Supplementary Information

Ubiquitous Memory Augmentation via Mobile Multimodal Embedding System

Dongqi Cai[1,2], **Shangguang Wang**[1*], **Chen Peng**[1], **Zeling Zhang**[1], **Zhenyan Lu**[1,3], **Tao Qi**[1], **Nicholas D. Lane**[2,4*], **Mengwei Xu**[1*]

[1]State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China
[2]Department of Computer Science and Technology, University of Cambridge, Cambridge, UK
[3]Pengcheng Laboratory, Shenzhen, China
[4]Flower Labs, London, UK

*Correspondence: sgwang@bupt.edu.cn, ndl32@cam.ac.uk, mwx@bupt.edu.cn

Background, Applications and Related Works

Unified multimodal embedding Embedding was initially proposed to vectorize text data for understanding similarities between different texts [1]. Large language models use embedding layers to generate text embeddings [2, 3]. Similarly, vision, audio, and sensor data can also be transformed into vectorized embeddings [4–6]. However, embedding methods focused on a single modality cannot access information across different modalities due to the gap between their embedding spaces.

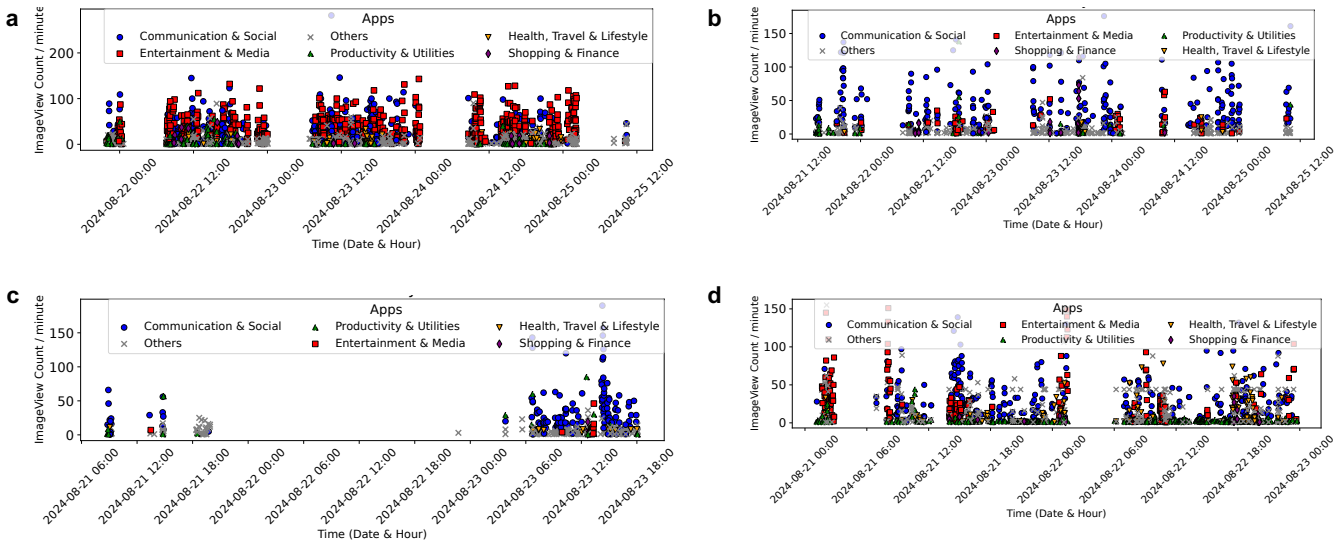
To bridge this gap, multimodal embedding models (MEMs) have been developed to unify different modalities into a single embedding space, enhancing the model's ability to understand and bind multimodal inputs. CLIP [7] aligns text and vision by jointly training on image-text pairs, using contrastive learning to map both modalities into a shared space while maintaining their distinction through a dual-tower architecture. ImageBind [8] extends this to align six modalities, including text, vision, audio, depth, thermal, and IMU readings. Each modality is processed by a separate encoder, and the embeddings are fused in a multimodal head to generate a unified embedding. ImageBind demonstrates strong zero-shot classification and retrieval performance across these modalities, matching or outperforming single-modality models. This is achieved through training on large-scale multimodal data.

According to Zhang et al. [9], over 80% (35 out of 43) of multimodal foundation models utilize ImageBind or its subset, CLIP, as their modality encoder. This widespread adoption highlights the efficiency and effectiveness of ImageBind in managing multimodal data. Further optimizations have enhanced the fusion of vision and text [10, 11], as well as the fusion of dynamic sensing [12]. However, these methods do not address new modalities in open-set recognition, as handled by ImageBind. In this manuscript, we enable efficient multimodal embedding in a unified space with usable search accuracy on mobile devices.

Multimodal mobile applications MEMs optimize alignment between high-quality representations across modalities. As such, multimodal information can be composed to enable a rich variety of mobile context-aware applications. For example, MEMs could embed visual, audio, text and sensor data experienced on a mobile device into a personalized memory palace [13, 14]. Whenever users want to recall a specific moment or items, they can query the memory palace with a multimodal query, and the system will retrieve the most relevant items. MEMs can also facilitate mobile agents to iteract with users in a more human-like manner [15–17]. Recently, Microsoft launches a project called Recall that makes a note of everything ever displayed on personal computer for AI-empowered retrospective search [18].

On-device multimodal embedding Data for embedding is continuously sourced from end users and is often private and sensitive. Evidence suggests that cloud service providers may be curious about uploaded data to improve their services [19], and database leaks and breaches pose significant threats [20]. Conducting embedding locally prevents the need to upload daily viewed, sensed, or heard data to the cloud, offering strong privacy protection. From the cloud perspective, a single user views over 6,000 images per day, according to our user study, requiring approximately 1065.6KJ of energy and 0.8 GPU hours. For 1 billion daily active users, cloud providers would need 1.1 TWh of energy and 0.8M GPU hours daily, costing over $100 million per day. On-device multimodal embedding shifts this cost to end users, making the service more practical to deploy.

Early-exiting While early exiting has been a known technique in both traditional CNNs and recent language models [21–24], it is rarely integrated with MEMs for mobile devices. BranchyNet [23] showed that features learned in early layers are often sufficient for classification, with only a few difficult samples requiring deeper layers. DynExits [25] introduced learnable early-exit branch weights to avoid the pitfalls of manually defined loss weights, while DVABatch [26] dynamically adjusted batch sizes to allow independent query



Supplementary Figure 1. Visualization for part of collected traces of viewed images of daily mobile applications. (a) Male, 23 years old; (b) Male, 25 years old; (c) Male, 55 years old; He quitted a while in the middle of the study. (d) Female, 24 years old.

exits, though with limited improvement. Layer Skip [27] and DeeCap [21] utilized early exits for tasks like text decoding and image captioning. Gromov et al. [24] demonstrated that removing deeper layers often does not degrade performance due to the similarity between adjacent deep layers. In this work, we propose an early-exiting system for on-device MEMs, providing a lightweight and efficient solution for mobile devices.

Predictive early exit Predictive Exit [22] designed a low-cost prediction engine for CNNs, using zero padding, filter generation, and one-dimensional convolution to predict exit points in computer vision tasks. Dong et al. [28] introduced an exit predictor that uses depthwise separable convolutions to generate scores for deciding whether to skip certain exits, reverting to confidence-based decisions when necessary. However, these methods are complex and not suited for attention-based transformers, where matrix multiplication dominates. In summary, while effective for CNNs, these approaches do not scale well to transformer-based models due to their convolution-specific designs. Hamed et al. [11] integrated early exits at the end of encoders to balance predictive performance and computational efficiency, but they did not address exit timing or hardware compatibility. Our system introduces a hardware-friendly, lightweight predictor for efficient early exits in transformers, tailored for mobile devices, ensuring both performance and accuracy.

User Study on Viewed Images

We collected statistics of viewed images of daily mobile applications from 8 volunteers, ranging from 20 to 52 years old. Uiautomator was executed in the background to dump daily android UI page (stored in XML file format) for final analysis. A sample of XML file is shown below:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<hierarchy rotation="0">
    <node>
        ...
        <node
            index="1"
            text="Brita Kettle..."
            resource-id="com.xingin.xhs:id/cy7"
            class="android.widget.ImageView"
            package="com.xingin.xhs"
            content-desc="product photo"
            checkable="false"
            checked="false"
            clickable="true"
            enabled="true"
            focusable="true"
            focused="false"
            scrollable="false"
            long-clickable="false"
            password="false"
            selected="false"
            bounds="[0,114][78,213]"
        />
        ...
    </node>
</hierarchy>
```

The top 10 applications that generated the most viewed images are shown to illustrate the user trace, as sampled in Supplementary Figure 1. The 'ImageView' elements are detected to monitor whether there is the new figures. Relevant images and their description are saved in storage for future retrieval. Each imageview xml element group is hashed to only include newly appeared images.

Multimodal Retrieval Demo

We utilize the original ImageBind-huge [8], downloaded from the official website, to demonstrate its multimodal retrieval

Supplementary Figure 2. Demo of cross-modal retrieval using MEMs. Given an audio clip of fireworks, the model retrieves the most semantically relevant images and text descriptions from user data, with high confidence scores.

capability. By aligning multimodal embeddings into a unified space, ImageBind can effectively retrieve semantically relevant content across different modalities using human-friendly input formats. As shown in Supplementary Figure 2, the sound of fireworks retrieves images of fireworks from the albums, along with their corresponding textual notes, with high confidence.

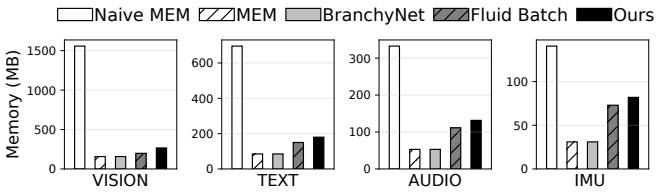
Memory Issue and Loading Latency

Storing the complete multimodal model weights for each modality is memory-intensive. For example, the visual encoder alone occupies more than 1GB of memory for its weights, not to mention the intermediate activations. This can cause the application to be killed by the OS due to memory pressure [29].

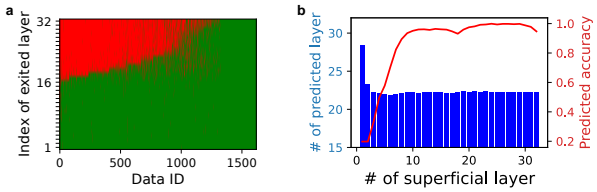
One common solution is to load the model layer by layer and sequentially remove the weights from the memory after finishing relevant operations. It can save memory by up to 10× for executing MEM inference. However, the loading latency for each transformer block is huge. For example, it takes around 0.43s to load one block on a commodity mobile phone, which is more than 2× larger than the encoding time of each block (around 0.19s for one image). This leads to a significant increase in the encoding time.

Reminisce is designed to address these issues by predicting the optimal number of layers to load and embedding for each sample, reducing memory usage and optimizing the encoding pipeline to hide the loading latency. We first test memory footprint of naive MEM in a memory-unlimited server environment. Compared to naive MEM, Reminisce can reduce memory usage by up to 5.8×. However, due to the extra memory overhead of batching parallelism, Reminisce has a slightly higher peak memory footprint compared to the naive layer-wise baseline. But it is still within a practical range, e.g., 82M for embedding IMU information, which is below the average Android application memory consumption of 100M as reported in 2020 [30]. After 5 years, the mobile RAM capacity has increased significantly, with up to 24GB available on high-end devices [31]. Less than 300MB of peaky memory usage is affordable for most modern mobile devices.

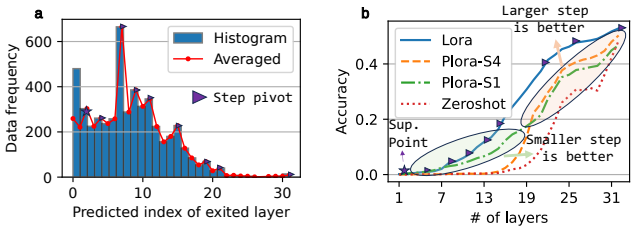
.



Supplementary Figure 3. Comparison of memory footprint across different modalities. Our system maintains a peaky memory usage of less than 300MB throughout the embedding process. Device: ORIN (INT8).



Supplementary Figure 4. Rationale behind pre-exit predictor. (a) Each dataset has a different optimal exit point. (b) Intermediate embeddings can serve as a cue for predicting the appropriate exit for each sample.

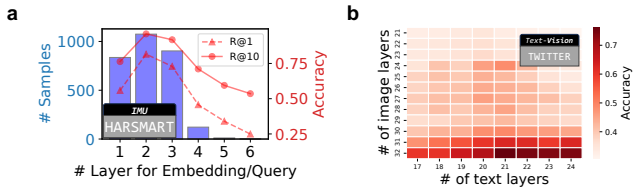


Supplementary Figure 5. P-LoRA step decision. (a) The pivot point of the pre-exit predictor guides the selection of the tuning step. (b) Relationship between progressive tuning steps and exit locations.

Micro Experiments for Design Rationales

Effectiveness of pre-exit predictor As shown in Supplementary Figure 4a, prediction accuracy improves with the increase of superficial embedding layers. As indicated by Supplementary Figure 4b, most samples require the complexity of more than 7 layers. With $N = 7$, the predicted accuracy is 85%, the average predicted layer is 15.5, and the average actual layer is 16.5. An interesting finding is that as the intermediate embeddings are fed layer by layer, the deeper the layers, the more accurately the predictor model can determine the exit value. This improvement occurs because deeper layer embeddings are more discriminative and better suited for predicting the final embedding.

P-LoRA step decision As shown in Supplementary 5, the optimal healing step varies across exit layers. In general, the



Supplementary Figure 6. Retrieval accuracy across embeddings generated by different MEM layer exits. (a) Single modality; (b) Cross modality.

larger the n, the greater the per-step healing capacity, due to the increased number of tunable parameters. However, if step 4 is applied to all exits, exits 2 and 3 will miss opportunities for healing. This is acceptable for the top layers, as they already have a strong feature representation from earlier healing. Larger steps benefit later layers by improving convergence performance. For smaller exits, earlier features are still weak and require healing at each exit.

Unbalanced retrieval performance As shown in Supplementary Figure 6b, query embeddings will prefer those data embedding counterparts generated by similar granularities, leading to an interesting observation: query with full-capacity embeddings not always retrieval the best results. For instance, for single-modality retrieval on the HARSMART dataset, using the full-capacity MEM to retrieve filtered embeddings results in a top-1 accuracy of only 24.9%, 56.6% lower than using a 2-layer query embedding, since over 99% of samples exit before 3 layers during local embedding. The same phenomenon occurs in the cross-modal TWITTER dataset.

Design Details and Cost Analysis

Supplementary Algorithm 1: *Our Pre-exit Predictor*

input : Superficial Embedding Layer N;
Predict model ϕ_S;
Burst-in Streaming Input, $\mathbb{X}$.
output : Embedding, $\mathbb{E}$.

Function *Data-aware_Coarse-grained_Embedding*(N, ϕ_S, $\mathbb{X}$):
  Embedding ← Batched_Layerwise_Encoding(0, N, $\mathbb{X}$);
  Predicted Exit $e \leftarrow \phi_S(\mathbb{E})$;
  Group $\mathbb{X}$ into $\mathbb{X}^e$ with the same exit seperately;
  forall $\mathbb{X}^e$ **do**
    Embedding ← Batched_Layerwise_Encoding(N, e, $\mathbb{X}^e$);
  Store Embedding $\mathbb{E}$ in the disk.
Function *Batched_Layerwise_Encoding*(i, j, $\mathbb{X}$):
  $\mathbb{X}_B \leftarrow$ Batching $\mathbb{X}$;
  forall $\mathbb{X}_B$ **do**
    while i<j **do**
      Encode $\mathbb{X}_b^i$; load layer i+1 concurrently;
  Embedding ← *Postprocessor*(Intermediate results);
  return Embedding.

Device	Specification	Model Precision
Jetson Orin [24]	1024-core NVIDIA Ampere GPU. Default operating mode: MAXQ.	INT8
Jetson TX2 [33]	256-core NVIDIA Pascal GPU. Default operating mode: MAXN.	INT8
Raspberry Pi 4B [34]	Broadcom BCM2711B0 quad-core A72 64-bit @ 1.5GHz CPU.	INT8
Redmi Turbo 3 [35]	Qualcomm Snapdragon 8 Gen 3 CPU. Default operating mode: Balance.	INT4

Supplementary Table 1. Experimental devices used in experiments.

ID	Name	CPU Online	Max CPU Freq	Max GPU Freq
0	MAXN	0–7 all	729.6 MHz–9.22 GHz	0–9.22 GHz
1	MAXQ	0–3	729.6–1190 MHz	0–612 MHz
2	MAXP_FREQ	0–3	729.6–1420 MHz	0–612 MHz
3	MAXP_CORE	0–7 all	729.6–1497.6 MHz	0–408 MHz

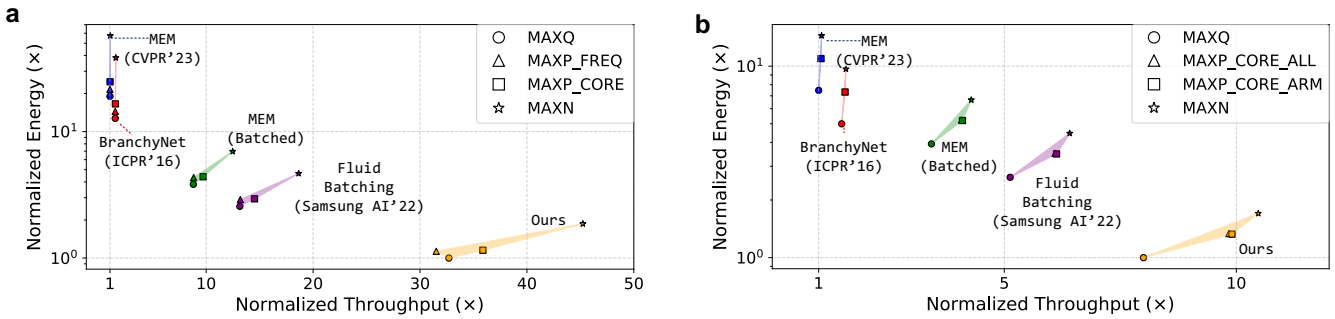
Supplementary Table 2. Jetson orin power modes overview.

Pre-exit predictor in detail We summarize the use of the pre-exit predictor in Supplementary Algorithm 1. First, we load Layer$_i$ and encode all input data as a batch, while Layer$_{i+1}$ is loaded concurrently to minimize loading time. This process iterates until all N layers are loaded. Next, we feed the intermediate embeddings (i.e., superficial embeddings) to the predictor model. Data are then batched according to the predicted exit values. These steps are repeated for each batch until all data reach their predicted exits.

Pre-exit predictor cost Training the predictor is efficient, requiring only tens of iterations on hundreds of samples, taking just a few minutes on a single GPU. The trained predictor is lightweight, with a memory footprint of around 1MB. The main concern is the cost of computing the superficial embedding. Fortunately, this embedding can be reused for subsequent coarse-grained embeddings.

Progressive LoRA training details The healing P-LoRA is designed to be parameter-efficient and highly transferable. Application developers can customize the personalized healing adapter during the testing phase. During deployment, healing occurs iteratively, and embedding granularity can be updated in real time to better fit the data and synchronize with their representations. In this work, for simplicity, the embedding granularity predictor was trained on zero-shot embeddings. The training objective is the fine-grained embedding, not the query embedding. We leave the output layer untuned to mitigate the dynamic embedding mismatch issue. Even without the healing P-LoRA, we demonstrate that Reminisce can still achieve usable retrieval performance.

Cache cost analysis We quantize the intermediate activations to INT4 and de-quantize them during reuse, which



Supplementary Figure 7. System performance under different operating modes. Dataset: COCO. (a) Performance on Jetson Orin (INT8); (b) Performance on Jetson TX2 (INT8).

ID	Name	CPU Online	Max CPU Freq	Max GPU Freq
0	MAXN	1–5 all	2.14 GHz	2.14 GHz
1	MAXQ	3–5 on	1200 MHz	0–850 MHz
2	MAXP_CORE_ALL	1–5 all	1400 MHz	0–1120 MHz
3	MAXP_CORE_ARM	3–5 on	2000 MHz	0–1120 MHz

Supplementary Table 3. Jetson TX2 power modes overview.

takes significantly less time than re-computation (around 10 ms per embedding). During pre-exiting prediction, the activations remains in RAM. Once coarse-grained embedding begins, these cached activations replace the intermediate variables typically stored in RAM during embedding, so no additional peak memory footprint is required. After the process ends, the activations are released sequentially. For cache reuse in the fine-grained embedding procedure, the activations of coarse-grained embeddings are quantized and temporarily stored in storage, which is less constrained than RAM, until the query occurs. This strategy is well-studied by mobile memory management literature [30], and is suitable for modern mobile devices with large storage. The loading time is approximately 1 ms for 10 activations. Once an image is queried, it is updated to the fine-grained embedding, and its storage can be freed.

Implementation Details

Reminisce is built on ImageBind-LoRA [32], an open-source multimodal embedding model fine-tuning framework. For matching, we use matrix multiplication, as it is not the primary bottleneck in query cost. Further vector database optimizations, such as FAISS [33], are orthogonal to Reminisce. LoRA tuning and embedding accuracy evaluations are emulated on a GPU server to enable faster iterations and energy savings. Embedding inference latency, power consumption, and memory usage are directly measured by running Reminisce on a mobile device using the open-source on-device multimodal inference engine mllm [34]. The devices used in the evaluation are summarized in Supplementary Table 1. Detailed hardware specifications of different modes for the two Jetson boards are summarized in Supplementary Table 2 and Supplementary Table 3. We adopt optimum-quanto [35] to quantize model weights to "qint8" on Jetson development boards.

Additional Throughput Analysis

Supplementary Table 4 summarizes the embedding throughput comparison, serving as a numerical supplementary of end-to-end performance. Reminisce maintains high throughput in a layer-wise setting, making it a more practical solution for resource-constrained devices. For instance, on the 8GEN3 mobile, Reminisce can process data up to 5.4× faster than the naive MEM without loading layers sequentially, while reducing memory footprint by up to 5.8×. Although Reminisce primarily targets layer-wise scenarios, we also evaluated throughput performance when loading all encoders simultaneously. Reminisce is still competitive in this setting, achieving 2.6× throughput improvement on average due to the enhanced early-exit mechanism.

Interestingly, we find that healing the exited larger MEMs is more effective than using a smaller-sized foundation model. For example, using CLIP-b/16 with 85.6M parameters results in embeddings that are 2.7× faster than ImageBind but significantly reduces the ability to embed different modalities concisely, leading to up to 39.8% accuracy loss. Though with our system design, the accuracy loss is mitigated to 3.1% with a 2.7× throughput improvement, its performance is still inferior to the healed ImageBind. Fortunately, Reminisce enables narrows the cost gap between the two models, while achieving much higher retrieval performance.

Furthermore, as shown in Supplementary Figure 7, more aggressive operating modes generally lead to higher system throughput. For example, on ORIN, using MAXN mode provides a 1.4× throughput improvement compared to MAXQ mode, albeit at a 6.3× increase in power consumption. Therefore, selecting an appropriate operating mode based on the specific device characteristics and requirements is crucial for optimizing the performance-energy tradeoff. Fortunately,

Dataset	COCO					FLICKR					CLOTHO					HARSMART				
Throughput (Contents / s)	Relative Accuracy	ORIN (INT8)	TX2 (INT8)	RPI4B (INT8)	8GEN3 (INT4)	Relative Accuracy	ORIN (INT8)	TX2 (INT8)	RPI4B (INT8)	8GEN3 (INT4)	Relative Accuracy	ORIN (INT8)	TX2 (INT8)	RPI4B (INT8)	8GEN3 (INT4)	Relative Accuracy	ORIN (INT8)	TX2 (INT8)	RPI4B (INT8)	8GEN3 (INT4)
MEM	100%	1.98	0.34	0.05	0.05	100%	1.98	0.34	0.05	0.05	100%	5.34	3.15	0.30	0.27	100%	34.15	4.30	1.03	0.81
MEM (Batched)		17.45	1.36	0.07	0.10		17.45	1.36	0.07	0.10		58.41	5.28	0.38	0.32		121.41	7.86	1.28	0.96
BranchyNet	71%	2.96	0.50	0.08	0.07	93%	2.27	0.38	0.06	0.06	81%	31.12	7.96	0.76	0.69	57%	108.98	13.72	3.29	2.58
Fluid Batch		26.08	2.03	0.11	0.16		20.01	1.55	0.08	0.12		100.71	13.34	0.96	0.80		387.49	25.07	4.08	3.05
Ours	95%	62.31	3.31	0.15	0.31	95%	44.33	2.32	0.10	0.22	98%	240.91	25.64	0.99	0.84	95%	787.46	69.51	6.88	5.15
MFM (w/o layerwise)	100%	36.49	1.70	0.07	0.16	100%	36.49	1.70	0.07	0.16	100%	125.52	5.85	0.40	0.34	100%	191.24	8.91	1.33	1.02
BranchyNet (w/o layerwise)	71%	54.53	2.54	0.11	0.25	93%	41.83	1.95	0.09	0.19	81%	317.11	14.77	1.00	0.85	57%	610.33	28.43	4.23	3.25
Ours (w/o layerwise)	95%	72.16	3.36	0.15	0.33	95%	50.44	2.35	0.10	0.23	98%	317.11	14.77	1.00	0.85	95%	1024.49	47.72	7.10	5.45

Supplementary Table 4. Throughput vs. relative retrieval accuracy.

Reminisce consistently demonstrates superior efficiency across different execution modes, enabling a more flexible trade-off between throughput and energy consumption.

Supplementary References

[1] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017.

[2] 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.

[3] 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.

[4] Alexey DOSOVITSKIY. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[5] 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.

[6] Danfeng Hong, Bing Zhang, Xuyang Li, Yuxuan Li, Chenyu Li, Jing Yao, Naoto Yokoya, Hao Li, Pedram Ghamisi, Xiuping Jia, et al. Spectralgpt: Spectral remote sensing foundation model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[7] 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.

[8] 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.

[9] Duzhen Zhang, Yahan Yu, Chenxing Li, Jiahua Dong, Dan Su, Chenhui Chu, and Dong Yu. Mm-llms: Recent advances in multimodal large language models. *arXiv preprint arXiv:2401.13601*, 2024.

[10] Shengkun Tang, Yaqing Wang, Zhenglun Kong, Tianchi Zhang, Yao Li, Caiwen Ding, Yanzhi Wang, Yi Liang, and Dongkuan Xu. You need multiple exiting: Dynamic early exiting for accelerating unified vision language model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10781–10791, 2023.

[11] Omar Hamed, Souhail Bakkali, Marie-Francine Moens, Matthew Blaschko, and Jordy Van Landeghem. Multimodal adaptive inference for document image classification with anytime early exiting. *arXiv preprint arXiv:2405.12705*, 2024.

[12] Soheil Hor and Amin Arbabian. Sense, predict, adapt, repeat: A blueprint for design of new adaptive ai-centric sensing systems. *arXiv preprint arXiv:2312.07602*, 2023.

[13] Eric Fassbender and Wolfgang Heiden. The virtual memory palace. *Journal of Computational Information Systems*, 2(1):457–464, 2006.

[14] Jan-Paul Huttner, Kathrin Robbert, and Susanne Robra-Bissantz. Immersive ars memoria: evaluating the usefulness of a virtual memory palace. 2019.

[15] 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.

[16] Zhao Yang, Jiaxuan Liu, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. Appagent: Multimodal agents as smartphone users. *arXiv preprint arXiv:2312.13771*, 2023.

[17] Dongfang Liu, Yiming Cui, Zhiwen Cao, and Yingjie Chen. Indoor navigation for mobile agents: A multimodal vision fusion model. In *2020 international joint conference on neural networks (IJCNN)*, pages 1–8. IEEE, 2020.

[18] Microsoft recall. https://support.microsoft.com/en-us/windows/retrace-your-steps-with-recall-aa03f8a0-a78b-4b3e-b0a1-2eb8ac48701c, 2024.

[19] CNBC. Apple apologizes for listening to Siri conversations. https://www.cnbc.com/2019/08/28/apple-apologizes-for-listening-to-siri-conversations.html, 2019. Accessed: 2024-09-06.

[20] Apple. https://www.apple.com/newsroom/2023/12/report-2-point-6-billion-records-compromised-by-data-breaches-in-past-two-years/, 2022.

[21] Zhengcong Fei, Xu Yan, Shuhui Wang, and Qi Tian. Deecap: Dynamic early exiting for efficient image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12216–12226, 2022.

[22] Xiangjie Li, Chenfei Lou, Yuchi Chen, Zhengping Zhu, Yingtao Shen, Yehan Ma, and An Zou. Predictive exit: Prediction of fine-grained early exits for computation-and energy-efficient inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 8657–8665, 2023.

[23] 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.

[24] Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Dan Roberts. The unreasonable ineffectiveness of the deeper layers. In *The Thirteenth International Conference on Learning Representations*, 2025.

[25] Meiqi Wang, Jianqiao Mo, Jun Lin, Zhongfeng Wang, and Li Du. Dynexit: A dynamic early-exit strategy for deep residual networks. In *2019 IEEE International Workshop on Signal Processing Systems (SiPS)*, pages 178–183. IEEE, 2019.

[26] 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.

[27] Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, et al. Layer skip: Enabling early exit inference and self-speculative decoding. *arXiv preprint arXiv:2404.16710*, 2024.

[28] Rongkang Dong, Yuyi Mao, and Jun Zhang. Resource-constrained edge ai with early exit prediction. *Journal of Communications and Information Networks*, 7(2):122–134, 2022.

[29] Android: Low memory killer dae- mon. https://source.android.com/docs/core/perf/lmkd, 2022.

[30] Niel Lebeck, Arvind Krishnamurthy, Henry M Levy, and Irene Zhang. End the senseless killing: Improving memory management for mobile operating systems. In *2020 USENIX Annual Technical Conference (USENIX ATC 20)*, pages 873–887, 2020.

[31] ASUS. ROG Phone 9 Pro. https://rog.asus.com/phones/rog-phone-9-pro/, 2024. Accessed: 2024-12-20.

[32] Fabawi. Imagebind-lora: Fine-tuning "imagebind one embedding space to bind them all" with lora, 2023. Accessed: 2024-12-06.

[33] Chunyuan Qin, Chuan Deng, Jiashun Huang, Kunxian Shu, and Mingze Bai. An efficient faiss-based search method for mass spectral library searching. In *2020 3rd International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE)*, pages 513–518. IEEE, 2020.

[34] Rongjie Yi, Xiang Li, and Mengwei Xu. mllm. https://github.com/UbiquitousLearning/mllm, 2024.

[35] Huggingface. optimum-quanto. https://github.com/huggingface/optimum-quanto, 2025. Accessed: 2025-03-31.