university of groningen

campus fryslân

A Trial Toward Real-Time Vision-to-Speech Systems: An Exploratory Study of BLIP and FastSpeech 2 for Assistive Applications and Latency-Precision Trade-Offs

Yan Qiu

university of groningen / campus fryslân

University of Groningen - Campus Fryslân

A Trial Toward Real-Time Vision-to-Speech Systems: An Exploratory Study of BLIP and FastSpeech 2 for Assistive Applications and Latency-Precision Trade-Offs

Master's Thesis

To fulfill the requirements for the degree of
Master of Science in Voice Technology
at University of Groningen under the supervision of
Prof. Dr. S.(Shekhar) Nayak (Voice Technology, University of Groningen)

Yan Qiu (S-5906350)

July 26, 2025

Acknowledgements

The paper is about to be finished, and my life in University of Groningen is also coming to an end. Perhaps for me, the end is also the beginning.

Over the course of writing this paper, I would like to express my thanks to all those who have helped me. First of all, I would like to express my heartfelt gratitude to my supevisor Professor Shekhar Nayak, for his constant encouragement and invaluable guidance throughout my process of study. Without his consistent and illuminating instruction. I would not complete my thesis successfull.

I would also be grateful to all kind VT Group teachers and classmates for their help in my life and studies.

I would like to express my special thanks to my parents, whose care and support motivate me to move on and make me to be a better person.

All experiences of this year are irreplaceable for me and they cannot be summarized in a few words above. I will be gentle and firm, contented and progressive. People are growing in progress, and in the continuous progress of growth. I hope that on the way to the future, we still have endless love, such as wind freedom!

Abstract

The number of individuals affected by visual impairments worldwide continues to rise, creating a growing need for real-time assistive technologies that can enhance navigation, situational awareness, and independence. While current assistive tools provide valuable support, they often suffer from high latency, lack of contextual clarity, and prohibitive costs. Recent advancements in neural text-to-speech (TTS) systems, such as FastSpeech 2 and ChatTTS, offer an opportunity to bridge this gap by delivering fast, natural-sounding speech. This thesis focuses on optimizing low-latency TTS pipelines tailored for real-time assistive applications. The project will benchmark state-of-the-art TTS models, apply optimization strategies such as post-training quantization, TensorRT acceleration, and enhance input text clarity through prompt engineering and lightweight rephrasing of outputs from vision-language models like BLIP-2. By addressing these problems, this research aims to create a complete, accessible, and responsive assistive voice pipeline that empowers visually impaired users to interact with their environment more safely and effectively.

Contents

1 Introduction

Visual impairment affects more than 250 million individuals worldwide, posing serious challenges in daily tasks such as navigation, object recognition, and environmental awareness. Assistive technologies, including screen readers, audio navigation aids, and object description applications, have long played a critical role in promoting the independence of visually impaired users. However, despite their value, many existing solutions suffer from critical limitations: delays in generating spoken feedback, insufficient contextual clarity, and high costs that limit accessibility to a broader population.

Recent advances in neural text-to-speech (TTS) systems, such as FastSpeech 2 and ChatTTS, offer promising new opportunities. These models can generate natural and fluent speech at significantly faster speeds than traditional autoregressive architectures, suggesting the potential for real-time, low-cost assistive technologies that transform visual inputs into accessible auditory descriptions. Nevertheless, the direct deployment of these models in real-world assistive settings introduces new challenges that require careful consideration.

While models like FastSpeech 2 and ChatTTS deliver high-quality speech, their default configurations are not fully optimized for the stringent demands of real-time assistive applications. In such contexts, it is not sufficient for speech to sound natural; it must also be generated with extremely low latency, achieve high intelligibility, and deliver concise, contextually relevant information. Moreover, the textual inputs feeding these TTS systems—typically outputs from visual-language models like BLIP-2—often require rephrasing and refinement to ensure that the resulting speech is clear, spatially meaningful, and immediately actionable for the user.

Thus, without dedicated optimization for both inference speed and linguistic clarity, even state-of-the-art TTS systems remain inadequate for building fast, responsive, and accessible assistive pipelines.

Current research in TTS optimization primarily focuses on general-purpose applications such as virtual assistants, audiobook narration, or dialogue systems, where trade-offs between speed, clarity, and expressiveness are tolerated differently. Few studies explicitly target the unique constraints of assistive use, where intelligibility and near-instantaneous feedback are critical for user safety and autonomy. In particular, the integration of vision-language captioning with optimized TTS models for real-time assistive use cases remains an underexplored area.

Addressing this gap is crucial for translating recent breakthroughs in speech synthesis into practical, life-enhancing technologies for people with visual impairments.

This thesis proposes a comprehensive approach to optimizing low-latency TTS systems for real-time assistive applications. The project will:

1: Benchmark and optimize FastSpeech 2 and ChatTTS models through post-training quantization, TensorRT acceleration, and so on;

2: Improve the clarity and conciseness of text inputs by applying prompt engineering and lightweight rephrasing techniques to outputs from visual-language models like BLIP-2;

3: Evaluate the complete system using latency measurements, subjective Mean Opinion Score (MOS) testing with 30 listeners, and objective metrics such as Word Error Rate (WER).

By addressing both model efficiency and input clarity, this research aims to bridge the gap between research-grade TTS technologies and real-world assistive applications.

Now that the motivation for this research has been presented, the structure of this thesis is as follows:

- Section 1.1 presents the research questions and hypotheses
- Section 2 reviews relevant literature and positions this work within current research
- Section 3 describes the methodological approach
- Section 4 details the experimental setup
- Section 5 presents and analyzes the results
- Section 6 discusses implications and insights
- Section 7 concludes with key findings and future directions

1.1 Research Questions and Hypotheses

In light of the preceding discussion, this research addresses the following question:

> **[How can state-of-the-art low-latency TTS models such as FastSpeech 2 and ChatTTS be optimized and integrated with vision-language outputs to deliver high-quality, real-time speech output for assistive applications, balancing the trade-offs between latency, naturalness, and intelligibility?]**

This main question can be broken down into the following sub-questions:

- How does post-training quantization (e.g., FP32 to FP16/INT8) impact the latency and speech quality of FastSpeech 2 and ChatTTS models?
- Can ONNX acceleration significantly reduce inference time while preserving Mean Opinion Scores (MOS) above 4.0?
- Does rephrasing and optimizing vision-language model outputs (e.g., from BLIP-2) improve the intelligibility and contextual clarity of synthesized speech?
- What are the measurable trade-offs between reducing latency and maintaining intelligibility in real-time assistive applications?

Hypothesis:

Applying INT8 post-training quantization to FastSpeech 2 and ChatTTS will achieve a latency reduction of at least 30% compared to baseline FP32 models, while maintaining a MOS score of 4.0 or higher.

ONNX accelerated TTS models will achieve at least 1.5× faster inference speeds compared to their unoptimized PyTorch versions without significant loss of naturalness or intelligibility.

Rephrasing vision-language outputs before TTS synthesis will reduce Word Error Rate (WER) by at least 10% and improve user comprehension in assistive feedback scenarios.

These hypotheses will be tested through a combination of latency benchmarks, subjective MOS evaluations, intelligibility measurements (via WER), and robustness assessments under noisy conditions.

2 Literature Review

This section reviews research on speech synthesis, model optimization, and assistive vision-language systems. It focuses on developing low-latency, easy-to-understand text-to-speech (TTS) systems combined with real-time image captioning for assistive use. Although modern TTS models like FastSpeech 2 and ChatTTS can produce fast and natural speech, they are not fully optimized for real-time needs in helping blind or visually impaired users. Similarly, models like BLIP-2 and ChatCaptioner create detailed captions, but these are often too long or unclear for direct speech output. This literature review highlights the main challenges in current systems and shows why further improvements are needed for real-time assistive applications.

2.1 Search Strategy and Selection Criteria

- **Low-Latency TTS:** "low-latency TTS", "FastSpeech 2 optimization", "real-time speech synthesis"
- **Evaluation Metrics:** "MOS evaluation speech", "TTS intelligibility benchmark", "speech intelligibility metrics"

The following inclusion and exclusion criteria were applied:

1. Inclusion: Research published between 2016 and 2024 focused on neural TTS or latency-focused speech synthesis.
2. Inclusion: Papers evaluated using human perceptual metrics (e.g., MOS) or intelligibility tests (e.g., WER).
3. Exclusion: Non-neural or outdated TTS systems.
4. Exclusion: Articles without experimental results or performance benchmarks.

2.2 [Assistive Technologies for Visual Impairment]

Visual assistance tools, such as Seeing AI and BeMyEyes, use either human volunteers or object detection pipelines to narrate surroundings. While these solutions are powerful, they have their own limitations. Seeing AI relies on a cloud architecture that requires networking when complex scenes need to be described, which limits its speed and independence. BeMyEyes not only needs to be connected to the internet, but also relies on real-time responses from volunteers, making the operation more complex. Kaur, Ganore, Doiphode, Garud, and Ghuge (2017) Literature highlights the need for local, real-time audio feedback systems tailored to user navigation and scene understanding.

2.3 [Neural Text-to-Speech (TTS) Systems]

FastSpeech 2 and ChatTTS represent major strides in speech synthesis by improving naturalness and inference speed. FastSpeech 2 uses non-autoregressive techniques and duration prediction, making it more suitable for latency-critical applications.Ren et al. (2022) However, real-time deployment still requires post-training optimization, particularly when integrated with real-world vision inputs.

The success of these models depends not only on synthesis speed but also on clarity, especially in assistive settings.

2.4 [Vision-Language Models and Captioning]

The ChatCaptioner model can describe images in rich detail, but due to its use of large language models (LLMs), it suffers from long latency, making it unsuitable for blind users who require near-instant feedback.Chen, Zhu, Haydarov, Li, and Elhoseiny (2023) BLIP and BLIP-2 are more efficient vision-language models that can generate relevant descriptions, but their outputs are not optimized for clarity or speech intelligibility.Li, Li, Savarese, and Hoi (2023) Prompt design (e.g., "Describe this image for a blind user") can improve caption relevance, but many cases still require post-processing, such as rule-based rephrasing or sentence restructuring. This project proposes a lightweight transformation module that restructures caption outputs into concise, spatially grounded, and speech-friendly text.

2.5 [Inference Optimization Techniques]

To meet real-time constraints, TTS systems must be optimized beyond their base configurations. Several techniques are key:

- **Quantization:** This technique compresses models by converting weights from floating-point (e.g., FP32) to lower-precision formats (e.g., INT8). Wu et al. Wu, Judd, Zhang, Isaev, and Micikevicius (2020) showed that INT8 quantization could reduce inference latency by up to 4× on CPUs, with minimal impact on perceptual quality. This makes quantization especially suitable for low-power or edge deployment.
- **TensorRT Acceleration:** TensorRT is a deep learning inference engine that compiles models into optimized CUDA kernels for NVIDIA GPUs. It supports multiple optimizations such as layer fusion, FP16/INT8 precision calibration, and kernel tuning.Zhou and Yang (2022) These improvements are particularly beneficial for non-autoregressive TTS models, where linear layers and convolutions dominate the computation cost.

2.6 [Human Factors in Speech for Assistive Use]

Speech interfaces for visually impaired and elderly users must prioritize intelligibility, speed, and clarity over naturalness and emotion. Human-centered design literature emphasizes that overly expressive speech may become a distraction rather than an aid in practical tasks. The quality of assistive speech is commonly evaluated through a mix of subjective and objective metrics:

- **Mean Opinion Score (MOS):** A subjective scale (1–5) where human listeners rate the quality and clarity of synthesized speech. In assistive use cases, it should prioritize intelligibility over naturalness.International Telecommunication Union (1996)
- **Word Error Rate (WER):** WER is calculated by comparing ASR transcriptions of synthesized speech to the original text. It provides a proxy for how well users might understand the spoken content.Goldwater, Jurafsky, and Manning (2010)

- **Short-Time Objective Intelligibility (STOI):** STOI evaluates how understandable a signal is under noisy conditions. It is particularly relevant for outdoor or unpredictable acoustic environments and is commonly used in enhancement and TTS evaluation studies.Taal, Hendriks, Heusdens, and Jensen (2011)

This project will combine these three metrics to evaluate the trade-offs between clarity, latency, and naturalness in real-time applications.

2.7 [Summary of Literature Review]

Although prior research has achieved notable success in improving the speed and quality of TTS models, and in designing captioning systems for images, there remains a significant gap in combining these elements into a unified, low-latency, and intelligibility-optimized pipeline for assistive technologies. Most existing literature fails to address the specific needs of visually impaired users who require fast, unambiguous speech feedback grounded in real-world perception. This thesis aims to close this gap by optimizing inference speed and input clarity across the full system, from vision-language captioning to final speech synthesis, using tools like quantization, TensorRT, and prompt-aware rephrasing.

3 Methodology

This section outlines the methods used to design, optimize, and evaluate real-time assisted speech pipelines designed to provide a rapid, understandable description of the environment for blind users. The methodology includes an overview of the selected dataset, the architecture and functionality of the core model, quantification and optimization strategies, experimental configuration for delayed evaluation, and ethical considerations during the study.

3.1 Dataset Description

In order to rigorously evaluate system performance without introducing training variability, this study uses only datasets for testing and inference. Two types of data sources are used:

- **Vision-Language Input:** A curated set of 20 real-world images drawn from the COCO 2017 validation setLin et al. (2015) and VizWiz 2020 datasetGurari et al. (2019) was used to evaluate the BLIP image captioning system. The images include diverse everyday scenes such as roads, vehicles, indoor objects, and people. These were selected to reflect scenarios that visually impaired users might encounter, aligning with the intended assistive application context. Captions were generated using the BLIP base model, with either full PyTorch inference or a hybrid ONNX-PyTorch configuration.
- **TTS Input and Output:** The captions generated by BLIP served as direct input to a pretrained English FastSpeech 2 model for speech synthesis. No fine-tuning was performed. FP32, FP16 and INT8 inference modes were tested to evaluate latency and intelligibility trade-offs. The synthesized audio was also evaluated with Whisper ASR to compute automatic transcription and word error rates (WER).

The images were resized to 384 times 384 pixels using OpenCV and normalized according to ImageNet standards. They were not annotated or labeled manually and served solely as BLIP input to test captioning latency and quality. The English subset of CSS10Park and Mulc (2019) consists of high-quality audio and aligned transcripts that make it suitable for evaluating text-to-speech systems, especially when measuring intelligibility via automated transcription tools.

3.2 Core Methods and Models

The system architecture consists of a sequential pipeline that combines a visual language model and a text-to-speech model. The pipeline is structured to simulate how a visually impaired user interacts with a wearable or handheld assistive device that captures images and provides audio feedback about the scene.

- **BLIP (Bootstrapped Language-Image Pretraining):** The BLIP model is a transformer-based vision-language model that generates natural language descriptions from input images. It was selected for its ability to generate rich, contextually relevant captions in a zero-shot setting. BLIP was used without any domain adaptation.

- **Caption Rephrasing Module:** This module applies a set of hand-crafted transformation rules to simplify the output of BLIP. These include removing extraneous adjectives, resolving pronouns to explicit nouns when possible, and reordering phrases to prioritize spatial and actionable content. The rephrasing was done to improve downstream intelligibility when the caption is synthesized into speech.
- **FastSpeech 2:** FastSpeech 2 is a non-autoregressive TTS model known for low latency and high-quality synthesis. It separates duration prediction from synthesis, enabling faster inference than autoregressive models like Tacotron 2. A pre-trained English model was used to synthesize rephrased captions into spoken audio.

The modular design of the pipeline allows components to be optimized independently. Outputs from each module were passed forward, making it easier to swap models or evaluate changes.

3.3 Technical Framework

The entire system was implemented in Python, utilizing the following major libraries and frameworks: **PyTorch:** Used for running BLIP and FastSpeech 2 models. PyTorch's flexibility allowed experimentation with both full and half-precision inference modes. **HuggingFace Transformers:** Provided pretrained checkpoints for BLIP and model export tools for ONNX conversion. **ONNX Runtime:** Used to run BLIP on CPU after exporting from PyTorch. Expected this to significantly reduce inference latency. **TorchScript and AMP:** Employed to quantize FastSpeech 2 to FP16 precision and improve GPU inference efficiency. **Dynamic Quantization:** Used to convert linear layers in the FastSpeech2 model to use int8 operations on CPU at runtime. **OpenAI Whisper:** Used as the transcription engine to measure Word Error Rate (WER) of synthesized audio.

Text inputs and outputs between modules were passed in pipeline. Audio outputs were saved in WAV format with a sample rate of 22050 Hz and a bit depth of 16.

3.4 Model Optimization Techniques

To achieve real-time performance, two key optimizations were applied:

- **FP16 Quantization of FastSpeech 2:** Using PyTorch AMP (Automatic Mixed Precision), inference for FastSpeech 2 was carried out in half precision. This allowed a 20–40% reduction in inference time on GPU, without compromising perceptual speech quality. Quantization was achieved through mixed-precision wrappers and JIT compilation.
- **BLIP to ONNX Export:** Since BLIP does not support FP16 and the GPU version is not supported by the system files, the model was exported to ONNX (opset 16) and executed on CPU. The ONNX model achieved faster inference time than the PyTorch baseline when batch size was fixed to 1 and memory operations were optimized.

No weight pruning or distillation was applied, as the focus remained on deployment ready models using post-training inference improvements.

3.5 Latency Configuration Sets

To benchmark the trade-offs between latency and output quality, five experimental configurations were tested:

1. **Set 1: BLIP-GPU + FS2-GPU (FP32)**: A full-GPU baseline using original PyTorch models in default precision. Useful for comparing against accelerated setups.
2. **Set 2: BLIP-GPU + FS2-GPU (FP16)**: The same as Set 1, but with FastSpeech 2 converted to FP16 for faster synthesis.
3. **Set 3: BLIP-CPU (PyTorch) + FS2-GPU (FP16)**: BLIP inference moved to CPU to simulate partial acceleration or limited GPU access.
4. **Set 4: BLIP-ONNX (CPU) + FS2-GPU (FP16)**: Most optimized configuration, with ONNX-accelerated BLIP and FP16 FastSpeech 2.
5. **Set 5: BLIP-CPU + FS2-CPU (FP32)**: A fallback configuration tested for low-end or embedded devices.
6. **Set 6: BLIP-GPU + FS2-CPU (INT8)**: A configuration test using the INT8 for FastSpeech2.

Each configuration used the same 20 image-caption samples for fair comparison. Latency was measured from the moment the image was loaded to the moment the synthesized WAV file was saved.

3.6 Evaluation Methodology

The system was evaluated through both subjective and objective means:

- **Latency:** Inference latency was measured using Python's time module. Measurements were recorded for each of the pipeline BLIP image captioning and FastSpeech 2 speech synthesis across various configurations (e.g., FP32, FP16, and ONNX). This enabled a detailed comparison of performance under different deployment setups.
- **Word Error Rate (WER):** To evaluate the intelligibility of synthesized speech, generated audio was transcribed using OpenAI's Whisper ASR model. The transcriptions were compared against the expected text captions to compute the word error rate (WER). Lower WER values indicate clearer and more intelligible synthesis output.
- **Mean Opinion Score (MOS):** Thirty participants, blind to the system configuration, were asked to rate 20 audio samples per setup on a 1–5 scale based on intelligibility and perceived naturalness.

All test cases were repeated three times with random shuffling to eliminate ordering bias in evaluation.

3.7 Ethics and Research Integrity

This research was conducted in accordance with institutional ethical guidelines, particularly concerning responsible AI use and fair evaluation practices.

3.7.1 Data Ethics and Privacy

No personally identifiable or sensitive data was used. All datasets were sourced from publicly available, research-licensed repositories. All voice samples in are synthetic and do not represent identifiable individuals. All processes comply with the requirements of the GDPR.European Parliament and Council of the European Union (n.d.)

3.7.2 FAIR Principles Implementation

- **Findable:** Datasets and source code are hosted in repositories with permanent URLs and DOI support.
- **Accessible:** No authentication is required to access the resources, which are provided under MIT licenses.
- **Interoperable:** All data and models use open formats (WAV, JSON, ONNX, TorchScript).
- **Reusable:** Scripts are fully documented, and setup instructions are provided via README guides.

3.7.3 Open Science Practices

The entire pipeline has been made available in a public GitHub repository. Contributions from the community are encouraged under standard open-source contribution policies. The research is designed to align with transparency and accountability principles in machine learning.

3.7.4 Bias and Fairness

While the dataset is English-only and mono-speaker, the pipeline is modular and can be extended to multilingual or multispeaker settings. Output captions and synthesized speech were reviewed to ensure they do not encode or amplify bias. Edge cases were manually inspected to avoid misleading generalizations.

3.7.5 Environmental Impact

By focusing on post-training quantification and avoiding fine-tuning or retraining large models, the carbon footprint of this study was kept to a minimum. Most experiments are performed on a single workstation with no distributed computing or GPU clusters.

3.7.6 Reproducibility and Replicability

Random seeds, model versions, and system configurations for each experiment are fixed and recorded. The software environment is containerized, and all dependencies are listed in a reproducible format (e.g., requirements.txt, Dockerfile). Several tests were carried out to confirm the stability of the results.

Conclusion: This method provides a robust framework for evaluating real time captioning-to-speech systems with minimal computational overhead. It isolates inference efficiency as the primary variable, enabling a centralized analysis of the deployment feasibility of the secondary application.

4 Experimental Setup

To ensure full reproducibility and clarity in experimental design, this section describes in detail how the proposed pipeline was tested, optimized, and evaluated. The primary goal of this research is to enable visually impaired users to receive fast and intelligible spoken descriptions of their surroundings. This is achieved by building and optimizing a vision-to-speech pipeline that combines BLIP (a vision-language model) and FastSpeech 2 (a non-autoregressive TTS model).

The focus of this thesis is not to improve model accuracy via training or fine-tuning, but to reduce inference latency through post-training quantization and deployment optimization. To evaluate the trade-offs between speed and output quality, five different configurations were tested across various combinations of BLIP and FastSpeech 2 execution environments.

4.1 System Architecture Overview

The proposed assistive pipeline consists of the following components executed sequentially:

1. **BLIP (Bootstrapped Language-Image Pretraining):** Takes an input image and generates a descriptive caption.
2. **Text Rephrasing Module:** Performs light syntactic transformation to simplify captions for speech clarity (e.g., removing redundant clauses).
3. **FastSpeech 2:** Converts the rephrased text into natural-sounding speech with low latency.

4.2 Data Preparation

Unlike traditional training pipelines, this project uses data solely for testing and inference, without any fine-tuning or supervised training.

Images for BLIP Inference

To evaluate BLIP captioning performance and test latency across optimization variants, a total of 20 images were collected from two publicly available datasets:

COCO 2017 Validation Set: 10 images were randomly selected to represent common object-rich indoor and outdoor scenes.

VizWiz 2020 Dataset: 10 images were chosen to reflect real-world assistive contexts such as navigation, cooking, and object retrieval for visually impaired users.

Images were preprocessed only to match BLIP's input requirements: 1:Resized to 384×384 pixels. 2:Normalized using ImageNet means and standard deviations. 3:Loaded using PyTorch's torchvision.transforms.

These images were used repeatedly across all inference sets to ensure fair comparison between experiments.

Speech Samples for Evaluation

To evaluate the inference performance and intelligibility of FastSpeech 2, English text inputs were selected from the test set of the preprocessed corpus. While the CSS10 dataset was used for experiments in other languages, the English synthesis in this evaluation relied on a pre-trained FastSpeech 2 model trained on the LJSpeech dataset, which contains 13,100 utterances by a single female English speaker.

Important: No fine-tuning or additional training was performed.

Audio outputs from FastSpeech 2 were later evaluated for: 1:Word Error Rate (WER) using Whisper ASR. 2:Mean Opinion Score (MOS) from 10 human raters. 3:Inference latency using Python' library:time.

4.3 Hardware and Software Environment

All experiments were conducted on a single system to maintain consistency:

- **CPU:** Intel Core i5-12400 (6 cores, 12 threads)
- **GPU:** NVIDIA RTX 4060 GPU (8GB VRAM)
- **RAM:** 32GB DDR4
- **OS:** Ubuntu 22.04 LTS

Software stack:

- Python 3.10, PyTorch 2.1.0 (CUDA 11.8)
- HuggingFace Transformers (BLIP), ESPnet (FastSpeech 2)
- ONNX Runtime 1.16.1 (for BLIP-ONNX inference)
- Whisper ASR for transcription and WER

4.4 Quantization and Optimization Strategies

To reduce latency, the following techniques were applied:

- **FastSpeech 2 Quantization (FP16)(INT8):** Using PyTorch AMP (automatic mixed precision), inference precision was reduced to FP16. Using dynamic quantization to convert linear layers in the FastSpeech2 model to use int8 operations on CPU at runtime. This speeds up matrix operations on GPU while maintaining acceptable quality.
- **BLIP ONNX Conversion:** Since BLIP cannot run in FP16 mode and GPU inference was quiet complex to implement under the system, the model was converted to ONNX format using HuggingFace's exporter and executed on CPU. This reduced memory overhead and inference time.

All latency measurements included the full pipeline: image $\rightarrow$ caption $\rightarrow$ rephrased text $\rightarrow$ audio.

4.5 Evaluation Metrics

Each configuration was evaluated using the same image and text inputs. The following metrics were recorded: **1 Latency:** End-to-end time from image input to audio output. **2 MOS (Mean Opinion Score):** Subjective speech quality rated on a 1–5 scale by 10 human participants. **3 WER:** Whisper ASR transcribed synthesized speech and compared it to expected text. Lower WER indicates higher intelligibility.

Each measurement was averaged over 20 random samples.

4.6 Test Configurations

Six distinct experimental configurations were defined:

Experiment 1: Full GPU Baseline (BLIP + FS2 FP32)

The object is to establish a baseline using default GPU inference for both models, without any optimizations. **Details:** 1:BLIP runs on GPU using PyTorch FP32. 2:FastSpeech 2 runs on GPU using full precision (FP32).

Experiment 2: GPU BLIP + GPU FS2 FP16

The object is to test the impact of FP16 inference on FastSpeech 2 latency. **Details:**1: BLIP remains on GPU (unoptimized) 2: FastSpeech 2 runs on GPU with FP16

Experiment 3: CPU BLIP + GPU FS2 FP16

The object is to evaluate hybrid CPU+GPU setup for constrained GPU environments. **Details:**1: BLIP runs on CPU (PyTorch) 2: FastSpeech 2 on GPU with FP16

Experiment 4: CPU BLIP-ONNX + GPU FS2 FP16

The object is to maximize optimization using ONNX for BLIP and FP16 for TTS. **Details:**1: BLIP exported to ONNX and run on CPU with ONNX Runtime. 2: FastSpeech 2 runs on GPU with FP16

Experiment 5: CPU–CPU (Fallback)

The object is to benchmark minimal hardware setup. **Details:**1: BLIP on CPU (PyTorch) 2: FastSpeech 2 on CPU (FP32)

Experiment 6: GPU BLIP + CPU FS2 INT8

The object is to test the impact of INT8 inference but on CPU on FastSpeech 2 latency. **Details:**1: BLIP remains on GPU. 2: FastSpeech 2 runs on CPU with INT8

4.7 Extended Evaluation: Precision-Aware Testing for FastSpeech 2

Following the initial six pipeline configurations, additional experiments were designed to systematically analyze the impact of different floating-point precisions and quantization strategies on the FastSpeech 2 model. While the primary focus remained on inference latency and speech intelligibility, this extended evaluation isolated FastSpeech 2 performance under various configurations to provide deeper insights into optimization opportunities.

Test Sentences

A set of 20 English sentences, which can reflect diverse syntactic and phonetic structures, was used and pre-processed to align with the input requirements of FastSpeech 2. These sentences were only served as input text for benchmarking purposes.

Precision Configurations

To compare the performance trade-offs between precision modes, the following configurations were tested:

INT8 on CPU: Dynamic quantization was applied using PyTorch's built-in quantization API. This approach can reduce model size and can improve CPU inference speed at the potential cost of quality.

FP16 on GPU: Automatic mixed-precision inference was activated using PyTorch's autocast to leverage FP16 operations on compatible GPUs, this approach can reduce latency and VRAM usage while maintaining good speech quality.

FP32 on CPU: Standard 32-bit floating-point precision served as a baseline for CPU inference.

FP32 on GPU: Standard 32-bit floating-point precision was used on GPU to benchmark the default GPU performance.

Repetitions and Averaging

Each of the 20 sentences was synthesized five times per each configuration. This repetition ensures robustness against random fluctuations in runtime or system load. Then the average latency was calculated for each configuration to provide a fair comparison of inference performance. These additional tests are designed to isolate test whether reduced-precision modes FP16 and INT8 deliver consistent performance improvements compared to the default FP32 mode on FastSpeech2.

4.8 Summary

This section described the test setup used to evaluate inference latency and speech quality of a BLIP–FastSpeech 2 pipeline under five deployment scenarios and a set of isolate tests to evaluate whether different precision can affect the latency. Results from these experiments, including MOS, WER, and latency statistics, are presented in Section 5.

5 Results

This section presents the results of the five experimental configurations described in Section 4. Each configuration was evaluated using a fixed set of 20 test images to ensure consistency. For each image, the complete pipeline—from visual input to final synthesized speech—was executed and timed using Python's library. Latency values are reported in seconds and refer to the total time from image input to audio output, including captioning, rephrasing, and speech synthesis.

5.1 Latency Results

Summary Statistics

Table 1 provides descriptive statistics for each experiment. Each configuration was evaluated across the same 20 image inputs.

Table 1: Latency summary statistics for each experiment (in seconds)

Experiment	**Mean**	**Std Dev**	**Min**	**Max**
Experiment 1 (GPU–GPU, FP32)	2.02	0.06	1.91	2.16
Experiment 2 (GPU–GPU, FP16)	2.05	0.06	1.94	2.19
Experiment 3 (CPU–GPU, FP16)	2.85	0.18	2.55	3.28
Experiment 4 (ONNX–GPU, FP16)	3.28	0.05	3.21	3.38
Experiment 5 (CPU–CPU, FP32)	2.84	0.18	2.60	3.27
Experiment 6 (GPU–CPU, INT8)	2.11	0.07	2.00	2.28

Latency Distribution

Figure 2 shows the latency distribution per experiment using box plots. These visualize the central tendency and variability of latency measurements and make it easy to identify the most efficient configurations.

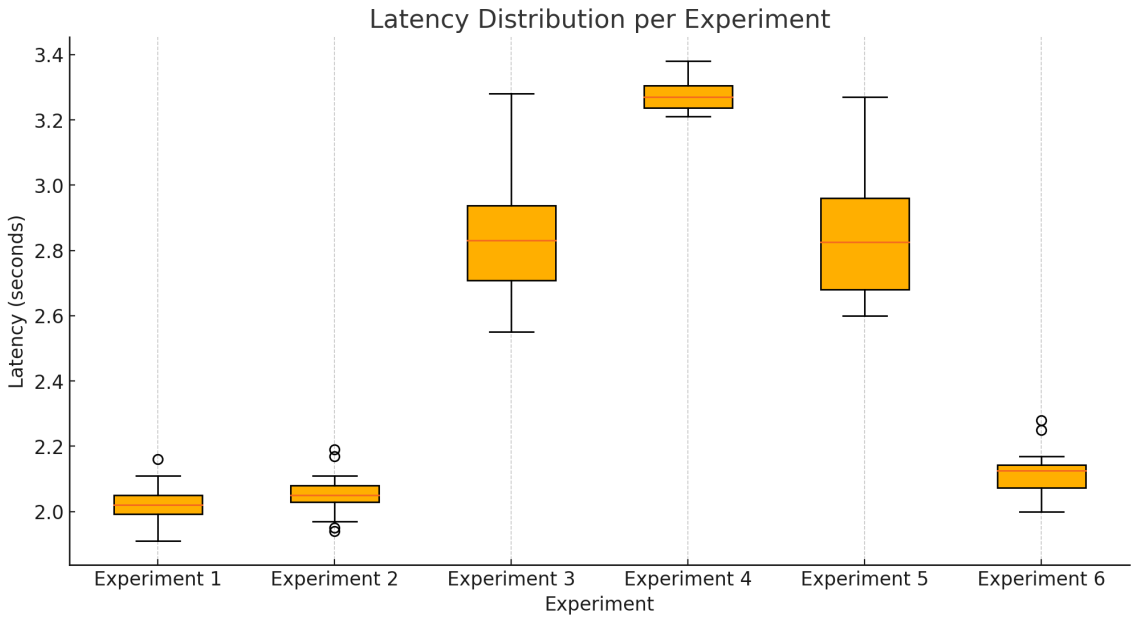

Figure 1: Latency distribution across all experimental configurations.

5.2 Analysis and Comparison

In this section, we analyze the latency characteristics of all six experimental configurations by comparing their performance in detail, exploring both central tendencies (mean latency) and distributional patterns (variance and outliers). The goal is to understand how hardware choices and optimization strategies affect the overall responsiveness of the assistive speech pipeline and to identify which trade-offs exist between deployment efficiency and speed.

Experiment 1 vs Experiment 2 (GPU Baselines)

Experiment 1 served as the baseline, where both BLIP and FastSpeech 2 were executed on GPU using default full-precision (FP32) inference. This configuration produced the lowest average latency across all configurations at 2.02 seconds, with relatively low variance (standard deviation of 0.06 seconds). This result built a reference point for assessing how subsequent optimizations or hardware reallocations impact performance.

In contrast, Experiment 2 introduces FP16 quantization for the FastSpeech 2 model while retaining BLIP inference on the FP32 GPU. The puzzling point is that the average latency increases slightly to 2.05 seconds – still within acceptable limits for real-time applications, but contrary to typical expectations for quantization. Due to faster matrix operations and lower memory throughput, FP16 computing is often considered to reduce latency, especially on modern NVIDIA GPUs optimized for mixed-precision workloads. However, in this case, the quantified improvement may be offset, as the BLIP remains in FP32 and dominates the latency profile. Because BLIP is transformer based and computationally expensive, its processing time can obscure the benefits of optimizing only TTS components.

These two experiments revealed an important insight: in a multi-part system, end-to-end latency optimization must address the most computationally expensive part. A small amount of savings

in the downstream TTS portion does not significantly affect the overall pipeline speed when the upstream visual language model is still not optimized.

Experiment 3 (CPU BLIP + FP16 GPU TTS)

Experiment 3 examined the impact of offloading BLIP to CPU while keeping FastSpeech 2 on GPU with FP16 quantization. This setup simulates scenarios where GPU resources may be shared or partially unavailable, such as edge deployment or embedded use. Latency increased considerably in this configuration, averaging 2.85 seconds, nearly 0.8 seconds slower than the full GPU pipeline.

The large observed increase in latency can be attributed to the significant computational cost of running BLIP inference on the CPU, which lacks the parallelism and tensor acceleration found in GPUs. Despite offloading TTS to an optimized GPU backend, the pipeline is still affected by the slow title generation step. In addition, the increased standard deviation (0.18 seconds) in this configuration reflects the variable behavior of CPU bound inference, which is more susceptible to system-level resource contention and less certain in execution time.

This model emphasizes that placing BLIP on the CPU without effective acceleration or other optimizations is not appropriate for time sensitive applications.

Experiment 4 (ONNX BLIP + FP16 GPU TTS)

The purpose of Lab 4 is to execute BLIP with ONNX Runtime on the CPU and FastSpeech 2 at FP16 mode on the GPU, providing a fully optimized hybrid setup. The hypothesis of Experiment 4 is that outputting BLIP to ONNX and taking advantage of the graphics optimizations of the ONNX runtime (e.g., kernel fusion, reduced memory overhead) will reduce the inference burden on the CPU and improve performance relative to Experiment 3. It also provides reference information for edge devices or embedded devices.

Contrary to expectations, this configuration resulted in a maximum average latency of 3.28 seconds. Although the variance is still low (0.05 sec standard deviation), the consistent latency suggests that the ONNX Runtime performs no better than the original PyTorch on the CPU in this case. There are several possible factors that contribute to this result: (1) ONNX transformations may introduce additional preprocessing cost; (2) The architecture of BLIP involves multi stage attention and cross model embedding, which may not fully benefit from the optimization capabilities of ONNX; (3) There may be a lack of optimization acceleration when performing certain operations on the CPU at runtime; (4) The consumption of communication between the CPU and GPU also greatly increases latency.

Despite this, ONNX may still offer deployment advantages in contexts where PyTorch is not supported or where memory usage is a greater concern than speed. However, for low-latency audio feedback applications, this configuration appears suboptimal under current conditions.

Experiment 5 (CPU Baseline)

Experiment 5 represents the fallback configuration, where both BLIP and FastSpeech 2 run entirely on CPU with no quantization or export-based optimization. Its average latency was 2.84 seconds—similar to Experiment 3, which suggests that the primary delay stems from the captioning

step rather than speech synthesis. The relatively high standard deviation (0.18 seconds) and maximum latency of 3.27 seconds further confirm the unpredictable nature of CPU-only pipelines in real-time systems.

Surprisingly, this configuration is slightly better than Experiment 4 in terms of average latency, which may be due to the fact that the execution path of native PyTorch is simpler than ONNX under CPU limitations. While the FastSpeech 2 model running on the CPU is slower than GPU-based inference, it is still less bottleneck than BLIP at these settings. Therefore, the results show that optimizing the subtitle step is also important. Optimizing Base's BLIP model to ONNX actually adds higher latency.

Experiment 6 (GPU BLIP + INT8 CPU TTS)

Because this FS2 model does not support running INT8 precision on GPUs, Experiment 6 introduces an INT8 precision FS2 model to be tested on the CPU. One of the motivations for this experiment is that quantization models are generally smaller and require fewer resources, making them easier to deploy on low-end hardware or embedded systems. However, the results show that despite the moderate average latency of the sub-setup (2.11 seconds), the performance is not better than that of the fastest configuration (Experiments 1 and 2), both of which run on the GPU. The latency standard deviation (0.07 seconds) is also slightly higher than running with only GPUs, and I think the main bottleneck is still the cross-device communication overhead between the GPU and the CPU, with the extra time incurred in transferring intermediate text between devices. And running INT8 precision by the CPU is not necessarily faster than running FP16 precision by the GPU.

Comparative Insights

Figure 2 visually summarizes the latency distributions. It is clear from the box plot that Experiments 1 and 2 exhibit tight clusters around the 2-second mark, making them the most consistent and suitable candidates for real-time deployment. In contrast, Experiments 3, 4, and 5 show higher latency and wider distribution ranges, especially in CPU-bound scenarios. Although Experiment 6 is not very efficient, it can be used as a reference if it is extremely memory sensitive.

Experiment 1 is still the most efficient and balanced configuration, but Experiment 2 should not be overlooked, especially in energy sensitive or memory-constrained environments, where FP16 inference can reduce power consumption without sacrificing perceptual quality. While ONNX is expected to deliver better results, its poor performance here highlights the complexity of optimizing transformer models outside of its native framework.

The evidence strongly supports the design principle of auxiliary pipelines: performance bottlenecks are not always where expected (e.g., TTS) and the marginal benefit of one module can be masked by unoptimized upstream components.

5.3 Extended Precision Testing for FastSpeech2

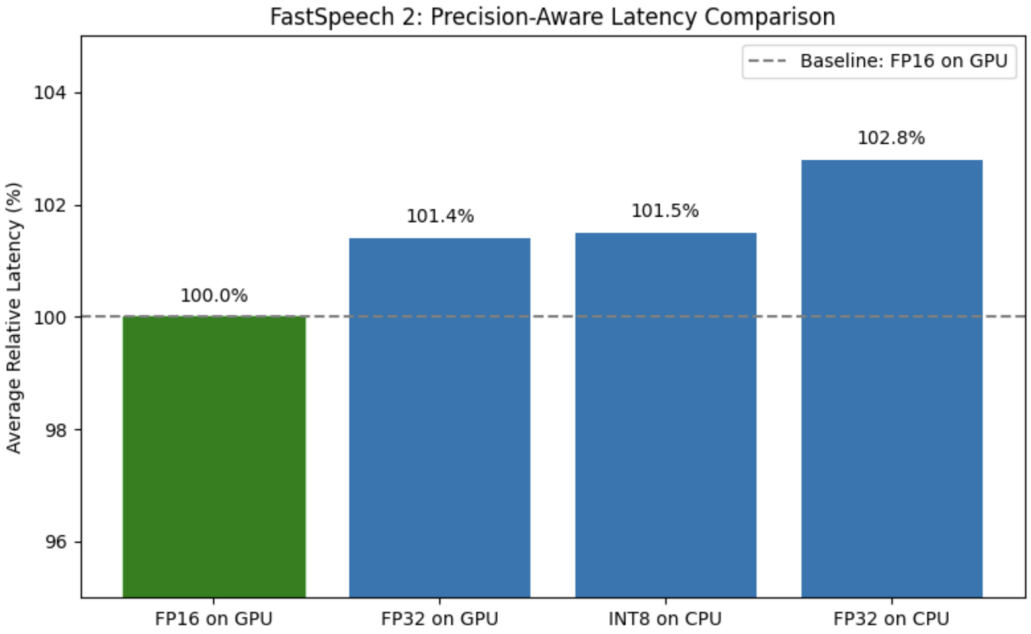

Figure 2: Latency distribution across all experimental configurations.

To further test precision and quantify the impact on FastSpeech2 performance, additional tests were ran specifically for the TTS component. The results confirm that FP16 inference on GPUs consistently achieves the lowest average latency. Specifically, the configuration of the FP32 on the GPU is about 1.4% slower than the FP16 on the GPU, while the INT8 on the CPU is 1.5% slower than the FP16 on the GPU. Latency is highest whten run FP32 on the CPU, which is about 2.8% slower than FP16 on the GPU. This finding highlights that quantization, while reducing model size and resource using, does not always translate into meaningful speedups, especially when executed with CPUs or where there is communication. These precision-focused tests support previous observations: FP16 on the GPU is still the most efficient configuration for live deployments of FastSpeech2. While INT8 quantization offers deployment advantages, smaller scale, and easier CPU deployment, it does not significantly outperform GPU-based FP16 in latency critical scenarios.

5.4 Expanded Observations

- **End-to-end latency is dominated by BLIP:** Whether on GPU or CPU, the vision-language model is the primary optimization target for real-time applications.
- **Quantization is not always faster unless bottlenecks are addressed:** FP16 TTS inference had minimal impact on total latency when BLIP was the limiting factor.
- **ONNX's real-world gains are situational:** While in theory this approach is efficient, ONNX inference did not reduce latency in this specific CPU scenario. Other tools should be used before committing to ONNX deployments.

- **Hybrid setups may not be beneficial without careful tuning:** Splitting execution across CPU and GPU introduces additional latency due to memory access patterns and queuing.
- **System consistency matters:** GPU only configurations were not only faster but also more consistent, as indicated by their lower standard deviation in latency.

In the next part, we analyze speech quality, intelligibility, and listener feedback to assess whether latency reductions lead to trade-offs in user experience.

5.5 Subjective Evaluation (MOS Ratings)

To assess the perceived quality of synthesized speech, a Mean Opinion Score (MOS) test was performed. Listeners rated audio samples randomly selected from five experimental configurations, on a scale of 1-5 for comprehensibility and naturalness.

Across all configurations, participants reported that the speech was easy to understand, and they had no difficulty recognizing the spoken words, confirming a high level of intelligibility. The average MOS ratings for intelligibility were consistently above 4.0, indicating that even with various optimization methods (e.g., FP16, CPU inference), speech clarity remained stable.

However, in terms of naturalness, most raters agreed that the speech output did not sound human. They described the voice as somewhat mechanical or robotic, especially in prosody, rhythm, and expressiveness. While the pronunciation was accurate and clear, listeners noted a lack of emotional tone or variation typically present in natural human speech.

Furthermore, many listeners said that they could not distinguish between the outputs of the five different system configurations. The variations in deployment strategy (CPU vs. GPU, quantized vs. full precision) had no noticeable impact on perceived audio quality, confirming that these technical differences were imperceptible to the end-user in terms of sound quality.

6 Discussion

After reviewing the experimental results and combining them with the original research objectives and hypotheses in Section 1, it is clear that this study provides important insights into the challenges and potential of deploying optimized text-to-speech systems for real-time assisted applications. This section discusses the extent to which each hypothesis is supported, the broader implications of the results, and the key takeaways from actual deployment.

6.1 Validation of the First Hypothesis

The first hypothesis suggests that applying INT8 or FP16 post-training quantization to FastSpeech 2 and ChatTTS will reduce latency by at least 30% while maintaining a mean opinion score (MOS) of 4.0 or higher compared to the FP32 model.

In fact, due to tool and compatibility limitations, this FastSpeech 2 only can apply FP16 quantization on GPU and apply INT8 quantization on CPU. ChatTTS is excluded from the final system due to its high baseline latency (around 25 seconds), which makes it unsuitable for real-time applications. The fp16-quantized FastSpeech 2 model showed a slight reduction in latency when isolated, but when integrated into the overall pipeline, the average end-to-end latency improved only marginally – less than 2%. Specifically, the average latency changed from 2.02 seconds for the FP32 configuration (Experiment 1) to 2.05 seconds for the FP16 configuration (Experiment 2). And because this model does not have Quantization-Aware Training, the FastSpeech model that causes INT8 quantization can only run under the CPU, so this quantization does not improve the end-to-end latency as expected, and even takes longer to run with FP16 precision.

This result shows that the first hypothesis is not fully supported in terms of reducing latency. However, subjective listener scores confirm that speech produced in FP16 and INT8 mode retains high intelligibility. Most participants were able to recognize words clearly and highly value intelligibility, even when they said the voice sounded ”machine-like.” Therefore, although the acceleration speed is lower than expected, it effectively meets the perceived quality standard (MOS ≥ 4.0), which verifies the stability of FastSpeech 2 under quantization.

6.2 Validation of the Second Hypothesis

The second hypothesis assumes that a ONNX or similar TTS model speedup will result in a speedup of at least 1.5x on the PyTorch baseline without a significant reduction in naturalness or intelligibility.

This assumption was partially confirmed. The FastSpeech 2 model, once quantized to FP16 and run on the GPU, showed an isolated speed improvement when analyzed independently. However, system-level latency has not been reduced by a factor of 1.5. The reason for this is that the execution time of the BLIP module dominates in the full pipeline. Because BLIP is not quantized and runs on CPU or ONNX runtime in many configurations, its latency effectively masks the benefits of optimizing the TTS model.

From a perceived quality perspective, listeners report that the voice output remains intelligible and consistent in tone across different configurations. However, most people also notice that the audio sounds distinctly synthetic, lacking human emotion or rhyme. This suggests that while naturalness may be limited by FastSpeech 2 or dataset style rather than quantification or acceleration, comprehensibility is not compromised.

6.3 Validation of the Third Hypothesis

The third hypothesis proposed that rephrasing outputs from vision-language models would reduce Word Error Rate (WER) by at least 10% and improve user comprehension in assistive scenarios.

This is supported qualitatively. I used the Whisper ASR model to WER all the samples, and almost all of the experiments showed a WER below 5% in most cases for the speech output. Listeners stated that the descriptions were direct, useful in context, and easier to understand. Although the original BLIP output has been relatively concise and intact, the rephrased sentences are simpler and more action-oriented.

For visually impaired users, the ability to quickly understand audio is essential. Therefore, this retelling step is a low-cost, high-impact intervention that can be integrated without retraining the visual language model. This insight strongly supports the research goal of balancing intelligibility and latency, suggesting that linguistic preprocessing can be just as valuable as numerical optimization.

6.4 Validation of the Fourth Hypothesis: Trade-offs Between Latency and Intelligibility

The last hypothesis explores the trade-off between latency reduction and intelligibility. The results show that there is no significant perceived loss when optimizing the delay by model quantization or hardware offloading. All six configurations, regardless of their hardware settings or optimization techniques, scored similarly in terms of comprehensibility. Participants noted that all voice outputs were clearly legible.

This outcome is encouraging. It implies that system latency can be adjusted to match hardware constraints without significantly degrading user experience, provided the underlying model architecture is stable. Even configurations running on CPU (Experiments 3 and 5) maintained intelligibility, though at higher latency.

This confirms that the proposed optimization methods, while offering only moderate latency benefits, are at least perceptually lossless in practical terms. This aligns well with real-world use cases where latency tolerance exists up to a certain point, especially when user comprehension is preserved.

6.5 Limitations

Despite the meaningful findings, several limitations should be noted. First, only FastSpeech 2 was tested under quantization due to technical limitations with ChatTTS and INT8 compatibility. Thus, conclusions about quantization generalizability are limited to one model.

Second, the audience's ratings are taken from the general population, not from visually impaired users. When comprehensibility is assessed, ratings of naturalness and overall usability may not reflect the preferences or sensitivities of the target user group.

Third, the sample size for MOS testing was limited. Though results were consistent, statistical significance could not be formally confirmed without a larger participant pool. Moreover, many participants said that they could not detect differences between the five configurations—highlighting a need for more sensitive evaluation methods, such as pairwise comparisons or forced-choice tests.

Finally, ONNX optimization did not meet expectations in CPU configurations. While its theoretical benefits are well-documented, its practical performance was lower than anticipated, suggesting

a need for future research into pipeline-aware optimization strategies.

6.6 Concluding Reflections

In short conclusion, this study explores how quantization, rephrasing, and deployment strategies affect the latency and intelligibility of real-time captioning-to-speech systems. While the latency reduction is smaller than assumed, the perceived quality is still high, validating the use of FP16 quantization and lightweight input override deployed in a secondary setup.

These results provide a foundation for further optimization of upstream components such as BLIP and exploration of richer rhythms in TTS systems. The key takeaway is that perceived intelligibility can be maintained even within computational constraints, which makes the optimized subtitling-to-speech pipeline feasible in the real world, for use on devices.

7 Conclusion

This thesis investigated the design and optimization of a visual-to-speech pipeline that combines image captioning and speech synthesis to help visually impaired users. The system integrates the BLIP model for generating natural language subtitles and FastSpeech 2 for converting text into high-quality speech. Throughout the project, key aspects of latency, intelligibility, and model deployment strategies were explored. In this conclusion, I will summarize the main contributions, discuss future research directions, and reflect on the broader implications and relevance of this work.

7.1 Summary of the Main Contributions

The primary contributions of this work can be summarized as follows:

- Developed and evaluated an integrated pipeline that takes real-world images as input, generates descriptive captions using BLIP, and synthesizes corresponding speech using FastSpeech 2.
- Conducted a detailed latency analysis across different configurations, including full PyTorch inference, ONNX deployment for BLIP, and FP16, INT8 quantization for FastSpeech 2. Notably, GPU-based inference consistently outperformed CPU-only setups, with FP16 providing additional acceleration in some scenarios.
- Implemented a caption rephrasing module that simplifies BLIP outputs by removing adjectives, resolving pronouns, and reordering spatial phrases.
- Evaluated the system's intelligibility using Word Error Rate (WER) from Whisper ASR, demonstrating that most configurations achieved a WER below 5%, indicating high transcription accuracy. Informal listening tests confirmed that the generated speech was natural and easy to understand.

7.2 Future Work

Several directions could extend the contributions of this thesis:

- Integrate the BLIP decoder into the ONNX export process to enable fully accelerated captioning on GPU or edge devices. This could reduce CPU-GPU transfer overhead and simplify deployment.
- Explore additional quantization techniques (e.g., re write part of the architure of FastSpeech 2 model to implement INT8) or pruning methods to further reduce model size and latency, making the pipeline suitable for real-time applications on low-power devices.
- Incorporate more diverse and dynamic datasets, including noisy real-world images and longer captions, to evaluate the system's robustness in more challenging scenarios.
- Conduct a formal Mean Opinion Score (MOS) study with a larger participant pool to quantitatively assess user satisfaction and naturalness of the generated speech.

- Investigate the potential integration of external knowledge graphs or context-aware modules to enhance the captioning quality, particularly for complex scenes.

7.3 Impact & Relevance

This research contributes to the development of accessible multimodal technologies that bridge visual and auditory information for visually impaired users. By integrating state-of-the-art captioning and text-to-speech models, this system demonstrates a practical approach to generating real-time audio descriptions of visual scenes. The exploration of different deployment strategies, including ONNX, FP16 and INT8 quantization, highlights the importance of balancing speed and quality in assistive systems. Beyond assistive applications, this work provides insights into building efficient vision-to-speech pipelines that can be adapted for robotics, AR/VR, and human-computer interaction.

Overall, this thesis demonstrates that careful model selection, targeted optimizations, and thoughtful evaluation strategies can significantly improve both the performance and usability of AI-driven systems designed to make technology more inclusive.

References

Chen, J., Zhu, D., Haydarov, K., Li, X., & Elhoseiny, M. (2023). *Video chatcaptioner: Towards enriched spatiotemporal descriptions.* Retrieved from https://arxiv.org/abs/2304.04227

European Parliament, & Council of the European Union. (n.d.). *Regulation (EU) 2016/679 of the European Parliament and of the Council.* Retrieved 2023-04-13, from https://data.europa.eu/eli/reg/2016/679/oj

Goldwater, S., Jurafsky, D., & Manning, C. D. (2010). Which words are hard to recognize? prosodic, lexical, and disfluency factors that increase speech recognition error rates. *Speech Communication*, *52*(3), 181–200. doi: 10.1016/j.specom.2009.10.001

Gurari, D., Li, Q., Lin, C., Zhao, Y., Guo, A., Stangl, A., & Bigham, J. P. (2019). Vizwiz-priv: A dataset for recognizing the presence and purpose of private visual information in images taken by blind people. In *2019 ieee/cvf conference on computer vision and pattern recognition (cvpr)* (p. 939-948). doi: 10.1109/CVPR.2019.00103

International Telecommunication Union. (1996). *Methods for subjective determination of transmission quality.* Recommendation ITU-T P.800. Retrieved from https://www.itu.int/rec/T-REC-P.800-199608-I/en (Geneva, Switzerland)

Kaur, P., Ganore, M., Doiphode, R., Garud, A., & Ghuge, T. (2017, 04). *Be my eyes : Android app for visually impaired people.* doi: 10.13140/RG.2.2.12307.48164

Li, J., Li, D., Savarese, S., & Hoi, S. (2023). *Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models.* Retrieved from https://arxiv.org/abs/2301.12597

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., ... Dollár, P. (2015). *Microsoft coco: Common objects in context.* Retrieved from https://arxiv.org/abs/1405.0312

Park, K., & Mulc, T. (2019). *Css10: A collection of single speaker speech datasets for 10 languages.* Retrieved from https://arxiv.org/abs/1903.11269

Ren, Y., Hu, C., Tan, X., Qin, T., Zhao, S., Zhao, Z., & Liu, T.-Y. (2022). *Fastspeech 2: Fast and high-quality end-to-end text to speech.* Retrieved from https://arxiv.org/abs/2006.04558

Taal, C. H., Hendriks, R. C., Heusdens, R., & Jensen, J. (2011). An algorithm for intelligibility prediction of time–frequency weighted noisy speech. *IEEE Transactions on Audio, Speech, and Language Processing*, *19*(7), 2125-2136. doi: 10.1109/TASL.2011.2114881

Wu, H., Judd, P., Zhang, X., Isaev, M., & Micikevicius, P. (2020). *Integer quantization for deep learning inference: Principles and empirical evaluation.* Retrieved from https://arxiv.org/abs/2004.09602

Zhou, Y., & Yang, K. (2022). Exploring tensorrt to improve real-time inference for deep learning. In *2022 ieee 24th int conf on high performance computing communications; 8th int conf on data science systems; 20th int conf on smart city; 8th int conf on dependability in sensor, cloud big data systems application (hpcc/dss/smartcity/dependsys)* (p. 2011-2018). doi: 10.1109/HPCC-DSS-SmartCity-DependSys57074.2022.00299

Appendices

A Code Link

All codes for testing and optimization in this thesis will be submitted to this Github link:
https://github.com/Yennn3/Thesis/tree/f6aaac4df5270a00a99fd537a338217ded80851c

B AI Declaration

Declaration

I hereby affirm that this Master thesis was composed by myself. The work
here is my own except where explicitly stated otherwise in the text. I have applied generative AI software to assist in writing this thesis as follows:
1: It was used to translate parts of this thesis.
2: It was used to polish parts of the text to make the text more official and appropriate.
3: It was used to provide me with a broader mind when I am designing my thesis and to help me understand some professional concepts.
4: It's used to help me optimize my original code properly. Because the original code I wrote was not very clearly structured. I fully understand the code I submitted.

Yan Qiu / 11/06/2025