Context Compression for Speech LLM's

Bachelor's Thesis of

Liam Derk Rembold

Artificial Intelligence for Language Technologies (AI4LT) Lab
Institute for Anthropomatics and Robotics (IAR)
KIT Department of Informatics

Reviewer:	Prof. Dr. Jan Niehues
Second reviewer:	Prof. Dr.-Ing. Rainer Stiefelhagen
Advisor:	M.Sc. Danni Liu

31. May 2025 – 30. September 2025

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

Karlsruhe, 30.09.2025

. . . . . .

(Liam Derk Rembold)

Abstract

Text-based Large Language Models (LLMs) have emerged as valuable tools for processing information based on instructions. However, their application also presents challenges, as LLMs often struggle to handle inputs that contain long sequences. The longer the input, the more computational resources are required, and the more difficult it becomes for the LLM to capture long-range dependencies within the text. These challenges become even more pronounced when dealing with speech instead of text. Text is usually tokenized word by word. Speech, in contrast, is inherently longer and produces much longer sequences of feature vectors that the LLM must process.

To address this issue, previous work has proposed adding intermediate processing steps that reduce the number of feature vectors extracted from input audio. Two papers explicitly introduced the concept of context compression, where groups of feature vectors are mapped to a more compact representation that aims to preserve the relevant information from the original sequence. These approaches utilize LLMs or other transformers. Methods that are highly dependent on training and therefore require additional computation resources. In this paper, two training-free approaches, namely skipping and averaging, and a trainable convolution-based approach, are evaluated to examine whether training-free methods provide a valid and less expensive alternative to training-based compression approaches.

To set up the convolution-based approach, a convolution layer is first integrated into the Phi-4-multimodal-instruct environment, serving as an additional compression step. Second, the layer is trained on an Automatic Speech Recognition (ASR) task while keeping the original model frozen. Training is carried out on the LibriSQA dataset. Skipping and averaging do not require any training. We then analyze how the adapted models generalize across different evaluation settings: Assessing ASR capabilities under different conditions, evaluating Question Answering (QA) capabilities on short-form speech, and testing summarization capabilities on long-form speech.

We used the ACL 60/60 dataset to evaluate ASR capabilities, LibriSQA to evaluate QA and ASR capabilities, and Nutshell to evaluate summarization capabilities.

The experiments show that, for ASR, training-free compression methods lead to substantial performance drops of up to 18.87 WER points (scaled by 100) compared to the baseline. In contrast, under the same conditions, the convolution-based method remains within 1.78 WER points (scaled by 100) of the baseline.

For tasks beyond ASR, such as summarization, the differences between compression and no-compression settings are marginal, with performance gaps of up to 0.41 ROUGE-L points (scaled by 100) and 0.04 BERTScore points (scaled by 100). In QA, however,

compression approaches result in more pronounced degradations relative to the baseline, reaching up to 7.94 ROUGE-L points (scaled by 100). For BERTScore, the differences between compression methods remain small, with a maximum of 0.49 points (scaled by 100). Compared to the baseline, they underperform slightly, by margins ranging from 0.65 to 1.14 points (scaled by 100).

Overall, these results demonstrate that compression methods can achieve performance levels comparable to the baseline in tasks beyond ASR, highlighting the quality and effectiveness of the compressed feature vector representation. Moreover, in QA and summarization, simple training-free approaches such as skipping and averaging perform nearly as well as the convolution-based method, indicating that even without task-specific training, compression can remain competitive with more complex, training-dependent approaches.

Zusammenfassung

Textbasierte Large Language Models (LLMs) zeigen große Schwierigkeiten, wenn sie mit langen Texteingaben konfrontiert werden. Je länger die Eingabe, desto mehr Hardwareressourcen werden vom Modell benötigt, und die Identifizierung von inhaltlichen Korrelationen über längere Textabschnitte hinweg wird zunehmend anspruchsvoller. Dieses Problem verstärkt sich, sobald Sprache statt Text verarbeitet wird: Sprache ist von Natur aus länger und erzeugt umfangreichere Sequenzen von Feature-Vektoren, die vom LLM verarbeitet werden müssen.

Um dieses Problem zu mitigieren, haben einige Studien Methoden entwickelt, um die Länge derartiger Sequenzen zu reduzieren. Zwei wissenschaftliche Arbeiten beschäftigen sich explizit mit dem Thema im Rahmen von Context Compression. Ziel der Context Compression ist es, eine Gruppe von Feature-Vektoren auf eine kompakte Repräsentation zu projizieren, wobei die semantische Information der Vektoren weitgehend erhalten bleibt. Die Kompression kann dabei unter anderem von LLMs oder anderen Transformern umgesetzt werden. Aufgrund der Größe dieser Modelle ist das Training jedoch kostenintensiv.

In dieser Arbeit werden daher folgende Ansätze untersucht: zwei Training-freie Kompressionsmethoden (Skipping und Averaging) sowie eine trainierbare, Convolution-basierte Methode. Für die Convolution Methode wird die Schicht zunächst in die Phi-4-Multimodal-instruct-Verarbeitungskette integriert. Anschließend erfolgt eine Trainingsphase, in der ausschließlich die Convolutionsschicht auf der Automatic Speech Recognition (ASR)-Aufgabe mit dem LibriSQA-Datensatz trainiert wird. Für Skipping und Averaging entfällt die Trainingsphase vollständig.

Die Leistung der modifizierten Verarbeitungsketten wird auf Basis verschiedener Aufgaben und Datensätze analysiert. Dazu zählen ASR-Aufgaben unter verschiedenen Bedingungen, Question Answering (QA)-Aufgaben über kürzere Audioeinheiten sowie Summarization-Aufgaben über längere Audioeinheiten. Für die Evaluation werden ACL 60/60 für ASR-Fähigkeiten, LibriSQA für QA und ASR-Fähigkeiten und Nutshell für Summarization-Aufgaben genutzt.

Die Experimente zeigen, dass trainingsfreie Kompressionsmethoden bei ASR zu erheblichen Leistungseinbußen von bis zu 18,87 WER-Punkten (skaliert um den Faktor 100) im Vergleich zur Baseline führen. Im Gegensatz dazu bleibt die Leistung der convolutionsbasierten Methode unter denselben Bedingungen innerhalb von 1,78 WER-Punkten (skaliert um den Faktor 100) zur Baseline.

Für Aufgaben jenseits von ASR, wie etwa die Summarization, sind die Unterschiede zwischen Kompression und keiner Kompression marginal, mit Leistungsunterschieden von bis zu 0,41 ROUGE-L-Punkten (skaliert um den Faktor 100) und 0,04 BERTScore-Punkten (skaliert um den Faktor 100). Beim QA hingegen führen Kompressionsansätze im Vergleich zur Baseline zu deutlich stärkeren Einbußen, mit bis zu 7,94 ROUGE-L-Punkten (skaliert um den Faktor 100). Hinsichtlich des BERTScore bleiben die Unterschiede zwischen den Kompressionsmethoden gering (bis zu 0,49 Punkte, skaliert um den Faktor 100), während sie im Vergleich zur Baseline leicht schlechter abschneiden, mit Abweichungen zwischen 0,65 und 1,14 Punkten (skaliert um den Faktor 100).

Insgesamt zeigen die Ergebnisse, dass Kompressionsmethoden bei Aufgaben jenseits von ASR Leistungen erzielen können, die mit der Baseline vergleichbar sind, was die Qualität und Effektivität der komprimierten Merkmalsvektorrepräsentation unterstreicht. Zudem schneiden bei QA und Summarization einfache, trainingsfreie Ansätze wie Skipping und Averaging nahezu so gut ab wie die convolutions-basierte Methode. Dies weist darauf hin, dass Kompression auch ohne aufgabenspezifisches Training mit komplexeren, trainingsabhängigen Ansätzen konkurrenzfähig bleiben kann.

Contents

List of Figures

List of Tables

1 Introduction

This chapter addresses the motivation of the paper and outlines the corresponding problem statement. It concludes by presenting the research questions the paper seeks to answer.

1.1 Motivation

A context window of a Large Language Model (LLM) defines the maximum length of an input sequence that the model can process effectively without significant performance degradation. Text-based LLMs are limited in this regard, as shown by [32], [19], and [20]. Consequently, extracting information from long texts is challenging and computationally expensive, increasing complexity, particularly when using self-attention layers, as elaborated in [54], [51], and [10]. Approaches that attempt to extend the context window (e.g., [9], [42]) still encounter limitations, showing performance drops for inputs exceeding the context window, as demonstrated by [32], [19], and [20].

Processing speech is accompanied by additional challenges. Speech produces much longer input sequences, especially at a sampling rate of typically 16 kHz (e.g., [43], [1]), which results in a significantly larger number of feature vectors to be considered by the LLM, in contrast to textual input. To illustrate this with an example, when processing text, the sentence "Hello World" is often represented as only two tokens. When processed as speech, the encoders must deal with a large number of frames, $x \gg 2$. If the audio has a length of 3 seconds and the sampling rate is 16,000 kHz, x would be 48,000 frames. Even when using a speech encoder with downsampling, as in Phi-4-multimodal [1], the number of feature vectors that an LLM must process remains higher than for textual input; 48,000 speech frames would correspond to 44 feature vectors.

In addition, speech contains extra features such as silent segments, background noise, and overlapping speech, all of which can degrade quality. Therefore, preprocessing steps are required to carefully prepare the audio prior to further processing, which in turn leads to additional computational cost.

One possible approach to mitigate the challenge of long-context processing is **context compression**, which aims to reduce the size of input representations while preserving the essential information needed for downstream tasks.

1.2 Problem Statement

In essence, context compression subsequently reduces the number of feature vectors extracted from the input audio, potentially losing valuable information. Compression modules that employ more complex compression strategies may require additional training to achieve adequate performance, thereby increasing the demand for computational resources.

This raises the question of how compression modules that require no training can compete across various tasks and datasets with compression modules that need training, and whether the model equipped with the compression module can maintain its original performance without compression, given the potential information loss. Another key aspect is whether compression modules, once integrated into the model architecture, allow the original model, previously trained on diverse tasks, to preserve its performance on these tasks, especially if the compression module was trained for only one of them.

1.3 Research Questions of the Study

The following research questions address the topic of context compression in the context of SpeechLLMs:

RQ1: How can we perform compression without additional training?

These are approaches that have a straightforward implementation and, as a result, can skip the training stage.

RQ2: How do these training-free approaches compare to the dedicated compression modules that are trained?

Demonstrating whether the capabilities of non-trainable compression modules, despite being easier to implement, can compete with trainable ones.

RQ3: How do these two types of approaches generalize to different acoustic conditions and different tasks?

Therefore, conducting a deep dive into the resilience of compression strategies and examining whether their ability to compress knowledge is effective across different tasks and datasets.

2 Background and Related Work

This chapter elaborates on the various types of Large Language Models (LLMs) and their construction and setup, while motivating the use of context compression. First, Section 2.1 introduces the background of Text LLMs by explaining the basic concept of language modeling, listing the architectures and training stages of LLMs, highlighting the attributes of Llama 3 and TOWER. Afterwards, Section 2.2 explains the background of Speech LLMs, starting with speech encoders like HuBERT and Whisper and subsequently covering ways to connect speech encoders with text LLMs. In Section 2.3, we present the challenge of processing long-form input and, finally, elaborate on the concepts of context compression by means of Section 2.4.

2.1 Text-Only LLM

This Section introduces the concept of language modeling and explains how it scales to LLMs. It then discusses the different stages of training and provides a brief overview of text-based LLMs such as Llama 3 and TOWER.

2.1.1 Formalization

Language Models (LMs) acquire a basic understanding of language generation by training. Thereby, applying training objectives such as Causal or Masked Language Modeling, as mentioned in [66] and studied by [35].

Causal Language Modeling (CLM)

In text generation, models must produce language sequentially and unidirectionally from left to right. The core idea is that a model takes an input text or prompt and then iteratively generates the next token until completion. This process is called Causal Language Modeling (CLM) because the probability of generating each next token is conditioned only on the tokens that appear before it in the sequence.

Given a sequence of tokens $\mathbf{x} = (x_1, x_2, \ldots, x_T)$, the model estimates the loss of the sequence in an autoregressive manner:

$$\mathcal{L}_{\text{CLM}} = -\sum_{t=1}^{T} \log P(x_t \mid x_{<t}) \tag{2.1}$$

where $x_{<t} = (x_1, \ldots, x_{t-1})$. At each step, the next token x_t is predicted and conditioned only on the tokens that appear before it.

Masked Language Modeling (MLM)

The unidirectional approach of Causal Language Modeling inevitably restricts the capacity to leverage the full context of a text, since models are prevented from incorporating or generating tokens that appear later in the sequence.

However, this limitation does not occur when using another approach called Masked Language Modeling (MLM). MLMs learn to generate tokens by looking at a bidirectional context. Instead of predicting the sequence sequentially, Given a sequence of tokens $\mathbf{x} = (x_1, \ldots, x_T)$, MLM randomly masks a subset of tokens $\mathcal{M} \subset \{1, \ldots, T\}$ in the input and trains the model to recover them:

$$\mathcal{L}_{\text{MLM}} = -\sum_{t \in \mathcal{M}} \log P(x_t \mid \mathbf{x}_{\setminus \mathcal{M}}), \tag{2.2}$$

where $\mathbf{x}_{\setminus \mathcal{M}}$ denotes the observed tokens (i.e., all tokens except the masked ones).

It is unusual to apply Masked Language Modeling (MLM) in Auto-regressive architectures. However, a speech encoder model like HuBERT [21], as mentioned in Section 2.2.1, which employs an Auto-encoding architecture, applies the MLM approach to learn discrete units by predicting masked frames. This shows that the concept of token prediction can be transferred to the speech domain.

2.1.2 Large Language Model (LLM)

By increasing the number of parameters and the amount of training data, following the scaling laws [24], studies (e.g., [57], [66]) have shown that LMs acquire additional capabilities. This discovery has led to the emergence of LLMs. In order for LLMs to become real general-purpose task solvers (e.g., [53], [59], [37]), LLMs are required to undergo instruction finetuning, as elaborated in [38]. The meaning of instruction finetuning will be elaborated in Section 2.1.3.

Scaling LLMs is one aspect, but it must take place within a specific architecture. Three well-known architectures for LLMs are Auto-encoding, Auto-regressive, and Encoder-Decoder models, as explored by [49]. The **Auto-encoding** architecture is usually applied to encoder-only models (e.g., [11], [21]), allowing bidirectional and masked training. In contrast, the **Auto-regressive** architecture is commonly used by decoder-only models

(e.g., [53], [2]) which generate text exclusively in an undirectional and autoregressive fashion. Lastly, the **Encoder–Decoder** architecture consists of an encoder and a decoder (e.g., [43]). The encoder utilizes self-attention layers to transform the input sequence into latent representations, while the decoder generates the target sequence autoregressively.

As mentioned above, with the increasing size of LLMs, several core capabilities have been discovered. One such capability is **In-Context Learning (ICL)**, as introduced by [6], which enables models to learn from natural language instructions or demonstrations provided in the prompt, without requiring parameter updates. Another capability is **Reasoning**, where models solve complex tasks through methods like Chain-of-Thought (CoT) prompting, as studied by [56], which encourages the generation of intermediate reasoning steps.

2.1.3 Overview of Training Stages

Training can be divided into pretraining and finetuning, as explored in [66].

Pretraining

Pretraining is an essential step that encodes general knowledge from a large-scale corpus into the massive model parameters. For LLM training, two commonly used pretraining objectives are language modeling and denoising auto-encoding, as studied in [28]. This stage equips the model with broad linguistic and semantic knowledge, thereby enhancing its language modeling and generalization abilities.

Compared with small-scale LMs, LLMs have a stronger demand for high-quality data during pretraining. Their general capacity largely depends on the size, diversity, and quality of the pretraining corpus. The pretraining corpus usually contains webpages, books (e.g., [17]), and conversational text (e.g., [45]). For more specialized use cases, it can be extended to multilingual corpora (e.g., [47]), scientific texts, and code datasets collected from programming QA communities (e.g., [61]).

After collecting a large pretraining corpus, it is important to preprocess the data by removing noisy, redundant, or irrelevant content. A typical preprocessing pipeline includes **filtering, selection** and **tokenization**, which segments the raw text into sequences of tokens that serve as basic input units for LLMs.

Instruction Finetuning

Instruction Finetuning is the approach to finetuning pretrained LLMs in a collection of formatted instances that contain instructions and their desired outcome or behavior. These formatted instances are used to finetune LLMs in a supervised learning way to follow instructions, align with human preferences, and acquire specialized capabilities while maintaining basic abilities, acquired by the pretraining objective, as further studied by

[38] and [64]. The formatted instances can be constructed based on preexisting datasets or using LLMs.

2.1.4 Specific Models

Llama 3 [13] and TOWER [2] are text-based LLMs.

Llama 3

Llama 3 is a general purpose model for natural language processing. Therefore, the Llama 3 family of models supports a wide variety of tasks. Llama 3's pretraining and finetuning stages correspond to those described in Section 2.1.3. In terms of pretraining, Llama 3 utilizes the language modeling objective on a multilingual text corpus constructed from formatted web documents. In terms of finetuning, Llama 3 utilizes a cross-entropy loss on data, which is largely comprised of synthetic data and human-annotated prompts.

TOWER

TOWER focuses in contrast to general-purpose models, which demonstrate strong performance across a broad spectrum of tasks and domains, specifically on translation-related tasks. TOWER is trained in three stages. First, the TOWERbase model is created by continuing pretraining on Llama 2 [52] with a large multilingual corpus, thus improving its multilingual capabilities. To further specialize the model on translation tasks, dedicated datasets are used to create TOWERblocks. These datasets allow training on tasks, including Named-Entity Recognition, sentence-level translation, error-span detection, and conversational-based tasks. Finally, through finetuning TOWERbase on TOWERblocks with standard cross-entropy loss, an instruction-following model, TOWERinstruct, is created.

2.2 Speech LLM

LLMs have advanced text-based processing, as explained in section 2.1. When combined with speech encoders, LLMs can be extended to the speech domain, enabling tasks such as Automatic Speech Recognition (ASR), Speech Translation (ST), or speech-based Question Answering (SQA) (e.g., [50]).

However, speech encoders face several unique challenges, as addressed by [21]. Unlike vision, where one instance typically corresponds to a single object, a single speech utterance may contain multiple overlapping sound units. Moreover, unlike NLP, speech pretraining lags a predefined vocabulary of sound units, making predictive losses harder to apply.

Finally, speech does not come with explicit boundaries between sound units, which complicates tasks such as masked prediction pretraining.

2.2.1 Speech Encoders

Speech encoders convert raw audio waveforms into structured feature vectors, providing semantic representations that facilitate various downstream applications.

HuBERT

Hidden Unit Bidirectional Encoder Representations from Transformers (HuBERT) [21] is a speech encoder that addresses the challenges outlined in Section 2.2 by employing a BERT-style masked prediction framework, as studied in [11]. HuBERT leverages the concept of simple discrete latent variable models (e.g., k-means), which categorize input sequences by assigning each data point to a representative cluster center, as shown in [26]. The model is trained on sequences of partially masked speech feature vectors, exclusively predicting the cluster assignments of the masked frames. This forces the model to build high-level representations of the unmasked inputs. This design implicitly encourages the model to learn both acoustic modeling, which captures meaningful latent representations from continuous speech, and language modeling, which models long-range temporal dependencies between these representations.

HuBERT is unsupervised, pretrained for two iterations on either the 960 hours of LibriSpeech audio from [40] or 60,000 hours of Libri-Light audio from [23], both derived from the LibriVox project, which contains English recordings of copyright-free audiobooks read by volunteers. Supervised finetuning of HuBERT is also performed on Libri-Light and LibriSpeech.

Whisper Encoder

Unlike many models that rely solely on an encoder to extract audio features (e.g., [21], [11]), Whisper employs an encoder-decoder architecture, as described in Section 2.1.

Whisper is trained in speech processing tasks, including multilingual speech recognition and translation. A unique aspect of Whisper is that tasks such as determining who spoke when and voice detection are trained together with speech recognition tasks by applying a multitask training format. These tasks are tokenized to be predicted by the decoder without significant standardization. For training, the dataset was constructed based on audio with transcripts gathered from the Internet covering a wide range of environments, recording setups, speakers, and languages. Furthermore, a set of heuristics was applied to the dataset to remove machine-generated transcripts and an audio language detector was used to ensure language consistency. Moreover, the model is trained on audio files that

are divided into 30-second segments, each paired with the corresponding portion of the transcript that falls within the segment.

2.2.2 Connection Speech Encoders and Text LLMs

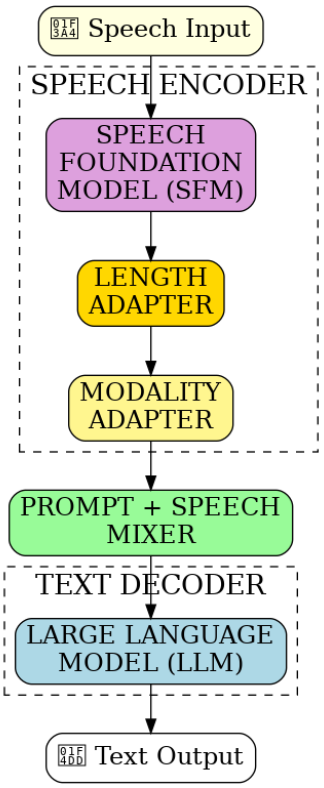

Figure 2.1: The figure illustrates a pipeline for connecting speech encoders with LLMs, as shown in [16]. The pipeline consists of a Speech Foundation Model (SFM), a length adapter, a modality adapter, a prompt speech mixer, and a LLM.

Speech LLMs are all about combining the capabilities of a speech encoder with those of a text LLM. However, the challenge lies in aligning the modules to ensure the encoder can work effectively with the LLM.

Yu et al. (2024) [62] propose three different ways to connect a speech encoder to an LLM. First, by using fully connected layers to compress adjacent feature vectors, serving as a one-dimensional convolution layer. Second, by using a one-dimensional convolution layer and afterwards a multi-headed cross attention layer to project the feature vectors into the embedding space of the LLM. Third, by using a Q-Former [29] that divides the audio into segments, processes them independently, and concatenates the outputs.

Gaido et al. (2024) [16] is a study that reviews different approaches that integrate Speech Foundation Models (SFMs) with LLMs. The basic pipeline is illustrated in Figure 2.1, starting with SFMs, which are typically built on Transformer or Conformer architectures.

SFM's are effective in extracting semantic representations. These representations are projected into the embedding space of the LLM through modality adapters. Since the raw sequence length of speech is much larger than what LLMs can process, a length adapter (e.g., [15], [29]) is used to compress and reduce the number of embeddings. Integration is often enhanced by additional components, such as prompt speech mixers, as used in [7], [60], and [14], which combine speech embeddings with textual prompts. The LLM then processes both modalities jointly and generates a final textual output.

Tung et al. 2024 [41] study ways to align the modalities of the speech encoder and the LLM by finetuning on ASR tasks. The paper reviews combinations of preexisting methods, such as freezing certain modules, utilizing parameter-efficient training strategies like LoRA [22], and applying adapters similar to those in [62]. Revealing that applying LoRA to both modules is the most effective finetuning strategy. In contrast, the strategy of keeping both frozen lags a bit behind in performance. The pretraining and finetuning stages correspond to those described in Section 2.1.3. Typical training and evaluation tasks include ASR (e.g., [55], [63]), ST (e.g., [55]), [63]) and SQA (e.g., [39], [50]).

The compression approaches evaluated in this paper can be viewed as length adapters, following the formulation in the second paper. The training-dependent compression variant, in particular, closely resembles the convolution-like approach introduced in the first paper. Moreover, while the third paper showed that the best performance is achieved when LoRA adapters remain active for both the encoder and the LLM, we keep them frozen in this work in order to isolate the effect of the compression modules.

2.3 LLM and Long Context

Most LLMs are built on Transformer architectures, whose memory and computational requirements grow with sequence length. This presents a fundamental trade-off: while longer input contexts provide models with more potentially useful information, they also increase the difficulty of reasoning over large amounts of content, often leading to a decrease in precision, as elaborated in [32].

The performance of language models on long contexts has been studied by Liu et al. (2024) [32] in multi-document question answering and key-value retrieval tasks. The results revealed that performance often depends on the position of relevant information. Accuracy tends to be highest when relevant content appears at the beginning or end of the input sequence, but performance degrades significantly when critical information is located in the middle. This degradation occurs even in models that exhibit extended context windows, indicating that simply increasing the context length does not guarantee better performance.

Hilgert et al. (2024) [19] have shown that reduced performance in question answering is observed for documents that exceed the original context length, particularly for questions where the relevant information lies beyond the model's context window. Even models with extended context windows struggle with such documents. However, finetuning

can enhance a model's performance on question-answering tasks for scientific papers. When applying finetuning, the performance loss for inputs exceeding the model's original context window is nearly eliminated.

Cheng-Ping Hsieh et al. 2024 [20] have shown that all reviewed models exhibit a significant reduction in question answering performance as sequence length increases. Additionally, it is suggested that increasing the input length based on the size of the context window significantly decreases performance. Moreover, Models that are larger in size tend to perform better overall than smaller models. When Models are trained on larger context sizes, it can lead to better performance, but this is not always the case. In fact, while larger context sizes can sometimes enhance overall performance, inconsistencies arise for longer inputs.

Although only the first study observed a U-shaped performance pattern when processing long contexts, all studies agree that performance degrades once the input sequence exceeds the model's context window. Moreover, all papers indicate that models with extended context windows do not necessarily exhibit better long-context performance.

2.4 Context Compression

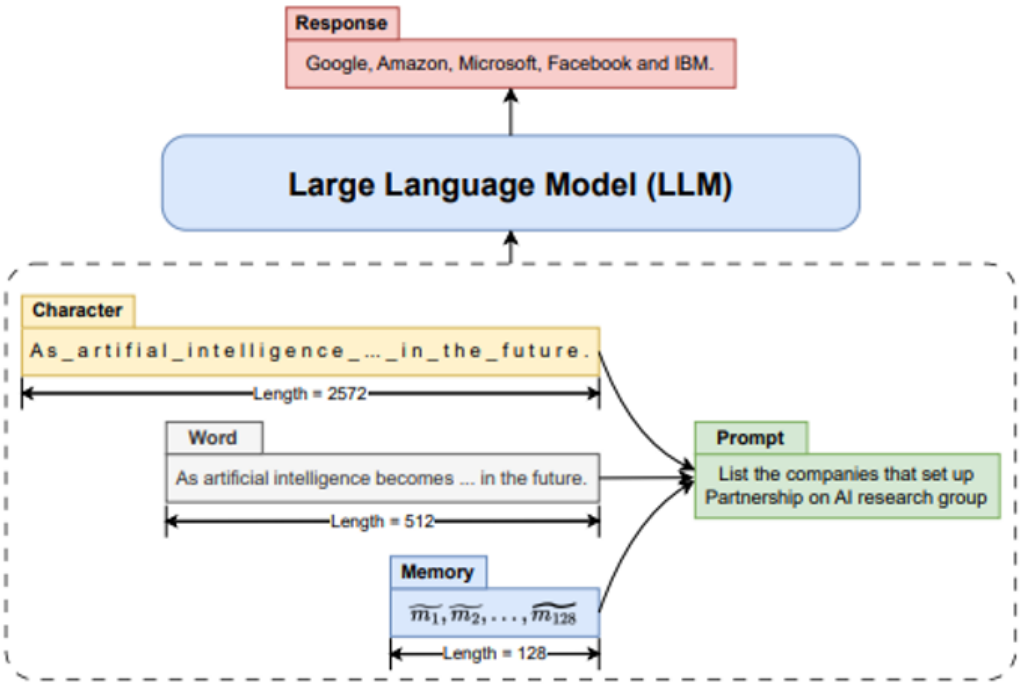

Figure 2.2: The concept of context compression, as illustrated in [18]. In the bottom section of the figure, it is emphasized how the prompt fed to the LLM can have representations of different lengths with M_1 to M_{128} being memory tokens, which lead to the same response

LLMs are constrained by a finite context window and the high computational cost associated with processing long documents. Previous research has attempted to address the limitation of LLMs through innovations in hardware (e.g., improvements in GPUs) or algorithms (e.g., [9], [42]). Although these methods can extend the size of context windows, they may still suffer from a drop in performance when applied to very long contexts, as shown in [32] and [19].

Another approach to reducing computational costs is **context compression**. Context compression is based on the observation that textual information can be represented at different levels of size while still preserving its core meaning, as illustrated in Figure 2.2. In this setting, an LLM can be trained to compress long contexts into shorter, compact memory vectors, which can then be directly used for a variety of downstream tasks.

An approach to context compression is described by Chevalier et al. [8]. In their work, language models are shown to be capable of compressing text into summary vectors that are significantly shorter than the original precompressed text. These summary vectors can be reused as soft prompts, i.e., newly initialized embeddings that are prepended to the input sequence. When long documents are divided into segments $S_1, \ldots, S_n$ and processed sequentially, the summary vectors from the previous segments can be accumulated and concatenated. This concatenation provides the model with a compact representation of the entire document, enabling it to reason over long contexts without exceeding its native context window.

Another approach is presented by Tao Ge et al. [18], who propose the *ICAE framework* for context compression. This framework introduces an encoder-decoder architecture, where the encoder is an LLM that compresses an input context $c = (w_1, w_2, \ldots, w_L)$ into a compact set of memory slots $(m_1, \ldots, m_k)$ with $k \ll L$. The decoder can condition these memory slots for downstream tasks such as autoencoding or text continuation.

All of the references base their approach on the idea of compressing the input into a more compact semantic representation, but they use additional transformers or LLMs that require extensive training. The approaches evaluated in this paper serve the same purpose but are lighter and rely on a more based strategy of slicing the input into compressed memory vectors. Therefore, the need for extensive training is reduced, and consequently, the computational cost is also reduced.

3 Methodology

This chapter introduces various approaches to context compression, with the goal of reducing resource usage while maintaining task performance. Section 3.1 presents compression methods that do not rely on training, while Section 3.2 introduces a training-based approach.

3.1 Inference-Time Compression

Speech, particularly at high sampling rates, generates very dense frame sequences even over short intervals. Because of this density, many segments contain similar or redundant information. This redundancy makes more aggressive, static compression strategies practical. Consequently, there is less need for training-dependent methods to achieve adequate performance.

The following approaches do not rely on trainable compression methods; They apply additional processing steps either directly to the speech waveform or to the corresponding feature vectors. As no training is required, these compression methods are lightweight, computationally inexpensive, and straightforward to implement. Therefore, they serve as a strong baseline for comparison with more advanced compression approaches.

Formal Definitions.

Let a raw speech waveform be denoted as

$$\mathbf{x} = (x_1, x_2, \ldots, x_n), \tag{3.1}$$

Formally, the encoder function can be defined as

$$E : \mathbb{R}^n \rightarrow \mathbb{R}^{m \times d}, \quad E(\mathbf{x}) = \mathbf{w} \tag{3.2}$$

and let the corresponding sequence of feature vectors extracted by the audio encoder be

$$\mathbf{w} = (\mathbf{w_1}, \mathbf{w_2}, \ldots, \mathbf{w_m}), \quad w_i \in \mathbb{R}^d, \tag{3.3}$$

where d is the hidden size of the encoder output and m depends on the encoder architecture .

3.1.1 Simple Chunking

The approach adds an additional preprocessing step by dividing the audio waveform into non-overlapping chunks of fixed length L. These chunks are then independently processed, and the results are concatenated. By segmenting and processing the input independently, the model requires fewer resources compared to processing the whole input in one run.

The simple chunking method was chosen because it is straightforward to implement and does not require additional training or modifications to the model architecture. In addition, chunking is a proven method to reduce computational cost, as many ASR models integrate a chunking-like approach (e.g., [43], [4]).

Chunking

The waveform x is split into consecutive, non-overlapping segments of fixed length L:

$$\mathbf{x} = (\mathbf{c_1}, \mathbf{c_2}, \ldots, \mathbf{c_K}), \quad \mathbf{c_k} = (x_{(k-1)L+1}, \ldots, x_{\min kL,n}), \tag{3.4}$$

where $K = \lceil n/L \rceil$. Each chunk c_k is processed independently by the encoder, and the resulting outputs are concatenated:

$$\mathbf{w} = (\mathbf{E}(\mathbf{c_1}) \parallel \mathbf{E}(\mathbf{c_2}) \parallel \ \ldots \ \parallel \mathbf{E}(\mathbf{c_K})). \tag{3.5}$$

The feature vector sequence obtained by encoding the entire audio without chunking will not be identical to w, since the attention span allows the model to attend to all positions in the given waveform when computing the feature vectors. With chunking, the attention span is restricted to each chunk. Moreover, the positional encodings differ, as each chunk resets its positional indices to zero.

3.1.2 Skipping and Averaging

As speech signals are continuous, sampling them at high frequencies produces sequences with high frame density. Due to the length of these sequences, one often encounters neighboring frames that contain similar information, resulting in redundancy. These redundant frames correspond to feature vectors that are suitable for compression. Since skipping and averaging focus on compressing neighboring frames, they are a compatible choice for this purpose. For implementation, these approaches introduce an intermediate processing step between the encoding and decoding stages. After extracting an array of feature vectors from the audio, the array is compressed along the time dimension by a constant factor x. In doing so, skipping and averaging reflect different degrees of roughness. Skipping simply discards feature vectors, whereas averaging incorporates the values of all vectors by computing their mean. This enables us to assess the impact of compression method complexity on model performance in ASR, particularly in summarization and QA tasks. Unlike in ASR, not every word or sentence in the audio contributes relevant information for

summarization or QA, which makes such compression strategies particularly interesting to investigate.

Overall, in terms of compression, skipping, and averaging offer a straightforward method, as they require no training and are relatively easy to implement.

Skipping

Given the feature sequence w, every z-th frame is retained and the others are discarded:

$$\mathbf{w}' = (\mathbf{w_1}, \mathbf{w_{1+z}}, \mathbf{w_{1+2z}}, \dots), \tag{3.6}$$

leading to a compressed length

$$|\mathbf{w}'| = \left\lceil \frac{m}{z} \right\rceil. \tag{3.7}$$

Averaging

The sequence w is divided into consecutive blocks of size z, and the mean vector is computed for each block:

$$\mathbf{w'_j} = \frac{1}{z} \sum_{i=(j-1)z+1}^{jz} \mathbf{w_i}, \quad j = 1, \dots, \left\lfloor \frac{m}{z} \right\rfloor. \tag{3.8}$$

If a remainder $r = m \bmod z$ exists, the last block is averaged accordingly:

$$\mathbf{w'_{\lfloor m/z \rfloor+1}} = \frac{1}{r} \sum_{i=m-r+1}^{m} \mathbf{w_i}. \tag{3.9}$$

3.2 Trained Compression Module

This approach modifies the model architecture by adding a trainable layer between the encoding and decoding stages. Compression is applied along the time dimension with a constant downsampling rate x on the sequence of audio feature vectors produced by the encoder. In contrast to the inference-time methods, this approach requires additional training to adapt the weights of the convolution layer to the specific environment and task. However, since this step is strongly data- and task-dependent, there is no guarantee of generalization across different datasets or tasks. Training in ASR does not guarantee equally strong performance in summarization or QA tasks. This opens the opportunity to evaluate how trainable compression methods generalize to different domains and tasks.

To illustrate the concept of trainable compression, convolution was chosen, as it represents a fundamental approach to downsampling, commonly used in Conformer models and other architectures (e.g., [1], [21]).

Convolution

The convolution layer with kernel size k and stride s produces a compressed output

$$\mathbf{w}' = (\mathbf{w}'_1, \mathbf{w}'_2, \ldots, \mathbf{w}'_{\mathbf{m}'}), \quad m' = \left\lfloor \frac{m - (k-1) - 1}{s} + 1 \right\rfloor \tag{3.10}$$

Each output vector $\mathbf{w}'_\mathbf{j}$ is computed as a weighted sum over a local window of k consecutive input vectors, with learnable weights $\mathbf{K} \in \mathbb{R}^{k \times d}$:

$$\mathbf{w}'_\mathbf{j} = \sum_{i=0}^{k-1} \mathbf{K_i} \cdot \mathbf{w}_{\mathbf{(j-1)s+i}} \tag{3.11}$$

During training, all parameters of the base model, including the LoRA parameters, are frozen, except for the weights of the newly added convolution layer. This allows for the direct evaluation of the convolution layer's impact on the model's performance across various tasks.

The convolution layer is trained exclusively on ASR data, as ASR datasets have the advantage of being available in large quantities compared to other datasets. Additionally, they are typically sentence-based, making it easier to train the model on them. The setup also enables us to investigate whether models trained on ASR can perform more complex tasks beyond ASR.

4 Experimental setup

This chapter begins with Section 4.1, which presents the tasks and datasets used for training and evaluation. Section 4.2 then discusses the baselines and their respective configurations. In Section 4.3, the compression pipeline, training setup, and characteristics of the compression modules are described. Finally, Section 4.4 introduces the metrics used for evaluation.

4.1 Tasks and Datasets

In this Section, the underlying tasks and datasets are introduced and motivated. Both were used to evaluate the baselines and compression approaches.

4.1.1 Tasks

To evaluate how compression influences the performance of LLMs on downstream capabilities, three types of speech-to-text tasks are introduced.

Automatic Speech Recognition Tasks

Automatic Speech Recognition (ASR) Tasks evaluate a model's ability to accurately convert input audio to text. Since ASR tasks involve transcribing spoken words in a one-to-one mapping, ASR models can exhibit vulnerabilities to information loss, as explored in [12]. This makes ASR a suitable choice to illustrate how increased compression rates relate to poorer ASR performance. ASR also provides the opportunity to evaluate how models perform under challenging conditions, such as audio containing background noise, distortions, or accented speech (e.g., [46]). Lastly, ASR combined with the Word Error Rate (WER) metric is a well-established evaluation setup, as it is applied by many papers to evaluate their models (e.g., [1], [21], [43]).

Speech Summarization Tasks

Summarization tasks assess a model's ability to extract the most representative information from the input audio and synthesize it into a coherent textual summary, as was even explored in the context of speech LLMs [48]. Therefore, such tasks can require a

deeper semantic understanding of audio content, as highlighted by [27]. Thus, it allows the evaluation of the model's semantic capabilities. Furthermore, since summarization itself can be viewed as a form of compression, it is particularly interesting to investigate how context compression on feature vectors influences a model's performance on summarization. Lastly, summarization tasks also provide datasets with long-form audio (e.g., [68]), which align with the intended purpose of compression methods.

Question Answering Tasks

Question Answering (QA) tasks require the model to generate answers to a given question based on the content of the input audio. Similarly to summarization, QA tasks can help models develop better semantic understanding, as explored in [5], making QA a suitable setup to illustrate how compression influences a model's semantic understanding capabilities. Moreover, the fact that QA is already an established task in speech LLMs makes it a suitable choice as an evaluating task (e.g., [36], [1], [50]).

4.1.2 Datasets

Table 4.1: Overview of Nutshell, LibriSQA-PartI and ACL 60/60 datasets, showing the amount of samples, the audio length in minutes, and the average word length per answer for both training and testing / development sets for each task.

	Task	Training set			Testing / Development set		
		amount of samples	average audio (min)	average words per answer	amount of samples	average audio (min)	average words per answer
LibriSQA-PartI	ASR	104,014	0.21	34.57	2,620	0.12	20.07
	Question Answering	104,014	0.21	17 ± 5	2,620	0.12	17 ± 5
Nutshell	Summarization	4,000	12.1 ± 11.2	142.8 ± 36.1	885	9.9 ± 3.6	141.9 ± 36.5
ACL 60/60	ASR	*no training set	*no training set	*no training set	468	0.11	15 ± 7.5

LibriSQA

LibriSQA-PartI [67] contains a total of up to 107,000 audio samples, as illustrated in Table 4.1. For evaluating ASR and QA capabilities, the complete LibriSQA-PartI test set is used, comprising 2,620 authentic human and clean speech samples. On average, each sample has a duration of 7.42 seconds, has an average length of 20.07 words, and is paired with a transcription, a natural question, and a corresponding free-form answer. For training ASR capabilities, the full LibriSQA-PartI training set is utilized, comprising 104,014 samples. On average, each sample has a duration of 12.58 seconds, an average length of 34.57 words, and is paired with a transcription, a natural question, and a corresponding free-form answer. The questions in the LibriSQA dataset have an average length of approximately 16 words, while the answers average around 17 words. The dataset is based on Librispeech [40] and is therefore limited to English.

Nutshell

For evaluating summarization tasks, the complete Nutshell [68] development set is used, which contains 885 audio samples from scientific talks, as illustrated in Table 4.1. The samples have an average duration of 9.9 ± 3.6 minutes. Each audio sample is paired with a corresponding abstract, averaging 141.9 ± 36.5 words. The dataset is built on the ACL Anthology data collection and is limited to English.

ACL 60/60

To evaluate ASR capabilities under challenging conditions, the complete ACL 60/60 [46] development set is used. It consists of sentence-level audio samples recorded under realistic conditions with speakers from diverse demographic backgrounds. The dataset comprises 468 audio samples with an average duration of 6.64 seconds and a length of 15 ± 7.5 words, as illustrated in Table 4.1. Each sample is paired with a corresponding transcription, averaging 16.9 word tokens in length. Unlike LibriSQA and Nutshell, ACL 60/60 supports multiple languages.

4.2 Baselines

To compare the impact of context compression on the performance of Speech LLMs across different tasks, the following baselines are defined.

4.2.1 Text-Based Two-Stage Pipeline

Whisper + Llama Pipeline

For audio-to-text transcription, Whisper [43] is employed as the ASR model. Whisper is a Transformer based on an encoder–decoder architecture, as elaborated in Section 2.1.2. Whisper segments input audio into sentence-level chunks of up to 30 seconds and supports up to 100 languages.

Two baselines are considered: Whisper-small and Whisper-large, differing in model size, with 244M and 1.55B parameters, respectively. For tasks beyond ASR, the generated transcriptions are processed by a text-based LLM. Llama 3, specifically the LLaMA-3.2-3B-Instruct model. Llama 3 is a decoder-only transformer that also supports up to 100 languages.

The Whisper+Llama combination is a suitable choice for this paper, as it represents an established pipeline for Speech LLMs both in the context of speech recognition [44] and speech understanding [30] and has been used in other prior works (e.g., [68]) .

Phi-4-multimodal-instruct self-cascade Pipeline

Phi-4-multimodal-instruct [1] is an open multimodal foundation model, allowing the pipeline to perform automatic speech recognition and a wide range of downstream tasks beyond ASR. The model has 5.6B parameters and supports multiple languages.

In this approach, all audios are first transcribed into English. Downstream tasks, such as summarization and QA, are then performed on the transcribed audio. With Nutshell containing long-form audio, the model cannot process the audio at once, given the limited GPU memory. Thus, the samples are first split into fixed-length segments and then concatenated when transcribed.

Phi-4-multimodal-instruct is a suitable choice for this paper, as it has been trained in speech recognition, QA, and summarization, and represents a unified solution, requiring no separate module for speech encoding.

4.2.2 Discrete-Token-Based Two-Stage Pipeline

Discrete-unit adapted ASR (HuBERT) + LLM (Spire) Pipeline

This pipeline employs an ASR approach based on discrete units. Specifically, the HuBERT-large-ll60k model [21], an encoder-only transformer, is used to extract feature vectors and convert them into discrete tokens. HuBERT primarily supports English. For tasks beyond ASR, Spire [3] is used as the LLM, adapted to process the discrete units generated by HuBERT. Spire has 7B parameters, follows a decoder-only architecture, and also primarily supports English.

As this approach is based on discrete units, a contrast is drawn with text-based approaches, thereby expanding the baseline variety.

4.2.3 End-To-End Pipeline

Phi-4-multimodal-instruct out of the box Pipeline

In this approach, summarization and QA are performed directly on the audio samples, without any interruptions of intermediate transcription or chunking steps.

Phi-4-multimodal-instruct without chunking Pipeline

In this approach, summarization and QA are also performed directly on the audio samples, without any transcription or chunking steps. As the out of the box approach encounters GPU memory shortages when processing Nutshell, the encoding and decoding steps are executed in separate runs. Specifically, the feature vectors extracted by the encoder are

first saved to disk. In a subsequent step, these vectors are reloaded, the encoder parameters are released from GPU memory, and decoding is performed. This baseline also serves as a sanity check, verifying that the results are identical to those obtained with the out of the box approach.

4.3 Model Configuration

The following implementations and their corresponding configurations of the baselines are based on the Hugging Face Transformer package [58] and are configured to produce results solely in English.

4.3.1 ASR (Whisper) + Text LLM (Llama) Pipeline

In the Whisper + Llama pipeline, the following settings are applied to the Whisper modules: timestamps are enabled and sampling is disabled (`do_sample=False`) to ensure deterministic results. The model is assigned to a single GPU (e.g., Nvidia Titan RTX) with 24 GB of memory, and the inference batch size is adapted per dataset: 1 for Nutshell and 20 for both LibriSQA and ACL 60/60. For the Llama module, the torch data type is set to `float16` to reduce GPU memory usage, and a device map distributes computation across four GPUs. During inference, tokenization is configured with `padding=True`. Generation is performed with a maximum of 300 new tokens and sampling disabled (`do_sample=False`). For ACL 60/60 and LibriSQA, two GPUs with a batch size of 30 are sufficient, whereas Nutshell requires four GPUs with a batch size of 5.

4.3.2 Discrete-Unit Adapted ASR (HuBERT) + LLM (Spire) Pipeline

In the HuBERT + Spire pipeline, the HuBERT module is configured with torch data type `float32`. For labeling inputs from LibriSQA and ACL 60/60, the module is assigned to a single GPU with a batch size of 20. In contrast, for Nutshell, one GPU with a batch size of 10 is used. Only Nutshell audio samples are chunked into 20-second segments to prevent GPU memory from overflowing. Chunks shorter than 31.25 ms are discarded. During labeling, `padding` is set to `true`. For Spire, the model uses torch data type `float32`. During inference across all datasets, three GPUs are assigned with a batch size of 10. The tokenizer is configured with `padding=True`. Text generation is performed with `max_new_tokens=300` and `do_sample=False`. Furthermore, `no_repeat_ngram_size=3` is set because Spire appears not to be adapted to accented speech and, therefore, vulnerable to generating repeating sequences when confronted with ACL 60/60. Additionally, regarding Nuthell, the summarization tasks are conducted on the transcriptions of the audio files, as using discrete units directly leads to GPU memory shortages.

4.3.3 Phi-4-Multimodal-Instruct Pipeline

Self-cascade

For the Phi-4-multimodal-instruct self-cascade setup, the processor is configured with `trust_remote_code=True`. The model is initialized with the following settings: attention implementation set to eager, `torch_dtype="auto"`, and `trust_remote_code=True`. For dataset transcription, the model is assigned two GPUs with a batch size of 20. For inference beyond ASR, a batch size of 20 is used for LibriSQA, while Nutshell uses a batch size of 5. For both cases, three GPUs are assigned. For transcription in Nutshell, audio samples are chunked into 30-second segments. During inference, text generation is configured with `max_new_tokens=300`, `do_sample=False`, and the default generation mode.

Out of the box

For the Phi-4 Multimodal-instruct out of the box setup, the processor and model are configured identically to those in the self-cascade setup. For inference beyond ASR, the model is assigned four GPUs, with a batch size of 15 for LibriSQA and a batch size of 1 for Nutshell.

Without chunking

For the Phi-4 multimodal-instruct setup without chunking, the processor and model are configured identically to the self-cascade setup. For ASR and QA, the model is assigned two GPUs with a batch size of 1, whereas summarization requires four GPUs with a batch size of 1. The batch size is reduced to 1 because when executing the model decoder independently, it no longer supports batching. During inference beyond ASR, text generation is configured with `do_sample=False`, `max_new_tokens=300`, and `no_repeat_ngram_size=3`. The processor settings remain the same as in the self-cascade setup.

Training

The training configuration is set as listed in Table 4.2: Exclusively, the convolution layer weights are adjusted during training while the parameters of the original model remain frozen, resulting in 28,31M trainable parameters. Training is conducted 10 epochs with a batch size of 8 per GPU, distributed across four GPUs, on 99% of the LibriSQA training set, which corresponds to 102,973 samples. Gradient checkpointing is enabled (`use_reentrant=False`) to reduce memory usage, and FlashAttention [10] is disabled. The optimizer used is AdamW [34] with $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-7}$, a learning rate of $4 \cdot 10^{-5}$, a weight decay of 0.01, and a maximum gradient norm of 1.0. The learning rate scheduler is linear with 50 warm-up steps. Logging occurs at every step, and checkpoints are saved every 200 steps, with a maximum of one checkpoint retained, storing only the model weights. Evaluation is performed every 200 steps on 1 percent of the LibriSQA training set. The early

Table 4.2: Hyperparameters for training the convolution layer

	Hyper Parameters
Training epochs	10
Batch size	8
Gradient checkpointing	True
Optim	Adamw torch
Adam beta1	0.9
Adam beta2	0.95
Adam epsilon	1e-7
Learning rate	$4 \cdot 10^{-5}$
Weight decay	0.01
Max grad norm	1.0
Lr scheduler type	linear
Warmup steps	50
Save strategy	steps
Save steps	200
Eval strategy	steps
Eval steps	200
Save total limit	1
Save only model	True
bf16	bf16
fp16	fp16
Remove unused columns	False
Disable tqdm	True
Dataloader num workers	4
Ddp find unused parameters	True
Load best model at end	True
Metric for best model	Eval loss
Greater is better	False

stopping patience is set to 2 to prevent the layer from overfitting. When training ends, the best model is selected based on `eval_loss` (lower is better). Training stops after 6.4 epochs due to early stopping kicking in. Mixed precision is enabled with `bf16` or `fp16` as specified. Unused columns in the dataset are retained (`remove_unused_columns=False`). DeepSpeed is not used. Data loading uses 4 worker threads and `ddp_find_unused_parameters=True` to handle any unused layers.

4.3.4 Compression Modules

To ensure proper compression, the processor first calculates the expected size of the compressed feature vector array that will be passed to the encoder for generation. To avoid issues when running the decoder, padding is applied to feature vector arrays with lengths between 4000 and 4096 tokens before they are fed to the decoder.

For the convolution, the following formula is applied, where L_{out} and L_{in} denote the sizes of the output and input, respectively, k is the kernel size, and s is the stride:

$$L_{out} = \left\lfloor \frac{L_{in} - (k-1) - 1}{s} + 1 \right\rfloor \tag{4.1}$$

and a compression ratio of

$$CR_{conv} = \frac{L_{out}}{L_{in}} = \frac{\lfloor (L_{in} - k + 1)/s \rfloor}{L_{in}}. \tag{4.2}$$

The convolution layer has the following parameters: `kernel_size=3`, `stride=2`, `bias=False`, which results in an output length of

$$L_{out} = \left\lfloor \frac{L_{in} - 1}{2} + 1 \right\rfloor \tag{4.3}$$

and a compression rate of

$$CR_{conv} = \frac{L_{out}}{L_{in}} = \frac{\lfloor (L_{in} - 2)/2 \rfloor}{L_{in}}. \tag{4.4}$$

For the skipping approach, the compression formula is:

$$L_{out} = \left\lceil \frac{L_{in}}{x_{te}} \right\rceil \tag{4.5}$$

For the averaging approach, the compression formula is:

$$L_{out} = \left\lfloor \frac{L_{in}}{x_{te}} \right\rfloor + \delta, \quad \delta = \begin{cases} 1 & \text{if } L_{in} \bmod x_{te} \neq 0 \\ 0 & \text{if } L_{in} \bmod x_{te} = 0 \end{cases} \tag{4.6}$$

Both approaches have an approximate compression ratio of

$$CR_{avg} = CR_{skip} = \frac{L_{out}}{L_{in}} \approx \frac{1}{x_{te}}. \tag{4.7}$$

4.4 Evaluation

4.4.1 Quality Metrics

To evaluate a model's performance on ASR tasks, the Word Error Rate (WER) is used. WER measures the similarity between two texts by counting the number of insertions, deletions, or substitutions required to align them. It is a standard ASR metric (e.g, [1], [43]).

For summarization and QA tasks, ROUGE-L [31] and BERTScore [65] are employed. ROUGE-L is suitable because the model may generate multiple valid answers or abstracts that are semantically correct but paraphrase the reference, where WER would be too strict. ROUGE-L evaluates similarity based on the longest common subsequence, capturing syntactic overlap while remaining flexible. To assess semantic similarity, BERTScore is used. BERTScore uses contextual embeddings that consider surrounding words, making it robust to paraphrasing and capable of capturing long-range dependencies in text. For this paper, we use the unnormalized version of BERTScore with RoBERTa-large [33] as the underlying model. Also, we use the F1 score of BERTScore. ROUGE-L and BERTScore are well-established metrics, as they have been used in both QA datasets, such as [67], and summarization datasets, like [68].

4.4.2 Compression Metrics

To quantify compression the compression factor is used, which measures the ratio between the size of the input feature vector array and the size of the output feature vector array. When quantifying the default compression rates of the baselines, the macro average of per-sample ratios between the number of frames when the input utterance is converted into waveform datapoints, and the number of output tokens is taken. For an end-to-end system that does not rely on intermediate discrete units, the macro average of per-sample ratios between the number of frames and the number of feature vectors is taken.

5 Results and Analysis

This chapter first presents the baseline results for ASR, QA, and summarization in Section 5.1. Section 5.2 then discusses the results of the inference-time compression approaches, including the compression rates of the baselines and the impact of increased compression on model performance. Finally, Section 5.3 reviews the results of the trained compression approach, evaluating its performance across different tasks and domains.

5.1 Baseline Results

This section presents the baseline results for ASR, QA, and summarization, as introduced in Section 4.1.

5.1.1 ASR Results

The Whisper-large model outperforms Whisper-small in all domains, with improvements of approximately 2.1 (WER) points in ACL 60/60 and 1.2 points in LibriSQA, as shown in Table 5.1. This gain can be attributed to its larger model size (1.54B parameters vs. 242M).

In contrast, the HuBERT + Spire combination yields the weakest results across all domains, lagging behind by up to 9.02 points in ACL 60/60 and 3.24 points in LibriSQA compared to Whisper-small, as reported in Table 5.1.

The best overall performance is achieved by the Phi-4-multimodal-instruct model, surpassing Whisper-large by approximately 0.8 points (17.47 vs. 18.27) on ACL 60/60 and 1.1 points (2.53 to 3.63) on LibriSQA, as shown in Table 5.1.

5.1.2 QA and Summarization Results

Whisper-large, despite its larger model size, performs nearly identically in summarization and QA across all metrics compared to Whisper-small, with differences within 0.34 ROUGE-L and BERTScore points, as shown in Table 5.2.

The Phi-4-multimodal-instruct self-cascade model slightly outperforms Whisper-large in both tasks, achieving gains of up to 1.08 ROUGE-L points (36.18 vs. 35.10) and 1.42 BERTScore points (90.33 vs. 88.91) in QA. In summarization, it yields an improvement of 0.87 BERTScore points (86.02 vs. 85.15), while the ROUGE-L scores remain almost

Table 5.1: Baseline ASR results in WER (↓) on two datasets of varying conditions: LibriSQA (clean, read speech) and ACL 60/60 (accented speech with varying recording conditions). For readability, the results have been scaled by a factor of 100.

	ASR	
	ACL 60/60	LibriSQA
Whisper-small	20.38	4.90
Whisper-large-v3	18.27	3.63
hubert+spire	29.40	8.14
Phi-4-multimodal-instruct	17.47	2.53

equal, differing by only 0.43 points (19.65 vs. 19.22), as shown in Table 5.2. However, Phi-4-multimodal-instruct self-cascade lags behind HuBERT + Spire in QA, with a gap of up to 13.31 ROUGE-L points (36.18 vs. 49.49); the difference in BERTScore is smaller at about 1.49 points (90.33 vs. 91.82). In contrast, HuBERT + Spire performs considerably worse in summarization, trailing the Phi-4-multimodal-instruct self-cascade model by 4.84 ROUGE-L points (14.81 vs. 19.65) and 2.48 BERTScore points (83.54 vs. 86.02). Due to limited GPU memory, summarization for HuBERT + Spire had to be carried out in a self-cascading setup, where summarization was applied to the transcriptions. As already noted, HuBERT + Spire does not perform strongly in ASR, which likely contributes to its weaker summarization results.

The best overall QA performance is achieved by the out of the box Phi-4-multimodal-instruct model, surpassing HuBERT + Spire by 8.78 ROUGE-L points (58.27 vs. 49.49) and 1.38 BERTScore points (93.20 vs. 91.82), as shown in Table 5.2. The greater performance of both the out of the box version of Phi-4-multimodal-instruct and the HuBERT + Spire pipeline compared to the Whisper + Llama pipeline may be explained by their training data. For Phi-4-multimodal-instruct, it is possible that LibriSQA or a related dataset was included in its training, although this cannot be confirmed. As a matter of fact, HuBERT in the HuBERT + Spire pipeline was trained on LibriSpeech, while LibriSQA is derived from LibriSpeech. This may also be attributed to the fact that SpireLM is the largest model among the baselines, and although it was exclusively trained on ASR and ST, its model size potentially enables improved generalization to QA tasks.

However, the out of the box Phi-4-multimodal-instruct approach encounters an Out Of Memory (OOM) error when processing long-form audio from Nutshell due to the limited GPU memory. The BERTScore for Nutshell is therefore taken from [25]. As [25] only reports BERTScore, we leave the ROUGE-L for this system open. When compared to the version without chunking, the Phi-4-multimodal-instruct out of the box pipeline performs almost identical in summarization, showing a difference of only 0.38 BERTScore points (86.72 vs. 86.34).

The without chunking variant of Phi-4-multimodal-instruct served as a sanity check to verify that separating the feature extraction step works as intended. A comparison of its LibriSQA ASR transcriptions on the first 100 samples with those from the out of

the box approach showed identical outputs. Although the without chunking approach delivers results close to the out of the box variant, it performs worse when generating abstracts in terms of ROUGE-L compared to the self-cascade approach, with a margin of 3.13 ROUGE-L points (16.52 vs. 19.65), as shown in Table 5.2.

Interestingly, both the without chunking and out of the box versions of Phi-4-multimodal-instruct outperform the self-cascade variant across all metrics, except for ROUGE-L in summarization. This may be attributed to the fact that there is no single correct way to write an abstract; this effect becomes even more pronounced as the input audio length increases. Since the BERTScore is higher in the setup without chunking, it indicates that the generated abstracts are closer in meaning to the references, which can be considered a more important signal. Regarding QA, the better performance of the out of the box pipeline compared to the self-cascading variant could be explained by the intermediate transcription step in the pipeline, which may lose or alter information from the short-form audio. Since QA tasks can be very sensitive to the specific information requested, sometimes down to a single word, such alterations can have a noticeable impact on performance.

Table 5.2: Baselines speech-to-text results in ROUGE-L (↑) and BERTScore (↑) on two datasets of varying conditions: LibriSQA (Audio books + questions and answers, short-form audio) and Nutshell (Conference talks + abstracts, long-form audio). For readability, the results have been scaled by a factor of 100.

	Summarization (Nutshell)		QA (LibriSQA)	
	ROUGE-L	BERTScore	ROUGE-L	BERTScore
Whisper-small+llama 3-3B	18.96	85.11	34.76	88.83
Whisper-large-v3+llama 3-3B	19.22	85.15	35.10	88.91
hubert+spire	14.81	83.54	49.49	91.82
Phi-4-multimodal-instruct self-cascade	19.65	86.02	36.18	90.33
Phi-4-multimodal-instruct e2e (out of the box)	–	86.72	58.27	93.20
Phi-4-multimodal-instruct e2e (without chunking)	16.52	86.34	58.27	93.20

5.2 Inference-Time Compression Results

In the following section, we examine both the compression rate inherent to the model and the effect of increasing compression on model performance. Additionally, we present the results of the inference-time compression approaches.

Table 5.3: Inference-Time Compression text-to-text results in ROUGE-L (↑) and BERTScore (↑) on two datasets of varying conditions: LibriSQA (Audio books + questions and answers, short-form audio) and Nutshell (Conference talks + abstracts, long-form audio). For readability, the results have been scaled by a factor of 100.

	Summarization (Nutshell)		QA (LibriSQA)	
	ROUGE-L	BERTScore	ROUGE-L	BERTScore
Phi-4-multimodal-instruct - e2e (without compression)	16.52	86.34	58.27	93.20
Phi-4-multimodal-instruct - e2e (skip 2)	16.27	86.34	50.33	92.06
Phi-4-multimodal-instruct - e2e (avg 2)	16.68	86.36	53.70	92.56
Phi-4-multimodal-instruct - e2e (skip 4)	14.46	85.71	33.85	89.10
Phi-4-multimodal-instruct - e2e (avg 4)	14.47	85.68	35.06	89.19
Phi-4-multimodal-instruct - e2e (skip 8)	13.04	84.75	26.41	87.83
Phi-4-multimodal-instruct - e2e (avg 8)	11.43	82.59	22.17	87.00

5.2.1 Compression Rate Comparison to Baselines

The baselines already yield a certain degree of compression by default. For instance, in ASR on LibriSQA, Phi-4-multimodal-instruct exhibits a compression rate of 5690 when considering the macro average of ratios between the number of audio frames and the number of generated text tokens per sample, whereas Whisper-large achieves a compression rate of 4315 under the same setup. When comparing the compression ratio regarding the macro average of ratios between the number of audio frames and the corresponding feature vectors per sample, the Phi-4 Multimodal-instruct encoder yields a compression rate of 1108, while HuBERT achieves 419. The compression approaches introduced in this paper further extend these rates by multiplying factors of 2, 4, and 8. The corresponding pipelines are summarized in Table 5.3.

5.2.2 Performance Comparison to Baselines

The Phi-4-multimodal-instruct pipeline without compression performs slightly better than the skipping approach: in summarization, the results are almost identical, differing by only 0.25 ROUGE-L points (16.52 vs. 16.27), and regarding BERTScore, they are equal. For QA, the baseline performs better by 7.94 ROUGE-L points (58.27 vs. 50.33) and 1.14 BERTScore points (93.20 vs. 92.06), as shown in Table 5.3. Regarding summarization, averaging and the pipeline without compression perform almost equally well, differing by only 0.16

ROUGE-L points (16.68 vs. 16.52) and 0.02 BERTScore points (86.36 vs. 86.34). For QA, the without compression approach outperforms averaging by 4.57 ROUGE-L points (58.27 vs. 53.70) and by 0.64 BERTScore points (93.20 vs. 92.56). Overall, performance scores in summarization are very close across all metrics. In contrast, for QA, the original pipeline performs noticeably better, particularly in terms of ROUGE-L, compared to the compression approaches. This may be due to the fact that LibriSQA contains short-form audio, making information loss from compression more impactful. Nevertheless, the results indicate that, at least for summarization of long-form audio, compression approaches (averaging with a compression rate of 2) can perform comparably to the no compression baseline. The fact that summarization tasks focus on compressing and removing redundant information, combined with the fact that speech frames often contain large amounts of redundant content, which are intended to be removed by the compression method, may explain why these types of compression approaches are particularly compatible.

The compression approach based on averaging performs slightly better than skipping across all metrics and tasks, as shown in Table 5.3. Specifically, they perform almost identically in summarization, differing by only 0.41 ROUGE-L points (16.68 vs. 16.27) and 0.02 BERTScore points (86.36 vs. 86.34). In QA, averaging improves by 3.37 ROUGE-L points (53.70 vs. 50.33), and they perform in terms of BERTScore almost equally with 0.5 BERTScore points (92.56 vs. 92.06) difference.

5.2.3 Impact of Increasing Compression Rate

The previous analyses exclusively considered compression pipelines with a compression rate of 2. When comparing skipping and averaging approaches at higher compression rates (4 and 8), performance is consistently degrading between tasks and metrics. When comparing rates 2 and 8 for averaging, in summarization performance decreases by 5.25 ROUGE-L points and by 3.77 BERTScore points, as shown in Figure 5.1 and Figure 5.2. For QA, the degradation is more severe, with performance drops by up to 31.53 ROUGE-L points and 5.56 BERTScore points, as shown in Table 5.3, Figure 5.4, and Figure 5.3. When comparing rates 2 and 8 for skipping, in summarization performance decreases by 3.23 ROUGE-L points and by 1.59 BERTScore points, as shown in Figure 5.1 and Figure 5.2. For QA, the degradation is more severe, with performance dropping by up to 23.92 ROUGE-L points and by up to 4.23 BERTScore points, as shown in Table 5.3, Figure 5.4, and Figure 5.3. Interestingly, as the compression rate increases, the performance gap between averaging and skipping narrows. As presented in the previous passage, averaging performs better than skipping at a compression rate of 2. At a compression rate of 8, skipping even surpasses averaging across all tasks and metrics in summarization, with a 1.61-point ROUGE-L advantage (13.04 vs. 11.43) and a 2.16-point BERTScore (84.75 vs. 82.59) advantage, favoring the use of skipping. In QA, by 4.24 ROUGE-L points (26.41 vs. 22.17) and 0.83 BERTScore points (87.82 vs. 87.00), also in favor of skipping. This can be explained by the fact that when compressing too many vectors into one, the resulting vector may deviate significantly from the original vectors, making it possible that simply removing vectors while keeping the rest authentic might have the same or even better effect.

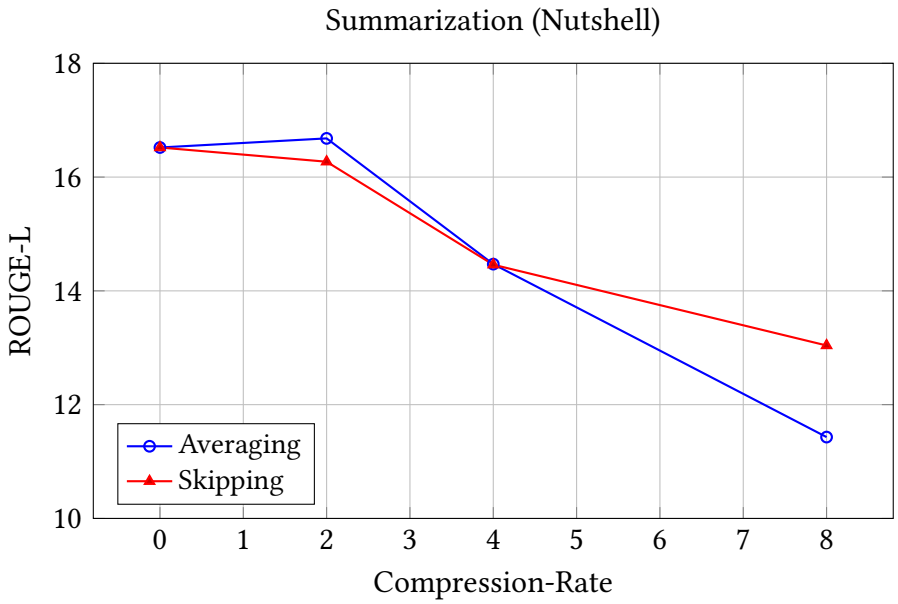

Figure 5.1: Averaging and Skipping approach over compression rates 0, 2, 4, 8 in ROUGE-L (↑) on one dataset: Nutshell (Conference talks + abstracts, long-form audio). For readability, the results have been scaled by a factor of 100.

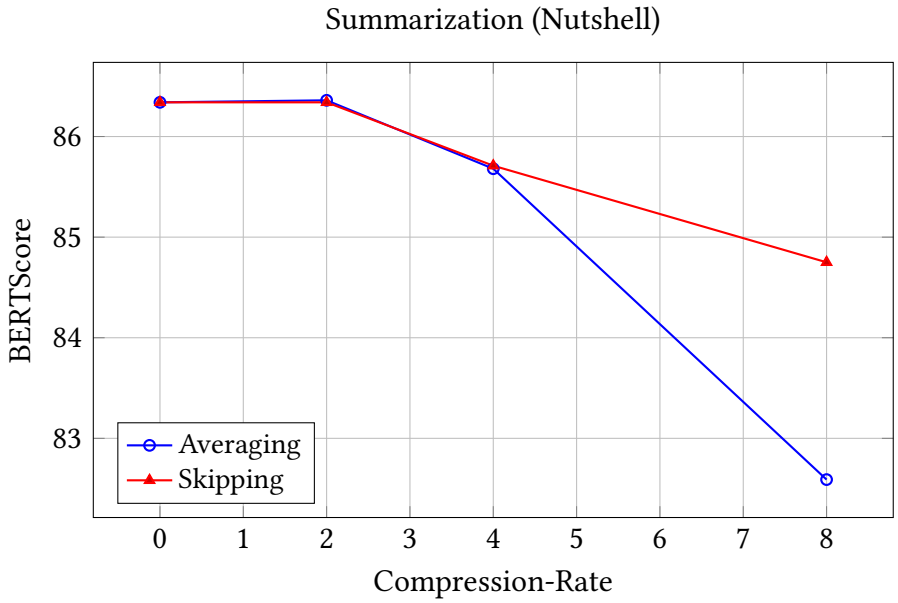

Figure 5.2: Averaging and Skipping approach over compression rates 0, 2, 4, 8 in BERTScore (↑) on one dataset: Nutshell (Conference talks + abstracts, long-form audio). For readability, the results have been scaled by a factor of 100.

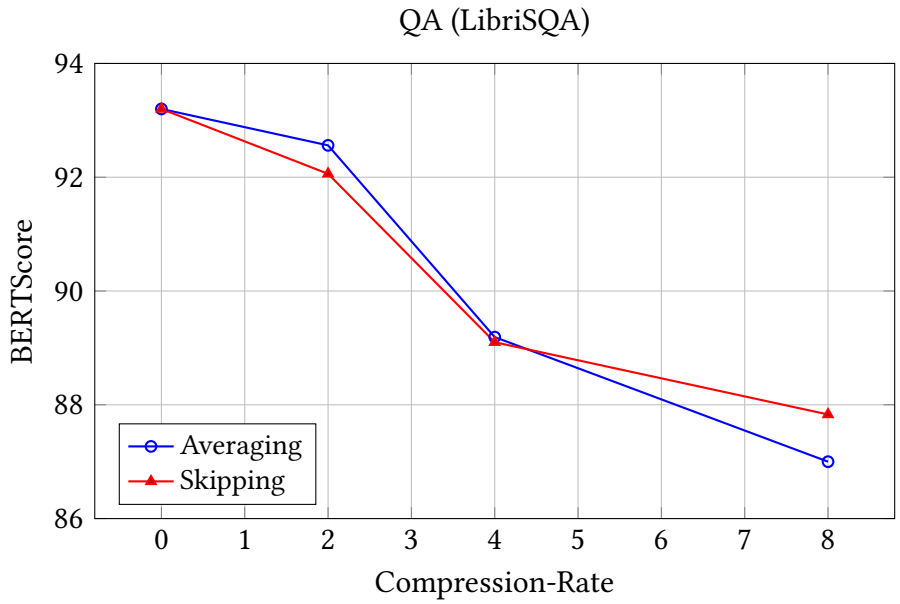

Figure 5.3: Averaging and Skipping approach over compression rates 0, 2, 4, 8 in ROUGE-L (↑) on one dataset: LibriSQA (clean, read speech). For readability, the results have been scaled by a factor of 100.

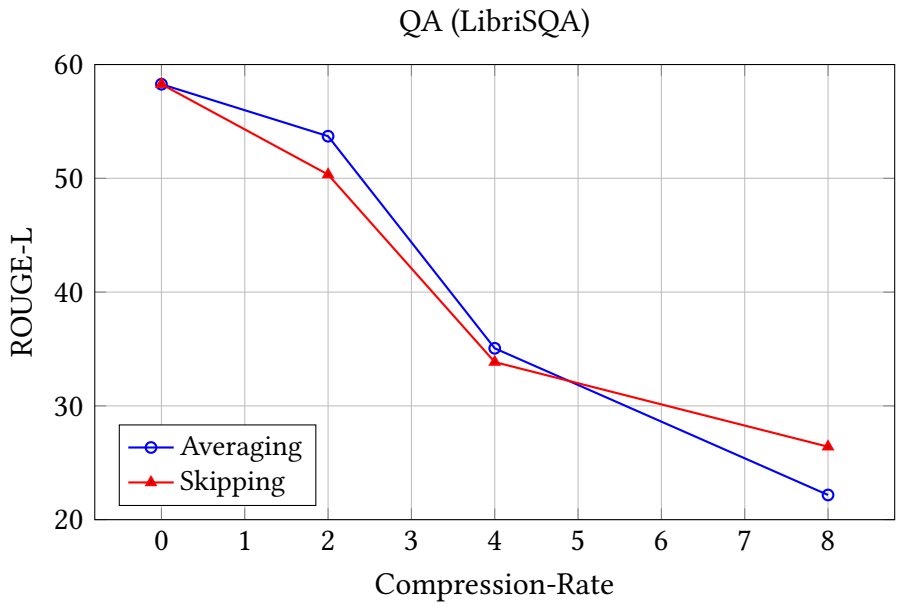

Figure 5.4: Averaging and Skipping approach over compression rates 0, 2, 4, 8 in BERTscore (↑) on one dataset: LibriSQA (clean, read speech). For readability, the results have been scaled by a factor of 100.

The Phi-4 Multimodal-instruct approach without compression performs best in ASR across both datasets. It surpasses the skipping approach (compression rate 2) by 16.59 points in ACL 60/60 (17.47 vs. 34.06) and by 18.87 points in LibriSQA (2.53 vs. 21.4). Compared to the averaging approach (compression rate 2), it performs better with 6.09 points in ACL 60/60 (17.47 vs. 23.56) and 5.07 points in LibriSQA (2.53 vs. 7.6), as shown in Table 5.4. As expected, since compression comes with potential information loss.

With higher compression rates (4 and 8), performance further degrades. Without compression, the model outperforms averaging (compression rate 8) by approximately 75.37 points in ACL 60/60 and by 87.22 points in LibriSQA and outperforms skipping (compression rate 8) by approximately 77.01 points in ACL 60/60 and by 85.05 points in LibriSQA , as shown in Table 5.4, Figure 5.6 and Figure 5.5. The gap between skipping and averaging narrows as the compression rate increases. At a compression rate of 2, averaging is ahead of skipping by 10.5 points (23.56 vs. 34.06) in ACL 60/60 and 13.8 points (7.60 vs. 21.40) in LibriSQA. At a rate of 8, the difference shrinks to 1.54 points (94.48 vs. 92.84) in favor of averaging in ACL 60/60, while skipping slightly outperforms averaging by 2.17 points (87.58 vs. 89.75) in LibriSQA. For ASR overall, skipping and averaging seem to exhibit a similar trend to the summarization and QA tasks.

Table 5.4: Inference-Time Compression ASR results in WER (↓) on two datasets of varying conditions: LibriSQA (clean, read speech) and ACL 60/60 (accented speech with varying recording conditions). For readability, the results have been scaled by a factor of 100.

	ASR	
	ACL 60/60	LibriSQA
Phi-4-multimodal-instruct - e2e (without compression)	17.47	2.53
Phi-4-multimodal-instruct - e2e (skip 2)	34.06	21.40
Phi-4-multimodal-instruct - e2e (avg 2)	23.56	7.60
Phi-4-multimodal-instruct - e2e (skip 4)	78.59	71.54
Phi-4-multimodal-instruct - e2e (avg 4)	71.36	64.32
Phi-4-multimodal-instruct - e2e (skip 8)	94.48	87.58
Phi-4-multimodal-instruct - e2e (avg 8)	92.84	89.75

5.3 Results of Trained Compression

5.3.1 Training and Testing Both on LibriSQA

As expected, the convolution-based compression approach achieved the best performance in ASR tasks with LibriSQA, outperforming the skipping method (compression rate 2) by 19.26 points (2.14 vs. 21.4) and averaging (compression rate 2) by 5.46 points (2.14 vs. 7.60), as shown in Table 5.5. It even slightly surpasses the no compression approach by 0.39

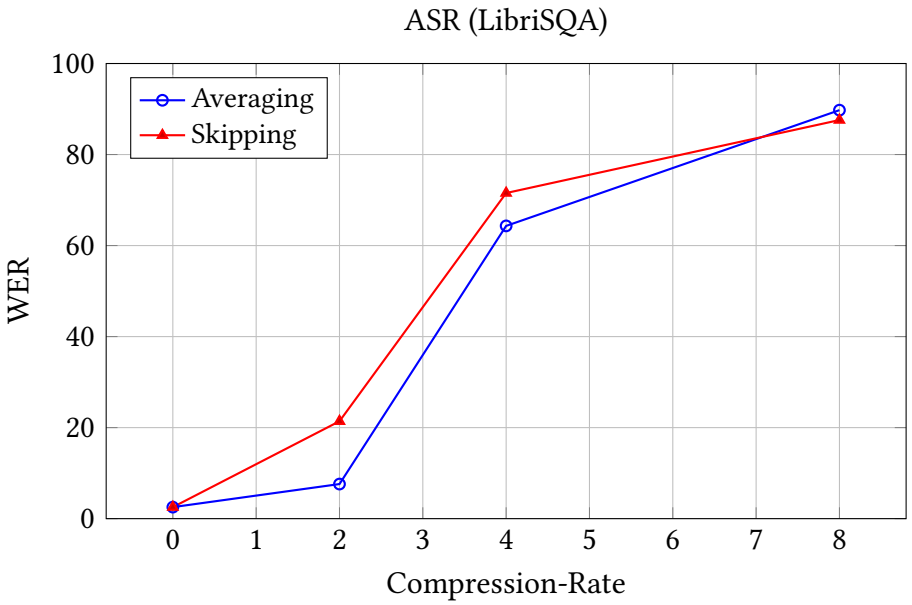

Figure 5.5: Averaging and Skipping approach over compression rates 0, 2, 4, 8 in WER (↓) on one dataset: LibriSQA (clean, read speech). For readability, the results have been scaled by a factor of 100.

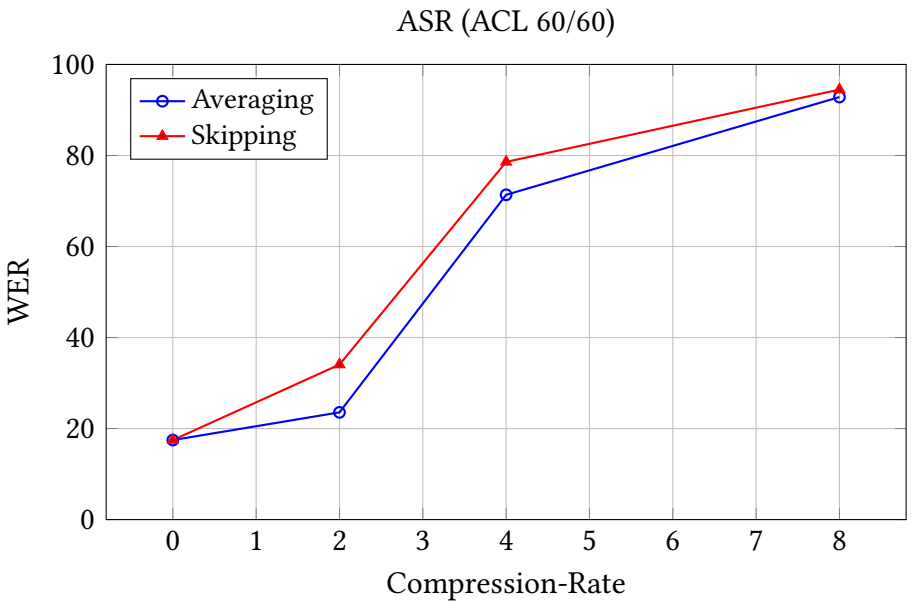

Figure 5.6: Averaging and Skipping approach over compression rates 0, 2, 4, 8 in WER (↓) on one dataset: ACL 60/60 (accented speech with varying recording conditions). For readability, the results have been scaled by a factor of 100.

points (2.14 vs. 2.53). This improvement can be attributed to the fact that the convolution layer was specifically trained on LibriSQA.

Table 5.5: Trained Compression ASR results in WER (↓) on one dataset: LibriSQA (clean, read speech). For readability, the results have been scaled by a factor of 100.

	ASR
	LibriSQA
Phi-4-multimodal-instruct - e2e (without compression)	2.53
Phi-4-multimodal-instruct - e2e (skip 2)	21.40
Phi-4-multimodal-instruct - e2e (avg 2)	7.60
Phi-4-multimodal-instruct - e2e (conv)	2.14

5.3.2 Same Domain, Different Tasks: Training on ASR, Testing on QA

The convolution-based compression approach performs for QA in regard to ROUGE-L slightly better than averaging (compression rate 2) by 1.05 ROUGE-L points (54.75 vs. 53.70) and skipping (compression rate 2) by 4.42 ROUGE-L points (54.75 vs. 50.33), as shown in Table 5.6. However, in terms of BERTScore, all compression methods perform almost equally, with the results of the convolution-based approach differing by 0.36 BERTScore points (92.19 vs. 92.55) from averaging and by 0.13 BERTScore points (92.19 vs. 92.06) from skipping. This indicates that training compression parameters on ASR tasks does not necessarily give an advantage in performance on QA tasks compared to non-trainable compression methods. On the other hand, the no compression approach still slightly outperforms all compression methods, with the convolution approach trailing by 3.52 ROUGE-L points (58.27 vs. 54.75) and the averaging approach by 0.65 BERTScore points (93.20 vs. 92.55).

An aspect to consider is the fact that LibriSQA contains short-form audio. QA is, by nature, very sensitive. Consequently, the answer to a question can depend on one word. Altering this word can lead to a change in context, thereby yielding a completely different answer. When compression is applied to short-form audio, it may induce a greater loss or change in relevant information, resulting in poorer QA performance. Furthermore, this results in the convolution-based approach being marginalized compared to the averaging method.

5.3.3 Same Task, Different Domains: Training on LibriSQA and Testing on ACL 60/60

The convolution-based compression approach achieved better performance in ASR tasks with ACL 60/60 than the training-free compression methods, outperforming averaging (compression rate 2) by 4.31 points (19.25 vs. 23.56) and skipping (compression rate 2) by

Table 5.6: Trained Compression text-to-text results in ROUGE-L (↑) and BERTScore (↑) on one dataset: LibriSQA (Audio books + questions and answers, short-form audio). For readability, the results have been scaled by a factor of 100.

	QA (LibriSQA)	
	ROUGE-L	BERTScore
Phi-4-multimodal-instruct - e2e (without compression)	58.27	93.20
Phi-4-multimodal-instruct - e2e (skip 2)	50.33	92.06
Phi-4-multimodal-instruct - e2e (avg 2)	53.70	92.55
Phi-4-multimodal-instruct - e2e (conv)	54.75	92.19

14.81 points (19.25 vs. 34.06), as shown in Table 5.7. The no compression approach still outperforms all other methods, including convolution, by 1.78 points (17.47 vs. 19.25). However, these results are expected because transcribing corresponds more or less to a one-to-one projection when it comes to words. By applying compression, words may get lost, leading to worse ASR performance. The superior performance of the convolution-based approach among the compression methods may be attributed to the fact that it was trained on ASR tasks.

Table 5.7: Trained Compression ASR results in WER (↓) on one dataset: ACL 60/60 (accented speech with varying recording conditions). For readability, the results have been scaled by a factor of 100.

	ASR
	ACL 60/60
Phi-4-multimodal-instruct - e2e (without compression)	17.47
Phi-4-multimodal-instruct - e2e (skip 2)	34.06
Phi-4-multimodal-instruct - e2e (avg 2)	23.56
Phi-4-multimodal-instruct - e2e (conv)	19.25

5.3.4 Different Tasks, Different Domains: Training on LibriSQA and Testing on Nutshell

The compression approaches and the baseline achieve nearly identical performances in summarization with respect to BERTScore, differing by only a small margin of up to 0.04 BERTScore points, as shown in Table 5.8. In terms of ROUGE-L, both the compression approaches and the baseline deliver almost identical results, differing by a small margin of up to 0.41 ROUGE-L points. Overall, for summarization tasks, the compression approaches deliver almost equal scores across all metrics, including the approach without compression, indicating that the non-trained compression approaches can compete with both the trained

compression methods and the approach without compression. With abstracts more or less describing the overall notion and purpose of a text and QA demanding specific details in the text, it is possible that Summarization is less sensitive to information loss as a result of compression, especially when applied to long-form audio. Compression can also facilitate the model in finding long-range dependencies by placing them numerically closer. In fact, the summarization task and the compression of feature vectors share a similar intention, which potentially translates into improved summarization performance.

Table 5.8: Trained Compression text-to-text results in ROUGE-L (↑) and BERTScore (↑) on one dataset: Nutshell (Conference talks + abstracts, long-form audio). For readability, the results have been scaled by a factor of 100.

	Summarization (Nutshell)	
	ROUGE-L	BERTScore
Phi-4-multimodal-instruct - e2e (without compression)	16.52	86.34
Phi-4-multimodal-instruct - e2e (skip 2)	16.27	86.34
Phi-4-multimodal-instruct - e2e (avg 2)	16.68	86.36
Phi-4-multimodal-instruct - e2e (conv)	16.62	86.38

6 Conclusion

This paper highlights the need for LLMs to consume substantial computational resources and the performance degradation they experience when processing lengthy input. The concept of context compression was introduced as a means to mitigate these challenges. Thus, both training-free and training-dependent compression approaches were introduced to investigate whether less resource-intensive, training-free methods can serve as a valid alternative to training-dependent approaches for compression. These approaches were tested, and the results were analyzed in light of the following research questions.

RQ1: How can we realize compression without additional training?

Compression can be implemented through simple methods such as *skipping*, which retains only every x-th feature vector, or *averaging*, which replaces a group of adjacent vectors with their mean representation. These approaches rely on linear operations without adjustable parameters, eliminating the need for training and thereby reducing computational cost.

RQ2: How do these training-free approaches compare to dedicated compression modules that are trained?

For long-form audio summarization tasks, training-free compression methods approach the baseline competitively, within a margin of 0.25 ROUGE-L points and 0.02 BERTScore points. In addition, the training-dependent convolution-based approach, which is trained on ASR, yields nearly identical results to the training-free approaches, with differences of only up to 0.04 BERTScore points and 0.35 ROUGE-L points. The findings suggest that the convolution approach, when trained in ASR, provides only a marginal advantage over training-free compression methods when applied to summarization tasks on long-form audio. The near-identical performance of the compression methods and the baseline can be attributed to the alignment between the goals of the summarization task and the objectives of the compression modules, both aiming to condense information. The fact that the audio consists of long-form recordings may also account for the minimal performance differences between trainable and non-trainable methods. As the relevant information density decreases with longer audio, the compression can be less fine-grained without significantly impacting the results, allowing for some tolerance to errors.

In QA, averaging, skipping, and the trainable convolution approach also yield results that are competitively close regarding BERTscore, differing by a small margin of up to 0.49 BERTScore points. Compared to the baseline, the compression methods perform slightly worse in terms of BERTScore by a margin of 0.65 to 1.14 BERTScore points. With ROUGE-L, the performance drops become even more pronounced, with the training-free compression

approaches performing worse than the convolution-based approach by up to 4.42 ROUGE-L points, and the convolution-based approach performing worse than the baseline by 3.52 ROUGE-L points. This may be because compression alters the original syntax, and when applied to short-form audio, it can further distort the structure. Combined with the tendency of models to paraphrase, this may lead to differences in syntactic accuracy.

As expected, regarding LibriSQA and ACL 60/60, the training-free approaches perform noticeably worse on ASR tasks by up to 18.87 points compared to the baseline without compression. This could be due to compression-induced information loss and the ASR task's sensitivity to such information loss. The trainable approach outperforms the training-free approach, achieving a performance comparable to the baseline (within 1.78 points). The trainable approach, which was explicitly trained in ASR, may be a reason why the performance drop is mitigated.

When increasing the compression rates for the training-free approaches, performance consistently degrades across summarization, QA, and ASR, with skipping even outperforming averaging at a compression rate of 8.

Nevertheless, the results show that the compression methods can compete with the baseline in both QA and summarization tasks. In particular, the training-free approaches are able to perform comparably to the training-dependent methods except for ASR.

RQ3: How do these two types of approaches generalize to different acoustic conditions and tasks?

When tested on ASR using LibriSQA, the training-dependent convolution-based approach, which was specifically trained on ASR in LibriSQA, performed the best, outperforming both the no-compression baseline and the training-free compression methods, as expected. The training-free compression methods performed the worst, trailing the convolution-based approach by up to 18.96 points, due to their indiscriminate discarding of feature vectors.

When transferred to the same dataset, LibriSQA, but for a different task, namely QA, all compression methods performed worse than the no-compression baseline by up to 1.14 BERTScore points and 7.94 ROUGE-L points. However, the training-free compression methods showed significantly better generalization abilities on QA compared to ASR, approaching the baseline, particularly in terms of BERTScore. The convolution-based approach, although it also showed some performance drop, especially in regard to BERTScore, maintained relatively stable performance.

When applying the same ASR task to a different dataset, ACL 60/60, which contains short-form audio similar to LibriSQA but with accented speech, making the ASR task more challenging, the overall performance of compression methods dropped considerably. The convolution-based approach, trained on LibrisQA ASR, trailed the no-compression baseline by 1.78 points, whereas the training-free methods lagged by up to 16.59 points. Thus, the training-free compression methods demonstrate similar generalization abilities on ACL 60/60 as they did on LibrisQA, while the convolution-based approach shows reduced generalization.

When applying a different task, summarization, to a different dataset, Nutshell, all compression methods exhibited strong generalization abilities, similar to those of the convolution-based approach on LibrisQA with ASR. The performances of all compression approaches and the baseline were almost identical, differing by no more than 0.41 ROUGE-L points and 0.04 BERTScore points, demonstrating the strongest overall generalization capabilities compared to the different setups.

All compression approaches generalized reasonably well to the summarization of long-form audio, often performing as strongly as the baseline. They also demonstrated strong performance in QA, although generalization was not as robust as in summarization.

In contrast, for ASR tasks, performance differences were more pronounced, particularly on ACL 60/60, which contains accented speech, with compression methods demonstrating one of the weakest generalization abilities. On LibriSQA, which contains clean audio, the performance drop was less significant for the convolution-based approach; in fact, the approach even outperformed the baseline slightly. However, for the training-free compression approaches, the results were nearly as poor as with ACL 60/60.

Notably, the experiments on long-form summarization and short-form QA showed that averaging and skipping, despite being training-free compression methods, generalized well to the tasks and performed competitively with the training-based compression approach. Overall, the results suggest that compression methods can achieve performance levels comparable to those of no-compression baselines for QA and summarization, although this effect is less pronounced for ASR.

Limitations

Due to time and computational resource limitations, only a limited number of compression methods could be tested. This study focuses on the most essential methods, including skipping, averaging, and convolution-based compression. Future work should evaluate other compression modules within the same setup, such as Max / Min pooling or windowed-level Q-formers [29].

In addition to different compression methods, other datasets, especially QA datasets with long-form audio, should be considered to draw a more comprehensive comparison with the summarization performance on Nutshell. Furthermore, datasets that allow testing the compression module capabilities beyond ASR, QA, and Summarization should be evaluated.

Bibliography

[1] Abdelrahman Abouelenin et al. “Phi-4-Mini Technical Report: Compact yet Powerful Multimodal Language Models via Mixture-of-LoRAs”. In: *CoRR* abs/2503.01743 (2025). DOI: 10.48550/ARXIV.2503.01743. arXiv: 2503.01743. URL: https://doi.org/10.48550/arXiv.2503.01743.

[2] Duarte M. Alves et al. “Tower: An Open Multilingual Large Language Model for Translation-Related Tasks”. In: *CoRR* abs/2402.17733 (2024). DOI: 10.48550/ARXIV.2402.17733. arXiv: 2402.17733. URL: https://doi.org/10.48550/arXiv.2402.17733.

[3] Kshitij Ambilduke et al. “From TOWER to SPIRE: Adding the Speech Modality to a Text-Only LLM”. In: *CoRR* abs/2503.10620 (2025). DOI: 10.48550/ARXIV.2503.10620. arXiv: 2503.10620. URL: https://doi.org/10.48550/arXiv.2503.10620.

[4] Alexei Baevski et al. “wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations”. In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle et al. 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/92d1e1eb1cd6f9fba3227870bb6d7f07-Abstract.html.

[5] Kinjal Basu et al. “SQuARE: Semantics-based Question Answering and Reasoning Engine”. In: *Proceedings 36th International Conference on Logic Programming (Technical Communications), ICLP Technical Communications 2020, (Technical Communications) UNICAL, Rende (CS), Italy, 18-24th September 2020*. Ed. by Francesco Ricca et al. Vol. 325. EPTCS. 2020, pp. 73–86. DOI: 10.4204/EPTCS.325.13. URL: https://doi.org/10.4204/EPTCS.325.13.

[6] Tom B. Brown et al. “Language Models are Few-Shot Learners”. In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle et al. 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

[7] Zhehuai Chen et al. “SALM: Speech-Augmented Language Model with in-Context Learning for Speech Recognition and Translation”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2024, Seoul, Republic of Korea, April 14-19, 2024*. IEEE, 2024, pp. 13521–13525. DOI: 10.1109/ICASSP48485.2024.10447553. URL: https://doi.org/10.1109/ICASSP48485.2024.10447553.

[8] Alexis Chevalier et al. "Adapting Language Models to Compress Contexts". In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Ed. by Houda Bouamor, Juan Pino, and Kalika Bali. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 3829–3846. DOI: 10.18653/v1/2023.emnlp-main.232. URL: https://aclanthology.org/2023.emnlp-main.232/.

[9] Zihang Dai et al. "Transformer-XL: Attentive Language Models beyond a Fixed-Length Context". In: *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. Ed. by Anna Korhonen, David R. Traum, and Lluís Màrquez. Association for Computational Linguistics, 2019, pp. 2978–2988. DOI: 10.18653/V1/P19-1285. URL: https://doi.org/10.18653/v1/p19-1285.

[10] Tri Dao et al. "FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness". In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*. Ed. by Sanmi Koyejo et al. 2022. URL: https://proceedings.neurips.cc/paper_files/paper/2022/hash/67d57c32e20fd0a7a302cb81d36e40d5-Abstract-Conference.html.

[11] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Association for Computational Linguistics, 2019, pp. 4171–4186. DOI: 10.18653/V1/N19-1423. URL: https://doi.org/10.18653/v1/n19-1423.

[12] Yehoshua Dissen et al. "Enhanced ASR Robustness to Packet Loss with a Front-End Adaptation Network". In: *25th Annual Conference of the International Speech Communication Association, Interspeech 2024, Kos, Greece, September 1-5, 2024*. Ed. by Itshak Lapidot and Sharon Gannot. ISCA, 2024. DOI: 10.21437/INTERSPEECH.2024-306. URL: https://doi.org/10.21437/Interspeech.2024-306.

[13] Abhimanyu Dubey et al. "The Llama 3 Herd of Models". In: *CoRR* abs/2407.21783 (2024). DOI: 10.48550/ARXIV.2407.21783. arXiv: 2407.21783. URL: https://doi.org/10.48550/arXiv.2407.21783.

[14] Yassir Fathullah et al. "AudioChatLlama: Towards General-Purpose Speech Abilities for LLMs". In: *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. Ed. by Kevin Duh, Helena Gomez, and Steven Bethard. Mexico City, Mexico: Association for Computational Linguistics, June 2024, pp. 5522–5532. DOI: 10.18653/v1/2024.naacl-long.309. URL: https://aclanthology.org/2024.naacl-long.309/.

[15] Marco Gaido et al. "CTC-based Compression for Direct Speech Translation". In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*. Ed. by

Paola Merlo, Jörg Tiedemann, and Reut Tsarfaty. Association for Computational Linguistics, 2021, pp. 690–696. DOI: 10.18653/V1/2021.EACL-MAIN.57. URL: https://doi.org/10.18653/v1/2021.eacl-main.57.

[16] Marco Gaido et al. "Speech Translation with Speech Foundation Models and Large Language Models: What is There and What is Missing?" In: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Lun-Wei Ku, Andre Martins, and Vivek Srikumar. Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 14760–14778. DOI: 10.18653/v1/2024.acl-long.789. URL: https://aclanthology.org/2024.acl-long.789/.

[17] Leo Gao et al. "The Pile: An 800GB Dataset of Diverse Text for Language Modeling". In: *CoRR* abs/2101.00027 (2021). arXiv: 2101.00027. URL: https://arxiv.org/abs/2101.00027.

[18] Tao Ge et al. "In-context Autoencoder for Context Compression in a Large Language Model". In: *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL: https://openreview.net/forum?id=uREj4ZuGJE.

[19] Lukas Hilgert, Danni Liu, and Jan Niehues. "Evaluating and Training Long-Context Large Language Models for Question Answering on Scientific Papers". In: *Proceedings of the 1st Workshop on Customizable NLP: Progress and Challenges in Customizing NLP for a Domain, Application, Group, or Individual (CustomNLP4U)*. Ed. by Sachin Kumar et al. Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 220–236. DOI: 10.18653/v1/2024.customnlp4u-1.17. URL: https://aclanthology.org/2024.customnlp4u-1.17/.

[20] Cheng-Ping Hsieh et al. "RULER: What's the Real Context Size of Your Long-Context Language Models?" In: *CoRR* abs/2404.06654 (2024). DOI: 10.48550/ARXIV.2404.06654. arXiv: 2404.06654. URL: https://doi.org/10.48550/arXiv.2404.06654.

[21] Wei-Ning Hsu et al. "HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units". In: *IEEE ACM Trans. Audio Speech Lang. Process.* 29 (2021), pp. 3451–3460. DOI: 10.1109/TASLP.2021.3122291. URL: https://doi.org/10.1109/TASLP.2021.3122291.

[22] Edward J. Hu et al. "LoRA: Low-Rank Adaptation of Large Language Models". In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL: https://openreview.net/forum?id=nZeVKeeFYf9.

[23] Jacob Kahn et al. "Libri-Light: A Benchmark for ASR with Limited or No Supervision". In: *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020*. IEEE, 2020, pp. 7669–7673. DOI: 10.1109/ICASSP40776.2020.9052942. URL: https://doi.org/10.1109/ICASSP40776.2020.9052942.

[24] Jared Kaplan et al. "Scaling Laws for Neural Language Models". In: *CoRR* abs/2001.08361 (2020). arXiv: 2001.08361. URL: https://arxiv.org/abs/2001.08361.

[25] Sai Koneru et al. "KIT's Offline Speech Translation and Instruction Following Submission for IWSLT 2025". In: *Proceedings of the 22nd International Conference on Spoken Language Translation (IWSLT 2025)*. Ed. by Elizabeth Salesky, Marcello Federico, and Antonis Anastasopoulos. Vienna, Austria (in-person and online): Association for Computational Linguistics, July 2025, pp. 232–244. ISBN: 979-8-89176-272-5. DOI: 10.18653/v1/2025.iwslt-1.22. URL: https://aclanthology.org/2025.iwslt-1.22/.

[26] Chia-ying Lee and James R. Glass. "A Nonparametric Bayesian Approach to Acoustic Model Discovery". In: *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 1: Long Papers*. The Association for Computer Linguistics, 2012, pp. 40–49. URL: https://aclanthology.org/P12-1005/.

[27] Hyunjae Lee et al. "Enhancing Semantic Understanding with Self-Supervised Methods for Abstractive Dialogue Summarization". In: *22nd Annual Conference of the International Speech Communication Association, Interspeech 2021, Brno, Czechia, August 30 - September 3, 2021*. Ed. by Hynek Hermansky et al. ISCA, 2021, pp. 796–800. DOI: 10.21437/INTERSPEECH.2021-1270. URL: https://doi.org/10.21437/Interspeech.2021-1270.

[28] Woong-Hee Lee et al. "Noise Learning-Based Denoising Autoencoder". In: *IEEE Commun. Lett.* 25.9 (2021), pp. 2983–2987. DOI: 10.1109/LCOMM.2021.3091800. URL: https://doi.org/10.1109/LCOMM.2021.3091800.

[29] Junnan Li et al. "BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models". In: *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*. Ed. by Andreas Krause et al. Vol. 202. Proceedings of Machine Learning Research. PMLR, 2023, pp. 19730–19742. URL: https://proceedings.mlr.press/v202/li23q.html.

[30] Mohan Li et al. "WHISMA: A Speech-LLM to Perform Zero-Shot Spoken Language Understanding". In: *IEEE Spoken Language Technology Workshop, SLT 2024, Macao, December 2-5, 2024*. IEEE, 2024, pp. 1115–1122. DOI: 10.1109/SLT61566.2024.10832156. URL: https://doi.org/10.1109/SLT61566.2024.10832156.

[31] Chin-Yew Lin. "ROUGE: A Package for Automatic Evaluation of Summaries". In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 74–81. URL: https://aclanthology.org/W04-1013/.

[32] Nelson F. Liu et al. "Lost in the Middle: How Language Models Use Long Contexts". In: *Transactions of the Association for Computational Linguistics* 12 (2024), pp. 157–173. DOI: 10.1162/tacl_a_00638. URL: https://aclanthology.org/2024.tacl-1.9/.

[33] Yinhan Liu et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *CoRR* abs/1907.11692 (2019). arXiv: 1907.11692. URL: http://arxiv.org/abs/1907.11692.

[34] Ilya Loshchilov and Frank Hutter. “Decoupled Weight Decay Regularization”. In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=Bkg6RiCqY7.

[35] Nicolo Micheletti et al. “Exploration of Masked and Causal Language Modelling for Text Generation”. In: *CoRR* abs/2405.12630 (2024). DOI: 10.48550/ARXIV.2405.12630. arXiv: 2405.12630. URL: https://doi.org/10.48550/arXiv.2405.12630.

[36] Eliya Nachmani et al. “Spoken Question Answering and Speech Continuation Using Spectrogram-Powered LLM”. In: *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL: https://openreview.net/forum?id=izrOLJov5y.

[37] OpenAI. “GPT-4 Technical Report”. In: *CoRR* abs/2303.08774 (2023). DOI: 10.48550/ARXIV.2303.08774. arXiv: 2303.08774. URL: https://doi.org/10.48550/arXiv.2303.08774.

[38] Long Ouyang et al. “Training language models to follow instructions with human feedback”. In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*. Ed. by Sanmi Koyejo et al. 2022. URL: https://proceedings.neurips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.

[39] Jing Pan et al. “COSMIC: Data Efficient Instruction-tuning For Speech In-Context Learning”. In: *25th Annual Conference of the International Speech Communication Association, Interspeech 2024, Kos, Greece, September 1-5, 2024*. Ed. by Itshak Lapidot and Sharon Gannot. ISCA, 2024. DOI: 10.21437/INTERSPEECH.2024-1346. URL: https://doi.org/10.21437/Interspeech.2024-1346.

[40] Vassil Panayotov et al. “Librispeech: An ASR corpus based on public domain audio books”. In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2015, South Brisbane, Queensland, Australia, April 19-24, 2015*. IEEE, 2015, pp. 5206–5210. DOI: 10.1109/ICASSP.2015.7178964. URL: https://doi.org/10.1109/ICASSP.2015.7178964.

[41] Van Tung Pham et al. “A Comprehensive Solution to Connect Speech Encoder and Large Language Model for ASR”. In: *CoRR* abs/2406.17272 (2024). DOI: 10.48550/ARXIV.2406.17272. arXiv: 2406.17272. URL: https://doi.org/10.48550/arXiv.2406.17272.

[42] Michael Poli et al. “Hyena Hierarchy: Towards Larger Convolutional Language Models”. In: *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*. Ed. by Andreas Krause et al. Vol. 202. Proceedings of Machine Learning Research. PMLR, 2023, pp. 28043–28078. URL: https://proceedings.mlr.press/v202/poli23a.html.

[43] Alec Radford et al. "Robust Speech Recognition via Large-Scale Weak Supervision". In: *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*. Ed. by Andreas Krause et al. Vol. 202. Proceedings of Machine Learning Research. PMLR, 2023, pp. 28492–28518. URL: https://proceedings.mlr.press/v202/radford23a.html.

[44] Srijith Radhakrishnan et al. "Whispering LLaMA: A Cross-Modal Generative Error Correction Framework for Speech Recognition". In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*. Ed. by Houda Bouamor, Juan Pino, and Kalika Bali. Association for Computational Linguistics, 2023, pp. 10007–10016. DOI: 10.18653/V1/2023.EMNLP-MAIN.618. URL: https://doi.org/10.18653/v1/2023.emnlp-main.618.

[45] Stephen Roller et al. "Recipes for Building an Open-Domain Chatbot". In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*. Ed. by Paola Merlo, Jörg Tiedemann, and Reut Tsarfaty. Association for Computational Linguistics, 2021, pp. 300–325. DOI: 10.18653/V1/2021.EACL-MAIN.24. URL: https://doi.org/10.18653/v1/2021.eacl-main.24.

[46] Elizabeth Salesky et al. "Evaluating Multilingual Speech Translation under Realistic Conditions with Resegmentation and Terminology". In: *Proceedings of the 20th International Conference on Spoken Language Translation, IWSLT@ACL 2023, Toronto, Canada (in-person and online), 13-14 July, 2023*. Ed. by Elizabeth Salesky, Marcello Federico, and Marine Carpuat. Association for Computational Linguistics, 2023, pp. 62–78. DOI: 10.18653/V1/2023.IWSLT-1.2. URL: https://doi.org/10.18653/v1/2023.iwslt-1.2.

[47] Teven Le Scao et al. "BLOOM: A 176B-Parameter Open-Access Multilingual Language Model". In: *CoRR* abs/2211.05100 (2022). DOI: 10.48550/ARXIV.2211.05100. arXiv: 2211.05100. URL: https://doi.org/10.48550/arXiv.2211.05100.

[48] Hengchao Shang et al. "An End-to-End Speech Summarization Using Large Language Model". In: *25th Annual Conference of the International Speech Communication Association, Interspeech 2024, Kos, Greece, September 1-5, 2024*. Ed. by Itshak Lapidot and Sharon Gannot. ISCA, 2024. DOI: 10.21437/INTERSPEECH.2024-1428. URL: https://doi.org/10.21437/Interspeech.2024-1428.

[49] Minghao Shao et al. "Survey of Different Large Language Model Architectures: Trends, Benchmarks, and Challenges". In: *IEEE Access* 12 (2024), pp. 188664–188706. DOI: 10.1109/ACCESS.2024.3482107. URL: https://doi.org/10.1109/ACCESS.2024.3482107.

[50] Changli Tang et al. "SALMONN: Towards Generic Hearing Abilities for Large Language Models". In: *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL: https://openreview.net/forum?id=14rn7HpKVk.

[51] Yi Tay et al. “Long Range Arena : A Benchmark for Efficient Transformers”. In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL: https://openreview.net/forum?id=qVyeW-grC2k.

[52] Hugo Touvron et al. “Llama 2: Open Foundation and Fine-Tuned Chat Models”. In: *CoRR* abs/2307.09288 (2023). DOI: 10.48550/ARXIV.2307.09288. arXiv: 2307.09288. URL: https://doi.org/10.48550/arXiv.2307.09288.

[53] Hugo Touvron et al. “LLaMA: Open and Efficient Foundation Language Models”. In: *CoRR* abs/2302.13971 (2023). DOI: 10.48550/ARXIV.2302.13971. arXiv: 2302.13971. URL: https://doi.org/10.48550/arXiv.2302.13971.

[54] Ashish Vaswani et al. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon et al. 2017, pp. 5998–6008. URL: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[55] Mingqiu Wang et al. “SLM: Bridge the Thin Gap Between Speech and Text Foundation Models”. In: *IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2023, Taipei, Taiwan, December 16-20, 2023*. IEEE, 2023, pp. 1–8. DOI: 10.1109/ASRU57964.2023.10389703. URL: https://doi.org/10.1109/ASRU57964.2023.10389703.

[56] Jason Wei et al. “Chain-of-Thought Prompting Elicits Reasoning in Large Language Models”. In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*. Ed. by Sanmi Koyejo et al. 2022. URL: https://proceedings.neurips.cc/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

[57] Jason Wei et al. “Emergent Abilities of Large Language Models”. In: *Trans. Mach. Learn. Res.* 2022 (2022). URL: https://openreview.net/forum?id=yzkSU5zdwD.

[58] Thomas Wolf et al. “Transformers: State-of-the-Art Natural Language Processing”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020*. Ed. by Qun Liu and David Schlangen. Association for Computational Linguistics, 2020, pp. 38–45. DOI: 10.18653/V1/2020.EMNLP-DEMOS.6. URL: https://doi.org/10.18653/v1/2020.emnlp-demos.6.

[59] Chenfei Wu et al. “Qwen-Image Technical Report”. In: *CoRR* abs/2508.02324 (2025). DOI: 10.48550/ARXIV.2508.02324. arXiv: 2508.02324. URL: https://doi.org/10.48550/arXiv.2508.02324.

[60] Jian Wu et al. “On Decoder-Only Architecture For Speech-to-Text and Large Language Model Integration”. In: *IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2023, Taipei, Taiwan, December 16-20, 2023*. IEEE, 2023, pp. 1–8. DOI: 10.1109/ASRU57964.2023.10389705. URL: https://doi.org/10.1109/ASRU57964.2023.10389705.

[61] Frank F. Xu et al. "A systematic evaluation of large language models of code". In: *MAPS@PLDI 2022: 6th ACM SIGPLAN International Symposium on Machine Programming, San Diego, CA, USA, 13 June 2022*. Ed. by Swarat Chaudhuri and Charles Sutton. ACM, 2022, pp. 1–10. DOI: 10.1145/3520312.3534862. URL: https://doi.org/10.1145/3520312.3534862.

[62] Wenyi Yu et al. "Connecting Speech Encoder and Large Language Model for ASR". In: *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2024, Seoul, Republic of Korea, April 14-19, 2024*. IEEE, 2024, pp. 12637–12641. DOI: 10.1109/ICASSP48485.2024.10445874. URL: https://doi.org/10.1109/ICASSP48485.2024.10445874.

[63] Hao Zhang et al. "Tuning Large language model for End-to-end Speech Translation". In: *CoRR* abs/2310.02050 (2023). DOI: 10.48550/ARXIV.2310.02050. arXiv: 2310.02050. URL: https://doi.org/10.48550/arXiv.2310.02050.

[64] Shengyu Zhang et al. "Instruction Tuning for Large Language Models: A Survey". In: *CoRR* abs/2308.10792 (2023). DOI: 10.48550/ARXIV.2308.10792. arXiv: 2308.10792. URL: https://doi.org/10.48550/arXiv.2308.10792.

[65] Tianyi Zhang et al. "BERTScore: Evaluating Text Generation with BERT". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=SkeHuCVFDr.

[66] Wayne Xin Zhao et al. "A Survey of Large Language Models". In: *CoRR* abs/2303.18223 (2023). DOI: 10.48550/ARXIV.2303.18223. arXiv: 2303.18223. URL: https://doi.org/10.48550/arXiv.2303.18223.

[67] Zihan Zhao et al. "LibriSQA: Advancing Free-form and Open-ended Spoken Question Answering with a Novel Dataset and Framework". In: *CoRR* abs/2308.10390 (2023). DOI: 10.48550/ARXIV.2308.10390. arXiv: 2308.10390. URL: https://doi.org/10.48550/arXiv.2308.10390.

[68] Maike Züfle et al. "NUTSHELL: A Dataset for Abstract Generation from Scientific Talks". In: *Proceedings of the 22nd International Conference on Spoken Language Translation (IWSLT 2025)*. Ed. by Elizabeth Salesky, Marcello Federico, and Antonis Anastasopoulos. Vienna, Austria (in-person and online): Association for Computational Linguistics, July 2025, pp. 19–32. ISBN: 979-8-89176-272-5. DOI: 10.18653/v1/2025.iwslt-1.2. URL: https://aclanthology.org/2025.iwslt-1.2/.