# FLAT: An Optimized Dataflow for Mitigating Attention Bottlenecks

Felix Kao, Suvinay Subramanian, Gaurav Agrawal, Amir Yazdanbakhsh, Tushar Krishna

Google Research

**March 28, 2023 @ASPLOS, Vancouver, Canada**

# Attention-based Models are Important to Accelerate

Model Traits and Trends:

- Attention-based models achieve SOTA quality on a wide range of tasks
  - Language modeling, Question answering, Image generation
- Increased interest in longer sequence lengths

Keyword Extraction
Question answering
Named Entity Recognition
Music Generation
Machine Translation
Summarization
Image Recognition
Composing music
Image Classification
Image Generation
Text classification
Information Retrieval
Speech Recognition
Semantic Analysis
Recommendation
Search
Information Grouping

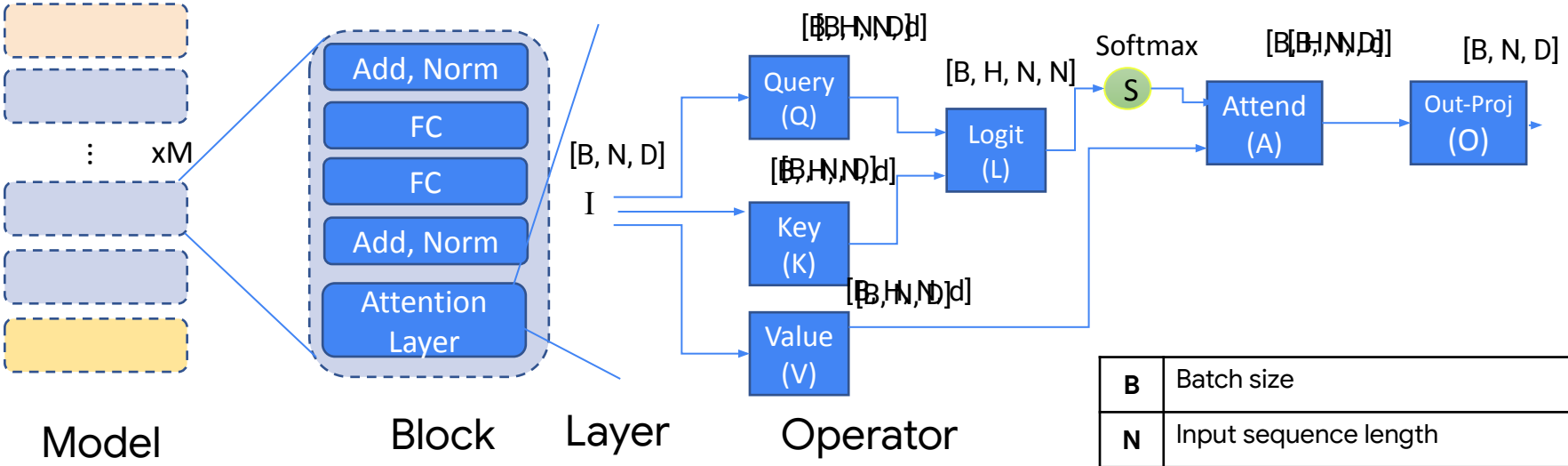


# Why is Attention Challenging to Accelerate?

1. Quadratic growth in complexity with sequence length
2. Compute under-utilized because of memory boundedness
3. Quadratic growth in memory (footprint, bandwidth) with sequence length

Focus of this work

# Canonical Attention-based Model




## Terminology

| | |
|---|---|
| **B** | Batch size |
| **N** | Input sequence length |
| **H** | Number of head |
| **D** | Hidden size of query, key and value |
| **d** | The split hidden size, d = D/H |

# Canonical Accelerator and Dataflow

## Accelerator

Compute Processing Element (PE) array

Memory On-chip scratchpad

Off-chip memory

High bandwidth low capacity

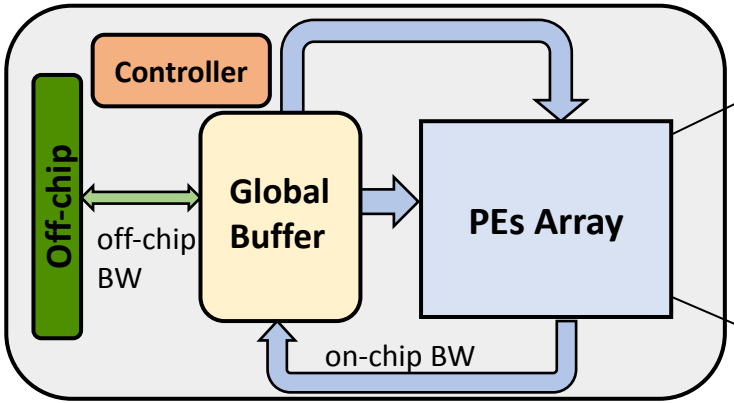
Low bandwidth high capacity

## Dataflow: Mapping & Scheduling

Tiling How tensors are sliced, fetched, stored

Schedule How operators, loops are ordered

Focus of this work




```
for b=[0, B):
 for h=[0,H):
  for m=[0, N):
   for n=[0, N):
    for k=[0, d):
        L[b,h,m,n] +=
      Q[b,h,m,k]*K[b,h,d,n]
```

# Operation Intensity and Roofline Model

## Operation Intensity

Captures relative compute- or memory-boundedness of an operation

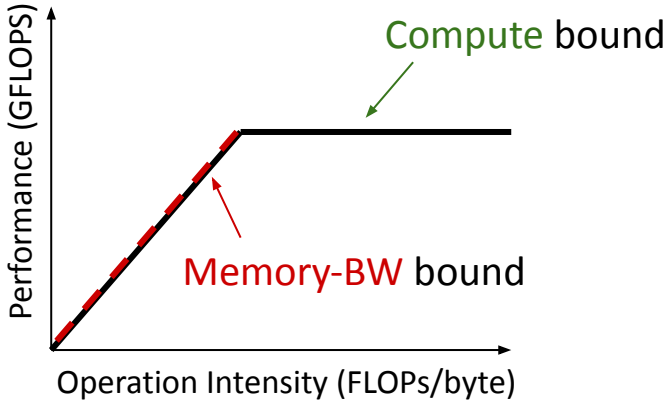
$$op.\ int. = \frac{\#\ ops}{\#\ mem\ accesses}$$

Higher operation intensity

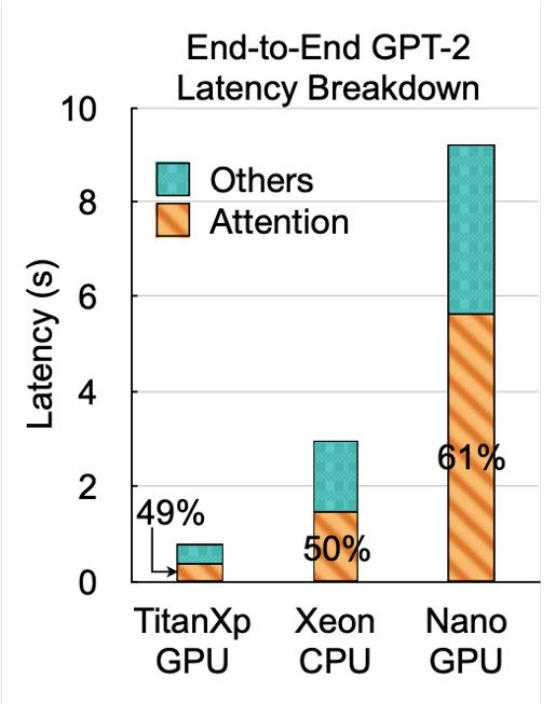
⇒ Higher compute utilization

## Roofline Model

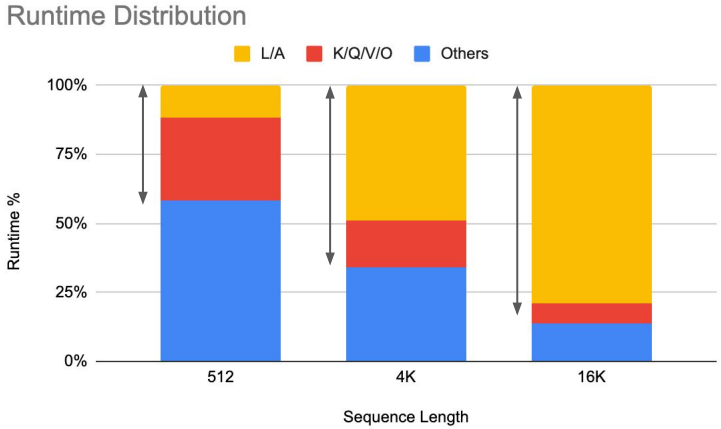
Visual model of achieved performance vs. operation intensity



# Attention Layer is Critical in Transformer-based Models




Attention accounts for >50% of total runtime.




With increase of sequence length, attention could accounts > 75% of total runtime.

Logit, Attend is the major contributors

**Well, larger GPU, TPU, problem solved?**

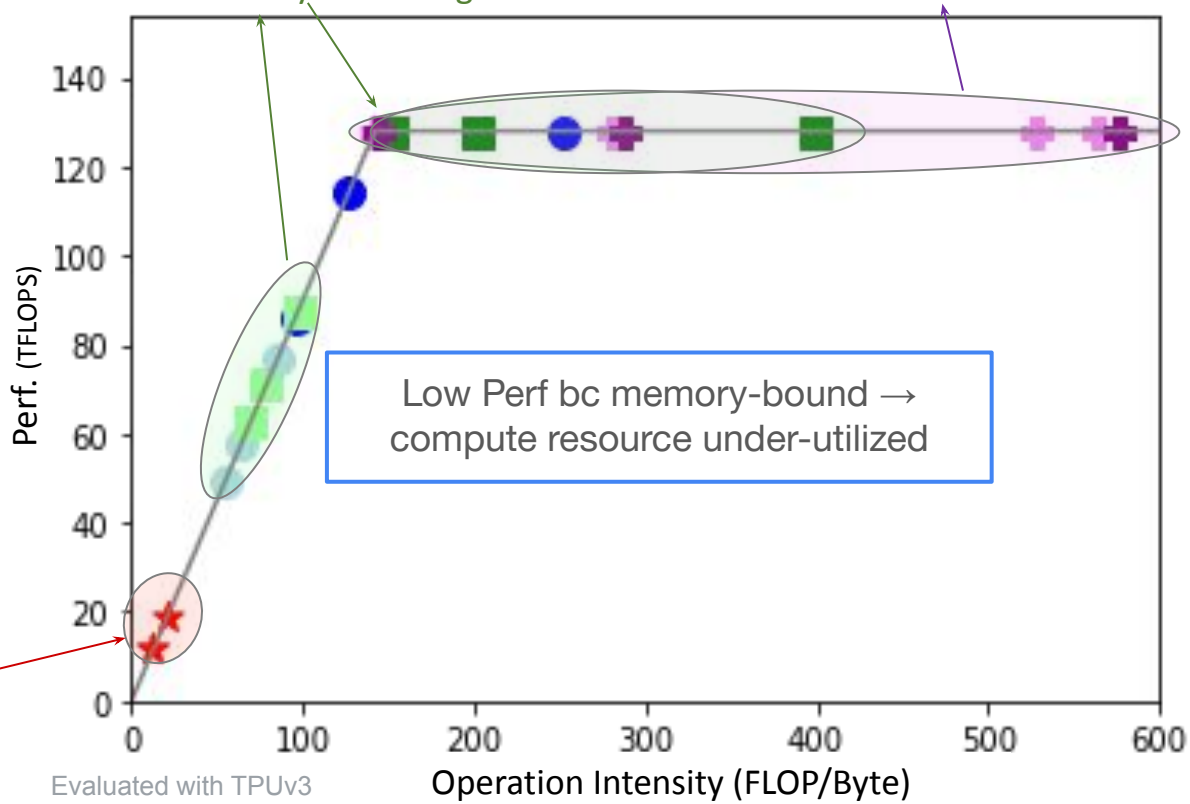
# Challenge 1: Low Compute Resource Utilization

Model: BERT, TrXL, XLM, Resnet

| Operator | |
|---|---|
| L/A | ★ ★ |
| FC (K/Q/V/O) | ● ● |
| FC (Others) | ■ ■ |
| CONV | ✚ ✚ B=1 B=128 |

FC's performance can often be increased by increasing batch size.

CONV usually compute-bound



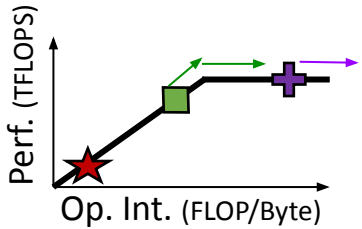

Computer resource seriously under-utilized.
Batch size cannot help.

Evaluated with TPUv3

# What's Missing?

Let's revisit **operation intensity** $op.\ int. = \frac{\#\ ops}{\#\ mem\ accesses}$

The easy trick often works: just pact as many inputs together (increase batch size)




Standard FC

Batch size — Increase batch size, increase op. Int.

Input size — Decided by the dataset

Hidden dimension — Fixed at design time

$$\mathcal{O}\left(\frac{BND^2}{BND+D^2+BND}\right)$$

L, A operator in attention

Batch size

Decided by the dataset — Sequence length

Hidden dimension — Fixed at design time

$$\mathcal{O}\left(\frac{BN^2D}{2BND+BHN^2}\right)$$

Algorithmically, our hands are tied.
**We propose to fuse L and A operators into one to open up new opportunity.**

# Other Fusions?

Fusion (in this work): Tensor operation fusion

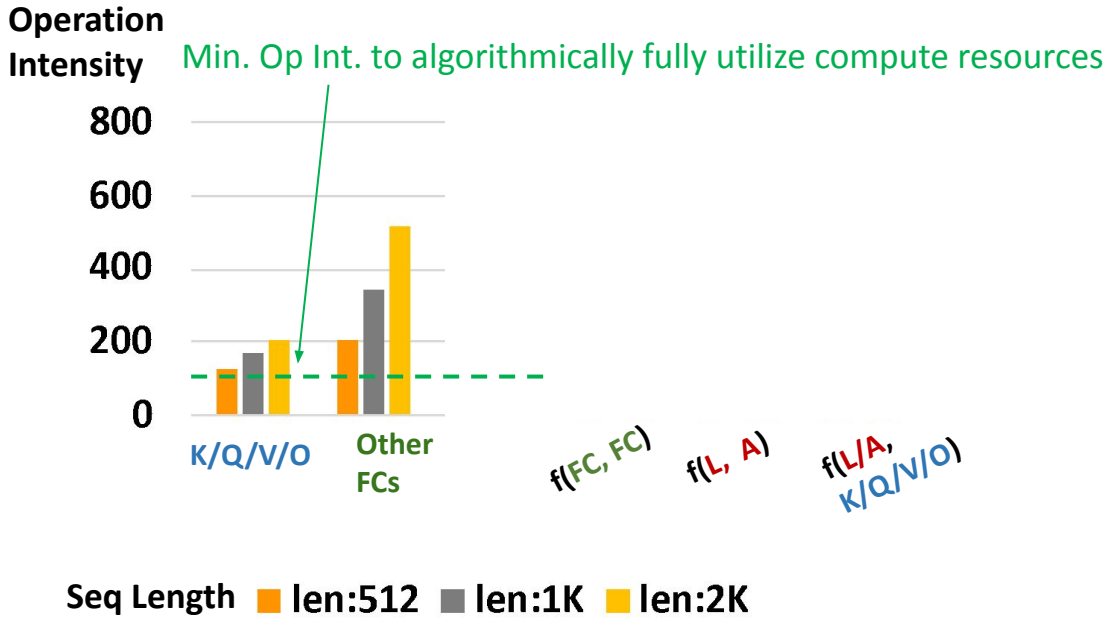
CONV-CONV, FC-FC, Einsum-Einsum

DL compilers are pretty decent at other fusion opportunities

Fusing other attention operators?

Already has enough operation intensity without fusion

Fusing them algorithmically don't give extra benefit

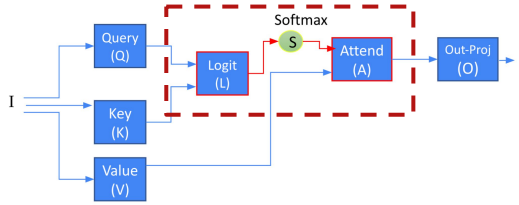



Evaluated with TPUv3

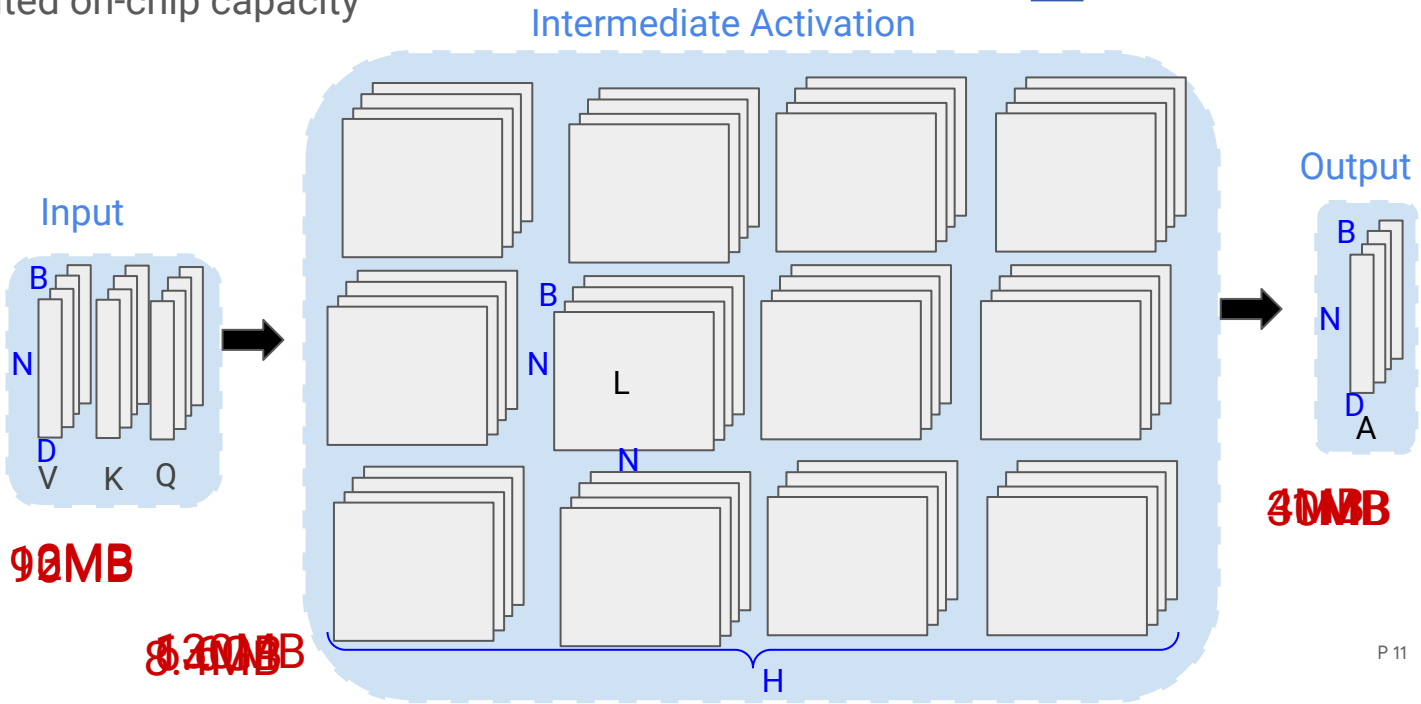
# Challenge 2: Large On-Chip Buffer Requirements

## Staging intermediate activation on-chip is expensive

- Ideally: Leveraging high on-chip memory BW
- In practice: Limited on-chip capacity



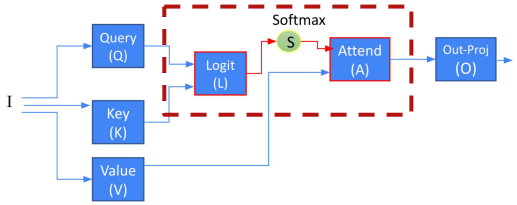

We optimize Dataflow (Tiling, Scheduling) in this work



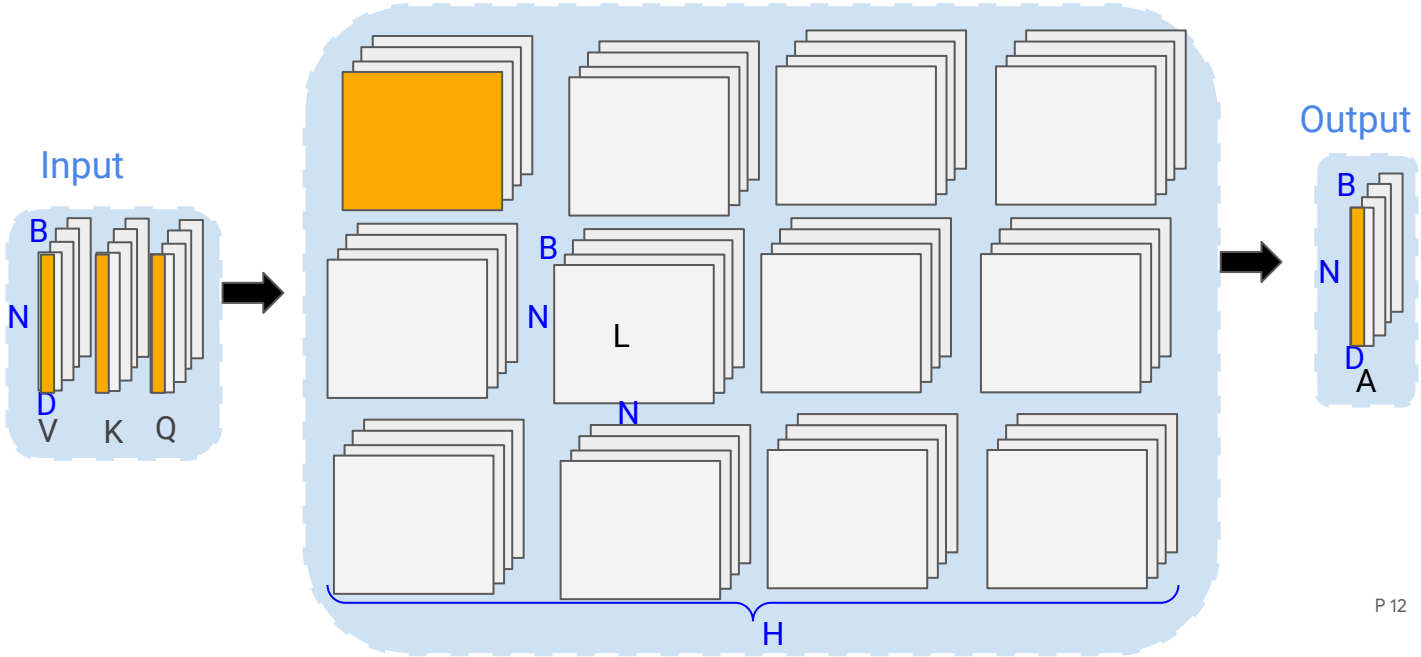
# Challenge 2: Large On-Chip Buffer Requirements

## Staging intermediate activation on-chip is expensive

- Ideally: Data reuse and leveraging high on-chip memory BW
- In practice: Limited on-chip capacity

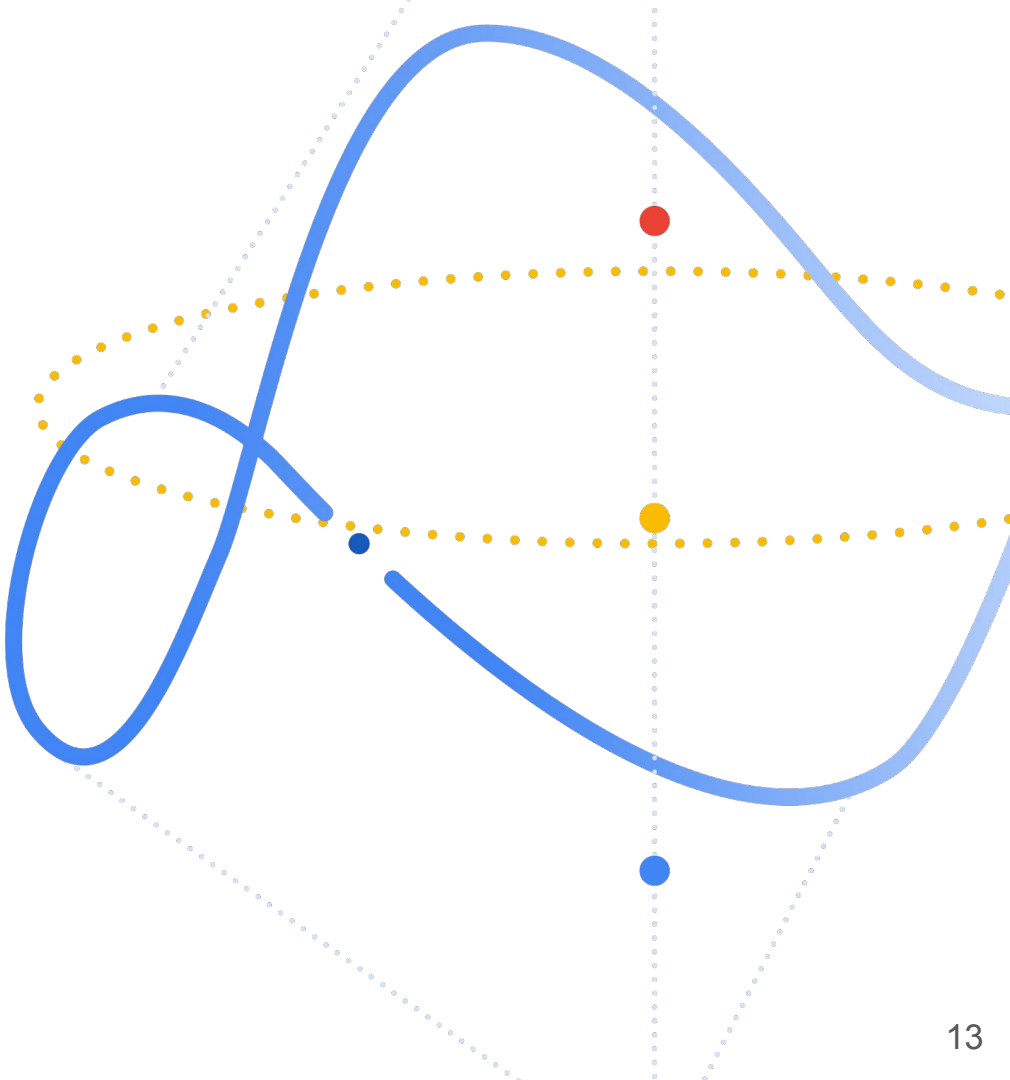



We optimize Dataflow (Tiling, Scheduling) in this work

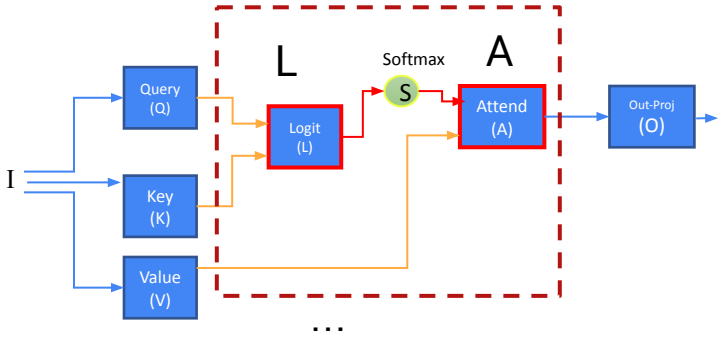


Google Research

# FLAT: Fused Logit Attend Tiling



# FLAT: Fused Logit-Attend Tiling




## FLAT-dataflow

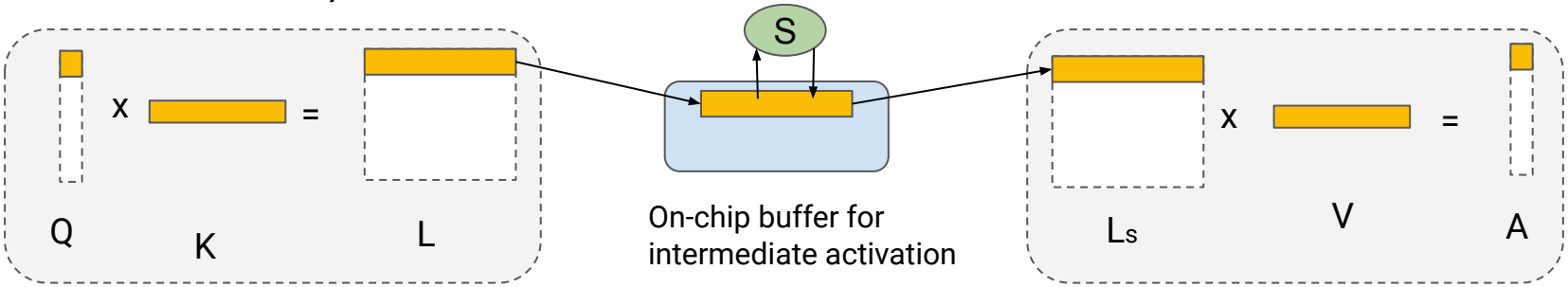
- Tiling: slice of data
- Scheduling: data movement

## Why FLAT-tile? Respect data dependency

- Softmax requires certain slices of data to reduce correctly

Row-wise softmax

Basic unit of FLAT-tile: Row-granularity

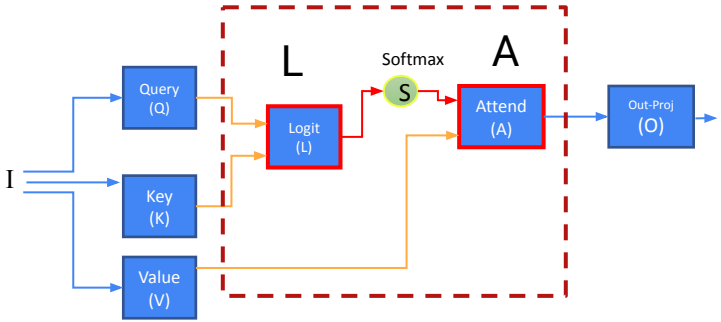


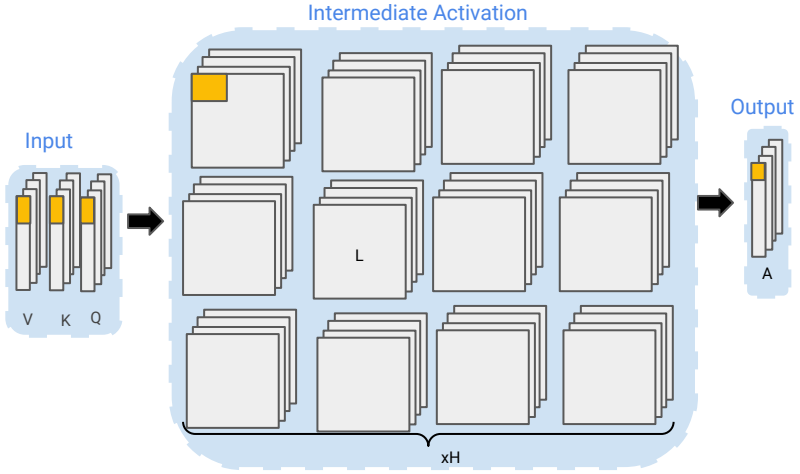
# Logit Attend Fusion

Fuse to compute L, A operations simultaneously

- Advantage: Immediate reuse of slice of intermediate activation
  - Buffer requirement reduces ↓

    ⇒ Able to stage on-chip



# FLAT Dataflow Features

**Tiling:**

Exploring different tile sizes, granularities

- Basic tiling unit: Row-granularity
  - Parameterized FLAT-tile size:
    R-(Row) H-(Head)
    B-(Batch) M-(Batch Multihead)
    granularity

**Scheduling:**

Cross-fused-operators:

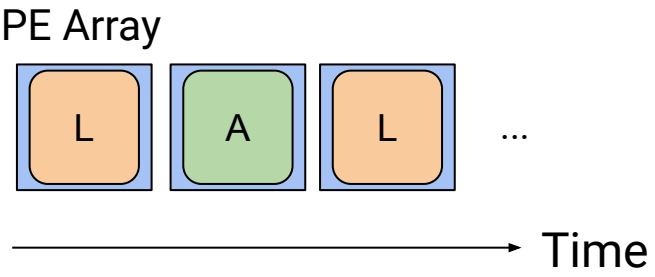
- Interleaved execution: better latency, simple control

Intra-fused-operator:

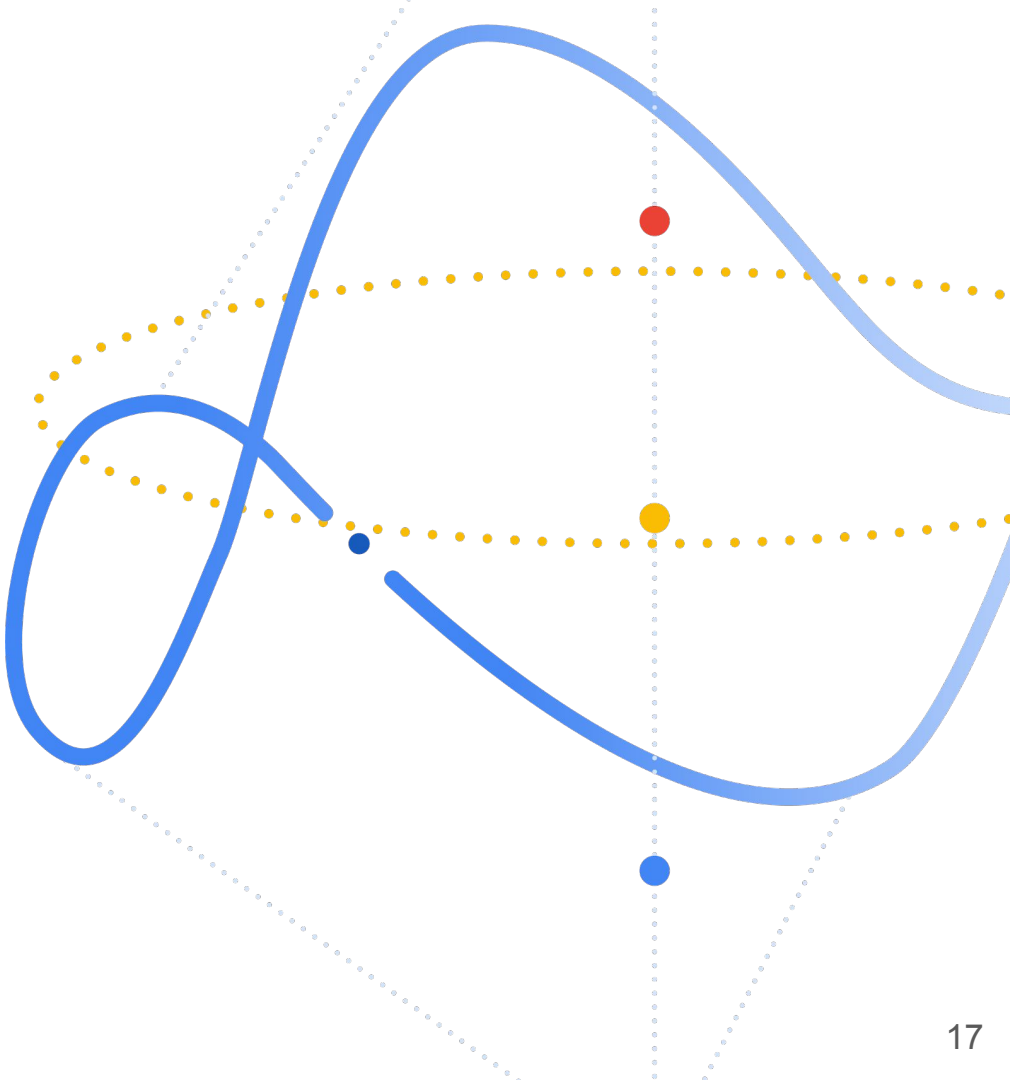
- Loop ordering: input/output/weight-stationary

Optimal tiling and scheduling found via simple Design Space Exploration

- Exploiting on-chip buffer
- Optimizing reuse



# Evaluation



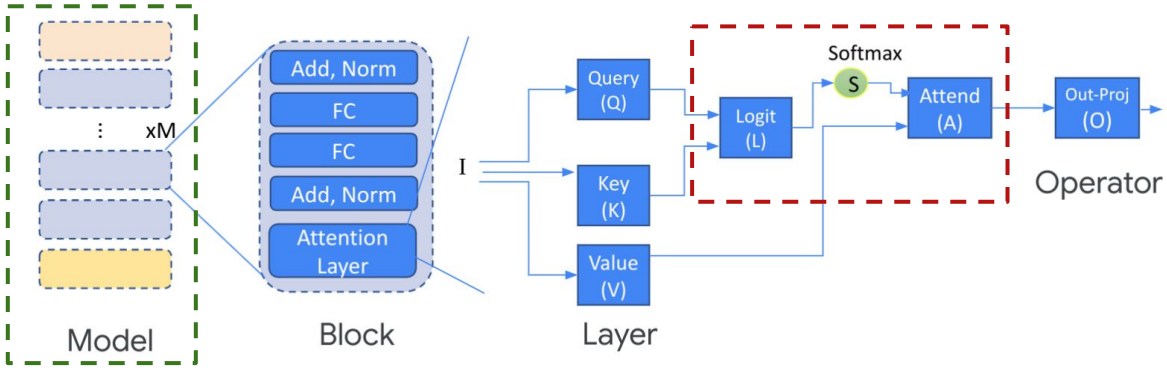
# Methodology

Progressive Evaluation:

Logit-Attend (L-A) → Model

Performance Modeling:

*https://github.com/maestro-project/frame*




## Dataflows Compared

1. Naive — Naive dataflow: Fixed tile size, schedule
2. FLEX — Optimal dataflow found via DSE, **no** fusion
3. **FLAT** — Optimal FLAT dataflow (fusion, tiling, scheduling) found via DSE

https://github.com/maestro-project/frame

# Dataflow Comparisons

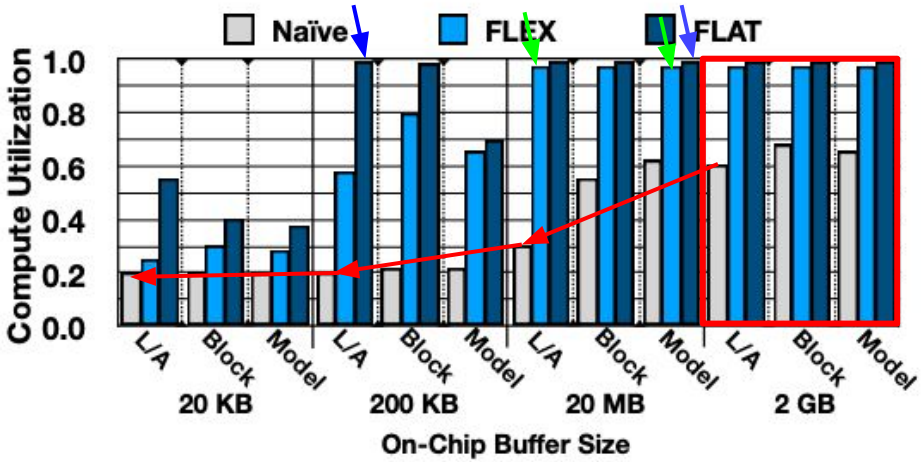
## Performance metric

- Compute utilization

$$Util = \frac{Runtime_{ideal}}{Runtime_{actual}}$$

Capturing time ratio that compute is not idling for memory

Sequence Length: 512




Given unlimited on-chip buffer, Naive dataflow can also works fine

When on-chip buffer become limited, performance goes down bc memory-boundedness

For L/A, FLEX (optimal baseline) needs **10x** more on-chip buffer to achieve close to 1.0 utilization

Model-wise, amortized by other operators, FLEX and FLAT are similar

https://github.com/maestro-project/frame

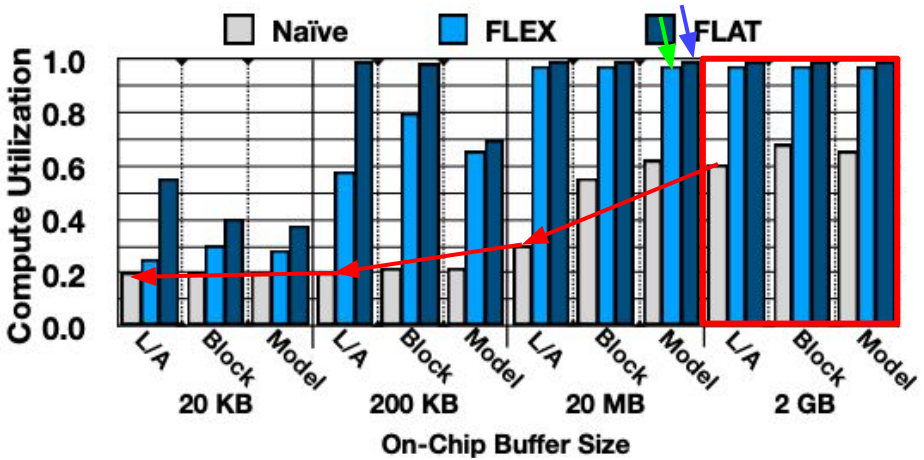
# Dataflow Comparisons

At >4K seq length, L/A becomes dominant

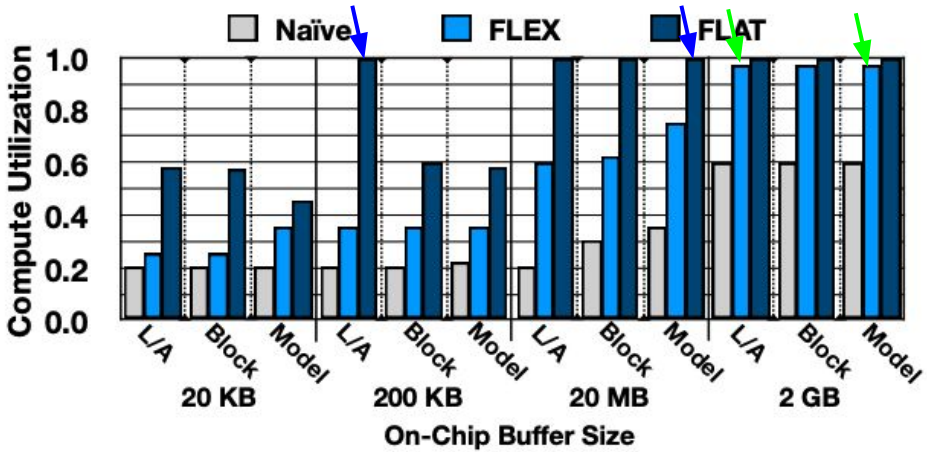
Model-wise, FLEX (optimal baseline) needs **2 order of magnitude** more buffer to achieve similar performance to FLAT

While for L,A, FLEX needs **4 order of magnitude** more buffer to achieve similar performance to FLAT

Sequence Length: 512




Sequence Length: 4K



# Memory Benefit

Platform: Evaluating using the configuration of Tesla T4 GPU

Dataset: TrEMBL protein sequencing dataset (require long sequence)

Model: Bert-based model with configurable number of blocks

| | Memory Req.(GB) | Number of Attention Blocks (Sequence Length=8K) | | | | | |
|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 |
| (Naive) | Baseline | 4.6 | 9.1 | 13.7 | 18.2 -OOM | 22.2 -OOM | 27.3 -OOM |
| | FLAT | 0.9 | 1.8 | 2.7 | 3.5 | 4.4 | 5.3 |

| | Memory Req.(GB) | Number of Attention Blocks (Sequence Length=16K) | | | | | |
|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 |
| (Naive) | Baseline | 17.5 -OOM | 35 -OOM | 52.5 -OOM | 70.0 -OOM | 87.5 -OOM | 105.0 -OOM |
| | FLAT | 2.8 | 5.6 | 8.5 | 11.3 | 14.1 | 16.9 -OOM |

Seq 8K, baseline can only launches 3 attention blocks before OOM, while FLAT can fit > 6 blocks

Seq 16K, baseline gets OOM with 1 attention block, while FLAT can fit 5 blocks

https://github.com/maestro-project/frame

# FLAT with SOTA Sparse Techniques

Platform: Evaluating using the configuration of TPU-v3

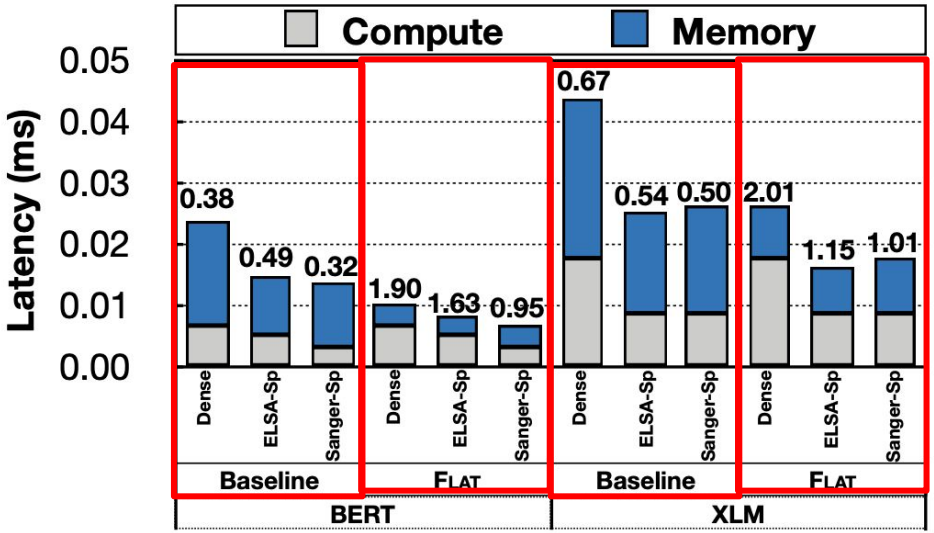
Model: BERT, XLM

Sparse Techniques:

- ELSA (ISCA'21)
- Sanger (MICRO'21)

Performance metric:

- Latency
- Compute/ Memory Ratio
  - < 1.0: memory-bound




Both ELSA and Sanger improves latency

FLAT further improves the runtime performance by **~1.7x**

FLAT increase the compute/memory ratio to close or over 1.0 → eliminate memory bottleneck

https://github.com/felix0901/flat_prototype

# FLAT Prototyping

Platform: Evaluating on Tesla T4 GPU
*(16GM memory)*

Prototyping Operator: Fused Logit-Attend using FLAT

Language: Jax
*https://github.com/felix0901/flat_prototype*

| Runtime (ms) | Batch Size (Sequence Length=256) | | | | | | |
|---|---|---|---|---|---|---|---|
| | 1 | 16 | 64 | 128 | 256 | 1K | 2K |
| (Naive) Baseline | 36 | 630 | 2,520 | 5,230 | OOM | OOM | OOM |
| FLAT | 28 | 480 | 1,870 | 3,740 | 7,560 | 34,010 | OOM |

Maximum batch size

| Runtime (ms) | Sequence Length (Batch Size=1) | | | | | | |
|---|---|---|---|---|---|---|---|
| | 128 | 512 | 2K | 4K | 16K | 64K | 128K |
| (Naive) Baseline | 12 | 74 | 697 | OOM | OOM | OOM | OOM |
| FLAT | 11 | 43 | 175 | 424 | 4,599 | 64,350 | OOM |

Maximum sequence length

FLAT accommodate **8x larger batch size** in memory over baseline

FLAT fits **32x longer sequences** over baseline

FLAT has on average **1.5x speedup** over baseline

# Conclusions

https://github.com/maestro-project/frame
https://github.com/felix0901/flat_prototype

The quadratic complexity of Logit and Attend operator in Attention layer causing two major challenges:

1. Low performance from memory boundedness
2. Large on-chip buffer requirement for staging intermediate activations

FLAT fused Logit and Attend operators and optimized tiling and scheduling

1. Increased the operation intensity → ameliorate the memory-boundedness
2. Reduced the on-chip buffer requirement for data staging

FLAT delivered

1. **1.5x speedup** (on average) in our performance model across models & platforms
2. **1.7x extra speedup** (on average) on top of ELSA and Sanger
3. **1.5x speedup** (on average) in our Jax prototype
   while enabling **8x larger** batch size or **32x larger** sequence length

Thank you!

felixkao.gatech@gmail.com