Speech Understanding - Programming Assignment 1

Bilas Chandra Tarafdar (M24CSA008)

February 2, 2025

1 Question 1: Analysis of a Speech Processing Task

1.1 Task Description and Importance

The selected task for analysis is **Code-Switching (CS) Automatic Speech Recognition (ASR)** and **Spoken Language Identification (LID)**. Code-switching ASR models are designed to transcribe speech containing alternating languages within a conversation. This is an important challenge in multilingual societies where speakers frequently switch between languages mid-sentence.

Real-world applications of code-switching ASR include:

- **Multilingual Voice Assistants**: Enhancing voice assistants like Siri, Google Assistant, and Alexa to understand and process code-switching speech.
- **Transcription Services**: Improving automated transcription accuracy for multilingual broadcasts, meetings, and academic research.
- **Healthcare and Customer Support**: Assisting professionals in multilingual interactions by providing real-time, high-accuracy transcriptions.
- **Language Learning Tools**: Enabling better feedback and assessments for students learning multiple languages simultaneously.

1.2 State-of-the-Art Models

Several approaches exist for handling code-switching ASR, with recent advancements integrating **deep learning-based architectures** such as Transformer and Conformer models.

1.2.1 Existing Models and Approaches

- **Monolingual ASR Models**: Traditional ASR systems rely on monolingual models, which perform poorly on code-switched speech due to lack of exposure to mixed-language training data.
- **Bilingual and Multilingual ASR**: Recent advancements train ASR models on mixed-language data, allowing for better adaptation to code-switching scenarios.
- **Concatenated Tokenizer Approach (Proposed in Paper)**: This novel method assigns separate token ID spaces to each language, allowing the model to retain language identity at a token level while utilizing pre-existing monolingual tokenizers.
- **Synthetic Data Generation**: The paper introduces an efficient pipeline for generating synthetic code-switching datasets using monolingual data sources, ensuring that models are exposed to diverse linguistic transitions during training.

1.2.2 Strengths and Limitations

1.3 Performance Metrics

ASR performance is evaluated using the following metrics:

- **Word Error Rate (WER)**: Measures transcription accuracy by comparing recognized text to ground truth.
- **Spoken Language Identification Accuracy**: Evaluates how well the model identifies the language of each token or utterance.
- **Dataset-Specific Evaluations**: Performance is analyzed on both synthetic datasets and real-world corpora like the Miami Bangor (English-Spanish) and MUCS 2021 (English-Hindi) datasets.

Model Type	Strengths	Limitations
Monolingual ASR	High accuracy for a single language	Fails in code-switching scenarios
Bilingual/Multilingual ASR	Supports multiple languages	May lose fine-grained linguistic distinctions
Concatenated Tokenizer (Proposed)	Retains language identity at the token level	Needs careful dataset balancing for optimal performance
Synthetic Data Generation	Enables scalable dataset creation	May not fully capture real-world code-switching complexity

Table 1: Comparison of Different ASR Approaches

1.3.1 Findings from the Paper

- The **concatenated tokenizer approach** achieved competitive ASR performance while providing superior LID capabilities.
- **98%+ accuracy** was observed for spoken language identification on out-of-distribution datasets.
- **WER for CS datasets**: Performance on the Miami Bangor (50-53% WER) and MUCS 2021 (28-30% WER) datasets showed that code-switching models significantly outperformed bilingual models for real-world CS scenarios.

1.4 Open Problems and Future Directions

Despite advancements in code-switching ASR, several challenges remain:

1.4.1 Challenges

- **Data Scarcity**: Real-world code-switching datasets are limited, requiring synthetic data generation.
- **Language-Specific Variations**: Code-switching behavior varies across speakers, dialects, and contexts.
- **Handling Low-Resource Languages**: Many languages lack large monolingual datasets, making multilingual ASR challenging.

- **Computational Efficiency**: Training large-scale multilingual ASR models requires substantial computational resources.

1.4.2 Potential Research Directions

- **Improved Synthetic Data Generation**: Enhancing methods for more realistic data augmentation.
- **Few-Shot Learning for CS ASR**: Exploring models that require minimal code-switching training data.
- **Multi-Modal Approaches**: Integrating visual and contextual cues for better ASR performance.
- **Cross-Lingual Transfer Learning**: Leveraging pre-trained monolingual models for multilingual adaptation.

1.5 Conclusion

The paper presents a novel **concatenated tokenizer** and **synthetic data generation** approach for training **code-switching ASR** models. The results demonstrate that these methods provide **state-of-the-art** performance on code-switching datasets while maintaining monolingual ASR accuracy. Future research should focus on enhancing data availability, improving low-resource language support, and optimizing multilingual model efficiency.

2 Question 2: Experimenting with Spectrograms and Windowing Techniques

2.1 UrbanSound8k Dataset

The **UrbanSound8K** dataset comprises **8732 labeled audio excerpts** ($\leq$4 seconds) of urban sounds, categorized into **10 classes** such as air_conditioner, car_horn, and dog_bark. This dataset is essential for sound recognition tasks, particularly in enhancing speech understanding systems.

Relevance to Speech Understanding and Windowing Methods

- **Sound Classification:** It aids in training models to distinguish between various urban sounds, which is crucial for filtering background noise in speech recognition.
- **Feature Extraction:** The dataset allows for the extraction of key audio features (e.g., MFCCs), improving the accuracy of speech processing algorithms.
- **Windowing Techniques:** The short duration of audio clips facilitates the use of windowing methods, essential for analyzing temporal aspects of speech.
- **Data Augmentation:** Structured metadata supports data augmentation, enhancing model robustness against diverse acoustic conditions.

2.2 Windowing Techniques and Spectrograms

In signal processing, windowing techniques are crucial for analyzing signals in the frequency domain. Three commonly used windows are the **Hann**, **Hamming**, and **Rectangular** windows.

- **Hann Window**:
  - Smooth transition to zero, minimizing discontinuities. Offers a good balance between main lobe width and side lobe suppression.

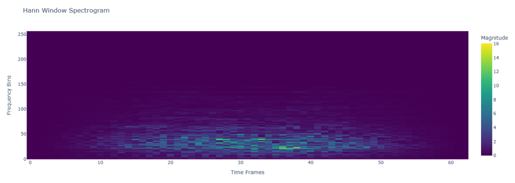

Figure 1: Hann Window Spectrogram : Least spectral leakage

- **Hamming Window**:
  - Similar to the Hann window but does not reach zero at the endpoints, providing better side lobe suppression at the cost of a slightly wider main lobe.

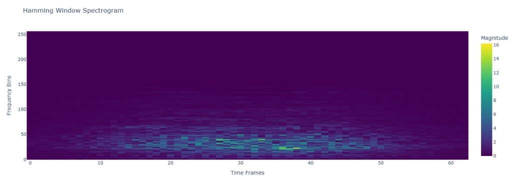

Figure 2: Hamming Window Spectrogram : Moderate spectral leakage

- **Rectangular Window**:
  - Abrupt transition from 1 to 0, resulting in a narrow main lobe but higher side lobes, leading to spectral leakage.

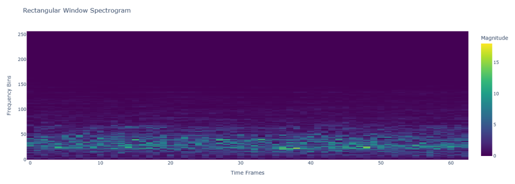

Figure 3: Rectangular Window Spectrogram : Most spectral leakage

2.3 Classification Experiment

2.4 AudioCNN Architecture and Results

The `AudioCNN` model consists of two convolutional layers followed by max-pooling, dynamic fully connected layers, and dropout for regularization. The model performs classification with a final fully connected layer.

Results: All windowing methods (Rectangular, Hamming, and Hann) yielded similar accuracy around 54.5%. The Rectangular window performed best, followed by Hamming, with Hann performing the least well.

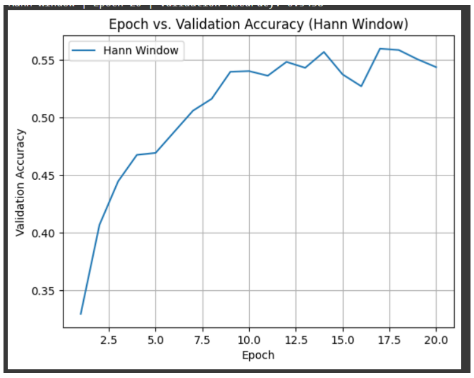

Figure 4: Hann Window Performance

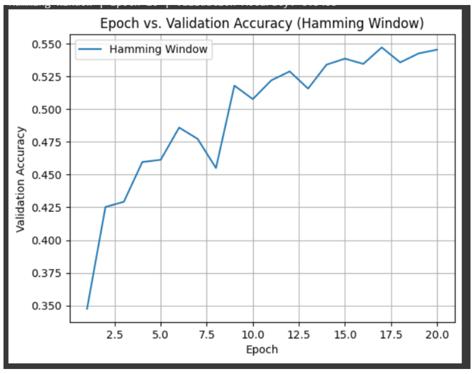

Figure 5: Hamming Window Performance

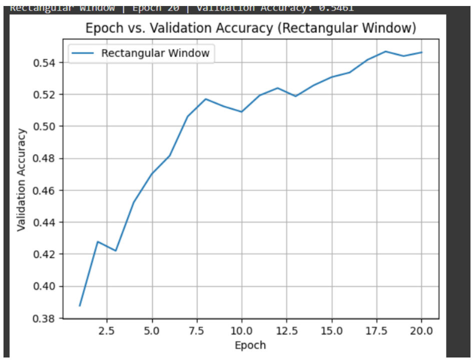

Figure 6: Rectangular Window Performance

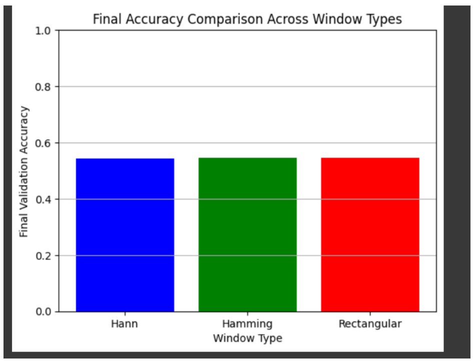

Figure 7: Comparison of Windowing Techniques

2.5 Task B: Genre-based Spectrogram Analysis

The spectrograms of different music genres exhibit distinct characteristics, reflecting the unique sound structures and production techniques used in each genre:

- **Classical:**
  - The spectrogram of classical music tends to show smooth, continuous frequency distributions with clear tonal separation. The presence of strings, wind, and brass instruments results in broad, low-amplitude side lobes and a focused energy in the lower to mid-frequency range.

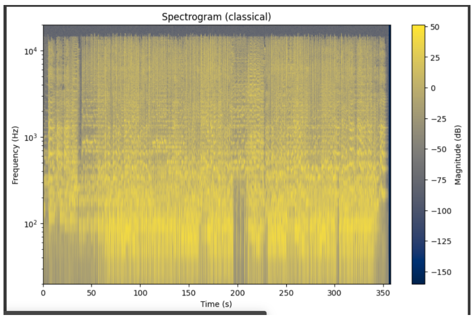

Figure 8: Classical Music Spectrogram

- **Pop:**
  - Pop music typically shows a higher concentration of energy in the mid and high frequencies. Spectrograms display quick transitions between frequencies with varying intensity, with noticeable periodicity due to beat-driven structure and synthetic sounds from electronic instruments.

- **Electronic/Rock:**
  - Electronic and rock music tend to exhibit sharp frequency bursts with a broad range of harmonics and distortion, especially in the high frequencies. In electronic music, there is often a higher degree of periodicity, with rhythmic basslines and heavy use of synthesized effects.

- **Metal:**

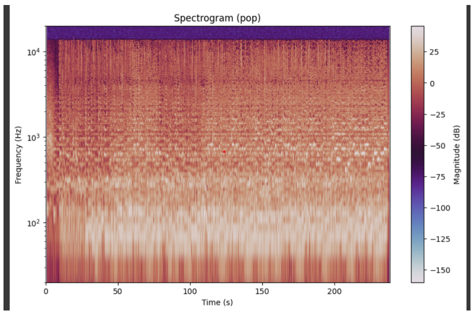

Figure 9: Pop Music Spectrogram

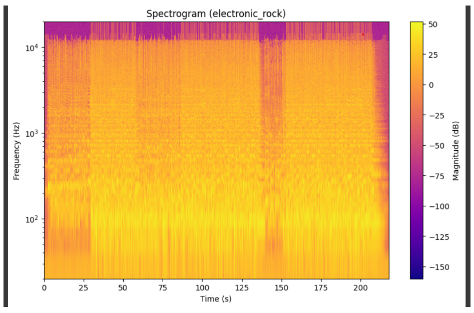

Figure 10: Electronic/Rock Music Spectrogram

– Metal music spectrograms show intense, chaotic frequency patterns with a high presence of percussive elements and distorted guitar sounds. These spectrograms typically show broad energy distributions across the frequency spectrum, especially in the mid and high frequencies, with less tonal coherence compared to classical music.

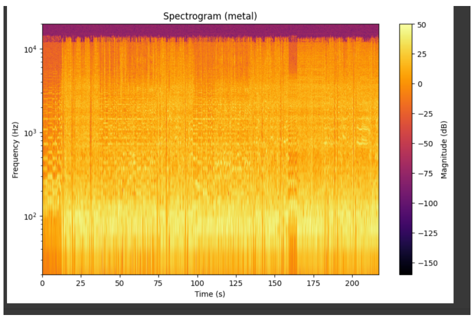

Figure 11: Metal Music Spectrogram

3 GitHub Repository

The code and resources for this project are available at: GitHub Repository

4 References

References

[1] NVIDIA, *NVIDIA NeMo: A toolkit for conversational AI*, Available: `https://github.com/NVIDIA/NeMo`

[2] K. Dhawan, D. Rekesh, and B. Ginsburg, *Unified model for code-switching speech recognition and language identification based on a concatenated tokenizer*, arXiv preprint arXiv:2306.08753, 2023. Available: `https://arxiv.org/abs/2306.08753`

[3] PyTorch Team, *PyTorch Audio Documentation*, Available: `https://pytorch.org/audio/stable/index.html`

[4] J. Salamon, C. Jacoby, and J. P. Bello, *UrbanSound8K: A dataset for urban sound classification*, In Proceedings of the 23rd ACM International Conference on Multimedia, 2014. Available: `https://urbansounddataset.weebly.com/urbansound8k.html`