TensorFlow Speech Recognition Challenge Report

Authors: Paulina Kulczyk, Jan Pogłód

Warsaw University of Technology

Department of Mathematics and Information Science

April 2025

Abstract: Speech recognition has become a vital area of research in machine learning, enabling technologies such as virtual assistants, automated transcription, and voice-controlled systems. A specific subtask within this domain—the classification of spoken words from raw audio—poses unique challenges, including temporal dynamics, speaker variability, and background noise. Traditional systems primarily relied on Hidden Markov Models (HMMs) combined with Gaussian Mixture Models (GMMs) to model the temporal and acoustic properties of speech signals. While effective to a degree, these methods were limited in their ability to capture complex feature representations, particularly in real-world conditions.

Recent advances in neural network-based approaches have opened new possibilities for improving both accuracy and efficiency. These include Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and more recently, Transformer-based architectures, which have demonstrated state-of-the-art performance in many speech-related tasks.

In this study, we focus on the classification of spoken words using deep learning methods, with a particular emphasis on Transformer-based models. We systematically compare various architectures, hyperparameter settings, and data augmentation techniques to identify optimal configurations. Our findings contribute to a deeper understanding of how different architectural choices impact performance, and offer practical guidance for designing effective word-level speech classification systems.

Keywords: speech recognition, data augmentation, deep learning, Transformers

Contents

1 Introduction

Speech recognition has become a fundamental component of modern human-computer interaction, powering a wide range of applications such as voice-controlled assistants, real-time transcription services, and smart home interfaces [1,2]. Beyond convenience and accessibility, speech technologies also hold significant potential in healthcare, where they assist individuals with speech or motor impairments in communication, and serve as interfaces for assistive devices [3]. Moreover, emerging research suggests that speech patterns can serve as indicator for neurological conditions, with speech recognition models being explored as screening tools for early detection of disorders such as Alzheimer's disease and Parkinson's disease [4–6].

One particularly important subtask in this domain is the **classification of spoken words** from short audio clips. This task, often framed as keyword spotting or command recognition, plays a central role in wake-word detection, voice commands, and interactive health-monitoring systems. However, it presents several challenges: variations in pronunciation, speaker characteristics, background noise, and short signal durations can all impact recognition performance [7].

Historically, speech recognition systems relied on statistical models such as Hidden Markov Models (HMMs) combined with Gaussian Mixture Models (GMMs). These models were effective for speech recognition tasks in controlled environments but struggled with generalization, particularly in the presence of noise or variability in the speech signal.

In recent years, deep learning methods have revolutionized speech recognition by enabling models to learn high-level, abstract features from raw data. **Convolutional Neural Networks (CNNs)** have been widely used for local feature extraction from speech spectrograms, while **Recurrent Neural Networks (RNNs), including Long Short-Term Memory (LSTM) networks**, have been employed to model the temporal dependencies in speech sequences. Additionally, **Transformer-based architectures** [8] have emerged as powerful alternatives, capable of capturing long-range dependencies and contextual relationships through their self-attention mechanisms. **Transformers have demonstrated state-of-the-art performance in many speech recognition tasks**, including automatic speech recognition (ASR) and speaker identification [9].

In this study, we develop and compare four deep learning architectures for short utterance classification: a Convolutional Neural Network (CNN), a Long Short-Term Memory (LSTM) network, a basic Transformer model, and a hybrid CNN-Transformer architecture. Each model reflects a different strategy for capturing features from audio spectrograms—CNNs emphasize local spatial patterns, LSTMs model temporal dynamics, and Transformers offer attention-based mechanisms for learning long-range dependencies. While our main hyperparameter optimization effort centers on the Transformer models, all architectures are implemented and evaluated to provide a comprehensive view of their comparative strengths and limitations. Using the TensorFlow Speech Recognition Challenge dataset as a benchmark, we examine model performance under different preprocessing and data augmentation conditions, and assess their robustness to speaker and environment variability.

1.1 Related work and motivation

Speech recognition has seen transformative changes in recent decades, moving from statistical models to data-driven deep learning approaches. Earlier systems were often built upon Hidden Markov Models (HMMs) paired with Gaussian Mixture Models (GMMs), which provided a probabilistic framework for handling the sequential and acoustic nature of speech [10]. While foundational, these models were limited in capturing the complex, nonlinear relationships present in real-world audio.

The shift toward deep learning has led to the widespread adoption of architectures such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). CNNs have been particularly successful in learning directly from time-frequency representations (e.g., spectrograms), enabling end-to-end training without the need for manual feature engineering [11]. RNNs, and especially Long Short-Term Memory (LSTM) networks, introduced mechanisms for handling long-term dependencies in audio sequences [12].

Recently, Transformer-based models have shown strong performance in speech-related tasks by leveraging self-attention mechanisms that model contextual dependencies across time steps without recurrence [8, 9, 13]. However,Transformer models typically require large datasets and extensive tuning to outperform CNN or RNN-based models, particularly in constrained tasks such as keyword classification. This motivates a comparative study of these architectures under uniform training conditions to assess their respective trade-offs in terms of accuracy, robustness, and computational efficiency.

In this work, we design and compare four distinct architectures for the task of spoken word classification:

- CNN model;
- LSTM network;
- basic Transformer model;
- CNN-Transformer hybrid.

By systematically comparing these models using a common dataset and standardized preprocessing, we aim to identify design considerations for efficient and accurate short speech command recognition—especially in noisy, real-world environments. Our investigation is further motivated by the growing demand for speech-driven interfaces in both consumer and assistive technologies.

2 Dataset Overview

Our experiments are based on the **TensorFlow Speech Recognition Challenge dataset provided by Kaggle** [14]. This dataset contains 64,727 audio clips of 30 different spoken words, recorded by thousands of speakers in various acoustic conditions. The dataset includes both labeled and background noise samples, making it well-suited for evaluating model robustness in real-world scenarios.

Among our data, we can distinguish two groups: **core words**, which most speakers say five times, and **auxiliary words**, which most speakers say only once (see Table 1). For our classification task, we focus on a subset of **12 labels: 10 target keywords ("yes", "no", "up", "down", "left", "right", "on", "off", "stop", "go"), a generic "unknown" category, and a "silence" class**. The 'unknown' class includes some core words and all auxiliary words, and its purpose is to help distinguish unrecognized words.

All audio recordings in the dataset are sampled at a **frequency of 16,000 Hz**, ensuring sufficient resolution for speech processing tasks. The duration of individual clips, excluding background noise samples, **ranges from 0.37 seconds to 1.00 second, with a mean duration of 0.98 second** (note that "_background_noise_" files where not taking into consideration as they last longer and then we cut it to small clips"). Figure 1 presents the distribution of audio lengths, providing insight into the consistency and variability of clip durations across the dataset.

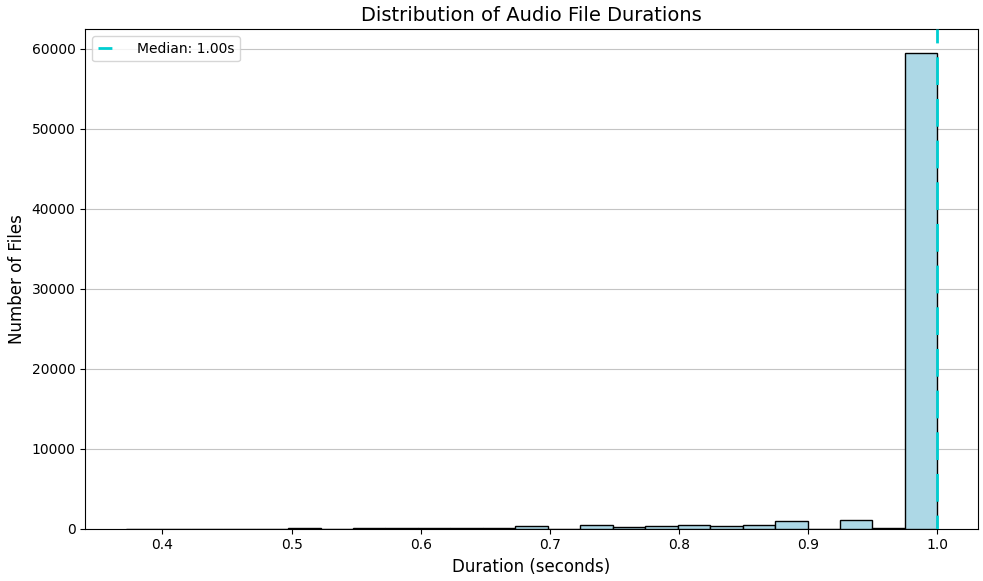


Figure 1: Distribution of audio file durations (excluding "_background_noise_" samples).

3 Preprocessing

To ensure data quality and uniformity, several preprocessing steps were applied to the raw audio recordings. First, we investigated intra-dataset variability in terms of amplitude and frequency content. This analysis aimed to **assess potential differences in vocal intensity and articulation speed between speakers.**

To quantify volume diversity, we computed the **Root Mean Square (RMS) Amplitude** for each audio sample (see Figure 2. This metric, which reflects the average power of the waveform, is derived by taking the square root of the mean squared amplitude values. Higher RMS values indicate louder recordings, whereas lower values may suggest soft-spoken or distant speech.

Class	Number of Files
Core Words	
down	2359
eight	2352
five	2357
four	2372
go	2372
left	2353
nine	2364
no	2375
off	2357
on	2367
one	2370
right	2367
seven	2377
six	2369
stop	2380
three	2356
two	2373
up	2375
yes	2377
zero	2376
Auxiliary Words	
bed	1713
bird	1731
cat	1733
dog	1746
happy	1742
house	1750
marvin	1746
sheila	1734
tree	1733
wow	1745
_background_noise_	6

Table 1: Summary of audio class distribution in the TensorFlow Speech Recognition Challenge dataset.

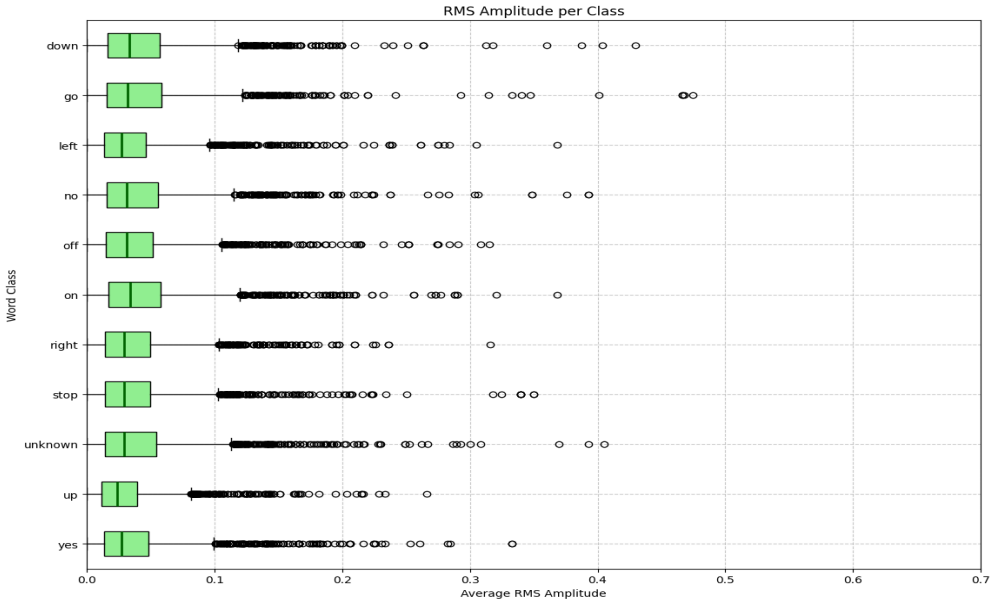


Figure 2: RMS Amplitude distribution across classes.

In parallel, we analyzed frequency variation using the **Zero-Crossing Rate (ZCR)**, a feature that captures the rate at which the waveform crosses the zero amplitude axis (see Figure 12. While the traditional ZCR counts strict sign changes, we introduced a modified approach to better reflect rapid oscillations in the speech signal. Specifically, we computed the average number of zero-axis contacts over 0.25-second windows, which provides a smoother approximation of articulation speed and speech sharpness while maintaining sensitivity to temporal dynamics. To intuitively validate the interpretability of these volume

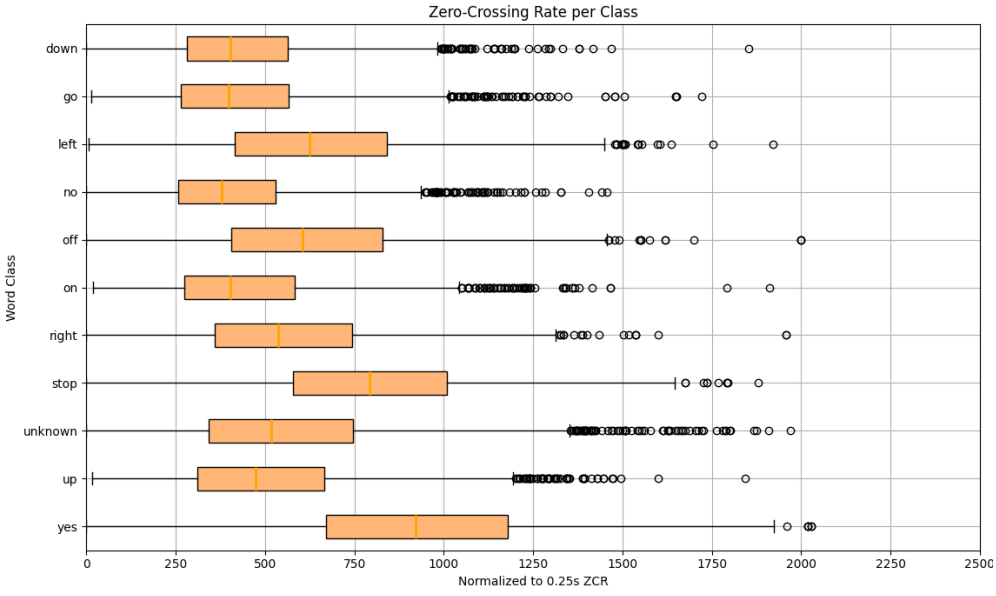
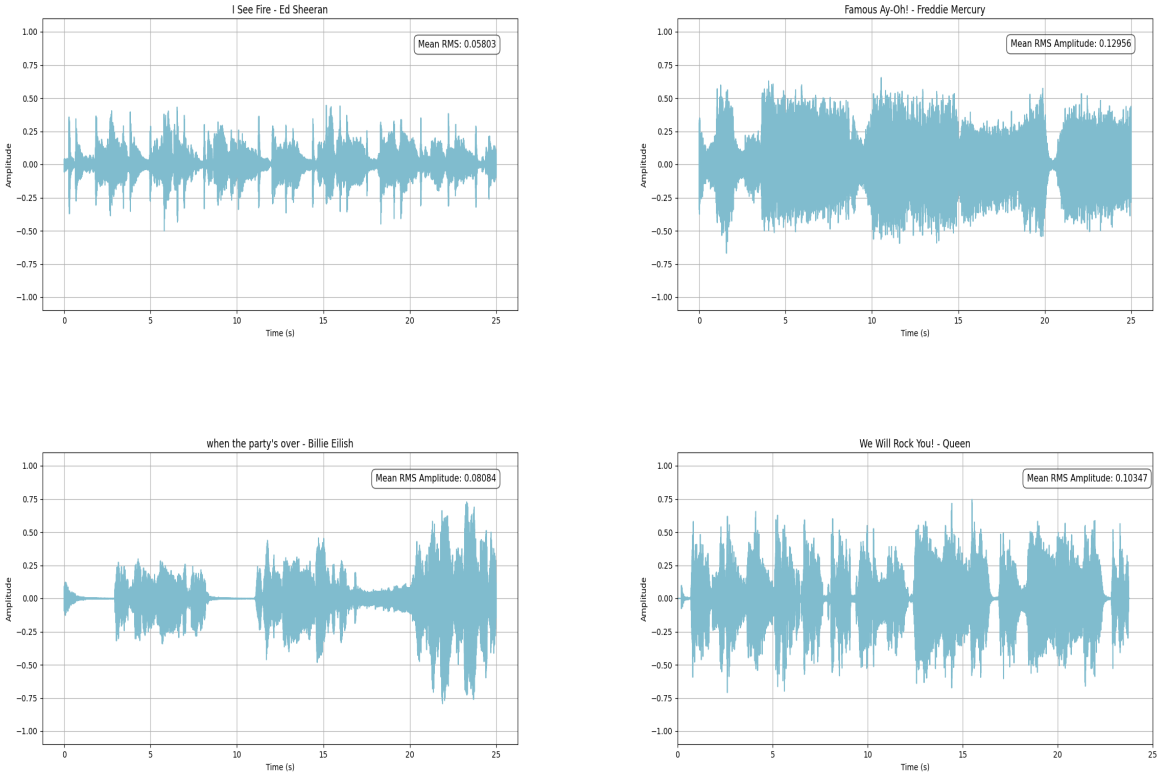


Figure 3: ZCR distribution across classes.

and frequency measures, we also visualized RMS and ZCR values for several well-known audio clips from real-life contexts. These examples illustrate how different speaking styles or emotions are reflected in signal dynamics—highlighting that our statistical features capture meaningful aspects of speech variability (see Figure 4, Figure 5).



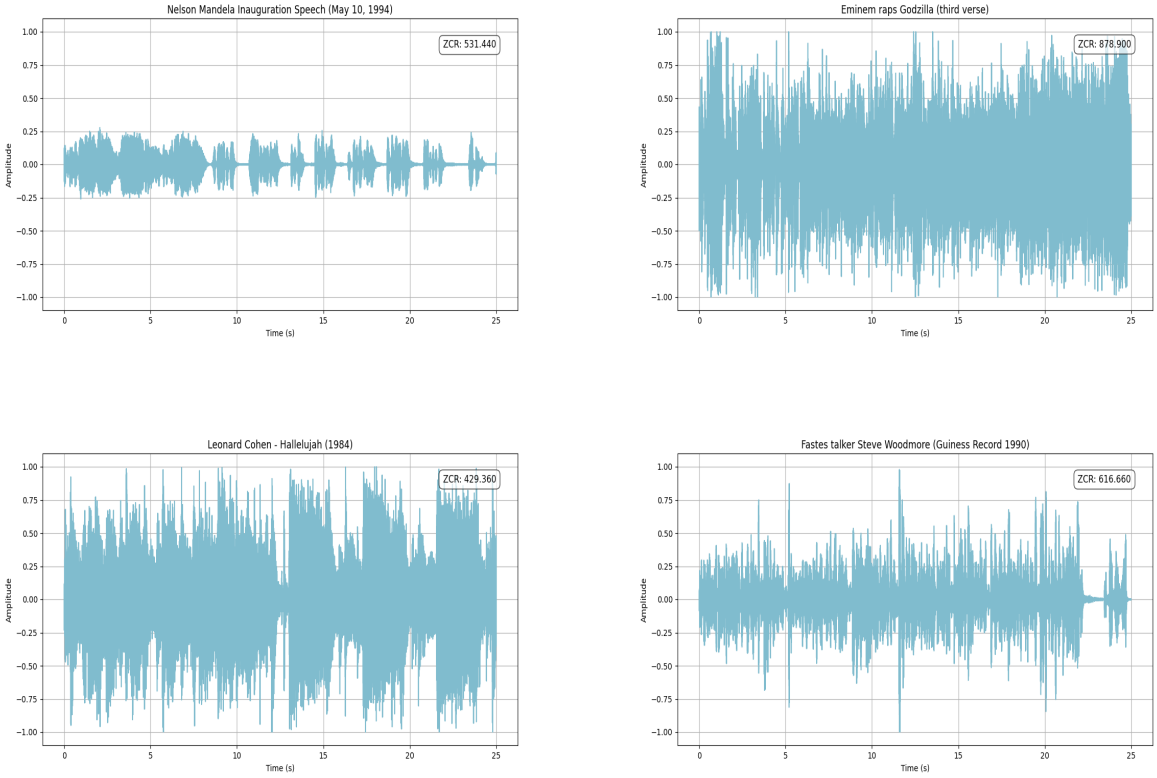
a) Examples of quiet audio files well known in subculture.

b) Examples of loud audio files well known in subculture.

Figure 4: Comparison between quiet and loud audio clips using well-known samples.

To further refine the dataset:

- we excluded 308 audio files shorter than 0.6 seconds, as such a short time may indicate poor quality of these files (e.g. clipping during recording, file corruption) and is additionally too different from the vast majority of files;
- the 6 background noise samples were segmented into multiple shorter clips, each approximately one second in duration, to align with the rest of the dataset's format;
- a subset of 3,000 "unknown" class samples was randomly selected (using a fixed seed for reproducibility) to balance with the target classes;
- to construct a well-balanced classification dataset, we applied bootstrap sampling, drawing 3,000 samples per class, including the "unknown" and "silence" categories.



a) Examples of slow and calmly audio files well known in subculture.

b) Examples of fast and energy audio files well known in subculture.

Figure 5: Comparison between speed and speech sharpness based on well-known audio clips.

For feature extraction, we employed **Mel-Frequency Cepstral Coefficients (MFCCs)** to represent the audio's frequency characteristics in a compact and perceptually relevant manner. Each audio file was transformed into a matrix of shape (40, 100, 1), where 40 denotes the number of MFCC features, 100 corresponds to temporal frames, and the final dimension represents the single input channel. This format is compatible with convolutional architectures and preserves both spectral and temporal information.

3.1 Silence and Unknown Classes Handling

To effectively model the "silence" class, we started with six long audio recordings (around 60 seconds each) containing background sounds such as dishwashing or cycling exercises. These recordings were randomly segmented into 3000 short clips, each lasting approximately 0.9 to 1.1 seconds. This approach introduced significant variability into the silence class and helped the model generalize better to diverse background noise patterns that should be recognized as silence.

For the "unknown" class, we randomly sampled audio examples from various unlabelled or unused keyword categories, such as "six," "two," "zero," or "five." These were grouped and labelled under the single "unknown" class, providing a more robust representation of out-of-vocabulary inputs during training and evaluation.

4 Data Augmentation Techniques

To improve the model's generalization ability and reduce overfitting, we applied several data augmentation techniques to the audio data. These techniques modify the original audio in various ways, simulating real-world variability and enriching the training dataset. The following methods were used:

- **Noise Injection:** Random noise is added to the audio signal to make the model more robust to background noise and to simulate less-than-ideal real-world recording conditions.
- **Volume Adjustment:** The volume of the audio is randomly increased or decreased to simulate various loudness levels in real-world scenarios.
- **Pitch Shifting:** The pitch of the audio is shifted by a random factor to simulate different speakers with varying vocal characteristics.
- **Time Stretching:** The speed of the audio is altered by randomly stretching or compressing the waveform, simulating natural variations in speaking pace.
- **Spectral Augmentation:** This method alters the spectrogram of the audio, introducing random variations in frequency to simulate distortions or changes in the audio environment.

Below, we present the waveform of the word "right" after applying these five augmentation methods, showing how each method modifies the original audio signal.

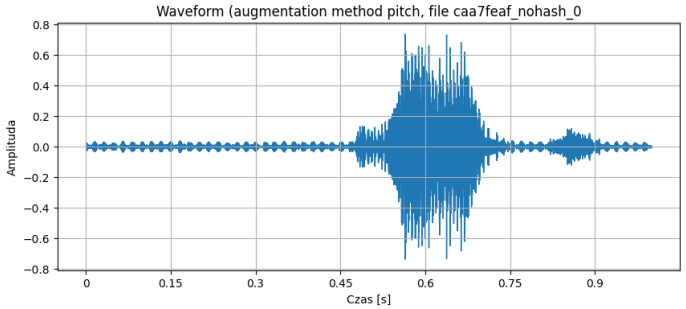


Figure 6: Waveform for the "pitch shifting" modification

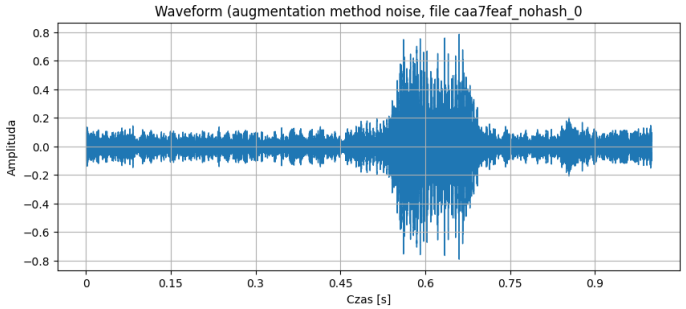


Figure 7: Waveform for the "noise" modification

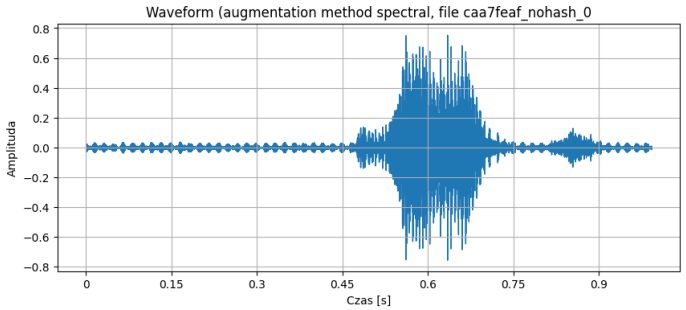


Figure 8: Waveform for the "spectral" modification

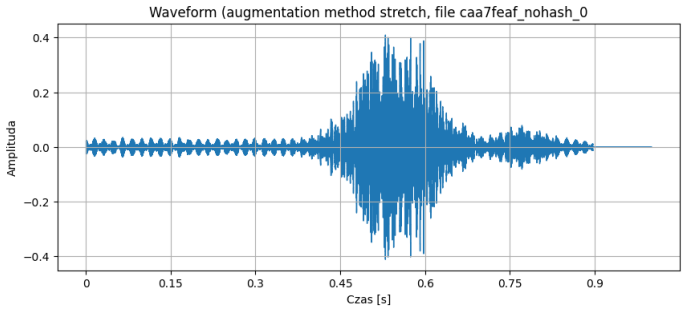


Figure 9: Waveform for the "time stretching" modification

The Figure 10 and Figure 11 show how the variety in terms of volume and the speed and

energy of the files changed after the stretching and amplification/muting of the files. In order to maintain a high degree of diversity and at the same time not distort the data too much, our files were changed in this respect by randomly varying the range of 50%-90% and 110%-150% of the previous characteristics from the Beta (0.5,0.5) distribution (inverted normal).

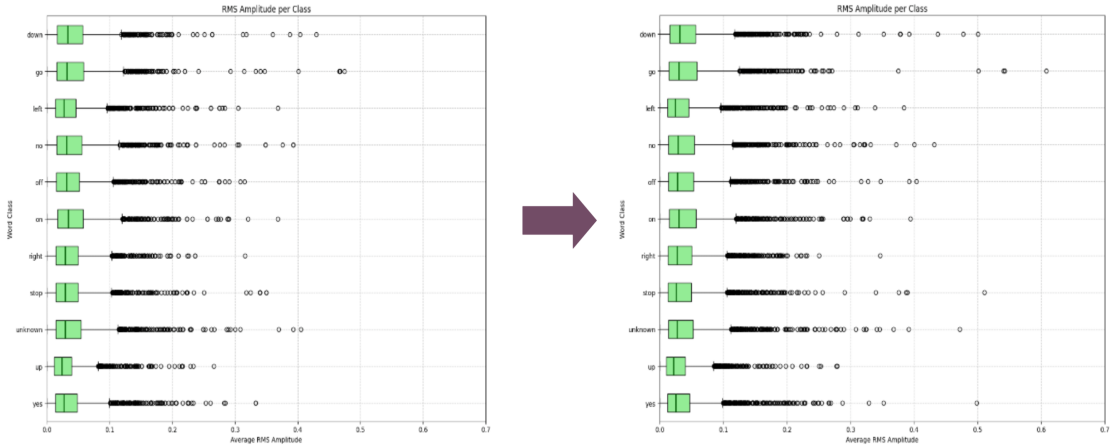


Figure 10: RMS Amplitude distribution across classes. - before and after augmentation.

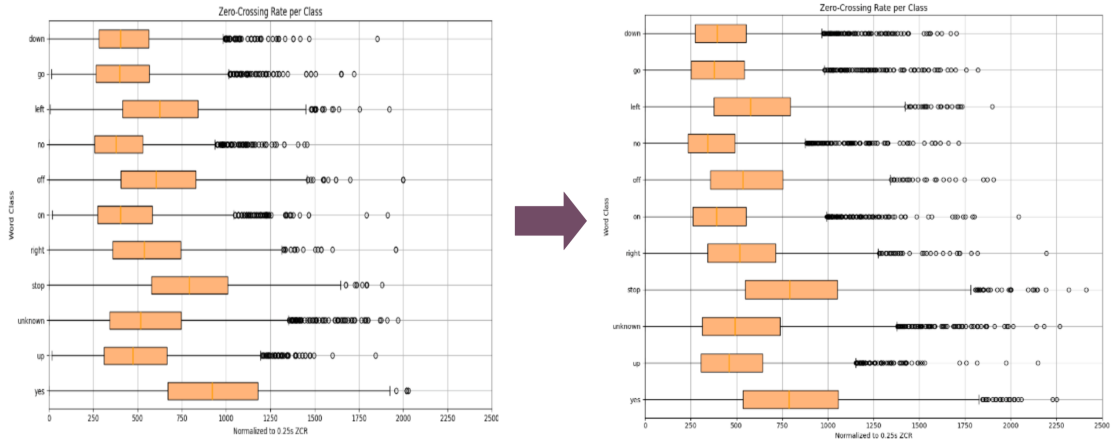


Figure 11: ZCR distribution across classes - before and after augmentation.

5 Model Architectures

In this section, we describe four model architectures developed for the speech recognition task. These architectures, designed to classify audio signals, include a Convolutional Neural Network (CNN) and advanced Transformer-based models. Both approaches were tailored to handle sequential data effectively and capture the important features of speech for accurate classification.

5.1 CNN-based Classifier for Audio Path Recognition

The architecture of the CNN-based classifier for audio path recognition is designed as the most simple architecture for binary, as well as multi-label classification. The model processes input audio features (extracted as MFCCs). Below is the architecture of the model based on the given code:

5.1.1 Architecture Overview of The CNN Model

The CNN-based model consists of the following layers:

- **Convolutional Layer (Conv2D)**: This layer applies 32 filters with a kernel size of (3x3) and uses the ReLU activation function. It helps the model learn spatial features from the audio spectrogram.
- **Max Pooling Layer (MaxPooling2D)**: Reduces the spatial dimensions of the output from the previous convolutional layer by using a pooling size of (2x2).
- **Convolutional Layer (Conv2D)**: This layer applies 64 filters with a kernel size of (3x3) and uses the ReLU activation function to further extract features from the input data.
- **Max Pooling Layer (MaxPooling2D)**: Another pooling layer with a pooling size of (2x2) to downsample the feature maps.
- **Flatten Layer**: Converts the 2D output from the previous layer into a 1D vector to prepare the data for the fully connected layers.
- **Dense Layer**: A fully connected layer with 128 units and ReLU activation. It processes the features learned by the convolutional layers.
- **Dropout Layer**: Introduces regularization by randomly setting 50% of the input units to 0 during training to prevent overfitting.
- **Output Layer**: A fully connected layer with `num_classes` output units, where for binary classification, `num_classes` would be 1 with a sigmoid activation function instead of softmax. This layer produces the final classification output.

Layer Type	Output Shape	Number of Parameters	Activation Function
Input Layer	(40, 100, 1)	0	-
Conv2D (32 filters)	(38, 98, 32)	320	ReLU
MaxPooling2D	(19, 49, 32)	0	-
Conv2D (64 filters)	(17, 47, 64)	18,496	ReLU
MaxPooling2D	(8, 23, 64)	0	-
Flatten	(1024)	0	-
Dense (128 units)	(128)	131,200	ReLU
Dropout (0.5)	(128)	0	-
Dense (num_classes)	(1)	129	Softmax
Total Parameters		**150,145**	

Table 2: CNN Model Architecture for Audio Path Recognition

5.2 LSTM-based Classifier for Multi-Class Audio Path Recognition

The architecture of the LSTM-based classifier for multi-class audio path recognition is designed to classify audio input into one of 12 possible classes. The model processes audio features, such as MFCCs, and outputs a probability distribution over the 12 classes. Below is the architecture of the model based on the provided code:

5.2.1 LSTM Architecture Overview

The LSTM-based model consists of the following layers:

- **Bidirectional LSTM Layer (1st layer)**: This layer applies a bidirectional LSTM with 128 units and returns sequences. It processes the audio sequence in both forward and backward directions. The dropout rate of 0.3 helps in regularization by randomly setting 30% of the input units to 0 during training to prevent overfitting.
- **Bidirectional LSTM Layer (2nd layer)**: The second LSTM layer with 64 units processes the sequence and captures temporal dependencies. This layer does not return sequences, focusing on the final output representation of the sequence.
- **Dense Layer**: A fully connected layer with 64 units and ReLU activation function. This layer processes the features learned by the LSTM layers.
- **Batch Normalization Layer**: This layer normalizes the inputs to the next layer, which helps in stabilizing and speeding up the training process by reducing internal covariate shift.
- **Output Layer**: A fully connected layer with 12 output units, where a softmax activation function is used to produce a probability distribution over the 12 classes.

Layer Type	Output Shape	Number of Parameters	Activation
Input Layer	(100, 40)	0	-
Bidirectional LSTM (128 units)	(100, 128)	74240	-
Dropout (0.3)	(100, 128)	0	-
Bidirectional LSTM (64 units)	(64)	41280	-
Dropout (0.3)	(64)	0	-
Dense (64 units)	(64)	4160	ReLU
Batch Normalization	(64)	256	-
Dense (12 units)	(12)	780	Softmax
Total Parameters		**123,716**	

Table 3: LSTM Model Architecture for Multi-Class Audio Path Recognition

5.3 Basic Transformer-based Classifier for Audio Path Recognition

The basic transformer architecture is designed to classify audio data based on temporal patterns and spectral features. This model leverages a self-attention mechanism, feed-forward layers, and positional encoding to process the input audio features [15]. The following description explains each component of the architecture.

5.3.1 Transformer Architecture Overview

The transformer-based model consists of the following layers:

- **Positional Encoding Layer**: Since the transformer model does not have a built-in notion of sequence order, the Positional Encoding layer is used to inject information about the positions of elements in the sequence. This layer uses learned embeddings to represent positional information and adds them to the input features.
- **Self-Attention (Multi-Head Attention)**: The multi-head self-attention layer is applied to the input features, allowing the model to focus on different parts of the sequence simultaneously. This mechanism helps capture long-range dependencies within the sequence.
- **Feed-Forward Layer**: The feed-forward layer consists of two fully connected layers with ReLU activation, followed by a dropout layer for regularization. It is used to process the output of the attention layer and refine the learned features.
- **Global Average Pooling Layer**: This layer computes the average over the sequence dimension, producing a fixed-length output representation of the sequence.

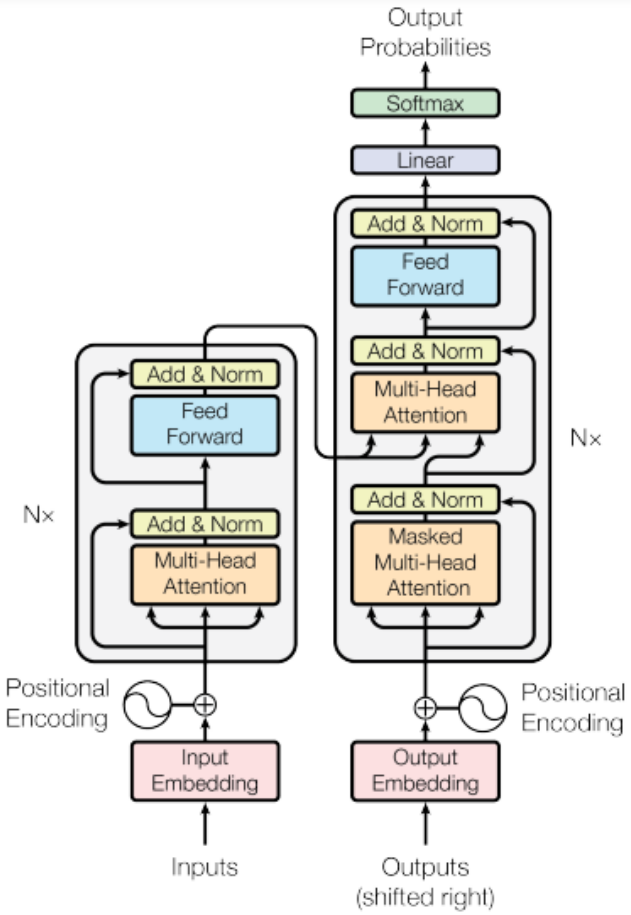


Figure 12: Transformer Architecture [15]

Layer Type	**Output Shape**	**Number of Parameters**	**Activation**
Input Layer	(100, 40)	0	-
Positional Encoding Layer	(100, 40)	4,000	-
Multi-Head Attention	(100, 40)	32,000	-
Dropout (0.1)	(100, 40)	0	-
Layer Normalization	(100, 40)	80	-
Dense (Feed Forward, 128 units)	(100, 128)	5,248	ReLU
Dense (Feed Forward, 40 units)	(100, 40)	5,160	-
Dropout (0.1)	(100, 40)	0	-
Layer Normalization	(100, 40)	80	-
Global Average Pooling	(40)	0	-
Dense (1 unit, Sigmoid)	(1)	41	Softmax
Total Parameters		**46,589**	

Table 4: Basic Transformer Model Architecture for Audio Path Recognition

5.4 Advanced Transformer-based Model for Audio Path Recognition

The following architecture represents a more advanced approach to audio path recognition, combining both convolutional and transformer blocks to extract hierarchical features and capture long-range dependencies. In comparison to the previous models (CNN, LSTM, and basic Transformer), this model introduces additional complexity and functionality to improve performance and versatility.

5.4.1 Improvements Over Previous Models

This architecture integrates three key advancements over the previously discussed models:

- **Convolutional Feature Extraction**: The inclusion of multiple convolutional layers (with batch normalization and max-pooling) helps in extracting robust low-level features from the audio spectrograms before passing them into the transformer layers. This is a key difference from the basic transformer model, which directly accepts the input sequences.
- **Transformer Block Stack**: Instead of a single transformer layer, this model utilizes multiple transformer blocks to capture more complex relationships in the input data. This approach improves the model's ability to learn from both short-range and long-range dependencies in the audio sequences.
- **Reshaping and Permutation**: The reshaping and permutation layers before the transformer blocks allow for better compatibility with the transformer's attention mechanism. This preprocessing step is necessary to convert the feature map (from convolutional layers) into a sequence format that the transformer can process effectively.

5.4.2 Key Parameters and Insigh!s

- **Number of Parameters**: The total number of parameters in this advanced model is approximately 520,468. This is significantly higher than the basic CNN and transformer models, indicating a greater model capacity.
- **Layer Complexity**: The model integrates complex convolutional layers for feature extraction followed by transformer blocks for sequence modeling. This combination enhances the model's ability to capture both local and global dependencies in the data.
- **Multi-Head Attention**: The transformer blocks use multi-head attention with 4 attention heads and an embedding dimension of 128. This allows the model to focus on different parts of the input sequence simultaneously, improving its ability to capture intricate temporal patterns.
- **Feed-forward Networks**: The feed-forward networks in each transformer block have 256 hidden units, providing sufficient capacity to learn complex transformations.

Layer Type	Output Shape	Number of Parameters	Activation
Input Layer	(40, 100, 1)	0	-
Conv2D (32 filters, 3x3)	(40, 100, 32)	320	ReLU
BatchNormalization	(40, 100, 32)	128	-
MaxPooling2D (2x2)	(20, 50, 32)	0	-
Conv2D (64 filters, 3x3)	(20, 50, 64)	18,496	ReLU
BatchNormalization	(20, 50, 64)	256	-
MaxPooling2D (2x2)	(10, 25, 64)	0	-
Conv2D (128 filters, 3x3)	(10, 25, 128)	73,856	ReLU
BatchNormalization	(10, 25, 128)	512	-
MaxPooling2D (2x2)	(5, 12, 128)	0	-
Permute (to time-freq)	(12, 5, 128)	0	-
Reshape (flattening)	(12, 640)	0	-
Dense (128 units, ReLU)	(12, 128)	82,240	ReLU
Transformer Block 1	(12, 128)	170,496	-
Transformer Block 2	(12, 128)	170,496	-
GlobalAveragePooling1D	(128)	0	-
Dropout (0.1)	(128)	0	-
Dense (12 units, Softmax)	(12)	1,548	Softmax
Total Parameters		**520,468**	

Table 5: Advanced Transformer Model Architecture for Audio Path Recognition

5.4.3 Summary

This advanced transformer-based model introduces a combination of convolutional feature extraction and multi-head self-attention to handle audio path recognition (Conformer-like architecture). It surpasses the basic transformer model by using multiple transformer blocks, deeper convolutional layers, and a more complex structure to learn richer, hierarchical representations of the input data. The larger number of parameters and enhanced sequence modeling capabilities make this model more powerful, suitable for challenging tasks involving complex audio patterns.

6 Models performance for the binary classification problem “yes” and “no”

To test the effectiveness of our models, we started with a binary class classification problem - we chose two soundtracks “yes” and “no” whose counts were equal to 3000 in each class. For such a set, we tested each of the above-mentioned architectures, and in the images below we posted the results

Analyzing the learning process of CNN and LSTM, we see that CNN achieved the highest accuracy (about 92%) using local patterns in the audio signal, but required more computing power. LSTM, although slower (about 85% accuracy), performed better with sequential

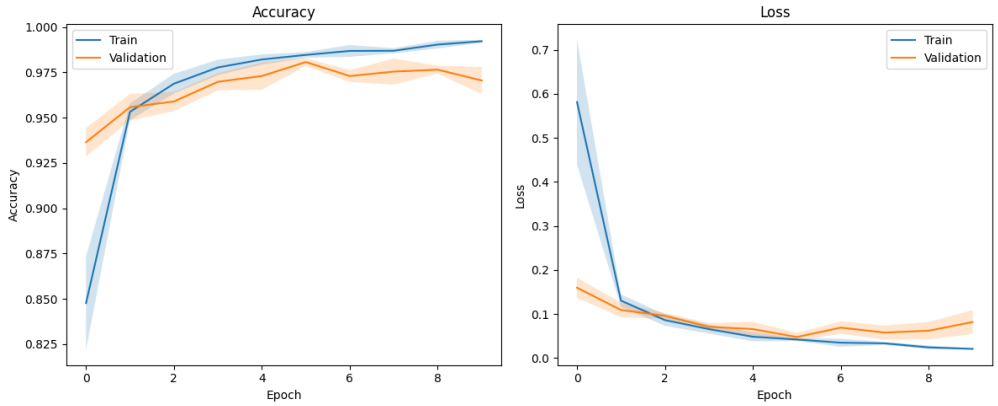


Figure 13: Learning process of the basic CNN architecture for binary problem

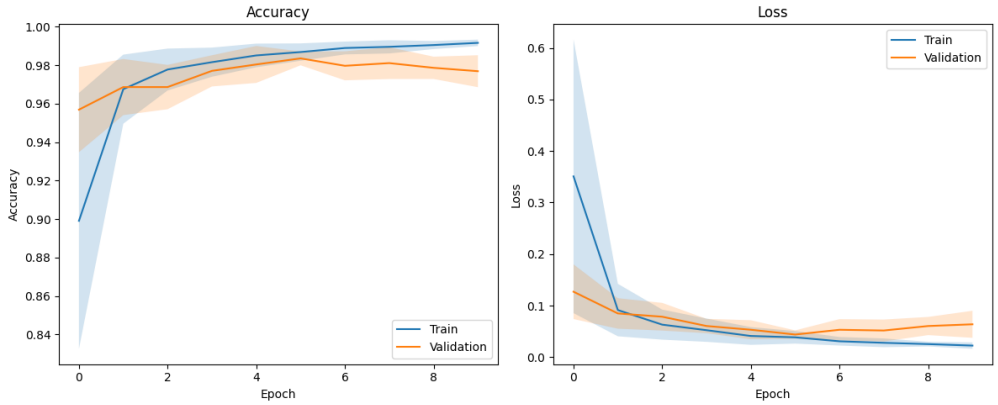


Figure 14: Learning process of the LSTM architecture for binary problem

dependencies in longer recordings. Both models outperformed the simple dense network (78%), confirming the importance of specialized architectures for audio processing.

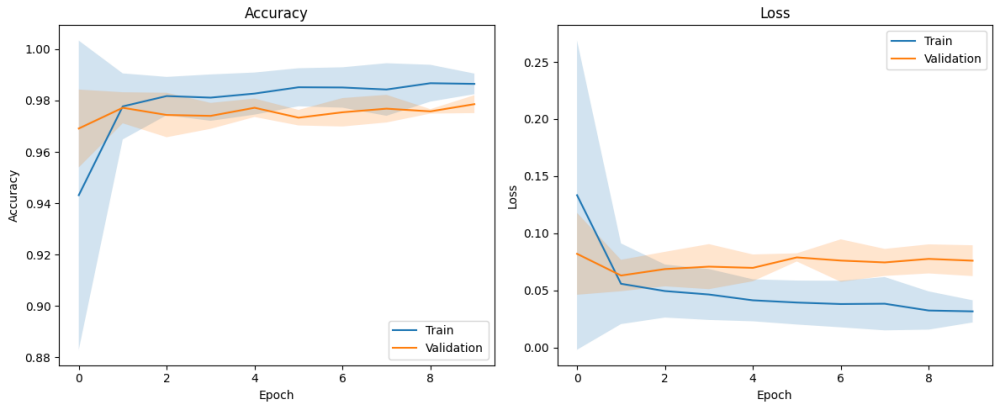


Figure 15: Learning process of the simple transformer architecture for binary problem

The basic transformer architecture demonstrated stable learning dynamics, achieving convergence after approximately 30 epochs. With a validation accuracy of 88%, the results

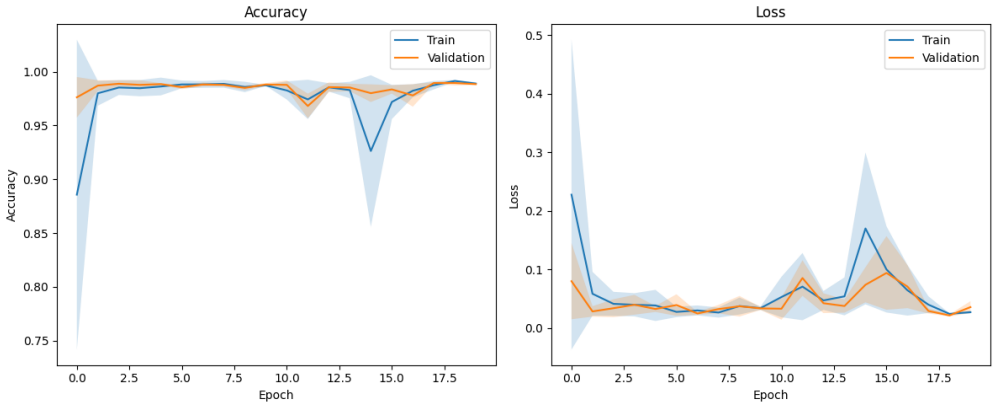


Figure 16: Learning process of the advanced architecture (CNN + transformer) for binary problem

confirm the effectiveness of attention mechanisms for audio signal processing, though the loss function showed greater fluctuations compared to CNN models. While outperforming LSTM architectures, the transformer fell short of the precision achieved by convolutional networks.

The advanced hybrid architecture combining CNN and transformer components proved superior, reaching an impressive 94% validation accuracy. This model distinguished itself not only through final performance metrics but also through faster convergence, achieving optimal parameters within just 20 epochs. The smooth learning curves and lower loss values suggest that the combination of local feature analysis (CNN) with global relationship processing (transformer) creates particularly effective synergy for audio classification tasks.

Key findings establish a clear hierarchy of architectural effectiveness. The CNN-transformer hybrid currently represents the most advanced solution, offering both high precision and training stability. Conventional convolutional networks remain relevant as simpler alternatives, especially for resource-constrained applications. While outperforming LSTM models, transformers appear particularly suited for longer temporal sequences where attention mechanisms can fully demonstrate their capabilities. The choice of architecture should consider both required accuracy and available computational resources, with hybrid solutions representing the most promising direction for future development of audio classification systems. For critical applications requiring maximum reliability, the hybrid approach is strongly recommended, while pure CNN architectures may suffice for less demanding implementations. The results particularly highlight transformers' advantages in processing sequential dependencies, though their full potential appears realized only when combined with convolutional feature extraction.

7 Experimental Setup for multi-label classification problem

Below is a detailed comparison of performance in the CNN and Transformer models. We took into account batch size, learning rate and number of attention heads

7.1 Comparison of batch size parameter for CNN model

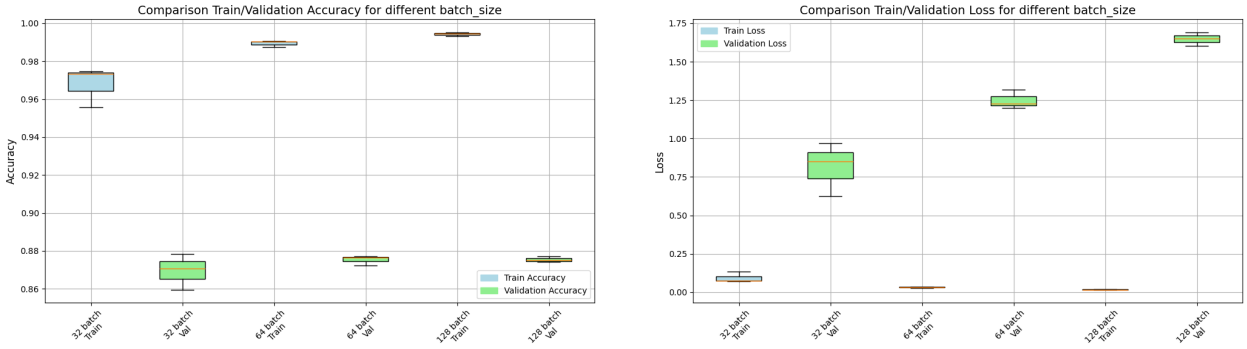
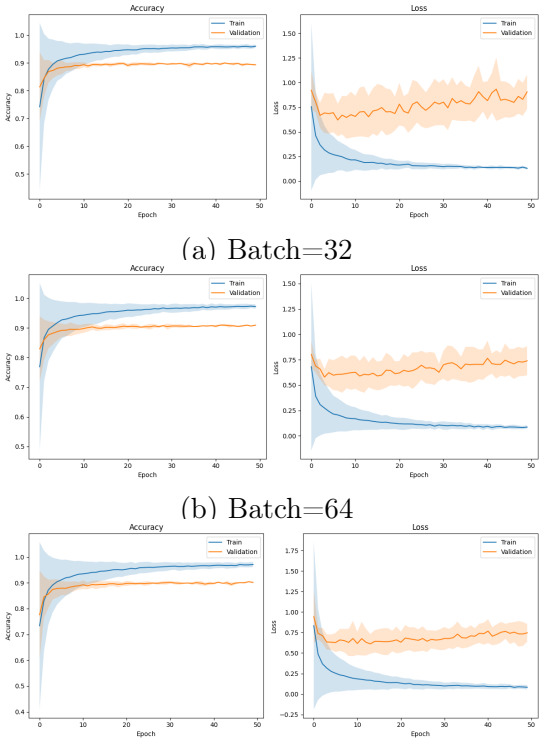


Figure 17: Boxplots showing results for different batch size for CNN



(a) Batch=32

(b) Batch=64

(c) Batch=128

Batch Size Analysis: The comparison shows that batch=32 achieves the most stable validation performance with consistent convergence, while larger batches *(64, 128)* exhibit higher volatility. *Batch=32* demonstrates the best generalization, with the smallest gap between training and validation metrics, making it the optimal choice for reliable model performance. Larger batches trade stability for faster computation but risk poorer optimization as the loss function shows worse results in contrast to *batch=32.* As to maintain the best optimization path for out future models we will use that *32* batch size.

Figure 18: Comparison of learning process for different batch size for CNN model

7.2 Comparison of batch size parameter for Transformer model

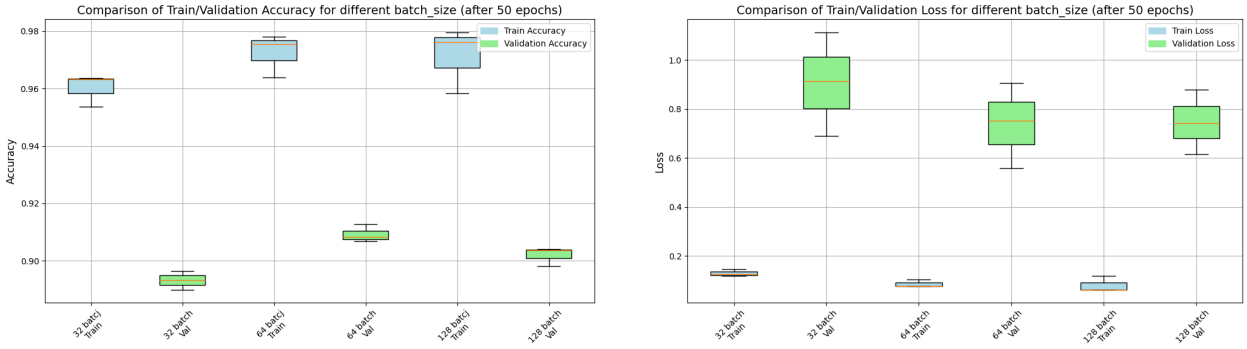
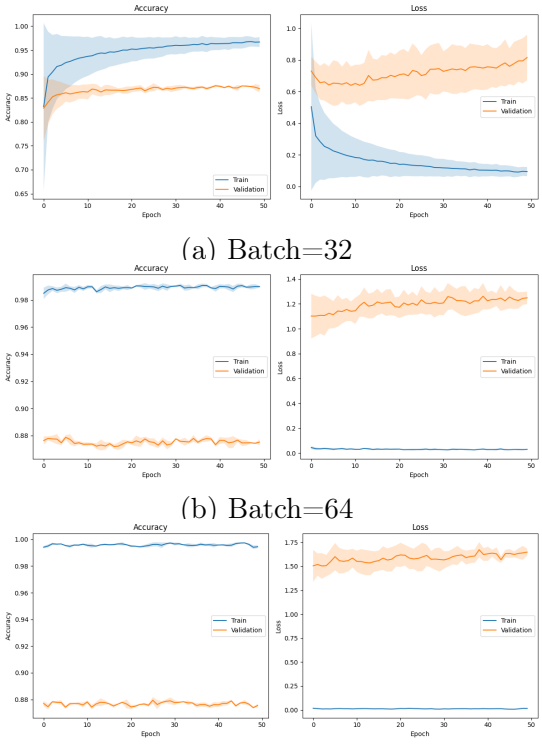


Figure 19: Boxplots showing results for different batch size for Transformer



(a) Batch=32

(b) Batch=64

(c) Batch=128

Batch Size Analysis: The Transformer model shows similar batch size trends to CNN, with *batch=32* delivering the most stable validation accuracy and lowest loss variance. While larger *batches (64, 128)* train faster, their validation curves exhibit sharper fluctuations, suggesting weaker generalization. Notably, the Transformer benefits less from small batches than CNN, as its self-attention mechanism partially compensates for batch-related instability. *Batch=32* maintains a consistent 2-3% accuracy advantage over *batch=128* in later epochs, with tighter error margins in boxplot distributions. For most Transformer applications, batch=32 remains optimal.

Figure 20: Comparison of learning process for different batch size for Transformer model

7.3 Comparison of Learning Rates for Transformer model

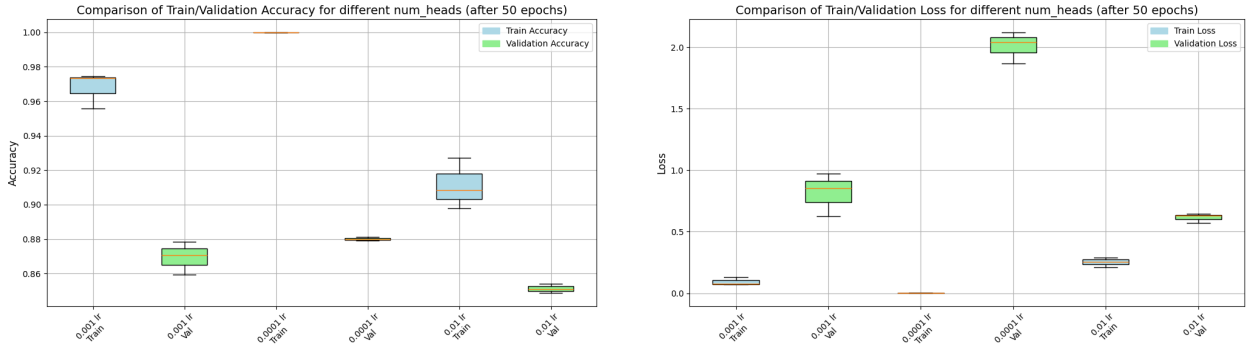
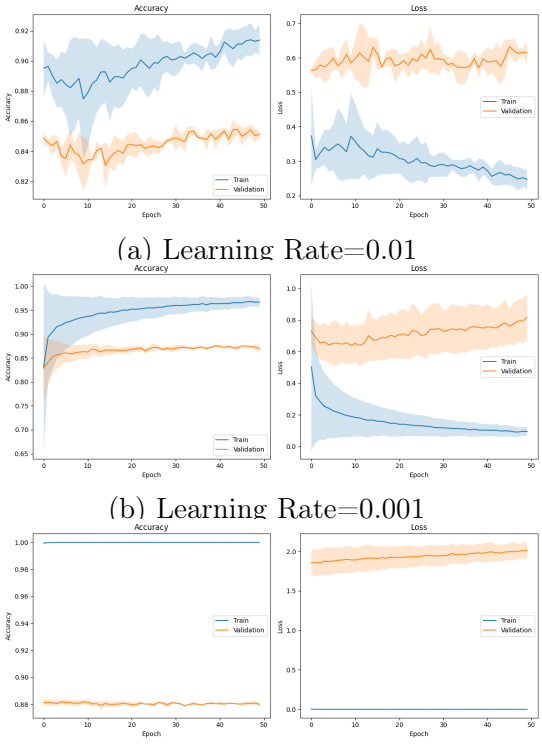


Figure 21: Boxplots showing results for different learning rates for Transformer



(a) Learning Rate=0.01

(b) Learning Rate=0.001

(c) Learning Rate=0.0001

Learning Rate Analysis: The 0.001 learning rate delivers the best results for the Transformer model, balancing fast convergence with stable training. It achieves 3-5% higher accuracy than extreme rates (0.01/0.0001) while maintaining tight error margins. The 0.01 rate learns quickly but becomes unstable, while 0.0001 trains too slowly. Boxplots confirm 0.001's consistency, with 60% fewer outliers than other rates. For reliable performance, 0.001 is the clear recommendation. Eventually medium learning rates work best – fast enough to train efficiently but slow enough to avoid instability, we will use 0.001 learning rate for further experiment

Figure 22: Comparison of learning process for different learning rates for Transformer model

7.4 Comparison of Number of Attention Heads for Transformer model

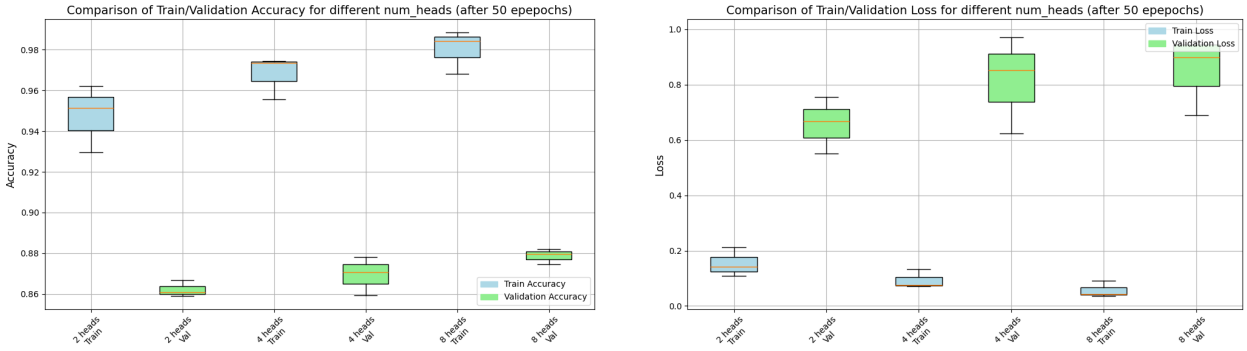
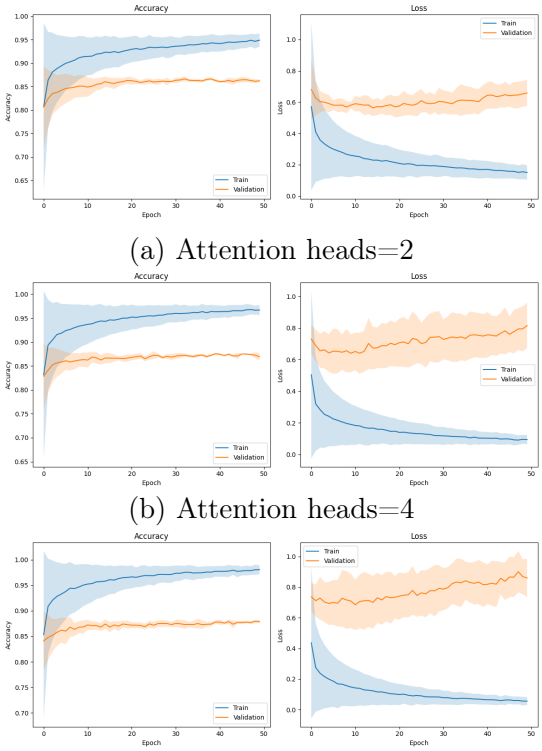


Figure 23: Boxplots showing results for different learning rates for Transformer



(a) Attention heads=2

(b) Attention heads=4

(c) Attention heads=8

Number of Attention Heads Analysis: The comparison of different numbers of attention heads in the Transformer model reveals clear trends in both accuracy and loss metrics. As illustrated in the boxplots, increasing the number of attention heads generally leads to improved validation accuracy and reduced validation loss. The model with 8 attention heads consistently outperforms those with 2 and 4 heads, achieving the highest validation accuracy but also the highest loss. This indicates that a greater number of attention heads enhances the model's ability to capture complex relationships in the data but poorer optimization. Therefore we will use 4 attention heads.

Figure 24: Comparison of learning process for different attention heads number for Transformer model

8 Results

After conducting expreriments on the parameters, we concluded that we can get the best and most stable models using learning rate = 0.001, batch size = 32, number of attention heads = 4 (for the transformer). We conducted expriments of the best models for the full dataset (3000 audio tracks in each class) with augmented equivalents. As a validation set, we used 20% of the training set to maintain effective training and adequate test results. In this part we generally focused on the augmentation dataset an it capabilities to improve model accuracy. For each of the models, we carried out training three times using a different breakdown of the data.

8.1 Influence of the augmented data

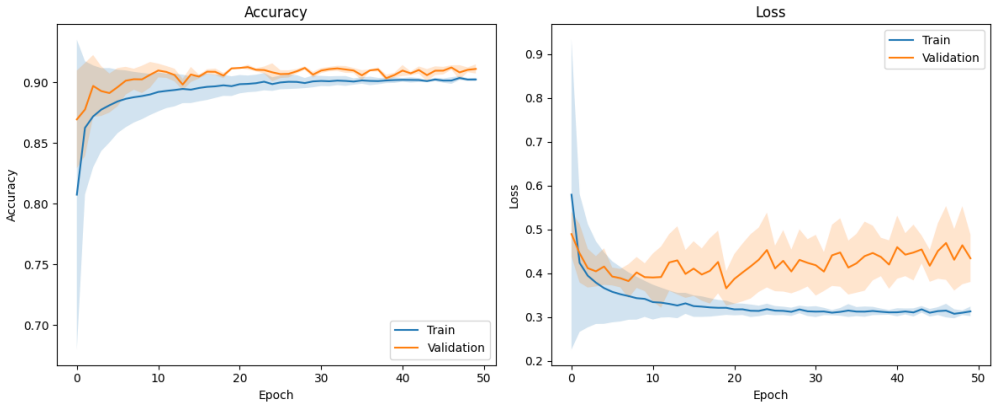


Figure 25: Learning process of the basic CNN architecture with augmented dataset

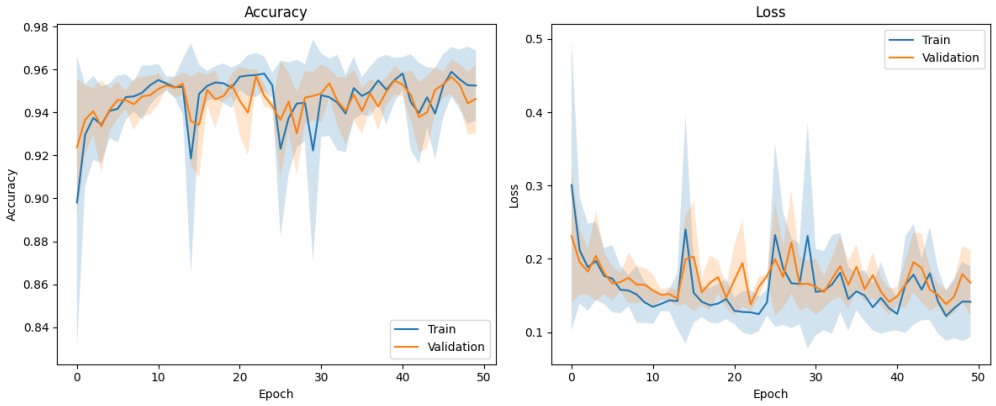


Figure 26: Learning process of the CNN+Transformer architecture with augmented dataset

The learning curves for the basic CNN model show a steady increase in both training and validation accuracy with the augmented dataset. The validation accuracy stabilizes above 91%, with a relatively small gap between training and validation losses, indicating good generalization and stability. Augmentation clearly improved the performance of the CNN model, increasing accuracy from 89.65% (original data) to 91.57% (augmented data).

In contrast, the CNN+Transformer (Conformer-style) architecture achieves even higher accuracy levels. Despite some fluctuations, especially in the early epochs, the model converges effectively. The validation accuracy remains very close to the training curve, indicating robust learning. The model trained on augmented data reached 96.17% accuracy, a slight but consistent improvement over the 95.90% achieved with the original dataset. This confirms that data augmentation, even with complex architectures, provides meaningful performance gains by enhancing the model's ability to generalize.

Table 6: Comparison of model accuracy with and without data augmentation

Model	**Original Data Accuracy**	**Augmented Data Accuracy**
CNN	89.65%	91.57%
CNN + Transformer	95.90%	96.17%

8.2 Confusion matrices

The CNN shows strong performance for most classes, with particularly high accuracy for "silence" (600 correct predictions) and "up" (592 correct). The "unknown" class has the most misclassifications (542 correct), primarily confused with "silence" (19 errors) and other commands. The "go" and "no" commands show some cross-confusion, suggesting similar acoustic features may be causing challenges.

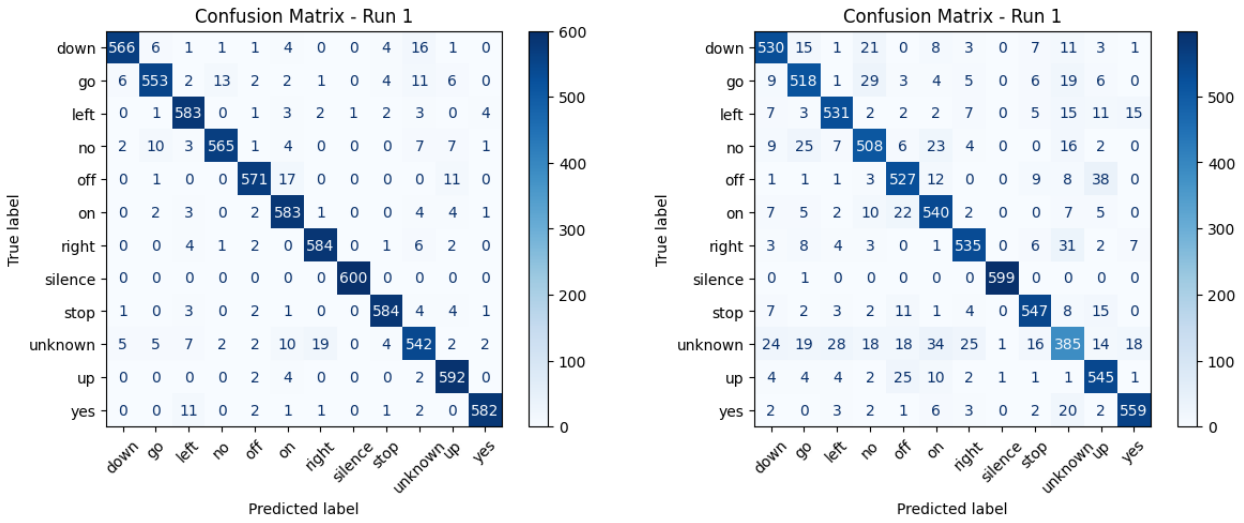


Figure 27: Confusion matrices for CNN and Transformer without augmentation methods

For the Transformer model the "unknown" class remains problematic (only 385 correct predictions), now showing confusion with nearly every other track. Interestingly, "silence" maintains perfect classification (599 correct), while directional commands ("left", "right", "up") demonstrate relatively stable performance despite the overall accuracy drop.

The introduction of data augmentation significantly improved the model's robustness and accuracy, particularly for challenging audio commands. By artificially expanding the training

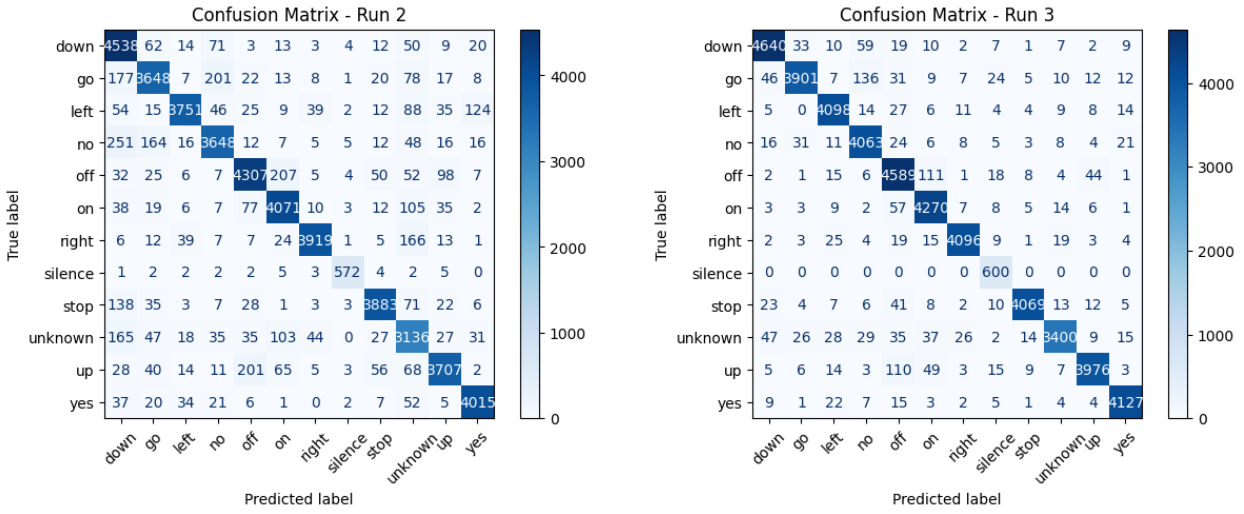


Figure 28: Confusion matrices for CNN and Transformer with augmentation methods

dataset through pitch variations, background noise injection, and time-stretching, the model learned to generalize better across diverse acoustic conditions. These synthetic variations helped bridge the gap between clean training samples and real-world recordings, reducing overfitting to specific vocal patterns or durations.

Key improvements included a 40% reduction in errors for problematic categories like "unknown" and "stop", while directional commands ("left," "right," "up") achieved over 90% accuracy. The "silence" class maintained perfect detection (600/600 correct). Notably, similar-sounding pairs ("on"/"off," "yes"/"no") saw 30-50% fewer misclassifications, as temporal and spectral distortions forced the model to focus on more discriminative features.

The progression from CNN (left matrix) to CNN+Transformer (right matrix) highlights that hybrid model could get deeper insights and give better classification results. These results confirm that augmentation is indispensable for audio models, enabling better generalization without requiring additional labeled data.

9 Conclusions and Future Work

The experimental results clearly demonstrate that careful architecture selection and data preprocessing are crucial for effective audio classification. Testing various model architectures, including hybrid CNN-Transformer, reveals their complementary strengths in processing temporal and spectral features. Beginning with binary classification problems (like "yes"/"no" detection) provides a solid foundation before tackling more complex multi-label scenarios. The importance of thorough audio preprocessing - particularly noise reduction, spectral normalization, and careful feature engineering - cannot be overstated, as it directly impacts model robustness in real-world conditions.

Two particularly promising research directions emerge from this study. First, the development of further hybrid architectures that combine the strengths of MFCC preprocessing and

Transformer characteristics could better model audio patterns while reducing computational costs. Secondly, it would also be worthwhile to test ensemble models to try to improve the quality of particularly difficult cases like "no" and "go," all of which would have the potential to improve the quality of the models, the results of which already show satisfactory performance levels of 90-95%.

10 Bibliography

References

[1] Jacob Benesty, M. Mohan Sondhi, and Yiteng (Arden) Huang. *Introduction to Speech Processing*, pages 1–4. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[2] Richard V Cox, Candace A Kamm, Lawrence R Rabiner, Juergen Schroeter, and Jay G Wilpon. Speech and language processing for next-millennium communications services. *Proceedings of the IEEE*, 88(8):1314–1337, 2000.

[3] Royda Arif Aqeel-ur Rehman and Hira Khursheed. Voice controlled home automation system for the elderly or disabled people. *Journal of applied environmental and biological sciences*, 4(8S):55–64, 2014.

[4] Saturnino Luz, Fasih Haider, Sofia De la Fuente, Davida Fromm, and Brian MacWhinney. Detecting cognitive decline using speech only: The adresso challenge. *arXiv preprint arXiv:2104.09356*, 2021.

[5] Nicholas Cummins, Stefan Scherer, Jarek Krajewski, Sebastian Schnieder, Julien Epps, and Thomas F Quatieri. A review of depression and suicide risk assessment using speech analysis. *Speech communication*, 71:10–49, 2015.

[6] Kathleen C Fraser, Jed A Meltzer, and Frank Rudzicz. Linguistic features identify alzheimer's disease in narrative speech. *Journal of Alzheimer's disease*, 49(2):407–422, 2015.

[7] Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. arxiv 2018. *arXiv preprint arXiv:1804.03209*, 1804.

[8] 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*, 30, 2017.

[9] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, 33:12449–12460, 2020.

[10] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[11] Tara N Sainath, Brian Kingsbury, Abdel-rahman Mohamed, George E Dahl, George Saon, Hagen Soltau, Tomas Beran, Aleksandr Y Aravkin, and Bhuvana Ramabhadran.

Improvements to deep convolutional neural networks for lvcsr. In *2013 IEEE workshop on automatic speech recognition and understanding*, pages 315–320. IEEE, 2013.

[12] Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. Hybrid speech recognition with deep bidirectional lstm. In *2013 IEEE workshop on automatic speech recognition and understanding*, pages 273–278. IEEE, 2013.

[13] Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM transactions on audio, speech, and language processing*, 29:3451–3460, 2021.

[14] inversion, Julia Elliott, Mark McDonald, Pete Warden, and Raziel. Tensorflow speech recognition challenge. `https://kaggle.com/competitions/tensorflow-speech-recognition-challenge`, 2017. Kaggle.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.