Dissertation for the Degree of *Doctor of Philosophy*

A Study of Acoustic Scene Classification in Concept Drift Situation

Ibnu Daqiqil Id

Graduate School of Interdisciplinary Science and Engineering in Health Systems
Okayama University
Okayama – Japan

September 2022

Dissertation submitted to

Graduate School of Interdiciplinary Science and Engineering in Health System

of

Okayama University

for

partial fulfillment of the requirements

for the degree of

Doctor of Philosophy.

Written under the supervision of

Professor Masanobu Abe

Abstract

The dynamics of real-world data pose significant challenges for implementing predictive machine learning (ML) models. Changes in data can cause ML models to experience performance degradation or model decay. This degradation is caused by the model assuming a static relationship between input and output variables, but as the distribution of train and evaluation data changes, the relationship will change. This phenomenon is known as the concept drift. Concept drift can occur in various cases, especially acoustic scene classification (ASC) where the data distribution may change due to many factors like the change of event sound in the scene, non-stationary noise, overlapping audio events in the time or frequency domain, and echoes or reverberant operating conditions.

In this study, we propose a Combine–merge Gaussian mixture model (CMGMM) and Kernel density drift detection (KD3) to solve the concept drift problem. The CMGMM is an incremental algorithm based on the Gaussian mixture model (GMM) that adapts to the concept drift by adding or modifying its components to accommodate the emerging concept drift. The algorithm's advantages are adaptation and continuous learning from stream data with a local replacement strategy to preserve previously learned knowledge and avoid catastrophic forgetting. KD3 is a window-based algorithm for concept drift detection. It works based on estimating the window density using the Kernel Density Estimation (KDE). The concept drift can be detected by comparing the probability functions between successive windows—the greater the variation between the windows, the more evidence obtained for the concept drifts.

In the first experiment, the training dataset consisted of audio signals extracted with a 10-second window from 15 scenes. We add new novel event sounds into datasets audio to simulate the concept drift in four concept drift types, namely abrupt concept drift (AB), gradual concept drift (GR), recurring concept drift type 1 (R1), and recurring concept drift type 2 (R2). In the evaluation, two adaptation strategies are used, namely active and passive. In the active adaptation strategy, the adaptation process is carried out when the concept drift detector detects concept drift. Whereas in the passive strategy, adaptation is carried out every specific time span or period.

The evaluation results demonstrated that the proposed algorithms work well in detecting and adapting to four types of concept drift and three scenarios. In the active adaptation strategy, the adaptations of CMGMM on R1 and R2 showed better accuracy than those on AB and GR. On average, AB exhibited the lowest accuracy, while R2 showed the highest accuracy. RI and R2 have better accuracy because the CMGMM is designed to preserve the old concept, then the model can recognise the previously learned concept if it is repeated in the future.

In the passive adaptation strategy, a short cycle could disrupt the model component because the model is trained with insufficient data, leading to an underfitting problem. The model requires a high-frequency adaptation in AB and GR to maintain the performance; therefore, sensitive hyperparameters or a short cycle size is required. Based on the experimental results, the passive method is more suitable for these concept drift types.

Furthermore, we try to improve the CMGMM performance by implementing a two-step scene classification using Pretrained Audio Neural Networks (PANNs) as a feature extractor for scene audio. PANNs is CNN based on a model trained under large-scale event audio. It processes the log-mel spectrogram of an audio scene to obtain the high-level features containing the occurrence probability of a particular sound event in the scene. CMGMM uses these vectors in the training and adaptation process.

The experiment result shows that PANNs have better accuracy than MFCCs both in the active and passive approaches. In the active approach, PANNs show significant improvement in AB and GR. In the passive approach, PANNs tend to have lower performance in rapid adaptation batches.

In the last experiment, we try to solve the concept drift problem where a new class emerges. We propose a framework to perform incremental learning using rehearsal strategies. We use representation data from past data and pseudo data generated by GAN (Generative adversarial networks) model to improve model performance and positive backward transfer in acoustic scene classification problems. A positive backward transfer means that learning a new task would increase the model's performance on the previously learned tasks.

In the experiment, we compare the accuracy of incremental learning using GAN only, representative data with GAN and augmentation in large and small representative data settings. The results show that the GAN's accuracy decline over time, so the use of this method is not suitable for a large number of tasks. The use of GAN and representative data show better results

than representative data with augmentation, both in small and large representative data. When using this combination, the classifier's performance improved and showed more stable accuracy. In small representative data with GAN, selecting representative data with a low logit value gives the best results and the random method shows the best results in large representative data.

Acknowledgments

I would like to extend my gratitude to my supervisor, Prof. Masanobu Abe, for giving me the trust, freedom, and support to pursue a thesis topic of my interest. I appreciate the kind understanding he has shown me; his helpful comments, guidance, and support have aided in producing a thesis I am truly proud of.

In addition, I would like to thank Dr Sunao Hara, Katsuki Inoue and all Abelab members for technical & language support during my studies.

I finally wish to acknowledge my family, whose help and support have been invaluable. It is due to them that I have been given the opportunity to pursue a degree at Okayama University; from them that I have gained my curiosity and desire to learn; and whose values have defined the person that I am.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ADWIN	Adaptive sliding window
ASC	Acoustic scene classification
BIC	Bayesian information criterion
CMGMM	Combine–merge Gaussian mixture model
CNN	Convolusional neural network
EM	Expectation maximisation
GAN	Generative adversarial model
GMM	Gaussian mixture model
KD3	Kernel density drift detection
KDE	Kernel density estimation
KSWIN	Kolmogorov–Smirnov statistic test
MFCC	Mel-frequency cepstral coefficients
ML	Machine learning
PANNs	Pretrained audio neural networks

Chapter 1.
Introduction

Traditionally, classification works on the assumption that the data distribution does not change. That means the data used to train and test the model is generated from the same distribution. However, this is different in many real-world applications where the system is implemented may change due to many factors, such as changes in behaviour, environment, or worn instruments or sensors. As a result, the data distribution should be expected to change. This problem is known as concept drift, and when this happens to the data, it causes the performance of the static model to degrade.

This chapter covers the background, the research problems and its contribution to the field. This chapter also presents an outline of the rest of the thesis.

1.1 Background

Sound carries a large amount of information and has an essential role in human communication, comparable to the role played by images or text. There are four major fields in audio processing research, namely Automatic Speech Recognition (ASR), Speaker Recognition (SR) and non-speech research into Acoustic Event Detection (AED) and Acoustic Scene Classification (ASC). The first two research areas are focused on human language, and excellent results have emerged from these research areas. For example, Apple's Siri and Amazon's Alexa are well-known applications of ASR. SR is also becoming popular in security systems that consider voice biometrics as one of the necessary security layers, especially in modern telephone banking applications. Regarding the two remaining research areas, these are considered to be newly (or recently) emerging fields over the past few years. One of the pioneers, Lyon[1], defined this as “machine hearing” and described it in terms of how accurately machines can hear and understand sounds in real environments compared to humans.

Human-computer interaction through audition requires devices to recognise the environment using acoustic sound analysis. One of the primary research topics in this area is acoustic scene classification (ASC), which attempts to classify digital audio signals into mutually exclusive scene categories. ASC is an important area of study covering various applications, including smart homes, context-aware audio services, security surveillance, mobile robot navigation, and wildlife monitoring in natural habitats. Machine audition applications have a high potential to lead to more innovative context-aware services.

We intend to develop an ASC system for environmental or scene audio in specific locations (i.e., beaches, shops, bus stations, and airports) with different acoustic characteristics. The scene audio contains an ensemble of background and foreground sounds. One of the most important aspects of the audio scene in real life is the concept drift [2], whose data distribution might evolve or change in the future. For example, at a bus station where at the time of recording the training data, the sound events that exist there are wind noise, engine noise and horns, but over time, physical conditions or human activities change, resulting in new event sounds appearing such as conversations, music and ambulances[3]. The addition of these new sounds will cause changes in the distribution of the data or concept drift. As a result, the model make predictions on unseen data that potentially reduce the performance of the model over time [4].

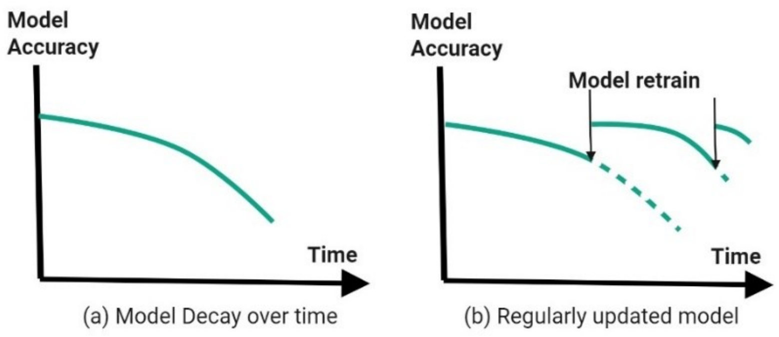


Figure 1.1. Model decay and regularly update model performance

In case of data drift, data evolves with time, potentially introducing a previously unseen variety of data and new categories or classes. The most straightforward solutions for handling the abovementioned problems to maintain the model performance are periodic retraining and redeployment of the model.

Nevertheless, these solutions can be time-consuming and costly. Moreover, deciding on the frequency of retraining and redeployment is difficult. Another promising approach is to use an evolving or incremental learning method [5][6], where the model is updated when a new subset of data arrives [7]. Each iteration is considered an incremental step toward revisiting the current model. This thesis investigates the acoustic scene classification under the concept drift problem by proposing an adaptive algorithm that can update its model when a concept drift is detected.

1.2 Research Objective

This research aims to solve the concept drift problem by developing a novel concept drift detection algorithm which can detect concept drift and perform adaptation on the drifted data to maintain the model performance. Furthermore, we propose a rehearsal strategy to retrain our model in incremental class problems efficiently.

1.3 Contribution

The main contributions of this research are summarised as follows:

- A novel concept of drift detection and adaptation algorithm is proposed: Kernel Density Drift Detector (KD3) and Combine Merge Gaussian Mixture Model (CMGMM). KD3 is a kernel density-based algorithm that aims to detect concept drift and supply drifted data for adaptation. Furthermore, CMGMM performs adaptations based on this data by modifying and adjusting existing components. The algorithm's advantages are adaptation and continuous learning from stream data with a local replacement strategy to preserve previously learned knowledge and avoid catastrophic forgetting (Chapter 3).
- A two-step classification improves the model performance when adapting to concept drift using high-level features. This method can improve model performance by using a large pre-train model as a feature extractor (Chapter 4).
- A framework for the incremental training of deep convolutional networks based on the rehearsal strategy is proposed. In the framework, a combination of representative data and pseudo data to maintain the model performance in the class increment problem (Chapter 5)

1.4 Research Significance

The theoretical and practical significance of this research is summarised as follows:

- Theoretical significance: The concept drift problem is the root cause of performance deterioration of machine learning systems. This study aims to improve supervised machine learning performance by detecting concept drift and adapting to the change. The proposed concepts and theorems are mathematically general and could be applied to other learning problems. Furthermore, the rehearsal strategy always uses a fixed number of pseudo-data which is effective for large number of training.

- Practical significance: This study proposes a series of experiments to improve the ASC model performance in the concept drift situation. The proposed algorithms can be implemented in many cases, such as noise monitoring systems [8], enhancing user experience [9], [10] and speech-processing[11], [12], audio surveillance, recognition and classification [13], mobile phone sensing[14][12], context-aware robots[15], intelligent wearable devices[16], robotics navigation systems[17] and audio archive management [18].

1.5 Dissertation outline

The remaining parts of this dissertation are organised as follows:

- **Chapter 2** provides an overview of Concept drift, Acoustic scene classification and the Gaussian mixture model.
- **Chapter 3** presents the proposed method (CMGMM and KD3) and its experiment result on acoustic scene classification based on Gaussian Mixture Model in the concept drift situation.
- **Chapter 4** explains the use of CMGMM with high-level features and its experimental result on concept drift adaptation for acoustic scene classification.
- **Chapter 5** describes the proposed rehearsal-based incremental learning and its experimental result for acoustic scene classification.
- Finally, **Chapter 6** draws the conclusions of this research, points out the study's limitations, and suggests some future works.

Chapter 2.

Literature Review

This chapter briefly covers concept drift and reviews published works about concept drift.

2.1 Acoustic Scene Classification

2.1.1 Definition

The ability of devices to understand their environment through the analysis of sound is the main objective of machine hearing research, a broad research area related to Computational Auditory Scene Analysis (CASA)[19]. Machine listening systems perform processing tasks similar to the human auditory system and are part of a broader research theme linking disciplines such as machine learning, robotics and artificial intelligence.

Acoustic scene classification (ASC) refers to the task of associating a semantic label to an audio stream that identifies the environment in which it has been produced. ASC, in this study, aims to classify a sound clip into one of the provided predefined classes that characterise the environment in which it was recorded. An acoustic scene denotes the label of the place where the sound was recorded (e.g., train, car, park, indoor), the situation (e.g., in a meeting, in an emergency), and the human activity involved (e.g., cooking, chatting, vacuuming) [20]. An acoustic event means a specific type of sound, such as bird sounds, footsteps, running water, or music. Many sound clips contain multiple acoustic events that overlap on the time axis. Figure 2.1 illustrates the relationship between acoustic scenes and acoustic events.

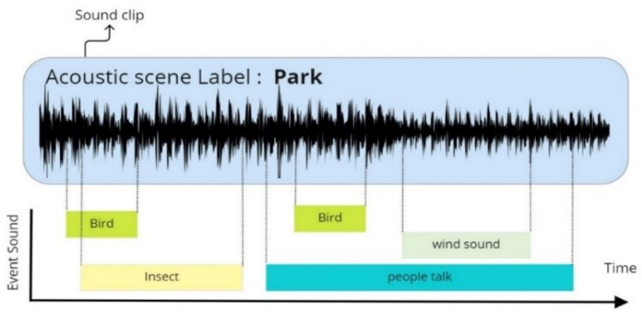


Figure 2.1. Relationship between acoustic scene and acoustic event [20]

In ASC, acoustic features are first extracted from a training dataset and an acoustic scene model is constructed using the acoustic features. After that, an acoustic feature extracted from a test sound clip is input to the acoustic scene classification system, and the system produces an acoustic scene label of the test sound clip.

2.1.2 ASC Application

By detecting the current location, devices could obtain useful information to enable machine to respond appropriately or adjust certain functions, opening a wide range of distinct applications. Applications which can directly benefit from ASC encompass existing technologies from smartphones to hearing aids:

- Hearing aid adapt their configuration to the user's environment, such as a quiet office, restaurant or music hall. Current hearing aid solutions are tuned according to general acoustic environments that do not adapt quickly to changes in context. [9], [10] show that automatic program switching using ASC is greatly beneficial for hearing aid users to enhance users' listening experience.
- Context-awareness devices include always-listening capabilities to adapt behaviour to the surrounding situation. For example, ASC enables smartphones to continuously sense their surroundings[8], switching their mode to silence every time we enter a concert hall. Another example of practical applications is reported in [21], where wearable devices adjust the rate or intensity of notifications depending on the context. The cost of being distracted by a device may be high: imagine receiving many notifications in the car while driving, at the restaurant with other people or while crossing the street. The decision to notify or not and how to notify the user, should be made considering the current context.
- Robot use information of "where I am" to switch behaviour and functions**.** Clarkson et al. [22] integrated ASC ability into a robotics system to help define the most appropriate actions. Frank et al. [23] adjusted robotic wheelchair function and speed based indoor or outdoor location.
- Smart city sensor use ASC systems for monitoring noise pollution in a distributed system using intelligent acoustic analysis [24].

- Speech-processing enhancement. El-Maleh[12] performs noise removal to enhance speech-processing systems by modifying the processing according to the type of background noise.
- Elderly assistance**.** Buyot et al. [25] employ ASC as part of a larger system aiming to monitor the daily activities of elderly people.

ASC applications are not only limited to these areas but can also benefit systems utilising context-aware services [45], audio segmentation[18] and retrieval[26], intelligent wearable devices [52], noise monitoring systems [8], audio archive management [32] and so on.

2.2 Concept Drift

2.2.1 Definition of Concept Drift

A concept can be formally defined as a set of instances generated by a stationary source function[27]–[29]. Therefore, concept drift can be defined as a change in the source generating the data. In other words, concept drift is a phenomenon in which the statistical properties of a target domain change over time in an arbitrary way[30]. It was first proposed by [31] to point out that noise data may turn to non-noise information at different times. These changes are usually caused by changes in hidden variables or features which cannot be measured directly. Formally, Concept drift defines as follows:

Definition 2.1. Concept Drift. In a t-period of time, a set of samples, denote as $D_{0,t} = \{s_0, \dots, s_t\}$, where $s_i = (X_i, y_i)$ is a data instance, X_i is the feature vector, y_i is the label and $D_{0,t}$ has a certain distribution $F_{0,t}(X, y)$. When the concept drift occurs in the next period, then the distribution of $P_{0,t}(X, y) \neq P_{t+1,2t}(X, y)$ [2], [4], [32]. Furthermore, Gama [1] also formalises the concept drift as a change in the joint probability $P(\mathrm{X}, \mathrm{y}) = P(\mathrm{y}|\mathrm{X})\, P(\mathrm{X})$ that is consistent and persistent.

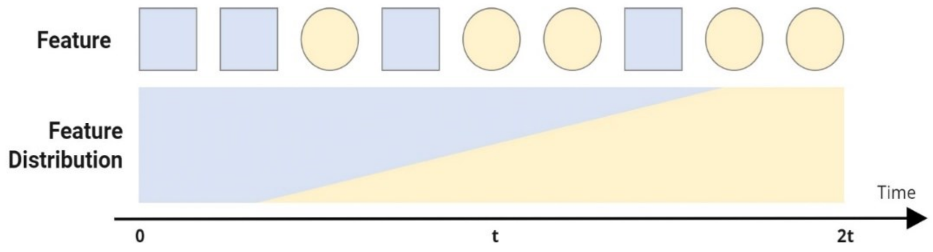


Figure 2.2. Feature and feature distribution illustration in concept drift

2.2.2 Types of Concept Drift

At present, there are different standards to categorise concept drift. This section categorises the types of concept drift from type of change and its impact. Concept drift is based on type of change categorised into two types namely virtual and real concept drift. In virtual concept drift, the change in the joint probability $P(x, y)$ is caused by the change of $P(x)$, but the $P(y|x)$ does not change. Meaning that there was a change in the underlying feature distribution, but the model's performance hasn't changed. In real concept drift, $P(y|x)$ is changed, then affects the model performance.

Concept drift can be broadly categorised into three types based on the impact on classifier performance over time [33]:

- Sudden drift occurs when the concept changes abruptly. This often manifests itself as a sudden drop in classifier performance, as illustrated in Figure 2.3,
- Gradual drift generally happens when the concept gradually changes from one concept to another.
- Recurring drift are trends or patterns which repeat themselves at intervals and might look something like Figure 2.3. Recurring trends are commonly found in seasonal data.

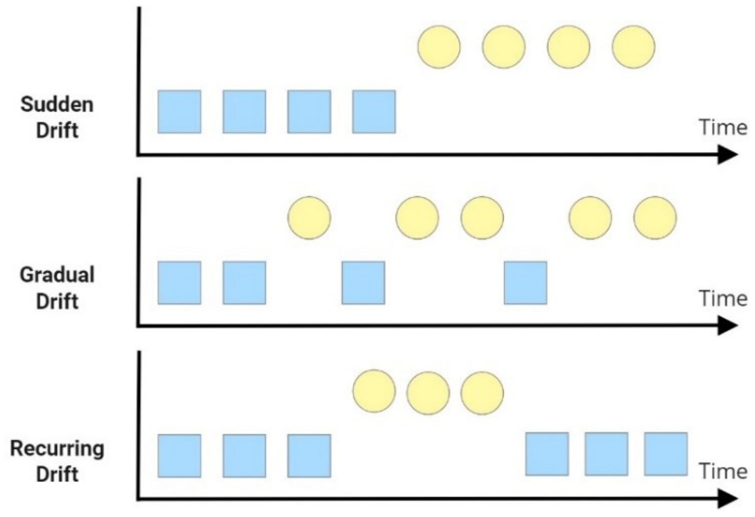


Figure 2.3. Type of Concept Drift

Other non-drift issues are similar to the effects and mechanisms of concept drift. Depending on your situation, they can have similarly big effects on your model. Here is a list of the most important ones:

- Prior Probability Shift: Prior probability shift means a change in the class priors in the output's statistical distribution $P_{0,t}(y) \neq P_{t+1,2t}(y)$. Other command terms for prior probability shift are class probability shift, label drift, real concept shift, and class drift.
- Novel Class Appearance: In a classification problem, your model needs to predict a label it hasn't seen before. $P_{0,t}(Y = y) = 0$ and $P_{t+1,2t}(Y = y) > 0$. New unseen labels typically result from a change in upstream data collection (e.g., a new field in a form) introducing a new value.

2.2.3 Concept Drift Detection methods

Concept drift detection methods are some detectors that can signal the change of data distributions based on the learning algorithm's performance or the statistics of the input data. The algorithm can be categorised into three main types:

A. Methods monitoring distributions of two different time windows

Test whether two fixed-length sequences are from the same distribution according to predetermined confidence. These methods usually use a fixed reference window representing the past concept's summary and a sliding detection window containing the instances to be tested on. Then some statistical tests were performed on the two windows with the null hypothesis stating that the distributions are equal. If the null hypothesis is rejected, a change is triggered on the test window.

B. Detectors based on sequential analysis

The sequential probability ratio tests, such as the Wald test, are the basis for change detection algorithms in this category. These methods detect changes in the following way: let $x_1, \dots, x_n$ be the sequence of the instances. Assume the subset of instances $x_1, \dots, x_w$ are generated from an unknown distribution P_0 and $x_{w+1}, \dots, x_n$ are generated from distribution P_1. If the probability of observing subsequences under P_1 is significantly higher (above some threshold) than that under P_0 then a drift is triggered at the point w.

C. Detectors based on Statistical Process Control

Statistical Process Control considers learning as a process and monitors the evolution of this process. For each example from the data stream, the prediction result of the model can be either true or false. For a set of examples, the error is a random variable from Bernoulli trials. The Binomial distribution gives a general form of the probability for the random variable that represents the number of errors in a set of n examples. For each point i in the sequence, the error rate is the probability p_i of observing false with the standard deviation $\sigma_i = \sqrt{p_i(1-p_i)/i}$. The drift detector manages two registers during the model operation, p_{min} and σ_{min}. At time i, after casting the prediction for the current example and verifying the prediction error if $p_i + \sigma_i$ is lower than $p_{min} + \sigma_{min}$, then $p_{min} = p_i$ and $\sigma_{min} = \sigma_i$

Here we briefly describe a few drift detection methods used in our approach and the experiments:

- **ADWIN** (ADaptive sliding WINdow) [34] is the best-known representative of methods from the window monitoring distributions. It takes a sequence of real values $x_1, x_2 \dots, x_n, \dots$ and a confidence parameter $\delta \in (0,1)$, minimum number of items n to start searching changes. ADWIN keeps a variable-length window W of recently seen items, which has the maximal length statistically consistent with the hypothesis "there has been no change in the average value inside the window". Each time the number of items in the window exceeds n, , it loop over all the partitions of $W = W_0W_1$ and it will trigger a change if the average value in one sub-window is significantly different from the other with confidence level δ. The split point of the two windows is the indication of concept drift.

- **HDDM$_A$** and **HDDM$_W$** [35] are also detect algorithms that compare two windows. The former compares the moving averages in different windows to detect drifts. The latter uses the EMWA forgetting scheme [42] to weight the moving averages. After that, weighted moving averages are compared to detect concept drifts. For both cases, the Hoeffding's inequality is used to set an upper bound to the level of difference between averages. The authors noted that the first and the second methods are ideal for detecting abrupt and gradual drifts, respectively.

- **KSWIN**[36] is a recent concept drift detection method based on the well-known Kolmogorov–Smirnov (KS) statistic test. The KS-Test is a non-parametric test that does not require any underlying data distribution assumption. KS-Test can handle only one-dimensional data. It compares the absolute dist(W, R) between two empirical cumulative distributions

2.3 Gaussian Mixture Model

Gaussian mixture model (GMM) is a probabilistic model representing normally distributed subpopulations within an overall population. GMM are a generalization of Gaussian distributions and can be used to represent any data set clustered into multiple Gaussian distributions. It assumes that all the data points are generated from a mix of Gaussian distributions with unknown parameters. A GMM can be used for clustering, which is the task of grouping a set of data points into clusters. GMM can find clusters in data sets that may not be clearly defined. Additionally, GMM can estimate the probability that a new data point belongs to each cluster.

Gaussian mixture models are also relatively robust to outliers, which can still yield accurate results even if some data points do not fit neatly into any clusters. This feature makes GMMs a flexible and powerful tool for clustering data. It can be understood as a probabilistic model where Gaussian distributions are assumed for each group and have means and covariances that define their parameters.

In GMM, a component of a Gaussian distribution is represented by (w, μ, P) and $\{(w_1, \mu_1, P_1), (w_2, \mu_2, P_2), \dots (w_K, \mu_K, P_K)\}$ to denote a mixture of K Gaussian components, where w, μ, and P are the weight or prior probability, distribution means, and covariance matrix, respectively. Let x be a d-dimensional random vector drawn from a K-component GMM. The probability density function (pdf) of x is given by

$$P(x|\theta) = \sum_{i=1}^{K} w_i \mathrm{N}(w_i; \theta_i), \tag{2.1}$$

where θ is the component parameter $\theta_i = \{\mu_i, P_i\}$ and prior probability must satisfy $w_1 + w_2 + \dots + w_K = 1$ and $w_K \geq 0 \ \forall K$. The definition of N shown in Eq. 2.2 and the log-likelihood of K-component GMM for sample data X where $X = \{x_i\}_{i=1}^{N}$ shown in Eq. 2.3.

$$N(x; \mu_i, P_i) = \frac{1}{\sqrt{(2\pi)^d \det P_i}} e^{\left[-\frac{1}{2}(x-\mu_i)^T P_i^{-1}(x-\mu_i)\right]}. \tag{2.2}$$

$$L(X|\theta) = \sum_{i=1}^{N}\left[\sum_{j=1}^{K} w_j N(w_j;\theta_j)\right] \tag{2.3}$$

When the number of components K is known, expectation-maximisation (EM) is the technique most used to estimate the mixture model's parameters. EM is an iterative algorithm that starts from some initial estimate of θ (e.g., random) and then proceeds to iteratively update θ until convergence is detected.

2.4 Kullback Leibler Divergence for the Gaussian Mixture Model

Kullback-Leibler (KL) discrimination, known as the Kullback-Leibler divergence or relative entropy, is a tool to measure the discrepancy between two probability distributions. The KL discrimination between $f(x)$, a probability distribution for random variable X and $g(x)$, another probability distribution is the expected value of the log-likelihood ratio. So, the KL divergence of $f(x)$ and $g(x)$ is defined as Eq. 7 where $\Re^d$ is the sample space of the random variable X.

$$d_{KL}(f,g) = \int_{\Re^d} f(x)\log\frac{f(x)}{g(x)}\ dx \tag{2.4}$$

Definition 2.2. Kullback Leiber Divergence for Gaussian pdf. Let $f(x)$ is a d-dimensional Gaussian pdf with a mean vector μ_1 and covariance matrix Σ_1, and $g(x)$ is a d-dimensional Gaussian pdf with mean vector μ_2 and covariance matrix Σ_2, then by subtitute Eq. 2.2 to Eq. 2.4 we got the KL Divergence for normal as shown in Eq. 2.5.

$$d_{KL}(f,g) = \frac{1}{2}\left[\mathrm{tr}(\Sigma_2^{-1}[\Sigma_1 - \Sigma_2 + (\mu_1-\mu_2)(\mu_1-\mu_2)^T]) + \log\frac{\det(\Sigma_2)}{\det(\Sigma_1)}\right] \tag{2.5}$$

Definition 2.3. Kullback Leiber Upper Bound [37]. Let $f(x), h_1(x), h_2(x)$ is any pdf over d-dimensional and $0 \le w \le 1,\ w + \bar{w} = 1$ then

$$d_{KL}(wh_1 + \bar{w}h_2, f) \le\ wd_{KL}(h_1, f) + \bar{w}d_{KL}(h_2, f) \tag{2.6}$$

$$d_{KL}(f, wh_1 + \bar{w}h_2) \le\ wd_{KL}(f, h_1) + \bar{w}d_{KL}(f, h_2) \tag{2.7}$$

2.5 Kernel Density Estimation

Kernel density estimation (KDE) is the application of kernel smoothing for probability density estimation, i.e., a non-parametric method to estimate the probability density function of a random variable based on kernels as weights. KDE is a fundamental data smoothing problem where inferences about the population are made based on a finite data sample. The formal definition of KDE is shown in Eq.2.8

$$P_{kde}(x) = \frac{1}{nh}\sum_{i=1}^{n} K(\frac{x_i - x}{h}) \tag{2.8}$$

K is the kernel function that is generally a smooth, symmetric function such as Gaussian and h is the smoothing bandwith that controls the amount of smoothing K.

A range of kernel functions are commonly used: uniform, triangular, biweight, triweight, Epanechnikov, normal, and others. The Epanechnikov kernel is optimal in a mean square error sense, though the loss of efficiency is small for the kernels listed previously. Due to its convenient mathematical properties, the normal kernel is often used, which means $K(x) = N(x)$, where N is the standard normal density function.

2.6 ASC using Convolutional Neural Networks

In ASC problems, we generally start with unstructured data in the form of audio files. Audio can be represented as 2-D matrices of amplitude, energy, or another property of sound against time. As we have a large volume of unstructured data, Deep Learning approaches such as Convolutional Neural Networks (CNN) would be very effective at extracting features. While they are commonly used for images, they can also be applied to other forms of data that are in 2-D matrix form like spectogram.

2.6.1 Architecture

The figure 2.4 shows a high-level view of a CNN organization. Apart from the input layer, the middle layers achieve feature extraction while the final fully connected part performs classification.

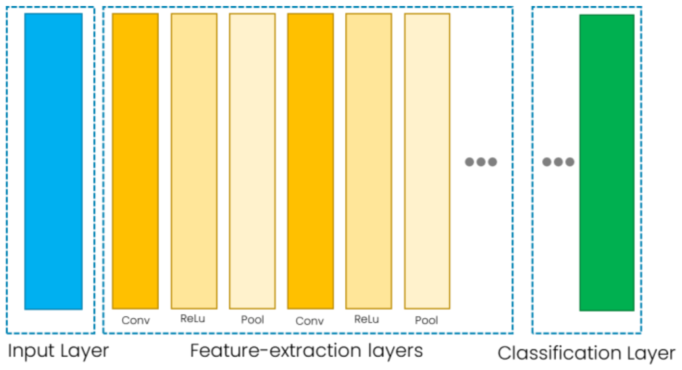


Figure 2.4. High level overview of CNN structures

In basic CNN architectures, feature extraction is performed by a repeated pattern. First, a convolutional layer is applied to the input, then an activation function and finally a Pooling layer, which reduces the information size.

2.6.2 Layers

The convolutional layer is the core building block of a CNN, and it is where the majority of computation occurs. Convolutional layers role is to find local conjunction of features from the previous layers [38]. They don't perform a simple matrix multiplication as the fully connected in a feed-forward neural network do, they rather execute a convolution. Weights in a convolutional neural network are grouped in matrices called kernels or filters.

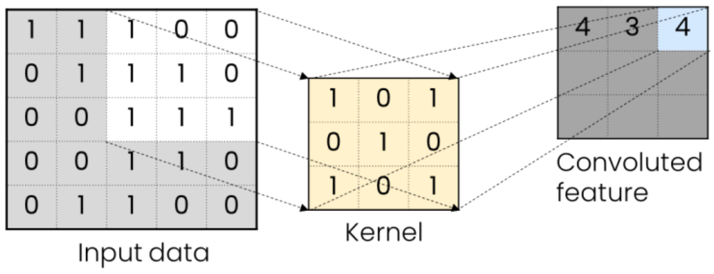


Figure 2.5. Convolution operation

As shown in figure 2.5 the convolution operation is applied by multiplying a kernel for an area in the input matrix, each value is multiplied by the corresponding weight, then the result is the sum of all multiplications. The result is stored in the output matrix called feature map or activation map. There are multiple of such filters. Once the input has been completely processed, the network uses the next filter. It is important to notice, that the depth of the activation map is not related to the input or the kernel depth, instead, it is equal to the number of kernels applied to the input image.

Some definitions for this layer:

- N: width/height of the input, in the simplified case of a square input image.
- P: the amount of padding used in the input. The padding is useful to obtain a desired dimension in the activation map, for example, to keep the same inputsizes.
- S: the stride. It expresses by how much a kernel is shifted during convolution.
- F: the kernel dimension.

Pooling layers, also known as downsampling, conducts dimensionality reduction, reducing the number of parameters in the input. Similar to the convolutional layer, the pooling operation sweeps a filter across the entire input, but the difference is that this filter does not have any weights. Pooling layers merge semantically similar features found in the previous activation map[38] and help control overfitting[39]. The most common layer is the Maxpooling which extract the maximum value.

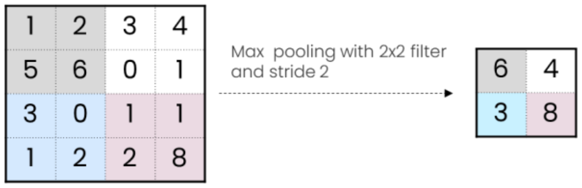


Figure 2.6. Maxpooling example

Activation layers apply an activation function to each element of their input typically a ReLU or LeakyReLY. The fully connected layer takes the output of convolution/pooling, flattens it and predicts the best label to describe the image. As in a normal feed-forward neural network, the inputs to the fully connected layer are multiplied by the weights and summed together. Then an activation function is used to produce the output. The results are propagated to the next fully connected layer. The last one has a neuron for each class label, and it produces the probability distribution.

2.7 Incremental Learning

Incremental learning studies the problem of learning from a growing amount of data and aims to integrate new tasks while gradually maintaining knowledge of old tasks. Incremental learning is also referred to as continual learning, lifelong or sequential learning. The basic principle is that the proposed learning algorithm is capable of learning sequentially from data. Also, the most important obstacle in incremental learning is catastrophic forgetting, which is the interference caused by learning on new data reducing the knowledge of previously learned data.

2.7.1 Challenges

The problem of catastrophic forgetting has emerged as one of the main problems facing incremental learning. Catastrophic forgetting is defined as a complete forgetting of previously learned information by a neural network exposed to new information[37].

The catastrophic forgetting phenomenon is a special case of another challenge in incremental learning training known as the "stability-plasticity dilemma". This dilemma describes the friction between incremental, parallel learning and plasticity. Too much plasticity will result in previously encoded data being constantly forgotten, whereas too much stability will impede the efficient coding of this data at the level of the connections between neurons. In other words, incremental learning requires the right balance between forgetting and stability. Reducing "forgetting" may improve network "stability" but it is not really addressing the greater problem if it comes at the cost of "plasticity," as it so often does.

2.7.2 Method

There are three main categories of recent incramental learning algorithms can solve the problem: **Regularization-based**, **Architecture-based,** and **Rehearsal-based.**

Regularization-based methods[38]–[41] restrict the model update ability by limiting the learning rate on important parameters for past tasks. These methods can address catastrophic forgetting without preserving past cases, but they do not provide satisfactory performance under challenging settings[42] or on complex data sets[43].

Architecture-based methods aim to expand or adjust the model architecture (network or components) during the learning process [44]–[48]. Some models use masks to activate a subset of the network [49]–[51] or modify the existing component in the model [52]–[54]. For instance, [44] proposed extending the network by augmenting each layer with a fixed number of neurons for new tasks but keeping the parameters of the old layers fixed to avoid forgetting. The limitations of this approach result in extensive networks. Furthermore, some methods require task identity for conditioning the network at test time, like [19], which is less flexible in the architectural adaptation of other models, and it isn't easy to improve its performance.

Rehearsal-based methods have shown successful results compared to other approaches. Rebu et al. [55] introduced iCarl to store a representative set of exemplars of classes encountered in older learning sessions, using a process called herding. They combine distillation and classification loss at each increment to train a classifier, and The Nearest Mean of Exemplars rule is used for classification. Jaehong et al. [56] explored the online core-set selection to select high-affinity samples of past tasks. The result shows that online core-set selection is more efficient than state-of-the-art techniques like EWC[39], A-GEM[57] and ER-Reservoir[58]. Prabhu[59] also introduces a method named Gdumb, wherein the sampler greedily stores samples while balancing the classes. Furthermore, a generative-based method like Generative feature replay[60], MER-GAN [61] also demonstrated that Generative Adversarial Networks could be used as a preserving knowledge alternative method. This method's limitation is that additional storage is required to store previously learned data.

2.8 Generative Adversarial Model

Generative Adversarial Networks (GANs) [40] are systems based on a min-max strategy where two algorithms are confronted: one algorithm generates data (the generator) and the other discriminates between fake and real data (the discriminator). Generator generates samples based on a vector sampled from latent space distribution, and discriminator network learns to determine whether a sample comes from the training data or Generator. The training procedure for the generator network is to maximize the probability of the discriminator misclassifying the generations. Meanwhile, the discriminator network is trained to distinguish between real data and generated data.

Hence, the objective function of the complete network is the following:

$$\min_{G} \max_{D} V(D, G) = \mathbb{E}_{x \sim p_{data}}[log\ D(x)] + \mathbb{E}_{z \sim p_z}[log(1 - D(\mathrm{G}(z))] \tag{2.9}$$

This expression represents value (*V*), which is a function of both, discriminator *D* and generator *G*. The goal is to maximize the discriminator (*D*) loss and minimize the generator (*G*) loss. Value *V* is the sum of expected log likelihood for real and generated data. Likelihoods (probabilities) are the discriminator outputs for real or generated images. Note that the discriminator output for a generated image is subtracted from 1 before taking the log. Maximizing the resulting values leads to optimization of the discriminator parameters such that it learns to correctly identify both real and fake data.

Chapter 3.

Acoustic Scene Classification in the Concept Drift Situation

In this chapter, we aim to solve concept drift problem in acoustic scene classification.

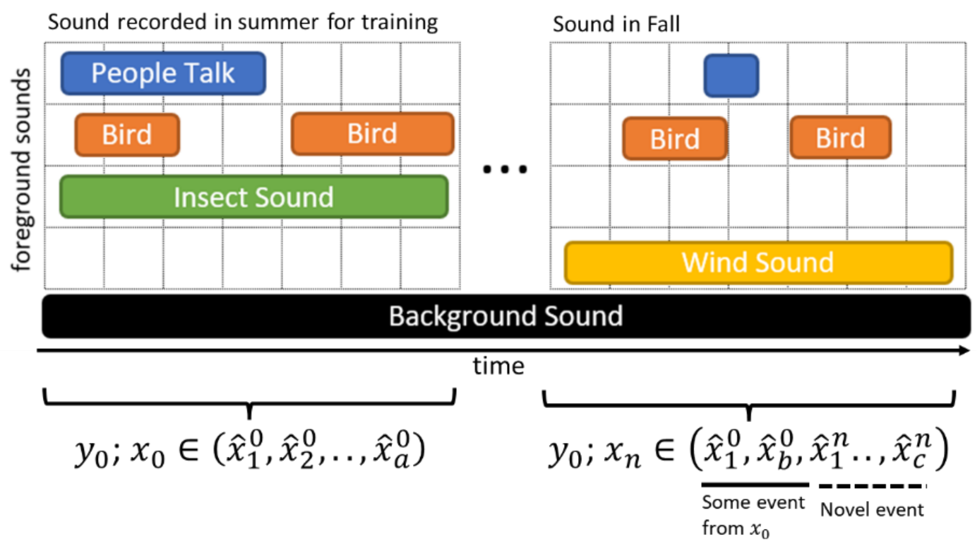
3.1 Concept Drift in Acoustic Scenes

An acoustic scene sound X consists of various specific event sounds $\hat{x}$ perceived and defined by humans [41]. For example, acoustic sound in the park consists of several event sounds like bird sounds, wind sounds, and insect sound. Mathematically, the relationship between of X and $\hat{x}$ determines $p(X, y)$, where $X \in (\hat{x}_1, \hat{x}_2, \hat{x}_3, .\hat{x}_i,)$ and i denotes the number of $\hat{x}$ in X. In the future, sound in the park, for example, might change due to weather, season, or human behaviour. As a result, it changes the distribution of sound events $\hat{x}$ in X, which then changes the relationship of $p(X, y)$. This situation is called the concept drift in acoustic scene, which is expressed as follows:

$$\exists X: p_{w_0}(X, y) \neq p_{w_n}(X, y). \tag{3.1}$$

Eq. 3.1 and Figure 3.1 describe concept drift as the change in the joint probability distribution between two-time windows, w_0 and w_n. Models built on previous data at w_0 might not be suitable for predicting new incoming data at w_n. This change may be caused by a change not only in the number of $\hat{x}$, but also in the underlying data distribution of $\hat{x}$. These changes require model adaptation because the model's error may no longer be acceptable with the new data distribution [42].

The change in the incoming data at w_n depends on a variety of different internal or external influences (e.g., event sounds that exist in a park depending on the season). The initial data recorded in the winter may only consist of people talking, bird calls, and dogs barking. However, the event sounds change in the summer, and new event sounds, such as insect and wind sounds, emerge.



Sound recorded in summer for training
Sound in Fall
foreground sounds
People Talk
Bird
Bird
Insect Sound
Bird
Bird
Wind Sound
Background Sound
time
$y_0; x_0 \in (\hat{x}_1^0, \hat{x}_2^0, .., \hat{x}_a^0)$
$y_0; x_n \in (\hat{x}_1^0, \hat{x}_b^0, \hat{x}_1^n .., \hat{x}_c^n)$
Some event from x_0
Novel event

Figure 3.1. Illustration of the concept drifts in an acoustic scene audio at a park

3.2 Kullback Leiber Discrimination

Based on Eq. 2.5, it's clear that KL divergence of two probability distributions is always non-negative $d_{KL}(f,g) \geq 0$ and the discrimination of the same two probability distribution is zero, $d_{KL}(f,f) = 0$. However, KL divergence is asymmetry, $d_{KL}(f,g) \neq d_{KL}(g,f)$ also does not satisfy the triangle inequality, $d_{KL}(f,g) + d_{KL}(g,h) \geq d_{KL}(f,h)$. Furthermore, there appears to be no closed-form expression for the KL discrimination of one (non-trivial) Gaussian mixture from another. However, Runnalls in [37] enable us to put an upper bound on the discrimination of the mixture after the merge from the mixture before the merge, so we can use it to measure the discrimination between the mixture.

Suppose we are given a GMM model that contains two Gaussian components $\{(w_1, \mu_1, \Sigma_1), (w_2, \mu_2, \Sigma_2)\}$ where $w_1 + w_2 = 1$ then we wish to merge this GMM into a single Gaussian $(1, \mu, \Sigma)$. A strong candidate is a distribution whose zeroth, first and second order moment match of (w_1, μ_1, Σ_1) and (w_2, μ_2, Σ_2). In other words, we refer $(1, \mu, \Sigma)$ as the moment-preserving merge of (w_1, μ_1, Σ_1) and (w_2, μ_2, Σ_2). The mean vector μ and covariance matrix P of the candidate is defined as

$$\mu = w_1\mu_1 + w_2\mu_2 \tag{3.2}$$

$$\Sigma_1 = w_1\Sigma_1 + w_2\Sigma_2 + w_1 w_2(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T \tag{3.3}$$

In the adaptation process, we must remove the restriction of $w_1 + w_2 = 1$ for GMM model that has more than two components. So, we wish to merge two Gaussian components $\{(w_i, \mu_i, \Sigma_i), (w_j, \mu_j, \Sigma_j)\}$, where $w_i + w_j \leq 1$, and approximate the result as a single Gaussian $(w_{ij}, \mu_{ij}, \Sigma_{ij})$. $(w_{ij}, \mu_{ij}, \Sigma_{ij})$ must preserve the zeroth-, first-, and second-order moments of the original Gaussian. The moment-preserving merge is shown in Eqs. 3.4–3.6.

$$w_{ij} = w_i + w_j \tag{3.4}$$

$$\mu_{ij} = \frac{w_i}{w_i + w_j}\mu_i + \frac{w_j}{w_i + w_j}\mu_j \tag{3.5}$$

$$\Sigma_{ij} = \frac{w_i}{w_i + w_j}\Sigma_i + \frac{w_j}{w_i + w_j}\Sigma_j + \frac{w_i w_j}{(w_i + w_j)^2}(\mu_i - \mu_j)(\mu_i - \mu_j)^T \tag{3.6}$$

Accordingly, we apply the KL dissimilarity by computing the Kullback–Leibler discrimination upper bound of the post-merge mixture with respect to the pre-merge mixture from definition 2.2 and Eq.2.5. In the case of the Gaussian mixture, where $f(x) = N(w_i, \mu_i, \Sigma_i), g(x) = N(w_j, \mu_j, \Sigma_j)$ and $w_i + w_j < 1$, the KL dissimilarity between $f(x)$ and $g(x)$ is shown in Eq. 3.7.

$$d_{KL}(f, g) = \frac{1}{2}\left[(w_i + w_j)\log(\det(\Sigma_{ij})) - w_i\log(\det(\Sigma_i)) - w_j\log(\det(\Sigma_j))\right] \tag{3.7}$$

However, overflow or underflow problems arise when we compute a very small or very large determinant of covariance Σ. In this research, we always use a symmetric covariance matrix by following the identity function in Eq. 3.8 to avoid overflow or underflow

$$\det(e^P) = e^{\mathrm{tr}(P)} \tag{3.8}$$

So finally, we use Eq. 3.9 to compute KL discrimination in the adaptation process.

$$d_{KL}(f, g) = \frac{1}{2}\left[(w_i + w_j)\mathrm{tr}(\log(\Sigma_{ij})) - w_i\mathrm{tr}(\log(\Sigma_i)) - w_j\mathrm{tr}(\log(\Sigma_j))\right] \tag{3.9}$$

The dissimilarity measure given by Eq.3.9 is reasonably easy to compute, with computational complexity at most $O(d^3)$ and symmetric.

3.3 Combine Merge Gaussian Mixture Model

The combine merge gaussian mixture model (CMGMM) is an incremental classification algorithm that has the ability to adapt to the new incoming concept. This algorithm is developed based on a GMM and inherits all the properties thereof, such as a set of parameters w, μ, and Σ denoting the non-negative weight, distribution means, and covariance matrix, respectively. Furthermore, this algorithm can add new components as new relevant information or concepts are identified in the current data, then merge some components to adjust the parameters of each distribution. The CMGMM algorithm pipeline is shown in Figure 3.2.

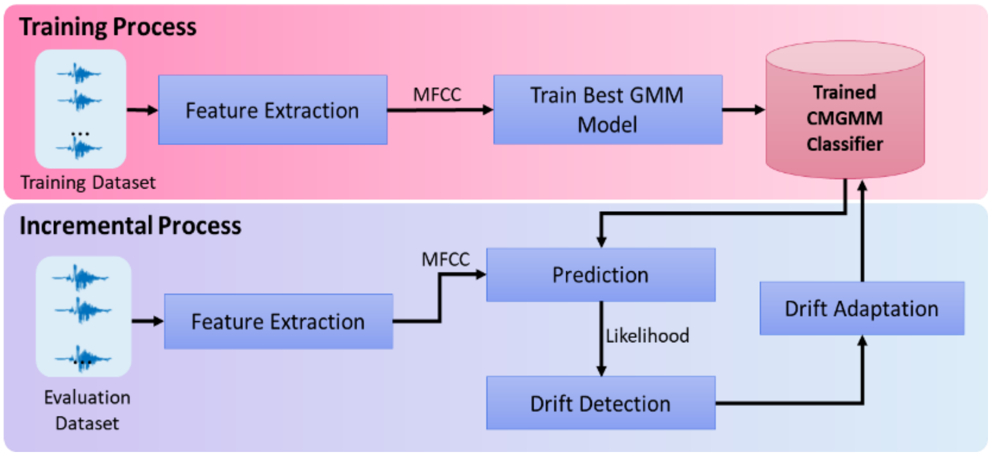


Figure 3.2. Combine–merge Gaussian mixture model (CMGMM) general workflow

In the training process, we extract the feature of the scene audio from the training dataset D_0 and train an optimal model $M_{optimal}$. We use the Expectation maximisation (EM) [43] algorithm to train the model and the Bayesian information criterion (BIC) [44] to select the best model.

In the incremental process, the $M_{optimal}$performance is observed through the prediction likelihood. When concept drift detector detects a significant likelihood change, the model activates the concept drift adaptation process. The concept drift adaptation process then begins by creating a local model $M_{drifted}$ from the new coming data. $M_{drifted}$ represents the new concepts or concept updates in the incoming data. Finally, we combine the $M_{optimal}$ and $M_{drifted}$ components to include any new concepts from $M_{drifted}$ that may not exist in the $M_{optimal}$ at the initial training and merge similar components to update the existing component in $M_{optimal}$.

The CMGMM pipeline process is detailed in the subsections that follow.

3.3.1 Feature Extraction

Feature extraction is the first step of both the training and incremental processes. In this research, we use normalised Mel-frequency cepstral coefficients (MFCCs) that represent the short-term power spectrum of audio in the frequency domain of the Mel scale. MFCCs are commonly used as features in audio processing and speech recognition. The first step is pre-emphasis for enhancing the quantity of energy in high frequencies. The next step is windowing the signal and computing the fast Fourier transformation to transform the sample from the time domain to the frequency domain. Subsequently, the frequencies are wrapped on a Mel scale, and the inverse DCT is applied [45]. Finally, each of the MFCCs is normalised using mean and variance normalisation based on Eq. 3.10.

$$MFCC_{norm} = \frac{(MFCC-\mu)}{S}, \tag{3.10}$$

where, μ and S denote the mean and the standard deviation of the training samples, respectively.

3.3.2 Model Training

The training process is intended to build a set of models from the training dataset D_0 containing training data $x = \{x_1, x_2, \dots, x_n\}$, where x_n denotes the MFCC vector. The models are trained Q times using the EM algorithm. For each training cycle, a different number of components K ranging from $K_{\min}$ to $K_{\max}$ is used, where $Q = K_{max} - K_{min}$. Consequently, a set of models $\mathcal{M} = \{\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, ., \mathcal{M}_Q\}$ is obtained based on the different numbers of components.

The next step is model selection using the BIC. In [46], the BIC value of a model $\mathcal{M}_K$ trained over the dataset X with K components, BIC(X, M_K), is defined as follows:

$$BIC(X, M_k) \equiv -2 \log L(X, M_K) + v \log N\,, \tag{3.11}$$

where, L denotes the model likelihood; v denotes the degree of freedom of the model parameters; and N denotes the number of training data points. The model with the lowest BIC value is selected because it maximises the log-likelihood [7]. Algorithm 3.1 presents the steps of the learning process.

Algorithm 3.1: Training the Optimal Model

```
Input: Initial Dataset $D_{init}$, Minimum Component Number $K_{min}$, and
       Maximum Component Number Kmax
Result: Best GMM Model
$BIC_{best} = \infty$
for $K_{min}$ to $K_{max}$ do
    $M_{candidate}$= EMTrain($D_{init}$, k)
    $BIC_{candidate}$ = ComputeBIC ($D_{init}$, $M_{candidate}$)
    if $BIC_{candidate} < BIC_{best}$ then
        $M_{best} = M_{candidate}$
    end
end
return $M_{best}$
```

3.3.3 Model Adaptation

Model adaptation aims to revise the current model upon newly incoming data that might contain new concepts or concept changes. The result of this adaptation is an adapted weighted mixture component that respects the original mixture.

The model adaptation method starts by training a new model M_{drift} from data drifts D_{drift} using Algorithm 3.1, and then combining the existing model $\mathcal{M}$. Consequently, the newly adapted model $\mathcal{M}$ accommodates the new concept represented by the components in $\mathcal{M}_{\text{drift}}$. The next step is to calculate the pairwise distance between the components in $\mathcal{M}$ using KL discrimination. The KL discrimination formula (Eq. 3.9) enables us to set an upper bound on the discrimination of the mixture before and after the merging process. By using KL discrimination CMGMM tend to select for merging:

- Component with low weights. In Eq. 3.9 the weight appears outside of logaritm so it has dominant effect
- Components whose means are close together in relation to their variances.
- Components whose covariance matrices are similar.

According to this formula, components with low weights means close to their variances, and similar covariance matrices are selected for merging. When two components are merged, the moment-preserving merging method [47] is used to preserve the mean and the covariance of the overall mixture (Eqs. 3.7–3.9). Figure 3.3 illustrates the CMGM adaptation process.

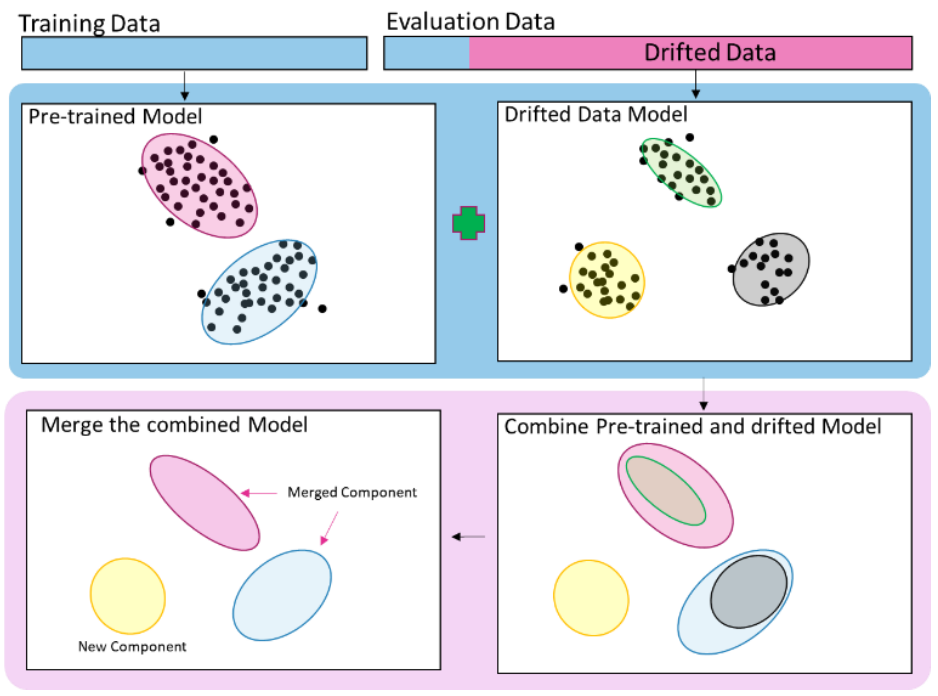


Figure 3.3. Illustration of the Combine–merge Gaussian mixture model (CMGMM) adaptation process

As a result, the reduction process generates a set of merged models $\mathcal{M}_{\text{merge}}$. To select the best $\mathcal{M}_{\text{merge}}$ model, the accumulative BIC is computed by combining sampling data from $\mathcal{M}_{\text{curr}}$ and D_{drift} then computes the BIC value using Eq. 3.14. The smaller the value of the accumulative BIC, the better the newly adapted model.

Based on [7] and [11], the CMGMM tends to increase the number of components because it combines and merges them. This mechanism leads to an overfitting problem because the adaptation frequency increases due to the sensitive KD3 hyperparameter.

To maintain the compactness of the CMGMM and avoid overfitting, we design a strategy to merge statistically equivalent components into one component, then prune the inactive components. The inactive components are identified by the proximity of the ratio of w and Σ of the merged component to zero. In practice, components with w that are very close to zero are ignored by the model, whereas those with a large covariance tend to overlap with other components. Algorithm 3.2 presents in detail the steps of the proposed CMGMM-based method.

Algorithm 3.2: Model adaptation

```
Input: Current Model $M$, Drifted Dataset $D_{drift}$
Result: Adapted Model
$M_{drift}$ = findBestGMM($D_{drift}$)
$M_{combine}$ = CombineGMMComponent($M_{drift}$, $M$)
distanceMatrix = KLDissimalarity($M_{combine}$)
ds = $D_{drift}$ + $M$.generateData()
$nComp_{min}$ = $M$.number_component
$nComp_{max}$ = $M_{combine}$.number_component
$BIC_{best} = \infty$
for targetComponent = $nComp_{min}$ to $nComp_{max}$
    $M_{merge}$ = mergeComponent(target, distanceMatrix)
    if useComponentPrune then
        ComponentPrune($M_{merge}$)
    End
    $BIC_{candidate}$ = ComputeBIC ($M_{merge}$, ds)
    if $BIC_{candidate} < BIC_{best}$ then
        $M_{best} = M_{merge}$
    End
end
return $M_{best}$
```

3.4 Kernel Density Drift Detection

We propose Kernel Density Drift Detection (KD3) to detect the concept drift. KD3 is a window-based algorithm for concept drift detection. It works based on estimating the window density using the Kernel Density Estimation (KDE) or the Parzen's window [48]. The KDE is a non-parametric probability density estimator that automatically estimates the shape of the data density without assuming the underlying distribution. The concept drift can be detected by comparing the probability functions between these windows. The greater the variation between the windows, the more evidence obtained for the concept drifts. Aside from detecting concept drifts, KD3 also collects data for adaptation (D_{drift}) by identifying a warning zone when data begin to show indications of concept drift.

KD3 requires three hyperparameters, namely α, β, and h, which denote the margins for detecting the concept drift and accumulating the density distance and the window length, respectively. α is used to determine the threshold of the density variation in the concept drift, while β is employed to determine the threshold of the density variation in the warning zone. Therefore, α must be greater than β. KD3 accepts a set of likelihood windows z_c as input. z_c is the current likelihood window that contains a sequence of log-likelihood ℓ from the model prediction, $z_c = \{\ell_1, \ell_2, \ell_3, \dots, \ell_h\}$.

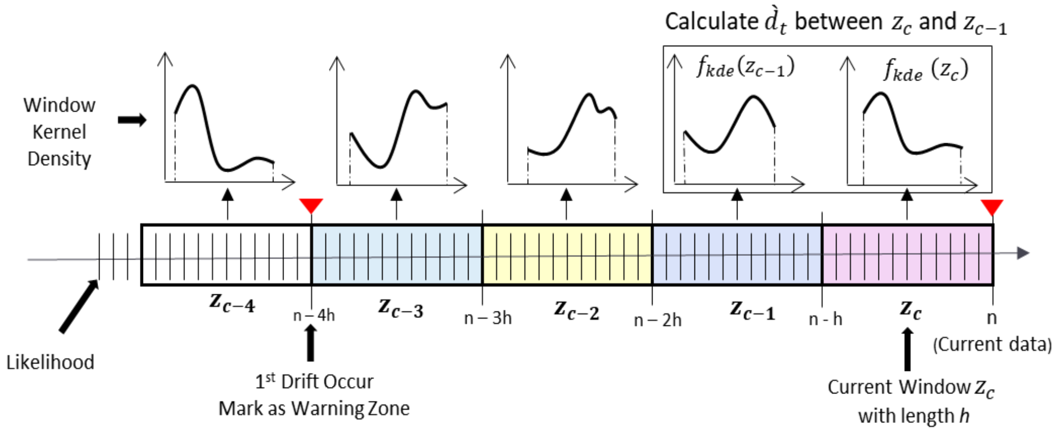


Figure 3.4. Illustration of the Kernel density drift detection (KD3) concept

First, this algorithm aims to estimate the density (f_{kde}) of the current z_c and previous z_{c-1} windows. Let ℓ_n be the latest generated ℓ. Let z_c contain the h-latest ℓ from ℓ_n, $z_c \in [\ell_{n-h}, \ell_n]$, and let z_{c-1} contain the h-latest ℓ from ℓ_{n-h}, $z_{c-1} \in [\ell_{n-2h}, \ell_{n-h}]$. To detect a concept drift, the distance $\grave{d}_t$ between f_{kde} of z_c and z_{c-1} is computed using Eq. (3.12) within the bounds of $b1$ and $b2$. The bounds are computed based on the maximum and minimum values of the joined ℓ of z_c and z_{c-1}.

$$\grave{d}_t = \frac{1}{2}\int_{b1}^{b2} |f_{kde}(z_c) - f_{kde}(z_{c-1})| \; dz, \text{where} \tag{3.12}$$
$$z_c \in [\ell_{n-h}, \ell_n],\ z_{c-1} \in [\ell_{n-2h}, \ell_{n-h}],$$
$$b1 = \min(\ell_{n-2h}, \ell_n),\, b2 = \max(\ell_{n-2h}, \ell_n).$$

Finally, the algorithm compares $\grave{d}_t$ to α and β. Suppose that the accumulative distance is equal to or greater than α. In that case, the algorithm sends the collected data to the model for adaptation. Figure 3.4 and Algorithm 3.3 illustrate the detailed KD3 process.

Algorithm 3.3: Detecting the Concept Drift

```
Input: Set of t likelihood $\ell$, drift margin $\alpha$, warning margin $\beta$ ($\alpha > \beta$), window
       length $h$,
Result: Drift Concept Signal, Drifted Dataset ($D_{drift}$)
Window_1 = $\ell[c - h: c]$;
Window_2 = $\ell[c - 2h: c - h]$;
$B_{min}$, $B_{max}$ = CalculateWindowBound($\ell[c - 2h: c]$);
KDE_1 = EstimateKDE(Window_1)
KDE_2 = EstimateKDE(Window_2)
diff = distance(KDE_1, KDE_2, $B_{min}$, $B_{max}$)
if (diff $\geq \alpha$) then
    resetWarningZoneData()
    return true, $[c - h: c]$
end
if (diff $\geq \beta$) then
    accumulativeWarning += diff
    if (accumulativeWarning $\geq \alpha$) then
        resetWarningZoneData();
        return true, $[c - h: c]$
    end
    return false, $[c - h: c]$
end
return false, null
```

3.5 Experiment Setting

This section provides information about the datasets and experimental setup used in this study to train, optimise, and evaluate the proposed method.

3.5.1 Datasets

We used three types of datasets in this experiment, that is, training, optimisation, and evaluation. The training dataset consisted of audio signals extracted with a 10-seconds window from 15 scenes in the TUT Acoustic Scenes 2017 [49] and TAU Urban Acoustic Scenes 2019 datasets [49]. The scenes were home, airport, beach, office, cafe, grocery store, bus, tram, metro, city center, residential area, street pedestrian, and shopping mall.

To simulate the concept drift in the datasets, the optimisation and evaluation datasets were generated by overlap and add new additional event sounds from and UrbanSound8K datasets [50] and the BBC Sound Class Library [51]. When the sounds were added, the numbers of additions (1–10), positions in the time axis (0–9000 ms), and loudness (−20,0) of the sounds were randomly changed at random. In total, the dataset had 46 new event classes that do not exist in the datasets, like food mixer sounds, fountain, sneeze, thunder, etc.

We generated four concept drift types with three scenarios. The four types of concept drift are as follows:

- abrupt concept drift (AB), where ongoing concepts are replaced with new concepts at a particular time;
- gradual concept drift (GR), where new concepts are started to add to an ongoing concept at a particular time gradually;
- recurring concept drift type 1 (R1), where an ongoing concept is replaced at a particular time with concepts that previously appeared; and
- recurring concept drift type 2 (R2), where concepts that previously appeared are added to an ongoing concept at a particular time.

Figure 3.5 illustrates the scenario generation. The three scenarios were designed to have different data distribution complexities. The scenario details are described below:

- Scenario 1 (Sc1): A unique event sounds from a specific event sounds is repeatedly introduced with a random number of times, gain, and timing. For example, in the airport scene, unique sounds representing the airplane sound, crowd background, and construction site are overlaid with a random number of times (1–10), position (0–9000 ms), and loudness (−20,0).
- Scenario 2 (Sc2): Several event sounds are randomly selected from a set of the same sound labels in Sc1 and added using the same rule as Sc1.
- Scenario3 (Sc3): This scenario differs from Sc1 and Sc2 in that event sounds coexist among scenes. For example, a set of rain sounds exists in other scenes (e.g., beach, city center, and forest paths). The methods of selection and addition are the same as those in Sc2.

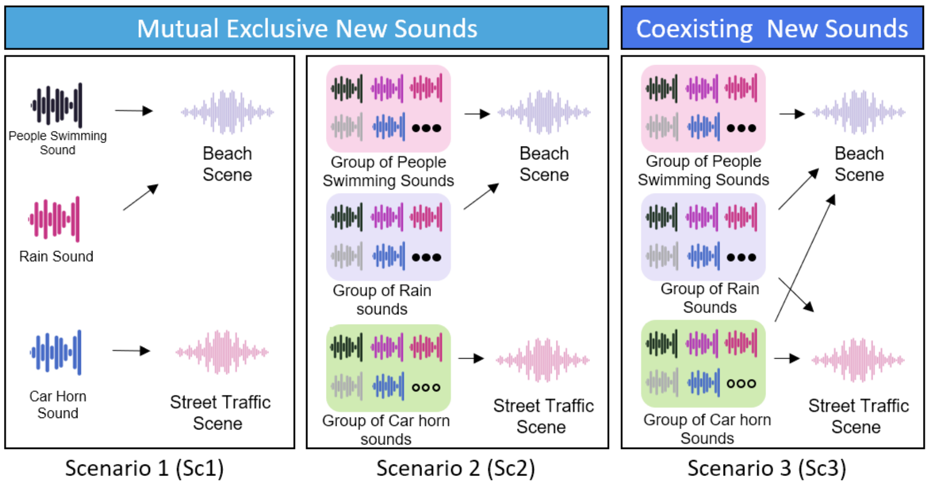


Figure 3.5. Concept drift scenario

Table 3.1 shows a list of event sounds that appear at scene types in every concept drift scenario. The mutually exclusive event sounds appear in all scenarios, but coexisting sounds only appear for Sc3.

Table 3.1. Setting of the novel sounds in scene audio for Sc1, Sc2, and Sc3

Scene	Mutually Exclusive Sounds in Sc1, Sc2, and Sc3	Additional Coexisting Sounds in Sc3
Airport	Helicopter, crowd, and construction site	Airplane, footsteps, and children playing
Beach	People swimming, footsteps on the sand, and rain	Teenage crowd, dog, and birds
Bus	Car horn, engine, and city car	Kitchenware, phone ringing, children playing, and teenage crowd
Café /restaurant	Washing machine, food mixer, and kitchenware	Phone ringing, children playing, and teenage crowd
City center	Sound of bird, ambulance, and wind	Footsteps, phone ringing, and children playing
Grocery store	Footsteps, children playing, and shopping cart	Vacuum cleaner, phone ringing, and footstep
Home	Frying, door, and vacuum cleaner	Clock and phone ringing
Metro station	Siren, road car, and thunder	Footsteps, crowd, and wind
Office	Typing, phone ringing, and sneeze	Broom, camera, and footsteps
Public square	People running, music, and airplane	Birds, rain, and teenagers talking
Residential area	Wind, camera, and cat	Birds, sneeze, and clock
Shopping mall	Clock, camera, and teenage crowd	Children playing, phone ringing, dog
Street pedestrian	Dog, bicycle, and bird	Footsteps and children playing
Street traffic	Motorcycle, horn, and train	Siren, airplane, and bell
Tram	Coughing, bell, and footsteps on the pavement	Teenage crowd and children playing

Finally, we have one training dataset, four optimisation datasets, and 12 evaluation datasets in this experiment. Each training dataset contained 3,000 scene audios, while the optimisation and evaluation dataset contained 15,000 scene audios. The datasets are available in our repository[1].

3.5.2 Experimental Scenario

The CMGMM accuracy was evaluated under four concept drift types in three scenarios (i.e., Sc1, Sc2, and Sc3). The evaluations are performed using the two following approaches:

- Active CMGMM adaptation: In this approach, the CMGMM actively detects the concept drift using a certain method and only adapts the model when the concept drift is detected. In this study, we compared KD3 to ADWIN [34], HDDMA, HDDMW [35], and KSWIN [36].
- Passive CMGMM adaptation: In this approach, the CMGMM adapts as soon as a particular datum is received without requiring the explicit prior detection of the concept drift. Several adaptation cycle sizes were tested, that is, 25, 50, 100, 150, and 200.

3.5.3 Hyperparameters

To evaluate the proposed algorithm, the following best hyperparameter values were selected based on the grid-search results over the test data.

- MFCC hyperparameters: The number of MFCCs was set to 13, the length of the fast Fourier transform (FFT) window was set to 2048, and the number of Mel bands was set to 128.
- KD3 hyperparameters: The window length h was set to 45, the drift margin α was set to 0.001, and the warning margin β was set to 0.00001.
- ADWIN hyperparameters: The delta parameter for the ADWIN was set to 0.002.
- HDDM hyperparameters: The drift confidence level was set to 0.001, and the warning confidence level was set to 0.005.
- CMGMM hyperparameter: Number of components to train the best model parameters ranged from 3 to 30 components.

[1]https://bit.ly/CMGMM_Dataset

- IGMM hyperparameters: The minimum and maximum number of components were set to 3 and 60, respectively.

3.5.4 Experimental Metric

To evaluate the effectiveness of the proposed method, the following metric was considered:

- Accuracy is the ratio of correctly predicted observations to the total number of observations. The accuracy formula is shown in Eq. 3.15

$$Accuracy = \frac{TP\ + TN}{TP + FP + FN + TN} \tag{3.13}$$

- F1 score – the weighted average of precision and recall.

$$Precision = \frac{TP}{TP + FP} \tag{3.14}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.15}$$

$$Precision = \frac{2(Precision\ \times Recall)}{Recall\ +\ Precision} \tag{3.16}$$

3.6 Experiment Result

The experimental results of the proposed method are presented herein.

3.6.1 CMGMM and IGMM comparison

The first experiment is to compare CMGMM and IGMM. Table 3.2 lists the experiment result for both active and passive approaches. For the active approach, results are shown for all combinations of the detector and adaptation methods. For the passive approach, the results are shown for different numbers of cycles of the adaptation process.

Table 3.2 Experiment Result in Active and Passive Scenario

ACTIVE APPROACH											
ADAPTOR	**DETECTOR**	**ACCURACY**			**F1 SCORE**			**EXECUTION TIME**			**Drift**
		Sc1	**Sc2**	**Sc3**	**Sc1**	**Sc2**	**Sc2**	**Sc1**	**Sc2**	**Sc3**	
CMGMM*	KD3*	0.8373	**0.7962**	**0.7409**	0.8432	**0.7993**	**0.7460**	128.06	115.07	110.49	39
	ADWIN	**0.8401**	0.6415	0.6332	**0.8418**	0.6379	0.6379	83.07	84.22	85.44	23
	HDDM	0.2762	0.2627	0.2990	0.3184	0.2992	0.3406	84.81	84.53	83.11	373
IGMM	KD3*	0.8283	**0.7574**	**0.6622**	0.8173	**0.7499**	**0.6488**	120.04	128.75	120.50	35
	ADWIN	**0.8419**	0.5711	0.6057	**0.8423**	0.5722	0.6063	82.80	84.219	83.08	21
	HDDM	0.2363	0.2507	0.2032	0.2436	0.3055	0.2675	84.37	87.55	84.87	350
PASSIVE APPROACH											
ADAPTOR	**CYCLE**	**ACCURACY**			**F1 SCORE**			**EXECUTION TIME**			**Drift**
		Sc1	**Sc2**	**Sc3**	**Sc1**	**Sc2**	**Sc2**	**Sc1**	**Sc2**	**Sc3**	
CMGMM*	50	0.5621	0.4451	0.4003	0.5719	0.4615	0.4373	83.178	82.945	83.163	-
	100	0.7424	0.6547	0.6434	0.7451	0.6583	0.6476	83.688	82.265	83.704	-
	150	0.8002	**0.7437**	**0.7301**	0.8043	**0.7482**	**0.7327**	84.527	82.298	89.555	-
	200	0.7602	0.7073	0.6904	0.7663	0.714	0.7001	83.414	85.922	84.594	-
IGMM	50	0.5365	0.4209	0.3687	0.5458	0.4319	0.3888	84.467	82.776	82.655	-
	100	0.7324	0.643	0.6371	0.736	0.6448	0.6388	82.693	82.652	82.199	-
	150	**0.8056**	0.7401	0.7291	**0.8107**	0.7431	0.7223	83.659	82.645	83.453	-
	200	0.7528	0.7149	0.6899	0.7609	0.722	0.6981	84.597	84.228	85.797	-

(*) Proposed Method

By comparing the best score in the passive and active approaches, it can be noticed from Table 3.2 that better scores were always achieved under the active approach. KD3 demonstrated a better performance than ADWIN except for Sc1. This is mainly because KD3 takes the entire data distribution into account, while ADWIN uses only its average. The drawback of KD3 is its high computational cost. HDDM demonstrated the worst performance among the tested drift detectors. This is because HDDM detected drifts even when they actually did not occur.

In terms of the overall accuracy, CMGMM demonstrated a better performance than IGMM except for Sc1. Different from CMGMM, IGMM has the parameter of the maximum number of Gaussian components for the distribution of the training data. Therefore, when a concept drift is simple as in Sc1, IGMM outperformed CMGMM. However, when the concept drift occurred with a complicated combination of event sounds, CMGMM outperformed IGMM because CMGMM allows flexible adaptation using the combining and merging mechanisms and has no limit for the number of Gaussian components.

The combination of CMGMM and KD3 achieved the best average performance in the experiment. Figure 3.6 shows the performance changes of this combination according to time for Sc1, Sc12, and Sc13 scenarios. It can be noticed from the figure that window accuracy almost always improves in Sc2 after the detection of the concept drift. In Sc3, accuracy is not improved from around 6,000th to 12,000th data point because there are a lot of deeper decrements compared to Sc1 and Sc2, as Sc3 simulates a more complicated distribution of the concept drift. In Sc3 the possibility of newly coming data having different concept with adapted model is higher than Sc1 and Sc2. While the performance is the best in Sc1, it seems that concept drifts are detected more often than they occurred.

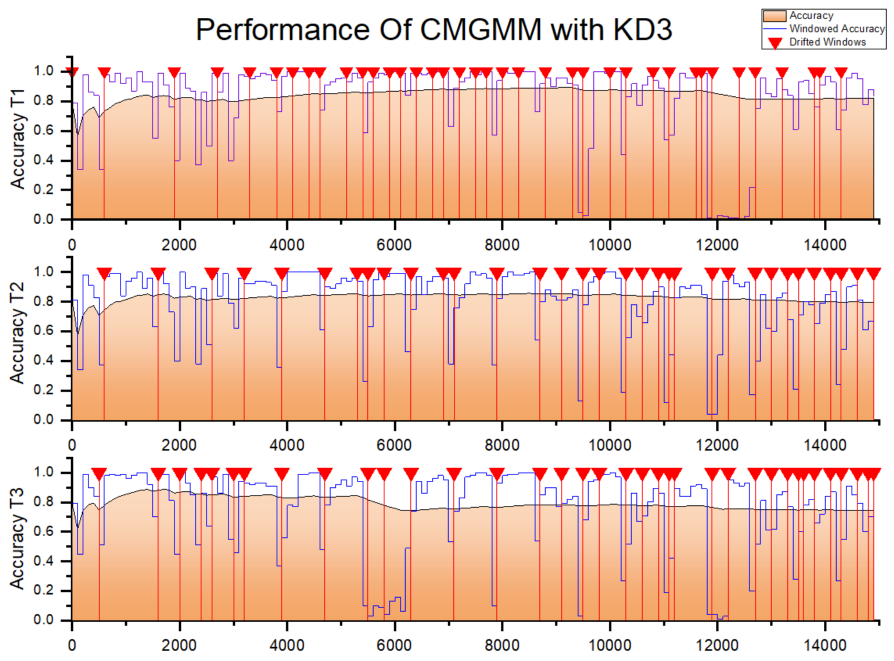


Figure 3.6. Overall and windowed performance of CMGMM and KD3 in the three scenarios:Sc1, Sc2, and Sc3.

Figure 3.7 shows the change in performance of CMGMM with ADWIN overtime for Sc1, Sc2, and Sc3 scenarios. In scenario Sc1, the performance of this combination is good, although the number of adaptations is reduced compared to the combination of CMGMM and KD3 in the same scenario. However, in the other two scenarios featuring complicated concept drifts, ADWIN demonstrates delays in detecting the concept drifts at several points. These detection delays lead to adaptation delays, causing the overall performance of CMGMM with ADWIN to decrease over time from around 90% to 70%. For example, in scenario Sc2, there is a significant decrease in performance from 3000th to 4000th data points. Hence, the accuracy of drift detection and speed of adaptation to drifts are vital aspects impacting the overall model performance.

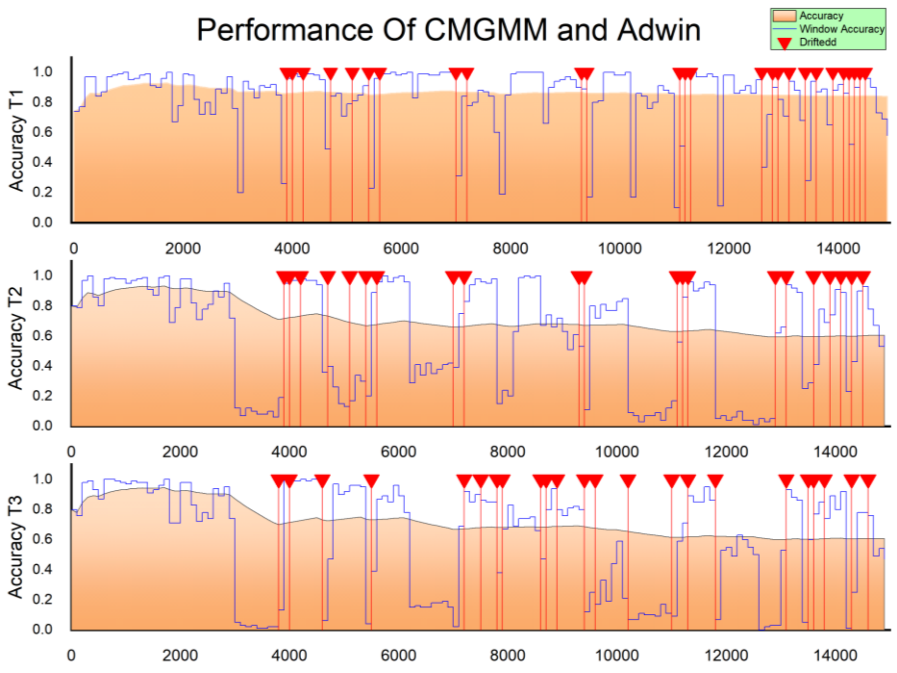


Figure 3.7. Overall and windowed performance of CMGMM and ADWIN in the three scenarios:Sc1, Sc2, and Sc3

Furthermore, the adaptation cycle (i.e., the number of data points to update) plays an important role in achieving a good performance in the passive approach. While IGMM shows its best performance in Sc1 and CMGMM in Sc2 and Sc3, both models have the same optimal adaptation cycle of 150 data points. This suggests the importance of finding the best adaptation cycle, which may vary from one dataset to another. For example, when the adaptation cycle for CMGMM in Sc1 increased from 50 to 100 and 150, the accuracy score increased from 0.5621 to 0.7424 and 0.8002, respectively. However, when the adaptation cycle further increased to 200 or more, the accuracy score decreased to 0.7602.

3.6.2 Hyperparameter Optimization

The next experiment is the systematic optimisation of the KD3 hyperparameter. We used the grid-search method using a combination of hyperparameters α from 0.1 to 0.001, β from 0.0001 to 0.000001, and ĥ from 45 to 300. We prepared a particular dataset for the KD3 hyperparameter optimisation in four types of concept drift.

In [52], we reported that hyperparameters β and ĥ did not significantly affect accuracy. Therefore, during the initial step, we observed the performance change according to β and ĥ. Figure 3.8 shows the average accuracy in all concept drift types according to hyperparameters β and ĥ. In this experiment, the best β and ĥ were set at 0.0001 and 45, respectively.

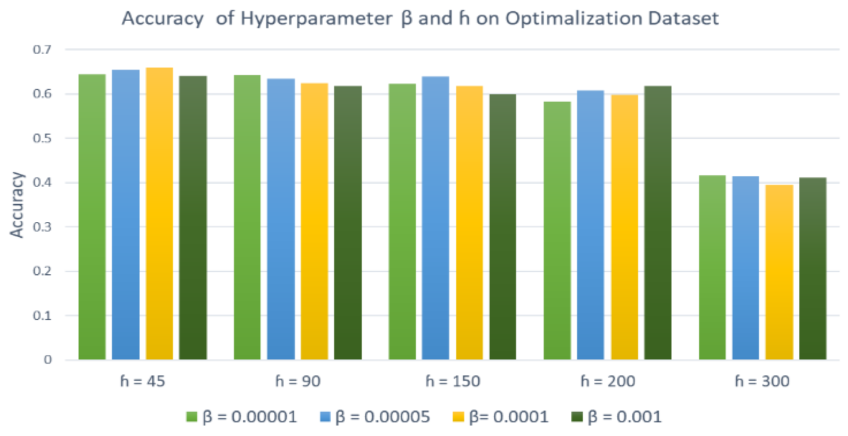


Figure 3.8. Result of hyperparameters β and ĥ in four types of concept drift

Table 3.3 lists the experimental results of α in the optimisation dataset in four types of concept drift. Based on this experiment, every concept drift type has its respective hyperparameter α according to the concept drift characteristics. AB and GR have similar characteristics. There are no repeating concepts in the future; hence, a more sensitive concept drift detector than R1 and R2 is required.

Table 3.3. KD3 hyperparameter optimisation result

Concept Drift Types	**Hyperparameter α (β = 0.001, ĥ = 45)**				
	α = 0.1	**α = 0.05**	**α = 0.01**	**α = 0.005**	**α = 0.001**
AB	0.6568	0.7113	0.7069	**0.7137**	0.7066
GR	0.6007	0.6173	**0.7050**	0.6922	0.7002
R1	**0.7332**	0.7232	0.7158	0.7054	0.6927
R2	**0.7268**	0.7133	0.7228	0.7090	0.6823
Overall	0.6793	0.6912	**0.7126**	0.7050	0.6950

In AB and GR, a sensitive hyperparameter α accelerated the update frequency. In the experimental result for these types of concept drifts, a high adaptation frequency reduced the loss received. However, a less-sensitive hyperparameter showed a better result in recurring concept drifts, where an old concept reappears in the future. A less-sensitive hyperparameter α provided the model with longer data than the sensitive hyperparameter.

We also selected the overall hyperparameter setting based on this experiment. The overall hyperparameter was selected from the best average performance of the hyperparameter optimization (α = 0.01, β = 0.001, and ﬁ = 45). We used this hyperparameter for further CMGMM and KD3 evaluations.

3.6.3 Active CMGMM Adaptation Result

Table 3.4 presents the experimental results of the active CMGMM adaptation. In general, the model performance without a concept drift detector is low in all concept drift types and scenarios. The adaptations of the CMGMM on R1 and R2 showed better accuracy than those on AB and GR. On average, AB exhibited the lowest accuracy, while R2 showed the highest accuracy. This high accuracy on recurring was caused by the CMGMM being designed to preserve the old concept, even though the new concept is adapted in the model. Thus, the model can recognise the previously learned concept if it is repeated in the future.

Table 3.4. Experiment result of the CMGMM with the concept drift detector

Concept Drift Detector		**Accuracy**				**F1**				**Number of Concept Drift Detection**			
		Sc1	**Sc2**	**Sc3**	**Avg**	**Sc1**	**Sc2**	**Sc3**	**Avg**	**Sc1**	**Sc2**	**Sc3**	**Avg**
AB	**ADWIN**	**0.6989**	**0.6525**	**0.6214**	**0.6369**	**0.713**	**0.6495**	**0.6353**	**0.6599**	83	89	85	85
	HDDM_A	0.4157	0.4476	0.4345	0.4326	0.4586	0.4804	0.477	0.472	3287	3350	3359	3332
	HDDM_W	0.4041	0.4455	0.4546	0.4347	0.4485	0.4902	0.5004	0.4797	489	413	409	437
	KD3*	0.6469	0.6359	0.6331	0.6386	0.6476	0.6467	0.6395	0.6446	207	236	220	221
	KSWIN	0.6611	0.6241	0.6345	0.6317	0.6722	0.6369	0.6411	0.6501	132	121	123	125
	Without Detector	0.4121	0.4054	0.4095	0.4090	0.4235	0.4125	0.4095	0.4151	-	-	-	-
GR	ADWIN	0.6723	0.6341	0.6363	0.6475	0.6845	0.6506	0.6478	0.6609	92	81	83	85
	HDDM_A	0.4134	0.4463	0.4311	0.4302	0.4580	0.4946	0.4701	0.4742	3306	3308	3265	3293
	HDDM_W	0.4131	0.4497	0.4544	0.439	0.4524	0.487	0.4816	0.4736	0	409	401	270
	KD3*	**0.6999**	**0.6942**	**0.6879**	**0.694**	**0.7044**	**0.7004**	**0.6867**	**0.6971**	190	218	221	209
	KSWIN	0.6532	0.6241	0.618	0.6322	0.6631	0.6326	0.6204	0.6387	129	125	107	120
	Without Detector	0.3554	0.3524	0.3489	0.3522	0.3571	0.3542	0.3501	0.3538	-	-	-	-
R1	ADWIN	0.7222	0.6948	0.6568	0.6912	0.7393	0.6981	0.6333	0.6902	78	83	87	82
	HDDM_A	0.4721	0.5204	0.4896	0.494	0.5206	0.5566	0.5392	0.5388	3154	3258	3201	3204
	HDDM_W	0.4847	0.4815	0.5068	0.491	0.5305	0.5215	0.5357	0.5292	0	662	647	436
	KD3*	0.7373	0.7334	0.7239	0.7315	0.7389	0.7341	0.7247	0.7325	208	201	204	204
	KSWIN	**0.7818**	**0.7351**	**0.7357**	**0.7508**	**0.7876**	**0.7404**	**0.7416**	**0.7565**	142	135	126	134
	Without Detector	0.4512	0.4458	0.4257	0.4409	0.4512	0.4458	0.4257	0.4409	-	-	-	-

R2	ADWIN	0.7353	0.7066	0.7089	0.7169	0.7398	0.7131	0.7163	0.723	75	86	82	81
	HDDM_A	0.5919	0.6034	0.6259	0.607	0.6369	0.6398	0.6658	0.6475	3052	2957	3017	3008
	HDDM_W	0.5789	0.5996	0.5875	0.5886	0.6369	0.6398	0.6658	0.6475	759	604	543	635
	KD3*	**0.7321**	**0.7325**	**0.7227**	**0.7291**	**0.7352**	**0.7315**	**0.7221**	**0.7296**	195	207	205	202
	KSWIN	0.6914	0.6964	0.677	0.6882	0.6977	0.6976	0.6811	0.6921	156	143	141	146
	Without Detector	0.4254	0.4205	0.3985	0.4148	0.4279	0.4198	0.3968	0.4148	-	-	-	-
Overall	ADWIN	**0.7072**	0.6720	0.6559	0.6731	**0.7191**	0.6778	0.6581	0.6835	82	84	84	83
	HDDM_A	0.4733	0.5044	0.4953	0.4910	0.5185	0.5428	0.5380	0.5331	3199	3218	3210	3209
	HDDM_W	0.4702	0.4941	0.5008	0.4883	0.5170	0.5346	0.5458	0.5325	312	522	500	444
	KD3*	0.7041	**0.6990**	**0.6919**	**0.6983**	0.7065	**0.7031**	**0.6932**	**0.7009**	200	215	212	209
	KSWIN	0.6969	0.6699	0.6663	0.6757	0.7051	0.6768	0.6710	0.6843	139	131	124	131
	Without Detector	0.4110	0.4060	0.3956	0.4042	0.4149	0.4080	0.3955	0.4061	-	-	-	-

The CMGMM experiment result depicted that KD3 outperformed other combinations in two of the four concept drift types in GR and R2. Meanwhile, ADWIN showed the best accuracy in AB. KSWIN demonstrated the best accuracy in R1, whereas HDDM was unsuitable for this case. Despite getting the highest overall accuracy score, the combination of the CMGMM and KD3 needed more frequent adaptations than ADWIN and KSWIN. In contrast, both HDDM-based methods showed worse performances compared to all others. HDDMA overdetected the concept drift in all concept drift types for more than 3000 times in GR.

The abovementioned results illustrated that the concept drift detector plays a vital role in the concept drift adaptation. The model performance decreased over time if the drift detector failed to detect or delay detecting or over detecting the concept drift.

Performance of the CMGMM and KD3

KD3 showed the best average accuracy of 0.6983 compared to ADWIN, KSWIN, and HDMM with 209 adaptations. This combination also showed its best results on R2 with 0.7321 accuracy, followed by R1 with 0.7373 accuracy, GR with 0.6999 accuracy, and AB with accuracy 0.6469. Furthermore, this combination was the most stable in all scenarios. The maximum performance decrements in AB, GR, R1, and R2 were 1.38%, 1.2%, 1.34%, and 0.94%, respectively.

Despite achieving a good performance in all concept drift types, the number of concept drifts detected in this combination was higher than ADWIN and KSWIN. The most significant number of adaptations occurred in AB. The disadvantage of a high number of adaptations is the higher computation time required to finish the task and possible overfitting. In this case, the higher numbers of adaptations in AB and GR are obtained because the concept constantly changes over time, and the learned concept becomes obsolete in the future; hence, the higher the adaptation, the better the performance.

Performance of CMGMM and ADWIN

In general, the combination of CMGMM and ADWIN showed a good performance in every concept drift type, especially on the abrupt datasets, where this combination showed its best performance. The overall accuracy was 0.6371 with 83 times of adaptation. The overall accuracies of this combination in AB, GR, R1, and R2 were 0.6369, 0.6475, 0.6912, and 0.7169, respectively. Furthermore, this combination had the advantage of a small number of adaptations in all concept drift types. Hence, ADWIN showed an effective performance in using resources and had a reasonably good performance. This combination performed very well on Sc1, but showed a performance drop in Sc2 and Sc3. For example, the accuracies of AB, GR, R1, and R2 decreased in Sc3 by 4.64%, 3.82%, 2.74%, and 287%, respectively.

Performance of CMGMM and HDDM

In this experiment, both Hoeffding's inequality-based algorithms showed underperformance results for all concept drift types. Both algorithms were less effective in detecting the concept drift in this case. The overall accuracies of HDDMA in AB, GR, R1, and R2 were 0.4326, 0.4302, 0.494, and 0.607, respectively. The number of HDDMA adaptations exceeded 3000 times of adaptation. This high adaptation process was ineffective because the amount of trained data for each adaptation was too small. This condition led to an overfitting and decreased the model performance.

HDDM W also experienced the same problem. In some cases, HDDMA failed to detect the drift concepts, such as GR, R1, and R2 in Sc1. The overall accuracies of this HDDMW in AB, GR, R1, and R2 were 0.4347, 0.439, 0.491, and 0.5886, respectively.

Performance of CMGMM and KSWIN

The combination of CMGMM and KSWIN showed the best performance in R1, with an overall accuracy of 0.750 with 134 adaptations. The accuracies of this combination in AB, GR, R1, and R2 were 0.6317, 0.6322, 0.7508, and 0.6882, respectively. On average, KSWIN required eight to nine adaptations per scene in all dataset types. This algorithm seems able to detect occurring changes in data and supports the concept drift handling process with good indicators at a given time.

3.6.4 Passive CMGMM Adaptation Result

Table 3.5 lists the experimental results of the passive CMGMM adaptation. The best performance in AB, GR, and R1 was obtained with a cycle size of 50, and that in R2 was obtained with a cycle size of 100. The best accuracies of AB, GR, R1, and R2 were 0.7152, 0.7139, 0.7323, and 0.7155, respectively.

Table 3.5. The experiment result of CMGMM without concept drift detector

Concept Drift Types	**Cycle Size**	**Accuracy**			
		Sc1	**Sc2**	**Sc3**	**Average**
AB	25	0.6290	0.6236	0.6256	0.6260
	50	**0.7122**	**0.7159**	**0.7177**	**0.7152**
	100	0.6285	0.5925	0.5955	0.6055
	150	0.6361	0.5776	0.5851	0.5996
	200	0.5580	0.5059	0.5119	0.5252
GR	25	0.6294	0.6232	0.6004	0.6176
	50	**0.7133**	**0.7169**	**0.7115**	**0.7139**
	100	0.6173	0.5887	0.6089	0.6049
	150	0.6371	0.5818	0.5797	0.5995
	200	0.5334	0.5094	0.5076	0.5168
R1	25	0.6186	0.5799	0.5608	0.5864
	50	**0.7235**	**0.7320**	**0.7416**	**0.7323**
	100	0.7211	0.7105	0.6988	0.7101
	150	0.7332	0.7086	0.7120	0.7179
	200	0.6639	0.7018	0.7146	0.6934
R2	25	0.5133	0.5444	0.5669	0.5415
	50	0.7396	0.6991	0.7011	0.7132
	100	**0.7431**	**0.7012**	**0.7023**	**0.7155**
	150	0.6865	0.6639	0.6803	0.6769
	200	0.6502	0.6011	0.6089	0.6200

Similar to active adaptation, R1 and R2 showed good performances compared to AB and GR, but better performances in passive adaptation. Although R1 exhibited the best adaptation at cycle size 50, it also showed good result at cycle sizes 100 and 150. If you consider the time and the computing resources used, then cycle sizes 100 and 150 are recommended.

In passive adaptation, the cycle size is vital in achieving a good performance. This cycle size determines the adequacy of the data for adaptation. If the cycle size is too short, the number of data adapted is small, leading to overfitting problems.

3.7 Summary

This chapter presents an incremental algorithm that can be used for classification in concept drift situations. This algorithm can be applied with a particular concept drift detector to have a better result. Moreover, we also proposed a concept drift detector that can be applied in multi-dimensional probability distribution; hence, it is more flexible in other models and cases. This algorithm outperforms other combinations, espesially in abrupt and recurring concept drift. To obtain the best results, adjusting the sensitivity level of the concept drift detector in the active scenario or the cycle size in the passive scenario determines the performance of the model. In the case of abrupt concept drift, a highly sensitive detector is required, but in the case of recurring, a less sensitive detector shows better results.

This research makes several contributions to the literature. First, we introduced the novel algorithm to adapt the drifted data using a local replacement strategy. Next, we optimize this algorithm under several types of concept drift to better understand how to deal with the concept drift.

Chapter 4.

Concept drift adaptation for acoustic scene classification using high-level features

4.1 Background

In classification tasks, one of the interesting challenges in machine learning is finding a good feature representation for training. For example, in image classification, image representations such as SIFT (Scale-Invariant Feature Transform) and HOG (Histogram of Oriented Gradients) have triggered significant advances in object matching and recognition. Interestingly, many successful representations are quite similar, essentially involving the calculation of edge gradients, followed by some histogram or pooling operation. While this is effective in capturing low-level image structures, the challenge is to find a suitable representation for mid- and high-level structures, i.e. corners, intersections, and object parts, which are certainly important for image understanding and improve its performance.

In scene audio detection, the event sounds in a scene audio determine the class of the scene. The challenge is that the event sounds can overlap and exist in many classes. For example, the sound of a conversation can exist in a restaurant and a bus station. However, in restaurants, there are other specific sounds such as the sound of dishes. So in a scene, the intensity level and combination of event sounds in the acoustic scene are the main identifiers of a scene.

Our previous experiment, we developed an ASC system that performs the task end-to-end by using scene audio recording directly using Combine–merge Gaussian mixture model (CMGMM). CMGMM is a Gaussian mixture model (GMM) based incremental algorithm. This algorithm can adapt concept drift by combining new components or revising existing components to adjust the emerging concepts. The algorithm's main benefits are its ability to adapt to new concepts with a local replacement strategy to maintain previously learned knowledge and avoid catastrophic forgetting.

CMGMM extracts Mel-frequency cepstral coefficients (MFCCs) from scene audio as an input vector and then classifies it. MFCCs is method non-parametric feature extraction method which models the human auditory perception. However, the way humans recognize the audio scene is different. We recognize it by identifying the current events sound in the scene[8]. The presence or intensity of an event sound can influence our decisions. For example, the presence of an airplane takeoff sound event infers that the scene is an airport and not a shopping mall.

Therefore, this chapter proposed a two-step scene classification using Pretrained Audio Neural Networks (PANNs) [9] as a feature extractor for scene audio and CMGMM as a classifier to solve the concept drift problem. Figure 4.1 illustrates the proposed method

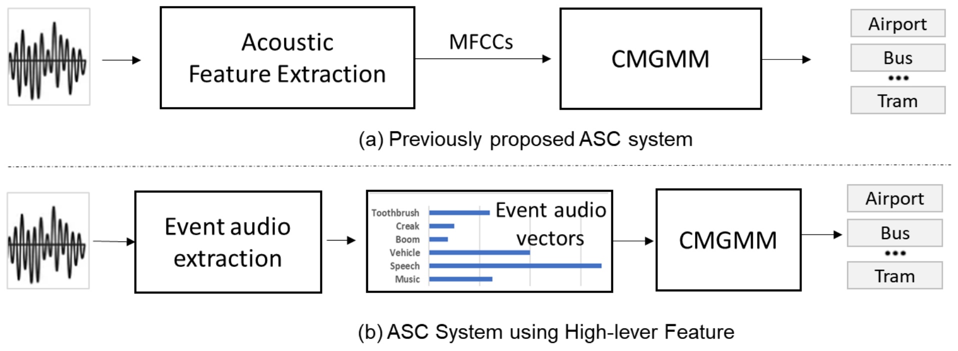


Figure 4.1. Conceptual illustration of (a): ASC system in chapter 3 and (b): the proposed method using PANNs to extract event sounds vector.

4.2 Pre-trained audio neural networks

Pre-trained audio neural networks (PANNs) is a convolutional neural networks (CNN) based model that can be used to solve audio tagging problem. Audio tagging is an essential task of audio pattern recognition, with the goal of predicting the presence or absence of audio tags in an audio clip. In this chapter, we use the PANNs model to extract audio tags for CMGMM.

The model is trained using balanced AudioSet[53] using 527 event sound classes. The AudioSet is a large-scale audio event dataset and contains 2,084,320 human-labeled 10-second sound clips representing 632 audio event classes. The video clips are in different lengths but the labels represent a10-second interval of the video clips.

PANNs has five convolusional blocks. Each convolutional block consists of 2 convolutional layers with a kernel size of 3 × 3. Batch normalization is applied between each convolutional layer, and the ReLU nonlinearity is used to speed up and stabilize the training. We apply average pooling of size of 2 × 2 to each convolutional block for downsampling 2 × 2 average pooling. Global pooling is applied after the last convolutional layer to summarize the feature maps into a fixed-length vector. The detail netwok topology as shown in Table 4.1. The number after symbol @ indicates the number of feature maps. "BN" and "FC" indicate batch normalisation and fully connected layer, respectively.

TABLE 4.1 Topology of the PANN CNN14 Model

Log Mel spectrogram
(Conv 3×3 @ 64, *BN, ReLU*)×2, Pooling 2×2
(Conv 3×3 @ 128, *BN, ReLU*)×2, Pooling 2×2
(Conv 3×3 @ 256, *BN, ReLU*)×2, Pooling 2×2
(Conv 3×3 @ 521, *BN, ReLU*)×2, Pooling 2×2
(Conv 3×3 @ 1024, *BN, ReLU*)×2, Pooling 2×2
(Conv 3×3 @ 2048, *BN, ReLU*)×2, Global pooling
FC 2048, ReLU
FC 527, Sigmoid

The PANNs processes the log-mel spectrogram of an audio scene to produce features containing the occurrence probability of a particular sound event in the scene. CMGMM uses these vectors in the training and adaptation process. This contrasts with our previous work [52], [54], [55], we used a low-level feature, MFCCs, directly to CMGMM. The example output of PANNs can be seen at Fig. 4.2

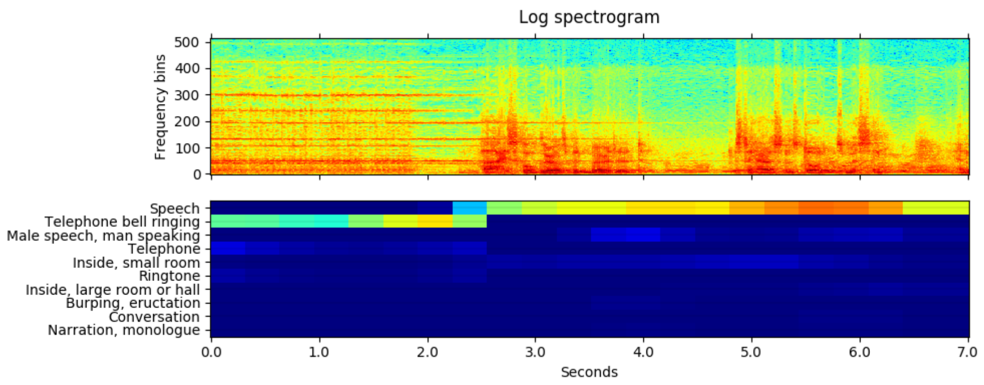


Figure 4.2. Example of PANNs output from a log spectogram input

We denotes the audio clips from spectrogram as x_{n}, where n is index audio clips and $f(x_{\mathrm{n}})\epsilon[0,1]^{K}$ is the output of PANNs representing the presence of K sound class in the audio clips.

4.3 Experimental Setup

This section presents information regarding datasets, experiment setup, and performance evaluation criteria used in this study. In this experiment, we use the same train and test dataset that we use in chapter 3.5.

4.3.1 Experimental Setup

The performance evaluations are performed using two approaches, namely:

- Active CMGMM adaptation. This approach assumes that the models detect concept drifts using a concept drift detector and adapt to the detected drifts. This study compared the proposed drift detection method KD3 to the ADWIN [34].
- Passive CMGMM adaptation. This approach assumes that the models adapt continuously as soon as new incoming data are received without requiring prior explicit concept drift detection. The adaptation batch size is 50, 100, 150, and 200 data.

4.3.2 Evaluation method and metrics

We evaluate the accuracy of the proposed method performance using prequential evaluation or the interleaved test-then-train method. All the classifiers are trained on the same initial fully labelled training set. In the evaluation, newly incoming data is evaluated in specific windows. The model is constantly tested on the data stream that has not been seen when evaluated in this order. The benefit of this approach does not require a holdout set for testing. It allows us to utilize the existing data efficiently.

4.4 Experiment Result

In this part, we present the experimental result of the proposed method.

4.4.1 Active CMGMM

Table 4.2 presents the experimental results of the active adaptation against two types of feature extraction, namely MFCCs and PANNs. PANN always shows better accuracy than MFCC in AB and GR for both KD3 and ADWIN. The combination of KD3 and PANNs increases model accuracy by 10-11% in GR and 3-10% in AB, and the combination of ADWIN and PANNs increases model accuracy by 8-10% in AB and 2-6% in GR.

Table 4.2. Experiment result of Active CMGMM Adaptation using KD3 and ADWIN

Concept Drift Types	Detector	MFCCs Accuracy				PANNs Accuracy			
		Sc1	Sc2	Sc3	Average	Sc1	Sc2	Sc3	Average
Abrupt (AB)	ADWIN	0.6989	0.6525	0.6214	0.6576	0.7521	0.6748	0.6882	**0.7050**
	KD3	0.6469	0.6359	0.6331	0.6386	0.7492	0.6702	0.6846	**0.7013**
Gradual (GR)	ADWIN	0.6723	0.6341	0.6363	0.6476	0.7783	0.7262	0.7227	**0.7424**
	KD3	0.6999	0.6942	0.6879	0.6940	0.8125	0.8019	0.805	**0.8065**
Recurring Type 1 (R1)	ADWIN	0.7222	0.6948	0.6568	0.6913	0.7394	0.6975	0.7	**0.7123**
	KD3	0.7373	0.7334	**0.7239**	0.7315	0.7231	0.6875	0.6198	0.6768
Recurring Type 2 (R2)	ADWIN	0.7353	0.7066	**0.7089**	0.7169	0.7337	0.6721	0.6876	0.6978
	KD3	0.7321	0.7325	0.7227	0.7291	0.7832	0.7778	0.6442	**0.7351**

On the other hand, different tendencies are observed in the recurring concept drift, R1 and R2. PANNs does not always show better performance than MFCCs. Moreover, the performance difference between PANNs and MFCCs is smaller than one in the case of AB and GR.

4.4.2 Passive CMGMM

Table 3 shows the experimental results of passive adaptation. Overall, PANNs show better accuracy than MFCC in 14 of 20 experiments. MFCC shows better performance in small batch sizes than PANNs. The best performance used in R1 and R2 is obtained with cycle size 200, and that in AB is obtained with cycle size 150. The best accuracy of AB, GR, R1, and R2 using PANNs are 0.7123, 0.7602, 0.7567, and 0.7412, respectively. PANNs tend to have lower performance in rapid adaptation batches.

Table 4.3. Experiment Result of CMGMM without Concept Drift Detector

Concept Drift Types	Batch Size	MFCCs Accuracy				PANNs Accuracy			
		Sc1	Sc2	Sc3	Average	Sc1	Sc2	Sc3	Average
Abrupt (AB)	25	0.629	0.6236	0.6256	**0.6261**	0.6254	0.6095	0.6015	0.6121
	50	0.7122	0.7159	0.7177	**0.7153**	0.7637	0.6905	0.6826	0.7123
	100	0.6285	0.5925	0.5955	0.6055	0.7776	0.6756	0.6842	**0.7125**
	150	0.6361	0.5776	0.5851	0.5996	0.7035	0.6987	0.6775	**0.6932**
	200	0.5580	0.5059	0.5119	0.5253	0.7349	0.6524	0.6827	**0.6900**
Gradual (GR)	25	0.6294	0.6232	0.6004	0.6177	0.6532	0.61058	0.6004	**0.6214**
	50	0.7133	0.7169	0.7115	**0.7139**	0.7225	0.7024	0.712	0.7123
	100	0.6173	0.5887	0.6089	0.6050	0.7287	0.6747	0.6802	**0.6945**
	150	0.6371	0.5818	0.5797	0.5995	0.8135	0.7422	0.7248	**0.7602**
	200	0.5334	0.5094	0.5076	0.5168	0.8024	0.7229	0.7291	**0.7515**
Recurring type 1 (R1)	25	0.6186	0.5799	0.5608	0.5864	0.6115	0.6024	0.5984	**0.6041**
	50	0.7235	0.7320	0.7416	**0.7324**	0.7487	0.7237	0.7207	0.7310

	100	0.7211	0.7105	0.6988	**0.7101**	0.7369	0.6524	0.6309	0.6734
	150	0.7332	0.7086	0.712	0.7179	0.7383	0.7289	0.7127	**0.7266**
	200	0.6639	0.7018	0.7146	0.6934	0.7925	0.7526	0.7251	**0.7567**
Recurring type 2 (R2)	25	0.5133	0.5444	0.5669	0.5415	0.6256	0.6209	0.6054	**0.6173**
	50	0.7396	0.6991	0.7011	**0.7133**	0.7201	0.654	0.6905	0.6882
	100	0.7431	0.7012	0.7023	0.7155	0.7578	0.7399	0.6951	**0.7309**
	150	0.6865	0.6639	0.6803	0.6769	0.7552	0.6881	0.7024	**0.7152**
	200	0.6502	0.6011	0.6089	0.6201	0.7819	0.7524	0.6892	**0.7412**

4.4.3 Comparation active and passive CMGMM

The passive approach shows that the majority of the use of PANN is a better result, but the highest accuracy values for AB, GR, R1, and R2 are in the active method. The results indicate that the concept drift detections make it possible to perform a more optimal adaptation. Moreover, to adapt AB and GR, PANNs should be used as feature vectors.

4.5 Summary

This chapter presents acoustic scene classification using a pre-train model PANNs as feature extractor and CMGMM in four types of concept drift. The experiment result shows that PANNs have better accuracy than MFCCs both in the active and passive approaches. In the active approach, PANNs show significant improvement in abrupt and gradual concept drift. In the passive approach, PANNs tend to have lower performance in rapid adaptation batches. As part of our future work, we plan to compare the usage of PANNs with other classification methods and approaches.

Chapter 5.
Class incremental learning for acoustic scene classification using rehearsal-based strategy

5.1 Background

Concept drift is one of the major challenges in machine learning. One of the concept drift cases is the novel class increment[56]. To deal with the concept drift, it is necessary to manage the memory of past concepts. Memory can be stored in various ways, e.g., as raw data, as representations or as model weights. An efficient memory management strategy should store important information and allows knowledge to be transferred to future tasks.

Chapters 3 and 4 proposed CMGMM as an incremental algorithm that manages new and old information through component merging. In other words, CMGMM manages memory that stored as the model component. The limitation of this solution is linked to specific architectures, which makes their use difficult in other tasks. To have architecture-free incremental learning, we can use a memory management approach through raw data training [57]. However, when we retrain the model to update the model using new data and update its parameters, the model only performs well on the new data. The parameter updates interfere with previously learned experience, leading to a drastic drop in performance on previously learned tasks—this phenomenon is known as catastrophic interference or forgetting [58]. Furthermore, there are also limitations in storing past training data, so not all data can be stored.

Recent studies have shown promising results on rehearsal-based methods that use a small portion of previously learned data can retain learned experience and mitigate catastrophic forgetting [59], [60]. However, this approach can cause overfitting problems to the incoming data because the stored samples are much smaller than the incoming data, or the samples are ignored during training due to their small size. A straightforward solution is to increase the data size as samples arrive gradually, but this does not preserve the important resource constraint of limited storage capacity in the problem settings. Therefore, we need a strategy that maintains the previously learned experience with only a small or limited sample.

This chapter focuses on a framework that allows an ordinary model to be trained incrementally using the rehearsal-based approach to maintaining information of the learned knowledge as much as possible with a few samples to solve class imbalance or overfitting problems. In [61], [62], we find that samples produced by the generative method are accurate enough to retain knowledge. However, the algorithm's efficacy heavily depends on the quality of the generator. So, our approach stores one generator to generate representatives of all the previous samples. Then we use the generator to generate samples and retrain the model with representative data to avoid bias in the model.

5.2 Rehearsal-based incremental learning

In this chapter, we propose an incremental rehearsal-based training framework that enables convolutional network training using rehearsal data consisting of task data, representative data and pseudo-data.

The method is inspired by real-world learning where there is a literal repetition of the exact words that students need to remember, in oral or written form, using simple repetition, cumulative repetition, note-taking, and marking or highlighting text[63]. In the context of computational neural networks, the rehearsal process can be modelled by storing data from previous training and then reusing that data to strengthen the learnt network and accommodate new knowledge into the network[64]. Figure 5.1 illustrates the proposed method from initial to rehearsal training.

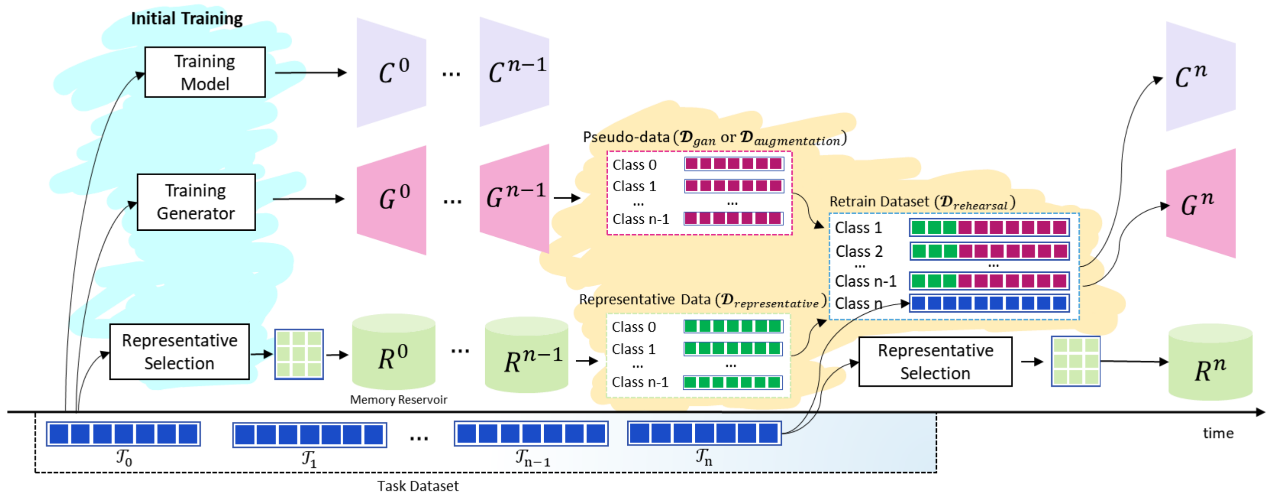


Figure 5.1. Illustration of incremental initial training, a model is trained and a data reservoir and generator are built to generate pseudo-data for the next training.

In this experiment, we consider a sequence of tasks $\mathcal{T} = (\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_N)$ of N tasks for training. Each of task $\mathcal{T}_t$ has dataset $\mathcal{D}^t_{task} = \{x^t_n, y^t_n\}^{N_t}_{n=1}$ that contain N_t datapoints and labels. In $\mathcal{T}_t$, we assume that y^t has unique classes, $y^t \cap \{y^0 .. y^{t-1}\} = \emptyset$. In the training process, the t-th task is only accesible at step t only and we also try to minimize standar loss function (e.g. cross-entropy loss).

The first step of this framework is train classifier M, select representative data $\mathcal{D}_{representative}$, and train generator G. After the initial training is carried out, the next stage is performed increamental training to update the knowledge in M. The details of the process are as follows

5.2.1 Classifier

In the experiment we use a convolutional neural networks (CNN) based model to classify a audio scene. The CNN consists of several convolutional layers where each convolutional layer contains several kernels that are convolved with the input feature maps to capture their local patterns. Detail network architecture showed in Fig. 5.2.

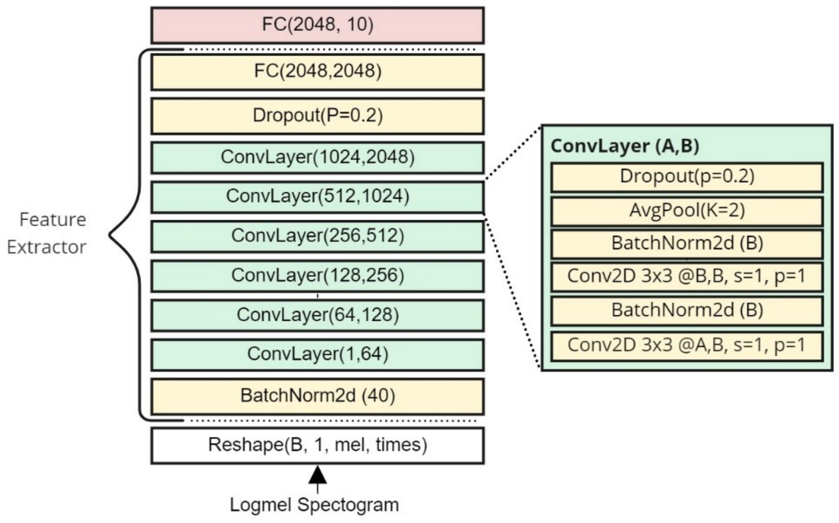


Figure 5.2. Classifier Architecture

A classifier M consists of feature extractor F and classifier head C, $M(F(x;\theta);C)$. To classify data x, M extract feature u using F that parameterise by θ. C needs V as a matrix that projects u to class score using $\mathcal{A}$ softmax function, $C = \mathcal{A}(Vu)$, to classify the feature. The F itself contains 6 convolutional layers inspired by the VGG-like CNNs. Each convolutional block consists of 2 convolutional layers with a kernel size of 3 × 3. Batch normalization is applied between each convolutional layer, and the ReLU nonlinearity is used to speed up and stabilize the training. We apply average pooling of size of 2 × 2 to each convolutional block for downsampling. Finally on top of F we add a fully connected layer to the fixed length vectors to extract features.

5.2.2 Representative Data

Representative data is a subset of representative samples of the previous dataset, $\mathcal{D}_{representative} \subset \mathcal{D}_{task}$. When the M is trained, the representative data $\mathcal{D}^t_{representative}$ is selected from $\mathcal{D}^t_{task}$ and they are stored in the data reservoir R^t. $\mathcal{D}^t_{representative}$ was selected using several methods:

- **High or low logits**. This method uses the number generated from $\mathcal{A}$ that indicates the level of probability of the classification result. The higher the value, the more likely the classifier is to be correctly classified. Using a high logit means we focus on storing learning data with a high probability of being classified correctly. Conversely, if we choose a low probability, we store the representative data containing data that is likely to be misclassified. In simple words, using this method the model can choose to rehearse with data that is often misclassified or always correctly classified.

- **Mean clustering**. Mean clustering utilizes the average feature u to select data stored in the data reservoir. The smaller the distance from the average means that the data selected is data that frequently appears in the dataset.
- **Barycenter**. The concept used in this method is similar to using mean clustering. However, the samples selected are samples whose u are the closest to their moving barycenter distance [65].
- **Random selections**. Randomly selected samples from the current dataset.

5.2.3 Pseudo-data Generator

Pseudo-data is a set of generated samples that are used in the retraining process. In this chapter, pseudo-data can be generated in two ways: using the generative method and data augmentation. In the generative method, we train our generator G^t using Memory Replay GAN (MER-GAN)[62] to generate dataset $\mathcal{D}_{gan}^t$. Generative adversarial networks (GAN) are a popular framework for image generation due to their capability to learn a mapping between a low-dimensional latent space and a complex distribution of interest, such as natural images. A GAN consists of two networks, a Generator, and a Discriminator, competing with each other in a zero-sum game framework.

MER-GAN consists of three components: generator, discriminator, and classifier. The discriminator and classifier share all layers but the last ones (task-specific layers). The conditional generator is parametrized by θ^G and generate a sample $\tilde{x} = G_{\theta^G}(z, c)$ given a latent vector z and a class c. Similarly, the discriminator parametrized by θ^D tries to discern whether an input x is real or generated, while the generator tries to fool it by generating a more realistic sample. In addition, MER-GAN uses an auxiliary classifier C with parameter θ^C to predict the label $\tilde{c} = C_{\theta^C}(x)$, and thus forcing the generator to generate images that can be classified in the same way as real images

To train MER-GAN, we use join-retraining with rehearsal dataset $\mathcal{D}_{rehearsal}$. The generator has an active role by replaying data of previous tasks via generative sampling and then using them during the training of the current task to prevent forgetting.

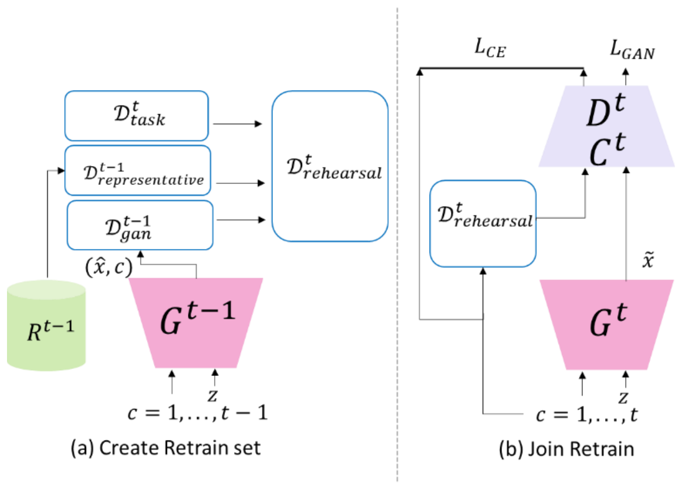


Figure 5.3. (a) Rehearsal data construction.(b) In incremental GAN training using join retraining.

Firstly, we generate dataset $\mathcal{D}^{t-1}_{gan}$ contain generated sample from all previous tasks from task 0 to $t-1$. Then we combine $\mathcal{D}^{t-1}_{gan}$, $\mathcal{D}^{t-1}_{representative}$, and $\mathcal{D}^t_{task}$ as retrain dataset $\mathcal{D}^t_{rehearsal}$. The illustration of the retraining procedure with GAN can be found in Fig. 5.2b and Eq. 5.1 show the content of rehearsal dataset using GAN.

$$\mathcal{D}^t_{rehearsal} = \mathcal{D}^t_{task} \cup \mathcal{D}^{t-1}_{representative} \cup \mathcal{D}^{t-1}_{gan} \tag{5.1}$$

Once the extended dataset is created, the network is trained using joint training as

$$\min_{\theta^G_t}(L^G_{GAN}(\theta_t, \mathcal{D}^t_{rehearsal}) + \lambda_{CLS} L^G_{CLS}(\theta_t, \mathcal{D}^t_{retrain})) \tag{5.2}$$

$$\min_{\theta^D_t}(L^D_{GAN}(\theta_t, \mathcal{D}^t_{rehearsal}) + \lambda_{CLS} L^D_{CLS}(\theta_t, \mathcal{D}^t_{rehearsal})) \tag{5.3}$$

$$L^G_{GAN}(\theta_t, \mathcal{D}^t_{rehearsal}) = -\mathbb{E}_{z\sim p_z, c\sim p_c}\left[D_{\theta^D}(G_{\theta^G}(z,c))\right] \tag{5.4}$$

$$L^G_{CLS}(\theta_t, \mathcal{D}^t_{rehearsal}) = -\mathbb{E}_{z\sim p_z, c\sim p_c}\left[y_c log\ C_{\theta^C}(G_{\theta^G}(z,c))\right] \tag{5.5}$$

$$\begin{aligned} L^D_{GAN}(\theta_t, \mathcal{D}^t_{rehearsal}) \\ &= -\mathbb{E}_{(x,c)\sim S}\left[D_{\theta^D}(x)\right] + \mathbb{E}_{z\sim p_z, c\sim p_c}\left[D_{\theta^D}(G_{\theta^G}(z,c))\right] \\ &+ \lambda_{GP}\mathbb{E}_{x\sim S, z\sim p_z, c\sim p_c, \epsilon\sim p_\epsilon}\left[\left(\left\|\nabla D_{\theta^D}(\epsilon x + (1-\epsilon)G_{\theta^G}(z,c))\right\| - 1\right)^2\right] \end{aligned} \tag{5.6}$$

$$L^D_{CLS}(\theta_t, \mathcal{D}^t_{rehearsal}) = -\mathbb{E}_{(x,c)\sim S}[\ C_{\theta^C}(G_{\theta^G}(z,c))] \tag{5.7}$$

The GAN loss uses the WGAN formulation with gradient penalty where $L^G_{GAN}(\theta_t, \mathcal{D}^t_{rehearsal})$ and $L^G_{CLS}(\theta_t, \mathcal{D}^t_{rehearsal})$ are loss for generator and the cross entropy loss for classification, respectively,

$p_c = U(1,t), p_z = N(0,1)$ are the sampling distributions (uniform and Gaussian, respectively), y_c is the one-hot encoding of c for computing the cross-entropy, ϵ are parameters of the gradient penalty term sampled as $p_\epsilon = U(0,1)$ and the last term of L_{GAN}^{D} is the gradient penalty.

In the augmentation method, the dataset $\mathcal{D}_{rehearsal}^{t}$ contain $\mathcal{D}_{task}^{t}$, $\mathcal{D}_{representative}^{t-1}$ and $\mathcal{D}_{augmentation}^{t-1}$. $\mathcal{D}_{augmentation}^{t-1}$ is the augmentation result of $\mathcal{D}_{representative}^{t-1}$ using Frequency masking[66]. This method masks several blocks of consecutive frequency channels by a uniform distribution. Eq. 5.8 show the content of rehearsal dataset using augmentation.

$$\mathcal{D}_{rehearsal}^{t} = \mathcal{D}_{task}^{t} \cup \mathcal{D}_{representative}^{t-1} \cup \mathcal{D}_{augmentation}^{t-1} \tag{5.8}$$

5.3 Experiment Setup

We use the TAU Urban Acoustic Scenes 2019. The dataset has 10 classes divided into 5 tasks, where each task contains two unique classes. In the experiment, we compared the use of Pseudo-data with GAN alone with small and large representative data $\mathcal{D}_{representative}$. The $|\mathcal{D}_{representative}|$ for small representative data is 10% of $|\mathcal{D}_{task}|$ and for small representative data is 75% of $|\mathcal{D}_{task}|$. Finally, we have five experiment scenarios: GAN Alone (GAN-Alone); Small representative data and GAN (SmallRep+GAN), Small representative data and augmentation (SmallRep+AUG), Large representative data and GAN (LargeRep+GAN) and Large representative data and augmentation (LargeRep+AUG). Table 5.1 summarize the experiment scenario and its data size configuration.

Table 5.1. Experiment Scenario configuration

Experiment Scenario	$\mathcal{D}_{representative}$	$\mathcal{D}_{gan}$	$\mathcal{D}_{augmentation}$
GAN-Alone	-	100%	-
SmallRep+GAN	10%	90%	-
SmallRep+AUG	10%	-	90%
LargeRep+GAN	75%	25%	-
LargeRep+AUG	75%		75%

To measure the proposed method, we use average accuracy (Eq. 5.9) and Backward Transfer (BWT). BWT measures the influence that learning a task has on the performance of previous tasks where *N* is the number of tasks and $Accuracy_{i,i}$ is the test accuracy score for task j after the model learned task i.

$$ACC = \frac{1}{N}\sum_{i=1}^{N} Accuracy_{N,i} \quad (5.9)$$

$$BWT = \frac{1}{N-1}\sum_{i=1}^{N-1} Accuracy_{N,i} - Accuracy_{i,i} \quad (5.10)$$

We train our classifier task for 100 epochs and GANs for 500 epochs. The Adam optimizer is used in all experiments, and the learning rate for classifier and GANs are 1e-4 and 1e-4, respectively.

5.4 Experiment Result

Using GAN-Alone to regenerate samples from prior knowledge shows that this strategy can overcome catastrophic forgetting, but the model's performance degrades over time. The overall average accuracy using GAN-Alone is 0.733. Figure 5.4 shows that the accuracy of GAN-Alone decreases as the number of classes increases. Furthermore, according to the detailed analysis, the task accuracy results show no positive backward transfer in retraining. A positive backward transfer is the influence of the current training that improves the accuracy of the previous tasks. So, GAN-Alone is not recommended in high incremental phases.

The use of $\mathcal{D}_{representative}$ both with $\mathcal{D}_{gan}$ or $\mathcal{D}_{augmentation}$ can be used to overcome the problem of data degradation. The higher number of $\mathcal{D}_{representative}$, the higher its accuracy. The experimental also results show that the use of $\mathcal{D}_{gan}$ outperforms $\mathcal{D}_{augmentation}$ both in large and small representative data. When using this combination, the classifier's performance improved and showed more stable accuracy, and some backward transfer was found, both in large and small representative data. The average accuracy of the GAN combination in SmallRep+GAN and LargeRep+GAN were 0.7651 and 0.8418, respectively.

For small representative data using GAN (SmallRep+GAN), the low logit selection method shows the highest average accuracy of 0.7805, while the Random Selection method show the lowest mean accuracy of 0.754. On average, all selection method in SmallRep+GAN shows better average accuracy than GAN-Alone, but SmallRep+AUG, only the cluster selection method shows better results than GAN-Alone by 0.741. Therefore, using augmentation on a small amount of representative data is not recommended. The comparison of SmallRep+GAN and SmallRep+AUG can be shown in Fig.5.4

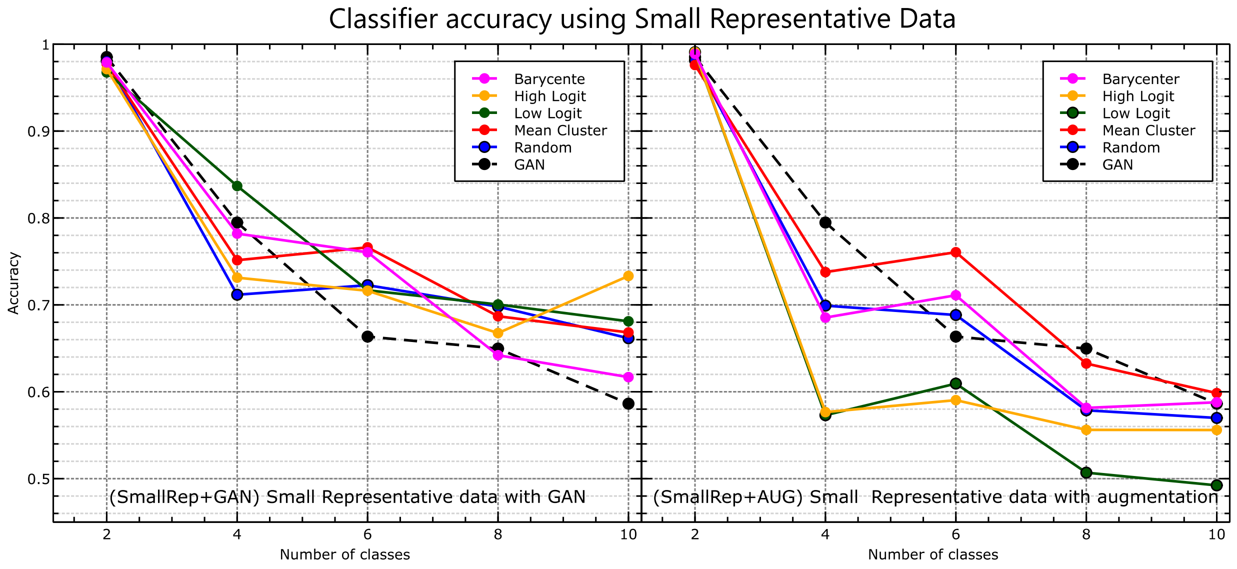


Figure 5.4. Classifier accuracy in small representative data

In large representative data, LargeRep+GAN using the random selection method shows the best results with an average accuracy of 0.8581. In addition, other methods also get high average results such as low logit of 0.8536 and barycenter of 0.8567. Although the use of LargeRep+AUG has a lower accuracy than LargeRep+GAN, the accuracy is quite good, and there is much backward transfer so that it keeps its performance stable. The method with the most significant average accuracy on augmentation is the mean cluster of 0.7735. The comparison of LargeRep+GAN and LargeRep+AUG can be shown in Fig.5.5

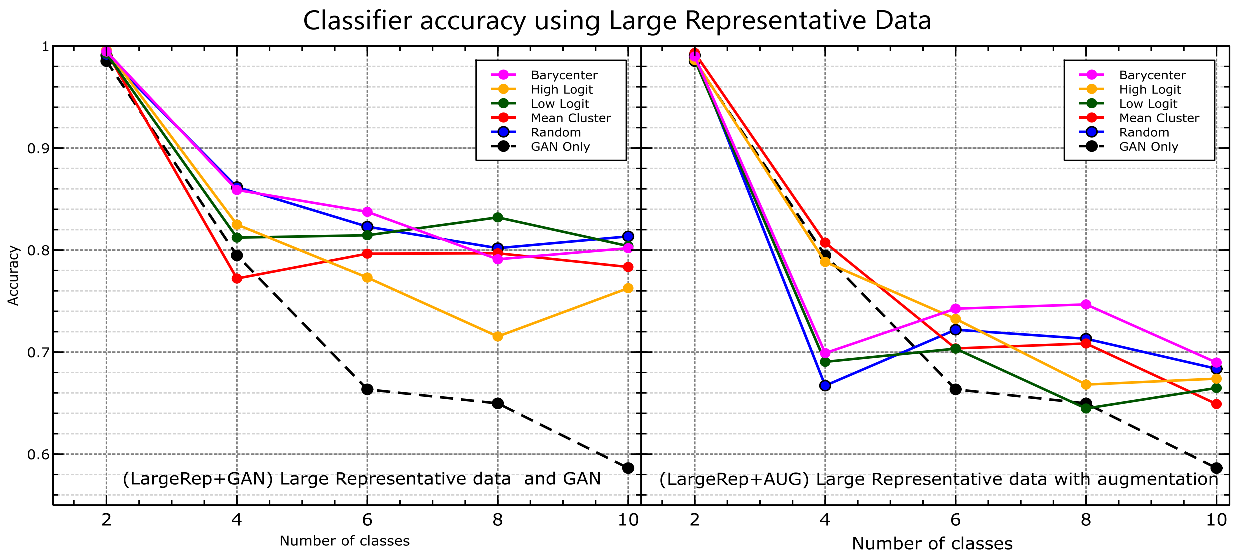


Figure 5.5. Average accuracy in large representative data

Positive BWT result

In this experiment, it is clear that the larger number of representative memories the larger the positive BWT. The most significant positive BWT was obtained using High logit on LargeRep+GAN and SmallRep+GAN of 0.0222 and 0.0041, respectively. Table 5.2 show the average BWT in all scenarios.

Table 5.2. BWT result

SCENARIO	Representative Selection method				
	High logit	Low logit	Random	Bary Center	Cluster
SmallRep+GAN	**0.0041**	0	0	0	0
SmallRep+AUG	0	0	0	**0.0116**	0
LargeRep+GAN	**0.0223**	0.0037	0.0055	0.0141	0.0034
LargeRep+AUG	0.0134	**0.0303**	0.0085	0.0470	0.0399

Storage requirement

We also compare the storage usage of the original dataset, representative data, and GAN model size. The original dataset stores 3.9 GB of audio per class. For large and small $\mathcal{D}_{representative}$ require 73.6 MB and 16 MB per class. Our generator G requires a constant storage requirement of 82.85 MB for all classes. So, by using a generator, the size of storage used remains the same, even if the number of experience models increases over time.

5.5 Summary

In this chapter, we propose a framework to train acoustic scene classifiers using a rehearsal-based strategy incrementally. This method consists of three main components: classifier, rehearsal data and pseudo-data. Rehearsal data obtained by selecting the small portion of the past dataset as representative data. Then we use GAN to generate the samples as an extension of the representative data to avoid overfitting. Experimental results show promising results, prevent catastrophic forgetting, and increase backward transfer. The model performs better when using a low logit sample rather than a high logit in selecting the representative data.

Chapter 6.

Conclusion

6.1 Conclusion

In this thesis we develop algorithms and a framework for acoustic scene classification in concept drift situations. The objective of the proposed methods is to maintain model performance, even in the presence of concept drifts, by detecting drifts and adapting the classification system to the new concepts.

The first proposed methods are CMGMM (Combine Merge Gaussian Mixture Model) and KD3 (Kernel Density Drift Detection). CMGMM is an incremental algorithm that has the ability to adapt to concept drift by adding or modifying its component, and KD3 is a window-based concept drift detector based on kernel density. The main advantages of CMGMM are adaptation and continuous learning from stream data with a local replacement strategy to preserve previously learned knowledge and avoid catastrophic forgetting. The experiment results showed that the combination of CMGMM and KD3 can adapt to concept drift and maintain the model performance over time.

In addition, CMGMM can also be used in active mode (by using a concept drift detector) and passive mode (time-specific adaptation). The use of the active mode is recommended in cases where unpredictable concept drift is more efficient to use. Understanding the characteristics of concept drift will also help to improve performance. In the case of Abrupt, a sensitive detector is needed, but on the contrary, in the case of recurring, a less-sensitive detector performs better.

In cases where the time or location of the concept drift can be predicted, the use of a passive adaptation strategy is more beneficial and has a lower computational cost than the active strategy. However, if the adaptation cycle is too far from the concept drift, then the model performance will decrease over time.

This paper thesis also improves CMGMM by using a pre-train model PANNs as a feature extractor that processes the log-mel spectrogram of an audio scene to produce features containing the occurrence probability of a particular sound event in the scene. CMGMM uses these vectors in the training and adaptation process. The experiment result shows that the usage PANNs with CMGMM have better accuracy than MFCCs both in the active and passive approaches. In the active approach, PANNs significantly improve abrupt and gradual concept drift. In the passive approach, PANNs tend to perform less in rapid adaptation batches.

In other cases of concept drift where a new class comes into existence, we propose a framework to perform incremental learning using a rehearsal strategy. In this framework, we use representative data and GAN to improve model performance and positive backward transfer. The advantage of this framework is that it can overcome the catastrophic forgetting problem and improve backward transfer with relatively stable storage size. Experimental results show that the size of representative data affects accuracy and backward transfer. In experiments without representative data, only pseudo-data, the performance of the model continues to decline, so the use of this method is not suitable for a large number of tasks.

On the other hand, when using representative data, the performance is more stable. By combining representative data and pseudo-data, there is backward transfer even though the amount of representative data is minimal. On a small amount of representative data, using the low-logit method in selecting representative data gives the best results, while on a large representative data, the random method shows the best results. In addition to representative data, using GAN as a pseudo-data generator also shows better results, especially if the amount of representative data is small. This is because the sample variation generated by GAN is better than augmentation.

6.2 Future Work

This thesis identifies the following directions as future work:

- Feature evolving analysis. The performance of CMGMM depends on the type of concept drift that occurs so that adjustments can be made to the concept drift detector or adaptation cycle to get better results. Therefore, a tool is needed to analyse the type of drift that will occur in the data.
- Compressed representative data. Rehearsal performance is affected by the size of the representative data so that the larger the amount, the performance will increase. Therefore, it is important to compress the data.

References

[1] Richard F. Lyon, *Human And Machine Hearing*. Cambridge University Press, 2017.

[2] R. Elwell and R. Polikar, “Incremental Learning of Concept Drift in Nonstationary Environments,” *IEEE Trans Neural Netw*, vol. 22, no. 10, pp. 1517–1531, 2011, doi: 10.1109/TNN.2011.2160459.

[3] S. Ntalampiras, “Automatic analysis of audiostreams in the concept drift environment,” Sep. 2016. doi: 10.1109/MLSP.2016.7738905.

[4] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, “A Survey on Concept Drift Adaptation,” *ACM Comput Surv*, vol. 46, no. 4, pp. 1–37, 2014, doi: 10.1145/2523813.

[5] T. R. Hoens, R. Polikar, and N. v. Chawla, “Learning from Streaming Data with Concept Drift and Imbalance: An Overview,” *Progress in Artificial Intelligence*, vol. 1, no. 1, pp. 89–101, 2012, doi: 10.1007/s13748-011-0008-0.

[6] I. Žliobaitė, “Learning under Concept Drift: an Overview,” pp. 1–36, 2010, doi: 10.1002/sam.

[7] C. Chen *et al.*, “Improving Accuracy of Evolving GMM under GPGPU-Friendly Block-Evolutionary Pattern,” *Intern J Pattern Recognit Artif Intell*, vol. 34, no. 3, pp. 1–34, 2020, doi: 10.1142/S0218001420500068.

[8] P. Gaunard, C. G. Mubikangiey, C. Couvreur, and V. Fontaine, “Automatic classification of environmental noise events by hidden Markov models,” in *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP ’98 (Cat. No.98CH36181)*, vol. 6, pp. 3609–3612. doi: 10.1109/ICASSP.1998.679661.

[9] J. Xiang, M. F. McKinney, K. Fitz, and T. Zhang, “Evaluation of sound classification algorithms for hearing aid applications,” in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2010, pp. 185–188. doi: 10.1109/ICASSP.2010.5496064.

[10] S. Ravindran and D. V. Anderson, “Audio Classification And Scene Recognition and for Hearing Aids,” in *2005 IEEE International Symposium on Circuits and Systems*, pp. 860–863. doi: 10.1109/ISCAS.2005.1464724.

[11] D. P. W. Ellis and K. Lee, "Minimal-impact audio-based personal archives," in *Proceedings of the the 1st ACM workshop on Continuous archival and retrieval of personal experiences - CARPE'04*, 2004, p. 39. doi: 10.1145/1026653.1026659.

[12] K. El-Maleh, A. Samouelian, and P. Kabal, "Frame level noise classification in mobile environments," in *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No.99CH36258)*, 1999, pp. 237–240 vol.1. doi: 10.1109/ICASSP.1999.758106.

[13] H. I. Jo and J. Y. Jeon, "Urban soundscape categorization based on individual recognition, perception, and assessment of sound environments," *Landsc Urban Plan*, vol. 216, p. 104241, Dec. 2021, doi: 10.1016/j.landurbplan.2021.104241.

[14] S. Micevska, A. Awad, and S. Sakr, "SDDM: an interpretable statistical concept drift detection method for data streams," *J Intell Inf Syst*, 2021, doi: 10.1007/s10844-020-00634-5.

[15] B. Schilit, N. Adams, and R. Want, "Context-Aware Computing Applications," in *1994 First Workshop on Mobile Computing Systems and Applications*, Dec. 1994, pp. 85–90. doi: 10.1109/WMCSA.1994.16.

[16] Y. Xu, W. J. Li, and K. K. C. Lee, *Intelligent Wearable Interfaces*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2008. doi: 10.1002/9780470222867.

[17] S. Chu, S. Narayanan, C. C. Jay Kuo, and M. J. Matarić, "Where am I? Scene recognition for mobile robots using audio features," *2006 IEEE International Conference on Multimedia and Expo, ICME 2006 - Proceedings*, vol. 2006, no. July, pp. 885–888, 2006, doi: 10.1109/ICME.2006.262661.

[18] I. Damnjanovic, J. Reiss, and D. Barry, "Enabling access to sound archives through integration, enrichment and retrieval," in *2008 IEEE International Conference on Multimedia and Expo*, Jun. 2008, pp. 1597–1598. doi: 10.1109/ICME.2008.4607756.

[19] D. Wang and G. J. Brown, "Computational auditory scene analysis: Principles, algorithms, and applications," *Computational Auditory Scene Analysis: Principles, Algorithms, and Applications*, pp. 1–395, 2006, doi: 10.1109/9780470043387.

[20] K. Imoto, "Introduction to acoustic event and scene analysis," *Acoust Sci Technol*, vol. 39, no. 3, pp. 182–188, May 2018, doi: 10.1250/ast.39.182.

[21] N. Kern and B. Schiele, "Context-aware notification for wearable computing," in *Seventh IEEE International Symposium on Wearable Computers, 2003. Proceedings.*, pp. 223–230. doi: 10.1109/ISWC.2003.1241415.

[22] B. Clarkson, N. Sawhney, and A. Pentland, "Auditory Context Awareness via Wearable Computing," *In Proceedings of The 1998 Workshop On Perceptual User Interfaces*, pp. 1–6, 1998, [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.28.8726

[23] A. O. Frank, J. Ward, N. J. Orwell, C. Mccullagh, and M. Belcher, "Introduction of a new NHS electric-powered indoor/outdoor chair (EPIOC) service: benefits, risks and implications for prescribers," *Clin Rehabil*, vol. 14, no. 6, pp. 665–673, Dec. 2000, doi: 10.1191/0269215500cr376oa.

[24] J. Abeßer, S. I. Mimilakis, R. Gräfe, and H. Lukashevich, "Acoustic Scene Classification by Combining Autoencoder-Based Dimensionality Reduction and Convolutional Neural Networks," *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, no. November, pp. 7–11, 2017.

[25] P. Guyot, J. Pinquier, and R. Andre-Obrecht, "Water sound recognition based on physical models," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 793–797. doi: 10.1109/ICASSP.2013.6637757.

[26] T. Zhang and C.-C. J. Kuo, "Audio content analysis for online audiovisual data segmentation and classification," *IEEE Transactions on Speech and Audio Processing*, vol. 9, no. 4, pp. 441–457, May 2001, doi: 10.1109/89.917689.

[27] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with Drift Detection," 2004, pp. 286–295. doi: 10.1007/978-3-540-28645-5_29.

[28] A. Narasimhamurthy and L. I. Kuncheva, "A framework for generating data to simulate changing environments," in *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications, AIA 2007*, 2007, pp. 384–389.

[29] I. Žliobaitė, "Learning under Concept Drift: an Overview," Oct. 2010, doi: https://doi.org/10.48550/arXiv.1010.4784.

[30] N. Lu, G. Zhang, and J. Lu, "Concept drift detection via competence models," *Artif Intell*, vol. 209, no. 1, pp. 11–28, 2014, doi: 10.1016/j.artint.2014.01.001.

[31] J. C. Schlimmer and R. H. Granger, “Incremental learning from noisy data,” *Mach Learn*, vol. 1, no. 3, pp. 317–354, Sep. 1986, doi: 10.1007/BF00116895.

[32] D. Brzezinski and J. Stefanowski, “Reacting to different types of concept drift: The accuracy updated ensemble algorithm,” *IEEE Trans Neural Netw Learn Syst*, vol. 25, no. 1, pp. 81–94, 2014, doi: 10.1109/TNNLS.2013.2251352.

[33] L. I. Kuncheva, “Classifier ensembles for detecting concept change in streaming data: Overview and perspectives,” in *Proceedings of the 2nd Workshop SUEMA*, 2008, pp. 5–10.

[34] A. Bifet and R. Gavaldà, “Learning from Time-changing Data with Adaptive Windowing,” *Proceedings of the 7th SIAM International Conference on Data Mining*, no. April, pp. 443–448, 2007, doi: 10.1137/1.9781611972771.42.

[35] I. Frías-Blanco, J. del Campo-Ávila, G. Ramos-Jiménez, R. Morales-Bueno, A. Ortiz-Díaz, and Y. Caballero-Mota, “Online and Non-Parametric Drift Detection Methods Based on Hoeffding’s Bounds,” *IEEE Trans Knowl Data Eng*, vol. 27, no. 3, pp. 810–823, 2015, doi: 10.1109/TKDE.2014.2345382.

[36] C. Raab, M. Heusinger, and F. M. Schleif, “Reactive Soft Prototype Computing for Concept Drift Streams,” *Neurocomputing*, Apr. 2020, doi: 10.1016/j.neucom.2019.11.111.

[37] A. R. Runnalls, “Kullback-Leibler Approach to Gaussian Mixture Reduction,” *IEEE Trans Aerosp Electron Syst*, vol. 43, no. 3, pp. 989–999, 2007, doi: 10.1109/TAES.2007.4383588.

[38] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.

[39] J. Patterson and A. Gibson, *Deep learning: a practitioners approach*. OReilly, 2017.

[40] S. Mahdizadehaghdam, A. Panahi, and H. Krim, “Sparse generative adversarial network,” *Proceedings - 2019 International Conference on Computer Vision Workshop, ICCVW 2019*, pp. 3063–3071, 2019, doi: https://doi.org/10.48550/arXiv.1406.2661.

[41] T. Zhang, J. Liang, and B. Ding, “Acoustic scene classification using deep CNN with fine-resolution feature,” *Expert Syst Appl*, vol. 143, no. 1, p. 113067, 2020, doi: 10.1016/j.eswa.2019.113067.

[42] A. Tsymbal, “The Problem of Concept Drift: Definitions and Related Work,” 2004.

[43] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum Likelihood from Incomplete Data via the EM Algorithm,” *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, Sep. 1977, [Online]. Available: http://www.jstor.org/stable/2984875

[44] A. D. R. McQuarrie and C.-L. Tsai, *Regression and Time Series Model Selection*, vol. 6, no. 2. WORLD SCIENTIFIC, 1998. doi: 10.1142/3573.

[45] P. Pal Singh, “An Approach to Extract Feature using MFCC,” *IOSR Journal of Engineering*, vol. 4, no. 8, pp. 21–25, 2014, doi: 10.9790/3021-04812125.

[46] C. Fraley, “How Many Clusters? Which Clustering Method? Answers Via Model-Based Cluster Analysis,” *Comput J*, vol. 41, no. 8, pp. 578–588, 1998, doi: 10.1093/comjnl/41.8.578.

[47] J. L. Williams and P. S. Maybeck, “Cost-function-based Gaussian mixture reduction for target tracking,” *Proceedings of the 6th International Conference on Information Fusion*, vol. 2, pp. 1047–1054, 2003, doi: 10.1109/ICIF.2003.177354.

[48] E. Parzen, “On Estimation of a Probability Density Function and Mode,” *The Annals of Mathematical Statistics*, vol. 33, no. 3, pp. 1065–1076, Sep. 1962, doi: 10.1214/aoms/1177704472.

[49] A. Mesaros, T. Heittola, and T. Virtanen, “TUT Acoustic scenes 2017, Development dataset,” Mar. 2017. doi: 10.5281/ZENODO.400515.

[50] J. Salamon, C. Jacoby, and J. P. Bello, “A Dataset and Taxonomy for Urban Sound Research,” *MM 2014 - Proceedings of the 2014 ACM Conference on Multimedia*, no. November, pp. 1041–1044, 2014, doi: 10.1145/2647868.2655045.

[51] BBC, “BBC Sound Effects.” https://sound-effects.bbcrewind.co.uk/search (accessed Mar. 20, 2019).

[52] I. D. Id, M. Abe, and S. Hara, “Evaluation of concept drift adaptation for acoustic scene classifier based on Kernel Density Drift Detection and Combine Merge Gaussian Mixture Model,” May 2021. doi: 10.5281/zenodo.4595866.

[53] J. F. Gemmeke *et al.*, “Audio Set: An ontology and human-labeled dataset for audio events,” *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pp. 776–780, 2017, doi: 10.1109/ICASSP.2017.7952261.

[54] I. D. Id, M. Abe, and S. Hara, “Concept Drift Adaptation for Acoustic Scene Classifier Based on Gaussian Mixture Model,” in *2020 IEEE REGION 10 CONFERENCE (TENCON)*, Nov. 2020, pp. 450–455. doi: 10.1109/TENCON50793.2020.9293766.

[55] I. D. Id, M. Abe, and S. Hara, “Acoustic Scene Classifier Based on Gaussian Mixture Model in the Concept Drift Situation,” *Advances in Science, Technology and Engineering Systems Journal*, vol. 6, no. 5, pp. 167–176, Sep. 2021, doi: 10.25046/aj060519.

[56] M. Masud, J. Gao, L. Khan, J. Han, and B. M. Thuraisingham, “Classification and Novel Class Detection in Concept-Drifting Data Streams under Time Constraints,” *IEEE Trans Knowl Data Eng*, vol. 23, no. 6, pp. 859–874, Jun. 2011, doi: 10.1109/TKDE.2010.61.

[57] B. Gepperth, Alexander Hammer, “Incremental learning algorithms and applications,” 2016. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01418129

[58] M. McCloskey and N. J. Cohen, “Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem,” *Psychology of Learning and Motivation*, vol. 24, pp. 109–165, 1989, doi: 10.1016/S0079-7421(08)60536-8.

[59] J. Yoon, D. Madaan, E. Yang, and S. J. Hwang, “Online Coreset Selection for Rehearsal-based Continual Learning,” in *International Conference on Learning Representations (ICLR)*, 2021, pp. 1–16.

[60] A. Prabhu, P. H. S. Torr, and P. K. Dokania, “GDumb: A Simple Approach that Questions Our Progress in Continual Learning,” in *ECCV 2020*, 2020, pp. 524–540.

[61] H. Shin, J. K. Lee, J. Kim, and J. Kim, “Continual Learning with Deep Generative Replay,” in *NIPS’17: Proceedings of the 31st International Conference on Neural Information Processing Systems*, May 2017, pp. 2994–3003. [Online]. Available: http://arxiv.org/abs/1705.08690

[62] C. Wu, L. Herranz, X. Liu, Y. Wang, J. van de Weijer, and B. Raducanu, “Memory Replay GANs: learning to generate images from new categories without forgetting,” in *32nd International Conference on Neural Information Processing*, Sep. 2018, pp. 1–12.

[63] L. M. Joseph, “Incremental Rehearsal: A Flashcard Drill Technique for Increasing Retention of Reading Words,” *Read Teach*, vol. 59, no. 8, pp. 803–807, May 2006, doi: 10.1598/RT.59.8.8.

[64] D. Muñoz, C. Narváez, C. Cobos, M. Mendoza, and F. Herrera, “Incremental learning model inspired in Rehearsal for deep convolutional networks,” *Knowl Based Syst*, vol. 208, p. 106460, Nov. 2020, doi: 10.1016/j.knosys.2020.106460.

[65] S. A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, “iCaRL: Incremental classifier and representation learning,” in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017, pp. 5533–5542. doi: 10.1109/CVPR.2017.587.

[66] D. S. Park *et al.*, “SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition,” in *Interspeech 2019*, Sep. 2019, pp. 2613–2617. doi: 10.21437/Interspeech.2019-2680.

List of Publications

Refereed Papers

[P1] Ibnu Daqiqil ID, Masanobu Abe, Sunao Hara. “Concept Drift Adaptation for Acoustic Scene Classifier Based on Gaussian Mixture Model”. Proceedings of the 2020 IEEE Region 10 Conference (TENCON), 2020

[P2] Ibnu Daqiqil ID, Masanobu Abe, Sunao Hara. “Acoustic Scene Classifier Based on Gaussian Mixture Model in the Concept Drift Situation”. Advances in Science, Technology and Engineering Systems Journal Vol. 6, No. 5, 167-176, 2021

[P3] Ibnu Daqiqil ID, Masanobu Abe, Sunao Hara. “Concept drift adaptation for audio scene classification using high-level features”. Proceedings of the 2022 IEEE International Conference on Consumer Electronics (ICCE), 2022

[P4] Ibnu Daqiqil ID, Masanobu Abe, Sunao Hara. “Incremental audio scene classifier using rehearsal-based strategy”. Proceedings of the 2022 IEEE 11th Global Conference on Consumer Electronics(GCCE), 2022

Other Papers

[P5] Ibnu Daqiqil ID, Masanobu Abe, Sunao Hara. “Evaluation of concept drift adaptation for acoustic scene classifier based on Kernel Density Drift Detection and Combine Merge Gaussian Mixture Model”. Proceedings of the Acoustical Society of Japan Spring Meeting 2021