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. In the context of japanase 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.

Most of the ASR pipelines is focusing on the acoustic or end-to-end model. 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 preprocessing 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 and PLP for traditional systems, and log-Mel filterbanks (with or without augmentation) for modern neural models. The preprocessing and feature extraction step can gave a big impact to the accuracy and stability of the model.

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.

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

An accurate and efficient Japanese transcription in ASR is not solely reliant on the model, but also on the preprocessing part like noise handling, VAD/segmentation, normalization (CMVN), and the choice of features (MFCC vs. log-Mel). Previous work in this domain usually focuses on the model itself and rarely discusses the impact of preprocessing choices side by side across model families, and reporting of character error rate (CER), word error rate (WER), and real-time factor (RTF) is often inconsistent. This will cause an issue when picking the correct model pipeline that is reliable and reproducible.

The issue remains that there is still no clear and evidence-based preprocessing and feature extraction that will create lower CER and RTF for formal Japanese language across traditional, hybrid, and fine-tuned transformers model. Because of this, there is still no obvious ASR model to pick that has the best accuracy and reliability in the context of formal Japanese language. Because of that, by comparing the preprocessing pipelines, testing MFCC and log-Mel variants under the same conditions, the performance of GMM-HMM, CRDNN-CTC, and fine-tuned Whisper can be fairly compared, filling a gap in the literature and providing actionable insights for practitioners.

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 preprocessing setup to reduce Character Error Rate (CER) and Real-Time Factor (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 Character Error Rate (CER) and Real-Time Factor (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 noise/reverberation reduction, VAD/segmentation, resampling and normalization (CMVN), and two feature families of MFCC ($\pm\Delta/\Delta\Delta$) and log-Mel (with/without SpecAugment or frame stacking).

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 like "eeto" and "ano", 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-domain Japanese speech with available transcripts. The data that will be used is JSUT corpus (**jsut**). The JSUT corpus consist of 10 hours of read speech from a single female speaker, covering various topics and is designed for TTS research.

*** dataset part is not final yet, need to find more formal Japanese dataset with more speakers and longer duration ***

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 preprocessing and feature extraction choices 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.

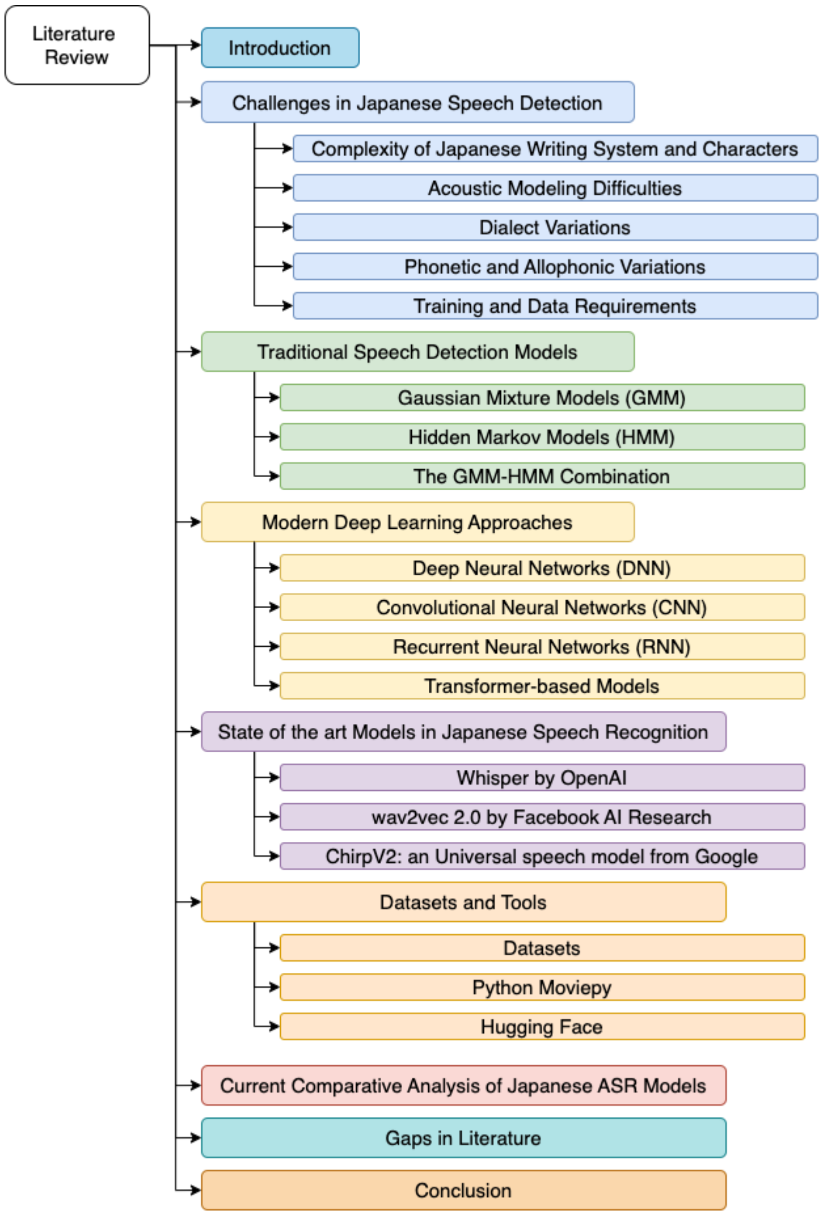
1.7 Conclusion

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
Acoustic Modeling Difficulties
Dialect Variations
Phonetic and Allophonic Variations
Training and Data Requirements
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)
Transformer-based Models
State of the art Models in Japanese Speech Recognition
Whisper by OpenAI
wav2vec 2.0 by Facebook AI Research
ChirpV2: an Universal speech model from Google
Datasets and Tools
Datasets
Python Moviepy
Hugging Face
Current Comparative Analysis of Japanese ASR Models
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, 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, 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 (**ravanelli2019pytorch**). 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 (**ravanelli2019pytorch**; 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 (**graves2006ctc**).

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 (**graves2006ctc**). 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 (**watanabe2018espnet**).

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 (**ravanelli2019pytorch**). 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 (**ravanelli2019pytorch**; **speechbrain2021toolkit**; **watanabe2018espnet**).

2.4.5 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 Transformers Models in Japanese Speech Recognition

2.5.1 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.

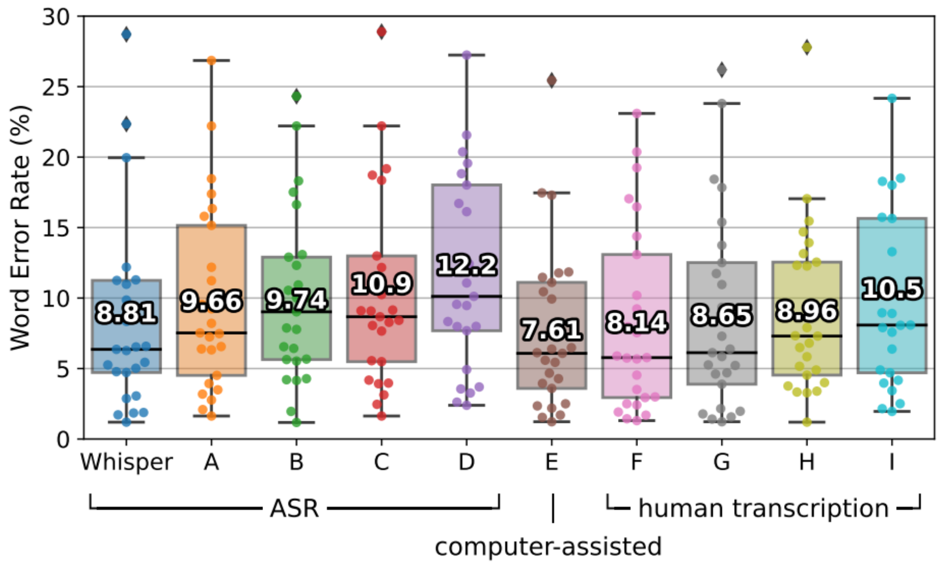


Figure 2.2 Whisper WER cited from Radford et al., 2023

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).

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

2.5.2 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 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.

2.5.3 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 Moviepy

MoviePy is a Python library for video editing that can be used to extract audio from video files. It provides a simple and intuitive interface for working with video and audio files, making it easy to extract audio clips from video files (Boishakhi et al., 2021). One of the key features to use MoviePy in this study is to extract audio from video files and convert them into a format that can be used for training ASR models.

2.7.3 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 explains 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. These models were chosen to represent three major paradigms in ASR: classic hybrid acoustic modeling with an explicit lexicon, recurrent end-to-end modeling with CTC alignment, and large-scale pretrained transformer modeling. All systems are trained and tested on the same curated set of formal Japanese speech (lecture-style and TED-style talks). The audio is collected, cleaned, segmented into utterances, normalized, and converted to a consistent 16 kHz mono WAV format. Transcripts are also normalized to produce a publishable-style reference that is used consistently across all models.

The evaluation focuses on both accuracy and usability. Accuracy is measured using Character Error Rate (CER) as the primary metric, with Word Error Rate (WER) as supporting context. Usability is measured using Real-Time Factor (RTF), which reflects how fast each model can decode speech relative to the length of the audio. The rest of this chapter describes the data pipeline (collection, cleaning, segmentation, and splits), the feature extraction and augmentation strategy, the training and decoding process for each model family, and the scoring protocol used to compute CER, WER, and RTF on the same held-out test set.

3.2 Research Design

This study is organised into four phases: Preparation, Data Collection, Analysis, and Discussion. An overview of the phases, activities, methods, and expected deliverables is shown in Table 3.1.

Table 3.1
Overview of the Research Methodology Plan

Phase	Activities	Methods	Deliverables
Phase 1 — Preparation	Define research area Define problem statement Define research objectives, scope, questions, and significance	Review of related articles and journals Discussion and guidance with supervisor	Approved research proposal Completed Chapter 1 (Introduction)
Phase 2 — Data Collection	Study prior work on Automatic Speech Recognition (ASR) Identify current challenges in Japanese ASR Identify promising techniques for improving transcription accuracy and latency Collect and prepare speech data	Literature review Data collection from: – TED Talks (YouTube) – COJADS dataset Data preprocessing: – Audio segmentation – Normalisation and cleaning of transcripts – Train/validation/test split	Selection of target ASR models (GMM–HMM, CRDNN–CTC, Whisper) Processed and standardised audio–text pairs Experimental environment and training configuration
Phase 3 — Analysis	Train and evaluate the selected ASR models Compute evaluation metrics Compare models quantitatively Analyse error patterns	Model training and inference on held-out test data Performance measurement: – Character Error Rate (CER) – Word Error Rate (WER) – Real-Time Factor (RTF, transcription latency)	Quantitative comparison of GMM–HMM, CRDNN–CTC, and Whisper Strengths, weaknesses, and trade-offs for each model Analysis of accuracy vs. latency
Phase 4 — Discussion	Interpret the results Discuss limitations and challenges Relate findings to the research objectives Outline implications and future work	Synthesis of findings across all models Identification of remaining gaps in Japanese ASR for formal speech	Final dissertation write-up Chapters on discussion, conclusion, and future work

3.3 Data Collection Pipeline

This section describes how the speech and transcript data used in this study were collected and prepared. The objective of the pipeline is to obtain high-quality formal Japanese speech together with aligned reference text that can be used consistently across all three model families evaluated in this thesis (GMM–HMM, CRDNN–CTC, and fine-tuned Whisper). The pipeline begins with source selection, continues through automated scraping of audio and transcripts from online talks, and proceeds through transcript cleaning, audio segmentation, audio standardization, and dataset splitting. The final output of this pipeline is a set of speaker-independent train, validation, and test partitions with aligned audio–text pairs that are suitable for training and evaluating automatic speech recognition (ASR) systems and for computing Word Error Rate (WER), Character Error Rate (CER), and Real-Time Factor (RTF).

3.3.1 Source Selection

The first step is to identify material that matches the target domain of this thesis, which is formal, planned, and relatively careful spoken Japanese. Rather than using spontaneous conversational audio, telephone speech, entertainment media, or noisy street recordings, this work focuses on lecture-style speech such as TED and TEDx talks, invited talks, conference-style presentations, public addresses, and university lectures. Speech in these settings is typically delivered in a polite or semi-formal register, often with deliberate pacing and clear articulation. This style is also closer to professional transcription use cases, such as captioning for broadcast, academic archiving, and government documentation, where clarity and correctness of formal Japanese are prioritized.

Candidate videos are screened manually to ensure that the spoken language is Japanese throughout the relevant portion, that there is a single primary speaker rather than overlapping dialogue, and that the audio is not dominated by background music, excessive reverberation, or heavy compression artifacts. Very short clips are avoided because they provide too little material, and extremely long multi-hour recordings are also avoided because they complicate processing; talks on the order of ten minutes or longer are preferred. Only videos that satisfy these criteria move forward to the scraping stage.

3.3.2 YouTube / TED Talk Scraping in Python

After identifying suitable talks, a Python-based scraping script is used to download both the audio tracks and any available Japanese subtitles. The script maintains a list of pre-screened YouTube URLs (or video IDs), each corresponding to a single talk. For every video in this list, the script retrieves metadata such as the title of the talk, the channel or speaker name, the upload date, and the total duration. This metadata is stored so that each audio segment can always be traced back to its source.

The audio is then downloaded in high quality (for example, as `.m4a` or `.webm`) and kept in its original form temporarily. In a later stage (Section 3.3.5), the audio will be converted into a consistent WAV format for all experiments. At the same time, the script attempts to extract Japanese subtitles. When human-curated Japanese captions are available, they are preferred. If they are not available, automatically generated Japanese captions are used instead. In both cases, the subtitles are obtained as a sequence of time-stamped segments, where each segment has a start time, an end time, and a corresponding text string.

For each subtitle segment, the script stores the source video identifier, the start and end timestamps in seconds, and the raw subtitle text exactly as provided by YouTube. By the end of this scraping stage, we have two aligned views of the same talk: a long-form audio file and a set of short text segments with timing information that serve as initial, noisy transcriptions.

3.3.3 Transcript Cleaning and Normalization

The raw subtitle text is not directly suitable as training or evaluation data. Subtitles often include non-speech annotations such as “[applause]” or “[laughter]”, stage directions, or musical cues. They may also contain line breaks that exist only for on-screen readability, repeated punctuation, or stylistic conventions that do not correspond exactly to what was spoken. In addition, numerical expressions, symbols, and punctuation may appear in inconsistent forms across videos.

To address this, each subtitle segment is passed through a text normalization stage. During normalization, non-speech annotations and audience reactions are removed so that only the intended spoken content remains. Redundant symbols and subtitle-line formatting are stripped out. Numbers and symbols are converted into a consistent written form according to a policy that matches the target of this thesis: formal, publishable Japanese rather than raw conversational disfluency. Where appropriate, filled pauses and hesitation markers such as elongated vowels (for example, eeeee or “anooo”) may be reduced or removed, so that the transcript reflects what a professional transcription service might deliver to a client.

The output of this stage is a cleaned transcript for each time span covered by the subtitle timestamps. This cleaned transcript becomes the reference text for evaluation. In particular, it is the text against which Character Error Rate (CER) is computed, and it is also used when training the end-to-end systems such as CRDNN–CTC and the fine-tuned Whisper model. Applying the same normalization rules consistently across the entire dataset is critical, because all downstream comparisons between models rely on having a single canonical ground truth.

3.3.4 Audio Segmentation

Lecture-style recordings are typically long, sometimes tens of minutes, but most ASR toolchains operate on short utterances paired with short text segments. The time stamps provided in the subtitle data create natural cut points that can be used to break a long recording into manageable clips. After the audio for a given talk has been downloaded, it is aligned with the subtitle segments obtained during scraping. For each subtitle segment, the corresponding region of the audio is extracted using the start and end timestamps.

If a single subtitle segment covers an unusually long span, that span may be further split so that no resulting audio clip exceeds a chosen maximum duration, for example around 15 seconds. When splitting, natural pauses or silence regions are preferred boundaries so that words are not cut mid-utterance. Leading and trailing silence is reduced where possible, but care is taken not to truncate the actual spoken content at the edges.

Each resulting audio clip is assigned a unique utterance identifier that encodes both its source video and its local segment index. This identifier forms the backbone of the dataset: it links the audio clip to its cleaned transcript, to its timing information, and later to any speaker label or talk label. This same identifier will also be used in Kaldi-style data directories for the GMM–HMM pipeline and in JSON or CSV manifests for the neural models.

3.3.5 Resampling and Format Standardization

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

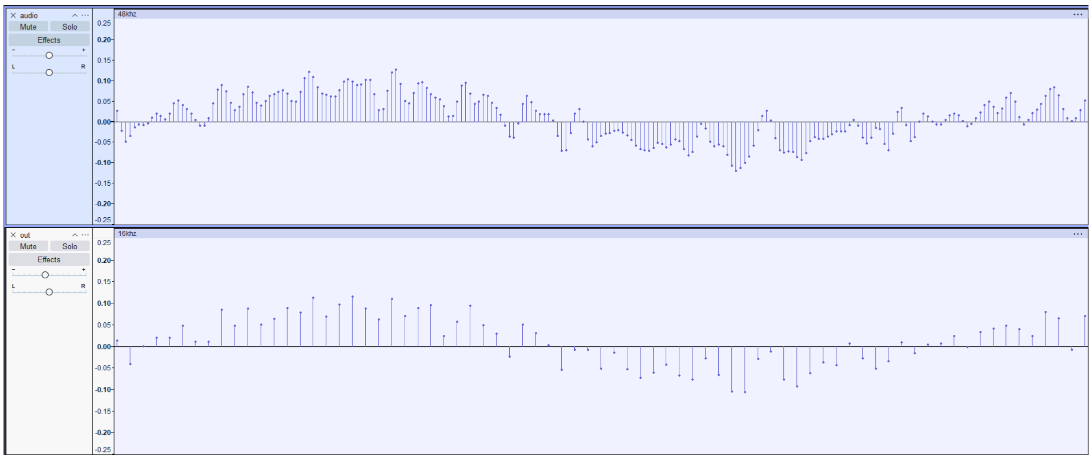


Figure 3.1 Audio resampling from 44.1 kHz(Top) to 16 kHz(Bottom)

Alongside the WAV files, machine-readable metadata is generated. For each utterance ID, the metadata records the path to the audio file, the audio duration, the cleaned transcript text from Section 3.3.3, and basic information about the speaker or the talk. For the GMM–HMM experiments, these fields are also exported into the conventional Kaldi-style files `wav.scp`, `text`, `utt2spk`, and `spk2utt`. For the CRDNN–CTC and Whisper pipelines, the same information is exported into JSON or CSV manifests, which are 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 is to partition the utterances into training, validation, and test sets. This split is constructed to satisfy two requirements. First, it is speaker-independent: the same speaker should not appear in both the training data and the evaluation data. Enforcing this separation prevents the system from overfitting to a particular voice and then benefiting from that familiarity at test time, which would give an unrealistically optimistic error rate. Second, the split is fixed and stable. The test set is defined once and is not touched during model development. A smaller validation set is used for tasks such as early stopping and hyperparameter tuning, while the training set contains the majority of the available material and is used to learn model parameters.

In practical terms, utterances are first grouped by talk (or by speaker, when the talk and speaker are effectively the same). Entire talks are then assigned to either train, validation, or test. The resulting structure is saved on disk using a consistent directory layout, for example `data/train`, `data/valid`, and `data/test`, each of which contains the audio clips and the corresponding transcripts or manifests. Because every model family in this thesis—the GMM–HMM baseline, the CRDNN–CTC model, and the fine-tuned Whisper model—is always trained and evaluated on these same splits, their performance can be compared directly.

The held-out test split defined here is the only data used when reporting objective metrics in later chapters. Specifically, Word Error Rate (WER), Character Error Rate (CER), and Real-Time Factor (RTF) are computed on this test set and only on this test set. By keeping the test set untouched during training and tuning, the reported scores in Chapter 4 reflect generalization to unseen speakers and unseen audio, rather than memorization of the training material.

3.4 Text Preparation for ASR Training

After collecting and segmenting the audio data, the next step is to prepare the corresponding text so that it can be used consistently across all model families in this thesis. This involves three main tasks. First, we must define the basic text units that each acoustic model will predict: phonetic units for the GMM–HMM system, character or subword units for the CRDNN–CTC system, and tokenizer units for Whisper. Second, we construct any additional text resources needed by specific model families, such as the pronunciation lexicon and language model text for the GMM–HMM pipeline. Third, we produce a final, normalized reference transcript for each utterance. This final reference is the single source of truth that will later be used to compute Character Error Rate (CER), Word Error Rate (WER), and Real-Time Factor (RTF). The goal is to make sure that every model is evaluated against the same canonical text under the same normalization rules.

3.4.1 Lexicon and Token Units

Different ASR model families represent “text” in different ways. The GMM–HMM system follows a traditional hybrid ASR design in which there is an explicit pronunciation lexicon and an explicit phone (or phoneme-like) inventory. In this setting, each written unit that appears in the transcript is mapped to one or more pronunciations, and each pronunciation is represented as a sequence of phones. The acoustic model is trained to predict these phones (or context-dependent phone states), and decoding proceeds by searching over phone sequences consistent with both the lexicon and the language model. Preparing this system therefore requires defining a consistent phone set for Japanese, determining which symbols count as silence or noise, and producing a lexicon that links each lexical item in the transcript to its allowed pronunciations. This lexicon becomes part of the standard Kaldi-style `lang/` directory and is later used to construct the decoding graph.

In contrast, the CRDNN–CTC model does not depend on an externally defined pronunciation dictionary. Instead, it is trained to map acoustic features directly to character-like units using the Connectionist Temporal Classification (CTC) loss. In this work, these prediction targets are typically Japanese characters or subword units derived from the cleaned transcripts described in Section 3.3. Because CTC training does not require frame-level alignments, the model can learn the alignment between audio and text implicitly. This simplifies the preparation pipeline: once the final normalized transcript text is available for each utterance, it can be used directly as the target sequence, and there is no need to define or maintain an explicit pronunciation lexicon.

The fine-tuned Whisper model follows yet another strategy. Whisper is a transformer encoder–decoder model that uses a multilingual tokenizer to represent text as a sequence of subword tokens. These tokens are not simple characters; instead, they are learned units from Whisper’s large-scale pretraining, which cover Japanese along with many other languages. During fine-tuning, the model is optimized to continue predicting these tokenizer units, conditioned on the input audio and any decoding settings (e.g., “transcribe” mode without translation). As a result, for

Whisper there is no need to design a new token set. However, it is important that the transcripts fed into Whisper fine-tuning are formatted in a way that matches Whisper's expectations: consistent script usage, stable punctuation rules, and no inserted tags that Whisper would never naturally emit. Any mismatch between training-time targets and Whisper's decoding style can cause degraded performance.

Although these three model families operate on different units (phones, raw characters, or subword tokens), they must all remain aligned to the same semantic content. A polite form such as saseteitadakimasu, for example, should not be normalized away in one system and kept in another, because that would make direct comparison of CER and WER unreliable. For this reason, the transcript normalization policy described in Section 3.3 is applied globally before any model-specific tokenization is carried out.

3.4.2 Language Model Text for the GMM–HMM System

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

Depending on the experiment configuration, additional external text may also be incorporated to improve language coverage. For example, publicly available written Japanese in a similar register (conference abstracts, formal articles, lecture summaries, or encyclopedic prose) can be added to increase lexical diversity and provide better coverage of technical vocabulary or polite forms. When such external data is used, it is subjected to the same normalization rules as the in-domain transcripts to keep punctuation, numerals, and stylistic conventions consistent. The goal is not to build a large, general-purpose language model for arbitrary Japanese text, but rather to model the specific style of well-structured, mostly single-speaker public speech.

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

3.4.3 Final Reference Transcripts for Scoring

All reported evaluation metrics in this thesis — including Character Error Rate (CER), Word Error Rate (WER), and Real-Time Factor (RTF) — are computed against a single, consistent set of reference transcripts. These reference transcripts come from the normalized text produced in the data collection stage (Section 3.3), after cleaning, disfluency handling, and punctuation standardization. No model is allowed to "see" a different or more favorable version of the truth. This is important for fairness: the GMM–HMM, the CRDNN–CTC model, and the fine-tuned Whisper model must all be judged against exactly the same textual target for each utterance in the held-out test set.

For CER, the reference transcript is converted into the agreed-upon character sequence. This sequence typically includes kanji, hiragana, and katakana, with punctuation either retained or removed according to a fixed scoring policy. CER reflects substitutions, insertions, and deletions at the character level, and is especially meaningful for Japanese because spaces are not required between words. For WER, the same reference transcript is segmented into word-like units using a consistent segmentation procedure. Although Japanese does not naturally include whitespace boundaries, defining an explicit segmentation allows us to report WER in a way that is at least internally consistent across experiments. WER is treated as a supporting metric in this thesis, while CER is treated as the primary metric.

It is worth emphasizing that text normalization and scoring policy are tightly connected. For example, if numbers are spoken aloud and then replaced in the transcript by Arabic numerals, a model that outputs the spoken form will be penalized even if it is "correct" from a speech perception point of view. To avoid this ambiguity, numerals and similar constructs are normalized to a stable written form before any model is trained, and that same written form is used everywhere in evaluation. Likewise, non-speech events such as laughter or applause are removed from the transcripts entirely, so that systems are not punished for failing to output tokens that they were never supposed to predict.

Finally, Real-Time Factor (RTF) is computed using the same test set and the same reference transcripts, but it is conceptually different: RTF measures how fast the model produces its hypothesis relative to the length of the audio, rather than how accurate that hypothesis is. By tying CER, WER, and RTF to the same held-out speakers and the same canonical reference text, this thesis ensures that downstream comparisons between traditional hybrid models, end-to-end recurrent models, and transformer-based models remain meaningful and reproducible.

3.5 Feature Extraction and Front-End Processing

Before any acoustic model can be trained, the raw waveform for each utterance must be converted into a numeric representation that is suitable for learning. This section describes the feature extraction pipeline used in this thesis. Although the three model families — GMM–HMM, CRDNN–CTC, and fine-tuned Whisper — ultimately rely on different assumptions about how speech should be represented, they all begin from the standardized 16 kHz mono WAV clips described in Section 3.3. The processing described here covers two main aspects: (i) how acoustic features are computed, normalized, and prepared for consumption by each model family; and (ii) how data augmentation is 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 uses Mel-Frequency Cepstral Coefficients (MFCCs) as its primary acoustic representation. MFCCs are a conventional hand-crafted feature set for ASR. The waveform is 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 is computed.

The resulting magnitude spectrum is then passed through a Mel-scaled filterbank that emphasizes perceptually relevant frequency regions. The log energies in these Mel bands are 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) are appended to capture local dynamics. The final MFCC feature at a given frame therefore encodes both the short-term spectral shape and how that shape is changing over time.

After MFCC extraction, Cepstral Mean and Variance Normalization (CMVN) is applied. The goal of CMVN is 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 can be computed per speaker, using all utterances from that speaker, or at minimum per utterance when speaker identity is not reliably known. In this thesis, because data is collected from multiple unrelated speakers and recording conditions, normalization is applied consistently across all clips so that the acoustic model sees features that are less sensitive to absolute loudness, microphone characteristics, or background coloration. This normalized MFCC representation is then used in all stages of the GMM–HMM acoustic training pipeline, including monophone training, triphone training, and subsequent alignment and re-alignment steps.

The MFCC+CMVN front end is important not only for historical reasons, but also because it provides a baseline reference point when comparing more modern approaches. Later chapters report how this classical front end performs under the same Japanese speech data used by end-to-end models, which helps answer whether such “older” features are still competitive when the domain is formal, relatively clean speech.

3.5.2 Log-Mel Filterbank and Spectrogram-Derived Features

While MFCCs are well matched to GMM–HMM systems, contemporary neural ASR models tend to operate on less aggressively compressed representations such as log-Mel filterbanks or related spectrogram features. In this thesis, the CRDNN–CTC model is trained on features derived from the short-time magnitude spectrum computed over 16 kHz audio. The waveform is framed and windowed in a similar way (for example, 25 ms windows with 10 ms stride), and a Fourier transform is 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 consumes the log-scaled Mel filterbank energies or a closely related log-Mel time–frequency matrix. This preserves more fine-grained structure in the frequency domain, which is especially useful for convolutional front ends. Convolutional layers in CRDNN architectures benefit from two-dimensional structure (time $\times$ frequency) because they can learn local patterns such as formant transitions, onsets, and consonant bursts.

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

The Whisper model follows a similar but more specialized approach. Whisper's original pretraining uses a log-Mel spectrogram computed with a fixed configuration (for example, a particular FFT size, hop length, and Mel filterbank definition) and expects audio resampled to a specific rate. During fine-tuning in this thesis, Whisper continues to consume (and internally normalize) log-Mel spectrogram features consistent with its pretraining regime. This is a key difference from MFCC-based systems: in Whisper and other transformer encoder–decoder models, the learned encoder expects the raw spectral structure, not a hand-designed compact representation. As a result, it is important that the front end for Whisper remains faithful to the preprocessing used during its large-scale pretraining; deviating from this can harm downstream accuracy.

Although MFCCs and log-Mel features share the same physical input (the standardized 16 kHz WAV clips), they embed different assumptions. MFCCs assume that a relatively low-dimensional, decorrelated cepstral space is best for a Gaussian mixture model with diagonal covariances. Log-Mel features assume that downstream neural layers will learn useful patterns directly from a higher-resolution time–frequency surface. One contribution of this thesis is to evaluate both styles of features under matched Japanese data and report how these front ends affect CER, WER, and real-time decoding speed.

3.5.3 Data Augmentation

In addition to extracting features, this work also considers controlled forms of data augmentation. Augmentation is useful because the dataset described in Section 3.3 is drawn from a limited number of speakers and recording setups. Without augmentation, a model might overfit to those specific voices, microphones, or rooms and then degrade noticeably on new speakers in the held-out test set.

One common augmentation strategy is speed perturbation, where the audio waveform is 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 produces additional training examples that mimic natural variations in speaking rate. Speed perturbation effectively changes both the temporal dynamics and the apparent formant trajectories, which encourages the acoustic model to become less sensitive to how quickly a speaker delivers each phrase. In traditional hybrid systems such as GMM–HMM, speed perturbation is often applied before MFCC extraction so that each perturbed version of the audio yields its own set of MFCC features. In end-to-end systems such as CRDNN–CTC, it is usually applied on the fly during training, so the model is repeatedly exposed to slightly different versions of the same utterance.

Another augmentation strategy, more common in neural end-to-end models, is spectrogram masking (often referred to as SpecAugment). After computing log-Mel features, small contiguous time regions or frequency bands are randomly masked out during training. The CRDNN–CTC model can be trained with this kind of masking so that it learns to rely on context rather than any single narrowband cue. Time masking imitates short dropouts or pauses, while frequency masking imitates mild channel distortion or bandwidth limitations. The net effect is improved robustness to local corruption and noise.

Additive noise or reverberation simulation can also be applied, although in this thesis the speech domain is relatively clean lecture-style audio rather than highly noisy conversational speech. For that reason, augmentation is chosen to be conservative: the goal is not to invent extreme background noise that does not exist in the target domain, but rather to introduce realistic variability in rate, bandwidth emphasis, and mild spectral dropouts. This approach keeps the augmented data close to the style of formal Japanese speech that this thesis aims to model.

It is important to note that augmentation policies must be applied consistently when comparing model families. If CRDNN–CTC benefits from aggressive augmentation while the GMM–HMM system is trained only on clean audio, then any accuracy gap might reflect augmentation rather than architectural differences. In this work, augmentation is therefore documented alongside each model's training recipe, and its impact is evaluated explicitly in later chapters. The final reported WER, CER, and RTF values are always computed on the unmodified held-out test set, so that improvements in robustness do not come at the cost of unfair evaluation.

3.6 Model Family A: GMM–HMM (Kaldi-Style Hybrid Baseline)

This section will be detailed the process taken to build the GMM–HMM baseline system used in this thesis. HMM-GMM system that carefully configured still remains a strong point of reference for controlled studies on lexicon design, tokenization, and LM effects although end-to-end systems like CRDNN–CTC, Whisper are become more mainstream. Kaldi recipes will be used for all steps, following best practices from existing Japanese ASR recipes since creating a new GMM–HMM pipeline from scratch will be taking a considerable amount of time and out of scope for this thesis. The GMM-HMM acoustic will be 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 will be split into step that is 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 need to prepare the Kaldi data directories. This involves creating a `data/` directory with subdirectories for training names `train/`, validation names `valid/`, and test names `test/`. In each subdirectory, the required files will be `wav.scp` for 16kHz mono WAV that point to the correct paths, `text` for normalized transcripts that map utterance IDs to their text, `utt2spk` for utterance-to-speaker mapping that maps utterance IDs to speaker IDs, and `spk2utt` derived from `utt2spk`. The speaker-ids will be assigned based on talk-level speakers to enforce speaker-independence between train/valid/test splits. The Folder structure will be like below:

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

  * `spk2utt`

There is 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 convert transcripts to readings and train with a *grapheme-level kana phoneset*. The lexicon uses kana characters (plus digits and a small set of normalized symbols) as phones, with `sil` as the optional silence phone. The dictionary includes:

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

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

For building language models, the cleaned training transcripts from data collection pipeline is used as LM text. Then the text is used to train pruned 3- gram LM using KenLM `lmplz` with `-discount_fallback` and light pruning. A test language directory (`lang_kana_test_3g`) is formatted. Then during decoding, this smaller 3-gram LM is used to build `HCLG` more efficiently, and a larger 4-gram LM is used for lattice rescoring to improve accuracy without creating a huge decoding graph.

3.6.2 Acoustic Features

For building the GMM–HMM acoustic models, MFCC features with per-speaker CMVN needs to be extracted. For this setup, 13-dim MFCCs per 25 ms frame with 10 ms shift and apply per-speaker CMVN is used. The details of MFCC extraction configuration are 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`

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

Dimensionality is increased in triphone stages to capture more context and improve discriminability. LDA+MLLT and fMLLR transforms are 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 is trained in multiple stages, each building on the previous one to increase modeling power and robustness. The first step of training is to build a monophone model using the MFCC+CMVN features. This model provides initial alignments between audio frames and phone sequences derived from the transcripts. The monophone model is relatively simple, but it establishes a foundation for more complex models. After the monophone model is trained, we proceed to context-dependent triphone models. The first triphone model (`tri1`) uses MFCC features augmented with delta and delta-delta coefficients to capture temporal dynamics. This model leverages 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 is more suitable for Gaussian modeling. This stage refines the acoustic model further by improving feature representation. Finally, the third triphone model (`tri3-SAT`) incorporates speaker-adaptive training (SAT) using fMLLR transforms. This allows 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 is to build the decoding graph `HCLG.fst` that combines acoustic, pronunciation, and language model information into a single weighted finite-state transducer (WFST). HCLG is 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 serves a specific purpose in constraining the decoding process.

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

In practice, large or dense `G` can cause memory spikes during `fstcomposecontext`. To avoid the "bad FST header" / OOM failures seen with bigger character LMs, we:

1. Build `LG` using *Kaldi's* `fstbin` utilities (`fsttablecompose` $\rightarrow$ `fstdeterminizestar` $\rightarrow$ `fstminimizeencoded` $\rightarrow$ `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 perform decoding on the held-out test set using the trained acoustic model and the constructed `HCLG.fst` graph. This step uses two-pass fMLLR decoding to adapt to speaker characteristics in the test data. In the first pass, we use a speaker-independent (SI) model to estimate fMLLR transforms for each speaker in the test set. These transforms are then applied to the features in the second pass, where we decode using the adapted features. This two-pass approach helps improve accuracy by normalizing speaker variability.

3.6.6 Scoring

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

WER. Another important metric reported in this thesis is Word Error Rate (WER). WER is 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 are not explicitly marked, we use a consistent segmentation procedure to tokenize both hypotheses and references into word-like units. This ensures that WER comparisons are fair and meaningful across different models.

RTF (real-time factor). The RTF will be used to measure the decoding speed of the GMM-HMM system. RTF is 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 indicates that the model can decode faster than real-time, which is desirable for realtime applications. To obtain a machine- and configuration-stable latency estimate, we perform a 1-job fMLLR decode on the full test set, record wall time, and divide 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 describes the end-to-end acoustic model used as the second major system in this thesis: a Convolutional Recurrent Deep Neural Network trained with the Connectionist Temporal Classification (CTC) loss, referred to here as CRDNN–

CTC. Unlike the GMM–HMM system in Section 3.6, which relies on an explicit pronunciation lexicon, phone-level alignment, and a separately trained language model during decoding, the CRDNN–CTC model is trained to map acoustic features directly to character-level (or subword-level) transcriptions. There is no requirement for forced alignment between frames and labels. Instead, the model learns both acoustic modeling and alignment jointly, by optimizing the CTC objective on pairs of audio and transcripts drawn from the dataset described in Section 3.3. The goal of including this model family is to represent a strong non-transformer baseline from the modern end-to-end ASR literature: it is neural, sequence-to-sequence in spirit, but still lighter and more conventional than large transformer encoder–decoder models such as Whisper.

3.7.1 Architecture and Rationale

The CRDNN architecture used in this work follows a common structure: an initial stack of convolutional layers that operate on time–frequency features, followed by recurrent layers that model longer temporal dependencies, and finally one or more fully connected layers that project into the symbol inventory used for decoding (for example, Japanese characters). The convolutional front end consumes log-Mel filterbank or spectrogram-derived features of the type described in Section 3.5, and learns to detect local acoustic patterns such as consonant bursts, vowel formants, or transitions in place of hand-designed features. Because these convolutional layers typically subsample or stride over time, they also reduce the effective sequence length, which lowers the computational cost for later recurrent processing.

On top of the convolutional stack, the model includes bidirectional recurrent layers, typically using gated units such as LSTMs or GRUs. These recurrent layers allow the network to integrate information over a longer time window than any single convolutional filter can see. Bidirectionality is important here because the task is offline transcription of pre-recorded speech rather than streaming recognition; the model is allowed to condition on both past and future frames when predicting the output sequence. After the recurrent block, a fully connected (feed-forward) projection layer maps each time step to a logit distribution over the output symbol set. In this thesis, the output symbols correspond directly to the normalized Japanese transcription units described in Section 3.4, such as characters or subword units.

The model is trained with the CTC loss. CTC introduces an additional special "blank" symbol and defines an objective that marginalizes over all possible frame-level alignments between the acoustic frames and the target label sequence. This removes the need for frame-aligned supervision or pre-computed phone boundaries. Conceptually, the CRDNN–CTC model learns both what to say (which character sequence best describes the utterance) and when to say it (how that sequence aligns in time), without external alignment steps. This property makes CRDNN–CTC attractive in low- and medium-resource setups, and it is one reason it is selected as a representative non-transformer end-to-end baseline in this thesis.

3.7.2 Data Manifest and Supervision

Training the CRDNN–CTC model requires a manifest that links each utterance-level audio clip to its cleaned transcript. For every utterance in the train split (as defined in Section 3.3), the manifest records the absolute or relative path to the audio file, the duration of the clip, and the normalized reference transcript text. This transcript is exactly the same text that will later be used for scoring Character Error Rate (CER) and Word Error Rate (WER). The manifest may also include identifying information such as the talk or speaker label, which can be useful for diagnostic analysis and optional speaker-dependent normalization, but the core requirement is simply that each clip is paired with its target transcription.

A parallel manifest is created for the validation split. The validation manifest is used during training to monitor overfitting and to select checkpoints. A third manifest is created for the test split, but the test data is not used for training or early stopping. By keeping these manifests aligned with the splits defined earlier, we ensure that the CRDNN–CTC model is evaluated under the same speaker-independent and talk-independent conditions as the GMM–HMM system, and later as the Whisper system.

3.7.3 Feature Pipeline

The CRDNN–CTC model operates on log-Mel or similar spectrogram-derived features extracted from the standardized 16 kHz mono WAV audio described in Section 3.3. Each utterance is framed into short overlapping windows (for example, 25 ms frames with a 10 ms stride), transformed into the frequency domain, and projected onto a Mel-scaled filterbank. The log-Mel energies over time form a two-dimensional time–frequency representation. This representation is then normalized, typically through mean and variance normalization either per utterance or using global statistics computed over the training set.

Unlike MFCCs, which are further decorrelated using a discrete cosine transform and reduced to a small number of cepstral coefficients, log-Mel features preserve local spectral structure. This is helpful for the convolutional front end, which expects a 2D “image-like” input and learns filters that can detect salient acoustic patterns directly. The combination of convolutional layers and log-Mel inputs also makes it straightforward to apply spectrogram-level data augmentation such as time masking and frequency masking. During training, segments of the time–frequency plane may be randomly masked so that the model is forced to rely on broader contextual cues instead of memorizing narrow, local spectral details. This masking strategy improves robustness to mild channel variation or short dropouts without requiring artificially heavy noise injection.

In addition to masking, modest speed perturbation may be applied to the raw waveform before feature extraction. By slightly compressing or stretching the audio in time, the model is exposed to natural variations in speaking rate. This is particularly relevant for public talk data, where individual speakers may have different pacing styles. The goal is to increase generalization across speakers and recording conditions while keeping the augmented data realistic for the target domain.

3.7.4 Training Procedure

The CRDNN–CTC model is trained end to end using minibatch stochastic optimization. Each minibatch consists of multiple utterances drawn from the training manifest. Because utterances vary in length, batching typically uses padding and length-aware masking so that the loss is only computed over valid frames for each sequence. The optimizer (for example, Adam or a similar adaptive method) updates all model parameters jointly: convolutional front end, recurrent stack, and final projection layer. A learning rate schedule is used to gradually reduce the step size over time, which helps stabilize training once the model enters a lower-loss regime.

Early stopping and checkpoint selection are based on the validation split. After some fixed number of optimization steps or epochs, the current model is evaluated on the validation manifest. The evaluation computes the CTC greedy transcription (described below) and measures CER on the validation references. The checkpoint that yields the best validation CER is kept as the final model for testing. This approach makes CER, rather than training loss alone, the main selection criterion, which aligns with how results are ultimately reported in Chapter 4.

One important practical detail is that CTC training does not require external forced alignments, which simplifies the pipeline compared to GMM–HMM training. There is no alternating loop of alignment and re-estimation. Instead, once the manifests and normalized transcripts are prepared, training can proceed directly. This property is valuable when assembling domain-specific datasets like the Japanese formal speech used in this thesis, because it reduces the amount of manual supervision and specialized linguistic resources required.

3.7.5 Decoding and Hypothesis Generation

At inference time, the trained CRDNN–CTC model produces, for each time step, a probability distribution over the output symbols plus the special CTC blank symbol. The simplest decoding strategy is greedy decoding: at each frame, select the most likely symbol, then collapse repeated symbols and remove blanks. This produces a raw character sequence for the utterance without requiring any external language model. Greedy decoding is fast and has low computational overhead, which is useful when measuring the Real-Time Factor (RTF) of the system.

A more accurate but slower alternative is beam search decoding with CTC. In a beam search, instead of keeping only the single most likely symbol at each time step, multiple partial hypotheses are maintained in parallel. These partial hypotheses are extended over time, and low-probability paths are pruned. Beam search can optionally incorporate an external character-level or subword-level language model, which biases the decoding toward sequences that are more plausible in well-formed Japanese. Such shallow fusion between the acoustic model (the CRDNN output probabilities) and the language model can reduce insertion/deletion noise and improve

CER and WER, especially on longer utterances. Whether or not a language model is used, the decoding output is ultimately a sequence of Japanese characters or subword units that is intended to match the style of the normalized reference transcripts.

In this thesis, decoding for evaluation on the test set is performed under controlled settings so that CER, WER, and RTF can be compared fairly across systems. The same test utterances used for GMM–HMM scoring are fed to the CRDNN–CTC model, and decoding is run in a consistent, documented configuration (for example, greedy versus beam). The wall-clock time required to decode the full test set is recorded so that RTF can later be computed in a comparable manner.

3.7.6 Post-Processing of Hypotheses

The raw output sequence from CTC decoding still requires light post-processing before it can be scored. First, CTC collapsing is applied: repeated characters that arise from the alignment structure of CTC are merged, and all blank symbols are removed. Second, the resulting character sequence is normalized so that it matches the same conventions used by the reference transcripts. This normalization includes applying the same punctuation policy, numeral formatting, and handling of formal expressions described in Section 3.4. The goal is that what the model outputs and what the reference contains differ only because of recognition errors, not because of mismatched formatting rules.

This step is important for fairness. If the CRDNN–CTC output preserved filler sounds or hesitations that were intentionally removed from the reference transcripts, the model would be penalized even when it correctly captured the spoken audio. By applying consistent normalization rules to the hypotheses, we ensure that CER and WER reflect genuine transcription quality rather than stylistic disagreements.

3.7.7 Scoring and Metrics

After post-processing, the CRDNN–CTC hypotheses for the held-out test set are scored using the same evaluation protocol as the GMM–HMM system. The primary metric is Character Error Rate (CER), computed by aligning the predicted and reference character sequences and counting substitutions, insertions, and deletions. CER is especially relevant for Japanese because it avoids ambiguities in word boundary definition. Word Error Rate (WER) is also reported for completeness. For WER, both the hypothesis and the reference are segmented into word-like units using the same segmentation method that was used to train and score the GMM–HMM system's language model. Although WER is less natural for Japanese than CER, it provides an additional point of comparison across model families.

In addition to CER and WER, Real-Time Factor (RTF) is measured by timing the decoding process on the test set. RTF is defined as the ratio between the total wall-clock decoding time and the total audio duration. This gives an estimate of how practical the system would be in real or near-real-time transcription scenarios. Because CRDNN–CTC decoding can be done greedily without an expensive search graph, it can often achieve relatively low RTF compared to heavier transformer-based models. On the other hand, adding beam search and an external language model can increase accuracy at the cost of higher RTF.

By evaluating CER, WER, and RTF on the same held-out speakers and the same test utterances used for the GMM–HMM baseline, this thesis is able to compare a traditional hybrid ASR pipeline against a modern end-to-end recurrent architecture under matched conditions. This comparison helps isolate which improvements in accuracy and latency come from architectural advances, which come from feature choices such as log-Mel versus MFCC, and which come from decoding strategies.

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

This section describes the third model family evaluated in this thesis: a transformer-based encoder–decoder model derived from Whisper, fine-tuned on the Japanese formal speech data described in Section 3.3. Whisper differs fundamentally from both the GMM–HMM system in Section 3.6 and the CRDNN–CTC system in Section 3.7. Unlike the GMM–HMM pipeline, Whisper does not require an explicit pronunciation lexicon, external language model, or pre-built decoding graph. Unlike CRDNN–CTC, it does not rely on CTC loss or a collapse step with blanks; instead, Whisper generates text autoregressively, one token at a time, using a decoder that is conditioned on a continuous acoustic representation learned by a transformer encoder. In this sense, Whisper can be seen as a sequence-to-sequence model trained to perform speech recognition directly as conditional generation.

The motivation for including Whisper in this thesis is twofold. First, Whisper represents the current generation of large-scale, multilingual transformer ASR systems, pretrained on massive amounts of weakly supervised speech–text pairs. Second, by fine-tuning Whisper on the domain-specific, cleaned, formal Japanese lecture speech assembled in this work, we can measure how far such pretrained models can be adapted to narrow domains in terms of both accuracy and latency. The evaluation in later chapters therefore compares Whisper not only to a traditional ASR baseline (GMM–HMM) and a recurrent end-to-end baseline (CRDNN–CTC), but also to a high-capacity transformer model that arrives with strong prior knowledge of Japanese.

3.8.1 Model Overview and Motivation

Whisper consists of two main components: an encoder and a decoder, both implemented as transformer stacks. The encoder consumes a log-Mel spectrogram derived from the input waveform. The spectrogram is computed at a fixed sample rate and with a fixed short-time analysis configuration that Whisper expects from its original pretraining. The encoder processes this time–frequency representation and produces a sequence of high-level latent vectors that summarize the acoustic content of the utterance. These encoder outputs function as a learned representation of "what was said", abstracted away from raw waveform details such as pitch, microphone color, or background artifacts.

The decoder then generates the transcription autoregressively, token by token, using cross-attention over the encoder outputs. At each decoding step, the model predicts the next subword token from Whisper's multilingual vocabulary. This vocabulary is based on a tokenizer learned during large-scale pretraining and is shared across many languages, including Japanese. The decoder behaves similarly to a standard transformer language model that happens to be conditioned on the speech representation from the encoder. Because decoding is autoregressive, Whisper is naturally able to insert punctuation, spacing, and other textual conventions it has learned during pretraining.

From an ASR perspective, this architecture has several advantages. First, it does not require an explicit alignment between acoustic frames and text positions. The decoder implicitly learns alignment through attention, rather than through CTC-style monotonic constraints. Second, the output text is produced in its final form rather than as an intermediate symbol sequence that must later be post-processed. This makes Whisper especially suitable for domains where stylistic output (for example punctuation and polite forms) matters, such as captioning or archival transcription of public speech in Japanese. Finally, because Whisper is pretrained on large multilingual corpora, it already has prior exposure to Japanese orthography, polite speech markers, and common discourse patterns. Fine-tuning can then specialize this broad ability to the narrower register studied in this thesis.

3.8.2 Dataset Formatting and Input Representation

Although Whisper comes pretrained, it still requires supervised fine-tuning data in a format that matches its expectations. The starting point for this data is the segmented, speaker-separated, 16 kHz mono audio prepared in Section 3.3, together with the cleaned and normalized transcripts described in Section 3.4. For each utterance in the training split, we create an entry that includes the absolute or relative file path to the audio clip, its duration, and the final reference transcript that will also be used during scoring. The transcript at this stage must follow a consistent style. For example, if the normalization policy removes explicit stage directions such as "[applause]" and regularizes numbers and punctuation to formal written Japanese, then the same policy is applied here so that Whisper is trained to output text in that style from the start.

Whisper expects audio to be resampled to its canonical sample rate and transformed into a log-Mel spectrogram with the same FFT parameters and hop sizes used during its original pretraining. To avoid mismatches, the fine-tuning pipeline does not invent a new feature extractor; instead, it reuses Whisper's own front-end computation. In practice, this means that even though the dataset has already been standardized to 16 kHz WAV for compatibility with the other model families, Whisper may internally resample or reinterpret the audio according to its own specification before feeding it into the encoder. Keeping Whisper's front end intact is important: if the spectrogram differs from what the model saw during pretraining, performance can degrade sharply.

The dataset is split into train, validation, and test using the same speaker-independent partitions introduced earlier. The validation portion is used to monitor fine-tuning progress and select the final checkpoint. The test portion is kept strictly held out and is used only for reporting CER, WER, and Real-Time Factor in later chapters, ensuring a fair comparison with the other systems.

3.8.3 Fine-Tuning Configuration and Optimization

Fine-tuning Whisper on the in-domain Japanese speech involves updating some or all of the model's parameters using supervised learning on the train split. In practical terms, each training sample consists of the input spectrogram from one utterance and the corresponding transcript tokens in Whisper's subword vocabulary. The model is optimized to maximize the likelihood of generating the correct token sequence, conditioned on the audio. Because the decoder is autoregressive, this is equivalent to minimizing the cross-entropy between the predicted token distribution at each step and the ground truth token at that step.

There are several design choices in fine-tuning. One choice is whether to update all layers of the encoder and decoder or to freeze some parts of the network and only train a subset of parameters. Freezing most of the encoder and training only higher layers can reduce memory usage and stabilize convergence on smaller datasets, but may also cap ultimate performance. Another choice is the effective batch size, which is limited by GPU memory because both the encoder activations and the decoder's autoregressive states must be stored for backpropagation. A learning rate schedule is typically applied so that the model takes relatively larger steps early in training and then gradually reduces the step size to refine accuracy.

During training, performance on the validation split is monitored at regular intervals. The model generates transcriptions for the validation utterances using the decoding strategy described in Section 3.8.4, and these hypotheses are scored against the cleaned reference text. Validation Character Error Rate (CER) is treated as the main early stopping signal. The checkpoint that achieves the best CER on the validation set is selected as the final model for evaluation on the held-out test set. This mirrors the procedure used for the CRDNN–CTC model in Section 3.7, ensuring that both neural systems are chosen according to transcription quality rather than raw training loss alone.

3.8.4 Inference and Decoding Strategy

At inference time, Whisper performs conditional text generation. The encoder first processes the log-Mel spectrogram of the input utterance and produces a sequence of latent acoustic embeddings. The decoder then generates the transcript token by token. Decoding can be done greedily (always selecting the most likely next token at each step) or using beam search (maintaining multiple candidate transcriptions in parallel and selecting the one with highest overall probability at the end). Greedy decoding is generally faster and is useful when measuring Real-Time Factor. Beam search typically improves accuracy, especially on longer or more formal utterances where punctuation and clause structure matter, but it also increases decoding cost.

Whisper includes additional decoding controls such as temperature settings, repetition penalties, and suppression of unwanted tokens. For example, since Whisper is multilingual and can also perform translation in some modes, it is important during fine-tuning and evaluation to force the model into "transcribe Japanese as Japanese" mode rather than "translate Japanese into another language." This is done by constraining the decoding setup so that the output language is fixed to Japanese and translation modes are disabled. Similarly, when the target domain expects well-formed written Japanese with standard punctuation, decoding parameters can be chosen to allow natural punctuation rather than aggressively stripping it.

The output of decoding is a sequence of subword tokens from Whisper's tokenizer. These tokens are then detokenized back into Japanese text. Because Whisper was pretrained on large-scale multilingual data, it tends to produce well-structured sentences directly, including punctuation and spacing conventions that resemble written text rather than raw disfluency. This behavior is desirable in the present work, where the reference transcripts are also normalized toward a polished, publishable style.

For evaluation, decoding is run on every utterance in the test split using a fixed configuration (for example, a specific beam size and temperature). Wall-clock decoding time is recorded for the full test set so that Real-Time Factor can be computed later. This procedure parallels the timing strategy for the GMM–HMM and CRDNN–CTC systems, enabling a fair comparison of latency.

3.8.5 Text Normalization and Style Consistency

Although Whisper often produces final-looking text, some light normalization is still applied before scoring. The goal is not to rewrite the model's output, but to make sure that formatting decisions do not artificially inflate error rates. The same normalization policy used for the reference transcripts in Sections 3.3 and 3.4 is applied to Whisper's hypotheses. For example, if the reference consistently writes numbers in a particular way, the hypothesis is converted to match that convention so that differences in numeral formatting are not counted as substitutions. Likewise, any non-speech tags or artifacts that Whisper may occasionally emit due to its pretraining (for example, language ID tokens or timestamp markers, depending on configuration) are removed if those elements were intentionally excluded from the reference transcripts.

This normalization step has the same purpose here as the post-processing step described for CRDNN–CTC in Section 3.7.6: to ensure that CER and WER reflect real recognition errors rather than stylistic mismatches. Because Whisper is capable of inserting punctuation and discourse markers that resemble written Japanese prose, the normalization also checks that this punctuation policy aligns with the reference. The goal of the thesis is to compare transcription quality, not to penalize minor stylistic punctuation choices that fall within the range of acceptable formal Japanese.

3.8.6 Scoring and Metrics

After normalization, the Whisper hypotheses for the held-out test set are evaluated using the same metrics applied to the other model families. The primary metric is Character Error Rate (CER). CER is computed by aligning the predicted and reference character sequences and counting substitutions, insertions, and deletions, then dividing by the number of reference characters. CER is particularly appropriate for Japanese because it does not require pre-defined whitespace boundaries. Word Error Rate (WER) is also reported for completeness. As with the GMM–HMM and CRDNN–CTC systems, WER is computed using a consistent segmentation procedure that defines word-like units in Japanese. WER is treated as a supporting metric, since its value depends more directly on the details of that segmentation.

In addition to CER and WER, Real-Time Factor (RTF) is measured for Whisper. RTF is defined as the ratio between total decoding time and total audio duration for the entire test set. Because Whisper is a transformer encoder–decoder model with autoregressive generation, it can be computationally heavier than a greedy-decoding CRDNN–CTC system, especially if beam search is enabled. Measuring RTF therefore provides a concrete sense of the latency trade-off: Whisper may deliver very strong accuracy after fine-tuning, but potentially at higher inference cost.

By evaluating CER, WER, and RTF on exactly the same held-out speakers and utterances used for the GMM–HMM baseline and the CRDNN–CTC model, this thesis is able to compare three distinct ASR paradigms under matched experimental conditions: a traditional hybrid system with an explicit pronunciation lexicon and language model, an end-to-end recurrent model trained with CTC, and a large pretrained transformer model fine-tuned for in-domain Japanese formal speech. The results in Chapter 4 quantify not only raw transcription accuracy, but also practical considerations such as inference speed and stylistic conformity to professional transcription norms.

3.9 Unified Evaluation Protocol

All models in this thesis — the GMM–HMM baseline from Section 3.6, the CRDNN–CTC model from Section 3.7, and the fine-tuned Whisper model from Section 3.8 — are evaluated using a single, shared protocol. The purpose of this protocol is to ensure that performance comparisons are fair and that reported differences in accuracy or latency reflect genuine model behavior rather than differences in preprocessing, scoring scripts, or data splits. The same held-out test set is used for every system, and the same reference transcripts are used as ground truth for all scoring. All evaluation is performed only after model selection has finished, so that no model is tuned directly on the test data.

3.9.1 Error Metrics: CER and WER

The primary accuracy metric in this thesis is Character Error Rate (CER). CER is well suited to Japanese because Japanese orthography does not require whitespace-delimited word boundaries in the way that English does. To compute CER, the predicted transcription from a model is aligned with the reference transcription at the character level. The alignment counts substitutions (characters that are incorrect), insertions (characters that appear in the hypothesis but not the reference), and deletions (characters missing from the hypothesis). CER is then defined as

$$\mathrm{CER} = \frac{S+D+I}{N_{\mathrm{ref}}},$$

where S is the number of substitutions, D is the number of deletions, I is the number of insertions, and N_{ref} is the total number of characters in the reference transcript. A lower CER indicates a better match between hypothesis and reference.

Word Error Rate (WER) is also reported, but it is treated as a supporting metric. WER is computed in the same general way as CER, except that the alignment is performed over word-like units rather than over individual characters. Because Japanese does not use spaces to mark word boundaries by default, word segmentation must be defined explicitly. In this work, the same segmentation strategy used to train and score the language model in the GMM–HMM pipeline is reused for all systems. This segmentation produces a token sequence that serves as a stand-in for "words." The predicted output and the reference transcript are both segmented in this way, and WER is computed from substitutions, insertions, and deletions at the token level. WER is informative because it approximates how often whole lexical units are wrong, but it depends on the particular segmentation method and therefore is less language-agnostic than CER.

Both CER and WER are always computed after applying the same normalization policy to the model output and to the reference transcript. The normalization rules, described previously in Sections 3.3 and 3.4, remove elements such as applause markers and enforce consistent formatting for punctuation, numerals, and polite forms. This is important because it prevents systems from being penalized for style differences that are irrelevant to transcription quality, and it ensures that all models are judged with respect to the same target text.

3.9.2 Latency Metric: Real-Time Factor

In addition to transcription accuracy, latency is measured using Real-Time Factor (RTF). RTF is defined as the ratio between the total amount of wall-clock time a system spends decoding the test audio and the total duration of that audio. If T_{decode} is the time in seconds that the model took to produce transcriptions for the entire test set, and T_{audio} is the sum of durations of all test utterances in seconds, then

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

An RTF below 1.0 indicates that the system can, in principle, run faster than real time on the measured hardware configuration. An RTF above 1.0 indicates that the system is slower than real time under those conditions. Because deployment settings such as live captioning and broadcast subtitling care strongly about end-to-end delay, RTF provides a practical complement to CER and WER. Two systems with similar CER may have very different RTF values if one depends on an expensive beam search or a large transformer decoder while the other uses a lighter acoustic model and greedy decoding.

For fairness, RTF is measured in a consistent way across models. Each system is run end-to-end on the same held-out test audio. GMM–HMM decoding time includes feature extraction, acoustic likelihood computation, and graph-based beam search. CRDNN–CTC decoding time includes feature extraction, neural inference, and either greedy or beam search decoding, depending on the chosen evaluation setting. Whisper decoding time includes spectrogram computation, transformer encoder inference, and autoregressive decoding with the transformer decoder. Timing is done at the utterance level and accumulated across the entire test set.

3.9.3 Consistency and Comparability

All reported results in this thesis follow three consistency rules. First, every system is evaluated on exactly the same test utterances, drawn from the test split defined in Section 3.3, with strictly held-out speakers. Second, every system is scored against the same reference transcripts, which were cleaned and normalized using a single policy designed to reflect publishable, formal Japanese rather than raw unedited speech. Third, the same scoring procedure is applied: CER is computed on normalized character sequences, WER is computed on consistently segmented token sequences, and RTF is computed using total decoding time divided by total audio duration.

These constraints allow direct comparison of traditional hybrid modeling (GMM–HMM), recurrent end-to-end modeling (CRDNN–CTC), and transformer encoder–decoder modeling (Whisper). When differences appear in Chapter 4, those differences can therefore be attributed to model architecture, feature choices, decoding strategy, or computational cost, and not to mismatched data conditions or unfair evaluation settings.

3.10 Ablation Studies and Controlled Comparisons

Beyond straightforward model evaluation, this thesis also conducts a series of controlled comparisons designed to isolate the impact of specific design decisions. The purpose of these ablations is to understand not only which model performs best overall, but also why: which front-end features matter most, how much benefit comes from data augmentation, how decoding choices affect both CER and RTF, and how much the text normalization policy contributes to final scores.

One axis of comparison is the choice of acoustic front end. The dataset described in Section 3.3 is always the same, but the feature representations differ across models. The GMM–HMM pipeline relies on MFCC features with cepstral mean and variance normalization, which reflect the assumptions of Gaussian mixture acoustic modeling. The CRDNN–CTC model consumes log-Mel or spectrogram-derived features that preserve more local time–frequency structure and are well matched to convolutional front ends. Whisper uses its own fixed log-Mel spectrogram front end inherited from large-scale pretraining. By evaluating all systems on the same held-out speakers, we can compare how these different front ends behave under formal Japanese speech. This addresses a practical question: do classical MFCC features remain competitive in relatively clean lecture-style audio, or do modern log-Mel features always dominate once neural sequence models are introduced?

Another axis of comparison is data augmentation. For CRDNN–CTC, runtime augmentation such as speed perturbation, light time masking, and frequency masking can make the model more robust to differences in speaking rate and mild channel variation. For the GMM–HMM system, speed perturbation can also be applied before MFCC extraction to simulate slower or faster speech and increase the effective size of the training set. Whisper, by contrast, begins with a strong prior from pretraining, so the marginal benefit of additional augmentation during fine-tuning may be smaller or more situational. By running versions of each model with and without augmentation (while keeping all other conditions stable), we can measure how much of the final performance gain is actually due to augmentation, rather than to the model architecture itself. This is important when interpreting results: a model that appears superior may simply have benefited from more aggressive augmentation.

A further controlled comparison concerns decoding strategy and search complexity. The CRDNN–CTC model can be decoded using a simple greedy strategy, which tends to be fast and gives a low Real-Time Factor, or using a beam search strategy, which tends to improve CER and WER but is slower. Whisper can also be decoded greedily or with beam search, and may include heuristics such as temperature control or biasing toward Japanese output only. The GMM–HMM system uses a beam search over a precompiled decoding graph, where pruning thresholds determine how aggressively unlikely hypotheses are discarded. By varying these decoding strategies in a controlled way and measuring both accuracy and RTF, we can observe the trade-off between accuracy and latency. In other words, ablation on decoding settings reveals whether a model's higher accuracy is intrinsically due to its architecture, or whether it is largely the result of spending more compute at inference time.

Another comparison concerns text normalization and "formalization." The transcripts used in this thesis are cleaned to remove elements such as explicit laughter tags and to regularize punctuation, numerals, and polite forms so that the final transcript resembles publishable Japanese. From a deployment standpoint, this matters: organizations interested in archival transcripts or broadcast subtitles often prefer clean written text instead of literal disfluency. However, from a modeling standpoint, enforcing this normalization can hide certain types of errors. By examining examples where the raw model output differs from the normalized reference only in terms of fillers or honorific style, and by contrasting those with examples where the model actually misrecognized content words, we can estimate how much of the error rate comes from linguistic substance and how much comes from stylistic alignment. This analysis helps clarify how usable a model's raw hypotheses are without post-editing, and how much human cleanup would still be required in a downstream workflow.

Taken together, these ablations provide a structured way to interpret the quantitative results in Chapter 4. They separate the effects of feature choice, augmentation strategy, decoding complexity, and stylistic normalization. This is important, because without such controls it would be easy to draw overly broad conclusions about "which model is best" without understanding the conditions under which that conclusion holds. The ablation studies show not only which system yields the lowest CER in absolute terms, but also how sensitive that performance is to choices that might change in a real deployment, such as whether beam search is allowed or how aggressively the output must be formalized.

3.11 Challenges and Limitations

Challenges in the methodology include ensuring data quality during scraping, handling diverse speaking styles within formal speech, and managing computational resources for training large models like Whisper. Limitations include 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 affect the conclusions drawn from the experiments.

3.12 Ethical Considerations

Ethical considerations in this research include respecting copyright and usage rights when scraping public talks, ensuring that speaker consent is 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, must be carefully weighed. The research adheres to ethical guidelines for data collection and model evaluation to mitigate these concerns.

3.13 Chapter Summary

This chapter has described the full experimental methodology used in this thesis, from raw data collection to evaluation. The process begins with assembling a domain-specific corpus of formal Japanese speech. Public talks, lecture-style presentations, and similar sources are scraped programmatically, and their audio and Japanese subtitles are harvested. The subtitles are cleaned to remove applause markers and other non-speech annotations, and the remaining text is normalized to match a professional, publishable style. The long-form audio is segmented into utterance-level cl ips aligned with timestamped transcript segments. All clips are standardized to a consistent technical format (16 kHz mono WAV), and each clip is 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 are trained and evaluated. The first is a traditional Kaldi-style GMM–HMM system, which uses MFCC features with cepstral mean and variance normalization, an explicit pronunciation lexicon, and a decoding graph that integrates acoustic likelihoods with a language model. The second is a CRDNN–CTC model, an end-to-end neural recognizer with a convolutional front end, recurrent temporal modeling, and CTC loss, which learns alignment implicitly and decodes either greedily or with beam search. The third is a fine-tuned Whisper model, which is a transformer encoder–decoder architecture pretrained on multilingual speech and adapted here to the target domain of formal Japanese. Whisper performs autoregressive decoding directly into text-like output, including punctuation and polite forms.

All three systems are evaluated on the same held-out speakers using the same reference transcripts. Character Error Rate (CER) is used as the primary metric, Word Error Rate (WER) is reported as a supporting metric, and Real-Time Factor (RTF) is measured to capture inference speed. The evaluation protocol enforces consistent normalization and scoring rules so that differences in CER, WER, and RTF can 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 are 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 aims to separate architectural gains from procedural gains. The next chapter presents the quantitative results of these experiments, including detailed CER, WER, and RTF measurements for each model family, as well as qualitative error analyses that highlight typical substitution patterns, honorific handling, and stylistic differences between raw hypotheses and the final cleaned transcripts.

REFERENCES

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.

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

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.

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.

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.

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

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

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