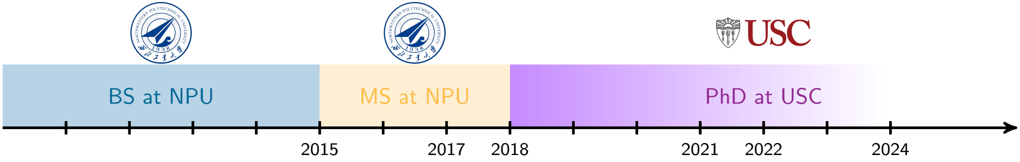
Efficient ML: Hardware to Algorithm

Yue (Julien) Niu
PhD candidate

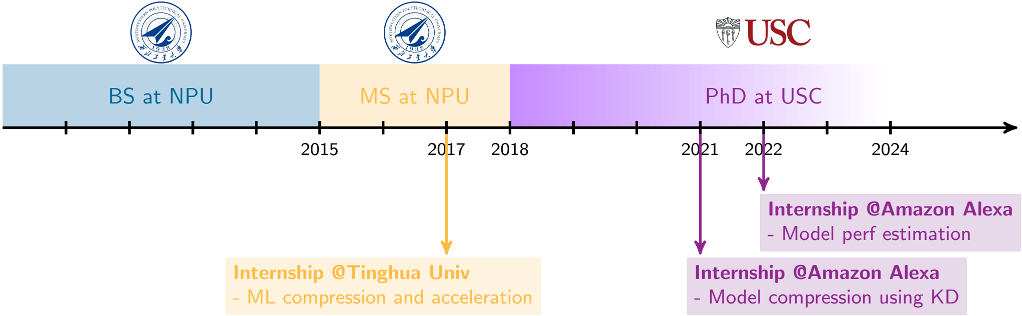
Dept. of Electrical and Computer Engineering
University of Southern California

USC Viterbi
School of Engineering
Ming Hsieh Department of Electrical and Computer Engineering

About Me



About Me


BS at NPU
MS at NPU
USC
PhD at USC
2015
2017
2018
2021
2022
2024
Internship @Amazon Alexa
- Model perf estimation
Internship @Tinghua Univ
- ML compression and acceleration
Internship @Amazon Alexa
- Model compression using KD

About Me

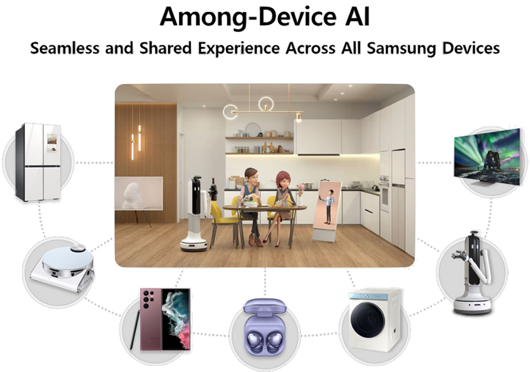
Research Areas:

- ML/LLM **compression** and **acceleration** [CVPR'24, ACL'24 (under review), FPGA'20, $\cdots$]
  - model pruning
  - low-rank compression
  - hardware architecture design
- **Efficient** privacy-preserving ML [CVPR'24, PETS'24, TMLR'23, NeurIPS-FL'23, PETS'22, $\cdots$]
  - differential privacy
  - federated learning
  - trusted execution environments
- LLM privacy, fairness and bias [NAACL'24, AAAI-ReLM'24]
- Stochastic optimization [TMLR'23, ICML'21 Workshop on Optimization]

This talk

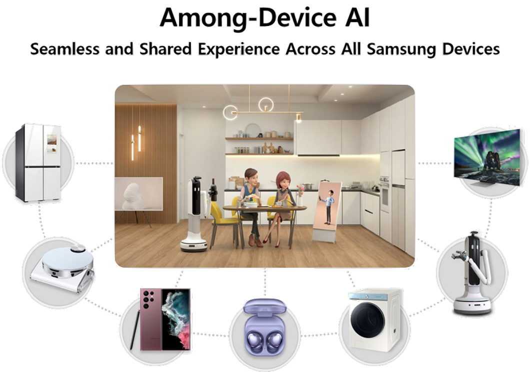
1. ML Acceleration: Hardware Design (brief)
2. Efficient Transformer: Self-Attention with Reduced Complexity
3. Efficient Learning: Model Design with Low-Rank Input

Efficiency Is Always the Goal



Source: Samsung, AI Vision

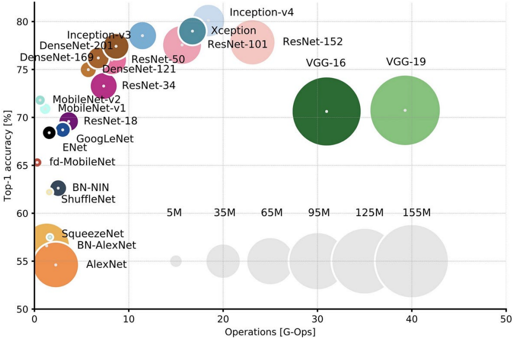
Efficiency Is Always the Goal



Source: Samsung, AI Vision

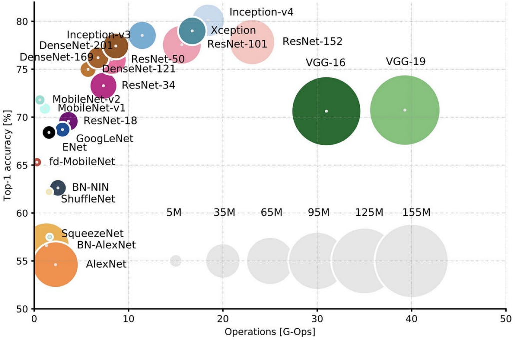
Fast,
Accurate,
Energy-Efficient

Efficiency Is Challenging

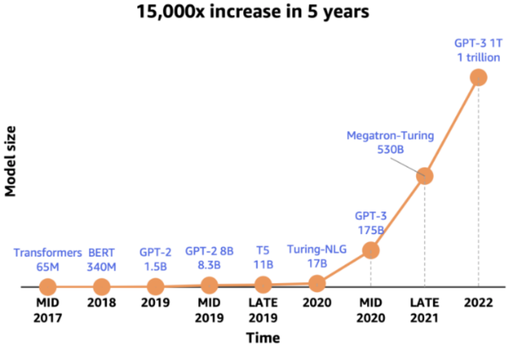


before 2017

Efficiency Is Challenging



before 2017



after 2017

This talk

1. ML Acceleration: Hardware Design (brief)
2. Efficient Transformer: Self-Attention with Reduced Complexity
3. Efficient Learning: Model Design with Low-Rank Input

Overview on ML Acceleration

ML Accelerator on FPGA

- 2017 - 2018 at Tsinghua
- low-rank CNN models
- 16-bit float point
- Tiling-based conv
- 200MHz working frequency
- low latency (200ms)
- Verilog, C++

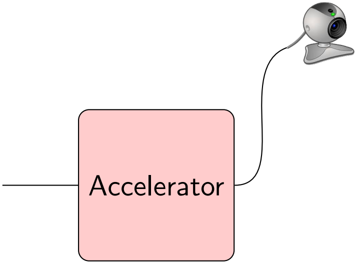
General ML Accelerator

- 2018 at Tsinghua Univ
- generic module
  * conv, matmul, ReLU, $\cdots$
- layer fusing
  * conv-relu
  * conv-relu-pooling
- Tiling-based module
- overlap compute, memory
- automatic model conversion
- Caffe, C++, Verilog

Sparse ML Accelerator

- 2020 at USC
- sparse DNN inference
- frequency-domain conv
- sparse model
- sparsity-aware training
- high model acc
- high throughput
- 16-bit fixed point
- Tensorflow, Verilog

Demo available at: https://www.youtube.com/watch?v=eFW8OTIur38

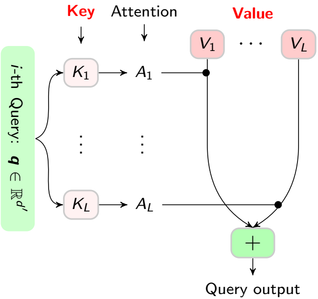


This talk

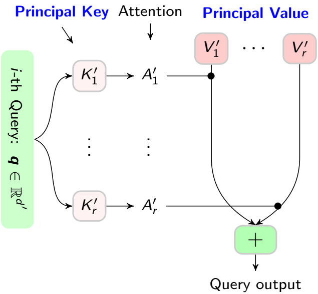
1. ML Acceleration: Hardware Design (brief)
2. Efficient Transformer: Self-Attention with Reduced Complexity
3. Efficient Learning: Model Design with Low-Rank Input

ATP: High-level Overview [ACL'24 under review]

ATP: reduce self-attention complexity from *quadratic to linear*.



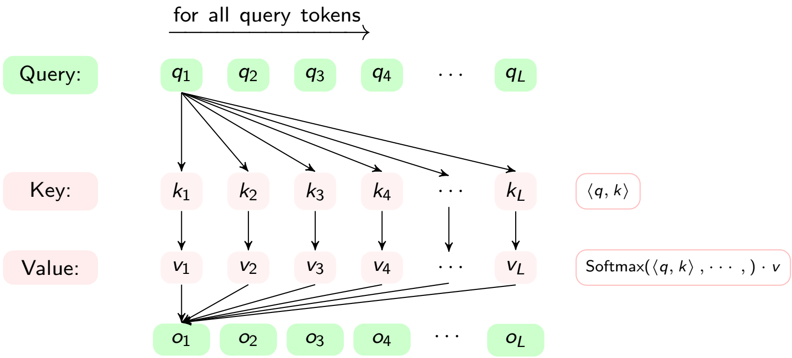
Standard self-attention.



Low-rank self-attention.

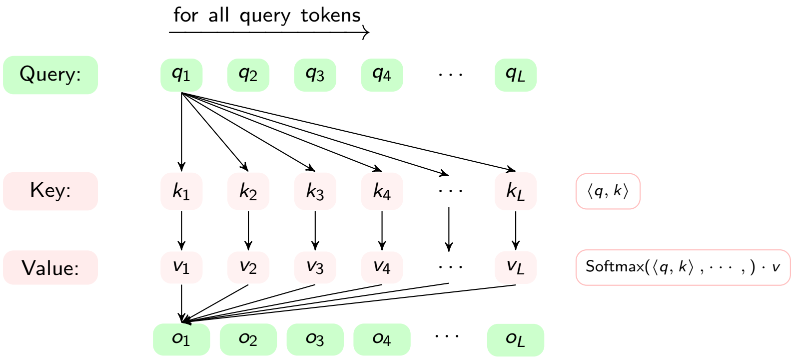
ATP Paper Link

LLMs/Transformers are bottlenecked by self-attention.



Problem and Background

LLMs/Transformers are bottlenecked by self-attention.



- For a sequence with L tokens, self-attention complexity is $O(L^2)$.

Problem and Background

LLMs/Transformers are bottlenecked by self-attention.

layer 1 head 1 head 2 head 3 $\cdots$

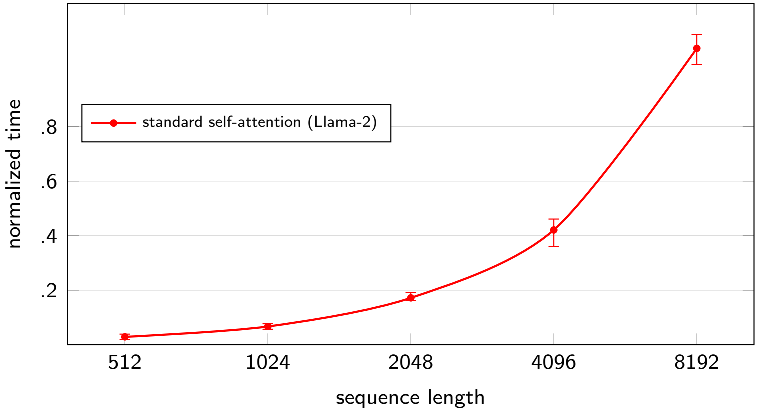
Problem and Background

LLMs/Transformers are bottlenecked by self-attention.

layer 1	head 1	head 2	head 3	$\cdots$
layer 2	head 1	head 2	head 3	$\cdots$
layer 3				
$\vdots$				

Problem and Background

Running time increases quatratically with the sequence length.



Related Works

Quantization

Weights/Activations/KV Cache: FP32/FP16

↓

Weights/Activations/KV Cache: INT8/INT4

- reduce memory footprints;
- reduce computation complexity, friendly to hardware;
- complexity still scales quadratically;
- calibration needed for activation quantization

Related Works

Quantization

Weights/Activations/KV

Cache: FP32/FP16

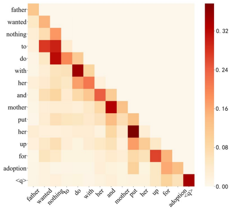
↓

Weights/Activations/KV

Cache: INT8/INT4

- reduce memory footprints;
- reduce computation complexity, friendly to hardware;
- complexity still scales quadratically;
- calibration needed for activation quantization

Sparse Attention



- reduce more redundancy
- good model utility
- sparse computation, not hardware-friendly
- longer running time
  - irregular compute flow
  - bad locality

Related Works

Quantization

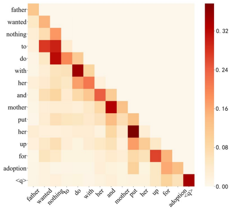
Weights/Activations/KV
Cache: FP32/FP16

↓

Weights/Activations/KV
Cache: INT8/INT4

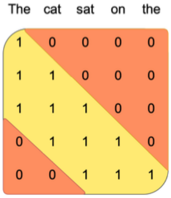
- reduce memory footprints;
- reduce computation complexity, friendly to hardware;
- complexity still scales quadratically;
- calibration needed for activation quantization

Sparse Attention



- reduce more redundancy
- good model utility
- sparse computation, not hardware-friendly
- longer running time
  - irregular compute flow
  - bad locality

Attention w. Small Window

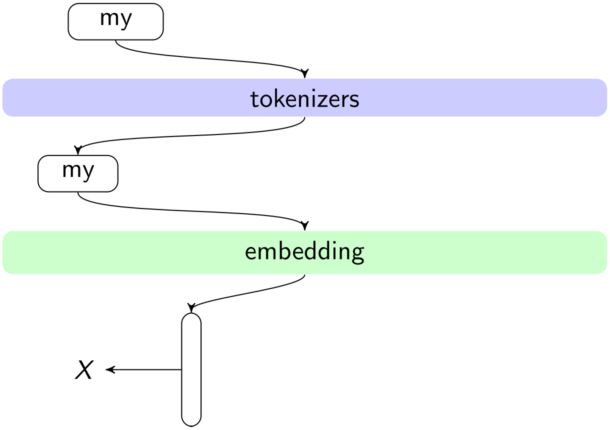


Sliding Window Attention

- support long input sequences
- truncating error due to small window
- unable to model long semantic relationships

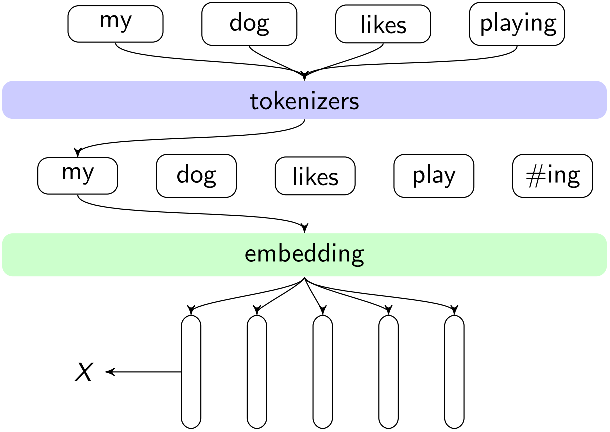
A Key Observation

Internal activations exhibit highly low-rank structure



A Key Observation

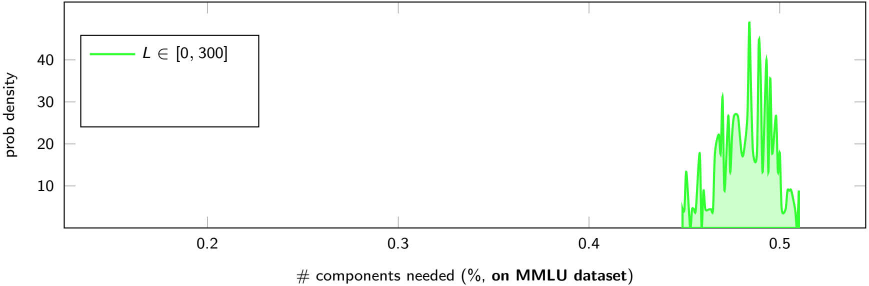
Internal activations exhibit highly low-rank structure



A Key Observation

Internal activations exhibit highly low-rank structure

$$\text{low-rank approximation: } X \xrightarrow{SVD} X_{lr} = U(:, 1:r) \cdot S(1:r, 1:r) \cdot V(1:r, :)$$

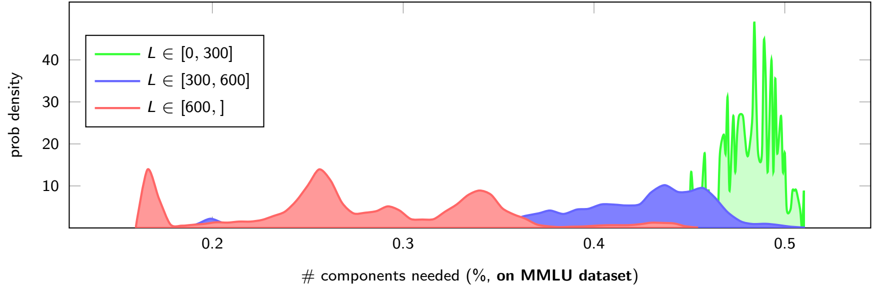


- input sequences can be approximated w. a few principal components

A Key Observation

Internal activations exhibit highly low-rank structure

$$\text{low-rank approximation: } X \xrightarrow{SVD} X_{lr} = U(:, 1:r) \cdot S(1:r, 1:r) \cdot V(1:r, :)$$



- input sequences can be approximated w. a few principal components
- long sequences exhibit more low-rank structure

A Key Observation

An example $\cdots$

Large Language Models are foundational machine learning models that use deep learning algorithms to process and understand natural language. These models are trained on massive amounts of text data to learn patterns and entity relationships in the language. Large Language Models can perform many types of language tasks, such as translating languages, analyzing sentiments, chatbot conversations, and more. They can understand complex textual data, identify entities and relationships between them, and generate new text that **is** coherent and grammatically accurate.	Large Language Models are foundational machine learning models that use deep learning algorithms to process and understand natural language. These models are trained on massive amounts of text data to learn patterns and entity relationships in the language. Large Language Models can perform many types of language tasks, such as translating languages, analyzing sentiments, chatbot conversations, and more. They can understand complex textual data, identify entities and relationships between them, and generate new text that **are** coherent and grammatically accurate.
(a) Original text	(b) approximated text with 20% principal vectors from Word2Vec.

ATP: Low-Rank Self-Attention

How to leverage the low-rank structure of inputs in self-attention
to reduce quadratic complexity?

ATP: Low-Rank Self-Attention

	Standard Self-Attention	Self-Attention w/ Low-Rank Inputs
Input:	$X \in R^{L \times d}$	$X = U \cdot X', \quad X' \in R^{r \times d}$

ATP: Low-Rank Self-Attention

	Standard Self-Attention	Self-Attention w/ Low-Rank Inputs
Input:	$X \in R^{L \times d}$	$X = U \cdot X', \quad X' \in R^{r \times d}$
Query:	$Q = \underline{X} \cdot W^Q$	$Q = \underline{U \cdot X'} \cdot W^Q$
Key:	$K = \underline{X} \cdot W^K$	$K = \underline{U \cdot X'} \cdot W^K$
Value:	$V = \underline{X} \cdot W^V$	$V = \underline{U \cdot X'} \cdot W^V$

ATP: Low-Rank Self-Attention

	Standard Self-Attention	Self-Attention w/ Low-Rank Inputs
Input:	$X \in R^{L \times d}$	$X = U \cdot X', \quad X' \in R^{r \times d}$
Query:	$Q = \underline{X} \cdot W^Q$	$Q = \underline{U \cdot X'} \cdot W^Q$
Key:	$K = \underline{X} \cdot W^K$	$K = \underline{U \cdot X'} \cdot W^K$
Value:	$V = \underline{X} \cdot W^V$	$V = \underline{U \cdot X'} \cdot W^V$
Att:	$A = \texttt{Softmax}(Q \cdot K^*) \cdot V$	$A =???$

Low-Rank Self-Attention

Input:	$X = U \cdot X' \quad X' \in R^{r \times d}$
Query:	$Q = U \cdot \underline{X' \cdot W^Q} = U \cdot \underline{Q'} \quad Q' \in R^{r \times d'}$
Key:	$K = U \cdot \underline{X' \cdot W^K} = U \cdot \underline{K'} \quad K' \in R^{r \times d'}$
Value:	$V = U \cdot \underline{X' \cdot W^V} = U \cdot \underline{V'} \quad V' \in R^{r \times d'}$

ATP: Low-Rank Self-Attention

Low-Rank Self-Attention

Input:	$X = U \cdot X' \quad X' \in R^{r \times d}$
Query:	$Q = U \cdot \underline{X' \cdot W^Q} = U \cdot \underline{Q'} \quad Q' \in R^{r \times d'}$
Key:	$K = U \cdot \underline{X' \cdot W^K} = U \cdot \underline{K'} \quad K' \in R^{r \times d'}$
Value:	$V = U \cdot \underline{X' \cdot W^V} = U \cdot \underline{V'} \quad V' \in R^{r \times d'}$

Attention: $\exp(\boldsymbol{q} \cdot K^T) \cdot V$

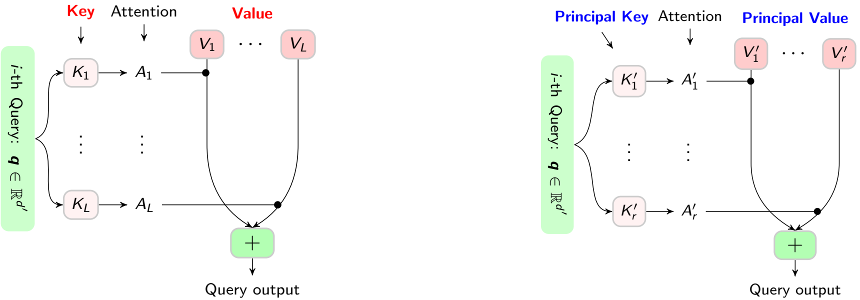
$= \exp(\boldsymbol{q} \cdot K'^T \cdot U^T) \cdot U \cdot V'$

$\simeq \mathbf{1} \cdot U \cdot V' + \boldsymbol{q} \cdot K'^T \cdot U^T \cdot U \cdot V'$ (Taylor expansion)

$= (\mathbf{1} \cdot U + \boldsymbol{q} \cdot K'^T) \cdot V'$

$= A' \cdot V' \leftarrow$ convert to low-rank attention

ATP: Low-Rank Self-Attention



Standard self-attention.

Low-rank self-attention.

Self-Attention Complexity:
$$O(L^2) \to O(r \cdot L)$$

Experiment Highlights

Models

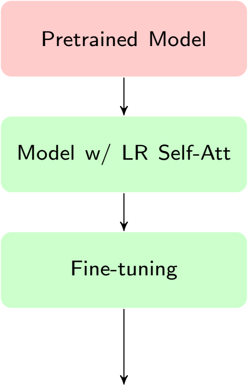
	BERT	Llama2-7B	Llama2-13B
# att layers	12	32	40
# heads/layer	12	32	40
# head dim	64	128	128

Datasets

BERT	Llama2-7B	Llama2-13B
SST2	MMLU	MMLU
Squad	BoolQ	BoolQ
IMDB		

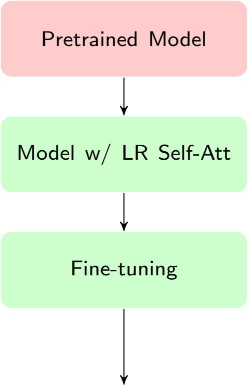
Experiment Highlights

Model Finetuning Procedure



Experiment Highlights

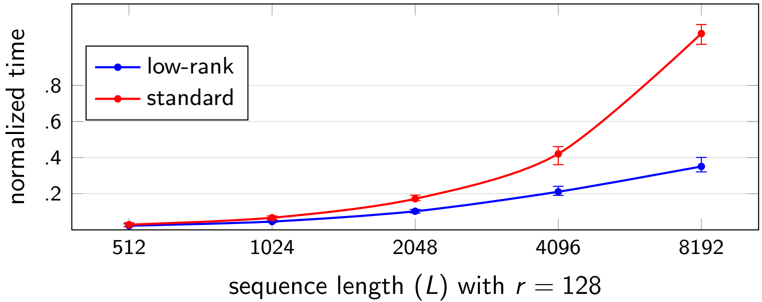
Model Finetuning Procedure



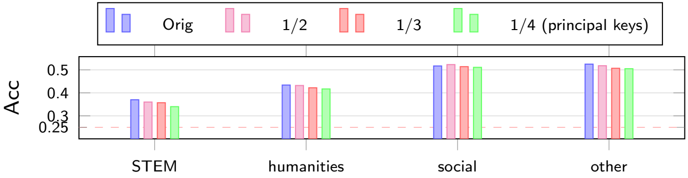
- load model in 4-bit quantized value
- load activations in 16-bit FP
- perform computation in 16-bit FP
- LoRA update w/ 16-bit FP

Experiment Highlights

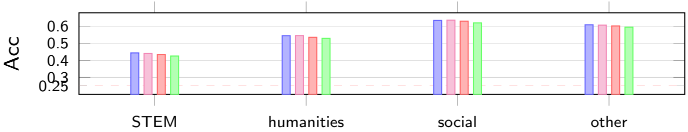
Actual Running Time (test on LLama-2/7B)



Model Accuracy on MMLU



(a) Llama2-7B.

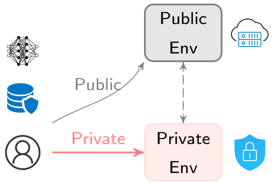


(b) Llama2-13B.

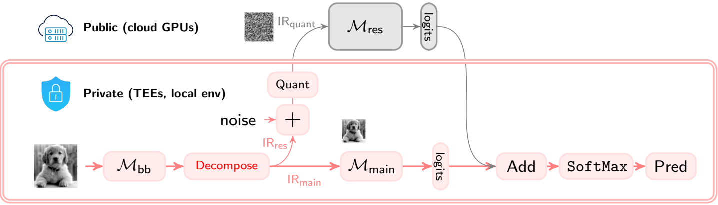
This talk

1. ML Acceleration: Hardware Design (brief)
2. Efficient Transformer: Self-Attention with Reduced Complexity
3. Efficient Learning: Model Design with Low-Rank Input

Low-rank activations enable small-model design



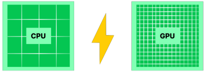
Problem Setup

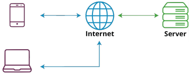


Method Overview

Delta Paper Link

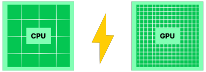
- computation/memory/communication bottleneck in private environments

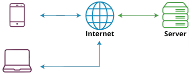




Critical Challenges in Private ML

- computation/memory/communication bottleneck in private environments

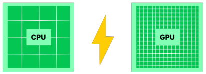


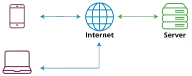


- balance privacy leakage in public environments

Critical Challenges in Private ML

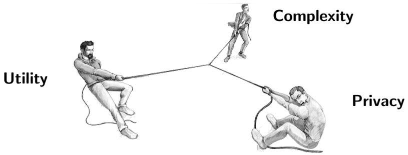
- computation/memory/communication bottleneck in private environments



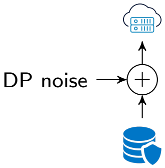


- balance privacy leakage in public environments

- The utility-privacy-complexity trilemma



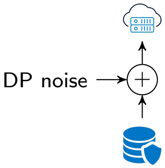
(Naive) DP-based ML



- Provable guarantee
- Severe accuracy drop

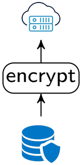
Privacy-Preserving ML Approaches

(Naive) DP-based ML



- Provable guarantee
- Severe accuracy drop

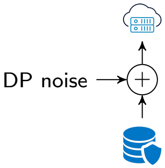
Crypto-based ML



- Strong protection
- High complexity

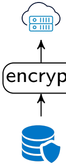
Privacy-Preserving ML Approaches

(Naive) DP-based ML



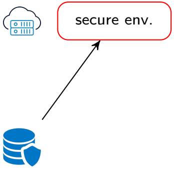
- Provable guarantee
- Severe accuracy drop

Crypto-based ML



- Strong protection
- High complexity

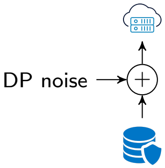
Secure Enclaves



- Hardware security
- Long running time

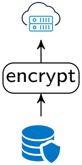
Privacy-Preserving ML Approaches

(Naive) DP-based ML



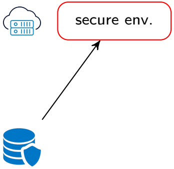
- Provable guarantee
- Severe accuracy drop

Crypto-based ML



- Strong protection
- High complexity

Secure Enclaves



- Hardware security
- Long running time

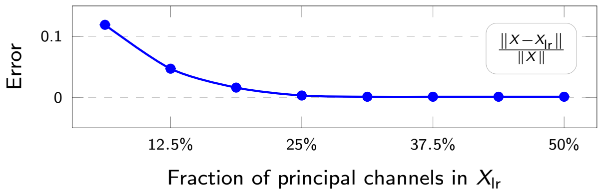
Delta: Private Learning with Asymmetric Flows

What does Delta do?

- Leverage both private (client-side, TEEs, ...) and public (cloud) environments.
- Offload as much computation to public envs as possible, but prevent minimal privacy leakage.
- Preserve as much information in private env as possible, but introduce minimal complexity for training and inference.

Delta: Private Learning with Asymmetric Flows

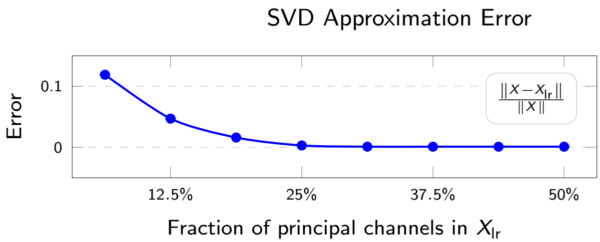
SVD Approximation Error



$$X \xrightarrow{\text{SVD}} U, s, V$$

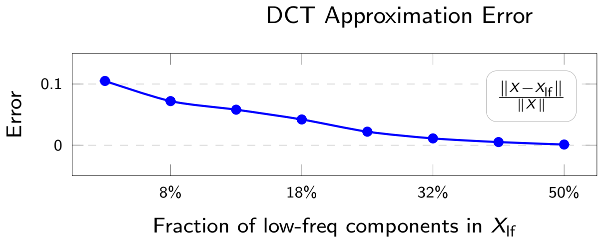
$$X_{\mathrm{lr}} = \sum_{i=1}^{r} s_i \cdot U_i \cdot V_i^*$$

Delta: Private Learning with Asymmetric Flows



$$X \xrightarrow{\mathrm{SVD}} U, s, V$$

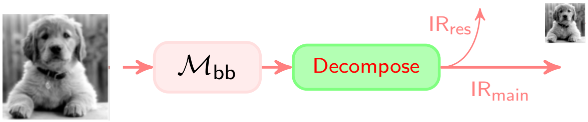
$$X_{\mathrm{lr}} = \sum_{i=1}^{r} s_i \cdot U_i \cdot V_i^*$$



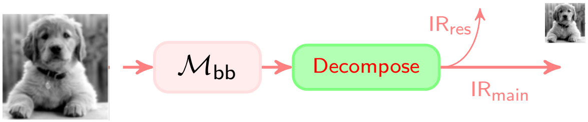
$$X \xrightarrow{\mathrm{DCT}} C$$

$$X_{\mathrm{lf}} = \sum_{i=1}^{r} \mathrm{IDCT}(C(i))$$

Asymmetric data decomposition



Asymmetric data decomposition



$$\mathbf{IR} \xrightarrow{\text{SVD}} \begin{cases} \mathbf{IR}_{\text{SVD main}} \xrightarrow{\text{DCT}} \begin{cases} \mathbf{IR}_{\text{main}} \\ \mathbf{IR}_{\text{DCT res}} \end{cases} \\ \mathbf{IR}_{\text{SVD res}} \end{cases}$$

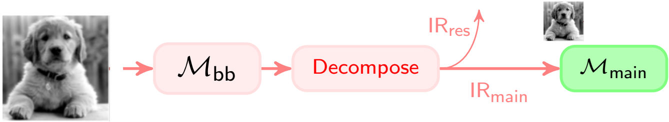
$$\mathbf{IR}_{\text{DCT res}} \oplus \mathbf{IR}_{\text{SVD res}} \rightarrow \mathbf{IR}_{\text{res}}$$

$\mathbf{IR}_{\text{main}}$: few channels & smaller feature size

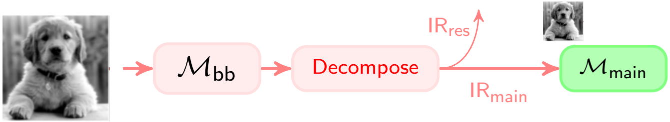
$\mathbf{IR}_{\text{res}}$: few info & large dimension

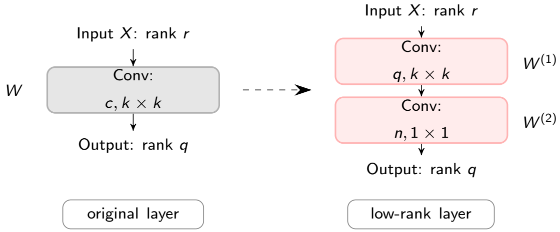
- SVD: asymmetric decomposition along channel dimension
- DCT: asymmetric decomposition along spatial dimension

Efficient model design with low-dimensional data



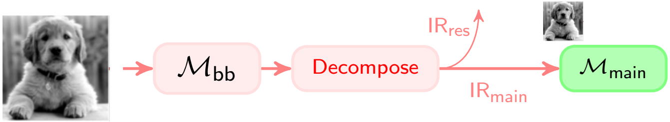
Efficient model design with low-dimensional data

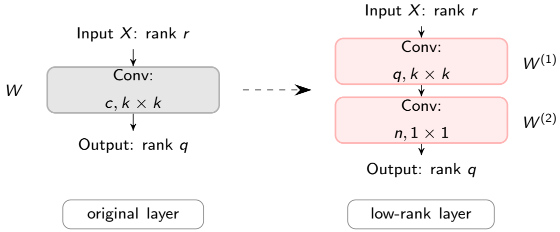




Delta: Private Learning with Asymmetric Flows

Efficient model design with low-dimensional data





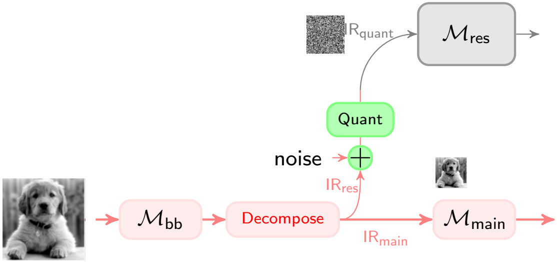
Theorem: By optimizing W^1, W^2, we can achieve:

$$\min_{W^1, W^2} \left\| \text{Op}(W, X) - \text{Op}(W^1, W^2, X) \right\|$$

$$= 0$$

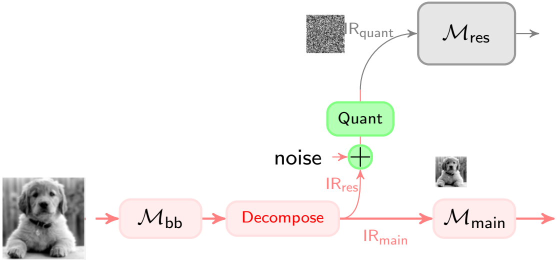
NOT low-rank model compression!

Minimize communication and privacy leakage with random binary quantization



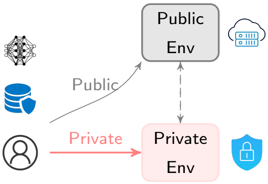
Delta: Private Learning with Asymmetric Flows

Minimize communication and privacy leakage with random binary quantization

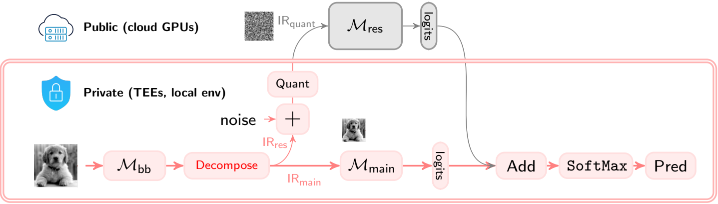


$$\mathrm{IR}_{\mathrm{quant}}(\cdot) = \mathrm{BinQuant}(\mathrm{IR}_{\mathrm{noisy}}(\cdot)) = \begin{cases} 0 & \mathrm{IR}_{\mathrm{noisy}}(\cdot) < 0 \\ 1 & \mathrm{IR}_{\mathrm{noisy}}(\cdot) \geq 0 \end{cases}$$

The full picture



Problem Setup

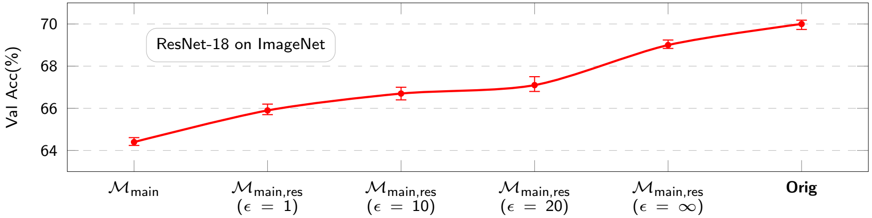


Method Overview

- Asymmetric data decomposition: decouple information from computation
- Efficient model design: reduce complexity in private environments
- Random binary quantization: reduce communication costs

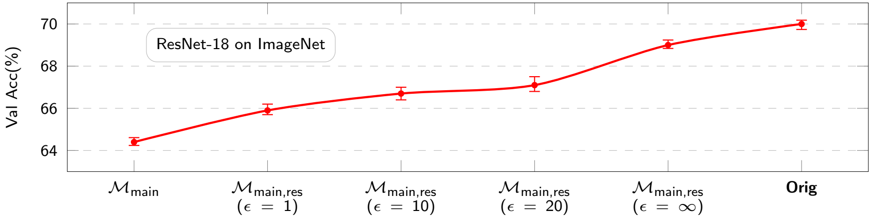
Experiment Highlights

Model Utility



Experiment Highlights

Model Utility



MAC of each module

	$\mathcal{M}_{\text{bb}}+\mathcal{M}_{\text{main}}$	SVD	DCT	$\mathcal{M}_{\text{res}}$
ResNet-18	48.3 M	0.52 M	0.26 M	547M
ResNet-34	437 M	1.6 M	0.7 M	3.5G

Experiment Highlights

Training and Inference Speedup

	Priv-only	3LegRace	Slalom	Delta
Train (ms/speedup)	1372	237 (6×)	-	62 (22×)
Inference (ms/speedup)	510	95 (5×)	84 (6×)	20 (25×)

3LegRace [Niu, et al, PETs 2022]: layer-wise feature decomposition on linear layers
Slalom [Tramer, et al, ICLR 2019]: layer-wise computation distribution on linear layers

- Significant speedup compared to solely using private envs
- Faster compared to baselines due to reduced communication

Summary

ML efficiency is achieved from: efficient hardware, software and algorithm optimization

Hardware
(accelerator, $\cdots$)

Algorithm
(quantization, low-rank, $\cdots$)

Software
(TFLite, MLIR, $\cdots$)