Mobile Foundation Model as Firmware
The Way Towards a Unified Mobile AI Landscape

Jinliang Yuan†, Chen Yang†, Dongqi Cai†,
Shihe Wang, Xin Yuan, Zeling Zhang, Xiang Li, Dingge Zhang, Hanzi Mei,
Xianqing Jia, Shangguang Wang, Mengwei Xu*
Beijing University of Posts and Telecommunications
Beijing, China

ABSTRACT

In the current AI era, mobile devices such as smartphones are tasked with executing a myriad of deep neural networks (DNNs) locally. It presents a complex landscape, as these models are highly fragmented in terms of architecture, operators, and implementations. Such fragmentation poses significant challenges to the co-optimization of hardware, systems, and algorithms for efficient and scalable mobile AI.

Inspired by the recent groundbreaking progress in large foundation models, this work introduces a novel paradigm for mobile AI, where mobile OS and hardware jointly manage a foundation model that is capable of serving a wide array of mobile AI tasks. This foundation model functions akin to firmware, unmodifiable by apps or the OS, exposed as a system service to Apps. They can invoke this foundation model through a small, offline fine-tuned "adapter" for various downstream tasks. We propose a tangible design of this vision called `M4`, and prototype it from publicly available pre-trained models. To assess its capability, we also build a comprehensive benchmark consisting of 38 mobile AI tasks and 50 datasets, spanning 5 multimodal inputs. Extensive experiments demonstrate `M4`'s remarkable results: it achieves comparable accuracy in 85% of tasks, offers enhanced scalability regarding storage and memory, and has much simpler operations. In broader terms, this work paves a new way towards efficient and scalable mobile AI in the post-LLM era.

†Authors contributed equally to this research. *Corresponding author.

ACM MobiCom '24, September 30–October 4, 2024, Washington D.C., DC, USA

ACM ISBN 979-8-4007-0489-5/24/09…$15.00
https://doi.org/10.1145/3636534.3649361

CCS CONCEPTS

• **Human-centered computing** → *Ubiquitous and mobile computing systems and tools.*

KEYWORDS

Mobile computing, multimodal foundation model, efficient and scalable mobile AI

ACM Reference Format:
Jinliang Yuan†, Chen Yang†, Dongqi Cai†,, Shihe Wang, Xin Yuan, Zeling Zhang, Xiang Li, Dingge Zhang, Hanzi Mei, Xianqing Jia, Shangguang Wang, Mengwei Xu*. 2024. Mobile Foundation Model as Firmware The Way Towards a Unified Mobile AI Landscape. In *International Conference On Mobile Computing And Networking (ACM MobiCom '24), September 30–October 4, 2024, Washington D.C., DC, USA.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3636534.3649361

1 INTRODUCTION

Machine learning is revolutionizing mobile applications by facilitating a more automated, intelligent, and efficient interaction between users and devices. These advancements enable humans to enjoy the convenience provided by deep models at all times and locations, from voice assistants [9, 61], image editers [112, 124, 129], to augmented reality [100, 148]. As reported in [39, 130], the number of deep models incorporated within individual devices is growing rapidly, making mobile devices a primary vehicle for AI.

Executing deep models on devices offers benefits in data privacy and service availability but also demands significant resources such as memory, energy, and time. For efficient and scalable on-device execution of these models, a comprehensive co-design approach that integrates hardware, system, and algorithms is needed. However, this task is challenged by the *fragmented ecosystem* of mobile deep models: they significantly differ in architecture, operators, and implementations [46, 56, 68, 77, 113, 135]. This fragmentation, which often results in ad-hoc optimization efforts [40, 99, 107], seems unavoidable. It originates from the complex nature of mobile AI tasks (CV/NLP/TTS/HAR/..), multimodal data from various sensors (camera, screen, microphone, etc.), and diverse application demands (high accuracy, low latency, etc.).

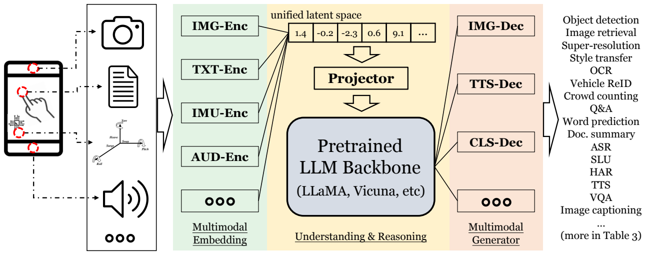


Figure 1: An overview of M4.

Such fragmentation fundamentally undermines the efficiency of constructing an efficient and scalable mobile AI stack, notably in the three following aspects:

- *Hardware aspect:* it complicates the design of ASIC-based accelerators (NPUs), by forcing difficult trade-offs between generality and performance. §2.2 shows that mobile NPUs can achieve up to a 22× speedup over multi-core CPUs on the Qualcomm Snapdragon 8+ Gen 1 platform. However, this advantage only extends to a small fraction (around 8%) of deep models due to the lack of operator support.
- *OS aspect:* It hampers the system-wise sharing of weights and computations across different applications. Mobile apps often perform similar meta-ML tasks (e.g., object detection for augmented reality, image enhancement, OCR apps), and there exist temporal, spatial, and semantic correlations among the input data [62]. However, exploiting such similarities to reduce memory or computing via cache-reuse is currently impractical, due to the model fragmentation and OSes' lack of visibility into those models managed at the application level.
- *Software aspect:* It makes library-level optimizations adhoc. As noted in [144], there is a wide array of frameworks available to developers, but their performance can vary significantly across different models and devices. No single solution excels universally, often leaving developers struggling to differentiate between them.

Mobile foundation model as firmware. In order to fundamentally tackle the aforementioned issues, we propose a novel paradigm for mobile AI in which the OS and hardware co-manage a foundation model that is capable of addressing most, if not all, mobile AI tasks. This model, akin to firmware, is exposed as a system service to applications similar to the unified ML interface NNAPI [60] in Android. It remains unaltered by apps or the OS. To utilize it, each application embeds it a lightweight "adapter" that is fine-tuned offline for downstream tasks. This approach could greatly simplify NPU design and allow the OS to take control of AI computing across applications, thereby facilitating the sharing of weights, computations, and task scheduling. This vision becomes feasible thanks to recent advancements in the ML community, specifically: (1) The establishment of pre-trained foundation models [92, 105, 117] that capture extensive knowledge from vast Internet data; (2) The development of algorithms to accurately align multimodal data input [57, 114]; (3) The demonstration of parameter-efficient fine-tuning (PEFT) methods like LoRA [65, 89] that efficiently adapt pre-trained models to diverse downstream tasks.

While the vision is intriguing, there are two key missing pieces to turn it into reality. (1) How to build such a one-size-fits-all foundation model to ubiquitously handle highly diversified, multimodal mobile AI tasks? While research on the multimodal foundation model has achieved impressive progress in recent years, they are still not adequate in our case: most of them [80, 91, 101] handle only a small fixed number of input/output modalities (e.g., text-image) and cannot be flexibly adapted to more; a concurrent effort CoDi [114] with this work enables any-to-any generation across three modalities (image-text-audio), but requires more than 34GBs of on-device storage/memory. (2) How to properly evaluate the performance of the proposed foundation model? To our best knowledge, there has been no comprehensive benchmark or a set of standard metrics for mobile AI tasks.

M4: a composable mobile foundation model (§3). We introduce M4, the first architectural design and implementation of a multimodal mobile foundation model, as illustrated in Figure 1. Unlike prior approaches like CoDi that directly use (Nx) heavy encoders to align multimodal input data and (Mx) heavy decoders to generate specific data format, M4 adds a backbone module in between (a "narrow waist") that comprehends and reasons for each downstream task. Through such "N-1-M" design, M4 is able to achieve better accuracy-to-parameter efficiency as compared to traditional "N-M" architecture. Moreover, M4 could be partially activated by various tasks based on their characteristics (input/output modality, the need for complex comprehension, etc.). We have fully prototyped M4 with only pre-trained models publicly available from HuggingFace [126], which guarantees the reproducibility of M4 and also demonstrates its compatibility with the existing LLM ecosystem. Overall, M4 contains 9.2B parameters and demands 7.5GBs of peak memory footprint. Such a size is only affordable on high-end mobile devices nowadays, but we deem it to be soon feasible for more commons whose memory/storage capacity is significantly increasing yearly.

eAIBench: a comprehensive edge-oriented AI benchmark (§4.1). To assess M4 and future endeavors, we have constructed the first comprehensive benchmark for diverse mobile AI tasks, named eAIBench. Through an extensive examination of real-world mobile AI and publications in mobile venues, M4 presently includes 38 important mobile AI tasks and 50 classic datasets. The tasks include five different input/output data modalities (vision, text, audio, IMU, and mix). Each task is also linked with a task-specific model, representative of the DNN in the pre-LLM era (e.g., ResNet-152 for image classification [63] and LSTM for input token prediction [67]). We also standardize a set of key metrics to quantify the capability of a foundation model.

Key results (§4). We then conduct extensive experiments to evaluate M4 using eAIBench on three kinds of hardware platforms: NVIDIA A100 GPU, NVIDIA Jetson ORIN NX and Pixel 7 Pro smartphone. We summarize our major results.

• ***Ubiquity***– M4 *effectively supports most tasks and datasets in* eAIBench. Compared with the models tailored for each task, M4 shows comparable accuracy on 85% of the 50 datasets and a significant improvement on 4 of them (including image captioning and text-to-image retrieval). In only six instances does M4 experience nontrivial accuracy degradation, marked by a greater than 10% gap. The system also demonstrates promising zero-shot and few-shot capabilities, achieving usable accuracy on certain tasks without any fine-tuning. Moreover, quantization minimally affects the performance of M4: when reduced to 8 bits on two tested tasks, accuracy degradation ranges only between 0.2% and 0.8%. To be noted, the backbone LLM used in the current prototype of M4, i.e., LLaMA (Feb. 2023), has been defeated by many other open-source LLMs since its release, such as LLaMA-2 (July. 2023) and Mistral-7B [71] (Oct. 2023). We expect the performance of M4 to improve substantially as well by using such more powerful backbone LLMs. This is also confirmed by our preliminary experiments by replacing LLaMA with LLaMA-2 on two tasks, as will be discussed in §4.2.

• ***Scalability*** – *Despite* M4 *foundation model's heavier footprint, its adaptation to downstream mobile tasks is lightweight and therefore more scalable.* The current implementation of M4 encompasses ~10 billion parameters, in contrast to the mere 1 million to 500 million parameters found in task-specific models. Nevertheless, the "adapters" of M4 require only 1,000 to 10 million parameters, which enhances scalability across various mobile AI tasks, given that the foundation model is shared. For example, on a device with 12GB of memory, M4 (4-bit quantized) with all 50 adapters can be hosted in memory, eliminating cold-start latency, whereas only 20 of 50 task-specific models would fit the same memory constraints.

• ***Velocity*** – M4 *is much slower than task-specific models, yet the gap might be mitigated through a highly-optimized NPU.* On a high-end autonomous board Jetson Orin NX (16GB memory), M4 runs 18× slower on average. We also test the performance of M4 on smartphone CPUs [1] , which shows that the prediction delay could be too slow, i.e., 2.1 secs to classify an image or 240 msecs to generate a token in QA. However, such a performance degradation might be addressed by running M4 on a highly optimized NPU, since existing NPUs already offer up to a 22-fold speedup over CPUs, as mentioned in §2.2.

[1]Currently, M4 cannot run on COTS smartphones GPU/NPU due to the lack of operator support.

• ***Simplicity*** – M4 *requires fewer operators for execution, greatly simplifying hardware design.* In the ONNX format, M4 utilizes a mere 39 different mathematical operators, in contrast to the cumulative 156 operators required by 50 task-specific models. More impressively, M4 can expand its capabilities using the same number of operators. The traditional approach, on the other hand, continuously introduces new operators [139, 149], thereby complicating NPU design.

In addition to conventional mobile AI tasks, M4 also enables more complex and innovative mobile applications, e.g., a sophisticated assistant capable of processing multimodal input data, understanding user intentions, and responding with precision as demonstrated in §4.7.

Contributions Major contributions are summarized below.

- We delineate a vision for a mobile foundation model, harnessing cutting-edge machine learning techniques to consolidate the mobile AI ecosystem and foster integrated hardware-system co-design.
- We design and prototype the first mobile foundation model with public, pre-trained LLMs.
- We have constructed the first comprehensive edge-oriented AI benchmark, through which our prototype demonstrates significant potential in catering to widespread mobile AI tasks, while exhibiting strong scalability, flexibility, and velocity in its performance.

Open-source M4 and eAIBench are publicly available at https://github.com/UbiquitousLearning/MobileFM.

2 BACKGROUND AND MOTIVATION

2.1 Mobile AI Characteristics

Mobile AI is pervasive. An important trend of AI deployment is the migration of deep learning inference tasks from data centers to smartphones, aiming to minimize user-perceived latency and better preserve data privacy [45, 49]. For instance, it is reported that Android apps embedded with on-device DNNs on the Google Play market have experienced a remarkable 60% growth from February 2020 to April 2021 [39]; Such DL-enhanced apps have been downloaded by users billions of times. Unsurprisingly, mobile devices like smartphones and laptops have become a major carriers of intelligence, where DNN inference happens frequently anywhere anytime even without users being aware of it.

Mobile DNNs are fragmented. Unlike cloud AI where each computing unit (e.g., an NVIDIA GPU) only serves one model for user requests [55, 90, 106], a mobile device needs to handle highly diversified mobile AI tasks by itself. Such diversification is inevitable since mobile AI tasks could

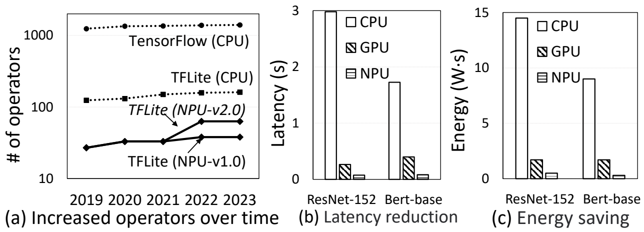


Figure 2: Empirical study on mobile NN processors: (1) A longitudinal analysis of operator support on CPU/NPU; (2) The performance gap between NPU and CPU/GPU on Google Pixel 7 Pro.

leverage multimodal sensor data from devices, including imagery data from cameras, audio data from microphones, IMU data from motion sensors, and textual/code data from users typing. Each modality itself has a wide spectrum of applications, e.g., Google Translate for NLP and Apple Siri for Audio. Meanwhile, there are a wide range of cross-modal applications in mobile scenarios: visual question answering [146], image captioning [48], and multimedia content retrieval [86]. Research suggests that the quantity of multimodal applications on mobile devices has almost doubled in the last two years due to rapid advancements in multimodal technologies [138]. A recent empirical in-the-wild study [39] reported significant heterogeneity in the architectures and internal operators of DNNs handling various modal data.

2.2 A Dilemma of Mobile NPU

Fragmented DNNs significantly strain mobile AI stack hardware, system, and library design, as discussed in Section 1. Here, we emphasize the challenges faced by mobile NPUs specifically. Pilot measurements are conducted to assess mobile NPUs' performance gains compared to mobile CPU/GPU when executing typical mobile DNNs.

The DNN operator support of mobile NPU is significantly lagged behind general-purpose processors. We conducted an investigation on the number of supported NN operators by TensorFlow and TFLite. As illustrated in Figure 2 (a), we have two key observations: (1) The number of NN operator types is still increasing noticeably lately, e.g., from 1240 to 1399 as supported by TensorFlow from 2019 to 2023. Such evolvement of DNN architecture poses significant challenges in designing ubiquitous and efficient mobile NPU design. (2) Mobile chips, especially its NPU, support only a small portion of existing NN operators. TFLite supports less than 160 operators on mobile CPUs, which is nearly 90% fewer than TensorFlow. Furthermore, the number of supported operators by mobile NPU (EdgeTPU on Pixel 7 Pro) is even fewer, i.e., 33 in 2022 and 63 in 2023. Consequently, mobile NPUs might benefit only a small number of DNNs.

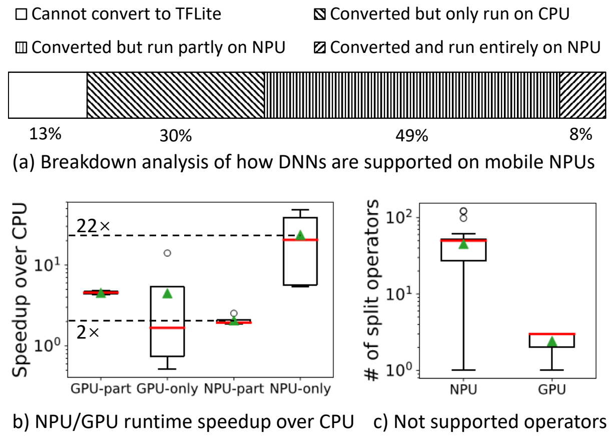


Figure 3: An empirical study of 110 in-the-wild DNNs crawled from public sources on Google Pixel 7 Pro.

For the lucky DNNs fully supported, mobile NPU is able to deliver significant inference speedup and energy reduction compared to mobile CPU/GPU. As an ASIC-based customized processor, mobile NPU is expected to offer faster and more energy-efficient DNN inference. To understand the performance of contemporary mobile NPU, we measure the inference latency and energy consumption of EdgeTPU on Google Pixel 7 Pro. The results are illustrated in Figure 2 (b) and (c). ResNet-152, the NPU achieves an inference latency of only 76ms, which is 39× and 11× faster than the accompanying CPU (4-cores used) and GPU, respectively. Similarly, on BERT-base, the NPU consumes only 0.3J of energy per image, while the CPU and GPU consume significantly higher amounts of energy, i.e., 1.7J (5.78×) and 9.0J (30×), respectively.

However, such significant benefits only apply to a very small portion of in-the-wild popular DNNs. To further understand the ubiquity of mobile NPU, we download more representative DNNs and test their performance. In total, we have found 110 TensorFlow-format DNNs from Model Zoo [140] and HuggingFace [53], prioritized based on their stars and download times. We then try to convert them to TFLite format using the official tool developed by Google and measure their performance on Google Pixel 7 Pro. As shown in Figure 3 (a), unfortunately, only 8% of those models can entirely run on mobile NPU, while for the rest: 13% fail to be converted to TFLite format; 30% fail to run on mobile NPU; and 49% require CPU-NPU co-running due to the lack of NN operator support by NPU. Figure 3 (b) further illustrates the performance of those DNNs on devices. It reveals that, only the 8% fully supported DNNs gain significant improvement over CPUs (i.e., > 20× median speedup), while the rest (either running partly on GPU/NPU or entirely on GPU) obtain much less profound speedup. In fact, for the DNNs that require CPU-NPU co-running, the inference speed is even not as good as running on mobile GPU. Figure 3 (c) further digs into the reason for such phenomenon: the NPU-incompatible DNNs need to be split into many sub-models to be scheduled between CPU and NPU (e.g., median number of 50); therefore the data movement and format exchange could severely delay the inference.

2.3 Emergence of Foundation Models

Foundation models are renovating AI ecosystem; the model design is converging. In recent years, significant advancements have been made in large-scale neural networks for language, image, and audio understanding and generation. GPT-3 [42] exemplifies this progress with impressive performance across various tasks, revolutionizing human-computer interaction and intelligent assistance. In the visual domain, Meta's SAM [75] demonstrates exceptional zero-shot proficiency. Additionally, models like Kosmos-1 [66] and PaLM-E [47] handle inputs from multiple modalities, enabling diverse task capabilities. These models share the transformer architecture [119], differing mainly in layer configurations or input modality processing. This convergence trend in AI model design is expected to continue in the future.

However, there has been no effort in building one model to fit highly diversified mobile AI tasks. None of the aforementioned foundation models is capable of (not even close to) solving all mobile AI tasks. A single modality model (such as GPT for NLP) cannot comprehend or generate data in other modalities. Existing multimodal models (such as CLIP for CV-NLP) can only deal with very limited multi-modal AI tasks. One might seek to include a foundation model for each $< input : M1, output : M2 >$ pair to solve the above issue, but: (1) It is not parameter-efficient as the comprehension and conversion between different modality data share inherent common sense [80, 101]; (2) It cannot support AI tasks that take multimodal input or output, such as visual question answering [146]. There have been ad-hoc approaches to deal with those issues [111], yet we are not aware of any systematic strategy to build a one-size-fits-all foundation model for diversified mobile AI tasks.

3 M4 DESIGN AND PROTOTYPING

3.1 Overview

Design principles M4 is a one-size-fits-all foundation model for diversified mobile AI tasks. It is designed with following principles: (1) *unified*: instead of building independent foundation models for different possible modalities, M4 provides a unified architecture that maximizes the capability sharing across different modalities, thus being more resource-efficient and extensible; (2) *elastic*: M4 can be easily scaled out to more modalities (either for input or output), e.g., for new types of sensor/app data; (3) *multimodal*: M4 can take multimodal input or generate multimodal output as needed, e.g., for advanced mobile applications like visual question answering or audio caption.

Model architecture Figure 1 illustrates the overall architecture of M4, which consists of three major components:
• *Multimodal Embedding* is to align the contents of different modalities by converting multimodal input data into a unified representation (i.e., a vector). It is typically implemented as a set of transformer encoders [57] for each modality, except that audio has two independent encoders to differentiate the context information (e.g., background noise, speaker emotions) and spoken language (e.g., automatic speech recognition).
• *Foundation Backbone (i.e., Pre-trained LLM Backbone)* is to comprehend and reason about the input data. It encapsulates abundant knowledge to understand complex embedded data, performs task-specific inference, and generates easily intelligible output for generator. It uses a decoder-based architecture trained on huge amount of textual dataset since language has been acknowledged as the most representative type of data [50, 92, 117]. The backbone is the most heavy part of M4.
• *Multimodal Generator* is to adapt the output from the foundation backbone to task-specific data format. For classification tasks, it is simply a MLP with softmax layer; for image tasks, it is a stable diffusion model [104]; etc.

Trainable parameters M4 contains three trainable parts to be fine-tuned for downstream mobile AI tasks: two PEFT modules inserted to the multimodal embedding and foundation backbone, respectively; and one MLP projection layer that adapts the output of multimodal embedding to the required representation of the foundation backbone. In later experiments, we use LoRA as the default PEFT method, but also report results for other PEFT methods. As will be demonstrated in §4, the trainable parameter size is trivial compared to the pre-trained part of M4 and is also much smaller than traditional state-of-the-art DNNs.

3.2 Prototyping with Off-the-Shelf LLMs

We have fully prototyped M4 with only pre-trained, off-the-shelf models publicly available from HuggingFace [126]. It guarantees the reproducibility of M4 and also demonstrates its compatibility with the existing LLM ecosystem.
• *Multimodal Embedding*. Multimodal embedding is composed of five parallel modules with transoformer encoder-only architecture: *Image (IMG_enc)*, *Text (TXT_enc)*, *Inertial Measurement Unit (IMU_enc)*, *Audio-Background (AUD-B_enc)*, and *Audio-Intent (AUD-I_enc)*. The *IMG_enc* employs the Vision Transformer (ViT) architecture and is utilized

	Types	Params (10^9)	Format	Architecture	GFLOPs
Embedding	IMG_enc [57]	0.6328	FP16	Encoder-only	167.5963
	TXT_enc [57]	0.354	FP16	Encoder-only	23.4189
	AUD-B_enc [57]	0.0862	FP16	Encoder-only	61.4679
	AUD-I_enc [102]	0.037	FP16	Encoder-Decoder	26
	IMU_enc [57]	0.0196	FP16	Encoder-only	5.1417
Backbone	LLaMA [117]	6.28	INT8	Decoder-only	312
Generator	TTS_dec	< 0.01	FP32	Encoder-Decoder	8.58
	IMG_dec [104]	1.0663	FP16	Encoder-Decoder	267
	GEN_dec [117]	< 0.01	FP16	MLP	125.0

Table 1: M4 sub-model parameters.

to encode visual information derived from input images into a sequential matrix of embeddings. The *TXT_enc* for input text is based on the CLIP architecture with a 12-layer encoder [101]. The *IMU_enc* for IMU data is a lightweight 6-layer encoder transformer model [101]. The *AUD-B_enc* encoder is also derived from ViT and is used for encoding audio backgrounds [59]. The pre-trained weights of the above four encoders are from ImageBind [57] multimodal model. The *AUD-I_enc* encoder is based on a sequence-to-sequence Transformer model for encoding audio intents, with pre-trained weights from Whisper.tiny.en [102].

• *Foundation Backbone.* We use LLaMA-7B (INT8 format) [117], pre-trained on one trillion tokens by Meta, as M4's backbone. Released in Feb. 2023, LLaMA is a research project aimed at creating a more versatile and efficient language model. It emphasizes training on a broad array of multilingual and multitask supervised data to enhance performance across various natural language processing tasks.

• *Multimodal Generator.* Multimodal generator is composed of three parallel decoders: *Text-to-Speech (TTS_dec)*, *Image (IMG_dec)* and *Generation (GEN_dec)*. The *TTS_dec* decoder is a integral element within the FAIRSEQ [93], the open-source sequence modeling toolkit released by Meta, tasked with converting the input text into corresponding speech signals. The *IMG_dec* decoder is a key component of the Diffusion Model, which generates image output from text input. The *GEN_dec* decoder serves as distinctive entities employed to lead the backbone language model to perform generation tasks, the parameters of which are initialized with the last layer of pre-trained LLaMA. Classification tasks could be reformulated with a generation prompt according to prompt learning literature [85] and re-use the generation decoder *GEN_dec*. Or it could re-initialize a new MLP decoder according to traditional classification literature.

System complexity Table 1 presents the model complexity of M4's different modules. The model comprises five types of embeddings (encoders), with parameter sizes ranging from 0.03B to 0.63B, and complexities ranging from 26GFLOPs to 167GFLOPs. The backbone of M4 is LLaMA, with a parameter count of 6.3B, and a complexity of 312GFLOPs. The LLaMA backbone is the largest component (86.1%) in terms of parameter size within the entire model. The generators (decoders) contribute trivially to the overall model size.

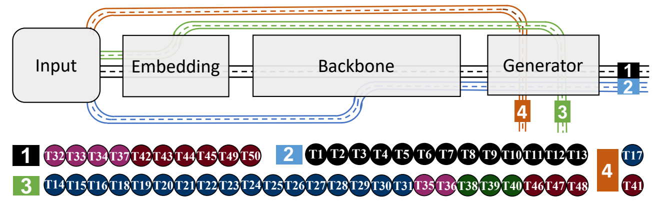


Figure 4: Execution path to each task in Table 3.

3.3 Multi-path Task Execution

Task-specific partial activation of M4. Not every task needs to go through the end-to-end workflow of M4, i.e., embedding-backbone-generator. Inspired by the early-exit inference [78, 115, 150] and multi-path design in hardware [121, 122], we propose a multi-path task execution design for M4. For simpler tasks that can be well solved by only part of M4's modules, we allow partial activation of M4 to reduce the computing complexity. In practice, developers could assign specific execution path to different tasks to achieve the best performance. In Figure 4, we have pre-defined 4 paths that can suffice the 50 mobile tasks we have investigated.

- **Path-1** means a full-model activation of M4. Tasks taking this path often require cross-modality alignment and complicated task compresion/reasoning. For instance: spoken language understanding and visual question answering.
- **Path-2** activates only the backbone and generator, mostly used by language tasks since M4's backbone can directly take raw textual input. For instance: input word prediction and machine translation.
- **Path-3** activates only the multimodal embedding, often used by tasks that can be accomplished through cross-modality (often language-X) alignment. For instance, image classification aligns the images with their corresponding textual labels (e.g., "cat"); human activity recognition tasks align the IMU data with the textual activity types (e.g., "walking"). In practice, the labels are embedded into a carefully designed prompt as shown below.
- **Path-4** activates only a specific generator, as taken by very few tasks. For instance: super-resolution can be accomplished by the visual generator (*IMG_dec*); text-to-speech is accomplished by the speech generator (*TTS_dec*).

Training details Fine-tuning M4 follows three steps: (1) *input data processing:* NLP tasks take as input the context information along with task-specific descriptive language prompts. CV/Audio/IMU tasks will do alike according to previous work [57]. For more intricate tasks, such as object detection, we utilize a region proposal network to generate a set of object proposals [151]. (2) *tunable weights setting:* For tasks going through backbone (Path 1, 2), only PEFT parameters in backbones are activated while freezing encoder PEFT

Tasks	Path	Prompts at Text Embedding and Backbone
Image classification	Path-3	[E]: There is a photo of a [Image_label: car].
Machine translation	Path-2	[B]: Translate the following sentences from [SRC_language: en] to [TGT_language: de].
Code generation	Path-2	[B]: Write an assembly code according to the [sentence] requirements.
HAR	Path-3	[E]: The human is [Activity_label: sitting].
Audio captioning	Path-1	[E]: Give a very short caption of the audio, the caption have 16 words at most.
Image captioning	Path-1	[E]: Give a very short caption of the image, the caption have 16 words at most.
Video classification	Path-3	[E]: There is a video of [Video_label: abseiling].
OCR	Path-3	[E]: A [negative / positive] review of a movie.

Table 2: A few prompt examples used in `M4`. [E] denotes the Text Embedding. [B] denotes the Backbone.

modules. For discriminative tasks that only go through encoders (Path 3), the encoder PEFT modules will be activated. Linear mapping contacting encoders and backbones will be always activated for shaping alignment. (3) *model weights updating:* During each training iteration, we compute the CrossEntropy loss from actual labels and predicted tokens, utilizing it to update the PEFT/MLP parameters.

Prompt design Two parts of `M4` need careful prompt engineering [73, 109, 152] to fully exploit its potential: the text embedding and foundation backbone. We have designed prompts for each mobile AI task in §4.1, and Table 2 lists a few of them as exemplifications.

4 EXPERIMENTS AND ANALYSIS

4.1 Benchmark and Setups

`eAIBench`: a comprehensive edge-oriented benchmark for AI tasks. As the very first effort for a one-size-fits-all mobile foundation model, a pivotal undertaking is the comprehensive evaluation of its versatility across diverse mobile AI tasks. Therefore, we embark on constructing an exhaustive edge-oriented benchmark for AI tasks, encompassing 38 tasks spanning 50 public datasets, as shown in Table 3. Those tasks are essential to real-world mobile applications (e.g., translation, object detection, and voice assistant). Many of these tasks have also received substantial attention within the mobile community itself [44, 72, 74, 76, 97, 108, 120, 133, 141, 145]. Each task is accompanied by its designated accuracy metric. `eAIBench` includes 5 modality domains: NLP, CV, Audio, Sensing (IMU), and Misc (Multimodal). While the majority of tasks are tailored to smartphones, we extend our scope to encompass pivotal devices such as laptops (code generation), autonomous cars (traffic sign classification), and IoT cameras (Counting).

To understand `M4`'s performance, we select one task-specific model (namely `TS-model`) for each dataset as a baseline. The selection of these models adheres to two primary criteria: (1) They must remain within the confines of mobile device resource constraints, specifically with fewer than 1 billion parameters; (2) The model accuracy shall be representative to the status quo on mobile devices, though not necessary to be absolute state-of-the-art. Consequently, our model selection draws primarily from two sources: open-source endeavors showcased at prominent mobile conferences like MobiCom and MobiSys during the past five years, and contemporary models showcased on the Paperwithcode platform [94]. A comprehensive list of employed `TS-models` is provided in Table 3, including instances such as ResNet-152 [63] for image classification, RoBERTa [87] for question answering, and CRDNN [128] for spoken language understanding.

Hardware We use three kinds of hardware platform:

- NVIDIA A100 GPU (A100), a high performance accelerator released in May. 2020 that has 40GB RAM, 384GB storage.
- NVIDIA Jetson ORIN NX (Orin), a high-end edge board for autonomous robotics or cars released in Feb. 2023 that has 16GB RAM, 64GB storage, and 1024-core NVIDIA Ampere architecture GPU with 32 Tensor Cores.
- Pixel 7 Pro smartphone (Pixel), a high-end smartphone released in Oct. 2022 that has 12GB RAM, 256GB storage, Octa-core CPU, Mali-G710 MP7 GPU, and an edge TPU.

All training experiments are conducted on two GPU servers, each equipped with 8 A100s. This encompassed the fine-tuning of `M4` and a portion of the training for the TS-model starting from scratch. To facilitate the accuracy of `TS-models` on each dataset quickly, we performe the inference experiments on A100 and Orin. We show the comprehensive results and corresponding platforms in Table 3. In total, our experiments take 100,000 GPU-hours. We use ORIN and Pixel, two typical mobile devices, to measure the runtime performance (memory, latency, energy, etc.) of `M4` on real-world devices.

Implementation The benchmark results are tested with PyTorch [98], and model parameters were obtained through two methods: (1) Directly downloading pre-trained parameters from open-source websites, such as Hugging Face [53]; (2) Training model parameters from scratch based on the open-source code, available on platforms like GitHub [58].

The `M4` prototype is built upon PandaGPT [111], a versatile instruction-following foundation model. Additionally, we implement two crucial modules using PyTorch [98]: (1) A redesigned multimodal generator to broaden output capabilities for mobile scenarios, including classification, text-to-speech, and image generation, surpassing the exclusive focus on text generation. (2) A multi-path controller aimed at enhancing compatibility with diverse mobile AI tasks. We directly acquire pre-trained parameters for the embedding and backbone from official releases [95, 96]. Following previous researches [65, 81], we fine-tune their adapters from scratch, as elaborated in Section 4.4.

The runtime-cost performance on Orin is obtained through jetson-stats [70], which is a powerful tool to analyze your board. Regarding the Pixel smartphone, the latency and

Category	Tasks	Mobile Application	Dataset	Specific-Models	Results	Metrics
NLP	Input word prediction T1	Input method (GBoard)	PTB	RNN [23]	0.17*	Accuracy
	Question answering T2 T3	Private assistant (Siri)	SQuAD v2.0	RoBERTa [35]	0.79*	F1
			TyDi QA	AraELECTRA [36]	0.87	F1
	Machine translation T4	Translator (Google Translate)	wmt22 en-de	Transformer [32]	0.34*	BLEU
	Emoji prediction T5	Input method (GBoard)	tweet_eval	RoBERTa [22]	0.33*	Accuracy
	Emotion prediction T6	Conversational analytics (Clarabridge)	go_emotion	RoBERTa [29]	0.57*	Accuracy
	Sentiment analysis T7	Conversational analytics (Clarabridge)	tweet_eval	RoBERTa [27]	0.77*	Accuracy
	Text classification T8 T9	Spam SMS filtering (Truecaller)	ag_news	BERT [37]	0.93*	Accuracy
			SST2	DistilBERT [38]	0.91*	Accuracy
	Grammatical error correction T10	Writing assistant (Grammarly)	JFLEG	FLAN-t5 [30]	0.68*	BLEU
	Text summary T11	Reading assistant (ChatPDF)	CNN Daily Mail	BART [5]	0.43*	ROUGE1
	Code document generation T12	Code editor (Javadoc)	CodeSearchNet	CodeT5-base [20]	0.33*	ROUGE1
	Code generation T13	Code editor (Copilot)	Shellcode_IA32	CodeBERT [21]	0.92	BLEU
CV	Object detection T14 T15	Augmented Reality (Google Lens)	COCO	Libra-rcnn [24]	0.43*	mAP
			LVIS	X-Paste [33]	0.51	AP
	Image retrieval T16	Image searcher (Google Photos)	Clothes Retrieval	Resnet50-arcface [31]	0.90*	Recall
	Super-resolution T17	Video/Image super-resolution (VSCO)	set5	Real-ESRGAN [19]	0.82*	SSIM
	Styler transfer T18	Painting & Beatifying (Meitu)	COCO, Wikiart	StyleGAN-nada [4]	0.23	CLIP score
	Semantic segmentation T19 T20	Smart camera (Segmentix)	ADE20K-150	Deeplabv3plus [25]	0.43*	mIoU
			PASCAL VOC	Deeplabv3plus [26]	0.79*	mIoU
	Optical character recognition T21	Intelligent document automation software (Ocrolus)	Rendered SST2	CLIP [34]	0.71	Accuracy
	Image classification T22 T23	Album management (Google Photos)	CIFAR100	GFNet-XS [103]	0.89	Accuracy
			ImageNet	Resnet-152 [64]	0.79	Accuracy
	Traffic sign classification T24	Intelligent transportation (Waze)	GTSRB	MicronNet [8]	0.98	Accuracy
	Vehicle re-identification T25	Surveillance camera (AI Re-ID)	Veri776	MSINet [6]	0.96	Rank
	Gender recognition T26	Smart camera (Face++)	Adience	MiVOLO-D1 [2]	0.96	Accuracy
	Location recognition T27	Navigation search (Google Maps)	Country211	CLIP [34]	0.46	Accuracy
	Pose estimation T28	AI fitness coach (Keep)	AP-10K	ViTPose [134]	0.69	AP
	Video classification T29	Video player (YouTube)	kinetics400	SlowFast [28]	0.79	Accuracy
	Crowd Counting T30	Smart camera (Fitness Tracking)	UCF-QNRF	CSS-CCNN [12]	437	MAE
	Image matting T31	Virtual backgrounds (Zoom)	RefMatte-RW100	MDETR [79]	0.06	MSE
Audio	Automatic speech recognition T32	Private assistant (Siri)	LibriSpeech	CTC+attention [14]	3.16%*	WER
	Spoken language understanding T33 T34	Private assistant (Siri)	FSC	Transformer [18]	0.37%	WER
			SLURP	CRDNN [3]	0.82*	Accuracy
	Emotion recognition T35	Emoji recommendation (WeChat)	IEMOCAP	ECAPA-TDNN [15]	0.64*	Accuracy
	Audio classification T36	Music discovery (Shazam)	ESC-50	ACDNet [1]	0.87	Accuracy
	Keyword spotting T37	Private assistant (Siri)	Speech command	Cnn-trad-fpool3 [17]	0.88*	Accuracy
Sensing	Human activity recognition T38 T39 T40	AI fitness coach (Keep)	Using Smartphones	TS-TCC [51]	0.90	Accuracy
			HHAR	LIMU-BERT [16]	0.84	Accuracy
			MotionSense	LIMU-BERT [16]	0.91	Accuracy
Multimodal	Text-to-speech T41	Voice broadcast (WeChat reading)	LJSpeech	Transformer [13]	3.26	MCD
	Audio captioning T42 T43	Hearing-impaired accessibility (Ava)	Clotho	Transformer [10]	0.52*	BLEU
			AudioSet	Transformer [10]	0.64	BLEU
	Image captioning T44 T45	Visual-impaired accessibility (Supersence)	MSCOCO'14	LSTM [7]	0.73*	BLEU
			Flickr8k	LSTM [110]	0.58	BLEU
	Text-to-image retrieval T46 T47	Image search (Google Photos)	Flickr8k	NAPReg [69]	0.39	Recall
			Flickr30k	CLIP [34]	0.69	Recall
	Audio/Text-to-image generation T48	Art creation (Verb Art)	VGGSound	Wav2clip [11]	99.89	FID
	Visual question answering T49 T50	Visual-impaired accessibility (Answerables)	VQA v2.0	MUTAN [41]	0.63	Accuracy
			VizWiz	MUTAN [41]	0.52	Accuracy

Table 3: A comprehensive benchmark of edge-oriented AI tasks. Circled abbreviations denotes specific task and dataset. * denotes the results obtained from Jetson ORIN, while others are obtained from A100.

memory results are measured using TFLite's benchmark tools [116], while power consumption data is extracted from Android's virtual file system (e.g., /sys and /proc).

4.2 Overall Accuracy

M4 can well support most mobile AI tasks and datasets. Figure 5 illustrates M4's overall performance improvement (or degradation) compared to TS-models. As observed, M4 can achieve comparable performance across 85% of tasks, with over 50% of these tasks showcasing considerable performance improvement. Note that the vertical axis, normalized in Table 3, reflects variations in accuracy across the 50 tasks. Our main focus is on evaluating whether M4 demonstrates superior or inferior accuracy compared to respective tasks,

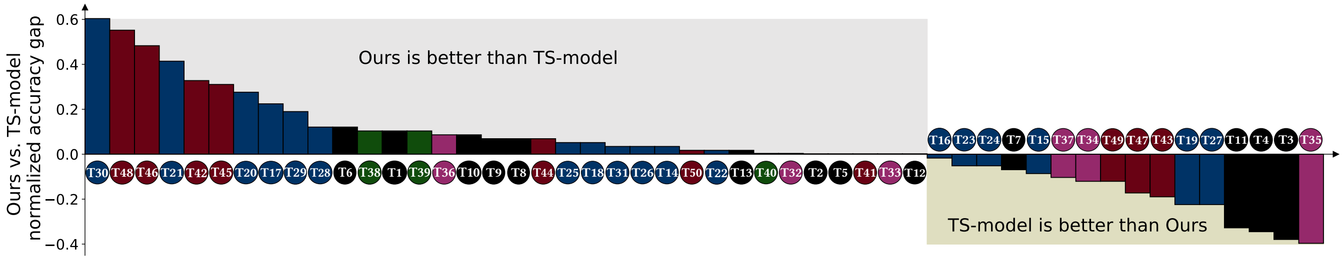


Figure 5: Normalized accuracy comparison of M4 and TS-models on 50 popular mobile tasks and datasets.

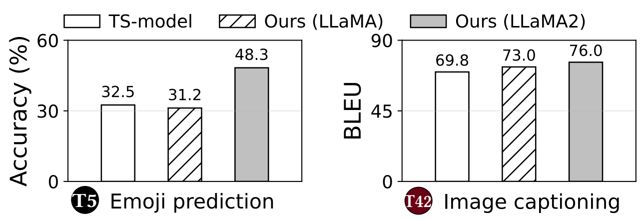


Figure 6: Performance improvement of M4 when replacing LLaMA with LLaMA2 as foundation backbone.

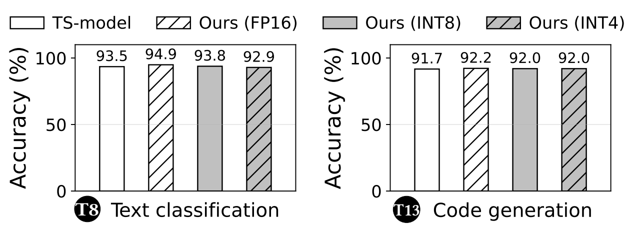


Figure 7: Accuracy of quantized M4 compared with TS-models. FP16/INT8/INT4 represent the numerical representation bit-width employed by LLaMA.

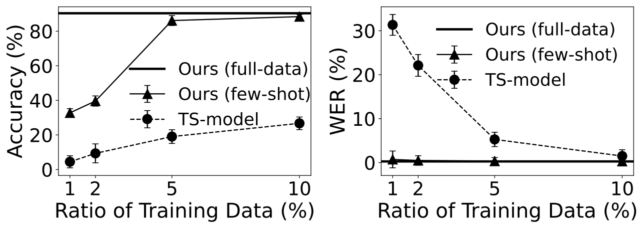
assessing its universal capability. Emphasis is placed on overall performance rather than specific task improvements. For example on T1 input word prediction, T42 audio captioning, and T46 text-to-image retrieval, M4 yields accuracy increments of 6%, 19%, and 28% respectively. Such commendable improvements are attributed to the well-engineered design of M4, characterized by its unified, adaptive, and multimodal foundation model. While M4's prowess is manifest, it is prudent to acknowledge marginal performance dips (not surpassing 10%) observed in specific tasks. Instances such as T7 sentiment analysis, T16 image retrieval, and T37 keyword spotting exemplify this trend, with accuracy experiencing nominal decrements of 4%, 1%, and 6%, respectively. However, even in these cases, M4 remains viable for deployment with usable performance. Additionally, M4 only showcases diminished performance in 4 tasks, with accuracy drop-offs of up to 20%. The reason behind this reduction stems from the unique requirements of certain low-resource translation tasks, necessitating extensive language knowledge that isn't inherently embedded within the current foundation model's pre-training phase.

M4 can be further enhanced with enhanced foundation models. Figure 6 illustrates the performance improvement realized by M4 through the integration of the latest LLaMA2, in comparison to LLaMA. LLaMA2, a refined evolution of its precursor, LLaMA, introduces heightened capabilities and marked improvements [117]. Released in July 2023, LLaMA2 marks a substantial leap forward, expanding the context window and ushering in the innovation of grouped-query attention. This novel architectural element empowers the model with rapid information processing capabilities. As observed, M4 using LLaMA2 attains a remarkable 15% accuracy enhancement and a 2% improvement in BLEU scores for T5 emoji prediction and T42 image caption tasks, respectively. This prowess is attributed to LLaMA2's optimized architectural schema, expansive training corpus (comprising 2T tokens), and elevated data quality [117]. As M4 is inherently adaptable to its foundation underpinnings, it seamlessly integrates and capitalizes upon the latest components.

M4 can efficiently preserve the performance with low-bit quantization. Figure 7 illustrates the performance comparison of M4 using quantized backbone with respect to the TS-model. As observed, M4 using 8-bit (INT8) and 4-bit (INT4) quantization both achieve nearly lossless accuracy, compared to M4 using 16-bit float representation (FP16). For example, on T8 text classification and T13 code generation, INT8 and INT4-based M4 achieved only a marginal decrease in accuracy compared to FP16-based M4, with reductions of 0.2%-0.9% and 0.2%-2%, respectively. The reasonable behind is that large models possess an abundance or even surplus knowledge representation, which contributes to more extensive knowledge even after quantization [54]. Therefore, we consistently employed M4 with the default INT8 quantization of the LLaMA backbone.

4.3 Zero/Few-shot Ability

We experiment on two tasks, image classification and spoken language understanding. For each task, we follow prior work [43] to randomly select gold labels, with the sample size



(a) Image classification **(b) SLU (lower is better)**

Figure 8: Few-shot testing of M4 and TS-model. SLU: spoken language understanding.

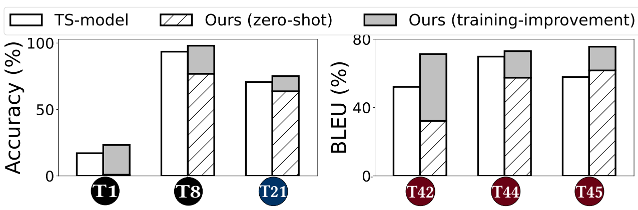


Figure 9: Zero-shot testing of M4 and TS-model.

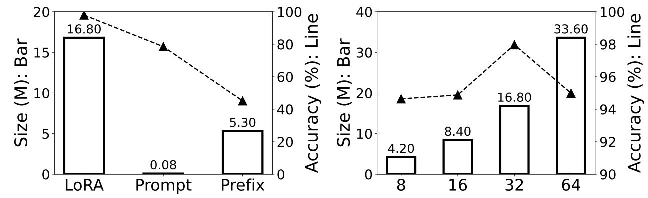
varying between 1% and 10% of the entire dataset. By default, the labels form a skewed distribution across clients to be more realistic to real-world situations. For each dataset, we conduct 5 repeated experiments and report the mean results.

M4 has better few-shot ability than TS-models that are trained from scratch. In Figure 8, few-shot M4 performs on par with or slightly lower than M4 with full data tuning. Notably, it consistently outperforms TS-model by up to 67.1%. For example on T33, even with a mere 1% sample (equating to just 231 samples), few-shot M4 achieves a Word Error Rate (WER) of 0.7%. It is a mere 0.4% higher than M4 with full data, but 25.4% lower than TS-model that is trained from scratch. This outcome underscores M4's prowess in leveraging pre-trained multimodal knowledge for swift adaptation to new tasks, even with scarce data.

M4 also has a certain zero-shot ability, but fine-tuning makes it much more accurate. Figure 9 illustrates M4's zero-shot capabilities on 6 tasks. Evidently, M4 demonstrates commendable zero-shot proficiency, attaining approximately 80% of the TS-model performance in most cases. Notable instances include T8 T21 T42 and T44, where M4's zero-shot performance remains acceptably close to the corresponding TS-models, with reductions ranging from 7% to 20%. In T45, M4 showcases a 4% improvement over TS-models, a testament to the efficacy of prompt learning methodologies. Notwithstanding these accomplishments, the application of fine-tuning to these datasets yields substantial accuracy enhancements for M4, surging by 11%-39%. This improvement arises from M4's robust attention-based architecture [125].

Tasks	PEFT settings		PEFT results	
	Techniques	Rank	Size (Ratio)	Acc (Dif)
Emoji prediction	LoRA	4	2M (0.03%)	31 (1↓)
Image classification	LoRA	4	8M (0.007%)	90 (1↑)
Human activity recognition	LoRA	1	5M (0.004%)	96 (5↑)
Audio captioning	LoRA	4	4M (0.06%)	72 (19↑)

Table 4: PEFT-enhanced M4's optimal results. Size (Ratio) denotes the trainable parameter size and its ratio to total parameters. Acc (Dif) denotes the performance of PEFT-enhanced M4 along with the differences compared to TS-model, and the units are %.



(a) PEFT techniques **(b) PEFT LoRA rank**

Figure 10: Impact analysis of PEFT-enhanced M4 on text classification. Size (M): trainable parameter size.

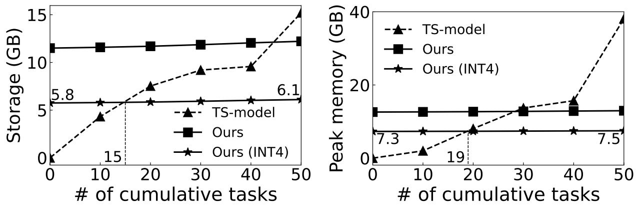
4.4 Parameter-efficient Fine-tuning

A proper PEFT technique and its configuration is crucial to trade off M4 performance and cost. Table 4 reports the optimal results of M4 on the trade-off between model accuracy and trainable parameter size. Our observations highlight the efficacy of the LoRA tuning technique, paired with well-suited rank settings, in yielding optimal results across a majority of tasks. PEFT-enhanced M4 attains a noteworthy 6% accuracy boost over TS-models, while engaging a mere 0.0253% of parameters for fine-tuning on average.

Diving deeper, Figure 10 provides a comprehensive analysis of the impact of diverse PEFT techniques and associated hyper-parameters on the performance of PEFT-enhanced M4. In Figure 10(a), the discernible trend showcases LoRA tuning as a standout performer, surpassing Prompt and Prefix tuning by 19% and 52% in terms of accuracy. Additionally, the fine-tuning process using LoRA mandates a mere 16.8 million trainable parameters, resulting in an exceptionally frugal training cost. Figure 10(b) offers further insights, indicating that selecting an appropriate LoRA rank value plays a pivotal role in propelling M4 towards heightened model accuracy while simultaneously minimizing trainable parameter size. For instance, with the LoRA rank set at 32, M4 attains a commendable accuracy of 98% on the task, leveraging a mere 16.8 million trainable parameters.

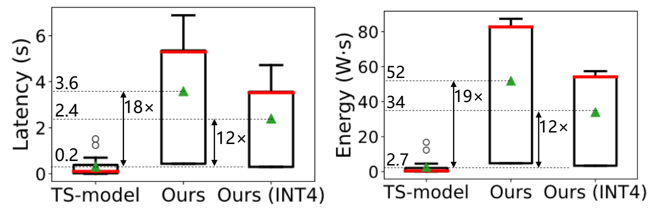
4.5 Runtime Cost

This subsection evaluates the storage, peak memory, latency, and energy consumption of running M4 and 50 TS-models on Jetson ORIN and Pixel 7 Pro.



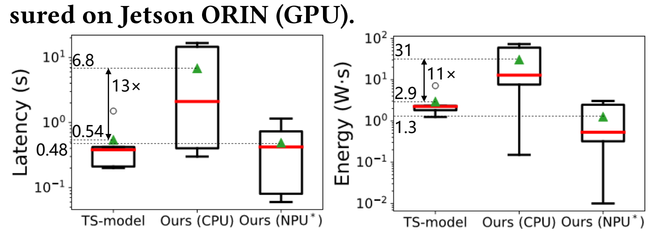
(a) Storage (b) Peak memory

Figure 11: M4's scalability analysis of storage and peak memory measured on Jetson ORIN.



(a) Average latency (b) Average energy

Figure 12: M4's runtime cost of latency and energy measured on Jetson ORIN (GPU).



(a) Latency (b) Energy

Figure 13: What-if cost analysis of latency and energy when running M4 on Pixel 7 Pro. TS-model: on CPU.

M4 is more storage-efficient when the model number scales out. Figure 11(a) presents a comparative analysis of storage between M4 and TS-models as the task count increases. As observed, M4's storage footprint is notably greater when serving a limited number of tasks compared to TS-models. However, the narrative changes as task diversity proliferates. With the deployment of a modest number of tasks (e.g. 15 tasks), the storage of M4, specifically those equipped with INT4 quantization, outpaces that of TS-models. This trend intensifies as the number of tasks expands. Ultimately, TS-models surpass the storage allocation of INT4-based M4, culminating at 15.2GB, signifying a substantial 2.5-fold escalation. This underscores M4's compelling storage scalability.

M4 is memory hungry, but is capable of holding more tasks for warm in-memory inference when task number scales out. Figure 11(b) shows that even when serving 50 tasks simultaneously, the cumulative peak memory usage of INT4-based M4 remains at a modest 7.5GB. This constitutes a mere 2.7% increase, while concurrently yielding a notable 5.1-fold reduction in peak memory consumption compared to TS-models. This exceptional memory efficiency can be attributed to M4's foundation design, which initially houses all requisite model parameters. Subsequently, the integration of new tasks necessitates only a marginal addition of fine-tuning parameters, typically amounting to less than 10MB each. The 7.5GB of peak memory cannot fit some mobile devices, but it is entirely affordable for many high-end smartphones with 12/16/32GB of RAM, like the Pixel 7 Pro we used. This underscores M4's practicality and potential to be effectively deployed across a spectrum of devices.

M4 is 18× slower and incurs 19× more energy than TS-models on the same processor. Figure 12 provides a comparison of the inference latency and energy consumption between M4 and TS-models across the spectrum of 50 tasks. As observed, M4 using INT8-format LLaMA exhibits 12× and 19× (on average) higher inference latency and more energy consumption, compared to TS-models. While the transition to INT4 quantization offers a marginal amelioration, the performance gap remains significant—a respective 8× increase in latency and 12× surge in energy consumption compared to TS-models. This substantial performance degradation is primarily due to M4's substantial parameter count and intricate computational demands.

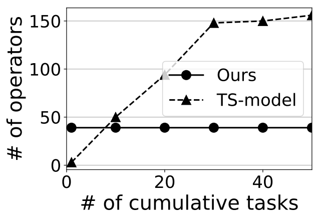
M4 could get on par execution speed as TS-models if it can be deployed to run on the NPU. Figure 13 provides a runtime cost comparison between M4 and TS-models on the CPU and NPU. We obtain the latency and energy of 50 tasks on CPU, denoted as TS-model and Ours (CPU). As observed, M4's inference latency and energy consumption on CPU are notably 13× and 11× (on average) higher than the TS-models.

Given the substantial performance advantage of NPU over CPU as shown in §2.2, we aim to evaluate the optimized runtime cost when deploying M4 on the NPU. However, the NPU currently supports a limited set of operators (details in §2.2) and cannot directly execute all components of M4. Similarly, the majority of TS-models cannot run directly on the NPU. Therefore, we conduct a what-if analysis to estimate its runtime latency and energy consumption on NPU, denoted as Ours (NPU^*). This projection is achieved by leveraging the observed performance ratio from TS-models between the NPU and CPU, as discussed in Section §2.2. Subsequently, utilizing the measured performance of M4 on the CPU, we can derive its estimated performance on NPU.

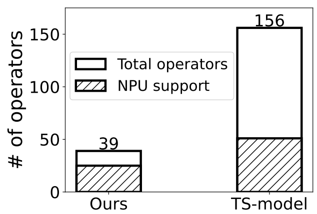
As observed, NPU-enabled M4 achieves an average latency of 0.48s and energy consumption of 1.3J, which are even 11.1% and 55.2% lower than TS-models on CPU. Furthermore, we delve into the architectural intricacies of M4 to analyze the latency breakdown performance in Table 5. From this table, we observe that the latency optimization bottleneck for M4

Tasks	Path	Latency (s)	
		CPU	NPU*
Image classification	Path-3	IMG_enc: 2.10	0.11
Audio classification	Path-3	AUD-I_enc: 0.28	0.014
Question answering	Path-2	First token: 6.34	0.32
		Sequent tokens: 0.24/token	0.012/token
Visual question answering	Path-1	First token: 6.47	0.32
		Sequent tokens: 0.25/token	0.013/token
Text-to-speech	Path-4	TTS_dec: 0.82	0.041

Table 5: An in-depth what-if cost analysis of latency when running M4 on Pixel 7 Pro. NPU*: M4's estimated latency based on the NPU acceleration rate of TS-model if it can be deployed on NPU.



(a) Total operators



(b) NPU supported operators

Figure 14: Simplicity analysis of M4's operators.

lies in the time taken by the IMG-encoder and the generation of the first token by the backbone, which is approximately 2.1s and 6.3s, respectively. These components collectively account for 31% and 93% of M4's average latency (6.8s in Figure 12(a)). However, the other components could exhibit near real-time inference (less than 100ms) if being deployed on NPU. These achievements demonstrate that, if M4 can be accelerated on NPU, it could get on par execution speed and energy consumption with TS-models. The aim of comparing M4's NPU performance with TS-models on CPU isn't to claim superiority, but to showcase how future NPU support can enhance efficiency for mobile foundation models.

4.6 Model Architecture Simplicity

M4's architectural design is much simpler and cleaner in terms of NN operators, therefore could greatly simplify accelerator design. Figure 14(a) shows that the number of operators in the TS-models increases rapidly with the growth in the number of tasks. Notably, as the task spectrum broadens to encompass 50 tasks, the number of operator types culminates at 156. In contrast, M4 engages a mere 39 operator types, encompassing both foundation model and task-specific "adapters". Furthermore, Figure 14(b) undertakes a granular exploration of NPU supported operators for both M4 and TS-models. It underscores that only 51 out of 156 operators in TS-models are supported by the NPU, with more than 2/3 of the operators unable to fully run on the NPU; But for M4, NPU-supported operators account for 64%. This

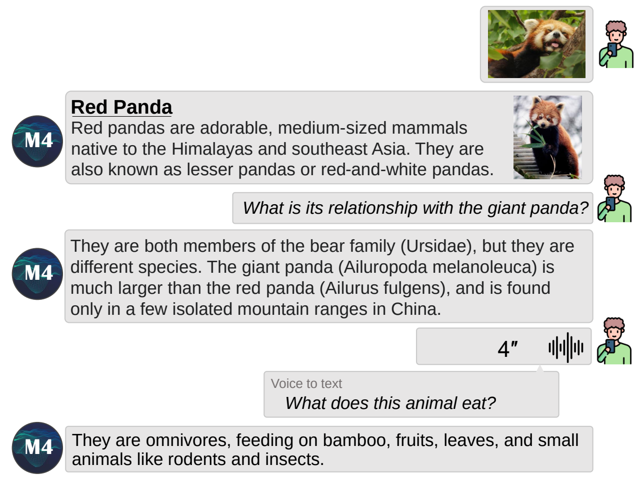


Figure 15: A demo of M4: multimodal chat.

phenomenon emanates from M4's transformer-based architecture, which inherently involves quantity and NPU friendly operators, thus enhancing operational efficiency [99].

4.7 Novel Application with M4

M4 enables complex, unpresent mobile applications. Based on our proposed M4, we build a demo of a multimodal chat use case as shown in Figure 15. Users engage in multi-turn chats with the M4 client using multimodal inputs such as images, text, and audio, thereby obtaining precise and tailored answers that meet their requirements. This multi-modal computing capability is also crucial for the recent popular mobile agents [82]. We build this prototype system of M4 based on the architecture depicted in Figure 1. It first aligns the contents of image, text, and audio by converting multimodal input data into a unified representation. Then, it encapsulates abundant knowledge to understand complex embedded data, perform task-specific inference, and generate the required information. This innate capability for multimodal processing harbors the potential to significantly enrich the landscape of mobile applications.

5 RELATED WORK

Foundation models. Building one-for-all foundation models to serve generic AI tasks has been a primary research goal of the machine learning community. The recent advancements of LLMs [92, 117, 118, 143], multimodalities alignment [57, 91, 111, 114], and parameter-efficient training methods [89, 136, 142] have shed lights on this challenging goal. For instance, ImageBind [57] and CoDi [114] focus on how to align the embeddings of each modality, and PandaGPT [111] further attempts to compose semantics from different modalities naturally based on LLaMA [117]. However, there have been no efforts like M4 that try to fit extremely diversified AI tasks into one model. Meanwhile, M4 leverages the most state-of-the-art pre-trained LLMs to reuse the wisdoms as well as the investments from the ML community&industry. The concurrent work NExT-GPT [127] shares a similar architecture as M4. Nonetheless, M4 introduces two distinctive contributions: (1) It marks the inaugural proposal of a transformer-based N-1-M architecture, aiming to curtail resource costs in any-to-any modal generation; (2) Its innovative multi-path execution design is tailored to enhance compatibility with highly diversified mobile AI tasks.

Hardware-system-algorithm co-design for mobile AI. AI workloads are highly compute-intensive and exhibit analogous patterns, therefore is better to be accelerated domain-specific accelerator (e.g., NPUs). For instance, SpAtten [122] and Sanger [88] focus on how efficient algorithm-architecture co-designs can reduce sparse attention computation and memory access. Besides, QNAS [83] and NAAS [84] focus on composing highly matched neural-hardware architectures by jointly searching for neural network architecture, accelerator architecture, and compiler mapping. However, all prior literature makes tradeoffs between the ubiquity of operator support and the performance, instead of for a foundation model that can serve generic AI tasks itself. The vision of the mobile foundation model could open a new research domain for cross-layer co-design of mobile AI. There have been preliminary attempts [131, 132, 137] to alleviate the huge resource cost of large foundation models for devices. Those work are orthogonal to this work.

Managing AI as a mobile system service. AI has been a ubiquitous workload on mobile devices, and managing it at a system aspect (instead of individual app) could facilitate OS-wise runtime scheduling and software deployment. Some early studies [52, 123, 133, 144, 147] attempt to mitigate the severe fragment across different libs in the mobile DL ecosystem. Google introduced a unified ML interface NNAPI [60] into Android in 2017, to relieve the gap between heterogeneous mobile processors and user-defined ML frameworks. Compared to the above work, M4 takes another giant step further that mobile devices shall manage a foundation model for each ML task and expose it as firmware.

6 LIMITATIONS AND FUTURE WORK

This study has several potential limitations. (1) eAIBench's results are evaluated on datacenter GPU (NVIDIA A100) and edge GPU (Jetson Orin), lacking assessment on mobile devices. It's mainly due to the highly diverse code implementation of baseline models and the huge time span of evaluating M4 on large test dataset. There might exist performance gap between different hardware architectures. Yet, the comparison is fair as both baseline models and M4 are evaluated on the same hardware. In fact, due to the simpler and cleaner architecture of M4, it would be much easier to design accelerator to support M4 with high precision. (2) M4 underperforms baseline models on certain ML tasks. This unveils the limitation of existing pre-trained foundation models, e.g., translation. On the one hand, we do not expect M4 to be able to solve all mobile AI tasks in the near future; it could co-exist with traditional DNNs that run on mobile CPU. On the other hand, the LLM capacity is still fast evolving: from LLaMA-1/2 used in this study, to the Mistral-7B [71] that ranks higher even than LLaMA-13B. Such continuous improvement endeavors our vision with much confidence.

To be noted, M4 is the very first step towards the vision of mobile foundation model. We believe it could potentially revolutionize the mobile AI landscape and open a new research domain. However, to fully realize the vision, there are a few key designs to be explored. For instance: (1) *Foundation model design*: As a preliminary prototype, M4 is currently built atop off-the-shelf, separately pre-trained LLMs from Internet instead of being tailored for mobile devices. Therefore, it is still highly inefficient in terms of accuracy and model parameter size. With enough resources (GPUs and data), hardware vendors can build a more compact mobile foundation model that is expected to deliver significantly higher accuracy with lower runtime cost than M4. (2) *Accelerator design*: fine-tuning for downstream tasks generates small "adapters" that are inserted into the mobile foundation model. The NPU better has the flexibility to run those adapters as well; otherwise the inference must involve CPU/GPU computation and data movement overhead. Fortunately, the adapters have simple structure (e.g., linear matrix operations) and very few weights. (3) *FM upgradating*: the foundation model capacity could evolve with better architecture/weights as shown in §4.2. Yet the adapters trained for the old foundation model cannot work with the new one. We therefore need a unified interface between LLMs and adapters to allow them to evolve independently without interfering with each other.

7 CONCLUSIONS

We envision a mobile hardware-OS co-managed multimodal foundation model that is exposed to mobile applications as a system service to serve almost every on-device AI task. We design and prototype the first such model using off-the-shelf LLMs. Evaluated on a comprehensive benchmark consisting of 50 representative mobile AI tasks, M4 shows good accuracy, better scalability and reduced runtime cost.

8 ACKNOWLEDGES

The authors thank the anonymous reviewers and the shepherd for their insightful feedbacks. This work was supported by NSFC (61921003, 62102045).

REFERENCES

[1] Audio classification task. https://github.com/mohaimenz/acdnet.
[2] Gender recognition task. https://github.com/wildchlamydia/mivolo.
[3] Spoken language understanding task2. https://drive.google.com/drive/folders/103t4_zqBZNqa_gGlIfteIs8_mdKhn3Rd?usp=sharing.
[4] Styler transfer task. https://github.com/rinongal/StyleGAN-nada.
[5] Text summary task. https://huggingface.co/facebook/bart-large-cnn.
[6] Vehicle re-identification task. https://github.com/vimar-gu/msinet.
[7] Image captioning task. https://github.com/maxy0524/image_captioning/tree/master, 2015.
[8] Traffic sign classification task. https://github.com/ppriyank/MicronNet, 2018.
[9] How apple personalizes siri without hoovering up your data. https://www.technologyreview.com/2019/12/11/131629/apple-ai-personalizes-siri-federated-learning/, 2019.
[10] Audio captioning task. https://github.com/lukewys/dcase_2020_T6/tree/master, 2020.
[11] Audio/text-to-image generation task. https://github.com/descriptinc/lyrebird-wav2clip, 2020.
[12] Crowd counting task. https://github.com/val-iisc/css-ccnn, 2020.
[13] Text-to-speech task. https://github.com/facebookresearch/fairseq/tree/main/examples/wav2vec, 2020.
[14] Automatic speech recognition task. https://huggingface.co/speechbrain/asr-crdnn-rnnlm-librispeech/tree/main, 2021.
[15] Emotion recognition task. https://drive.google.com/drive/folders/1U9SiO4KkCNBKfxilXzJqBZ_k-vHz4ltV?usp=sharing, 2021.
[16] Human activity recognition task. https://github.com/dapowan/LIMU-BERT-Public, 2021.
[17] Keyword spotting task. https://github.com/tensorflow/tensorflow/tree/master/tensorflow/examples/speech_commands, 2021.
[18] Spoken language understanding task1. https://github.com/speechbrain/speechbrain/tree/develop/recipes/fluent-speech-commands, 2021.
[19] Super resolution task. https://github.com/open-mmlab/mmagic/tree/main/configs/real_esrgan, 2021.
[20] Code document generation task. https://huggingface.co/Salesforce/codet5-base-multi-sum, 2022.
[21] Code generation task. https://huggingface.co/microsoft/codebert-base, 2022.
[22] Emoji prediction task. https://huggingface.co/cardiffnlp/twitter-roberta-base-emoji, 2022.
[23] Input word prediction task. https://github.com/friedrichor/Language-Model-Next-Word-Prediction/tree/main, 2022.
[24] Object detection task1. https://github.com/open-mmlab/mmdetection/tree/main/configs/libra_rcnn, 2022.
[25] Semantic segmentation task1. https://github.com/open-mmlab/mmsegmentation/tree/main/configs/deeplabv3plus, 2022.
[26] Semantic segmentation task2. https://github.com/open-mmlab/mmsegmentation/tree/main/configs/deeplabv3plus, 2022.
[27] Sentiment analysis task. https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment-latest, 2022.
[28] Video classification task. https://github.com/open-mmlab/mmaction2/tree/main/configs/recognition/slowfast, 2022.
[29] Emotion prediction task. https://huggingface.co/SamLowe/roberta-base-go_emotions, 2023.
[30] Grammatical error correction task. https://huggingface.co/pszemraj/flan-t5-large-grammar-synthesis, 2023.
[31] Image retrieval task. https://github.com/open-mmlab/mmpretrain/tree/main/configs/arcface, 2023.
[32] Machine translation task. https://github.com/huggingface/transformers, 2023.
[33] Object detection task2. https://github.com/yoctta/xpaste, 2023.
[34] Optical character recognition, location recognition, and text-to-image retrieval task. https://github.com/mlfoundations/open_clip/, 2023.
[35] Question answering task1. https://huggingface.co/deepset/roberta-base-squad2, 2023.
[36] Question answering task2. https://huggingface.co/aubmindlab/araelectra-base-discriminator, 2023.
[37] Text classification task1. https://huggingface.co/JiaqiLee/bert-agnews, 2023.
[38] Text classification task2. https://huggingface.co/distilbert-base-uncased-finetuned-sst-2-english, 2023.
[39] Mario Almeida, Stefanos Laskaridis, Abhinav Mehrotra, Lukasz Dudziak, Ilias Leontiadis, and Nicholas D Lane. Smart at what cost? characterising mobile deep neural networks in the wild. In Proceedings of the 21st ACM Internet Measurement Conference, pages 658–672, 2021.
[40] Manoj Alwani, Han Chen, Michael Ferdman, and Peter Milder. Fused-layer cnn accelerators. In 2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), pages 1–12. IEEE, 2016.
[41] Hedi Ben-Younes, Rémi Cadene, Matthieu Cord, and Nicolas Thome. Mutan: Multimodal tucker fusion for visual question answering. In Proceedings of the IEEE international conference on computer vision, pages 2612–2620, 2017.
[42] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. Advances in neural information processing systems, 33:1877–1901, 2020.
[43] Dongqi Cai, Yaozong Wu, Shangguang Wang, Felix Xiaozhu Lin, and Mengwei Xu. Fedadapter: Efficient federated learning for modern nlp. arXiv preprint arXiv:2205.10162, 2022.
[44] Qingqing Cao, Noah Weber, Niranjan Balasubramanian, and Aruna Balasubramanian. Deqa: On-device question answering. In Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services, pages 27–40, 2019.
[45] Xingyu Chen, Yufeng Liu, Yajiao Dong, Xiong Zhang, Chongyang Ma, Yanmin Xiong, Yuan Zhang, and Xiaoyan Guo. Mobrecon: Mobile-friendly hand mesh reconstruction from monocular image. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 20544–20554, 2022.
[46] Yinpeng Chen, Xiyang Dai, Dongdong Chen, Mengchen Liu, Xiaoyi Dong, Lu Yuan, and Zicheng Liu. Mobile-former: Bridging mobilenet and transformer. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 5270–5279, 2022.
[47] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. arXiv preprint arXiv:2204.02311, 2022.
[48] Marcella Cornia, Lorenzo Baraldi, and Rita Cucchiara. Show, control and tell: A framework for generating controllable and grounded captions. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 8307–8316, 2019.
[49] Weihao Cui, Han Zhao, Quan Chen, Hao Wei, Zirui Li, Deze Zeng, Chao Li, and Minyi Guo. {DVABatch}: Diversity-aware {Multi-Entry}{Multi-Exit} batching for efficient processing of {DNN} services on {GPUs}. In 2022 USENIX Annual Technical Conference (USENIX ATC 22), pages 183–198, 2022.
[50] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.
[51] Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee Keong Kwoh, Xiaoli Li, and Cuntai Guan. Time-series representation learning via temporal and contextual contrasting. arXiv

preprint arXiv:2106.14112, 2021.

[52] Amin Eslami Abyane and Hadi Hemmati. Robustness analysis of deep learning frameworks on mobile platforms. In IFIP International Conference on Testing Software and Systems, pages 160–177. Springer, 2021.

[53] Hugging Face. Hugging face. https://huggingface.co/, 2023.

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

[55] X Yu Geoffrey, Yubo Gao, Pavel Golikov, and Gennady Pekhimenko. Habitat: A {Runtime-Based} computational performance predictor for deep neural network training. In 2021 USENIX Annual Technical Conference (USENIX ATC 21), pages 503–521, 2021.

[56] Petko Georgiev, Nicholas D Lane, Kiran K Rachuri, and Cecilia Mascolo. Leo: Scheduling sensor inference algorithms across heterogeneous mobile processors and network resources. In Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking, pages 320–333, 2016.

[57] Rohit Girdhar, Alaaeldin El-Nouby, Zhuang Liu, Mannat Singh, Kalyan Vasudev Alwala, Armand Joulin, and Ishan Misra. Imagebind: One embedding space to bind them all. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 15180–15190, 2023.

[58] github. github. https://github.com.

[59] Yuan Gong, Yu-An Chung, and James Glass. Ast: Audio spectrogram transformer. arXiv preprint arXiv:2104.01778, 2021.

[60] Google. Android neural networks api. https://github.com/android/ndk-samples/tree/main/nn-samples, 2017.

[61] Filip Granqvist, Matt Seigel, Rogier Van Dalen, Aine Cahill, Stephen Shum, and Matthias Paulik. Improving on-device speaker verification using federated learning with privacy. Proceedings of International Speech Communication Association, 2020.

[62] Peizhen Guo and Wenjun Hu. Potluck: Cross-application approximate deduplication for computation-intensive mobile applications. In Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems, pages 271–284, 2018.

[63] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

[64] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

[65] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In International Conference on Learning Representations, 2021.

[66] Shaohan Huang, Li Dong, Wenhui Wang, Yaru Hao, Saksham Singhal, Shuming Ma, Tengchao Lv, Lei Cui, Owais Khan Mohammed, Qiang Liu, et al. Language is not all you need: Aligning perception with language models. arXiv preprint arXiv:2302.14045, 2023.

[67] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional lstm-crf models for sequence tagging. arXiv preprint arXiv:1508.01991, 2015.

[68] Loc N Huynh, Youngki Lee, and Rajesh Krishna Balan. Deepmon: Mobile gpu-based deep learning framework for continuous vision applications. In Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services, pages 82–95, 2017.

[69] Bhavin Jawade, Deen Dayal Mohan, Naji Mohamed Ali, Srirangaraj Setlur, and Venu Govindaraju. Napreg: nouns as proxies regularization for semantically aware cross-modal embeddings. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pages 1135–1144, 2023.

[70] jetson stats. jetson-stats. https://developer.nvidia.com/embedded/community/jetson-projects/jetson_stats, 2023.

[71] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. arXiv preprint arXiv:2310.06825, 2023.

[72] Shiqi Jiang, Zhiqi Lin, Yuanchun Li, Yuanchao Shu, and Yunxin Liu. Flexible high-resolution object detection on edge devices with tunable latency. In Proceedings of the 27th Annual International Conference on Mobile Computing and Networking, pages 559–572, 2021.

[73] Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. How can we know what language models know? Transactions of the Association for Computational Linguistics, 8:423–438, 2020.

[74] Liuyi Jin, Tian Liu, Amran Haroon, Radu Stoleru, Michael Middleton, Ziwei Zhu, and Theodora Chaspari. Emsassist: An end-to-end mobile voice assistant at the edge for emergency medical services. In Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services, pages 275–288, 2023.

[75] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. arXiv preprint arXiv:2304.02643, 2023.

[76] Rui Kong, Yuanchun Li, Yizhen Yuan, and Linghe Kong. Convrelu++: Reference-based lossless acceleration of conv-relu operations on mobile cpu. In Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services, pages 503–515, 2023.

[77] Nicholas D Lane, Sourav Bhattacharya, Petko Georgiev, Claudio Forlivesi, Lei Jiao, Lorena Qendro, and Fahim Kawsar. Deepx: A software accelerator for low-power deep learning inference on mobile devices. In 2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), pages 1–12. IEEE, 2016.

[78] Stefanos Laskaridis, Alexandros Kouris, and Nicholas D Lane. Adaptive inference through early-exit networks: Design, challenges and directions. In Proceedings of the 5th International Workshop on Embedded and Mobile Deep Learning, pages 1–6, 2021.

[79] Jizhizi Li, Jing Zhang, and Dacheng Tao. Referring image matting. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 22448–22457, 2023.

[80] Junnan Li, Ramprasaath Selvaraju, Akhilesh Gotmare, Caiming Xiong, and Steven Chu Hong Hoi. Align before fuse: Vision and language representation learning with momentum distillation. Advances in neural information processing systems, 34:9694–9705, 2021.

[81] Yixiao Li, Chen Liang, Pengcheng He, Nikos Karampatziakis, Weizhu Chen, and Tuo Zhao. Loftq: Lora-fine-tuning-aware quantization for large language models. arXiv preprint arXiv:2310.08659, 2023.

[82] Yuanchun Li, Hao Wen, Weijun Wang, Xiangyu Li, Yizhen Yuan, Guohong Liu, Jiacheng Liu, Wenxing Xu, Xiang Wang, Yi Sun, et al. Personal llm agents: Insights and survey about the capability, efficiency and security. arXiv preprint arXiv:2401.05459, 2024.

[83] Yujun Lin, Driss Hafdi, Kuan Wang, Zhijian Liu, and Song Han. Neural-hardware architecture search. NeurIPS WS, 2019.

[84] Yujun Lin, Mengtian Yang, and Song Han. Naas: Neural accelerator architecture search. In 2021 58th ACM/IEEE Design Automation Conference (DAC), pages 1051–1056. IEEE, 2021.

[85] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. ACM Computing Surveys, 55(9):1–35, 2023.

[86] Song Liu, Shengsheng Qian, Yiru Chen, Wenkui Ding, and Zhongyuan Wang. Hit: Hierarchical transformer with momentum contrast for

video-text retrieval. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 11915–11925, 2021.

[87] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692, 2019.

[88] Liqiang Lu, Yicheng Jin, Hangrui Bi, Zizhang Luo, Peng Li, Tao Wang, and Yun Liang. Sanger: A co-design framework for enabling sparse attention using reconfigurable architecture. In MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture, pages 977–991, 2021.

[89] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, and Sayak Paul. Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft, 2022.

[90] Harini Muthukrishnan, David Nellans, Daniel Lustig, Jeffrey A Fessler, and Thomas F Wenisch. Efficient multi-gpu shared memory via automatic optimization of fine-grained transfers. In 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA), pages 139–152. IEEE, 2021.

[91] Ivona Najdenkoska, Xiantong Zhen, and Marcel Worring. Meta learning to bridge vision and language models for multimodal few-shot learning. arXiv preprint arXiv:2302.14794, 2023.

[92] OpenAI. Gpt-4 technical report. https://cdn.openai.com/papers/gpt-4.pdf, 2023.

[93] Myle Ott, Sergey Edunov, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. arXiv preprint arXiv:1904.01038, 2019.

[94] paperswithcode. Paper with code. https://paperswithcode.com/, 2023.

[95] Imagebind parameters. Imagebind parameters. https://github.com/facebookresearch/ImageBind, 2023.

[96] LLaMa parameters. Llama parameters. https://github.com/facebookresearch/llama, 2023.

[97] Seonghoon Park, Yeonwoo Cho, Hyungchol Jun, Jeho Lee, and Hojung Cha. Omnilive: Super-resolution enhanced 360° video live streaming for mobile devices. In Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services, pages 261–274, 2023.

[98] Adam Paszke, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. arXiv preprint arXiv:1912.01703, 2019.

[99] Suchita Pati, Shaizeen Aga, Nuwan Jayasena, and Matthew D Sinclair. Demystifying bert: System design implications. In 2022 IEEE International Symposium on Workload Characterization (IISWC), pages 296–309. IEEE, 2022.

[100] Siddhant Prakash, Alireza Bahremand, Linda D Nguyen, and Robert LiKamWa. Gleam: An illumination estimation framework for real-time photorealistic augmented reality on mobile devices. In Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services, pages 142–154, 2019.

[101] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In International conference on machine learning, pages 8748–8763. PMLR, 2021.

[102] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In International Conference on Machine Learning, pages 28492–28518. PMLR, 2023.

[103] Yongming Rao, Wenliang Zhao, Zheng Zhu, Jiwen Lu, and Jie Zhou. Global filter networks for image classification. Advances in neural information processing systems, 34:980–993, 2021.

[104] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 10684–10695, 2022.

[105] Teven Le Scao, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. arXiv preprint arXiv:2211.05100, 2022.

[106] Haichen Shen, Lequn Chen, Yuchen Jin, Liangyu Zhao, Bingyu Kong, Matthai Philipose, Arvind Krishnamurthy, and Ravi Sundaram. Nexus: A gpu cluster engine for accelerating dnn-based video analysis. In Proceedings of the 27th ACM Symposium on Operating Systems Principles, pages 322–337, 2019.

[107] Yongming Shen, Michael Ferdman, and Peter Milder. Maximizing cnn accelerator efficiency through resource partitioning. ACM SIGARCH Computer Architecture News, 45(2):535–547, 2017.

[108] Cong Shi, Xiangyu Xu, Tianfang Zhang, Payton Walker, Yi Wu, Jian Liu, Nitesh Saxena, Yingying Chen, and Jiadi Yu. Face-mic: inferring live speech and speaker identity via subtle facial dynamics captured by ar/vr motion sensors. In Proceedings of the 27th Annual International Conference on Mobile Computing and Networking, pages 478–490, 2021.

[109] Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. arXiv preprint arXiv:2010.15980, 2020.

[110] Lingyun Song, Jun Liu, Buyue Qian, and Yihe Chen. Connecting language to images: A progressive attention-guided network for simultaneous image captioning and language grounding. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 8885–8892, 2019.

[111] Yixuan Su, Tian Lan, Huayang Li, Jialu Xu, Yan Wang, and Deng Cai. Pandagpt: One model to instruction-follow them all. arXiv preprint arXiv:2305.16355, 2023.

[112] Yanan Sun, Chi-Keung Tang, and Yu-Wing Tai. Semantic image matting. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 11120–11129, 2021.

[113] Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. Mobilebert: a compact task-agnostic bert for resource-limited devices. arXiv preprint arXiv:2004.02984, 2020.

[114] Zineng Tang, Ziyi Yang, Chenguang Zhu, Michael Zeng, and Mohit Bansal. Any-to-any generation via composable diffusion. arXiv preprint arXiv:2305.11846, 2023.

[115] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via early exiting from deep neural networks. In 2016 23rd international conference on pattern recognition (ICPR), pages 2464–2469. IEEE, 2016.

[116] TFLite tools. Tflite tools. https://www.tensorflow.org/lite/performance/measurement, 2023.

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

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

[119] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.

[120] Dan Wang, Haibo Lei, Haozhi Dong, Yunshu Wang, Yongpan Zou, and Kaishun Wu. What you wear know how you feel: An emotion inference system with multi-modal wearable devices. In Proceedings of the 26th Annual International Conference on Mobile Computing and Networking, pages 1–3, 2020.

[121] Hanrui Wang, Yongshan Ding, Jiaqi Gu, Yujun Lin, David Z Pan, Frederic T Chong, and Song Han. Quantumnas: Noise-adaptive search for robust quantum circuits. In 2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA), pages 692–708. IEEE, 2022.

[122] Hanrui Wang, Zhekai Zhang, and Song Han. Spatten: Efficient sparse attention architecture with cascade token and head pruning. In 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA), pages 97–110. IEEE, 2021.

[123] Haoyu Wang, Hao Li, and Yao Guo. Understanding the evolution of mobile app ecosystems: A longitudinal measurement study of google play. In The World Wide Web Conference, pages 1988–1999, 2019.

[124] Xinlong Wang, Wen Wang, Yue Cao, Chunhua Shen, and Tiejun Huang. Images speak in images: A generalist painter for in-context visual learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 6830–6839, 2023.

[125] Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. arXiv preprint arXiv:2109.01652, 2021.

[126] Thomas Wolf, Lysandre Debut, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. arXiv preprint arXiv:1910.03771, 2019.

[127] Shengqiong Wu, Hao Fei, Leigang Qu, Wei Ji, and Tat-Seng Chua. Next-gpt: Any-to-any multimodal llm. arXiv preprint arXiv:2309.05519, 2023.

[128] Yusheng Xiang and Marcus Geimer. Optimization of operation strategy for primary torque based hydrostatic drivetrain using artificial intelligence. arXiv preprint arXiv:2003.10011, 2020.

[129] Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. arXiv preprint arXiv:2304.05977, 2023.

[130] Mengwei Xu, Jiawei Liu, Yuanqiang Liu, Felix Xiaozhu Lin, Yunxin Liu, and Xuanzhe Liu. A first look at deep learning apps on smartphones. In The World Wide Web Conference, pages 2125–2136, 2019.

[131] Mengwei Xu, Yaozong Wu, Dongqi Cai, Xiang Li, and Shangguang Wang. Federated fine-tuning of billion-sized language models across mobile devices. arXiv preprint arXiv:2308.13894, 2023.

[132] Mengwei Xu, Wangsong Yin, Dongqi Cai, Rongjie Yi, Daliang Xu, Qipeng Wang, Bingyang Wu, Yihao Zhao, Chen Yang, Shihe Wang, et al. A survey of resource-efficient llm and multimodal foundation models. arXiv preprint arXiv:2401.08092, 2024.

[133] Mengwei Xu, Mengze Zhu, Yunxin Liu, Felix Xiaozhu Lin, and Xuanzhe Liu. Deepcache: Principled cache for mobile deep vision. In Proceedings of the 24th annual international conference on mobile computing and networking, pages 129–144, 2018.

[134] Yufei Xu, Jing Zhang, Qiming Zhang, and Dacheng Tao. Vitpose++: Vision transformer for generic body pose estimation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2023.

[135] Zheng Yang, Lirong Jian, Chenshu Wu, and Yunhao Liu. Beyond triangle inequality: Sifting noisy and outlier distance measurements for localization. ACM Transactions on Sensor Networks (TOSN), 9(2):1–20, 2013.

[136] Zheng Yang, Yi Zhao, Fan Dang, Xiaowu He, Jiahang Wu, Hao Cao, Zeyu Wang, and Yunhao Liu. Caas: Enabling control-as-a-service for time-sensitive networking. In IEEE INFOCOM, 2023.

[137] Rongjie Yi, Liwei Guo, Shiyun Wei, Ao Zhou, Shangguang Wang, and Mengwei Xu. Edgemoe: Fast on-device inference of moe-based large language models. arXiv preprint arXiv:2308.14352, 2023.

[138] Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. A survey on multimodal large language models. arXiv preprint arXiv:2306.13549, 2023.

[139] Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. Orca: A distributed serving system for {Transformer-Based} generative models. In 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22), pages 521–538, 2022.

[140] Koh Jing Yu. Model zoo. https://modelzoo.co/, 2023.

[141] Mu Yuan, Lan Zhang, Fengxiang He, and Xiang-Yang Li. Infi: end-to-end learnable input filter for resource-efficient mobile-centric inference. In Proceedings of the 28th Annual International Conference on Mobile Computing And Networking, pages 228–241, 2022.

[142] Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. arXiv preprint arXiv:2106.10199, 2021.

[143] Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. Glm-130b: An open bilingual pre-trained model. arXiv preprint arXiv:2210.02414, 2022.

[144] Qiyang Zhang, Xiang Li, Xiangying Che, Xiao Ma, Ao Zhou, Mengwei Xu, Shangguang Wang, Yun Ma, and Xuanzhe Liu. A comprehensive benchmark of deep learning libraries on mobile devices. In Proceedings of the ACM Web Conference, pages 3298–3307, 2022.

[145] Wuyang Zhang, Zhezhi He, Zhenhua Jia, Yunxin Liu, Marco Gruteser, Dipankar Raychaudhuri, and Yanyong Zhang. Elf: accelerate high-resolution mobile deep vision with content-aware parallel offloading. In Proceedings of the 27th Annual International Conference on Mobile Computing and Networking, pages 201–214, 2021.

[146] Xi Zhang, Feifei Zhang, and Changsheng Xu. Vqacl: A novel visual question answering continual learning setting. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 19102–19112, 2023.

[147] Yi Zhao, Zheng Yang, Xiaowu He, Jiahang Wu, Hao Cao, Liang Dong, Fan Dang, and Yunhao Liu. E-tsn: Enabling event-triggered critical traffic in time-sensitive networking for industrial applications. In 2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS), pages 691–701. IEEE, 2022.

[148] Yiqin Zhao and Tian Guo. Xihe: a 3d vision-based lighting estimation framework for mobile augmented reality. In Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services, pages 28–40, 2021.

[149] Quan Zhou, Haiquan Wang, Xiaoyan Yu, Cheng Li, Youhui Bai, Feng Yan, and Yinlong Xu. Mpress: Democratizing billion-scale model training on multi-gpu servers via memory-saving inter-operator parallelism. In 2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA), pages 556–569. IEEE, 2023.

[150] Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. Bert loses patience: Fast and robust inference with early exit. Advances in Neural Information Processing Systems, 33:18330–18341, 2020.

[151] Xingyi Zhou, Rohit Girdhar, Armand Joulin, Philipp Krähenbühl, and Ishan Misra. Detecting twenty-thousand classes using image-level supervision. In European Conference on Computer Vision, pages 350–368. Springer, 2022.

[152] Yongchao Zhou, Andrei Ioan Muresanu, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. arXiv preprint arXiv:2211.01910, 2022.