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

LIST OF TABLES

LIST OF FIGURES

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.

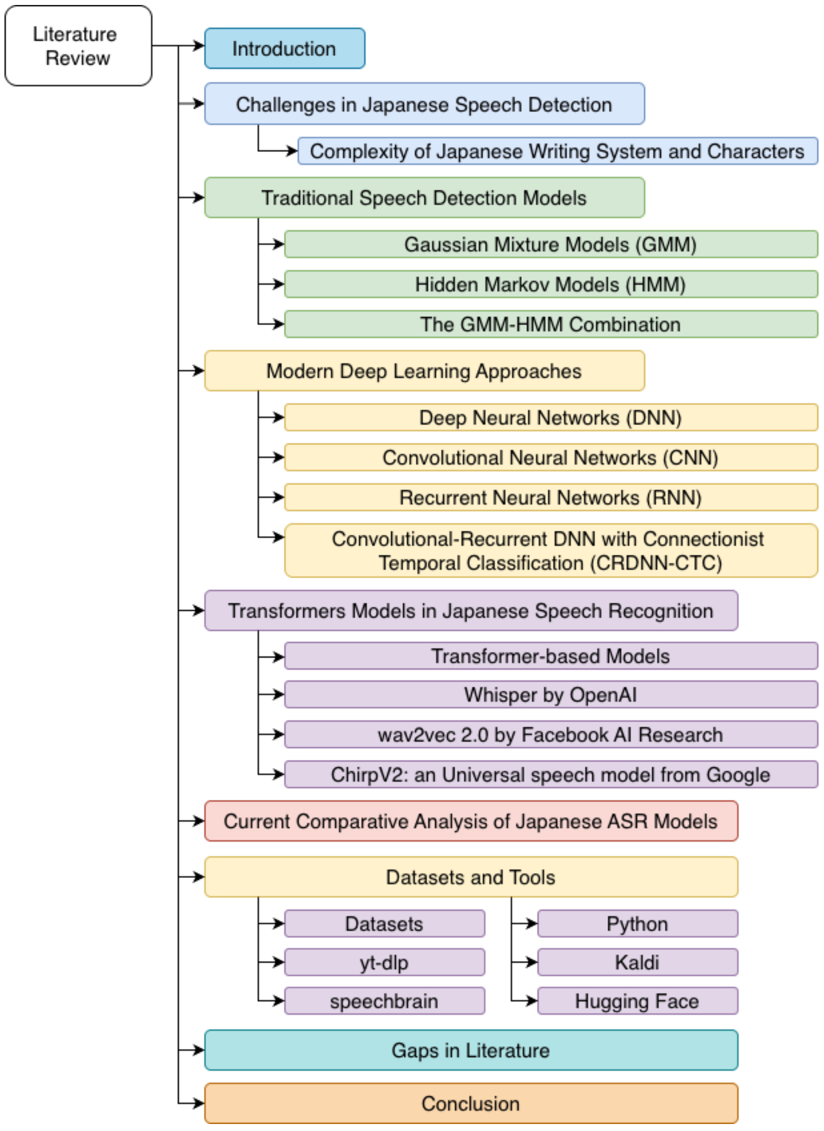
1.7 Conclusion

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.

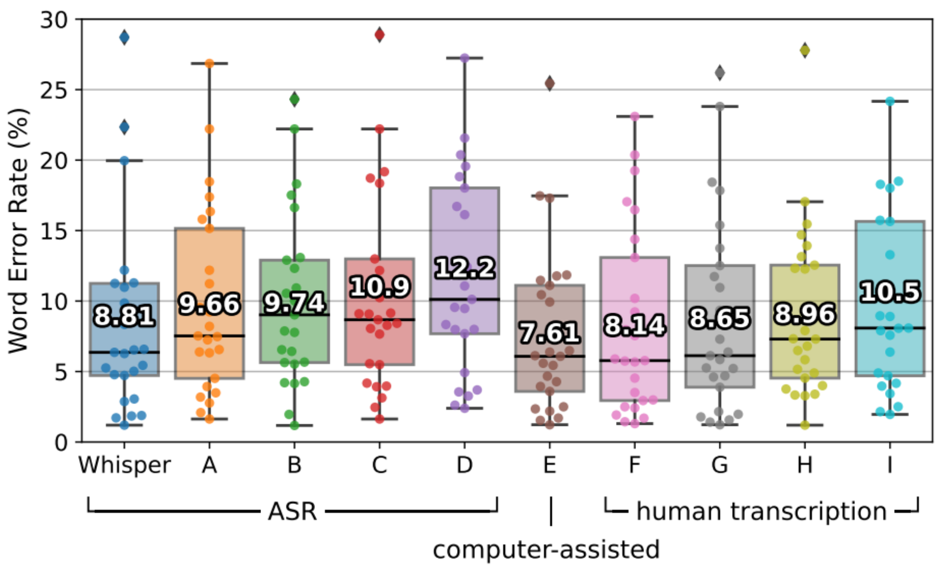


Figure 2.2 Whisper WER cited from Radford et al., 2023

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 is 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 moving 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. Table 3.1 below shows an overview of the phases, activities and deliverables.

Table 3.1
Overview of Research Project

Phase	**Activities**	**Deliverables**
Planning	Define below items: • Project title • Problem statements • Objectives • Scopes • Significance	Chapter 1 • Title defined • Problems defined • Objectives defined • Scope defined • Significance defined
Prior Work	• Review Japanese ASR studies • List preprocessing methods • List model families • List evaluation metrics • Summarize key gaps	Chapter 2 • Literature review written • Preprocessing chosen • Models selected • Metrics defined • Gaps summarized
Data Collection and preprocessing	• Collect Japanese TEDx talks • Download audio and subtitles • Clean and normalize transcripts • Resample audio to 16 kHz mono • Split audio using VAD • Create train/valid/test splits • Export manifests and metadata	Chapter 3 • Dataset compiled • Collection method documented • Transcripts normalized • Segments generated • Splits finalized • Manifests prepared
Model Training and Fine Tune	• Train GMM-HMM • Train CRDNN-CTC • Fine-tune Whisper variants • Set decoding parameters • Save configs and checkpoints	Chapter 4 • Models trained • Checkpoints saved • Settings recorded • Outputs generated
Analysis	• Score with CER and WER • Measure speed using RTF • Compare models and decoders • Inspect common errors	Chapter 4 • Results tables and plots • Error analysis summary • Best model identified
Discussion	• Explain main findings • Link to prior work • Note limitations • Suggest future work	Chapter 5 • Findings discussed • Limitations stated • Recommendations given • Future work proposed

3.2.1 Planning Phase

Table 3.2 shows 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 are described in the table shown below.

Table 3.2
Overview of Research Project

Phase	Activities	Deliverables
Planning	Define 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 begin in the planning phase. This phase is important because it will serve as the blueprint and become guidance to the project direction. This phase purpose is to identify the problem in the domain and finds a way to solve the problem stated. The activities that will be carried out in this pahse is defining key aspect that is the title of the project, problem statements, objective, scope and significance of the project. The outcome from this phase will be the title of the project, problems that has been defined, objective of the project, scope of the project which explained what is covered and what is not covered in this project and last deliverables in this phase is 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.

3.2.2 Prior Work

Table 3.3 shows the second phase of this project where prior work in the domain of ASR is reviewed and some decision is made. The activities carried out and the outcome deliverables are described in the table shown below.

Table 3.3
Overview of Research Project

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

The starting point of this project begin in the planning phase. This phase is important because it will serve as the blueprint and become guidance to the project direction. This phase purpose is to identify the problem in the domain and finds a way to solve the problem stated. The activities that will be carried out in this pahse is defining key aspect that is the title of the project, problem statements, objective, scope and significance of the project. The outcome from this phase will be the title of the project, problems that has been defined, objective of the project, scope of the project which explained what is covered and what is not covered in this project and last deliverables in this phase is 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.

3.3 Data Collection Pipeline

In this section, each step of the data collection pipeline was described in detail, highlighting the methods and tools used to ensure the quality and consistency of the collected data. The process began with identifying suitable sources of formal spoken Japanese, was followed by programmatically scraping of audio and subtitle data from YouTube and TED Talk platforms. The raw subtitles were then cleaned and normalized to produce high-quality transcripts. The long-form audio recordings were segmented into utterance-level clips aligned with the cleaned transcripts. All audio clips were standardized to a consistent 16 kHz, mono and WAV format. Finally, the dataset was split into speaker-independent for training, validation, and test sets.

3.3.1 Data Source Selection

The first step in data collection was to identify the suitable data sources that would be used in this study. The focus was on formal speech rather than spontaneous conversation or noisy environments. Suitable sources included TED and TEDx talks, invited lectures, conference presentations, public addresses, and university lectures. These audio usually had clear speech, minimal background noise, and well-structured content. However to gather data from private lectures or talks that were not publicly available, permission from the content owners might have been required. Because of that, this work focused on publicly accessible content on platforms like YouTube and the official TEDx talks website.

Video from TEDx talks had 2 types that was manually curated subtitles and automatically generated subtitles. Manually curated subtitles became the primary choice because they tended to be more accurate and better aligned with the spoken content. Another strong reason to choose TEDx talks was that they often covered a wide range of topics, which helped ensure lexical diversity in the collected dataset. The speakers were usually professionals or experts in their fields, which meant the speech was generally clear and well-articulated.

3.3.2 Data Scraping in Python

After selecting the data sources, the next step was to programmatically scrape the audio and subtitle data. A python library yt-dlp was used to download audio and subtitles from YouTube videos. This tools had the option to specify download option so that only video that had manual japanese transcription only that would be downloaded. Additionally, yt-dlp supported downloading subtitles in various formats, including SRT and VTT, which were commonly used for captioning.

The video was downloaded using a custom Python script that leveraged the yt-dlp library. The script took a list of video URLs as input and iterated through each URL to perform the following actions. First, it checked whether the video had Japanese subtitles available, either human-curated or automatically generated. If subtitles were present, the script proceeded to download the audio track of the video. Then, the audio was then downloaded in high quality for example, as `.m4a` and kept in its original form temporarily before the audio files was converted into wav using ffmpeg.

3.3.3 Transcript Cleaning and Normalization

The downloaded subtitle files contained time-aligned text segments that corresponded to the spoken content in the audio. However, they also included non-speech elements such as speaker labels, overlapping dialogue, or background noise descriptions. There was also the issue of inconsistent formatting, such as different use of punctuation, capitalization, and spacing. Because of this, the raw subtitle text could not be used directly as the reference transcript for ASR training and evaluation. Instead, the subtitles had to be normalized by removing non-speech annotations so that only the spoken content remained.

A custom Python script was developed to process each subtitle file. The script read the subtitle data and applied a series of normalization rules. These rules included removing any tags or annotations that did not correspond to spoken words, such as `[laughter]`, `(applause)`, or speaker identifiers like `Speaker 1:`. The output of this stage was a cleaned transcript for each time span covered by the subtitle timestamps. This cleaned transcript became the reference text for evaluation. Applying the same normalization rules consistently across the entire dataset was important, because the comparisons between models were based on this single source of truth.

3.3.4 Audio Segmentation

Most of the downloaded audio consisted of long-form recordings, often ranging from 5 minutes to over an hour in length. However, ASR models were typically trained and evaluated on much shorter utterances, often in the range of a few seconds. This was because long audio would cause memory and computational challenges during both training and decoding. To address this issue, the long recordings had to be segmented into shorter clips that aligned with the ASR training requirements. The time stamps provided in the subtitle data created natural cut points that could be used to break a long recording into manageable clips.

A custom Python script was used to perform the segmentation. The script read the cleaned subtitle timestamps and used them to extract corresponding audio segments from the long-form recordings. Each segment was saved as a separate audio file, typically in WAV format, along with its associated cleaned transcript. If the subtitle was less than 1 second, it might have been discarded to avoid training on extremely short utterances that provided little context. The final output of this stage was a collection of utterance-level audio clips, each paired with its cleaned transcript.

3.3.5 Resampling and Format Standardization

After segmentation, all clips were converted to a consistent audio format so that different model families could be trained and evaluated under the same acoustic conditions. In this paper, every utterance-level clip was resampled to 16 kHz, converted to mono, and stored as 16-bit PCM WAV. Standardizing at this stage had two main benefits. First, traditional hybrid systems such as GMM–HMM in Kaldi commonly assumed 16 kHz WAV input. Second, by storing all clips in an identical format, end-to-end neural systems such as CRDNN-CTC and Whisper could operate directly on the same files without requiring model-specific resampling or channel conversion later.

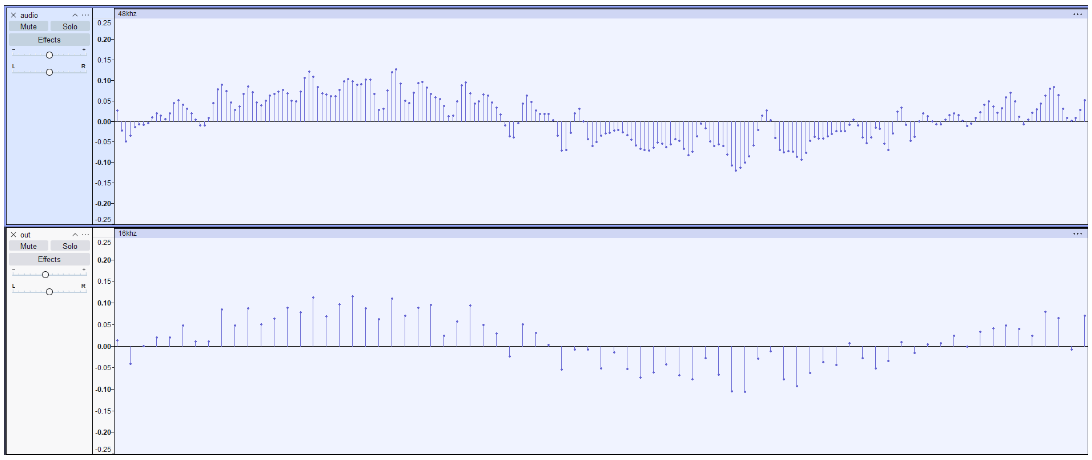


Figure 3.1 Audio resampling from 44.1 kHz(Top) to 16 kHz(Bottom)

Alongside the WAV files, machine-readable metadata was generated. For each utterance ID, the metadata recorded the path to the audio file, the audio duration, the cleaned transcript text from transcript cleaning, and basic information about the speaker or the talk. For the GMM–HMM experiments, these fields were also exported into the conventional Kaldi-style files `wav.scp`, `text`, `utt2spk`, and `spk2utt`. For the CRDNN-CTC and Whisper pipelines, the same information was exported into

JSON or CSV manifests, which were the common input format for modern end-to-end ASR training recipes.

3.3.6 Train / Validation / Test Split

The final step in the data collection pipeline was to split the data into training, validation, and test folders. This data had to be split because the ASR models needed to be trained on one portion of the data (training set), tuned on another portion (validation set), and evaluated on a completely held-out portion (test set). Another reason to split the data was to prevent the system from overfitting to a particular voice and then benefiting from that familiarity at test time, which would have given an unrealistically optimistic error rate.

Since the data was collected from multiple speakers, the data was split in a folder level so that no speaker appeared in more than one split. And then the data was divided into three disjoint sets: 80% for training, 10% for validation, and 10% for testing. Since every model in this thesis was always trained and evaluated on these same splits, their performance could be compared directly.

3.4 Text Preparation for ASR Training

After collecting and segmenting the audio, the corresponding text was prepared for consistent use across all models. The process had three steps. First, the prediction units for each model family—phonetic units for GMM–HMM, characters or subwords for CRDNN–CTC, and tokenizer units for Whisper had to be defined. Secondly, the auxiliary text resources required by each model, such as the pronunciation lexicon and language model text for GMM–HMM, had to be built. Finally, a final normalized reference transcript for each utterance had to be created to serve as the ground truth for evaluation metrics. This section described each of these steps in detail.

3.4.1 Lexicon and Token Units

Different ASR model families represented “text” in different ways. The GMM–HMM system followed a traditional hybrid ASR design in which there was an explicit pronunciation lexicon and an explicit phone or phoneme-like inventory. In this setting, each written unit that appeared in the transcript was mapped to one or more pronunciations, and each pronunciation was represented as a sequence of phones. The acoustic model was trained to predict these phones, and decoding proceeded by searching over phone sequences consistent with both the lexicon and the language model.

In contrast, the CRDNN–CTC model did not depend on an externally defined pronunciation dictionary. Instead, it was trained to map acoustic features directly to character-like units using the CTC loss. In this work, these prediction targets were typically Japanese characters or subword units derived from the cleaned transcripts described in data collection. Because CTC training did not require frame-level alignments, the model could learn the alignment between audio and text implicitly.

The fine-tuned Whisper model followed yet another strategy. Whisper was a transformer encoder–decoder model that used a multilingual tokenizer to represent text as a sequence of subword tokens. These tokens were not simple characters; instead, they were learned units from Whisper's large-scale pretraining, which covered Japanese along with many other languages. During fine-tuning, the model was optimized to continue predicting these tokenizer units, conditioned on the input audio and any decoding settings.

3.4.2 Buildiing Language Model Text for the GMM–HMM

The GMM–HMM pipeline used a separate language model (LM) during decoding to constrain which word sequences were likely. Preparing this component required assembling a clean text corpus that reflected the type of Japanese we expected the system to transcribe. The starting point for this text corpus was the set of cleaned transcripts obtained from the scraping and normalization steps. Because these transcripts came from formal, presentation-style speech, they were already well matched to the target domain of this thesis.

Once the text corpus was prepared, it was tokenized according to the unit definition adopted for the GMM–HMM decoding stage. In many Japanese ASR recipes, words" might have been approximated using segmentation heuristics or morphological analyzers, since Japanese did not naturally separate words with spaces. Whatever segmentation strategy was chosen, it had to remain stable from training through evaluation, because WER depended directly on how word boundaries" were defined. The processed text was then used to train an *n*-gram language model. This language model, together with the lexicon and the phone set described above, was compiled into the decoding graph (often represented in Kaldi as `HCLG.fst`) that would be used at inference time.

3.5 Feature Extraction and Front-End Processing

Before any acoustic model could be trained, the raw waveform for each utterance had to be converted into a numeric representation that was suitable for learning. This section described the feature extraction pipeline used in this thesis. Although the three model families — GMM–HMM, CRDNN–CTC, and fine-tuned Whisper — ultimately relied on different assumptions about how speech should be represented, they all began from the standardized 16 kHz mono WAV clips described in data collection. The processing described here covered two main aspects: (i) how acoustic features were computed, normalized, and prepared for consumption by each model family; and (ii) how data augmentation was applied in a controlled way to improve robustness and allow fair comparison across systems.

3.5.1 MFCC with Cepstral Mean and Variance Normalization

The GMM–HMM system in this work used Mel-Frequency Cepstral Coefficients (MFCCs) as its primary acoustic representation. MFCCs were a conventional hand-crafted feature set for ASR. The waveform was first divided into short, overlapping frames using a fixed frame length (for example, 25 ms) and a fixed frame shift (for example, 10 ms). For each frame, a short-time Fourier transform was computed. The resulting magnitude spectrum was then passed through a Mel-scaled filterbank that emphasized perceptually relevant frequency regions. The log energies in these Mel bands were decorrelated using a discrete cosine transform, yielding a compact cepstral vector per frame. In many ASR recipes, first- and second-order temporal derivatives (deltas and delta-deltas) were appended to capture local dynamics. The final MFCC feature at a given frame therefore encoded both the short-term spectral shape and how that shape was changing over time.

After MFCC extraction, Cepstral Mean and Variance Normalization (CMVN) was applied. The goal of CMVN was to reduce channel mismatch and long-term amplitude drift by forcing each feature dimension to have approximately zero mean and unit variance. In traditional GMM–HMM pipelines, CMVN could be computed per speaker, using all utterances from that speaker, or at minimum per utterance when speaker identity was not reliably known. In this thesis, because data was collected from multiple unrelated speakers and recording conditions, normalization was applied consistently across all clips so that the acoustic model saw features that were less sensitive to absolute loudness, microphone characteristics, or background coloration.

3.5.2 Log-Mel Filterbank and Spectrogram-Derived Features

While MFCCs were well matched to GMM–HMM systems, contemporary neural ASR models tended to operate on less aggressively compressed representations such as log-Mel filterbanks or related spectrogram features. In this thesis, the CRDNN–CTC model was trained on features derived from the short-time magnitude spectrum computed over 16 kHz audio. The waveform was framed and windowed in a similar way, for example, 25 ms windows with 10 ms stride, and a Fourier transform was applied to obtain the frequency content over time. Instead of applying the discrete cosine transform to decorrelate these spectra into cepstral coefficients, the model directly consumed the log-scaled Mel filterbank energies or a closely related log-Mel time frequency matrix.

These log-Mel features were typically mean/variance normalized before being fed into the network. Normalization could be applied globally, as example, using statistics computed over the entire training set or on a per-utterance basis. In practice, either approach reduced variation due to recording conditions and helped stabilize training. The CRDNN–CTC model then learned to map these normalized log-Mel features to character or subword predictions under the CTC loss. Because the model had recurrent or bidirectional recurrent layers on top of the convolutional stack, it was able to integrate long-range temporal information that went well beyond a single frame.

The Whisper model followed a similar but more specialized approach. Whisper's original pretraining used a log-Mel spectrogram computed with a fixed configuration, as example, a particular FFT size, hop length, and Mel filterbank definition and expected audio resampled to a specific rate. During fine-tuning in this thesis, Whisper continued to consume internally normalize log-Mel spectrogram features consistent with its pretraining regime. This was a key difference from MFCC-based systems: in Whisper and other transformer encoder–decoder models, the learned encoder expected the raw spectral structure, not a hand-designed compact representation.

3.5.3 Data Augmentation

In addition to extracting features, this work also considered controlled forms of data augmentation. Augmentation was useful because the dataset described in data collection was drawn from a limited number of speakers and recording setups. Without augmentation, a model might have overfit to those specific voices, microphones, or rooms and then degraded noticeably on new speakers in the held-out test set.

One common augmentation strategy was speed perturbation, where the audio waveform was played slightly faster or slower (for example, by factors such as $0.9\times$, $1.0\times$, and $1.1\times$) without altering the pitch too aggressively. This produced additional training examples that mimicked natural variations in speaking rate. Speed perturbation effectively changed both the temporal dynamics and the apparent formant trajectories, which encouraged the acoustic model to become less sensitive to how quickly a speaker delivered each phrase.

In traditional hybrid systems such as GMM–HMM, speed perturbation was often applied before MFCC extraction so that each perturbed version of the audio yielded its own set of MFCC features. In end-to-end systems such as CRDNN–CTC, it was usually applied on the fly during training, so the model was repeatedly exposed to slightly different versions of the same utterance.

Another augmentation strategy, more common in neural end-to-end models, was spectrogram masking (often referred to as SpecAugment). After computing log-Mel features, small contiguous time regions or frequency bands were randomly masked out during training. The CRDNN–CTC model could be trained with this kind of masking so that it learned to rely on context rather than any single narrowband cue.

3.6 Model Family A: GMM–HMM Acoustic Model

This section was detailed the process taken to build the GMM–HMM baseline system used in this thesis. HMM-GMM system that carefully configured still remained a strong point of reference for controlled studies on lexicon design, tokenization, and LM effects although end-to-end systems like CRDNN–CTC, Whisper became more mainstream. Kaldi recipes were used for all steps, following best practices from existing Japanese ASR recipes since creating a new GMM–HMM pipeline from scratch was taking a considerable amount of time and out of scope for this thesis. The GMM-HMM acoustic was trained in stages like below:

- Monophone (mono)
- Context-dependent triphone with delta features (tri1)
- Context-dependent triphone with LDA+MLLT (tri2)
- Context-dependent triphone with SAT (tri3-SAT)
- Decoding with WFST HCLG graph

The conducted steps were split into step that was data and language preparation, acoustic features, training schedule, and decoding graph construction to get CER and WER.

3.6.1 Data and Language Preparation

Before proceeding to acoustic model training, we needed to prepare the Kaldi data directories. This involved creating a `data/` directory with subdirectories for training names `train/`, validation names `valid/`, and test names `test/`. In each subdirectory, the required files were `wav.scp` for 16kHz mono WAV that pointed to the correct paths, `text` for normalized transcripts that mapped utterance IDs to their text, `utt2spk` for utterance-to-speaker mapping that mapped utterance IDs to speaker IDs, and `spk2utt` derived from `utt2spk`. The speaker-ids were assigned based on talk-level speakers to enforce speaker-independence between train/valid/test splits. The Folder structure was like below:

- `data/`
  - `train/`
    - `wav.scp`
    - `text`
    - `utt2spk`
    - `spk2utt`
  - `valid/`
    - `wav.scp`
    - `text`
    - `utt2spk`
    - `spk2utt`
  - `test/`
    - `wav.scp`
    - `text`
    - `utt2spk`
    - `spk2utt`

There were 1000s of kanji characters in Japanese, each with multiple possible readings depending on context. To avoid a combinatorial phone explosion with mixed kanji/kana, we converted transcripts to readings and trained with a *grapheme-level kana phoneset*. The lexicon used kana characters (plus digits and a small set of normalized symbols) as phones, with `sil` as the optional silence phone. The dictionary included:

- `lexicon.txt`: `<token> <space-separated kana symbols>`
- `nonsilence_phones.txt`: full kana inventory (typically $\sim$80–120 symbols)
- `silence_phones.txt` and `optional_silence.txt`: `sil`

This low number of feature made sure model training and decoding were efficient. Also without a very large number of "character" as phones, this helped to mitigate the risk of overfitting and improve generalization.

Model	Features	Dimensionality
Mono	MFCC	13
Tri1	MFCC + Δ + $\Delta\Delta$	39
Tri2	LDA+MLLT (spliced context ± 3)	40
Tri3-SAT	fMLLR (spliced context ± 3)	40

For building language models, the cleaned training transcripts from the data collection pipeline were used as LM text. The text was used to train a pruned 3-gram LM using KenLM's `lmplz` with `-discount_fallback` and light pruning. A test language directory (`lang_kana_test_3g`) was formatted. During decoding, this smaller 3-gram LM was used to build `HCLG` more efficiently, while a larger 4-gram LM was reserved for lattice rescoring to improve accuracy without creating an excessively large decoding graph.

3.6.2 Acoustic Features

For building the GMM–HMM acoustic models, MFCC features with per-speaker CMVN needed to be extracted. For this setup, 13-dim MFCCs per 25 ms frame with 10 ms shift and apply per-speaker CMVN was used. The details of MFCC extraction configuration were as follows:

- `-sample-frequency=16000`, `-frame-length=25`, `-frame-shift=10`
- `-num-ceps=13`, `-num-mel-bins=23`, `-low-freq=20`, `-high-freq=0`
- `-snip-edges=false`, `-use-energy=true`

For monophone training we used only the static 13-dim MFCCs. However, for triphone training we appended dynamic features to capture temporal context. The specifics were as in table below:

Dimensionality was increased in triphone stages to capture more context and improve discriminability. LDA+MLLT and fMLLR transforms were learned during training to optimize feature representation for the GMM–HMM models. (Praveen Kumar P S, 2019)

3.6.3 Training Schedule

The acoustic model was trained in multiple stages, each building on the previous one to increase modeling power and robustness. The first step of training was to build a monophone model using the MFCC+CMVN features. This model provided initial alignments between audio frames and phone sequences derived from the transcripts. The monophone model was relatively simple, but it established a foundation for more complex models. After the monophone model was trained, we proceeded to context-dependent triphone models. The first triphone model (`tri1`) used MFCC features augmented with delta and delta-delta coefficients to capture temporal dynamics. This model leveraged the alignments from the monophone stage to learn context-dependent phone representations. Then the second triphone model (`tri2`) used LDA+MLLT transforms to project the spliced features into a lower-dimensional space that was more suitable for Gaussian modeling. This stage refined the acoustic model further by improving feature representation. Finally, the third triphone model (`tri3-SAT`) incorporated speaker-adaptive training (SAT) using fMLLR transforms. This allowed the model to adapt to individual speaker characteristics, improving robustness across different voices and recording conditions.

3.6.4 Decoding Graph Construction

After training the acoustic model, the next step was to build the decoding graph `HCLG.fst` that combined acoustic, pronunciation, and language model information into a single weighted finite-state transducer (WFST). HCLG was constructed by composing several components: the HMM topology for context-dependent phones (`H`), the context-dependency transducer (`C`), the lexicon transducer (`L`), and the language model transducer (`G`). Each component served a specific purpose in constraining the decoding process.

HCLG $= H \circ C \circ L \circ G.$

In practice, large or dense `G` could cause memory spikes during `fstcomposecontext`. To avoid the "bad FST header" / OOM failures seen with bigger character LMs, a two-step approach was used:

1. Build `LG` using *Kaldi's* `fstbin` utilities (`fsttablecompose` $\rightarrow$

`fstdeterminizestar` → `fstminimizeencoded` → `fstpushspecial`).

2. Compose context with the kana `disambig` symbols to produce `CLG`.
3. Create `Ha` with `make-h-transducer`, then compose `Ha` with `CLG` and finalize (`fstrmsymbols`, `fstrmepslocal`, `fstdeterminizestar`, `add-self-loops`).

Using a compact 3-gram for `HCLG` and deferring the 4-gram to lattice rescoring eliminates the memory pathologies while preserving final accuracy.

3.6.5 Inference and Lattice Rescoring

Before scoring the model, we performed decoding on the held-out test set using the trained acoustic model and the constructed `HCLG.fst` graph. This step used two-pass fMLLR decoding to adapt to speaker characteristics in the test data. In the first pass, we used a speaker-independent (SI) model to estimate fMLLR transforms for each speaker in the test set. These transforms were then applied to the features in the second pass, where we decoded using the adapted features. This two-pass approach helped improve accuracy by normalizing speaker variability.

3.6.6 Scoring

After the model was trained and decoded using GMM-HMM, we evaluated its performance using Character Error Rate (CER) as the primary metric. Since Japanese text was not whitespace-delimited, CER was particularly well-suited for this language. The CER was computed by comparing the decoded hypotheses against the reference transcripts at the character level. Both hypotheses and references were converted into sequences of characters, and we computed the edit distance (substitutions, insertions, deletions) between these sequences. The CER was then calculated as the total number of errors divided by the total number of characters in the reference transcript.

Another important metric reported in this thesis was Word Error Rate (WER). WER was computed by aligning the decoded hypotheses with the reference transcripts at the word level, counting substitutions, insertions, and deletions, and normalizing by the total number of words in the reference. For Japanese, where word boundaries were not explicitly marked, we used a consistent segmentation procedure to tokenize both hypotheses and references into word-like units. This ensured that WER comparisons were fair and meaningful across different models.

The last metric that was used was the Real-Time Factor (RTF). The RTF was used to measure the decoding speed of the GMM-HMM system. RTF was defined as the ratio of the time taken to decode an audio segment to the actual duration of that audio segment. An RTF less than 1 indicated that the model could decode faster than real-time, which was desirable for realtime applications. To obtain a machine- and configuration-stable latency estimate, we performed a 1-job fMLLR decode on the full test set, recorded wall time, and divided by the total audio duration:

RTF = wall-clock decode time (1 job) / total audio seconds

3.7 Model Family B: CRDNN–CTC (End-to-End, Non-Transformer)

This section documented the end-to-end baseline used in this study: a Convolutional Recurrent Deep Neural Network trained with the Connectionist Temporal Classification (CTC) loss. Unlike the GMM–HMM pipeline in previous section, this model mapped speech features directly to character or subword sequences without an explicit pronunciation lexicon or frame-level alignments. It served as a strong non-transformer reference under the same data, normalization, and scoring policies.

3.7.1 Data and Supervision

Training used the speaker-independent splits defined in data collection. For each utterance, a manifest recorded the path to the 16 kHz mono WAV, the duration, and the cleaned and normalized transcript from text preparation. Parallel manifests were created for validation for checkpoint selection and for test for final scoring. No additional alignment files were required.

3.7.2 Acoustic Features and Augmentation

The model consumed log-Mel or closely related spectrogram features computed from the standardized audio. Frames used a typical 25 ms window and 10 ms shift. Mean and variance normalization was applied per utterance or using global statistics for robustness across recording conditions. Augmentation followed the policy in data augmentation. Modest speed perturbation and spectrogram masking in time and frequency improved generalization while staying close to the formal-speech domain.

3.7.3 Training Procedure

Optimization used minibatches with padding and length masks. An adaptive optimizer such as Adam and a scheduled learning rate were applied. Early stopping and model selection were based on validation Character Error Rate computed from decoded hypotheses. This tied checkpoint choice to transcription quality rather than training loss.

3.7.4 Decoding

Two decoding modes were considered. The first was greedy collapse of per-frame argmax predictions, which was fast and had low overhead. The second was beam search with optional shallow-fusion language modeling, which could improve accuracy at higher computational cost. Evaluation used a fixed and documented configuration for fair comparison across systems. Wall-clock time for the full test set was recorded for Real-Time Factor.

3.7.5 Post-Processing and Normalization

CTC outputs were collapsed by removing blanks and merging repeats, then normalized with the same policy applied to references in data collection. Errors then reflected recognition differences rather than formatting.

3.7.6 Scoring

On the held-out test set, Character Error Rate was the primary metric. Word Error Rate was reported using the same word-like segmentation as elsewhere. Real-Time Factor was computed as total decode time divided by total audio duration, following the unified protocol in RTF section.

3.8 Model Family C: Whisper (Transformer Encoder–Decoder, Fine-Tuned)

This section presented the transformer encoder–decoder system based on Whisper, fine-tuned on the curated formal Japanese speech described in data collection. Whisper differed from both GMM–HMM and CRDNN–CTC. It generated text autoregressively with a decoder conditioned on an encoder's learned acoustic representation. Large-scale multilingual pretraining provided strong prior knowledge that was then adapted to the target domain.

3.8.1 Model Overview

The encoder ingested a log-Mel spectrogram with the configuration Whisper expected from pretraining and produced a sequence of latent acoustic embeddings. The decoder generated subword tokens from Whisper's multilingual vocabulary using cross-attention over these embeddings. This setup naturally produced punctuation and formal written style consistent with pretraining.

3.8.2 Dataset Formatting and Inputs

Fine-tuning used the same speaker-independent train, validation, and test partitions and the same normalized transcripts from data collection and text preparation. Audio was supplied as 16 kHz WAV. The Whisper pipeline then applied its canonical front end to compute the spectrogram expected by the pretrained model. Consistency with Whisper's original preprocessing was maintained to avoid feature mismatch.

3.8.3 Fine-Tuning Configuration

Supervised training minimized token-level cross-entropy over the decoder's autoregressive outputs. Depending on resource constraints, either all layers were updated or lower encoder blocks were partially frozen. Effective batch size and a scheduled learning rate were tuned within GPU memory limits. Validation Character Error Rate was monitored periodically, and the checkpoint with the best validation Character Error Rate was selected for testing.

3.8.4 Inference and Decoding

At test time, decoding was performed in Japanese transcribe mode with translation disabled. Greedy decoding provided a fast baseline. Beam search could improve accuracy on longer clauses at additional cost. Subword outputs were detokenized to text. The decoding configuration was fixed for reporting, and total wall time over the test set was recorded for Real-Time Factor.

3.8.5 Normalization and Style Consistency

Light and deterministic normalization aligned hypotheses with the reference style used in this thesis. Numerals and punctuation followed the same policy as the references, and any tags not present in references were removed. Scoring then reflected recognition quality rather than stylistic variation.

3.8.6 Scoring

Evaluation followed the unified protocol. Character Error Rate was the primary metric. Word Error Rate was a supporting metric with the shared segmentation. Real-Time Factor was decode time over audio time. Results were directly comparable to the GMM–HMM and CRDNN–CTC systems because data splits, normalization, and scoring rules were identical.

3.9 Unified Evaluation Protocol

All models in this study—the GMM–HMM baseline, the CRDNN–CTC model, and the fine-tuned Whisper model were evaluated with one protocol. The same held-out test set and the same reference transcripts were used for every system. Evaluation was performed only after model selection; no system was tuned on the test set.

3.9.1 Accuracy: CER and WER

Character Error Rate (CER) is the primary metric because it suits Japanese orthography. Given substitutions S, deletions D, insertions I, and reference length

N_{ref},

$$\text{CER} = \frac{S+D+I}{N_{\text{ref}}}.$$

Word Error Rate (WER) is reported as a secondary metric. It is computed on a shared tokenization (“word-like” units) applied consistently to all systems, using the same segmentation policy as the language model in the GMM–HMM pipeline. Before scoring, the same normalization rules in data collection and text preparation are applied to hypotheses and references to standardize punctuation, numerals, and markup.

3.9.2 Latency: Real-Time Factor

Latency is measured with Real-Time Factor (RTF), the ratio of total decoding time to total audio duration:

$$\text{RTF} = \frac{T_{\text{decode}}}{T_{\text{audio}}}.$$

RTF < 1 indicates faster-than-real-time processing. Timing is end-to-end on the same hardware: GMM–HMM includes feature extraction, likelihoods, and beam search; CRDNN–CTC includes features, neural inference, and greedy/beam decoding; Whisper includes spectrograms, encoder inference, and autoregressive decoding.

3.9.3 Consistency and Comparability

All results followed three rules: (1) identical test utterances with held-out speakers; (2) identical, normalized references; (3) identical scoring (CER on characters, WER on shared tokens, RTF as above). Under these conditions, differences reported in Chapter 4 reflected model architecture, features, and decoding strategy rather than data or scoring mismatches.

3.10 Challenges and Limitations

Challenges in the methodology included ensuring data quality during scraping, handling diverse speaking styles within formal speech, and managing computational resources for training large models like Whisper. Limitations included potential biases in the selected public talks, the representativeness of the dataset for all forms of formal Japanese speech, and the generalizability of results to other languages or dialects. The limitations of each model architecture, such as the GMM–HMM's reliance on handcrafted features or Whisper's dependence on large-scale pretraining, also affected the conclusions drawn from the experiments.

3.11 Ethical Considerations

Ethical considerations in this research included respecting copyright and usage rights when scraping public talks, ensuring that speaker consent was obtained where necessary, and being mindful of privacy concerns related to audio data. Additionally, the potential societal impact of deploying ASR systems, such as accessibility improvements versus risks of misrepresentation or bias in transcription, had to be carefully weighed. The research adhered to ethical guidelines for data collection and model evaluation to mitigate these concerns.

3.12 Chapter Summary

This chapter described the full experimental methodology used in this thesis, from raw data collection to evaluation. The process began with assembling a domain-specific corpus of formal Japanese speech. Public talks, lecture-style presentations, and similar sources were scraped programmatically, and their audio and Japanese subtitles were harvested. The subtitles were cleaned to remove applause markers and other non-speech annotations, and the remaining text was normalized to match a professional, publishable style. The long-form audio was segmented into utterance-level cl ips aligned with timestamped transcript segments. All clips were standardized to a consistent technical format (16 kHz mono WAV), and each clip was paired with its cleaned transcript and assigned to a speaker-independent train, validation, or test split.

On top of this dataset, three different ASR model families were trained and evaluated. The first was a traditional Kaldi-style GMM–HMM system, which used MFCC features with cepstral mean and variance normalization, an explicit pronunciation lexicon, and a decoding graph that integrated acoustic likelihoods with a language model. The second was a CRDNN–CTC model, an end-to-end neural recognizer with a convolutional front end, recurrent temporal modeling, and CTC loss, which learned alignment implicitly and decoded either greedily or with beam search. The third was a fine-tuned Whisper model, which was a transformer encoder–decoder architecture pretrained on multilingual speech and adapted here to the target domain of formal Japanese. Whisper performed autoregressive decoding directly into text-like output, including punctuation and polite forms.

All three systems were evaluated on the same held-out speakers using the same reference transcripts. Character Error Rate (CER) was used as the primary metric, Word Error Rate (WER) was reported as a supporting metric, and Real-Time Factor (RTF) was measured to capture inference speed. The evaluation protocol enforced consistent normalization and scoring rules so that differences in CER, WER, and RTF could be traced back to meaningful differences in modeling approach, feature design, or decoding strategy, rather than to inconsistencies in preprocessing.

Finally, this chapter outlined how ablation studies were used to interpret the results. By contrasting MFCC-based and log-Mel-based front ends, testing augmentation versus no augmentation, varying decoding complexity, and examining the role of transcript normalization, the thesis aimed to separate architectural gains from procedural gains. The next chapter presented the quantitative results of these experiments, including detailed CER, WER, and RTF measurements for each model family, as well as qualitative error analyses that highlighted typical substitution patterns, honorific handling, and stylistic differences between raw hypotheses and the final cleaned transcripts.

CHAPTER FOUR
RESULTS AND DISCUSSIONS

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.

Praveen Kumar P S, H. S. J. (2019). Creation and instigation of triphone based big-lexicon speaker-independent continuous speech recognition framework for kannada language. *International Journal of Innovative Technology and Exploring Engineering*. https://api.semanticscholar.org/CorpusID:241128661

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 multi-channel 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*.