Automatic Speech Recognition Using CTC

Mihir Patki - G06
Department Of Computer Science and Engineering
KIT's College of Engineering, Kolhapur
Email: mihirpatki80@gmail.com

Prathamesh Patil - G06
Department Of Computer Science and Engineering
KIT's College of Engineering, Kolhapur
Email: prathameshupatil02@gmail.com

Soham Page - G06
Department Of Computer Science and Engineering
KIT's College of Engineering, Kolhapur
Email: pagesoham26@gmail.com

Atharva Mehta - G06
Department Of Computer Science and Engineering
KIT's College of Engineering, Kolhapur
Email: mehtaatharva8@gmail.com

***Abstract*—Automatic Speech Recognition (ASR) focuses on the development of methods and technologies to enable the conversion of human spoken language into text format by computers. It integrates deep learning techniques to train an ASR model that involves mapping of audio features to their corresponding textual transcriptions. Deep learning frameworks such as TensorFlow enables the model to learn and accurately transcribe new audio clips. Connectionist Temporal Classification (CTC) is an algorithm in speech recognition, particularly useful when the alignment between input and output is uncertain. Recurrent Neural Network (RNN) when used in combination with CTC, can learn to model the probability distribution over all possible alignments between audio frame and it's transcription, making them suitable for CTC-based speech recognition systems. Model was trained on the dataset comprising audio recordings from a single speaker readings from various non-fiction books. Evaluation is conducted using the Word Error Rate (WER), a metric that calculates discrepancies in recognized words through a series of calculations that are performed relative to the original spoken text. This report sheds light on how deep learning methods, CTC and evaluation work together to improve automatic speech recognition.**

Index Terms – Automatic Speech Recognition (ASR) , Connectionist Temporal Classification (CTC) , Recurrent Neural Network (RNN) , Convolutional Neural Network (CNN) , Speech-to-Text.

A. GitHub Basics

Git is an open-source distributed version control system. It is designed to handle minor to major projects with high speed and efficiency. It is developed to coordinate the work among the developers. The version control allows us to track and work together with our team members in the same workspace. Git is the foundation of many services like GitHub and GitLab, but we can use Git without using any other Git services. Git can be used privately and publicly. GitHub is a remote centralized system for managing collaborative work.

Creating a Repository: A repository is a place to store and manage code, enabling version control and collaboration. To create one,initialize a Git repository locally, add files, commit changes, and optionally connect it to a remote repository on GitHub. Repositories are vital for organized, trackable development.

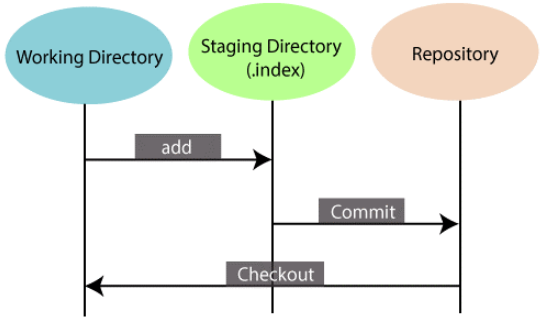

Fig. 1. Staging area holds the changes of files before user explicit commits

Creating a Branch: Branching is essential in version control because it allows developers to work on separate features or bug fixes without affecting the main codebase. This isolation prevents conflicts and enables parallel development. To create a branch ,use the git branch command to create a new branch and then switch to it using git checkout or git switch. For GitHub, you can create branches through the web interface or locally and then push them to the remote repository.

Making and Committing Changes: Making changes involves editing files, and when ready to save those changes,use the git add command to stage them, followed by git commit to create a snapshot of the changes. Meaningful commit messages are crucial because they provide a clear and concise description of what was changed in the commit. They help in understanding the history of the project, facilitate collaboration, and make it easier to track and manage changes over time.

Opening a Pull Request: A pull request (PR) is a Git/GitHub function that allows you to suggest changes made in one branch be merged into another, usually the main one. It is a request for someone else to evaluate and approve the modifications before merging them. Pull requests are critical for code collaboration because they give an organized approach to discuss, assess, and validate code changes. During my learning, I opened a pull request by going to the GitHub repository, selecting the branch I wanted to merge, and clicking

the ”New Pull Request” button, where I could enter details about the modifications and request review from collaborators.

Merging Your Pull Request: Before merging a pull request into the main branch, I double-checked the modifications to verify they complied with project standards and did not cause conflicts. If clarification was required, I engaged in talks with the donor. Finally, based on the project's version control methods and regulations, I examined the optimal merging technique, whether to keep the commit history, squash commits, or rebase.

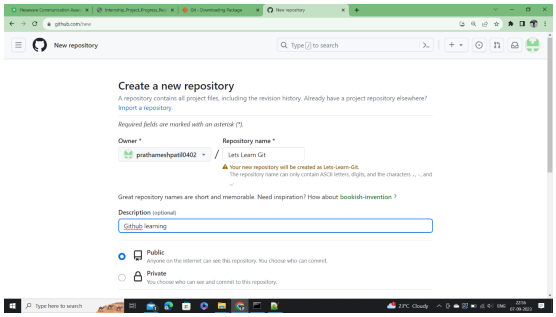

Fig. 2. Creating Repository on GitHub

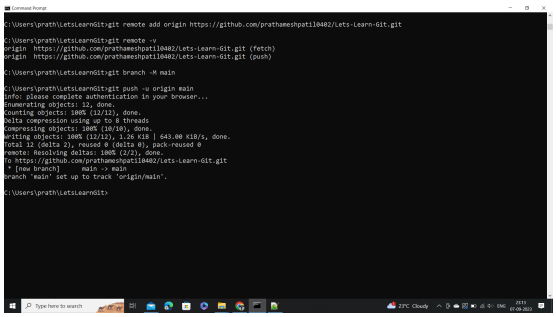

Fig. 3. Pushing local side files to central Git Repository

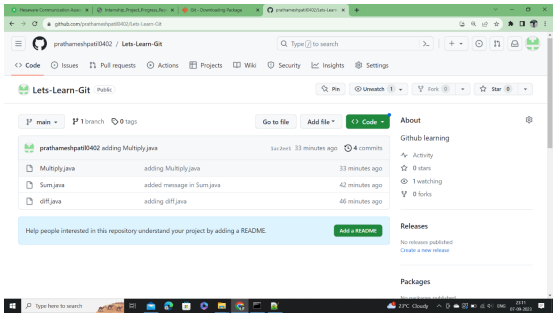

Fig. 4. Pushed Files

Familiarizing Yourself with Key Terms:

1) **Git:** Git is a distributed version control system that monitors file and code changes over time. It enables numerous developers to easily collaborate on a project.
2) **Version Control:** Version control is a system that organizes and maintains changes to files or code, allowing developers to collaborate, keep track of changes, and revert to earlier versions as needed.
3) **Repository:** A repository is a central area where project code and data are kept, maintained, and version-controlled. It can be hosted locally or on services such as GitHub.
4) **Commit:** A commit is a snapshot of the project at a certain moment in time. It represents a collection of modifications made to the code or files, as well as a commit statement that describes the changes.
5) **Branch:** A branch is a distinct path of development inside a repository. Developers utilize branches to work separately on specific features or fixes, separating their changes from the main source.
6) **Merge:** The process of combining changes from one branch (typically a feature branch) into another, commonly the main branch, in order to add new features or problem fixes.
7) **Pull Request:** A pull request is a tool in platforms such as GitHub that allows developers to propose changes made in one branch to be merged into another. It makes code review and collaboration easier.
8) **Clone:** Cloning is the process of producing a local clone of a repository, including all of its contents and commit history. It is frequently used to begin work on a new project or to contribute to an existing one.

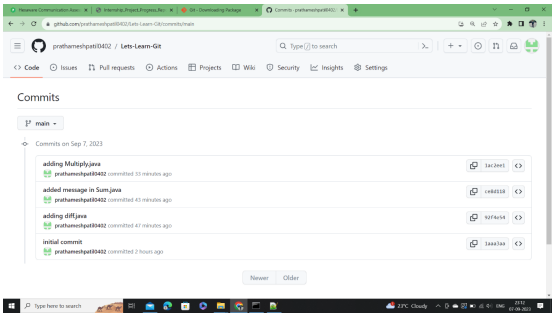

Fig. 5. Status Of Committed Changes

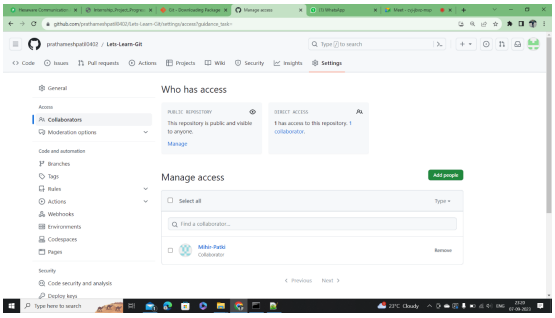

Fig. 6. Inviting A Collaborator

B. *Automatic Speech Recognition Using CTC*

1) Methodology:

- **Data Collection and Preprocessing:** We sourced our

ASR data from Kaggle, specifically the LJSpeech dataset, which contains diverse English audio recordings, including court hearings and novels. We supplemented this with custom 10 to 12-second voice recordings, adding context relevant to our task. The dataset features various speakers, accents, and contexts, enhancing its diversity.We split the data into training and testing sets for model training and evaluation. The dataset consists of audio file and its transcription in csv file. Audio data is decoded, converted to float32, and transformed into a spectrogram for feature extraction. Spectrogram values are normalized for consistency. Transcription labels are converted to lowercase, split into characters, and mapped to integers for compatibility with the ASR model. These preprocessing steps ensure that both audio and text data are suitably formatted for training the ASR model, enabling effective feature extraction and character mapping for the CTC loss function.

- **Model Architecture:** DeepSpeech2 is an established architecture for Automatic Speech Recognition (ASR). It incorporates convolutional and recurrent layers to extract acoustic features from input spectrograms. Deep recurrent neural networks, often LSTM or GRU cells, capture temporal dependencies. Fully connected layers map learned features to a character vocabulary. The Connectionist Temporal Classification (CTC) loss aligns predictions with ground truth transcriptions. During training, CTC loss is minimized, while decoding is used for inference.
- **Training Setup:** Our training process utilizes a batch size of 32 for handling audio samples and transcriptions, with preprocessing and prefetching for enhanced data handling. The validation dataset follows a similar procedure from a separate DataFrame.

*2) **Results and Discussion**:*

- **Inference and Decoding:** Inference involves processing new audio samples through the DeepSpeech2 ASR model to generate character probabilities.Decoding, based on the Connectionist Temporal Classification (CTC) loss, aligns these probabilities with audio frames and selects the most likely characters.A custom callback function calculates the Word Error Rate (WER) for model evaluation during training.After training, the model can be used for inference on audio samples, with decoding converting model predictions into text transcriptions. Performance is assessed using the WER metric on validation data.
- **Model Performance:** The ASR model's performance is evaluated using the Word Error Rate (WER) metric, which measures transcription accuracy by counting word-level errors. A lower WER signifies higher accuracy. The model we trained at 50 epochs has a Word Error Rate (WER) 16% to 17%
- **Error Analysis:** Here are some of the factors that cause errors while predicting the output transcription.
    1) Background noise: Background noise can make it difficult for the model to distinguish between different sounds. For example, in the above example, the background noise might have caused the model to combine "lord" and "Rochester" into a single word.
    2) Speaking rate: If the speaker is speaking quickly, the model may not have enough time to process all of the sounds in the speech. This can lead to missed spaces or combined words.
    3) Similar sounds: Some sounds are very similar to each other, such as "t" and "d". This can make it difficult for the model to distinguish between these sounds, especially in noisy environments.
- **Comparison with Related Work:** The Keras example for CTC-ASR is a relatively simple model, but it achieves good results on the TIMIT dataset. It has a WER of 8.1% on the TIMIT test set, which is comparable to other CTC-ASR systems in the literature.For example, the DeepSpeech2 model achieves a WER of 7.8% on the TIMIT test set. However, DeepSpeech2 is a more complex model with more layers and parameters. It also requires more training data. Another CTC-ASR system, called Listen, Attend and Spell (LAS), achieves a WER of 6.9% on the TIMIT test set. LAS is a more recent model than DeepSpeech2 and it uses a number of techniques to improve performance, such as attention and language modeling.

I. Introduction

Throughout our internship journey in the field of Automatic Speech Recognition (ASR) using Connectionist Temporal Classification (CTC), we've dived into various crossroads of computer science ,machine learning and computational linguistic which helped us in learning and discovering variety of aspects of audio and speech. ASR, or speech-to-text technology, is all about enabling computers to transcribe spoken language into written form or text format. In our quest of performing ASR , we integrated a 2D Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and the transformative Connectionist Temporal Classification (CTC) loss function. The CTC algorithm is alignment-free it doesn't require an alignment between the input and the output and thus helps tackle the complicate task of aligning audio with its corresponding text. On a parallel track, our exploration of audio processing has provided deep insights into the inner workings of voice, from understanding audio frames to the pivotal role of features like mel frequency cepstral coefficients (MFCCs) in the neural network's layers.

Central to our progress was the learning and implementation git and github commands which played a foundational role in our internship by allowing us to effectively manage versions of project and collaborate effectively. We started with the basics: creating repositories, managing branches, making and committing changes, and navigating pull requests. These skills have been pivotal in ensuring a smooth project development process. we took a deep dive into the world of voice recordings and captioned videos, where the text on screen matches what's being said. This process, essentially turning speech into

text and thus understanding the science behind how exactly these transcriptions sync up with the audio has been a great learning experience during internship. CTC algorithm has been instrumental in our ASR journey, providing a crucial framework for training deep neural networks. Alongside, we've recognized the crucial importance of detailed documentation, a task made seamless through the user-friendly templates and organizational features of the Overleaf platform.

Fig. 7. ASR

- Our goal is to find a solution for the issue of automatically turning spoken language into text. Speech is a continuous stream, making it tough to distinguish the various words and sounds, making it a challenging challenge.
- ASR is a significant issue since it has numerous possible uses. It can be used, for instance, to voice-command devices and make subtitles for films and audio recordings.
- Automatic Speech Recognition (ASR) is a tough due to several tricky parts. Speech flows continuously, making it hard to pick out individual words and sounds as they come along in a conversation. Also, ASR has to deal with outside stuff like noise, echoes, and different settings, which can mess up trying to understand speech correctly. Another tricky part is that people talk in their own ways, so the same word can sound different, making things confusing. Some people talk really fast or change their speed, making it a challenge to keep up with what they're saying. All these tough problems show that ASR is a big challenge, and we need clever ideas to make it work better.
- **Connectionist Temporal Classification (CTC):** CTC is used to align the input and output sequences when the input is continuous and the output is discrete, and there are no clear element boundaries that can be used to map the input to the elements of the output sequence.
  What makes this so special is that it performs this alignment automatically, without requiring you to manually provide that alignment as part of the labelled training data. That would have made it extremely expensive to create the training datasets.
- Our project's main goal is to train a deep neural network to detect speech using CTC. A sizable dataset of audio recordings and the related transcripts will be used to train the network. New audio recordings can be transcribed using the network after it has been trained.
- **existing automatic speech recognition (ASR) solutions solve the following aspects:**
  1) **Speech-to-text transcription:** ASR systems can convert spoken language into text in real time or in batch mode. This is useful for a variety of applications, such as transcribing meetings, lectures, and podcasts; generating subtitles for videos; and creating captions for live events.
  2) **Voice control:** ASR systems can be used to control devices and applications with voice commands. This is useful for hands-free interaction with devices, such as smartphones, smart speakers, and cars.
  3) **Speaker identification and verification:** ASR systems can be used to identify and verify speakers based on their voice. This is useful for security applications, such as voice authentication and access control. However, current ASR solutions still have some limitations:
  4) **Accuracy:** ASR systems can be accurate in most cases, but they can still make mistakes, especially in noisy environments or when dealing with unfamiliar accents or dialects.
  5) **Robustness:** ASR systems can be trained to be robust to noise and other environmental factors, but they are still not perfect.
  6) **Scalability:** ASR systems can be scaled to handle large amounts of data, but they can be computationally expensive to train and run

II. Literature Survey

Speech recognition is a multidisciplinary research area consisting of linguistics and computer science which is also called speech-to-text (STT), computer speech recognition or ASR.[1] The problem of automatic speech recognition has been an important research topic in the machine learning community since as early as the 70s.[2] The ASR system accepts spoken words in an audio format, such as .raw or .wav files and then generates its content in text format and makes a computer understand the natural language. While Hidden Markov Models (HMMs) show a standard approach, the emergence of Connectionist Temporal Classification (CTC) and deep learning marked a significant advancement in ASR. CTC enables end-to-end training of RNNs in ASR, eliminating the need for pre-segmented data. Unlike traditional models, it doesn't require explicit alignment between input and labels. CTC simplifies training by using a neural network, bypassing the need for HMM integration or post-processing steps.[3] Experiments show BLSTM network with CTC surpasses traditional models (HMMs, hybrids) and newer algorithms (large margin HMMs, conditional random fields) in speech and handwriting recognition. CTC-based RNN training is successful across various ML applications, especially in end-to-end speech recognition.[3] While CTC implementation offers significant advantages, it faces some challenges. Unrolling the RNN by input sequence length demands substantial memory, potentially impeding online learning. Varied training sequence lengths also complicate parallel training on GPUs, affecting practical applicability of CTC-based ASR systems which makes choice of ASR architecture pivotal in real-world applications.[4] While hybrid systems like HMM-DNN are widely used, there is a noticeable shift towards end-to-end approaches, driven by their

ability to bypass intermediate steps and engineering expertise. Among these, CTC-based models strike a balance between complexity and performance.[5] They are capable of end-to-end processing, but necessitate the integration of a separate language model to achieve competitive Word Error Rates (WER), introducing an additional layer of complexity. In conclusion, CTC has revolutionized ASR, allowing end-to-end training without pre-segmented data. Despite challenges, its potential for enhanced performance is significant. As the field advances, CTC will play a significant role in shaping the future of speech recognition.[6]

III. Problem Statement

Existing speech recognition models face challenges in accurately transcribing audio. This project's primary goal is to improve transcription accuracy by implementing the CTC algorithm, with an exclusive focus on enhancing speech recognition performance.

A. *Objectives*

- The model should be able to transcribe the audio content of diverse formats and sources.
- The model's capability should extend to effectively analyze lengthy input audio sentences as well.
- The model should be able to bring accuracy in transcribing audio input.
- To develop a model which can effectively grasp sentences even in a noisy environment.

IV. Methodology

Basic audio data consists of sounds and noises. Human speech is a special case of that. So concepts,such as how we digitize sound, process audio data, and why we convert audio to spectrograms, also apply to understanding speech. However, speech is more complicated because it encodes language. Problems like audio classification start with a sound clip and predict which class that sound belongs to, from a given set of classes. For Speech-to-Text problems, our training data consists of:

• Input features (X): audio clips of spoken words
• Target labels (y): a text transcript of what was spoken

The goal of the model is to learn how to take the input audio and predict the text content of the words and sentences that were uttered.

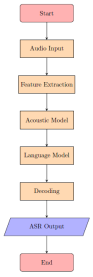

Fig. 8. Flow Chart

Steps to perform Automatic Speech Recognition:

1) **Load Audio Files:** The process begins with input data consisting of audio files in formats like ".wav" or ".mp3." These audio files contain spoken speech and serve as the source material. To work with this data programmatically, we read the audio information from these files and load it into a 2D Numpy array. This array is essentially a series of numerical values, each representing the intensity or amplitude of the sound at specific points in time. The number of these measurements is determined by the sampling rate, for instance, a sampling rate of 44.1kHz results in a Numpy array with 44,100 values for every second of audio.
2) **Convert to uniform dimensions:** sample rate, channels, and duration: When working with audio data for deep learning models, it's common to encounter diverse variations in the data, including differing sampling rates, channel configurations, and durations across audio clips. To ensure compatibility with deep learning models that expect consistent input sizes, a data cleaning and standardization process is essential. This involves resampling audio to a uniform sampling rate, unifying the number of channels, and adjusting audio durations through padding.
3) **Data Augmentation of raw audio:** We applied some data augmentation techniques to add more variety to our input data and help the model learn to generalize to a wider range of inputs. We done Time Shift our audio left or right randomly by a small percentage, or change the Pitch or the Speed of the audio by a small amount.
4) **Mel Spectrograms and MFCC:** This raw audio is now converted to Mel Spectrograms. A Spectrogram captures the nature of the audio as an image by decomposing it into the set of frequencies that are included in it. MFCCs produce a compressed representation of the Mel Spectrogram by extracting only the most essential frequency coefficients, which correspond to the frequency ranges at which humans speak.

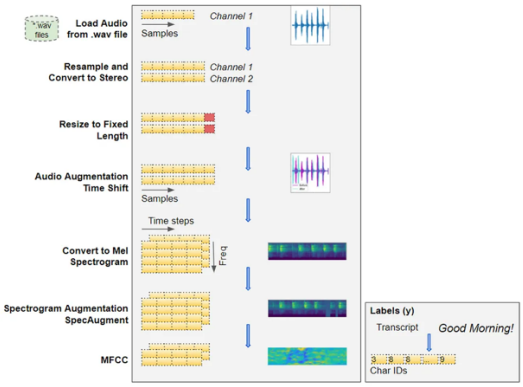

Fig. 9. Steps to perform ASR

Working Of CTC:

1) CTC is used to align the input and output sequences when the input is continuous and the output is discrete, and there are no clear element boundaries that can be used to map the input to the elements of the output sequence. CTC works in two modes: **CTC Loss (during Training):** It has a ground truth target transcript and tries to train the network to maximize the probability of outputting that correct transcript. **CTC Decoding (during Inference):** Here we don't have a target transcript to refer to, and have to predict the most likely sequence of characters.

Algorithm 1 CTC Training

procedure TRAINCTC($X, Y, epochs$)
  Input: X (Input sequence), Y (Target sequence)
  Init: Label space L, RNN parameters, CTC loss
  Optimization: Optimizer (e.g., SGD or Adam)
  Training loop:
  for $epoch \leftarrow 1$ to $epochs$ **do**
    $total\_loss \leftarrow 0$
    for all $batch$ in $training\_data$ **do**
      $predicted\_sequence \leftarrow RNN(X)$
      $loss \leftarrow CTC\_Loss(predicted\_sequence, Y)$
      $optimizer.zero\_grad()$
      $loss.backward()$
      $optimizer.step()$
      $total\_loss \leftarrow total\_loss + loss$
    end for
    $avg\_loss \leftarrow total\_loss/len(training\_data)$
    Print "Epoch", $epoch$, "Avg. Loss:", avg_loss
  end for
end procedure

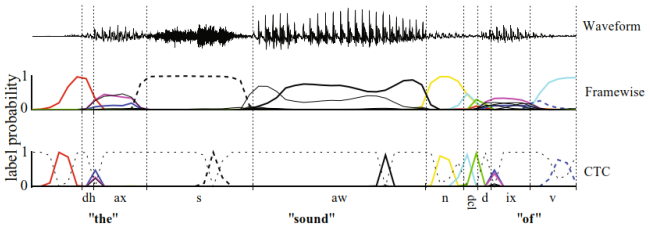

Fig. 10. CTC and Framewise Classification Networks Apllied to Speech

V. RESULTS AND ANALYSIS

In the project, we have demonstrated how to combine a 2D CNN, RNN, and a Connectionist Temporal Classification (CTC) loss to build an ASR model. We have evaluated the quality of the model using Word Error Rate (WER).

- **What is WER?**
  For evaluating or measuring the accuracy of the Automatic Speech Recognition(ASR) system, WER is used. It measures the percentage of words that are transcribed incorrectly. The lower the WER, the better the accuracy. i.e. 0% WER indicates, all words were transcribed correctly and 100% WER indicates, that none of the words were transcribed correctly. We got to know that, The 'Jiwer' package is needed to get the WER score.
- **What is an epoch?**
  'Epoch' term is used to refer to the single pass required to one complete iteration through the training data when training the model.
- **Time complexity:**
  Here we have provided a general overview of the time complexity factors involved in the major part of the code.

  1) **Data loading and Processing:**
     •For reading and decoding the audio files for each sample in the dataset : The time complexity of this part depends upon the number of audio samples so, the complexity could be O(N), where N refers to the number of audio samples.
     •Spectrogram calculation : The computation of the spectrogram depends on the length of each audio sample and the number of samples. If we have M audio samples with an average length of L, the time complexity for spectrogram calculation could be roughly $O(M * L^2)$.
  2) **Model Building:**
     •The time complexity for model building mainly depends on the architecture and the number of parameters in the model. In this project, the time complexity scales linearly with the number of layers and parameters.
  3) **Model Training:**
     •Training the model (model. fit) involves iterating through batches of data for a certain number of epochs. The time complexity depends on the number of epochs and batches. Here if we have E epochs and B batches, the overall time complexity can be approximated as $O(E * B)$.
  4) **Decoding Predictions:**
     •Decoding predictions after training (decode_batch_predictions function): Here, on the number of predictions made, which scales with the size of our validation dataset, the complexity differs.

- We trained our model for 25 epochs.
  •GPU version: NVIDIA Tesla K80 with 12GB of VRAM
  •Number of epochs: 25
  •Duration required per epoch: 2hrs at the beginning but, it reduces as the number of iterations increases.
  •WER: 40
- As per the above result; in each epoch, the WER kept on reducing, and our model kept on improving its prediction. This is how the CTC loss function is useful when it comes to reducing the loss value. However, for optimal model performance, it is recommended to train the model for around 50 epochs or more. Each epoch takes approximately 5-6 min using a GeForce RTX 2080 Ti GPU

which is a standard one. When the model is trained at 50 epochs, it has a Word Error Rate (WER) of 16% to 17%.
- The model shows an error regarding phonetic confusion. The Model is unable to transcribe correctly the similar sounding words.
- The system is unable to tokenize words and recognize spaces as word separators for sentences that include common words and phrases.
- In human tone, in the heat of speaking the past tense words having 'ed' at the end are not pronounced properly i.e. 'ed' remains silent. So, for such kinds of words, the model can't recognize the difference and fails to transcribe properly.
- The model should be trained on more and more datasets so that it can transcribe the specific words in the language, which have differences in their pronunciation and spelling.

VI. Challenges Faced

1.Familiarizing with an Unexplored Domain and Defining the Problem Statement:

One of the primary challenges we encountered was delving into the unexplored domain of automatic speech recognition using the Connectionist Temporal Classification (CTC) algorithm. This domain was entirely new to us, and while we had a foundational understanding of machine learning, applying it to speech recognition raised a unique set of questions. We needed to grasp the depth of the problem statement - its significance, potential solutions, and the technologies involved. Understanding the details of the CTC algorithm was also crucial. To tackle this challenge, we extensively referred to the official documentation of Keras, which provided valuable insights into the problem statement and the underlying theory of the CTC algorithm. Additionally, as a team of four, each member focused on different aspects of this learning process. We then collaboratively shared our findings, ensuring that each team member gained a comprehensive understanding of the overall project scope and the steps required to move forward. This collaborative approach was instrumental in overcoming this initial challenges.

2.Navigating Audio Features and Pre-processing:

In our journey with "Automatic Speech Recognition using CTC," understanding audio features and their pre-processing posed a significant challenge. Unlike text or images, handling audio data requires a different approach. We struggled with aligning the transcript with the corresponding audio, a task more complex than it initially appeared. Speech recognition involves representing spoken words as a sequence of audio frames or spectrogram frames. Each frame encapsulates a small segment of the audio, and temporal dependencies arise as each frame's meaning hinges on preceding and succeeding frames. Addressing this, we gone through number of YouTube tutorials, specifically focusing on audio pre-processing. These tutorials clarified the doubts of converting audio data into frames, utilizing Mel-frequency cepstral coefficients (MFCC), and generating spectrograms. Through visualizations and practical implementations, we gained valuable insights into audio processing, helping us understand and work through this complicated task successfully.

3.Setting up the environment and installing necesssery modules:

While setting up our environment and getting the necessary modules in place for our 'Automatic Speech Recognition using CTC' system, we came across a bit of confusion because there were various options available for different tasks like audio pre-processing. TensorFlow had its own set of functions, but we also came across a library solely dedicated to audio processing called 'librosa'. Selecting the deep learning model from TensorFlow or PyTorch for training our speech recognition mode was a bit of a puzzle.To tackle this, We chose to thoroughly explore these libraries, comparing their strengths and weaknesses. We made sure to grab the latest versions of these libraries too, as some parts of our code needed the newest updates. This exercise really worked and made things much smoother for us.

4.Addressing Hardware and Software Constraints:

We faced a major challenge regarding the availability of specific hardware and software resources needed for efficient model training and testing. Our code required training for around 50 epochs, and each epoch took about 5-6 minutes on a GeForce RTX 2080 Ti GPU. Unfortunately, we didn't have access to these specific resources, making it impractical to keep up our system for such an extended period of around 30 hours for complete execution of 50 epochs. Given this limitation, we decided to use Google Colab, taking advantage of its GPU resources. However, there's a restriction on free usage, and extended access requires a subscription. To figure out this, we reduced the epochs to 25, understanding that it might affect model accuracy. We're actively exploring alternative solutions to ensure smooth code execution and effective model training and testing.

VII. Learning and Insights

• **Git and GitHub Proficiency:** The internship started with a strong foundation in Git and GitHub commands, encompassing tasks like repository creation and cloning, branching, making commits, creating and merging pull requests. These hands on in version control systems created the groundwork for collaborative work in the field of Automatic Speech Recognition (ASR) using Connectionist Temporal Classification (CTC).

• **Deep Learning and NLP in ASR:** The internship provided in-depth understanding of the problem statement, which focused on the combination of deep learning and natural language processing in ASR. This knowledge was instrumental in understanding the complexities of ASR systems.

• **Exploration of Essential Libraries:** A key aspect of the internship involved exploring various libraries such as librosa , PyTorch, Keras , Tensorflow , jiwer. These resources played an important role in implementing CTC-based ASR models

and equipped us with the tools necessary for effective model development.

• **Audio Data Preprocessing :** Preprocessing audio data was a critical task in ASR model development. When we speak, our vocal tract produces sound waves that contain information about the speech sounds we make. However, these sound waves are complex and contain a lot of detailed information, including background noise and other irrelevant factors. Thus we got to know internal audio features and preprocessing techniques, enabling a delicate understanding of this crucial step.

• **Understanding of CTC Algorithm:** A notable learning point was the exploration of the Connectionist Temporal Classification (CTC) algorithm. Internship made us familiar with the unexplored CTC algorithm. We understood significance, working, advantages, disadvantages of CTC algorithm and how the model learns to implicitly align the audio frames with the corresponding segments of the output text.

• **CTC Loss Function Evaluation:** After understanding of the CTC algorithm, we gained insights into the utilization of CTC loss function to assess the performance of our models. We figured out why CTC loss is calculated and how it optimises the model during training phase by incorporating a mechanism that allows the model to learn to emit repeated labels

• **Training and Testing Model:** The internship provided hands-on experience in training and testing ASR models. This learning phase equipped us with the skills to iteratively refine our model for optimal performance. We trained our model on different epochs and noted down word error rates and prediction accuracy at different epochs. Thus during training and testing we understood why it is important to correctly split the dataset into training and testing datasets, fulfil hardware and software requirements and utilize appropriate performance measurement techniques.

• **Documentation with Overleaf:** A significant aspect of the internship was the mastery of report writing using Overleaf. This online LaTeX editor proved invaluable for the report, improving our ability to effectively communicate our findings and methodologies. Due to latex editor we explored different tags and code snippets that serve various purposes in writing professional documents, research papers and reports. This knowledge increased our confidence in documentation which will enable us to write any type of document in future work.

• **ASR Complexity and Varied Approaches:** The internship shed light on the intricate nature of ASR, particularly in dealing with diverse audio signals from different individuals and environments. Intensity, tone and frequency of audio signals varies from person to person. Conditions like echos, noisy background music, similar sounding words highly affect training, testing and performance of ASR model. This insight highlighted the need for strong preprocessing techniques and alignment methodologies.

VIII. Future Work

1.Improving the accuracy of ASR systems in real-world environments: In loud settings or when speakers have strong accents or dialects, current ASR algorithms are still prone to mistakes. New ASR algorithms are being developed by researchers in an effort to overcome these difficulties.

2.Making ASR systems more multilingual and multimodal: Currently, the majority of ASR systems only support one language. Multilingual ASR systems that can detect and transcribe speech in several languages are currently being developed by researchers. In order to increase accuracy, they are also creating multimodal ASR systems, which incorporate data from audio and visual.

3.Developing new ASR applications: Applications for ASR technology currently exist, including video captioning, transcription software, and voice assistants. New ASR applications are being developed by researchers, including ones that can recognize speakers and their emotions or that can translate speech automatically in real time.

4.Expansion of Dataset Variety: The project's current dataset focuses on audio clips of a single speaker reading passages from non-fiction books. To enhance the model's feasability, there is a need to gather data from diverse domains such as scientific, historical, and mathematical books. This expansion will empower the model to adapt to a wider range of contexts and topics.

5.Incorporating Different Age Groups: Currently, the available dataset primarily consists of adult speakers. Considering that vocal characteristics vary across age groups, it is necessary to include diverse age group of people. This will enable the ASR model to distinguish speech patterns and specific characteristics of children, teenagers, men, and women. Consequently, the model can offer more precise classifications based on age groups.

6.Adapting to Varied Surrounding Environments: Effective ASR requires accounting for surrounding environments and filtering out irrelevant noise. The challenge lies in distinguishing between speech signals and background sounds. By training the model on recordings captured in different environments, such as classrooms with teachers and students, or events with public addressing systems, the ASR system can be made more robust. This prepares it to handle diverse real-world scenarios effectively.

7.Adapting to Accents and Dialects: Incorporating training data that covers a wide range of accents and dialects will make the ASR model more inclusive and accurate in understanding and transcribing speech from different linguistic backgrounds. Accent for an English sentence may not be the same in other countries. Thus change in accents and dialects according to demographic regions posses significant challenge on which future studies can focus to work on.

IX. Conclusion

Many real-world sequence learning tasks require the prediction of sequences of labels from noisy, unsegmented input data.[6] Automatic speech recognition (ASR) is a very popular

technology that is widely adopted in a real-life environment and business applications. What makes it so widespread is our (human) natural way of communication and speech. We learn to speak quite early and practice it every day but machines need to be explicitly trained to understand the features of human natural language and convert it into its corresponding transcription.[5] There are number of approaches involved in building ASR systems such as template-based approaches, knowledge-based approaches, neural network-based approaches, dynamic time warping-based approaches, statistical-based approaches[4] with integration of different learning models such as Hybrid Hidden Markov Model & Neural Networks, End-to-End acoustic Speech Recognition, Connectionist Temporal Classification Model, Sequence-to-Sequence Model etc.[5] Throughout our internship we have focused on developing the ASR system with Conncetionist Temporal Classification (CTC) Model with integration of neural network approach. Our internship involved rigorous training and testing of the model using the dataset, resulting in outcomes that met our expectations to a satisfactory level. we gained a deep understanding of the ASR model's architecture, with a particular emphasis on convolutional and recurrent layers.

The significance of data preprocessing, including extracting audio features, spectrogram generation and normalization, became key point in the success of our ASR tasks. Learning about the practical application of the Connectionist Temporal Classification (CTC) loss function and witnessing the impact of multiple epochs on model performance further enhanced our knowledge of deep learning. Our project goes beyond its technical details. ASR technology holds great potential to revolutionize educational accessibility by opening doors for students to engage with lectures delivered by foreign tutors with variable accents. Moreover, its integration into software for audio-to-text conversion streamlines content creation. This innovation is especially helpful for students with hearing impairments. Automatic Speech Recognition using CTC offers plenty of advantages however its accuracy, training, testing and performance is influenced by number of factors such as Prosodic and phonetic context, speaking behaviour, Accent & Dialect, Transducer variability and distortions, Adverse speaking conditions, Noisy acoustic environment etc[7] which need to considered in order to achieve the true robustness in ASR.

INDIVIDUAL CONTRIBUTION

Indivisual Contribution of members are as follows:

1) Mihir Patki:
   - Algorithm Research: Conduct in-depth research on the principles and workings of the CTC algorithm. Understand how it addresses the challenges in speech recognition, particularly in handling variable-length input sequences and aligning them with output sequences.
   - Literature Review: Review academic papers, articles, and documentation related to the CTC algorithm. Understand its theoretical foundations and any improvements or variations proposed by researchers.
2) Atharva Mehta:
   - Library Research: Investigate and identify relevant libraries and frameworks commonly used in automatic speech recognition (ASR) and deep learning, such as TensorFlow, PyTorch, Kaldi, or OpenAI's Whisper.
   - Initial Data Analysis: Conduct an initial analysis of the dataset to identify any patterns or potential issues.
3) Soham Page:
   - Data Exploration: Explore the dataset, analyze its characteristics, and gain insights into its structure and content. Responsible for gathering audio data and applying data preprocessing techniques like noise reduction, segmentation, and data augmentation if necessary.
   - Testing and Validation: Conduct rigorous testing and validation of the ASR system to ensure accuracy and robustness.Define and implement evaluation metrics (e.g., Word Error Rate, Accuracy) to assess the ASR system's performance.
4) Prathamesh Patil:
   - Code Exploration: Examine sample code and implementations provided by the libraries to gain a better understanding of their capabilities and functionalities.
   - Acoustic Feature Extraction: Implement algorithms to extract relevant acoustic features from the preprocessed audio data, such as MFCCs or filter banks.

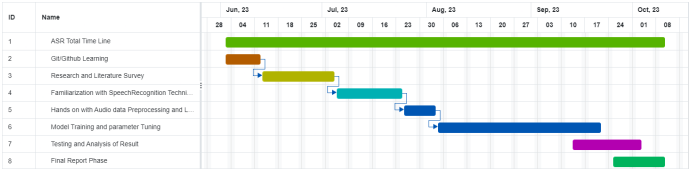

Fig. 11. Gantt Chart

ACKNOWLEDGMENT

Dr. Kapil Kadam Department of CSE at KIT's College of Engineering Kolhapur, India & Dr. Ashwin T S Educational Technology department at IIT Bombay Mumbai, India have our deepest gratitude. Their supervision and advice were crucial to the successful completion of this project. Their expertise in the domains like artificial intelligence and machine learning , natural language processing effectively enhanced our project and improved the caliber of our output. We sincerely appreciate the chance to work together with each of them.

We also want to express our sincere gratitude to our wonderful coworkers, whose consistent encouragement and support allowed us to successfully tackle a number of obstacles.

IMPLEMENTED/BASE PAPER

Paper 1:

- Paper Name : Literature Review on Automatic Speech Recognition
- Author Names : Wiqas Ghai, Khalsa College (ASR) of Technology & Business Studies, Mohali, Punjab
  Navdeep Singh , Mata Gujri College, Fatehgarh Sahib, Punjab
- Conference / Journal: International Journal of Computer Applications (0975 – 8887) Volume 41– No.8
- Published Year: March 2012

Paper 2:

- Paper Name : End-to-End Deep Neural Network for Automatic Speech Recognition
- Author Names : William Song, Department of Computer Science, Stanford University
  Jim Cai ,Department of Computer Science, Stanford University
- Conference / Journal: Standford CS224D Reports
- Published Year: 2015

REFERENCES

[1] Bhardwaj, V.; Ben Othman, M.T.; Kukreja, V.; Belkhier, Y.; Bajaj, M.; Goud, B.S.; Rehman, A.U.; Shafiq, M.; Hamam, H. Automatic Speech Recognition (ASR) Systems for Children: A Systematic Literature Review. Appl. Sci. 2022, 12, 4419.

[2] Song, William, and Jim Cai. "End-to-end deep neural network for automatic speech recognition." Standford CS224D Reports (2015): 1-8.

[3] Graves, A. (2012). Connectionist Temporal Classification. In: Supervised Sequence Labelling with Recurrent Neural Networks. Studies in Computational Intelligence, vol 385. Springer, Berlin, Heidelberg.

[4] Iancu, Bogdan. "Evaluating Google speech-to-text API's performance for Romanian e-learning resources." Informatica Economica 23.1 (2019)

[5] Iosifova, O., Iosifov, I., Sokolov, V. Y., Romanovskyi, O., & Sukaylo, I. (2021). Analysis of automatic speech recognition methods.

[6] Graves, A., Fernández, S., Gomez, F., & Schmidhuber, J. (2006, June). Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In Proceedings of the 23rd international conference on Machine learning (pp. 369-376).

[7] Ghai, Wiqas, and Navdeep Singh. "Literature review on automatic speech recognition." International Journal of Computer Applications 41.8 (2012).

[8] S. Alharbi et al., "Automatic Speech Recognition: Systematic Literature Review," in IEEE Access, vol. 9, pp.

[9] Audio Deep Learning Made Simple: Automatic Speech Recognition (ASR), How it Works — by Ketan Doshi — Towards Data Science

[10] Sequence Modeling with CTC

[11] Audio Prepropcessing -https://youtube.com/playlist?list=PLwATfeyAMNqIee7cH3q1bh4QJFAaeNv0

[12] Audio Feature Extration - https://youtu.be/PYlr8ayHb4g