WWDC

2024看AI未来

WWDC24

WWDC 2024 启示录

1. 洞察端测模型、云测模型的未来，以及Apple自研的大模型
2. 端测（C 端）AI 应用是否会进入爆发期
3. AI 手机只是一个概念还是会迎来换机潮
4. 对于大模型厂商和 AI 芯片厂商的启发

视频内容

1. 5 分钟了解 WWDC AI 内容
2. Apple Intelligence 大模型架构
3. Apple 大模型细节分析
4. 行业趋势洞察与思考

2024.06.10

Apple WWDC 2024


A more personal Siri
Your mom's flight lands at 11:18 AM.
Private Cloud Compute
Clean Up in Photos
Summaries in Messages
Writing Tools
Reduce Interruptions
in Focus
Apple Intelligence
Image Playground
Genmoji
Priority messages in Mail
Inbox
Priority notifications
Image Wand
Audio recording
summaries
Describe a Memory
Natural language search


Apple Intelligence

APPLE 股价

- WWDC Day2，Apple 股价强势反弹超过7.2%，创 2022.11.10 以来最大涨幅，刷新2023.12.14 所创的收盘历史新高。

交易量最大 | 股票 | 在美国上市的证券 | 总部位于美国

昨日收盘价	$210.62
当日价格范围	$211.92 - $213.09
年度波幅	$164.08 - $220.20
市值	3.32万亿 USD
平均交易量	7,524.06万
市盈率	33.11
股息率	0.47%


smartisan


重新定义下一个
十年的个人电脑

重新定义了个人电脑，

重新定义了 Office 办公套件，

重新定义了搜索信息的方式，

重新定义了即时通讯工具…

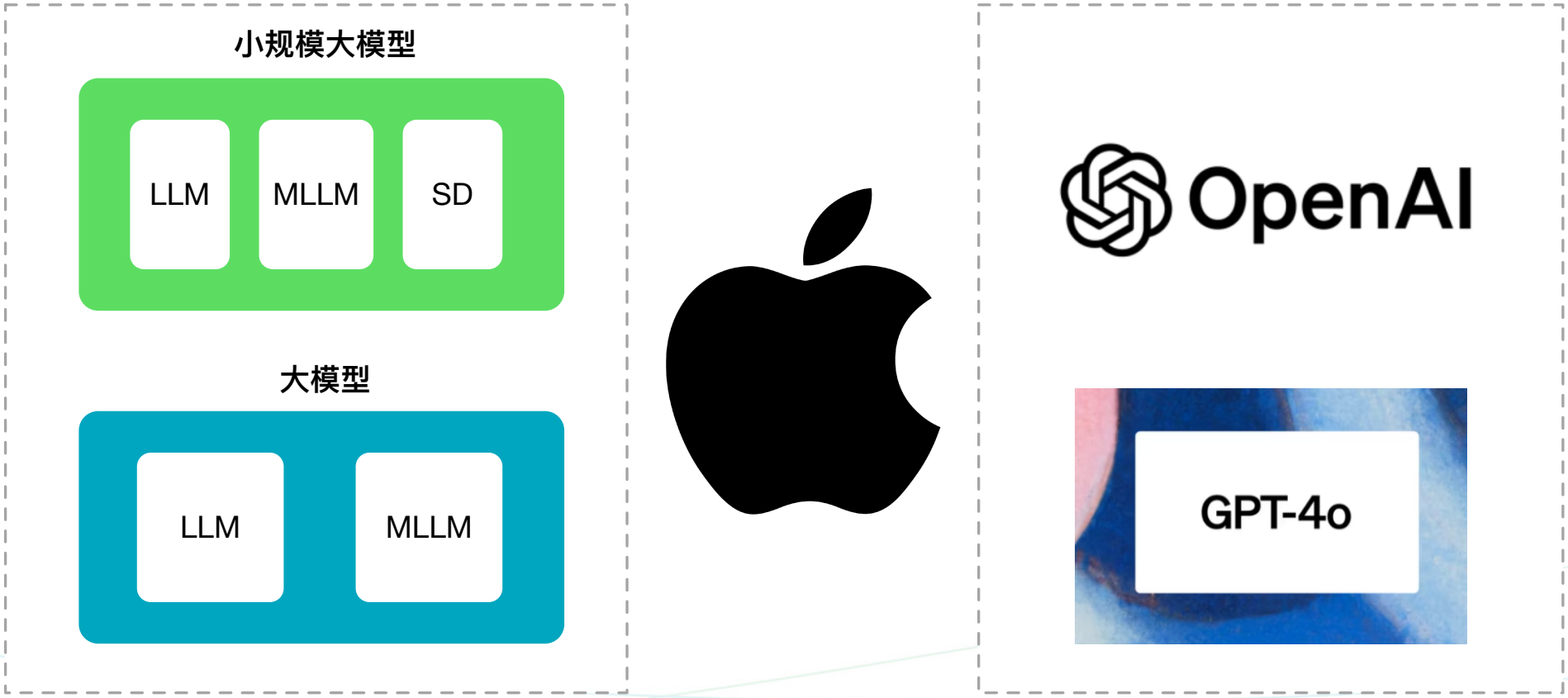
01. 5 分钟了解 WWDC AI

- 插入视频，通过视频进行讲解；只要音频，不需要录屏内容；

02. Apple Intelligence 架构

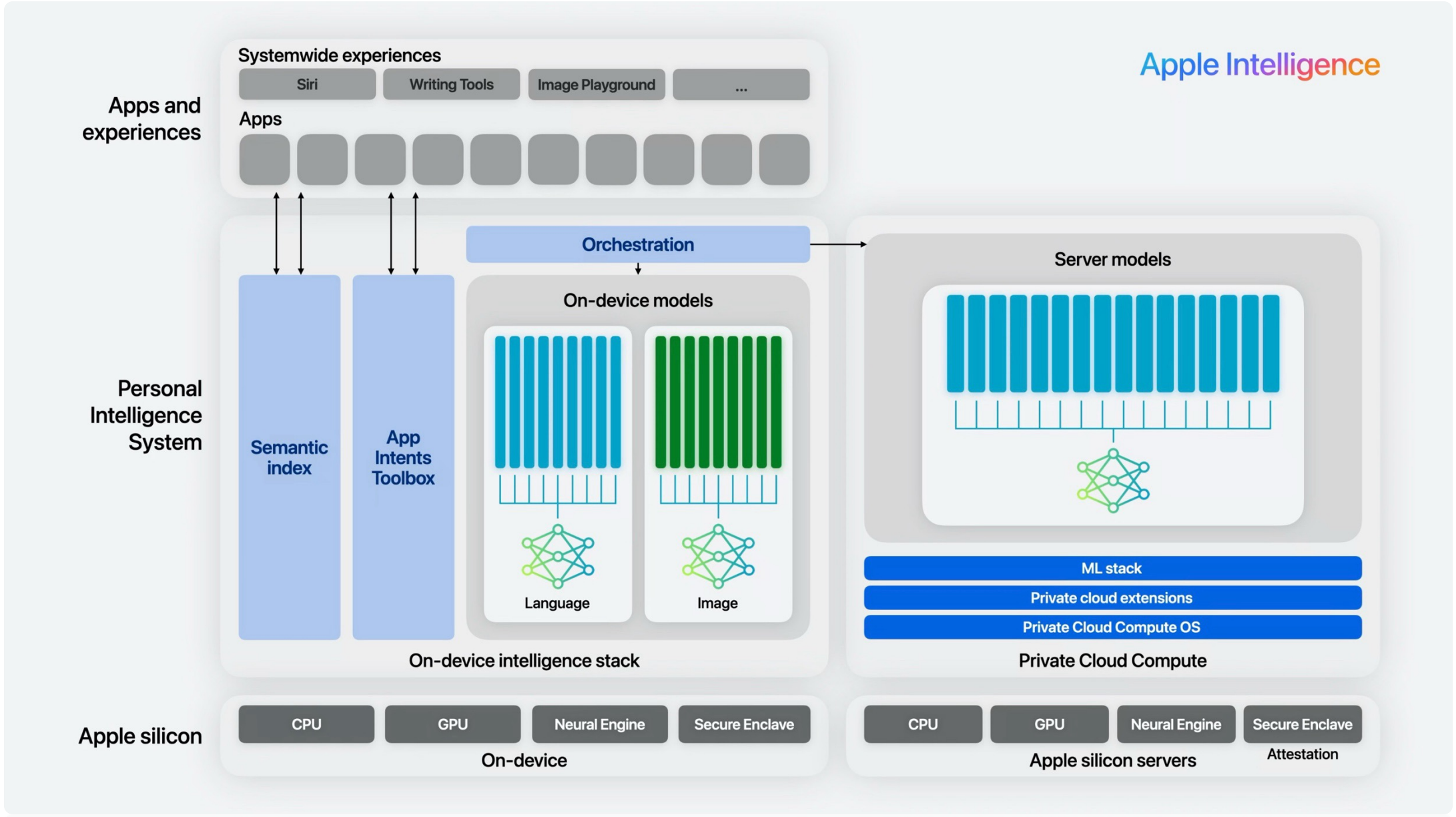
Apple Intelligence 大模型组成

- Apple Intelligence 分成三部分：1）自研端侧模型、2）自研云端模型，3）OpenAI GPT大模型。

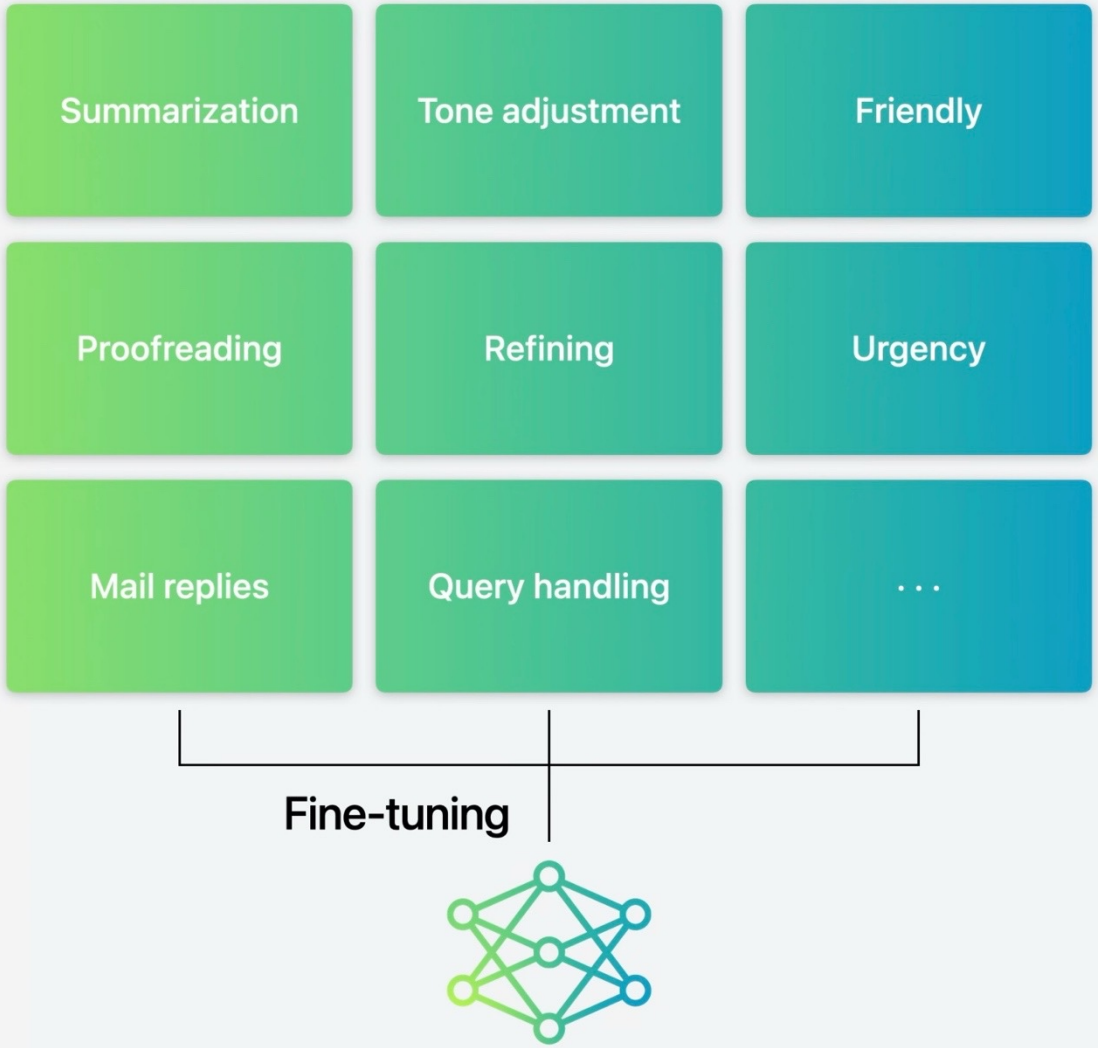


Apple Intelligence 大模型组成

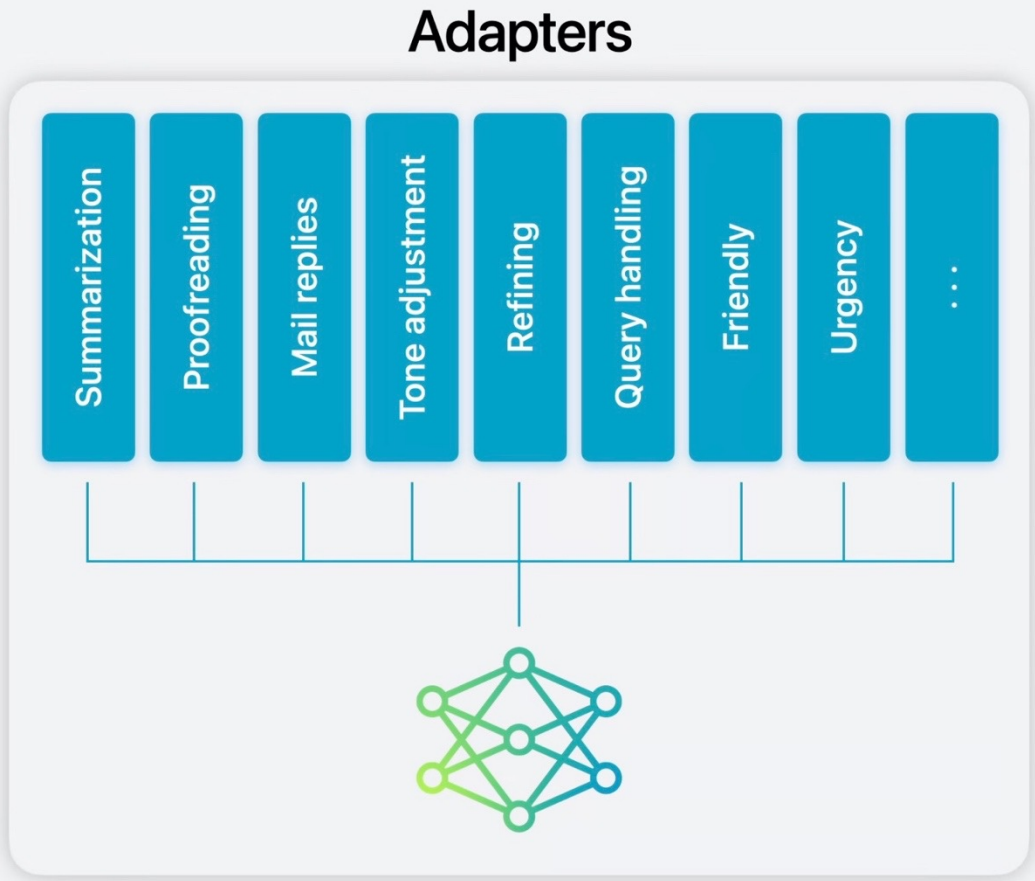
- 看官网相关介绍材料 O(∩_∩)O ~~


Apple Intelligence
Apps and experiences
Systemwide experiences
Siri
Writing Tools
Image Playground
...
Apps
Personal Intelligence System
Semantic index
App Intents Toolbox
Orchestration
On-device models
Language
Image
On-device intelligence stack
Server models
ML stack
Private cloud extensions
Private Cloud Compute OS
Private Cloud Compute
Apple silicon
CPU
GPU
Neural Engine
Secure Enclave
On-device
CPU
GPU
Neural Engine
Secure Enclave
Attestation
Apple silicon servers

LoRA 微调针对不同任务



Adapters 快速加载适配任务



技术总结

- **端测大模型组合：**
  1. LLM 语言大模型：MMI 3B 系列，配套Query、Summary 等不同风格 LoRA Adapters；
  2. LMM多模态大模型：MMI 3B 系列，配套 Emoji 等不同风格 LoRA Adapters；
  3. Stable Diffusion 模型：参数unknow，配套等不同风格 LoRA Adapters ；
- **云测大模型组合：**
  1. LLM + MLM 大模型：30B 在 Private Cloud 上进行微调，采用联邦学习的方式；
- **OpenAI 大模型：**
  1. 内置 OpenAI GPT4 处理 LLM；
  2. 提供 OpenAI GPT-4o APP 应用；

03. Apple
大模型细节

相关文章

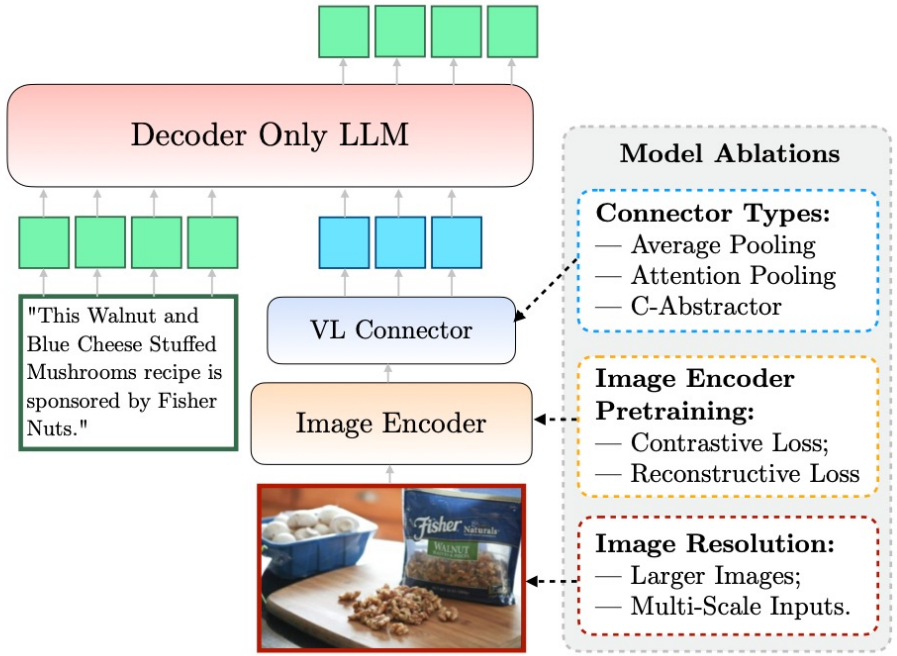
- MM1: Methods, Analysis & Insights from Multimodal LLM Pre-training -- **大模型**
- LLM in a flash:Efficient Large Language Model Inference with Limited Memory -- **端测部署**
- ReALM: Reference Resolution As Language Modeling -- **感知终端屏幕**

MM1 多模态大模型

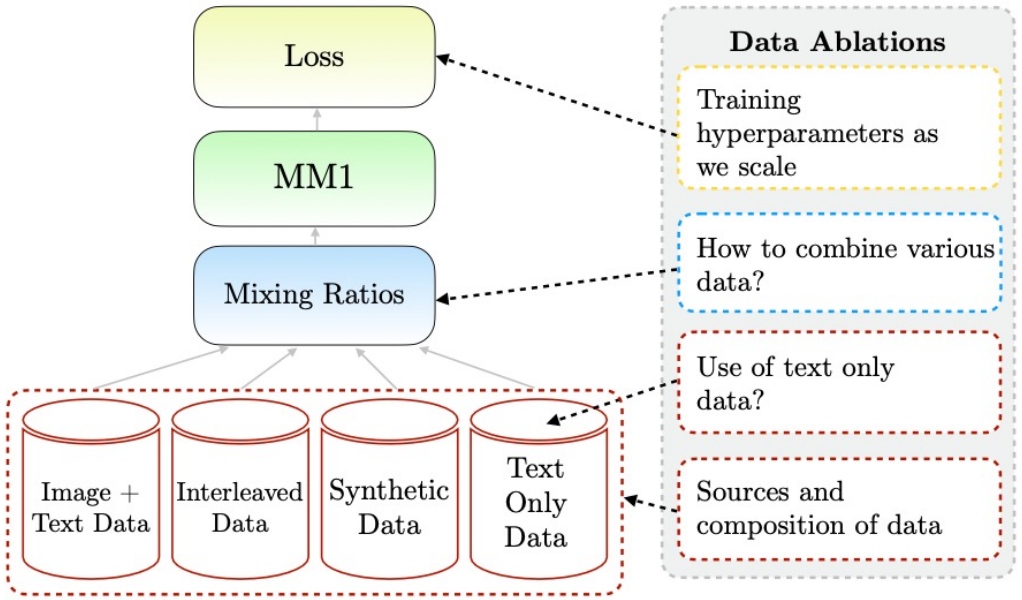
- Methods, Analysis & Insights from Multimodal LLM Pre-training
- 通过大量的实验，研究架构组件、数据选择对多模态大模型的重要性

MM1 多模态大模型

- **架构：**研究不同预训练图像 Encoder ， 探索LLM 与不同 Encoder 的连接方式
- **数据：**我不同类型数据及其相对混合权重比例，对模型效果的影响
- **训练过程：**探索如何训练MLLM，包括超参 + 模型组件 + 训练阶段



Left: Model ablations



Right: Data ablations

MM1 多模态大模型理解

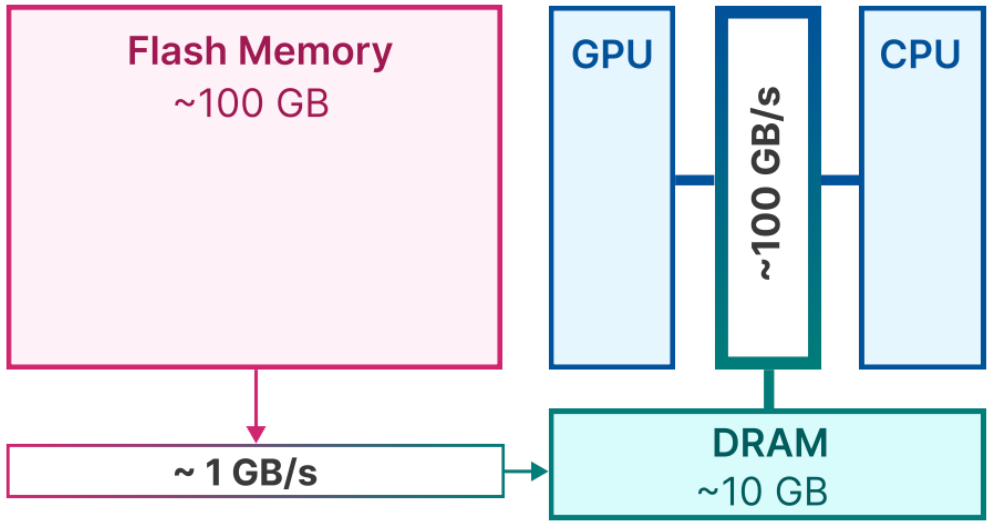
- MM1 系列有多 3B/7B/30B 不同组合的 MLLM，3B/7B/30B Chat/MoE Chat 不同组合的 LLM；
- So，Apple Intelligence 端测部署的小模型应该是以 3B 为主，云测以 30B 为主。

Model	Shot	Captioning			Visual Question Answering			
		COCO	NoCaps	TextCaps	VQAv2	TextVQA	VizWiz	OKVQA
MM1-3B Model Comparisons								
Flamingo-3B [3]	0†	73.0	–	–	49.2	30.1	28.9	41.2
	8	90.6	–	–	55.4	32.4	38.4	44.6
MM1-3B	0	73.5	55.6	63.3	46.2	29.4	15.6	26.1
	8	**114.6**	**104.7**	**88.8**	**63.6**	**44.6**	**46.4**	**48.4**
MM1-7B Model Comparisons								
IDEFICS-9B [58]	0†	46.0*	36.8	25.4	50.9	25.9	35.5	38.4
	8	97.0*	86.8	63.2	56.4	27.5	40.4	47.7
Flamingo-9B [3]	0†	79.4	–	–	51.8	31.8	28.8	44.7
	8	99.0	–	–	58.0	33.6	39.4	50.0
Emu2-14B [105]	0†	–	–	–	52.9	–	34.4	42.8
	8	–	–	–	59.0	–	43.9	–
MM1-7B	0	76.3	61.0	64.2	47.8	28.8	15.6	22.6
	8	**116.3**	**106.6**	**88.2**	**63.6**	**46.3**	**45.3**	**51.4**
MM1-30B Model Comparisons								
IDEFICS-80B [58]	0†	91.8*	65.0	56.8	60.0	30.9	36.0	45.2
	8	114.3*	105.7	77.6	64.8	35.7	46.1	55.1
	16	116.6*	107.0	81.4	65.4	36.3	48.3	56.8
Flamingo-80B [3]	0†	84.3	–	–	56.3	35.0	31.6	50.6
	8	108.8	–	–	65.6	37.3	44.8	57.5
	16	110.5	–	–	66.8	37.6	48.4	57.8
Emu2-37B [105]	0	–	–	–	33.3	26.2	40.4	26.7
	8	–	–	–	67.8	49.3	54.7	54.1
	16	–	–	–	68.8	50.3	57.0	57.1
MM1-30B	0	70.3	54.6	64.9	48.9	28.2	14.5	24.1
	8	123.1	111.6	92.9	70.9	49.4	49.9	58.3
	16	**125.3**	**116.0**	**97.6**	**71.9**	**50.6**	**57.9**	**59.3**

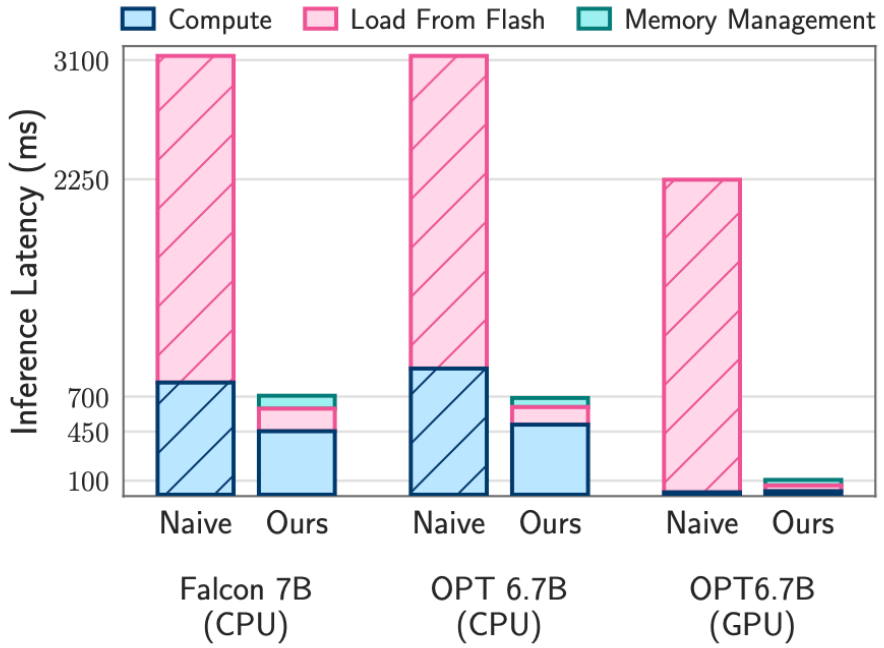
Model	VQA^{v2}	VQA^{T}	SQA^{I}	MMMU	MathV	MME^{P}	MME^{C}	MMB	SEED	POPE	$LLaVA^{W}$	MM-Vet
3B Model Comparison												
MobileVLM [20]	–	47.5	61.0	–/–	–	1288.9	–	59.6	–/–	84.9	–	–
LLaVA-Phi [135]	71.4	48.6	68.4	–/–	–	1335.1	–	59.8	–/–	85.0	–	28.9
Imp-v1 [99]	79.45	59.38	69.96	–/–	–	1434.0	–	66.49	–	88.02	–	33.1
TinyLLaVA [133]	79.9	59.1	69.1	–/–	–	1464.9	–	66.9	–/–	86.4	75.8	32.0
Bunny [42]	79.8	–	70.9	38.2/33.0	–	1488.8	289.3	68.6	62.5/–	86.8	–	–
Gemini Nano 2 [106]	67.5	65.9		32.6/–	30.6							
MM1-3B-Chat	82.0	71.9	69.4	33.9/33.7	32.0	1482.5	279.3	67.8	63.0/68.8	87.4	72.1	43.7
MM1-3B-MoE-Chat	82.5	72.9	76.1	38.6/35.7	32.6	1469.4	303.1	70.8	63.9/69.4	87.6	76.8	42.2
7B Model Comparison												
InstructBLIP-7B [24]	–	50.1	60.5	–/–	25.3	–	–	36.0	53.4/–	–	60.9	26.2
Qwen-VL-Chat-7B [5]	78.2	61.5	68.2	35.9/32.9	–	1487.5	360.7	60.6	58.2/65.4	–	–	–
LLaVA-1.5-7B [74]	78.5	58.2	66.8	–/–	–	1510.7	316.1	64.3	58.6/66.1	85.9	63.4	31.1
ShareGPT4V-7B [15]	80.6	60.4	68.4	–/–	–	1567.4	376.4	68.8	–/–	–	72.6	–
LVIS-Ins4V-7B [113]	79.6	58.7	68.3	–/–	–	1528.2	–	66.2	60.6/–	86.0	67.0	31.5
VILA-7B [71]	79.9	64.4	68.2	–/–	–	1531.3	–	68.9	61.1/–	85.5	69.7	34.9
SPHINX-Intern2 [36]	75.5	–	70.4	–/–	35.5	1260.4	294.6	57.9	68.8/–	86.9	57.6	36.5
LLaVA-NeXT-7B [75]	81.8	64.9	70.1	35.8/–	34.6	1519	332	67.4	–/70.2	86.53	81.6	43.9
MM1-7B-Chat	82.8	72.8	72.6	37.0/35.6	35.9	1529.3	328.9	72.3	64.0/69.9	86.6	81.5	42.1
MM1-7B-MoE-Chat	83.4	73.8	74.4	40.9/37.9	40.9	1597.4	394.6	72.7	65.5/70.9	87.8	84.7	45.2
30B Model Comparison												
Emu2-Chat-37B [105]	84.9	66.6	–	36.3/34.1	–	–	–	–	62.8/–	–	–	48.5
CogVLM-30B [114]	83.4	68.1	–	32.1/30.1	–	–	–	–	–	–	–	56.8
LLaVA-NeXT-34B [75]	83.7	69.5	81.8	51.1/44.7	46.5	1631	397	79.3	–/75.9	87.73	89.6	57.4
MM1-30B-Chat	83.7	73.5	81.0	44.7/40.3	39.4†	1637.6	431.4	75.1	65.9/72.1	87.6	89.3	48.7
Gemini Pro [106]	71.2	74.6	–	47.9/–	45.2	–	436.79	73.6	–/70.7	–	–	64.3
Gemini Ultra [106]	77.8	82.3	–	59.4/–	53.0	–	–	–	–	–	–	–
GPT4V [1]	77.2	78.0	–	56.8/55.7	49.9	–	517.14	75.8	67.3/69.1	–	–	67.6

LLM in a flash 端测部署

- Flash Memory 特点大存储、低带宽；DRAM 特点是小存储，高带宽。普通 LLM 推理直接模型参数加载到 DRAM，7B BF16 LLM 完全加载需14GB，DRAM 面临巨大挑战。
- Apple 解决方案将 LLM 放在 Flash Memory，每次推理时，仅将部分必要参数加载到 DRAM中。

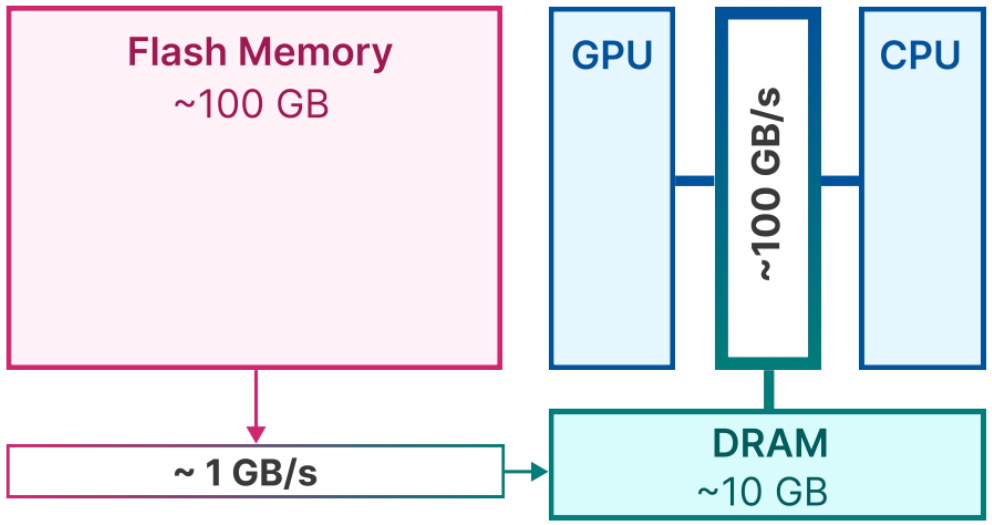


(a) Bandwidth in a unified memory architecture

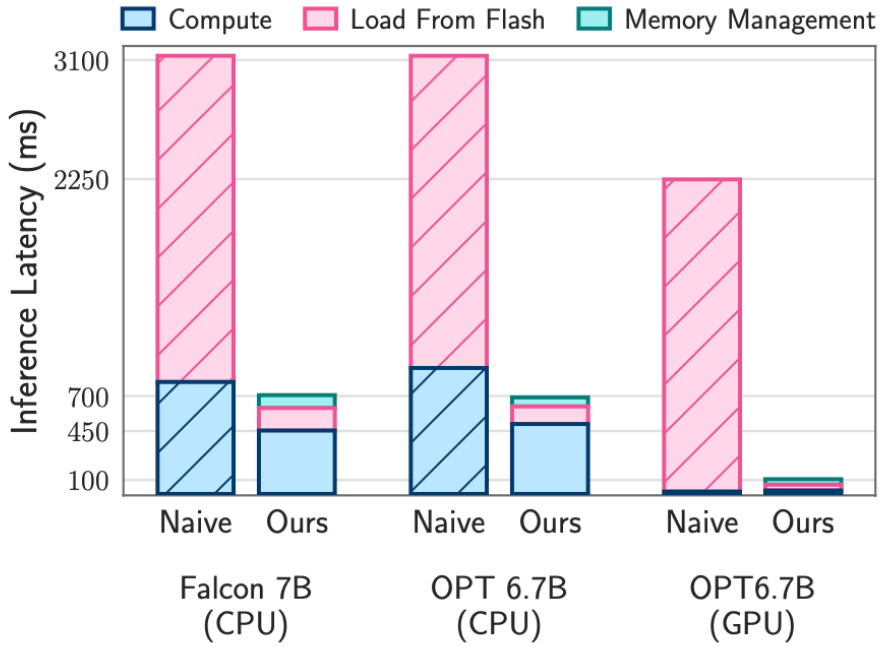


LLM in a flash 端测部署

- LLM in a flash: Efficient Large Language Model Inference with Limited Memory
- 1）减少数据传输量；2）提高传输吞吐量。
- So，让 LLM/MLLM大模型能够跑在苹果的终端设备上（手机），为 Apple Intelligence 奠基。



(a) Bandwidth in a unified memory architecture

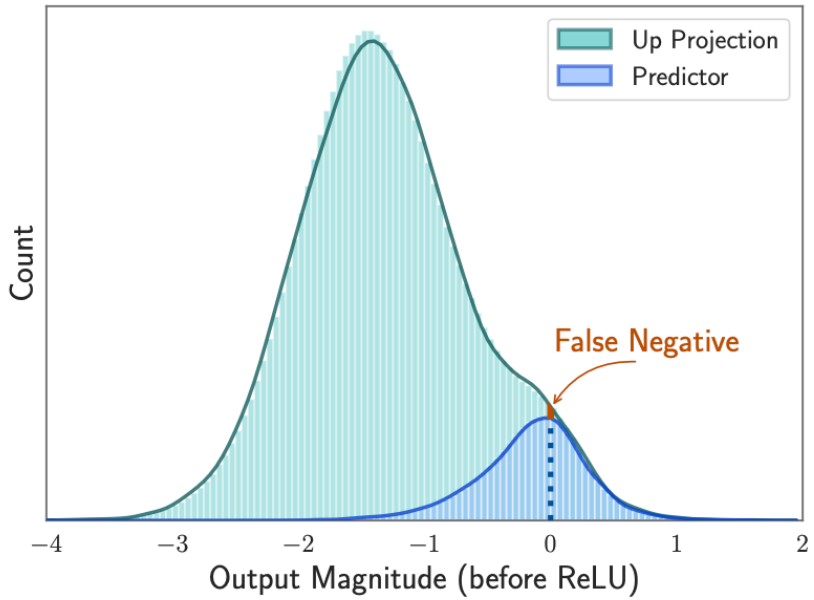


LLM in a flash：Selective Persistence Strategy

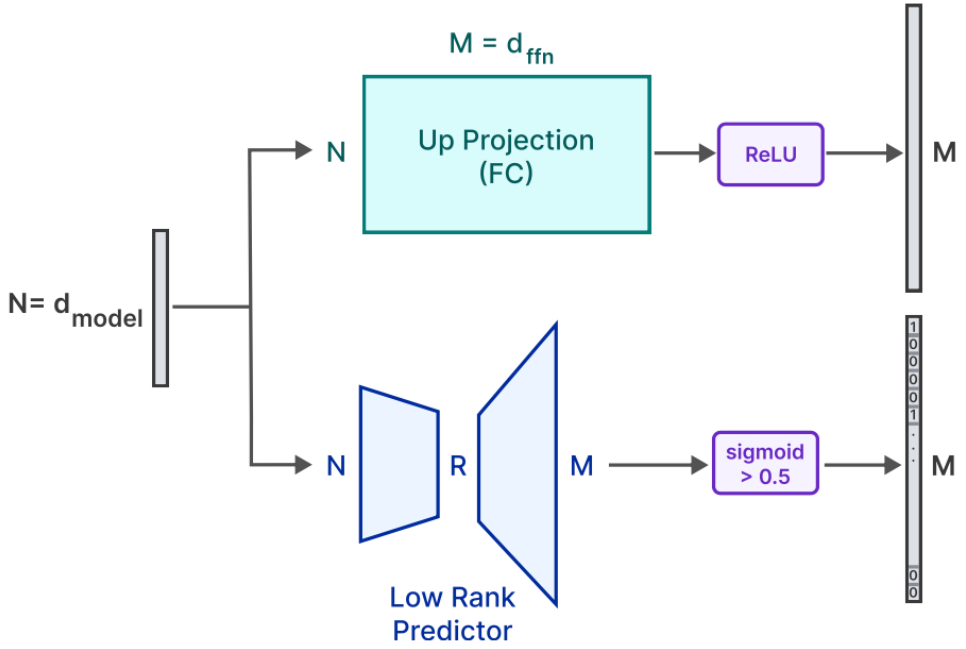
- **内存占用：**LLM 模型主要参数由 Attention 和 MLP 构成，其中 Attention 占1/3，MLP 占 2/3，此外还有 Embedding 层。
- **解决思路：**Attention 参数量相对较少，直接将 Attention + Embedding 参数直接加载到 DRAM，即有选择性地把部分参数常驻 DRAM。

LLM in a flash：Anticipating ReLU Sparsity

- MLP 输出只有不到 10% 值是激活状态（不为0），MLP 层神经元的激活与当前输入相关。
- 使用两层 MLP 低秩来预测会被激活的神经元，输出使用 sigmoid 输出 0/1 区分是否激活。
- 在能够准确预测前提下，每次在推理时动态加载预测为激活神经元对应的参数。



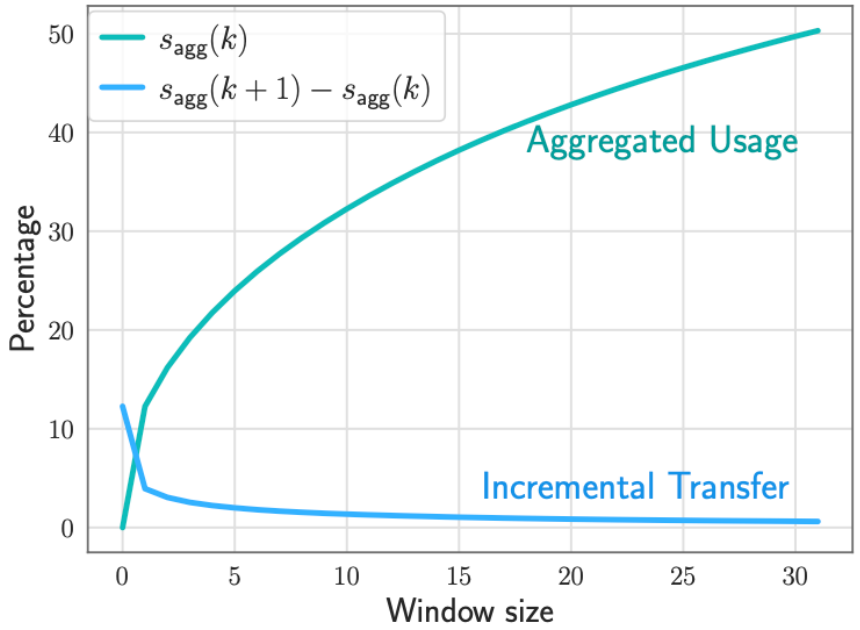
(a) predictor vs relu



(b) low rank predictor

LLM in a flash：Sliding Window

- 保留处理过去 k 个 Token 时激活神经元所对应参数在 DRAM 中，处理当前 Token 时：1）多余参数删除；2）缺少的参数进行加载。



(a) aggregated neuron use

(b) sliding window

LLM in a flash：Bundling Columns and Rows

- 为了提高稀疏参数的传输吞吐量，当第 i 个神经元被预测为激活时，需要同时读取 W_1 的第 i 列和 W_2^T 的第 i 行。
- 为了提高读取速度，提出列行捆绑数据读取方式，可以将 W_1 每一列列和 W_2^T 对应行拼接起来存储。
- 原本需要两次 I/O，现在只需要一次。总数据量没有变，但读取大块、连续数据比大量小块、非连续数据更高效，因此提升整体 IO 吞吐。

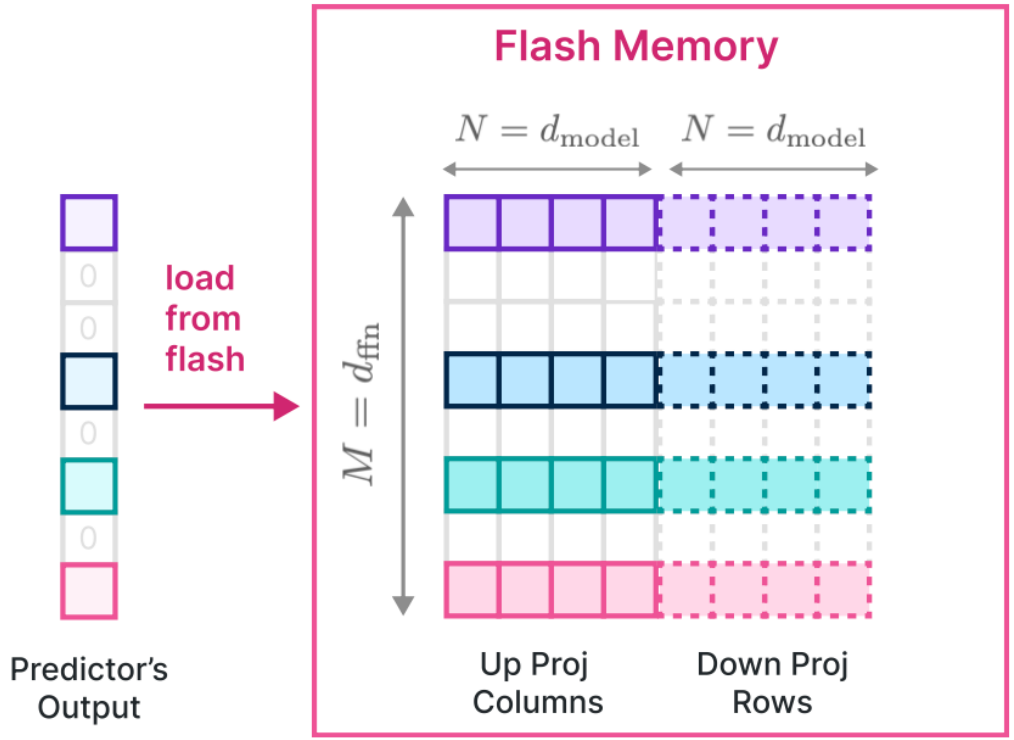


Figure 5: By bundling columns of up project and rows of down project in OPT 6.7B we will load 2x chunks instead of reading columns or rows separately.

LLM in a flash：Optimized Data Management in DRAM

- DRAM中 优化数据在内存的管理策略。涉及预分配必要内存，建立相应数据结构（数据结构包括指针、矩阵、偏置、num_used 和 last_k_active），以实现高效管理。

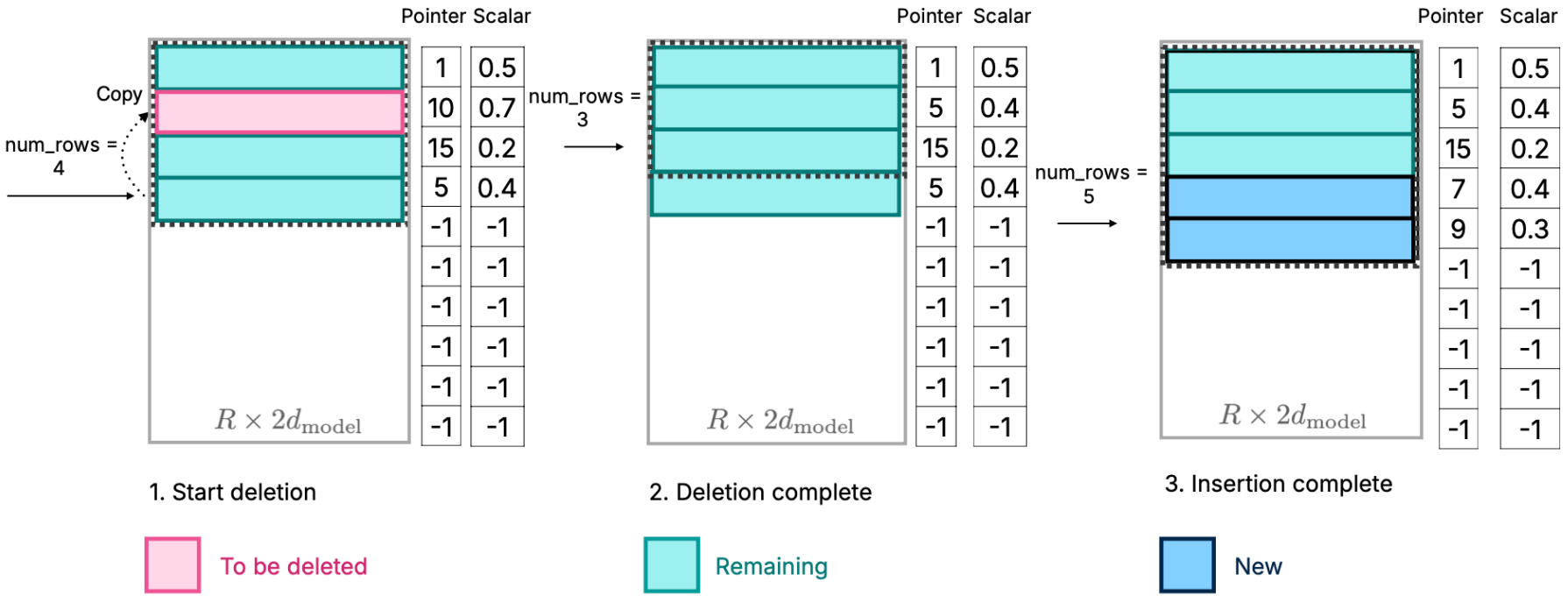
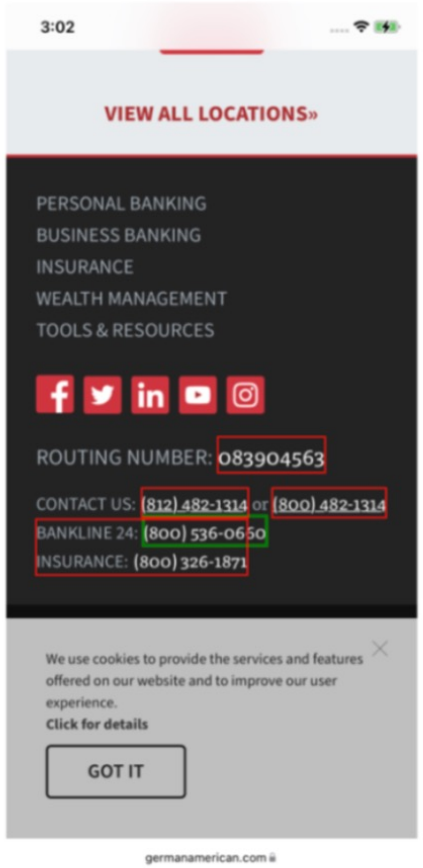
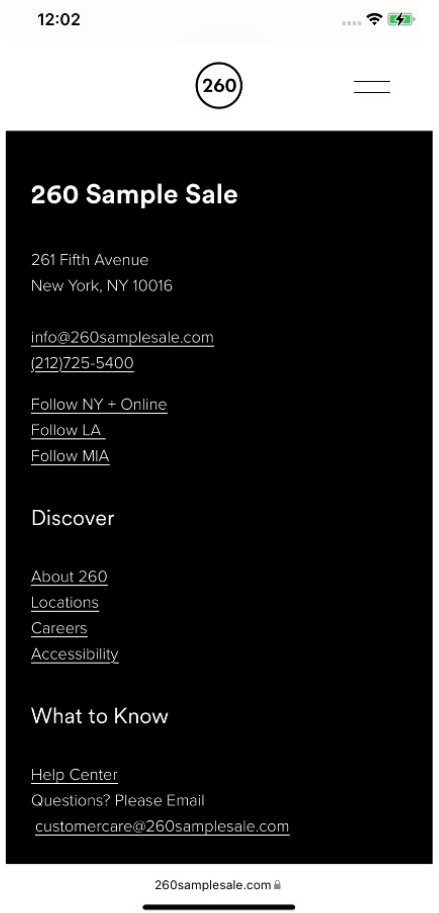


Figure 7: Memory management, first we copy last elements to deleting neurons to maintain a consecutive block of memory then the required ones are stack to the end, this prevents from copying whole data multiple times

ReALM 感知终端屏幕信息



(a) Screenshot example used in first annotation project



(b) Screenshot example used in second annotation project

Algorithm 1: Surrounding Object Clustering and Prompt Generation

Data: List of MDF turn objects
Result: Updated turn objects with surrounding object prompts

for *each MDF turn object* t **do**
  // Step 1: Get unique surrounding objects
  $surrounding\_objects \leftarrow$ Set of unique surrounding objects for t;
  // Step 2: Spatially cluster surrounding object bounding boxes
  $clusters \leftarrow DBScan(surrounding\_objects, rect\_dista$
  // Step 3: Predict the cluster for turn object
  $t\_cluster \leftarrow$ Predicted cluster for t;
  for *each surrounding object* s *in* $surrounding\_objects$ **do**
    if s *belongs to cluster* $t\_cluster$ **then**
      // Step 4: Process non-overlapping surrounding objects
      if *no string overlap between* t *and* s **then**
        Add s to the prompt under key 'surrounding_object';
  // Step 5: Provide global positioning information
  $t.distance\_from\_top \leftarrow$ Compute distance from the top for t;
  $t.distance\_from\_left \leftarrow$ Compute distance from the left for t;
return *prompt*;

Algorithm 2: Onscreen Parse Construction with Turn Object Injection

Data: List of turn objects
Result: Onscreen parse constructed with turn object injection

$onscreen\_parse \leftarrow$ Empty list of onscreen parse elements;
// Step 0: Get all text boxes present in the screen
for *each turn object* t, *index* i **do**
  // Step 1: Get unique surrounding objects
  $surrounding\_objects \leftarrow$ Set of surrounding objects for t;
  // Step 2: Insert turn objects into the set (as turn objects are also text)
  $surrounding\_objects \leftarrow surrounding\_objects \cup \{[[i.t]]\}$;
// Step 3: Sorting the centers of all surrounding objects
$sorted\_objects \leftarrow$ Sort objects in $surrounding\_objects$ by center (Top $\rightarrow$ Bottom, Left $\rightarrow$ Right);
// Step 4: Determine vertical levels
$margin \leftarrow$ Margin for considering objects at the same level;
$levels \leftarrow$ List of vertical levels;
for *each object* o *in* $sorted\_objects$ **do**
  $same\_level \leftarrow$ List of objects at the same level as o;
  for *each object other in* $sorted\_objects$ **do**
    if o *is not the same as other and* $|o.center\_top - other.center\_top| \leq margin$ **then**
      $same\_level \leftarrow same\_level \cup \{other\}$;
  $levels \leftarrow levels \cup \{same\_level\}$;
// Step 5: Construct onscreen parse
for *each level* l *in* $levels$ **do**
  $level\_parse \leftarrow$ Empty string;
  for *each object* obj *in* l **do**
    $level\_parse \leftarrow level\_parse + "\t" + obj$;
  $onscreen\_parse \leftarrow onscreen\_parse + "\n" + level\_parse$;
return $onscreen\_parse$;

04. 行业趋势洞察与思考

4.1

AI应用会促生换机潮吗

手机厂商围绕 AI 在做什么?

- 2023 年下半年开始，国内手机厂商 AI 布局声量已经渐起，All in AI 手机：
  - 2023.08，华为 HarmonyOS 4 全面接入盘古大模型；
  - 2023.10，小米自研AI大模型“MiLM-6B”接入澎湃OS；
  - 2023.11，vivo发布自研“蓝心大模型”；
  - 2023.11，OPPO 在 ColorOS 14引入“安第斯大模型”；
  - 2024.01，荣耀也发布自研70亿参数端侧AI大模型“魔法大模型”；
  - 2024.06，WWDC Apple 推出自研端云协同的大模型架构，接入 OpenAI；
  - 2024.06，华为 HDC 发布 Harmony Intelligence，官宣《智能 AI 终端白皮书》。

AI Phone 功能点洞察

- 现有 AI Phone 的 AI 功能主要围绕应用层，效果在于提升个别功能使用效率，好像并没有一个 AI 功能会成为刚需，大部分是 Make better，让智能无处不在。
- 实时通话翻译、会议录音总结等场景 AI 成为刚需，其他方向 AI 应用效果还没有足够惊艳。短期内 AI 对屏幕实体理解还不够，用户可以发起智能交互范畴受限。

但这会是用户想要新 iPhone 吗?

- 从文字、图片、语音，没有一个模态所在的场景，在智能终端中找不到的。
- ZOMI 相信 AI 无处不在很快就实现，但 AI 不是无所不能不是刚需，是 Make things Pretty。

但这会是用户想要新 iPhone 吗?

- Apple Intelligence 更多是让更多普通用户和传统 APP 开发者转到 AI、了解到 AI。
- 这一轮 AIGC 苹果玩不出花，APP 厂商和 AIGC 创业公司才是这一轮 AI 大模型革命缔造者。
- 所以如果是要强大到撬动用户换机诉求，赛道中还没有出现对应刚需应用（4G 短视频）。

4.2

大模型如何跟手机厂商结合?

苹果与 OpenAI 合作方式

- OpenAI 作为苹果手机三方预置大模型提供商，一个手机厂商不能预置太多三方大模型；
- 三方大模型安全问题归还给 OpenAI 和User，有种甩锅嫌疑，提出新概念所谓 PPC 安全；

手机厂商自研大模型

- 头部手机厂商 6+（华为、苹果、小米、Oppo、Vivo、荣耀、传音？）自研大模型：
  - 小米自研套皮开源大模型--MiLM-6B；
  - Vivo 自研套皮开源开源 -- 蓝心大模型；
  - OPPO 与三方合作套皮大模型 -- 安第斯大模型；
  - 荣耀自研套皮开源大模型 -- 魔法大模型 7B；
  - 华为背靠诺亚实验室 + 华为云提供盘古系列大模型；
  - 苹果自研 MM1杀入大模型修罗场 – MM1 3B；

与 APP 厂商提供的大模型如何分成?

- 终端厂商自研大模型与 APP 厂商自研的大模型如何区分?
- Apple 仅限于带货 AIGC? APP 厂商才是 AIGC 的缔造者/终结者?

2024.06.12 第三届美图影像节

与 OpenAI 合作的洞察

- 手机厂商不会/不能预置太多三方大模型，最坏情况未来终端厂商联合大模型厂商，强强联合；
- 剩下头部大模型公司真正能够让 AI 走向落地的机会越来越少，出口收窄在6+手机厂商；
- 中国工信部注册 117 个大模型&对应公司，最终会剩下几个？
- 安全出问题了肯定是大模型厂商背锅，就像自动驾驶一样。

4.3

对计算产业带来什么冲击?

回家吧~还是计算香

- **劝回：**当年离开昇腾去搞大模型创业公司的兄弟们赶紧回来吧~还是计算产业香，做上层应用变化太快，所以 ZOMI 才扎根 AI 系统。

计算产业的洞察

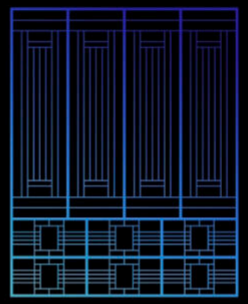
- OpenAI 目标是 AGI 能力，谷歌目标自家产品通过 AI 提升商业化增量，微软目标是企业 AI 化原有产品，英伟达目标是提供 AI 算法突破的底层算力，因此 NV 称为计算产业。
- 在新一轮 AIGC 浪潮中，Apple 也希望能够成为计算产业的第一梯队要去冲锋陷阵、验证技术能否落地。因此推出的 M1、M2、M3 到 M4 都自带 AI 内核。

120GB/s

Unified memory bandwidth

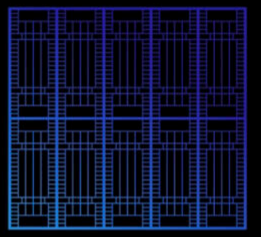
Dynamic Caching

Up to



10-core

CPU



10-core

GPU

Hardware-accelerated mesh shading

Up to

50%

faster CPU than M2

Up to

4x

faster GPU than M2

ProRes

AV1

Over

28 billion transistors

Second-generation

3 nm technology

Neural Engine with

38 trillion ops/sec

计算产业的洞察

- 终端厂商也参与到在推理场景的计算产业，高通、联发科 MTK、OPPO、VIVO、小米都想做外挂 AI Chip，AIGC 在计算的推理场景将会被瓜分得更加分散。

Apple 这次是跟 AI 黄昏恋 or 回到当初那个改变世界的少年?

Thank you

把AI系统带入每个开发者、每个家庭、每个组织，构建万物互联的智能世界

Bring AI System to every person, home and organization for a fully connected, intelligent world.

ZOMI

Course chenzomi12.github.io

GitHub github.com/chenzomi12/DeepLearningSystem

参考文献 Reference

1. https://machinelearning.apple.com/research/introducing-apple-foundation-models
2. https://machinelearning.apple.com/research/talaria
3. https://huggingface.co/posts/Jaward/185242785650522