

# SlimLM: An Efficient Small Language Model for On-Device Document Assistance

Thang M. Pham<sup>†</sup>  
thangpham@auburn.edu

Phat T. Nguyen<sup>‡</sup>  
pnguyen340@gatech.edu

Seunghyun Yoon<sup>§</sup>  
syoon@adobe.com

Viet Dac Lai<sup>§</sup>  
daclai@adobe.com

Franck Dernoncourt<sup>§</sup>  
franck.dernoncourt@gmail.com

Trung Bui<sup>§</sup>  
bui@adobe.com

<sup>†</sup>Auburn University

<sup>‡</sup>Georgia Tech

<sup>§</sup>Adobe Research

## Abstract

While small language models (SLMs) show promises for mobile deployment, their real-world performance and applications on smartphones remains underexplored. We present SlimLM, a series of SLMs optimized for document assistance tasks on mobile devices. Through extensive experiments on a Samsung Galaxy S24, we identify the optimal trade-offs between model size (ranging from 125M to 7B parameters), context length, and inference time for efficient on-device processing. SlimLM is pre-trained on SlimPajama-627B and fine-tuned on DocAssist, our constructed dataset for summarization, question answering and suggestion tasks. Our smallest model demonstrates efficient performance on S24, while larger variants offer enhanced capabilities within mobile constraints. We evaluate SlimLM against existing SLMs, showing comparable or superior performance and offering a benchmark for future research in on-device language models. We also provide an Android application, offering practical insights into SLM deployment. Our findings provide valuable insights and illuminate the capabilities of running advanced language models on high-end smartphones, potentially reducing server costs and enhancing privacy through on-device processing.

## 1 Introduction

The evolution of language models is diverging along two paths: large language models (LLMs) pushing the boundaries of artificial general intelligence in data centers (Chowdhery et al., 2022; OpenAI, 2023a; Team et al., 2023; Touvron et al., 2023a,b; Alibaba, 2023.11, 2024.09), and small language models (SLMs) designed for resource-efficient deployment on edge devices like smartphones (Meituan, 2023.12; MBZUAI, 2024.02; Zhang et al., 2024; Liu et al., 2024). While LLMs have attracted significant attention, the practical implementation and performance of SLMs on real

mobile devices remain understudied, despite their growing importance in consumer technology.

Recent developments, such as Qwen-2 (Alibaba, 2024.06), SmoLM (HuggingFace, 2024.07), Gemini Nano (Reid et al., 2024), Apple Intelligence (Apple, 2024.09) or LLaMA-3.2 (Meta, 2024.09) underscore the increasing relevance of SLMs in mobile applications. However, a comprehensive understanding of how these models perform on high-end smartphones is lacking. Unlike previous works that primarily focus on developing smaller models without extensive real-device testing (Meituan, 2023.12; MBZUAI, 2024.02; Zhang et al., 2024; Liu et al., 2024), our approach aims to bridge that gap by presenting an in-depth study of SLM development and deployment on a Samsung Galaxy S24 (also known as S24), focusing on three document assistance tasks: summarization (SUMM), question suggestion (QS), and question answering (QA). By enabling efficient on-device document processing, our approach has the potential to significantly reduce server costs associated with API calls to cloud-based services, while enhancing user privacy.

We address critical questions about optimal model size, maximum context length, inference latency, memory constraints, and performance trade-offs on mobile devices. To answer these questions, we introduce SlimLM, a series of small language models specifically designed and optimized for mobile deployment. SlimLM is pretrained on the SlimPajama-627B (Soboleva et al., 2023) and fine-tuned on DocAssist, our specialized dataset constructed based on ~83K documents for document assistance. Our models range from 125M to 1B parameters, allowing us to explore the full spectrum of what is possible on current mobile hardware.

Our results show that SlimLM models perform comparably or even better than existing SLMs of similar sizes across standard metrics such as BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), Semantic Textual Similarity (STS), Self-BLEU (Zhu

et al., 2018) for text diversity and GEval (Liu et al., 2023). The smallest model SlimLM-125M demonstrates efficient performance on S24, making it suitable for widespread deployment. Larger variants, up to 1B parameters, offer enhanced capabilities while still operating within mobile constraints. To demonstrate real-world applicability, we develop an Android application showcasing SlimLM’s document assistance capabilities<sup>1</sup> (Sec. 4).

Our key contributions are:

1. We identify the sweet spot between model size, inference time, as well as the longest context length that can be efficiently processed on the latest Samsung device S24 (Sec. 2.1).
2. We construct DocAssist, a specialized dataset for finetuning models on three critical document assistance tasks (Sec. 2.2).
3. We propose a set of small language models pretrained on SlimPajama with 627B tokens and finetuned on the DocAssist dataset (Sec. 2.3).
4. SlimLM outperforms or performs comparably to existing SLMs of similar sizes while handling a maximum of 800 context tokens (Sec. 3).

## 2 Approach

To develop and deploy an efficient model for document assistance tasks on mobile devices, we propose a 3-step approach: (1) Determine an ideal model size that can handle sufficiently long context inputs in reasonable time; (2) Construct a dataset for instruction-finetuning models to enhance their document assistance capabilities; and (3) Train and fine-tune SlimLM, a series of models from scratch to perform document assistance tasks while running efficiently on mobile devices.

### 2.1 Sweet Spot: Model Size, Context Length and Inference Time

Finding the sweet spot between model size, context length and inference time is important because larger models may take much time to handle and memory for being loaded so it cannot handle long context despite higher performance. Similarly, smaller models can handle longer contexts in a shorter time but it remains unknown how much their performance degrades.

<sup>1</sup>Code and data will be released at [anonymous.slimlm](https://anonymous.slimlm)

**Model Selection and Deployment** We select a list of state-of-the-art (SoTA) models ranging from 125M to 8B parameters as those larger than 8B are very challenging to be deployed even after quantization (Murthy et al., 2024). For quantization and deployment, we use the MLC-LLM framework (MLC-team, 2023) as it supports a wide range of SoTA models and GPU usage on mobile devices. All models are quantized in 4-bit using the group quantization method with a group size of 32.

**Context-length Selection** As document assistance tasks require handling long context inputs, we conduct experiments with different context lengths  $L$  up to 1,000 tokens to measure the models’ efficiency such as input token per second (ITPS), output token per second (OTPS), time to first token (TTFT) and total runtime in seconds. A document is tokenized and the tokens are divided into  $N = 5$  chunks, each chunk has a maximum of  $\frac{\max(L)}{N} = 200$  tokens. We prepare one ( $L = 200$ ), two ( $L = 400$ ) and up to five chunks as context inputs to the models for summarizing.

**Experiment** We first start by asking five different short questions (less than 12 tokens) e.g. “Who was the first president of USA” (Table 7) and measure their efficiency metrics to compute the average (Table 1a). Next, we gradually add more input contexts i.e. chunks extracted from five different documents as described along with different requests (Table 8) to prompt the models for the summarization task and record the average results (Table 1b–e).

**Results** Table 1 presents a clear trade-off between model size and speed, with smaller models like SmolLM or Qwen2 showing higher inference speeds (ITPS, TTFT) but potentially lower accuracy compared to larger models (e.g. Gemma-2, Phi-3.5, Mistral or Llama-3.1). As input length increases, most models experience decreased inference speeds, highlighting the impact of prompt size on efficiency. When the input context reaches approximately 1,000 tokens (5 chunks), smaller models (e.g. SmolLM, Qwen2) struggle to complete multiple experimental runs, while larger models face memory constraints on these long inputs. Mid-sized models like Qwen2-0.5B-Instruct often strike a balance between speed, accuracy, and input handling capacity, potentially offering the best compromise for practical applications within certain input length constraints.

Model	ITPS (t/s)	OTPS (t/s)	TTFT (s)	Runtime (s)
(a) Prompt: "Who was the first president of USA?"				
SmolLM-135M-Instruct	68.48	59.72	0.46	1.42
SmolLM-360M-Instruct	27.56	56.68	0.85	3.71
Qwen2-0.5B-Instruct	23.84	51.78	1.90	2.38
Qwen2-1.5B-Instruct	3.42	17.12	13.01	14.39
Gemma-2-2b-it	1.82	18.64	10.56	13.52
Phi-3-mini-4k-instruct	0.86	14.78	39.81	48.29
Phi-3.5-mini-instruct	0.88	15.60	39.90	47.49
Mistral-7B-Instruct-v0.3	0.44	9.36	127.60	135.12
Llama-3.1-8B-Instruct	0.10	2.20	261.65	269.99
(b) Prompt: 1 chunk ~ 200 tokens (157 words)				
SmolLM-135M-Instruct	167.80	60.80	1.91	4.22
SmolLM-360M-Instruct	28.42	36.12	10.62	16.82
Qwen2-0.5B-Instruct	23.02	39.42	13.15	14.96
Qwen2-1.5B-Instruct	3.86	14.70	78.78	86.14
Gemma-2-2b-it	2.20	11.68	122.06	141.15
Phi-3-mini-4k-instruct	1.05	12.68	327.09	339.87
(c) Prompt: 2 chunks ~ 400 tokens (269 words)				
SmolLM-135M-Instruct	130.66	40.42	4.84	8.14
SmolLM-360M-Instruct	23.28	27.90	30.40	41.07
Qwen2-0.5B-Instruct	18.62	24.72	29.49	38.36
(d) Prompt: 3 chunks ~ 600 tokens (368 words)				
SmolLM-135M-Instruct	174.10	45.70	4.89	8.26
SmolLM-360M-Instruct	31.50	33.94	27.16	33.52
Qwen2-0.5B-Instruct	20.53	25.04	37.94	47.05
(e) Prompt: 4 chunks ~ 800 tokens (529 words)				
SmolLM-135M-Instruct	134.66	32.96	8.47	11.83
SmolLM-360M-Instruct	23.60	25.52	48.06	58.15
Qwen2-0.5B-Instruct	19.74	19.52	54.90	66.65

Table 1: Performance comparison of language models across varying input lengths ranging from single questions to chunks of around 800 tokens. Smaller models demonstrate higher efficiency but potentially lower accuracy, while larger models generally exhibit slower inference speeds but better handling of longer inputs.

## 2.2 Document Assistance Dataset

While smaller models offer faster inference speeds, they often have limited document-handling capabilities. To address this, we develop DocAssist, a specialized dataset designed for fine-tuning these models to enhance their ability to process and assist with longer documents.

### 2.2.1 Data Collection

We utilize our proprietary tools to compile a diverse collection of 82,850 publicly available documents, primarily consisting of illustrations, presentation slides, and spreadsheets. This dataset also includes machine-generated documents to ensure a comprehensive representation of various document types. We extract the document contents and prepare them for pre-processing to ensure the data is suitable for model fine-tuning.

**Pre-processing** We employ Tiktoken (OpenAI, 2023b) in conjunction with GPT-3.5-turbo to tokenize the documents. Each document is segmented into 5 chunks, with each chunk containing a maximum of 200 tokens. This segmentation ensures that the maximum number of tokens per document

after pre-processing is 1,000. Consequently, documents with fewer than 1,000 tokens remain unaltered, while longer documents are truncated. Table 2 presents the statistical analysis of token distribution per document, including the mean, standard deviation, and range of token counts, both before and after pre-processing.

Processing Stage	Mean $\pm$ STD	Token Range
Pre-processing	8,635 $\pm$ 24,235	1 – 1,675,639
Post-processing	879 $\pm$ 252	1 – 1,000

Table 2: Statistical comparison of token distribution per document before and after pre-processing 82,850 documents. The table shows the mean  $\pm$  standard deviation and the range of token counts for each processing stage.

### 2.2.2 Data Annotation

We propose a novel approach for annotating documents using GPT-4o-mini (OpenAI, 2024) to generate comprehensive annotations for three key tasks in DocAssist: SUMM, QS, and QA. For each document, our method produces five distinct examples: one summary, one set of three suggested questions, and three question-answer pairs.

**Prompt Design** Our annotation process employs a carefully designed prompt (Table 3) that instructs the model to perform these tasks sequentially. The prompt is applied to each processed document, replacing the `{{document}}` placeholder with the actual content. The annotation prompt elicits a JSON response containing a document summary, three suggested questions, and their corresponding answers. To ensure high-quality and diverse annotations, we incorporate task-specific requirements:

1. `{{summ_req}}`: to produce concise, informative overviews that capture the document’s essence, enabling models to recognize and respond to requests for document overview.
2. `{{suggestion_req}}`: to generate diverse, relevant questions probing different aspects of the document’s content, allowing models to assist users seeking guidance on what to ask about a document or topic.
3. `{{qa_req}}`: to provide accurate, contextually appropriate answers to document-specific questions, training models to recognize and respond to user queries for specific information or explanations from the document.

Our approach serves several crucial functions: it facilitates intent classification training, enables task-specific response generation, enhances contextual understanding, ensures versatility in document handling, and maintains quality control in annotations. By leveraging the capabilities of GPT-4o-mini, we aim to generate high-quality annotations that capture the nuances and complexities of the documents. The in-context examples and detailed requirements are provided in Tables 9 to 12.

<p>You will be given a document. Your task is to provide a summary of the document, suggest relevant questions, and then answer those questions.</p> <p><b>Task Requirements:</b></p> <ol style="list-style-type: none"> <li>1. Summarization: <code>{{summ_req}}</code></li> <li>2. Question Suggestion: <code>{{suggestion_req}}</code></li> <li>3. Question Answering: <code>{{qa_req}}</code></li> </ol> <p>Format your response in JSON as shown in the examples below.</p> <pre>{   "tasks": {     "summarization": "Your summary here...",     "question_suggestion": [...],     "question_answering": [...]   } }</pre> <p><b>Examples:</b></p> <p>[Two in-context examples here]</p> <p><b>DOCUMENT CONTEXT</b> (may be truncated)</p> <p><code>{{document}}</code></p> <p><b>RESPONSE</b></p>		
---	--	--

Table 3: A prompt designed to annotate data for three tasks given a document in DocAssist: SUMM, QS and QA. `{{document}}` is replaced with each pre-processed document. Please see the complete prompt with in-context examples and requirements for each task `{{summ_req}}`, `{{suggestion_req}}` and `{{qa_req}}` in Tables 9 to 12, respectively.

**Result** Table 4 provides insight into the token usage statistics for the GPT-4o-mini model in annotating 82,850 documents. The relatively low standard deviation in completion tokens suggests consistent-length responses across different documents, which is desirable for maintaining annotation quality and consistency. The annotation process yields ~414K examples for DocAssist. Of these, ~2K examples were randomly selected for the test set, with the remaining examples allocated to the training set.

Token Type	Mean $\pm$ STD	Token Range
Prompt Tokens	2,126.04 $\pm$ 260.81	1,273 – 2,617
Completion Tokens	169.07 $\pm$ 17.61	107 – 312

Table 4: Token usage statistics for GPT-4o-mini model in annotating 82,850 documents.

## 2.3 Slim Language Model

SlimLM is based on the MPT (Mosaic Pre-trained Transformer) architecture by [MosaicML-NLP-Team, 2023](#) with specific modifications to optimize for document assistance tasks. Specifically, we opt not to use the ALiBi ([Press et al., 2022](#)) positioning method as document assistance tasks primarily deal with fixed-length inputs and outputs. Unlike the original MPT, SlimLM incorporates biases in its layers to enhance the model’s flexibility in capturing and representing document-specific nuances. Biases can help the model learn task-specific offsets, potentially improving its ability to distinguish between SUMM, QS, and QA tasks. Based on the sweet-spot findings (Sec. 2.1), we create and train a range of models from 125M to 1B parameters by adjusting the number of layers and heads.

### 2.3.1 Pre-training

We pre-trained SlimLM on the SlimPajama dataset ([Soboleva et al., 2023](#)), comprising 627B tokens. The pre-training objective follows the standard autoregressive language modeling approach, where the model learns to predict the next token in the sequence. The loss function for pre-training can be expressed as:

$$L_{pt} = - \sum_{i=1}^n \log P(x_i | x_{<i}) \quad (1)$$

where  $x_i$  represents the  $i^{th}$  token in the input sequence,  $x_{<i}$  denotes all tokens preceding  $x_i$ , and  $n$  is the length of the sequence.

### 2.3.2 Fine-tuning

Following pre-training, we fine-tuned the models on the training set of DocAssist that comprises ~412K examples to enhance document assistance capabilities by teaching them to handle specific tasks based on user requests. The process instructs the model to first identify the appropriate task from the user’s input and then generate a response that match the quality of GPT-4o-mini for the identified task. The fine-tuning loss function is also an autoregressive objective, defined as:

$$L_{ft} = - \sum_{i=1}^m \log P(y_i | y_{<i}, x) \quad (2)$$

where  $x$  is the input sequence (system prompt, document and user request),  $y_i$  is the  $i^{th}$  token in the target response generated by GPT-4o-mini,  $y_{<i}$  denotes all tokens preceding  $y_i$  in the target response  $m$  is the length of the target response.



### 3 Experiments and Results

#### 3.1 Experiment Setup

We pre-train SlimLM from scratch on the SlimPajama dataset using 128-256 A100/H100 GPUs using Lion optimizer (Chen et al., 2023) with different learning rates (LRs), global batch size, and number of trained tokens. All models are fine-tuned on DocAssist using 8 A100 GPUs using AdamW optimizer (Loshchilov, 2017) with the same LR of 5e-6 and global batch size of 48. The models’ configurations and hyperparameters are in Table 17.

##### 3.1.1 Baselines

Our selection is based on the sweet-spot results that demonstrate a clear trade-off between model size, speed, and context length. Specifically, we compare with the following models: SmolLM-135M-Instruct, SmolLM-360M-Instruct (HuggingFace, 2024.07), Qwen2-0.5B-Instruct and Qwen2-1.5B-Instruct (Alibaba, 2024.06). These models represent SoTA performance at their respective sizes, making them strong baselines for comparison.

##### 3.1.2 Evaluation Metrics

We employ a diverse set of metrics to evaluate models’ performance across the DocAssist tasks. For Intent Detection, we use Accuracy to measure classification precision. SUM, QS, and QA tasks are evaluated using BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), and Semantic Textual Similarity (STS) scores, which assess the quality, overlap, and semantic similarity of generated outputs compared to references. GEval (Liu et al., 2023) provide a comprehensive quality assessment with human alignment for SUMM and QA<sup>2</sup> outputs. While other metrics have scores in the range [0, 1], GEval scores range from 1 to 4.5. To ensure consistency across metrics, we rescale GEval scores to the same interval. Additionally, we use Self-BLEU for Text Diversity (Zhu et al., 2018) for QS to ensure varied outputs.

#### 3.2 Results

Before finetuning, all models cannot perform document assistance tasks or detect user intents. After finetuning, most models achieve perfect accuracy, with the lowest score being 99.86% from SmolLM-360M-Instruct (Table 6). Table 5 demonstrates the effectiveness of our SlimLM models compared to the baselines across the three DocAssist tasks.

<sup>2</sup>We adjust GEval prompts originally designed for summarization task accordingly for the evaluation of QA task.

Specifically, SlimLM models consistently outperform or match the performance of similar-sized counterparts, indicating the efficiency of our architecture. SlimLM-125M surpasses SmolLM-135M-Instruct, while both SlimLM-270M and SlimLM-350M outperform SmolLM-360M-Instruct. Notably, SlimLM-450M and SlimLM-760M achieve comparable results to Qwen2-0.5B-Instruct, despite the latter being pre-trained and fine-tuned on a substantially larger dataset. Detailed results for each task are presented in the appendix (Tables 14 to 16).

As model size increases (Table 5), we observe consistent improvement across all metrics, suggesting good scalability. Our largest model, SlimLM-1B, approaches the performance of the much larger model Qwen2-1.5B-Instruct, highlighting the potential for SlimLM to achieve competitive results with reduced computational requirements. While GPT-4o-mini still leads in overall performance, our SlimLM models offer a range of efficient options for various computational constraints and privacy concerns in document assistance tasks.

### 4 Use Case

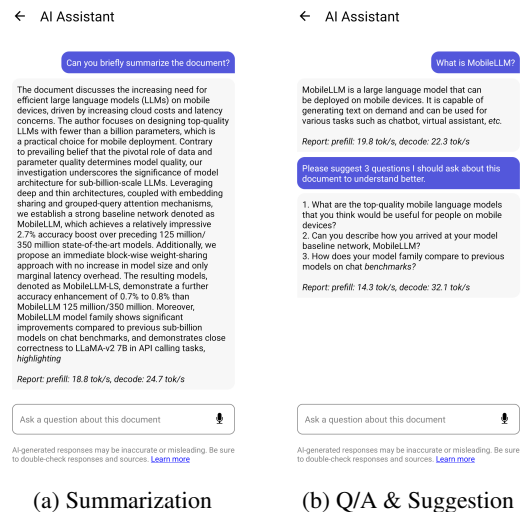


Figure 1: Loading MobileLLM paper (Liu et al., 2024) and interacting with AI assistant without internet access.

Adobe’s Acrobat relies on API calls for SUMM, QS and QA, incurring significant costs and raising privacy concerns. SlimLM can be deployed into the Acrobat mobile apps, enabling local processing of documents. This approach eliminates the need for external API calls, substantially reducing operational costs while enhancing user privacy by keeping document content on the device.

Model	BLEU $\uparrow$	ROUGE-1 $\uparrow$	ROUGE-2 $\uparrow$	ROUGE-L $\uparrow$	STS Score $\uparrow$	GEval $\uparrow$	Average
GPT-4o-mini	1.00	1.00	1.00	1.00	1.00	0.88	0.9795
SmolLM-135M-Instruct	0.10	0.37	0.17	0.34	0.64	0.60	0.3694
SmolLM-360M-Instruct	0.14	0.42	0.21	0.38	0.68	0.69	0.4202
Qwen2-0.5B-Instruct	0.21	0.49	0.28	0.45	0.74	0.79	0.4934
Qwen2-1.5B-Instruct	0.26	0.53	0.33	0.50	0.77	0.84	0.5396
LLaMA-3.2-1B-Instruct	0.26	0.53	0.33	0.50	0.77	0.86	<b>0.5442</b>
<b>Slim Language Models (ours)</b>							
SlimLM-125M <sup>a</sup>	<b>0.14</b>	<b>0.41</b>	<b>0.21</b>	<b>0.38</b>	<b>0.66</b>	<b>0.64</b>	<b>0.4052</b>
SlimLM-270M	0.17	0.45	0.24	0.42	0.71	0.72	0.4497
SlimLM-350M <sup>b</sup>	<b>0.18</b>	<b>0.45</b>	<b>0.25</b>	<b>0.42</b>	<b>0.71</b>	<b>0.73</b>	<b>0.4541</b>
SlimLM-450M <sup>c</sup>	0.20	0.48	0.27	0.44	0.73	0.76	0.4806
SlimLM-760M	0.21	0.48	0.28	0.45	0.74	0.79	0.4911
SlimLM-1B <sup>d</sup>	0.23	0.51	0.31	0.48	0.76	0.81	0.5182

Table 5: **Comparison of model performance on average of three tasks: SUMM, QS and QA.** Green highlighting indicates superior performance of SlimLM models compared to similar-sized counterparts. Key comparisons: (a) SlimLM-125M outperforms SmolLM-135M-Instruct, (b) SlimLM-350M exceeds SmolLM-360M-Instruct, (c) SlimLM-450M is comparable to Qwen2-0.5B-Instruct, and (d) SlimLM-1B approaches Qwen2-1.5B-Instruct despite being smaller. Tables 14 to 16 present detailed results for each task.

When a document is loaded, such as a legal contract, the app instantly generates a summary, suggests relevant questions, and provides quick answers to user queries, all without internet connectivity. This streamlined process allows professionals to grasp essential information rapidly and identify areas needing closer examination while maintaining document confidentiality and improving overall user experience. Users can also interact with the document by chatting with the AI assistant (Fig. 1).

## 5 Related Work

### 5.1 Small and Large Language Models

Large language models (Chowdhery et al., 2022; Chung et al., 2022; Touvron et al., 2023a,b) have demonstrated impressive capabilities across various NLP tasks. However, their massive size limits practical deployment, especially on resource-constrained devices. This has spurred interest in small language models (Microsoft, 2023.12, 2024.04; Bai et al., 2023; Google, 2024.07) that balance performance and efficiency. While some approaches focus on compressing LLMs through techniques like knowledge distillation (Gu et al., 2023; Zhang et al., 2024), our work aligns more closely with efforts to design and train efficient SLMs from scratch (Liu et al., 2024; MBZUAI, 2024.02). These approaches aim to achieve competitive performance with smaller model sizes and less training data. Our SlimLM builds on these efforts by focusing specifically on optimizing SLMs for document processing tasks on mobile devices.

### 5.2 SLMs for Mobile Devices

Deploying language models on mobile devices presents unique challenges, including memory constraints, inference latency, and energy efficiency (Liu et al., 2024; MBZUAI, 2024.02; Chen et al., 2024). The growing importance of efficient on-device language models is further underscored by recent developments from major tech companies (Reid et al., 2024; Apple, 2024.09; Meta, 2024.09). Our work extends this line of research by identifying the optimal balance between model size, context length, and performance specifically for real mobile devices e.g. Samsung Galaxy S24. We focus on enhancing document assistance abilities by designing and training SlimLM (Sec. 2.3) from scratch on SlimPajama and DocAssist (Sec. 2.2), advancing the SoTA in mobile-deployed language models for document processing applications.

## 6 Conclusion

In this work, we introduce SlimLM models optimized for document assistance tasks. We identify the optimal balance between model size, inference time, and maximum context length for efficient processing on real mobile devices. Our specialized DocAssist dataset, constructed from ~83K documents, enabled fine-tuning of SlimLM for three critical document assistance tasks. SlimLM models, ranging from 125M to 1B parameters, demonstrate comparable or superior performance to existing SLMs of similar sizes across standard metrics, while efficiently handling up to 800 context tokens. To showcase real-world applicability, we develop an Android application featuring SlimLM’s document assistance capabilities, paving the way for

widespread deployment of efficient, on-device language models for enhanced user privacy and reduced server costs.

## References

- Alibaba. 2023.11. Qwen 1. <https://huggingface.co/alibaba/Qwen-1>.
- Alibaba. 2024.06. Qwen 2. <https://qwenlm.github.io/blog/qwen2/>.
- Alibaba. 2024.09. Qwen 2.5. <https://qwenlm.github.io/blog/qwen2.5/>.
- Apple. 2024.09. apple-intelligence. <https://www.apple.com/apple-intelligence/>.
- Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-vl: A frontier large vision-language model with versatile abilities. *arXiv preprint arXiv:2308.12966*.
- Wei Chen, Zhiyuan Li, and Mingyuan Ma. 2024. Octopus: On-device language model for function calling of software apis. *arXiv preprint arXiv:2404.01549*.
- Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. 2023. *Symbolic discovery of optimization algorithms*. Preprint, arXiv:2302.06675.
- Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.
- Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.
- Google. 2024.07. Gemma-2. <https://huggingface.co/google/Gemma-2>.
- Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2023. Knowledge distillation of large language models. *arXiv preprint arXiv:2306.08543*.
- HuggingFace. 2024.07. SmolLM. <https://huggingface.co/huggingface/SmolLM>.
- Chin-Yew Lin. 2004. *ROUGE: A package for automatic evaluation of summaries*. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.
- Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. *G-eval: NLG evaluation using gpt-4 with better human alignment*. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522, Singapore. Association for Computational Linguistics.
- Zechun Liu, Changsheng Zhao, Forrest Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, Liangzhen Lai, and Vikas Chandra. 2024. *MobileLLM: Optimizing sub-billion parameter language models for on-device use cases*. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 32431–32454. PMLR.
- I Loshchilov. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.
- MBZUAI. 2024.02. Mobillama. <https://huggingface.co/mbzuai/MobileLlama>.
- Meituan. 2023.12. Mobilellama. <https://huggingface.co/meituan/MobileLLaMA>.
- Meta. 2024.09. Llama-3.2. <https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/>.
- Microsoft. 2023.12. microsoft/phi-2. <https://huggingface.co/microsoft/phi-2>.
- Microsoft. 2024.04. microsoft/phi-3-mini. <https://huggingface.co/microsoft/phi-3-mini>.
- MLC-team. 2023. *MLC-LLM*.
- MosaicML-NLP-Team. 2023. *Introducing mpt-7b: A new standard for open-source, commercially usable llms*. Accessed: 2023-05-05.
- Rithesh Murthy, Liangwei Yang, Juntao Tan, Tulika Manoj Awalganekar, Yilun Zhou, Shelby Heinecke, Sachin Desai, Jason Wu, Ran Xu, Sarah Tan, et al. 2024. *Mobileaibench: Benchmarking llms and lmms for on-device use cases*. *arXiv preprint arXiv:2406.10290*.
- OpenAI. 2023a. GPT-4 Technical Report. <https://arxiv.org/pdf/2303.08774v3.pdf>.
- OpenAI. 2023b. *Tiktoken*.
- OpenAI. 2024. *Hello gpt-4o*.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. *Bleu: a method for automatic evaluation of machine translation*. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Ofir Press, Noah Smith, and Mike Lewis. 2022. [Train short, test long: Attention with linear biases enables input length extrapolation](#). In *International Conference on Learning Representations*.

Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. 2023. [SlimPajama: A 627B token cleaned and deduplicated version of RedPajama](#). <https://www.cerebras.net/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-version-of-redpajama>.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. [Tinyllama: An open-source small language model](#). *Preprint*, arXiv:2401.02385.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. [Texygen: A benchmarking platform for text generation models](#). In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR '18*, page 1097–1100, New York, NY, USA. Association for Computing Machinery.



## A Appendix

Model	Accuracy (%)
GPT-4o-mini	100.00
SmoLM-135M-Instruct	99.86
SmoLM-360M-Instruct	99.81
Qwen2-0.5B-Instruct	100.00
Qwen2-1.5B-Instruct	100.00
SlimLM-125M	100.00
SlimLM-270M	100.00
SlimLM-350M	100.00
SlimLM-450M	100.00
SlimLM-760M	99.95
SlimLM-1B	99.90

Table 6: Intent Classification accuracy of various language models after fine-tuning on DocAssist dataset.

Q1: Who was the first president of USA?  
 Q2: What is the capital city of France?  
 Q3: Who was the first person to walk on the moon?  
 Q4: What is the chemical symbol for gold?  
 Q5: In what year did World War II end?

Table 7: Fact-checking questions asked to measure a model’s efficiency on real mobile devices.

R1. Please summarize the document excerpt(s) below:  
 R2. Kindly provide a concise overview of the following document excerpt(s):  
 R3. Briefly outline the main points from the passage(s) below:  
 R4. Highlight the key ideas from the following text sample(s):  
 R5. Capture the key points of the document snippet(s) provided:

Table 8: Summarizing requests used to measure a model’s efficiency with different input contexts on real mobile devices.

Summarize the main topic and key points of this document in one concise sentence. Ensure the summary gives a clear overview of the document’s content without including minor details.

Table 9: `{{summ_req}}`. Instructional prompt designed to guide GPT-4o-mini how to summarize the document contents.

**Provide answers to the suggested questions, adhering to the following guidelines:**

- Answer each question directly and completely based on the information in the document.
- Provide specific details, explain your reasoning and, if applicable, cite relevant parts of the document.
- Keep answers concise but informative, typically 1-3 sentences each.
- If a question cannot be fully answered based solely on the document, state this clearly and provide the best possible answer with the available information.
- Ensure that answers are accurate and directly related to the corresponding questions.

Table 10: `{{qa_req}}`. Instructional prompt designed to guide GPT-4o-mini how to answer questions for the Q/A task.

Generate *three insightful questions* so a user can explore and understand the document better and more quickly.

**When generating the questions, please consider the following:**

- What questions am I interested in asking as a reader?
- What questions does this document actually answer?

**Please make sure to adhere to the following specifications:**

- Questions must be short and simple.
- Each question must be less than 12 words.
- You must not write questions that are too general. For example, “what is this document about?” or “what is the purpose of this document” are bad questions.
- Questions must be specific to the document. For example, you should consider using entities and proper nouns that appear in the document to write your question, whenever possible.
- Questions must have an answer based on the document I am reading.
- Questions must be diverse, covering different parts of the document.
- Please generate exactly 3 questions.

Table 11: `{{suggestion_req}}`. Instructional prompt designed to guide GPT-4o-mini how to generate suggested questions for a given document. The suggested questions aims to guide users what should be asked to understand the document.

You will be given a document. Your task is to provide a summary of the document, suggest relevant questions, and then answer those questions.

**Task Requirements:**

1. Summarization: `{{summ_req}}`
2. Question Suggestion: `{{suggestion_req}}`
3. Question Answering: `{{qa_req}}`

Format your response in JSON as shown in the examples below.

```
{
  "tasks": {
    "summarization": "Your summary here...",
    "question_suggestion": ["Question 1?", "Question 2?", "Question 3?"],
    "question_answering": ["Answer to question 1.", "Answer to question 2.", "Answer to question 3."]
  }
}
```

**Examples:**

**DOCUMENT CONTEXT** (may be truncated)

XYZ Corporation has released its Q3 earnings report. The company reported a net income of \$5 million, up from \$3 million in Q2. The earnings per share (EPS) increased to \$0.25 from \$0.15 in the previous quarter. The revenue for Q3 was \$50 million, a 10% increase compared to Q2. The company's operating expenses increased by 5% to \$20 million. The table below summarizes the key financial figures:

Metric	Q3	Q2	% Change
Net Income	\$5M	\$3M	66.67%
EPS	\$0.25	\$0.15	66.67%
Revenue	\$50M	\$45M	11.11%
Operating Expenses	\$20M	\$19M	5.26%

**RESPONSE**

```
{
  "tasks": {
    "summarization": "XYZ Corporation's Q3 earnings report shows significant growth with net income rising to $5 million from $3 million in Q2. EPS increased to $0.25, revenue grew by 11.11% to $50 million, and operating expenses rose by 5.26% to $20 million.",
    "question_suggestion": [
      "What's the Q2 to Q3 revenue percentage increase?",
      "What's the EPS increase from Q2 to Q3?",
      "What's the operating expenses percentage change between Q2 and Q3?"
    ],
    "question_answering": [
      "The Q2 to Q3 revenue percentage increase is 11.11%.",
      "The EPS increased from $0.15 in Q2 to $0.25 in Q3, which is a $0.10 or 66.67% increase.",
      "The operating expenses percentage change between Q2 and Q3 is 5.26%."
    ]
  }
}
```

**DOCUMENT CONTEXT** (may be truncated)

# Document Excerpt 1: In the land of X, fairies and trolls have always been at odds. The fairies, known for their beauty and grace, live in the enchanted forest, while the trolls, known for their strength and cunning, reside in the dark caves. The two groups have had friction for centuries, with each side trying to outwit...

# Document Excerpt 2: The fairies use their magical powers to protect their forest, while the trolls rely on their dancing abilities to enter the fairies' territory. The story of their ongoing struggle is a tale of cunning, bravery, and the power of friendship.

**RESPONSE**

```
{
  "tasks": {
    "summarization": "In the land of X, fairies and trolls have been in conflict for centuries. Fairies, living in the enchanted forest, use magic for protection, while trolls from dark caves use dancing to infiltrate fairy territory. Their struggle is characterized by cunning, bravery, and unexpected friendship.",
    "question_suggestion": [
      "What traits define fairies and trolls?",
      "How do fairies defend their forest?",
      "What strategies do trolls use for entering the fairies' territory?"
    ],
    "question_answering": [
      "Fairies are defined by their beauty and grace, while trolls are known for their strength and cunning.",
      "Fairies use their magical powers to protect their forest.",
      "Trolls rely on their dancing abilities to enter the fairies' territory."
    ]
  }
}
```

**DOCUMENT CONTEXT** (may be truncated)

`{{document}}`

**RESPONSE**

Table 12: Full prompt designed to annotate data for three tasks given a document in DocAssist: Summarization, Question Answering and Question Suggestion. Please see the requirements for each task `{{summ_req}}`, `{{suggestion_req}}` and `{{qa_req}}` in Tables 9 to 11, respectively.

<p>You are an AI assistant for document analysis, performing summarization, question suggestion, and question answering.</p> <p><b>For each task:</b></p> <ol style="list-style-type: none"> <li>1. Analyze the given document</li> <li>2. Determine the task (summarization, question suggestion, or question answering)</li> <li>3. Perform the requested task</li> </ol> <p><b>Respond using this format:</b></p> <pre>&lt;intent&gt;: [summarization question_suggestion question_answering] &lt;response&gt; [Task-specific response here] &lt;/response&gt;</pre> <p>Now, analyze the following document and respond to the request:</p> <pre>&lt;document&gt; {{document}} &lt;/document&gt;  &lt;request&gt; {{request}} &lt;/request&gt;</pre>
---

Table 13: Full prompt designed to finetune SMLs to detect and handle three tasks given a user-uploaded document in DocAssist: Summarization, Question Answering and Question Suggestion.

Table 14: **Summarization task performance comparison.** SlimLM models show competitive performance: (a) SlimLM-125M outperforms SmolLM-135M-Instruct, (b) SlimLM-350M surpasses SmolLM-360M-Instruct, (c) SlimLM-450M performs comparably to Qwen2-0.5B-Instruct, and (d) SlimLM-1B approaches Qwen2-1.5B-Instruct’s performance despite being smaller.

Model	BLEU ↑	ROUGE-1 ↑	ROUGE-2 ↑	ROUGE-L ↑	STS Score ↑	GEval ↑	Average
GPT-4o-mini	1.00	1.00	1.00	1.00	1.00	0.86	0.9760
SmolLM-135M-Instruct	0.09	0.37	0.14	0.32	0.69	0.63	0.3762
SmolLM-360M-Instruct	0.13	0.42	0.18	0.36	0.74	0.71	0.4233
Qwen2-0.5B-Instruct	0.20	0.50	0.25	0.43	0.82	0.79	0.4985
Qwen2-1.5B-Instruct	0.26	0.54	0.31	0.48	0.84	0.83	0.5433
Slim Language Models (ours)							
SlimLM-125M <sup>a</sup>	0.12	0.40	0.17	0.35	0.73	0.66	0.4061
SlimLM-270M	0.17	0.46	0.22	0.40	0.79	0.74	0.4620
SlimLM-350M <sup>b</sup>	0.16	0.45	0.22	0.39	0.78	0.74	0.4570
SlimLM-450M <sup>c</sup>	0.20	0.49	0.25	0.43	0.80	0.77	0.4893
SlimLM-760M	0.20	0.49	0.25	0.43	0.81	0.78	0.4921
SlimLM-1B <sup>d</sup>	0.23	0.52	0.28	0.46	0.82	0.81	0.5194

Table 15: **Question Answering task performance comparison.** SlimLM models demonstrate strong performance: (a) SlimLM-125M outperforms SmolLM-135M-Instruct, (b) SlimLM-350M surpasses SmolLM-360M-Instruct, (c) SlimLM-450M and SlimLM-760M perform comparably to Qwen2-0.5B-Instruct, and (d) SlimLM-1B approaches Qwen2-1.5B-Instruct’s performance.

Model	BLEU ↑	ROUGE-1 ↑	ROUGE-2 ↑	ROUGE-L ↑	STS Score ↑	GEval ↑	Average
GPT-4o-mini	1.00	1.00	1.00	1.00	1.00	0.90	0.9830
SmolLM-135M-Instruct	0.18	0.45	0.26	0.42	0.72	0.56	0.4300
SmolLM-360M-Instruct	0.22	0.49	0.31	0.46	0.76	0.67	0.4860
Qwen2-0.5B-Instruct	0.30	0.57	0.39	0.54	0.81	0.79	0.5687
Qwen2-1.5B-Instruct	0.36	0.62	0.44	0.59	0.84	0.85	0.6157
Slim Language Models (ours)							
SlimLM-125M <sup>a</sup>	0.22	0.49	0.30	0.46	0.75	0.62	0.4731
SlimLM-270M	0.24	0.52	0.33	0.49	0.78	0.69	0.5077
SlimLM-350M <sup>b</sup>	0.26	0.53	0.35	0.50	0.78	0.72	0.5246
SlimLM-450M <sup>c</sup>	0.29	0.56	0.37	0.53	0.80	0.75	0.5491
SlimLM-760M <sup>c</sup>	0.30	0.57	0.39	0.54	0.81	0.79	0.5679
SlimLM-1B <sup>d</sup>	0.32	0.60	0.41	0.57	0.83	0.81	0.5907

Table 16: **Question Suggestion task performance comparison.** SlimLM models show competitive results: (a) SlimLM-125M outperforms SmolLM-135M-Instruct, (b) SlimLM-350M surpasses SmolLM-360M-Instruct, (c) SlimLM-450M and SlimLM-760M perform comparably to Qwen2-0.5B-Instruct, and (d) SlimLM-1B approaches Qwen2-1.5B-Instruct’s performance in most metrics. As Self-BLEU measures text diversity where lower scores indicate higher diversity (better), it is not included in the average scores.

Model	BLEU ↑	ROUGE-1 ↑	ROUGE-2 ↑	ROUGE-L ↑	STS Score ↑	Diversity ↓	Average
GPT-4o-mini	1.00	1.00	1.00	1.00	1.00	0.04	1.0000
SmolLM-135M-Instruct	0.04	0.29	0.11	0.29	0.49	0.05	0.2434
SmolLM-360M-Instruct	0.07	0.34	0.15	0.33	0.53	0.03	0.2837
Qwen2-0.5B-Instruct	0.12	0.39	0.20	0.38	0.59	0.02	0.3381
Qwen2-1.5B-Instruct	0.16	0.44	0.25	0.43	0.63	0.02	0.3837
Slim Language Models (ours)							
SlimLM-125M <sup>a</sup>	0.07	0.33	0.14	0.32	0.52	0.04	0.2754
SlimLM-270M	0.10	0.37	0.18	0.36	0.56	0.03	0.3122
SlimLM-350M <sup>b</sup>	0.10	0.36	0.18	0.35	0.56	0.03	0.3109
SlimLM-450M <sup>c</sup>	0.11	0.39	0.20	0.38	0.59	0.02	0.3326
SlimLM-760M <sup>c</sup>	0.12	0.39	0.20	0.38	0.59	0.02	0.3389
SlimLM-1B <sup>d</sup>	0.15	0.43	0.24	0.42	0.62	0.02	0.3713

	# Layers	# Heads	Model Dimension	Learning Rate	Global Batch Size	# Trained Tokens (billions)
SlimLM-125M	12	12	2,048	3e-4	2,048	627
SlimLM-270M	16	64	2,048	4e-4	2,048	627
SlimLM-350M	24	16	2,048	3e-4	2,048	627
SlimLM-450M	20	64	2,048	3e-4	2,048	627
SlimLM-760M	24	12	2,048	3e-4	2,048	627
SlimLM-1B	24	16	2,048	2e-4	2,048	627

Table 17: Specifications of SlimLM models and hyperparameters for pre-training. Fine-tuning parameters are consistent across all models: learning rate of 5e-6, global batch size of 48, and 2 epochs (~725M trained tokens).