LinguaLinked: Distributed Large Language Model Inference on Mobile Devices

Junchen Zhao*
UC Irvine
junchez3@uci.edu

Yurun Song*
UC Irvine
yuruns@uci.edu

Simeng Liu
UC Irvine
simenl3@uci.edu

Ian G. Harris
UC Irvine
harris@ics.uci.edu

Sangeetha Abdu Jyothi
UC Irvine, VMware Research
sangeetha.aj@uci.edu

Abstract

Deploying Large Language Models (LLMs) locally on mobile devices presents a significant challenge due to their extensive memory requirements. In this paper, we introduce LinguaLinked, a system for decentralized, distributed LLM inference on mobile devices. LinguaLinked enables collaborative execution of the inference task across multiple trusted devices and ensures data privacy by processing information locally. LinguaLinked uses three key strategies. First, an optimized *model assignment technique* segments LLMs and uses linear optimization to align segments with each device's capabilities. Second, an optimized *data transmission mechanism* ensures efficient and structured data flow between model segments while also maintaining the integrity of the original model structure. Finally, LinguaLinked incorporates a *runtime load balancer* that actively monitors and redistributes tasks among mobile devices to prevent bottlenecks, enhancing the system's overall efficiency and responsiveness. We demonstrate that LinguaLinked facilitates efficient LLM inference while maintaining consistent throughput and minimal latency through extensive testing across various mobile devices, from high-end to low-end Android devices.

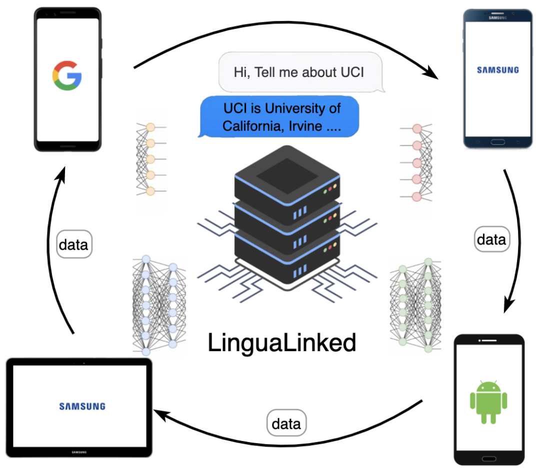

Figure 1: 'Trusted' mobile devices working collaboratively for LLM inference in LinguaLinked.

1 Introduction

The past decade has witnessed a seismic shift in the machine learning (ML) landscape, particularly with the rise of large language models (LLMs), which are built atop transformer decoders (Vaswani et al., 2023). These LLMs (Brown et al., 2020; Kaplan et al., 2020, Hoffmann et al., 2022; Chowdhery et al., 2022; Zhang et al., 2022; Touvron et al., 2023; Workshop et al., 2023) have achieved state-of-art performance on Natural Language Processing (NLP) benchmarks such as text generation, question answering, machine translation, and text summarization, and led to commercial offerings such as OpenAI ChatGPT and Github Copilot. Recent research has established that as the number of parameters in these models increases, they demonstrate enhanced capabilities in various language tasks (Alabdulmohsin et al., 2022; Clark et al., 2022; Huang et al., 2020; Patel and Pavlick, 2022, Hendrycks et al., 2021; Cobbe et al., 2021).

However, deploying these LLMs on mobile devices is challenging due to their significant memory and processing requirements. Traditional server-based inference raises privacy and bandwidth issues. An alternative is distributed inference, where LLMs are split into smaller segments across multiple devices, reducing the need for heavy model weight quantization and maintaining accuracy. While previous studies have looked into distributed model deployment on mobile computing platforms (Hu et al., 2019; Naveen et al., 2021; Zeng et al., 2021; Zhou et al., 2019), these have largely concentrated on smaller-scale models used in computer vision applications, which have a much smaller memory footprint compared to LLMs and

[0]* Equally contributed.

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), pages 160–171
August 11-16, 2024 ©2024 Association for Computational Linguistics

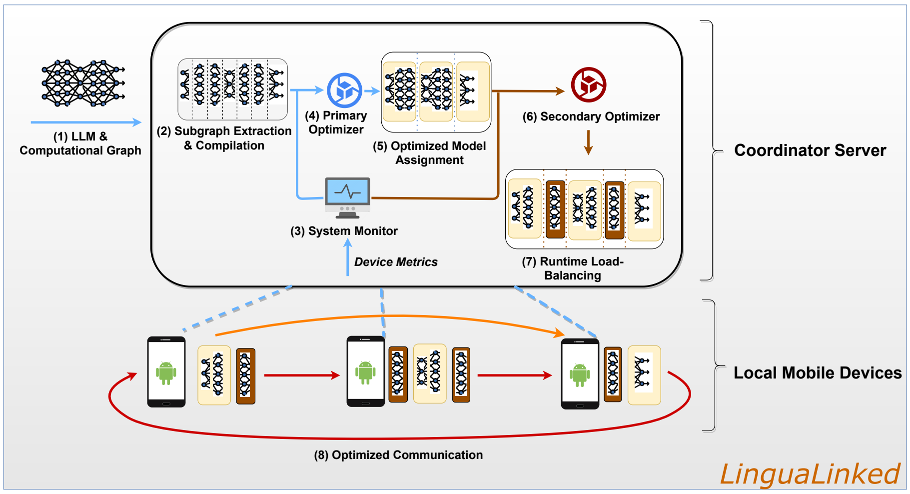

Figure 2: Overview of LinguaLinked System Design.

typically do not need iterative inference.

In this paper, we present LinguaLinked[1], a decentralized distributed inference system for LLM deployment on mobile devices. The core concept behind LinguaLinked is distributing model segments across 'trusted' devices shown in Figure 1, such as personal smartphones and tablets. This approach overcomes the limitations of individual device capacities, privacy concerns, and bandwidth constraints. However, it faces challenges like managing diverse device capabilities, handling data dependencies between model segments, and adjusting to dynamic resource availability.

LinguaLinked addresses these challenges with three key components: optimized model assignment that aligns model segments with device capabilities while minimizing data transmission, runtime load balancing to redistribute tasks and prevent bottlenecks, and optimized communication to ensure efficient data exchange between model segments. We perform a thorough evaluation of LinguaLinked on high-end and low-end Android devices. In a single-threaded setting, compared to the baseline, LinguaLinked achieves an inference performance acceleration of approximately $1.11\times$ to $1.61\times$ across both quantized and full-precision models. With multi-threading, the system exhibits further improvements, achieving acceleration rates of approximately $1.73\times$ to $2.65\times$ for both quantized and full-precision models. Runtime load balancing yields an overall inference acceleration of $1.29\times$ to $1.32\times$. Importantly, our findings indicate that LinguaLinked's performance gains are more pronounced with larger models, suggesting enhanced scalability and effectiveness in handling complex, resource-intensive tasks. We develop an Android application to demonstrate LinguaLinked's effectiveness in a typical mobile computing environment, showing how LLMs can operate on devices with diverse capabilities.

2 Related Works

Autoregressive LLMs and Computational Challenges. Recent advancements in NLP have been driven by autoregressive LLMs like GPT-3 (Brown et al., 2020), OPT (Zhang et al., 2022), and LLaMA (Touvron et al., 2023), which generate text sequentially. While effective for tasks such as language generation and translation, their sequential nature leads to computational inefficiencies, especially for longer texts (Lin et al., 2021; Floridi and Chiriatti, 2020; Lee, 2023). Traditionally, these models have been processed on centralized servers (Aminabadi et al., 2022; Borzunov et al., 2022; Du et al., 2023), with innovations aimed at reducing latency and enhancing efficiency (Wang et al., 2023; Romero et al., 2021; Gunasekaran et al., 2022). However, centralization raises privacy concerns and can introduce latency due to data transmission requirements (Khowaja et al., 2023; Sebastian, 2023; Renaud et al., 2023; Kshetri, 2023; Elbamby et al., 2019; Liang et al., 2020a; Park et al., 2019; Mao et al., 2017b).

Mobile Constraints and Model Optimization. Deploying LLMs on mobile devices presents challenges due to limited computational and memory resources (Wu et al., 2019; Zhao et al., 2022; Chen and Ran, 2019; Zhang et al., 2019). Techniques like

[1] https://github.com/zjc664656505/LinguaLinked-Inference

quantization (Gholami et al., 2022; Bondarenko et al., 2021; Coelho et al., 2021), distillation (Liang et al., 2020b; Gu et al., 2023; Jiao et al., 2019), and pruning (Blalock et al., 2020; Hoefler et al., 2021; Liang et al., 2021) help mitigate these issues but may compromise model performance. Frameworks such as TensorFlow Lite (TensorFlow, 2023), TVM (Chen et al., 2018), and ONNXRuntime (Runtime, 2023), along with advanced quantization methods (Yao et al., 2022; Frantar et al., 2022; Xiao et al., 2023), facilitate mobile deployment, yet challenges persist, especially on lower-end devices.

Distributed Inference Solutions. Addressing the limitations of single-device deployment, distributed inference strategies like LinguaLinked partition LLMs across multiple devices, reducing the memory load on individual devices and enabling broader device participation in inference tasks. Frameworks such as DeepHome (Hu et al., 2019), MODNN (Mao et al., 2017a), and EdgeFlow (Hu and Li, 2022) have explored data parallelism and model partitioning, primarily for vision models. However, the adaptation of these strategies for LLMs on mobile devices remains an underexplored area, with a need for solutions that consider real-time device performance fluctuations and load balancing (Xu et al., 2022).

Figure 3: Android chat application that runs full-precision BLOOM 1.7b on 2 Google Pixel 7 pro. The demo video can be found at https://youtu.be/4UhXzKUkOuI

3 LinguaLinked

As shown in Figure 2, the LinguaLinked system facilitates LLM distribution across mobile devices by transforming the LLM into a computational graph on a coordinator server, then partitioning it into sub-modules for optimized allocation to devices based on their performance metrics. It employs primary and secondary optimizers for task distribution and load balancing, with a communication strategy that minimizes data transmission between devices, ensuring efficiency and privacy as all data remains local to the devices.

3.1 System Monitor

The system monitor in LinguaLinked is comprised of server and device modules to track and manage performance metrics like bandwidth, latency, memory, and processing speed across devices. The server module controls monitoring activities and processes data for optimization, while the device module assesses performance indicators. Bandwidth is measured by transferring data between devices and calculating the transfer rate, while processing speed (FLOP/s) is determined by timing a test model's execution on each device, offering insights into the system's operational efficiency.

3.2 Optimized Model Assignment

Subgraph Extraction from LLMs. The first step in preparing LLMs for mobile deployment involves converting them into computational graphs and then segmenting these graphs into smaller, independent subgraphs. These subgraphs are designed to operate separately on different devices, and their extraction is based purely on the computational characteristics of the LLM, without considering the current state of the devices. Nodes within the graph that process inputs from a single node and output to multiple nodes are identified as key points for partitioning. These nodes typically represent distinct layers or operations within the model, making them ideal for creating subgraphs that can be independently executed. The process results in a series of subgraphs, each representing a functional segment of the original LLM, allowing for efficient distribution across the available mobile devices.

Subgraph Dependency Search. To handle dependencies between nodes in separate subgraphs of an LLM, we employ a subgraph dependency search algorithm, creating two key maps: the residual dependency map (RDM) and the sequential dependency map (SDM). The SDM tracks direct dependencies between adjacent subgraphs, ensuring that outputs from one subgraph serve as inputs for the next. The RDM identifies dependencies between non-adjacent subgraphs, capturing instances where a subgraph relies on nodes from an earlier subgraph, not directly preceding it.

Model Assignment Optimization. After segmenting LLMs into subgraphs, the next step involves assigning these subgraphs as executable sub-modules to mobile devices, considering device constraints and aiming to minimize computation and data transmission times. This involves compiling subgraphs into sub-modules, profiling each for FLOP count, memory needs, and data output size, and then using this information alongside device performance metrics to optimize sub-module allocation. The optimization termed as a primary optimizer, formulated as a linear optimization problem, balances local computation and data transmission efforts to reduce total inference time. Constraints ensure that the memory usage of sub-modules on any device does not exceed a predetermined portion of the device's available memory.

3.3 Runtime Load Balancing

Load Balancing Optimization. The load balancing mechanism refines the initial model assignment optimization termed as a secondary optimizer by introducing a strategy to overlap sub-modules across devices, categorizing them as movable or unmovable. Movable sub-modules can be dynamically allocated or removed to balance the load, whereas unmovable ones stay fixed. This approach uses linear programming to minimize data transmission and optimize memory usage, enhancing system performance and robustness during intensive tasks. The optimization allows for potential overlaps of sub-modules to the left or right of their current allocation, improving memory utilization within device constraints and facilitating efficient load distribution across the network.

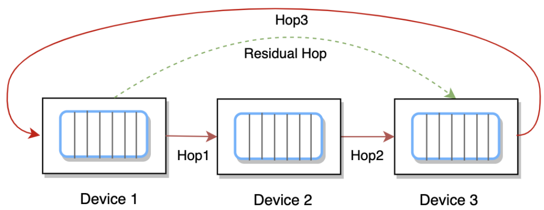

Figure 4: System Design For Device Communication.

3.4 Optimized Communication

Model Deployment with Load Balancing. LinguaLinked initiates load balancing based on real-time device performance metrics. When an imbalance is detected, it combines the initial model assignment with the secondary optimization to update the task distribution. Unmovable modules remain in place, while movable modules are reassigned as dictated by the load balancing optimization. Devices then adjust their loads according to this new strategy, pausing computations only locally to reduce disruptions.

Decentralized Device Communication. In our system, devices communicate in a decentralized, ring-structured manner, where each device sequentially receives, computes, and forwards data to the next device until it reaches the Header again, as illustrated with a solid red line in Figure 4. This efficient communication is facilitated by a message queue utilizing the ROUTER-DEALER pattern, allowing devices to alternate between sending (ROUTER) and receiving (DEALER) roles, which enhances scalability and ensures balanced load distribution. The cycle of data processing involves devices acting first as receivers to perform computations, then as senders to pass on results, maintaining a continuous and organized flow of data throughout the network.

Multi-Threaded Inference. Our system implements multi-threaded inference, allowing parallel processing where each thread independently manages a task and progresses to the next one immediately after completion. Due to the non-threadsafe nature of message queue sockets, we ensure thread safety by using multiple sockets and ports instead of sharing or locking sockets in multi-threads, thereby preventing performance bottlenecks and reducing communication latency. Furthermore, multi-threaded inference boosts CPU efficiency by accommodating varying batch sizes and enabling flexible processing strategies, such as dividing large batches into smaller mini-batches or adjusting batch sizes dynamically, optimizing system performance.

Sequential & Residual Communication. In sequential communication, devices in our system form a circuit where each transmits data to the next in line, creating a flow where only sequential data is exchanged. It leads to inefficiencies as devices pass along residual data not immediately needed by them. To overcome these limitations, we introduced a residual communication strategy, allowing for direct transmission of data to target devices, as depicted by green dashed lines in Figure 4. It reduces unnecessary data carriage and latency.

	Pixel 7 pro	CUBOT X30
SoC	Google Tensor G2	Mediatek MT6771
CPU	Cortex-X1/A78/A55	Cortex-A73/A53
RAM	12GB	8GB
OS	Android 13	Android 10

Table 1: Test Hardware Platforms in Evaluation.

3.5 Implementation and Methodology

LinguaLinked Prototype. We build LinguaLinked atop PyTorch (Paszke et al., 2019), leveraging the `torch.fx` library (Reed et al., 2022) for computational subgraph extraction and PyTorch sub-module compilation, with performance profiling done via Deepspeed (Rasley et al., 2020). These sub-modules are then prepared for mobile deployment by conversion to ONNX format (Bai et al., 2019) and optimized using `int8` precision quantization through ONNXRuntime (Runtime, 2023). Optimization for model distribution and load balancing is achieved with the MILP solver Gurobipy (Gurobi Optimization, LLC, 2023), allowing the deployment of optimized sub-modules on mobile devices through an Android application that utilizes ONNXRuntime's C++ API. For efficient mobile device communication and distributed inference, ZeroMQ (Zer) with a ROUTER-DEALER socket pattern is integrated, enhancing asynchronous communication in the mobile environment.

Chat Application. We develop an Android application that allows users to chat with LLMs in a distributed decentralized way as shown in Figure 3. Our application features two distinct modes: header and worker. The header mode focuses on direct user interaction with the LLM such as sending prompts and receiving responses. In contrast, the worker mode dedicates itself to the heavy lifting of model computation, showcasing the synergy between devices to accomplish LLM inference tasks efficiently.

4 Evaluation

4.1 Evaluation Setup

Hardware. In evaluating the LinguaLinked system, we utilize four mobile devices: three Google Pixel 7 Pros and one CUBOT X30. The specific hardware configurations of these devices are detailed in Table 1. Our analysis focuses on CPU performance, reflecting the system's compatibility with the current CPU-only support of ONNXRuntime for LLMs. This approach is deliberate, anticipating future integration with GPU acceleration capabilities as ONNXRuntime evolves, thereby highlighting our system's adaptability and the consistency of its performance evaluation across varying hardware source.

Evaluation Tasks. Our system's performance is assessed through text generation tasks, using the Wikitext-2 (Merity et al., 2016) with 100 randomly selected samples.

Evaluation Models. Evaluation leverages the BLOOM series LLMs (Workshop et al., 2023), including BLOOM 3b, BLOOM 1.7b, and BLOOM 1.1b models, in both full and `int8` precision formats.

Baseline for Comparison. We established a baseline for assessing on-device distributed inference of LLMs, assigning an equal number of sub-modules (m/n) to each of the n mobile devices, irrespective of their specific hardware or network conditions. This approach allows for a comparison of system throughput between our uniform distribution strategy and both our optimized model assignment and runtime load balancing strategies, in the absence of prior focused research in this area.

Model	Device Config	Device Number	Thread Number	Avg. Time/Token (s)
Baseline				
BLOOM3b-int8	2P+1C	3	1	2.526
BLOOM1.7b-int8	2P+1C	3	1	1.466
BLOOM1.1b-int8	2P+1C	3	1	1.017
BLOOM3b-full	3P+1C	4	1	5.222
BLOOM1.7b-full	3P+1C	4	1	3.159
BLOOM1.1b-full	3P+1C	4	1	1.967
Optimized				
BLOOM3b-int8	2P+1C	3	1	1.576
BLOOM1.7b-int8	2P+1C	3	1	0.944
BLOOM1.1b-int8	2P+1C	3	1	0.653
BLOOM3b-full	3P+1C	4	1	3.944
BLOOM1.7b-full	3P+1C	4	1	2.520
BLOOM1.1b-full	3P+1C	4	1	1.793

Table 2: Inference Throughput Comparison of Baseline and Optimized Strategies for Model Assignment in Heterogeneous Devices. On the Device Config column, P indicates Google Pixel 7pro and C indicates CubotX30.

4.2 Performance

Optimized Model Assignment Performance. In heterogeneous device environments, optimized model assignment significantly enhances inference throughput compared to baseline strategies, as indicated by the data in Table 2. For `int8` quantized models, throughput increases are notable: BLOOM 3b achieves a 1.61$\times$ improvement, while BLOOM

1.7b and 1.1b models see 1.55× and 1.56× enhancements, respectively. Full-precision models also benefit, with BLOOM 3b, 1.7b, and 1.1b models experiencing improvements of 1.32×, 1.25×, and 1.11×, respectively.

The trend is clear—larger models, such as the BLOOM 3b, exhibit greater gains, suggesting that optimization strategies yield more significant benefits for models with higher computational needs. This pattern underscores the efficacy of optimized model assignment in improving the efficiency of model deployment and inference performance in diverse computing landscapes.

Multi-threaded Inference Performance. Our study investigates the benefits of multi-threading on inference throughput for both `int8` quantized and full-precision BLOOM models, focusing on text generation task as indicated by the data in Table 3. Conducted on three Google Pixel 7 Pro devices, we report that quantized models show a marked performance improvement in multi-threaded setups for text generation. Specifically, the BLOOM 3b quantized model's throughput increases by 1.81 × with two threads and 2.52 × with five threads compared to a single-threaded baseline. Similarly, the BLOOM 1.7b and 1.1b models demonstrate significant speed-ups, with the 1.7b model doubling its throughput with two threads and reaching a 2.65× increase with five threads, and the 1.1b model achieving a 1.73 × speed-up with two threads and a 2.3× increase with five threads.

Full-precision models also benefit from multi-threading, albeit to a lesser extent. The BLOOM 3b full-precision model sees a 1.67 × speed-up with two threads and a 1.97 × increase with five threads. The 1.7b and 1.1b models exhibit speed-ups of 1.54 and 1.58 ×, respectively, with two threads, and 1.83 and 1.79 × with five threads.

Quantized Experiment/Int8					
Model	Avg. Compute Time/Token (s)/Thread Number				
Thread Number	1	2	3	4	5
BLOOM3b-int8	1.145	0.634	0.526	0.472	0.455
BLOOM1.7b-int8	0.740	0.389	0.336	0.302	0.279
BLOOM1.1b-int8	0.464	0.269	0.230	0.207	0.202
Full Precision Experiment					
Model	Avg. Compute Time/Token (s)/ Thread Number				
Thread Number	1	2	3	4	5
BLOOM3b-full	2.687	1.611	1.492	1.444	1.368
BLOOM1.7b-full	1.675	1.088	0.972	0.903	0.918
BLOOM1.1b-full	1.105	0.700	0.623	0.632	0.617

Table 3: Multi-threading Throughput for Text Generation on 3 Google Pixel 7 Pro using Optimized Model Assignment Strategy.

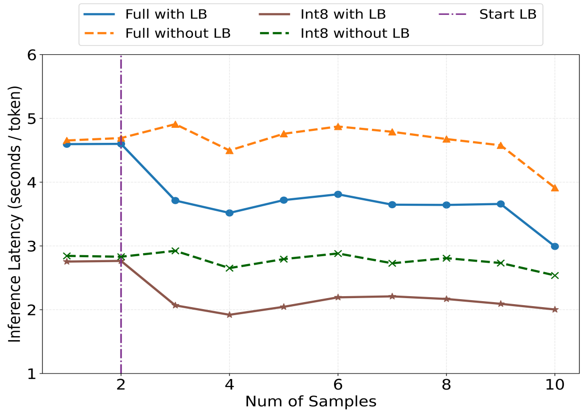

Figure 5: Load Balancer Launched at Runtime.

Micro-Benchmarking the Runtime Load Balancer. In our study, we also investigate the effects of runtime load balancing on the BLOOM 1.7b model in both full precision and `int8` quantized formats across three devices, including two high-end and one low-end phone. Initially, model partitions are unevenly distributed, causing the low-end phone to be overloaded. Upon activating the load balancer after processing two samples and continuing with ten more, we observe a significant improvement in processing efficiency.

Figure 5 illustrates that enabling load balancing from the second sample noticeably decreases inference latency. For the full precision model, enabling load balancing cuts down latency from 4.624 seconds to 3.587 seconds per token, showcasing an improvement by reallocating 8 sub-modules and overcoming a notable overhead from reloading sessions onto alternate devices.

For the quantized model, activating the load balancer reduces latency from 2.756 to 2.087 seconds per token, with a less substantial reloading overhead compared to the full precision model due to the smaller size of quantized sub-modules.

Inference times for the quantized model prove to be more stable across generation tasks than for the full precision model.

Overall, the implementation of runtime load balancing results in an average improvement of 30% in acceleration, effectively demonstrating the benefits of dynamically adjusting workloads among devices with varying processing capabilities.

4.3 Sequential and Residual Communication

To demonstrate the efficacy of residual over sequential communication, we conducted experiments measuring communication times, employing the Network Temporal Protocol (NTP) for precise synchronization and utilizing the system clock for nanosecond accuracy in runtime measurement.

By comparing these methods using three low-end smartphones and the quantized BLOOM 3b model, recording average delays from ten trials. The results, detailed in Table 4 and Figure 4, show that sequential communication incurs higher delays due to multiple transmission steps and the need for activations to be processed before residual data can be sent. Sequential transmission involved hops with an average data size of 1.5 MB, while residual communication transmitted about 15 KB directly, allowing it to operate in parallel and more efficiently than the sequential method. Note that as the number of residual connections and the size of the model increase, the saved time will likewise increase.

	Hop1	Hop2	Hop3	Total	Res Hop
Seq	0.2489s	0.2580s	0.0770s	0.5839s	——
Res	0.2347s	0.2463s	0.0779s	0.5589s	0.0111s

Table 4: Residual and Sequential Communication Performance

5 Discussion

A major direction for expanding LinguaLinked involves adapting it for distributed fine-tuning on mobile devices, allowing model customization based on user interactions and local data, paving the way for personalized AI applications while preserving data privacy. We also envision extending LinguaLinked to handle multi-modality models, enhancing its applicability in diverse real-world scenarios.

To further improve LinguaLinked, we envision more advanced model computational graph partitioning strategies involving further optimizations on task divisions better aligned with device capabilities. Moreover, integrating advanced load balancing algorithms that account for not only computational capabilities but also battery life and user engagement patterns will ensure a holistic approach to distributed computing on mobile platforms.

Finally, the implications of this research are significant for AI policy, as it challenges the prevailing reliance on cloud infrastructure and centralized data centers for AI deployment. By demonstrating the potential to deploy AI systems from a network of mobile devices, our work suggests a paradigm where the 'means of production' for AI can be decentralized and localized. This model of deployment could lead to a future where AI systems are both operated and fine-tuned locally using a diverse array of small devices. Such a setup could make AI systems more difficult to regulate, as the distribution and localization of AI technologies allow for widespread, generic hardware use.

6 Conclusion

In this work, we introduce LinguaLinked, a system for decentralized LLM inference on mobile devices. To the best of our knowledge, LinguaLinked is the first work that exploits deploying LLM distributively on mobile devices. LinguaLinked implemented optimized model assignment strategy, network communication and runtime load balancing mechanism to accelerate the distributed LLM inference on mobile devices. This approach tackles the complexities of deploying both full precision and quantized LLMs of various sizes within mobile computing environments.

7 Limitations

Our results demonstrate promising advancements in distributed LLM inference on mobile devices but also underscore several limitations. Key among these are the overheads from load balancing and constraints of current hardware and software frameworks. As tools like ONNXRuntime evolve to support GPU acceleration, we expect significant enhancements in LinguaLinked's performance. Furthermore, exploring advanced quantization techniques and communication mechanisms could lead to more efficient distributed inference systems.

At the same time, a critical focus for future iterations of LinguaLinked is energy efficiency. We find that the continuous intensive inference tasks, especially with full-precision models, significantly drain battery life and cause overheating, leading to performance degradation. To address this, we aim to incorporate energy-efficient computing strategies that balance computational demands with energy consumption and thermal management. This could include adaptive algorithms to modulate computational load based on the device's energy state, and hardware-specific optimizations leveraging low-power processing cores for specific tasks.

Furthermore, when designing our system with privacy in mind, we primarily contrast it with traditional server-based inference systems, operating under the assumption that all inference tasks are conducted locally on 'trusted' mobile devices. This setup should inherently protect user privacy. However, we recognize potential scenarios where this assumption may fail. For instance, if a device becomes compromised or if unauthorized access is obtained, sensitive local data could be at risk. Moreover, our trust model does not address potential side-channel attacks, which could allow attackers to derive sensitive information from the model's intermediate activations. These vulnerabilities underscore the need for more comprehensive, multi-layered security protocols that extend beyond simple device trust, aiming to robustly safeguard user data in diverse and adversarial environments.

Finally, due to the absence of standardized evaluation benchmarks for distributed LLM inference on mobile devices, we have created our own baselines to assess our system's performance. However, the lack of universally accepted benchmarks and previous research in this domain complicates the task of conducting thorough comparisons for future work. It is crucial for future research to focus on developing standardized benchmarks in this field. Establishing such benchmarks would facilitate more uniform comparisons between different systems, enhance the clarity of potential improvements, and identify the most effective strategies for distributed LLM inference on mobile platforms.

References

Zeromq. https://zeromq.org/. (Accessed on 11/29/2023).

Ibrahim Alabdulmohsin, Behnam Neyshabur, and Xiaohua Zhai. 2022. Revisiting neural scaling laws in language and vision.

Reza Yazdani Aminabadi, Samyam Rajbhandari, Ammar Ahmad Awan, Cheng Li, Du Li, Elton Zheng, Olatunji Ruwase, Shaden Smith, Minjia Zhang, Jeff Rasley, et al. 2022. Deepspeed-inference: enabling efficient inference of transformer models at unprecedented scale. In *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15. IEEE.

Junjie Bai, Fang Lu, Ke Zhang, et al. 2019. Onnx: Open neural network exchange. https://github.com/onnx/onnx.

Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. 2020. What is the state of neural network pruning? *Proceedings of machine learning and systems*, 2:129–146.

Yelysei Bondarenko, Markus Nagel, and Tijmen Blankevoort. 2021. Understanding and overcoming the challenges of efficient transformer quantization. *arXiv preprint arXiv:2109.12948*.

Alexander Borzunov, Dmitry Baranchuk, Tim Dettmers, Max Ryabinin, Younes Belkada, Artem Chumachenko, Pavel Samygin, and Colin Raffel. 2022. Petals: Collaborative inference and fine-tuning of large models. *arXiv preprint arXiv:2209.01188*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Jiasi Chen and Xukan Ran. 2019. Deep learning with edge computing: A review. *Proceedings of the IEEE*, 107(8):1655–1674.

Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Meghan Cowan, Haichen Shen, Leyuan Wang, Yuwei Hu, Luis Ceze, Carlos Guestrin, and Arvind Krishnamurthy. 2018. Tvm: An automated end-to-end optimizing compiler for deep learning.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways.

Aidan Clark, Diego de Las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake A. Hechtman, Trevor Cai, Sebastian Borgeaud, George van den Driessche, Eliza Rutherford, T. W. Hennigan, Matthew G. Johnson, Katie Millican, Albin Cassirer, Chris Jones, Elena Buchatskaya, David Budden, L. Sifre, Simon Osindero, Oriol Vinyals, Jack W. Rae, Erich Elsen, Koray Kavukcuoglu, and Karen Simonyan. 2022. Unified scaling laws for routed language models. In *International Conference on Machine Learning*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems.

Claudionor N Coelho, Aki Kuusela, Shan Li, Hao Zhuang, Jennifer Ngadiuba, Thea Klaeboe Aarrestad, Vladimir Loncar, Maurizio Pierini, Adrian Alan Pol, and Sioni Summers. 2021. Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors. *Nature Machine Intelligence*, 3(8):675–686.

Jiangsu Du, Jiazhi Jiang, Jiang Zheng, Hongbin Zhang, Dan Huang, and Yutong Lu. 2023. Improving computation and memory efficiency for real-world transformer inference on gpus. *ACM Transactions on Architecture and Code Optimization*, 20(4):1–22.

Mohammed S Elbamby, Cristina Perfecto, Chen-Feng Liu, Jihong Park, Sumudu Samarakoon, Xianfu Chen, and Mehdi Bennis. 2019. Wireless edge computing with latency and reliability guarantees. *Proceedings of the IEEE*, 107(8):1717–1737.

Luciano Floridi and Massimo Chiriatti. 2020. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30:1–14.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*.

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. 2022. A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*, pages 291–326. Chapman and Hall/CRC.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2023. Knowledge distillation of large language models. *arXiv preprint arXiv:2306.08543*.

Jashwant Raj Gunasekaran, Cyan Subhra Mishra, Prashanth Thinakaran, Bikash Sharma, Mahmut Taylan Kandemir, and Chita R Das. 2022. Cocktail: A multidimensional optimization for model serving in cloud. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 1041–1057.

Gurobi Optimization, LLC. 2023. Gurobi Optimizer Reference Manual.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding.

Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. 2021. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *The Journal of Machine Learning Research*, 22(1):10882–11005.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training compute-optimal large language models.

Chenghao Hu and Baochun Li. 2022. Distributed inference with deep learning models across heterogeneous edge devices. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, pages 330–339. IEEE.

Zhiming Hu, Ahmad Bisher Tarakji, Vishal Raheja, Caleb Phillips, Teng Wang, and Iqbal Mohomed. 2019. Deephome: Distributed inference with heterogeneous devices in the edge. In *The 3rd International Workshop on Deep Learning for Mobile Systems and Applications*, pages 13–18.

Chien-Chin Huang, Gu Jin, and Jinyang Li. 2020. Swapadvisor: Pushing deep learning beyond the gpu memory limit via smart swapping. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '20, page 1341–1355, New York, NY, USA. Association for Computing Machinery.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models.

Sunder Ali Khowaja, Parus Khuwaja, and Kapal Dev. 2023. Chatgpt needs spade (sustainability, privacy, digital divide, and ethics) evaluation: A review. *arXiv preprint arXiv:2305.03123*.

Nir Kshetri. 2023. Cybercrime and privacy threats of large language models. *IT Professional*, 25(3):9–13.

Minhyeok Lee. 2023. A mathematical interpretation of autoregressive generative pre-trained transformer and self-supervised learning. *Mathematics*, 11(11).

Fan Liang, Wei Yu, Xing Liu, David Griffith, and Nada Golmie. 2020a. Toward edge-based deep learning in industrial internet of things. *IEEE Internet of Things Journal*, 7(5):4329–4341.

Kevin J Liang, Weituo Hao, Dinghan Shen, Yufan Zhou, Weizhu Chen, Changyou Chen, and Lawrence Carin. 2020b. Mixkd: Towards efficient distillation of large-scale language models. *arXiv preprint arXiv:2011.00593*.

Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. 2021. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, 461:370–403.

Chu-Cheng Lin, Aaron Jaech, Xin Li, Matthew R. Gormley, and Jason Eisner. 2021. Limitations of autoregressive models and their alternatives.

Jiachen Mao, Xiang Chen, Kent W Nixon, Christopher Krieger, and Yiran Chen. 2017a. Modnn: Local distributed mobile computing system for deep neural network. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pages 1396–1401. IEEE.

Yuyi Mao, Changsheng You, Jun Zhang, Kaibin Huang, and Khaled B Letaief. 2017b. A survey on mobile edge computing: The communication perspective. *IEEE communications surveys & tutorials*, 19(4):2322–2358.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models.

Soumyalatha Naveen, Manjunath R Kounte, and Mohammed Riyaz Ahmed. 2021. Low latency deep learning inference model for distributed intelligent iot edge clusters. *IEEE Access*, 9:160607–160621.

Jihong Park, Sumudu Samarakoon, Mehdi Bennis, and Mérouane Debbah. 2019. Wireless network intelligence at the edge. *Proceedings of the IEEE*, 107(11):2204–2239.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library.

Roma Patel and Ellie Pavlick. 2022. Mapping language models to grounded conceptual spaces. In *International Conference on Learning Representations*.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506.

James K. Reed, Zachary DeVito, Horace He, Ansley Ussery, and Jason Ansel. 2022. Torch.fx: Practical program capture and transformation for deep learning in python.

Karen Renaud, Merrill Warkentin, and George Westerman. 2023. *From ChatGPT to HackGPT: Meeting the Cybersecurity Threat of Generative AI*. MIT Sloan Management Review.

Francisco Romero, Qian Li, Neeraja J Yadwadkar, and Christos Kozyrakis. 2021. {INFaaS}: Automated model-less inference serving. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 397–411.

ONNX Runtime. 2023. Onnx runtime. Available from: `https://onnxruntime.ai/`.

Glorin Sebastian. 2023. Do chatgpt and other ai chatbots pose a cybersecurity risk?: An exploratory study. *International Journal of Security and Privacy in Pervasive Computing (IJSPPC)*, 15(1):1–11.

TensorFlow. 2023. Tensorflow lite. Available from: `https://www.tensorflow.org/lite`.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need.

Yiding Wang, Kai Chen, Haisheng Tan, and Kun Guo. 2023. Tabi: An efficient multi-level inference system for large language models. In *Proceedings of the Eighteenth European Conference on Computer Systems*, pages 233–248.

BigScience Workshop, :, Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurençon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu, Chenghao Mou, Chris Emezue, Christopher Klamm, Colin Leong, Daniel van Strien, David Ifeoluwa Adelani, Dragomir Radev, Eduardo González Ponferrada, Efrat Levkovizh, Ethan Kim, Eyal Bar Natan, Francesco De Toni, Gérard Dupont, Germán Kruszewski, Giada Pistilli, Hady Elsahar, Hamza Benyamina, Hieu Tran, Ian Yu, Idris Abdulmumin, Isaac Johnson, Itziar Gonzalez-Dios, Javier de la Rosa, Jenny Chim, Jesse Dodge, Jian Zhu, Jonathan Chang, Jörg Frohberg, Joseph Tobing, Joydeep Bhattacharjee, Khalid Almubarak, Kimbo Chen, Kyle Lo, Leandro Von Werra, Leon Weber, Long Phan, Loubna Ben allal, Ludovic Tanguy, Manan Dey, Manuel Romero Muñoz, Maraim Masoud, María Grandury, Mario Šaško, Max Huang, Maximin Coavoux, Mayank Singh, Mike Tian-Jian Jiang, Minh Chien Vu, Mohammad A. Jauhar, Mustafa Ghaleb, Nishant Subramani, Nora Kassner, Nurulaqilla Khamis, Olivier Nguyen, Omar Espejel, Ona de Gibert, Paulo Villegas, Peter Henderson, Pierre Colombo, Priscilla Amuok, Quentin Lhoest, Rheza Harliman, Rishi Bommasani, Roberto Luis López, Rui Ribeiro, Salomey Osei, Sampo Pyysalo, Sebastian Nagel, Shamik Bose, Shamsuddeen Hassan Muhammad, Shanya Sharma, Shayne Longpre, Somaieh Nikpoor, Stanislav Silberberg, Suhas Pai, Sydney Zink, Tiago Timponi Torrent, Timo Schick, Tristan Thrush, Valentin Danchev, Vassilina Nikoulina, Veronika Laippala, Violette Lepercq, Vrinda Prabhu, Zaid Alyafeai, Zeerak Talat, Arun Raja, Benjamin Heinzerling, Chenglei Si, Davut Emre Taşar, Elizabeth Salesky, Sabrina J. Mielke, Wilson Y. Lee, Abheesht Sharma, Andrea Santilli, Antoine Chaffin, Arnaud Stiegler, Debajyoti Datta, Eliza Szczechla, Gunjan Chhablani, Han Wang, Harshit Pandey, Hendrik Strobelt, Jason Alan Fries, Jos Rozen, Leo Gao, Lintang Sutawika, M Saiful Bari, Maged S. Al-shaibani, Matteo Manica, Nihal Nayak, Ryan Teehan, Samuel Albanie, Sheng Shen, Srulik Ben-David, Stephen H. Bach, Taewoon Kim, Tali Bers, Thibault Fevry, Trishala Neeraj, Urmish Thakker, Vikas Raunak, Xiangru Tang, Zheng-Xin Yong, Zhiqing Sun, Shaked Brody, Yallow Uri, Hadar Tojarieh, Adam Roberts, Hyung Won Chung, Jaesung Tae, Jason Phang, Ofir Press, Conglong Li, Deepak Narayanan, Hatim Bourfoune, Jared Casper, Jeff Rasley, Max Ryabinin, Mayank Mishra, Minjia Zhang, Mohammad Shoeybi, Myriam Peyrounette, Nicolas Patry, Nouamane Tazi, Omar Sanseviero, Patrick von Platen, Pierre Cornette, Pierre François Lavallée, Rémi Lacroix, Samyam Rajbhandari, Sanchit Gandhi, Shaden Smith, Stéphane Requena, Suraj Patil, Tim Dettmers, Ahmed Baruwa, Amanpreet Singh, Anastasia Cheveleva, Anne-Laure Ligozat, Arjun Subramonian, Aurélie Névéol, Charles Lovering, Dan Garrette, Deepak Tunuguntla, Ehud Reiter, Ekaterina Taktasheva, Ekaterina Voloshina, Eli Bogdanov, Genta Indra Winata, Hailey Schoelkopf, Jan-Christoph Kalo, Jekaterina Novikova, Jessica Zosa Forde, Jordan Clive, Jungo Kasai, Ken Kawamura, Liam Hazan, Marine Carpuat, Miruna Clinciu, Najoung Kim, Newton Cheng, Oleg Serikov, Omer Antverg, Oskar van der Wal, Rui Zhang, Ruochen Zhang, Sebastian Gehrmann, Shachar Mirkin, Shani Pais, Tatiana Shavrina, Thomas Scialom, Tian Yun, Tomasz Limisiewicz, Verena Rieser, Vitaly Protasov, Vladislav Mikhailov, Yada Pruksachatkun, Yonatan Belinkov, Zachary Bamberger, Zdeněk Kasner, Alice Rueda, Amanda Pestana, Amir Feizpour, Ammar Khan, Amy Faranak, Ana Santos, Anthony Hevia, Antigona Unldreaj, Arash Aghagol, Arezoo Abdollahi, Aycha Tammour, Azadeh HajiHosseini, Bahareh Behroozi, Benjamin Ajibade, Bharat Saxena, Carlos Muñoz Ferrandis, Daniel McDuff, Danish Contractor, David Lansky, Davis David, Douwe Kiela, Duong A. Nguyen, Edward Tan, Emi Baylor, Ezinwanne Ozoani, Fatima Mirza, Frankline Ononiwu, Habib Rezanejad, Hessie Jones, Indrani Bhattacharya, Irene Solaiman, Irina Sedenko, Isar Nejadgholi, Jesse Passmore, Josh Seltzer, Julio Bonis Sanz, Livia Dutra, Mairon Samagaio, Maraim Elbadri, Margot Mieskes, Marissa Gerchick, Martha Akinlolu, Michael McKenna, Mike Qiu, Muhammed Ghauri, Mykola Burynok, Nafis Abrar, Nazneen Rajani, Nour Elkott, Nour Fahmy, Olanrewaju Samuel, Ran An, Rasmus Kromann, Ryan Hao, Samira Alizadeh, Sarmad Shubber, Silas Wang, Sourav Roy, Sylvain Viguier, Thanh Le, Tobi Oyebade, Trieu Le, Yoyo Yang, Zach Nguyen, Abhinav Ramesh Kashyap, Alfredo Palasciano, Alison Callahan, Anima Shukla, Antonio Miranda-Escalada, Ayush Singh, Benjamin Beilharz, Bo Wang, Caio Brito, Chenxi Zhou, Chirag Jain, Chuxin Xu, Clémentine Fourrier, Daniel León Periñán, Daniel Molano, Dian Yu, Enrique Manjavacas, Fabio Barth, Florian Fuhrimann, Gabriel Altay, Giyaseddin Bayrak, Gully Burns, Helena U. Vrabec, Imane Bello, Ishani Dash, Jihyun Kang, John Giorgi, Jonas Golde, Jose David Posada, Karthik Rangasai Sivaraman, Lokesh Bulchandani, Lu Liu, Luisa Shinzato, Madeleine Hahn de Bykhovetz, Maiko Takeuchi, Marc Pàmies, Maria A Castillo, Marianna Nezhurina, Mario Sänger, Matthias Samwald, Michael Cullan, Michael Weinberg, Michiel De Wolf, Mina Mihaljcic, Minna Liu, Moritz Freidank, Myungsun Kang, Natasha Seelam, Nathan Dahlberg, Nicholas Michio Broad, Nikolaus Muellner, Pascale Fung, Patrick Haller, Ramya Chandrasekhar, Renata Eisenberg, Robert Martin, Rodrigo Canalli, Rosaline Su, Ruisi Su, Samuel Cahyawijaya, Samuele Garda, Shlok S Deshmukh, Shubhanshu Mishra, Sid Kiblawi, Simon Ott, Sinee Sang-aroonsiri, Srishti Kumar, Stefan Schweter, Sushil Bharati, Tanmay Laud, Théo Gigant, Tomoya Kainuma, Wojciech Kusa, Yanis Labrak, Yash Shailesh Bajaj, Yash Venkatraman, Yifan Xu, Yingxin Xu, Yu Xu, Zhe Tan, Zhongli Xie, Zifan Ye, Mathilde Bras, Younes Belkada, and Thomas Wolf. 2023. Bloom: A 176b-parameter open-access multilingual language model.

Carole-Jean Wu, David Brooks, Kevin Chen, Douglas Chen, Sy Choudhury, Marat Dukhan, Kim Hazelwood, Eldad Isaac, Yangqing Jia, Bill Jia, et al. 2019. Machine learning at facebook: Understanding inference at the edge. In *2019 IEEE international symposium on high performance computer architecture (HPCA)*, pages 331–344. IEEE.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR.

Yuzhe Xu, Thaha Mohammed, Mario Di Francesco, and Carlo Fischione. 2022. Distributed assignment with load balancing for dnn inference at the edge. *IEEE Internet of Things Journal*, 10(2):1053–1065.

Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. 2022.

Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 35:27168–27183.

Liekang Zeng, Xu Chen, Zhi Zhou, Lei Yang, and Junshan Zhang. 2021. Coedge: Cooperative dnn inference with adaptive workload partitioning over heterogeneous edge devices. *IEEE/ACM Trans. Netw.*, 29(2):595–608.

Chaoyun Zhang, Paul Patras, and Hamed Haddadi. 2019. Deep learning in mobile and wireless networking: A survey. *IEEE Communications surveys & tutorials*, 21(3):2224–2287.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models.

Tianming Zhao, Yucheng Xie, Yan Wang, Jerry Cheng, Xiaonan Guo, Bin Hu, and Yingying Chen. 2022. A survey of deep learning on mobile devices: Applications, optimizations, challenges, and research opportunities. *Proceedings of the IEEE*, 110(3):334–354.

Li Zhou, Mohammad Hossein Samavatian, Anys Bacha, Saikat Majumdar, and Radu Teodorescu. 2019. Adaptive parallel execution of deep neural networks on heterogeneous edge devices. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, SEC '19, page 195–208, New York, NY, USA. Association for Computing Machinery.