RevMUX: Data Multiplexing with Reversible Adapters for Efficient LLM Batch Inference

Yige Xu, Xu Guo, Zhiwei Zeng, Chunyan Miao

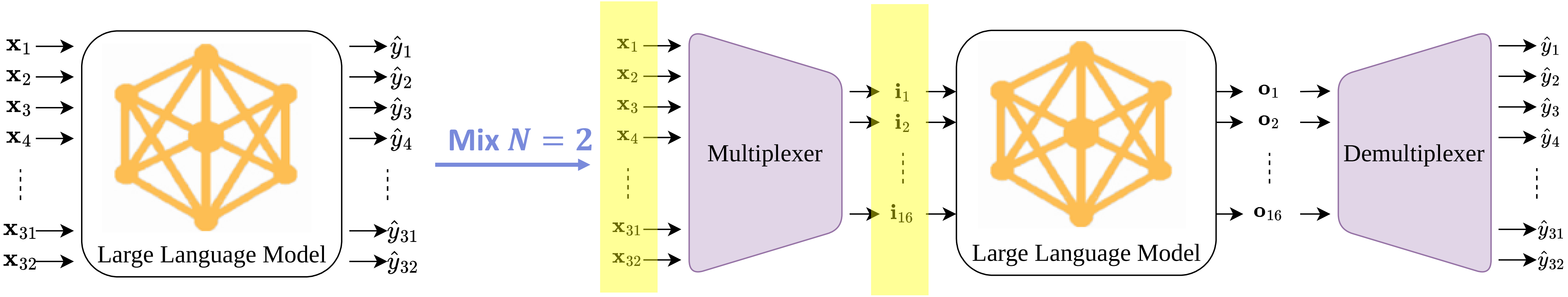
Research Background

- LLM-as-a-Service requires efficient inference.
- Three primary concerns:
  - Latency (the response time): Quantization[1], Model distillation[2].
  - Memory usage (during long-sequence processing): Speculative decoding[3], KV Cache[4].
  - Throughput (number of concurrent queries): DataMUX[5], MUX-PLM[6].

Batch Inference

Data Multiplexing for Batch Inference

- The Multi-Input Multi-Output (MIMO) Architecture
  - Multiplexer: mix a batch of N data samples into **one**
  - Demultiplexer: de-mix the models' output into **a batch**



Mini-batch processing

DataMUX

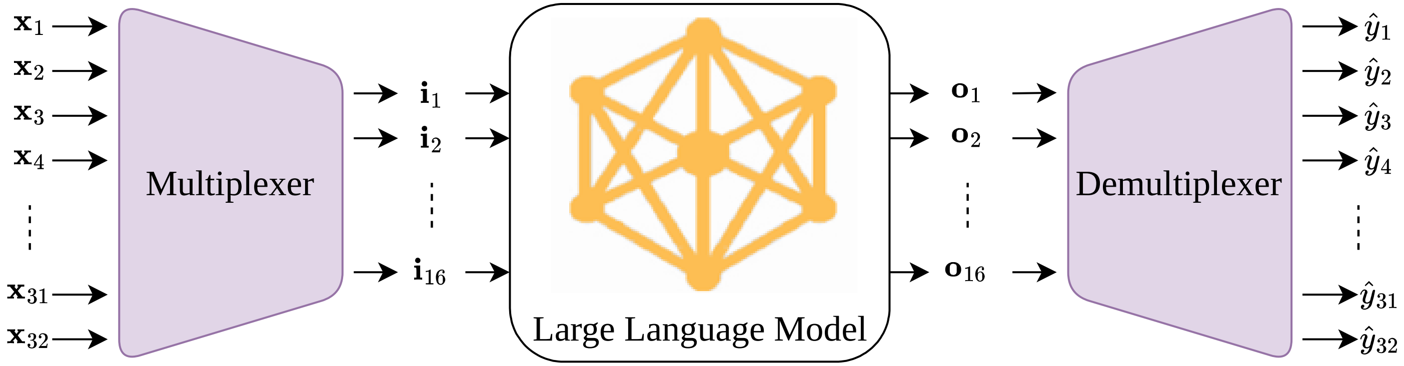
Motivation

Current MIMO-style models consider MIMO as a new task and train LLM together with the multiplexer and demultiplexer [5,6], which results in …

- Failing to handle the original Single-Input Single-Output scenario
- Not applicable to increasingly larger models

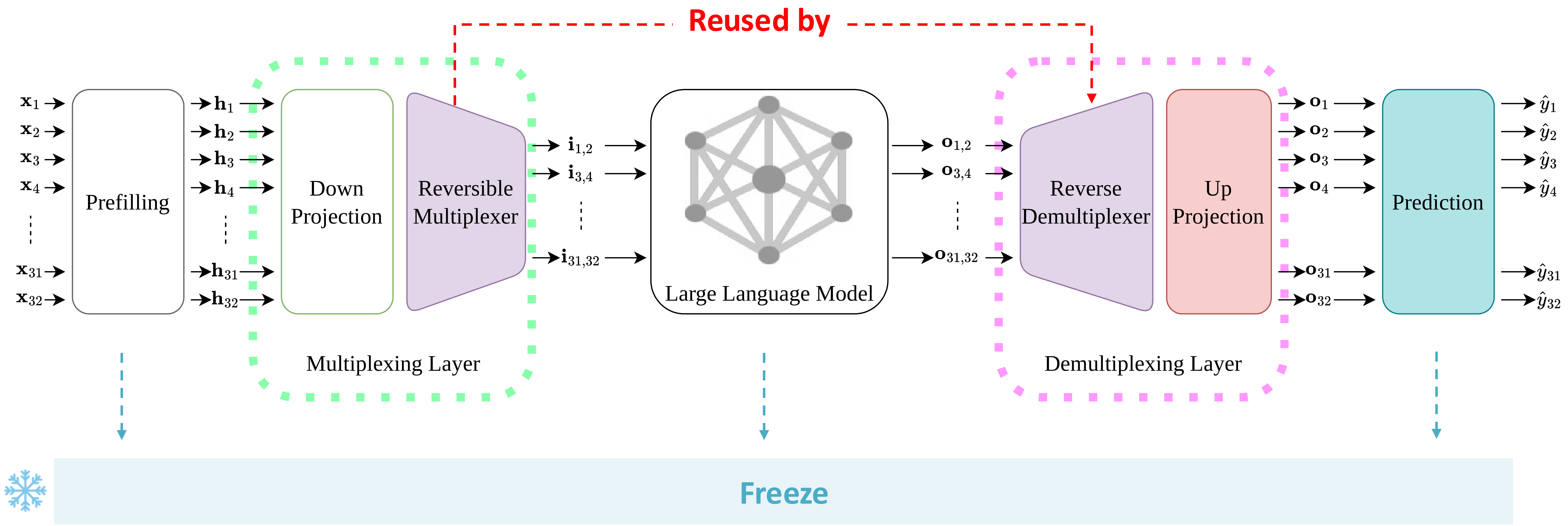
Can we **Freeze LLM** while still achieve data multiplexing?

Challenge: the fixed LLM can struggle to differentiate individuals within the consolidated inputs.

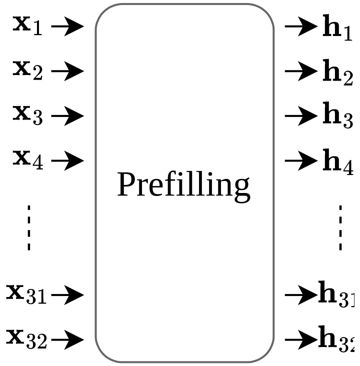


How to trace and preserve the inputs?

RevMUX: Overall Architecture

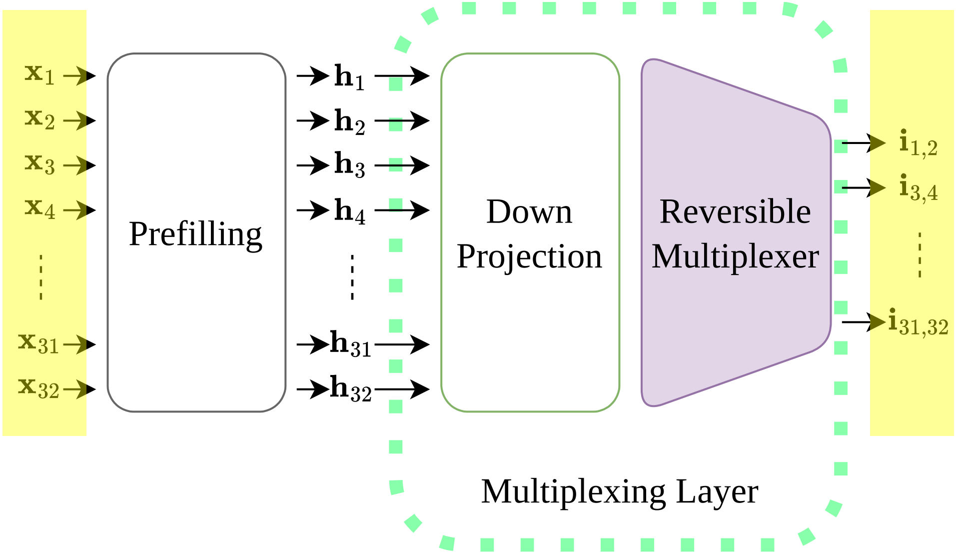


RevMUX: Prefilling

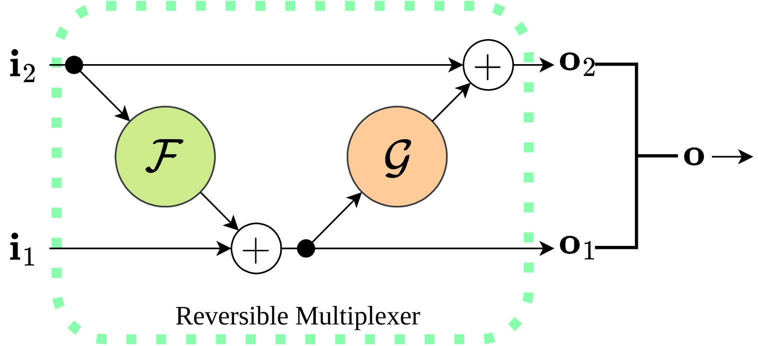


- Prefilling
  - Use first l layers to convert the input instances to dense representations: $N \times \mathbb{R}^d \rightarrow N \times \mathbb{R}^d$
  - Ensure the feature space becoming more similar to the feature space seen during the backbone pre-training

RevMUX: Reversible Multiplexer



Mix $\boldsymbol{N} = \mathbf{2}$ inputs into one



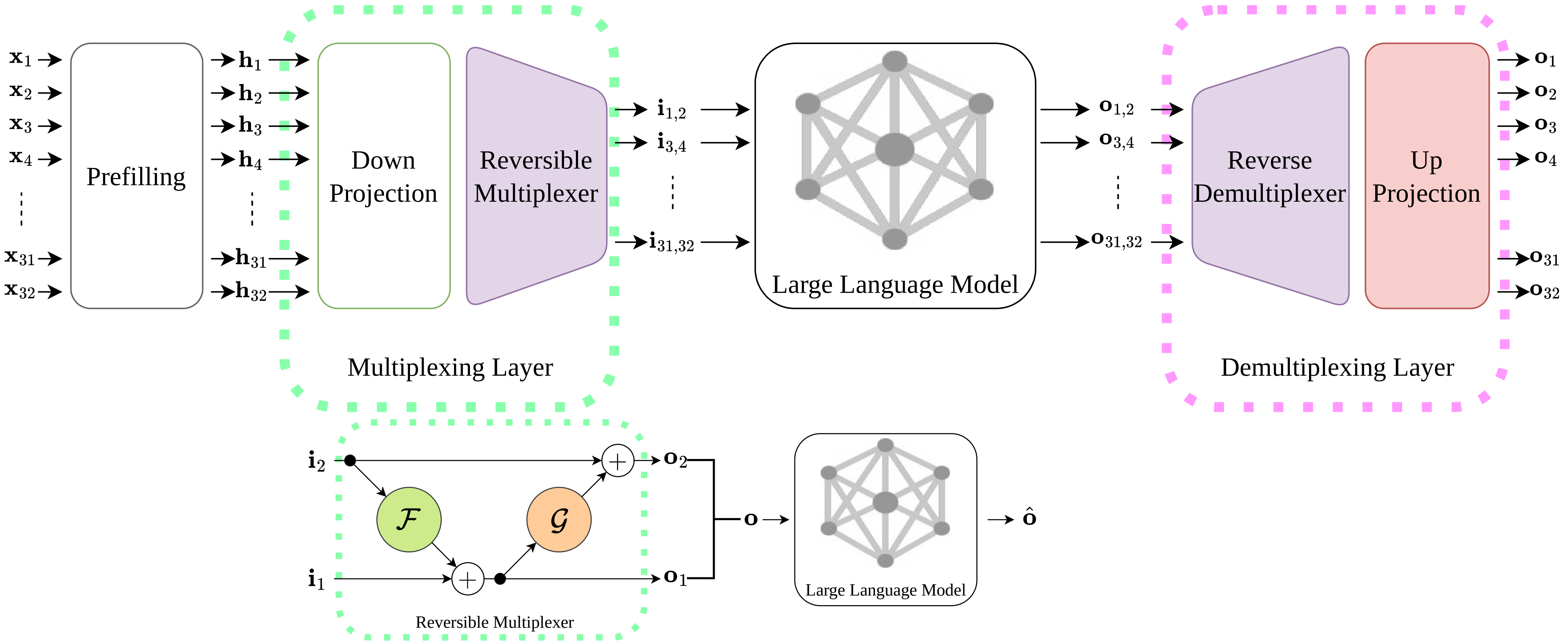
Down Projection: $N \times \boldsymbol{h} \rightarrow N \times \boldsymbol{i}$

$$\boldsymbol{h} \in \mathbb{R}^d, \boldsymbol{i} \in \mathbb{R}^{\frac{d}{N}}$$

$$\mathbf{o}_1^l = \mathbf{i}_1^l + \mathcal{F}(\mathbf{i}_2^l),$$
$$\mathbf{o}_2^l = \mathbf{i}_2^l + \mathcal{G}(\mathbf{o}_1^l),$$
$$\mathbf{o}^l = \text{concat}[\mathbf{o}_1^l, \mathbf{o}_2^l],$$

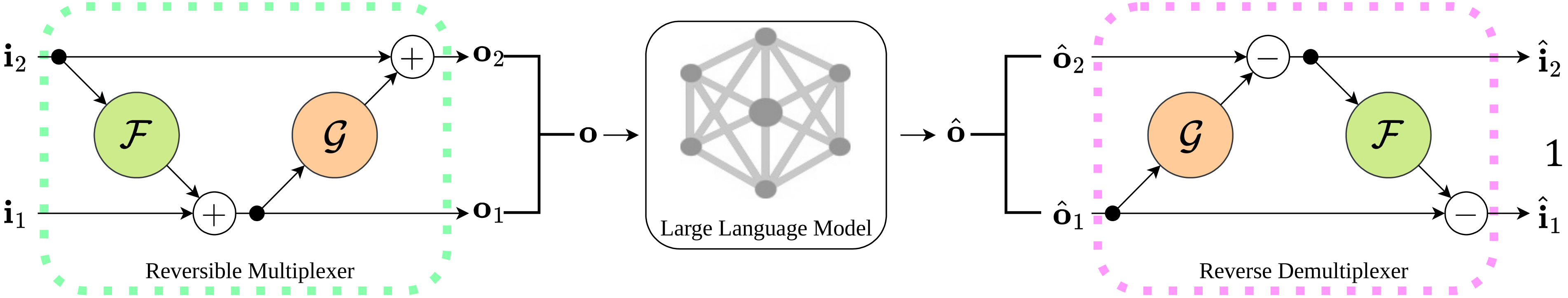
multiplexer: $N \times \boldsymbol{i} \rightarrow 1 \times \boldsymbol{o}$

RevMUX: Reverse Demultiplexer



RevMUX: Reverse Demultiplexer

- The reverse demultiplexer decouples the mixed inputs using the same F and G



$$1 \times \mathbb{R}^d \rightarrow N \times \mathbb{R}^{\frac{d}{N}}$$

$$\mathbf{o}_1^l = \mathbf{i}_1^l + \mathcal{F}(\mathbf{i}_2^l),$$
$$\mathbf{o}_2^l = \mathbf{i}_2^l + \mathcal{G}(\mathbf{o}_1^l),$$
$$\mathbf{o}^l = \mathrm{concat}[\mathbf{o}_1^l, \mathbf{o}_2^l],$$

$$[\hat{\mathbf{o}}_1, \hat{\mathbf{o}}_2] = \hat{\mathbf{o}},$$
$$\hat{\mathbf{i}}_2 = \hat{\mathbf{o}}_2 - \mathcal{G}(\hat{\mathbf{o}}_1),$$
$$\hat{\mathbf{i}}_1 = \hat{\mathbf{o}}_1 - \mathcal{F}(\hat{\mathbf{i}}_2),$$

RevMUX: Training Loss

- Loss function
  - Cross-entropy: classification

$$\mathcal{L}_{\mathrm{ce}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\log(\hat{y}_{i,c})$$

  - InfoNCE loss: match the demultiplexed outputs to those from the standard SISO forward pass

$$\mathcal{L}_{\mathrm{info}} = -\frac{1}{N}\sum_{k=1}^{N} \mathbb{E}\left[\log\frac{\exp(\hat{\mathbf{h}}_k \cdot \mathbf{h}_k)}{\sum_{j=1}^{N}\exp(\hat{\mathbf{h}}_k \cdot \mathbf{h}_j)}\right]$$

  - Joint loss: $\mathcal{L} = \mathcal{L}_{\mathrm{ce}} + \lambda\mathcal{L}_{\mathrm{info}}$

Comparison with baselines

	Model	N	↗	Tuned	Params	SST-2	MRPC	RTE	QNLI	Avg. Score
Backbones	BERT$_{\text{BASE}}$ (Devlin et al., 2019)	1	-	🔥	110M	92.20	87.01	62.96	90.55	83.18
	MUX-BERT$_{\text{BASE}}$ (Murahari et al., 2023)	1	100%	🔥	112M	91.74	87.75	63.18	90.54	83.30
Baselines	DataMUX (Murahari et al., 2022)	2	180%	🔥	166M	90.50	85.05	<u>60.87</u>	<u>88.39</u>	81.20
	MUX-BERT$_{\text{BASE}}$ (Murahari et al., 2023)	2	201%	🔥	112M	90.62	83.77	58.19	88.17	80.19
Ours	Vanilla Adapters	2	156%	❄️	16.53M	90.42	84.78	60.06	88.19	80.86
	Only Multiplexer Reversible	2	161%	❄️	20.07M	90.65	84.60	60.41	88.14	80.95
	RevMUX (❄️)	2	154%	❄️	9.45M	<u>90.85</u>	<u>85.06</u>	60.72	88.25	<u>81.22</u>
	RevMUX (🔥)	2	154%	🔥	120M	**91.21**	**85.78**	**61.41**	**88.72**	**81.78**

- The reversible multiplexer and the reverse demultiplexer together enhance the performance
- RevMUX (❄️) is comparable to DataMUX (🔥)

Inference Efficiency Comparison

	Model	N	$\nearrow$	Tuned	SST-2	MRPC	RTE	QNLI	Avg. FLOPs
Backbones	MUX-BERT$_{\text{BASE}}$ (Murahari et al., 2023)	1	100%	🔥	25.824	11.477	7.651	162.593	51.886
Baselines	DataMUX (Murahari et al., 2022)	2	180%	🔥	13.866	6.400	4.267	90.664	28.799
	MUX-BERT$_{\text{BASE}}$ (Murahari et al., 2023)	2	201%	🔥	12.439	5.741	3.827	81.330	25.834
Ours	Vanilla Adapters	2	156%	❄️	16.545	7.741	5.263	103.663	33.303
	Only Multiplexer Reversible	2	161%	❄️	16.019	7.495	5.096	100.363	32.243
	RevMUX	2	154%	❄️	16.749	7.837	5.328	104.938	33.713

Table 11: Inference efficiency comparison using BERT$_{\text{BASE}}$ as backbone model. (Unit: T FLOPs)

- Although RevMUX (❄️) has slightly higher average FLOPs than DataMUX (🔥), it achieves about 55% to 60% speedups on average, without fine-tuning the backbone language models.

Scalability to Larger Model

- RevMUX is scalable to different model types.
- RevMUX is scalable to billion-scale decoder-only LLMs
- Both the reversible multiplexer and reverse demultiplexer remains effective on larger-scale LLMs

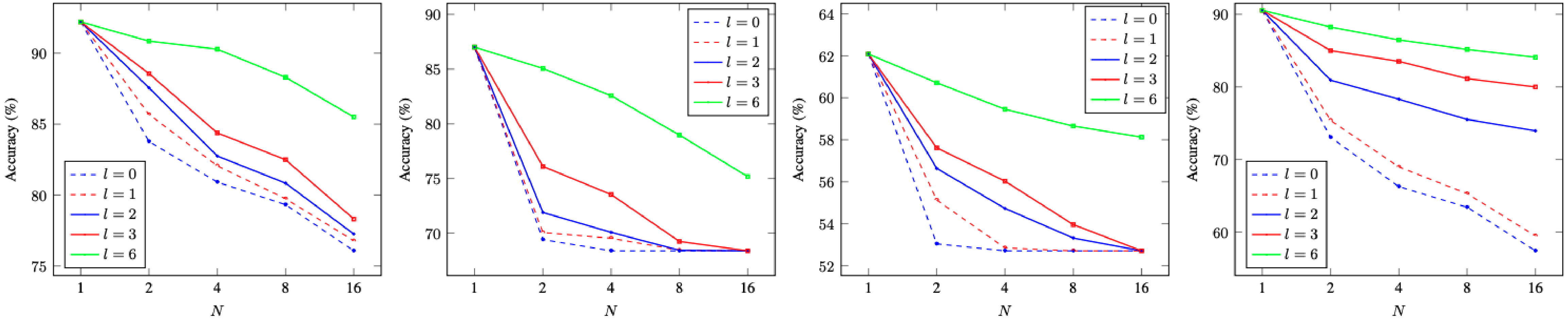
Backbones	Params.	Model	SST-2	MRPC	RTE	QNLI	Avg. Score
$\text{BERT}_{\text{BASE}}$	110M	Vanilla Adapters	90.42	84.78	60.06	88.19	80.86
		Only Multiplexer Reversible	90.65	84.60	60.41	88.14	80.95
		RevMUX	**90.85**	**85.06**	**60.72**	**88.25**	**81.22**
T5_{Small}	60M	Vanilla Adapters	89.00	81.72	57.22	85.36	78.33
		Only Multiplexer Reversible	89.04	82.30	57.51	85.44	78.57
		RevMUX	**89.14**	**82.45**	**60.22**	**85.63**	**79.36**
T5_{Base}	220M	Vanilla Adapters	92.36	82.94	63.28	87.58	81.54
		Only Multiplexer Reversible	92.54	83.19	64.01	88.14	81.98
		RevMUX	**92.70**	**83.80**	**64.73**	**88.65**	**82.47**
T5_{Large}	770M	Vanilla Adapters	92.58	83.16	64.22	88.42	82.10
		Only Multiplexer Reversible	92.67	83.46	64.43	88.56	82.28
		RevMUX	**92.81**	**83.86**	**65.01**	**88.89**	**82.64**
LLaMA3-8B	8B	Vanilla Adapters	94.01	80.96	82.72	85.99	85.92
		Only Multiplexer Reversible	94.09	81.08	82.82	86.24	86.06
		RevMUX	**94.38**	**81.30**	**83.18**	**86.53**	**86.35**

Scalability to Larger N

Model	N	Tuned	SST-2	MRPC	RTE	QNLI	Avg. Score
MUX-BERT$_{\text{BASE}}$	1	🔥	91.74	87.75	63.18	90.54	83.30
RevMUX	2	❄️	90.85	85.06	60.72	88.25	81.22
MUX-BERT$_{\text{BASE}}$	2	🔥	90.62	83.77	58.19	88.17	80.19
RevMUX	4	❄️	90.28	82.57	59.46	86.48	79.70
MUX-BERT$_{\text{BASE}}$	5	🔥	86.88	80.10	59.13	85.58	77.92
RevMUX	8	❄️	88.30	78.97	58.66	85.17	77.78
MUX-BERT$_{\text{BASE}}$	10	🔥	83.44	78.63	58.27	82.08	75.61
RevMUX	16	❄️	85.50	75.17	58.13	84.08	75.72

- RevMUX outperforms MUX-PLM when N=2
- RevMUX maintains comparable or superior performance with larger N

Model Analysis – Number of Prefilling Layer



(a) SST-2 (b) MRPC (c) RTE (d) QNLI

- Increase the number of prefilling layers retains a better performance
- With a sufficient number of prefilling layers (e.g., $l = 6$), the model can maintain relatively high accuracy even when $N = 16$.

Takeaway messages

- We addresses the need for high throughput through data multiplexing, handling batches of concurrent queries while maintaining satisfactory downstream performance
  – Freezing the backbone LLM and allow it to be reused in all tasks.
  – Creating a reversible adapter to enhance the decoupling of mixed inputs.
- RevMUX has demonstrated that
  – it has a better downstream performance than baselines that require finetuning the LLM.
  – it can be scaled to larger billion-scale LLMs
  – it can be scaled to 16-inputs.

References

- [1] Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, Ping Luo. ***Omnidirectionally calibrated quantization for large language models***. ICLR 2024.
- [2] Chengyuan Liu, Yangyang Kang, Fubang Zhao, Kun Kuang, Zhuoren Jiang, Changlong Sun, Fei Wu. ***Evolving knowledge distillation with large language models and active learning***. LREC-COLING 2024.
- [3] Yaniv Leviathan, Matan Kalman, Yossi Matias. ***Fast inference from transformers via speculative decoding***. ICML 2023.
- [4] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Re, ClarkW. Barrett, Zhangyang Wang, Beidi Chen. ***H2O: heavy-hitter oracle for efficient generative inference of large language models***. NeurIPS 2023.
- [5] Vishvak Murahari, Carlos E. Jimenez, Runzhe Yang, Karthik Narasimhan. ***DataMUX: Data Multiplexing for Neural Networks***. NeurIPS 2022.
- [6] Vishvak Murahari, Ameet Deshpande, Carlos E. Jimenez, Izhak Shafran, Mingqiu Wang, Yuan Cao, Karthik Narasimhan. ***Data multiplexing for high-throughput language models***. Findings of EMNLP 2023.