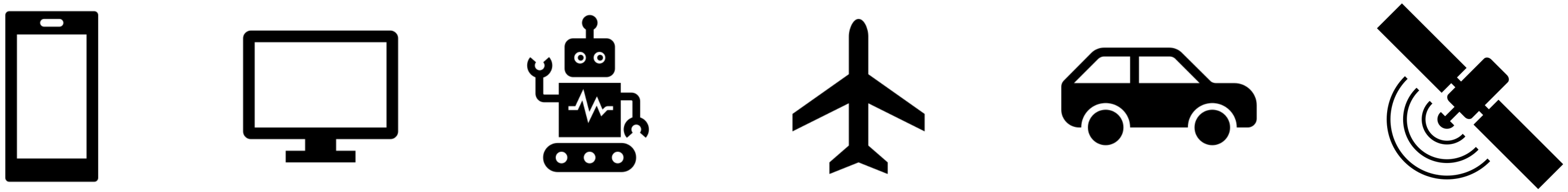
LLM Making Mobile Devices Smart at Next Level

Mengwei Xu (徐梦炜)
Beijing University of Posts and Telecommunications
https://xumengwei.github.io



Aren't they smart..Already?

- Yes, to a certain extent.

DNN-embedded mobile apps

- Increased by almost 10x (2018 to 2021)[1,2]
- Downloaded billions of times in one year
- Include almost every high-popularity app
- Up to 200+ DNNs in a single app[3]

[1] Mengwei Xu, et al. "A First Look at Deep Learning Apps on Smartphones". In the Web Conference (WWW) 2019
[2] Mario Almeida, et al. "Smart at what cost? Characterising Mobile Deep Neural Networks in the wild". In IMC 2021.
[3] Through offline communication with application developers.

Aren’t they smart.. Already?

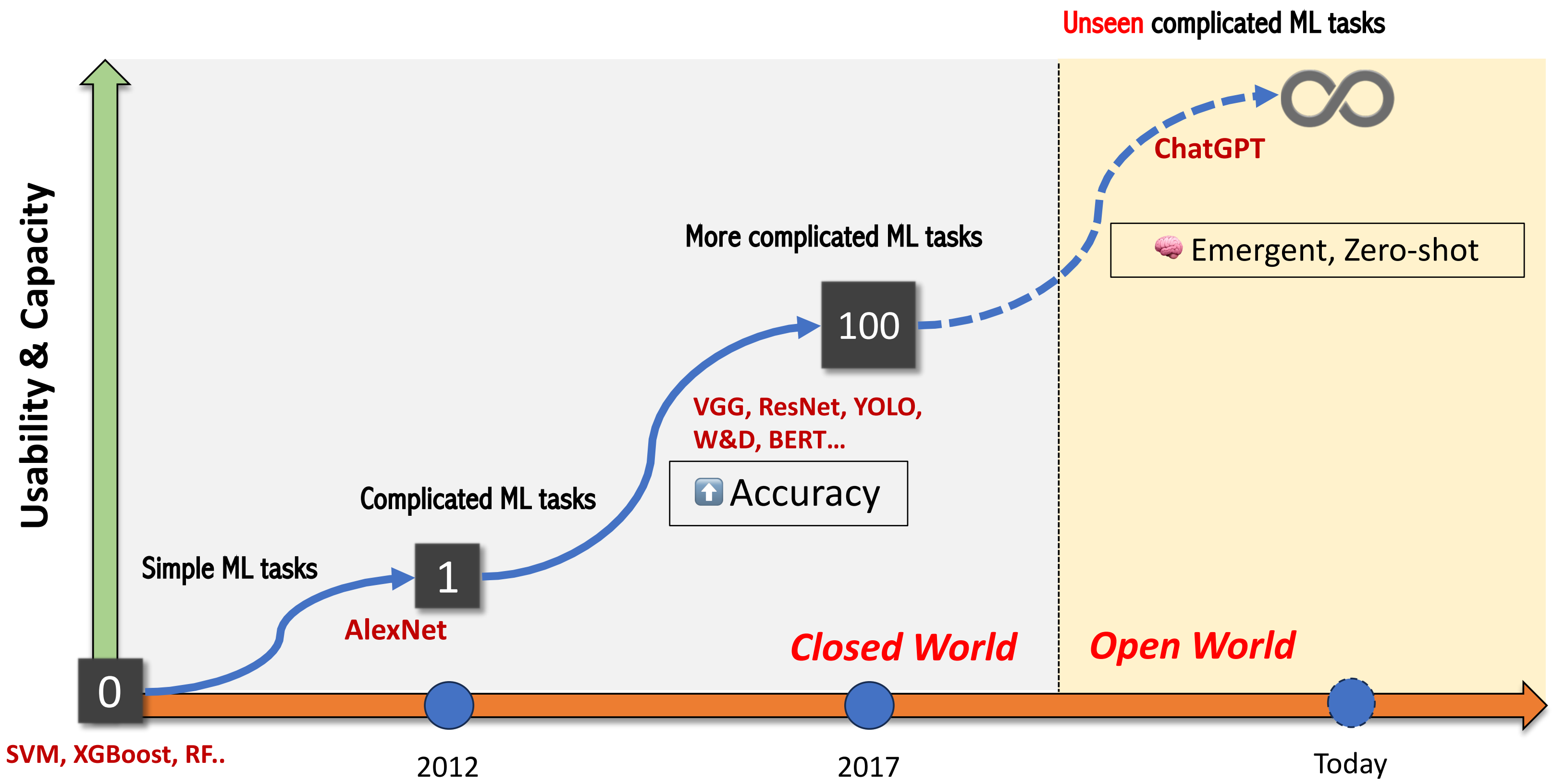
- Yes, to a certain extent.
- Yet, not even close to our expectation.

We expect a smart device to

- Understand open-vocabulary human language
- Operate/control itself accurately as humans do
- Action to human tasks in end-to-end manner

(Multimodal) LLM is an opportunity

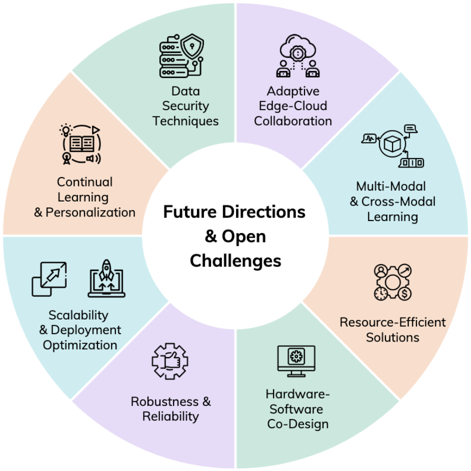
- To bring mobile devices the "next-level" intelligence



- Comprehend human language
- Zero-shot & in-context learning
- Multimodal alignment and input/output
- Reasoning & Planning
- Long context

On-device LLM is crucial

- On-device LLMs handle language tasks in a way that is ..
  - ✓ cost-efficient (important, obviously)
  - ✓ more available (even w/o network)
  - ✓ faster (not always)
  - ✓ privacy-preserving (very important, LLMs can leverage almost every bits of local data)



- LLMs on devices does not obviate mega-scale LLMs on clouds!
  - Creating music/poetry, solving math problems, etc.

[1] Jiajun Xu, et al. "On-Device Language Models: A Comprehensive Review". In preprint'24.

On-device LLM is crucial

- We already have a mobile device that can function with high intelligence!

A *mobile device* that can comprehend, reason, and plan without a cloud!

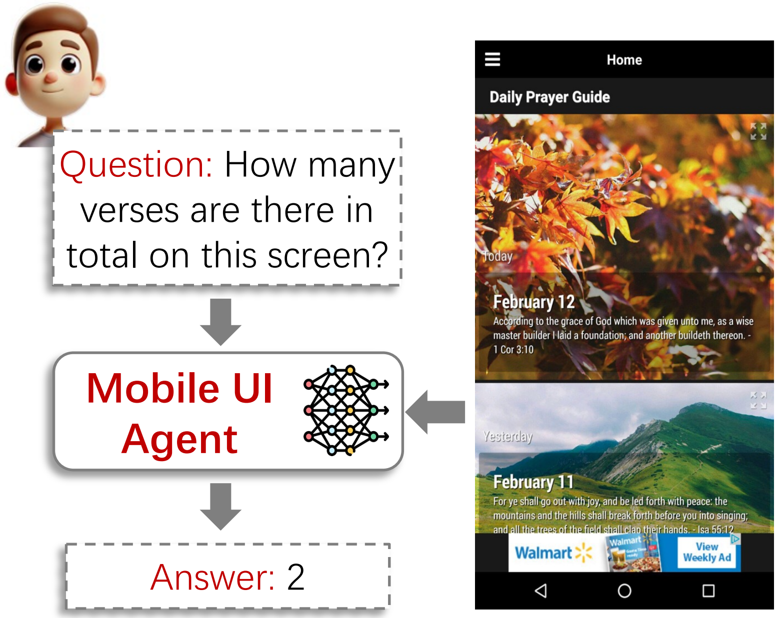
The fundamental differences for integrating LLM into mobile devices

(compared to traditional DNN-powered apps)

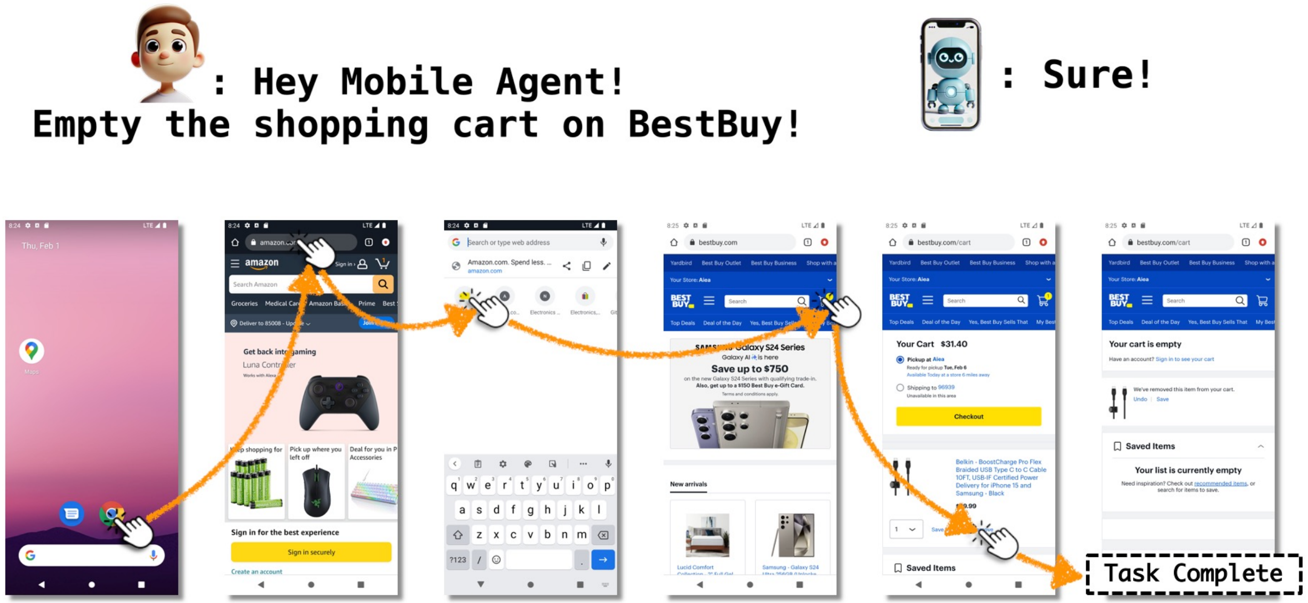
Workload: **Agent**	OS: **LLM-native**	Hardware: **DSA**

Fundamental diffs in LLM Era(1/3)

- An all-in-one killer app: **personal agent (assistant)**



Screen QA



Task Automation

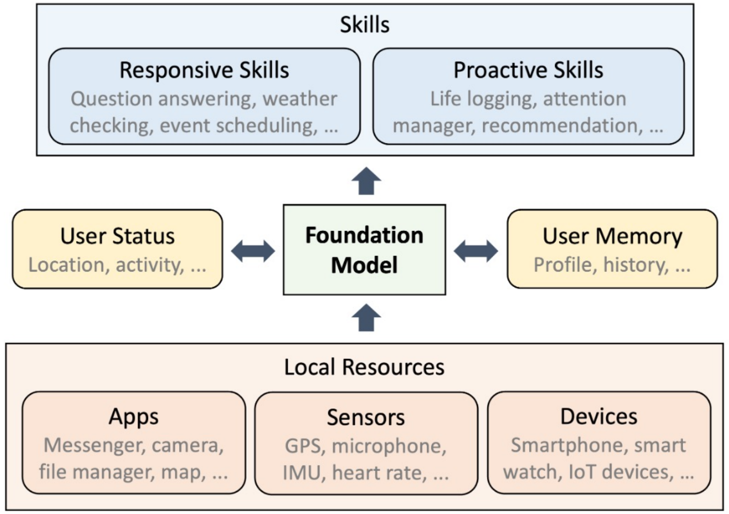
Freeing us from tedious work; making electronic devices benefit more people

Fundamental diffs in LLM Era[1/3]

- An all-in-one killer app: **personal agent (assistant)**

With following unique and exciting features:

1. End-to-end: instruction in, response out
2. Context-aware, personalized, and stateful
   - Long prompt (prefill time often dominates)
   - Can leverage almost every bit of data. Privacy!
3. No longer event-driven (only), but also proactively sense and plan when device is not in use



[1] Yuanchun Li, et al. "Personal LLM Agents: Insights and Survey about the Capability, Efficiency, and Security". In preprint'24.

Fundamental diffs in LLM Era(1/3)

- An all-in-one killer app: **personal agent (assistant)**

	1980s	2000s	2010s	2020s
	PC Era	Internet Era	Mobile Era	AI Era
Software Paradigm	Computer Programs	Websites	Smartphone Apps	Personal LLM Agents
Key Characteristic	Efficient GUI-based user interaction	Convenient information access	Always-connected personalized services	Ubiquitous intelligent automation
Examples	Word, Excel, ...	Google, Yahoo, ...	WeChat, U	

2016~2017	2018~2020	2021~2022	2023	2024~
开启Mobile AI时代	个人终端AI化	全场景设备AI化	AI大模型赋能终端	全新原生智能OS
搭载神经网络处理单元	智慧语音	智能座舱	接入大模型的全新小艺	统一AI系统底座
AI智慧影像	智慧识屏	语音助手覆盖全场景设备	高阶智能驾驶	小艺超级智能体
AI翻译	智慧视觉	小艺建议	智慧搜图	原生智能应用
AI慧眼识物	HiAI能力开放	智慧屏/智能座舱可视可说	文档摘要	全面开放AI生态
AI自拍	情景智能	多设备协同唤醒	照片AI云增强	
AI高倍变焦	AI隔空操控			
AI引擎	AI智感支付			
	AI信息保护			
	AI字幕			

Hype or Reality?

苹果发布多模态模型 Ferret-UI，部分手机 UI 任务超越 GPT-4V

赖文昕 AI科技评论 2024年04月10日 12:49 广东

支付宝突然推出新App，竟想用AI让日常生活开挂

荣耀MagicOS 9.0来了个全局智能体，AI手机方向变了

Original 关注AI智能体的 机器之心 2024年10月23日 20:21 北京

八年磨一剑，国产首款AI Agent手机跑赢苹果！手机学会「自动驾驶」秀翻全场

让AI像人类一样操作手机，华为也做出来了

机器之心 2024年10月25日 17:29 北京

控制电脑手机的智能体人人都能造，微软开源OmniParser

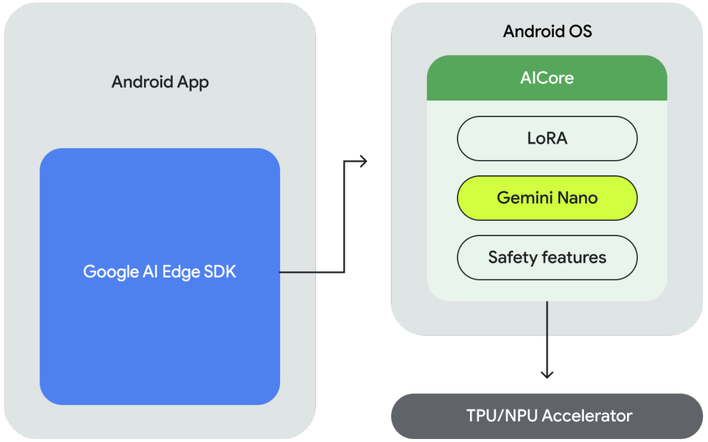
关注智能体的 机器之心 2024年10月26日 12:19 北京

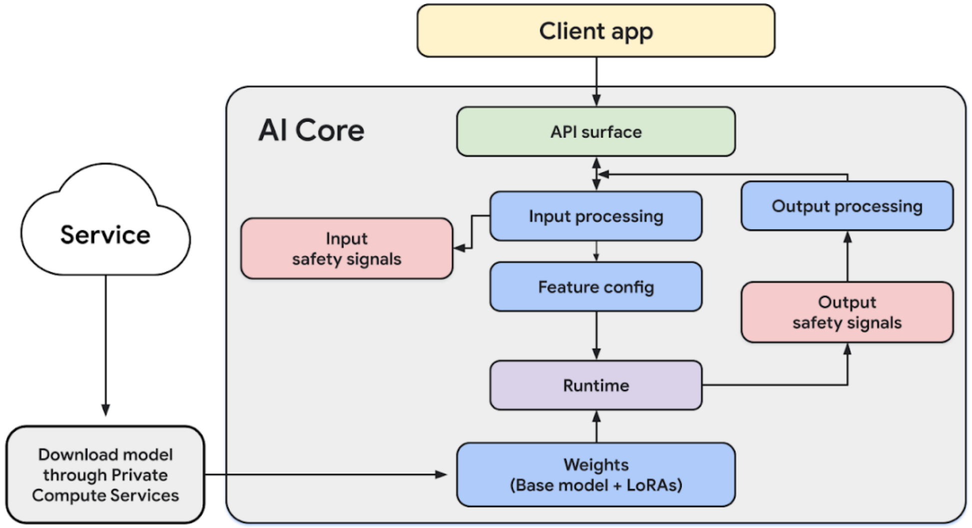
[1] Yuanchun Li, et al. “Personal LLM Agents: Insights and Survey about the Capability, Efficiency, and Security”. In preprint’24.
[2] 华为. “AI终端白皮书：AI与人协作、服务于人”. 2024.

Fundamental diffs in LLM Era(2/3)

- LLM integrated into OS as a **system service** (LLMaaS)
  - Scales to infinite number of tasks
  - Hardware-design-friendly
  - OS gains full visibility into LLM requests

Google AI Core Service





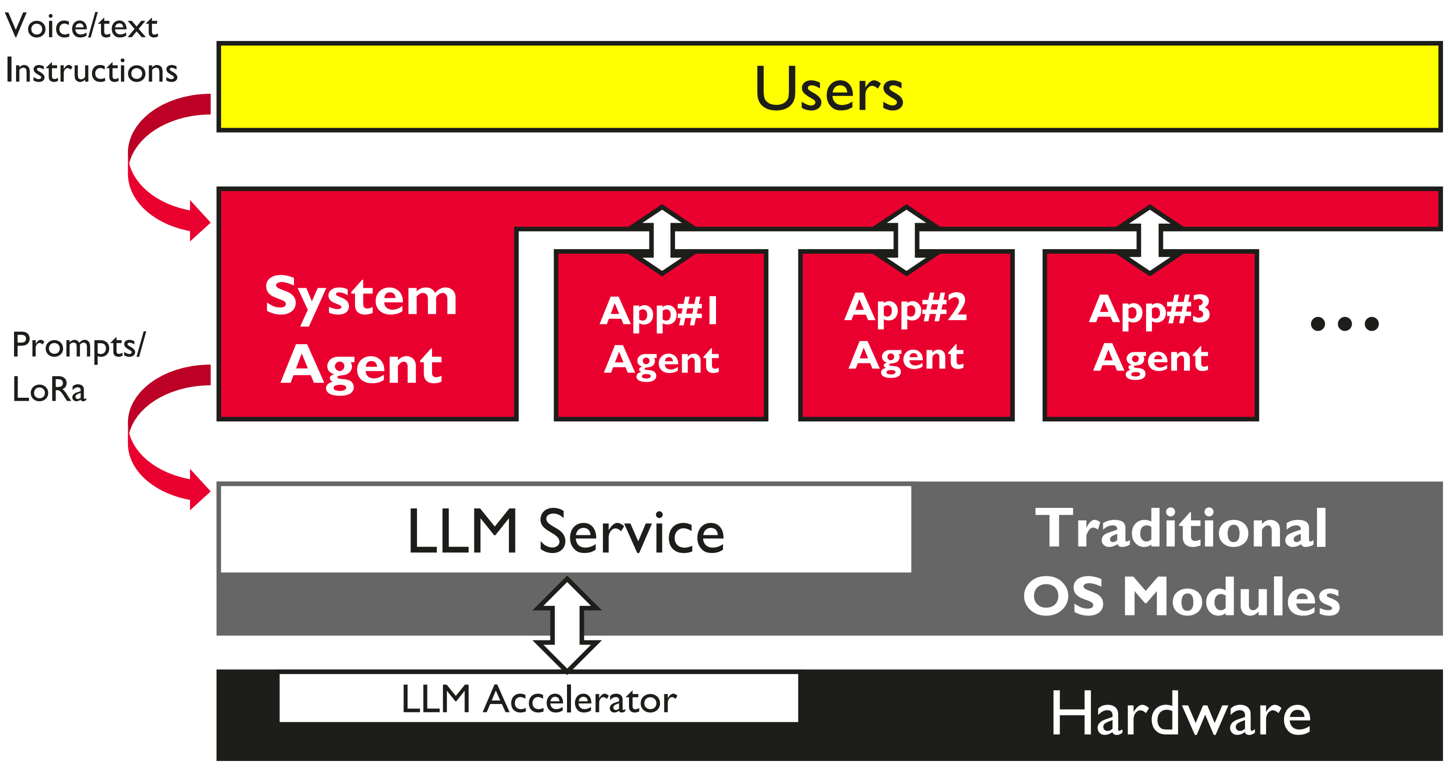
[1] Source: https://developer.android.com/ai/gemini-nano

Fundamental diffs in LLM Era$^{(2/3)}$

- LLM integrated into OS as a **system service** (LLMaaS)
  - Scales to infinite number of tasks
  - Hardware-design-friendly
  - OS gains full visibility into LLM requests
- Opening new research opportunities and challenges
  - *Efficiency*: how to schedule, batch, and cache-reuse system-wise LLM requests? How to manage the LLM context states across apps?
  - *Security*: how to protect app-owned LoRa? How to isolate cross-app requests?
  - *Usability*: how to upgrade LLM? How to design LLMaaS interface?
  - Etc..

Fundamental diffs in LLM Era(2/3)

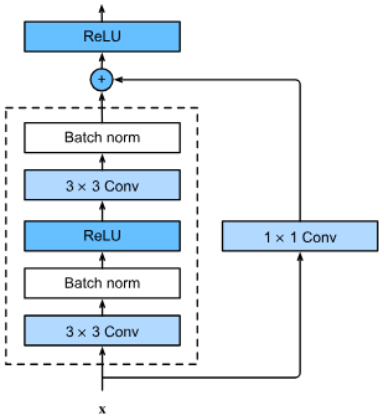
- LLM integrated into OS as a **system service** (LLMaaS)



One LLM
Many Agents

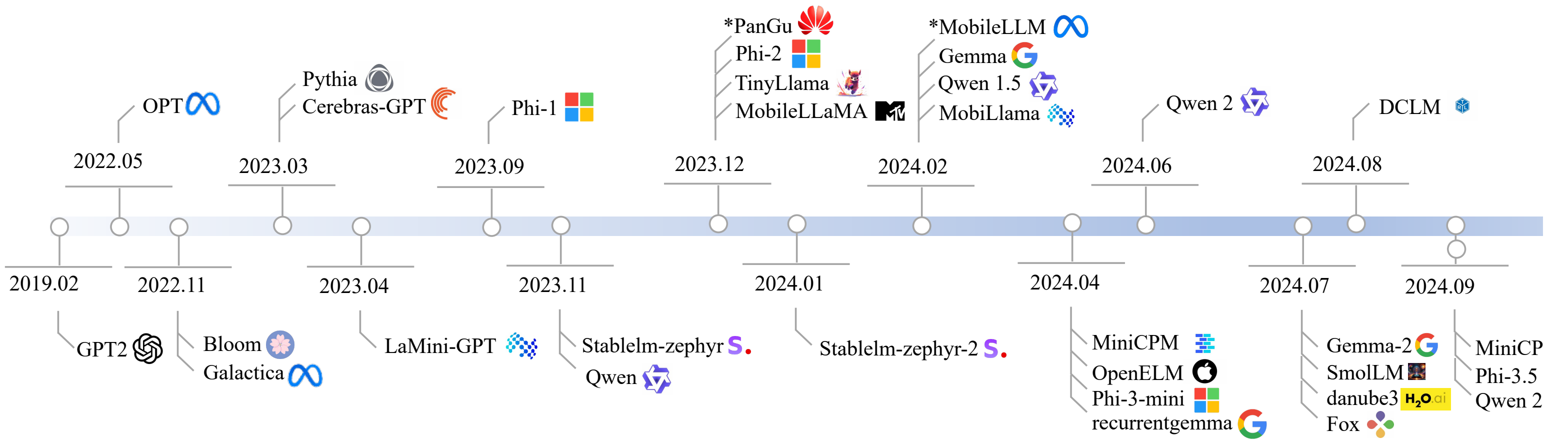
Fundamental diffs in LLM Era(3/3)

- On-device **resource scarcity** further exacerbated.



ResNet, YoLo, LSTM, etc
(<200M)
<100ms to process one image
<100MB memory footprint
Easy to quantize (integer-only)
Static shape and cost

vs.



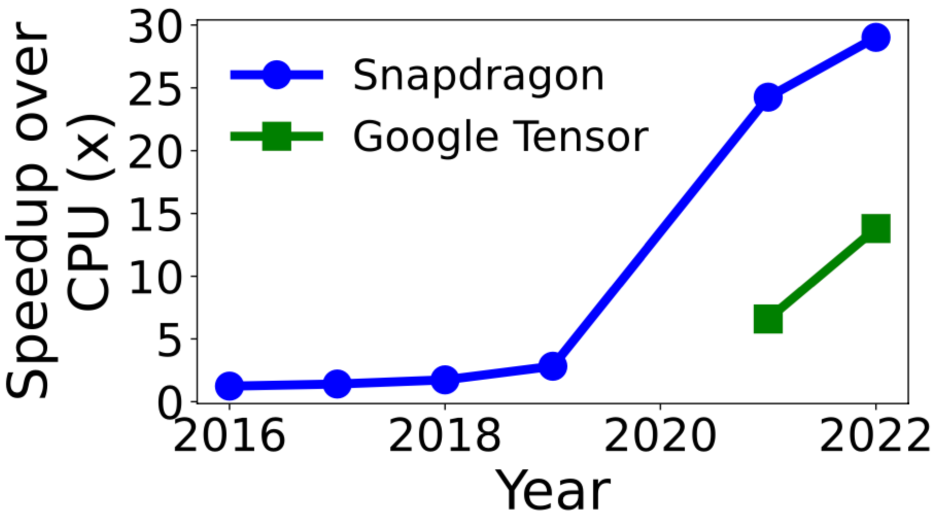
Small Language Models (1B~5B)

>10sec to process one prompt on CPU
>1GB memory footprint
Difficult to quantize (FP required)
Dynamic shape and increased cost with longer prompt

[1] Zhenyan Lu, et al. "Small Language Models: Survey, Measurements, and Insights". In preprint'24.

Fundamental diffs in LLM Era(3/3)

- On-device **resource scarcity** further exacerbated.
- DSA (NPU) is the answer to practical on-device LLM.



- *The gap between CPU/GPU and NPU increases over time*
- *Moore's law still stands for NPU*

[1] Jinliang Yuan. "Mobile Foundation Model as Firmware". In MobiCom'24.

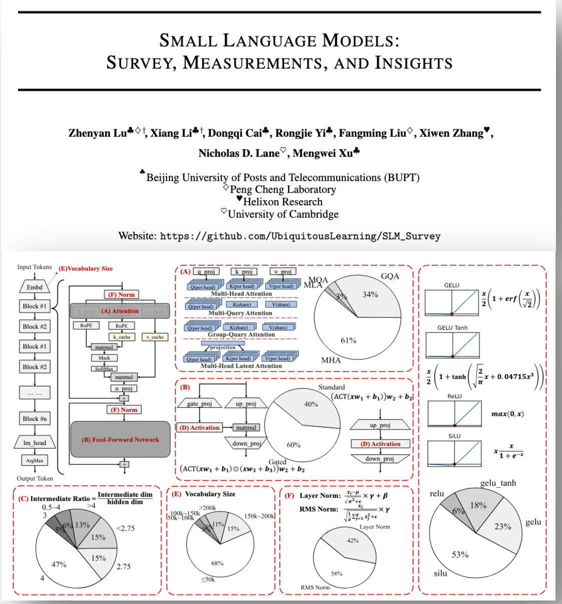
Call for full-stack design and opts

- Our response: **agent-model-runtime-OS** co-design

Agent	Device control and GUI agents **testbed** [LlamaTouch, UIST'24], **datasets** [PhoneLM, preprint'24][DroidCall, preprint'24], and **privacy enhancements**[SILENCE, NeurIPS'24]
Model	A training-from-scratch, fully-reproducible **SLM family** [PhoneLM, preprint'24], Any-to-any modality **mobile foundation model** [M4, MobiCom'24] , and **Federated LLM** techniques [FwdLLM, ATC'24][AdaFL, MobiCom'23] [FeS, MobiCom'23]
Runtime	Acceleration through **NPU** [llm.npu, ASPLOS'25], **SpecDecoding** [LLMCad, preprint'23] , **Sparsity** [EdgeMoE, preprint'23], **Early Exiting** [Recall, preprint'24], etc
OS	**LLMaaS Context Management** [LLMS, preprint'24] and **QoS** [ELMS, preprint'24]

Call for full-stack design and opts

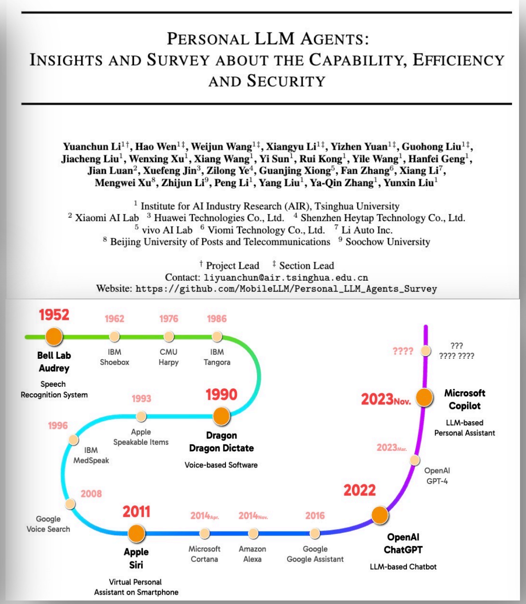
- Our response: **agent-model-runtime-OS** co-design


Small Language Models: Survey, Measurements, and Insights

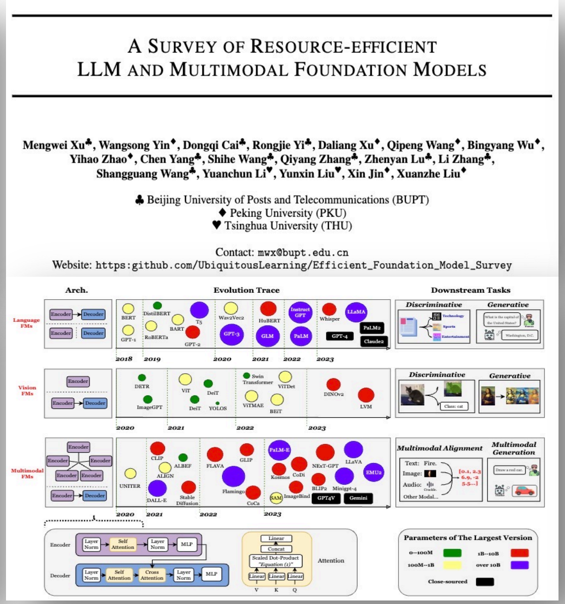
Zhenyan Lu, Xiang Li, Dongqi Cai, Rongjie Yi, Fangming Liu, Xiwen Zhang, Nicholas D. Lane, Mengwei Xu

Beijing University of Posts and Telecommunications (BUPT)
Peng Cheng Laboratory
Helixon Research
University of Cambridge

Website: https://github.com/UbiquitousLearning/SLM_Survey

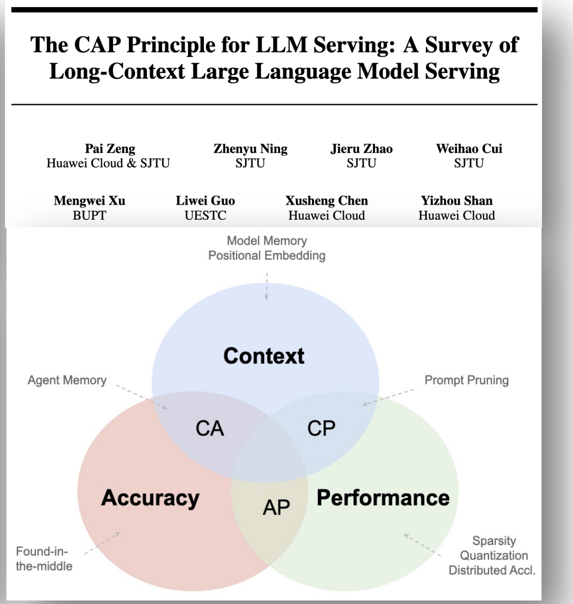

Personal LLM Agents: Insights and Survey about the Capability, Efficiency and Security

Website: https://github.com/MobileLLM/Personal_LLM_Agents_Survey


A Survey of Resource-efficient LLM and Multimodal Foundation Models

Beijing University of Posts and Telecommunications (BUPT)
Peking University (PKU)
Tsinghua University (THU)

Contact: mwx@bupt.edu.cn
Website: https://github.com/UbiquitousLearning/Efficient_Foundation_Model_Survey


The CAP Principle for LLM Serving: A Survey of Long-Context Large Language Model Serving

Pai Zeng (Huawei Cloud & SJTU), Zhenyu Ning (SJTU), Jieru Zhao (SJTU), Weihao Cui (SJTU), Mengwei Xu (BUPT), Liwei Guo (UESTC), Xusheng Chen (Huawei Cloud), Yizhou Shan (Huawei Cloud)

[1] "Small Language Models: Survey, Measurements, and Insights", Zhenyan Lu, et al.
[2] "Personal LLM Agents: Insights and Survey about the Capability, Efficiency and Security", Yuanchun Li, et al.
[3] "A Survey of Resource-efficient LLM and Multimodal Foundation Models", Mengwei Xu, et al.
[4] "The CAP Principle for LLM Serving: A Survey of Long-Context Large Language Model Serving", Pai Zeng, et al.

An e2e demo

DroidCall: A Dataset for LLM-powered Android Intent Invocation

Weikai Xie[1] Li Zhang[1] Shihe Wang[1] Rongjie Yi[1] Mengwei Xu[1]

Agent

PhoneLM: an Efficient and Capable Small Language Model Family through Principled Pre-training

Rongjie Yi[1] Xiang Li[1] Weikai Xie[1] Zhenyan Lu[1] Chenghua Wang[1] Ao Zhou[1] Shangguang Wang[1] Xiwen Zhang[2] Mengwei Xu[1]

Model

Code · Issues 9 · Pull requests 2 · Actions · Security 2 · Insights

mllm Public · Watch 17 · Fork 58 · Starred 516

Empowering 1000 tokens/second on-device LLM prefilling with mllm-NPU

Daliang Xu[♦], Hao Zhang[◇], Liming Yang[♦], Ruiqi Liu[♦], Gang Huang[♦], Mengwei Xu[◇*], Xuanzhe Liu[♦]
[♦]Peking University, [◇]Beijing University of Posts and Telecommunications

Runtime

Call for full-stack design and opts

- Our response: **agent-model-runtime-OS** co-design

Agent

Model

Runtime

OS

Device control and GUI agents **testbed** [LlamaTouch, UIST'24], **datasets** [PhoneLM, preprint'24][DroidCall, preprint'24], and **privacy enhancements**[SILENCE, NeurIPS'24]

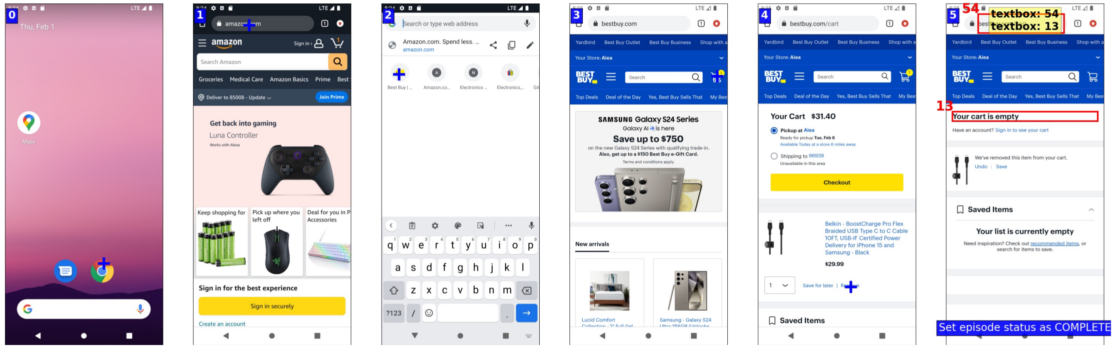
A training-from-scratch, fully-reproducible **SLM family** [PhoneLM, preprint'24], Any-to-any modality **mobile foundation model** [M4, MobiCom'24] , and **Federated LLM** techniques [FwdLLM, ATC'24][AdaFL, MobiCom'23] [FeS, MobiCom'23]

Acceleration through **NPU** [llm.npu, ASPLOS'25], **SpecDecoding** [LLMCad, preprint'23] , **Sparsity** [EdgeMoE, preprint'23], **Early Exiting** [Recall, preprint'24], etc

LLMaaS Context Management [LLMS, preprint'24] and **QoS** [ELMS, preprint'24]

Prefilling stage is the bottleneck

- **On-device LLM tasks demand long prompt (context)**



Each UI is hundreds to thousands of tokens

(either as image or view hierarchy)

- **Mobile LLMs can support long context**

Model	Max Context	Year	Model	Max Context	Year
Opt-1.3B	2K	2022.5	TinyLLaMA-1.1B	2K	2023.9
StableLLM-3B	4K	2023.10	phi-2-2.7B	2K	2023.12
Gemma-2B	8K	2024.2	Qwen1.5-1.8B	32K	2024.2
Phi3-mini-3.8B	128K	2024.5	Qwen2-1.5B	32K	2024.6

From 2K to 128K

- **Mobile Processors (CPU/GPU) do not support high parallelism as A100**

So, prefilling dominates the end-to-end LLM inference delay (>90% in most cases)

The opportunity: mobile NPU

- **Almost every smartphone has NPU**

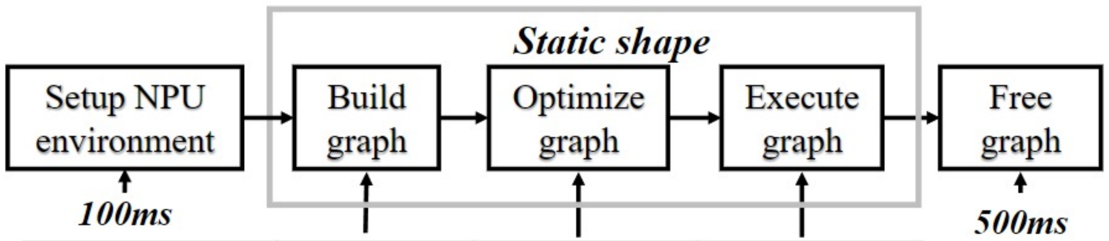
Vendor	Latest NPU	SDK	Open	Group	INT8 Perf.
Qualcomm	Hexagon NPU [15]	QNN [23]	×	×	73 TOPS
Google	Edge TPU [17]	Edge TPU API [7]	×	×	4 TOPS
MediaTek	MediaTek APU 790 [11]	NeuroPilot [13]	×	N/A	60 TOPS
Huawei	Ascend NPU [6]	HiAI [9]	×	×	16 TOPS

"Open": Open-source?; "Group": Support per-group quantization MatMul? "N/A": No available documents for public; "INT8 Perf.": Int8 performance.

Up to 73 TOPS, way more powerful than CPU/GPU

The opportunity: mobile NPU

- **Almost every smartphone has NPU**
- But, none of existing LLM systems support mobile NPU.
  1) NPU supports only static shape
  2) NPU is good at Integer Ops (INT8), but bad at FP Ops
  3) NPU vendor libs do not support group-level quantization



Qwen1.5-1.8B	450 ms	3.30 s	149 ms
Gemma-2B	360 ms	11.54 s	108 ms

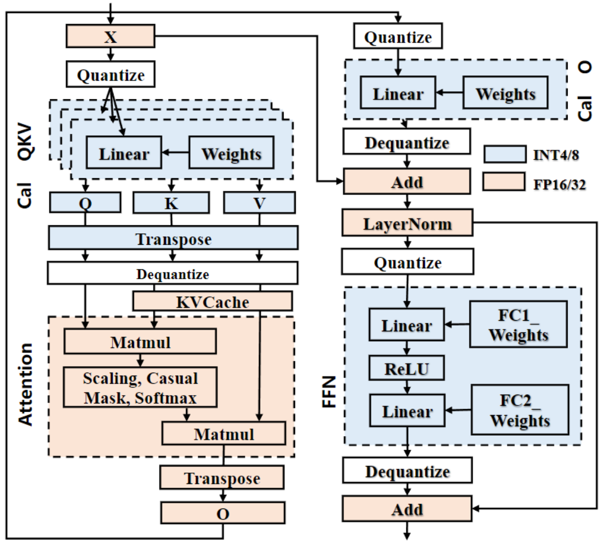
Re-building NPU execution graph is too costly!

The opportunity: mobile NPU

- **Almost every smartphone has NPU**
- But, none of existing LLM systems support mobile NPU.
  1) NPU supports only static shape
  2) NPU is good at Integer Ops (INT8), but bad at FP Ops
  3) NPU vendor libs do not support group-level quantiz

Quantization	Type	Acc.	Cal QKV	Atten.	Cal O	Norm.	FFN
K-Quant [54]	Per-Group	Low	INT8	FP16	INT8	FP16	INT8
GPTQ [33]	Per-Group	High	FP16	FP16	FP16	FP16	FP16
AWQ [52]	Per-Group	High	FP16	FP16	FP16	FP16	FP16
SmoothQuant [77]	Per-tensor	Low	INT8	FP16	INT8	FP16	INT8

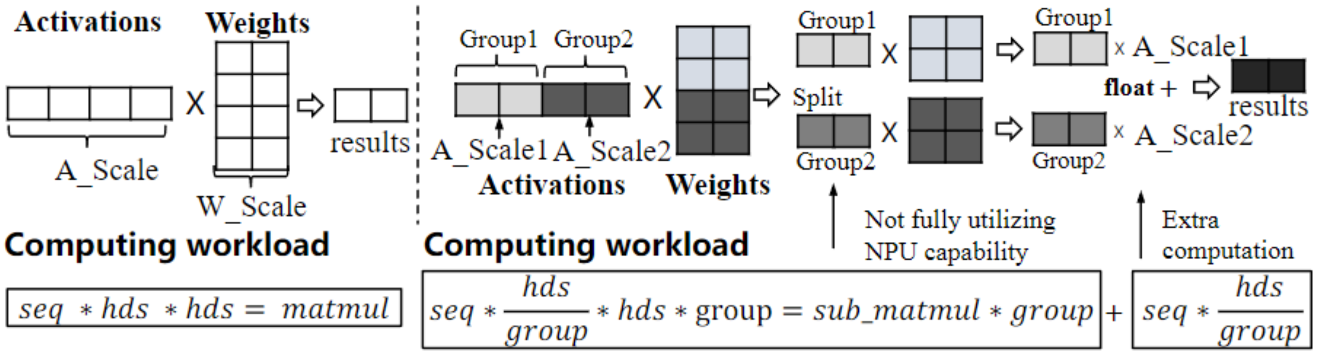
"Atten.":Attention; "Norm.": Normalization.



FP operations cannot be eliminated!

The opportunity: mobile NPU

- **Almost every smartphone has NPU**
- But, none of existing LLM systems support mobile NPU.
    1) NPU supports only static shape
    2) NPU is good at Integer Ops (INT8), but bad at FP Ops
    3) NPU vendor libs do not support group-level quantization

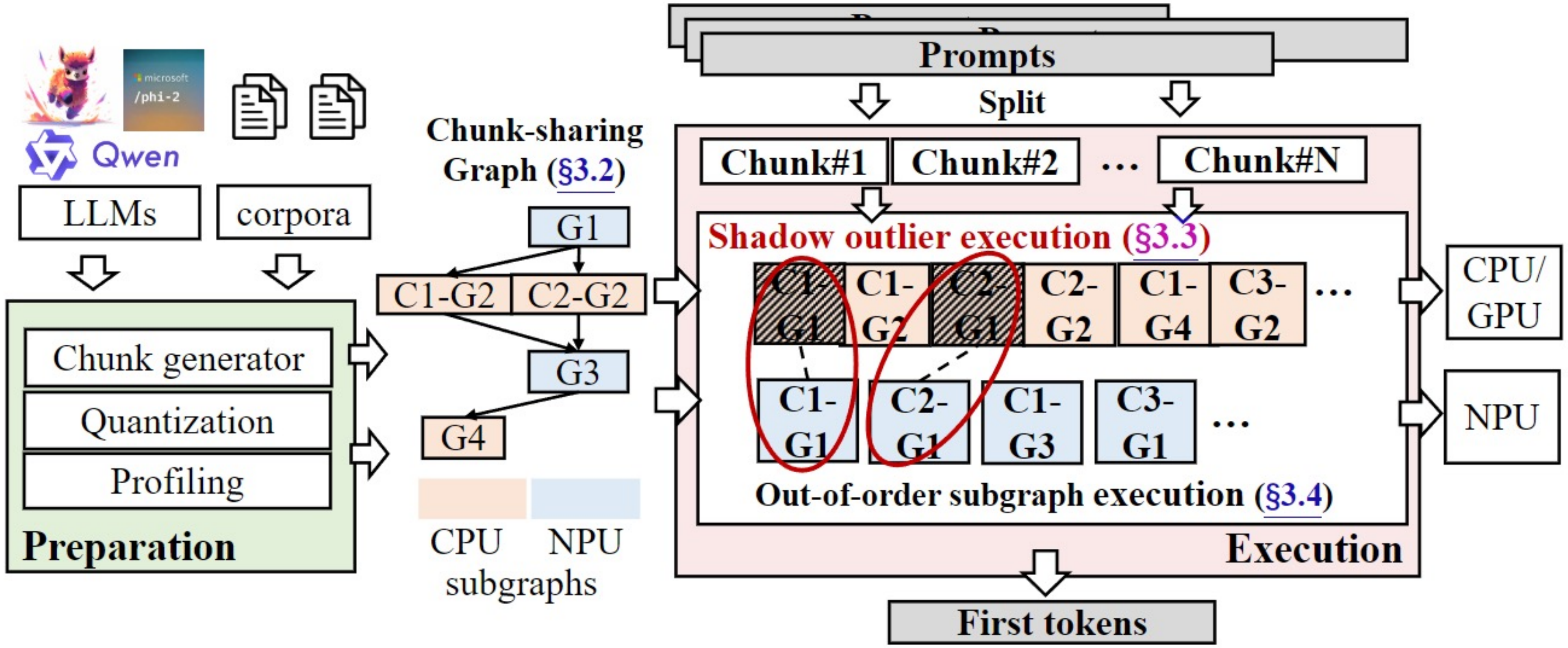


(a) Per-tensor (b) Per-group

Why group-level quantization, not tensor-level? Outliers!

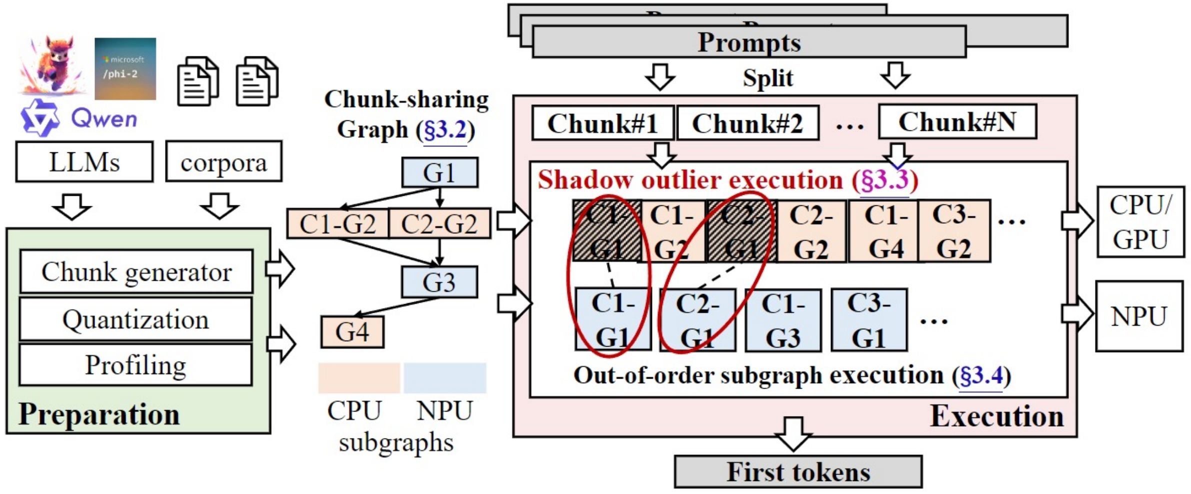
llm.npu: the first LLM engine for mobile NPU

- Overall idea: split the prompts to fixed-sized chunk; offloading FP Ops and outlier execution to CPU/GPU; properly scheduling them across NPU and CPU/GPU.

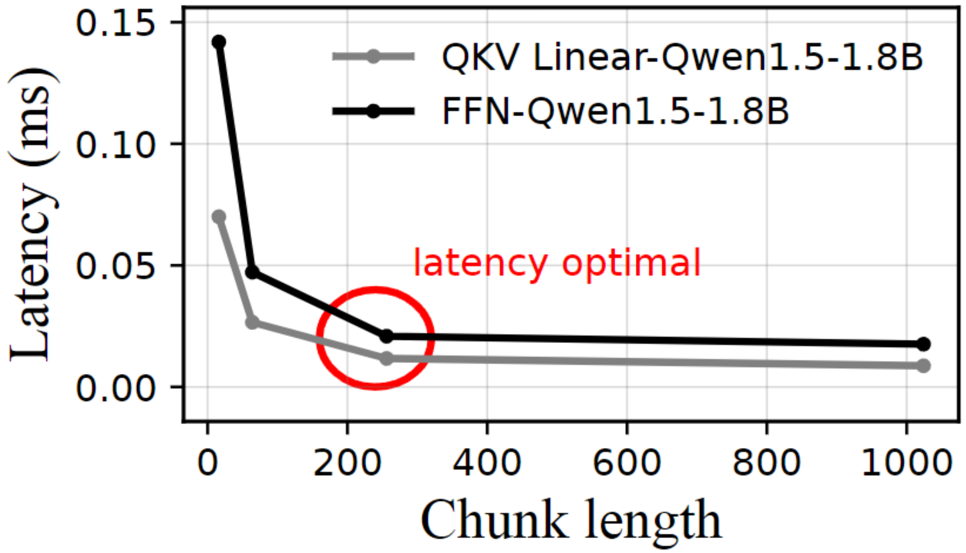


llm.npu: the first LLM engine for mobile NPU

- Key Technique #1: chunk-sharing graph execution
  - Challenge: too many subgraphs
  - Solution: shared static-shaped Ops across chunks; 75% memory saved.
    - ✓Static shaped: Linear, LayerNorm, etc
    - ✓Dynamic shaped: Attention

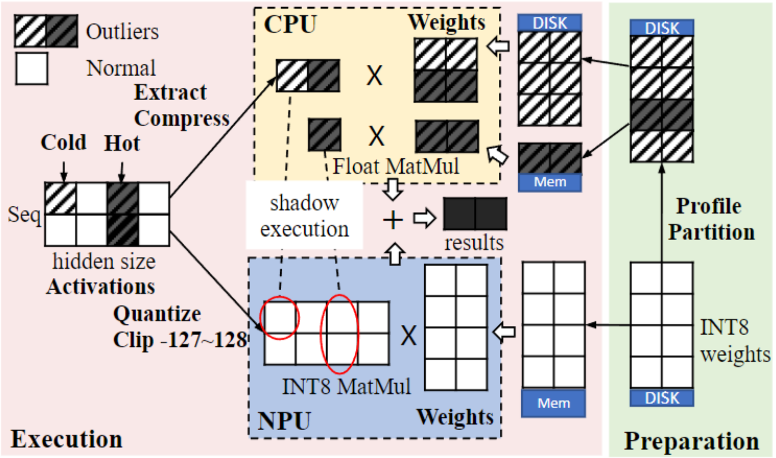


Empirically determining a chunk size

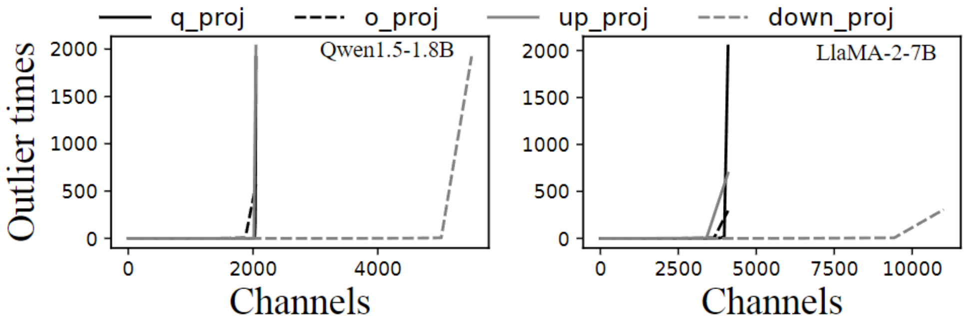


llm.npu: the first LLM engine for mobile NPU

- Key Technique #2: shadow outlier execution
  - Challenge(1/2): weights memory doubled since NPU and CPU do not share memory space
  - Solution: keep only hot weights channels (needed by outlier execution) in CPU memory space, and retrieve others from disk on demand.

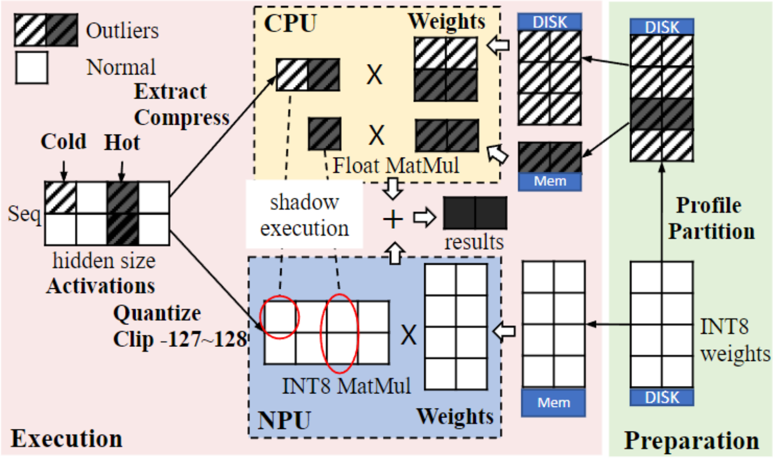


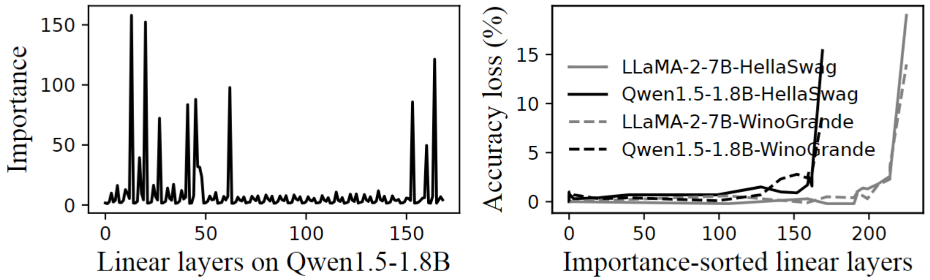
Overall: <3% channels contribute >80% outliers



llm.npu: the first LLM engine for mobile NPU

- Key Technique #2: shadow outlier execution
  - Challenge(2/2): while outlier exec. is fast, its synchronization with CPU incurs non-trivial overhead
  - Solution: outlier pruning.
    - ✓>85% layers' outliers can be pruned without compromising accuracy



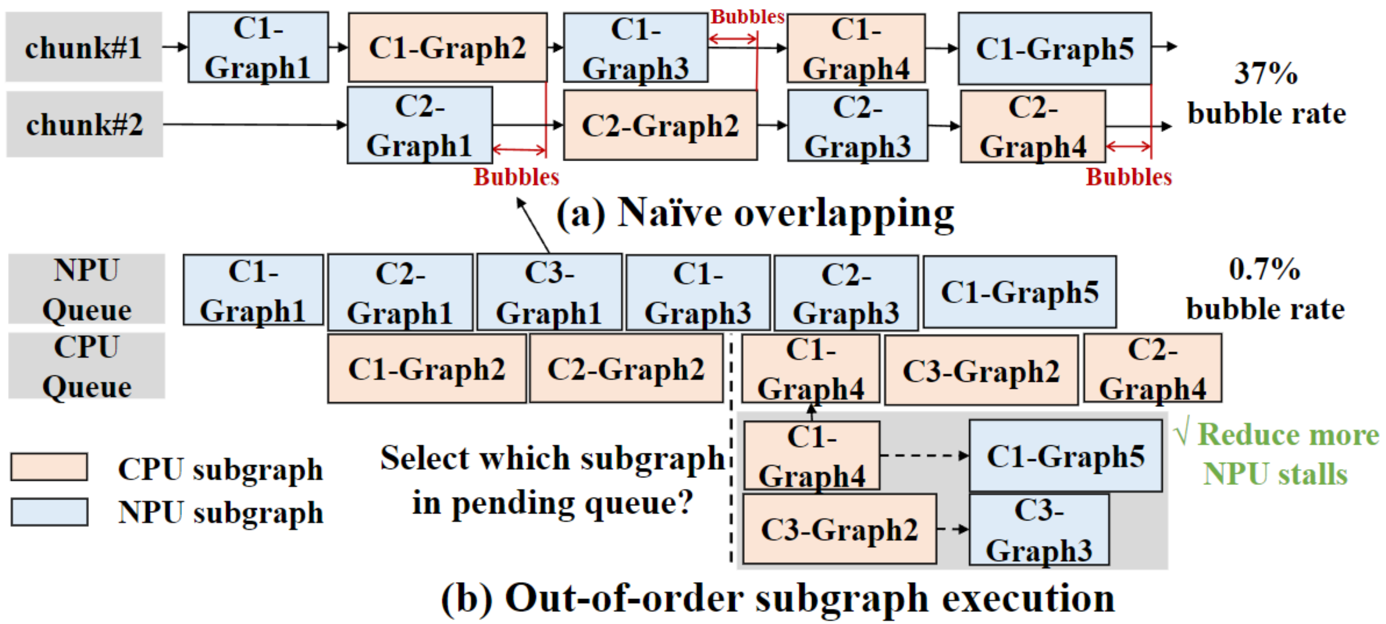


Top and bottom layers' outliers are more importance

It's okay to prune most layers' outliers

llm.npu: the first LLM engine for mobile NPU

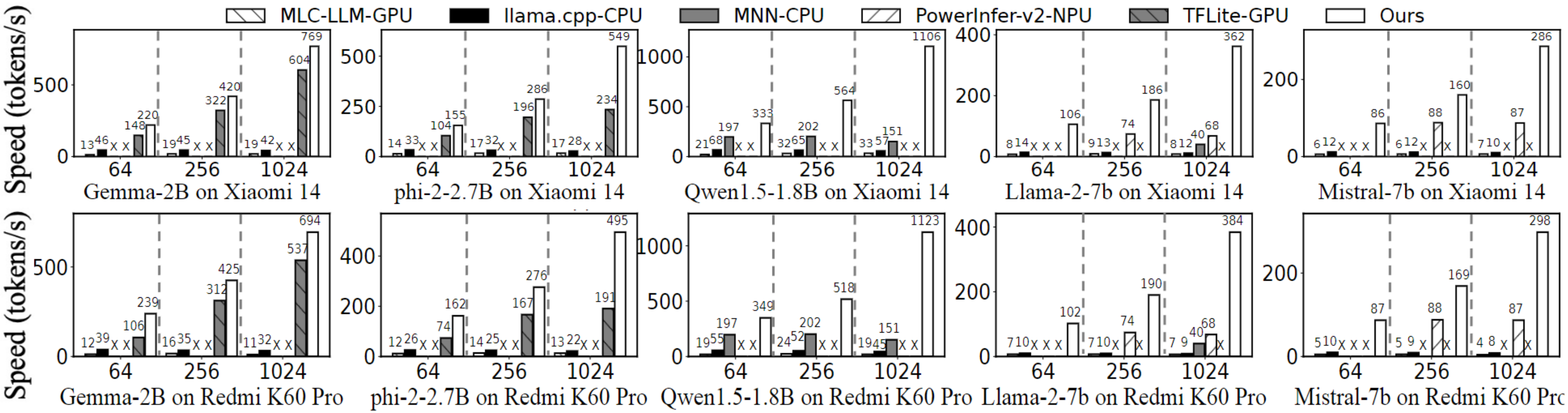
- Key Technique #3: out-of-order subgraph execution
  - Challenge: low HW utilization; NP-hard complexity
  - Solution: out-of-order execution (dependency-aware); a heuristic scheduling algorithm.



Minimize the bubble of NPU.

Highlighted results

Prefill speed under different prompt lengths on different devices (datasets: Longbench-2wiki-Multi-doc QA)
Baselines: MLC-LLM (GPU), llama.cpp (CPU), MNN (CPU), PowerInfer-v2 (NPU), TFLite (GPU)



7.3×–18.4×faster than baselines on CPU, and 1.3×–43.6× on GPU with prompt length of 1024
Achieves >1000 tokens/second on Qwen1.5-1.8B (for the first time)

Highlighted results

End-to-end latency comparison across different frameworks using real mobile applications execution on Xiaomi 14
Baselines: MLC-LLM (GPU), llama.cpp (CPU), MNN (CPU), PowerInfer-v2 (NPU), TFLite (GPU)

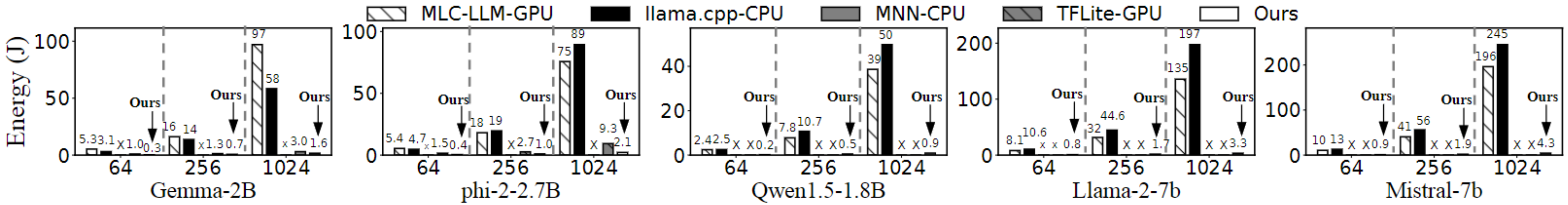
LLM	**Datasets**	**MLC**	**LCPP**	**MNN**	**PI**	**TFLite**	**Ours**	**Speedup**	**Datasets**	**MLC**	**LCPP**	**MNN**	**PI**	**TFLite**	**Ours**	**Speedup**
Qwen1.5-1.8B	Longbench: 2wiki -Multi-doc QA (prompt length: 1500 tokens)	45.6	26.7	10.6	-	-	1.7	**6.2-26.8×**	Longbench: TriviaQA (prompt length: 1500 tokens)	46.0	27.0	11.2	-	-	2.0	**5.6-23.0×**
Gemma-2B		78.4	34.6	-	-	2.6	1.9	**1.4-41.3×**		81.8	36.2	-	-	2.8	2.2	**1.3-37.2×**
Phi-2-2.7B		87.0	53.3	13.0	-	6.3	3.1	**2.0-28.1×**		91.4	56.3	14.7	-	6.8	3.6	**1.9-25.4×**
LlaMA-2-7B		184.7	146.0	22.4	19.8	-	5.3	**3.7-34.8×**		197.3	156.2	23.8	21.8	-	6.2	**3.5-31.8×**
Mistral-7b		254.2	200.2		20.0	-	5.5	**3.6-46.2×**		266.2	210.0	-	21.5	-	6.4	**3.4-41.6×**
Geo-mean (speedup)		**34.7×**	**21.8×**	**4.8×**	**3.7×**	**1.7×**	-			**31.0×**	**19.6×**	**4.4×**	**3.4×**	**1.6×**	-	
LLM	**Datasets**	**MLC**	**LCPP**	**MNN**	**PI**	**TFLite**	**Ours**	**Speedup**	**Datasets**	**MLC**	**LCPP**	**MNN**	**PI**	**TFLite**	**Ours**	**Speedup**
Qwen1.5-1.8B	DroidTask: clock (prompt length: 800 tokens))	21.0	10.4	3.9	-	-	1.4	**2.8-15.0×**	DroidTask: applauncher (prompt length: 600 tokens)	16.2	8.1	3.1	-	-	1.1	**2.8-14.7×**
Gemma-2B		39.4	16.5	-	-	2.5	1.2	**2.1-32.8×**		29.4	12.3	-	-	1.9	0.9	**2.1-32.7×**
Phi-2-2.7B		46.6	25.0	7.4	-	4.2	3.1	**1.4-15.0×**		35.4	19.0	5.9	-	3.2	2.4	**1.3-14.8×**
LlaMA-2-7B		87.7	60.4	10.6	11.1	-	4.8	**2.2-18.3×**		63.7	43.9	7.7	8.2	-	3.6	**2.1-17.7×**
Mistral-7b		122.3	68.6	-	12.0	-	4.9	**2.4-25.0×**		90.1	50.6	-	8.9	-	3.8	**2.3-23.7×**
Geo-mean (speedup)		**20.2×**	**10.8×**	**2.4×**	**2.4×**	**1.7×**	-			**19.7×**	**10.5×**	**2.5×**	**2.3×**	**1.7×**	-	

**LCPP* and *PI* in the first row represent llama.cpp and PowerInfer-V2, respectively.

23.0–46.2× over llama.cpp-CPU, 16.5–36.4×over MLC-LLM-GPU, 4.08–4.19× over MNN-CPU, 3.51–3.73×over PowerInfer-V2-NPU, and 1.27–2.03× over TFLite-GPU

Highlighted results

Energy consumption under different prompt lengths on Redmi K60 Pro (datasets: Longbench-2wiki-Multi-doc QA)
Baselines: MLC-LLM (GPU), llama.cpp (CPU), MNN (CPU), PowerInfer-v2 (NPU), TFLite (GPU)



1.9×–59.5× energy reduction compared to baselines

Room to improve: eliminate CPU/GPU in end-to-end execution

Takeaways

- On-device LLM is reinventing the mobile devices
  - A total paradigm shift of mobile AI ecosystem
  - The future of LLM is hybrid (device-cloud)

- It calls for full-stack LLM research
  - OS, runtime, model, and application (agent)