TUNIS BUSINESS SCHOOL
UNIVERSITY OF TUNIS

Personal Research Project

Speaker Recognition and ASR

Speaker Recognition and Automatic Speech Recognition

Sinene Fradj

A personal project exploring methods and techniques in
Speaker Recognition and Automatic Speech Recognition

March 26, 2025

Abstract

As artificial intelligence (AI) continues to evolve, its potential to transform everyday experiences becomes increasingly clear. This project was born out of a personal desire to explore how AI can be used to address a meaningful challenge—recognizing the reciter of a Quranic verse and identifying the exact verse being recited. This task, while seemingly straightforward, holds deep significance for many who seek to engage with the Quran in a more accessible and efficient way.

The project tackles the problem of speaker recognition and automatic speech recognition (ASR) in Quranic recitations. Initially, the speaker recognition task was approached using MatchboxNet, but it soon evolved to a more sophisticated Xembedding FTDNN architecture, enhanced with a NetVLAD pooling layer for improved accuracy. For ASR, the pretrained Whistle model was employed alongside RapidFuzz to search for the closest match within a vast collection of Quranic texts.

The system achieved a speaker recognition accuracy of 78%. While this result is promising, it also unveiled several challenges that will be explored in detail throughout the paper. These include improving recognition accuracy across diverse reciters and ensuring precise verse alignment. The findings provide valuable insights into the intersection of AI and Quranic studies, offering a glimpse into the future of AI applications in deeply personal contexts.

Contents

1 Introduction

Automatic speaker recognition and automatic speech recognition (ASR) have been extensively studied in recent years, with advancements driven by deep learning techniques and the increasing availability of large-scale datasets. Speaker recognition aims to determine the identity of a speaker from an audio recording, while ASR focuses on transcribing spoken language into text. These technologies have found applications in security, virtual assistants, and multimedia indexing. However, their application to religious recitations, such as the Quran, presents unique challenges due to variations in recitation styles, speaker accents, background noise and recording sound quality .

In this project, we focus on the task of identifying the reciter of a Quranic recitation and determining the corresponding verse and chapter. Given an audio recording of Quran recitation, our system performs two tasks: (1) speaker recognition to predict the identity of the reciter, and (2) ASR to transcribe the audio and match it to the closest Quranic verse. While previous research has explored speaker recognition and ASR separately, few studies have focused on integrating them for religious recitations, where accuracy is critical.

Our approach initially involved using MatchboxNet for speaker recognition at the outset our choice of this architecture was arbitrary ,I read about this architecture while competing in a zindi hackathon and wanted to explore its an unfamiliar mechanisms ,I write an article explaining matchoxNet deeply [5]. However, after further experimentation, we adopted an X-embedding Factorized TDNN (FTDNN) architecture with a NetVLAD pooling layer, which provided better performance since it is a text-independent architecture. For ASR, we used the Whisper pretrained model due to its robustness across different languages and accents. To match the transcribed text with the Quran, we leveraged RapidFuzz, a fuzzy string-matching library, to find the closest verse even when transcription errors occur.

Our experimental results show that the speaker recognition model achieves 78% accuracy, and we discuss the challenges faced throughout the project, including dataset limitations, variability in recitation styles, and noise interference.

The remainder of this paper is structured as follows: chapter 1 describes the Speaker Recognition System .first section will tackle the data sources and the pre-processing steps Section 2 explains the features extraction process used for speaker recognition and finally we explained the architecture layers and the results we got. chapter 3 will discusses the ASR and the matching strategy and evaluation metrics. Finally, chapter 4 concludes the paper and suggests future improvements.

2 Speaker Recognition System

Neural Network embeddings (a.k.a x-vectors,i-vectors) are obtained using a neural network trained to classify the speakers in the training set .x-vector architectures are divided into 3 main parts the backbone for deep speaker feature extraction in general we use even CNN or LSTM or a hybrid model of both layers this part extracts frame level representations from the acoustic features this is followed by a temporal pooling layer to aggregate our features into X-vectors and finally a feed forward classification network to produce speaker class posteriors [2]

2.1 Datasets

2.1.1 Training Data

For training data, I constructed a JSON file containing recitations from 17 distinct Quranic reciters. Each reciter's entry includes 4–8 YouTube links of their recitations, along with the corresponding chapter (Surah) names. I developed a Python program to interact with the YouTube API and download videos in .wav format. This was followed by another program that preprocesses the audio by trimming silence and splitting it into smaller segments.

For the initial training phase, each audio was divided into 50 segments, with each segment lasting 5 seconds, resulting in a dataset of 6,651 audio clips and a total of 92 hours of training data. However, empirical analysis revealed that due to the nature of Quranic recitation—particularly "Madd" (the prolongation of words)—the 5-second segments were not sufficiently informative.

To address this, I expanded the dataset by increasing the segment length to 10 seconds and the number of segments per audio to 75, resulting in 11,456 audio clips and a total of 318 hours of training data. This adjustment provided more context within each segment, improving the model's ability to learn meaningful patterns from the recitations

for data augmentation ,I applied room impulse response (RIR) and added Background noise .

2.1.2 Evaluation Data

For the evaluation dataset, I followed the same preprocessing steps as in training, using an unseen audio source.

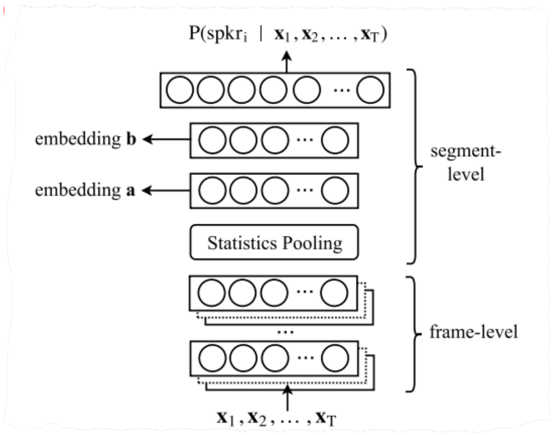
One key insight I gained after making predictions with my trained model was that the evaluation data was not optimally chosen. Since I split the raw audios into sequential segments, consecutive segments were highly similar. As a result, when the model misclassified one segment, it often misclassified the next ones as well.

I believe that improving this segmentation strategy—both in the training and evaluation datasets—would help increase the model's accuracy by ensuring more diverse and representative samples.

2.2 Feature Extraction

For feature extraction, I considered three options: using 23 MFCCs for 8 kHz audio, or 30 (HLT-COE) and 40 MFCCs (CLSP-MIT) for 16 kHz audio. After experimenting with different architectures, I observed that the architecture incorporating a NetVLAD pooling layer performed best with CLSP-MIT features, while the one utilizing statistical pooling was more effective with HLTCOE features. A detailed discussion of NetVLAD and statistical pooling follows in Section 2.3.2.1.
FYI: the mel-frequency cepstrum (MFCC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency. Mel-frequency cepstral coefficients are coefficients that collectively make up an MFC..

2.3 Segment-level embedding — x-vector



P(spkr$_i$ | $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T$)
embedding **b**
embedding **a**
Statistics Pooling
segment-level
frame-level
$\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T$

Figure 1: Segment-level embedding Architecture

2.3.1 Frame level:BackBone

The backbone is responsible for extracting key features from the input data, such as images, audio signals, or other structured inputs. Once these features are extracted, they can be further processed or enhanced by adding additional layers or modules

2.3.1.1 TDNN

Time-Delay Neural Networks (TDNNs) are a type of deep learning architecture particularly well-suited for processing sequential data like audio signals. They have been successfully applied in speaker identification tasks, especially as part of X-vector systems.

2.3.1.2 FTDNN with Skip connection

The factorized TDNN (F-TDNN) , reduces the number of parameters of the network by factorizing the weight matrix of each TDNN layer into the product of two low-rank matrices. The first of those factors is constrained to be semi-orthogonal [3], which helps to assure that we do not lose information when projecting from the high dimension to the low-rank dimension. Enforcing a parameter matrix to be semi-orthogonal during training involves a periodic adjustment to the matrix to maintain its semi-orthogonal property while allowing it to be optimized via standard gradient descent (SGD) [4]. I use 3 stage sliceing :3-stage splicing it is to have a layer with a constrained 2x1 convolution to dimension 256, followed by another constrained 2x1 convolution to dimension 256, followed by a 2x1 convolution back to the hidden-layer dimension (e.g. 1280).The dimension now goes from, for example, 1280 $\rightarrow$ 256 $\rightarrow$256 $\rightarrow$ 1280, within one layer. The effective temporal contextof this setup is of course wider than the TDNN baseline, due to the extra 2x1 convolution To enhance model generalization, I implemented a dimension-wise continuous scaled dropout layer, where dropout rates remain constant along selected feature dimensions during training

2.3.2 Utterance level:Temporal Pooling Layer

Temporal pooling represents the transition layer of a neural network that transforms frame-level embedding features to utterance-level embedding features Temporal pooling is applied to aggregate the frame-level features (hth) across the entire duration of the input speech segment (T) into a single representation (u).proposed a vector-based attentive pooling method, which adopts vectorial attention weights for each frame-level vector. I experienced with 2 different options:

2.3.2.1 statistical pooling

computes both first-order (mean) and second-order (standard deviation) moments of input features across a specified dimension, producing a fixed-length representation that encapsulates the distributional characteristics of the data.

2.3.2.2 NetVLAD Layer

NetVLAD is an aggregation layer that takes a set of local descriptors (extracted, for example, from convolutional feature maps) and encodes them into a single, fixed-length representation suitable for tasks like image retrieval or Speaker recognition. A key idea in NetVLAD is to compute a weighted sum of the "residuals" (the differences between the descriptors and some learned cluster centers). Instead of assigning each descriptor to a single cluster (hard assignment), NetVLAD uses a soft assignment. This "hard" assignment is non-differentiable, meaning you cannot compute gradients with respect to the cluster centers or other parameters during backpropagation

By instead assigning descriptors to clusters in a "soft" manner (i.e., each descriptor contributes to multiple clusters with different weights), the assignment becomes a smooth, differentiable function. This allows the entire network—including the NetVLAD layer—to be trained end-to-end using gradient descent.

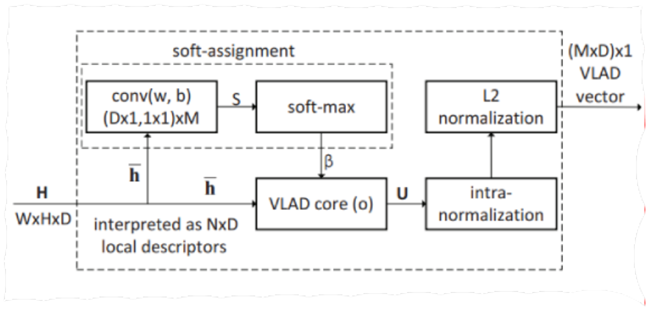


Figure 2: Diagram of the NetVLAD pooling layer [1]

2.3.3 Objective Function

Objective functions affect the effectiveness of speaker recognition much. x-vector adopt softmax as the output layer and take the cross-entropy minimization as the objective function, which may not be optimal. Recently, many objectives were designed to further improve the performance.I chose to use an Additive angular margin softmax (AAMSoftmax).This is myfinal classification layer that computes logits with an angular margin.

- It takes the 512-dimensional speaker embedding and calculates logits by applying an angular margin constraint and a scaling factor.
- The logits represent the speaker similarity scores before applying softmax

2.3.4 Results and Discussion

As shown in Table 1 Our experiments demonstrate that architectural choices and data configuration significantly impact speaker recognition performance. The statistical pooling variant achieved 69.3% accuracy with extended 10-second segments (V4), outperforming shorter 5-second inputs (62.3% for V1), confirming that prolonged phonetic units (Madd) in Quranic recitation require longer temporal contexts for robust feature extraction. Conversely, NetVLAD showed superior scalability, with its best configuration (10-second augmented data) achieving 78.3% accuracy—a 9% absolute improvement over statistical pooling under equivalent conditions. This suggests that learnable feature aggregation (NetVLAD's cluster-based pooling) better captures speaker identity in variable-length recitations compared to fixed statistical moments. Notably, increasing MFCCs to 40 degraded performance in both architectures (44.2% and 58.3%), indicating potential over-parameterization or noise amplification. The results collectively highlight: (1) input length as critical for Quranic speaker recognition due to Madd-dominated prosody, and (2) dynamic pooling (NetVLAD) as more adaptable to sacred recitation patterns than static statistical summaries. Future work could explore hybrid architectures or domain-specific cluster initialization for further gains. While the current x-embedding architecture achieves promising accuracy (78.32% with NetVLAD), its 826mb size presents challenges for real-world mobile deployment. To bridge this gap, several optimizations could be implemented: First, INT8 quantization could reduce the model size by 75% (to 207.25MB) with minimal accuracy loss, as demonstrated in similar speaker recognition tasks. Second, structured pruning could be applied to the fully connected layers, potentially compressing the model further to under 150MB while preserving the critical NetVLAD clustering behavior

Table 1: Performance of x-Embedding Architectures with Different Configurations

Model	MFCCs	Data Size	Length	Optimizer	Size (MB)	Acc (%)
Statistical Pooling Variants						
V1	30	6,649	5s	SGD	356.65	62.33
V2	40	6,649	5s	SGD	-	44.20
V3	23	6,649	5s	SGD	-	59.56
V4	30	11,865	10s	SGD	658.83	69.30
NetVLAD Variants						
V1	30	6,649	5s	AdamW	611.85	52.00
V2	40	6,649	5s	AdamW	-	58.30
V3	40	11,865	10s	AdamW	826.14	78.32

Notes: Missing size entries (-) indicate unchanged values from previous version. NetVLAD uses 8 clusters.

3 Automatic Speech Recognition

We mentioned in Chapter 1

3.1 Dataset

To ensure consistency in our textual analysis, I retrieved the complete Quranic text through the Alquran Cloud API, which provides the scripture in JSON format with full diacritical marks (tashkīl). As these phonetic annotations were not required for our Automatic Speech Recognition, I standardized the text by programmatically removing all diacritics (including fatḥa, kasra, ḍamma, sukūn, and shadda..), preserving only the base Arabic characters. This normalization step: (1) reduced lexical variation in our processing pipeline, (2) aligned with modern Arabic script conventions where diacritics are typically omitted, and (3) minimized potential noise while maintaining the core textual content.

3.2 ASR pretrained model :Whisper

Initially, I used the VOSK model for Arabic transcription, but the results did not meet my expectations. As a result, I switched to Whisper, OpenAI's free speech-to-text model.

3.3 Search System

After experimenting with different approaches, I finalized my method using the RapidFuzz library, which leverages string similarity calculations from FuzzyWuzzy. To achieve the best results after transcribing the audio, I removed all spaces from the Quranic text, converting it into a single continuous string. This decision was based on empirical analysis, where I observed that ASR models frequently introduce spacing errors due to the nature of Quranic recitation, including prolongation and tartil.

Next, I structured the search process into two phases:

- Surah-Level Search: RapidFuzz first searches for the transcribed text within each surah as a single string.
- Ayah-Level Search: Once the relevant surah is identified, a second search is performed to determine the specific ayah.

3.4 Results and Discussion

After testing the system with various audio samples, it was able to correctly identify the specific surah most of the time. However, the model may still require some fine-tuning, particularly in adjusting the similarity threshold. An alternative approach could be implementing a dynamic or soft threshold, allowing the system to adapt by searching for the highest similarity score.

4 Conclusion

This project has been a major milestone in my journey, providing me with valuable insights into both audio intelligence and software development. Throughout the process, I have explored different approaches to speech-to-text transcription, text similarity matching, and optimizing search algorithms for Quranic recitation. These experiences have deepened my understanding of ASR models, string-matching techniques, and the challenges associated with handling spoken language variations.

Moving forward, I am committed to continuously refining and optimizing this project. My goal is to enhance its accuracy, efficiency, and overall performance while making it more adaptable to different use cases. I also aim to integrate it seamlessly into websites and applications, expanding its accessibility and practical applications. By doing so, I hope to create a more robust and user-friendly tool that can effectively assist in Quranic audio analysis and transcription.

References

[1] Relja Arandjelović, Petr Gronát, Akihiko Torii, Tomáš Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5297–5307. IEEE, 2016.

[2] Philipp Koehn and Rebecca Knowles. Finding universal grammatical relations in multilingual bert. *arXiv preprint arXiv:2012.00931*, 2020. URL `https://arxiv.org/abs/2012.00931`.

[3] Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahremani, Vimal Manohar, Xingyu Na, Yiming Wang, and Sanjeev Khudanpur. Semi-orthogonal low-rank matrix factorization for deep neural networks. In *Proceedings of Interspeech*, pages 3743–3747, 2018. URL `https://www.danielpovey.com/files/2018_interspeech_tdnnf.pdf`.

[4] Daniel Povey, Gaofeng Cheng, Yiming Wang, Chao Li, Hainan Xu, Mohammad Yarmohammadi, and Sanjeev Khudanpur. Nist sre 2018: Speaker recognition evaluation in new domains. In *Proceedings of Interspeech*, pages 1483–1487, 2019. URL `https://danielpovey.com/files/2019_interspeech_nist_sre18.pdf`. Accessed: 2024-03-26.

[5] Fradj Sinen. All you need to know about matchboxnet with pytorch, 2024. URL `https://medium.com/@fradjsinen/all-you-need-to-know-about-matchboxnet-with-pytorch-47a36feddb9c`. Accessed: 2024-03-26.