修 士 論 文 概 要 書

Master's Thesis Summary

Date of submission: 01/27/2025 (MM/DD/YYYY)

専攻名 (専門分野) Department	Computer Science and Communications Engineering	氏 名 Name	Minghao Duan	指 導 教 員 Advisor	Hiroshi Watanabe 印 Seal
研究指導名 Research guidance	Research on Audiovisual Information Processing	学籍番号 Student ID number	CD 5123FG10-1		
研究題目 Title	A Multi-Component Framework for Speaker Recognition: Leveraging Differentiable Architecture Search and Temporal Dependencies				

1. Introduction

Speaker recognition aims to identify or verify individuals based on their unique voice features, with application in various fields such as virtual assistants and security systems. Traditional models typically rely on Convolutional Neural Network (CNN) backbone networks such as VGG-Net [1] or ResNet [2]. However, research have shown that these architectures are not optimally suited for speaker recognition tasks, as they struggle to effectively capture the temporal and global dependencies in speech data [4]. Manually searching for better architecture is also time-consuming. Differentiable Architecture Search (DARTS) [3] automates the search for optimal architectures and has shown success in image classification tasks. AutoSpeech [4] was the first to apply a pure DARTS-optimized network to speaker recognition, demonstrating the feasibility and effectiveness of this approach in speaker recognition.

This thesis proposes a hybrid framework that extends a DARTS-optimized CNN with self-attention pooling and Long Short-Term Memory (LSTM) networks. Self-attention pooling dynamically captures global dependencies and speaker-specific features, while LSTM modules address the sequential nature of speech by modeling long-term temporal patterns. Experimental results on VoxCeleb1 [6] and VoxCeleb2 [7] datasets demonstrate that the proposed framework surpasses AutoSpeech in both speaker identification and verification tasks. These results validate the effectiveness of proposed multi-component framework to address the complex challenges of speaker recognition tasks.

2. Related Works

2.1 Differentiable Architecture Search (DARTS)

Neural Architecture Search (NAS) automates the process of designing neural network architectures by exploring a defined search space to identify optimal structures. Early NAS methods automate the design of neural networks and achieve notable performance improvements, however, since the method relies on reinforcement learning or evolutionary algorithms, which are computationally expensive due to the independent training of sampled architectures. DARTS address this limitation by introducing a gradient-based approach. By relaxing the search space into a continuous domain, DARTS significantly reduces the computational cost while maintaining high-quality architecture discovery. This efficient optimization process allows for faster and more effective architecture design, making it a cornerstone for automated neural network development.

2.2 A Pure DARTS-based CNN Framework for Speaker Recognition: AutoSpeech

AutoSpeech is the first to apply DARTS to speaker recognition tasks. VGG-Net and ResNet architectures that were originally designed for image classification, these architectures are not adequately adapted to the characteristics of speaker data. Instead, AutoSpeech leverages DARTS to automate the search for network architectures optimized for speaker recognition tasks. By using a gradient-based optimization approach, AutoSpeech avoids the need for a large number of independently trained architectures in the early NAS method, achieves better performance compared to VGG-Net and ResNet.

2.3 Self-Attention Pooling

Self-attention pooling [5] is a feature aggregation mechanism, which can dynamically capture global dependencies, and is suitable for processing sequence data such as speech. Unlike traditional average or maximum pooling, self-attention pooling emphasizes the most important information by assigning different weights to each frame, thus generating a more semantically expressive fixed-length representation. This adaptive mechanism effectively enhances the model's ability to capture key features while preserving global contextual information.

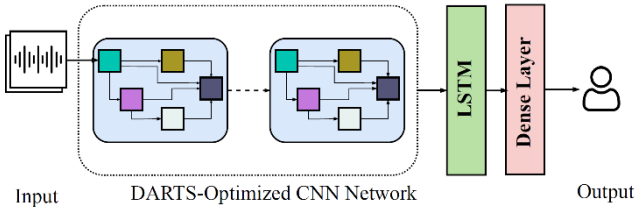

Fig. 1. Overview of the proposed method

3. Proposed Method

As shown in Fig. 1, our proposed approach consists of the following three steps: (1) DARTS architecture search, (2) replacing max pooling with self-attention

pooling, (3) integrating long short-term memory network (LSTM).

3.1 DARTS Architecture Search

The first step in the proposed method is to search for the optimal CNN architecture suitable for the speaker recognition task using DARTS. DARTS transforms the architecture search problem into a differentiable optimization process. By relaxing the search space into a continuous domain, the candidate operations on each edge are represented as weighted hybrid forms, and these weights are continuously optimized by gradient descent during the search process. As the search process advances, the operations with the highest weights are selected, resulting in optimal architecture.

3.2 Replacing Max Pooling with Self-Attention Pooling

After determining the CNN architecture through DARTS, the next step is to replace the maximum pooling layer in the network with a self-attention pooling module. Compared to the direct introduction of self-attention pooling in the DARTS search phase, this step is performed after the architecture search is complete, which significantly reduces the computational cost. Introducing the attention mechanism in the search phase would require additional resources to compute the weights, thus increasing the training overhead. By separating architecture search from module replacement, the advantages of self-attentive pooling in capturing long time dependencies can be fully exploited while ensuring computational efficiency.

3.3 Integrating Long Shot-Term Memory Network (LSTM)

The final step is to integrate the LSTM module after the DARTS-optimized CNN network in the final training phase. To avoid overfitting, LSTM uses a single-layer structure with hidden states of the same dimension as the CNN output to ensure compatibility. At this stage, the sequence of feature maps generated by the CNN is used as input to the LSTM, which processes the data sequentially to capture temporal patterns in speech. In addition, to address the instability that may be induced by training with long sequences, a gradient cropping technique is introduced in the training phase to prevent gradient explosion and ensure stable convergence.

Method	VoxCeleb1			VoxCeleb2
	Top-1(%)	Top-5(%)	EER(%)	EER(%)
ResNet-34	81.37	94.49	11.53	5.10
AutoSpeech	87.57	95.98	8.96	4.32
Proposed	**88.13**	**96.71**	**8.91**	**4.24**

Table 1. Comparative experimental results with ResNet and AutoSpeech

4. Experiment

4.1 Comparative experiment

We trained and evaluated our proposed model on the VoxCeleb1 and VoxCeleb2 datasets. We have selected ResNet and AutoSpeech for comparative evaluation. The experiments were performed with the same setup; for AutoSpeech and our method, we set the cell setting to 8 and the initial channel size to 128. As shown in Table 1, the proposed method demonstrated superior performance across all metrics, including Top-1 accuracy, Top-5 accuracy, and equal error rate (EER) for both speaker identification and verification tasks.

Method	Top-1(%)	Top-5(%)	EER(%)
Proposed w/o LSTM	88.04	96.42	8.93
Proposed w/o self-attention pooling	88.09	96.39	8.95
Proposed	**88.13**	**96.71**	**8.91**

Table 2. Experimental results of the ablation study

4.2 Ablation Study

In addition to comparing the overall performance, we conducted an ablation study to assess the contributions of individual components in our model. The study involved evaluating two variations: one without the LSTM module and the other without the self-attention pooling. The result is shown in Table 2, removing the LSTM resulted in a slight decrease in Top-1 and Top-5 accuracies, as well as a marginal increase in EER, highlighting its importance in capturing temporal dependencies. Similarly, replacing self-attention pooling with max pooling led to a decline in both identification accuracy and verification performance, emphasizing its role in capturing global feature dependencies.

5. Conclusion

This study introduces a hybrid framework for speaker recognition, combining a DARTS-optimized CNN architecture with self-attention pooling and LSTM modules. The proposed method effectively enhances performance by addressing both temporal and global dependencies in speech data, improving flexibility and accuracy for speaker identification and verification tasks. Ablation studies confirm the effectiveness of LSTM in capturing temporal dependencies and self-attention pooling in representing global features, enabling a balanced feature extraction process. Overall, this work demonstrates the effectiveness of proposed multi-component framework that integrates DARTS with feature extraction mechanisms in enhancing speaker recognition systems.

Reference

[1] K. Simonyan et al., "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv:1409.1556, Sep. 2014.

[2] K. He et al., "Deep Residual Learning for Image Recognition," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778, Jun. 2016.

[3] H. Liu et al., "DARTS: Differentiable Architecture Search," arXiv:1806.09055, Jun. 2018.

[4] S. Ding et al., "AutoSpeech: Neural Architecture Search for Speaker Recognition," in Proceedings of INTERSPEECH, Oct. 2020.

[5] P. Safari et al., "Self-Attention Encoding and Pooling for Speaker Recognition," in Proceedings of INTERSPEECH, pp. 921–925, Oct. 2020.

[6] A. Nagrani et al., "VoxCeleb: A Large-Scale Speaker Identification Dataset," in Proceedings of INTERSPEECH, pp. 161–165, Aug. 2017.

[7] J. S. Chung et al., "VoxCeleb2: Deep Speaker Recognition," in Proceedings of INTERSPEECH, pp. 1086–1090, Sep. 2018.

A Multi-Component Framework for Speaker Recognition: Leveraging Differentiable Architecture Search and Temporal Dependencies

A Thesis Submitted to the Department of Computer Science and Communications Engineering, the Graduate School of Fundamental Science and Engineering of Waseda University in Partial Fulfillment of the Requirements for the Degree of Master of Engineering

Submission Date: Jan 27th, 2025

Minghao Duan
(5123FG10-1)

Advisor: Prof. Hiroshi Watanabe
Research guidance: Research on Audiovisual Information Processing

Acknowledgement

First and foremost, I would like to express my sincere gratitude to my advisor, Professor Hiroshi Watanabe, for his kind support and guidance throughout my graduate studies. His approach allowed me the freedom to independently explore my research interests while always being available to provide valuable advice when needed. I am also deeply thankful for his genuine care and encouragement in my daily life. His kindness and understanding have made my graduate studies more manageable and enjoyable, for which I am truly grateful.

I am also sincerely thankful to all the professors who have taught me over the past two years. Their dedication and insightful lectures have broadened my knowledge and provided me with a strong foundation in the field of computer science and engineering.

I would like to extend my heartfelt thanks to the members of the Advanced Multimedia Systems Laboratory. Their collaboration, feedback, and camaraderie have made my time in the lab both productive and enjoyable. The stimulating discussions and mutual support have been truly inspiring.

Lastly, I would like to express my deepest appreciation to my family and friends. Their unwavering support, understanding, and encouragement have been my greatest source of strength throughout this journey. Without them, this achievement would not have been possible.

Abstract

Speaker recognition is a critical field with applications in security, user authentication, and personalized services. Despite the success of traditional Convolutional Neural Networks (CNNs), their limited capacity to capture temporal and global dependencies in speech data constrains their performance in speaker recognition tasks. To address this limitation, we propose a hybrid framework combining a Differentiable Architecture Search (DARTS)-optimized CNN with self-attention pooling and long short-term memory (LSTM) modules.

The proposed framework comprises three stages: first, DARTS is utilized to search for the optimal neural architecture by formulating the problem as a differentiable optimization task. Second, self-attention pooling is incorporated to replace max pooling, enhancing the model's ability to capture global dependencies across the feature space. Finally, LSTM is integrated into the training phase to effectively model long-term temporal dependencies.

Experimental evaluations on the VoxCeleb1 and VoxCeleb2 datasets demonstrate that the proposed framework achieves superior performance compared to baseline models, including traditional CNNs and purely DARTS-optimized architectures. Ablation studies reveal the effectiveness of LSTM and self-attention pooling, demonstrating their contributions to feature representation. This study demonstrates the effectiveness of the DARTS-based hybrid architecture in improving speaker recognition accuracy.

Keywords: Deep Learning, Speaker Recognition, Differentiable Architecture Search (DARTS), Self-Attention Pooling, long short-term memory (LSTM)

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Research Background

Speaker recognition is a technology that recognizes or verifies identity based on an individual's unique acoustic characteristics, and is widely used in areas such as voice assistants, telephone banking, identity verification systems, and network security. By analyzing features such as frequency, duration and pitch of voice signals, speaker recognition systems are able to differentiate between different speakers, enabling personalized services and secure access control [1].

With the development of deep learning, convolutional neural networks (CNNs) have gradually become one of the main methods for speaker recognition tasks. CNN architectures such as VGG-Net [2] and ResNet [3], known for their excellent performance in image classification tasks, have been widely adopted in speaker recognition [4] [5] [6]. However, since these networks were originally designed for image classification related tasks, directly applying them to speech related tasks may leads to performance bottlenecks since the differences between audio and image signal [7]. The differences between these signals may lead to low efficiency and high model complexity in speech feature extraction. In addition to this, manually adjusting these architectures for speaker recognition tasks is time-consuming and limited by human expertise, making it challenging to achieve optimal performance. Consequently, designing network architectures that better align with the needs of speaker recognition has become a key focus for researchers.

Neural Architecture Search (NAS) [8] [9] offers a novel solution for optimizing CNNs

in speaker recognition tasks. NAS can automatically explore and optimize neural network structures, reducing the complexity of manual design and discovering efficient architectures for specific tasks. Among NAS technologies, Differentiable Architecture Search (DARTS) [10] is a notable innovation. DARTS transforms the discrete architecture search space into a continuous and differentiable space, allowing the search process to be optimized through gradient descent. This significantly enhances search efficiency and reduces computational costs. DARTS has achieved remarkable success in image classification tasks, inspiring researchers to explore its potential applications in other domains.

AutoSpeech [7] is the first method to apply DARTS to speaker recognition tasks, demonstrating the effectiveness of automated architecture search in speech applications. AutoSpeech automates the search for CNN architectures optimized for speaker recognition, reducing the need for manual intervention and improving accuracy and flexibility. However, considering AutoSpeech is a pure CNN network, limitations in speech signal processing still exist, and there is still room for further optimization, particularly in modeling long-term sequences and extracting complex speaker features.

1.2 Research Objectives

In this work, we aim to develop a refined framework for speaker recognition by integrating Differentiable architecture search (DARTS) with LSTM and self-attention pooling. The specific objectives of this study are summarized as follows:

1. We incorporate self-attention pooling and LSTM layers into the purely DARTS optimized network. These modules are designed to improve the model's ability to capture both global dependencies and temporal features in speech data, enabling better representation of speaker-specific characteristics.

2. We evaluate the proposed framework on standard speaker recognition benchmarks, such as VoxCeleb1 and VoxCeleb2 datasets. Experimental results demonstrate that the proposed framework achieves superior performance compared to traditional CNN methods and purely DARTS-optimized architectures in both speaker identification and verification tasks.

3. We conduct extensive ablation studies to analyze the contributions of individual

components in the proposed architecture. The results reveal the effectiveness of LSTM and self-attention pooling in improving feature extraction, showing the advantages of combining these modules with DARTS-based designs.

1.3 Outlines of Thesis

The structure of this thesis is as follows:

Chapter 1: In this chapter, we introduce the background, challenges and the applications of speaker recognition. We discuss the limitations of traditional CNN-based methods and the motivation for utilizing DARTS to improve speaker recognition performance.

Chapter 2: This chapter reviews related works, including the principles of convolutional neural networks (CNNs), neural architecture search (NAS), and DARTS. It also discusses key components such as self-attention pooling and long short-term memory (LSTM) networks, which are essential for our proposed method.

Chapter 3: In this chapter, we detail our proposed method. We describe the DARTS-based architecture search, the incorporation of self-attention pooling, and the integration of LSTM for enhanced feature representation. The theoretical foundations and mathematical formulation of DARTS and self-attention pooling are also presented in this chapter.

Chapter 4: This chapter focuses on experiments conducted to evaluate the proposed framework. We introduce the datasets, evaluation metrics, and baseline models used for comparison. Results are presented for both speaker identification and verification tasks. Additionally, we perform an ablation study to analyze the contributions of LSTM and self-attention pooling to the overall performance.

Chapter 5: The conclusion summarizes the findings and advantages of the proposed framework. We also discuss potential future directions, such as extending the framework to speaker-conditioned generation tasks and optimizing it for real-time applications.

Chapter 2

Related Works

2.1 Convolutional Neural Network

2.1.1 Overview of Convolutional Neural Network

Convolutional neural networks (CNNs) have become a popular model in various deep learning tasks.Compared with traditional convolution, the traditional fully connected layer or standard convolution of CNN shows higher computational efficiency and generalization ability when dealing with high-dimensional data due to mechanisms such as local receptive fields, weight sharing and hierarchical feature extraction. Local receptive fields enable the model to focus on specific regions of the input, effectively capturing fine details while avoiding redundant information processing. Weight sharing allows the same convolutional kernels to be applied across different spatial locations, significantly reducing the number of parameters and computational complexity, thereby accelerating training and mitigating overfitting. Hierarchical feature extraction leverages stacked convolutional layers to progressively learn more abstract and complex features, from simple edges and textures to intricate patterns and high-level representations.

In the following sections, we will introduce some key components of CNN, including convolutional layers, pooling layers. Additionally, we will introduce two critical optimization techniques for CNN: Separable Convolution and Dilated Convolution.

2.1.2 Convolutional Layer

The process of convolution involves applying a filter, or convolution kernel, to a block of data rather than processing individual elements. This method enables neural networks to capture local patterns and hierarchical structures, allowing for a deeper understanding of the input. By recognizing features such as edges and textures in images, shifts and trends in time-series data, or patterns like n-grams in text, convolution helps the network extract meaningful representations from diverse data sources.

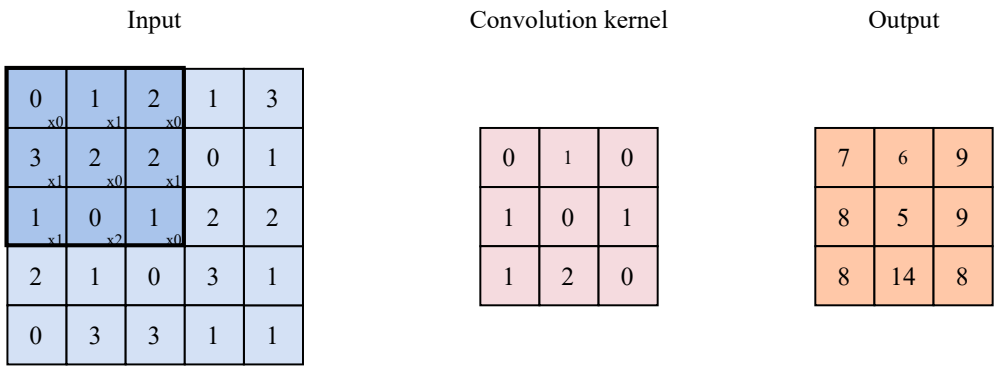

Fig. 2.1. Convolution layer.

As shown in Fig. 2.1, the 3×3 block in the center represents the convolution kernel. This kernel slides across the input data, extracting features and producing an output known as the feature map. In addition to the kernel size, convolution involves two key parameters: stride and padding. Stride defines the step size of the kernel's movement, typically set to 1 or 2, depending on the task. Without padding, the feature map will be smaller than the input data. Padding addresses this by adding zeros around the input, effectively enlarging it and ensuring that the output feature map maintains the same dimensions as the input.

2.1.3 Separable Convolution

Separable convolution is a convolutional operation that decomposes standard convolution into two smaller steps: depthwise convolution and pointwise convolution. This technique significantly reduces the number of parameters and computations required in CNNs while maintaining competitive performance.

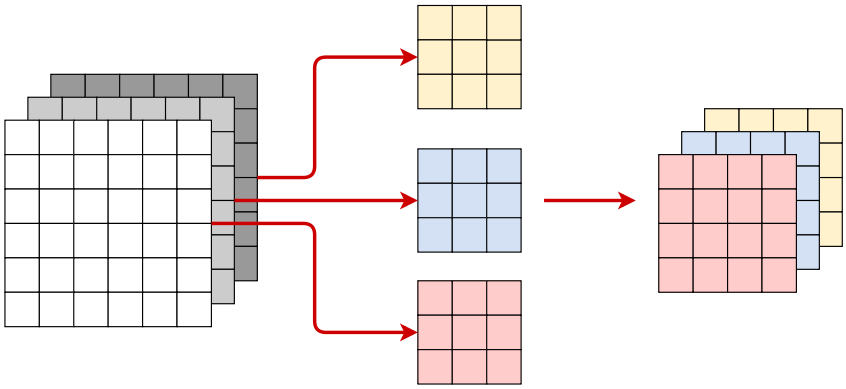

Fig. 2.2. Depthwise convolution.

There are two types of convolution involved in the process. It begins with depthwise convolution, where the convolution operation is applied independently to each channel of the input, using a separate convolution kernel for each channel. As the Fig. 2.2 shows, This step extracts spatial features from each channel without mixing information across channels. Following this, pointwise convolution is applied across all channels as shown in Fig. 2.3, which is actually a 1×1 convolution, combining the spatial features and enabling interaction between channels.

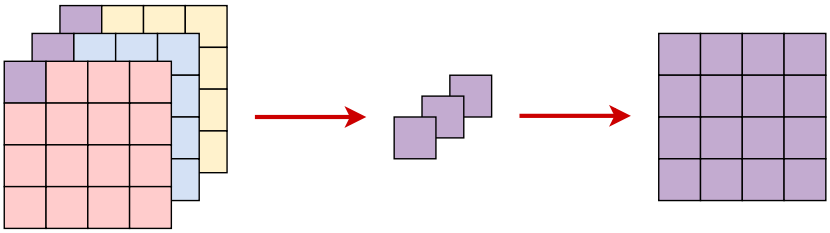

Fig. 2.3. Pointwise convolution.

By separating spatial filtering and channel-wise feature aggregation, separable convolution drastically reduces the computational cost compared to standard convolution, which applies multiple kernels simultaneously to all channels.

2.1.4 Dilated Convolution

Dilated convolution is another convolutional technique that expands the receptive field of a CNN without increasing the number of parameters or computational cost. By inserting spaces between the kernel elements, dilated convolution allows the model to capture multi-

scale contextual information more effectively.

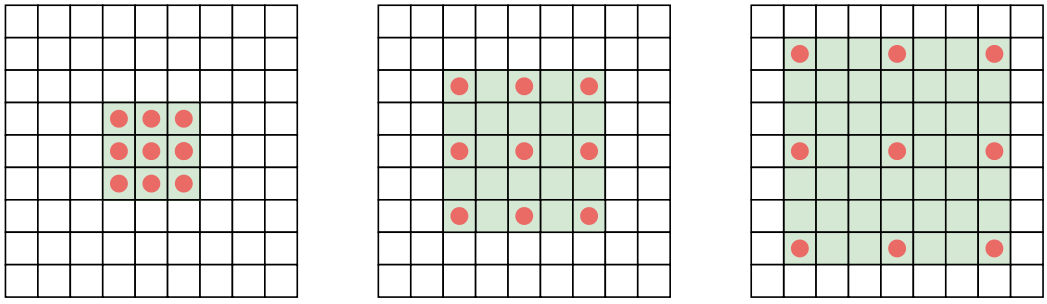

Fig. 2.4. Overview of dilated convolution.

Compared to standard convolution, dilated convolution introduces a hyper-parameter called dilation rate, this hyper-parameter defines the spacing of the values when the convolution kernel processes the input data. The receptive field F can be computed as:

$$F = k + (k-1)(r-1), \tag{2.1}$$

Where k represent the kernel size, r is the dilation rate. Fig. 2.4 emphasizes the effect of different dilation rates on the receptive field:

(a) is the convolution process when the dilation rate is 1. At this dilation rate, the dilation convolution can be treated as an ordinary convolution with a convolved receptive field of 3.

(b) is the convolution process when the dilation rate is 2, the convolved receptive field is 5.

(c) is the convolution process when the dilation rate is 3, the convolved receptive field is 7.

2.1.5 Pooling Layer

Pooling layers are also key components of CNNs which reduce the spatial dimensions of feature maps while remaining the most important information. This downsampling operation helps to decreasing computational complexity and mitigating the risk of overfitting. There are two common pooling operation: max pooling and average pooling.

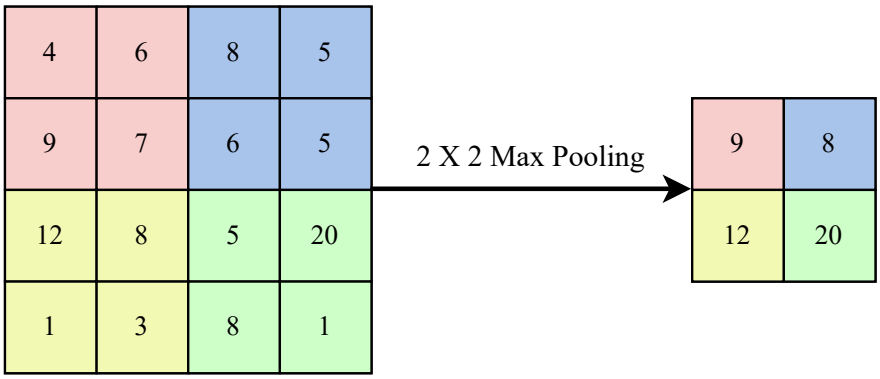

Fig. 2.5. Overview of max pooling operation.

Max Pooling

The max pooling is one of the most common pooling operations, which aims to select the maximum value within a localized region of the feature map to represent the activation intensity of that localized region. As the Fig. 2.5 shows, max pooling slides a fixed receptive field across the feature map according to a predefined window size and stride. It then performs a "take maximum" operation on all elements within that local region, using the resulting maximum value as the output for that region.

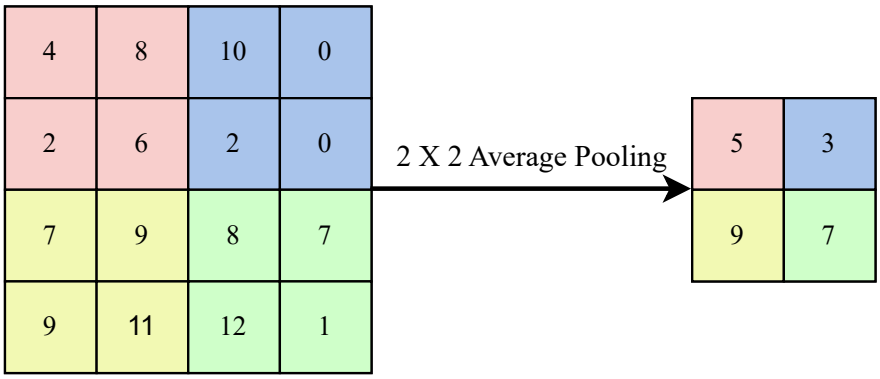

Fig. 2.6. Overview of average pooling operation.

Averge Pooling

average pooling is a commonly used pooling operation that aims to compute the average value within a localized region of the feature map, representing the overall activation

intensity of that region. As shown in Fig. 2.6, average pooling slides a fixed receptive field across the feature map based on a predefined window size and stride. For each local region, it calculates the mean of all elements and uses this average value as the output for that region. This process effectively reduces the spatial resolution of the feature map while retaining the overall distribution of activations, resulting in smoother downsampling.

2.2 Neural Architecture Search (NAS)

Neural Architecture Search (NAS) [8] automates the process of designing neural network architectures by formulating it as a search problem over a defined architecture space.

2.2.1 Early NAS Method

Early NAS method employs a controller network to generate candidate architectures by predicting architecture decisions as a sequence of tokens.

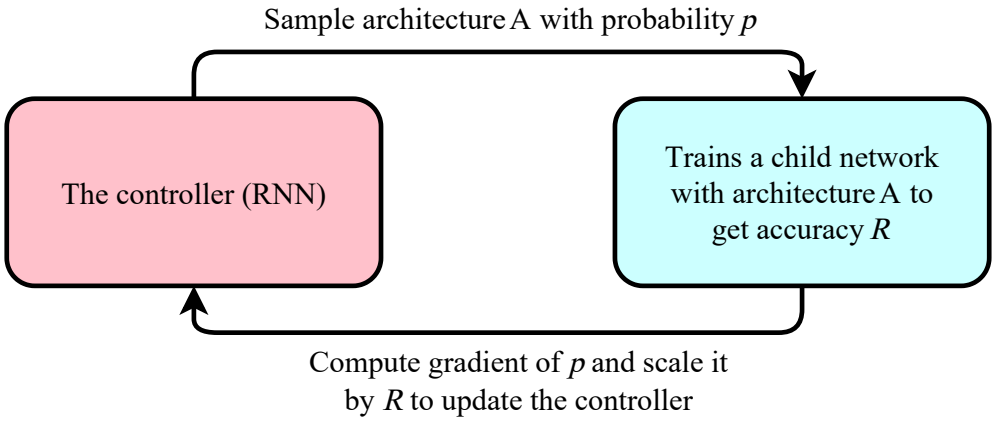

Fig. 2.7. Overview of early NAS method.

As shown in Fig. 2.7, the controller samples an architecture A from the search space with probability p. The sampled architecture is instantiated as a child network and trained to obtain an accuracy R on the validation set. The accuracy R will serve as the reward signal to update the controller, reinforcing the likelihood of generating architecture that yield higher performance in subsequent iterations.

The NAS framework applies the REINFORCE algorithm [11] [12] to update the controller. The policy gradient can be computed as:

$$\nabla_{\theta_c} J(\theta_c) = \sum_{t=1}^{T} E\left[\nabla_{\theta_c} \log P(a_t | a_{1:t-1}; \theta_c) R\right], \tag{2.2}$$

where θ_c denotes the controller's parameters, a_t denotes the action at step t, and R is the reward signal. In order to reduce the variance of the gradient estimate, a baseline b is applied, which is the moving average of previous rewards. The baseline function can be described as:

$$\nabla_{\theta_c} J(\theta_c) = \sum_{t=1}^{T} E\left[\nabla_{\theta_c} \log P(a_t | a_{1:t-1}; \theta_c)(R - b)\right]. \tag{2.3}$$

Although the early NAS method is effective, it requires high computational cost, as each sampled architecture required full training. Subsequent approaches, such as Differentiable Architecture Search (DARTS), addressed this limitation by introducing weight sharing and differentiable search spaces, significantly reducing the search overhead.

2.2.2 Differentiable Architecture Search (DARTS)

Compared to early NAS methods, which rely on reinforcement learing or evolutionary algorithms to explore the architecture space, DARTS [10] introduces a more efficient gradient-based method.

In early NAS method, the searching process involves discrete selection of candidate architectures, requiring the independent training of numerous child networks, which results in significant computational overhead. DARTS effectively reduces computational cost by converting the discrete search space into a continuous differentiable space, using standard gradient descent methods for architecture optimization.

In DARTS, the architecture search process revolves around the concept of continuously relaxing the selection of candidate operations into a weighted sum, which allows smooth transitions between different architectures during the search phase.

Specifically, DARTS models the neural network search space as a directed acyclic graph (DAG), where each node represents a feature representation, and each edge repre-

sents a possible operation. Each edge in DAG will be assigned a weighted combination of the candidate operations. During the training phase, the weights of these operations are optimized through gradient descent, which allows the model to converge to the most effective architecture. This approach avoids the inefficiencies of discrete search, allowing multiple architectures to be evaluated and optimized simultaneously in a single training.

After the search, the operations with the highest weights are selected to obtain the final architecture by discretizing the learned parameters.

The core mathematical formulations and optimization method will be introduced in Chapter 3.

2.2.3 A Pure DARTS-based CNN Framework for Speaker Recognition: AutoSpeech

Network architectures such as VGG-Net and ResNet are widely used as base networks in traditional speaker recognition systems. These architectures were originally designed for image classification and were later applied for speaker recognition tasks. While they perform reasonably well, they are not modified for the unique characteristics of speaker data. This mismatch between image architectures and speech tasks often leads to poor performance as well as excessive model complexity. In addition, the process of manually tuning these architectures for speaker recognition tasks is time-consuming.

AutoSpeech [7] address these limitations to some extent by applying DARTS to the speaker recognition tasks. AutoSpeech converts the architectural search space into a continuous domain and optimizes the architecture using gradient descent, thus avoiding the need to train multiple independent networks as required by early NAS methods. This automated process generates models that are better suited for speaker recognition, thereby improving performance.

AutoSpeech's approach is to search for the optimal combination of operations in the neural cells that form the basic building blocks of the final CNN model. By stacking these optimized cells, AutoSpeech builds a deep network suitable for speaker recognition. In the search phase, AutoSpeech follows the DARTS framework and uses a weighted sum of candidate operations on each edge of the neural units. These weights are dynamically

adjusted by gradient descent to gradually converge the model to the optimal architecture. Once the search is complete, the architecture is discretized by selecting the operations with the highest weights to finalize the network structure.

Compared to VGG and ResNet, AutoSpeech leverages DARTS for task-specific optimization, generating models that are not only more efficient but also achieve higher recognition performance. By focusing on the unique properties of speaker recognition, AutoSpeech is able to reduce unnecessary complexity while enhancing model effectiveness, demonstrating its improved performance in speaker recognition.

2.3 Self-Attention Pooling

2.3.1 Self-Attention Functions

The self-attention mechanism [13] aims to fully capture the long-range dependencies within a sequence and realize efficient parallel computation without relying on loops or convolutional structures. The main idea of Self-attention is to dynamically retrieve contextual information for each element in The input sequence X by measuring its relationship with other elements, thereby producing more semantically expressive representations. The sequence $X \in \mathbb{R}^{n \times d}$ is first projected into three vector spaces: the query vectors, the key vectors and the value vectors. Each of them can be expressed as:

$$Q = XW_Q, \tag{2.4}$$

$$K = XW_K, \tag{2.5}$$

$$V = XW_V, \tag{2.6}$$

where W_Q, $W_K \in \mathbb{R}^{d \times d_k}$, $W_V \in \mathbb{R}^{d \times d_v}$ are learnable parameter matrices. The output for each time instance o_t can be expressed as:

$$o_t = Attn(q_t, K)V, \tag{2.7}$$

where q_t represents the query at the time step t. $Attn$ is the attention function which

is used to compute the attention score. There are two most commonly used attention functions: dot-product attention [14] and additive attention [15].

Dot-Product Attention

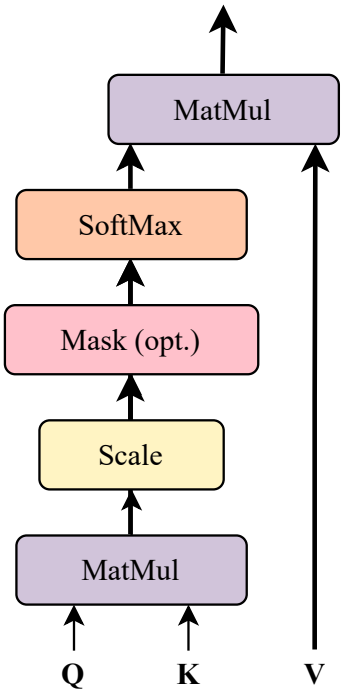

Fig. 2.8. Overview of dot-product attention.

As Fig. 2.8 shows, the dot products of the query with all keys are computed, each result is divided by $\sqrt{d_k}$, and a softmax function is applied to obtain the weights on the values. The output can be expressed as:

$$O = \text{softmax}\Big(\frac{QK^T}{\sqrt{d_k}}\Big)V, \tag{2.8}$$

where d_k is the dimensionality of the key vectors K.

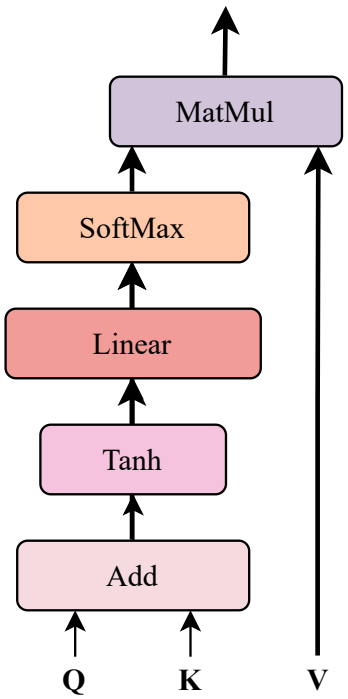

Fig. 2.9. Overview of additive attention.

Additive Attention

As Fig. 2.9 shows, additive attention computes the similarity score between each query and all keys using the formula:

$$E = v^T tanh(W_q Q + W_k K^T), \tag{2.9}$$

where $v \in \mathbb{R}^{d_k}$ is a learnable weight vector, W_q and W_k are learnable matrices which converts query q_t and k_i into the same space for comparison. A softmax function is applied to these scores to obtain the attention weights. The output is computed as the weighted sum of the value vectors. The output can be computed as:

$$O = softmax(E)V. \tag{2.10}$$

2.3.2 Self-Attention Pooling

Self-attention pooling [16] is used to convert frame-level features into fixed-length representations, which is suitable for tasks involving sequential data such as audio processing. Unlike traditional average pooling or maximum pooling, self-attention pooling is able to dynamically adjust its impact based on the importance of each frame. This adaptive mechanism enables the model to retain critical information and thus perform better in downstream tasks.

The main idea of self-attention pooling is to learn the importance of each frame rather than relying on fixed aggregation rules. By assigning different weights to different frames, the model is able to highlight the most informative parts of the input sequence, while less important frames contribute less to the final representation. This process is inspired by the additive attention mechanism, where the importance of each frame is computed from a set of trainable parameters.

In self-attention pooling, the model learns to ”focus” on the most valuable frames by comparing each frame to a trainable reference point. The final output is a weighted combination of all the frames that reflects how well each frames is aligned to that reference point. This approach is conceptually similar to dot product attention, but differs in that self-attention pooling uses a simple trainable vector as the query, rather than extracting the query from the input sequence.

By adopting this approach, self-attentive pooling enhances the model's ability to handle variations in the input sequence, ensuring that the generated representation captures the most critical parts of the data. This property makes self-attentive pooling well-suited for tasks with significant frame-wise variability, as it allows the model to prioritize and integrate the most informative parts of the sequence effectively.

2.4 Long Short-Term Memory Network

Long short-term memory network (LSTM) [17] is a special recurrent neural networks (RNN) which is designed to solve the vanishing or gradient exploding that occurs when traditional RNNs process long sequence data. When dealing with long sequence data, traditional RNNs usually have difficulty in retaining early information in the model due to

gradient vanishing or gradient explosion during backpropagation, resulting in poor performance in drawing long-term information. LSTM introduces a “gating mechanis” that effectively controls the storage and forgetting of information, allowing the model to capture dependencies over longer time spans while preventing gradient-related issues.

As usual for RNNs, the LSTM consists of repeating neural network modules. However, the repeating modules of LSTM have a different structure from RNNs.

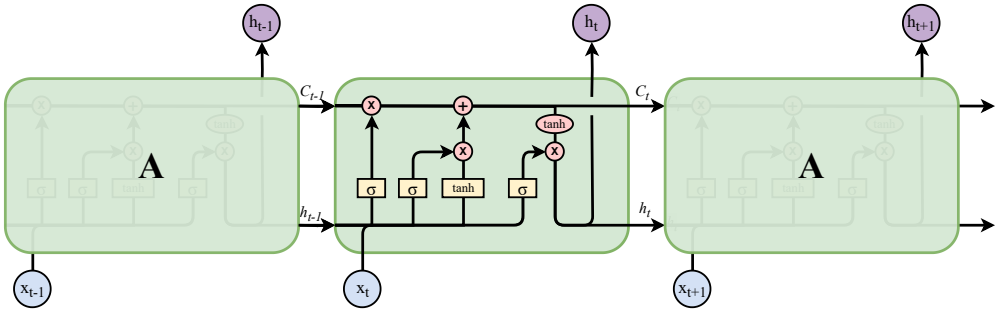

Fig. 2.10. Structure of LSTM.

Fig. 2.10 shows the chain structure of LSTM and the composition of a repeating module. The key to LSTM is the cell state, the horizontal line across the top of the graph. The cell state is similar to a conveyor belt. It runs directly along the entire chain with only a few small linear interactions. Information can easily flow on unchanged. In addition to this, each module uses three gates to determine the information that needs to be retained:

1. Forget Gate. The forget gate performs the selection of the last cell state C_{t-1}, which determines how much of the previous cell state C_{t-1} will be retained for the current moment C_t. The output f_t can be expressed as:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \tag{2.11}$$

where σ is the sigmoid activation function, which compress the output to a range of $[0, 1]$ to control the degree of forgetting. W_f represents the weight matrix of the forget gate, which learns the influence of the input and hidden state on forgetting. h_{t-1} is the hidden state from the previous time step. b_f is the bias term for the forget gate, which adjusts the output of the forget gate.

2. Input Gate. The input gate determines how much information of the input x_t will be saved to the cell state C_t. The output i_t can be expressed as:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \tag{2.12}$$

where W_i represents the weight matrix of the input gate, which controls how the input influences the cell state update. b_i is the bias term for the input gate. The candidate cell state can be expressed as:

$$\widetilde{C}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_C), \tag{2.13}$$

where W_c represents the weight matrix for the candidate cell state. b_C is the bias term for the candidate cell state. The cell state can be updated as:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \widetilde{C}_t. \tag{2.14}$$

3. Output Gate. The fianl output and the hidden state can be computed as:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \tag{2.15}$$

$$h_t = o_t \cdot tanh(C_t), \tag{2.16}$$

where W_o represents the weight matrix of the output gate and b_o is the bias term for the output gate.

LSTM is widely used in sequential data tasks, such as speech recognition, machine translation, and stock price prediction. CNNs are effective for extracting spatial features from spectrograms or mel-frequency cepstral coefficients (MFCCs), while LSTMs are adept at capturing temporal dependencies across frames. This combination leverages the strengths of both architectures, enabling accurate modeling of speaker characteristics and temporal patterns, which is crucial for speaker-related applications [18].

Chapter 3

Proposed Method

3.1 Overview of Proposed method

In this chapter, we mainly discuss the proposed method along with its architecture, the principles behind DARTS, and the working mechanism of self-attention pooling.

As Fig. 3.1 shows, the workflow of the proposed method can be divided into three key steps: DARTS Architecture Search, Replacing Max Pooling with self-attention Pooling, and Integrating LSTM in the Training Phase.

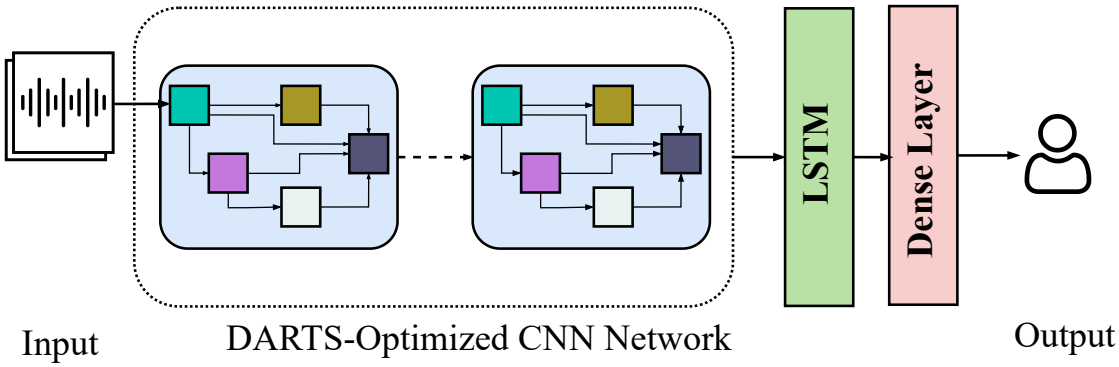

Fig. 3.1. Structure of proposed method.

First, we employ DARTS to identify the optimal neural network architecture. DARTS transforms the architecture search problem into a differentiable optimization task by relaxing the search space, represented as a weighted mixture whose weights are optimized by gradient descent. During the search process, these weights are gradually adjusted to determine the most efficient operations and connections. After a certain number of itera-

tions, the operation with the highest weight on each edge is selected, resulting in a discrete optimal architecture.

layers in the network with self-attention pooling. Directly incorporating self-attention pooling into the candidate operations during the DARTS search phase significantly increases computational overhead due to the resource-intensive nature of the attention mechanism for weight computation. By contrast, our approach defers the integration of self-attention pooling until after the architecture is finalized, which substantially improves training efficiency. This design saves search time while leveraging the ability of self-attention pooling to capture long-term dependencies, effectively balancing computational cost and model performance.

Finally, in the training phase, we integrate an LSTM network after the DARTS optimized CNN network.The CNN is first trained to extract local and spatial features from the input, and the resulting feature maps are subsequently fed into an LSTM. The LSTM is implemented with a single-layer structure to avoid overfitting, and its hidden state size is set to be dimensionally identical to the CNN output. At this stage, the sequence of feature maps generated by the CNN is used as input to the LSTM, which processes the data in sequence to capture long time dependencies. In order to address the instability that may result from training on long sequences, a gradient cropping technique is introduced at this stage.

3.2 Derivation of the DARTS Procedure

In our method, we employ DARTS [10] to automatically search an optimal CNN architecture for speaker recognition tasks. The network backbone is constructed by stacking neural cells, where each cell is designed as a DAG. Nodes in the graph represent feature representations, while edges denote candidate operations such as convolutions, pooling, or skip connections.

Fig. 3.2 shows the DARTS search steps as follows:

3.2.1 Search Space Definition

The search space $\mathcal{O}$ contains 8 candidate operations:

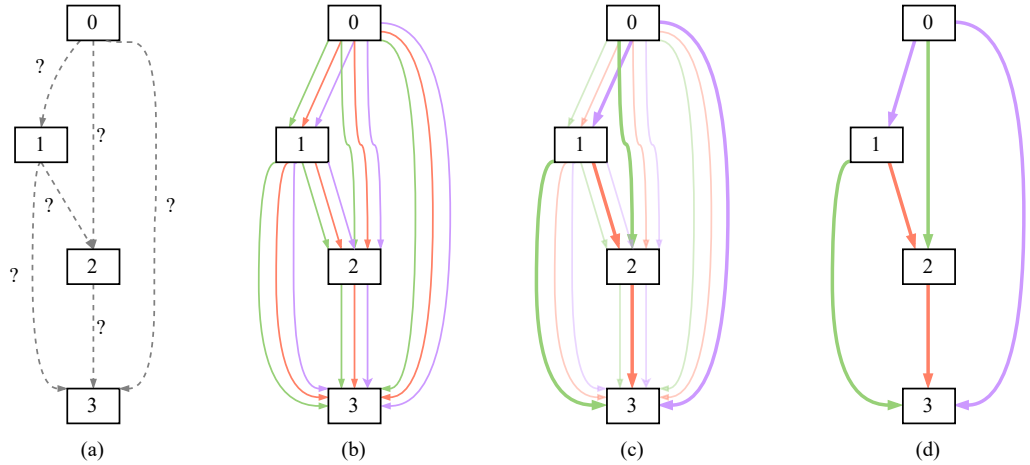

Fig. 3.2. Overview of DARTS search process: (a) Operations on the edges are unknown, (b) Continuous relaxation of the search space, (c) Bi-level optimization, (d) Final architecture.

- 3×3 separable convolution
- 5×5 separable convolution
- 3×3 dilated convolution
- 5×5 dilated convolution
- 3×3 average pooling
- 3×3 max pooling
- skip connection
- no connection (zero)

To balance the feature resolution and computational efficiency, two types of cells are defined as:

- **Normal cells:** Maintain spatial resolution and compute intermediate feature representations.
- **Reduction cells:** Reduce spatial resolution by a factor of two and double the number of channels.

Each cell is defined as having 7 nodes, with 2 input nodes and 4 intermediate nodes, and 1 output cell.

The placement of reduction cells follows a consistent strategy, located at $1/3$ and $2/3$ of the total depth of the stacked cells [10] [19]. Each type of cell shares its architecture across the entire network to simplify design complexity.

3.2.2 Relaxation of the search space

In early NAS method, the architecture is defined by discrete choices, such as selecting specific operations or connection paths. This makes the optimization process non-differentiable, requiring techniques like reinforcement learning or evolutionary algorithms.

Instead of selecting discrete operations for each edge, DARTS assigns a weighted combination of all candidate operations:

$$\bar{o}^{(i,j)}(x) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})} \cdot o(x), \tag{3.1}$$

where $\alpha_o^{(i,j)}$ is a learnable parameter associated with operation o on edge (i, j). This relaxation enables the architecture search to be formulated as a differentiable optimization task.

The feature at node j is then calculated as the summation over all its predecessors:

$$x^{(j)} = \sum_{i<j} o^{(i,j)}(x^{(i)}). \tag{3.2}$$

3.2.3 Bi-level Optimization

The search process involves two sets of parameters:

- **Network weights ω:** Parameters of the candidate operations.
- **Architecture parameters α:** Weights determining the importance of each operation.

The bi-level optimization problems [20] can be formulated as:

$$\begin{aligned} \min_{\alpha} \quad & \mathcal{L}_{\text{val}}(w^*(\alpha), \alpha) \\ s.t. \quad & w^*(\alpha) = \arg\min_{w} \mathcal{L}_{\text{train}}(w, \alpha), \end{aligned} \tag{3.3}$$

where $\mathcal{L}_{\text{train}}$ and $\mathcal{L}_{\text{val}}$ represent the training and validation losses, respectively. Both $\mathcal{L}_{\text{train}}$ and $\mathcal{L}_{\text{val}}$ are defined as cross-entropy losses:

$$L := -\sum_{k=1}^{K} \mathbb{I}(y = k) \log p_k, \tag{3.4}$$

where K is the number of speakers, y represents the ground-truth speaker label and p_k is the softmax probability of speaker k.

To reduce computational costs, we employ a one-step gradient approximation:

$$\begin{aligned} &\nabla_\alpha \mathcal{L}_{\text{val}}(w^*(\alpha), \alpha) \\ \approx\ &\nabla_\alpha \mathcal{L}_{\text{val}}(w - \xi \nabla_w \mathcal{L}_{\text{train}}(w, \alpha), \alpha). \end{aligned} \tag{3.5}$$

In this approximation format, ω is updated for a single step with learning rate ξ to approximate the solution.

3.2.4 Architecture Derivation

After the search phase, DARTS converts the continuous architecture into a discrete one by selecting the operation with the highest weight for each edge:

$$p_o^{(i,j)} = \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})}. \tag{3.6}$$

The derived architecture is then stacked to form the CNN backbone, with reduction and normal cells arranged according to the predefined placement.

To determine the convergence of the architecture search process, we evaluate the entropy of the architecture parameters α. The entropy is computed as:

$$E = \sum_{(i,j)} \sum_{o \in O} \alpha_o^{(i,j)} \log \alpha_o^{(i,j)}, \tag{3.7}$$

where α_{ij}^o represents the weight of operation o on edge (i, j) in the search space $\mathcal{O}$. A smaller entropy value indicates a higher confidence in selecting a specific operation among all possible candidates for each edge. The architecture search is considered converged when the entropy stabilizes and no longer decreases.

3.3 Derivation Self-Attention Pooling

Self-attention pooling [16] is an effective mechanism for aggregating frame-level features into a fixed-length utterance-level representation. Traditional pooling methods, such as average and max pooling, often fail to dynamically adjust to the varying importance of different frames in the input sequence. Self-attention pooling addresses this limitation by learning the importance of each frame, thereby retaining more informative representations for downstream tasks.

The main idea of self-attention pooling is based on additive attention, which computes the importance of each frame by training a set of trainable weights. Given an input sequence of frame-level features, which can be expressed as:

$$H = [h_1, h_2, ..., h_T]^T \in \mathbb{R}^{T \times d_m}, \tag{3.8}$$

where T denotes the number of time steps and d_m is the feature dimension, the segment-level representation is computed as:

$$C = Softmax(W_c H^T)H, \tag{3.9}$$

where $W_c \in \mathbb{R}^{d_m}$ is a learnable parameter representing the weight vector applied to each frame, H_T denotes the transposed feature matrix, which enables the attention mechanism to compute the importance score for each time step.

In particular, in self-attention pooling, the keys and values are from the same feature sequence H, the query is represented by a learnable vector W_c.

Conceptually, this process can be treat as dot-product attention, where the query, key, and value interact to compute attention scores. However, in self-attention pooling, the query is simplified to a learnable vector rather than being derived from the input sequence. Thus, the output representation C is a weighted average of the output sequence, with the weights reflecting the alignment learned by the self-attention mechanism.

Chapter 4

Experiment

4.1 Datasets

We evaluated our method on the VoxCeleb1 [5] and VoxCeleb2 [6] datasets, which consisted of large-scale collections of real-world speech data, providing a challenging and diverse environment for training and evaluating speaker-related models.

VoxCeleb1 contained voice recordings of 1,251 speakers, totaling more than 140,000 utterances from YouTube. The data was derived from online interviews involving speakers of different nationalities, occupations, and genders.

VoxCeleb2 provided a larger dataset containing 6,112 speakers and more than 1 million utterances, which differed from VoxCeleb1.

We evaluated both speaker identification and verification on VoxCeleb1 and speaker verification on VoxCeleb2..

4.2 Evaluation Metrics

For speaker recognition, we used top-1 accuracy and top-5 accuracy as evaluation metrics. Top-1 accuracy measured the proportion of test samples for which the model's highest confidence prediction agreed with the true label. This metric reflected the model's ability to correctly recognize a speaker in a single attempt and served as a straightforward and rigorous way of evaluating performance. The top-5 accuracy rate, on the other hand, treated a prediction as correct if the true label appeared in the top 5 predictions of the

model with the highest confidence.

For speaker verification, we used the equal error rate (EER) as the primary evaluation metric. EER was the point at which the false acceptance rate (FAR) equaled the false rejection rate (FRR). The FAR measured the proportion of impostors incorrectly accepted as the claimed speaker, while the FRR measured the proportion of genuine speakers incorrectly rejected. A lower EER indicated a better trade-off between these two error types, signifying a more balanced and robust verification system.

4.3 Experimental Result

To evaluate the effectiveness of our proposed model, we conducted a series of experiments to compare its performance against ResNet-34 [21] and AutoSpeech. Specifically, we evaluated our method and AutoSpeech under the same settings, with the number of cells set to 8 and the initial channel size set to 128. For speaker identification, the models were trained on the identification split of VoxCeleb1, and the testing set was used to evaluate identification accuracy. We reported both top-1 and top-5 accuracies as the primary evaluation metrics. For speaker verification, the models were trained on the verification splits of VoxCeleb1 and VoxCeleb2. The EER was employed as the key metric for verification tasks, which quantified the trade-off between false acceptance and false rejection.

Table 4.1 Comparative experimental results with ResNet and AutoSpeech

	VoxCeleb1			VoxCeleb2
Method	Top-1(%)	Top-5(%)	EER(%)	EER(%)
ResNet-34	81.37	94.49	11.53	5.10
AutoSpeech	87.57	95.98	8.96	4.32
Proposed	**88.13**	**96.71**	**8.91**	**4.24**

As Table 4.1 shows, for speaker identification, our model achieved a top-1 accuracy of 88.13% and a top-5 accuracy of 96.71%, significantly outperforming ResNet-34 and AutoSpeech. Specifically, compared to ResNet-34, the top-1 and top-5 accuracies increased by 6.76% and 2.22%, respectively, while compared to AutoSpeech, the improvements

were 0.56% and 0.73%. For speaker verification, our model achieved an EER of 8.91% on VoxCeleb1 and 4.24% on VoxCeleb2. These values represented reductions in error rates compared to ResNet-34 and AutoSpeech. On VoxCeleb2, the EER was reduced by 0.86% compared to AutoSpeech. These results validated the effectiveness of our proposed method, demonstrating its superior ability to capture and represent speaker-specific features.

4.4 Ablation Study

To better validate and analyze the effectiveness of each individual component in our model, we conducted an ablation study focusing on the LSTM module and the self-attention pooling mechanism. We designed two model variations:

1. Proposed w/o LSTM: In this configuration, the LSTM module was removed from the proposed architecture. After the convolutional backbone, a simple fully connected layer was used to aggregate the extracted features. This modification isolated the influence of the LSTM in modeling temporal dependencies.

2. Proposed w/o self-attention pooling: In this configuration, the self-attention pooling mechanism was replaced with max pooling. The rest of the architecture, including the LSTM module, remained intact. This setup isolated the effectiveness of self-attention pooling in capturing global feature dependencies.

All other hyperparameters and training settings were kept the same for a fair comparison.

Table 4.2 Experimental result for ablation study

Method	Top-1(%)	Top-5(%)	EER(%)
Proposed w/o LSTM	88.04	96.42	8.93
Proposed w/o self-attention pooling	88.09	96.39	8.95
Proposed	**88.13**	**96.71**	**8.91**

We evaluated the performance of both variants on VoxCeleb1 in the speaker identification and verification tasks. The results of the ablation study are presented in Table 4.2.

The ablation study results are summarized as follows:

1. Removing the LSTM module caused a slight decline in both Top-1 and Top-5 accuracies for speaker identification, dropping from 88.13% to 88.04% and from 96.71% to 96.42%, respectively. This showed the importance of LSTM in modeling temporal dependencies inherent in sequential speech data.

2. Replacing self-attention pooling with max pooling also resulted in a slight decrease in speaker identification performance. The Top-1 accuracy dropped from 88.13% to 88.09%, and the Top-5 accuracy decreased from 96.71% to 96.39%. This demonstrated the role of self-attention pooling in capturing global speaker-specific dependencies, which were critical for distinguishing speakers in identification tasks.

3. Replacing self-attention pooling with max pooling further caused a marginally higher EER on VoxCeleb1, increasing from 8.91% to 8.95%. This indicated that self-attention pooling improved the representation of speaker-specific features by capturing global dependencies, which helped reduce errors in speaker verification tasks.

4. The impact of removing the LSTM module was most noticeable in the Top-5 accuracy, which decreased by 0.29%. This suggested that the LSTM excelled at capturing fine-grained information within the sequence, which was critical for improving the model's performance in complex identification tasks. On the other hand, replacing the self-attention Pooling with Max Pooling had a more significant effect on both the EER, which increased by 0.04%, and the Top-5 accuracy, which dropped by 0.32%. This indicated that Self-Attention Pooling played a crucial role in modeling global features, significantly enhancing the overall discriminative power of the model.

Chapter 5

Conclusion and Future Works

5.1 Conclusion

Our work presents a refined approach to speaker recognition by integrating a DARTS-optimized CNN architecture with self-attention pooling and LSTM modules. Compared to traditional CNN methods and purely DARTS-optimized architectures, the proposed framework effectively enhances speaker recognition performance by leveraging both temporal and global dependencies in speech data. By combining neural architecture search with task-specific modules, the proposed framework addresses limitations in feature representation, resulting in improved flexibility and accuracy in both speaker identification and verification tasks.

The results of the ablation study demonstrate the unique contributions of LSTM and self-attention pooling to the proposed architecture. LSTM enhances the model's ability to capture temporal dependencies, which is essential for effectively processing sequential speech signals. Self-attention pooling, on the other hand, refines the representation of global dependencies across the feature space, enabling the model to focus on critical speaker-specific features. Collectively, these components enable the model to focus on sequential and global aspects of feature extraction, achieving superior performance over baseline architectures.

Overall, our method shows the effectiveness of combining neural architecture search with advanced feature extraction modules in addressing the challenges of speaker recognition. The proposed framework benefits from the flexibility and adaptability of DARTS-

optimized architectures while leveraging the Combined advantages of LSTM and self-attention pooling to enhance feature representation. This integration improves model performance, demonstrating the effectiveness of hybrid architectures in advancing speaker recognition systems.

5.2 Future Work

While the proposed framework achieves promising results, several directions remain open for future exploration. The computational complexity introduced by the self-attention pooling mechanism, although beneficial for capturing global dependencies, increases the overall model size and may pose challenges in real-time or resource-constrained applications. Developing lightweight alternatives or approximation methods to reduce this overhead will be critical for broadening the framework's applicability.

Furthermore, extending the framework to speaker-conditioned generation tasks (e.g., speech cloning and speech synthesis) is a direction worth exploring. Such tasks require high accuracy in speaker-specific representations to ensure that the generated speech achieves a good level of sound quality and naturalness. Future research could consider combining multi-task learning or contrastive learning frameworks to enhance the model's adaptability in multi-task scenarios by simultaneously optimizing the recognition and generation objectives. This direction is expected to further enhance the unified modeling of speaker features and promote the synergistic development of speaker recognition and generation tasks, thus offering the possibility of developing more comprehensive speaker-related systems.

List of Publication

- Minghao Duan, and Hiroshi Watanabe: “A Refined DARTS-based Method for Speaker Recognition,” IEICE General Conference, Mar. 2025 (to appear)

Bibliography

[1] John H. L. Hansen and Taufiq Hasan. Speaker recognition by machines and humans: A tutorial review. *IEEE Signal Processing Magazine*, 32(6):74–99, Nov. 2015.

[2] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. Sep. 2014.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. Dec. 2015.

[4] Maros Jakubec, Eva Lieskovska, and Roman Jarina. Speaker recognition with resnet and vgg networks. In *Proc. 2021 31st International Conference Radioelektronika (RADIOELEKTRONIKA)*, pages 1–5, Apr. 2021.

[5] Arsha Nagrani, Joon Son Chung, and Andrew Zisserman. Voxceleb: a large-scale speaker identification dataset. In *Proc. INTERSPEECH*, Sep. 2017.

[6] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman. Voxceleb2: Deep speaker recognition. In *Proc. INTERSPEECH*, Sep. 2018.

[7] Shaojin Ding, Tianlong Chen, Xinyu Gong, Weiwei Zha, and Zhangyang Wang. AutoSpeech: Neural Architecture Search for Speaker Recognition. In *Proc. Interspeech 2020*, pages 916–920, Oct 2020.

[8] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, Jan. 2019.

[9] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. arXiv:1802.01548, Feb. 2019.

[10] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. Jun. 2018.

[11] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. arXiv:1611.01578, Nov. 2017.

[12] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, Nov. 1992.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, Dec. 2017.

[14] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. arXiv:1508.04025, Aug. 2015.

[15] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. arXiv:1409.0473, Sep. 2016.

[16] Pooyan Safari, Miquel India, and Javier Hernando. Self-attention encoding and pooling for speaker recognition. arXiv:2008.01077, Aug. 2020.

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, Nov. 1997.

[18] Tara N. Sainath, Oriol Vinyals, Andrew Senior, and Haşim Sak. Convolutional, long short-term memory, fully connected deep neural networks. In *Proc. 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4580–4584, Apr. 2015.

[19] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. arXiv:1707.07012, Jul. 2018.

[20] Benoît Colson, Patrice Marcotte, and Gilles Savard. An overview of bilevel optimization. *Annals of Operations Research*, 153:235–256, Apr. 2007.

[21] Weidi Xie, Arsha Nagrani, Joon Son Chung, and Andrew Zisserman. Utterance-level aggregation for speaker recognition in the wild. arXiv:1902.10107, Feb. 2019.