

CLOUD COMPUTING

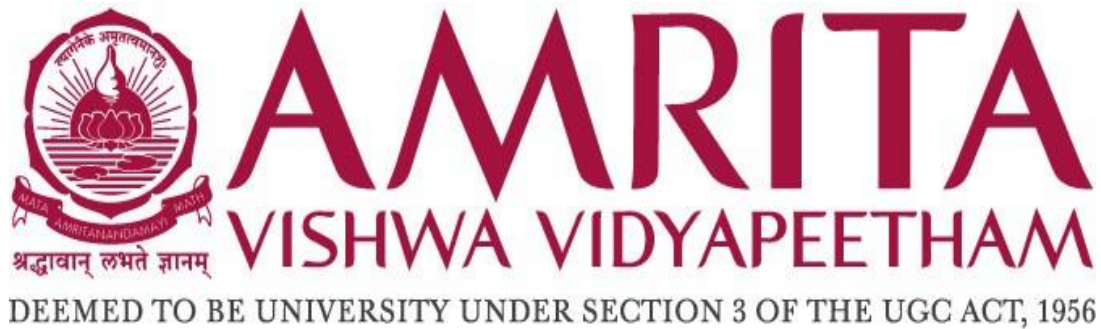
A Project Report
SUBMITTED BY GROUP 10

Poojitha Sai Manikandan - 22041

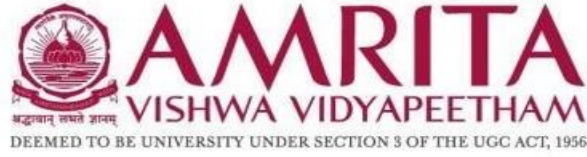
Rasha Sharma - 22043

Siddhaarth S - 22051

Subhashini Sudhakar - 22061



Centre for Computational Engineering and Networking
AMRITA SCHOOL OF ENGINEERING
AMRITA VISHWA VIDYAPEETHAM
COIMBATORE - 641 112 (INDIA)
February - 2023



BONAFIDE CERTIFICATE

This is to certify that the report entitled Dysarthria Speech Intelligibility Application submitted by Poojitha Sai Manikandan (CB.EN.U4AIE22041), Rasha Sharma (CB.EN.U4AIE22043), Siddhaarth S. (CB.EN.U4AIE22051), and Subhashini Sudhakar (CB.EN.U4AIE22061) of Batch A of the year 2022-2026 for the award of the Degree of Bachelor of Technology in the “CSE(AI)” is a Bonafide record of the work carried out by them under our guidance and supervision at Amrita School of Artificial Intelligence, Coimbatore.

Prajisha Sheejith
Project Guide

Dr. K.P. Soman
Dean, School of AI

Submitted for the university examination held on 12th November, 2024

DECLARATION

We solemnly declare that the Cloud Computing project, Dysarthric speech intelligibility application implementation based on our own work carried out during the course of our study under the supervision of MS. Prajisha Sheejith, Assistant Professor, CEN, and has not formed the basis for the award of any other degree or diploma, in this or any other Institution or University. In keeping with the ethical practice in reporting scientific information, due acknowledgement has been made wherever the findings of others have been cited.

Place: Ettimadai
Date: 12-11-2024

Contents

- 1 Introduction 8
- 2 Methodology 8
 - 2.1 Dataset 8
 - 2.2 Model..... 8
 - 2.3 Jarvis Labs 9
 - 2.4 Streamlit..... 9
 - 2.5 FastAPI..... 10
 - 2.6 Deployment..... 10
- 3 Results 11
- 4 Discussion 14
- 5 Conclusion..... 14
- 6 References..... 15

List of Figures

Figure 1 The Streamlit App in our GitHub Repository.....	11
Figure 2 Our Home Page	11
Figure 3 About Page	12
Figure 4 Information Page.....	12
Figure 5 Useful Links Page	13
Figure 6 Page to Record and Translate	13
Figure 7 Transcribed Word for 'Artist'	14

Acknowledgement

This project has been possible due to the sincere and dedicated efforts of many. First, we would like to thank the department and our professors for giving us the opportunity to get involved in a project and express our skill. We thank our Cloud Computing teacher, Ms Prajisha Sheejith, for her guidance and support without which this project would have been impossible.

Last but not least; we thank our parents and our classmates who encouraged us throughout the project.

Abstract

Dysarthria is a motor speech disorder caused by damage to the nervous system, leading to impaired control of the muscles involved in speech production. This results in slurred or unclear speech, making it challenging for listeners to understand. Recent ASR systems have benefited from readily available pretrained models such as wav2vec2 to improve the recognition performance. Speaker adaptation using fMLLR and xvectors have provided major gains for dysarthric speech with very little adaptation data. In our application we perform ASR on dysarthric speech using wav2vec2 embeddings.

To ensure scalability, accessibility, and real-time processing, our solution is developed and deployed using Streamlit after being trained on the cloud-based platform JarvisLabs, on their GPUs. Streamlit is a lightweight and efficient framework for creating interactive web interfaces. provides a easy to use cloud platform that enables seamless deployment, management, and scaling of web applications. By utilizing the cloud, our system can handle large volumes of speech data, perform intensive computations, and deliver enhanced speech outputs to users in real time. This cloud-based approach not only improves the performance and reliability of our speech enhancement system but also makes it easily accessible to users across different regions.

1 Introduction

Dysarthria is a motor speech disorder caused by damage to the nervous system, resulting in impaired control of the muscles involved in speech production, such as the tongue and vocal cords. This condition often leads to slurred or unclear speech, making it difficult for listeners to understand. Our application aims to enhance the clarity and comprehensibility of dysarthric speech by utilizing finetuned word2vec embedding to convert the audio into text.

In this project, this dynamic web application was developed and hosted on the web using Streamlit, a lightweight and efficient framework for creating interactive web interfaces. The application uses the computational resources of Jarvis Labs, which provides a scalable and efficient backend environment for seamless performance.

To bridge the gap between the user interface and backend functionalities, FastAPI was integrated, ensuring fast and reliable API communication. This approach combines the simplicity of Streamlit for frontend development, the computational power of Jarvis Labs for backend processing, and the high-performance capabilities of FastAPI to handle API requests efficiently.

This report delves into the design, implementation, and deployment of the web application, showcasing how modern tools can be harnessed to create interactive and scalable web solutions.

2 Methodology

2.1 Dataset

The dataset that we are using for training is UAspeech [5] This database consist of dysarthric speech produced by 19 speakers with cerebral palsy. Speech materials consist of 765 isolated words per speaker consists of 30 repetitions of 46 isolated words (10 digits, 26 alphabet letters, and 10 'control' words) and 35 words from the Grandfather passage produced by each of six individuals with cerebral palsy. This database includes one normal speaker's production of 15 repetitions of the same materials.

2.2 Model

The wav2vec2 adapter module using supervised speaker adaptive features "fMLLR" and unsupervised speaker vectors "xvectors". These are described as follows:

fMLLR: The fMLLR features are obtained from the MLLR transforms trained with supervised adaptation data. Conventional fMLLR transformation is applied over MFCC or filterbank features. we extract input features for computing fMLLR transforms from the output (1024 dims) of the final block EB in by forward propagating the raw waveforms as wav2vec2 input. We

arrived to the usage of wav2vec2 based features for fMLLR computation and its integration at the Bth MHA block empirically. The collected 1024 dim. wav2vec2 features are transformed using LDA followed by GMM-HMM based training. L211 is used for LDA+MLLT+SAT training and speaker dependent fMLLR transforms computation to get 40 dimensional fMLLR features **F**.

XVectors: Xvectors **G** are extracted from MFCC features using a separately trained xvector extractor. Since the xvectors are not directly extracted from wav2vec2 features, we integrate the xvectors with the output of the intermediate block, here b=2 is decided empirically.

Finetuning with wav2vec2 adapter involves two stages: 1) the model parameters within the green and pink block are initially updated by fixing the rest of parameters. This first update helps the wav2vec2 encoder establish connections with the auxiliary information. 2) The whole model is then finetuned till convergence.

2.3 Jarvis Labs

Jarvis Labs is a cloud-based platform designed to provide computational resources for various machine-learning and data science tasks. For this project, Jarvis Labs was utilized to train the Convolutional Neural Network (CNN) model. The platform's accessible GPU-based infrastructure enabled efficient training of the CNN, significantly reducing the time required compared to traditional CPU systems.

The trained model was then integrated into the web application, allowing users to interact with it seamlessly through a simple interface. Jarvis Labs' flexibility in scaling resources ensured that the model training could be conducted without interruptions or resource constraints. This made it an ideal choice for deploying machine learning workflows, from training to inference.

By using Jarvis Labs, the project achieved a streamlined development process, ensuring that the CNN model was trained effectively and ready for deployment as part of the web application.

2.4 Streamlit

Streamlit is an open-source Python framework designed to create interactive and user-friendly web applications with minimal effort. For this project, Streamlit was used to build and host the web application that incorporates the trained Convolutional Neural Network (CNN) model. Its simplicity and compatibility with Python made it a natural choice for the development process.

Streamlit seamlessly integrates with Python code, allowing developers to quickly transform scripts into fully functional web applications without requiring extensive knowledge of frontend technologies. By leveraging Streamlit, the Python program for the CNN model could be directly embedded into the web interface, ensuring a smooth connection between the application logic and user interaction.

The framework also supports real-time interactivity, enabling users to input data and receive model predictions instantly. This made it ideal for showcasing the trained CNN model in a web-based format, offering both functionality and ease of use.

2.5 FastAPI

FastAPI is a high-performance web framework for building APIs with Python. In this project, it was used to connect the trained Convolutional Neural Network (CNN) model with the web application built using Streamlit. FastAPI's compatibility with Python and its ability to handle asynchronous requests made it an excellent choice for ensuring efficient communication between the frontend and backend.

Using FastAPI, the model's functionality was exposed as API endpoints, allowing the Streamlit web app to send data to the backend and receive predictions seamlessly. Its straightforward syntax and detailed documentation streamlined the development process, making it easy to define and test the API routes.

FastAPI also ensured low-latency responses, which was crucial for the real-time interaction required by the application. This enabled users to experience quick and reliable predictions, enhancing the overall functionality and user experience of the web app.

2.6 Deployment

The application was deployed using Continuous Integration and Continuous Deployment (CI/CD), which is an approach that has improved the speed and efficiency of software delivery. The CI/CD pipeline allows for software changes to be tested, integrated, and deployed to production automatically.

The pipeline is as follows:

- **Code Integration:** Developers commit the code changes to a remote repository, such as GitHub, GitLab, or BitBucket. The code is integrated with the main codebase and it is ensured that the code changes are compatible. For the purpose of this project, GitHub was used to store the contents of the Dysarthric Speech Converter App.
- **Automated Testing:** This step involves running a set of tests to check the functionality of the code changes. This helps identify issues early in the development process.
- **Continuous Deployment:** In this stage, the code changes are automatically deployed to a staging environment. This step updates the software as and when there are changes made, and deploys it on its own.
- **Production deployment:** In this step, software changes are released to end-users. It also monitors the production environment for errors.

Using Streamlit Community Cloud and GitHub for CI/CD covers key stages of the pipeline in a simplified, automated setup. In the code integration stage, the code was uploaded to GitHub, where updates are merged into the main codebase.

Though optional, automated testing can be added through GitHub Actions, which is useful for running tests on code changes before deployment, which catches issues early. For continuous deployment, Streamlit Community Cloud automatically deploys the app whenever changes are pushed to the main branch in GitHub, making updates instantly live without manual steps.

While there is no separate production deployment phase, Streamlit essentially treats every deployment as a production update. This means users see the latest version of the app immediately, though any extra monitoring would need separate setup. Overall, Streamlit Community Cloud and GitHub simplify CI/CD by automating integration and deployment, with optional testing to ensure quality.

3 Results

The following images consist of the results of the various steps of our methodology.

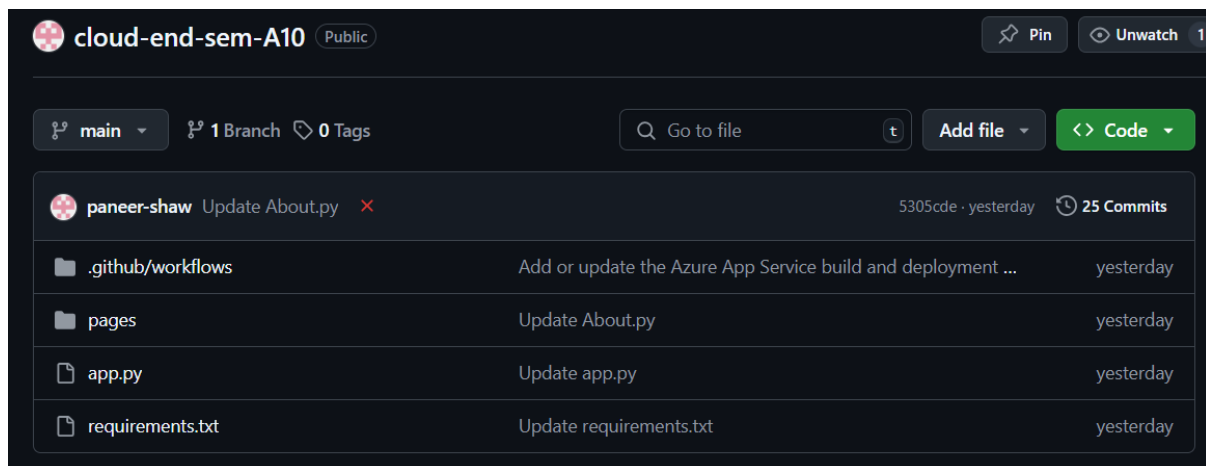


Figure 1 The Streamlit App in our GitHub Repository

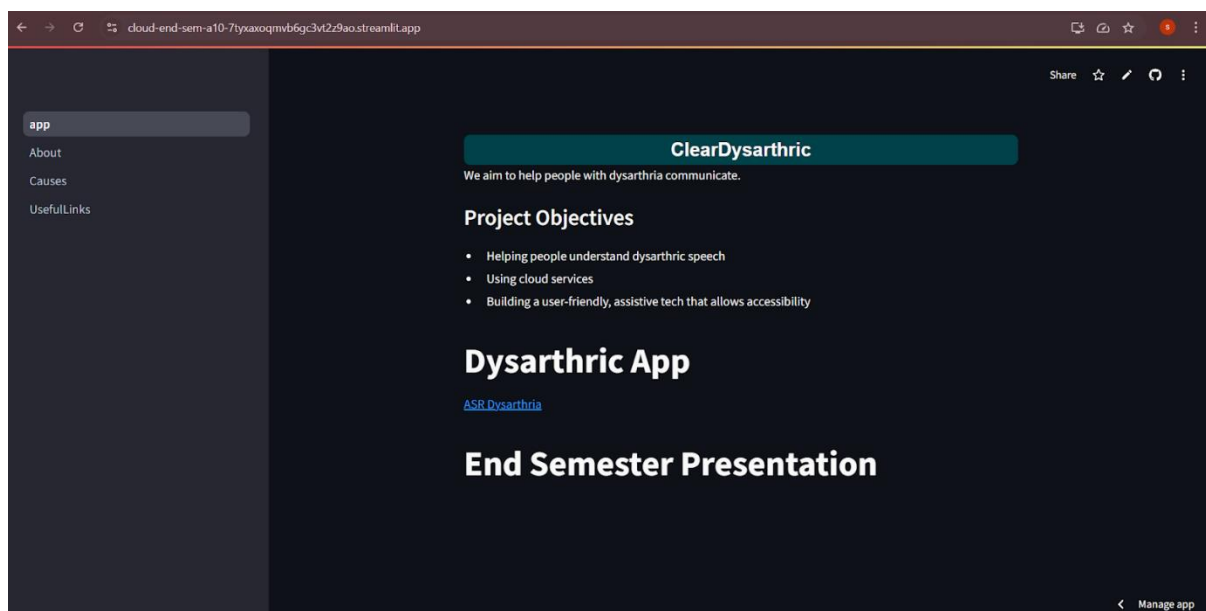


Figure 2 Our Home Page

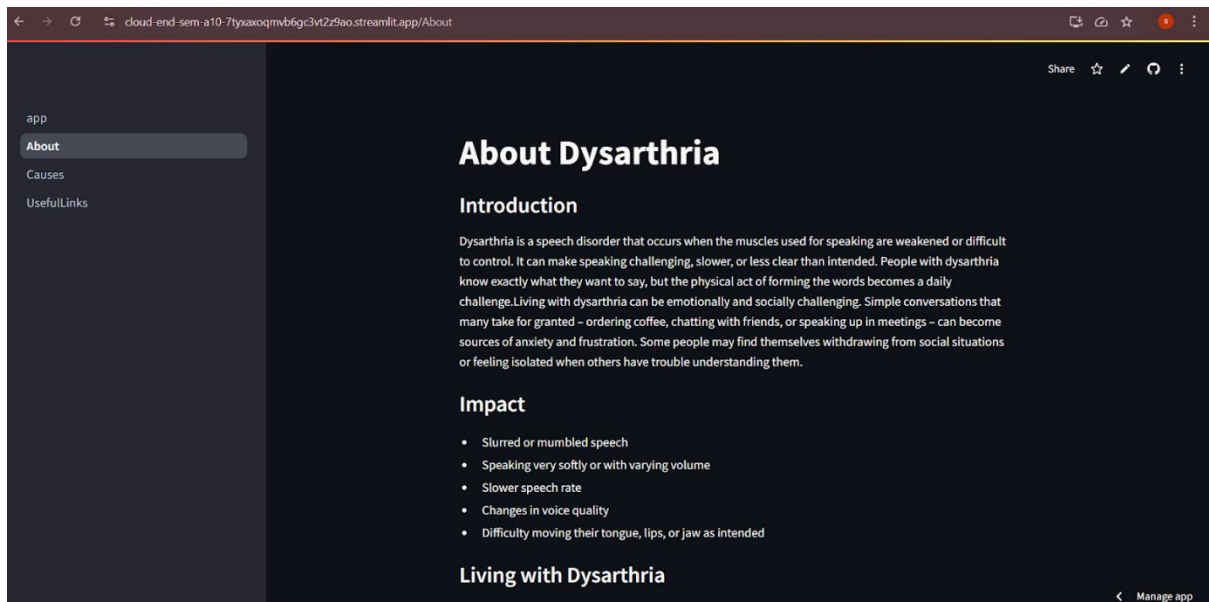


Figure 3 About Page

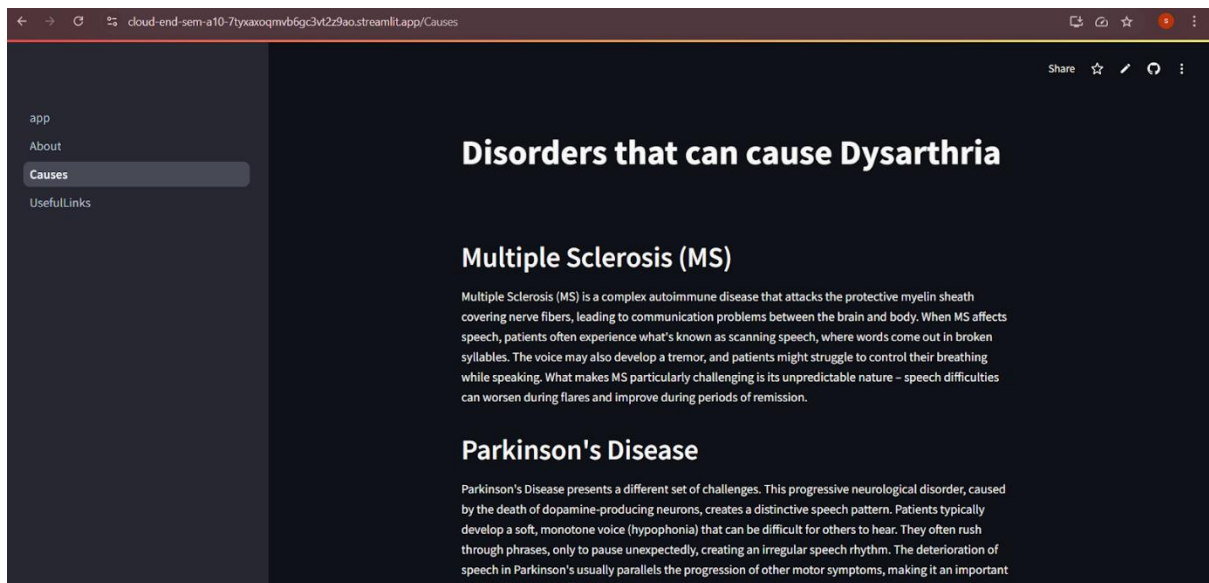


Figure 4 Information Page

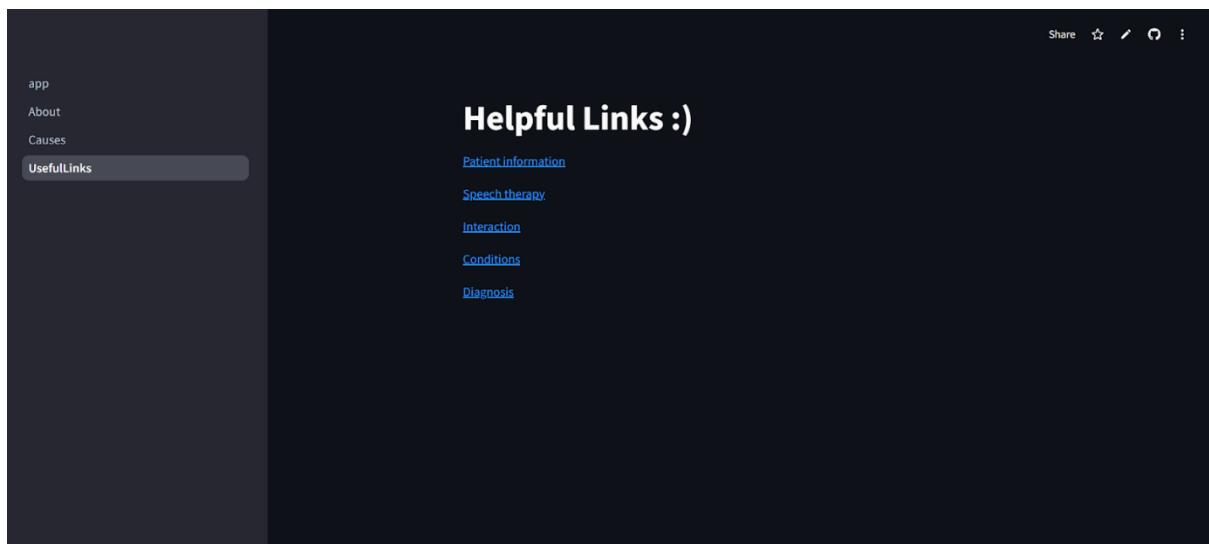
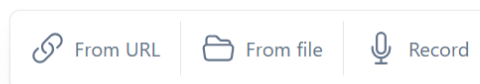


Figure 5 Useful Links Page

ASR Dysarthria

Automatic Speech Recognition for dysarthric speech

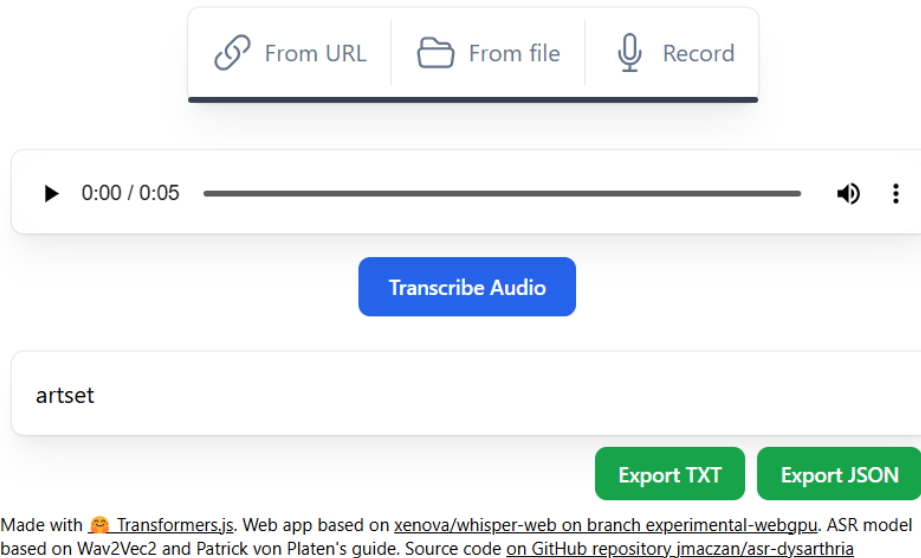


Made with 🍷 [Transformers.js](#). Web app based on [xenova/whisper-web](#) on branch [experimental-webgpu](#). ASR model based on Wav2Vec2 and Patrick von Platen's guide. Source code [on GitHub repository jmaczan/asr-dysarthria](#)

Figure 6 Page to Record and Translate

ASR Dysarthria

Automatic Speech Recognition for dysarthric speech



The screenshot shows the ASR Dysarthria web application interface. At the top, there are three buttons: 'From URL' (with a link icon), 'From file' (with a folder icon), and 'Record' (with a microphone icon). Below these is a media player showing a progress bar at 0:00 / 0:05. A blue 'Transcribe Audio' button is centered below the player. Underneath is a text input field containing the word 'artset'. To the right of the input field are two green buttons: 'Export TXT' and 'Export JSON'. At the bottom, a small text block states: 'Made with 🥰 Transformers.js. Web app based on [xenova/whisper-web](#) on branch `experimental-webgpu`. ASR model based on Wav2Vec2 and Patrick von Platen's guide. Source code [on GitHub repository jmaczan/asr-dysarthria](#)'.

Figure 7 Transcribed Word for 'Artist'

4 Discussion

This project successfully demonstrates the integration of modern technologies to address the challenge of transcribing dysarthric speech to text. The Convolutional Neural Network (CNN) model, trained on high-performance GPUs provided by Jarvis Labs, forms the core of the system, effectively handling the complex patterns inherent in dysarthric speech. Jarvis Labs enabled the training process to be conducted efficiently, ensuring optimal model performance within a manageable timeframe.

Streamlit played a pivotal role in bridging the gap between the trained model and end-users. Its seamless integration with Python allowed the development of a simple and accessible web interface, empowering users to interact with the system effortlessly. Through this interface, users could input speech samples and receive accurate transcriptions in real time.

FastAPI ensured a smooth communication channel between the frontend and backend. By exposing the model as an API, it enabled the Streamlit app to process user inputs dynamically and deliver predictions with low latency. The asynchronous capabilities of FastAPI were particularly beneficial in maintaining responsiveness, even during concurrent usage.

Overall, the integration of these technologies resulted in a cohesive solution that is both effective and user-centric, addressing the specific needs of individuals with dysarthria.

5 Conclusion

The development of this speech-to-text application for dysarthric speech highlights the

potential of leveraging advanced machine learning models and modern frameworks in tackling specialized challenges. By combining Jarvis Labs for training, Streamlit for the user interface, and FastAPI for backend communication, the project achieved a balance of performance, accessibility, and scalability.

This application provides a valuable tool for individuals with dysarthria, enhancing their ability to communicate effectively. While the project achieved its primary objectives, future work could focus on expanding the dataset, refining the model for greater accuracy, and incorporating additional features, such as multi-language support or adaptive learning based on user feedback. The results of this project underscore the importance of technological innovation in improving accessibility and inclusivity.

6 References

<https://jarvislabs.ai/>

<https://fastapi.tiangolo.com/>

<https://streamlit.io/>

<https://medium.com/codex/streamlit-fastapi-%EF%B8%8F-the-ingredients-you-need-for-your-next-data-science-recipe-ffbeb5f76a92>

<https://www.geeksforgeeks.org/mel-frequency-cepstral-coefficients-mfcc-for-speech-recognition/>

<https://asmp-eurasipjournals.springeropen.com/articles/10.1186/s13636-024-00357-3>

