Addressing Imbalance in Speech Recognition: Comparative Study of CNN, CRNN, and Autoencoder-SVM Models

Elnur Isgandarov[†]

***Abstract*—The field of speech recognition continues to evolve, driven by the growing demand for intelligent voice-enabled systems in various applications. This paper addresses the challenges associated with improving classification accuracy and robustness in speech recognition tasks. We propose novel architectures and techniques, including Convolutional Recurrent Neural Networks (CRNNs) and an Autoencoder-SVM hybrid approach, to enhance the performance of speech recognition models. By integrating recurrent layers and leveraging latent representations learned by autoencoders, our models aim to mitigate issues such as data imbalance and phonetic ambiguity. We demonstrate the effectiveness of our approach through comprehensive experiments on speech recognition datasets, achieving notable improvements in classification accuracy compared to existing methods. The results highlight the potential of our proposed architectures and techniques to advance the state-of-the-art in speech recognition technology. Furthermore, our findings offer insights for future research directions and potential applications in commercial products, paving the way for more robust and accurate voice-enabled systems.**

***Index Terms*—Speech Recognition, Convolutional Neural Networks, Deep Learning, Recurrent Neural Networks, Autoencoders, Support Vector Machines.**

I. Introduction

Speech recognition plays a crucial role in various applications, from virtual assistants to automated transcription systems. With the increasing demand for accurate and efficient speech recognition technologies, addressing the challenge of class imbalance in speech datasets has become paramount. In this paper, we focus on investigating and comparing the performance of three different models: Convolutional Neural Networks (CNN), Convolutional Recurrent Neural Networks (CRNN), and an Autoencoder-SVM hybrid approach, in the context of speech recognition tasks. By examining the effectiveness of these models, we aim to contribute to the advancement of speech recognition technology by providing insights into the handling of imbalanced speech datasets.

Despite significant advancements in speech recognition technology, class imbalance remains a persistent challenge, particularly in real-world applications where certain words or phrases occur less frequently than others. Previous attempts to address this issue have yielded suboptimal results, highlighting the need for more robust and efficient solutions. By exploring the limitations of existing approaches and identifying the gaps in current research, we strive to offer a comprehensive understanding of the challenges posed by imbalanced speech datasets and propose novel strategies to overcome them.

In this paper, we propose a comparative study of three distinct models for speech recognition tasks: CNN, CRNN, and an Autoencoder-SVM hybrid approach. Our approach involves evaluating the performance of these models on imbalanced speech datasets and analyzing their effectiveness in mitigating class imbalance. We highlight the novelty and relevance of our study by showcasing the unique characteristics and capabilities of each model, and how they contribute to the advancement of speech recognition technology. Additionally, we provide insights into the practical applicability of our findings and their potential implications for future research and development in the field.

This report is structured as follows: In Section II, we provide an overview of the state of the art in speech recognition technology. Section III presents the methodology and experimental setup used in our study. The results and analysis are discussed in Section IV, followed by a discussion of the implications and future directions in Section V. Finally, we conclude our findings and contributions in Section VI.

II. Related Work

Previous research in the field of speech recognition has explored various approaches to address the challenges associated with modeling audio data and extracting meaningful features for classification tasks. Chorowski et al. [1] introduced attention-based models, which have since become a cornerstone in speech recognition architectures. These models demonstrate the importance of capturing long-range dependencies and focusing on relevant parts of the input sequence, leading to improved performance in various speech recognition tasks.

Tang et al. [2] proposed the use of residual networks (ResNets) in convolutional neural networks (CNNs) for speech recognition. By leveraging residual connections, ResNets enable the training of deeper networks while mitigating the vanishing gradient problem. This approach has shown to improve the overall performance of CNNs in speech recognition tasks, particularly in handling complex and hierarchical features present in audio data.

Despite these advancements, existing approaches often struggle with data imbalance and phonetic ambiguity, which can hinder classification accuracy, particularly in distinguishing words with similar phonemes. Luo et al. [3] presented

[†]email: {elnur.isgandarov}@studenti.unipd.it

a speech augmentation-based unsupervised learning approach for keyword spotting, contributing to the exploration of unsupervised learning techniques in speech recognition tasks. Their work highlights the potential of unsupervised learning methods in improving the robustness and generalization capabilities of speech recognition models.

The work presented in this paper builds upon these findings by investigating the effectiveness of Convolutional Recurrent Neural Networks (CRNNs) and an Autoencoder-SVM hybrid approach for speech recognition tasks. By integrating recurrent layers and leveraging the latent representations learned by autoencoders, our models aim to address these challenges and achieve more robust performance in speech recognition tasks.

In summary, while previous research has made significant strides in advancing speech recognition technologies, there remain important challenges to be addressed. The contributions of this paper extend upon existing work by exploring novel architectures and techniques to improve classification accuracy and address the limitations of current approaches in speech recognition.

III. Processing Pipeline

The processing pipeline for our keyword spotting task involves several stages to enhance the quality of the input data and extract meaningful features for model training. Initially, the raw audio data from the `speech_commands_v0.02` dataset, sampled at 16,000 Hz and one second in duration, undergoes preprocessing steps to improve audio quality. This includes the removal of reverberation and noise reduction techniques, employing both spectral subtraction and simple thresholding to mitigate background noise and sudden spikes effectively.

As shown in Figure 2, the data processing pipeline consists of several stages. Following data preprocessing, feature extraction is performed. This process involves several substeps, beginning with trimming the audio samples to ensure consistency. Subsequently, the audio signals are converted into Mel spectrograms and transformed into a logarithmic scale to better capture human auditory perception. Normalization is applied to standardize the feature values, followed by the computation of Mel Frequency Cepstral Coefficients (MFCCs) and the calculation of delta coefficients to capture temporal dynamics.

To ensure uniform input size for model training, a special normalization step is applied to crop each input block to a size of 101 MFCCs. Finally, the preprocessed data is fed into various neural network architectures, including Convolutional Neural Networks (CNN), Recurrent Convolutional Neural Networks (RCNN), and an Autoencoder-Support Vector Machine (AE-SVM) hybrid model. These models are trained and tested on the processed data to perform the keyword spotting task effectively.

IV. Signals and Features

In this section, we describe the signals used in our study, the preprocessing steps applied to them, the feature extraction

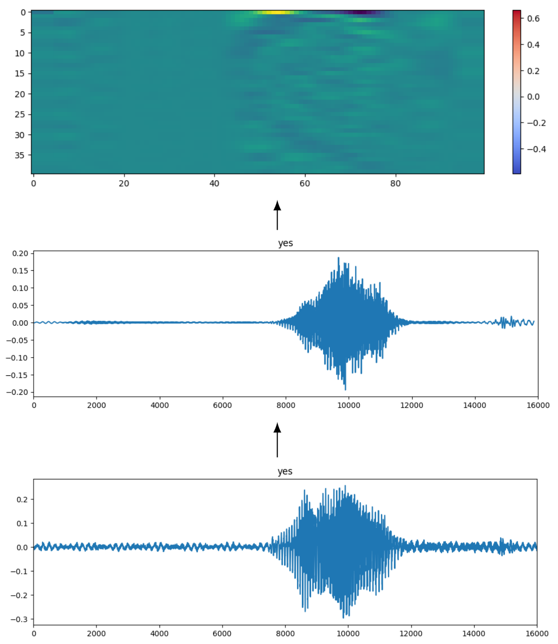

Fig. 1: Data processing steps.

process, and how the dataset was split for training, validation, and testing purposes.

A. *Measurement Setup*

The dataset utilized in this study is the Speech Commands Dataset: A Dataset for Limited-Vocabulary Speech Recognition [4]. The dataset consists of 1-second-long WAV files with a sampling rate of 16,000 Hz. Each audio clip represents a spoken word from a predefined vocabulary. To address the imbalance in audio clip lengths due to background noise, the longer audio clips were divided into 1-second chunks. Additionally, augmentation techniques were applied to increase the occurrence of background noise instances. These techniques included random mixing of time shifting, speed tuning, background noise mixing, and volume tuning. After augmentation, all audio clips were resampled to ensure consistency in length and sampling rate.

B. *Signal Preprocessing*

Prior to feature extraction, the audio signals underwent preprocessing to enhance their quality and robustness for model training. A threshold-based approach was used to identify low-quality audio clips containing background noise and spikes. Despite potentially degrading model performance, these clips were retained to improve model robustness. Data enhancement techniques were then applied selectively to mitigate the impact of background noise and improve overall signal quality. Finding the optimal thresholds and applying appropriate enhancements proved critical for achieving optimal performance in real-world scenarios. However, for the reported results, no enhanced data was used in training.

C. Feature Extraction

The feature extraction process involved several steps aimed at capturing relevant information from the preprocessed audio signals:

- **Trimming**: The audio clips were trimmed to ensure consistency in length.
- **Conversion to Mel Spectrogram**: The audio signals were converted into Mel spectrograms to represent their frequency content.
- **Conversion to Log Scale (dB)**: Inspired by human hearing, the Mel spectrograms were converted to a logarithmic scale to better reflect perceptual differences.
- **Normalization**: The feature values were normalized to ensure uniformity across the dataset.
- **Calculation of Mel Frequency Cepstral Coefficients (MFCCs)**: MFCCs were computed to capture spectral features relevant for speech recognition. The calculation involves several steps, including mel filtering, discrete cosine transform (DCT), and feature extraction.
- **Calculation of Delta**: Delta coefficients were calculated to capture temporal dynamics and emphasize the temporal information inherent in speech signals.

D. Dataset Splitting

The Speech Commands dataset was split into training, validation, and test sets according to predefined guidelines. The dataset provided explicit instructions on which files should be used for each set. The test set was constructed to have a balanced distribution of files for each class, while the training and validation sets maintained the same distribution. To address the low occurrence in the *silence* (background noise) class, additional instances were augmented and distributed across all sets while preserving set balance. Specifically, for the Autoencoder-SVM model, a bash script was utilized to create a more balanced dataset for training and validation by reducing the occurrence of the *unknown* class while maintaining word-wise balance within the class. This adjustment was necessary as the Autoencoder-SVM model is not robust to imbalanced data and resulted with having twice less input data for the model.

V. Learning Framework

In the experimentation phase, a common set of configurations and evaluation metrics were applied across all models. Specifically, a batch size of 64 was utilized during training. To prevent overfitting, an early stop callback mechanism was employed for all models. The Adam optimizer, known for its efficiency in handling sparse gradients and noisy data, was chosen as the optimization algorithm.

Across all experiments, the dataset comprised 12 distinct classes and the classification performance of each model was evaluated using various metrics, including Top-One error accuracy, precision, recall, and F1 score. Additionally, a confusion matrix was generated for each model, providing insights into its classification behavior and potential areas for improvement. These evaluation metrics collectively offered a comprehensive assessment of the models' capabilities in accurately identifying spoken keywords.

A. Convolutional Neural Network (CNN)

The CNN architecture (table 1) utilized in this study follows a design inspired by the *cnn-trad-fpool3* model, as proposed in the literature [5]. The model, which has convolution kernels with a larger dimension in time than in spatial dimensions, is tailored for the keyword spotting task and consists of several convolutional and pooling layers.

The CNN architecture employed in this study is designed for keyword spotting tasks, comprising multiple convolutional and pooling layers. The model starts with a convolutional layer with 32 filters and a kernel size of (20, 8), followed by max-pooling and batch normalization layers to capture low-level features and standardize activations. Subsequently, another convolutional layer with 64 filters and a kernel size of (10, 6) is applied, followed by max-pooling and batch normalization to extract higher-level features. Finally, a third convolutional layer with 128 filters and a kernel size of (5, 3) further enhances feature representation, followed by max-pooling and batch normalization. The output of the convolutional layers is flattened and passed through a dropout layer with a dropout rate of 0.2 to mitigate overfitting. A fully connected layer with softmax activation is then used to output class probabilities. During training, class weights are calculated to address data imbalance, and early stopping with a patience of 4 is employed to prevent overfitting.

TABLE 1: CNN Architecture

Layer Type	**Details**
Conv2D	32 filters, kernel size (20, 8) MaxPooling2D (pool size (2, 2)) BatchNormalization
Conv2D	64 filters, kernel size (10, 6) MaxPooling2D (pool size (2, 2)) BatchNormalization
Conv2D	128 filters, kernel size (5, 3) MaxPooling2D (pool size (2, 2)) BatchNormalization
Flatten	-
Dropout	Dropout rate 0.2
Dense	Output layer (softmax activation)

B. Convolutional Recurrent Neural Network (CRNN)

The CRNN architecture (table 2) seamlessly combines convolutional and recurrent layers to effectively capture spatial and temporal dependencies in the input data. It starts with one-dimensional convolutional layers to extract low-level features, followed by max-pooling and batch normalization for downsampling and normalization. Subsequent convolutional layers extract higher-level features while maintaining balance through pooling and normalization.

A crucial component is the Gated Recurrent Unit (GRU) layer, which processes sequential input data, capturing temporal patterns over time. Dropout regularization is employed to prevent overfitting, ensuring robust model performance. The output is flattened and passed through a fully connected layer for classification.

During training, the model utilizes the same hyperparameters as the CNN, including early stopping and class weights to handle data imbalance. This comprehensive architecture offers a robust solution for keyword spotting tasks, leveraging both spatial and temporal information effectively.

TABLE 2: CRNN Architecture

Layer Type	Details
Conv1D	32 filters, kernel size 20, strides 1 MaxPooling1D (pool size 2) BatchNormalization
Conv1D	64 filters, kernel size 10, strides 1 MaxPooling1D (pool size 2) BatchNormalization
Conv1D	128 filters, kernel size 5, strides 1 MaxPooling1D (pool size 2) BatchNormalization
Conv1D	256 filters, kernel size 3, strides 1 MaxPooling1D (pool size 2) BatchNormalization
GRU	128 units, return sequences
Dropout	Dropout rate 0.2
Flatten	-
Dense	Output layer (softmax activation)

C. Autoencoder-SVM (AE-SVM)

1) Feature Extraction with Autoencoders: The autoencoder architecture (table 3) includes a bottleneck size of 128, which allows for the extraction of compact latent representations from the input data. The autoencoder is trained using the Adamax optimizer with a mean squared error (MSE) loss function for 50 epochs. Despite being trained on a relatively smaller portion of the dataset due to data imbalance issues, the autoencoder achieves a satisfactory MSE of 0.001 on the test data. Increasing the bottleneck size improves the model's learning capacity, although this improvement is constrained by the availability of data.

2) Classification with SVM: To determine the optimal hyperparameters for the SVM classifier, GridSearch is employed on a subset of the training data. The grid search spans over a range of values for the regularization parameter (C) and kernel coefficient (gamma), denoted as C' and gamma'. The values of C' tested include 0.1, 1, 10, 100, 1000, while gamma' is explored within 1, 0.1, 0.01, 0.001, 0.0001. By iteratively evaluating various combinations of these parameters, the SVM model's discrimination capabilities are systematically assessed. Among the candidate configurations, the best-performing model is identified based on its ability to achieve balanced classification performance.

For the selected model, the chosen SVM parameters include C=1, gamma=0.001, and a radial basis function (RBF) kernel. These parameters are instrumental in achieving balanced classification performance by appropriately adjusting the trade-off between model complexity and generalization, as well as capturing the non-linear relationships in the data. It is important to note that the selected hyperparameters are specific to each distribution of the dataset and require separate optimization for different autoencoder models.

TABLE 3: Autoencoder Architecture

Encoder	
Layer Type	**Details**
Conv2D	Kernel: (3, 3), Filters: 32 Activation: ELU
Max Pooling	Pooling Kernel: (2, 2)
Conv2D	Kernel: (3, 3), Filters: 64 Activation: ELU
Max Pooling	Pooling Kernel: (2, 2)
Flatten	
Dropout (Rate: 0.2)	
Dense	Units: 128 Activation: None
Decoder	
Layer Type	**Details**
Dense	Units: 26 * 10 * 128 Activation: ELU
Reshape	Target Shape: (26, 10, 128)
Conv2DTranspose	Kernel: (3, 3), Filters: 64, Strides: (2, 2) Activation: ELU
Conv2DTranspose	Kernel: (3, 3), Filters: 32, Strides: (2, 2) Activation: ELU
Conv2DTranspose	Kernel: (3, 3), Filters: 1, Strides: (1, 1) Activation: Sigmoid

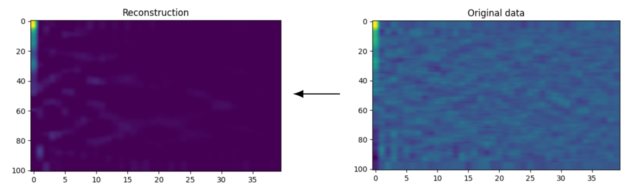

Fig. 2: Reconstruction

VI. Results

In this section, we present the performance and findings of our trained models. We begin by discussing the performance metrics of each model, followed by an analysis of their training characteristics and memory usage.

A. Performance Metrics

We evaluated the performance of our models using standard metrics such as accuracy, precision, recall, and F1-score. Among the models, the CRNN architecture demonstrated superior performance across all metrics.(table 4) Despite its longer training time compared to the CNN model, CRNN exhibited more stable loss and accuracy curves, reliably converging by the 20th epoch. In contrast, the CNN model achieved convergence by the 10th epoch but with slightly lower overall performance metrics. The AE-SVM model, although effective, exhibited reduced accuracy and F1-score compared to the CRNN and CNN models. In the imbalanced data scenario, the application of data sampling methods had varying effects on AE's bias towards the unknown class. Not all methods improved the results; some even worsened them. However, after applying SMOTEENN [6], we observed a notable increase in accuracy from approximately 65% to around 78%, accompanied by a corresponding improvement in the F1-score by approximately 10%. This highlights the effectiveness of data sampling techniques in addressing class imbalance and enhancing the overall performance of the models.

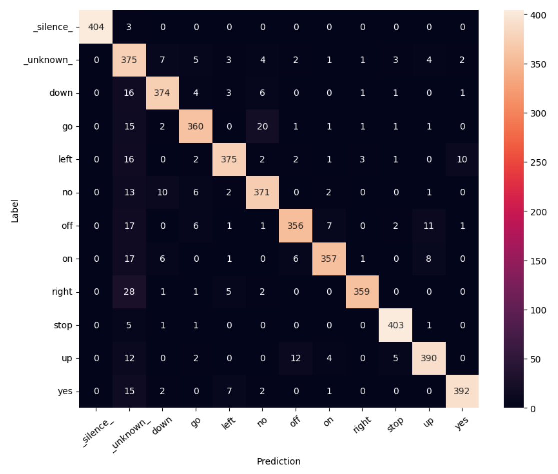

Fig. 3: Confusion Matrix (CRNN model)

B. Training Characteristics

We observed distinct training characteristics for each model. The CNN model, with its simpler architecture, required less training time and memory compared to CRNN and Autoencoder-SVM. However, the CRNN model, leveraging both convolutional and recurrent layers, exhibited longer training times but achieved higher overall performance. The Autoencoder-SVM model, utilizing an autoencoder for feature extraction followed by SVM classification, demonstrated competitive performance but faced challenges with data imbalance.

C. Memory Usage

An analysis of memory usage revealed varying requirements among the models. The CNN model had the smallest memory footprint, making it suitable for resource-constrained environments. The CRNN model required additional memory due to the inclusion of recurrent layers but still remained within acceptable limits. In contrast, the Autoencoder-SVM model, despite its effectiveness, consumed the most memory. Notably, only the encoder's memory usage was considered for comparison, as it is responsible for feature extraction in the AE-SVM model.

TABLE 4: Model Performance Metrics

	CRNN	**CNN**	**AE-SVM**
Top-One Error Accuracy (%)	92.39	91.76	83.16
Precision (%)	93.03	92.09	83.75
Recall (%)	92.37	91.75	83.13
Fscore (%)	92.55	91.80	83.30

VII. Concluding Remarks

Despite the promising performance of our models, several challenges remain. The difficulty in differentiating words with similar phonemes highlights the need for further investigation into phonetic representation and classification techniques.(fig. 3) Future directions may include exploring alternative feature extraction methods tailored to capturing phonetic nuances, as well as investigating the application of attention-based models and residual networks, as demonstrated in prior research [1], [2]. Additionally, the development of cost-sensitive autoencoder [7] models may offer a potential solution to address data imbalance issues and further improve classification accuracy. Additionally, investigating the application of recurrent layers within the autoencoder framework may also offer further improvements in classification accuracy.

Throughout this project, I encountered numerous challenges and gained valuable insights into speech recognition and machine learning. A significant difficulty was establishing an appropriate baseline due to the unique characteristics of the dataset, necessitating a deep understanding of data distribution and preprocessing. Despite initial hurdles, I developed proficiency in handling various types of audio data, including raw signals and 1D representations. Another challenge was interpreting the behaviors of different models, particularly with the autoencoder architecture, and identifying areas for improvement, especially related to data distribution issues. Additionally, grappling with TensorFlow-specific issues consumed considerable time, emphasizing the importance of mastering the chosen tools and platforms.

References

[1] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, “Attention-based models for speech recognition,” 2015.

[2] R. Tang and J. Lin, “Deep residual learning for small-footprint keyword spotting,” 2018.

[3] J. Luo, J. Wang, N. Cheng, H. Tang, and J. Xiao, “Speech augmentation based unsupervised learning for keyword spotting,” 2022.

[4] P. Warden, “Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition,” *ArXiv e-prints*, Apr. 2018.

[5] T. Sainath and C. Parada, “Convolutional neural networks for small-footprint keyword spotting,” in *Interspeech*, 2015.

[6] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “Smoteenn: A hybrid approach to alleviating the class imbalance problem,” *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[7] A. Telikani and A. H. Gandomi, “Cost-sensitive stacked auto-encoders for intrusion detection in the internet of things,” *Internet of Things*, vol. 14, p. 100122, 2021.