UNIVERSITI TEKNOLOGI MARA

COMPARATIVE ANALYSIS OF SPEECH DETECTION MODELS WITH A FOCUS ON THE JAPANESE LANGUAGE

MUHAMMAD ALIFF AIMAN BIN ZOLKIFELI

MSc

March 2025

UNIVERSITI TEKNOLOGI MARA

COMPARATIVE ANALYSIS OF SPEECH DETECTION MODELS WITH A FOCUS ON THE JAPANESE LANGUAGE

MUHAMMAD ALIFF AIMAN BIN ZOLKIFELI

Dissertation submitted in partial fullfillment
of the requirements for the degree of
Master of Computer Science

College of Computing, Informatics and Mathematics

March 2025

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figures Title Page

CHAPTER ONE
INTRODUCTION

1.1 Research Background

The advancement of technology has become the driver of the importance of human-computer interaction. Human-computer interaction has been evolved from manual input into more natural interace and one of the natural interface is Automatic Speech Recognition (ASR). ASR have the capibilities to convert spoken language into text which is really useful in in application such as captioning, meeting minutes, media archiving, and voice-enabled search(Wei Xu & Gao, 2023). In the context of japanese language, the quality of the transcription is still a challenge because of the multiple writing system that is kanji, hiragana and katakana and also context-sensitive readings that exist in the language (Koenecke et al., 2020).

Most of the ASR pipelines is focusing on the acoustic or end-to-end model (Xu et al., 2023). However the real-world performance really depends on the preprocessing and feature extraction decision made before the data is fed into the model. The important preproccessing step is noise and reverberation reduction, voice activity detection (VAD), segmentation, resampling, channel and gain normalization, and cepstral mean/variance normalization (CMVN). Meanwhile the important feature extraction that need to be carried out is MFCC for traditional systems, and log-Mel filterbanks with augmentation for modern neural models. The preprocessing and feature extraction step can gave a big impact to the accuracy and stability of the model (Latif et al., 2020).

In addition to the preprocessing steps, the model's output also must be clean for easy reading, quoting, and storing. This creates additional considerations, such as post-processing and text formalization. For example, inconsistent punctuation and stopwords like "eeto" and "ano" must be handled during post-processing. This step is important to convert raw ASR output into a standardized format that can be utilized for lecture notes, reports, or subtitles (Ando & Fujihara, 2021).

This paper will be focusing on formal Japanese speech transcription and compare three representative modeling approaches under controlled pre-processing and feature extraction. The first model is GMM-HMM pipeline that is traditional model, the second model is CRDNN-CTC model that is hybrid model, and the last model is fine-tuned Whisper model that is transformer model. This paper will also evaluate the model performance based on Character Error Rate (CER) as the primary metric, and Word Error Rate (WER) and Real-Time Factor (RTF) as secondary metrics.

1.2 Problem Statement

Despite rapid progress in ASR,a reliable transcription of Japanese remains a challenge due to its mixed writing systems (kanji, hiragana, katakana) and context-sensitive readings, which challenge both tokenization and error scoring. At the same time, much prior work emphasizes acoustic/end-to-end architectures, while real-world accuracy and stability also rely on preprocessing and feature extraction steps like noise handling, segmentation,normalization, and feature design such as MFCC or log-Mel. However, there is a lack of systematic comparisons because of the pre-processing is configured inconsistently across studies. These inconsistencies, together with huge diverse datasets and scoring conventions has led to uncertainty about which model family is the most effective for formal Japanese transcription, and how accuracy trades off with practical decoding latency. As a result, there is still a documented gap in adapting and evaluating state-of-the-art systems for Japanese-specific contexts with attention to both linguistic and operational considerations (koenecke2020racial; Ando & Fujihara, 2021; Xu et al., 2023).

This study addresses that gap by conducting a controlled, apples-to-apples comparison of three representative traditional statistical GMM–HMM, hybrid CRDNN–CTC, and fine-tuned Whisper transformer models by standardizing preprocessing, feature extraction, and evaluation protocols. All models will be trained and evaluated on the same formal Japanese speech dataset with matched preprocessing configurations such as segmentation, resampling, CMVN and the feature types such as MFCC and log-Mel. Then the outputs will be post-processed uniformly to ensure readability and standardization. Finally, the same evaluation metrics CER, WER and RTF will be computed consistently to quantify both accuracy and usability. This systematic comparison will clarify which model family and preprocessing/feature setup is most effective for formal Japanese transcription under matched conditions, filling a critical gap in the literature and informing best practices for ASR system design in Japanese contexts.

1.3 Research Objectives

1. To identify the key requirements for constructing speech-to-text model within the context of Japanese language.
2. To analyze speech-to-text models to determine the most effective pre-processing setup to reduce CER and RTF in Japanese language processing.
3. To evaluate the CER and the transcription latency using RTF of different speech-to-text model when transcribing Japanese formal and informal language.

1.4 Research Questions

1. What is the key requirements for constructing speech-to-text model within the context of Japanese language.
2. What is the most effective pre-processing setup to reduce CER and RTF in Japanese language processing.
3. How to calculate the performance and effectiveness using CER and RTF of different speech-to-text model in context of Japanese language?

1.5 Scope of Study

This study will be focusing on speech-to-text transcription of Japanese language using only formal speech. The dialect speech will not be included in this study. The main focus of this study is to produce a readable and standardized text rather than detect dialects or classify speaking styles. For the preproccessing and feature extraction, this study will be focusing on segmentation, resampling and normalization, and two feature families of MFCC and log-Mel filterbanks.

The models that will be compared in this study are GMM-HMM, CRDNN-CTC, and fine-tuned Whisper. The evaluation metrics that will be used in this study are Character Error Rate (CER) as the primary metric, and Word Error Rate (WER) and Real-Time Factor (RTF) as secondary metrics. The RTF will be calculated by dividing the total decoding wall-clock time by the total audio duration, using a fixed hardware/software setup.For the text post-processing, this study will be focusing on normalizing the output for formal use by removing fillers words, numeric and punctuation normalization, and consistent script conventions. Semantic editing and translation are out of scope for this study.

For the dataset, this study will only be using formal Japanese speech with available transcripts. The data that will be used is TEDx talks in Japanese language from YouTube that is scraped using yt-dlp tool and then the audio is extracted using ffmpeg tool.

1.6 Significance of Study

This study is aimed to address the gap of effective speech-to-text solution that focusing on Japanese language. Most of the developed models is focusing on English language or a generic transcribe model that is developed for multi-language. Organization that rely on accurate transcript like broadcasters, government agencies, and archives rely on this kind of technology. This thesis will be a guidance to determine which pre-processing and feature configurations most improve outcomes for formal Japanese across different model types.

This study also contributes to the research by providing a systematic comparison of model choices in ASR across traditional, hybrid, and fine-tuned transformer models in the context of formal Japanese transcription. By filling this gap, the findings will inform best practices for ASR system design in Japanese, supporting both academic research and practical applications in industries that depend on accurate speech-to-text conversion.

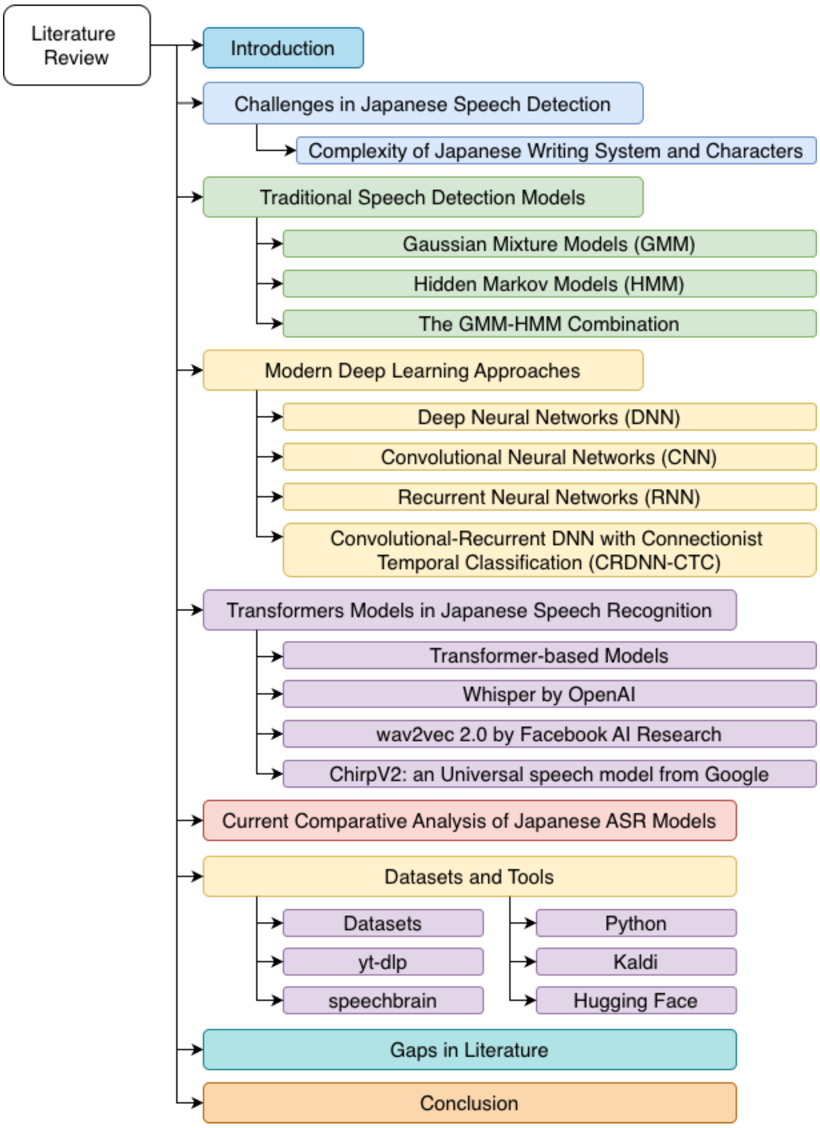
1.7 Conclusion

In this chapter, the advancement in machine learning and artificial intelligence that made the computer can understand human better by improving the speech to text model accuracy and speed has been discussed. However, there is still challenges to transcribe a language that has complex structure like Japanese that include syllable-based formation and the use of multiple writing systems. Because of this, a study to find which implementation and which model is the most performance for handling Japanese language. The finding from this study is very important to answer the question of which model is the best for speech-to-text solution in Japanese language. By identifying the specific linguistic challenges and comparing these models, this study will provide a valuable information that will be able to guide future advancements in speech-to-text technology in Japanese language and ultimately will be able to support its broader application across the industries that rely heavily on precise and efficient transcription.

CHAPTER TWO
LITERATURE REVIEW

2.1 Introduction

The technology for Automatic Speech Recognition (ASR) has advanced rapidly in these years. Starting from traditional models like GMM and HMM into more sophisticated deep learning approaches such as DNN, CNN, RNN, and Transformer-based architectures.



Literature Review
Introduction
Challenges in Japanese Speech Detection
Complexity of Japanese Writing System and Characters
Traditional Speech Detection Models
Gaussian Mixture Models (GMM)
Hidden Markov Models (HMM)
The GMM-HMM Combination
Modern Deep Learning Approaches
Deep Neural Networks (DNN)
Convolutional Neural Networks (CNN)
Recurrent Neural Networks (RNN)
Convolutional-Recurrent DNN with Connectionist Temporal Classification (CRDNN-CTC)
Transformers Models in Japanese Speech Recognition
Transformer-based Models
Whisper by OpenAI
wav2vec 2.0 by Facebook AI Research
ChirpV2: an Universal speech model from Google
Current Comparative Analysis of Japanese ASR Models
Datasets and Tools
Datasets
yt-dlp
speechbrain
Python
Kaldi
Hugging Face
Gaps in Literature
Conclusion

Figure 2.1 Literature Review Mind Map

However, to apply these technologies to the Japanese language may pose few challenges due to its complex writing systems. In this chapter, the challenges of Japanese speech detection and the traditional and modern ASR models will be reviewed. The state-of-the-art model like Whisper, wav2vec, and Chirp will also be discussed based on their applicability to Japanese. By identifying the key challenges and gaps in existing research, this chapter prepared for a focused analysis of Japanese-specific ASR systems.

2.2 Challenges in Japanese Speech Detection

2.2.1 Complexity of Japanese Writing System and Characters

The complexity of Japanese writing system and character can cause challenge in ASR system especially in end-to-end neural network architectures. Japanese writing system is a combination of multiple character sets, such as the Hiragana, Katakana, Kanji (ideographic characters), Roman letters and various symbols, leading to a considerably larger and more varied character (Rose, 2019). As mentioned by Ito et al. (2016), Ito et al. (2017), the number of possible Japanese character labels can exceed several thousand.

A single character of kanji may have a few ways to pronounce it because each character of kanji has Onyomi (Chinese derived) and Kunyomi (native Japanese) readings, and these readings can change depending on the word context (Curtin, 2020). Because of this ambiguity, the ASR system must be able to model and distinguish numerous acoustic differences in the speech data with same sound. The training model must be able to handle thousands of character and each of the character is potentially linked to multiple context dependent phonetic outcome which require a significant computational resources and large scale training data to ensure adequate coverage (Ito et al., 2016; Ito et al., 2017).

2.3 Traditional Speech Detection Models

2.3.1 Gaussian Mixture Models (GMM)

GMM have been the earliest technology used for developing Japanese speech detection and recognition systems because of their capability in capturing the statistical distribution of speech features very well (Imaishi & Kawabata, 2022). Because of the absence of word boundaries and the nuances of pitch accent in the Japanese language, it is really complicated to understand the context of the spoken words. However, GMM would be useful by employing probabilities to manage and characterize intricate patterns (Sun & Chol, 2020). For example, Povey et al. (2011) were able to use GMM to model phoneme-based acoustic features, and this approach led to a good performance of speech recognition systems.

Imaishi and Kawabata (2022) developed an approach within the EM algorithm that leads to the stabilization of the GMM parameters as well as increasing the discriminative power of the model in cases where there is not much evaluation data available. In other work, Povey et al. (2011) point out that it is possible to represent the distribution of speech features in GMM mode by employing a combination of several Gaussian components. This way the GMM ca n account for the phonetic or speaker variability which is known to be present during word is being pronounce.

Takami and Kawabata (2020) emphasized a different direction which starts with the creation of the Universal Background Model, which is a Gaussian Mixture Model calculated from the collection of a large number of speech samples. To develop a model of the characteristics of a given UBM, the UBM is modified through Maximum A Posteriori (MAP) Adaptation. This method adjusts parameters of the UBM such as mean vectors, covariance matrices, and mixture weights depending on the individual's data (Dehak et al., 2009). Studies also have shown that the use of speaker factor space constructed in the GMM and Joint Factor Analysis (JFA) can greatly improves the accuracy and efficiency of GMM systems (Matrouf et al., 2011).

2.3.2 Hidden Markov Models (HMM)

HMM is working quite well with Japanese speech detection because of the incorporation of the acoustic and temporal characteristics of speech, including the difficulties found in the encoding of Japanese speech (Tokuda, 1999). Moreover, HMM is so useful in ASR because they are very efficient in the representation of time varying systems by a succession of discrete time states. A unique segment of the speech signal is represented in each state, and the segment is described using a specific set of acoustic features (Juang & Rabiner, 1991). ASR systems incorporated with

HMM are more superior in portraying Japanese speech characteristics' rhythm and tone including essential features like pitch accent and moraic timing which features will enhance the performance of the systems on the phonology aspects of the language (Tokuda, 2000).

ASR systems based on HMMs give quite satisfactory results especially on languages like Japanese because it is a possible to interpolate between a discrete set of states, where each state stands for a segment of the speech signal that has distinct acoustic features like pitch, duration, and phoneme quality (Juang & Rabiner, 1991). To further Increase Japanese ASR project, few other models is used along with HMM which is context-sensitive such as Tri-phone method. Tri-phone method is a phonetic expansion that employs phonetics of the neighbor sounds to the phoneme as context in order to increase the recognition accuracy by taking into account the co-articulation that takes place during fast speech production (Tokuda, 2000). Other models by Gales and Young (2008) were used together with HMM are Maximum Mutual Information and Minimum Phone Error which are useful for optimizing the parameters of the HMM and improve the recognition performance.

2.3.3 The GMM-HMM Combination

The GMM-HMM model uses GMM for the observation probabilities corresponding to each state of the HMM. Each state of an HMM is assumed to have a library of Gaussian mixtures with which the state's acoustic feature is pooled. Because transition probabilities of each state are determined by the HMM, temporal dependency of speech is well modelled. This combination allows the system to account for some of the variations in speech signals, such as those related to accent and the differences in the pronunciation of words in the Japanese language (Taheri & Taheri, 2006). While HMMs trained with large datasets under maximum likelihood criteria may have limited discriminative power, incorporating GMMs as observation models captures a broader range of acoustic variations. This method works really well for Japanese language, which are sensitive to the duration of phonemes in the context of the language.

Furthermore, the integration of GMM and HMM eliminates the need for applying state-of-the-art feature extraction techniques like Mel-Frequency Cepstral Coefficients (MFCC), hence increasing recognition performance (Sonali Nemade, 2019). This hybrid approach has been successful in speaker-dependent as well as in speaker-independent systems. When fuzzy clustering and the expectation-maximisation algorithm are used, lower error rates are usually obtained by GMM-HMM than the methods used in isolation. For example, in a paper on speech data collected in a noisy environment, it was demonstrated that GMM-HMM provided much improvement in recognition performance over the conventional HMM scheme (Sonali Nemade, 2019; Taheri & Taheri, 2006).

2.4 Modern Deep Learning Approaches

2.4.1 Deep Neural Networks (DNN)

The use of DNN in conjunction with HMM, also known as DNN-HMM has been shown to improve performance in Japanese speech recognition tasks. Seki et al. (2014) compared syllable-based and phoneme-based DNN-HMM and found that the syllable-based DNN-HMM was better, as its parameter space is less coupled with the context of the syllables. They reported that an 11% relative decrease in the WER for triphone DNN-HMMs over syllable-based DNN-HMMs when used on large databases such as ASJ+JNAS. The multilayered structure of DNNs makes it much suitable for developing models of contextual dependencies for speech signals (Hojo et al., 2018). GMM-HMM models are less effective compared to DNN when the task involves the estimation of posterior probabilities. In particular, pre-training with restricted Boltzmann machines has been quite useful for weight initialization, the vanishing gradient problem, and overall performance (Masato Mimura et al., 2013).

Mu et al. (2020) developed a double-deep neural network for the evaluation of Japanese pronunciation to address the problems of text-to-speech alignment and scoring. The DDNN integrated CNN and RNNs with attention and it is effective for detecting pronunciation mistakes. Lin et al. (2017) noted the importance of addressing the particular problem of the lack of annotated Japanese speech corpora by emphasizing the use of transfer learning with DNN. First, pre-training on large universal datasets increases the generalization ability. Then, fine-tuning on Japanese databases enhances the performance that is critical in low-resource applications. The authors were also able to use CNN and recurrent architectures to attend to the granularity features of the Japanese language.

2.4.2 Convolutional Neural Networks (CNN)

There is difficulties in the visual speech recognition areas and specifically within lipreading because a limitation for the use of CNNs for phoneme recognition tasks was considered to be the number of training datasets (Noda et al., 2014). The research was conducted using elastic net regression on a seven-layer CNN structure and 58% of phoneme recognition accuracy was obtained for Japanese datasets. Building upon this work, Yalta et al. (2019) constructed a functional speech recognition framework inclusive of several types of words spoken intended for tight spots like houses. There are more focused methods for connecting microphones such as incorporating residual connections and batch denormalization.

Noda et al. (2014) investigated the use of CNNs for solving the problem of creating a Japanese speech acoustic model. CNN used to encode the frequency-time domain images and properly exploit the spatial and temporal aspects. The C-nets employed in this model aided in recognizing fine speech traits that improved performance in terms of recognition in contrast to the prevalent GMMs and HMMs methods. The combination of CNNs with attention mechanisms has yielded some results in the accurate detection of Japanese speech. This integration has been beneficial in increasing accuracy and interoperability during the detection of long utterances and multi-speaker datasets (Kohei Mukohara et al., 2015).

2.4.3 Recurrent Neural Networks (RNN)

In the work of Takeuchi et al. (2020), a novel design of the RNN is introduced, which enables the processing of input speech while removing noise caused by the room impulse response. This network mitigates the vanishing and exploding gradient problems often seen in RNNs while also keeping the parameter count low, making it very suitable for real-time applications. Yusuke Kida et al. (2016) investigated linear prediction filters based on LSTM. Their method trained an LSTM which did not require direct access to raw information and thus can extract features from distorted signals, as an LSTM estimated linear prediction coefficients.

Kubo (2014) broadened approaches incorporating RNNs into synthesizing speech for Japanese, particularly focusing on improving prosody and intonation. Their work underscored the necessity to consider the sequential modelling features of RNNs units, especially LSTMs, techniques for natural voice synthesis of Japanese language sounds. Takeuchi et al. (2020) took advantage of the RNN-based architectures for the acoustic modelling for Japanese ASR. They showed that even though GRUs have a simpler gating strategy than LSTMs, they could achieve a similar level of classification accuracy with lower compute requirements. Then, the studies on bidirectional LSTMs (BLSTMs). Imaizumi et al. (2022) revealed that they could utilise the past context and the future context of the signal for better performance of the speech recognition device. Many applications of automatic speech recognition in which the Japanese language is used have demonstrated that BLSTMs are particularly helpful for modelling complex phonological and prosodic structures of the Japanese language.

2.4.4 Convolutional-Recurrent DNN with Connectionist Temporal Classification (CRDNN-CTC)

The CRDNN-CTC architecture combines a convolutional front end, a recurrent middle block, and fully connected layers at the output, and is trained using the Connectionist Temporal Classification (CTC) objective. In this setup, the convolutional layers operate directly on spectral features such as MFCC or log-Mel filterbanks and learn local time-frequency patterns that are robust to small shifts due to noise or channel variation (Ravanelli et al., 2019). The recurrent layers (typically bidirectional LSTM or GRU) then model longer-range temporal dependencies and phonotactic structure across frames, capturing context that spans multiple morae or syllables in continuous Japanese speech (Ravanelli et al., 2019; Takeuchi et al., 2020). The final fully connected layers project these sequence representations to label posteriors (characters, kana, or subword units), and CTC is used to align the predictions to the transcription without requiring frame-level labels (Fujita et al., 2024).

A key property of CRDNN-CTC is that it removes the need for an explicit pronunciation lexicon or an HMM-based alignment stage during training. Instead, the model directly learns to map acoustic frames to symbol sequences under a monotonic alignment constraint enforced by CTC (Fujita et al., 2024). This is especially attractive for Japanese ASR because Japanese can be transcribed at the character or kana level, and word boundaries are often not explicitly marked in continuous speech. The CTC formulation avoids manual segmentation of long utterances and is tolerant of small timing mismatches, which simplifies data preparation for large-scale or semi-supervised corpora where detailed alignments are expensive to obtain (Watanabe et al., 2018).

CRDNN-CTC has also proven effective in settings where compute and latency matter. Compared to purely recurrent systems, the convolutional front end reduces redundancy by downsampling in time, which lowers the number of recurrent steps and thus reduces inference cost (Ravanelli et al., 2019). At the same time, compared to purely convolutional encoders, the recurrent block helps preserve long-span prosodic cues and coarticulation effects that are important in Japanese, such as vowel length and pitch accent, which affect meaning (Takeuchi et al., 2020). Toolkits such as PyTorch-Kaldi and SpeechBrain have standardized CRDNN-CTC style models as strong, lightweight baselines for character-level ASR in multiple languages, including Japanese broadcast and read speech, where they report competitive character error rates without resorting to very large transformer encoders (Ravanelli et al., 2019; Ravanelli et al., 2021; Watanabe et al., 2018).

2.5 Transformers Models in Japanese Speech Recognition

2.5.1 Transformer-based Models

Taniguchi et al. (2022) propose a series of Transformer-based ASR models aimed at improving Japanese speech recognition, particularly in the context of simultaneous interpretation. They investigate the possibility of utilizing auxiliary input like the source language text to resolve issues such as disfluencies, hesitations, and self-repairs commonly observed in the interpreter speech which helps to improve the transcription quality (Futami et al., 2020). The models combined audio and text data via multi-modal transformer encoders and decoders, which offers a broader scope of recognition by using previously provided source language text for interpreter training programs (Taniguchi et al., 2022).

A wide range of datasets for source text and simultaneous interpretation speech are however not readily available, so the authors use a adapted speech translation corpora from MuST-C and CoVoST 2 while also introducing TED based Japanese texts for evaluation purposes (Taniguchi et al., 2022). With an additional goal of enhancing performance, the authors fine-tune the source language text encoder by using large machine translation corpora which helps in lowering the word error rates during translation of English, Dutch, German and Japanese (Taniguchi et al., 2024). Results consistently demonstrate that incorporating source language text into Transformer-based ASR models significantly improves recognition performance, with the greatest impact observed when auxiliary input is introduced at later stages of the audio encoding and decoding process (Futami et al., 2020).

2.5.2 Whisper by OpenAI

Large scale weak supervision has emerged as one of the major approaches in speech recognition as noted by Radford et al. (2023) in their development of whisper model that has been trained on multilingual and multitask audio datasets that has a combined duration of 680,000 hours. This work is a continuation to the self-supervised methods such as Wav2Vec 2.0 (Baevski et al., 2020), which demonstrated learning without supervision from audio without any human-provided labels. However, dataset-specific fine-tuning is often necessary to obtain good performance, whereas with Whisper such reliance is reduced because of the efficacy of weak supervision.

By scaling weak supervision across diverse datasets, Whisper able to bypass the need for dataset-specific adaptation while able offer a robust zero-shot performance across languages and tasks. The authors also mentioned that by using this method, it will ensure the generalization and the robustness of the model while at the same time addressing main limitation in traditional models that is struggled to transcribe unfamiliar audio. This method also resulting in the models to have similar trends with other state of the art model in machine learning where a large, diverse datasets will improve model resilience which is align the with the advancements in computer vision (Kolesnikov et al., 2020) and NLP (Radford et al., 2019).The Whisper model's architecture, a simple encoder-decoder Transformer, reinforces the effectiveness of minimal preprocessing and sequence-to-sequence training, simplifying the transcription pipeline while achieving near-human-level accuracy.

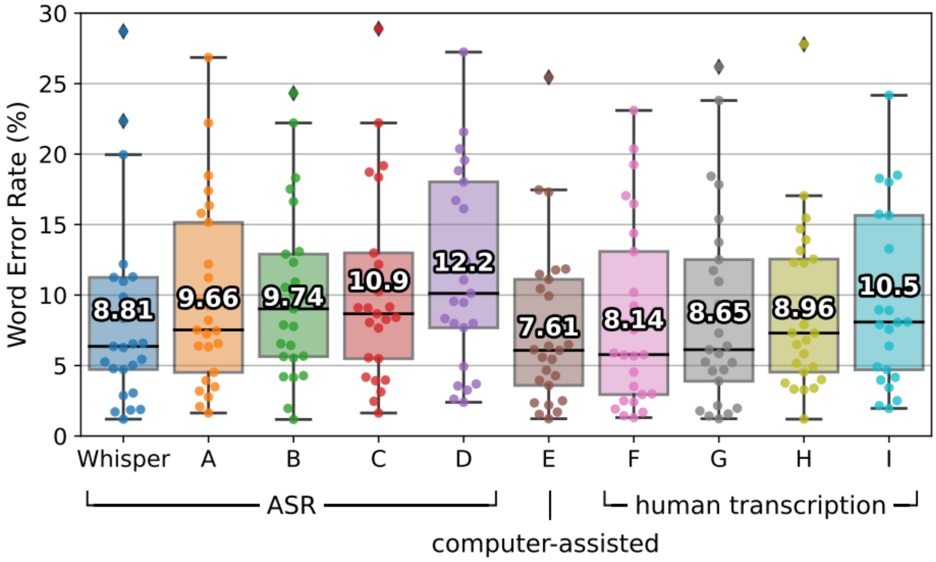


Figure 2.2 Whisper WER cited from Radford et al., 2023

Based on the work of Radford et al. (2023), there is further research that seeks to improve the performance of multilingual models on tasks that involve a japanese language. Bajo et al. (2024) detail their work on adapting OpenAI's Whisper model to enhance its performance in ASR for the Japanese language. The research draws attention to the dilemma faced in balancing the multilingual being and the accuracy of an English-only product, ReazonSpeech, that seeks to maximize on the Japanese language ASR, which is monolingual in nature. By using a Japanese dataset while utilizing Low-Rank Adaptation (LoRA) and fine-tuning methods, they were able to lower Whisper-Tiny's Cumulative Expenditure Rate (CER) from 32.7% to 14.7%. This fine tuning method showed that smaller multilingual models give more promising result, after being tuned for the desired language outperform their larger baseline models, for example the case of the Whisper-Base model (Bajo et al., 2024).

2.5.3 wav2vec 2.0 by Facebook AI Research

The research conducted by Baevski et al. (2020) proved that self-supervised learning greatly reduces the dependency on large amounts of labeled data in speech recognition. They achieved this by using a technique called wav2vec 2.0. With this method, models are trained over a significant set of unlabeled speech data by masking

Table 2.1
WER and CER performance of Whisper models. Reproduced from Bajo et al., 2024.

Model	WER (%)	CER (%)
Whisper Tiny	47.48	32.74
Whisper Base	29.81	20.20
Whisper Small	16.14	9.89
Whisper Medium	10.84	6.86
Whisper Large	7.41	4.77
Whisper Tiny + LoRA	33.16	20.83
Whisper Base + LoRA	23.36	14.50
Whisper Small + LoRA	14.90	9.16

the raw audio inputs and then treating a contrastive task. Thereafter, a model can be fine tuned using a limited set of labeled data which enables it to perform better when compared to semi-supervised techniques. Furthermore, according to Baevski et al. (2020), while their approach performed well with all the available labeled data by raising the WER to 1.8% for clean data and 3.3% for other data, it went even better with 10 mins of labeled and 53k hours of unlabeled data which had WER rates of 4.8% and 8.2%.

Table 2.2
WER on Librispeech dev/test sets using 10 minutes of labeled data and different unlabeled data setups.

Model	Unlabeled Data	LM	dev (clean)	test (other)
Discrete BERT	LS-960	4-gram	15.7	25.2
BASE	LS-960	4-gram+Transf.	8.9	15.6
		Transf.	6.6	12.9
LARGE	LS-960	Transf.	5.0	10.0
	LV-60k	Transf.	4.6	8.2
Highlighted Result	LV-60k (53k hours)	Transf.	4.8	8.2

In Japanese speech recognition, self-supervised learning (SSL) has emerged as one of the major tools for tackling the problems of dialectal diversity and low-resourced datasets. Miwa and Kai (2023) showcased what they refer to as successful adaptation of the wav2vec 2.0-based XLSR model to the Corpus of Japanese Dialects (COJADS), a collection of data capturing various dialects from different regions of Japan. They reported significant gains in ASR metrics for dialectal speech, achieving CER reductions of as much as 8.9% relatively to the models only trained on tagged data.

2.5.4 ChirpV2: an Universal speech model from Google

Zhang et al. (2023) demonstrated a novel technique that scales ASR to more than a hundred languages, this is achieved with the aid of large multilingual datasets with self-supervised learning, they refer to their model as the Universal speech model. The model was pretrained on 12 million hours of unlabelled audio data collection of 300 languages, in addition to 90 thousand hours of multilingual labelled audio data. One of the crucial innovations is BEST-RQ (BERT-based Speech pretraining with Random-projection Quantizer) because it improves the performance of speech representation without complicated quantization modules.

The model also outperformed specialized models including Whisper that have previously been trained with more data. In addition to this, chunk-wise attention is used to solve the performance drop-off problem that USM has with long audio, allowing USM to transcribe long audio. Other language resource enabling techniques such as noisy student training and adapter modules have enhanced USM performance with low resource and unseen languages considerably, as it did with low resource languages ensuring a robust ASR system (Zhang et al., 2023). USM proves the efficacy of self-supervised models in minimizing multilingualism and far supersedes existing standards for ASR systems.

Table 2.3
Word Error Rate (WER) Comparison of ASR Models

Dataset	USM-CTC (%)	USM-LAS (%)	Whisper (%)
YouTube (en-US)	13.7	14.4	17.7
YouTube (CORAAL)	18.7	19.0	27.8
SpeechStew (en-US)	26.7	29.8	-
FLEURS (62 languages)	12.1	11.2	13.2
Multilingual (YouTube)	15.5	12.5	23.9

2.6 Current Comparative Analysis of Japanese ASR Models

A comparative analysis that carried out by Karita et al. (2021) shows that Conformer-based models perform better than Conformer BLSTM architectures, as they obtained 4.1, 3.2, and 3.5 character error rates for CSJ in eval1, eval2, and eval3 tasks respectively. It is noted that both the BLSTM and Conformer models have character error rates below 7% and the character error rate is lower when using Conformer Itself. Conformer encoders also offer increased accuracy and efficiency, with a throughput of 628.4 utterances processed per second and 430.0 for the BLSTM models. The scope of the work also emphasizes the importance of the analysis of the specific problem of training parameters optimization, noting the importance of the implementation of SpecAugment, exponential moving average (EMA) and variational noise (VN). The SpecAugment technique results in the largest shifts which affect the performance. The integration of the Conformer transducers with the described set of training approaches surpasses all existing solutions in Japanese ASR and open the path for further development (Karita et al., 2021).

Table 2.4
Character error rates on CSJ dev/eval1/eval2/eval3 sets cited from Karita et al., 2021.

Encoder	Decoder	Param	Utt/sec	CER [%]
BLSTM	CTC	258M	430.0	3.9 / 5.2 / 3.7 / 4.0
BLSTM	attention	309M	365.5	3.8 / 5.3 / 3.7 / 3.7
BLSTM	transducer	274M	297.6	3.8 / 5.1 / 3.7 / 4.0
Conformer	CTC	117M	628.4	3.1 / 4.1 / 3.2 / 3.5
Conformer	attention	124M	534.8	3.3 / 4.5 / 3.3 / 3.5
Conformer	transducer	120M	376.1	3.1 / 4.1 / 3.2 / 3.5

Another comparative analysis in the domain of the Japanese language is presented by Takahashi et al. (2024), focusing on the accuracy of speech recognition for different dialects. The study evaluates three models: Whisper, XLSR, and XLS-R, which are self-supervised learning frameworks. The Whisper model significantly underperformed for any Japanese outside of standard Japanese, recording a 4.1% CER only after it has gone through fine tuning. However, when the accuracy is low when the language identification marker is absent where some instances of being higher than 100%.This marks the weakness of Whisper in terms of its application for wide ranging applications in different dialects of Japanese. However, Whisperer and XLS-R both of which were trained on multilingual speech data show improvement in the recognition of Japanese dialects. These models apply multi-task learning paradigms such as DID and ASR to increase their efficiency. Multi tasking adds significantly to the dialect accuracy and a three-step efficient training of the models reduce the CER by 3-4% relative to conventional transfer learning. Some dialects, especially those from Kyushu and Chubu, have larger CER than those spoken in Kanto, where there is a greater linguistic affinity (Takahashi et al., 2024).

Current comparative analysis from these studies shows that several challenges need to be addressed. A study to compare the existing models in Japanese ASR is needed because of the unique characteristics of the Japanese language. The comparative analysis from Karita et al. (2021) and Takahashi et al. (2024) shows that while models like Conformer and Whisper have made significant strides in ASR, they still face challenges in handling the complexities of Japanese. The results indicate that while Conformer-based models excel in standard Japanese. Similarly, Whisper's performance is notably affected by the presence or absence of language identification markers. These findings underscore the need for further research and development to enhance the adaptability and accuracy of ASR systems in Japanese language contexts.

Table 2.5
Comparison of ASR accuracy on two datasets, Standard Japanese (CSJ) and Japanese dialects (COJADS) cited from Takahashi et al., 2024.

Pre-Trained Model Name	Adaptation Method	CER [%] CSJ	CER [%] COJADS
Whisper-medium (zeroshot)	-	25.6*	116.0*
Whisper-medium	full finetuning	4.1	32.9
XLSR	full finetuning	6.5	34.1
XLSR	3-steps finetuning	-	30.0
XLS-R	full finetuning	6.1	32.6
XLS-R	3-steps finetuning	-	29.2

*After post-processing with kanji-to-kana conversion.

2.7 Datasets and Tools

2.7.1 Datasets

Dataset is a crucial component in training and evaluating ASR models. In this study, the dataset that will be used is JSUT, which is a large-scale Japanese speech corpus that contains over 10 hours of read speech from 100 speakers (sonobe2017jsut). The dataset includes a variety of speech styles, such as formal and informal speech, and covers a wide range of topics. The JSUT dataset is suitable for training ASR models because it provides a diverse set of speech samples that can help improve the model's ability to generalize to different speakers and speech styles.

2.7.2 Python

Python is a popular programming language that is widely used in the field of machine learning and natural language processing. It provides a variety of libraries and frameworks that can be used to develop ASR models, such as TensorFlow, PyTorch, and Keras (Boishakhi et al., 2021). Python also has a large community of developers who contribute to open-source projects, making it easy to find resources and support for developing ASR models. In this study, Python will be used to implement the ASR models and to preprocess the dataset.

2.7.3 yt-dlp

yt-dlp is a command-line tool that allows users to download videos from various websites, including YouTube. It is a fork of the popular youtube-dl tool and provides additional features and improvements (yt-dlp). In this study, yt-dlp will be used to download TED talk videos that will be used as part of the dataset for training and evaluating the ASR models. The audio from the downloaded videos will be extracted and processed to create a suitable dataset for ASR model training.

2.7.4 Kaldi

Kaldi is an open-source toolkit for speech recognition that provides a wide range of tools and algorithms for developing ASR models (povey2011kaldi). It includes tools for feature extraction, acoustic modeling, and decoding, as well as support for various machine learning algorithms. Kaldi is widely used in the research community and has been used to develop state-of-the-art ASR models for various languages. In this study, Kaldi will be used to implement and evaluate the ASR models for Japanese speech recognition.

2.7.5 speechbrain

SpeechBrain is an open-source toolkit for speech processing that provides a wide range of tools and algorithms for developing ASR models (Ravanelli et al., 2021). It includes tools for feature extraction, acoustic modeling, and decoding, as well as support for various machine learning algorithms. SpeechBrain is designed to be easy to use and provides a modular architecture that allows users to easily customize and extend the toolkit. In this study, SpeechBrain will be used to implement and evaluate the ASR models for Japanese speech recognition.

2.7.6 Hugging Face

Hugging Face is a platform that provides a wide range of pre-trained models for natural language processing tasks, including speech recognition. It offers a variety of models that can be fine-tuned on custom datasets to improve performance on specific tasks (Boishakhi et al., 2021). In this study, Hugging Face will be used to access pre-trained models for Japanese ASR. These model will be downloaded from Hugging Face and analyze the performance of the models based on the dataset used in this study.

2.8 Gaps in Literature

Based on the literature review, shows that there is several gaps in the research on Japanese ASR. One of the area is the insufficient exploration of how state-of-the-art models can be adapted to the specific nuances of the Japanese language. Although there is few study that evaluate the performance of these model, but there is no notable research that focusing on evaluating the state of the art models in Japanese ASR. Another research gap from the literature review is the lack of focus on the performance of these models when dealing with dialectical variations of Japanese. Although research has been conducted on their effectiveness with standard Japanese, there is a clear need for further work on how these newly developed models perform when handling dialectical speech. Lastly, another gap identified is the underexplored area of these models application in real-time environments. Despite some investigations into their use in real-time scenarios, additional research is required to optimize these models for applications where quick response times are essential.

2.9 Conclusion

This chapter highlighted the evolution of traditional ASR model to the cutting-edge technologies also has been highlighted in this chapter. Despite the advancements of the model and ASR framework, there is still a gap in adapting these models to Japanese-specific contexts. There is still work to be done to further refine the accuracy, speed, and dataset availability to advance Japanese ASR. This result can be used as a foundation for developing more effective and inclusive speech-to-text solutions tailored for Japanese language. For the dataset and tools, TED talks, CSJ, and CO-JADS are the most commonly used datasets for Japanese ASR research. To extract audio from video files, Python Moviepy is used and to access pre-trained models for Japanese ASR, the model will be obtain from Hugging Face.

CHAPTER THREE
RESEARCH METHODOLOGY

3.1 Introduction

This chapter explained the methodology used to evaluate Japanese automatic speech recognition (ASR) across three model families: a traditional Kaldi-style GMM–HMM system, an end-to-end CRDNN–CTC model, and a fine-tuned transformer encoder–decoder model based on Whisper model from OpenAI. This chapter described the data pipeline such as data collection, audio and transcript cleaning, and audio splits. The preprocessing, training, decoding process and the scoring protocol used for each model family was also discussed in this chapter.

3.2 Research Design

This study was organised into six phases where started from planning and followed by reviewing prior work. And then moved to data collection and preprocessing phase where dataset was compiled and processed. After that, the selcted model were trained and fine tuned using the data. Then the model was scored using selected metrics and the the data finally were analyzed and discussed in the later phase. ?? below showed an overview of the phases, activities and deliverables.

Table 3.1
Overview of Research Project

Phase	Activities	Deliverables
Planning	Defined below items: • Project title • Problem statements • Objectives • Scopes • Significance	Chapter 1 • Title defined • Problems defined • Objectives defined • Scope defined • Significance defined
Prior Work	• Reviewed Japanese ASR studies • Listed preprocessing methods • Listed model families • Listed evaluation metrics • Summarized key gaps	Chapter 2 • Literature review written • Preprocessing chosen • Models selected • Metrics defined • Gaps summarized
Data Collection and preprocessing	• Collected Japanese TEDx talks • Downloaded audio and subtitles • Cleaned and normalized transcripts • Resampled audio to 16 kHz mono • Split audio using VAD • Created train/valid/test splits • Exported manifests and metadata	Chapter 3 • Dataset compiled • Collection method documented • Transcripts normalized • Segments generated • Splits finalized • Manifests prepared
Model Training and Fine Tune	• Trained GMM-HMM • Trained CRDNN-CTC • Fine-tuned Whisper variants • Set decoding parameters • Saved configs and checkpoints	Chapter 4 • Models trained • Checkpoints saved • Settings recorded • Outputs generated
Evaluation	• Scored with CER and WER • Measured speed using RTF • Compared models and decoders	Chapter 4 • Results tables and plots • Best model identified
Discussion	• Explained main findings • Linked to prior work • Noted limitations • Suggested future work	Chapter 5 • Findings discussed • Limitations stated • Recommendations given • Future work proposed

3.3 Planning Phase

?? showed the first phase of this project where most of the planning and decision about this projectis done. The activities carried out and the outcome deliverables were described in the table shown below.

Table 3.2
Planning Phase

Phase	Activities	Deliverables
Planning	Defined below items: • Project title • Problem statements • Objectives • Scopes • Significance	Chapter 1 • Title defined • Problems defined • Objectives defined • Scope defined • Significance defined

The starting point of this project began in the planning phase. This phase was important because it served as the blueprint and became guidance to the project direction. This phase purpose was to identify the problem in the domain and found a way to solve the problem stated. The activities that was carried out in this pahse was defining key aspect that was the title of the project, problem statements, objective, scope and significance of the project. The outcome from this phase was the title of the project, problems that has been defined, objective of the project, scope of the project which explained what was covered and what was not covered in this project and last deliverables in this phase was the significance that has been identified. During this planning phase, things like time, cost, resource, benefit and difficulty level also have been take into consideration.

3.4 Prior Work Phase

?? showed the second phase of this project where prior work in the domain of ASR was reviewed and some decision was made. The activities carried out and the outcome deliverables were described in the table shown below.

Table 3.3
Prior Work Phase

Phase	Activities	Deliverables
Prior Work	• Reviewed Japanese ASR studies • Listed preprocessing methods • Listed model families • Listed evaluation metrics • Summarized key gaps	Chapter 2 • Literature review written • Preprocessing chosen • Models selected • Metrics defined • Gaps summarized

After planning phase was completed and the reasearch domain has been set, next phase was reviewing prior work phase. During this phase, multiple sources such as related article, journal, conference paper and textbook was reviewed. This resource was used to evaluate and identified what has been discussed or discovered. In this phase, the activity involved was list down the preprocessing mrthod that currently being used in speech recognition and a few preprocessing technique was choosen based on the requirements of the models as the deliverables. The next activity was to choose which model to compared to, instead of comparing model from same family, this research has conclude to compare 3 model from different model architecture and these 3 models was the deliverables for this phase. The next activity was to conclude which metrics was used to compare these models. As the deliverables, three metrics has been choose to do comparitive analysis between each models. In this phase, the gaps in the literature was also defined and as the deliverables the gaps in the literature was summarized.

3.5 Data Collection and preprocessing Phase

?? showed the third phase of this project where the source of the data was selected, aqquired and then the data went through preprocessing to make sure data was ready to be input into models. The activities carried out and the outcome deliverables were described in the table shown below.

Table 3.4
Data Collection and preprocessing Phase

Phase	Activities	Deliverables
Data Collection and preprocessing	• Collected Japanese TEDx talks • Downloaded audio and subtitles • Cleaned and normalized transcripts • Resampled audio to 16 kHz mono • Split audio using VAD • Created train/valid/test splits • Exported manifests and metadata	Chapter 3 • Dataset compiled • Collection method documented • Transcripts normalized • Segments generated • Splits finalized • Manifests prepared

The focus of this phase was to build a cleaned and useable dataset from publicly available dataset into a consistent training and evaluation format so that all three models from different family can use the sama dataset for traiing and testing. The pipeline for this phase could be seperated into four section. The first task was collecting TEDx Japanese talks from YouTube with only Japanese subtitles that came with manual subtitles by human. Then the audio was downloaded with the subtitle files. The second task was to clean and normalize subtitle text into reference transcripts and standardized the audio into 16 kHz mono PCM. Third task was to split the audio into chunks using voice activity detection (VAD) and the audio that did not have audio activity such as intro music was discarded. The last task was to produce train/validation/test splits and manifests with audio paths, durations, and transcripts.

Figure 3.1 Placeholder: End-to-end data collection and preprocessing pipeline for TEDx Japanese talks (YouTube → cleaned transcripts → VAD chunks → manifests).

3.5.1 Data Source and Selection Criteria

The source of the data was collected from Japanese TEDx talks that was hosted on video sharing platform, Youtube. This particular source was choosen beacuse TEDx talks was a collection of public speaking where commonly the speaker had a clear speech with structured monologue-style delivery. However, not all video that published came with manual Japanese subtitle by human, only part of the video had manual Japanese subtitle. The rest either did not have subtitle or the subtitle was in other language. To make sure that the transcripts were sufficiently reliable to serve as reference text during ASR training and evaluation, only videos that had Japanese manual subtitle was choosen and the rest was excluded. The videos that contained only auto-generated subtitles were also excluded to reduce transcription noise and alignment inconsistencies.

All audio and subtitle was downloaded using a python library name yt-dl. The script first queried the video metadata to find if the video in playlist had Japanese subtitle or did not without downloading the video using `extract_info` function from yt-dl library.

Listing 3.1: Audio splitting script

```
with yt_dlp.YoutubeDL(check_opts) as ydl:
    info = ydl.extract_info(video_url, download=False)

    # Check manual subtitles
    if 'subtitles' in info and 'ja' in info['subtitles']:
        has_jp_subs = True

    # Check auto-generated subtitles (ASR)
    elif ('automatic_captions' in info
          and 'ja' in info['automatic_captions']):
        pass
```

For the video that had Japenese subtitle, audio and transcript was downloaded

Listing 3.2: Audio splitting script

```
download_opts = {
    'subtitleslangs': ['ja'],
    'subtitlesformat': 'vtt',
}
if has_jp_subs:
    with yt_dlp.YoutubeDL(download_opts) as ydl:
        ydl.download([video_url])
```

3.5.2 Transcript Cleaning and Normalization

After the transcript was downloaded, the transcript needed to be cleaned since it could not be directly used as reference transcript. The raw downloaded trasncriptt contained non-speech markers, formatting tags, and artifacts introduced for readability rather than ASR training. All subtitle files were normalized into cleaned text using a custom script. The cleaning rules implemented included:

- Removing HTML and formatting tags (e.g., `<i>`, brace tags).
- Removing speaker labels (e.g., `Speaker 1:`, `NARRATOR:`).
- Removing non-speech annotations inside brackets or parentheses when they represent events such as `[laughter]`, `(applause)`, `[music]`, or `(inaudible)`.
- Normalizing whitespace and punctuation (e.g., collapsing multiple spaces, removing spaces before punctuation).

The script produced a transcript file where one cleaned line per subtitle cue with the timestamp which was useful for later alignment between subtitle cues and speech segments. The timestamp was used to seperate the long audio format into chunks.

3.5.3 Audio Standardization to 16 kHz Mono

The downloaded audio may came with different format based on the download option and video quality. To ensure the compatibility with all selected model, the audio was standardized into 16 kHz Mono. This was also to make sure the fairness across all ASR model families when compare to each other. The standardized choosen was 16 kHz sample rate, mono channel, 16-bit PCM because this format was a standard configuration used in Kaldi recipes and also aligned with typical front-end assumptions in many end-to-end ASR pipelines.

Listing 3.3: Audio resampling to 16 kHz

```python
import librosa, soundfile as sf

def resample_16khz(in_wav, out_wav, target_sr=16000):
    y, sr = librosa.load(in_wav, sr=None)
    if sr != target_sr:
        y = librosa.resample(y, orig_sr=sr, target_sr=target_sr)
        sr = target_sr
    sf.write(out_wav, y, sr)
```

3.5.4 Speech Segmentation using WebRTC VAD

The downloaded audio was averaging between 20 minutes to 30 minutes long. This duration was not suitable to be used as trining data because it was chalenging for the model to learn effectively from very long utterances, and it also increased memory usage and training time. To deal with this issue, WebRTC voice activity detection (VAD) was used to split a long audio files into chunks audio that suitable to be used in training phase. The segmentation strategy was:

- Audio was split into 30 ms frames.
- Speech frames were grouped into continuous speech regions.
- A region was closed when silence exceeded a threshold 500 ms.
- Very short segments were removed $\leq$2 s.
- Very long segments were further sliced into $\leq$20 s chunks.

The created a lot of short audio files and to manage all the chunks of file, below naming convention was used to make sure the original audio name still available

<originalname>_chunk_000.wav, <originalname>_chunk_001.wav, . . .

This naming convention was later reused as utterance IDs in the transcript mapping and manifests.

3.5.5 Remove non speaker audio

To improve the quality of the dataset, a filtering step was applied to remove the audio that did not have any Japanese speaker in it. This was because this audio became noise and could cause the the quality of the training data become lower. The type of audio that removed was audio clips that dominated by applause/laughter, heavy background noise, or non-Japanese speech. A python script was created to filter out this type of audio by applied three checks:

- Speaker structure check (pyannote diarization): clips were preferred when a single dominant speaker is present (high dominant speaker ratio).
- Acoustic event detection (CLAP zero-shot audio classification): clips with high confidence of laughter or clapping were removed based on threshold scores.

Listing 3.4: Automatic audio quality filtering

```
diar = Pipeline.from_pretrained("pyannote/speaker-diarization-3.1",
    use_auth_token=HF_TOKEN)
clap = hf_pipeline("zero-shot-audio-classification",
    model="laion/clap-htsat-fused",device=0 )

for audio_path in wav_files:
    # keep only clear speech (no laughter/clapping)
    dom_ratio = dominant_speaker_ratio(diar(audio_path))
    scores = clap(audio_path,
        candidate_labels=["people laughing", "applause or hand clap"],
        top_k=None)
    clear = ((dom_ratio >= 0.85) and (score(scores, "laugh") < 0.30)
        and (score(scores, "clap") < 0.30))

    if not clear:
        os.remove(audio_path)
```

Files failing the clarity or language criteria were deleted automatically, leaving a cleaner set of speech segments for the downstream ASR experiments.

3.5.6 Transcript-to-Audio Mapping and Manifest Preparation

The next step after splitting the audio into chunks and cleaning the subtitle was to align the audio with a matching reference transcript to enable supervised training and evaluation. The cleaned subtitle text served as the transcript source and the final transcript mapping was stored in a simple format like this:

```
utt_id:  transcript
```

where `utt_id` matched the chunk filename and the `transcript` was the clean subtitle file that has been extracted the value and aligned based on the timestamp in the subtitle file. After reference transcript was aligned with audio file, for training and evaluation reproducibility, metadata manifests were generated containing:

- `utt_id` (utterance identifier),
- `wav` (absolute path to the audio file),
- `duration` (in seconds),
- `transcript` (cleaned reference text).

3.5.7 Train/Validation/Test Splits

The final dataset was divided into three splits to support model development and unbiased evaluation. The split ratio used was 80% training, 10% validation, and 10% testing. The script implemented a two-step split using train test split:

1. Split the full set into 80% train and 20% temporary.
2. Split the temporary set into 50% validation and 50% test (10% each).

The data was split like this beacuse the training set must be large enough for the models to learn the language patterns, while the validation set was required to tune hyperparameters and select checkpoints without leaking information from the test set. The test set was kept completely unseen until the final evaluation to provide an unbiased estimate of real-world transcription performance.

3.6 Model Training and Fine Tune Phase

?? showed the fourth phase of this project where cleaned audio and transcript earlier was used to train or fine tune models. The main objective of this phase was to make sure each model was trained using the same data split so that the evaluation in the next phase was fair and consistent.

Table 3.5
Model Training and Fine Tune Phase

Phase	Activities	Deliverables
Model Training and Fine Tune	• Trained GMM-HMM • Trained CRDNN-CTC • Fine-tuned Whisper variants • Set decoding parameters • Saved configs and checkpoints	Chapter 4 • Models trained • Checkpoints saved • Settings recorded • Outputs generated

There model was used in this phase that was which was a traditional Kaldi GMM–HMM, an end-to-end CRDNN–CTC using SpeechBrain framework that was trained from zero while a transformer encoder–decoder Whisper model from OpenAI was fine tuned using the same dataset.

3.6.1 GMM–HMM Training (Kaldi)

The first model family was the traditional Kaldi-style GMM–HMM system. This model was trained using a standard Kaldi recipe flow which started from feature extraction, followed by acoustic model training in multiple stages (mono, tri1, tri2, tri3). In this experiment, MFCC features with CMVN normalization were used because it was a common baseline setup in Kaldi and suitable for classical HMM-based modelling.

Figure 3.2 Placeholder: Model training workflow for GMMHMM

3.6.1.1 Lexicon, Dictionary, and Language Preparation

A pronunciation dictionary and lexicon were prepared using a custom script, `prepare_dict.py`. The script extracted a vocabulary list from the training transcripts and generated a grapheme-based lexicon by converting each Japanese token into Hepburn romanisation using `pykakasi`. The romanised output was normalised to lowercase and filtered to `a-z` characters only. Each word was then mapped to a sequence of letters, which served as the basic pronunciation units (phones).

Listing 3.5: Grapheme-based lexicon generation (Japanese → Hepburn letters)

```python
from pykakasi import kakasi

k = kakasi()
k.setMode('J','a'); k.setMode('K','a'); k.setMode('H','a'); k.setMode('r','Hepburn')
conv = k.getConverter()

for w in vocab:  # unique tokens from training transcripts
    roma = "".join(x["hepburn"] for x in conv.convert(w)).lower()
    phones = list(re.sub(r"[^a-z]", "", roma))      # keep a-z only
    if phones:
        lexicon.write(f"{w} {' '.join(phones)}\n")  # lexicon
        phone_set.update(phones)                    # nonsilence_phones
```

Next, an unknown token `<unk>` was inserted into the lexicon to handle out-of-vocabulary (OOV) terms during decoding. This was for silent between speech or unknown features when decoding. After that, a language directory `data/lang` which contained symbol tables and language resources, was created using kaldi internal receipe. The output of this step was a complete language directory that included `words.txt`, `L.fst`, and other required files used by Kaldi for graph construction.

3.6.1.2 Feature Extraction (MFCC + CMVN)

MFCC (Mel-Frequency Cepstral Coefficients) was a compact numeric features that extracted from audio to represent the way human hear sound. It converted audio into numerical value and used as input feature in the ASR system. MFCC features were extracted for train, dev, and test sets using `steps/make_mfcc.sh` from kaldi receipe.

After that, CMVN (Cepstral Mean and Variance Normalization) statistics were computed using `steps/compute_cmvn_stats.sh`. CVMN was a normalization step that applied to the MFCC to make the features more consistent by removing channel/mic/loudness differences. It also applied variance normalization for scale the unit variance and reduced scale differences.Below was the script to run MFCC and CMVN:

Listing 3.6: Grapheme-based lexicon generation (Japanese → Hepburn letters)

```
echo "=== Extracting MFCCs & CMVN ==="
steps/make_mfcc.sh --mfcc-config conf/mfcc.conf --nj "$NJ_MFCC_TRAIN"
  --cmd run.pl data/train exp/make_mfcc/train mfcc
steps/make_mfcc.sh --mfcc-config conf/mfcc.conf --nj "$NJ_MFCC_DEV"
  --cmd run.pl data/dev   exp/make_mfcc/dev   mfcc
steps/make_mfcc.sh --mfcc-config conf/mfcc.conf --nj "$NJ_MFCC_TEST"
  --cmd run.pl data/test  exp/make_mfcc/test  mfcc

steps/compute_cmvn_stats.sh data/train exp/make_mfcc/train mfcc
steps/compute_cmvn_stats.sh data/dev   exp/make_mfcc/dev   mfcc
steps/compute_cmvn_stats.sh data/test  exp/make_mfcc/test  mfcc
```

3.6.1.3 Acoustic Model Training Stages

To train the acoustic model, the training was done gradually to improve the performance. The acoutic model was train using 4 stages as described below:

- Monophone (mono): a baseline context-independent model trained using `train_mono.sh`.
- Triphone 1 (tri1): a delta-based triphone model trained using `train_deltas.sh`.
- Triphone 2 (tri2): a triphone model with LDA+MLLT transforms trained using `train_lda_mllt.sh`.

- Triphone 3 (tri3): a speaker-adaptive training (SAT) model trained using `train_sat.sh`.

Between each stage, the training set was aligned using `align_si.sh` to produce improved alignments for the next model. This staged approach was important in Kaldi because better alignments usually led to better acoustic models. Below was the snippet of the bash script used to run the training receipe from Kaldi library.

Listing 3.7: Audio resampling to 16 kHz

```bash
echo "=== Training mono ==="
steps/train_mono.sh --nj "$NJ_TRAIN" --cmd run.pl \
  data/train data/lang exp/mono

echo "=== Aligning with mono -> mono_ali ==="
steps/align_si.sh --nj "$NJ_ALIGN" --cmd run.pl \
  data/train data/lang exp/mono exp/mono_ali

echo "=== Training tri1 (deltas) ==="
steps/train_deltas.sh 2000 10000 \
  data/train data/lang exp/mono_ali exp/tri1

echo "=== Aligning with tri1 -> tri1_ali ==="
steps/align_si.sh --nj "$NJ_ALIGN" --cmd run.pl \
  data/train data/lang exp/tri1 exp/tri1_ali

echo "=== Training tri2 (LDA+MLLT) ==="
steps/train_lda_mllt.sh 2500 15000 \
  data/train data/lang exp/tri1_ali exp/tri2

echo "=== Aligning with tri2 -> tri2_ali ==="
steps/align_si.sh --nj "$NJ_ALIGN" --cmd run.pl \
  data/train data/lang exp/tri2 exp/tri2_ali

echo "=== Training tri3 (SAT) ==="
steps/train_sat.sh 3500 20000 \
  data/train data/lang exp/tri2_ali exp/tri3
```

3.6.1.4 Language Model Construction and Decoding Graph

For decoding using a 3-gram, a language model was created using KenLM. An external language model from wikipedia dump was used because it had a huge number of vocabluary to cover huge amount of language pattern. After that `lmplz` was used to produce an ARPA LM, and Kaldi tools were used to convert the ARPA file into `G.fst`.

Finally, decoding graphs were created for each acoustic model using `utils/mkgraph.sh`. This produced graphs such as `exp/mono/graph`, `exp/tri1/graph`, and so on, which were used for decoding the test set.

Listing 3.8: Audio resampling to 16 kHz

```
echo "=== Making decoding graphs for each system ==="
utils/mkgraph.sh data/lang exp/mono exp/mono/graph
utils/mkgraph.sh data/lang exp/tri1 exp/tri1/graph
utils/mkgraph.sh data/lang exp/tri2 exp/tri2/graph
utils/mkgraph.sh data/lang exp/tri3 exp/tri3/graph
```

3.6.2 CRDNN–CTC Training (SpeechBrain)

The second family model was from neural network model and CRDNNCTC model has been choosen based on the literature review. This model was trained using SpeechBrain framework with the goal was to learn the direct mapping from acoustic feature and output character sequesnce. For the decoder setup, CTC was used because Japanese transcripts could be handled naturally as a sequence of characters, and it avoided the need for word segmentation.

Figure 3.3 Placeholder: Model training workflow for CRDNN

3.6.2.1 Manifest Preparation

Similar to GMM-HMM setup earlier, CRDNN-CTC setup also utilized the same CSV manifests that included `ID`, `wav`, `duration`, and `transcript`. The manifests were generated from the prepared wav directory and transcript mapping file. The same 80/10/10 split strategy train/valid/test was used.

3.6.2.2 Character Vocabulary (CTC Token Set)

Next step was to create a character vocabuluary from the training script by scanning all characters in the training set and writing them into a charset file. The vocabulary file began with a special `<blank>` token to represent the CTC blank symbol. After that this charset was loaded into SpeechBrain using `CTCTextEncoder`. This step was important because CTC required a fixed label set, and the model output layer size depended directly on the number of characters in this vocabulary. By generating the charset from the training transcripts, the label set stayed consistent with the dataset and reduced unexpected errors when unseen symbols appeared during training.

Listing 3.9: Grapheme-based lexicon generation (Japanese → Hepburn letters)

```python
# (1) Build charset.txt (CTC vocab)
import pandas as pd, unicodedata as ud
from collections import Counter
from pathlib import Path

df = pd.read_csv("manifests/jsut_train.csv")
```

```
chars = Counter(ch for t in df.transcript.astype(str)
                  for ch in ud.normalize("NFKC", t).strip() if ch != " ")

Path("tokenizer").mkdir(exist_ok=True)
with open("tokenizer/charset.txt", "w", encoding="utf-8") as f:
    f.write("<blank>\n" + "\n".join(sorted(k for k in chars if k.strip()
        )) + "\n")
```

3.6.2.3 Model Configuration and Hyperparameters

The CRDNN model hyperparameters were defined in `hyperparams.yaml`. In this experiment, filterbank (FBank) features with mean-variance normalization were used. The architecture consisted of CNN blocks, followed by bidirectional LSTM layers, and a DNN block before a final linear layer that projected to the CTC vocabulary size. The optimizer used was Adam with a learning rate of 1×10^{-3}, and learning rate scheduling was handled using NewBob scheduling based on validation improvement. Dropout was also applied to reduce overfitting because the dataset size was limited compared to large-scale ASR corpora. In addition, batch sizes for training and validation were configured separately to make sure validation could run stably under limited GPU memory.

3.6.2.4 Training Procedure and Checkpointing

A custom SpeechBrain `Brain` class (`ASRBrain`) was implemented to define forward computation, CTC loss, and CER tracking during validation. During training, checkpoints were saved using SpeechBrain checkpointer so that the best model could be recovered and used during evaluation. The model was trained for a fixed number of epochs, and validation CER was monitored across epochs. The learning rate was automatically adjusted when validation improvement became small, which helped stabilize training and avoided over-updating the model. At the end of training, the best checkpoint was selected based on validation performance

3.6.3 Whisper Fine-Tuning (Transformer Encoder–Decoder)

The third model family was Whisper, a transformer encoder–decoder ASR model from OpenAI. In this research, multiple Whisper variants were fine tuned (tiny, base, small, medium, and large-turbo). Whisper was fine tuned using supervised learning by providing audio features as inputs and reference transcripts as labels.

Figure 3.4 Placeholder: Model training workflow for Whisper

3.6.3.1 Dataset Preparation for Fine-Tuning

The dataset was prepared by converting `transcript_utf8.txt` into a DataFrame that mapped each utterance ID to its corresponding `.wav` path and transcript. The data was shuffled using a fixed random seed for reproducibility and split into training, validation, and testing sets. Each split was stored as CSV files (`train.csv`, `valid.csv`, `test.csv`) and later loaded using HuggingFace `datasets`.

3.6.3.2 Feature Extraction and Label Encoding

Whisper used log-Mel filterbank features extracted by the Whisper feature extractor. In preprocessing, each audio sample was resampled to 16 kHz if needed and converted into `input_features`. The transcript text was tokenized using Whisper tokenizer to generate `labels`. To avoid extremely long samples affecting training stability, audio longer than a fixed threshold (example: 30 seconds) was filtered out before training.

3.6.3.3 Fine-Tuning Configuration

For fine tuning, a pretrained Whisper checkpoint (example: `openai/whisper-base`) was loaded. The decoder was configured with Japanese language prompts using forced decoder IDs (`language="japanese", task="transcribe"`). Gradient checkpointing-related settings were applied by setting `use_cache=False`. Training used AdamW optimizer with a small learning rate (example: 1×10^{-5}) and cosine learning rate scheduling with warmup. Mixed precision (FP16) training was enabled when running on GPU to speed up training and reduce memory usage.

Listing 3.10: Grapheme-based lexicon generation (Japanese → Hepburn letters)

```python
from transformers import WhisperProcessor, WhisperForConditionalGeneration

MODEL_NAME = "openai/whisper-medium"  # later you can try tiny/base/medium/etc.

processor = WhisperProcessor.from_pretrained(MODEL_NAME)
model = WhisperForConditionalGeneration.from_pretrained(MODEL_NAME)

# configure language + task
forced_decoder_ids = processor.get_decoder_prompt_ids(
    language="japanese",
    task="transcribe",
)
model.config.forced_decoder_ids = forced_decoder_ids
model.config.suppress_tokens = []    # often helps for fine-tuning
model.config.use_cache = False       # must be False for gradient checkpointing
```

3.6.3.4 Training Loop and Validation

Training was performed for 10 epochs using mini-batch gradient updates. During training, the loss value was monitored at each step and average loss was computed at the end of each epoch. After training, validation decoding was performed using `model.generate()` and metrics were computed using WER and CER. The model and processor were then saved for later decoding and test evaluation.

3.6.3.5 Model Saving and Inference Check

After fine-tuning completed, the model weights and processor configuration were saved using `save_pretrained`. A quick inference check was performed by decoding a sample from the test split and comparing the predicted transcription against the ground truth reference. This step was important to confirm that the model and tokenizer were saved correctly and could be loaded again for the evaluation phase.

3.7 Evaluation Phase

?? showed the fifth phase of this project where all trained models were evaluated using the same test split and the same scoring protocol. The main objective of this phase was to measure transcription accuracy and decoding efficiency in a consistent and reproducible way so that comparisons across the three model families were fair. The activities carried out and the outcome deliverables were described in the table shown below.

Table 3.6
Evaluation Phase

Phase	Activities	Deliverables
Evaluation	• Scored with CER and WER • Measured speed using RTF • Compared models and decoders	Chapter 4 • Results tables and plots • Best model identified

In this phase, each trained system produced hypotheses on the test split and those hypotheses were scored against the same reference transcripts. The evaluation used three metrics that matched the research objectives that is character error rate (CER), word error rate (WER), and real-time factor (RTF).

3.7.1 Decoding and Hypothesis Generation

For the Kaldi GMM–HMM systems, decoding was performed using the test set features and the corresponding decoding graphs created earlier. Hypotheses were produced for each acoustic model stage (mono, tri1, tri2, tri3) using the standard Kaldi decoding pipeline. For the CRDNN–CTC system, decoding was performed using CTC decoding and hypotheses were generated for the test set, including the beam decoding variant when configured. For Whisper variants, hypotheses were generated using the HuggingFace `model.generate()` decoding procedure, and the output token sequences were converted back into Japanese text using the Whisper tokenizer.

3.7.2 Scoring Protocol (CER and WER)

After hypotheses were generated, each prediction was aligned against the reference transcripts and scored using CER and WER. CER was computed by measuring the normalized Levenshtein distance between predicted and reference character sequences. WER was computed using the same edit distance approach but applied at the word level after tokenization. The scoring protocol was applied consistently across all three model families so that differences in results reflected model performance rather than differences in evaluation handling.

3.7.3 Speed Measurement (RTF)

Decoding speed was evaluated using real-time factor (RTF), where total decoding wall time was divided by total audio duration. RTF values below 1.0 indicated faster-than-real-time decoding, while values above 1.0 indicated slower-than-real-time decoding. RTF was recorded for each system to compare computational efficiency, and the same measurement approach was applied across models under the same hardware environment to support fair comparisons.

3.8 Discussion Phase

?? showed the sixth phase of this project where the evaluation outcomes were interpreted and discussed in relation to the research questions and prior work. The main objective of this phase was to analyze the trends observed across the three model families, explain the trade-offs between accuracy and speed, and outline the limitations and future improvements. The activities carried out and the outcome deliverables were described in the table shown below.

Table 3.7
Discussion Phase

Phase	Activities	Deliverables
Discussion	• Explained main findings • Linked to prior work • Noted limitations • Suggested future work	Chapter 5 • Findings discussed • Limitations stated • Recommendations given • Future work proposed

In this phase, the results from CER, WER, and RTF were analyzed to explain how each model family behaved under the same dataset and evaluation setup. The discussion compared classical and neural approaches by focusing on how performance improved from mono to tri3 in the Kaldi pipeline, how CRDNN–CTC handled Japanese character sequences under end-to-end training, and how Whisper variants achieved different accuracy and speed trade-offs depending on model size. The outcomes were linked back to prior work to show whether observed trends matched or differed from findings in the literature, especially regarding the advantages of pre-trained transformer-based models for low-resource or domain-mismatched settings.

3.9 Summary

This chapter described the methodology for comparing Japanese ASR across three model families. The study followed six phases (planning, prior work, data preparation, training, evaluation, and discussion) and used a single, consistent dataset pipeline to ensure fair comparison. Japanese TEDx talks with manual subtitles were collected, transcripts were cleaned, audio was standardised to 16 kHz mono, and long recordings were segmented using WebRTC VAD, with low-quality non-speech clips filtered out. All models were trained on the same 80/10/10 splits and evaluated on the same test set using CER, WER, and RTF to compare accuracy and decoding speed.

CHAPTER FOUR
RESULTS AND DISCUSSION

4.1 Introduction

This chapter reported the outcomes of the end-to-end pipeline described in Chapter 3, starting from data collection and preprocessing, followed by model training and fine-tuning, and finally evaluation on a held-out test split. Results were organized to reflect the methodology phases and to support a fair comparison across three ASR model families (Kaldi GMM–HMM, CRDNN–CTC, and fine-tuned Whisper). Performance was measured using character error rate (CER), word error rate (WER), and real-time factor (RTF).

4.2 Data Collection and Preprocessing Results

4.2.1 Data Source Selection Outcomes

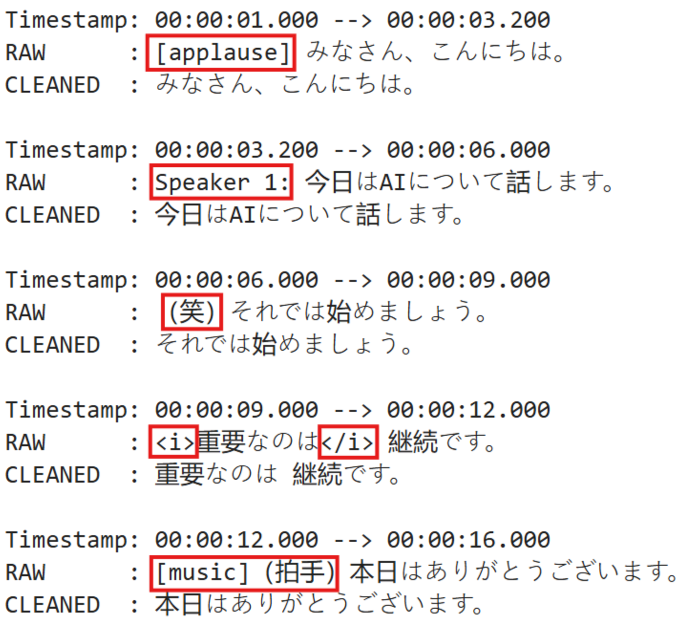
This study collected Japanese TEDx talks from YouTube, restricted to videos that provided manual Japanese subtitles. Videos with only auto-generated captions or no Japanese subtitles were excluded to reduce transcript noise.

Table 4.1
Video selection outcomes for TEDx Japanese data collection.

Item	Count
Playlist items queried	1574
Manual Japanese subtitles found	862
Auto-generated only / none	712
Download completed	862

4.2.2 Transcript Cleaning and Normalization Outcomes

After subtitle download, raw subtitle text was normalized into reference transcripts by removing formatting tags, speaker labels, and non-speech annotations (e.g., `[laughter]`, `(applause)`). The output of this step was a cleaned transcript stream aligned by subtitle cue timestamps.



```
Timestamp: 00:00:01.000 --> 00:00:03.200
RAW       : [applause] みなさん、こんにちは。
CLEANED   : みなさん、こんにちは。

Timestamp: 00:00:03.200 --> 00:00:06.000
RAW       : Speaker 1: 今日はAIについて話します。
CLEANED   : 今日はAIについて話します。

Timestamp: 00:00:06.000 --> 00:00:09.000
RAW       :  (笑) それでは始めましょう。
CLEANED   : それでは始めましょう。

Timestamp: 00:00:09.000 --> 00:00:12.000
RAW       : <i>重要なのは</i> 継続です。
CLEANED   : 重要なのは 継続です。

Timestamp: 00:00:12.000 --> 00:00:16.000
RAW       : [music] (拍手) 本日はありがとうございます。
CLEANED   : 本日はありがとうございます。
```

Figure 4.1 Audio resampling from 44.1 kHz(Top) to 16 kHz(Bottom)

4.2.3 Audio Standardization Outcomes

All audio was converted to a consistent format (16 kHz, mono, PCM16). This ensured compatibility across Kaldi, SpeechBrain, and Whisper training pipelines.

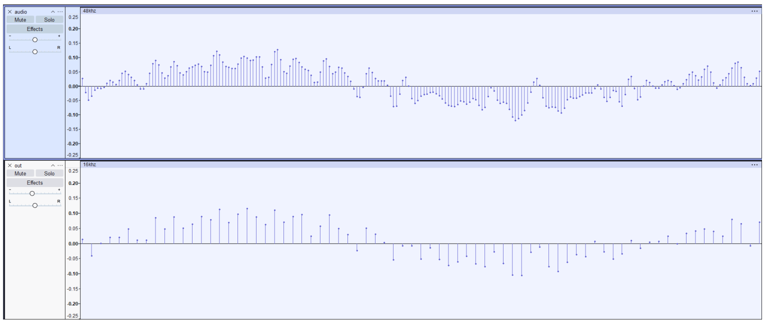


Figure 4.2 Audio resampling from 44.1 kHz(Top) to 16 kHz(Bottom)

4.2.4 VAD Segmentation Outcomes

Long-form TEDx audio was segmented into shorter training utterances using WebRTC VAD. Very short segments ($\leq$2 s) were removed, and very long segments were sliced into $\leq$20 s chunks to stabilize training.

Table 4.2
VAD segmentation results.

Item	Count / Value	Notes
Total chunks produced	74802	After segmentation and slicing
Chunks removed (too short)	11881	≤2 s
Avg chunk duration (s)	6.605	After filtering short clips
Max chunk duration (s)	20	Enforced by slicing

4.2.5 Non-Speaker / Low-Quality Audio Filtering Outcomes

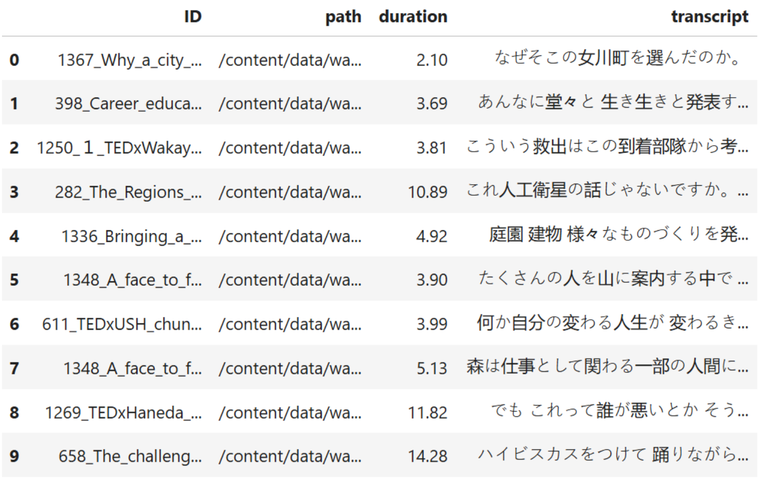
A filtering step was applied to remove chunks dominated by applause/laughter or unclear speech using diarization (dominant speaker ratio) and CLAP-based acoustic event detection. This step aimed to reduce training noise.

Table 4.3
Audio filtering outcomes (diarization + CLAP).

Item	Count	Notes
Chunks before filtering	62921	Output from VAD
Removed by speaker-structure check	13758	Dominant speaker ratio threshold
Removed by clap/laughter threshold	5596	CLAP score thresholds
Chunks after filtering (final)	43567	Used to build manifests

4.2.6 Transcript-to-Audio Mapping and Manifest Outcomes

After segmentation and cleaning, each chunk was paired with a reference transcript using timestamp alignment, and manifests were produced for reproducible training and evaluation.



	ID	path	duration	transcript
0	1367_Why_a_city_...	/content/data/wa...	2.10	なぜそこの女川町を選んだのか。
1	398_Career_educa...	/content/data/wa...	3.69	あんなに堂々と 生き生きと発表す...
2	1250_１_TEDxWakay...	/content/data/wa...	3.81	こういう救出はこの到着部隊から考...
3	282_The_Regions_...	/content/data/wa...	10.89	これ人工衛星の話じゃないですか。...
4	1336_Bringing_a_...	/content/data/wa...	4.92	庭園 建物 様々なものづくりを発...
5	1348_A_face_to_f...	/content/data/wa...	3.90	たくさんの人を山に案内する中で ...
6	611_TEDxUSH_chun...	/content/data/wa...	3.99	何か自分の変わる人生が 変わるき...
7	1348_A_face_to_f...	/content/data/wa...	5.13	森は仕事として関わる一部の人間に...
8	1269_TEDxHaneda_...	/content/data/wa...	11.82	でも これって誰が悪いとか そう...
9	658_The_challeng...	/content/data/wa...	14.28	ハイビスカスをつけて 踊りながら...

Figure 4.3 Audio resampling from 44.1 kHz(Top) to 16 kHz(Bottom)

4.2.7 Train/Validation/Test Split Results

The final dataset was split into 80% training, 10% validation, and 10% testing. ?? reported split sizes.

Table 4.4
Dataset statistics and train/validation/test split summary.

Split	#Utterances	Duration (hrs)	Average (s)
Train	32852	63.3	6.941
Valid	4357	8.3	6.861
Test	4358	8.1	6.731

4.3 Model Training and Fine-Tuning Results

4.3.1 Kaldi GMM–HMM Training Outcomes

Four Kaldi acoustic model stages were trained (mono, tri1, tri2, tri3). Later stages consistently improved accuracy in the final evaluation (reported in ??), indicating that staged training and improved alignments produced stronger acoustic models.

Table 4.5
Kaldi GMM–HMM training dynamics based on objective function improvement per frame. Peak improvements occur early and decrease toward near-zero values, indicating convergence.

Stage	Peak impr./frame	Peak iter	Final impr./frame	Stable iter (< 0.02)
mono	0.4289	1	0.0110	32
tri1	0.2022	3	0.0026	30
tri2	0.4820	1	0.0034	30
tri3	0.2975	3	0.0034	30

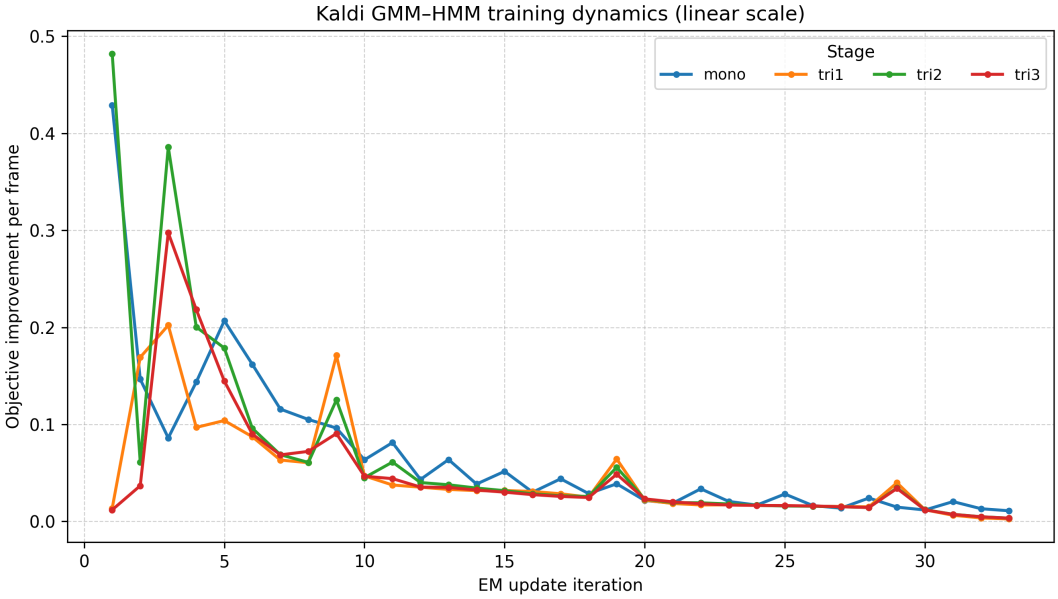


Figure 4.4 Audio resampling from 44.1 kHz(Top) to 16 kHz(Bottom)

4.3.2 CRDNN–CTC Training Outcomes

The CRDNN–CTC model was trained using the prepared manifests and a character vocabulary derived from the training transcripts. Over 70 epochs, training loss decreased from 5.44 to 0.0775, while validation CER reached its lowest value of 0.2272 at epoch 68. The best checkpoint was selected using validation tracking and later evaluated on the test split (reported in ??). ?? summarized the epoch-by-epoch training dynamics from `data.txt`.

Table 4.6
CRDNN–CTC training metrics across epochs.

Epoch	Train loss	Valid loss	CER
1	5.4400	5.2833	0.9983
2	4.0100	3.0769	0.6428
5	1.9100	1.6922	0.4555
10	1.1200	1.3025	0.3372
15	0.7760	1.0662	0.2906
20	0.5770	1.0610	0.2783
25	0.3130	1.0613	0.2629
30	0.2250	1.0297	0.2483
35	0.1740	1.0951	0.2404
40	0.1360	1.1158	0.2363
50	0.2050	1.0767	0.2437
60	0.1090	1.1674	0.2352
70	0.0775	1.2433	0.2309

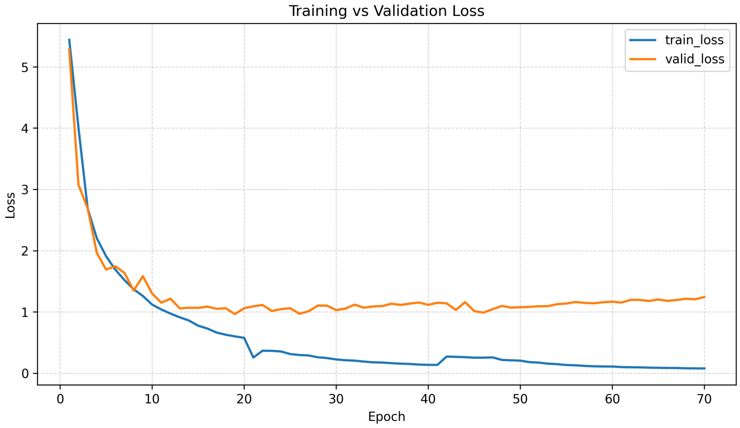


Figure 4.5 Audio resampling from 44.1 kHz(Top) to 16 kHz(Bottom)

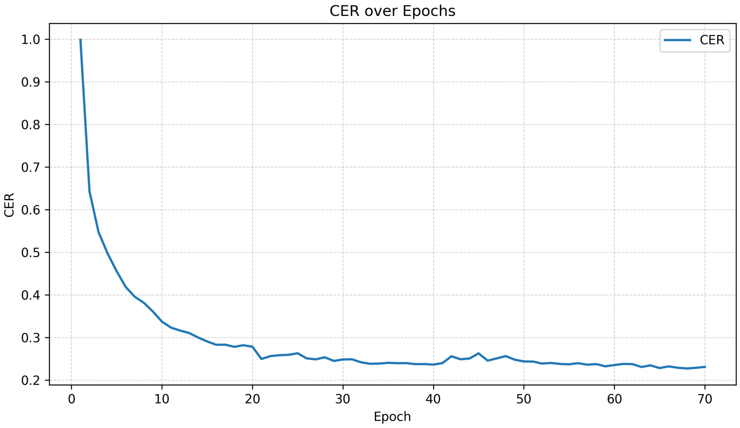


Figure 4.6 Audio resampling from 44.1 kHz(Top) to 16 kHz(Bottom)

4.3.3 Whisper Fine-Tuning Outcomes

Multiple Whisper variants were fine-tuned (tiny, base, small, medium, turbo). Training loss decreased steadily over 10 epochs for all variants, showing stable convergence. ?? reported the training loss trajectories.

Table 4.7
Whisper fine-tuning loss across epochs (lower is better).

Epoch	tiny	base	small	medium	turbo
1	0.6626	0.3683	0.1988	0.0747	0.2137
2	0.3636	0.2106	0.0820	0.0415	0.1361
3	0.2862	0.1466	0.0413	0.0236	0.0911
4	0.2327	0.1034	0.0216	0.0130	0.0591
5	0.1930	0.0731	0.0115	0.0071	0.0362
6	0.1636	0.0520	0.0061	0.0035	0.0198
7	0.1426	0.0380	0.0033	0.0016	0.0095
8	0.1289	0.0296	0.0016	0.0012	0.0041
9	0.1205	0.0252	0.0008	0.0006	0.0016
10	0.1173	0.0234	0.0006	0.0003	0.0003

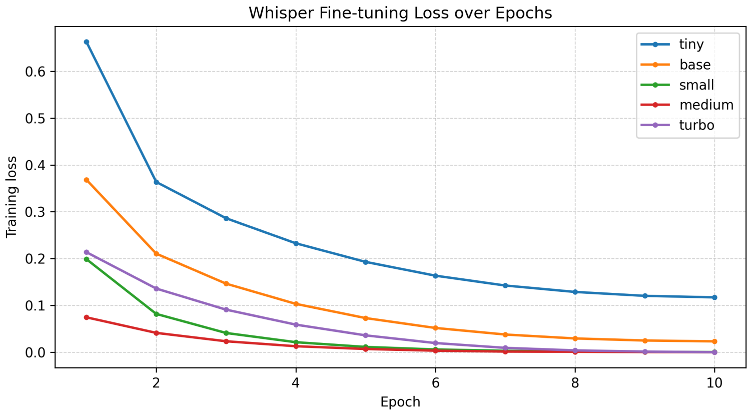


Figure 4.7 Audio resampling from 44.1 kHz(Top) to 16 kHz(Bottom)

4.4 Evaluation Results (CER, WER, and RTF)

4.4.1 Overall Performance Comparison

?? presented the main comparison across all systems. Whisper variants achieved the lowest error rates overall, CRDNN–CTC achieved the fastest decoding speed, and Kaldi showed strong gains from staged training but remained less accurate than the neural approaches.

Table 4.8
Overall comparison of ASR systems on the test split (lower is better for CER/WER/RTF).

Model	CER (%)	WER (%)	RTF
GMMHMM-mono	49.49	53.43	0.6500
GMMHMM-tri1	24.95	29.31	0.3410
GMMHMM-tri2	21.30	25.59	0.2480
GMMHMM-tri3	19.48	23.57	0.2510
CRDNNCTC-1	15.23	22.87	0.0129
CRDNNCTC-2 (beam)	15.12	22.70	0.0147
Whisper-tiny	10.72	11.61	0.0297
Whisper-base	5.67	5.89	0.0413
Whisper-small	4.34	4.30	0.0737
Whisper-medium	4.29	4.22	0.1605
Whisper-turbo	4.28	4.23	0.0959

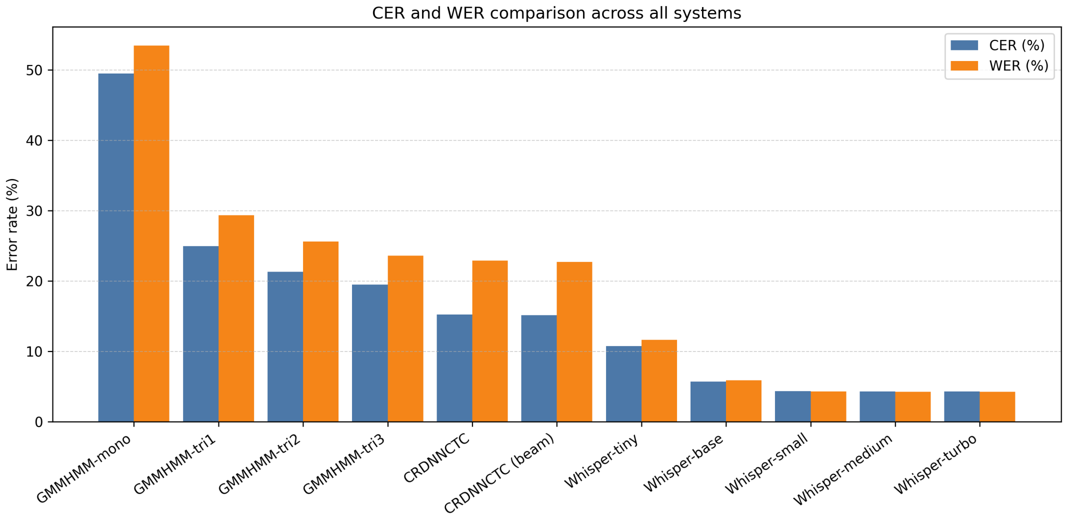


Figure 4.8 Audio resampling from 44.1 kHz(Top) to 16 kHz(Bottom)

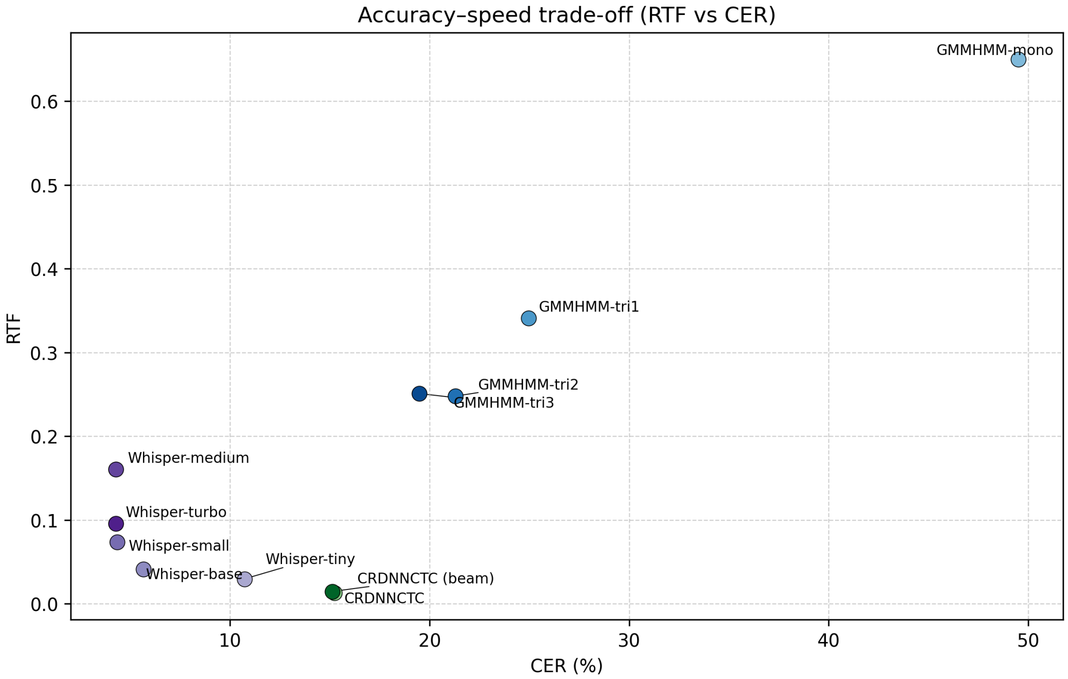


Figure 4.9 Audio resampling from 44.1 kHz(Top) to 16 kHz(Bottom)

4.4.2 Kaldi GMM–HMM Results (mono to tri3)

Kaldi showed consistent improvements across staged training. From mono to tri3, CER decreased from 49.49% to 19.48% (60.64% relative reduction), while WER decreased from 53.43% to 23.57% (55.89% relative reduction). The largest improvement occurred from mono to tri1, while later stages provided smaller but consistent gains.

Table 4.9
Kaldi GMM–HMM results across training stages on the test split.

Stage	CER (%)	WER (%)	RTF
mono	49.49	53.43	0.6500
tri1	24.95	29.31	0.3410
tri2	21.30	25.59	0.2480
tri3	19.48	23.57	0.2510

4.4.3 CRDNN–CTC Results (Greedy vs Beam)

Beam decoding provided a small improvement over greedy decoding (WER reduced from 22.87% to 22.70%; 0.74% relative reduction), but increased decoding cost (RTF increased by approximately 13.95%). Both CRDNN variants remained substantially faster than the other model families.

Table 4.10
CRDNN–CTC results on the test split (greedy vs beam).

Decoder	CER (%)	WER (%)	RTF
Greedy (CRDNNCTC-1)	15.23	22.87	0.0129
Beam (CRDNNCTC-2)	15.12	22.70	0.0147

4.4.4 Whisper Fine-Tuning Results (Model Size Effects)

Whisper fine-tuning produced the strongest transcription accuracy. Accuracy improved substantially from tiny to base (WER reduced from 11.61% to 5.89%), and continued improving from base to small (WER reduced to 4.30%). Gains from small to medium were marginal (WER 4.30% to 4.22%; about 1.86% relative), while decoding became much slower (RTF increased from 0.0737 to 0.1605). Whisper-turbo achieved nearly the same accuracy as medium (WER 4.23%) with faster decoding (RTF 0.0959).

Table 4.11
Whisper fine-tuning results by model size on the test split.

Variant	CER (%)	WER (%)	RTF
tiny	10.72	11.61	0.0297
base	5.67	5.89	0.0413
small	4.34	4.30	0.0737
medium	4.29	4.22	0.1605
turbo	4.28	4.23	0.0959

4.5 Discussion

4.5.1 Main Findings and Trade-offs

The results showed clear differences between model families. Whisper variants achieved the lowest CER/WER, indicating the strongest accuracy under the TEDx Japanese domain. CRDNN–CTC achieved the fastest decoding (lowest RTF), indicating strong suitability for low-latency use cases. Kaldi showed large improvements across staged training but remained less accurate than the neural approaches.

4.5.2 Interpretation by Model Family

4.5.2.1 Kaldi: staged training gains and diminishing returns

The large improvement from mono to tri1 reflected the benefit of moving from context-independent to context-dependent modelling. Later stages (tri2 and tri3) further improved accuracy through feature transforms (LDA+MLLT) and speaker-adaptive training (SAT), but with diminishing gains as the system approached its best performance under the chosen feature and language model configuration.

4.5.2.2 CRDNN–CTC: decoding strategy impact

The small gap between greedy and beam decoding suggested that most performance was driven by the acoustic model, and the chosen beam settings provided limited additional benefit relative to the added decoding cost. However, both CRDNN variants were consistently faster than Whisper and Kaldi in RTF.

4.5.2.3 Whisper: strong accuracy with model-size cost

Whisper performance improved with model size, but the speed cost increased substantially for larger variants. Whisper-medium achieved the best WER, while Whisper-turbo provided nearly the same accuracy with improved speed, making it a more practical choice when latency was important.

4.5.3 Error Analysis

To complement aggregate metrics, this section can summarize recurring error patterns across systems (e.g., deletions under fast speech, named entities, subtitle mismatch, or segmentation boundary effects). Detailed examples can be included using ??.

Table 4.12
Qualitative transcription examples (named entity + minor deletions).

System	Text
Reference	AI
Kaldi (tri3)	AI
CRDNN–CTC (best)	AI
Whisper (best)	AI

Table 4.13
Qualitative transcription examples (fast-speech particle deletions).

System	Text
Reference	
Kaldi (tri3)	
CRDNN–CTC (best)	
Whisper (best)	

Table 4.14
Qualitative transcription examples (segmentation boundary effects).

System	Text
Reference	
Kaldi (tri3)	
CRDNN–CTC (best)	
Whisper (best)	

Table 4.15
Qualitative transcription examples (subtitle mismatch / paraphrase).

System	Text
Reference	
Kaldi (tri3)	
CRDNN–CTC (best)	
Whisper (best)	

Table 4.16
Qualitative transcription examples (numbers + loanwords).

System	Text
Reference	2024
Kaldi (tri3)	
CRDNN–CTC (best)	2024
Whisper (best)	2024

4.5.4 Impact of Preprocessing Choices on Results

Because all systems used the same cleaned and standardized dataset, differences in accuracy and speed mainly reflected model architecture and decoding strategy rather than evaluation handling. Nevertheless, the preprocessing steps in Chapter 3 (manual subtitle selection, cleaning, VAD chunking, and filtering) were designed to reduce transcript noise and non-speech audio, which supported stable training and fair comparison across model families.

4.5.5 Limitations

Key limitations include TEDx-only speaking style, possible subtitle-to-speech mismatch, sensitivity of Japanese WER to tokenization, and limited ablation studies (e.g., VAD thresholds, filtering thresholds, or language model variants).

4.6 Summary

This chapter presented results for each stage of the Chapter 3 pipeline and compared ASR performance across classical and neural approaches. Kaldi improved substantially across staged training, CRDNN–CTC provided the fastest decoding, and fine-tuned Whisper variants achieved the best transcription accuracy with clear speed trade-offs by model size. The next chapter concludes the study and proposes future improvements.

REFERENCES

Ando, S., & Fujihara, H. (2021). Construction of a large-scale japanese asr corpus on tv recordings. ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 6948–6952.

Baevski, A., Zhou, Y., Mohamed, A., & Auli, M. (2020). Wav2vec 2.0: A framework for self-supervised learning of speech representations. Advances in neural information processing systems, 33, 12449–12460.

Bajo, M., Fukukawa, H., Morita, R., & Ogasawara, Y. (2024). Efficient adaptation of multilingual models for japanese asr. arXiv preprint arXiv:2412.10705.

Boishakhi, F. T., Shill, P. C., & Alam, M. G. R. (2021). Multi-modal hate speech detection using machine learning. 2021 IEEE International Conference on Big Data (Big Data), 4496–4499.

Curtin, K. (2020). Japanese kanji power: A workbook for mastering japanese characters.

Dehak, N., Kenny, P., Dehak, R., Glembek, O., Dumouchel, P., Burget, L., Hubeika, V., & Castaldo, F. (2009). Support vector machines and joint factor analysis for speaker verification. 2009 IEEE International Conference on Acoustics, Speech and Signal Processing, 4237–4240.

Fujita, Y., Watanabe, S., Chang, X., & Maekaku, T. (2024). Lv-ctc: Non-autoregressive asr with ctc and latent variable models. https://arxiv.org/abs/2403.19207

Futami, H., Ueno, S., Mimura, M., Sakai, S., Kawahara, T., et al. (2020). Rescoring hypotheses of automatic speech recognition with bidirectional transformer language model. Proceedings of the 82nd National Convention of IPSJ, 2020(1), 175–176.

Gales, M., & Young, S. (2008). The application of hidden markov models in speech recognition. Foundations and Trends® in Signal Processing, 1(3), 195–304.

Hojo, N., Ijima, Y., & Mizuno, H. (2018). Dnn-based speech synthesis using speaker codes. IEICE TRANSACTIONS on Information and Systems, 101(2), 462–472.

Imaishi, R., & Kawabata, T. (2022). Examination of iterative estimation counts in gaussian mixture model-based speaker recognition. Journal of the Acoustical Society of Japan, 78(11), 650–653.

Imaizumi, R., Masumura, R., Shiota, S., & Kiya, H. (2022). End-to-end japanese multi-dialect speech recognition and dialect identification with multi-task learning. APSIPA Transactions on Signal and Information Processing, 11. https://doi.org/10.1561/116.00000045

Ito, H., Hagiwara, A., Ichiki, M., Mishima, T., Sato, S., & Kobayashi, A. (2016). End-to-end neural network modeling for japanese speech recognition. Journal of the Acoustical Society of America, 140, 3116–3116. https://doi.org/10.1121/1.4969755

Ito, H., Hagiwara, A., Ichiki, M., Mishima, T., Sato, S., & Kobayashi, A. (2017). End-to-end speech recognition for languages with ideographic characters. 2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), 1228–1232. https://doi.org/10.1109/APSIPA.2017.8282226

Juang, B. H., & Rabiner, L. R. (1991). Hidden markov models for speech recognition. Technometrics, 33(3), 251–272.

Karita, S., Kubo, Y., Bacchiani, M. A. U., & Jones, L. (2021). A comparative study on neural architectures and training methods for japanese speech recognition. Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH, 2, 2092–2096. https://doi.org/10.21437/INTERSPEECH.2021-775

Koenecke, A., Nam, A., Lake, E., Nudell, J., Quartey, M., Mengesha, Z., Toups, C., Rickford, J. R., Jurafsky, D., & Goel, S. (2020). Racial disparities in automated speech recognition. Proceedings of the National Academy of Sciences of the United States of America, 117, 7684–7689. https://doi.org/10.1073/PNAS.1915768117/SUPPL_FILE/PNAS.1915768117.SAPP.PDF

Kohei Mukohara, S. N., Koichiro Yoshino, et al. (2015). Investigation of dnn and cnn bottleneck features in emotional speech recognition. Research Report on Speech and Language Processing (SLP), 2015(15), 1–6.

Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., & Houlsby, N. (2020). Big transfer (bit): General visual representation learning. Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16, 491–507.

Kubo, Y. (2014). Deep learning for speech recognition (series explanation: Deep learning [part 5]). Artificial Intelligence, 29(1), 62–71.

Latif, S., Qadir, J., Qayyum, A., Usama, M., & Younis, S. (2020). Speech technology for healthcare: Opportunities, challenges, and state of the art. IEEE Reviews in Biomedical Engineering, 14, 342–356.

Lin, S., Tsunakawa, T., Nishida, M., & Nishimura, M. (2017). Dnn-based feature transformation for speech recognition using throat microphone. 2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), 596–599.

Masato Mimura, T. K., et al. (2013). Application of dnn-hmm to japanese lecture speech recognition using csj and investigation of speaker adaptation. Research Report on Speech and Language Processing (SLP), 2013(9), 1–6.

Matrouf, D., Verdet, F., Rouvier, M., Bonastre, J.-F., & Linarès, G. (2011). Modeling nuisance variabilities with factor analysis for gmm-based audio pattern classification. Computer Speech & Language, 25(3), 481–498.

Miwa, S., & Kai, A. (2023). Dialect speech recognition modeling using corpus of japanese dialects and self-supervised learning-based model xlsr. Proc. INTERSPEECH 2023, 4928–4932.

Mu, D., Sun, W., Xu, G., & Li, W. (2020). Japanese pronunciation evaluation based on ddnn. IEEE Access, 8, 218644–218657.

Noda, K., Yamaguchi, Y., Nakadai, K., Okuno, H. G., Ogata, T., et al. (2014). Lipreading using convolutional neural network. Interspeech, 1, 3.

Povey, D., Burget, L., Agarwal, M., Akyazi, P., Kai, F., Ghoshal, A., Glembek, O., Goel, N., Karafiát, M., Rastrow, A., et al. (2011). The subspace gaussian mixture model—a structured model for speech recognition. Computer Speech & Language, 25(2), 404–439.

Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., & Sutskever, I. (2023). Robust speech recognition via large-scale weak supervision. International conference on machine learning, 28492–28518.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. OpenAI blog, 1(8), 9.

Ravanelli, M., Parcollet, T., & Bengio, Y. (2019). The pytorch-kaldi speech recognition toolkit. https://arxiv.org/abs/1811.07453

Ravanelli, M., Parcollet, T., Plantinga, P., Rouhe, A., Cornell, S., Lugosch, L., Subakan, C., Dawalatabad, N., Heba, A., Zhong, J., Chou, J.-C., Yeh, S.-L., Fu, S.-W., Liao, C.-F., Rastorgueva, E., Grondin, F., Aris, W., Na, H., Gao, Y., … Bengio, Y. (2021). Speechbrain: A general-purpose speech toolkit. https://arxiv.org/abs/2106.04624

Rose, H. (2019). Unique challenges of learning to write in the japanese writing system. L2 writing beyond English, 66.

Seki, H., Yamamoto, K., & Nakagawa, S. (2014). Comparison of syllable-based and phoneme-based dnn-hmm in japanese speech recognition. 2014 International Conference of Advanced Informatics: Concept, Theory and Application (ICAICTA), 249–254.

Sonali Nemade, R. D. P., Yogesh Kumar Sharma. (2019). To improve voice recognition system using gmm and hmm classification models. International Journal of Innovative Technology and Exploring Engineering (2019) 8(11) 2724-2726.

Sun, R. H., & Chol, R. J. (2020). Subspace gaussian mixture based language modeling for large vocabulary continuous speech recognition. Speech Communication, 117, 21–27.

Taheri, A., & Taheri, M. (2006). Fuzzy hmm and gmm models for speech recognition. 2006 2nd International Conference on Information & Communication Technologies, 1, 1242–1245.

Takahashi, N., Miwa, S., Kamiya, Y., Toyama, T., Nahar, R., & Kai, A. (2024). Comparison of large pre-trained models and adaptation methods for japanese dialects asr. 2024 IEEE 13th Global Conference on Consumer Electronics (GCCE), 811–814.

Takami, J., & Kawabata, T. (2020). Speaker recognition performance metric for small-scale voice dialogue systems based on adaptation speed and convergence accuracy of gaussian mixture models. Journal of the Acoustical Society of Japan, 76(5), 254–261.

Takeuchi, D., Yatabe, K., Koizumi, Y., Oikawa, Y., & Harada, N. (2020). Real-time speech enhancement using equilibriated rnn. ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 851–855.

Taniguchi, S., Kato, T., Tamura, A., & Yasuda, K. (2022). Transformer-based automatic speech recognition with auxiliary input of source language text toward transcribing simultaneous interpretation. INTERSPEECH, 2813–2817.

Taniguchi, S., Kato, T., Tamura, A., Yasuda, K., et al. (2024). Pre-training of transformer-based asr for simultaneous interpretation with auxiliary input of source language text using large machine translation corpus. Proceedings of the 86th National Convention of IPSJ, 2024(1), 397–398.

Tokuda, K. (1999). Application of hidden markov models to speech synthesis. IEICE Technical Report, SP99-61, 48–54.

Tokuda, K. (2000). Fundamentals of speech synthesis using hmm. IEICE Technical Report, SP2000-74, 43.

Watanabe, S., Hori, T., Karita, S., Hayashi, T., Nishitoba, J., Unno, Y., Soplin, N. E. Y., Heymann, J., Wiesner, M., Chen, N., Renduchintala, A., & Ochiai, T. (2018). Espnet: End-to-end speech processing toolkit. https://arxiv.org/abs/1804.00015

Wei Xu, L. G., Marvin J. Dainoff, & Gao, Z. (2023). Transitioning to human interaction with ai systems: New challenges and opportunities for hci professionals to enable human-centered ai. International Journal of Human–Computer Interaction, 39(3), 494–518. https://doi.org/10.1080/10447318.2022.2041900

Xu, C., Ye, R., Dong, Q., Zhao, C., Ko, T., Wang, M., Xiao, T., & Zhu, J. (2023). Recent advances in direct speech-to-text translation. arXiv preprint arXiv:2306.11646.

Yalta, N., Watanabe, S., Hori, T., Nakadai, K., & Ogata, T. (2019). Cnn-based multichannel end-to-end speech recognition for everyday home environments. 2019 27th European Signal Processing Conference (EUSIPCO), 1–5.

Yusuke Kida, T. T., et al. (2016). Reverberant speech recognition based on linear predictive filter estimation using lstm. Research Report on Speech and Language Processing (SLP), 2016(25), 1–6.

Zhang, Y., Han, W., Qin, J., Wang, Y., Bapna, A., Chen, Z., Chen, N., Li, B., Axelrod, V., Wang, G., et al. (2023). Google usm: Scaling automatic speech recognition beyond 100 languages. arXiv preprint arXiv:2303.01037.