Small footprint automatic speech recognition free keyword spotting

Thesis submitted to the

Indian Institute of Technology, Bhubaneswar

For award of the degree

of

Master of Technology

by

Bijoyashree Das
23SP06004

Under the guidance of

Dr. Himanshu Pramod Padole

SCHOOL OF ELECTRICAL & COMPUTER SCIENCES

INDIAN INSTITUTE OF TECHNOLOGY, BHUBANESWAR

May 2025

APPROVAL OF THE VIVA-VOCE BOARD

05/05/2025

Certified that the thesis entitled **Small footprint automatic speech recognition free keyword spotting**, submitted by **Bijoyashree Das** to the Indian Institute of Technology Bhubaneswar, for the award of the degree of Master of Technology has been accepted by the external examiners and that the student has successfully defended the thesis in the viva-voce examination held today.

(Examiner 1)

(Examiner 2)

(Supervisor)

CERTIFICATE

This is to certify that the thesis entitled **Small footprint automatic speech recognition free keyword spotting**, submitted by **Bijoyashree Das** to Indian Institute of Technology Bhubaneswar, is a record of bonafide research work under my supervision and I consider it worthy of consideration for the award of the degree of Master of Technology of the Institute.

Date:05/05/2025

Dr. Himanshu Pramod Padole
Assistant Professor
School of Electrical & Computer Sciences
Indian Institute of Technology Bhubaneswar
Bhubaneswar, India

DECLARATION

I certify that

1. the work contained in the thesis is original and has been done by myself under the general supervision of my supervisor.
2. the work has not been submitted to any other institute for any degree or diploma.
3. I have followed the guidelines provided by the institute in writing the thesis.
4. I have conformed to the norms and guidelines given in the ethical code of conduct of the institute.
5. wherever I have used materials (data, theoretical analysis, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.
6. whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Bijoyashree Das .

Bijoyashree Das

Contents

List of Figures

List of Abbreviations

KWS	Keyword spotting
MFCC	Mel-frequency cepstral co-efficients
DTW	Dynamic time warping
LCSS	Longest common sub sequence
TWE	Time warp edit
EDR	Edit distance on real sequences
MSM	Move split merge
RDST	Random Dilated Shapelet Transform
TSF	Time Series Forest
CATCH22	Canonical Time Series Characteristics 22
RISE	Random Interval Spectral Ensemble
ROCKET	Random Convolutional Kernel Transform
MultiROCKET	Multi Random Convolutional Kernel Transform
MR-Hydra	Multi ROCKET HYbrid Dictionary-ROCKET Architecture
Dr-CIF	Diverse Representation Canonical Interval Forest

Abstract

In the domain of speech analysis, keyword spotting (KWS) is referred to as one of the most important challenges, where the task is to determine the presence of a particular word in the audio data. With its increasing applications in a variety of fields, various methods have been proposed for KWS in recent times, predominantly using deep learning-based techniques. However, despite their appreciable performance, the recent state-of-the-art deep learning-based KWS methods are often computationally demanding, involve a large number of parameters, and hence are not appropriate for deployment on devices with limited resources like edge devices. To address this challenge, this work proposed a novel small-footprint KWS approach that uniquely formulated KWS as a time series classification problem wherein the audio data was modeled as a univariate or multivariate time series and the target keywords formed the classes. The time series classification problem was then solved using efficient unsupervised time series feature extractors coupled with simple machine learning classifiers, thus eliminating the need for huge deep learning architectures. By exploiting the unique unsupervised temporal features in audio data, along with the acoustic features, the proposed approach provided a lightweight, state-of-the-art solution for KWS with the accuracy reaching to 98%. Performance comparison of the proposed KWS models with the existing KWS methods attested to their applicability, especially in resource-constrained environments, making them ideal for small-footprint devices.

Chapter 1

Introduction

The goal of keyword spotting (KWS), a speech analysis problem, is for the model to find particular terms in a voice stream, which can then be used to trigger the predefined downstream task, as shown in Fig. 1.1. KWS has spread itself across a variety of applications in numerous fields. For smart devices, KWS is integral to voice-activated assistants allowing users to interact with smart home devices, set reminders, or play music using simple voice commands. In driving applications, KWS enhances in-car voice assistants, enabling drivers to control navigation, make calls, or manage entertainment systems safely and hands-free. In defense, it enables the detection of sensitive information, hands-free communication, and control. Other applications involve robotics, wearable technology, gaming, interactive voice response systems, etc.

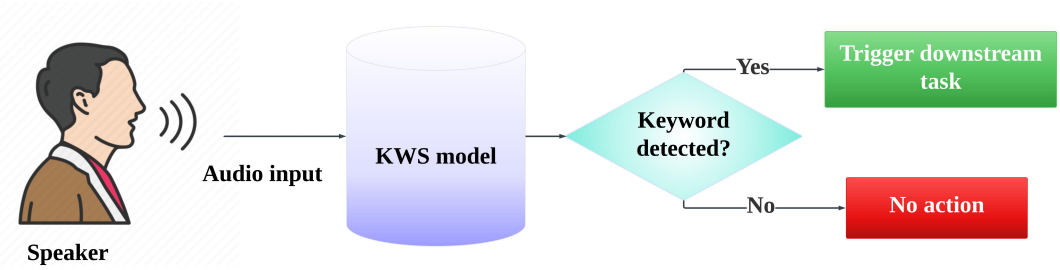


Figure 1.1: Standard KWS system

1.1 Literature review

Foundational template-based methods like Dynamic Time Warping (DTW) [1] are among the very early ASR-free approaches that finds optimal alignment between two sequences of speech. Few variants of DTW include Fast-DTW [2], Segmental DTW [3], Weighted DTW [4]. Other distance based algorithms involve Longest Common Sub Sequence (LCSS) [5], Time Warp Edit (TWE) [6], Edit Distance on Real Sequences (EDR) [7], and Move-Split-Merge (MSM) [8]. However, these distance-based approaches for keyword spotting suffer from high computational complexity, sensitivity to noise, and poor generalization across speakers and environments. They lack discriminative power, require manual threshold tuning, and are outperformed by modern deep learning methods.

With the recent increase in its applications, although a variety of methods have been proposed in the literature for keyword spotting, most of them follow one of the two basic

approaches: 1. automatic speech recognition (ASR)-based approach, and 2. automatic speech recognition (ASR)-free approach [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20]. The conventional ASR-based KWS models first translate the spoken audio into text using different ASR techniques, after which they process the text to spot the required keyword [9] [10] [11] [12]. However, such systems often employ huge advanced neural network architectures that require substantial computational resources, training on large language datasets, and cloud-based infrastructure to achieve the required performance in real-time. They need acoustic modeling, language modeling, and decoding, all of which demand high processing capabilities and significant memory. Also, as these models rely on speech recognition for keyword spotting, they are inherently language-specific, and their performance for low-resource languages is often questionable.

In contrast, the ASR-free approach tries to detect a keyword from the audio input solely using the audio characteristics, without actually recognizing the speech in the audio, thus drastically reducing the computational burden and memory requirements of the KWS models and allowing them to be language-agnostic. This makes ASR-free KWS techniques an attractive choice for lightweight, language-independent systems without heavy computational and memory demands to function on stand-alone, low-resource devices. With this advantage, in recent times, various ASR-free keyword detection methods have been developed, although most of them employed some kind of advanced deep learning models such as transformers, long short-term memory (LSTM), recurrent neural networks (RNN), convolutional neural networks (CNN), and attention mechanisms [13] [14] [15] [16] [17] [18] [19] [20], for obtaining the optimal detection performance.

CNNs that effectively learn spatial hierarchies of features, especially from the image data, have been widely adopted for identifying speech commands, wherein the audio data is often transformed into windowed-MFCC features to obtain the corresponding image representation. Rybakov et al. in [13] proposed a DNN model employing fully-connected layers with ReLU activation, a frame stacking mechanism, and a pooling layer, achieving 91.2% accuracy. Xingjian et al. in [14] gave a DenseNet structure with a trainable window function and mixup data augmentation to yield an accuracy of 92.8%. In [15], Ewald et al. presented an ASR-free feature learning model that trains a CNN using dynamic time warping (DTW) scores, yielding an AUC of 83.28% for features extracted using a correspondence auto-encoder. However, DTW has a time complexity of $O(n^2)$ for comparing two N-length sequences, thus making it computationally unsuitable for large-scale or real-time applications. In quest for better detection performance, the more recent models stacked multiple CNN blocks, making the overall architectures deeper, leading to a huge number of trainable parameters. Architectures like ResNets, having skip connections, tend to offer a solution to this and hence have been heavily used in keyword spotting models. Parnami et al. proposed the architecture of a few-shot learning-based KWS in [16], which exploited a ResNet architecture and reported 94% accuracy on the 2-way 5-shot keyword spotting task. Similarly, [17] by Huh Jaesung et al. employed metric learning upon a ResNet-based baseline architecture to achieve an accuracy of 83.82%. A. Bittar in [18] replaced the BC-ResNet encoder in the vision-inspired KWS framework with an I3D gated conformer for streaming audio, aiming to enhance keyword detection and localization with 94.8% accuracy.

Apart from the CNNs, more sophisticated deep learning techniques like RNNs, LSTMs, attention mechanisms, and transformers have also been used extensively to model temporal dependencies in audio signals [19] [20]. KWS models based on transformers are the

most recent developments in this field [19] [20]. In order to capture dependency relationships in audio data, transformer models, which are well-known for their effectiveness in natural language processing, have been modified for keyword-spotting tasks. The latest works involving transformers, like [19] by Berg et al., led to an impressive accuracy of 97.49%. Sun et al. [20] proposed the adoption of Swin transformer-based architecture for KWS with an accuracy of 87.49%, whereas the TCN+Swin transformer yielded an accuracy of 98.07%. However, the high-dimensional feed-forward networks in each layer and the self-attention mechanism's quadratic complexity make transformers resource-intensive. Their large model sizes, deep stacking, and reliance on massive datasets amplify computational and memory demands. Also, calculating attention weights in such attention-based models can be computationally intensive, especially for large models and datasets.

1.2 Objective

Therefore, while a few of these recent deep learning-based models have demonstrated state-of-the-art performance in keyword detection, they rely heavily on large architectures such as transformers, LSTMs, and attention modules. This ultimately increases the required training time, run time, memory, and computational resources, making it challenging to deploy for a real-time application in low-resource environments like edge devices with limited memory and computational power. To overcome these limitations, in this work, we proposed a novel lightweight time series classification-based KWS approach that reduces the overall memory and computational requirements for KWS. Specifically, the proposed method uniquely formulated the keyword detection task as a time-series classification problem wherein the input audio signal was modeled as univariate or multivariate time-series data, and different keywords formed the target classes. Various state-of-the-art time-series-specific feature extractors like ROCKET and RDST were then employed to extract the efficient unsupervised temporal features from the audio input, followed by simple classifiers like ridge or logistic regression for the final classification. The efficient unsupervised temporal feature extraction coupled with the simple classifiers essentially avoided the use of huge deep learning architectures without compromising the detection performance much, thus making it an ideal model to be deployed for the real-time language-agnostic KWS in resource-constrained environments.

1.3 Major contributions

The major contributions of the present work can be summarized as follows:

- Proposed a novel language-agnostic, ASR-free, time-series classification-based KWS approach wherein the input audio was modeled as a time series and the keywords corresponded to the class labels.
- Designed a unique univariate time series classification-based KWS model that employed state-of-the-art time-series feature extractors like ROCKET and RDST for the efficient unsupervised feature extraction and a simple ridge classifier, resulting in an effective lightweight model, bypassing the need for huge deep learning architectures.

- Developed a multivariate extension of the univariate time series classification-based KWS model that uniquely modeled the windowed-MFCC features from the audio as a multivariate time series to additionally exploit the inherent acoustic features from the audio data.
- Designed an integrated multivariate time series classification-based KWS model that fused the information from different time series feature extractors to yield state-of-the-art detection performance with minimal memory and processing requirements, ensuring feasibility for low-resource systems.

1.4 Organization of thesis

This thesis is divided into five chapters.

- Chapter 1, which is here, gives some introduction to keyword spotting, presents a literature review of the existing state-of-the-art approaches and lays out the premise for the study behind small footprint KWS.
- Chapter 2, provides the background and concepts involved in building the proposed KWS models.
- Chapter 3, presents the preliminary KWS model based on traditional distance based methods like DTW.
- Chapter 4, describes the design methodology and performance of a basic MFCC-CNN-based KWS model.
- Chapter 5, presents the proposed time series based KWS models, namely, univariate and multivariate time series classification-based KWS model and an integrated multivariate model.

Chapter 2

Background

The context for the methods used for aquostic feature extraction, algorithms for distance-based techniques and time series classifier types is given in this chapter. Section 2.1 describes the types of audio feature extraction techniques explored, section 2.2 provides the algorithms for the various distance based approaches utilized in chapter 3. Section 2.3 presents a brief overview of time series feature extractors and classifiers in chapter 5, laying the foundation of the proposed KWS models, with an emphasis on ROCKET and RDST, the two methods found to be optimal in the present application.

2.1 Aquostic feature extraction techniques

An essential stage in audio processing is aquostic feature extraction, which is the conversion of raw audio inputs into a collection of numerical features that can be used for additional processing. Direct computations on raw audio result in increased computational complexity because it is a continuous signal comprising many data points. Therefore, extracting the necessary features from it can aid in simplifying things and producing better performance. The standard audio feature extraction techniques are listed below:

1. Mel-Frequency Cepstral Coefficients (MFCC)
2. Spectrograms

2.1.1 Mel-Frequency Cepstral Coefficients (MFCC)

MFCC is a popular feature extraction technique in speech recognition applications. It is a chain of processes described fig. 2.1.

2.1.2 Spectrograms

It is a 2D visual representation of audio signals in the frequency domain, which shows how the frequencies in a sound change over time by dividing the audio signal into discrete segments and calculating the intensity of each segment's various frequency components. Few types of spectrograms are:

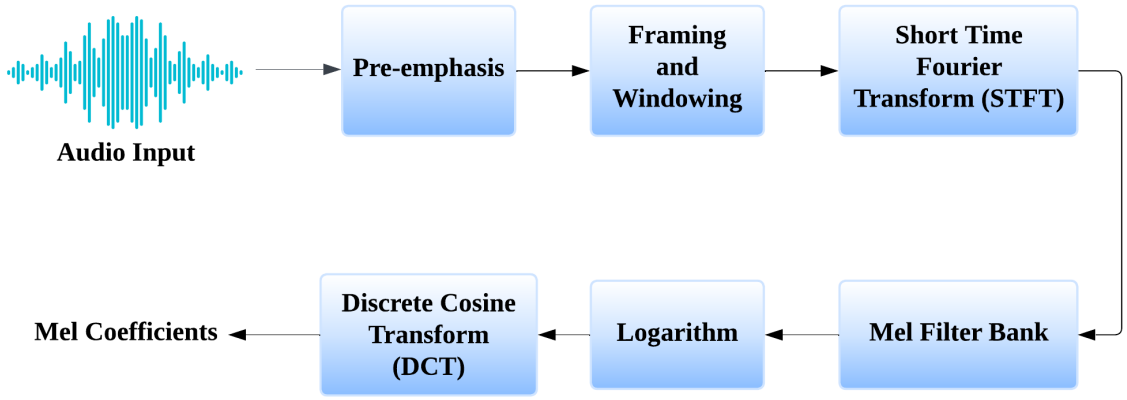


Figure 2.1: Steps involved in MFCC computation

1. Linear spectrograms : These are squared magnitude of short-time fourier transform (STFT) obtained on the audio input plotted against time.

2. Mel-spectrograms : Here the frequency scale is in Mels (non-linear). It uses a series of overlapping triangular filters, known as Mel filter bank. Conversion from Hertz to Mels is according to below formula:

$$m = 2595 \cdot \log_{10}\left(1 + \frac{f}{500}\right)$$

   where, m= Frequency in mels
   f= Frequency in hertz

3. Log-Mel spectrograms : Here the Mel frequency is converted to logarithmic scale to better mimic the human perception of sound.

2.2 Distance based approaches

The ability of distance-based techniques to assess similarity across sequences which makes them popular for use in tasks like keyword spotting. For instance, Dynamic Time Warping (DTW) is one technique that works especially well for comparing time-series data that may vary in length or show temporal fluctuations. These techniques optimize the distance or cost between related elements by aligning two sequences, enabling flexible comparisons. The algorithms for the below mentioned distance based techniques are provided in Appendix A.

2.2.1 Dynamic Time Warping (DTW)

DTW [1] is an algorithm used to measure the similarity between two temporal sequences which may vary in speed or length. It aligns the sequences in a way that minimizes the distance between them. Common distance measures like Euclidean distance calculate the distances between the corresponding values occurring at the same time instants in the two sequences. DTW, on the other hand, matches the peak with peak and trough with trough

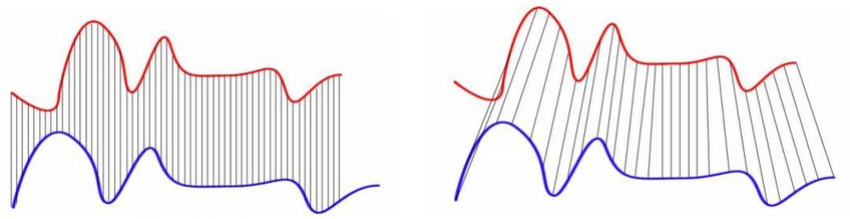


Figure 2.2: Alignment of two temporal sequences using Euclidean distance (left) vs. DTW (right)

for the sequences, hence bringing in the concept of warping and finds a more appropriate match. The steps involved in finding the DTW distance between two sequences is given below:

1. Let $X = \{x_1, x_2, \ldots, x_n\}$ and $Y = \{y_1, y_2, \ldots, y_m\}$ be the two sequences to be compared.
2. Create a cost matrix of size n×m and let d(i,j)=distance(x_i, y_j). Set value of first cell as d(1,1).
3. Fill in the cost matrix using the relation:

$$D[i][j] = d(i,j) + \min\left(D[i-1][j],\ D[i][j-1],\ D[i-1][j-1]\right)$$

4. Use backtracking to trace back from D[n][m] to D[1][1] using the path that led to the minimum cost.
5. The points corresponding to X and Y that are common to this least distance path get aligned/ warped.

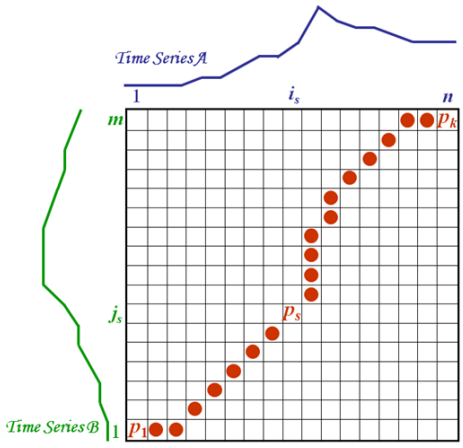


Figure 2.3: DTW cost matrix formation

A few DTW variants are examined in the following section in order to compare and choose the best one between conventional DTW and its other varieties.

2.2.2 Fast DTW

Traditional DTW is high in time complexity i.e. $O(mn)$ for unequal length sequences or $O(n^2)$ for equal length sequences. Fast DTW [2] is a faster alternative which downsamples the sequences before processing. The DTW is performed at the coarse level and then refinement is performed with iterations until original resolution is reached. The time complexity comes down to $O(n)$ for Fast DTW.

2.2.3 Segmental DTW

Keogh, Eamonn J., and Michael J. Pazzani, in [3] developed the concept of SDTW. The comparison between two sequences here is done over specific segments rather than the entire sequences. The sequences are divided into segments and segment-level cost matrix is calculated using DTW, on which segment-level optimal paths are found. Then a backtrace is performed through the cumulative cost matrices of corresponding segments to retrieve frame-level optimal alignment path for entire sequence.

2.2.4 Weighted DTW

Weighted DTW [4] computes DTW with an additional weight penalty that accounts for the difference in alignment between the sequences. The weight penalty function adjusts the contribution of each alignment in the DTW computation, penalizing or favoring alignments. This penalty function is used to obtain the distance matrix on which the DTW is performed. The penalty value is controlled by gamma parameter and the position index. We have set the gamma parameter to 0.3 and maximum weight allowed to 1.

2.2.5 Longest Common Sub Sequence (LCSS)

Longest Common Subsequence (LCSS) [5] is a technique to find the longest possible common sequence out of the input sequences. It does not give an optimal path to align two sequences as done by DTW, instead it uses few operations on the sequences to find the length of the longest common subsequence. There is also a threshold factor till which sets the limit for two elements to match. First, a matrix of zeroes with dimension equal to the (length of sequence 1, length of sequence 2) is defined. For a cell (i,j), if the corresponding elements from the sequences match, then increament the cell value at (i-1, j-1) and fill in the current cell. If no match is found, fill with the maximum of the neighboring cells. The last element of the matrix will give the maximum LCSS length required.

2.2.6 Time Warp Edit (TWE)

The TWE algorithm [6] finds the minimum edit distance between two time series sequences while considering warping costs and editing costs. We have three edit operations namely insert, delete and match. When elements are edited or deleted, an edit cost is applied. There is also a warping penalty for the matched points like weighted DTW. Based on the insert, delete and alignment costs, each cell in the matrix is filled with the minimum value of the three cost components. Once all the cells are filled, the last cell of the matrix gives the required TWE distance.

2.2.7 Edit Distance on Real Sequences (EDR)

Edit Distance [7] is very similar to LCSS in the sense that it uses a threshold to find the difference between two element values to be considered a match or not. It penalizes mismatches when the difference between elements exceeds threshold and computes the minimal edit distance using a dynamic programming approach. If the difference is less than the defined threshold, the alignment cost is set to zero, else it is set to one. Each cell is filled with the minimum value out of match cost, insert cost and delete cost. Match cost involves the diagonal cell, deletion cost involves the horizontal cell and insertion cost involves the vertical cell. The last cell gives the value of the required distance.

2.2.8 Move Split Merge (MSM)

The MSM algorithm [8] is very similar to ERP except the definition of the cost function. The alignment is adjusted based on three operations: moving, splitting and merging. The cost function is described below. Here 'c' is the minimum cost.

$$Cost(p;q;r) = \begin{cases} c & \text{if p lies between q and r} \\ c + \min(|p-q|, |p-r|) & \text{otherwise.} \end{cases}$$

2.3 Time Series Classifiers (TSC)

A continuous, sequential collection of real-valued data (a time series) is mapped to a categorical response variable using time series classification (TSC). We propose bringing these time series based extractors and classifiers into use to classify our speech data (as speech is also a continuous and sequential set of real values). One variable is recorded at each time step in univariate time series data, while multiple variables are measured at each time point in multivariate time series data. The TSC are mainly feature extractors followed by machine learning based classifiers. The types of TSC that are used in the implementation are shown in fig. 2.4.

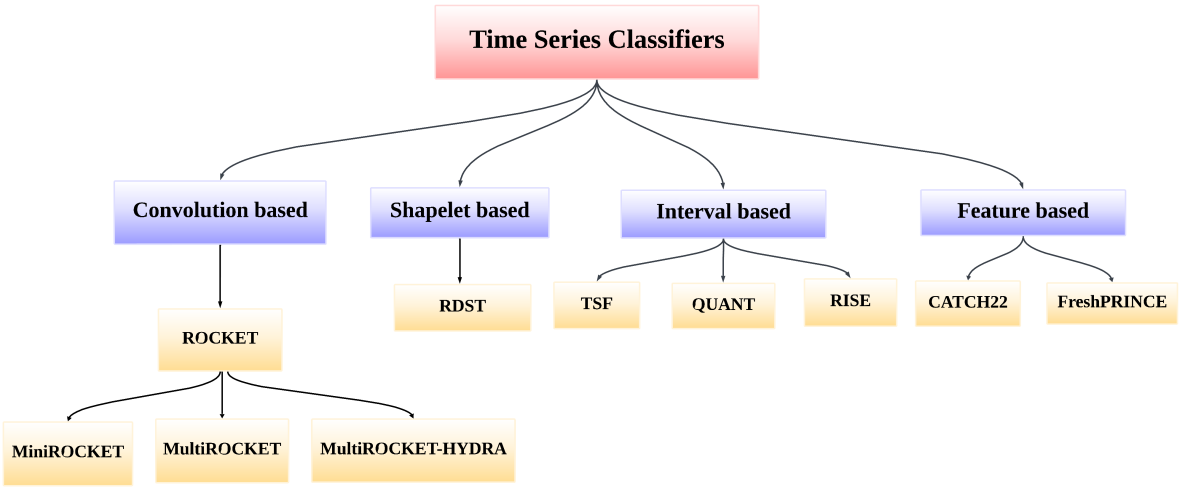


Figure 2.4: Types of Time Series Classifiers

2.3.1 Convolution based transforms

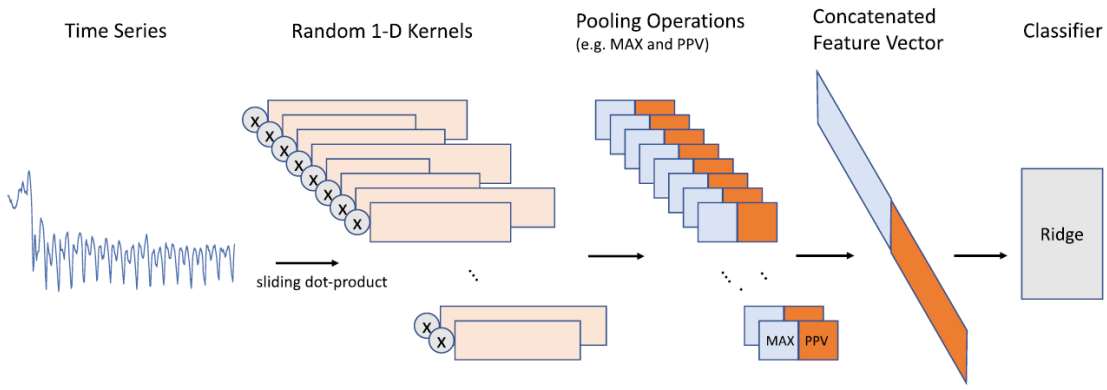


Figure 2.5: Pipeline of convolution based TSC as illustrated in [21]

Kernel or Convolution classifiers classify time series by using kernels, which are small sub-patterns designed to capture unique features within the data. Each kernel is applied to the time series through a sliding dot product, resulting in an activation map that shows where and how strongly the kernel's pattern aligns with various segments of the series. This process transforms the time series data into a new form where activation intensities highlight the presence of the kernel's features over time.The process involved in convolution based time series classifiers: generating activation maps, pooling for feature extraction, creating a feature vector and classification.

ROCKET (Random Convolutional Kernel Transform)

ROCKET [22] is a pipeline-based method that falls under the convolution-based time series classifier category. To transform the raw time series input into a more distinguishable representation, ROCKET first employs a large number of convolutional kernels that are generated by randomly varying certain kernel parameters, e.g., length, dilation, padding, random weights, and bias. Specifically, a given time series T, using a convolutional kernel W is transformed as

$$T_p * W = \sum_{k=0}^{L-1} T_{p+(k\times s)} \times W_k \tag{2.1}$$

where s denotes the dilation, L indicates the length of the time series T, and p decides the position in T. Following this, the two pooling operators, viz., maximum value and the proportion of positive values (PPV), are applied to extract features from the transformed time series. With k kernels, this essentially results in a final feature vector with $2k$ features, which are then provided to a ridge classifier for the final classification. Fig. 2.5 summarizes the entire ROCKET pipeline for a simpler visualization. Importantly, in contrast to CNN, the kernel weights here are not learned, and there is only one kernel layer. This unsupervised feature extraction in ROCKET significantly reduces the overall computational complexity.

Recently, two extensions of the basic ROCKET method have been proposed: Mini-ROCKET [23] and MultiROCKET [24]. MiniROCKET eliminates a large number of random elements of ROCKET and uses only PPV pooling. MultiROCKET, on the other hand, includes three more pooling operations in addition to the two pooling operations

of ROCKET, viz., mean of positive values (MPV), mean of indices of positive values (MIPV), and longest stretch of positive values (LSPV). All the variations of the ROCKET method were explored in the present study, and the one yielding the optimal performance was selected for each KWS model.

2.3.2 Shaplet based transform

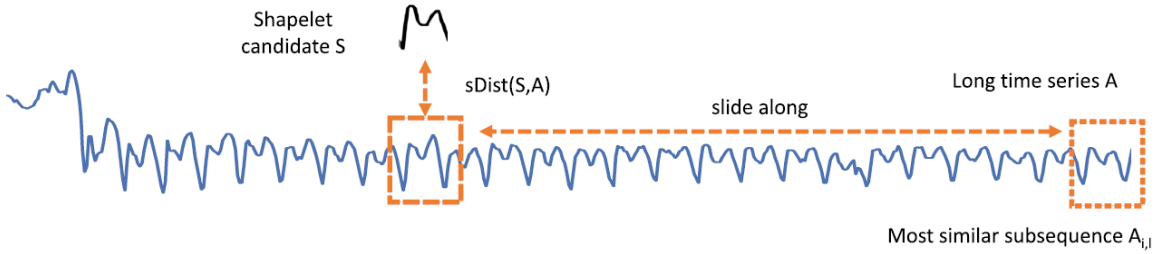


Figure 2.6: Visualisation of the shapelet distance operation sDist() between a shapelet S and a series A as described in [21]

Shapelets are short, discriminative subsequences within time series data that help distinguish between classes based on their presence or absence. Shapelet-based classification involves extracting candidate shapelets, measuring their distance to all series in the dataset, and selecting the most class-representative ones. These distances are then used to transform the original data into a new feature space, where each feature represents the distance to a selected shapelet. This transformed dataset can then be used with standard classifiers for accurate time series classification.

RDST (Random Dilated Shapelet Transform)

RDST [25] belongs to the shapelet-based category of time series classifiers. In time series data, shapelets [26] are segments that are very typical of a particular class. Shapelet-based classification works by finding these unique, discriminative patterns and using them to represent the time series data in a transformed feature space. From every time series in the dataset, all feasible subsequences of a given length are extracted to create a fixed number of potential shapelets. RDST, in particular, introduces randomness and dilation to the shapelet extraction process. Instead of exhaustively searching for the best shapelets, it randomly samples subsequences from the time series. This sampling reduces the computational load while maintaining meaningful patterns. Dilation adds spacing between points in the shapelet, allowing the classifier to capture patterns that may not be sequentially compact. So in RDST, each candidate shapelet W slides along the windows of the time series, and three parameters are recorded. These include the minimal distance discovered for that shapelet, its location, and a threshold-based indicator of the shapelet's frequency of occurrences. The minimum distance vector $D(W,T) = \{D_1, \ldots, D_{M-(L-1)\times s}\}$ for a time series $T = \{t_1, \ldots, t_M\}$ and shapelet $W = \{\{v_1, \ldots, v_L\}, s\}$, where L is the length parameter and s the dilation parameter, is computed as [25]:

$$D_i = \sqrt{\sum_{j=1}^{L} \left(T_{i+(j-1)\times s} - W_j\right)^2} \tag{2.2}$$

Fig. 2.6 depicts the process of finding the minimum distance between a shapelet and all possible windows of the time series in RDST. With the three features per shapelet, as described above, for k shapelets, the RDST produces a feature of length $3k$ which is used to train a ridge classifier. Interestingly, similar to ROCKET, the RDST also provides an efficient unsupervised method for time series feature extraction, thus making it a preferred choice for the present small footprint KWS application.

2.3.3 Interval based tranforms

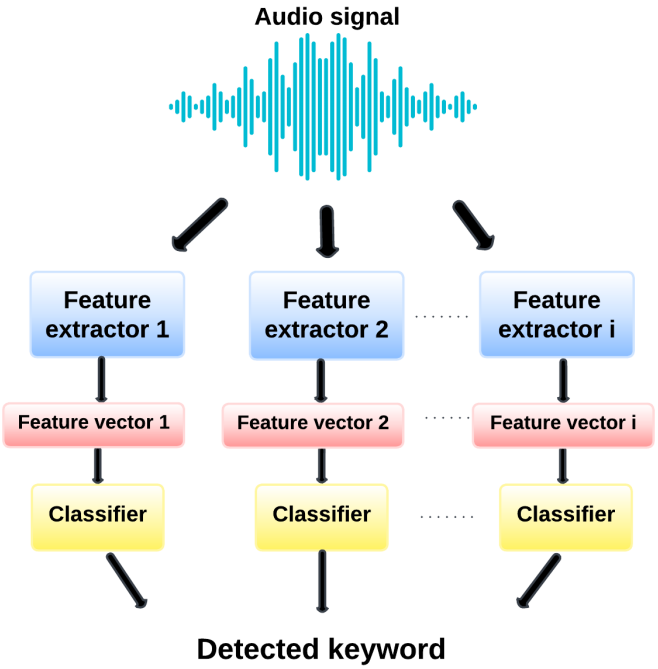


Figure 2.7: Visualisation of an ensemble of pipeline classifiers, as used in interval classifiers

In this category of classifiers, the entire time series is not utilized for feature vector creation, rather the time series is divided into intervals and then feature extraction is performed to transform each of these intervals to feature vectors to further apply classifiers. TSF, QUANT and RISE are some of the popular interval based classifier approaches.

QUANT

The QUANT [28] classifier is based on quantile-based feature extraction and multiple representations of time series data. It transforms each segment into a compact set of quantiles, efficiently summarizing the data while reducing sensitivity to noise. QUANT employs four representations—raw series, first-order differences, Fourier coefficients, and second-order differences—to capture diverse signal characteristics. A dyadic interval pyramid structure is used to segment the time series at varying granularities, allowing both broad and fine-grained pattern analysis. Quantile features are computed for each segment and concatenated into a large feature vector spanning all representations. Finally, classification is performed using an Extra Trees classifier.

Diverse Representation Canonical Interval Forest (DrCIF)

The Diverse Representation Canonical Interval Forest (DrCIF) [29] incorporates: standard raw interval-based features (mean, standard deviation, etc.), periodograms and first order differences, second order differences concatenated into a feature vector.

Time Series Forest (TSF)

It is a tree based ensemble technique. TSF [30] randomly selects multiple intervals from each time series with a random position and length. Then it extracts statistical features (such as mean, standard deviation, and slope) from each interval. TSF builds a decision tree classifier for each set of features and combines them into an ensemble. The ensemble makes the prediction using a majority vote of base classifiers. The extension of TSF is RISE.

Random Interval Spectral Ensemble (RISE)

RISE [31] is a frequency-based algorithm for TSC that leverages the spectral features of time series to build an ensemble classifier. It randomly select a single interval from each time series and extract spectral features from each interval then builds a decision tree classifier for each set of features and combines them into an ensemble. The spectral features used by RISE are: autocorrelation function, partial autocorrelation function and power spectrum. Spectral features can capture important characteristics of time series data, such as periodicity, seasonality, trend, and noise.

2.3.4 Feature based transforms

These extract descriptive statistics as features from time series to be used in classifiers. They follow series-to-vector transformations. These features are used in a simple pipeline of transformation followed by a classifier as shown in the figure 2.8.

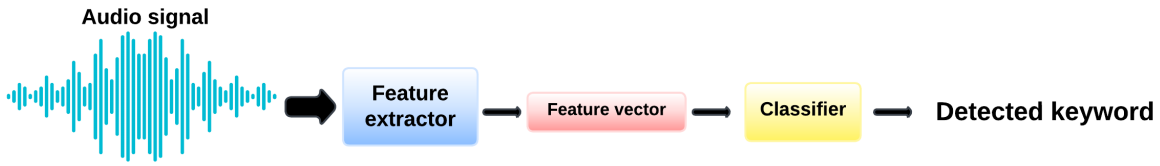


Figure 2.8: Visualization of a feature based classifier involving feature extraction followed by classification

Canonical Time Series Characteristics (Catch22):

Data-driven feature selection that captures the important aspects in a given dataset has been done using the hctsa (highly comparative time-series analysis) toolbox. Comparisons are made between thousands of different time-series features in a comprehensive collection. 22 features taken from the exploratory time series analysis toolbox of the hctsa. Entropy, linear correlations, and the fundamental statistics of time series values are only a few of the many covered by the Catch22 features [32].

FreshPRINCE

TSFresh is a set of less than 800 time series features. After comparing several feature extractor and classifier pipelines, it was discovered that using the entire set of TSFresh features without feature selection in conjunction with a Rotation Forest classifier was the most successful strategy. This pipeline was known as FreshPRINCE [33].

Chapter 3

Keyword spotting using distance based approaches

In this chapter, a simple keyword spotting model is developed based on the traditional distance matching approaches. The distance measures used for matching as namely dynamic time warping (DTW), fast dtw, weighted dtw, segmental dtw, longest common subsequence (LCSS), time warp edit (TWE), edit distance on real penalty (EDR) and move split merge(MSM), the details of which are provided in chapter 2. The exact methodology used to built the KWS model is described in section 3.1, where the distance measure used are the ones listed above. Note that, DTW is one of the most prominent distance methods that have been used in classical KWS models, where as the other techniques are not often very widely used. In this chapter, we have brought the use of these least used distance measures to check their efficacy in KWS applications and also compared their performances with traditional DTW based model.

3.1 Methodology

Pre-processing

The preprocessing involves extracting the MFCC coefficients from the raw audio files which will then be used for distance matching as detailed in next section. This input shape represents 13 coefficients and 99 time steps of the MFCC features extracted upon selecting window size as 25 ms and hop size as 10 ms. This window size and hop size combination was selected after trials on multiple such combinations.

Feature vector dimension calculation:

$$\text{MFCC feature vector dimension} \quad = \quad [\text{MFCC coefficients}, \text{Number of time frames}]$$

The number of time frames can be computed as follows. The total number of time points for our signal is $T = 16000$ samples (corresponding to 1 second of audio at a sample rate of 16 kHz). The window size is 25 ms, or 400 samples, and the hop size is 10 ms, or 160 samples (refer next section for details), the number of time frames is:

$$\text{Number of time frames} = \left(\frac{T - \text{Window size}}{\text{Hop size}} \right) + 1$$

Substituting the values:

$$\text{Number of time frames} = \left(\frac{16000 - 400}{160}\right) + 1 = 99$$

Thus, the MFCC feature vector has the following dimensions:

$$\text{MFCC feature vector for 13 coefficients} = (13, 99)$$

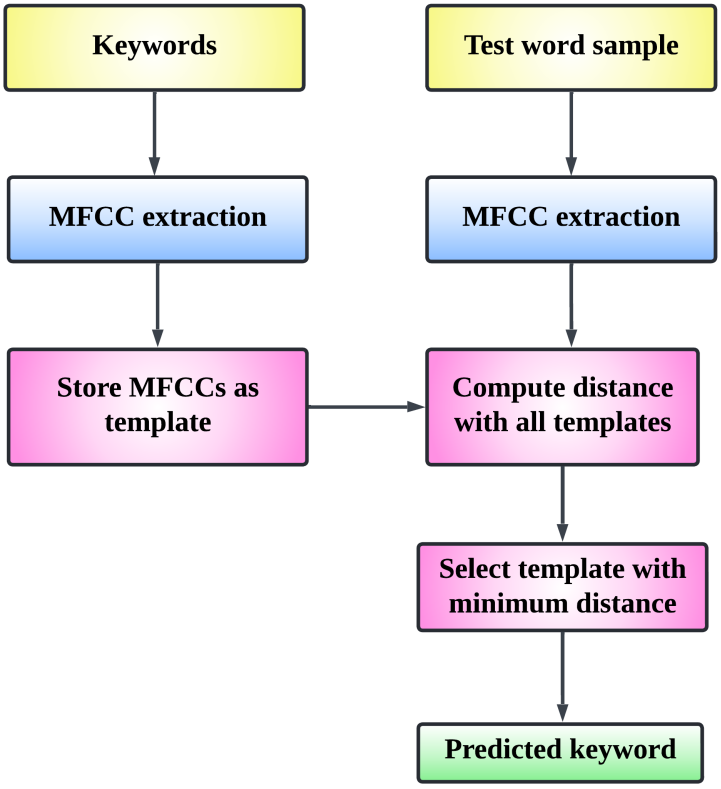


Figure 3.1: Block diagram representing the process of distance based KWS model

The block diagram presents a two-stream process for performing keyword spotting using distance-based comparison of audio features. In the first flow, a set of known keywords is used as reference data. MFCC (Mel-Frequency Cepstral Coefficient) features are extracted from each keyword sample to capture its important acoustic characteristics. These extracted MFCCs are then stored as templates, representing the unique patterns of each keyword in a compact form.

In the second flow, a test word sample is provided as input. MFCC features are similarly extracted from this test sample. These features are then compared against all the stored keyword templates by calculating a distance score that reflects how similar the test sample is to each template. The system identifies the keyword template with the minimum distance, and the corresponding keyword is returned as the predicted output. This process allows the system to match unknown speech inputs with the most acoustically similar keyword. Note that no threshold is set to decide for non-keywords. The number of correct classifications is used to obtain the detection accuracy for each DTW algorithm.

The accuracy is given by:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Test Samples}}$$

3.2 Dataset

The study presented herein utilized the widely used dataset: Google Speech Command (GSC) dataset V1 [34]. The dataset consist of 1-second audio snippets sampled at 16kHz. The V1 contains 65,000 samples of 30 different words. All the classification models in this study utilized ten command words: “up”, “down”, “left”, “right”, “yes”, “no”, “on”, “off”, “go”, and “stop” as the target keywords. The train-test split ratio was chosen to be 80:20.

3.3 Results

Out of the variants of DTW, traditional DTW comparatively performes better with an accuracy of 72% followed by Fast DTW with an accuracy of 64%, Weighted DTW with 62% and Segmental DTW also with 62%. The same is shown in table 3.1. Out of the elastic distance measures, time warped edit distance (TWE) showed comparatively higher performance with 65% accuracy, followed by MSM and LCSS with accuracies 59% and 58%. EDR has least accuracy of 57%.

Algorithm	**Accuracy**
DTW	72%
Fast DTW	64%
Segmental DTW	62%
Weighted DTW	62%
LCSS	58%
TWE	65%
EDR	57%
MSM	59%

Table 3.1: Accuracies obtained for each distance measure

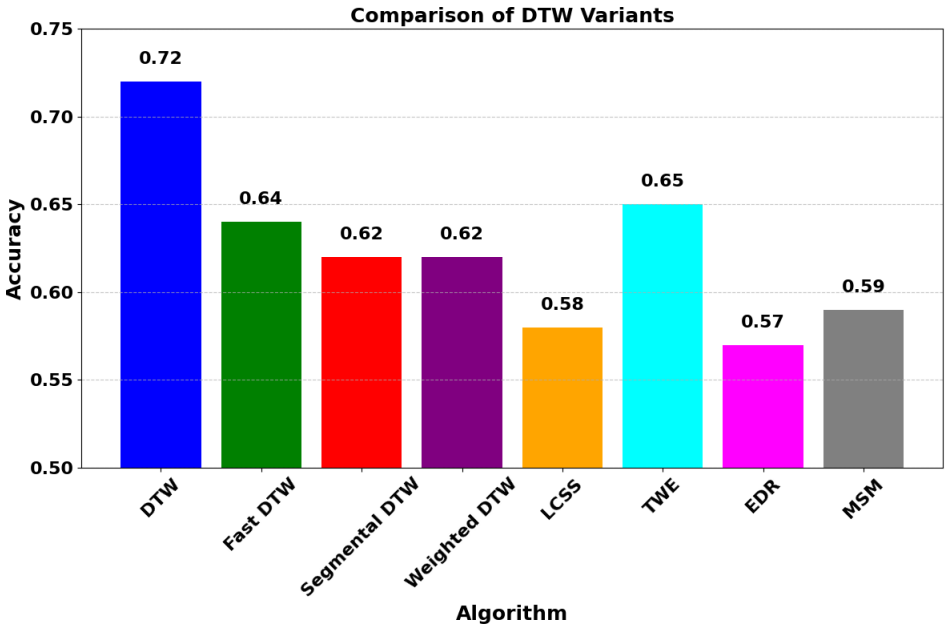


Figure 3.2: Bar graph showing the accuracies obtained for each distance algorithm

3.4 Conclusion

In this chapter, we have explored different distance based algorithms to be used as part of building a KWS models. We have tested the model with majorly used algorithm like DTW and also with measures like TWE, EDR, MSM that are not usually explored in KWS applications. The resulting model has served with a moderate performance of 72% but there lies a scope of major improvement in the accuracy. To utilise the advantages of recent developments in deep learning approaches like CNN based KWS models, we have explored in the next chapter one such basic model which uses MFCC again as the aqoustic feature extractor and then employs a simple CNN architecture to build a keyword classifier. This basic model will serve as a benchmark technique to have a comparison between non-deep learning vs. deep learning technique in building a small footprint KWS system.

Chapter 4

MFCC-CNN-based basic KWS model

This chapter introduces a simple MFCC (Mel Frequency Cepstral Coefficients) and CNN (Convolution Neural Network) based keyword spotting model which utilizes the CNN for classifying the MFCC features extracted from the audio input. The methodology follows a comprehensive data preparation, feature extraction, model design, training, and evaluation pipeline. The raw audio files are processed to extract MFCCs following similar approach as shown in previous chapters. Then, we apply these features to a CNN classifier.

4.1 Dataset

The entire study presented herein utilized the widely used dataset: Google Speech Command (GSC) dataset V1 [34]. The dataset consist of 1-second audio snippets sampled at 16kHz. The V1 contains 65,000 samples of 30 different words. All the classification models in this study utilized ten command words: “up”, “down”, “left”, “right”, “yes”, “no”, “on”, “off”, “go”, and “stop” as the target keywords, in addition to two classes, viz., “silence”, and “unknown” where the prior corresponds to the non-speech input while the latter comprised of the non-target keywords. The train-test split ratio was chosen to be 80:20.

4.2 Methodology

Pre-processing

The preprocessing involves extracting the MFCC coefficients from the raw audio files which are then applied to a simple CNN classifier, the architecture of which is provided in next subsections. This input shape represents 13 coefficients and 99 time steps of the MFCC features extracted upon selecting window size as 25 ms and hop size as 10 ms. This window size and hop size combination was selected after trials on multiple such combinations.

Feature vector dimension calculation:

$$\text{MFCC feature vector dimension} \quad = \quad [\text{MFCC coefficients}, \text{Number of time frames}]$$

The number of time frames can be computed as follows. The total number of time points for our signal is $T = 16000$ samples (corresponding to 1 second of audio at a sample rate of 16 kHz). The window size is 25 ms, or 400 samples, and the hop size is 10 ms, or 160 samples (refer next section for details), the number of time frames is:

$$\text{Number of time frames} = \left(\frac{T - \text{Window size}}{\text{Hop size}}\right) + 1$$

Substituting the values:

$$\text{Number of time frames} = \left(\frac{16000 - 400}{160}\right) + 1 = 99$$

Thus, the MFCC feature vector has the following dimensions:

$$\text{MFCC feature vector for 13 coefficients} = (13, 99)$$

CNN Model Architecture

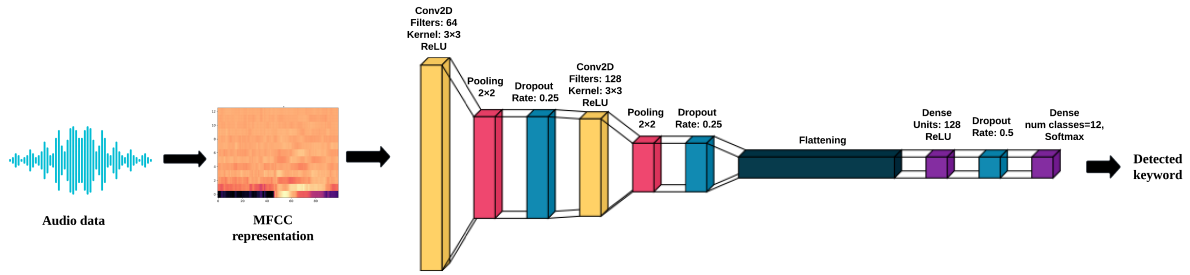


Figure 4.1: Block diagram for basic MFCC-CNN-based KWS model.

Learning distinctive characteristics from the MFCC representations is the goal of the classifier in this basic model. As CNN is well-suited for handling the 2D nature of the resulting MFCC matrix, the same has been employed here as a classifier for distinguishing between different audio classes. Following two convolutional layers, the CNN model employed dropout for regularization (with dropout rates of 0.25 following the convolutional layers) and max-pooling with a kernel size of 2. After every convolutional layer in the forward pass, a ReLU activation function was applied to add non-linearity. The convolutional layers in the first and second layers employed 64 and 128 filters, respectively, with the same kernel size of 3×3. Following the convolutional blocks, the feature maps underwent two fully connected layers after flattening. To prevent overfitting, a ReLU activation and a dropout layer (with a rate of 0.5) were applied after the first fully connected layer, having 128 neurons. The final layer then outputted class probabilities that match with the number of target classes. Adam optimizer was used to train the above model for 150 epochs with a learning rate of 0.001, batch size of 32, and cross-entropy loss as the objective function. The aforementioned optimal configuration of the CNN

architecture, tabulated in table 4.1, was obtained through comprehensive experimentation, as presented in the results section. Fig. 4.1 illustrates the pipeline of the basic MFCC-CNN-based keyword spotting model.

Layer	**Configuration**	**Output Shape**
Input	-	(1, 13, 99)
Conv2D	Filters: 64, Kernel: 3×3, ReLU	(64, 11, 97)
MaxPool2D	Pool size: 2×2	(64, 5, 48)
Dropout	Rate: 0.25	(64, 5, 48)
Conv2D	Filters: 128, Kernel: 3×3, ReLU	(128, 3, 46)
MaxPool2D	Pool size: 2×2	(128, 1, 23)
Dropout	Rate: 0.25	(128, 1, 23)
Flatten	-	(2944)
Dense	Units: 128, ReLU	(128)
Dropout	Rate: 0.5	(128)
Dense	Units: num_classes=12, Softmax	(num_classes=12)

Table 4.1: Architecture details of the basic MFCC-CNN-based KWS model.

4.3 Results

As the proposed basic MFCC-CNN-based KWS model employs a CNN classifier to classify the MFCC feature matrix, optimizing the CNN architecture is crucial for obtaining the best detection performance. So, we tried to optimize the CNN architecture by varying the number of filters and training epochs. Table 4.2 summarizes the classification performance of the basic model for various filter combinations in CNN. A CNN architecture with 64 and 128 filters performed the best and hence was the choice in the final baseline KWS model. Similarly, table 4.3 shows the detection performance of the MFCC-CNN model, obtained by training the CNN model using a varying number of epochs, fig. 4.3 shows the confusion matrix obtained for the model for CNN architecture (64,128) for 150 epochs while fig. 4.3 shows the corresponding training loss. It is evident that although the training loss decreases with an increase in epochs, training the CNN for above 150 epochs results in overfitting; hence, the number of epochs was set to 150, leading to the highest accuracy of 91.695%.

CNN Architecture	Accuracy (%)
(32,64)	90.594
(64,128)	**91.695**
(128,256)	90.576

Table 4.2: Classification performance comparison of different CNN architectures.

Number of Epochs	Performance Metrics on GSC V1			
	Accuracy	***Precision***	***Recall***	***F1-Score***
50	85.925	85.971	85.923	85.904
100	91.316	91.335	91.316	91.269
150	**91.695**	**91.807**	**91.695**	**91.692**
200	91.442	91.568	91.442	91.437

Table 4.3: Classification performance of CNN architecture (64,128) over varying number of epochs.

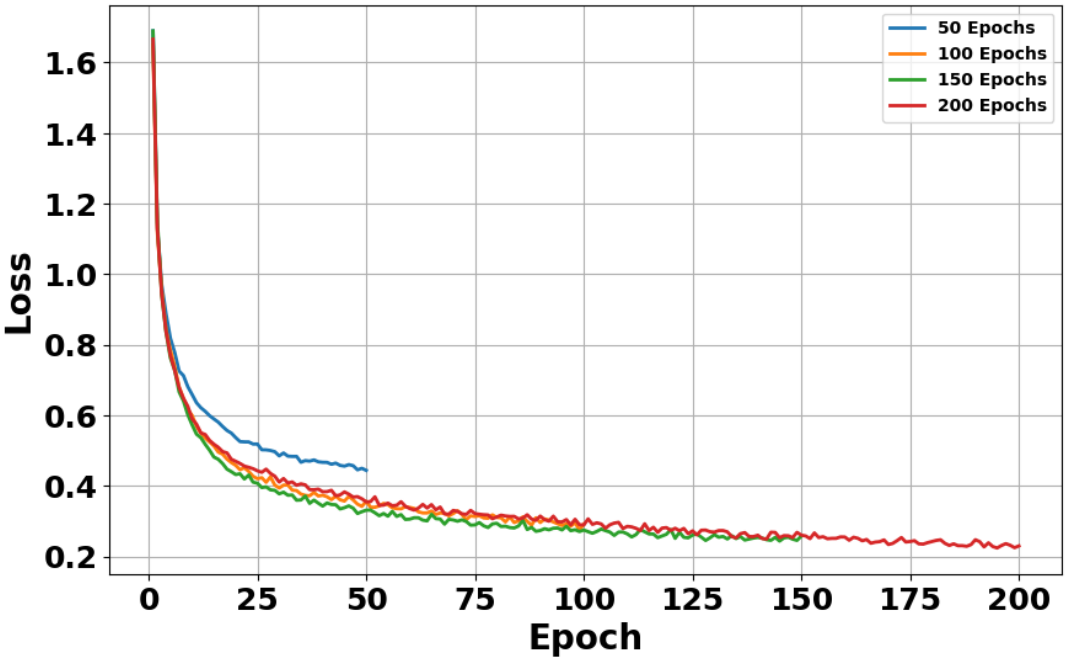


Figure 4.2: Training loss for CNN baseline model over different number of epochs

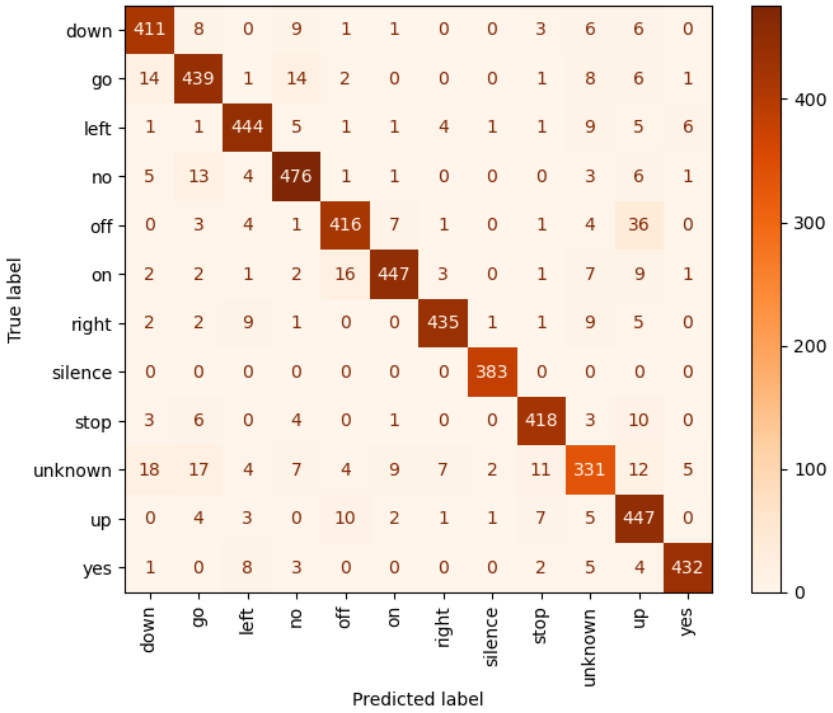


Figure 4.3: Confusion matrix for MFCC-CNN based baseline KWS model for CNN architecture (64,128) for 150 epochs

4.4 Challenges

As detailed in the next chapter, although this basic model performed reasonably well with a classification accuracy of 91.69%, it considers the MFCC matrix as an image data, neglecting the temporal evolution of the MFCC coefficients, thus losing the important temporal characteristics from the inherently time-evolving audio data. To overcome this limitation, in the next chapter, we proposed a time series classification-based keyword spotting model that exploits the intrinsic time series nature of the audio data.

Chapter 5

Time series classification based KWS

As seen in the previous chapter, the basic keyword spotting (KWS) model achieved a respectable classification accuracy of 91.69%, indicating its potential for real-world deployment in speech-enabled applications. Despite this promising performance, the model is based on a representation that treats the Mel-Frequency Cepstral Coefficients (MFCC) as a static two-dimensional image. This perspective inherently disregards the temporal progression of the MFCC features, which is a critical aspect of speech signals. Speech is fundamentally a time-dependent phenomenon where the ordering and evolution of features over time carry essential information. By ignoring these temporal patterns, the model sacrifices the ability to recognize subtle yet important dynamics in how keywords are articulated, especially in cases where different words may share similar spectral characteristics but differ in their temporal structure.

To overcome this limitation, this chapter introduces a time series classification-based KWS model that is explicitly designed to exploit the sequential nature of audio data. Rather than flattening or treating the data as an image, this approach processes the MFCC feature matrix as a sequence of vectors evolving over time. The coefficients of the MFCC matrix are considered as multichannel inputs evolving over the time frames. By doing so, it captures the temporal dependencies within the audio signal. This shift not only aligns better with the intrinsic properties of speech but also paves the way for improved recognition accuracy and robustness. Ultimately, the proposed time series-based model offers a more principled and effective solution for the keyword spotting task. In this regard, three types of KWS models are proposed: the univariate time series classification (UTSC)-based KWS model, the multivariate time series classification (MTSC)-based KWS model and an integrated MTSC based KWS model as described in detail in the upcoming sections. The background for the time series classifiers used is provided in chapter 2.

5.1 Dataset

The entire study presented herein utilized two widely used publicly available datasets: the Google speech command (GSC) datasets V1 and V2 [34]. Both datasets consist of 1-second audio snippets sampled at 16 kHz. The V2 contains 105,000 samples of 35 distinct words, whereas the V1 contains 65,000 samples of 30 different words. All the classification models in this study utilized ten command words: “up”, “down”, “left”, “right”, “yes”, “no”, “on”, “off”, “go”, and “stop” as the target keywords, in addition to two classes, viz., “silence”, and “unknown” where the prior corresponds to the non-speech

input while the latter comprised of the non-target keywords. The train-test split ratio was chosen to be 80:20. In addition to the two versions of GSC dataset, another dataset called the Free Spoken Digit Dataset (FSDD) [35] has been used to validate the proposed models. The Free Spoken Digit Dataset (FSDD) consists of approximately 3,000 audio samples of spoken digits from 0 to 9, making up 10 distinct classes. These recordings are contributed by multiple speakers, typically 6, each pronouncing digits multiple times to ensure variability. Each audio clip is around 1 second long, saved in WAV format, and recorded at a sampling rate of 8,000 Hz. Additionally, "unknown" and "silence" classes have been added similar to that of the GSC dataset.

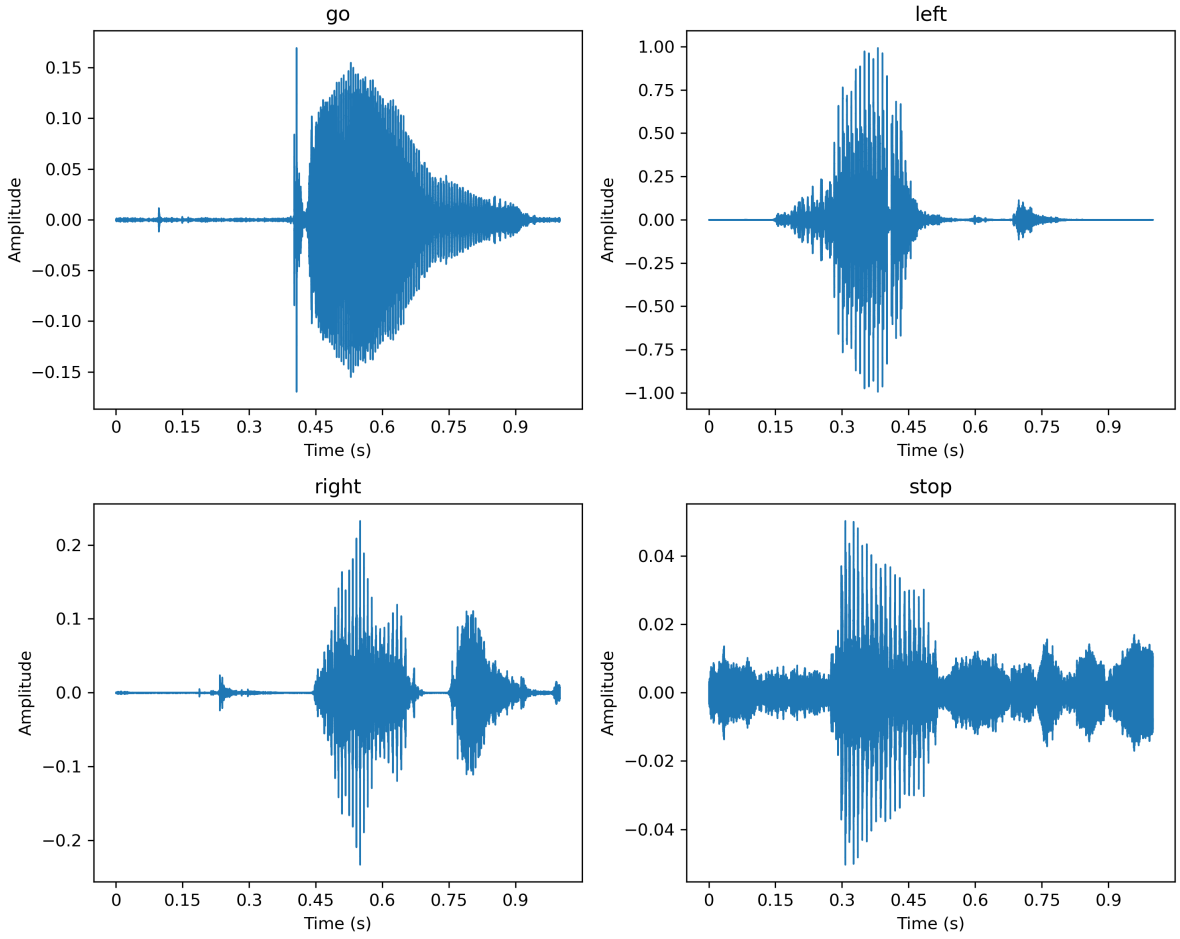


Figure 5.1: Representation of a few sample audios from the GSC V1 dataset

5.2 Univariate time series classification (UTSC)-based KWS model

As we propose the use of time series classifiers (TSC) for KWS, we first utilize the univariate form of TSC to validate the idea and then move on to the actual proposition using multivariate TSC. Although the existing works on time series classification deal with different applications involving data from sensors, motion, video, and biomedical devices, audio or speech data has hardly been explored, especially for keyword spotting applications [21]. So, here, for the first time, we bring the time series classifiers into use

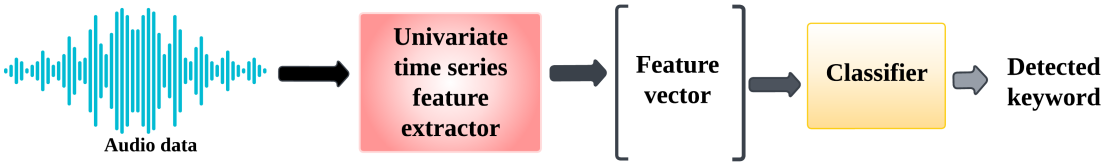


Figure 5.2: Block diagram of the proposed UTSC-based KWS model.

for keyword spotting, since the audio signal in raw form, which records the temporal variation in signal amplitude value, can efficiently be modeled as univariate time series data. Specifically, we formulated the problem of keyword spotting as a univariate time series classification task, wherein the input audio served as a univariate time series to be classified while the different keywords formed the class labels. To validate our hypothesis, we employed different time series classifiers described in chapter 2, on the GSC dataset and analyzed their performance as described in upcoming sections.

To make the audio data suitable to be applied as an input to the time series classifiers, the data was first reshaped in the form $(n_{\text{cases}}, n_{\text{channels}}, n_{\text{timepoints}})$, where n_{cases} = total number of speech samples in the dataset, $n_{\text{channel}} = 1$ for univariate signals, $n_{\text{timepoints}}$ = sampling rate in Hz×length of audio in seconds. Thus, we reshaped our audio dataset to $(27692, 1, 16000)$, where the total number of audio samples = 27692 and the sampling rate of each audio of length 1 second was 16000 Hz. This was further broken down to the 8:2 train-test split. Next, the different univariate time series feature extractors, described earlier in chapter 2, were provided with this matrix to generate the feature representations that were used further to train different classifiers for keyword detection as depicted in Fig. 5.2. Finally, performance metrics were evaluated for classification on the test data with leave-one-out cross-validation. Multiple combinations of time series feature extractors and classifiers were explored to obtain the best-performing model. As detailed in the results section, the ROCKET and the RDST features, coupled with the ridge classifier, performed optimally, with the classification accuracy of 77.88% and 77.67% , respectively.

So, although these results corroborated our idea of employing the time series classifiers for keyword spotting, the proposed UTSC-based KWS model works directly on the raw audio time series, and hence, unlike the MFCC-CNN-based model, can not explicitly exploit the potentially useful acoustic features like MFCC. Hence, to combine the strengths of both the MFCC-CNN-based approach and the UTSC-based approach, we next propose a multivariate extension of the univariate approach that uniquely models the windowed-MFCC features from the audio as a multivariate time series to additionally exploit its acoustic features.

5.3 Multivariate time series classification (MTSC)-based KWS model

As stated earlier, the MTSC-based keyword spotting model aims to combine the strengths of acoustic features like MFCC with time series-specific feature extractors. To achieve this, similar to the basic MFCC-CNN-based model, firstly the MFCC features from the overlapping windows of the audio were extracted, which transformed the univariate audio

Figure 5.3: Block diagram of the proposed MTSC-based KWS model.

time series data into a 2-D MFCC matrix of size $M \times N$, where M denotes the number of MFCC coefficients while N corresponds to the number of time frames. Now, this $M \times N$ matrix, consisting of the MFCC features extracted from the consecutive time frames, essentially represents the temporal evolution of the M MFCC coefficients and hence can be treated as an M-dimensional multivariate time series. This, in turn, translates the keyword spotting problem into the multivariate time series classification problem where the input to a classifier is an M-dimensional time series of length N, and the output class labels correspond to the required keywords. To determine the optimal multivariate time series representation of the audio, apart from the different combinations of window size and hop size, different acoustic features, like spectrograms, were also explored. As detailed in the later section, the 13-length MFCC features extracted from a window size of 25 ms and a hop size of 10 ms, which resulted in a 13×99 matrix, yielded the optimal results and hence was used in the further analysis presented below.

Similar to the UTSC-based model, the application of MTSC also requires an input of the form $(n_{\text{cases}}, n_{\text{channels}}, n_{\text{timepoints}})$, where n_{cases} = total number of speech samples in the dataset, n_{channels} = number of multivariate channels, $n_{\text{timepoints}}$ = number of total time frames of audio considered to calculate MFCC. So in the present application, n_{cases} = 27692 (representing the total number of audio samples in our dataset), n_{channels} = 13 (representing 13 MFCC coefficients), and $n_{\text{timepoints}}$ = 99 (representing a total number of time frames upon which MFCC is computed), thus resulting in the input matrix of size $(27692, 13, 99)$.

Once this matrix was formed, the same was applied to the different multivariate time series feature extractors to get the feature space representations. These feature representations were then employed as the inputs to different classifiers for the final keyword detection. Through the comparative analysis presented in the results section, it was observed that, similar to the UTSC-based model, the ROCKET and RDST features, applied to the ridge classifier, performed the best. The same were then explored further for enhancing the model performance as detailed below.

As discussed in chapter 2, ROCKET produces $2k$ number of features for k number of kernels, and RDST produces $3k$ number of features for k number of shapelets. So, deciding the optimal number of kernels and shapelets is extremely important as it affects the time series feature extraction and hence ultimately the detection performance. To select the optimal number of kernels and shapelets, experimentation was performed by varying their default values (of 10K kernels for ROCKET and 10K shapelets for RDST, respectively), as detailed in ablation studies of section 5.5. It was found that ROCKET with increased 120K kernels, and RDST with increased 50K shapelets performed the best with the accuracies of 95.02% and 95.14%, respectively. Now, although employing the increased number of kernels and shapelets improved the classification accuracy, it also increased the resulting number of features to $2 \times 120\text{K} = 240\text{K}$ for ROCKET and

3×50K = 150K for RDST. These feature vectors, when inputted to the 12-class classifier, essentially resulted in an increase in total trainable parameters for the KWS model, as detailed below:

- For ROCKET:
  Total trainable parameters $= 12 \times 240K + 12 = 2880012 = 2.88M$.
- For RDST:
  Total trainable parameters $= 12 \times 150K + 12 = 1800012 = 1.80M$.

Now this huge number of trainable parameters also increases the model's complexity and may restrict its applicability to resource-constrained systems like edge devices. So, to circumvent this, the number of model parameters was brought down by reducing the feature vector length using dimensionality reduction. Specifically, as shown in Fig. 5.3, principal component analysis (PCA) [36], which retains only the essential dimensions of the input, was applied to the original feature vector before feeding it to the classifier so that the number of trainable parameters is reduced. As discussed in detail in the next section, the drastic reduction in the number of parameters using PCA, marginally affected the overall classification accuracy, thus making the model deployable in low-resource environments without compromising the accuracy significantly. Both the proposed ROCKET and the RDST-based KWS models, even with the reduced feature vectors, performed really well with an accuracy of 94.82% and 93.36%, respectively. To improve the detection performance further, we finally combined them to design the proposed integrated MTSC-based KWS model, which is presented next.

5.4 Integrated MTSC-based KWS model

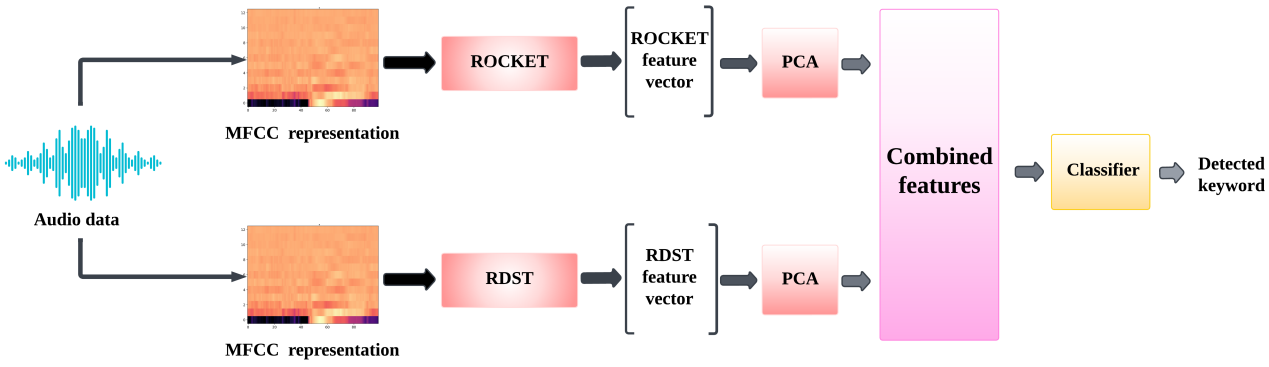


Figure 5.4: Block diagram of the proposed integrated MTSC-based KWS model.

Impressed by the individual keyword detection capabilities of the proposed ROCKET and the RDST-based MTSC KWS models, here we propose an integrated MTSC-based KWS model that combines their strengths by employing an intermediate-level feature fusion approach [37]. Specifically, as depicted in Fig. 5.4, the final integrated MTSC-based KWS model concatenated the PCA-reduced ROCKET-based 20K features with the PCA-reduced RDST-based 5K features to get a combined feature space representation for a given audio input. This concatenated feature of length 25K, which essentially fused

the ROCKET's kernel-based information with the RDST's shapelet-based information, was then used to train different classifiers used in the earlier analysis. As expected and also verified by the experimental analysis presented in the next section, this fusion of complementary information extracted using different feature extractors further enhanced the detection performance for the proposed integrated MTSC-based KWS model. With a classification accuracy of ∼95% using only 300K trainable parameters, the proposed lightweight integrated MTSC-based KWS model thus establishes itself as a strong contender for deployment to edge devices for keyword spotting tasks. This is further justified through detailed performance evaluation, ablation study, and comparative analysis of different KWS models presented in the following section.

5.5 Experimental Analysis and Results

This section provides a thorough analysis of each suggested KWS model's keyword detection performance, which is followed by a comprehensive ablation study. The performance of the suggested KWS models is also contrasted with the current state-of-the-art KWS techniques to appreciate the contribution further.

To quantify the effectiveness of various KWS approaches, the entire performance analysis presented herein employed four widely used classification performance metrics, viz.,

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

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

where TP, TN, FP, and FN indicate true positive, true negative, false positive, and false negative, respectively.
The remainder of this section quantifies and analyzes the performance of all the KWS methods using the above four performance metrics.

5.5.1 UTSC-based KWS Model

As described in detail in section 5.2, the proposed UTSC-based KWS model performs the feature extraction using various time series feature extractors like ROCKET, and RDST. As the overall classification performance of the model depends on the efficacy of the features, various time series feature extractors were employed to obtain the temporal features from the audio that were then fed as input to the classifier for keyword detection. Table 5.1 summarizes the performance metrics of the proposed UTSC-based KWS model using various univariate time series feature extractors. Among all, RDST and ROCKET yielded the most prominent results on univariate audio data with an accuracy of 77.67%

and 77.88%, respectively. To improve the classification performance further, these feature sets were subjected to different standard classifiers like ridge, support vector classifier (SVC), logistic regression, and random forest, the results of which are summarized in ablation studies. However, as can be observed therein, the model with the default ridge classifier performed the best.

UTSC	**Performance Metrics on GSC V1**			
	Accuracy	***Precision***	***Recall***	***F1-Score***
ROCKET [22]	**77.8844**	**77.6135**	**77.9844**	**77.1902**
RDST [25]	**77.6674**	**78.0127**	**77.7479**	**76.9627**
Hydra [27]	71.4693	71.2655	71.2396	71.2525
MR-Hydra [27]	72.7456	72.7156	72.3649	72.5398
QUANT [28]	66.2574	67.2638	66.7151	65.8078
DrCIF [29]	64.8974	64.2989	64.3987	64.3487
TSF [30]	44.1776	44.2236	45.1140	43.5484
RISE [31]	59.4331	60.9941	59.9488	58.3955
CATCH22 [32]	48.1494	48.1400	48.7251	47.6679
FreshPRINCE [33]	68.1346	68.1726	68.2544	68.2134

Table 5.1: Classification performance of UTSC-based KWS model using different time series feature extractors.

5.5.2 MTSC-based KWS Models

MTSC	**Performance Metrics on GSC V1**			
	Accuracy	***Precision***	***Recall***	***F1-Score***
ROCKET [22]	**89.7996**	**89.8721**	**89.9091**	**89.8648**
RDST [25]	**93.6270**	**93.7773**	**93.6418**	**93.6600**
Hydra [27]	87.1998	87.3470	87.2800	87.2342
MR-Hydra [27]	87.3803	87.5443	87.4619	87.4489
QUANT [28]	77.1258	78.6164	77.2386	76.6211
DrCIF [29]	81.0000	81.1707	81.1162	80.9578
TSF [30]	76.1328	77.1186	76.4026	75.8296
RISE [31]	28.0014	28.3751	29.0613	28.3689
CATCH22 [32]	72.5762	73.4686	72.7820	71.8176
FreshPRINCE [33]	76.3352	76.4210	76.5101	76.2306

Table 5.2: Classification performance of MTSC-based KWS model using different time series feature extractors.

Similar to the UTSC-based KWS model, the efficacy of the proposed MTSC-based KWS model is also dictated by the multivariate time series features extracted from the MFCC representation of the audio data. So, to obtain the optimal classification performance, again, various time series feature extractors were applied to the MFCC matrix, the results of which are tabulated in table 5.2. Similar to the univariate case, the best classification accuracy was obtained by the RDST and ROCKET in the multivariate case as well. Inspired by this, the RDST and ROCKET-based models were tuned further by changing the number of shapelets and kernels, as detailed in the ablation studies. As can be observed

therein, the highest accuracy results were achieved when the number of shapelets was increased 50K for RDST, and the number of kernels was increased to 120K for ROCKET. However, with this increased number of shapelets and kernels, the number of trainable parameters of the model also increased drastically. As calculated earlier in section 5.3, the 120K kernels in ROCKET result in 2.88M trainable parameters, whereas the 50K kernels in RDST result in 1.80M trainable parameters, thus restricting the model's deployability in low-resource environments.

To circumvent this, as discussed earlier, the dimensionality reduction method, specifically the PCA, was introduced before the actual classification module. To determine the optimal dimensions of the features, different numbers of components were retained using PCA, and the resulting classification performance was assessed as tabulated in table 5.3. As clearly seen from the table, ROCKET-based model almost retains its best performance with a slightly reduced accuracy of 94.82% when the feature size is reduced from 240K to 20K, to drastically bring down the trainable model parameters from 2.88M to 240K. Similarly, the RDST-based model also maintains its performance with a marginally lower accuracy of 93.36% when the feature size is reduced from 150K to 5K, again significantly bringing down the trainable model parameters from 1.8M to 60K. This huge reduction in the number of trainable parameters essentially facilitates the model's applicability in resource-constrained environments. Apart from fine-tuning the feature extraction part, we also tried different standard classifiers for classifying the above extracted time series features. The comparison results presented in the ablation studies, indicated that the ridge classifier turned out to be the best for both the RDST and the ROCKET-based features and hence were used in the final MTSC-based KWS models.

MTSC	**PCA Components**	**Trainable Parameters**	**Performance Metrics on GSC V1**			
			Accuracy	***Precision***	***Recall***	***F1-Score***
ROCKET	20K	$20K \times 12 + 12 = 240012 = 240K$	**94.8185**	**94.8519**	**94.8635**	**94.8213**
	10K	$10K \times 12 + 12 = 120012 = 120K$	94.7463	94.8098	94.7726	94.7707
	5K	$5K \times 12 + 12 = 60012 = 60K$	94.6199	94.6877	94.6555	94.6206
RDST	20K	$20K \times 12 + 12 = 240012 = 240K$	93.1034	93.3064	93.1309	93.1705
	10K	$10K \times 12 + 12 = 120012 = 120K$	93.3200	93.5130	93.3310	93.3603
	5K	$5K \times 12 + 12 = 60012 = 60K$	**93.3562**	**93.5421**	**93.3516**	**93.3602**

Table 5.3: Classification performance of MTSC-based KWS models with different PCA components.

Finally, table 5.4 shows the results of integrating both the MTSC-based models by combining the ROCKET and RDST-based features. The integrated KWS model, with a combined optimal reduced set of features obtained from the ROCKET and RDST, performed the best, surpassing their individual performance. Fig. 5.5 provides the resulting confusion matrix for a better visualization. This essentially validated our idea of fusing different time series features to benefit from their complementary information. To assess the impact of the classifier on the integrated MTSC-based KWS model, similar to earlier analysis, the concatenated features were applied to various standard classifiers, the results of which are summarized in the ablation studies. Similar to all the earlier time series-based KWS models, here also, the simple ridge classifier was found to be optimal.

MTSC	PCA Components	Combined Trainable Parameters	Performance Metrics on GSC V1			
			Accuracy	*Precision*	*Recall*	*F1-Score*
ROCKET RDST	20K 5K	$(20K + 5K) \times 12 + 12 = 25K \times 12 + 12 = 300012 = 300K$	**94.9449**	**94.9643**	**94.9935**	**94.9558**
ROCKET RDST	5K 5K	$(5K + 5K) \times 12 + 12 = 10K \times 12 + 12 = 120012 = 120K$	94.4755	94.5719	94.5008	94.5068

Table 5.4: Classification performance integrated MTSC-based KWS model with different PCA components.

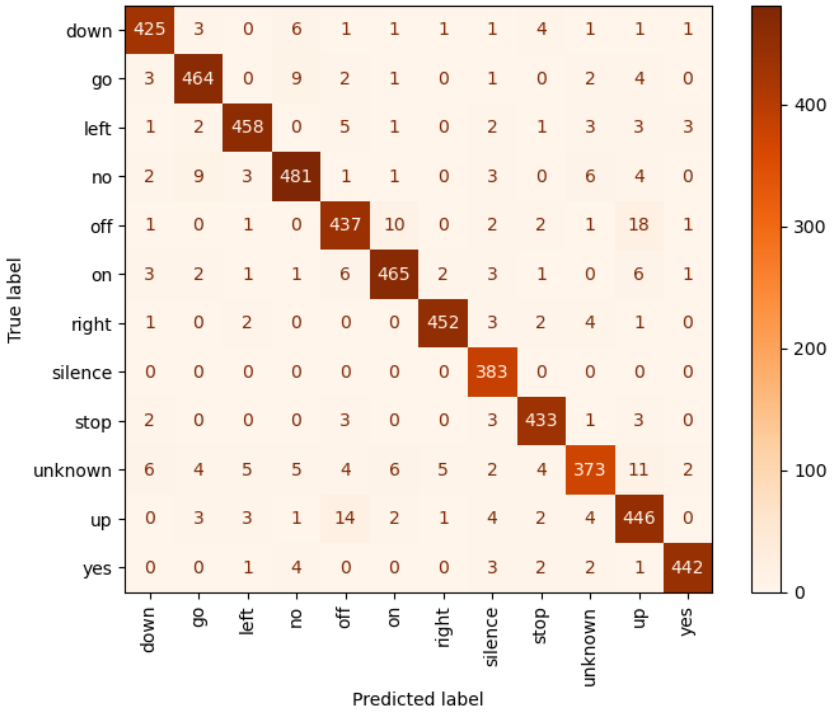


Figure 5.5: Confusion matrix for integrated MTSC-based KWS model.

Further, to attest to the generalizability of the proposed MTSC-based KWS models, their performance was examined on GSC V2, the results of which are provided in Fig. 5.6. The impressive performance of the proposed integrated MTSC-based KWS model, with an accuracy of 94.34%, on this dataset as well, validated its applicability in practical applications.

Having validated the performance of the proposed KWS models on different datasets, we finally compare their performance with that of the existing state-of-the-art KWS methods in table 5.5. The results in the table clearly indicate that the proposed integrated MTSC-based KWS model generally performs better than the existing methods, especially when the number of trainable parameters is also considered. Specifically, with an accuracy of 94.95%, the proposed integrated MTSC-based KWS model outperforms all the existing KWS methods, except for the two transformer-based models KWT-3 [19] and Swin-Transformer [20]. However, although the classification accuracy of these transformer-based models is slightly higher, the number of trainable parameters, essentially indicating the model complexity, for these models is magnitudes higher as compared to our proposed integrated MTSC-based KWS model. Precisely, the KWT-3 model [19] has total trainable parameters of 5.36M, whereas the two models namely the Swin-Transformer-based model and TCN+Swin-Transformer-based model in [20], have total trainable parameters of 3.07M and 4.94M, respectively. In contrast to such huge models, the proposed ROCKET-

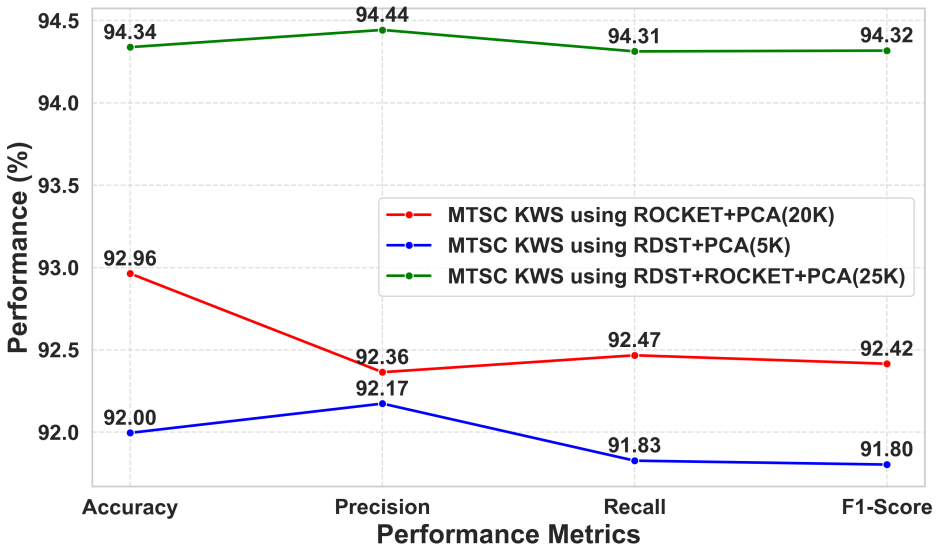


Figure 5.6: Classification performance of the proposed KWS models on GSC V2 dataset.

based MTSC KWS model has only 240K trainable parameters, the RDST-based MTSC KWS model has a total of 60K trainable parameters, and the integrated MTSC-based KWS model has a total of 300K trainable parameters. Fig. 5.7 shows the performance comparison of different KWS models along with the corresponding number of trainable parameters, reiterating the efficacy of the proposed KWS models. Additionally, unlike the other state-of-the-art deep learning-based KWS methods, our proposed integrated MTSC-based KWS model employs a simple ridge classifier for the classification, thus lowering its computational complexity even further. Overall, this makes the proposed integrated MTSC-based KWS model an efficient and ideal choice for small-footprint applications. Having presented the detailed classification performance of the proposed KWS models and their comparison with the contemporary KWS methods, next we present an ablation study conducted for the optimal design of the proposed KWS models.

Model	**Accuracy (%)**
Enhanced DNN with Frame Stacking and Pooling [13]	91.2
DenseNet structure [14]	92.8
Few shot learning based KWS [16]	94.0
Metric learning upon ResNet-based baseline [17]	83.82
Vision transformer based KWS [18]	94.8
KWT-3 [19]	97.49
Swin-Transformer [20]	87.49
TCN + Swin-Transformer [20]	98.01
MFCC+CNN-based KWS model (Ours)	91.69
MTSC ROCKET+PCA based KWS model(Ours)	94.82
MTSC RDST+PCA based KWS model (Ours)	93.36
MTSC ROCKET+RDST+PCA based KWS model(Ours)	94.94

Table 5.5: Classification performance of different KWS methods.

To validate our results further, we have implemented all the three proposed KWS

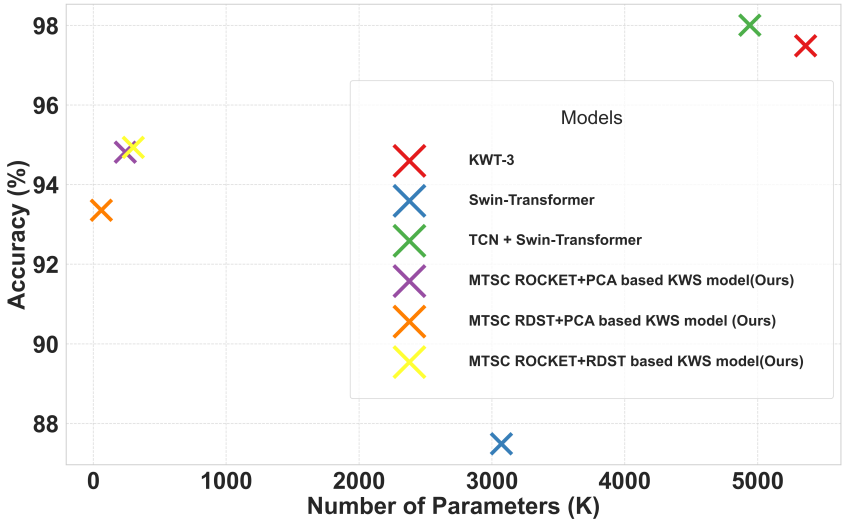


Figure 5.7: Comparison of trainable parameters vs achieved accuracies for different KWS methods for GSC V1.

models on a different speech dataset called Free Spoken Digit Dataset (FSDD). The accuracy obtained on the MTSC based KWS models for both ROCKET and RDST are both 98% which is the highest obtained performance so far. The integrated MTSC KWS model boosts it further to 98.7% accuracy. The details about the kernels/shapelets used and the total parameters are given in table 5.6.

MTSC	Kernels/Shapelets	PCA Components	Trainable Parameters	Accuracy
ROCKET	50K	2K	24K	98.06
RDST	50K	2K	24K	98.25
ROCKET+RDST	25K+25K	2K	24K	**98.77**

Table 5.6: Classification performance of MTSC-based KWS models on FSDD

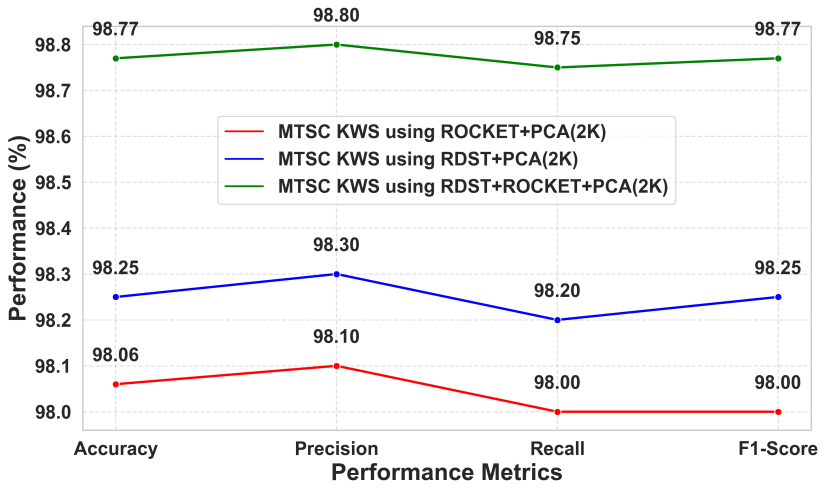


Figure 5.8: Classification performance of the proposed KWS models on FSSD

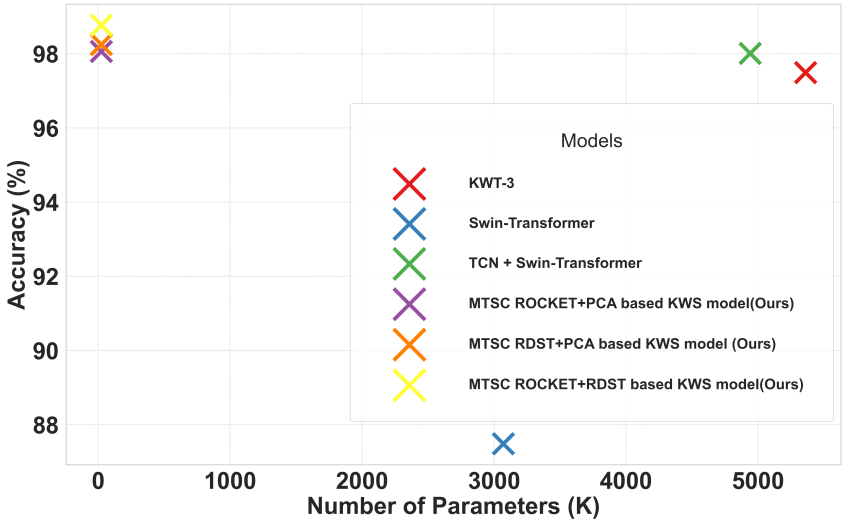


Figure 5.9: Comparison of trainable parameters vs achieved accuracies for different KWS methods for FSSD

Figure 5.8 shows a linegraph of all the performance metrics obtained for the FSDD using the three proposed models. Figure 5.9 plots the accuracy vs. parameters for the proposed models using FSDD against the state-of-the-art KWS models.

5.5.3 Ablation studies

Optimal window and hop size

As described in the design of the proposed MTSC-based KWS approach, the values of the window length and hop size determine the MFCC matrix and thereby, the resulting multivariate time series. So to obtain the optimal multivariate time series representation from the audio, we tried using different combinations of window length and hop size as shown in Fig. 5.10, wherein the combination of 25 ms window with a hop size of 10 ms was found to be the best, and hence was used in the subsequent analysis.

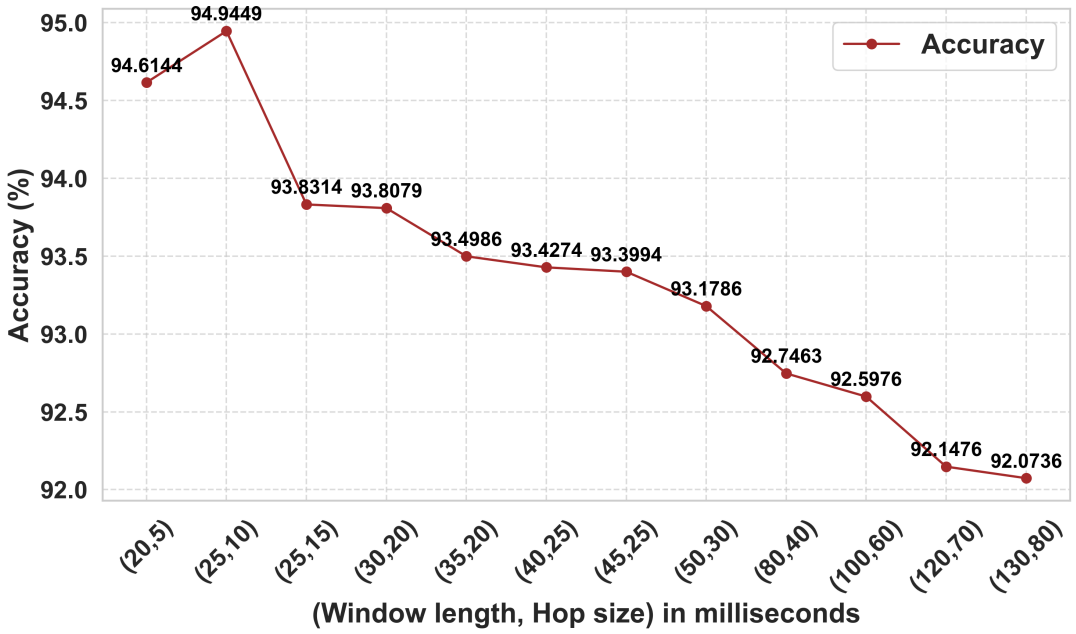


Figure 5.10: Classification accuracy of MTSC-based KWS model for different window size and hop size combinations.

Acoustic feature selection

In the process of obtaining the multivariate time series from the windowed audio signal, we examined a few standard acoustic feature extractors like MFCC (both 13 and 39 coefficients) and spectrograms to determine the most suitable one. The analysis, as summarized in Fig. 5.11, indicated that 13-length MFCC features produced the optimal classification result and hence was the choice in entire MTSC-based KWS models. With this choice of acoustic features and the aforementioned window length and the hop size, the dimension of the final multivariate time series representation of the audio came out to be 13×99, with the calculations as detailed below.

Feature vector dimension calculation:

$$\text{MFCC feature vector dimension} = [\text{number of MFCC coefficients}, \text{number of time frames}]$$

The number of time frames can be computed as follows. The total number of time points for our signal is $T = 16000$ samples (corresponding to 1 second of audio at a sample rate of 16 kHz). Given that our window size is 25 ms, or 400 samples, and the hop size is 10 ms, or 160 samples, the number of time frames is:

$$\text{Number of time frames} = \left(\frac{T - \text{Window size}}{\text{Hop size}}\right) + 1$$

Substituting the values:

$$\text{Number of time frames} = \left(\frac{16000 - 400}{160}\right) + 1 = 99$$

Thus, the final multivariate time series representation of the audio acquires the dimension of 13×99. For visualization purpose, Fig. 5.12, 5.13 and 5.14 shows the representative multivariate time series obtained from one sample audio input (corresponding to the word "left").

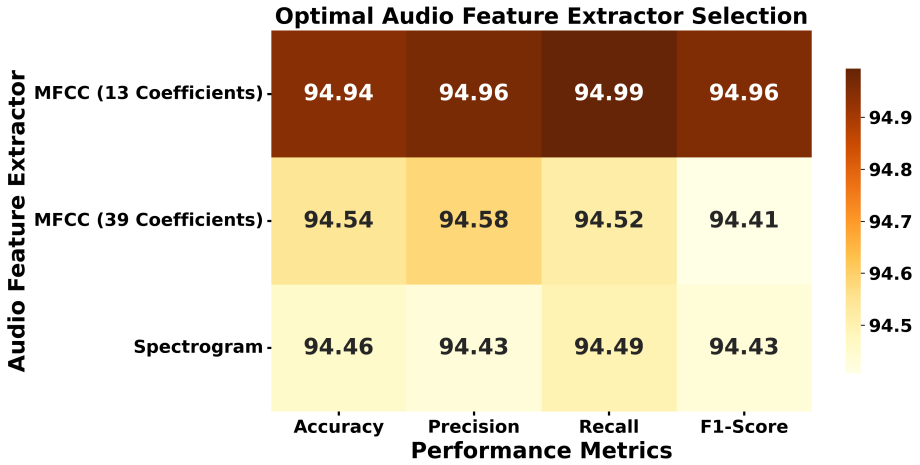


Figure 5.11: Classification performance of MTSC-based KWS model using different acoustic features.

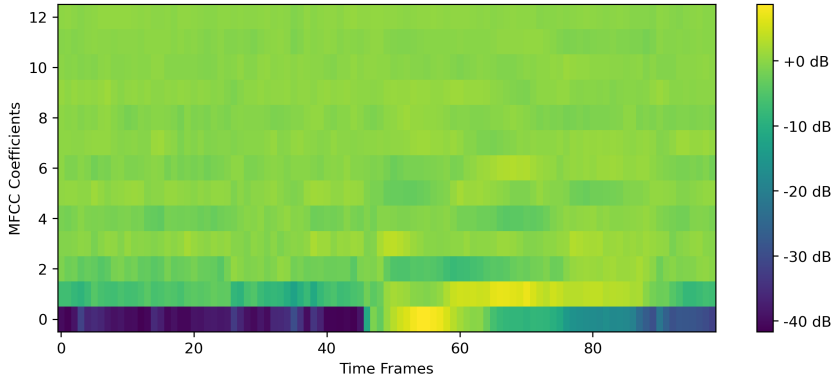


Figure 5.12: MFCC (13 coefficients) representation of an audio sample.

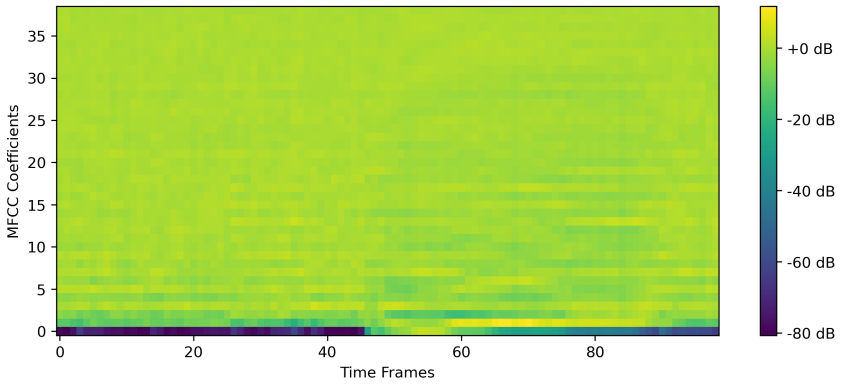


Figure 5.13: MFCC (39 coefficients) representation of an audio sample.

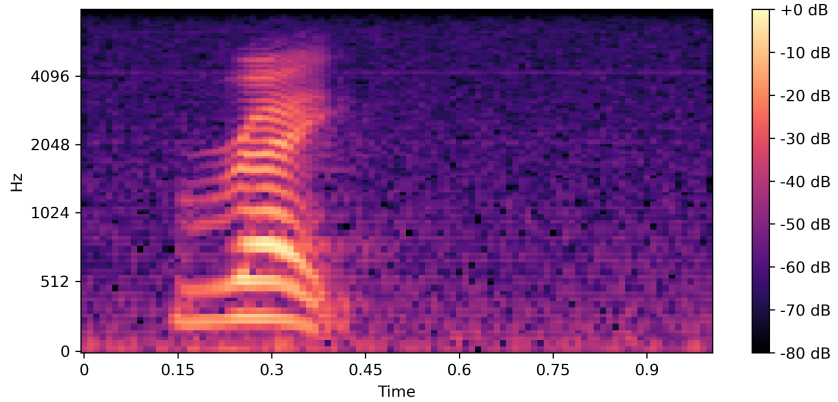
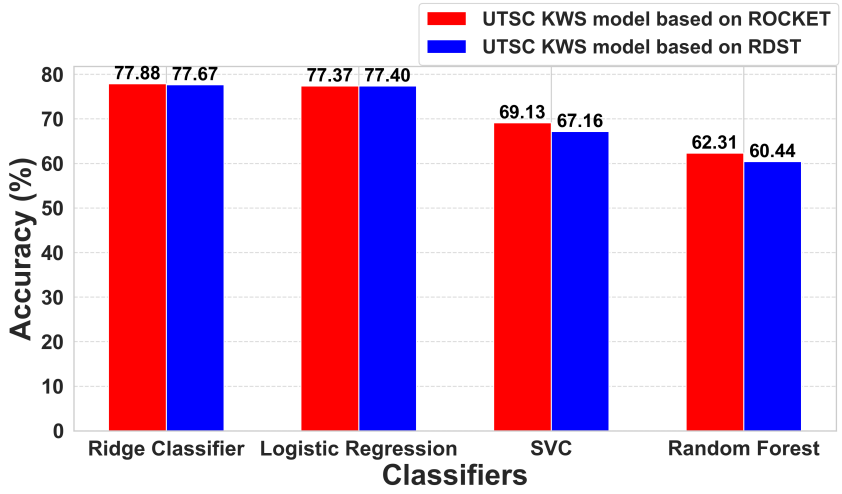
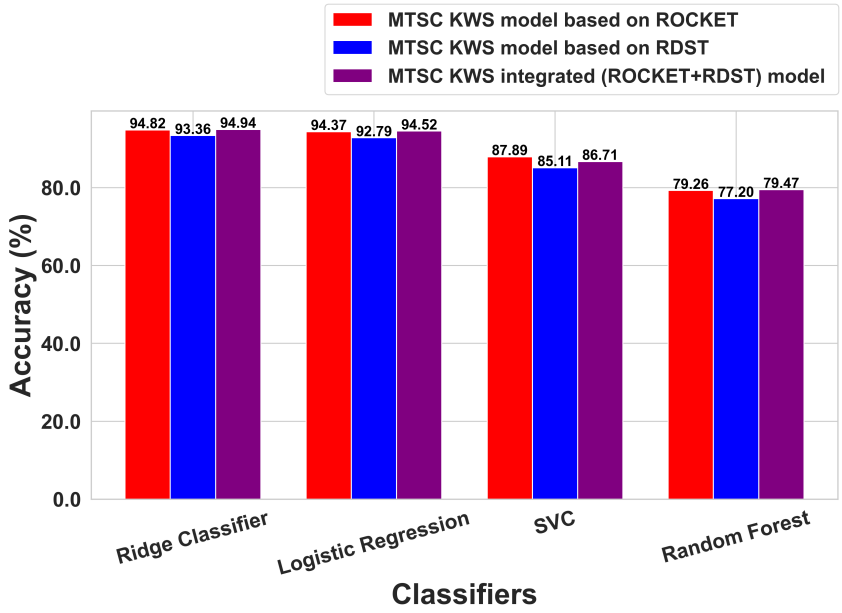


Figure 5.14: Spectrogram representation of an audio sample.

Optimal classifier selection



(a)



(b)

Figure 5.15: Classification accuracy comparison of ROCKET and RDST-based KWS models using different classifiers for (a) UTSC and (b) MTSC models.

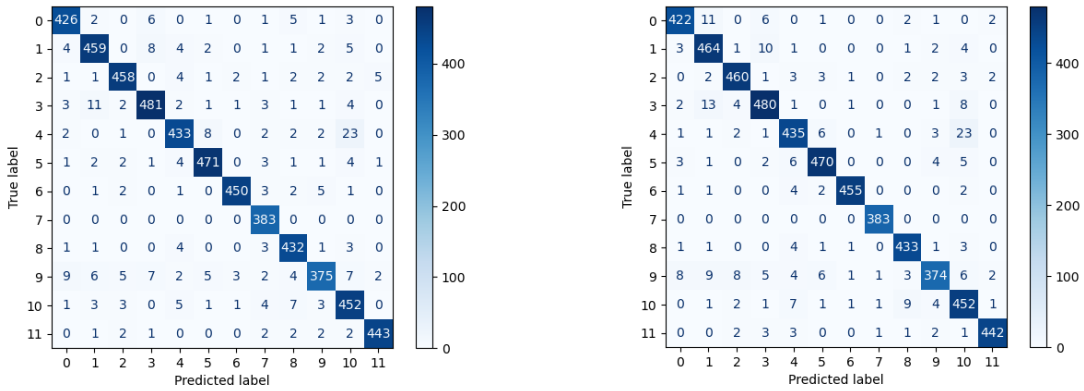
As mentioned earlier in this section, to optimize the classification performance of the proposed UTSC and MTSC-based KWS models, different classifiers such as ridge, logistic regression, SVC, and random forest were examined to determine the most suitable one. However, the comparison results, presented in Fig. 5.15(a) and 5.15(b), indicated that the default ridge classifier remains optimal for both models, yielding the best classification performance.

Number of kernels and shapelets

As discussed in the analysis of the proposed MTSC-based KWS model, to fine-tune the feature extraction and hence the classification performance of the model, different number of kernels and shapelets were tried, for the ROCKET and the RDST, respectively. Table 5.7 summarizes the performance of the ROCKET and the RDST-based MTSC models, before parameter reduction, for varying numbers of kernels and shapelets, respectively. It indicates that 120k kernels for ROCKET and 50k shapelets for RDST provide the most accurate results and hence were used in the final integrated MTSC-based KWS model.

MTSC	Kernels/ Shapelets	Performance Metrics on GSC V1			
		Accuracy	***Precision***	***Recall***	***F1-Score***
ROCKET	10K	89.7996	89.8721	89.9091	89.8648
	20K	93.4645	93.5073	93.5174	93.4890
	30K	94.3852	94.4188	94.4175	94.3952
	50K	94.7102	94.7543	94.7368	94.7200
	100K	94.9629	94.9781	95.0160	94.9776
	120K	**95.0171**	**95.0503**	**95.0648**	**95.0277**
RDST	10K	93.6270	93.7773	93.6418	93.6600
	30K	94.8366	94.9713	94.8888	94.9088
	50K	**95.1435**	**95.2802**	**95.1829**	**95.2094**
	70K	95.0171	95.1595	95.0688	95.0942

Table 5.7: Classification performance of RDST and ROCKET-based MTSC KWS models for varying input parameters



(a) **(b)**

Figure 5.16: Confusion matrices for MTSC KWS models: (a) ROCKET-based with 120K kernels and (b) RDST-based with 50K kernels.

5.6 Discussion

The findings presented in the previous section provide important insights into the performance and effectiveness of the proposed keyword spotting (KWS) models. In this section, we analyze these outcomes further to understand their significance and highlight the contributions of the proposed approaches.

The results demonstrate that both feature extraction techniques and model architecture choices significantly influence KWS accuracy, precision, recall, and F1-score. The Univariate Time Series Classification (UTSC)-based KWS models offered a lightweight yet effective alternative to CNN-based systems. The method resulted in 77.88% accuracy with ROCKET features. Although the results support our hypothesis of employing time series feature extractors and classifiers for keyword spotting, the proposed UTSC-based model operates directly on raw audio time series. As a result, unlike the MFCC-CNN-based model, it cannot explicitly exploit spectral information available in acoustic features such as MFCC. This inherent limitation in feature representation contributed to its relatively lower performance compared to the basic MFCC-CNN-based model.

In contrast to the UTSC-based model, the results indicate the superior performance of the proposed Multivariate Time Series Classification (MTSC)-based techniques in the keyword spotting task. In particular, the proposed MTSC-based KWS model using ROCKET achieved an accuracy of 94.82% on the GSC V1 and 98.06% on FSDD, outperforming the UTSC and the MFCC-CNN-based model. This improvement can be attributed to the ability of the MTSC-based approach to effectively exploit both temporal and spectral dimensions of the audio signal. By operating on multivariate representations—such as MFCC feature sequences—these model is capable of capturing complex temporal patterns as well as spectral characteristics essential for distinguishing between different keywords. The integration of ROCKET and RDST-based features further improved classification outcomes. The integrated MTSC-based model, even with reduced feature sizes and a simple ridge classifier, achieved an impressive accuracy of 94.94% on the GSC V1 and 98.77% on FSDD, outperforming individual models and validating the advantage of feature fusion.

Furthermore, the proposed integrated MTSC-based model eliminates the need for complex backpropagation during training as required by the deep learning-based methods, thereby substantially reducing both training time and computational overhead. The proposed MTSC-based KWS model achieves a competitive accuracy of 94.94% on the GSC V1 and 98.77% on FSDD while utilizing a significantly lower number of trainable parameters compared to transformer-based models, thus maintaining a low memory footprint. The adoption of a simple ridge classifier further minimizes computational complexity, making the model highly suitable for deployment in edge devices and low-resource environments.

Conclusion and Future Work

In this study, we addressed the challenge of designing a small-footprint, language-agnostic, ASR-free keyword spotting model that can be deployed in low-resource environments like edge devices with limited computation power and memory. As proposed, the keyword spotting task was formulated as a classification model. The distance based approaches utilized MFCC feature extraction coupled with different traditional distance-based methods like DTW to perform the keyword matching task. Out of all the methods, DTW has shown the maximum accuracy of 72%. Then, a basic MFCC-CNN-based KWS model was proposed which has resulted in an accuracy of 91.70%.

As part of the major contribution of this work, the problem of keyword spotting was uniquely formulated as a time series classification task, which was solved using efficient unsupervised time series-specific feature extractors and simple machine learning classifiers, thus bypassing the need for huge deep learning-based models, resulting in a lightweight model. Specifically, the proposed KWS models modeled the audio data as either univariate or multivariate time series wherein the former extracted only the temporal features from the audio, whereas the latter exploited both the temporal and acoustic features, thus resulting in improved detection performance. To obtain the optimal detection performance, various time series feature extractors were explored, and finally, the best-performing features obtained using ROCKET and RDST were combined to design the integrated MTSC-based KWS model. To verify the utility of the proposed KWS models, their performance was evaluated on two publicly available datasets. Comparison with existing KWS methods showed that, with a classification accuracy of $\sim$95% on GSC daataset and 98% on FSDD, the proposed integrated MTSC-based KWS model produces state-of-the-art performance with much lower memory and computational requirements, thus advocating its application in low-resource environments such as edge devices.

Although the proposed integrated MTSC-based KWS model provided an attractive alternative for the low-footprint KWS, the models can be further tuned to achieve more precise results. The current models have utilized only MFCC and spectrograms for acoustic feature extraction; hence, other time-frequency representations, optimal for the present application, can also be explored. Further, the current models employed only PCA for reducing the feature dimensions. Apart from this, more sophisticated compression techniques like encoders can also be exploited, although this may increase the overall model complexity, which will need further optimizations.

Appendix A

Distance algorithms

FastDTW(A, B, radius):
1. Input:
A' = Downsample(A)
B' = Downsample(B)
2. D' = ComputeDTW(A', B',radius) // Compute DTW at coarse level
3. RefinedWindow = GetWindowAroundCoarseAlignment(D') // Refine alignment
4. D = ComputeDTW(A, B, window = RefinedWindow) // Compute DTW at original level
5. return D[n][m]

Table A.1: Fast DTW Algorithm

SegmentalDTW(A, B, ϵ)
1. Input:
A: Time series sequence 1 (length n)
B: Time series sequence 2 (length m)
ϵ: Threshold for segmenting the sequences
2. Output:
D: DTW distance matrix between A and B
2. Segments_A = DivideIntoSegments(A, ϵ)
3. Segments_B = DivideIntoSegments(B, ϵ)
4. For each segment A_i in Segments_A:
For each segment B_j in Segments_B:
Compute DTW distance D_ij = ComputeDTW(A_i, B_j)
5. Aggregate the segment distances into the overall DTW distance matrix D
6. Segment_Alignment_Path = ComputeSegmentAlignmentPath(D)
7. Frame_Level_Alignment_Path = BacktrackOptimalPath(D, Segment_Alignment_Path)
8. Return D[n][m]

Table A.2: Segmental DTW Algorithm

WeightedDTW(X, Y, w_max, gamma)
1. *Input:*
   X: Time series sequence 1 (length n)
   Y: Time series sequence 2 (length m)
   w_max: Maximum weight (e.g., 1)
   gamma: Parameter controlling the penalty level for warping

Output:
   weighted_dtw_distance: Weighted DTW distance between X and Y

2. *Initialize distance matrix M_w with size* $(n+1) \times (m+1)$*:*
3. *Define weight penalty function:*
   $w(i) = \frac{w\_max}{1+\exp(-gamma \cdot (i-m/2))}$
4. *Fill in the distance matrix M_w:*
   For $i = 1$ *to* n*:*
     For $j = 1$ *to* m*:*
       Compute weight penalty: $w_{ij} = w(|i-j|)$
       Compute squared distance: $d_{ij} = (X[i-1] - Y[j-1])^2$
       Update distance matrix $M\_w[i][j]$*:* $M\_w[i][j] = w_{ij} \cdot d_{ij}$
5. *Compute the DTW distance using the weighted distance matrix M_w:*
   Initialize DTW matrix D with size $(n+1) \times (m+1)$*:*
     For $i = 1$ *to* n*:*
       For $j = 1$ *to* m*:*
         $D[i][j] = M\_w[i][j] + \min(D[i-1][j], D[i][j-1], D[i-1][j-1])$
6. *Return weighted_dtw_distance* $= D[n][m]$

Table A.3: Weighted DTW Algorithm

LCSS(X, Y, δ)
1. Input:
X: Time series sequences of length m
Y: Time series sequences of length n
δ: Equality threshold
2. Output:
$L_{m,n}$: LCSS length between sequences X and Y
3. Initialize L as an $(m+1) \times (n+1)$ matrix with zeros, indexed from zero
For $i = 0$ to m:
For $j = 0$ to n:
If $|X_i - Y_j| < \delta$: // Difference is less than allowed threshold
$L_{i,j} = L_{i-1,j-1} + 1$ *// Match found, increment the subsequence length*
Else:
$L_{i,j} = \max(L_{i-1,j}, L_{i,j-1})$ *// No match, take max of neighbors*
4. Return LCSS Distance = $1 - \frac{L_{m,n}}{\min(n,m)}$

Table A.4: LCSS Algorithm

TWE(X, Y, e_cost, warp_penalty, dist_func)
1. Input:
X, Y: Time series sequences of length m
e_cost: Editing cost
warp_penalty: Warping penalty factor
dist_func: Pointwise distance function
2. Output:
$D_{m,n}$: TWE distance between sequences X and Y
3. Initialize D as an $(m+1) \times (n+1)$ matrix with zeros
$D_{0,0} = 0$
4. For $i = 1$ to m:
$D_{i,0} = \infty$
5. For $j = 1$ to n:
$D_{0,j} = \infty$
6. For $i = 1$ to m:
For $j = 1$ to n:
$alignment = D_{i-1,j-1} + dist\_func(X_i, Y_j) + dist\_func(X_{i-1}, Y_{j-1}) + 2 \cdot warp\_penalty \cdot |i - j|$
$remove = D_{i-1,j} + dist\_func(X_i, X_{i-1}) + e\_cost + warp\_penalty$
$add = D_{i,j-1} + dist\_func(Y_j, Y_{j-1}) + e\_cost + warp\_penalty$
$D_{i,j} = \min(alignment, add, remove)$
7. Return $D_{m,n}$

Table A.5: TWE Algorithm

EDR(X, Y, θ)
1. *Input:*
   X: *Time series sequences of length* m
   Y: *Time series sequences of length* n
   θ: *Equality threshold*
2. *Output:*
   $E_{m,n}$: *EDR distance between sequences* X *and* Y
3. *Initialize* E *as an* $(m+1) \times (n+1)$ *matrix with zeros, indexed from zero*
   For $i = 0$ *to* m:
     For $j = 0$ *to* n:
       If $i = 0$ *and* $j = 0$:
         $E_{i,j} = n$
       If $|X_i - Y_j| < \theta$:
         $cost = 0$
       Else:
         $cost = 1$
       Compute $match = E_{i-1,j-1} + cost$
       Compute $insert = E_{i-1,j} + 1$
       Compute $delete = E_{i,j-1} + 1$
       $E_{i,j} = \min(match, insert, delete)$
4. *Return* $E_{m,n}$

Table A.6: EDR Algorithm

MSM(x, y, c, p, $Cost$)
1. *Input:*
   x: *Time series sequences of length* m
   y: *Time series sequences of length* n
   c: *Minimum cost*
   p: *Pointwise distance function*
   $dist\_func$: *Cost function for operations*
2. *Output:*
   $D_{m,n}$: *MSM distance between sequences* x *and* y
3. *Initialize* D *as an* $m \times n$ *matrix with zeros*
   $D_{1,1} = p(x_1, y_1)$
4. *For* $i = 2$ *to* m:
     $D_{i,1} = D_{i-1,1} + Cost(x_i, x_{i-1}, y_1)$
5. *For* $j = 2$ *to* n:
     $D_{1,j} = D_{1,j-1} + Cost(y_j, x_1, y_{j-1})$
6. *For* $i = 2$ *to* m:
     For $j = 2$ *to* n:
       compute $match = D_{i-1,j-1} + p(x_i, y_j)$
       compute $insert = D_{i-1,j} + Cost(x_i, x_{i-1}, y_j)$
       compute $delete = D_{i,j-1} + Cost(y_j, y_{j-1}, x_i)$
       $D_{i,j} = \min(match, insert, delete)$
7. *Return* $D_{m,n}$

Table A.7: MSM Algorithm

Bibliography

[1] Lines, J., Bagnall, A. "Time series classification with ensembles of elastic distance measures." *Data Min Knowl Disc* 29, 565–592 (2015). https://doi.org/10.1007/s10618-014-0361-2

[2] Yasushi Sakurai, Masatoshi Yoshikawa, Christos Faloutsos, "FTW: Fast Similarity Search under the Time Warping Distance," PODS 2005, June 13–15, 2005, Baltimore, Maryland. ACP 1–59593–062–0/05/06. pp 1–11.

[3] Keogh, Eamonn J., and Michael J. Pazzani. "Scaling up dynamic time warping to massive datasets," In European Conference on Principles of Data Mining and Knowledge Discovery, pp. 1–11. Springer, Berlin, Heidelberg, 1999.

[4] Jeong Y, Jeong M, Omitaomu O (2011) "Weighted dynamic time warping for time series classification," *Pattern Recognition* 44:2231–2240

[5] Paterson, M., Dančík, V. (1994). "Longest common subsequences," In: Prívara, I., Rovan, B., Ruzička, P. (eds) *Mathematical Foundations of Computer Science 1994. MFCS 1994. Lecture Notes in Computer Science*, vol 841. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-58338-6_63

[6] Marteau PF (2009) "Time warp edit distance with stiffness adjustment for time series matching," *IEEE Trans Pattern Anal Mach Intell* 31(2):306–318

[7] Chen L, Özsu MT, Oria V (2005) "Robust and fast similarity search for moving object trajectories," In: *Proceedings of the 2005 ACM SIGMOD international conference on management of data*, pp 491–502. ACM

[8] Stefan A, Athitsos V, Das G (2012) "The move-split-merge metric for time series," *IEEE Trans Knowl Data Eng* 25(6):1425–1438

[9] A. H. Michaely, X. Zhang, G. Simko, C. Parada, and P. Aleksic, "Keyword spotting for Google Assistant using contextual speech recognition," in *Proc. IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, Okinawa, Japan, 2017, pp. 272–278, doi: 10.1109/ASRU.2017.8268946.

[10] I. López-Espejo, Z.-H. Tan, J. H. L. Hansen, and J. Jensen, "Deep spoken keyword spotting: An overview," *IEEE Access*, vol. 10, pp. 4169–4199, 2022, doi: 10.1109/ACCESS.2021.3139508.

[11] J. Macoskey, G. P. Strimel, and A. Rastrow, "Bifocal neural ASR: Exploiting keyword spotting for inference optimization," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, Toronto, ON, Canada, 2021, pp. 5999–6003, doi: 10.1109/ICASSP39728.2021.9414652.

[12] M. Pudo, M. Wosik, and A. Janicki, "Improved small-footprint ASR-based solution for open vocabulary keyword spotting," *IEEE Access*, vol. 12, pp. 91289–91299, 2024, doi: 10.1109/ACCESS.2024.3421605.

[13] O. Rybakov, et al., "Streaming keyword spotting on mobile devices," *arXiv preprint*, arXiv:2005.06720, 2020.

[14] X. Du, M. Zhu, M. Chai, and X. Shi, "End to end model for keyword spotting with trainable window function and DenseNet," in *Proc. IEEE 23rd Int. Conf. Digital Signal Processing (DSP)*, Shanghai, China, 2018, pp. 1–4, doi: 10.1109/ICDSP.2018.8631574.

[15] E. van der Westhuizen, H. Kamper, R. Menon, J. Quinn, and T. Niesler, "Feature learning for efficient ASR-free keyword spotting in low-resource languages," *Comput. Speech Lang.*, vol. 71, p. 101275, 2022.

[16] A. Parnami and M. Lee, "Few-shot keyword spotting with prototypical networks," in *Proc. 7th Int. Conf. Machine Learning Technologies (ICMLT)*, ACM, 2022, doi: 10.1145/3529399.3529443.

[17] J. Huh, M. Lee, H. Heo, S. Mun, and J. S. Chung, "Metric learning for keyword spotting," in *Proc. 2021 IEEE Spoken Language Technology Workshop (SLT)*, Shenzhen, China, 2021, pp. 133–140, doi: 10.1109/SLT48900.2021.9383571.

[18] A. Bittar, P. Dixon, M. Samragh, K. Nishu, and D. Naik, "Improving vision-inspired keyword spotting using dynamic module skipping in streaming conformer encoder," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, Seoul, South Korea, 2024, pp. 10386–10390, doi: 10.1109/ICASSP48485.2024.10447485.

[19] A. Berg, M. O'Connor, and M. Cruz, "Keyword Transformer: A self-attention model for keyword spotting," in *Proc. Interspeech*, 2021, pp. 4249–4253, doi: 10.21437/Interspeech.2021-1286.

[20] C. Sun, B. Chen, F. Chen, Y. Leng, and Q. Guo, "Speech keyword spotting method based on Swin-Transformer model," *Int. J. Comput. Intell. Syst.*, vol. 17, 2024, doi: 10.1007/s44196-024-00448-1.

[21] M. Middlehurst, P. Schafer, and A. Bagnall, "Bake off redux: A review and experimental evaluation of recent time series classification algorithms," *Data Mining and Knowledge Discovery*, vol. 38, pp. 1958–2031, 2024, doi: 10.1007/s10618-024-01022-1.

[22] A. Dempster, F. Petitjean, and G. Webb, "ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels," *Data Mining and Knowledge Discovery*, vol. 34, pp. 1454–1495, 2020, doi: 10.1007/s10618-020-00720-y.

[23] A. Dempster, D. Schmidt, and G. Webb, "MiniROCKET: A very fast (almost) deterministic transform for time series classification," in *Proc. 27th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD)*, 2021, pp. 248–257, doi: 10.1145/3447548.3467400.

[24] C. W. Tan, A. Dempster, C. Bergmeir, et al., "MultiROCKET: Multiple pooling operators and transformations for fast and effective time series classification," *Data Mining and Knowledge Discovery*, vol. 36, pp. 1623–1646, 2022, doi: 10.1007/s10618-022-00867-0.

[25] A. Guillaume, C. Vrain, and W. Elloumi, "Random dilated shapelet transform: A new approach for time series shapelets," in *Proc. Int. Conf. Pattern Recognition and Artificial Intelligence (ICPRAI)*, 2022.

[26] L. Ye and E. Keogh, "Time series shapelets: A novel technique that allows accurate, interpretable and fast classification," *Data Mining and Knowledge Discovery*, vol. 22, no. 1–2, pp. 149–182, 2011, doi: 10.1007/s10618-010-0179-5.

[27] A. Dempster, D. F. Schmidt, and G. I. Webb, "HYDRA: Competing convolutional kernels for fast and accurate time series classification," *arXiv preprint*, arXiv:2203.13652, 2022.

[28] A. Dempster, D. F. Schmidt, and G. I. Webb, "Quant: A minimalist interval method for time series classification," *arXiv preprint*, arXiv:2308.00928, 2023.

[29] M. Middlehurst, J. Large, M. Flynn, et al., "HIVE-COTE 2.0: A new meta-ensemble for time series classification," *Machine Learning*, vol. 110, pp. 3211–3243, 2021, doi: 10.1007/s10994-021-06057-9.

[30] H. Deng, G. Runger, E. Tuv, et al., "A time series forest for classification and feature extraction," *Information Sciences*, vol. 239, pp. 142–153, 2013, doi: 10.1016/j.ins.2013.02.030.

[31] M. Flynn, J. Large, and A. Bagnall, "The contract random interval spectral ensemble (c-RISE): The effect of contracting a classifier on accuracy," in *Proc. Hybrid Artificial Intelligence Systems*, *Lecture Notes in Computer Science*, vol. 11734, Springer, 2019, pp. 381–392, doi: 10.1007/978-3-030-27477-1_32.

[32] C. Lubba, S. Sethi, P. Knaute, et al., "catch22: Canonical time-series characteristics," *Data Mining and Knowledge Discovery*, vol. 33, no. 6, pp. 1821–1852, 2019, doi: 10.1007/s10618-019-00647-x.

[33] M. Middlehurst and A. Bagnall, "The FreshPRINCE: A simple transformation-based pipeline time series classifier," in *Proc. Int. Conf. Pattern Recognition and Artificial Intelligence*, Springer, 2022, pp. 150–161.

[34] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint*, arXiv:1804.03209, 2018.

[35] Jackson, Z. (2016). "Free Spoken Digit Dataset," GitHub repository. https://github.com/Jakobovski/free-spoken-digit-dataset

[36] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley Interdiscip. Rev. Comput. Stat.*, vol. 2, no. 4, pp. 433–459, 2010, doi: 10.1002/wics.101.

[37] J.-H. Choi and J.-S. Lee, "EmbraceNet: A robust deep learning architecture for multimodal classification," *Inf. Fusion*, vol. 51, pp. 259–270, Nov. 2019, doi: 10.1016/j.inffus.2019.02.010.

[38] "Aeon Toolkit," accessed Jan. 28, 2025. [Online]. Available: https://www.aeon-toolkit.org/en/stable/