Efficient Deployment of Large Language Models on Resource Constrained Edge Computing Platforms

Ismail Bustany, Yiyu Shi, Jinjun Xiong

THE CHIPS TO SYSTEMS CONFERENCE

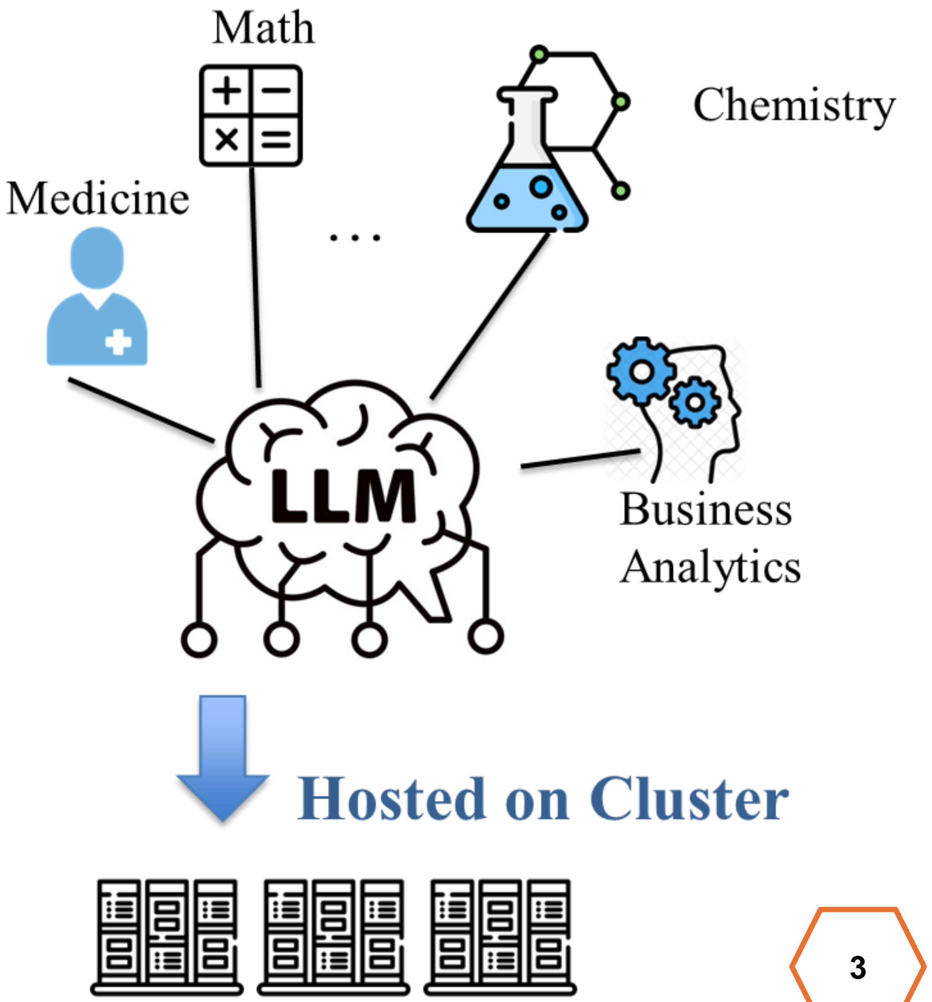
SPONSORED BY

Introduction

The Success of Large Language Models

“As models scale, they approach or surpass task-specific baselines, showing promise as universal systems for natural language understanding”

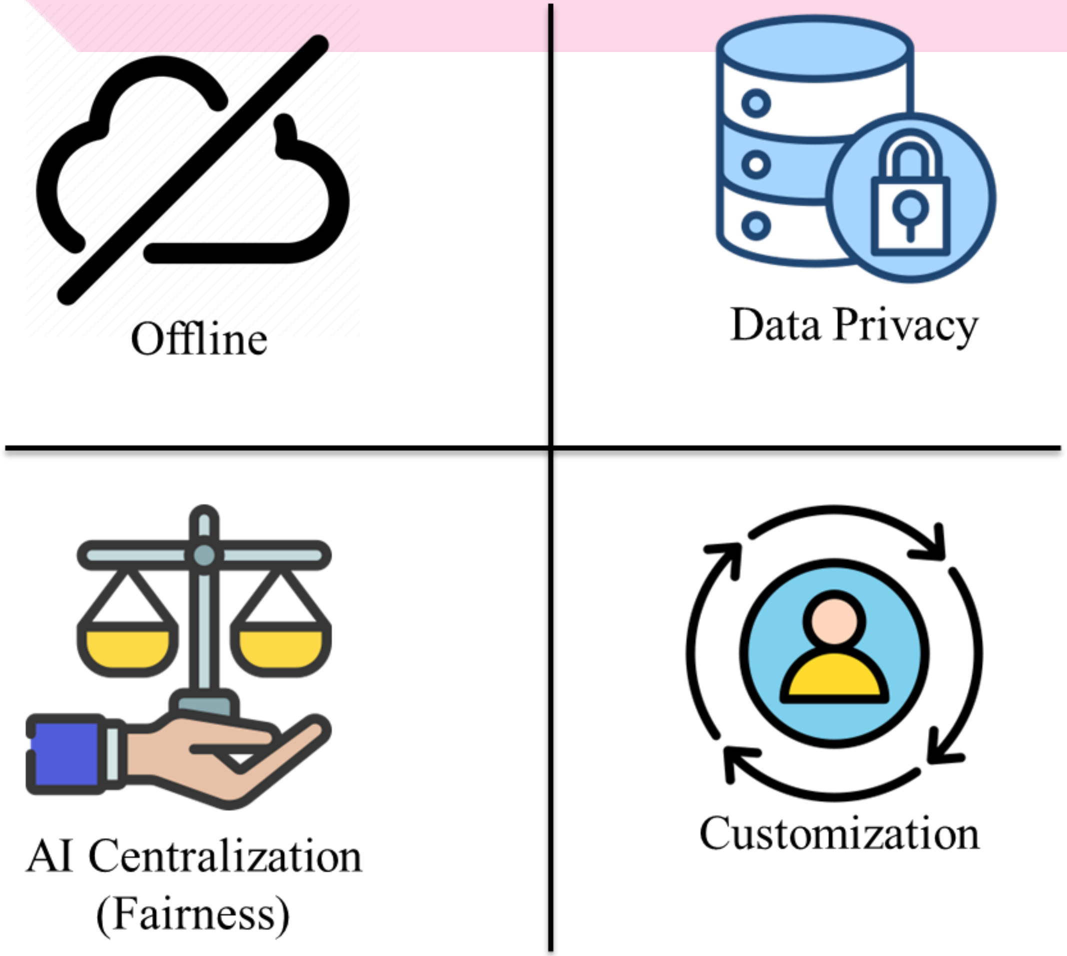
-- By Scaling Law from OpenAI



LLM is powerful, but…

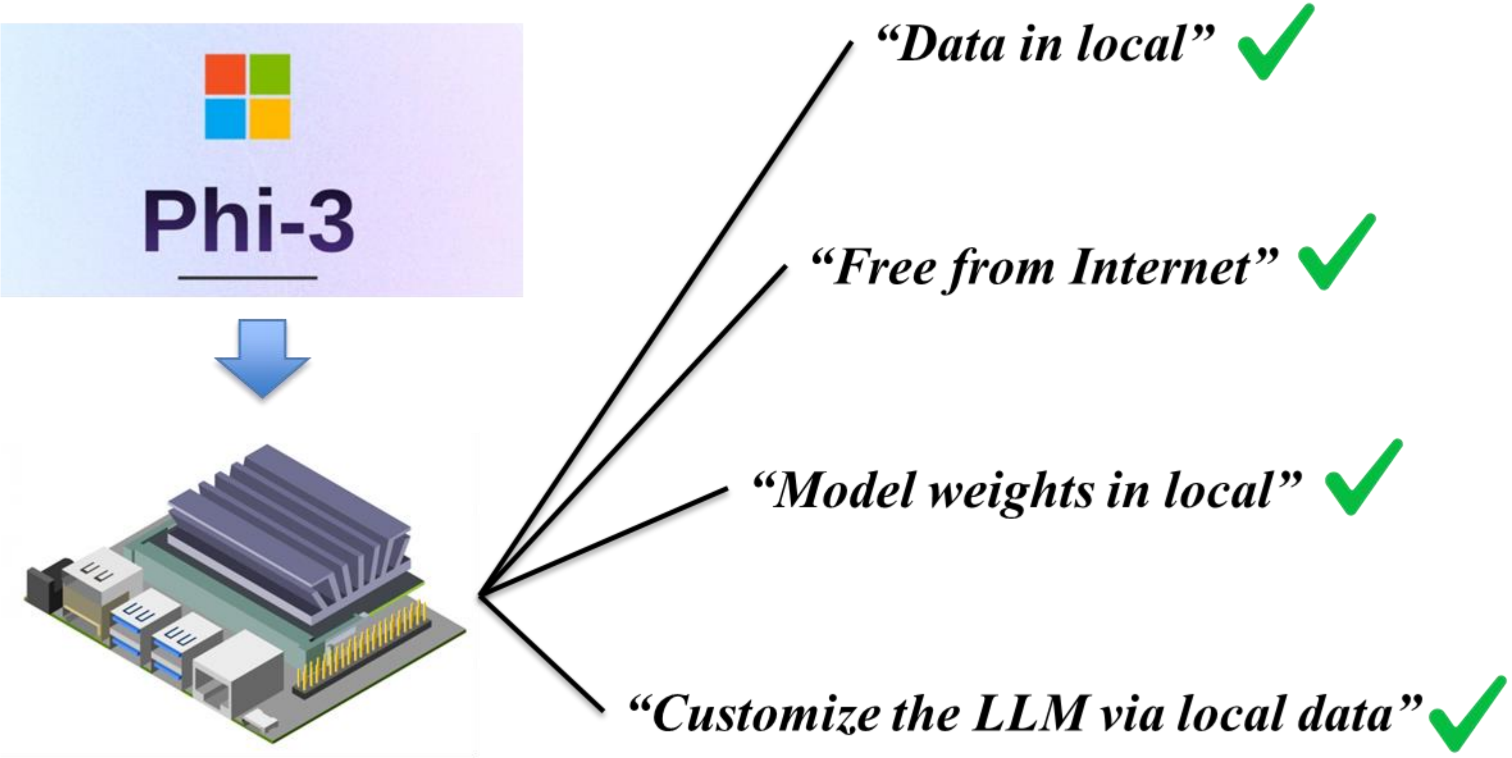
Vision: LLM hosted on cluster can achieve many tasks, but is compromised by **certain concerns**:

- **Offline** → Internet is unavailable/unstable, but real-time reaction is required (suicide detection, auto-drive)
- **Data Privacy** → Medical history, personal information
- **AI Centralization** → Only large corps can own models, data, and computational resources (clusters)
- **Customization** → LLM needs to adapt users with distinct situations



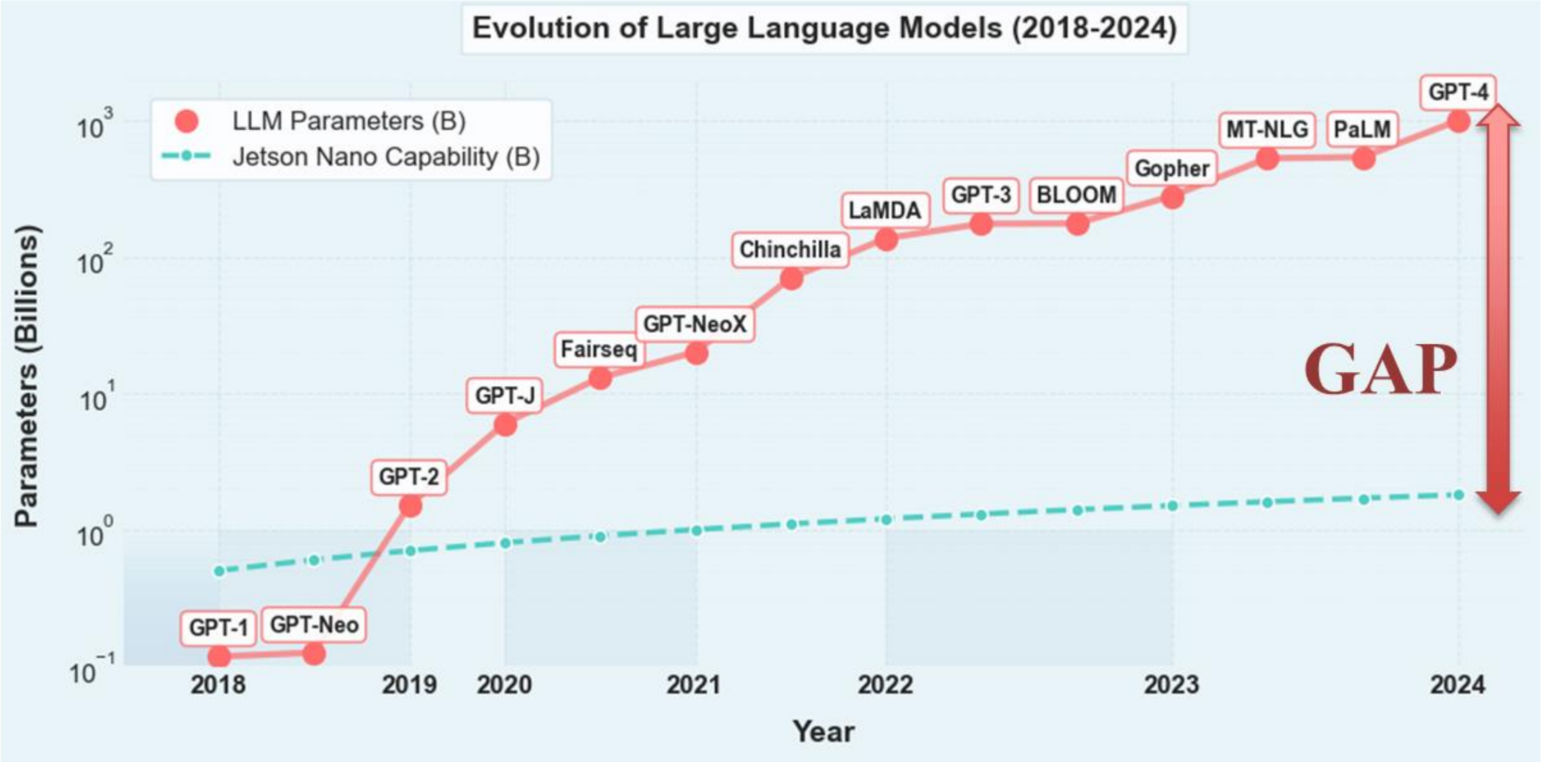
Edge-based LLM can be a solution

- LLM deployed on the edge device can **avoid these concerns**.
- Microsoft's Phi model, has successfully demonstrated the **power of edge-friendly** LLM



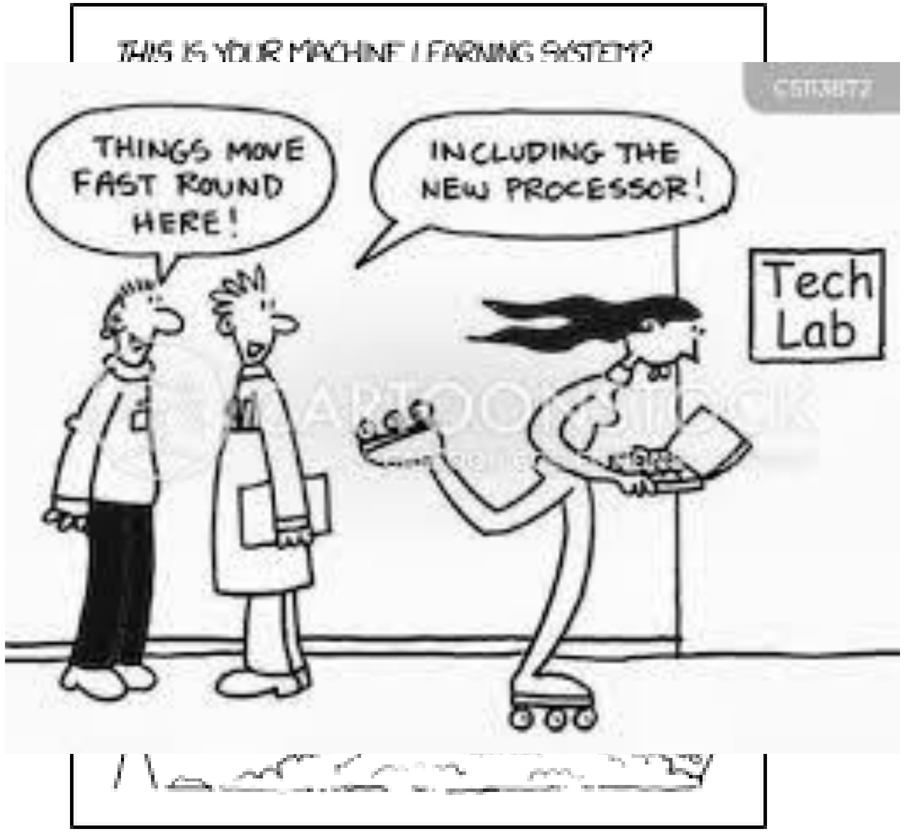
Gap Between LLM and Edge Devices

- LLM is growing much faster than the upgrade of edge devices
- Challenges:
  - Computation complexity
  - Memory capacity
  - Energy efficiency

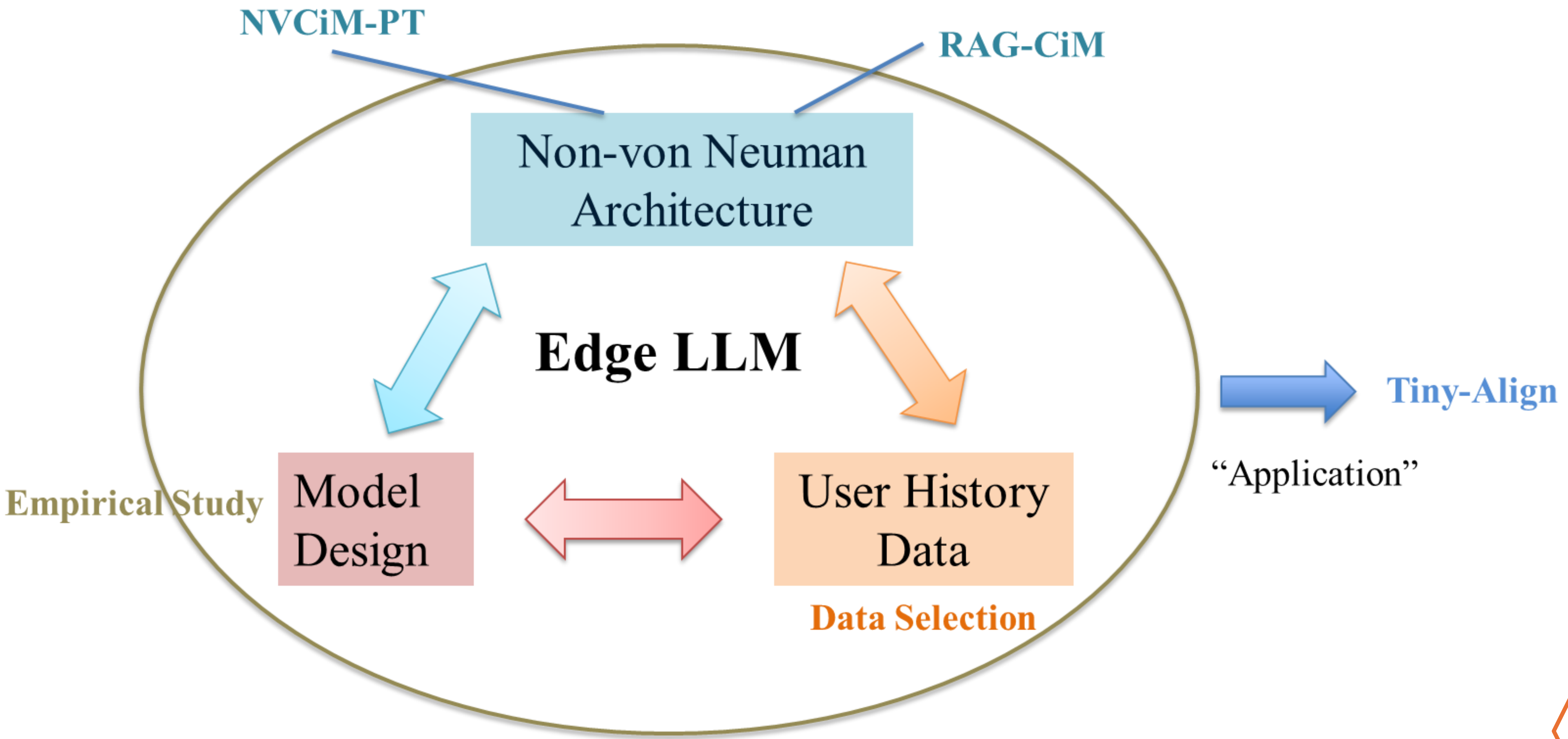


A Successful Edge LLM should be able to …

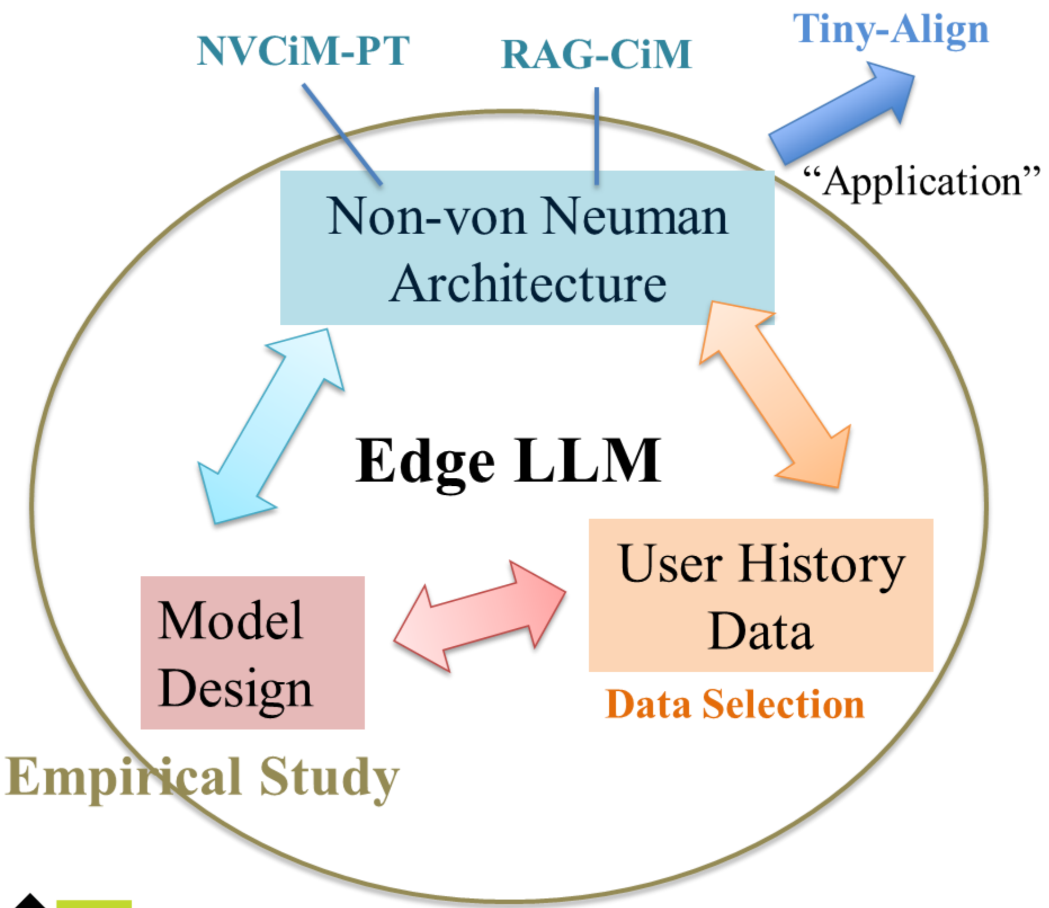
- **Tradeoff**: Use resource wisely among model weights and user data during training/inference
- **Personalization**: Generate user-preferred/related response
- **Robustness**:
  - Continuously growing performance over experience
  - Handle out-of-distribution scenarios



Build Up Efficient LLM on Edge Devices

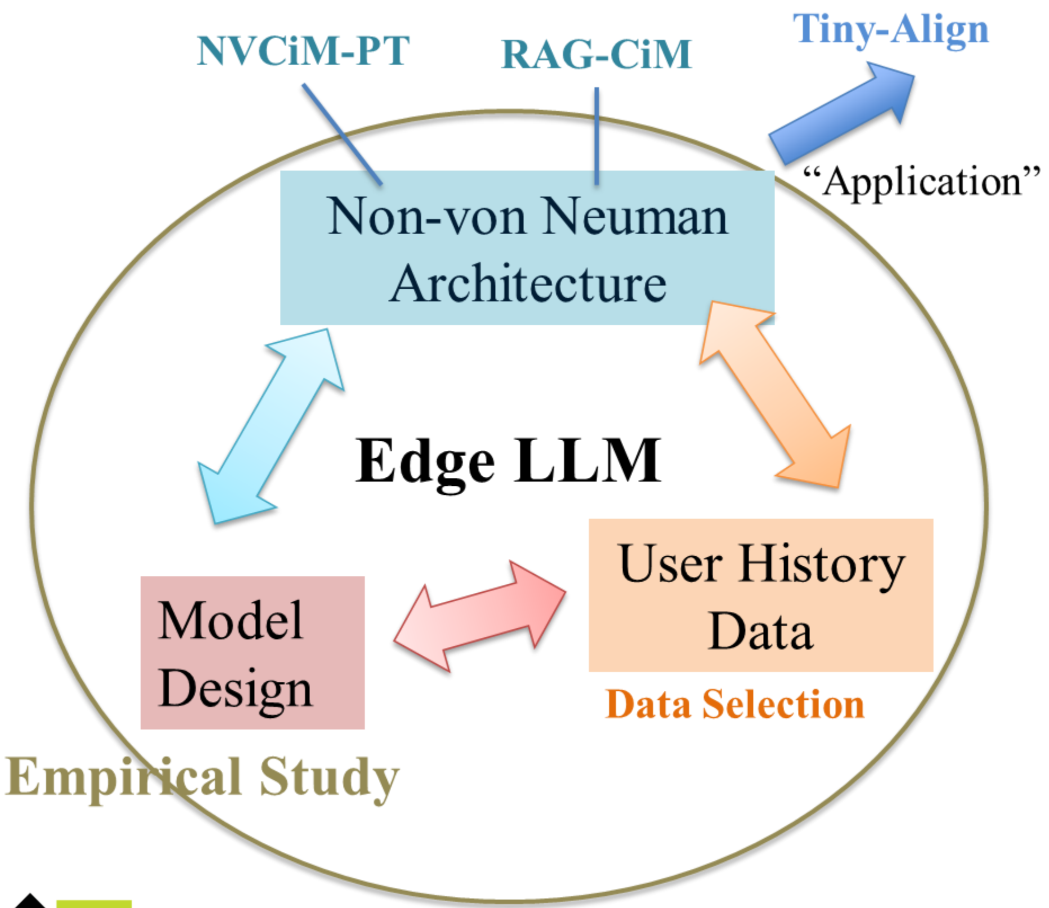


Section 1: Model Design



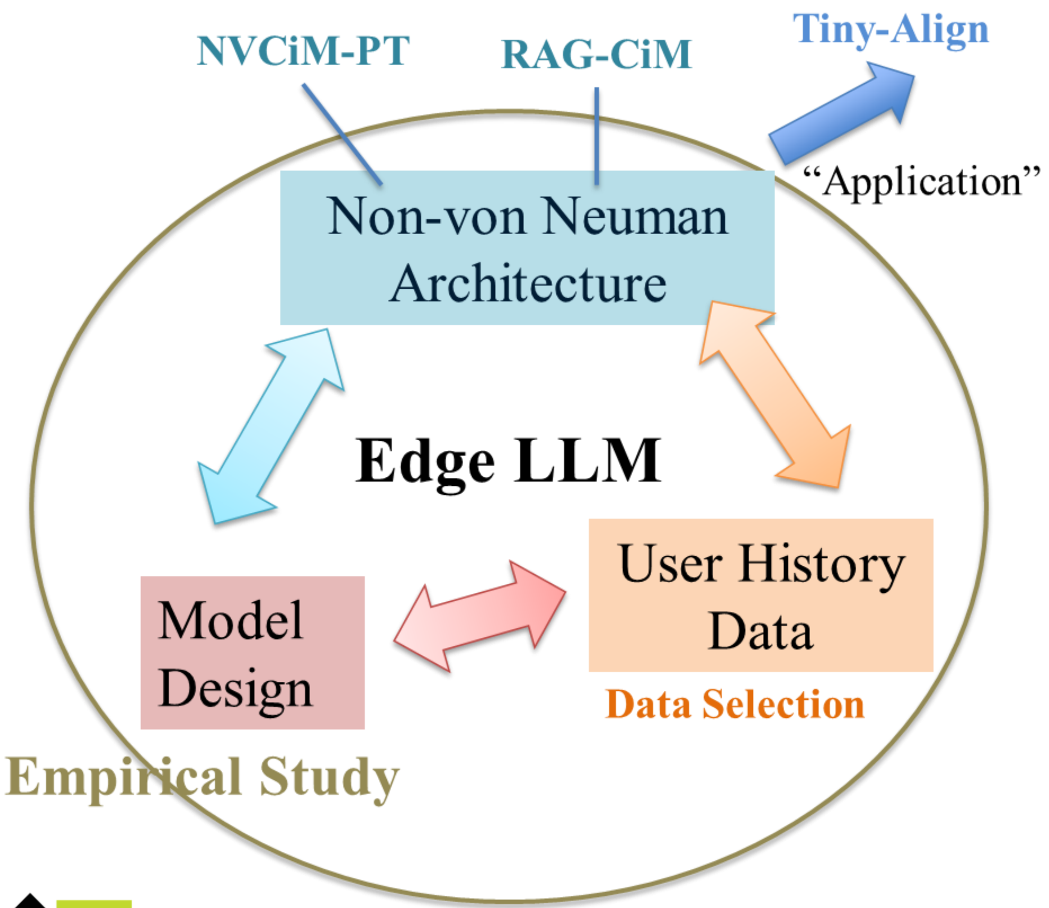
- **Overview**: Comprehensively evaluation of the tradeoff between learning, model weights, and user data
- **Contributions**:
  - First comprehensive study on edge LLM deployment
  - Guidelines for deployment and usages of LLM
  - Insights for future research/engineering questions

Section 2: Data Selection



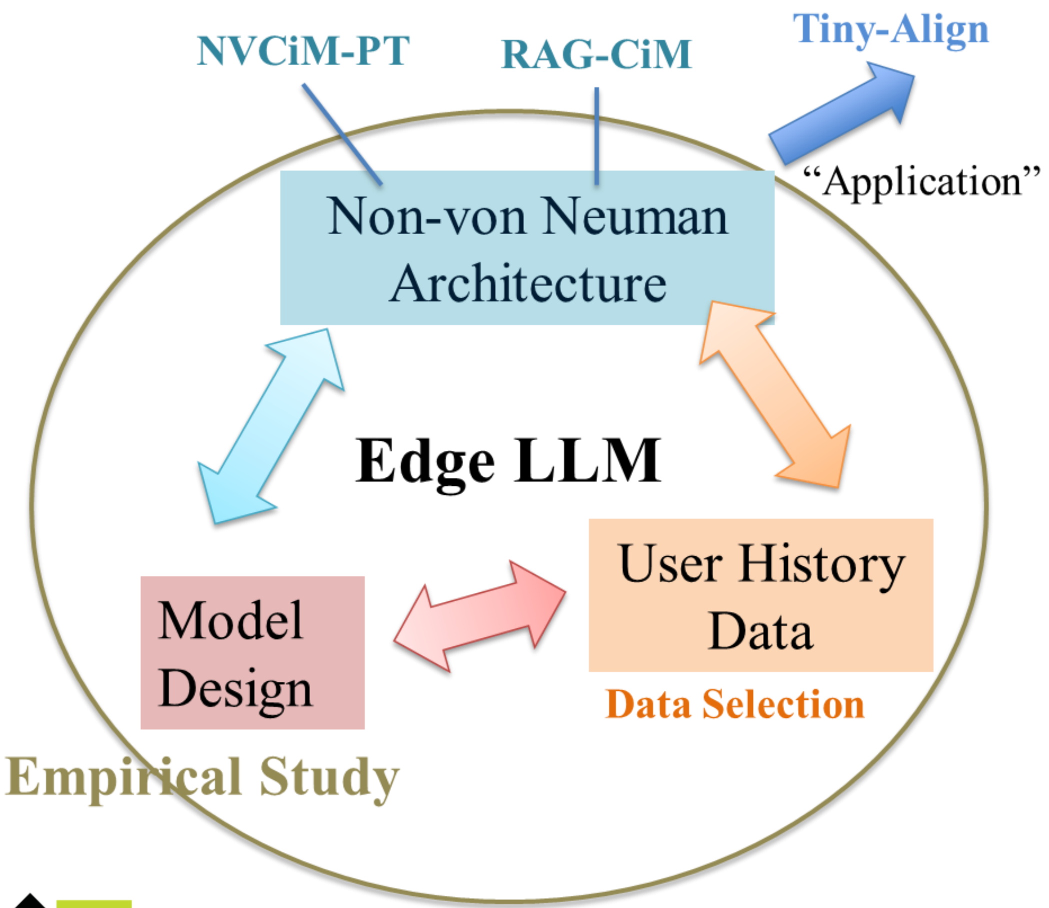
- **Overview**: Maintain a high-quality and compact user-generated data chunk on edge devices for training and inference
- **Contributions**:
  - First on-device data selection frameworks for LLM training
  - Resource-efficient data selection method

Section 3: RAG-CiM



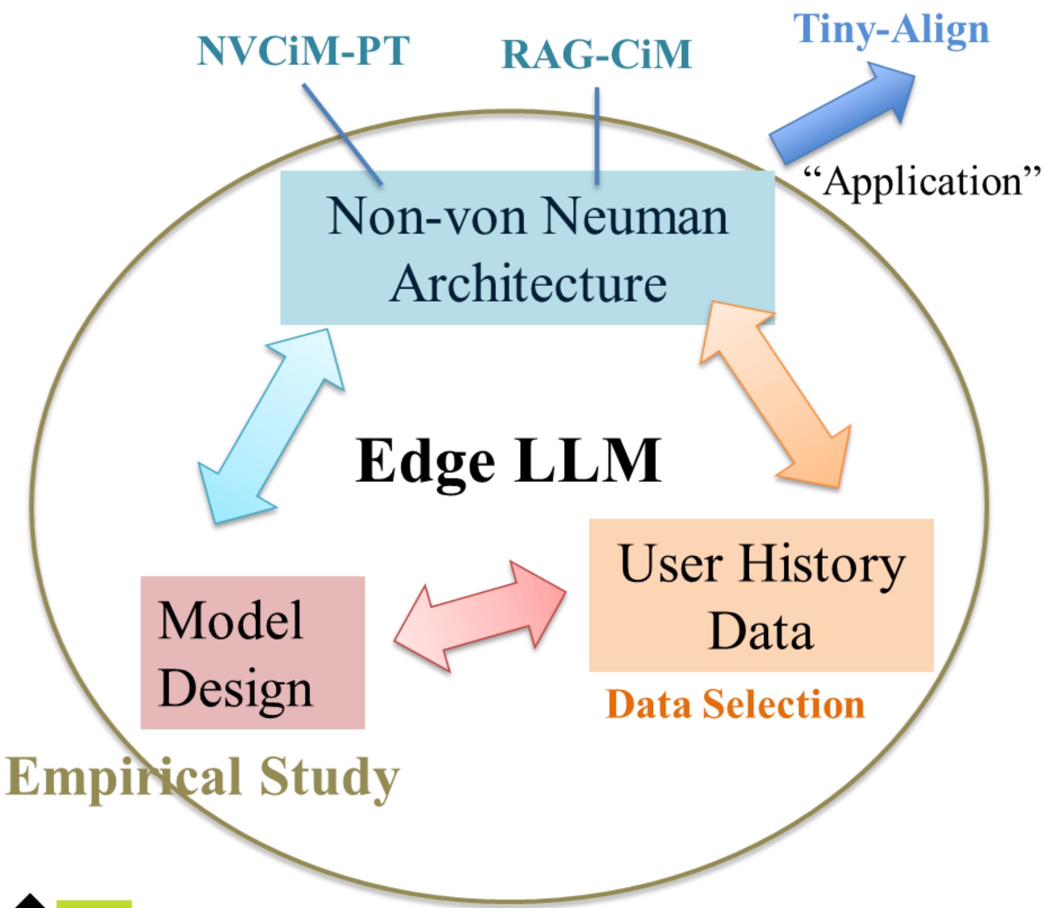
- **Overview**: Accelerate RAG via Compute-in-Memory (CiM) for LLM personalization in inference stage
- **Contributions**:
  - First using CiM to optimize the functionality in LLM
  - Accelerate retrieval-augmented generation (RAG) via applying max inner product search (MIPS) into in-memory computing

Section 4: NVCiM-PT



- **Overview**: Optimize prompt tuning, a LLM training method based on non-volatile CiM architectures
- **Contributions**:
  - Co-design the search algorithm and non-volatile memory (NVM) devices
  - Demonstrate the potential of LLM personalization acceleration via NVCiM

Section 5: Tiny-Align



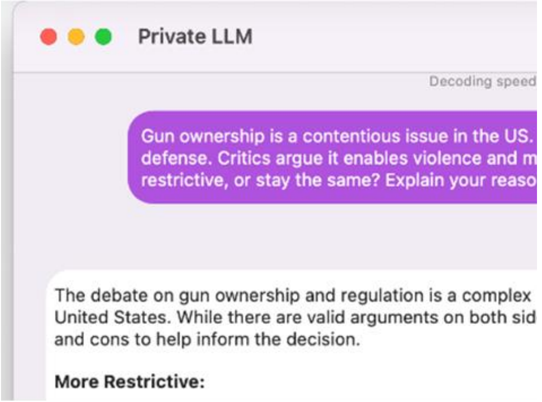
- **Overview**: Resource-efficient learning method to enable audio-based interaction between LLM and user
- **Contributions**:
  - First on-device cross-modal (audio, text) alignment framework
  - Largely benefit people with healthcare needs (Dementia, Aphasia, and Specific Language Impairment)

Section 1: Empirical Study

THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY CEDA IEEE Council on Electronic Design Automation sigda acm

Empirical Study onto Edge LLM



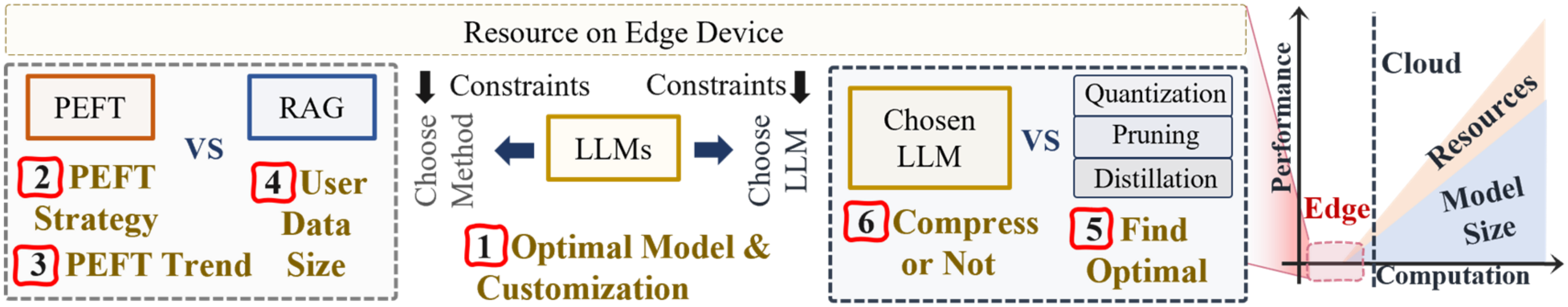
Llama on iPad

Chat with RTX

- Increasing needs to deploy LLMs on edge devices with various resource settings
- One single LLM might not work best for all user cases
- LLM learning in-situ can better fit the user knowledge domain and generate user-preferred response
- Various factors need to be considered, tradeoffs between model, resource, and data need to made, during LLM deployment on edge

Overview of Empirical Study

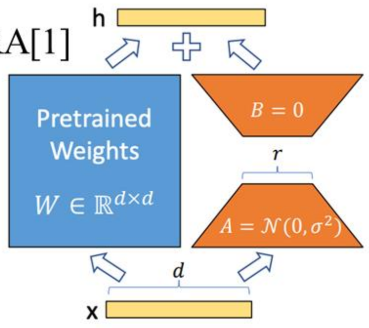
- This is the first work to systemically study the deployment of LLM on edge including
- It can provide guidelines to the future edge LLM usage (inference, training, deployment)
- It states the edge-appropriate LLM format



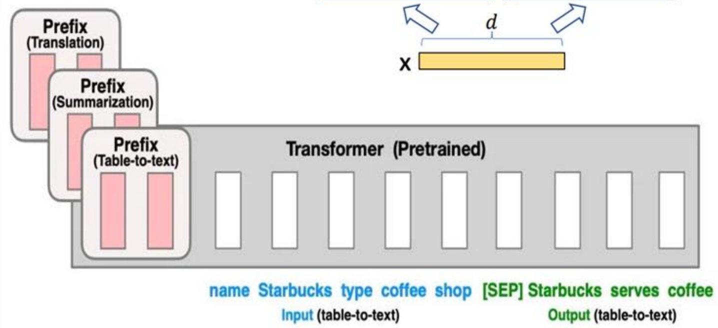
Overview of investigations on Edge LLM including model selection, parameter efficient fine-tuning (PEFT), retrieval-augmented generation (RAG), data size, compression and optimization

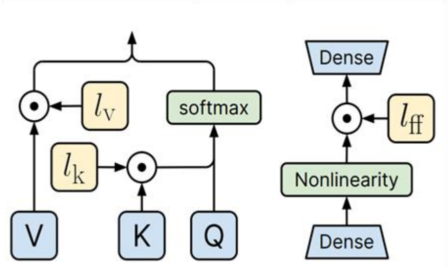
Concept Heads-up: Parameter Efficient Fine-Tuning (PEFT)

LoRA[1]



Prefix Tuning[2]





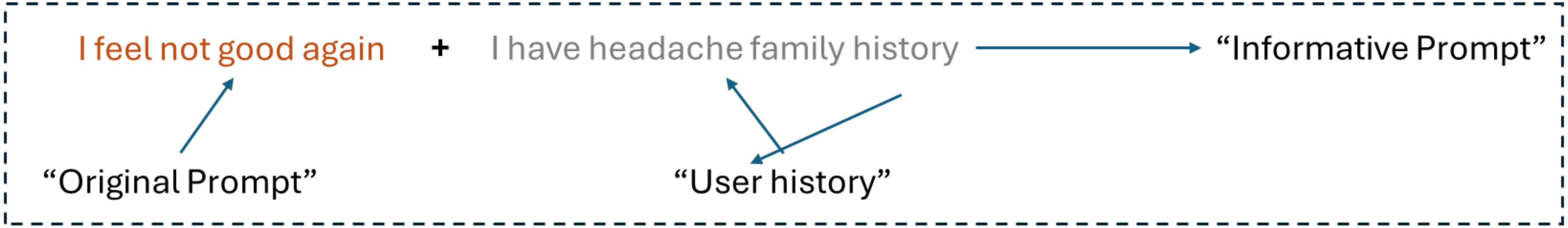
IA3[3]

- Employ a small portion of parameters in a model to fine-tune based on personal history data
- Enable LLM fine-tuning on edge device
  - Llama-7B: over **24GB** DRAM in **full-scale** fine-tune vs **5GB** DRAM in **PEFT**
- Implementation: LoRA (Low-Rank Adaption), Prefix/Prompt Tuning, IA3
- Train the additional adapters

[1] Hu, Edward J., et al. "Lora: Low-rank adaptation of large language models." arXiv preprint arXiv:2106.09685 (2021).
[2] Li, Xiang Lisa, and Percy Liang. "Prefix-tuning: Optimizing continuous prompts for generation." arXiv preprint arXiv:2101.00190 (2021).
[3] Liu, Haokun, et al. "Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning." Advances in Neural Information Processing Systems 35 (2022): 1950-1965.

Concept Heads-up: Retrieval-Augmented Generation (RAG)

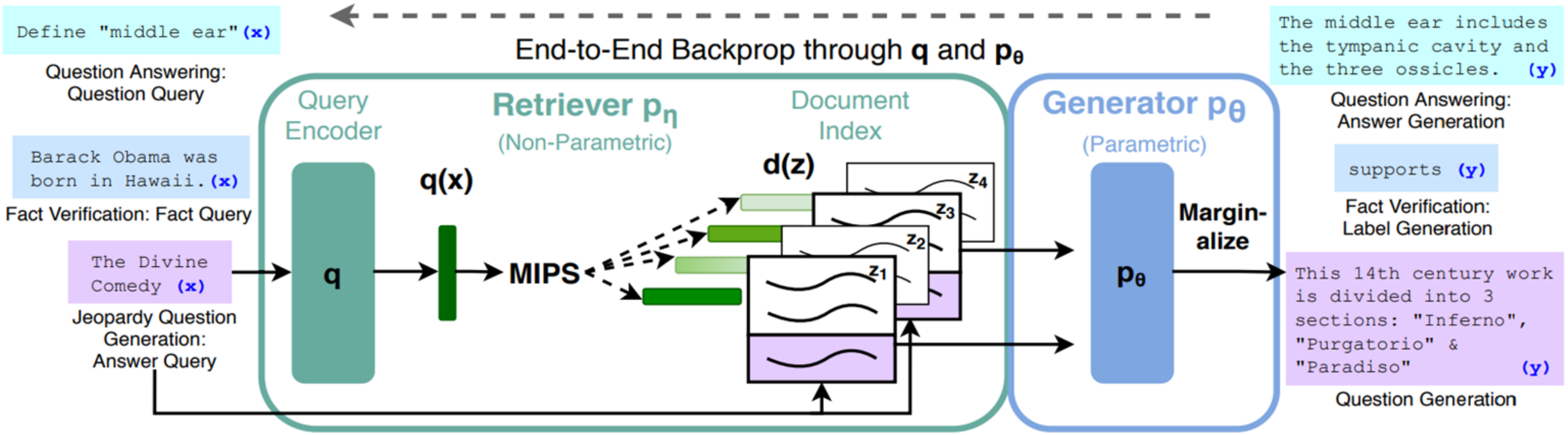
- LLM on edge: LLM can use the user **past data** to provide **personalized response (retrieval)**
- Why RAG? Parameter Learning can be computationally expensive; **RAG** uses **much less resources**.



- But data sample can accumulate → need to find the most relevant data

Concepts Heads-up: Retrieval-Augmented Generation (RAG)

- Mechanism:
  - Store user-related data (sentence embedding, in 1D vector)
  - Retrieve the data that mostly semantically relevant to user query (Retrieval algorithm like max inner product search - MIPS)
  - Concatenate the retrieved data with query
- Rationale: Provide each query with more context information

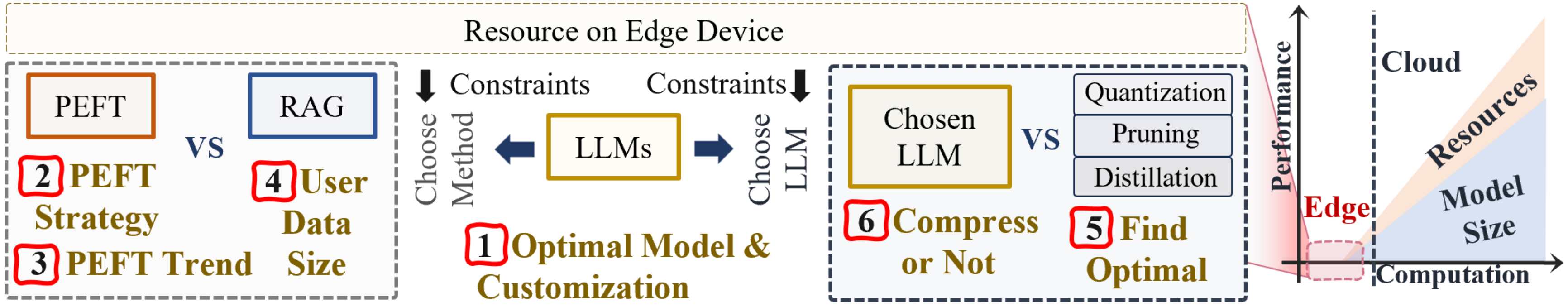


RAG[1]

[1] Lewis, Patrick, et al. "Retrieval-augmented generation for knowledge-intensive nlp tasks." Advances in Neural Information Processing Systems 33 (2020): 9459-9474.

Highlight Findings

- As the Increasing of Task Difficulty: PEFT → RAG → PEFT.
- Having a huge user data corpus
  - Burden the memory
  - Ineffective learning
  - No significant performance improvement in PEFT or RAG
- Compression Methods:
  - Pruning: Not recommended (If used, heavy training might be required)
  - Knowledge Distillation: Most stable
  - Quantization: highest peak performance



Overview of investigations on Edge LLM

Evaluation Datasets and Their Difficulties

- Each Dataset: contain one task like classify movie tags or summarize conversation content
- Normalized Accuracy (Difficulty):
  - Classification: LaMP-2 < LaMP-3 < LaMP-1
  - Generation: LaMP-6 < LaMP-5 < LaMP-7 < LaMP-4
  - Classification in general is easier than generation

Dataset	GPT-4	Claude 3 Opus	Gemini 1.0 Pro	Llama 3 70B	Average	**Normalized Accuracy**	Task Type
LaMP-1 (Accuracy)	0.539	0.539	0.490	0.422	0.4975	0.995	Classification
LaMP-2 (Accuracy)	0.355	0.320	0.300	0.400	0.3438	5.157	Classification
LaMP-3 (Accuracy)	0.667	0.657	0.510	0.755	0.6472	3.236	Classification
LaMP-4 (ROUGE-1)	0.143	0.171	0.139	0.131	0.1460	0.1460	Generation
LaMP-5 (ROUGE-1)	0.386	0.374	0.405	0.084	0.3123	0.3123	Generation
LaMP-6 (ROUGE-1)	0.351	0.356	0.405	0.278	0.3475	0.3475	Generation
LaMP-7 (ROUGE-1)	0.326	0.136	0.237	0.255	0.2385	0.2385	Generation

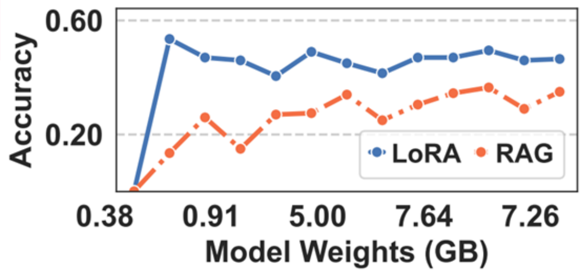
Task Difficulty Measurement

Optimal Model and Customization for LLMs on the Edge

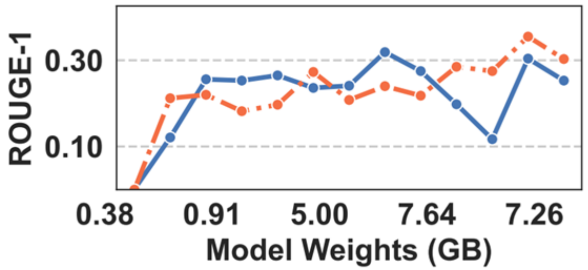
- Easy classification task → small LLM with PEFT
- Difficulty increases → RAG + Quantized LLM
- Difficulty even higher → PEFT + Quantized LLM

	Models	Py-2.8b	OPT-2.7b	Ll2-3b	Sta-3b	Ge-2b	Phi-2	Mis-7b-G	OpCt-3.5-G	Ge-7b-G	S-Ll-2.7b-P
LaMP-2	PfixT	0.223	0.105	0.035	0.145	0.130	0.122	0.205	0.145	0.185	0.033
	PmptT	0.055	0.025	0.045	0.087	0.070	0.204	0.085	0.215	0.170	0.050
	IA3	0.055	0.080	0.085	0.090	0.108	0.115	0.189	0.105	0.155	0.055
	LoRA	**0.475**	**0.480**	**0.430**	**0.480**	**0.430**	**0.450**	**0.485**	**0.460**	**0.420**	**0.455**
	RAG	0.320	0.110	0.295	0.245	0.205	0.340	0.375	0.290	0.365	0.035
LaMP-3	PfixT	0.490	0.392	0.520	0.412	0.500	0.451	0.206	0.627	0.275	0.265
	PmptT	0.373	0.304	0.461	0.471	0.451	0.324	0.353	0.647	0.324	0.451
	IA3	0.480	0.314	0.451	0.343	0.539	0.567	0.425	0.605	0.314	0.382
	LoRA	**0.716**	0.627	**0.765**	**0.784**	**0.784**	**0.745**	**0.814**	0.647	**0.775**	**0.725**
	RAG	0.716	**0.696**	0.461	0.667	0.765	0.627	0.755	**0.814**	0.480	0.578

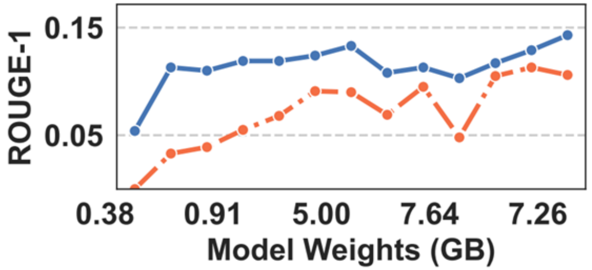
Performance (Accuracy) comparisons between parameter learning and RAG



Easy classification task (LaMP-2)



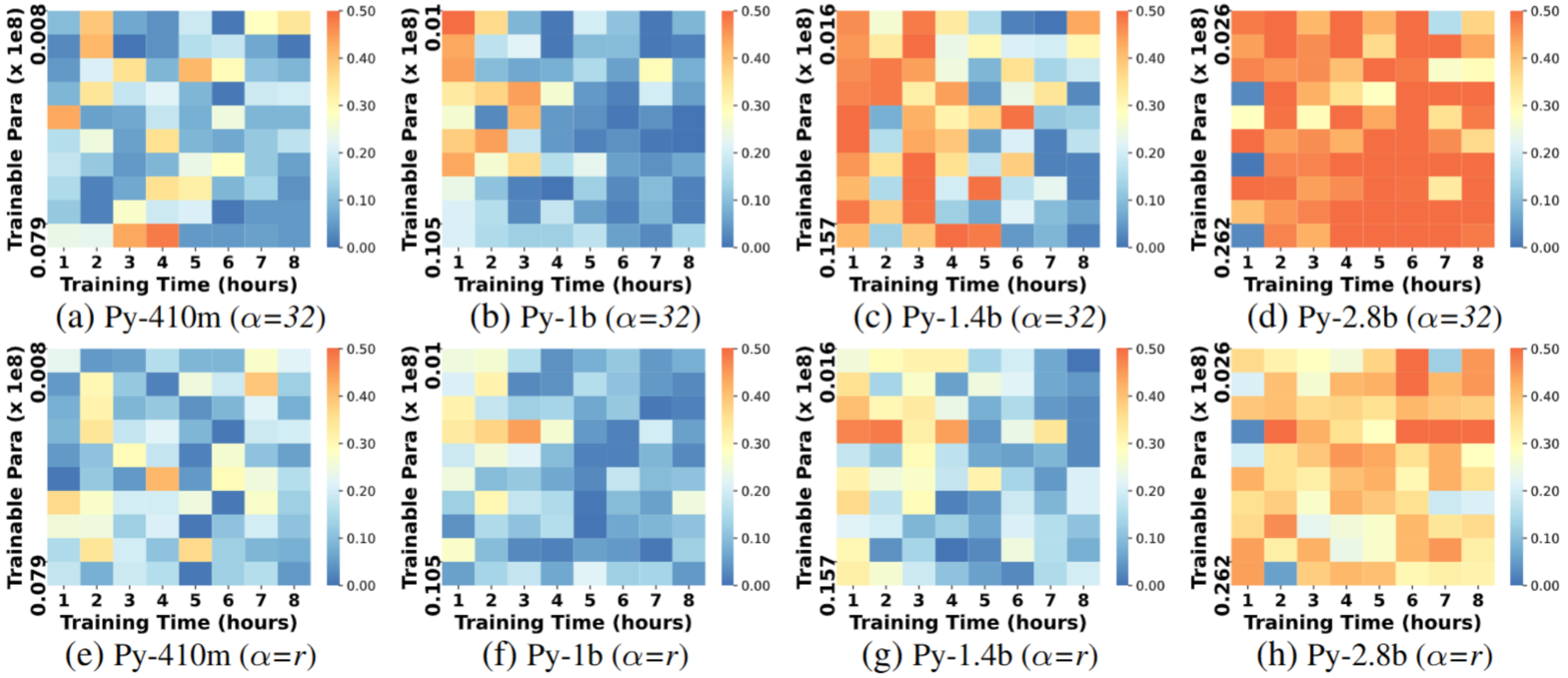
Easy summarization task (LaMP-6)



Hard summarization task (LaMP-4)

Choice of Parameter Efficient Fine-tuning (PEFT) Strategies

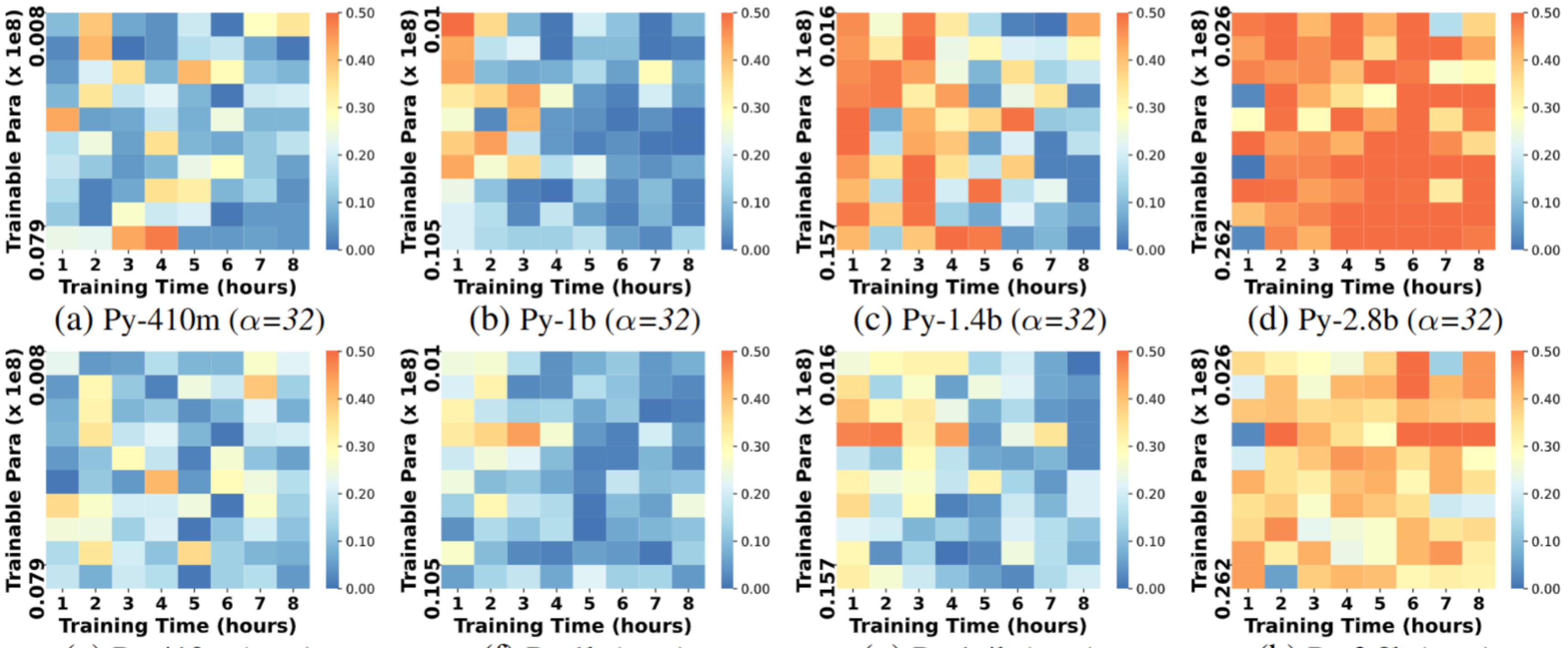
- In previous experiments, we find the optimal PEFT implement – LoRA.
- In the following investigations, we will dig in LoRA and information retrieval method – RAG.
- For LoRA, its **rank (r)** decides the number of trainable parameters, and **alpha (a)** decides how much impact that adapter on the original LLM



Performance (classification accuracy) for Pythia models with different sizes on LaMP-1, given alpha = rank = 16 or alpha = 32 but rank increases from 8 to 256

Choice of Parameter Efficient Fine-tuning (PEFT) Strategies

- Fixing alpha can benefit edge LLM PEFT more
- Setting alpha and rank to **(16, 16) or (16, 32)** can work in **most cases**.



(a) Py-410m (α=32) (b) Py-1b (α=32) (c) Py-1.4b (α=32) (d) Py-2.8b (α=32)

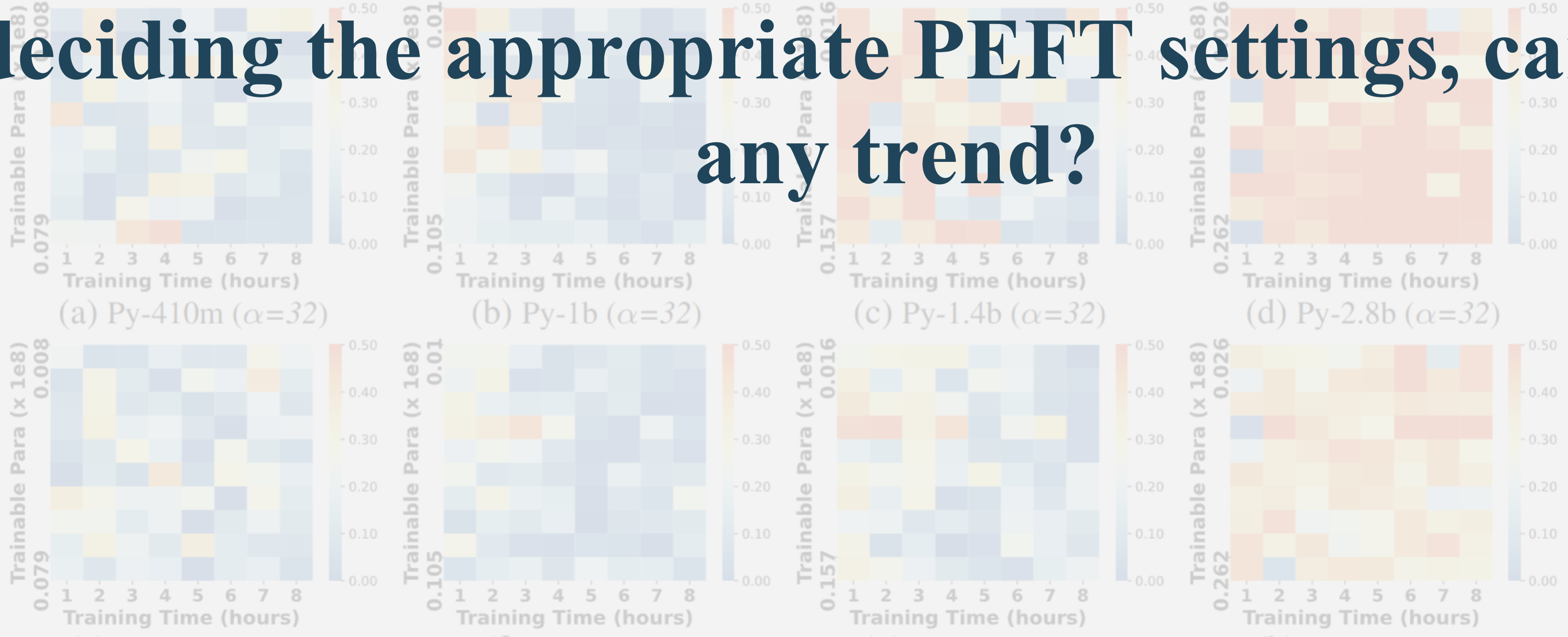
(e) Py-410m (α=r) (f) Py-1b (α=r) (g) Py-1.4b (α=r) (h) Py-2.8b (α=r)

Performance (classification accuracy) for Pythia models with different sizes on LaMP-1, given alpha = rank = 16 or alpha = 32 but rank increases from 8 to 256

Choice of Parameter Efficient Fine-tuning (PEFT) Strategies

- Fixing alpha can benefit edge LLM PEFT more
- Setting alpha and rank to **(16, 16) or (16, 32)** can work in **most cases**.

After deciding the appropriate PEFT settings, can we observe any trend?

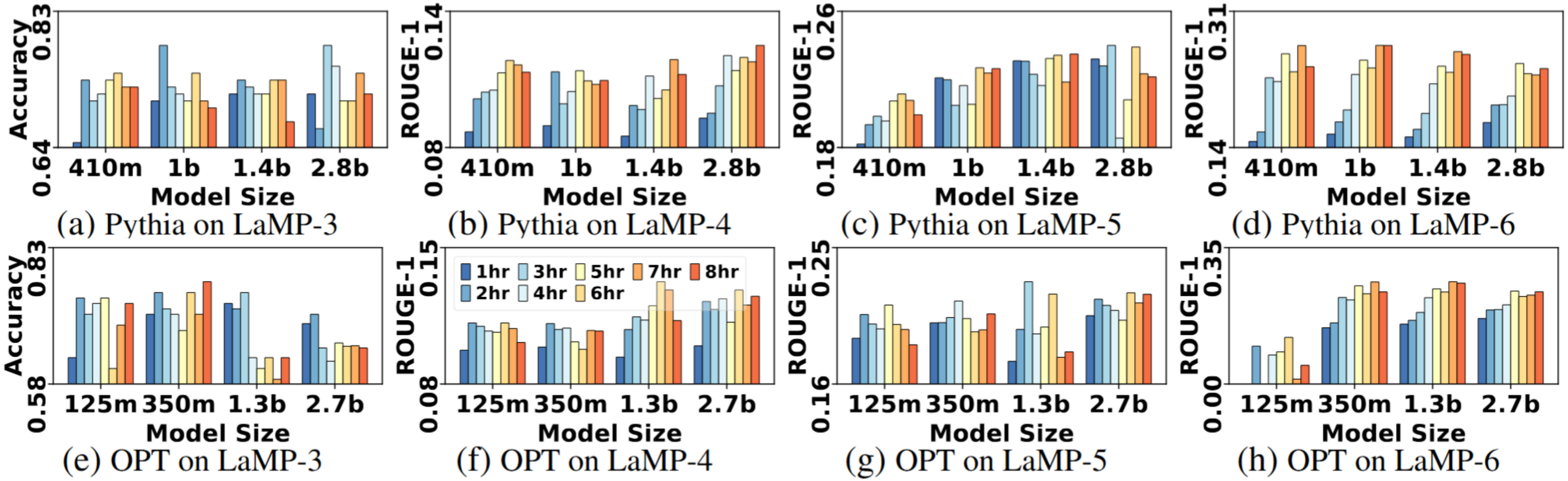


(a) Py-410m (α=32) (b) Py-1b (α=32) (c) Py-1.4b (α=32) (d) Py-2.8b (α=32)
(e) Py-410m (α=r) (f) Py-1b (α=r) (g) Py-1.4b (α=r) (h) Py-2.8b (α=r)

Performance (classification accuracy) for Pythia models with different sizes on LaMP-1, given alpha = rank = 16 or alpha = 32 but rank increases from 8 to 256

Identification of PEFT Trends for Edge LLMs

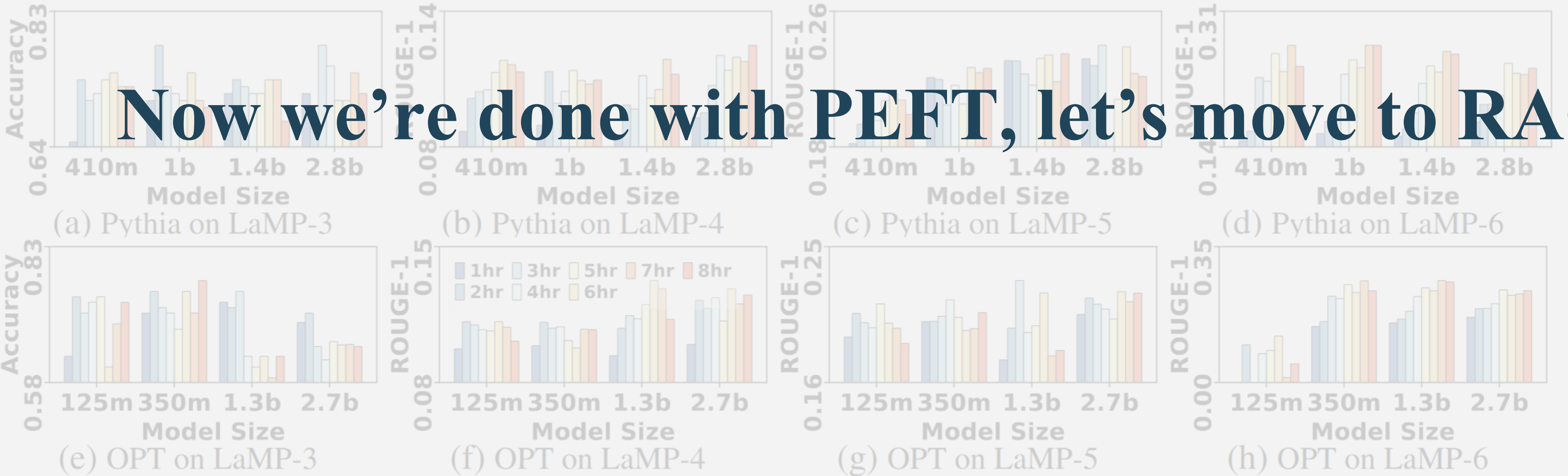
- More training data → NOT necessarily better performance
- Appropriate training time can be 3-4 hours
- Increasing training time only brings improvement on easiest task



Performance comparisons on multiple sized Pythia and OPT models on different amounts of training data

Identification of PEFT Trends for Edge LLMs

- More training data → NOT necessarily better performance
- Appropriate training time can be 3-4 hours
- Increasing training time only brings improvement on easiest task

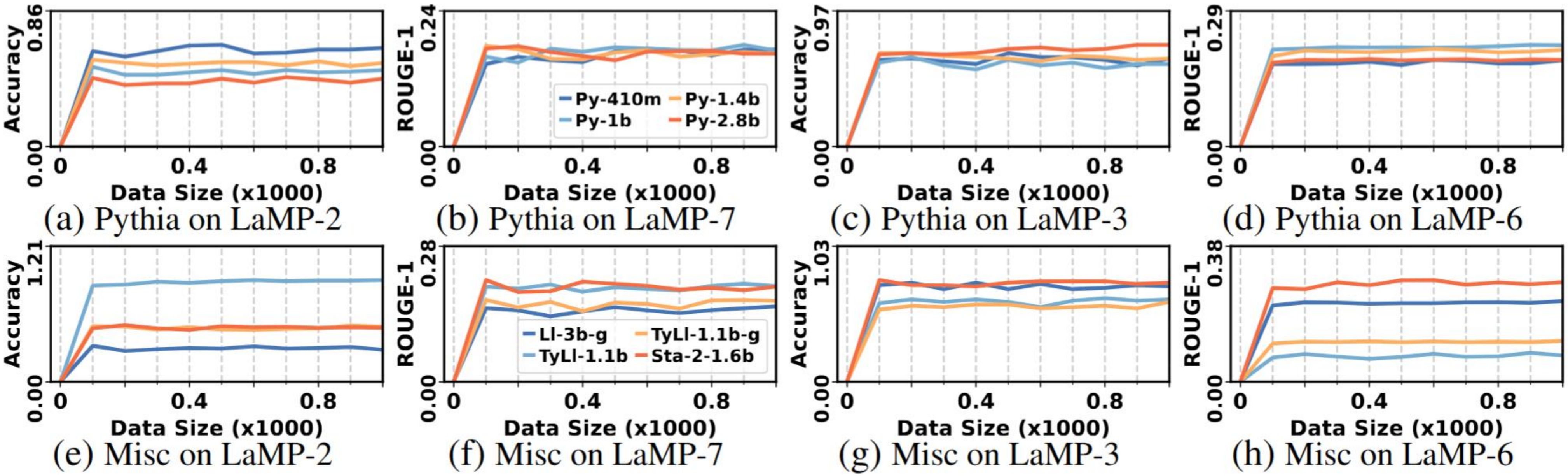


Now we're done with PEFT, let's move to RAG

Performance comparisons on multiple sized Pythia and OPT models on different amounts of training data

Impact of User History Data Volume on RAG Performance

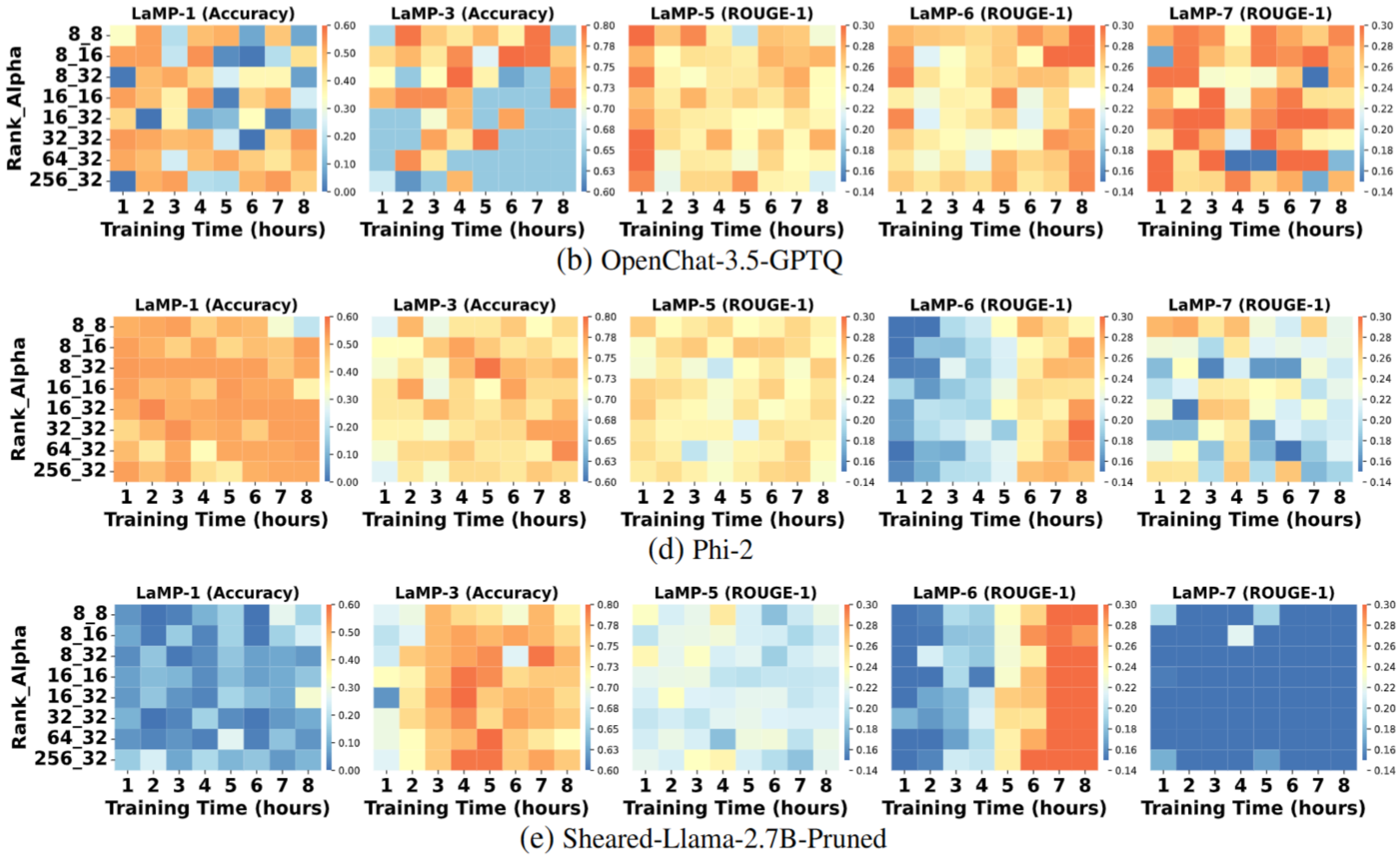
- Small size of data (i.e. 100) can support a decent RAG performance
- The **RAG** performance based on eight models is quite **consistent** across all **different sizes** of user **history** data



Performance improvement brought by RAG on four Pythia models and four miscellaneous (Misc) models across different sizes of user history data

Comparison of Model Compression Techniques on Edge LLMs

- Knowledge distillation (Phi) can be a safe option
- Different compression techniques are good at different types of tasks.
- Shearing is a very promising technique that preserves better performance, but they are less efficient at saving RAM compared to quantization.



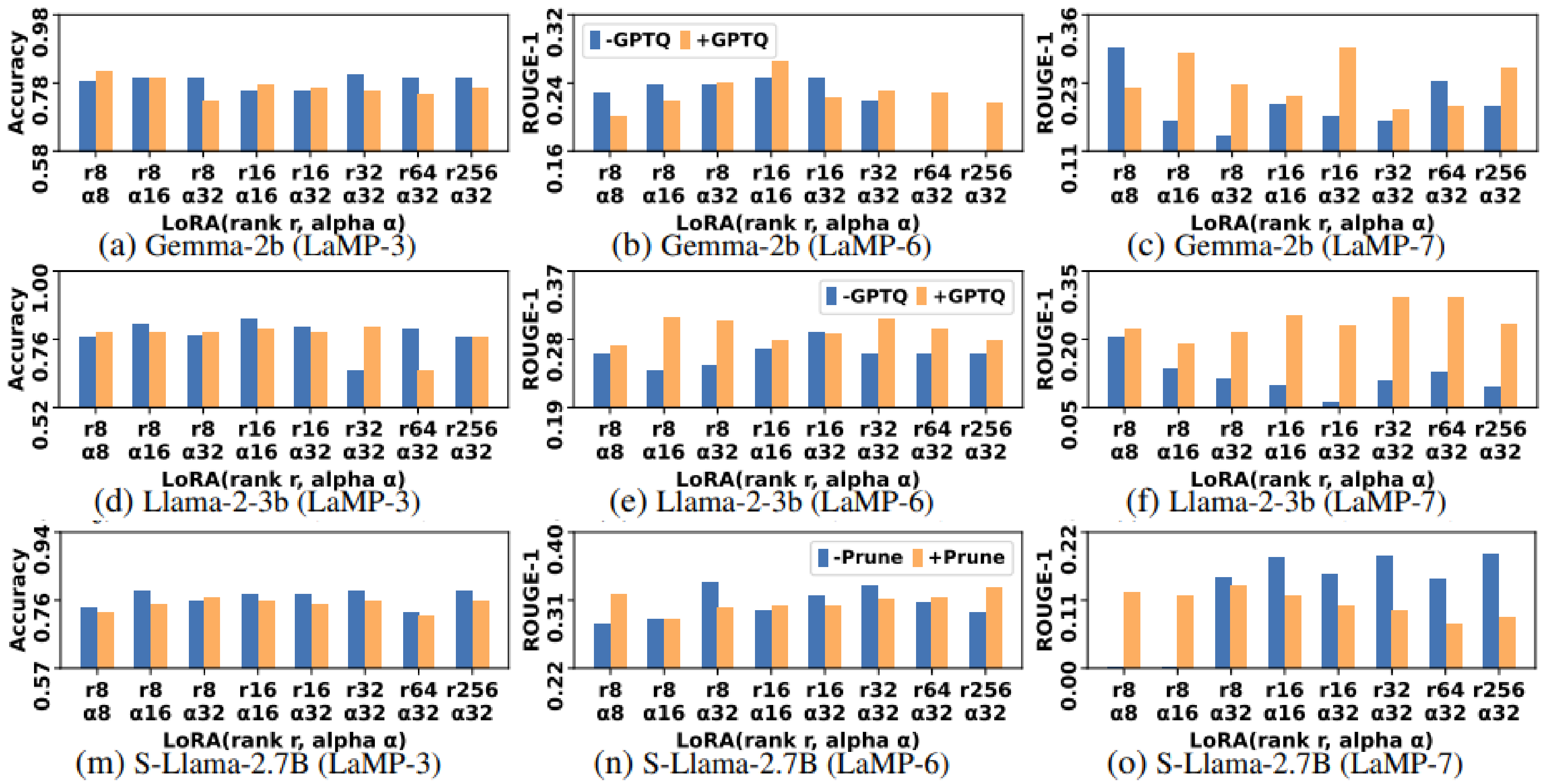
Performance comparisons between quantization (b), distillation (d), and pruning (e) LLMs.

Comparison between Compressed and Uncompressed Edge LLMs

- On challenging tasks (LaMP-7), quantization can improve the model performance via PEFT
- On the contrary, pruning can lower the model performance (LaMP-7)

Remark: what factors makes quantization better than pruning for LLM (Possible future research topic)

- Structure?
- Pretrained knowledge?
- Easy for fine-tuning?



Performance comparisons between quantization (a - f) and pruning (g - i) LLMs and their original versions.

Section 2:
Data Selection

THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY

Data Selection for LLM Personalization

LLM personalization in general

Learn from user-generated data

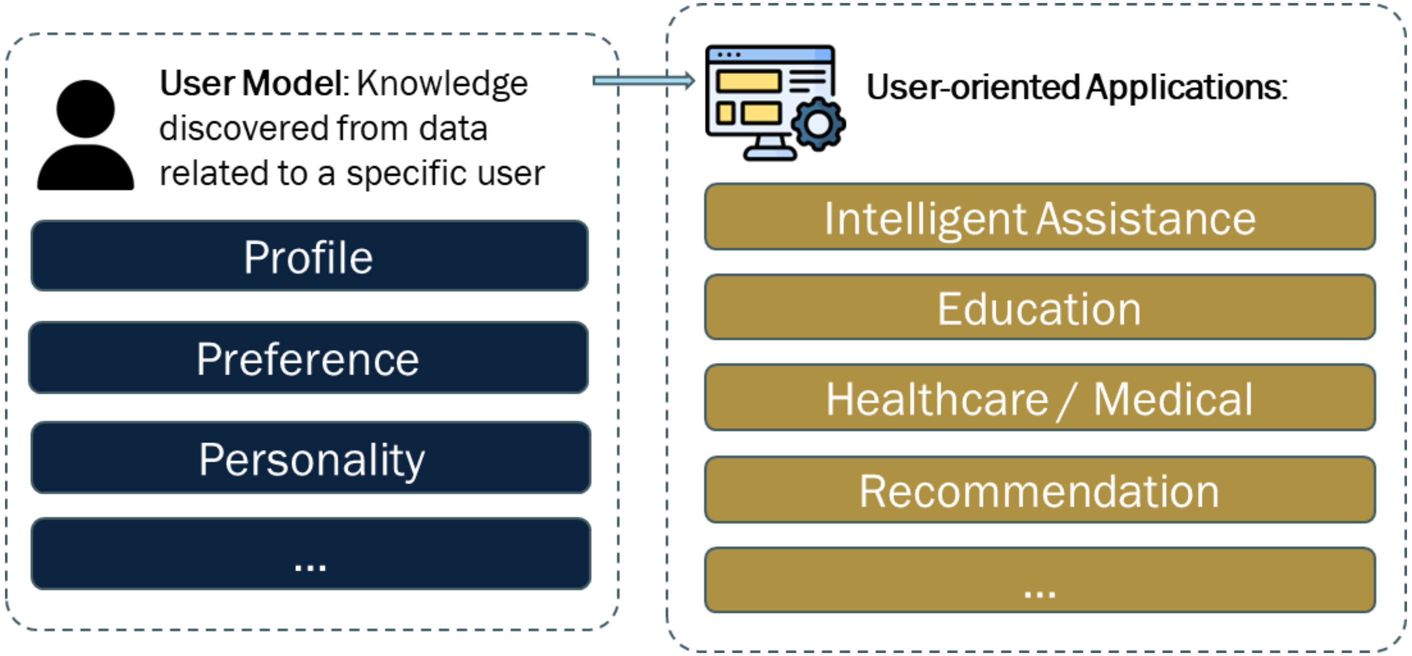
- Extract information from raw data
- Learn from labeled and annotated data

Generate user preferred content

- Generating should consider user-related data

Being expert in a few domains

- Need knowledge in more than one area



Different Things to Consider

Computational Power

- Ten to **hundreds times less** than Server's computational power
- Given the same time, **limited tokens** can be processed on edge

Limited Data Buffer

- DRAM is near to **drain out** by the model weights
- Data **buffer** for user-generated data can be **small** (i.e. 50MB)
- Data **in-stream** need to be process in **real-time**

Data Quality

- Low quality data contains **few user-related** information
- Learning from **high quality** data can **save resources**

Llama-7B	Edge		Server	
	Jetson TX2	AGX Xavier	Nvidia A10	Nvidia A100
FLOPS	1.33	11.33	150	312
Tokens per sec	<1	1 to 5	50 to 200	500 to 1000
DRAM	8GB	16GB	24GB	40 to 80GB

Example of Llama-7B on selected devices

uh oh it's twenty twenty-two.
however you'd like to.
no no and so you're not.
okay yep i can. okay.
let me hang on one second.

Low quality data

I have heard disease history,
but recently I am doing well.
When I get depressed for a
long time, then I usually will
have heard disease

High quality data

Background and Issues

Select data from real-time streaming

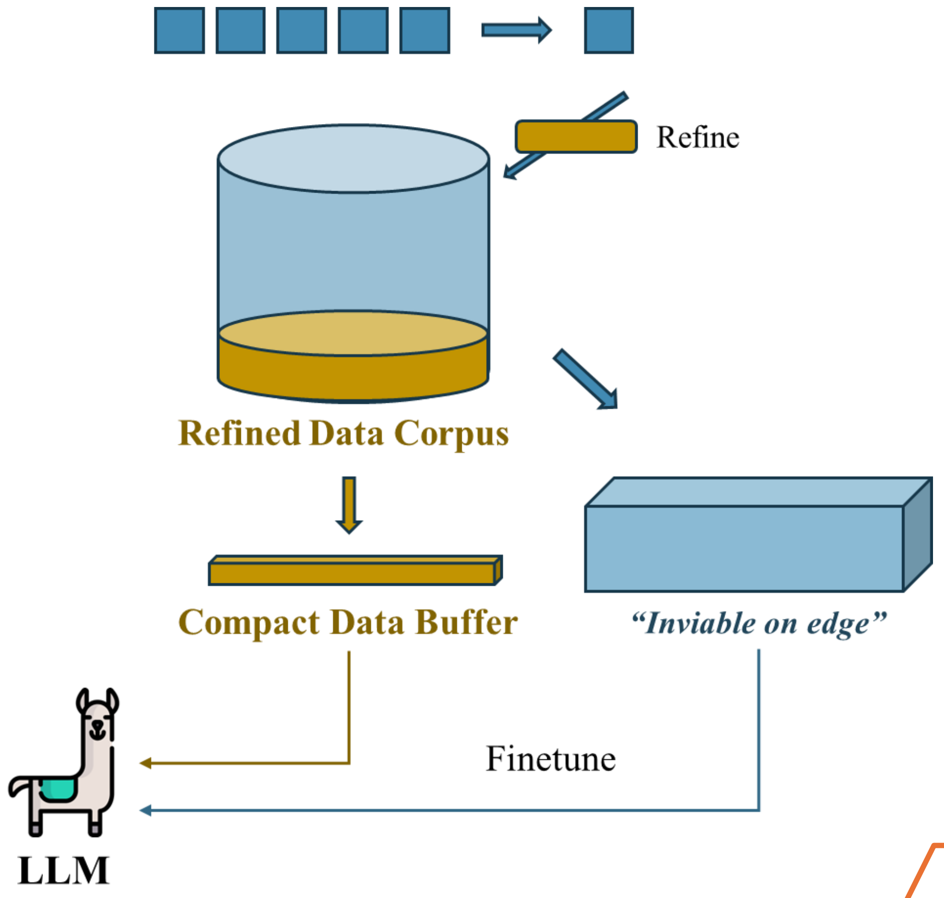
- User-generated data **continuously** get into data buffer
- Streaming data can be **temporally correlated** within mini-batch

Maintain a compact data buffer

- **Store** data on disk, **move** data to DRAM when using them
- It takes time to **retrieve** the proper data, and
- **Data movement** can cost more latency

Fine-tune LLM with scarce data

- When we select **high-quality** data, and maintain a small volume
- Such data may **not be enough** to finetune LLM



Our Data Selection (Enrich by Data Synthesis)

On-device LLM personalization framework

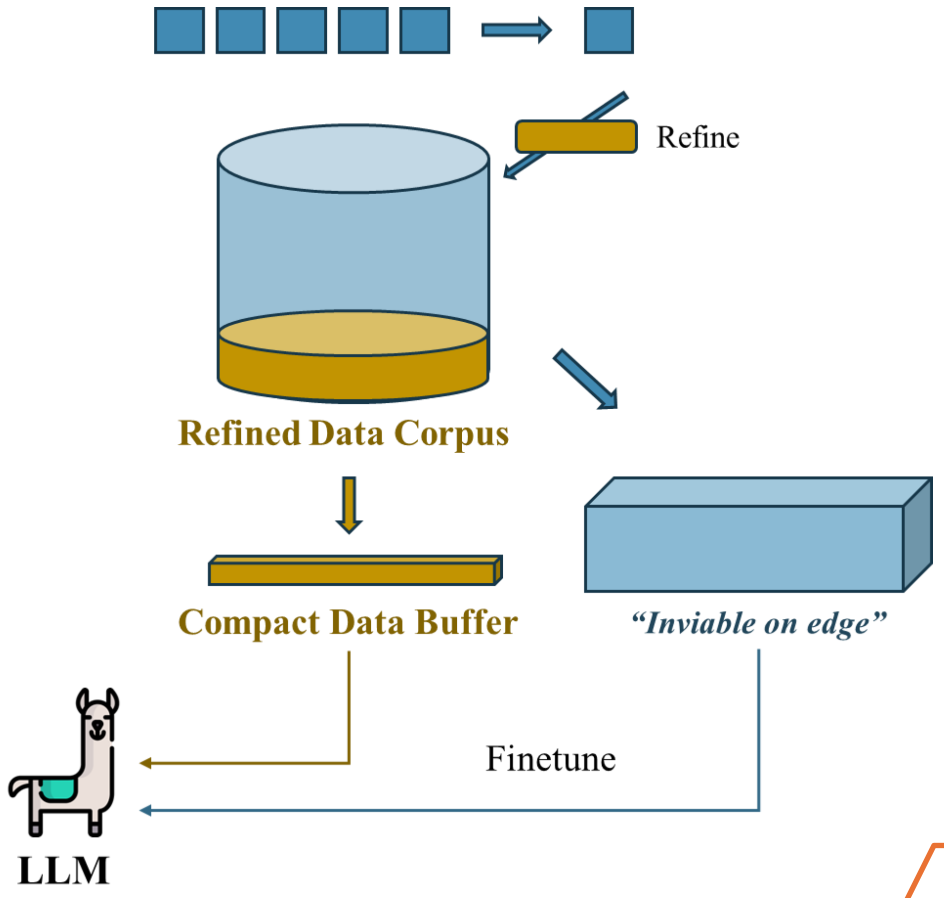
- Form mini-batch and **train on the fly**
- Use a small data buffer and eliminate the necessity of storing all streaming data

Data selection based on quality metrics

- A data **replacement** policy based on three quality metrics
- Save the most representative data
- **Annotation** is **not needed** in data replacement

Data synthesis for labeled data

- Utilize the embedded LLM as a data **synthesizer**
- Form **semantically similar** data from selected data



Data Selection

Estimate the information volume, Keep domain-specific data, Drop correlated data

Metric 1: Entropy of Embedding

- Get **embedding** from pretrained LLM (PLM)
- Use **Shannon Entropy**, normalized by **logistic sequence length** to estimate the amount of information in data
- Keep **high** information data

Metric 2: Domain Specific Score

- Get the input domain
- Count the **domain-related tokens**
- Keep the **most domain-related** input data

Metric 3: In-Domain Dissimilarity

- Within the same domain, we keep the **most distinct** data
- Reuse embeddings from PLM

	Domain	Example Lexicons
medical	Admin	dose vial inhale inject ml pills ingredient
	Anatomy	Pelvis arm sinus breast chest lymph tonsil
	Drug	ACOVA ACTONEL CARTIA EMGEL
emotion	Fear	bunker cartridge cautionary chasm cleave
	Surprise	amazingly hilarious lucky merriment
	Trust	advocate alliance canons cohesion
GloVe	GloVeTW26	extreme potential activity impact movement
	GloVeCC41	symptomatic thrombosis fibrillation
	GloVeTW75	nyquil benadryl midol pepto midol ritalin

Extendable Lexicons

Metric 1 $$\mathrm{EOE}(\vec{\mathbf{E}}_i) = \frac{-\sum_{e_i \in \vec{\mathbf{E}}} p(e_i) \log p(e_i)}{\log(n)}$$

Metric 2 $$\mathrm{DSS}(T, L) = \frac{1}{m} \sum_{i=1}^{m} \frac{|T \cap l_i|}{n}$$

Metric 3 $$\mathrm{IDD}(\vec{\mathbf{E}}, B) = \frac{1}{R} \sum_{i=1}^{R} (1 - \cos(\vec{\mathbf{E}}, \vec{\mathbf{E}}^{i}_{Dom_d}))$$

Data Selection Pipeline

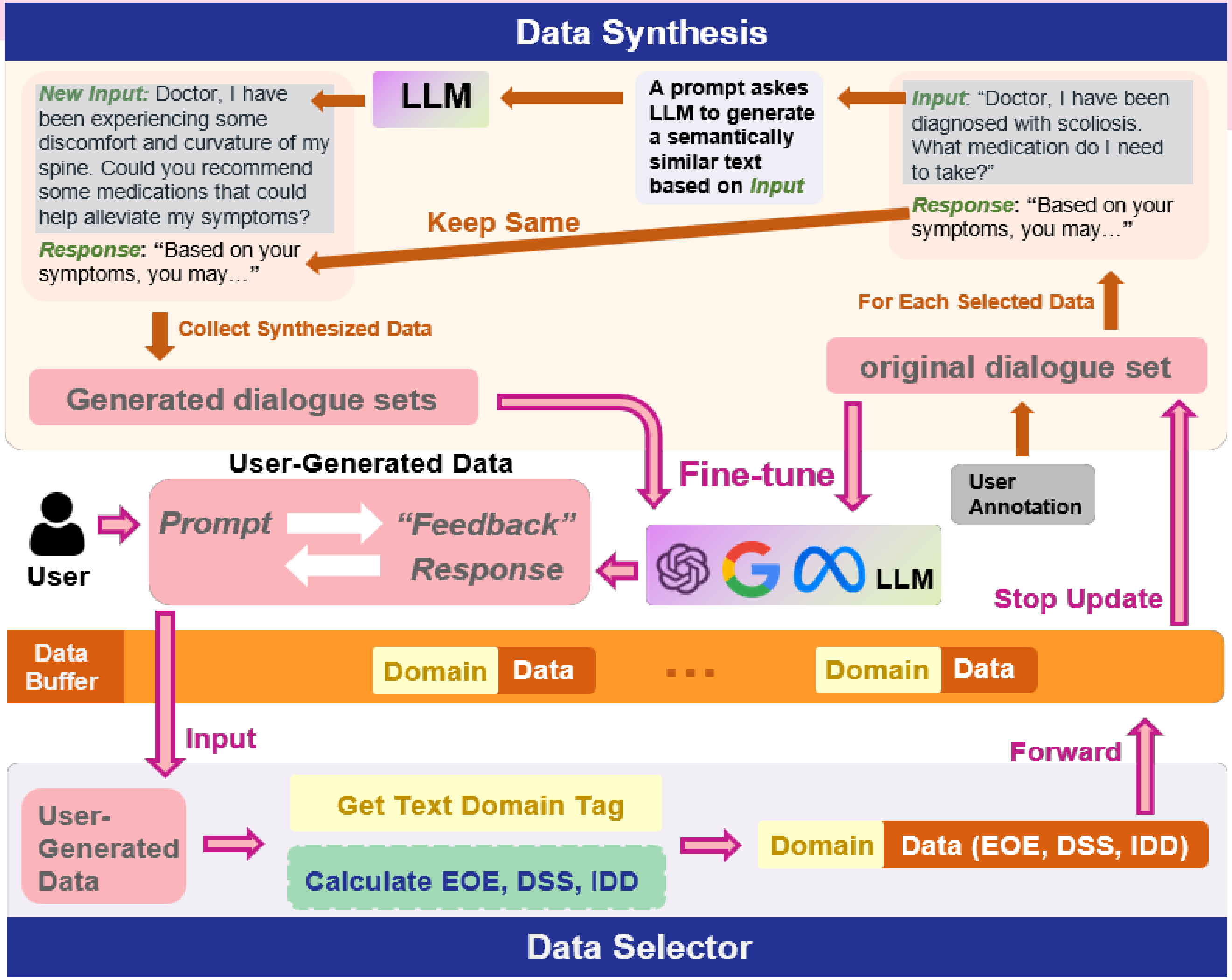
When buffer is not full

- For a new input, get its **embedding** and domain **tag**
- **Save** it on the buffer

When buffer is full

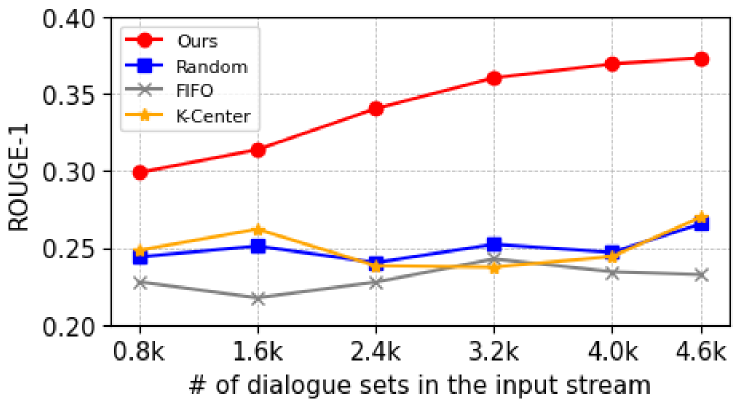
- Decide whether **discard** input data or **replace the data in the buffer**
- Calculate its **EOE**, if larger than the current min **EOE** on buffer
  - Calculate DSS given its domain tag
  - Within the domain, calculate IDD
- **Replace** with the one whose EOE, DSS, and IDD are all **smaller** than the current input data
- If the current one is **minimum, drop it**

Overview of Data Selection and Data Synthesis

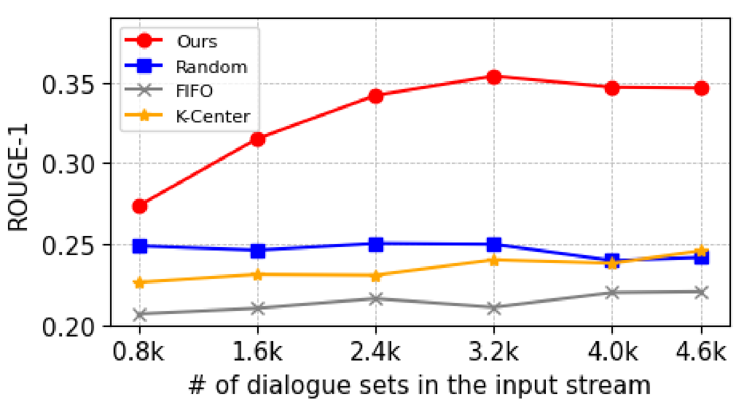


Performance and Conclusion

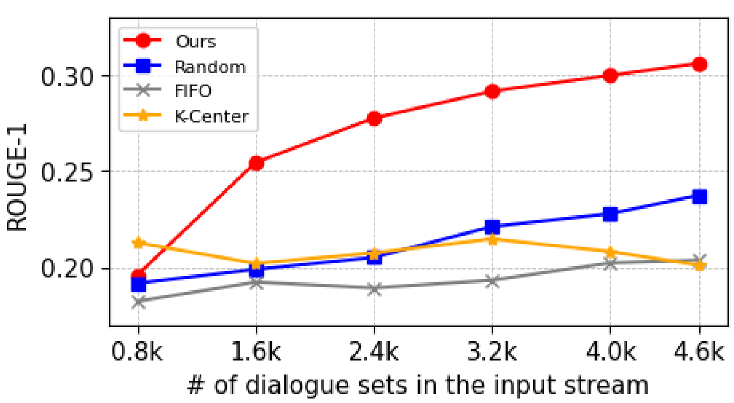
- Demonstrate decent performances on various datasets
- Highlight the potential of on-device data selection towards efficient LLM learning based on LORA



Performance on ALPACA dataset



Performance on DOLLY dataset



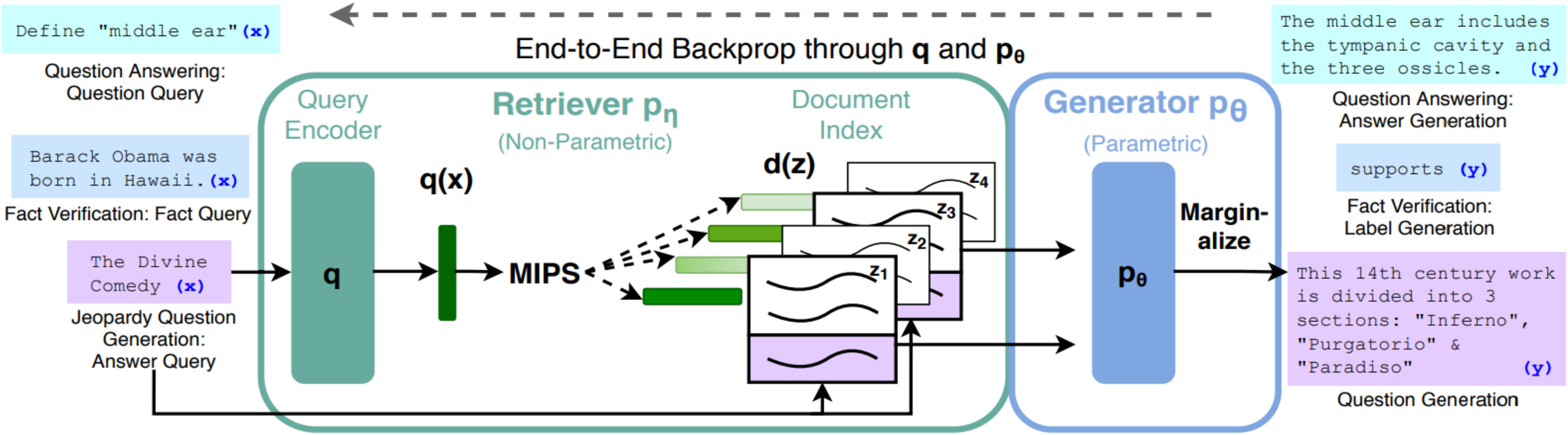
Performance on Prosocial dataset

Section 3:
RAG-CiM

THE CHIPS TO SYSTEMS CONFERENCE

Concept Recall: Retrieval-Augmented Generation (RAG)

- Mechanism:
  - Store user-related data (sentence embedding, in 1D vector)
  - Retrieve the data that mostly semantically relevant to user query (Retrieval algorithm like max inner product search - MIPS) – computationally expensive
  - Concatenate the retrieved data with query
- Rationale: Provide each query with more context information



RAG[1]

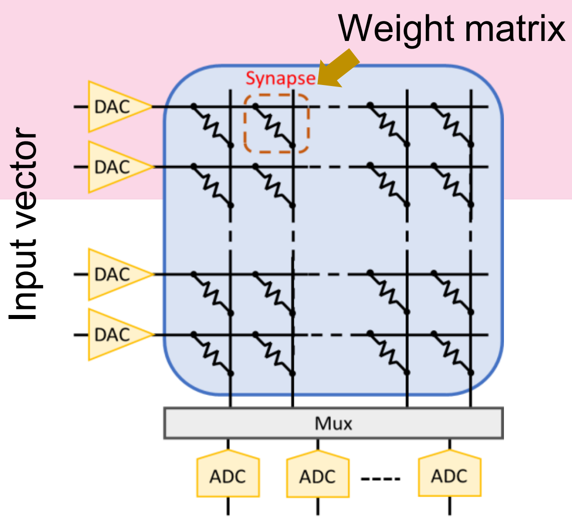
[1] Lewis, Patrick, et al. "Retrieval-augmented generation for knowledge-intensive nlp tasks." Advances in Neural Information Processing Systems 33 (2020): 9459-9474.

MIPS in RAG

- **What to store:**
  - One user-generated text → *Sentence Embedding Model* → One **stored vector**
  - Many user-generated texts → Vectors → "Can formalize a **matrix**"
- **Query**:
  - Original prompt → *Sentence Embedding Model* → One **query vector**
- **Find the appropriate user-generated text:**
  - **Inner products** between **query** vector and **every** store vector
  - **Rank** the stored vectors based on the **products**
  - Max Inner Product Search (**MIPS**)

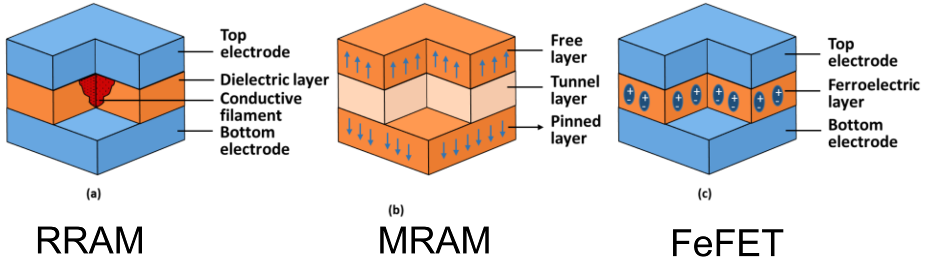
- **User-generated data:**
  - **Save all** the user-generated data → **Resource** on edge is **limited**
- **Manage data:**
  - Save on RAM: Easy for compute, but take up resources for other applications
  - Save on Disk: Data movement can lead to latency (Longer than LLM Inference)

Background: Nonvolatile Memory (NVM) and Computing-in-Memory (CiM)



The crossbar array architecture

- Vector-matrix multiplication in one clock cycle
- Concept:
  - Input Vector into each row [voltage]
  - Matrix stored in each cross point [conductance]
  - Output from each column [current]
- NVM



Emerging non-volatile devices

Name	# of Levels	Device Variations σ_v			
		L_0	L_1	L_2	L_3
$RRAM_1$ (Device-1)	1	0.0100	0.0100	0.0100	0.0100
$FeFET_2$ (Device-2)	4	0.0067	0.0135	0.0135	0.0067
$FeFET_3$ (Device-3)	4	0.0049	0.0146	0.0146	0.0049
$RRAM_4$ (Device-4)	4	0.0038	0.0151	0.0151	0.0038
$FeFET_6$ (Device-5)	4	0.0026	0.0155	0.0155	0.0026

Sample Device Variations (Noise) Pattern

Pros and Cons of NVCiM

- **MIPS: Vector-Matrix multiplication**
  - Vector: Query
  - Matrix: Many vectors from user-generated text
- **Maintained all user-generated text on NVM:**
  - Remove the latency due to data movement
  - In crossbar array: Preform MIPS robustly and efficiency (energy and time)

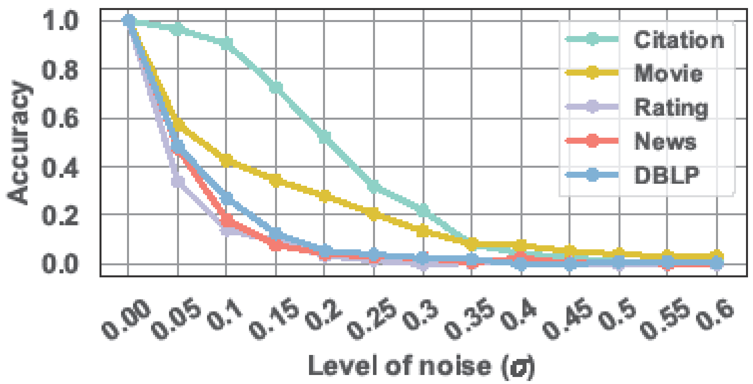


Figure: MIPS accuracy when device variation occurs during document embedding is written

Use NVCiM for RAG?

- Reduce the **retrieval latency** due to the growing user history data
- Bridge the gap between **NVCiM** and **RAG acceleration**

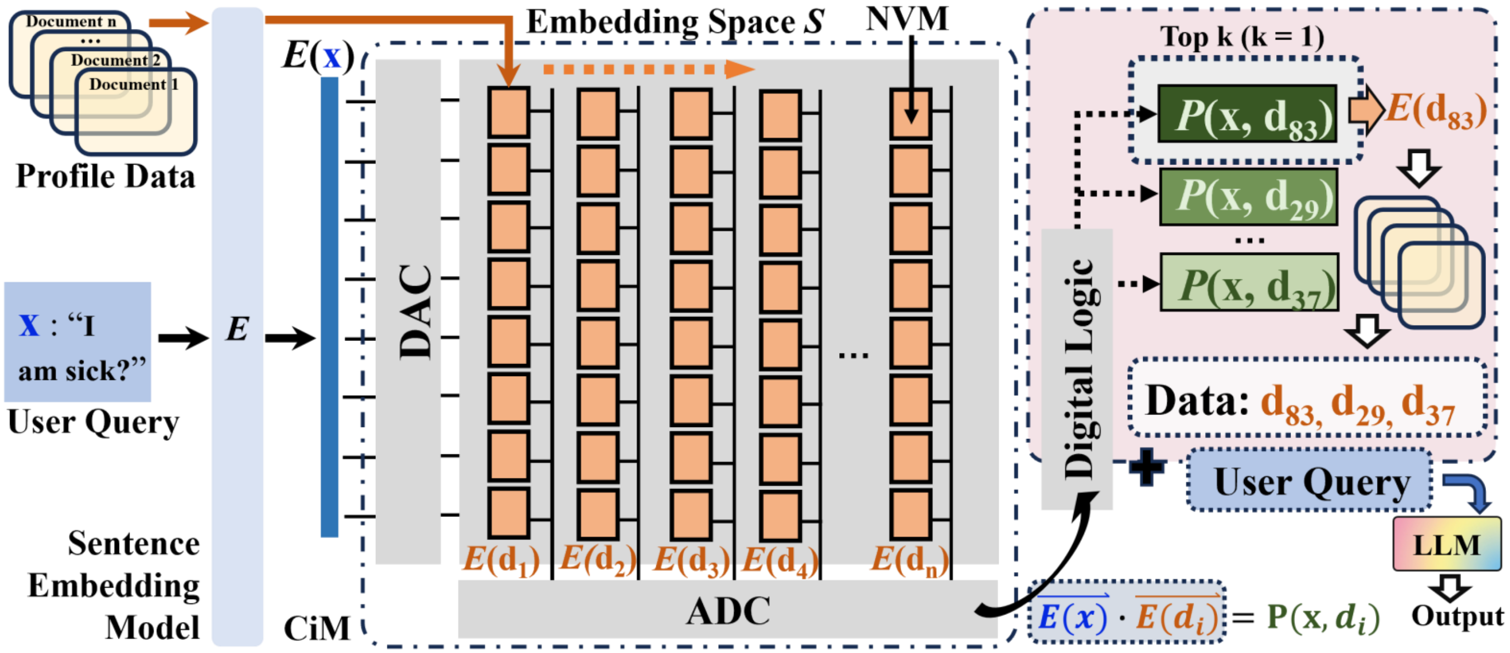
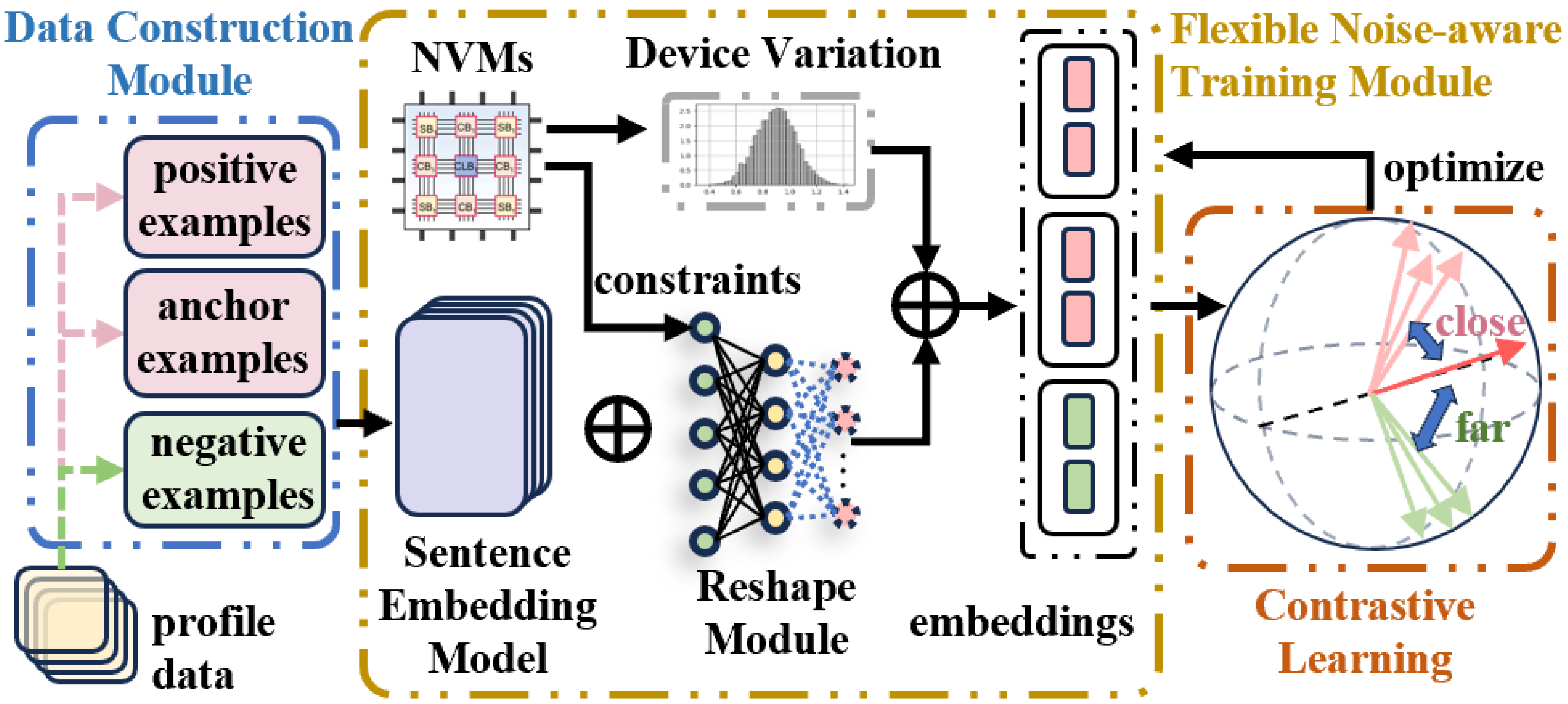


Figure: Implement RAG on NVCiM

Robust CiM-backed RAG (RoCR)

Data construction Noise-ware Training Contrastive Learning



RoCR: Contrastive Learning

- **Core**:
  - Push semantically **similar** vectors **closer**
  - Pull semantically **distinct** vectors **further**
- **Construct Data for contrastive learning:**
  - **Anchor**: The original input (prompt)
  - **Positive**: Semantically similar to the anchor
  - **Negative**: Semantically distinct from the anchor
- **Rationale**:
  - Contrastive learning is used to **train** embedding model that generate **noise-resilient** vectors

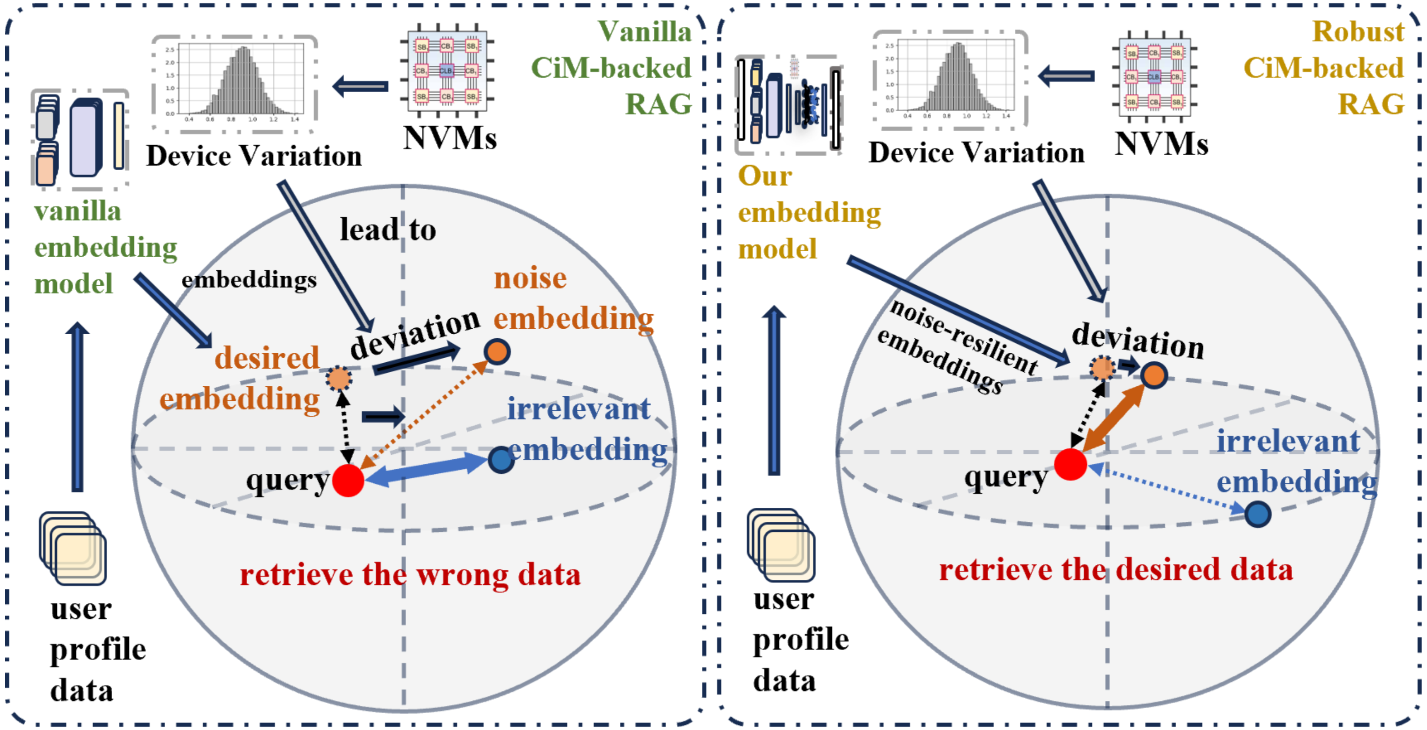
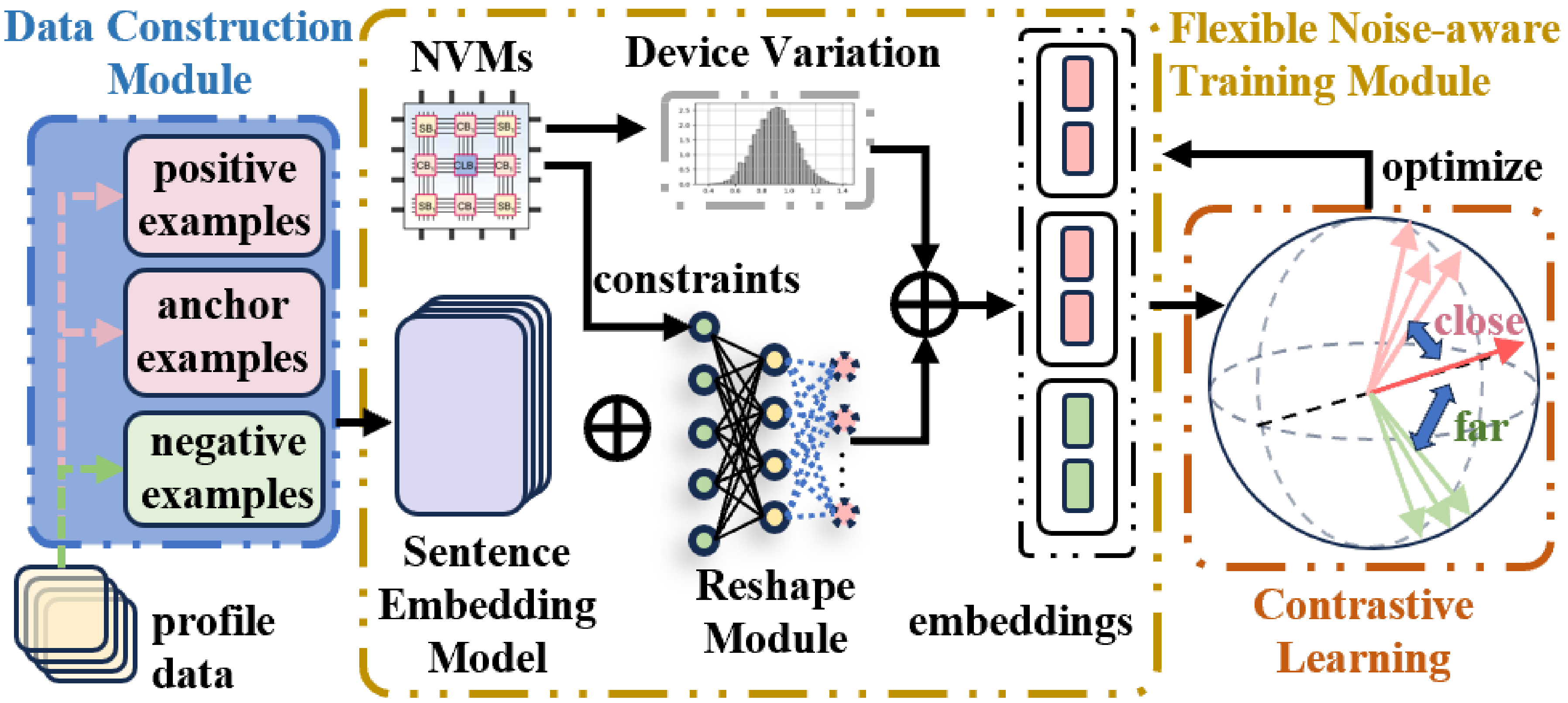


Figure: Improvement by RoCR

RoCR: Data Construction

Data construction | Noise-ware Training | Contrastive Learning



RoCR: Data construction

- **Use dropout rate (r) to generate:**
  - Large r: similar embeddings
  - Small r: distinct embeddings
- **When data has explicit labels (CDE):**
  - Anchor: emb(prompt + proper label)
  - Positive: emb(prompt + proper label, r = 0.1)
  - **Negative: emb(prompt + mismatching label)**
- **When data has no explicit labels(CDI)**
  - Anchor: emb(prompt)
  - Positive: emb(prompt, r = 0.1)
  - **Negative: emb(prompt, r = 1 – 0.1)**
- **Rationale:**
  - Our data construction methods work with the contrastive learning framework
  - Handling the cases when the user-generated data **with** or **without** have labels.

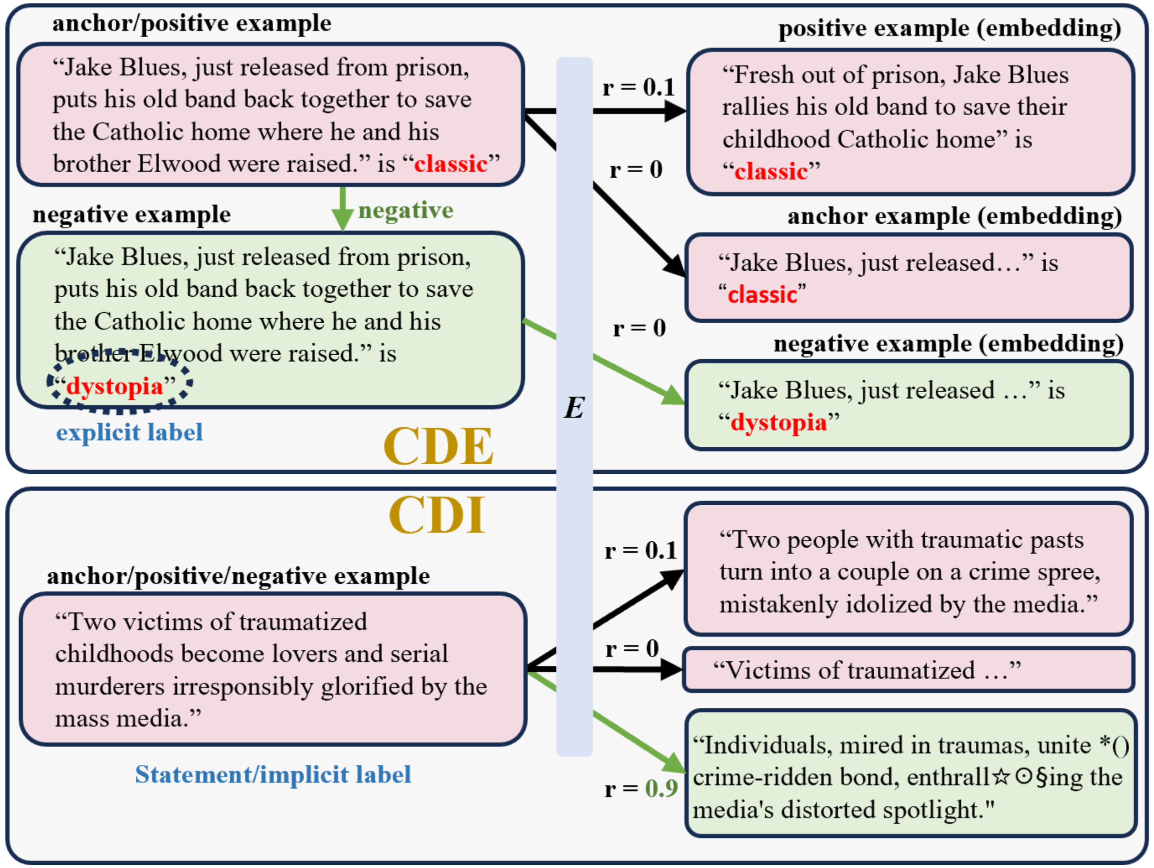
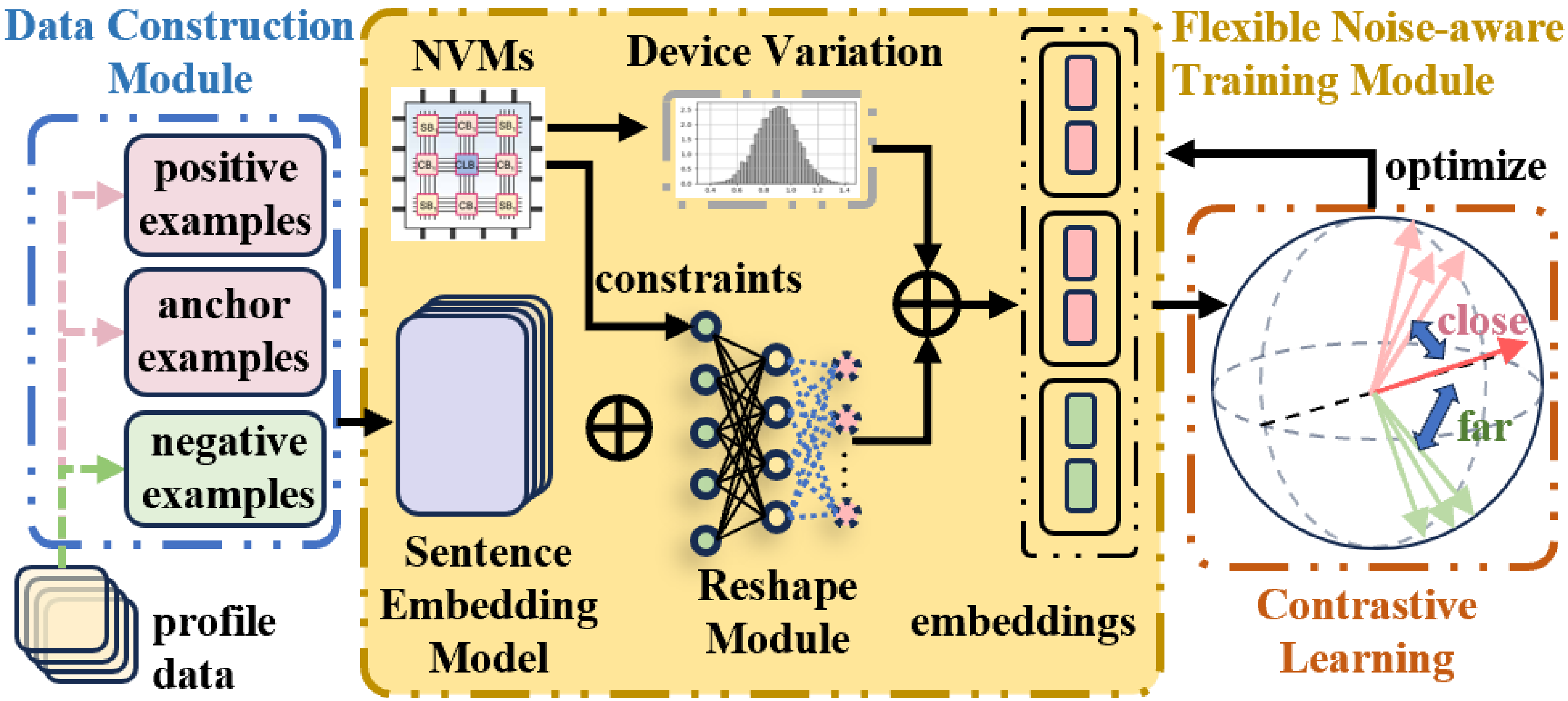


Figure: Examples of the two data construction methods

RoCR: Noise-ware Training

Data construction Noise-ware Training Contrastive Learning



RoCR: Noise-ware Training

- **Noise injection:**
  - [1, 0.75], [0.75, 0.5], [0.5, 0.25], [0.25, 0] (4 states in NVM), each range corresponding a variance level, shown as Figure 1
  - Concatenating with gaussian distribution (default to 0.1)
- **During training:**
  - Noise are added to embedding, shown as Figure 2
- **Rationale:**
  - When injected noise will **not** lead to **undesirable** LLM generating, we **stop** training

Name	# of Levels	Device Variations σ_v L_0	L_1	L_2	L_3
$RRAM_1$ (Device-1)	1	0.0100	0.0100	0.0100	0.0100
$FeFET_2$ (Device-2)	4	0.0067	0.0135	0.0135	0.0067
$FeFET_3$ (Device-3)	4	0.0049	0.0146	0.0146	0.0049
$RRAM_4$ (Device-4)	4	0.0038	0.0151	0.0151	0.0038
$FeFET_6$ (Device-5)	4	0.0026	0.0155	0.0155	0.0026

Figure 1: Device non-ideality modeling for different real and synthesized devices

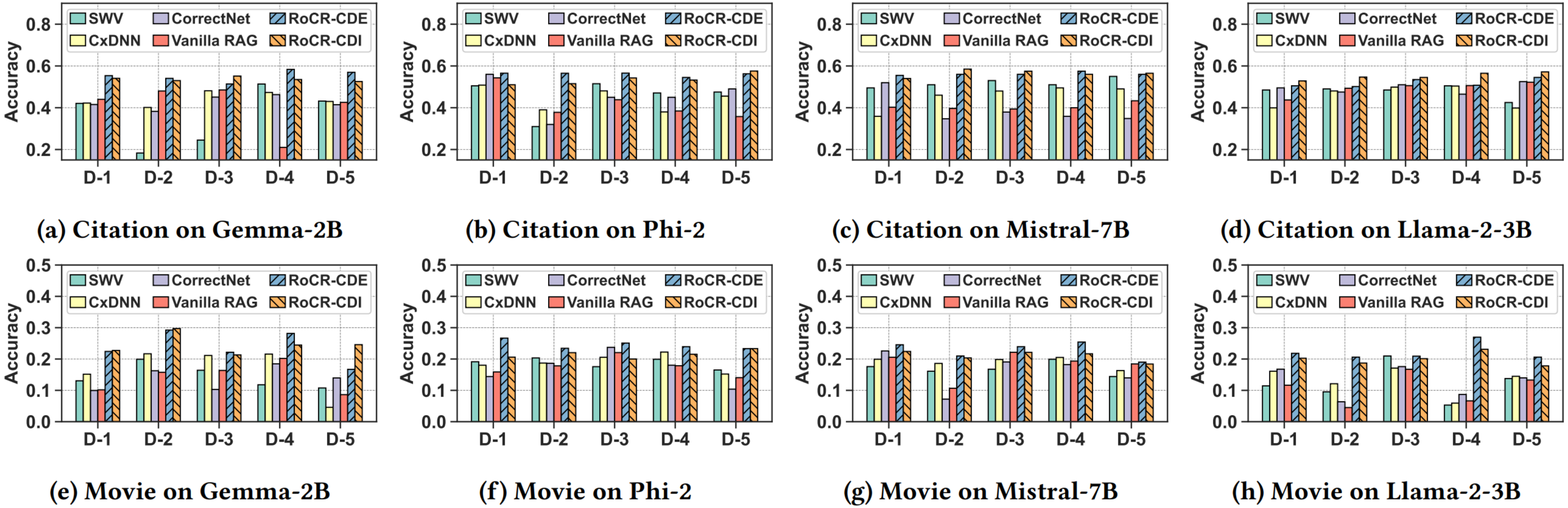
```python
def add_noise(embeddings, noise_factor_1, noise_factor_2, noise_factor_3, noise_factor_4, gaussian_noise_sigma):
    w_n = embeddings / embeddings.abs().max().item()
    w_n[w_n > 0.75] = w_n[w_n > 0.75] + torch.randn_like(w_n[w_n > 0.75]) * noise_factor_1 * gaussian_noise_sigma
    mask = (w_n <= 0.75) * (w_n >= 0.5)
    w_n[mask] = w_n[mask] + torch.randn_like(w_n[mask]) * noise_factor_2 * gaussian_noise_sigma
    mask = (w_n <= 0.5) * (w_n >= 0.25)
    w_n[mask] = w_n[mask] + torch.randn_like(w_n[mask]) * noise_factor_3 * gaussian_noise_sigma
    mask = (w_n <= 0.25)
    w_n[mask] = w_n[mask] + torch.randn_like(w_n[mask]) * noise_factor_4 * gaussian_noise_sigma

    return w_n
```

Figure 2: Noise injection

Performance and Conclusion

- After noise mitigations done by our work and other baselines, the processed data stored on NVM, will be used for RAG. Our work demonstrates decent RAG performance
- Highlight the potential of CiM architecture in optimizing LLM-related functions (like RAG)



Performance comparison between our framework and four baselines

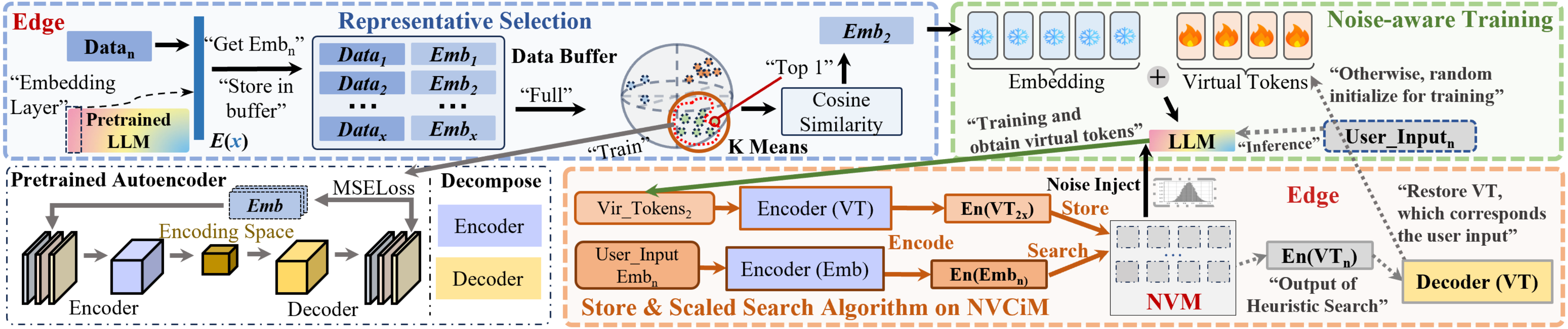
Section 4: NVCiM-PT

THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY

NVCiM-PT

- Scaled-search algorithm: Co-design circuit and algorithms
- Noise-aware Training: Mitigate the noise impact during NVM usage
- Representative Selection: Refine user data and formalize domain-specific data chunk



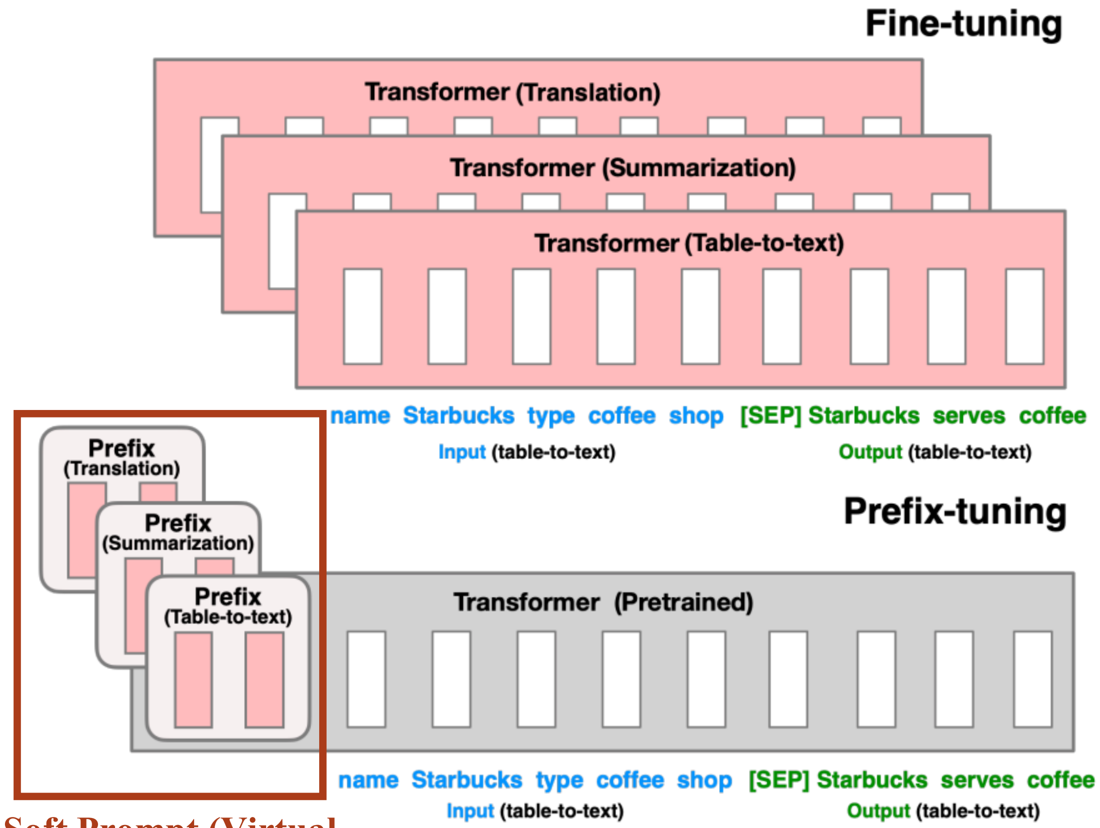
Background

Prefix-tuning VS Fine-tuning:

- Train only 0.1% parameters
- Saving resources
- Outperform fine-tuning in low-data settings

Challenges remain:

- Frequent domain shift
- Optimal sets of virtual tokens (OVT) → Specific Task
- Resource usage can be costly for edge



Soft Prompt (Virtual Tokens)

Background

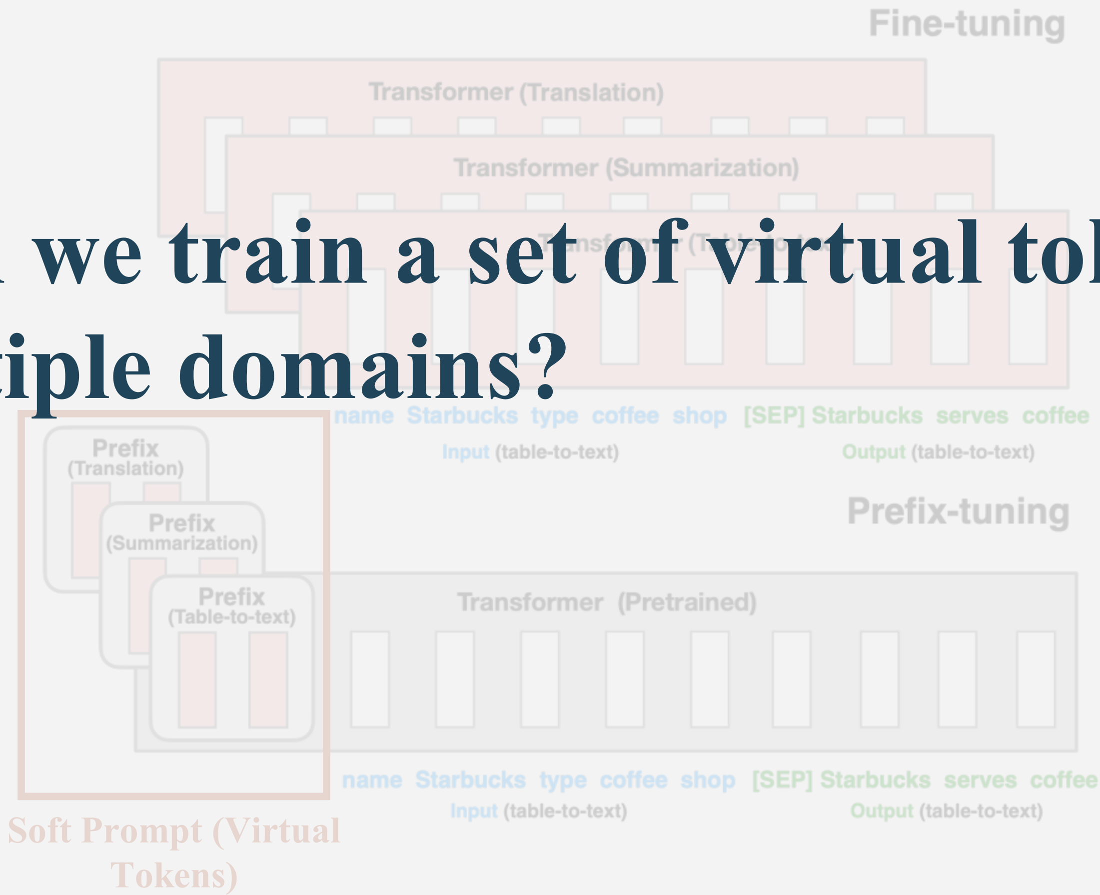
Prefix-tuning VS Fine-tuning:

- Train only 0.1% parameters
- Saving resources
- Outperform fine-tuning in low-data settings

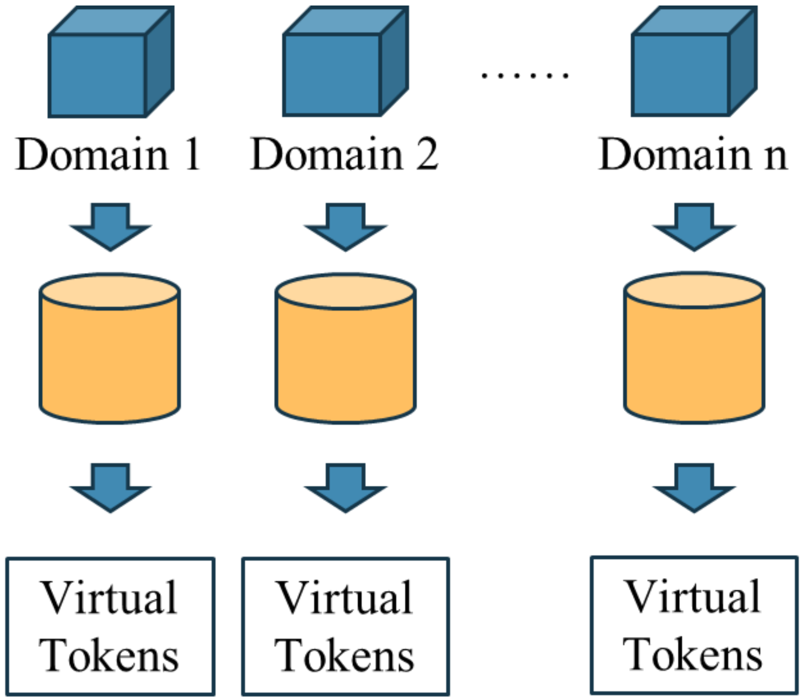
Challenges remain:

- Frequent domain shift
- Optimal sets of virtual tokens (OVT) → Specific Task
- Resource usage can be costly for edge

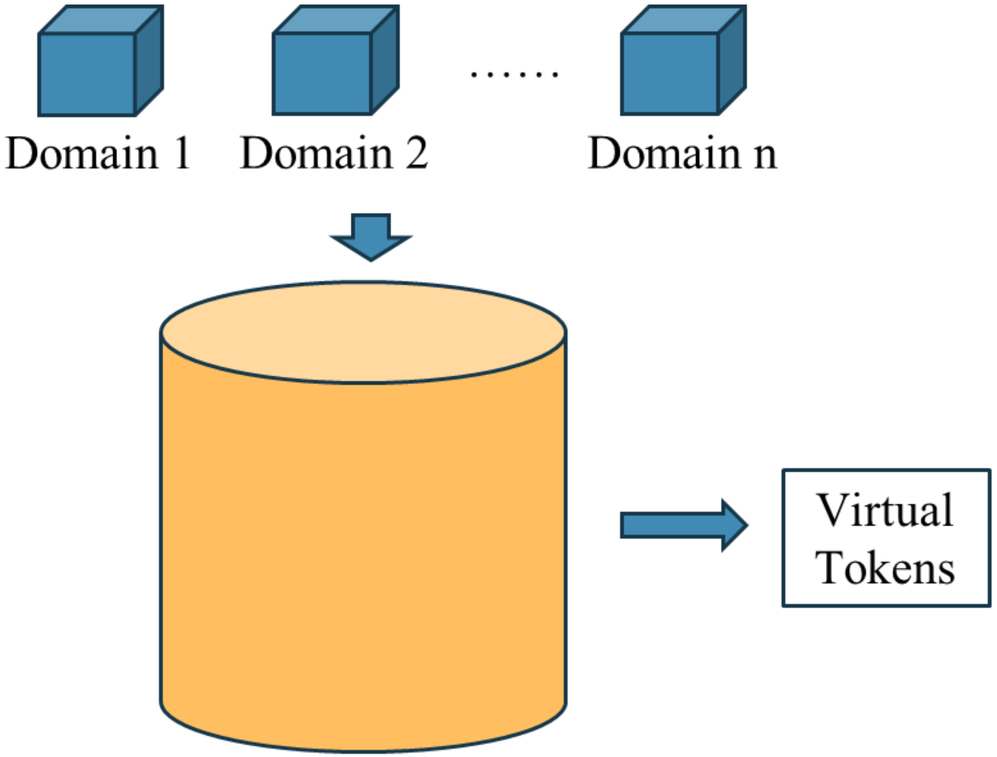
To overcome domain shift, can we train a set of virtual tokens to adapt multiple domains?



Motivation



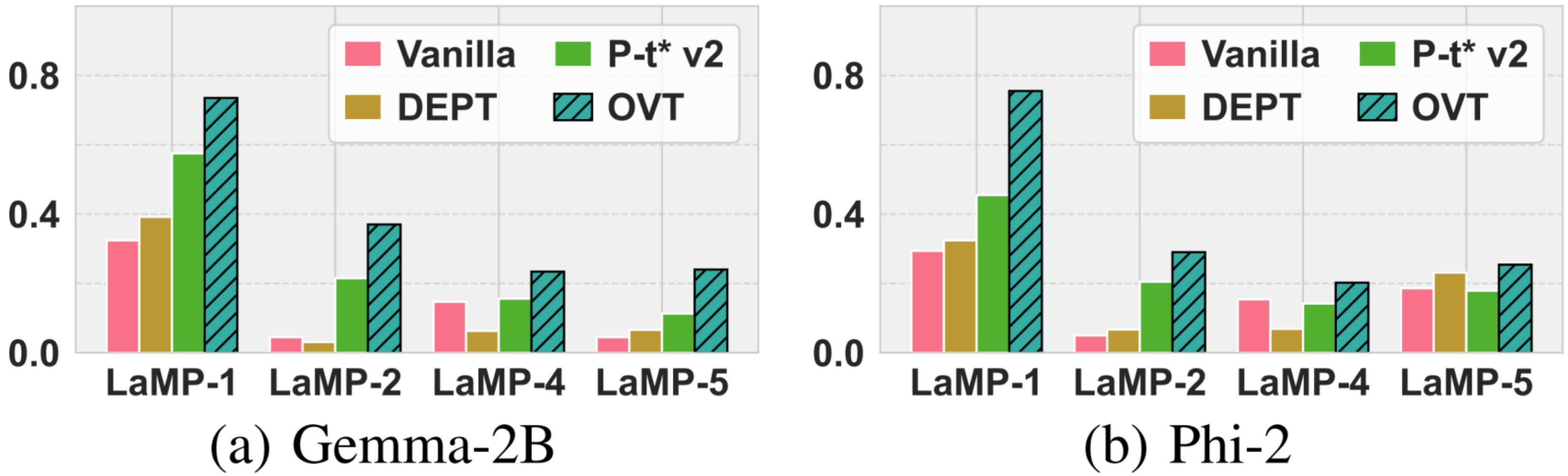
Smaller data volume,
faster training,
better performance



Accumulate a relatively large data volume

Impact of OVT Selection

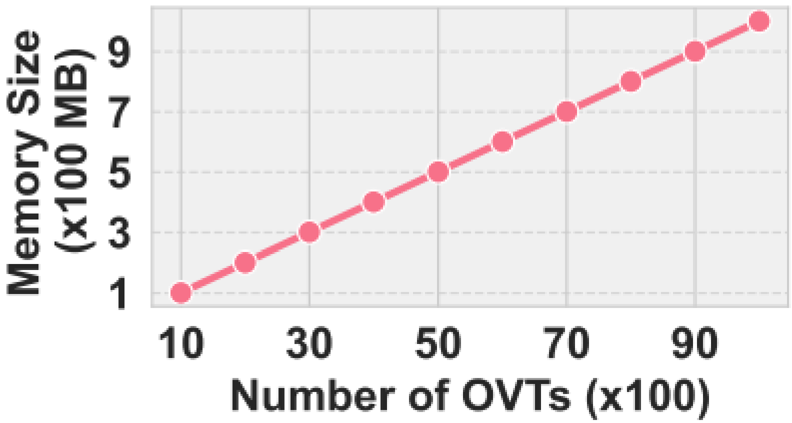
- When optimal virtual tokens (OVT) can be selected properly
- Compare the performance when every input can have its OVT and when all inputs have the same virtual tokens (Vanilla, P-t* v2, DEPT)



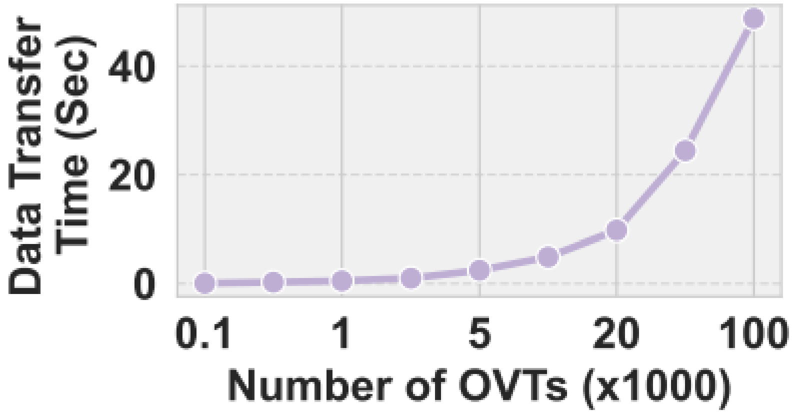
Challenges of Building OVT Bank

Instead of training OVT once domain shift, can we **store** and **retrieve** OVT to/from an OVT bank?

- Memory Consumption (a): RAM usage
- Latency (b): Data movement between disk and RAM if storing data on disk

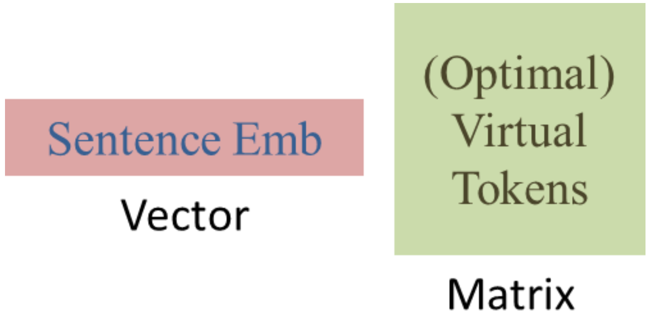


(a) Increasing memory usage



(b) Increasing data moving time

Sentence Embedding VS Virtual Tokens



Sentence Embedding

- Entire sentence is converted into a vector

Virtual Tokens :

- Consists of many tokens
- Each token is a vector

Retrieving sentence embedding

- Operation: vector * vector
- Rationale:
  - Semantic information is easy to interpret
  - Sentence embedding model converts textual input into sentence embedding

Finding optimal virtual tokens

- Sentence embedding model is not viable for virtual tokens
- Operation: "*Between matrix (input) and matrices (OVTs)*"
- Challenge:
  - Semantic information is hidden
  - *Simple matrix-matrix multiplication provides limited meaning*

OVT Bank based on NVCiM

"Simple matrix-matrix multiplication provides limited meaning"

Multi-scale (pooling):

- Scale = 1: Original token-level information
- Scale = 2: Medium across-token information
- Scale = 4: Long distance semantic information

Motivation:

- Why not just using scale 1: Only provides token-level information
- Seeking the multi-level vision
- Then synthesize (get average) of these "visions"

More Scale?

- Cost of chips (more scale → more resources are needed
- Balance and tradeoff:
    - Tri-level: Small-Medium-Large covers enough vision
    - More scales can lead to confusing information

$$P_1(\text{prompt}) = \begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 3 & 3 \\ 4 & 4 \end{bmatrix}$$

$$P_2(\text{prompt}) = \begin{bmatrix} 1.5 & 1.5 \\ 3.5 & 3.5 \end{bmatrix}$$

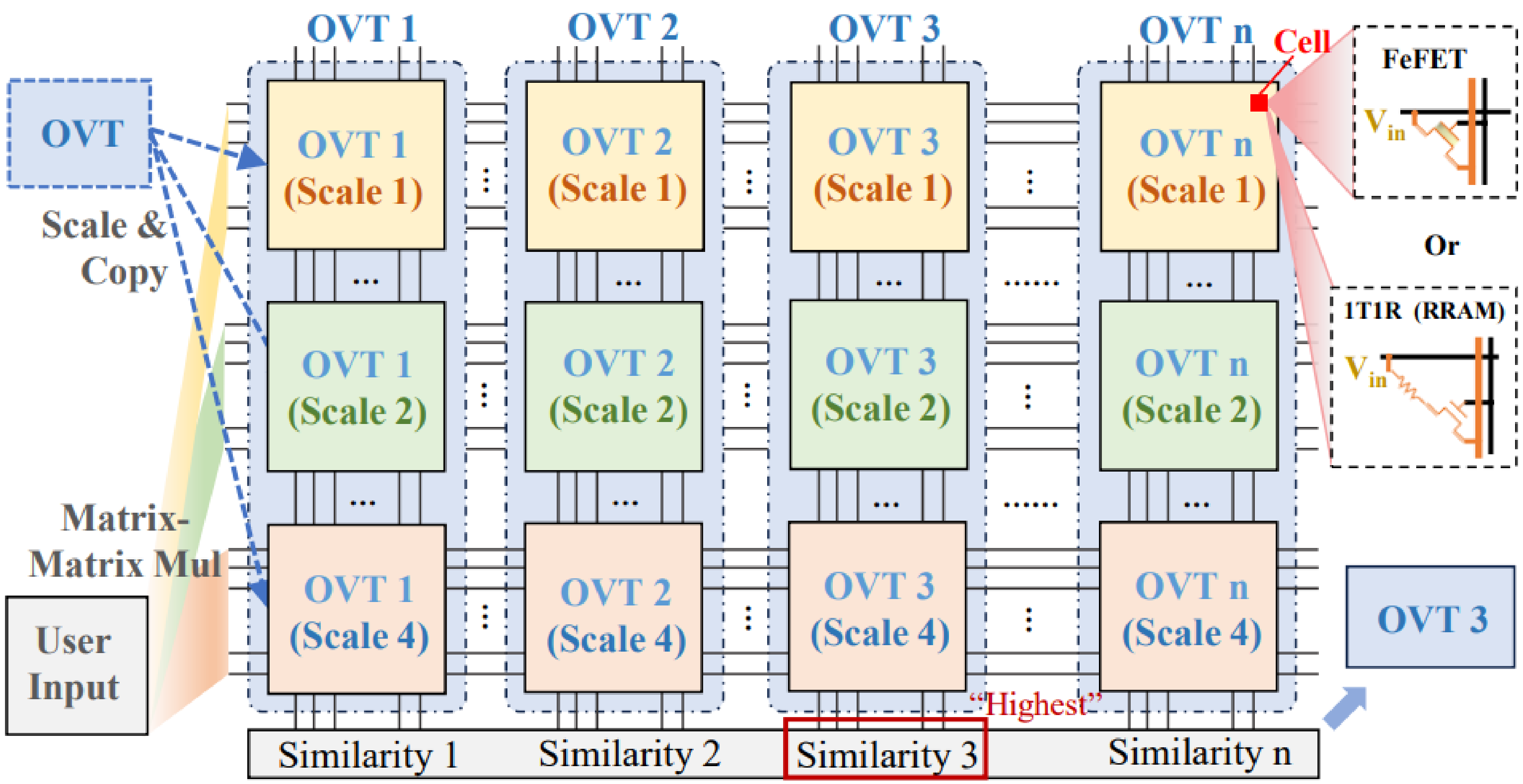
$$P_4(\text{prompt}) = \begin{bmatrix} 2.5 & 2.5 \end{bmatrix}$$

Example: Scale 1 2 4

Co-design NVCiM and Prefix Tuning (PT)

Core components to enable virtual tokens retrieval based on CiM:

- Retrieval algorithm: Adapter-level search (More complicated than MIPS)
- Circuit operation: matrix-matrix multiplication

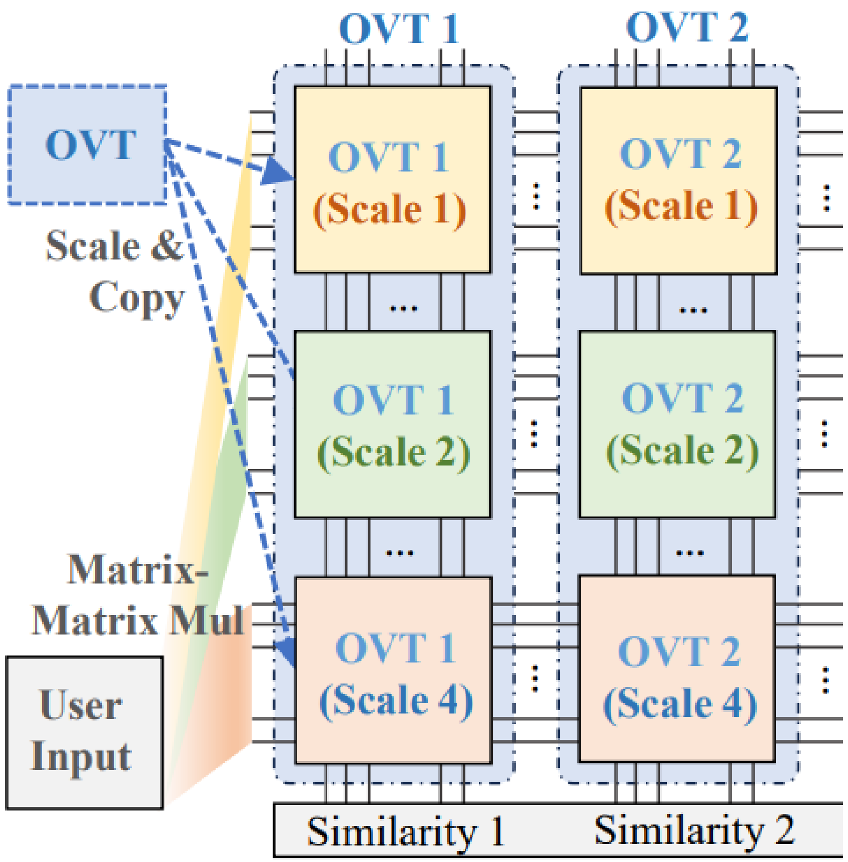


Co-design: Retrieval algorithm

- High-level Description: Adapter-level search (More complicated than MIPS)
- Concept:
  - Virtual Tokens (adapter), different from that in RAG, are integrated into a matrix.
  - Instead of vector (input) and matrix (stored data) multiplication, **matrix** (input) and **matrices** (stored adapters) **multiplications** are need
- Propose: Weighted Multi-Scale Dot Product search (WMSDP)
  - Scale: average pooling adapters
  - Weighted: on designed factors 1, 2, 4
  - Dot Product: Between input matrix and every stored matrix
- Rationale: Information stored in adapter is **more hidden**, compared to sentence embedding data in RAG

Co-design: Circuit operation

- High-level description： matrix-matrix multiplication
- Input: Entire matrix instead of single vector. Each row is a set of voltages
- Storage: Each OVT is copied into three scales 1, 2, 4
- Output: Sum and average, the similarity score is a single value, for ranking

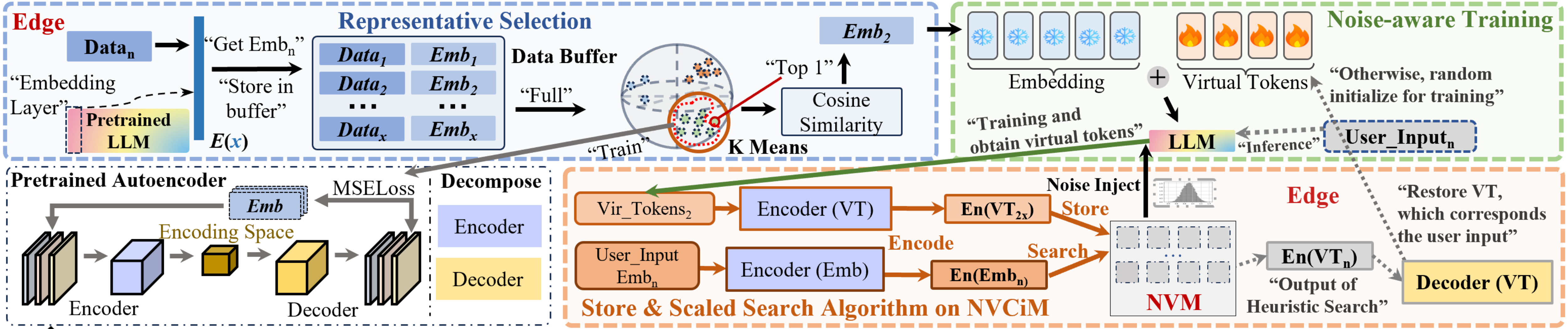


NVCiM-PT Framework

- Representative Selection:
  - Echo back "***Data Selection***"
- Noise-aware Training:
  - Echo back "***RAG-CiM (RoCR)***"
- ***Scaled Search Algorithm with Co-design***

Noise-aware Training:

- During generating (prefix tuning) the OVT
- Adding noise to virtual tokens
- Use default CE loss



Performance and Conclusion

- Demonstrate decent performance on various datasets, multiple LLMs and different NVM devices
- CiM architecture has potential to optimize LLM-related functions (RAG, prefix tuning)
  - *Maybe we can do more in the future!*

LLM		Gemma-2B					Mistral-7B-GPTQ					Phi-2				
Device	**Dateset** / **Method**	**LaMP-1** Acc	**LaMP-2** Acc	**LaMP-3** Acc	**LaMP-5** Rouge-1	**LaMP-7** Rouge-1	**LaMP-1** Acc	**LaMP-2** Acc	**LaMP-3** Acc	**LaMP-5** Rouge-1	**LaMP-7** Rouge-1	**LaMP-1** Acc	**LaMP-2** Acc	**LaMP-3** Acc	**LaMP-5** Rouge-1	**LaMP-7** Rouge-1
NVM-1	SWV	0.347	0.179	0.667	0.081	0.080	0.417	0.026	0.550	0.091	0.112	0.605	0.079	0.635	0.098	0.123
	CxDNN	0.392	0.230	0.706	0.067	0.107	0.477	0.083	0.466	0.065	0.160	0.661	0.154	0.671	0.136	0.209
	CorrectNet	0.447	0.199	0.705	0.185	0.107	0.506	0.191	0.438	0.049	0.076	0.696	0.179	0.634	0.045	0.187
	No-Miti(MIPS)	0.318	0.153	0.634	0.128	0.081	0.421	0.011	0.466	0.03	0.011	0.593	0.148	0.627	0.052	0.148
	NVP*(MIPS)	0.385	0.155	0.668	0.020	0.051	0.393	0.111	0.251	0.053	0.118	0.409	0.104	0.354	0.118	0.216
	NVCiM-PT	**0.549**	**0.250**	**0.732**	**0.199**	**0.139**	**0.529**	**0.205**	**0.559**	**0.166**	**0.209**	**0.707**	**0.250**	**0.725**	**0.201**	**0.257**
NVM-2	SWV	0.360	0.119	0.702	0.107	0.044	0.341	0.038	0.554	0.105	0.160	0.691	0.083	0.681	0.085	0.049
	CxDNN	0.359	0.305	0.755	0.151	0.124	0.487	0.063	0.452	0.117	0.101	0.667	0.227	0.687	0.085	0.255
	CorrectNet	0.469	0.209	0.685	0.226	0.147	0.440	0.107	0.507	0.147	0.093	0.672	0.207	0.643	0.138	0.128
	No-Miti(MIPS)	0.397	0.183	0.642	0.112	0.057	0.382	0.048	0.442	0.015	0.049	0.571	0.113	0.61	0.101	0.159
	NVP*(MIPS)	0.469	0.136	0.537	0.052	0.044	0.506	0.069	0.256	0.023	0.129	0.455	0.029	0.395	0.066	0.244
	NVCiM-PT	**0.510**	**0.300**	**0.763**	**0.158**	**0.170**	**0.529**	**0.205**	**0.569**	**0.166**	**0.207**	**0.718**	**0.265**	**0.716**	**0.203**	**0.266**

Performance comparison between our framework with existing noise mitigation methods

Section 5: Tiny-Align

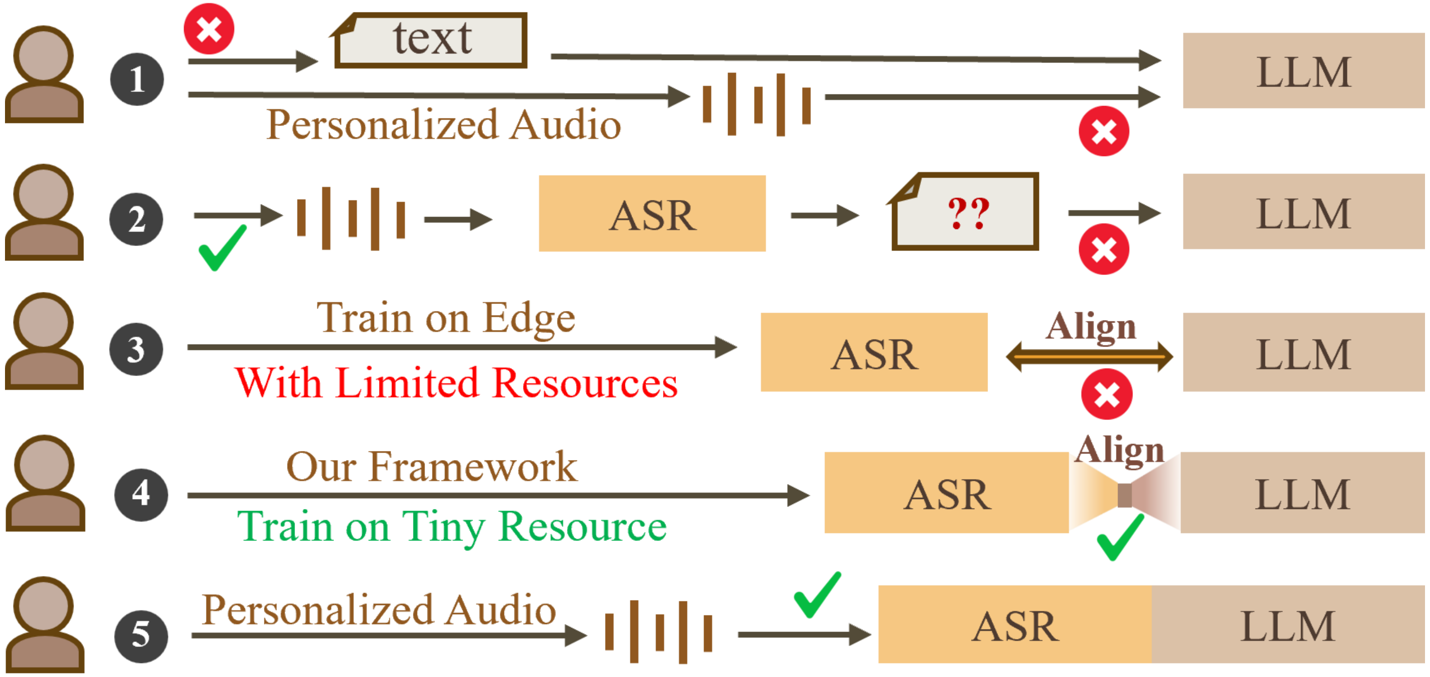
THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY

Cross-modal Alignment: Tiny-Align

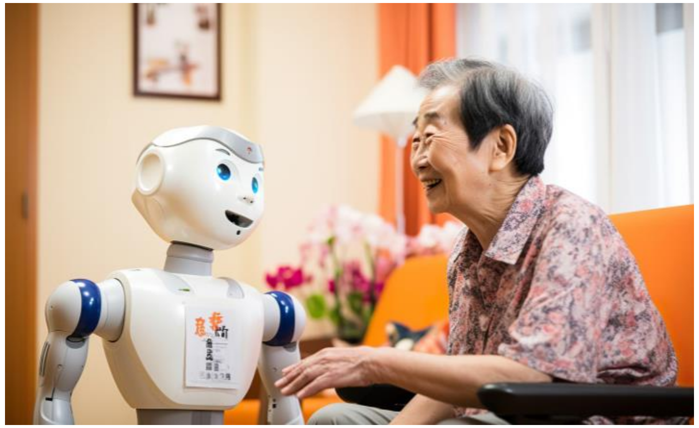
Interaction beyond text:

- Personalization (speech pattern/behavior)
- Benefit users with typing difficulties
- Align audio with text-based LLM is challenging



Cross-modal Alignment → ASR-LLM:

- ASR- Automatic Speech Recognition models
- Applications: People with dementia/aphasia/SLI
  - What's special: difficulties with typing, highly personalized interaction, privacy



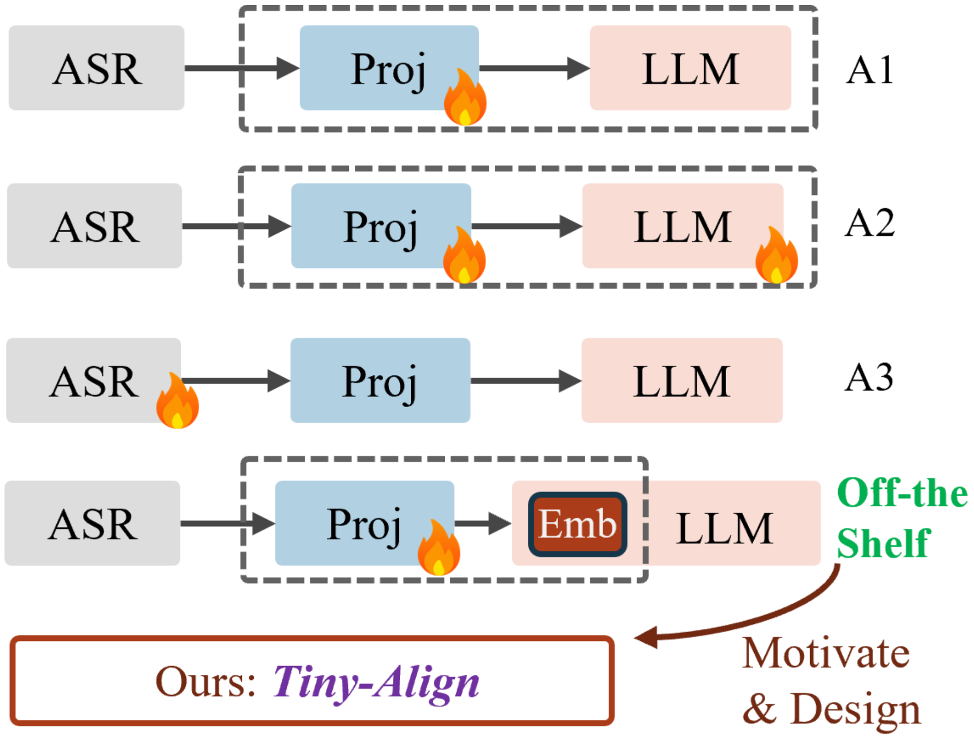
Dementia

Aphasia

Specific Language Impairments (SLI)

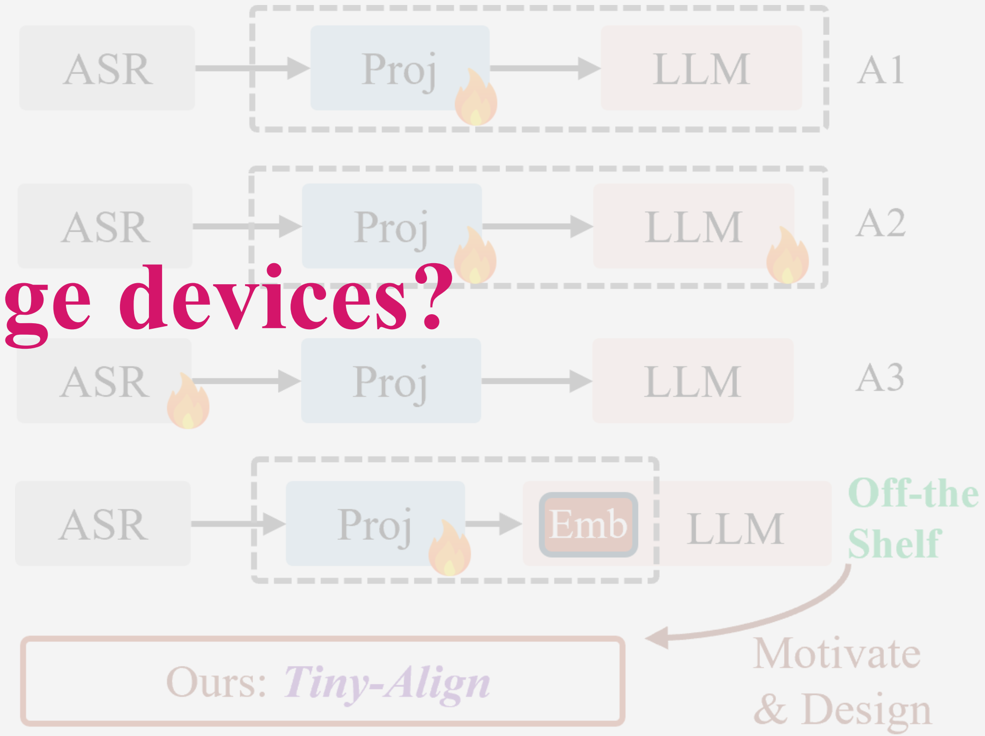
Existing Approaches of Cross-modal Alignment

- Heads-up: **Projectors** are used to map ASR features into LLM
- Approach 1: Train the projector based on LLM inference
- Approach 2: Train the projector and the LLM at the same time
- Approach 3: Train the ASR and keep LLM unchanged



Existing Approaches of Cross-modal Alignment

- Heads-up: **Projectors** are used to map ASR features into LLM
- Approach 1: Train the projector based on LLM inference
- Approach 2: Train the projector and the LLM at the same time
- Approach 3: Train the ASR and keep LLM unchanged



Do they work on edge devices?

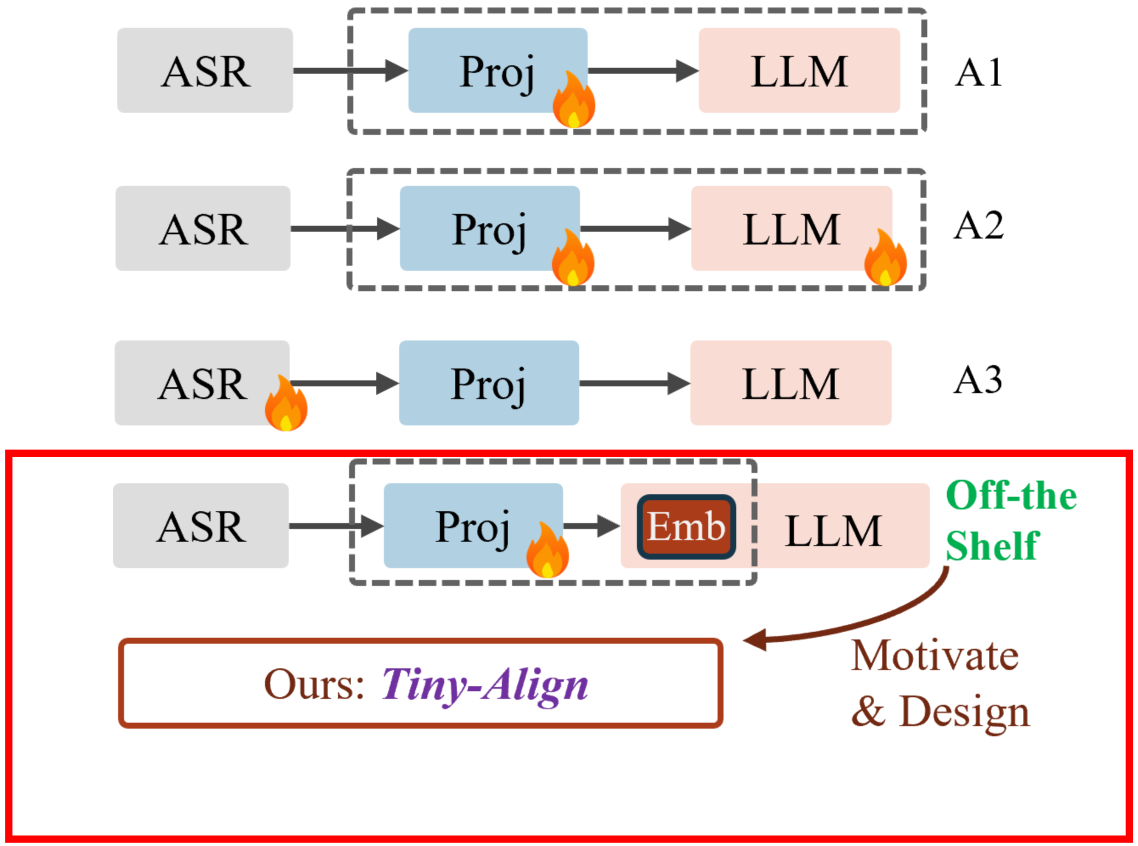
Motivation

Problems in existing approaches:

- Without ASR and LLM alignment, performance may degrade
- Given small data volume (1k~5k samples), end-to-end alignment may be unnecessary and burdensome

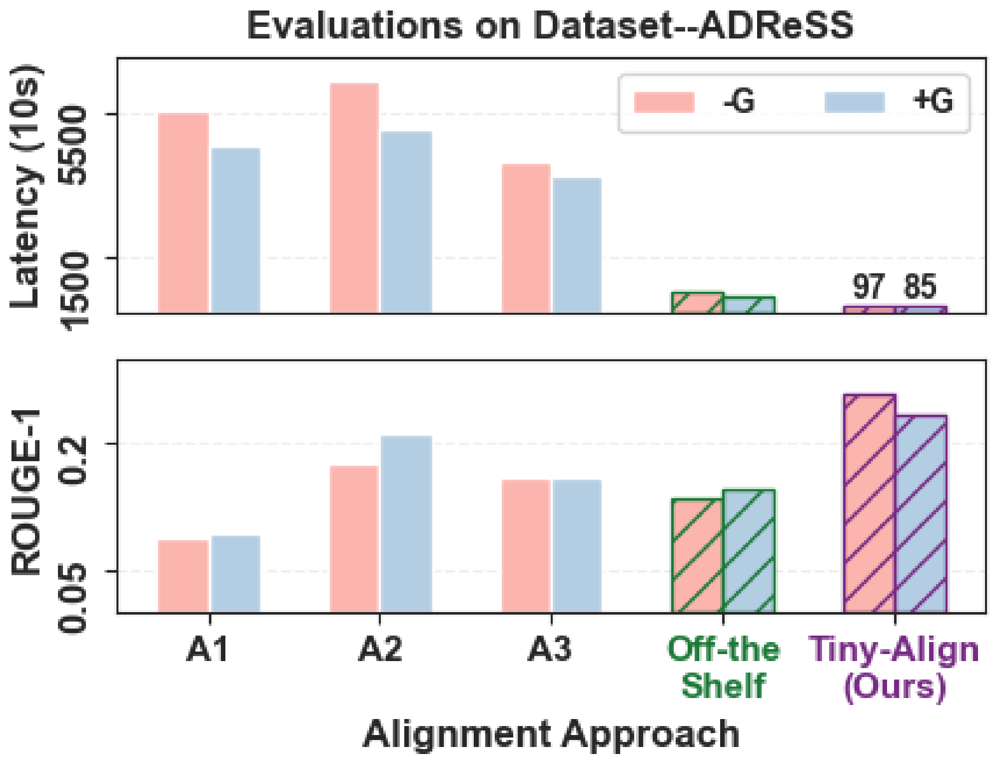
Core of the edge solution:

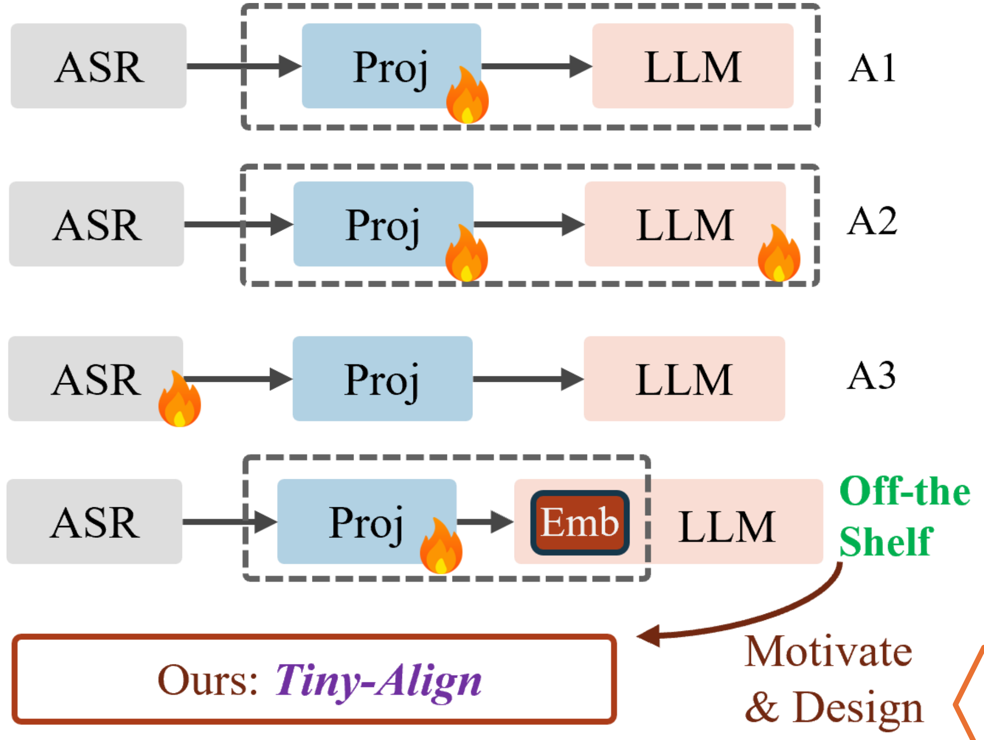
- Train only the projector with fast feedback
- Map ASR features into LLM recognizable content (input embedding)



Preliminary Evaluations

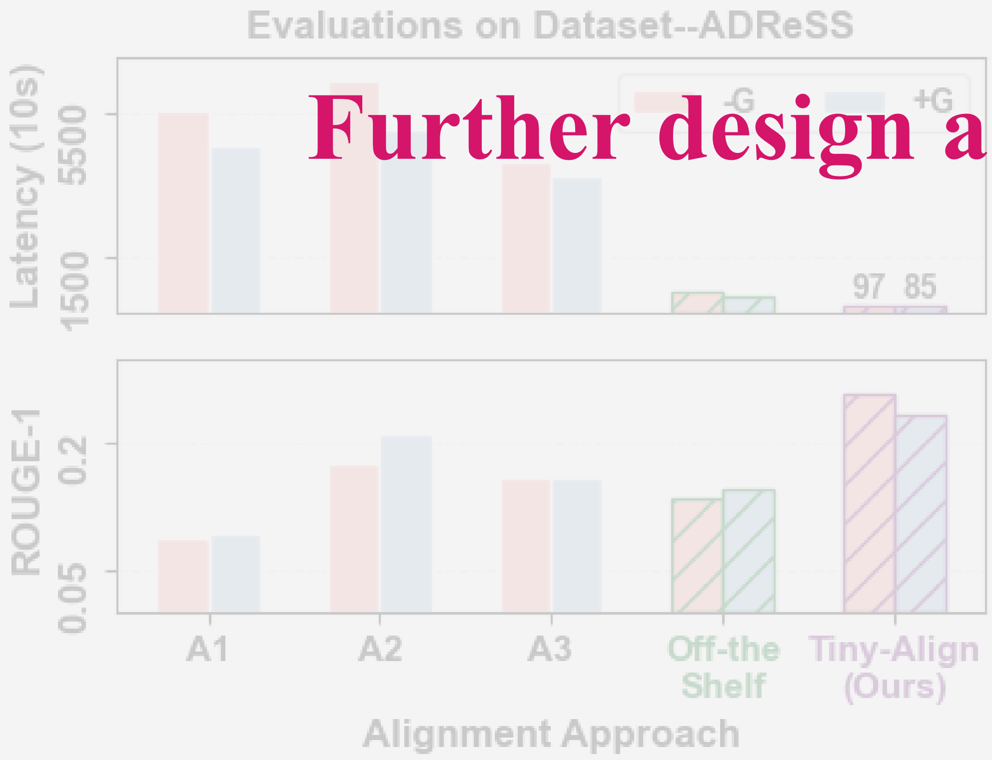
- Compress LLM does not bring significant benefit
- Projector only (off-the-shelf) method outperforms in performance and efficiency
- Using off-the-shelf projectors do not work as well as optimized ones (Tiny-Align)

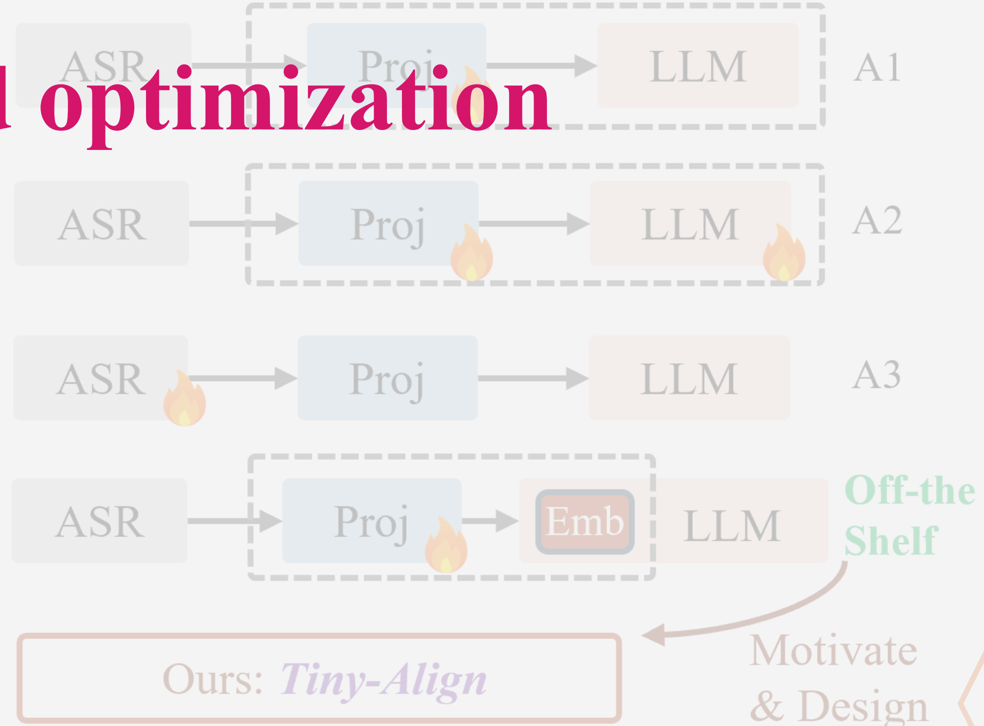




Preliminary Evaluations

- Compress LLM does not bring significant benefit
- Projector only (off-the-shelf) method outperforms in performance and efficiency
- Using off-the-shelf projectors do not work as well as optimized ones (Tiny-Align)



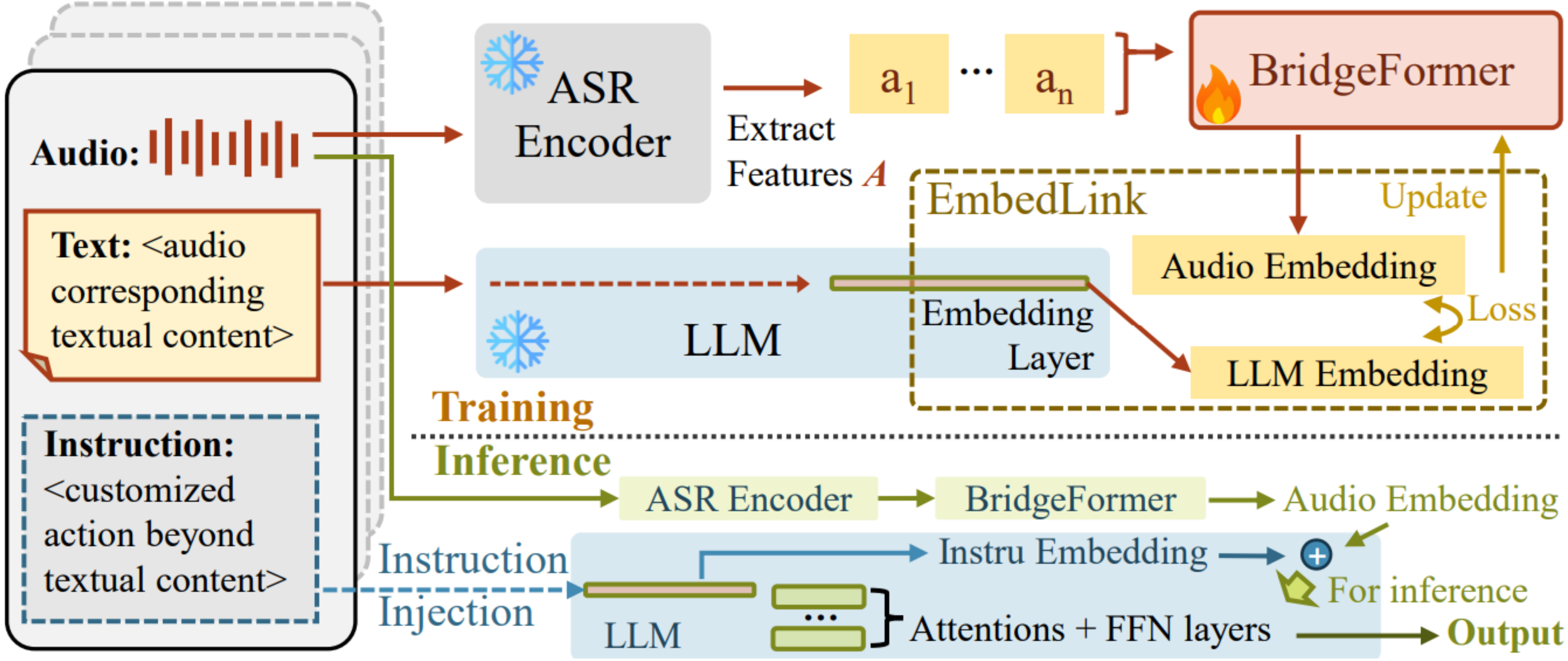


Further design and optimization

Motivate & Design

Cross-modal Alignment: Tiny-Align

- BridgeFormer
- EmbedlLink
- Feature-based ASR
- Instruction Injection



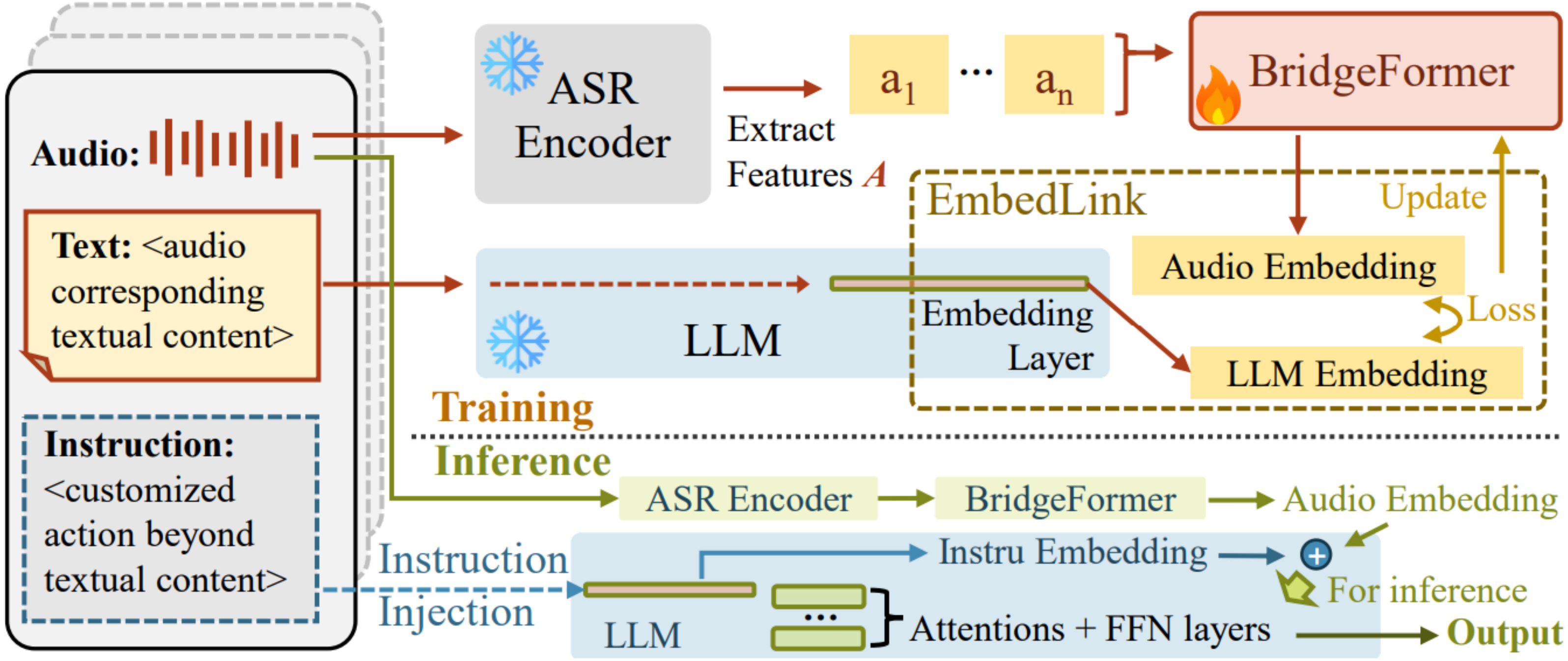
Cross-modal Alignment: Tiny-Align

Training:

- Input: Audio + Corresponding Text
- Output: Textual Response

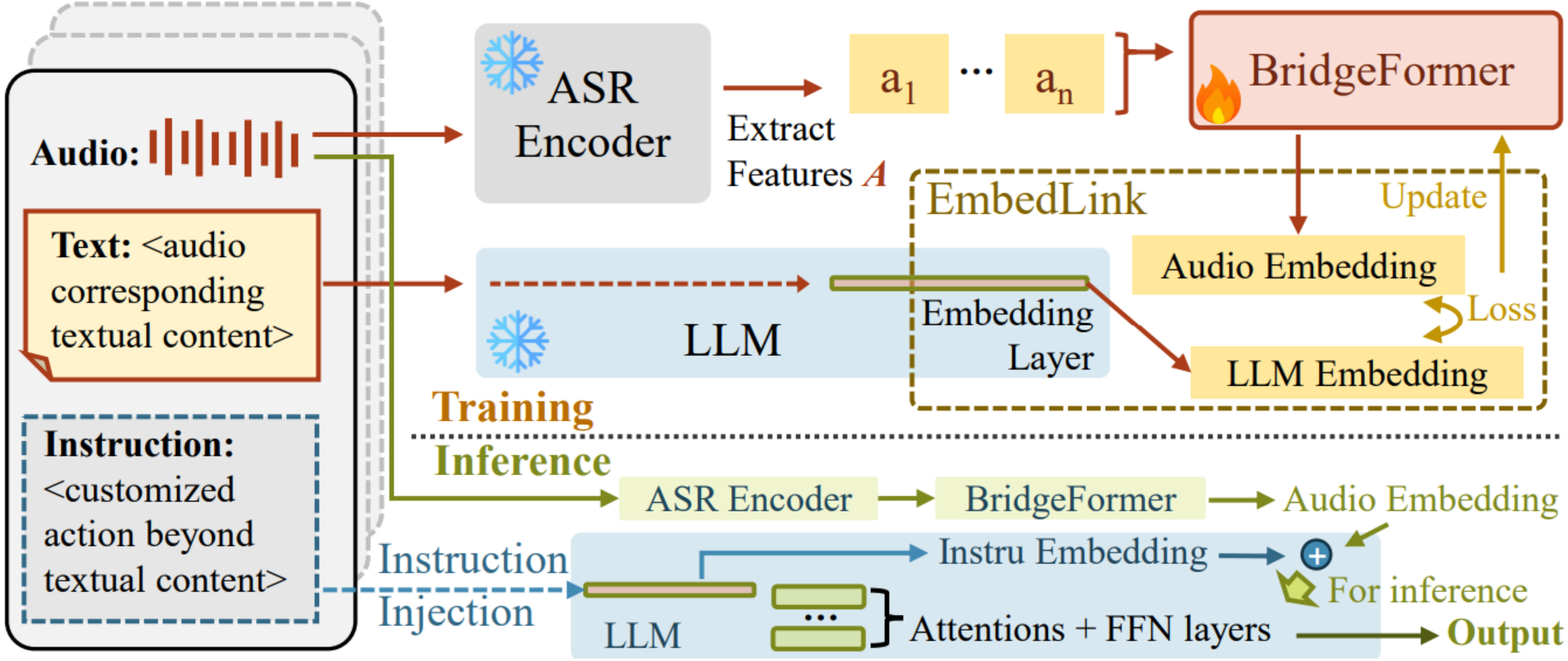
Inference:

- Input: Audio + [Instruction]
- Output: Textual Response



Cross-modal Alignment: Tiny-Align

- **BridgeFormer**
- EmbedlLink
- Feature-based ASR
- Instruction Injection



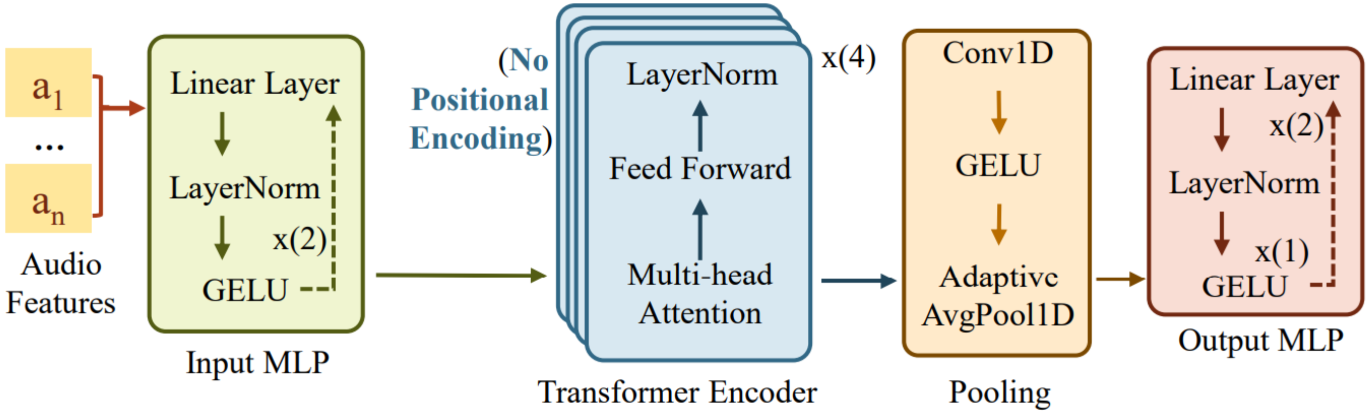
Cross-modal Alignment: Tiny-Align

Normal Projectors in Cross-Modal Alignment:

- Simple MLP
- Limited representational space

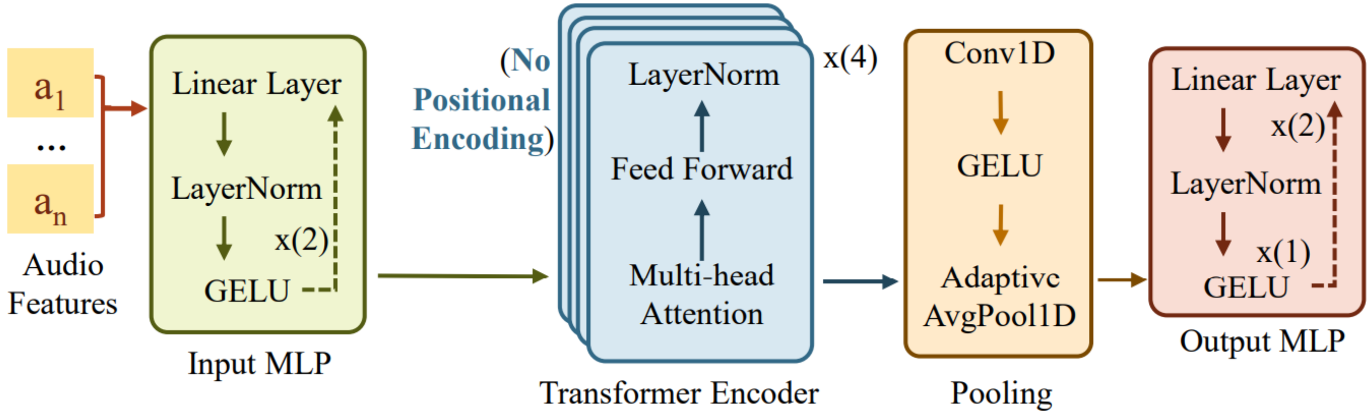
Ours (BridgeFormer):

- Add multi-head attentions, remove positional encoding
- Use MLP to reshape the input and output
- Rich representational space



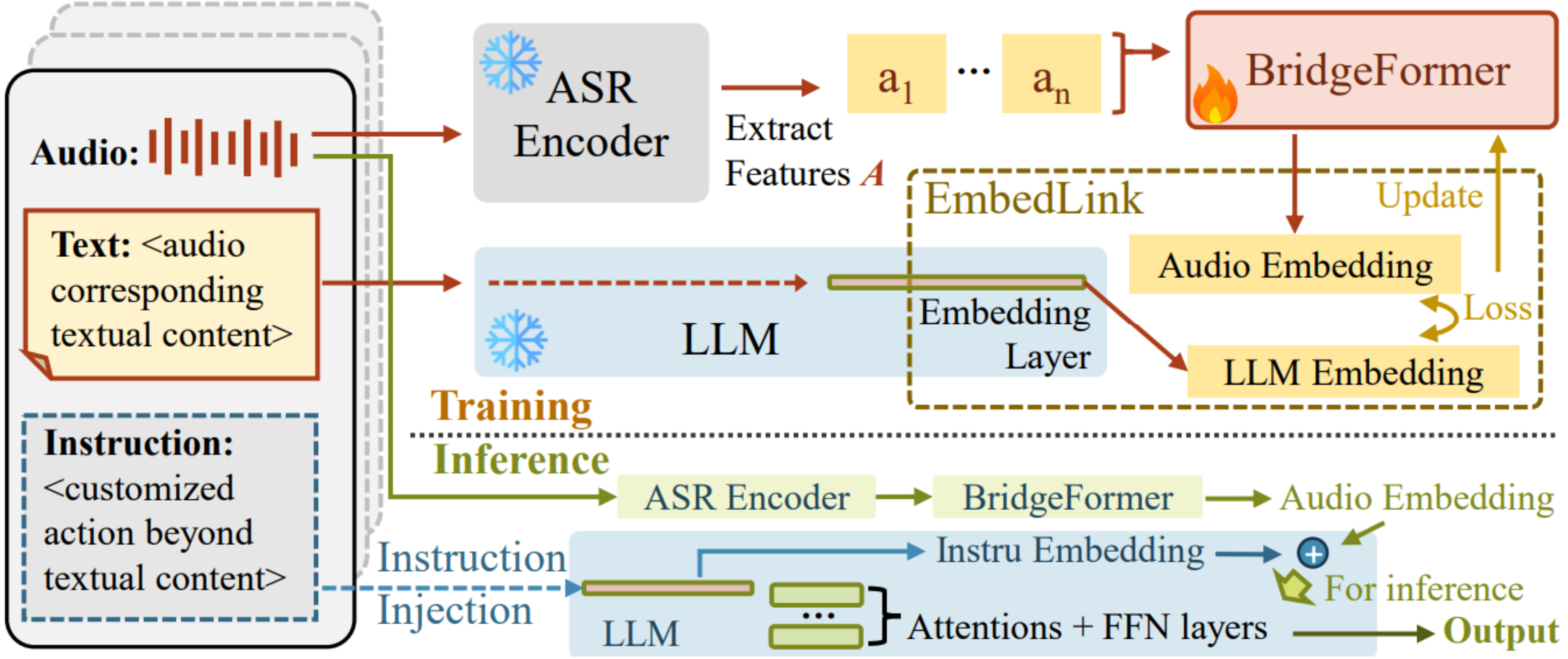
Benefits of transformer-based projector (BridgeFormer):

- **Capability**: Better capture hidden semantic information
- **Architecture**: Correspond to the attention mechanisms in LLM (and possibly ASR, depending on ASR choice)
- **Rationale**: Flexible size, robust performance, better scalability (increase/decrease attention head)
- **Budget**: Tolerable increase in training workload



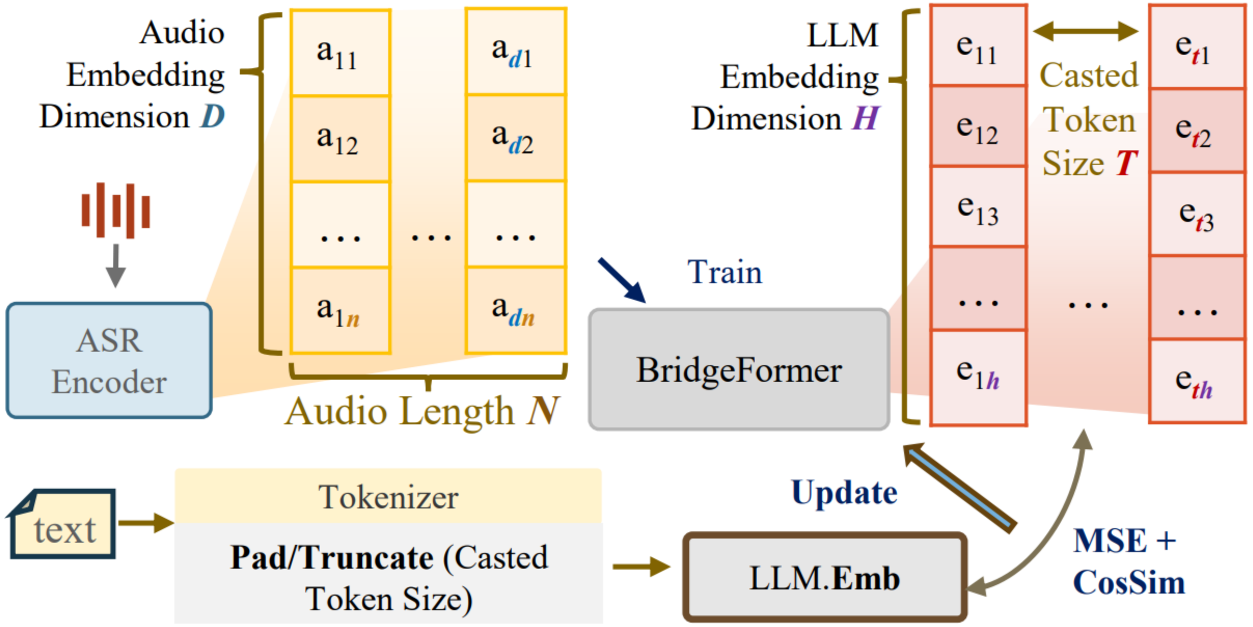
Cross-modal Alignment: Tiny-Align

- BridgeFormer
- **EmbedlLink**
- Feature-based ASR
- Instruction Injection



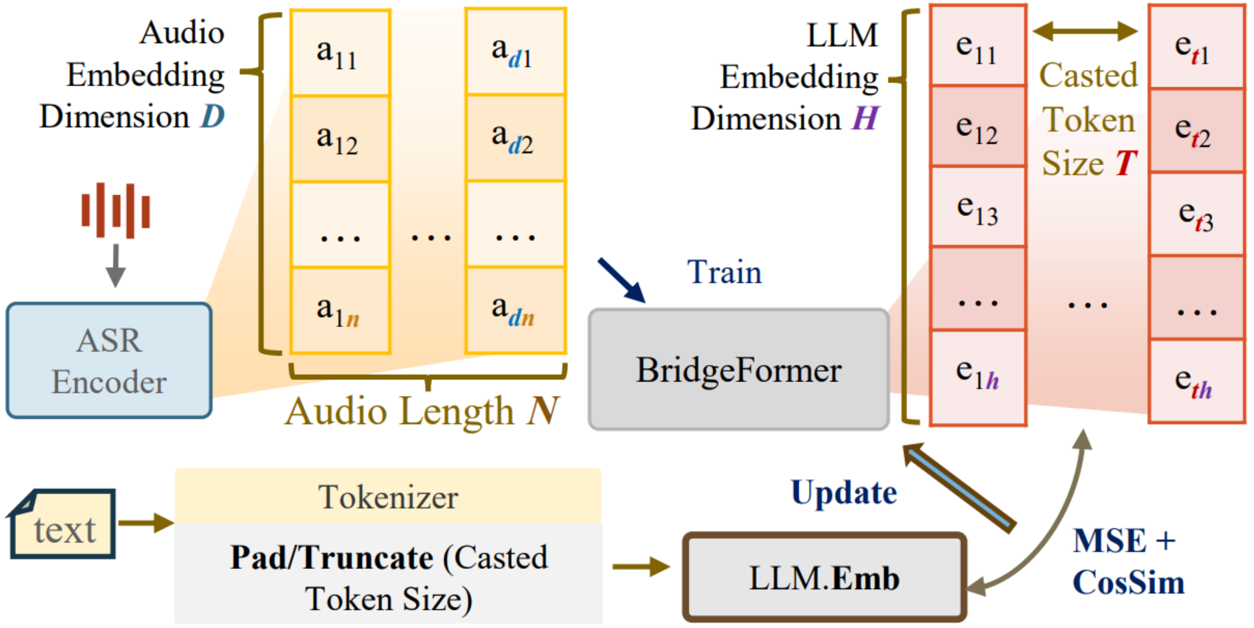
Mismatch dimensions between ASR and LLM

- ASR features have flexible embedding size
- LLM takes fixed embedding size
- Example:
  - Audio: Noise, pause, word speed
  - Text: Highly condensed information
  - Audio >> Text



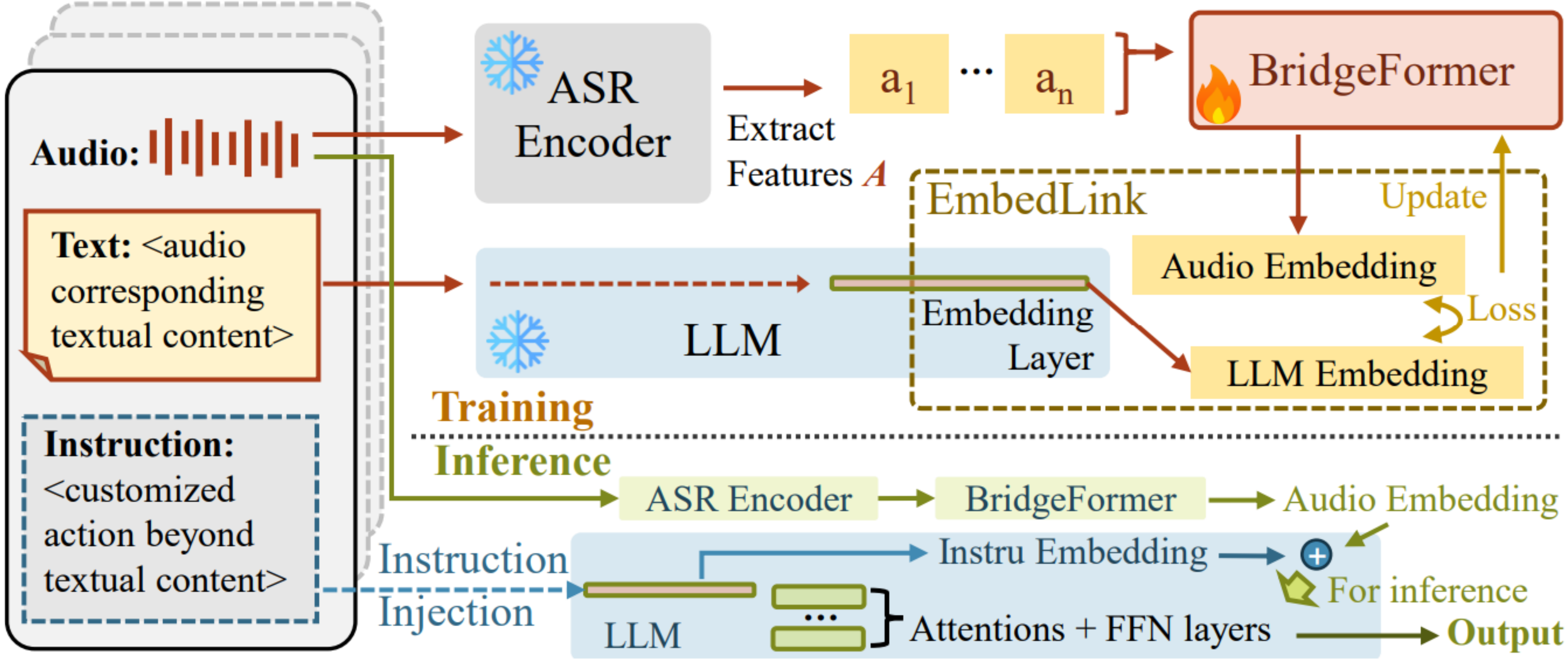
Dealing with mismatch dimensions (EmbedLink)

- Choose an embedding size wisely
  - Smaller than ASR feature size, larger than LLM embedding size
- From ASR, the dimension reduction can be done by MLP in Bridgeformer
- To LLM, the dimension can be cased by padding or truncation
- Default: 30 tokens (1 minute talking).



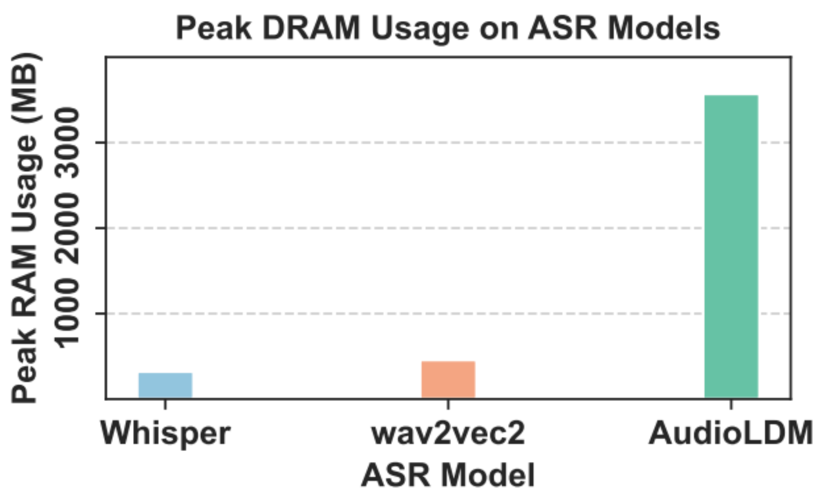
Cross-modal Alignment: Tiny-Align

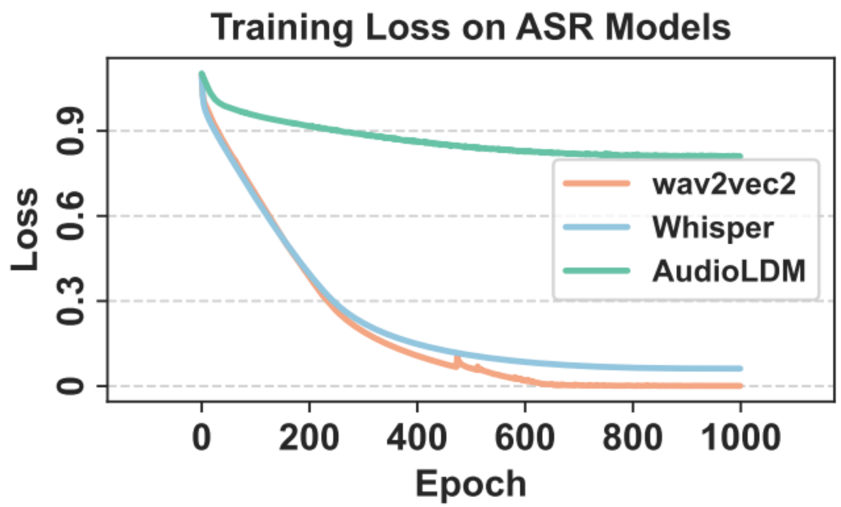
- BridgeFormer
- EmbedlLink
- **Feature-based ASR**
- Instruction Injection



Choose the Appropriate ASR Model

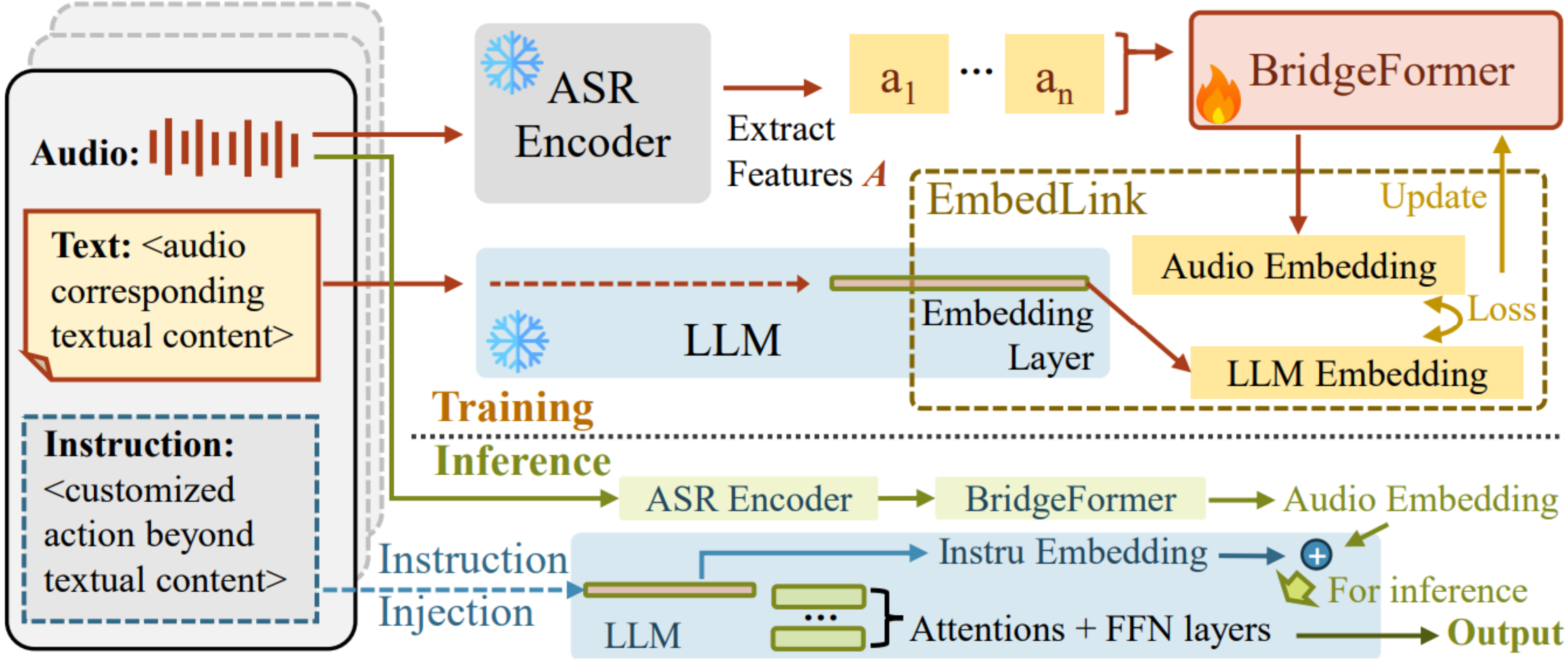
- Input: speech, non-semantic input (i.e., music, mumble by aphasia patient)
- Generative ASR (i.e. AudioLDM) handles non-semantic input well, but 10 times heavier than lite feature-based ASR (i.e. whisper and wav2vec)
- Evaluations also demonstrate the superior performance and efficiency of feature-based ASR.





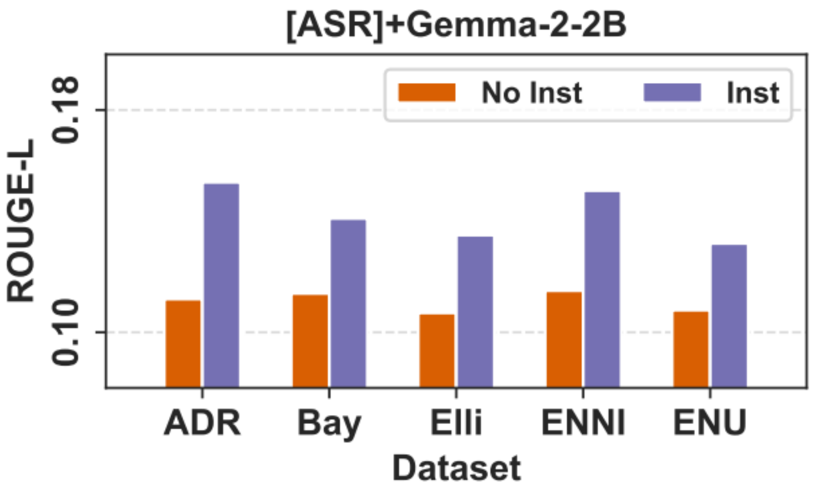
Cross-modal Alignment: Tiny-Align

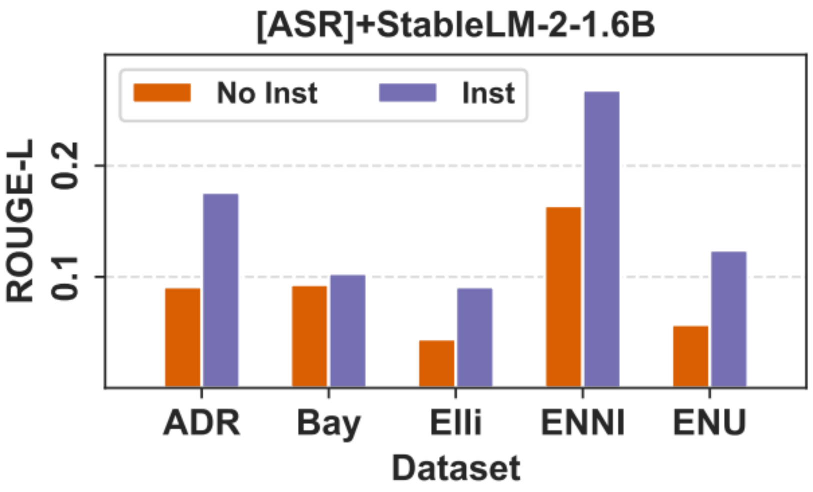
- BridgeFormer
- EmbedlLink
- Feature-based ASR
- **Instruction Injection**



Independent Instruction Provides Flexibility and Performance Improvement

- During projector training, instruction is excluded.
- In inference of ASR-LLM, instruction is concatenated with projector output
- Compared to instruction-included projector training, independent instruction injection provide **more flexibility**.





Performance and Conclusion

- Our framework (Tiny-Align) demonstrates decent performance and efficiency on different audio datasets
- It can benefit people with Dementia, Aphasia, and Specific Language Impairment. Why? On-device learning for personalized audio input, so the LLM can understand such audio input, and process with its strong reasoning capability
- Echo:
  - ***RAG-CiM*** and ***NVCiM-PT***: Can we use **novel circuits** and **energy efficient hardware** to further optimize Tiny-Align?
  - ***Empirical Study*** and ***Data Selection***: Can we use **traditional** devices and algorithms to optimize Tiny-Align?

Dataset	Method	ASR + Llama-3.2-1B			ASR + Llama-3.2-3B			ASR + Gemma-2-2B			ASR + Phi-3.5-mini		
		R-1	R-L	C-T(10s)	R-1	R-L	C-T(10s)	R-1	R-L	C-T(s)	R-1	R-L	C-T(10s)
ADReSS	A1	0.021	0.025	293	0.032	0.027	293	0.051	0.049	2249	0.103	0.072	2520
	A2	0.152	0.138	3107	0.134	0.128	3107	0.196	0.112	7338	0.213	0.119	11875
	A3	0.104	0.096	1128	0.108	0.103	1128	0.185	0.119	1841	0.029	0.026	2093
	Ours	0.192	0.164	111	0.205	0.193	111	0.268	0.154	97	0.225	0.121	104
Baycrest	A1	0.024	0.024	792	0.024	0.024	792	0.017	0.016	11517	0.050	0.047	7375
	A2	0.141	0.104	949	0.152	0.109	949	0.183	0.114	4614	0.202	0.110	3847
	A3	0.065	0.057	3015	0.071	0.067	3015	0.088	0.060	9951	0.012	0.011	6903
	Ours	0.184	0.167	167	0.197	0.173	167	0.233	0.141	133	0.205	0.112	183

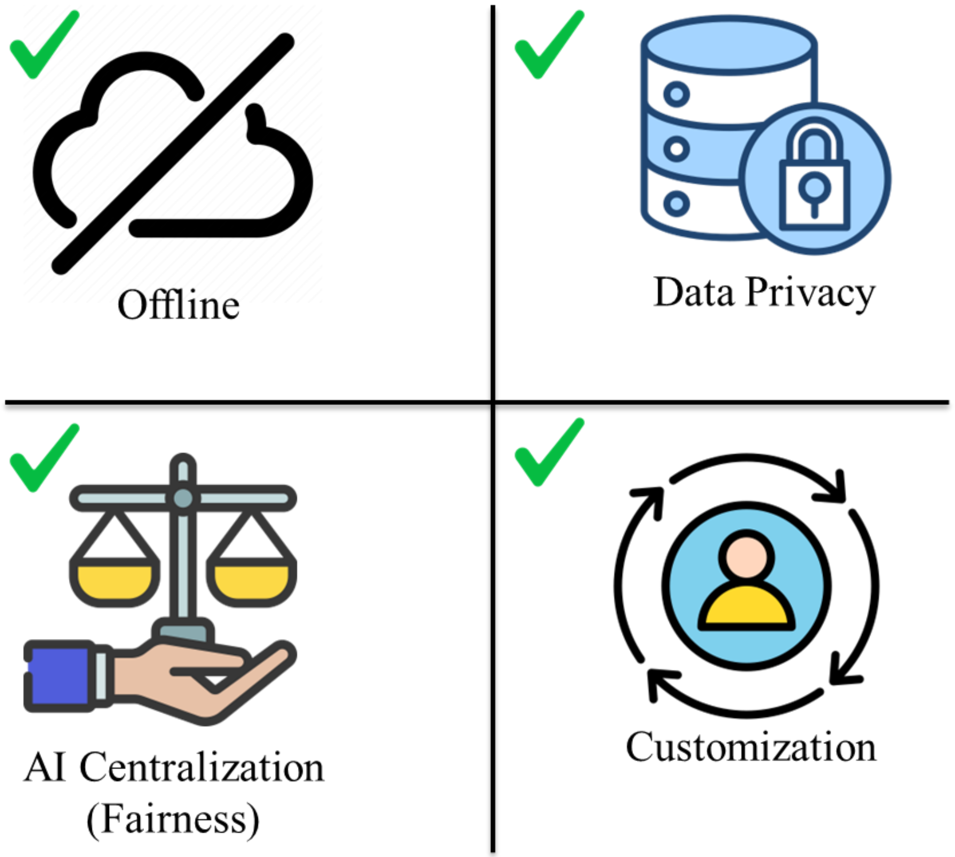
Performance comparison between our framework with existing methods. Metrics including ROUGE-1 (R-1), ROUGE-L (R-L), and Convergence Time (C-T)

Walk Through

THE CHIPS
TO SYSTEMS
CONFERENCE

SPONSORED BY CEDA
IEEE Council on Electronic Design Automation
sigda

Key Takeaways



- Edge LLM can build up an environment towards:
  - AI personalization
  - Data Privacy
  - Robustness and Fairness
- Cross-layer design and optimization can help and optimal decision and strategies depend on edge hardware capacities
- Unleash the potential of emerging techs like (CiM and FeFET) on edge LLM optimization

References

- **Empirical Study**: **Ruiyang Qin**, D. Liu, C. Xu, Z. Yan, Z. Tan, Z. Jia, A. Nassereldine, J. Li, M. Jiang, A. Abbasi and **Yiyu Shi**, "*Empirical guidelines for deploying LLMs onto resource-constrained edge devices*," [TODAES]
- **Data Selection**: **Ruiyang Qin**, J. Xia, Z. Jia, M. Jiang, A. Abbasi, P. Zhou, J. Hu, and **Yiyu Shi**, "*Enabling on-device large language model personalization with self-supervised data selection and synthesis*," [DAC '24]
- **RAG-CiM**: **Ruiyang Qin**, Z. Yan, D. Zeng, Z. Jia, D. Liu, J. Liu, Z. Zheng, N. Cao, K. Ni, J. Xiong and **Yiyu Shi**, "*Robust implementation of retrieval-augmented generation on edge-based computing-in-memory architectures*," [ICCAD '24]
- **NVCiM-PT**: **Ruiyang Qin**, P. Ren, Z. Yan, L. Liu, D. Liu, A. Nassereldine, J. Xiong, K. Ni, S. Hu, and **Yiyu Shi**, "*NVCiM-PT: An NVCiM-assisted prompt tuning framework for edge LLMs*," [DATE '25]
- **Tiny-Align**: **Ruiyang Qin**, D. Liu, G. Xu, Z. Yan, C. Xu, Y. Hu, X. S. Hu, J. Xiong, and **Yiyu Shi**, "*Tiny-align: Bridging automatic speech recognition and large language model on the edge*," [preprint arXiv:2411.13766]

62 THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY CEDA IEEE Council on Electronic Design Automation sigda acm