Towards Ubiquitous Edge Intelligence: Efficient ML Algorithm and Hardware Co-Design

Presenter: Haoran You   Advisor: Yingyan Lin

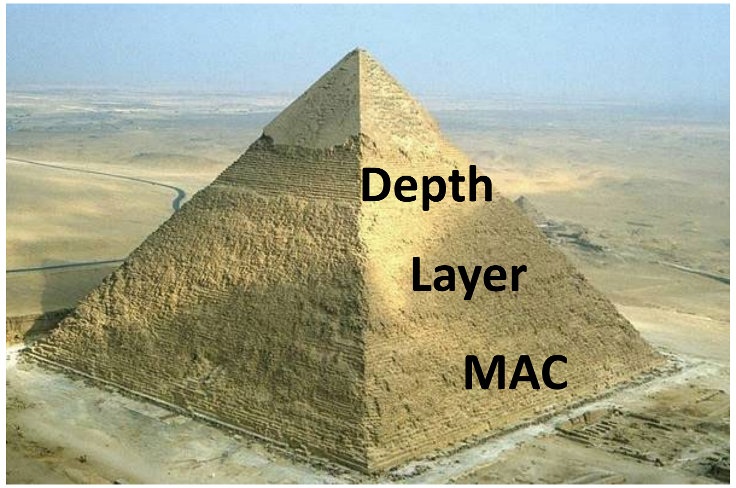
Georgia Institute of Technology

Efficient and Intelligent Computing Lab

Georgia Tech School of Computer Science College of Computing

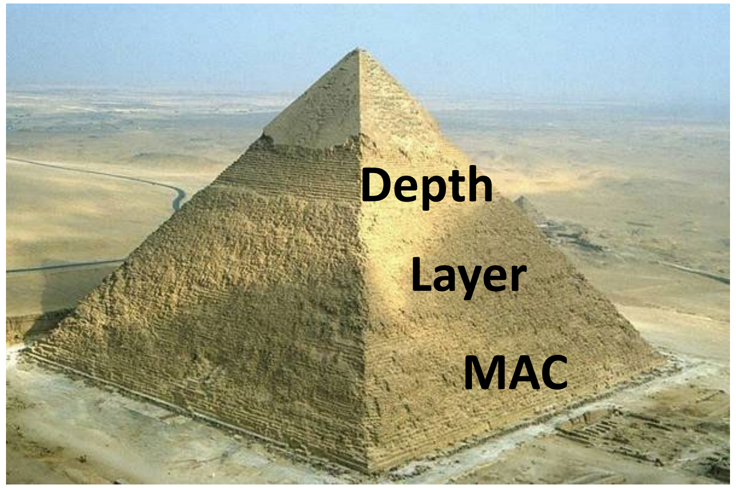
Research Project Summary

Energy Pyramid of DNNs



Research Project Summary

Energy Pyramid of DNNs

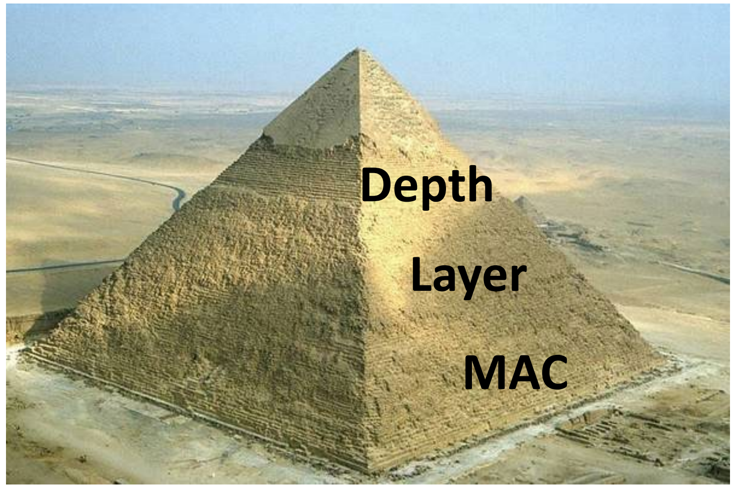


Algorithm

Pruning
Quantization
Efficient Training & Inference
NAS
ShiftAddNet
...

Research Project Summary

Energy Pyramid of DNNs



Algorithm-Hardware Co-Design

Algorithm

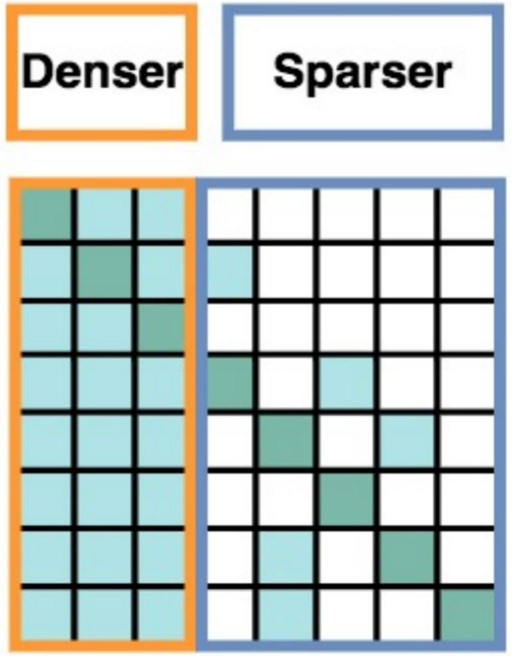
Pruning
Quantization
Efficient Training & Inference
NAS
ShiftAddNet
...

Hardware

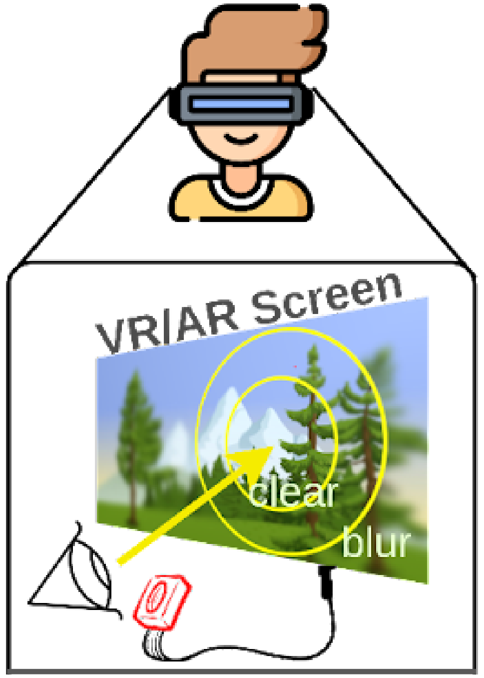
Accelerator
Scheduling
Micro-architecture
Memory Hierarchy
Dataflow & Mapping
...

Table of Content

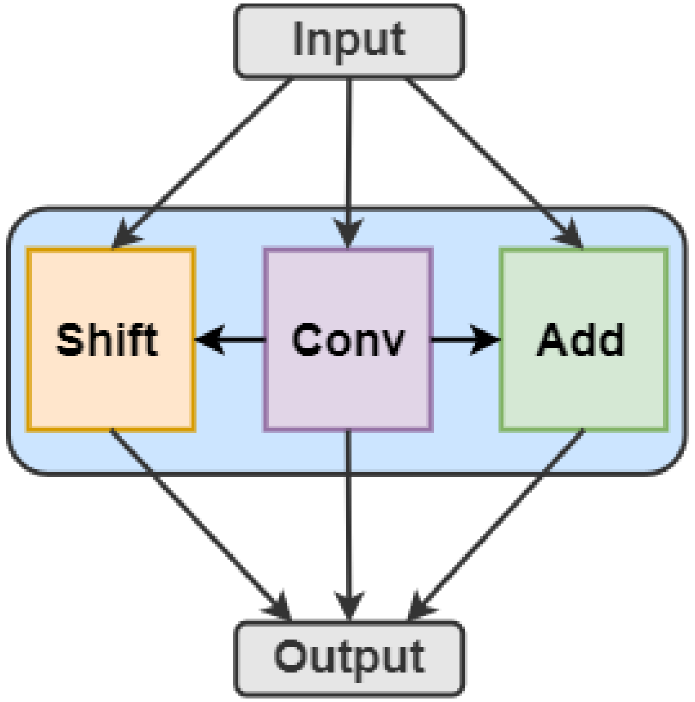
- **Algorithm-Hardware Co-Design**



ViTCoD [HPCA'23]



EyeCoD [ISCA'22]



ShiftAddNAS [ICML'22]
& NASA [ICCAD'22]

ViTCoD: Vision Transformer Acceleration via Dedicated Algorithm and Accelerator Co-Design

Haoran You[1], Zhanyi Sun[2], Huihong Shi[1], Zhongzhi Yu[1], Yang Zhao[2], Yongan Zhang[1], Chaojian Li[1], Baopu Li[3], and Yingyan Lin[1]

[1]Georgia Institute of Technology

[2]Rice University

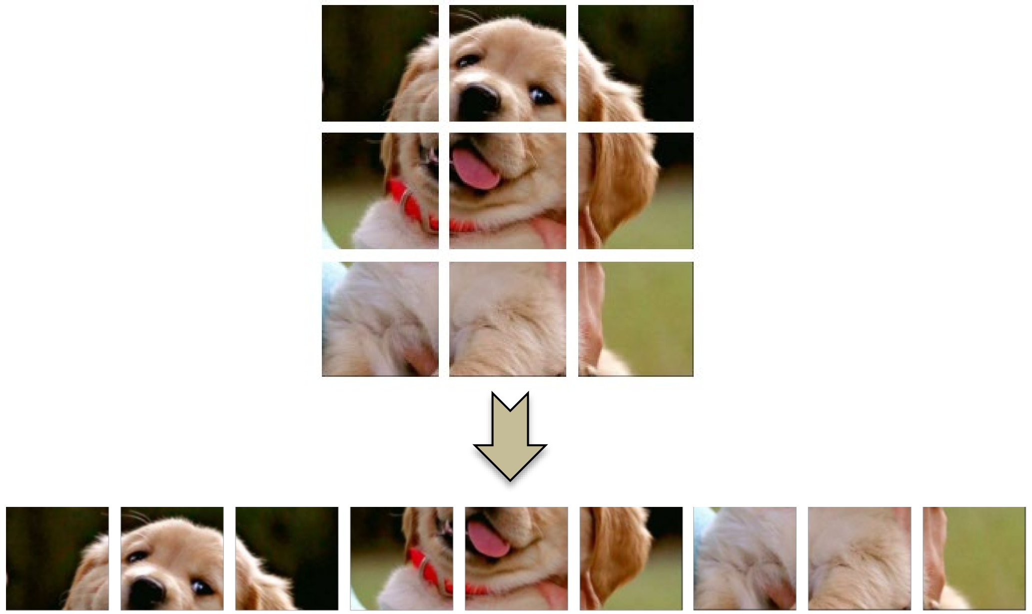
[3]Oracle Health and AI

The 29th IEEE International Symposium on High-Performance Computer Architecture (HPCA 2023)

Efficient and Intelligent Computing Lab

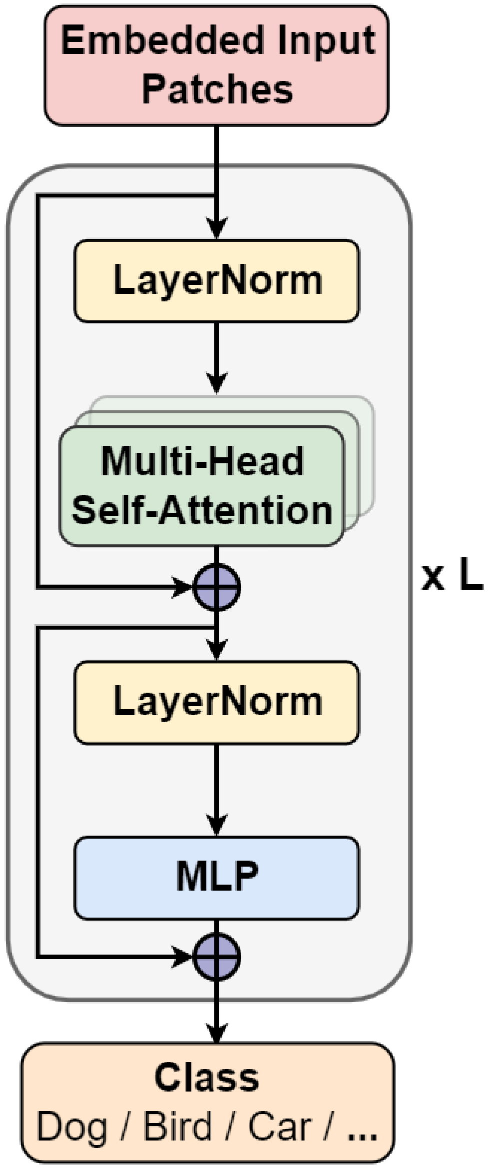
Background of Vision Transformer (ViTs)

- ViTs achieve SOTA performance on various vision tasks
  - Input: 2D image → input tokens/patches



Input Tokens

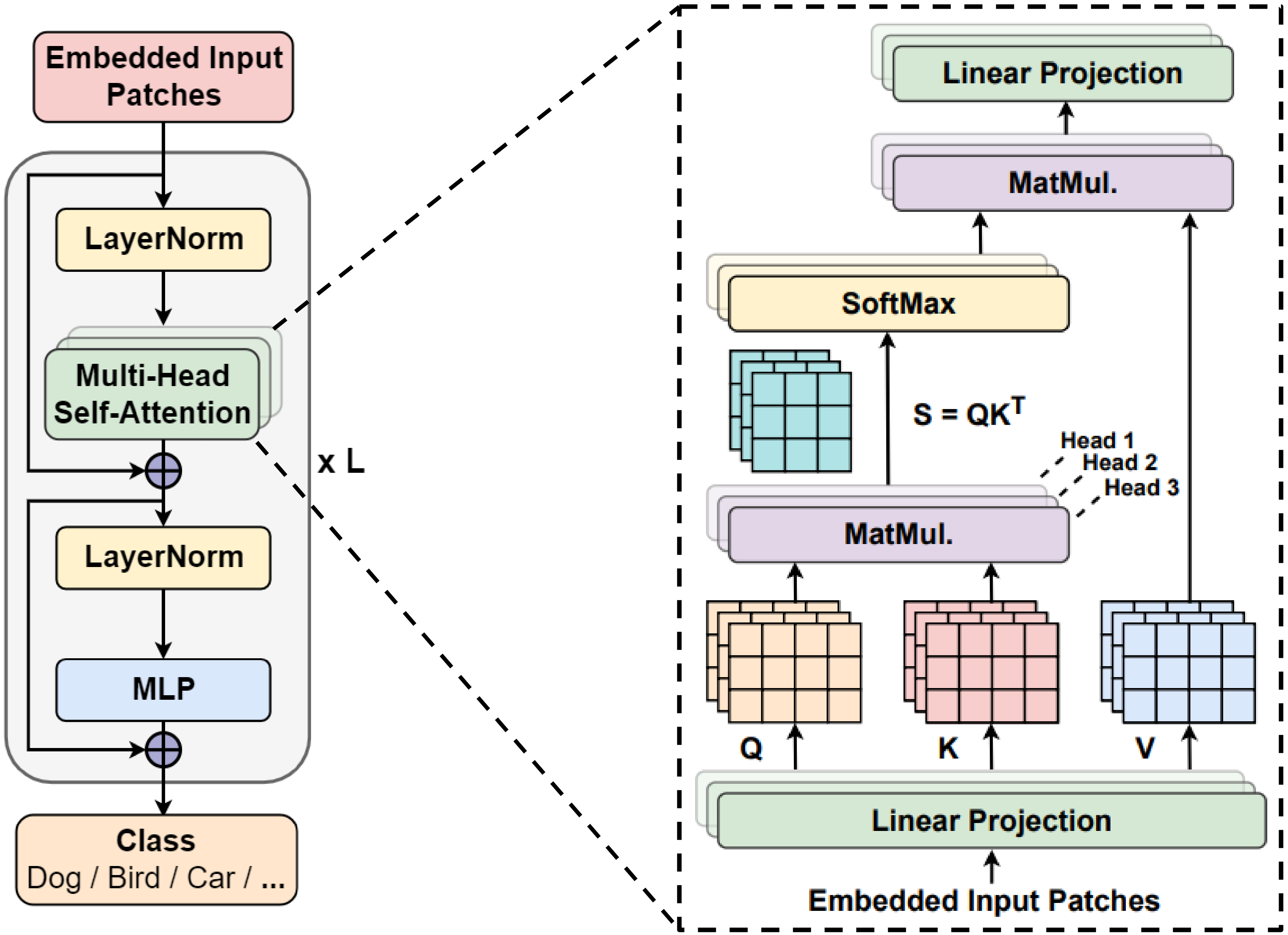
  - Core Model: Self-Attention and MLP



ViT Models

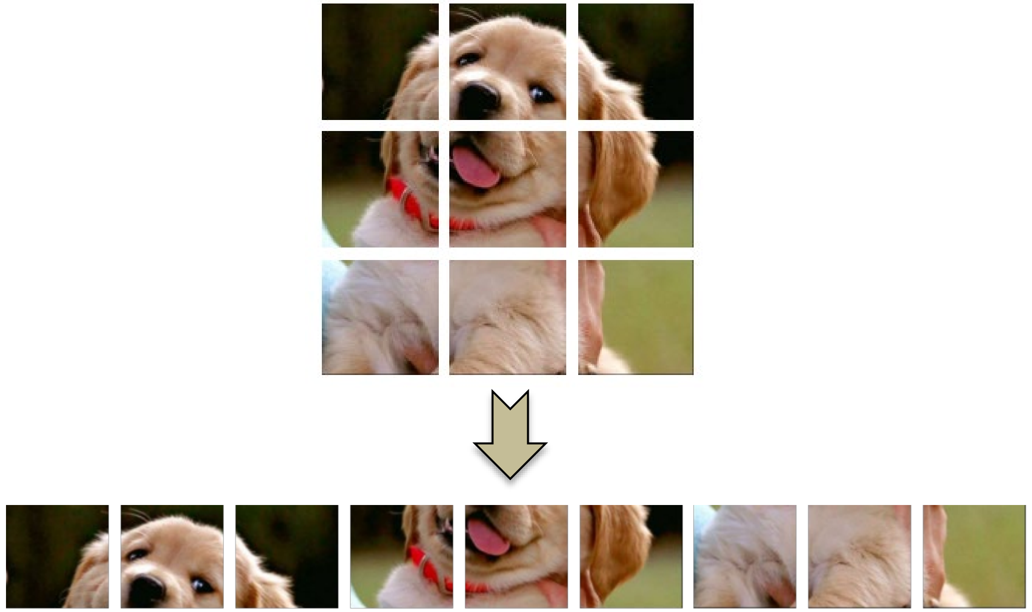
Background of Vision Transformer (ViTs)

- ViTs achieve **SOTA performance** on various vision tasks



Background of Vision Transformer (ViTs)

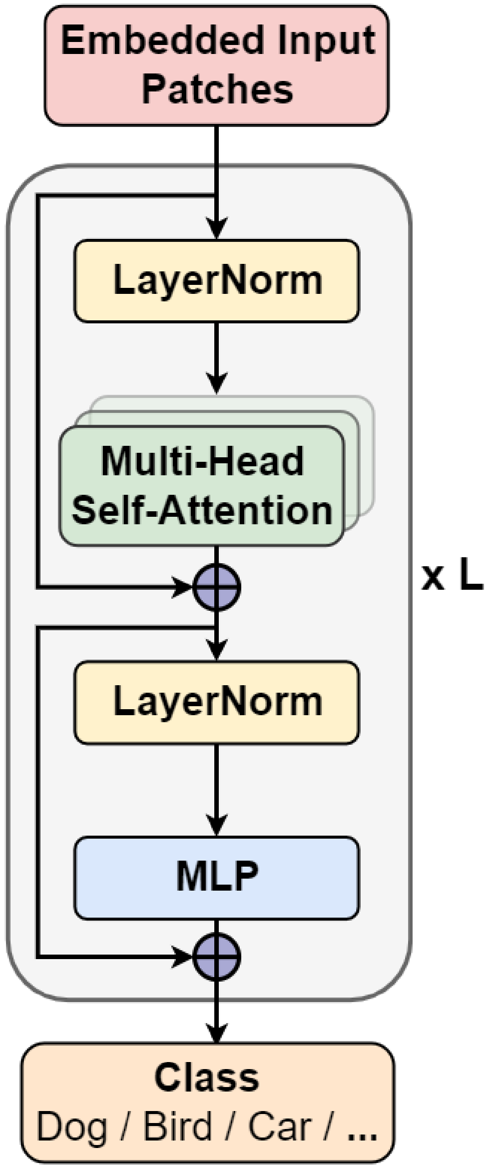
- ViTs achieve SOTA performance on various vision tasks
  - Input: 2D image → input tokens



Input Tokens

  - Core Model: Self-Attention and MLP

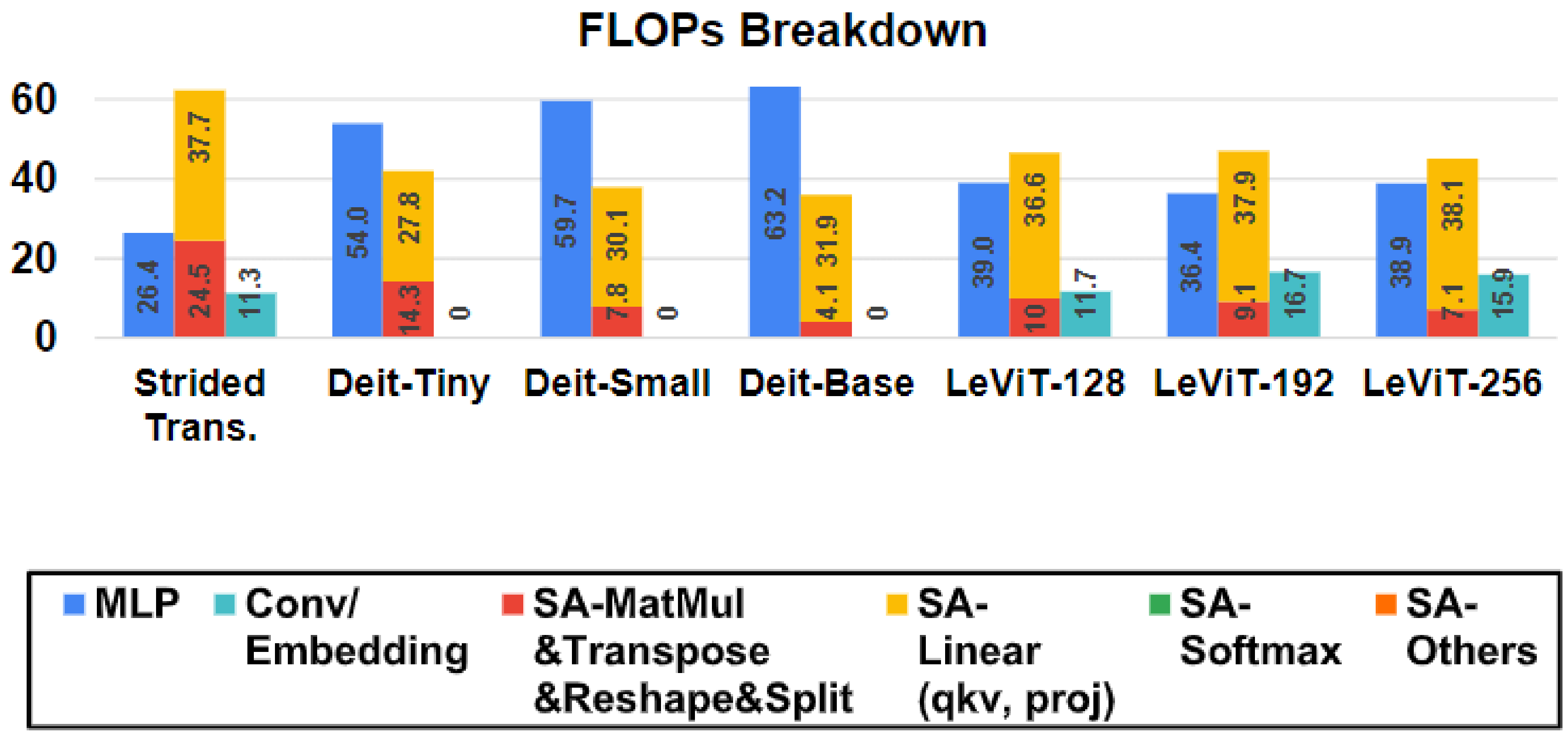
- But ViTs still require a high computational cost as compared to convolutional networks (CNNs)



ViT Models

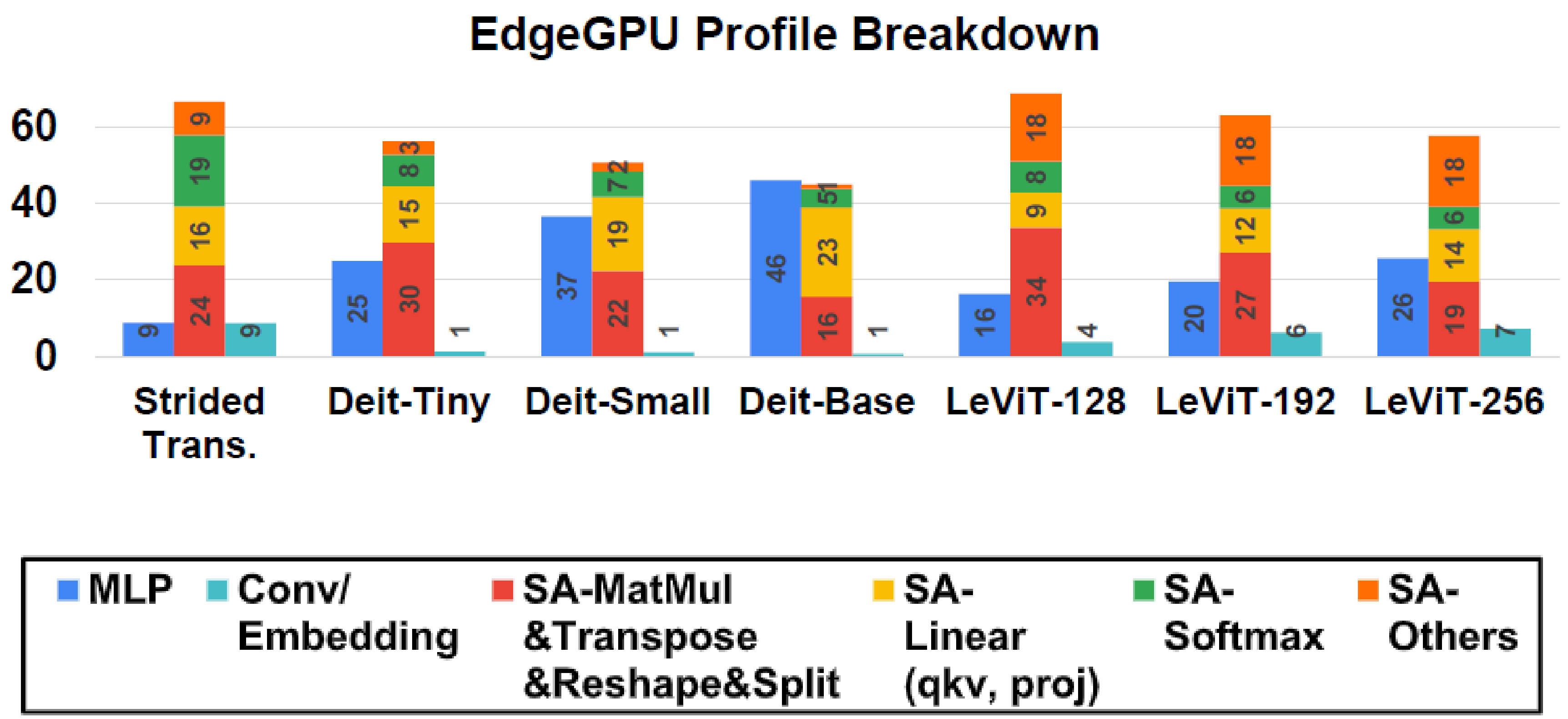
What are the Bottlenecks in ViTs?

- **The bottleneck is the self-attention module**
  - **We profile seven ViT models to show the breakdown**
    - In terms of FLOPs, self-attention is not as dominant as MLPs



What are the Bottlenecks in ViTs?

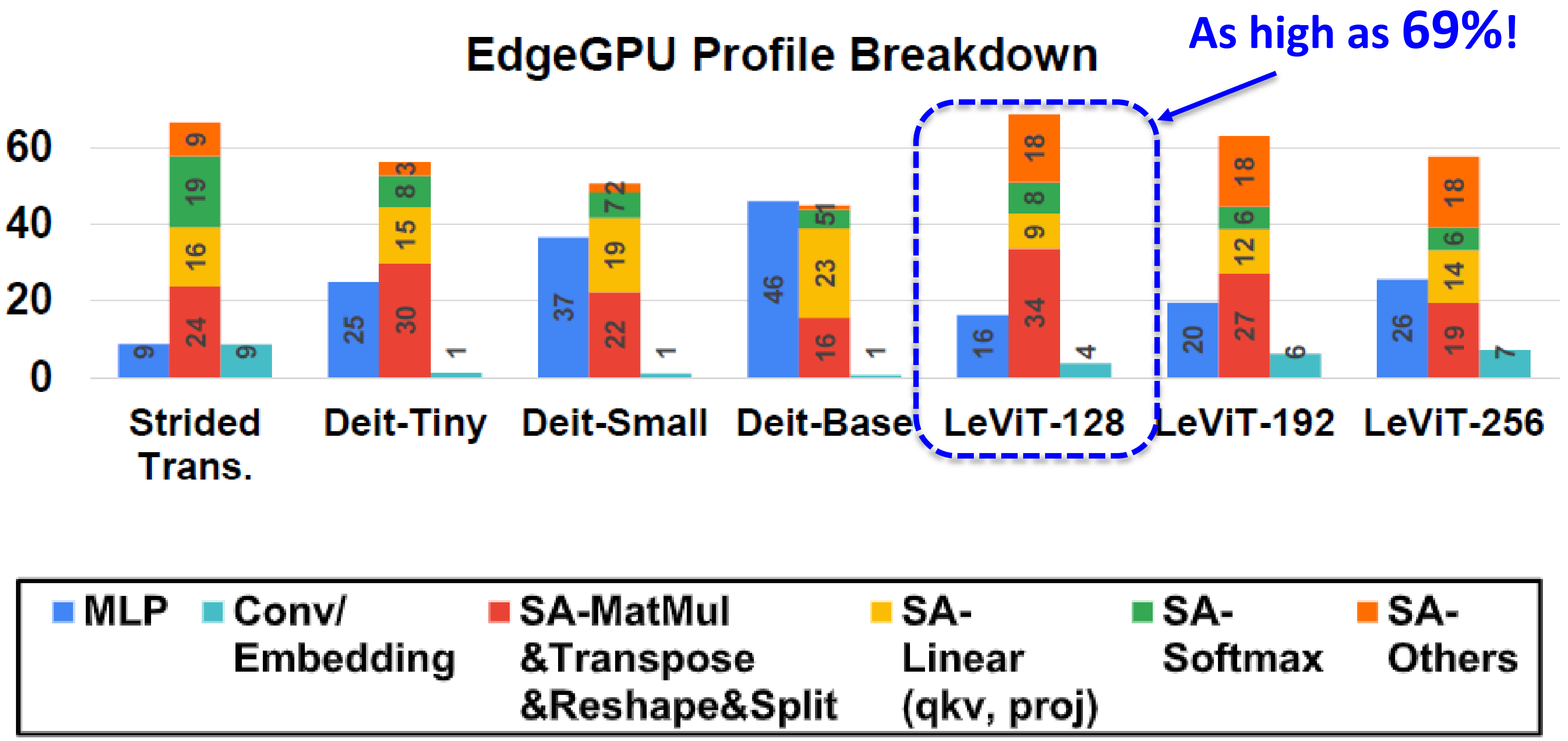
- **The bottleneck is the self-attention module**
  - **We profile seven ViT models to show the breakdown**
    - In terms of FLOPs, self-attention is not as dominant as MLPs
    - In terms of real Latency, it consistently accounts **for over 50% latency**



EdgeGPU: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-tx2/

What are the Bottlenecks in ViTs?

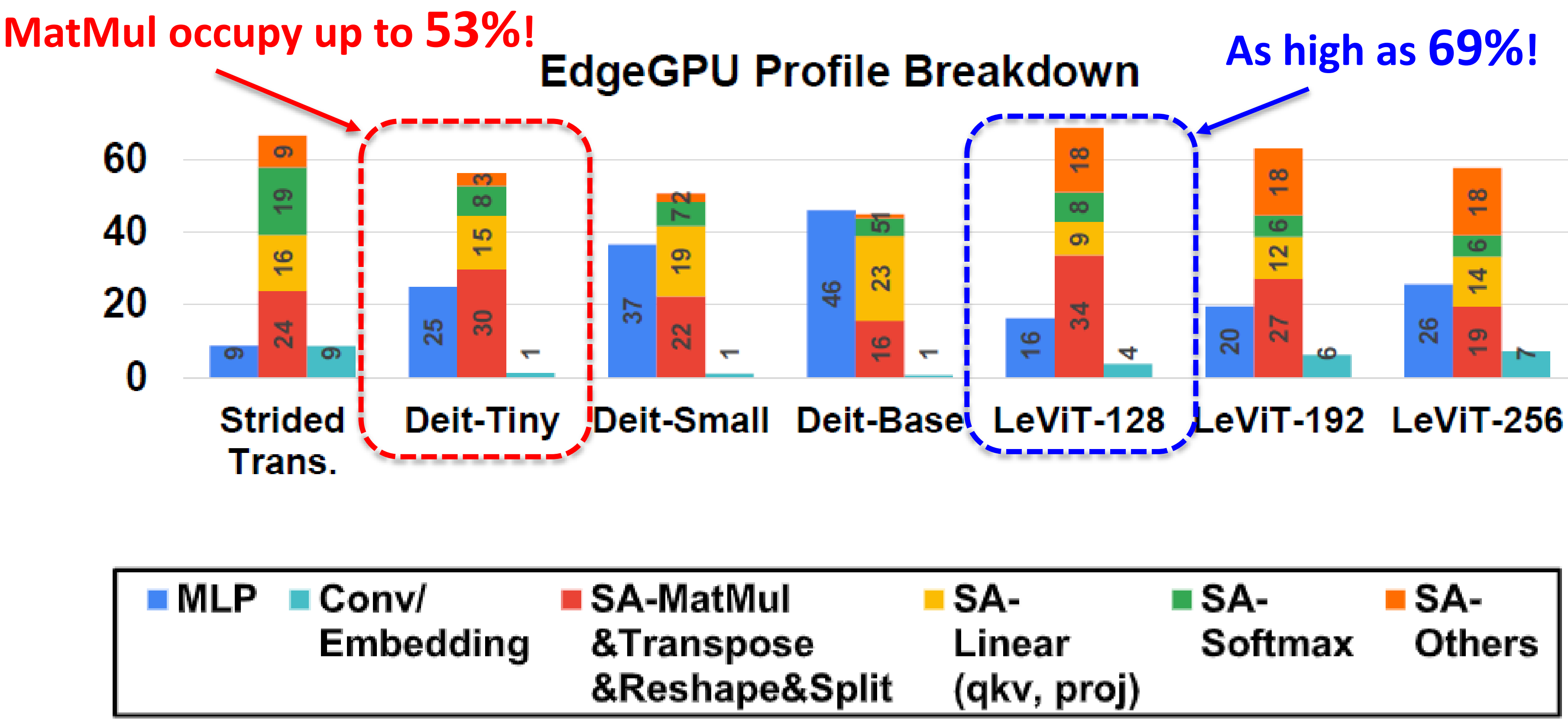
- **The bottleneck is the self-attention module**
  - **We profile seven ViT models to show the breakdown**
    - In terms of FLOPs, self-attention is not as dominant as MLPs
    - In terms of real Latency, it consistently accounts **for over 50% latency**



EdgeGPU: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-tx2/

What are the Bottlenecks in ViTs?

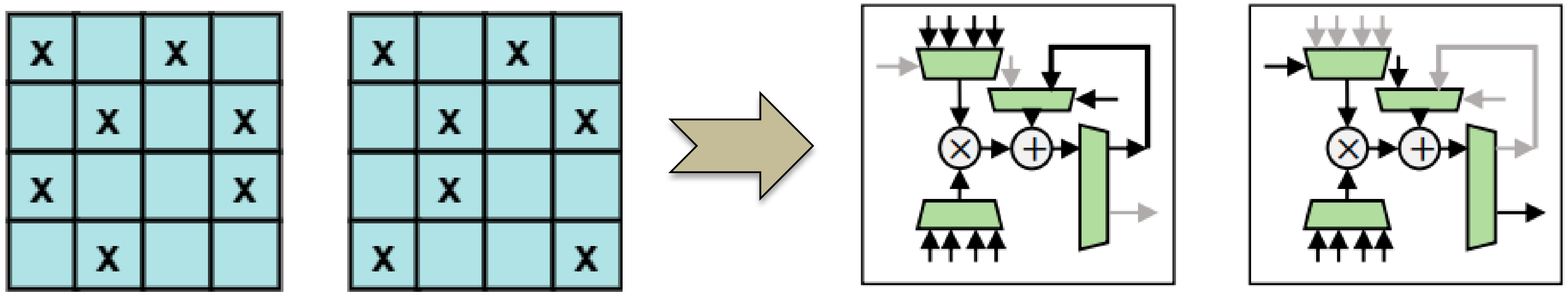
- **The bottleneck is the self-attention module**
  - **We profile seven ViT models to show the breakdown**
    - In terms of FLOPs, self-attention is not as dominant as MLPs
    - In terms of real Latency, it consistently accounts **for over 50% latency**



EdgeGPU: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-tx2/

Can Previous Attention Accelerators Help?

- **The bottleneck is the self-attention module**
  - **We profile seven ViT models to show the breakdown**
    - In terms of FLOPs, self-attention is not as dominant as MLPs
    - In terms of real Latency, it consistently accounts **for over 50% latency**
  - **Can we use previous sparse attention accelerator to handle it?**
    - No, they are dedicated to NLP Transformers



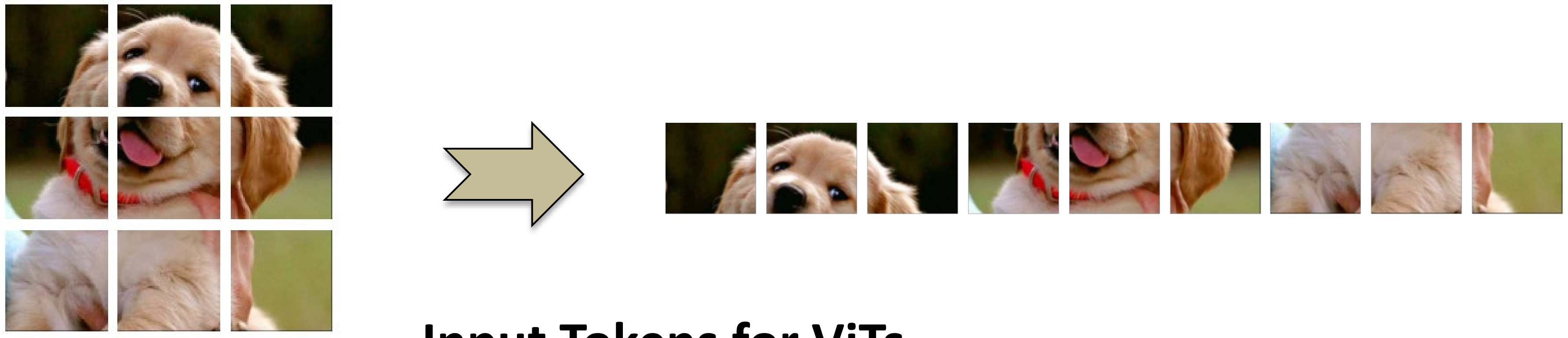
Dynamic Sparsity Patterns for Different Inputs

Reconfigurable Architecture E.g., Sanger [1], DOTA [2], etc

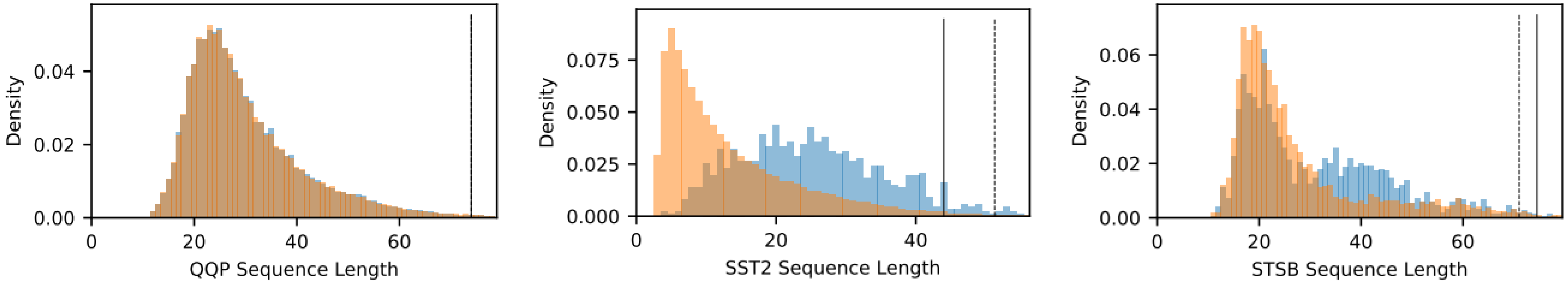
[1] Sanger: A Co-Design Framework for Enabling Sparse Attention using Reconfigurable Architecture, MICRO 2021
[2] DOTA: detect and omit weak attentions for scalable transformer acceleration, ASPLOS 2022

Attention in ViTs and NLP Transformers

- **Comparison of self-attentions in ViTs and NLP Transformers**
  - **Difference 1**:
    Fixed number of input tokens vs. dynamic number of input tokens



Input Tokens for ViTs

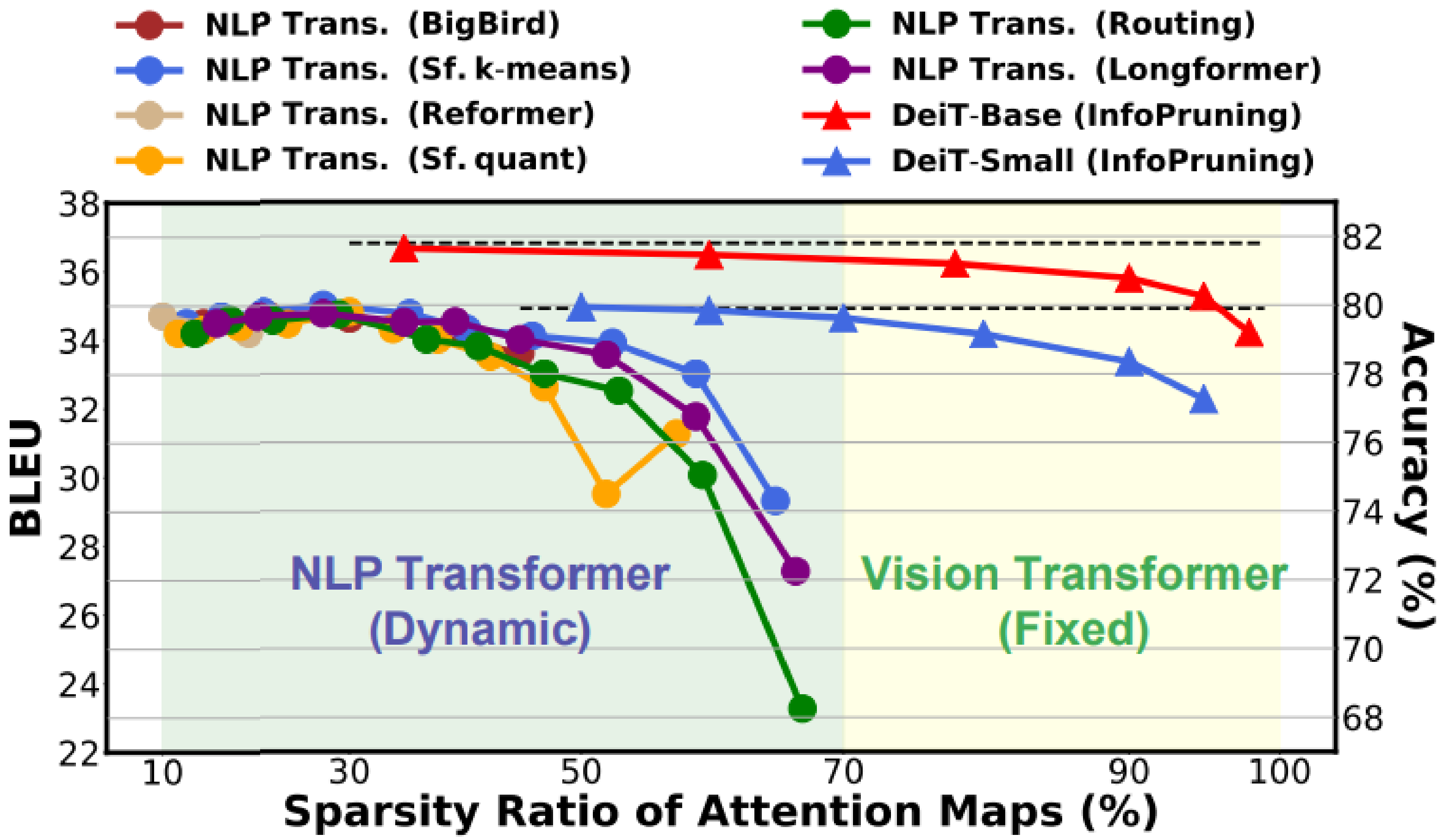


Input Tokens for NLP Transformer [1]

[1] Learned Token Pruning for Transformers, KDD 2022

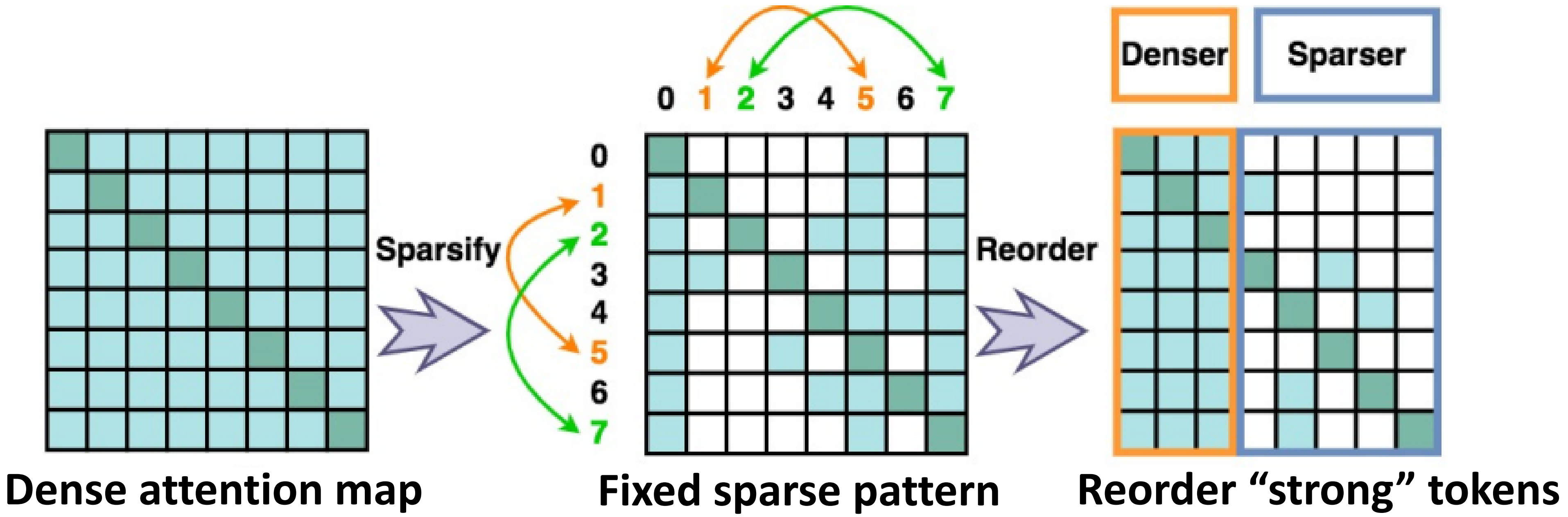
Attention in ViTs and NLP Transformers

- **Comparison of self-attentions in ViTs and NLP Transformers**
  - **Difference 2**:
    Up to 90% sparsity in ViTs' attention maps vs. 50% ~ 60% in NLP Transformer's attention maps



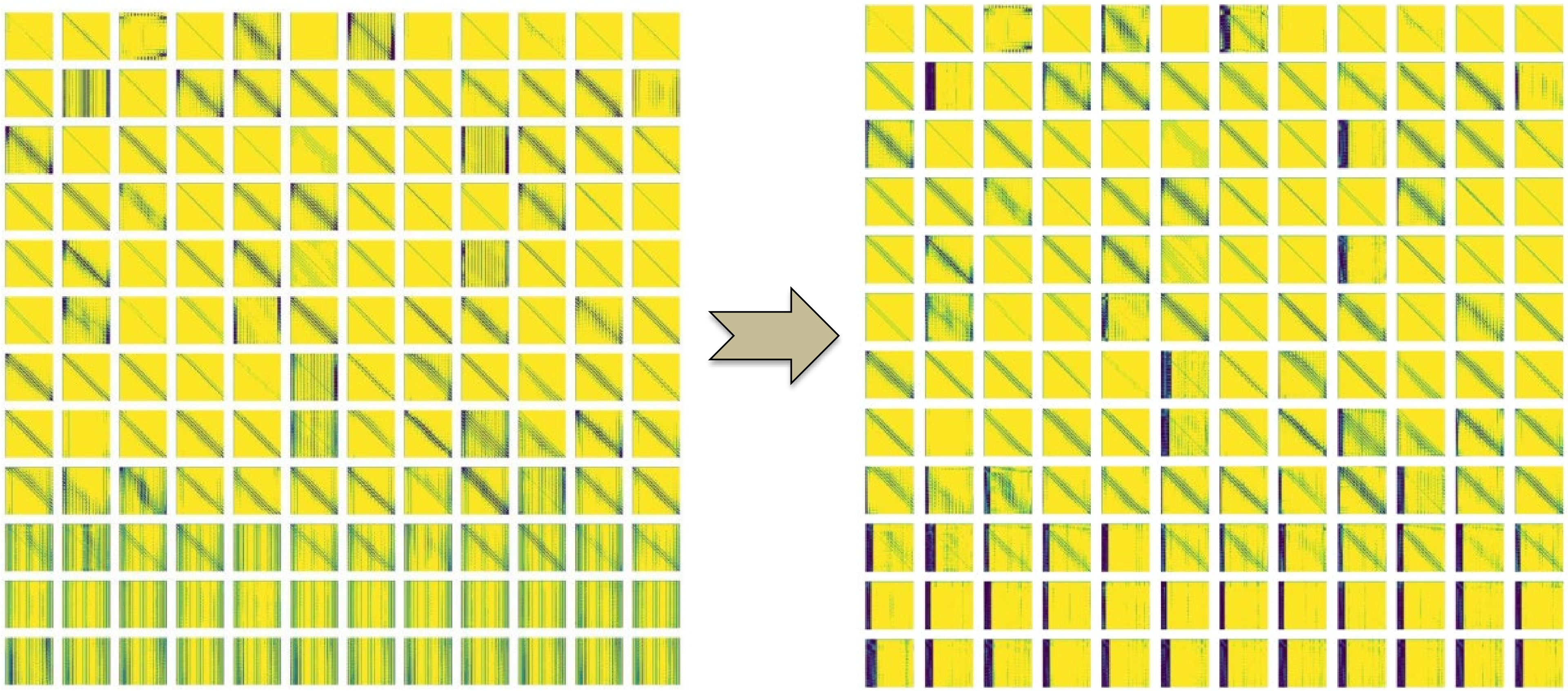
Challenges and Unexplored Opportunities for ViTs?

- **Challenge 1: Accelerate ViTs w/o on-the-fly reconfiguration?**
  - **Opportunity 1: Fixed attention sparse patterns in ViTs**
    - ☑ Fixed sparse patterns and thus stationary data accesses
    - ☑ Strong "tokens"



Challenges and Unexplored Opportunities for ViTs?

- **Challenge 1: Accelerate ViTs w/o on-the-fly reconfiguration?**
  - **Opportunity 1: Fixed attention sparse patterns in ViTs**
    - ☑ Fixed sparse patterns and thus stationary data accesses
    - ☑ Strong "tokens"

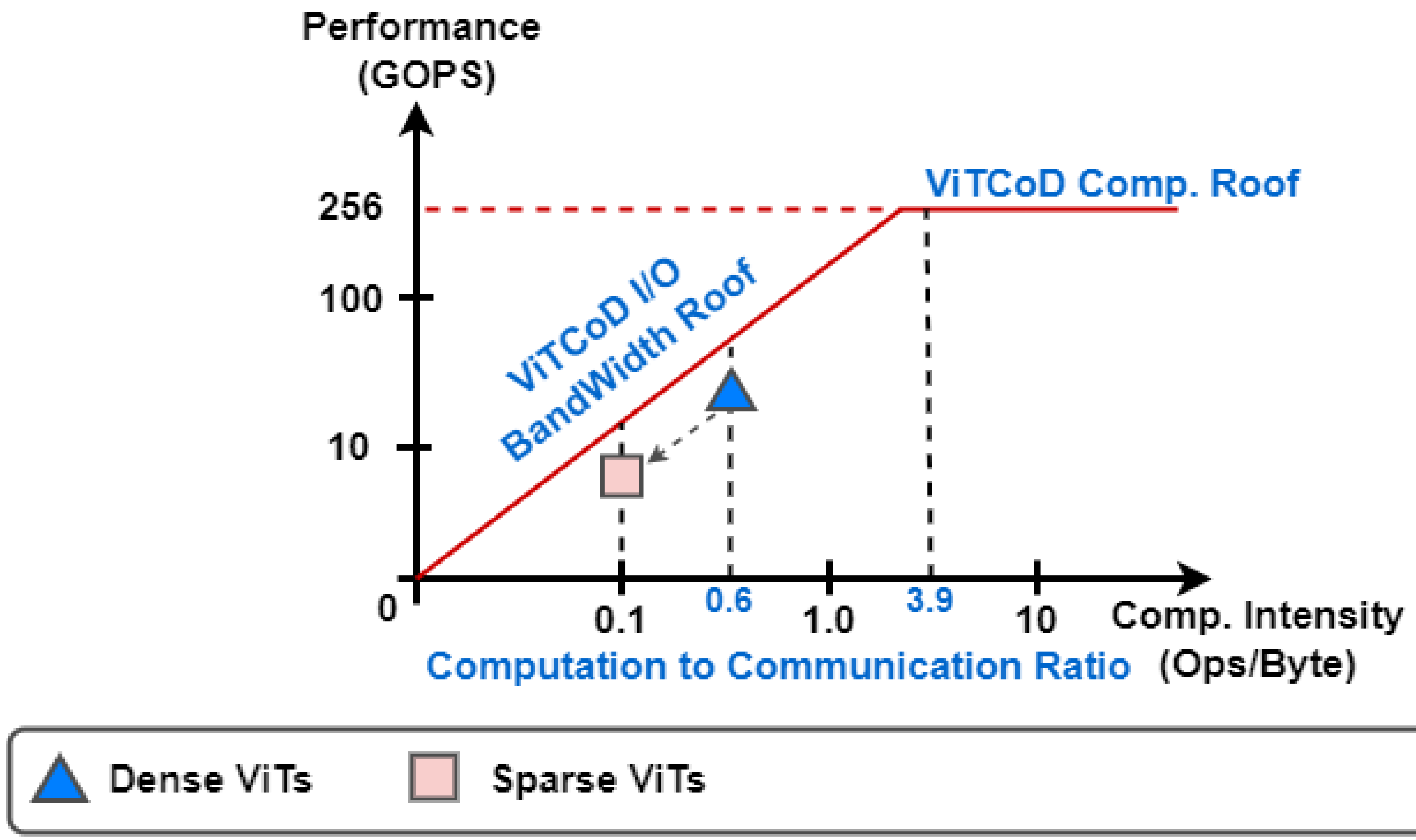


Fixed sparse pattern **Reorder "strong" tokens**

Challenges and Unexplored Opportunities for ViTs?

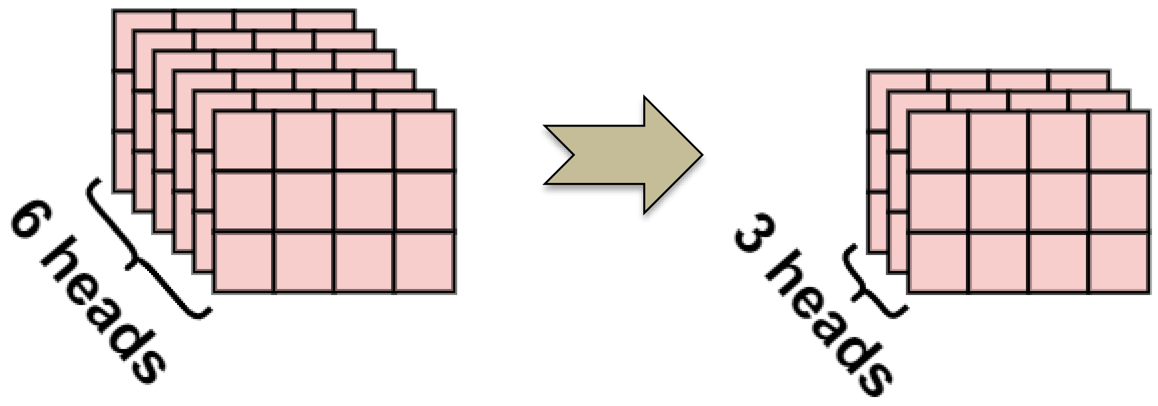
- Challenge 1: Accelerate ViTs w/o on-the-fly reconfiguration?
  - Opportunity 1: Fixed attention sparse patterns in ViTs

- Challenge 2: How to balance computations vs. data movements?
  - Sparse attention makes data movements a bigger problem



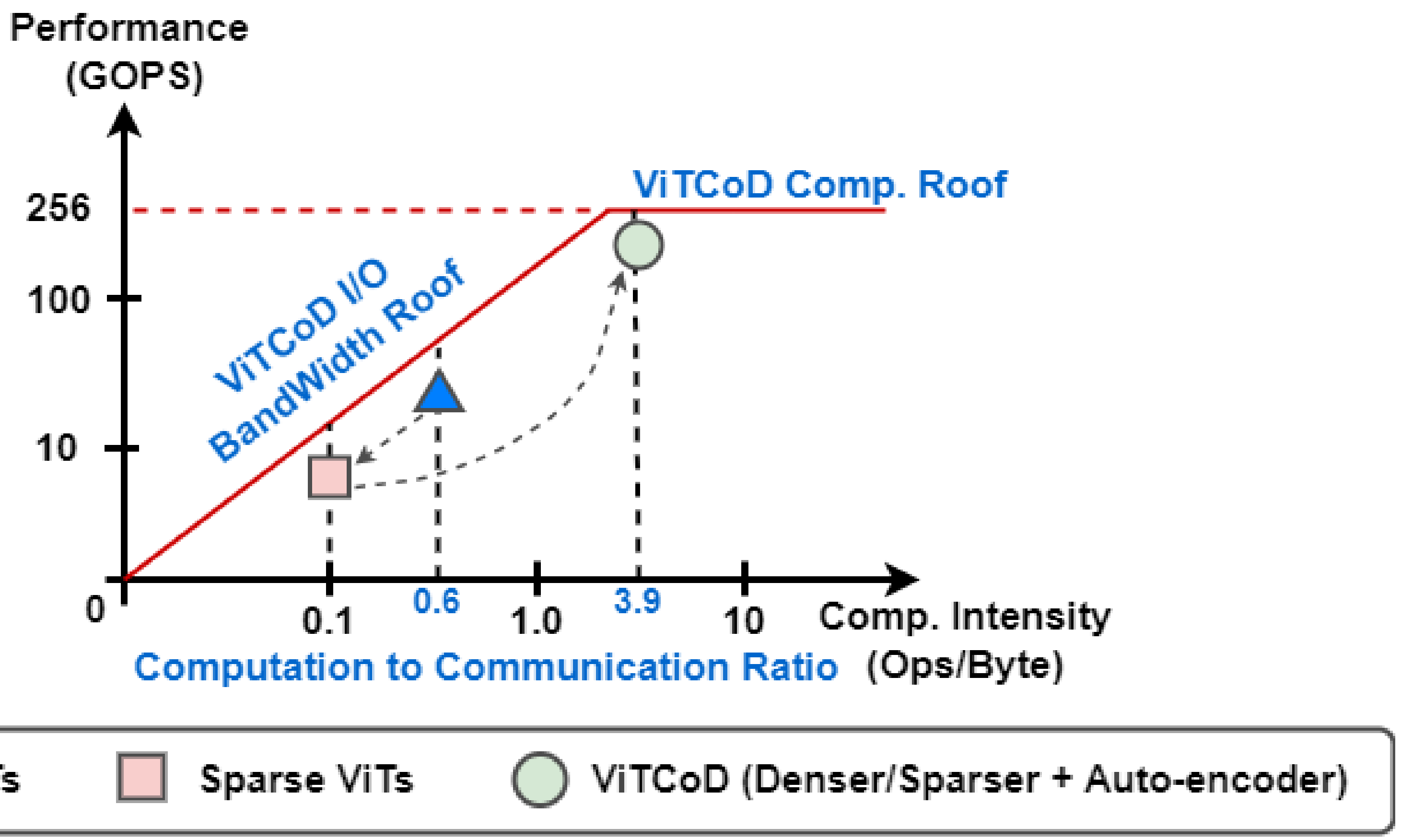
Challenges and Unexplored Opportunities for ViTs?

- Challenge 1: Accelerate ViTs w/o on-the-fly reconfiguration?
  - Opportunity 1: Fixed attention sparse patterns in ViTs
- Challenge 2: How to balance computations vs. data movements?
  - Opportunity 2: Redundancy across attention heads



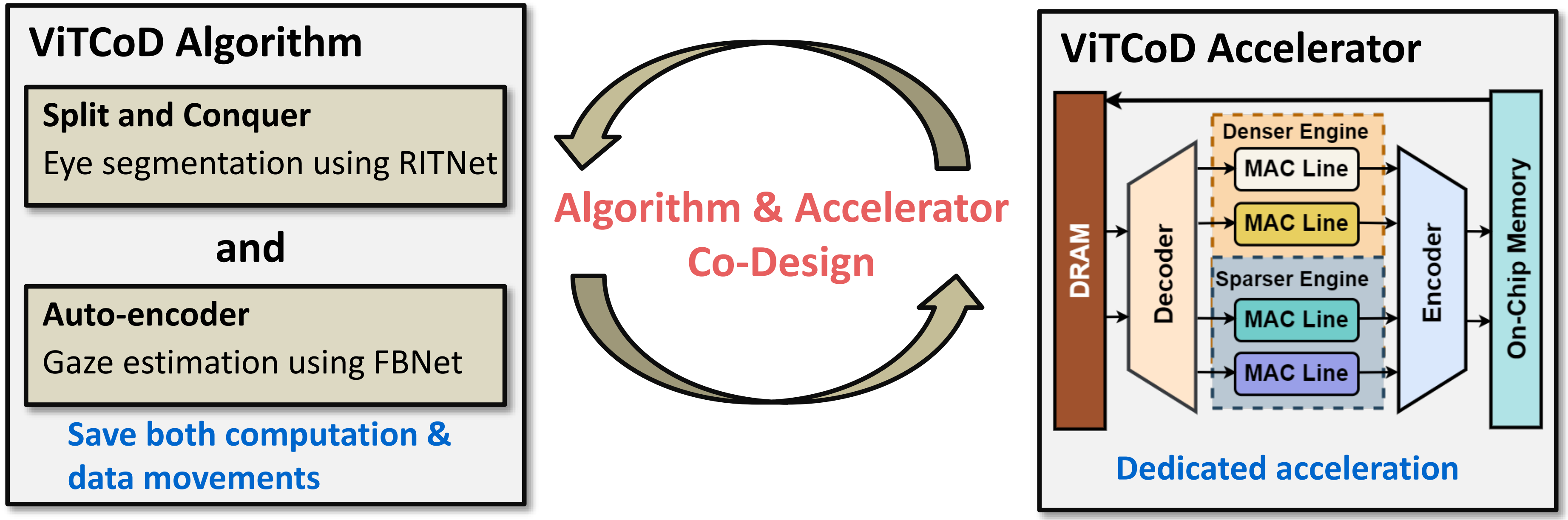
Challenges and Unexplored Opportunities for ViTs?

- Challenge 1: Accelerate ViTs w/o on-the-fly reconfiguration?
  - Opportunity 1: Fixed attention sparse patterns in ViTs
- **Challenge 2**: How to balance computations vs. data movements?
  - **Opportunity 2**: Redundancy across attention heads



Proposed ViTCoD: Algorithm & Accel. Co-Design

- **Proposed ViT algorithm & accelerator co-design (ViTCoD) for accelerating ViTs with sparse attention**
  - Split and conquer algorithm to cluster the workloads into denser/sparser
  - Auto-encoder module to compress attention heads before transmitting

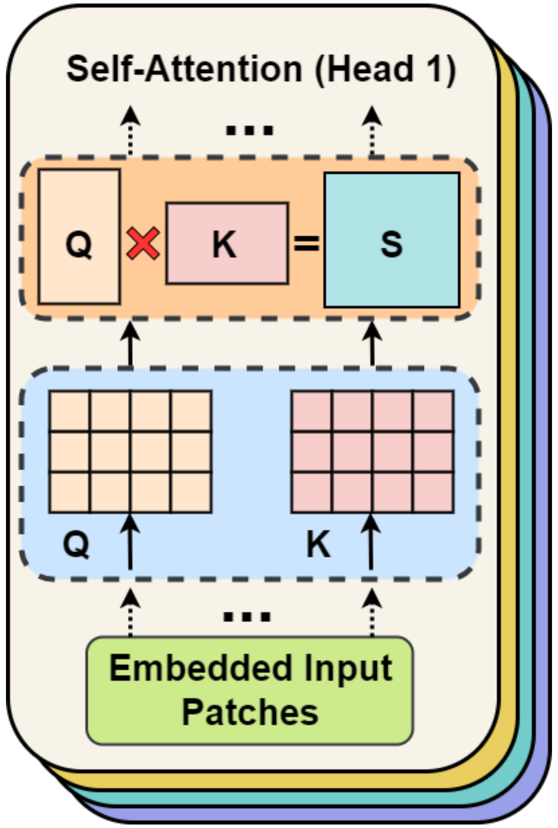


Our Overall Contributions in ViTCoD

In this work, we

- Propose the first Vision Transformer algorithm & accelerator co-design framework, dubbed ViTCoD
- **On the algorithm level,** ViTCoD
  - prunes and polarizes the attention maps to have either denser or sparser fixed patterns for regularizing two levels of workloads
  - integrate a lightweight and learnable auto-encoder module to enable trading dominant high-cost data movements for lower-cost computations
- **On the hardware level,** ViTCoD
  - adopts a dedicated accelerator to simultaneously handle the enforced denser and sparser workloads
  - integrates on-chip encoder and decoder engines to reduce data movements

ViTCoD Overview

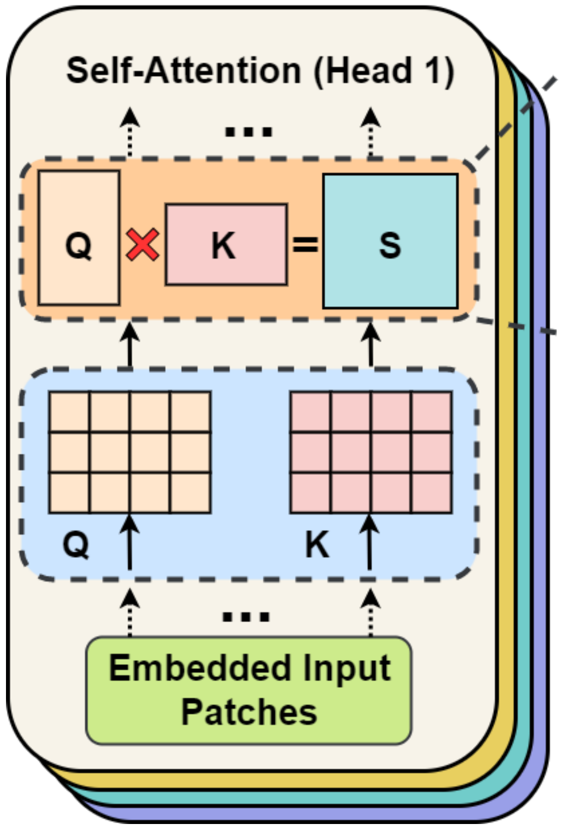


ViT Self-attention

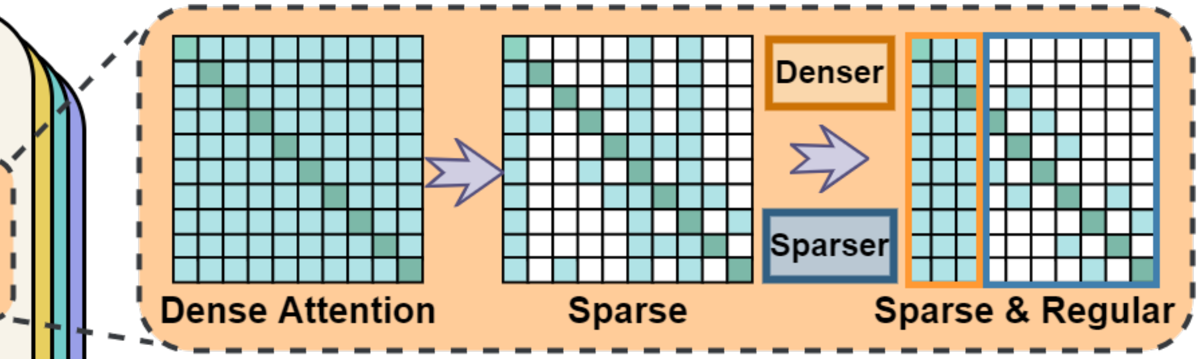
ViTCoD Algorithm:

The core idea on the algorithm level is to reduce both computations and data movements in core self-attention modules.

ViTCoD Overview



ViT Self-attention

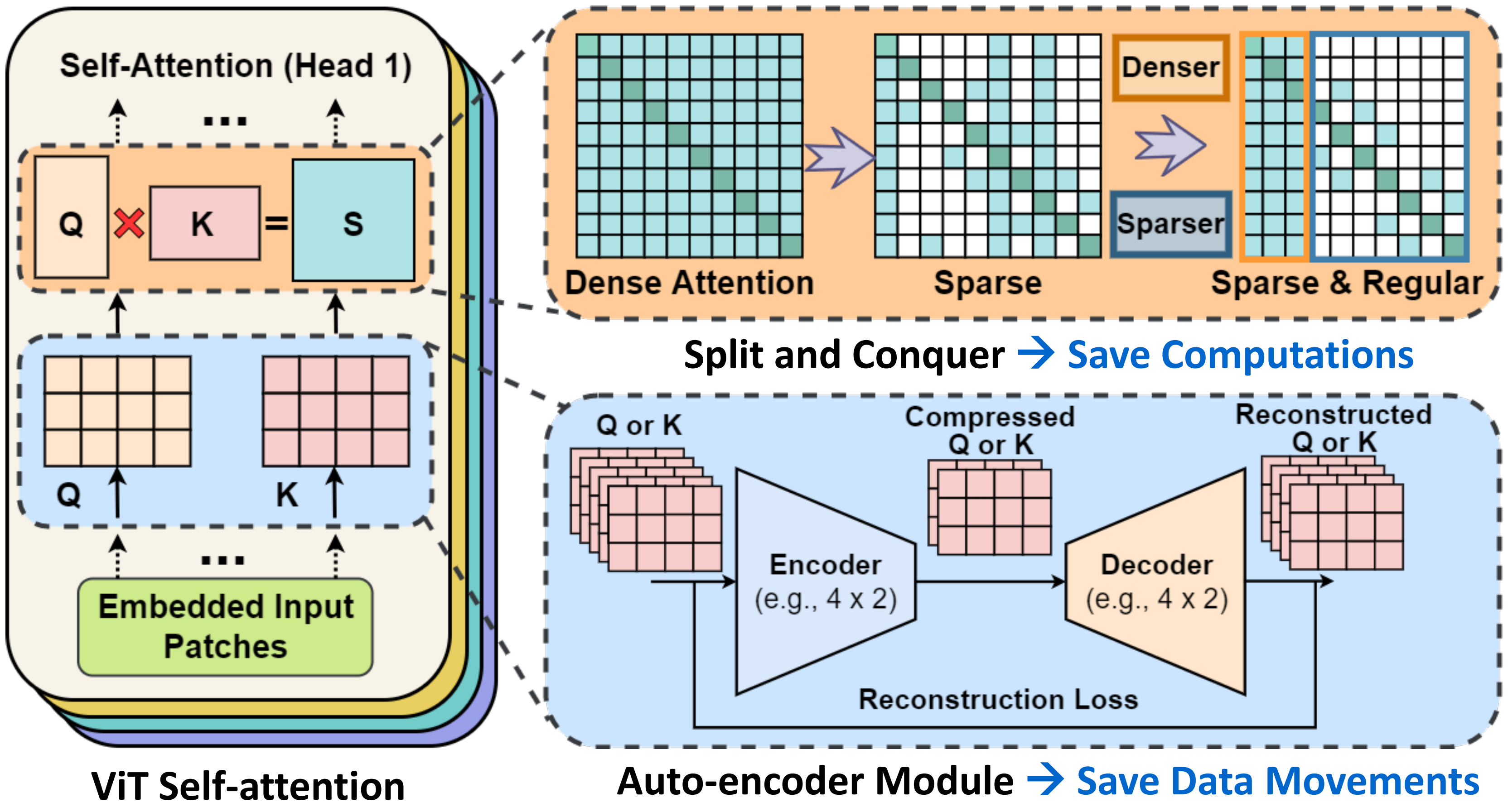


Split and Conquer → Save Computations

ViTCoD Algorithm:

The core idea on the algorithm level is to reduce both computations and data movements in core self-attention modules.

ViTCoD Overview



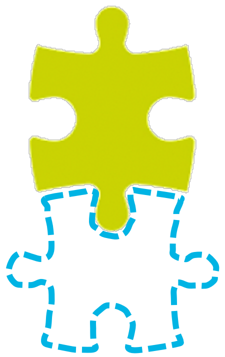
ViTCoD Algorithm:

The core idea on the algorithm level is to reduce both computations and data movements in core self-attention modules.

ViTCoD Overview

Split and Conquer
→ Save Computations

Auto-encoder Module
→ Save Data Movements



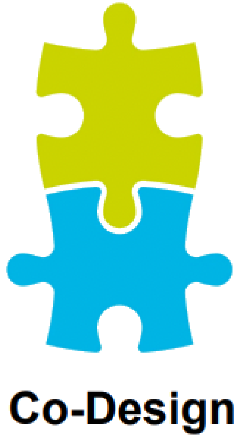
Co-Design

ViTCoD Accelerator:

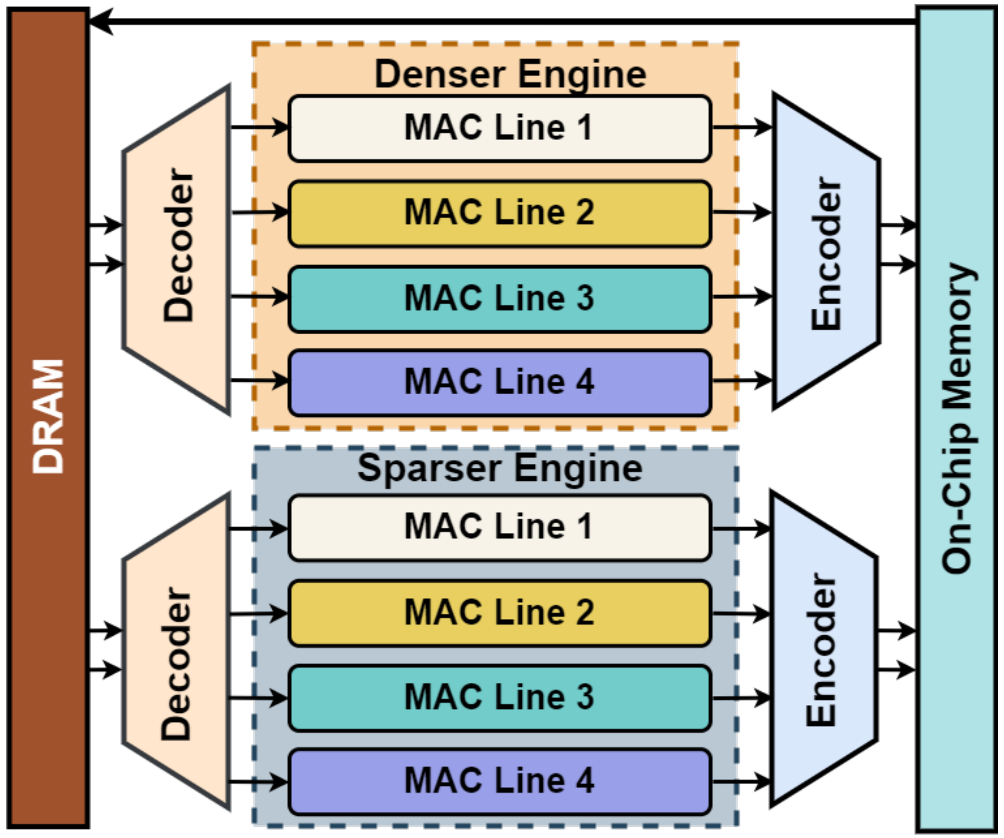
The core idea on the accelerator level is to develop a dedicated accelerator for supporting algorithms → accelerated computations and data movements

ViTCoD Overview

Split and Conquer
→ Save Computations



Auto-encoder Module
→ Save Data Movements

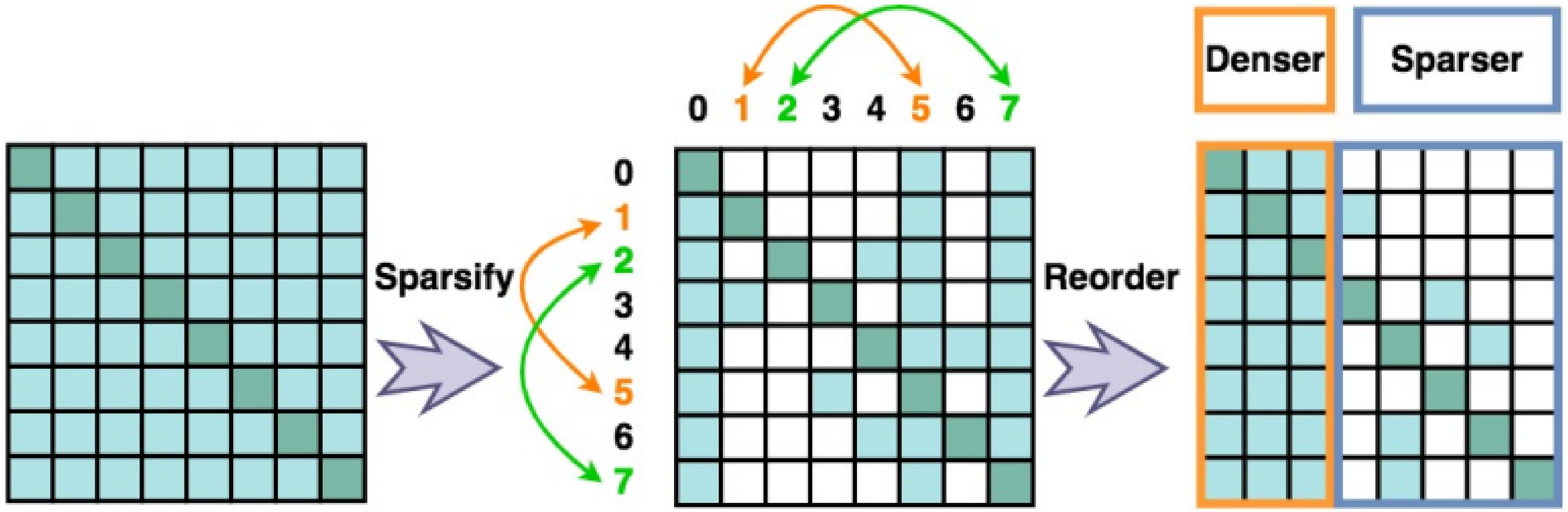


ViTCoD Accelerator:

The core idea on the accelerator level is to develop a dedicated accelerator for supporting algorithms → accelerated computations and data movements

ViTCoD Algorithm: Split and Conquer

- *Challenge 1:* How to aggressively reduce the computation?
  - Design insights:
    - Pruning with fixed masks
    - Attention map reordering
  - ViTCoD leverages the following observation:
    - The attention maps can be pruned up to 90% sparsity with fixed masks
    - There are "strong" tokens in the attention



Dense attention map | Fixed sparse pattern | Reorder "strong" tokens

ViTCoD Algorithm: Split and Conquer

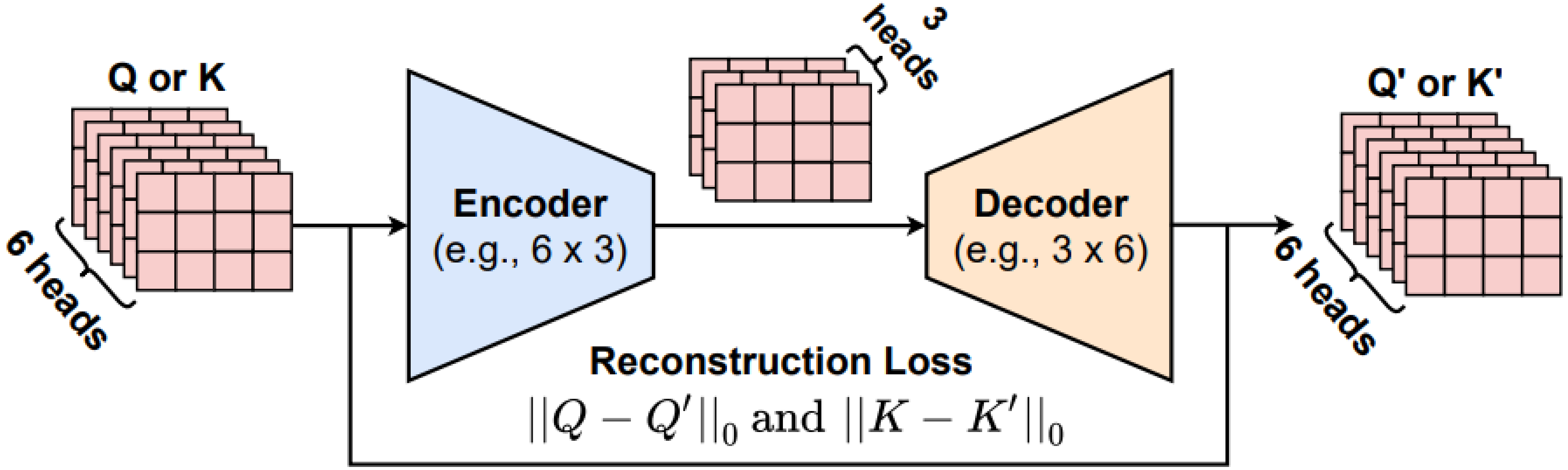
- **Visualizing the pruned or reordered attention maps on DeiT-B**

(a) Prune (b) Reorder (c) Prune and Reorder

ViTCoD Algorithm: Auto-Encoder

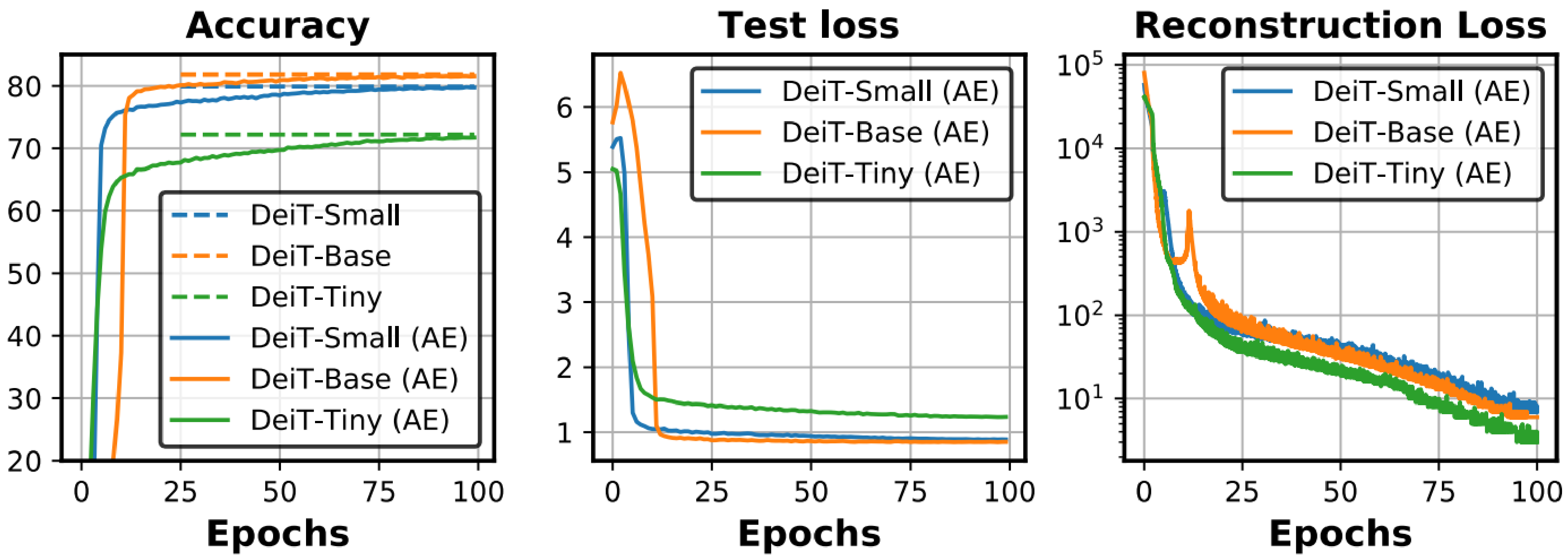
- *Challenge 2:* How to aggressively reduce the data movements?
  - Design insights:
    - Trade costly data movements with computations
  - ViTCoD leverages the following observation:
    - There is redundancy among attention heads

    → Compress the Q/K data before transmitting from off-chip to on-chip



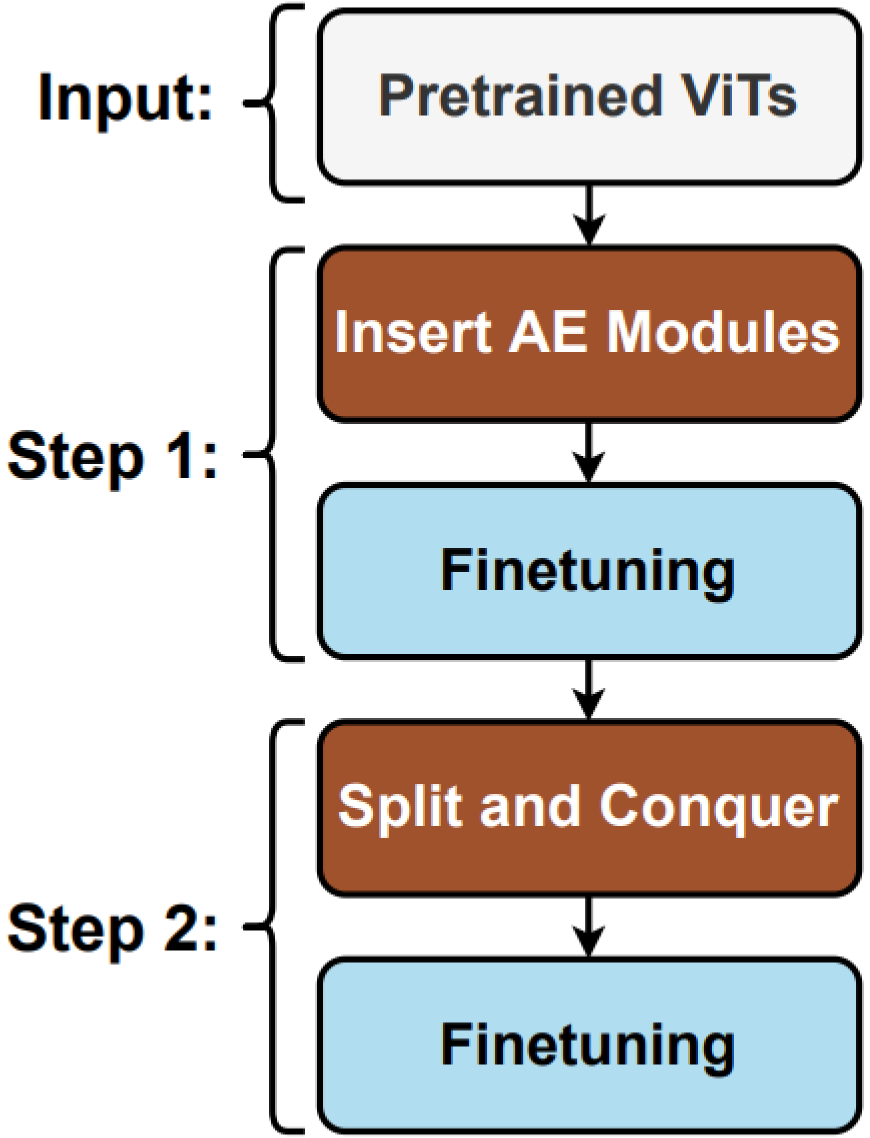
ViTCoD Algorithm: Auto-Encoder

- **Visualizing the training trajectory of DeiT-T/S/B with our proposed auto-encoder (AE) modules**



ViTCoD Algorithm: Training Pipeline

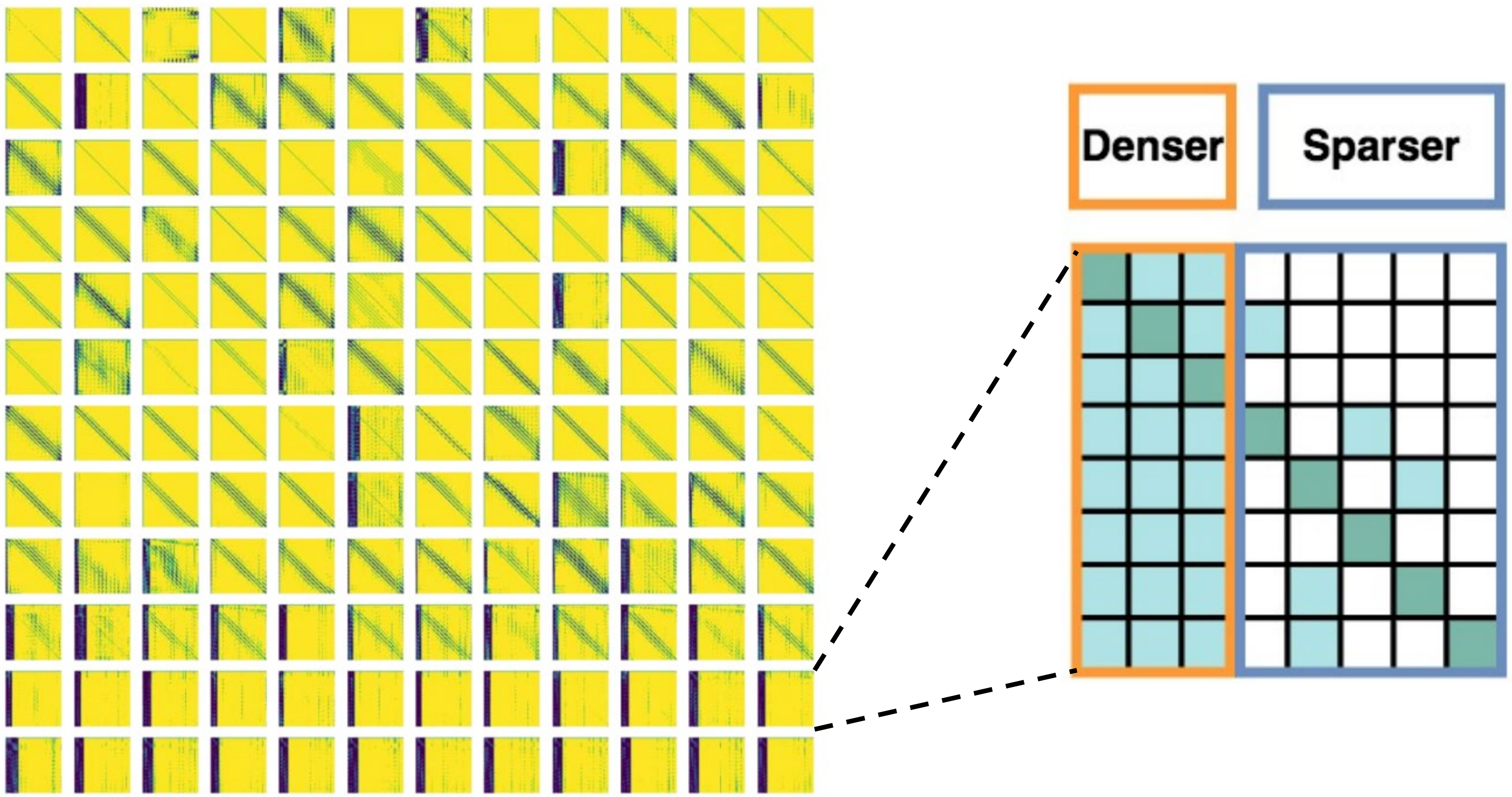
- **Overall training pipeline**
  - **Input:**
    - Pretrained ViT models
  - **Step 1: Insert AE modules**
    - Finetuning for 100 epochs
  - **Step 2: Split and conquer**
    - Prune and reorder
    - Finetuning for 100 epochs



ViTCoD Accelerator: Opportunities

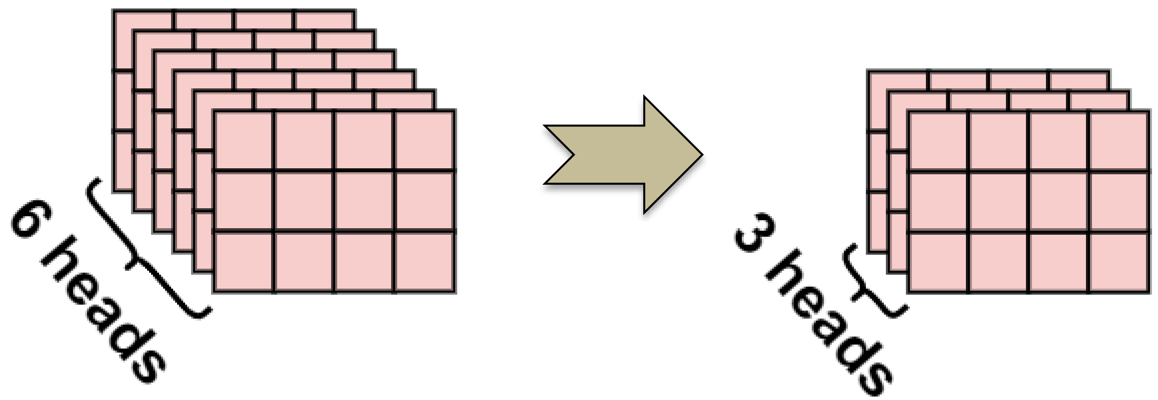
- *Challenge: How to fully exploit the potential of ViTCoD algorithm?*

  - **Opportunities:**
    - Fixed and structurally sparse Attention



ViTCoD Accelerator: Opportunities

- *Challenge: How to fully exploit the potential of ViTCoD algorithm?*
  - **Opportunities:**
    - **Fixed and structurally sparse Attention**
    - **Compact Q and K representation**



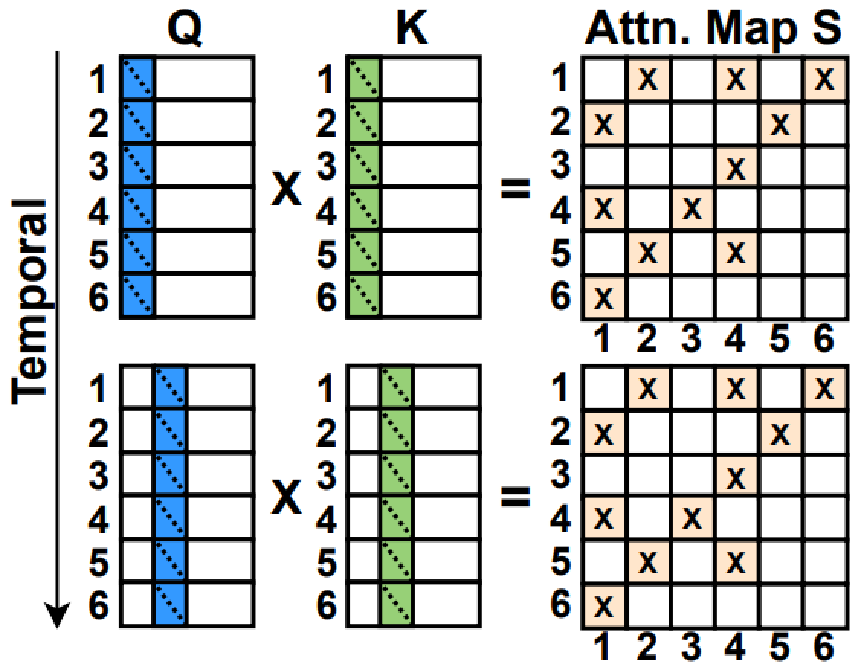
ViTCoD Accelerator: Design Explorations

- ***Challenge: How to fully exploit the potential of ViTCoD algorithm?***
  - **Design explorations:**
    - **Micro-architecture:** ***single one or multiple sub-accelerator?***
      - ***Latter with merely two diverse workloads: denser or sparser***

ViTCoD Accelerator: Design Explorations

- ***Challenge: How to fully exploit the potential of ViTCoD algorithm?***

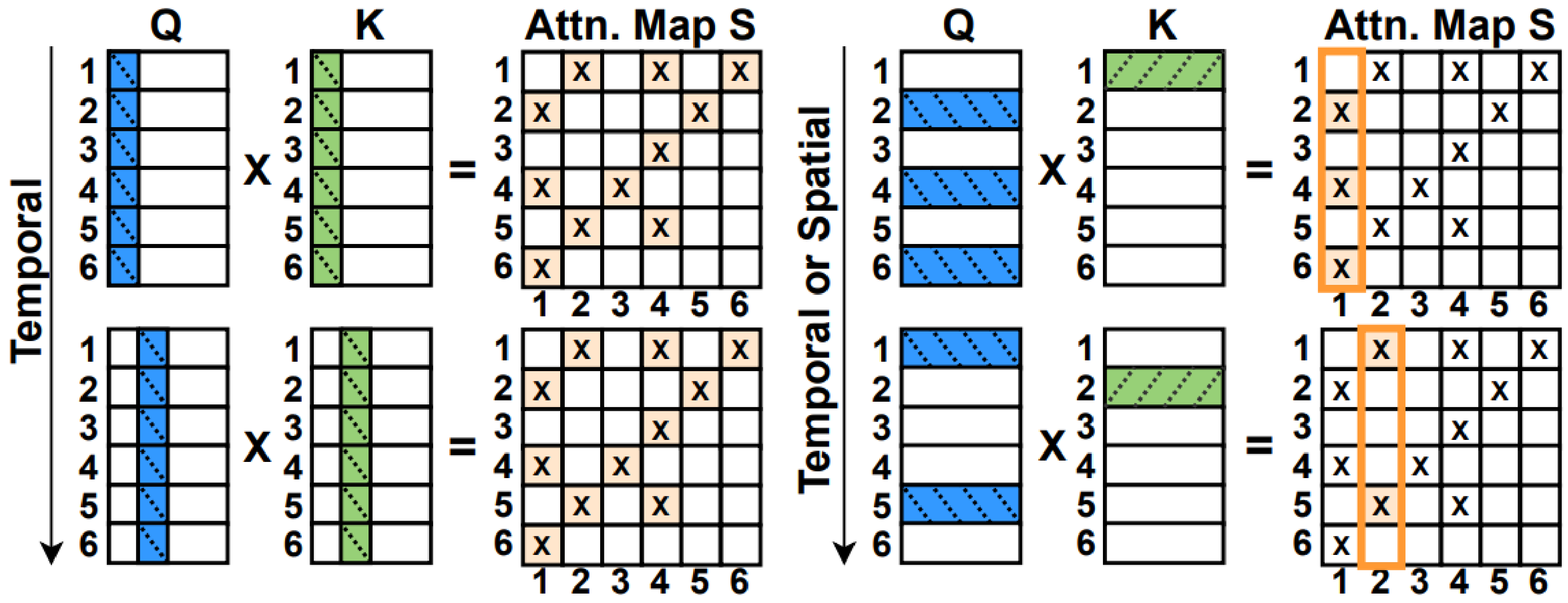
  - **Design explorations:**
    - **Dataflows:** ***S-stationary or K-stationary?***



S-stationary

ViTCoD Accelerator: Design Explorations

- *Challenge: How to fully exploit the potential of ViTCoD algorithm?*
  - **Design explorations:**
    - Dataflows: *S-stationary or K-stationary?*
      - *The latter is better suited for resulting sparse attention patterns*

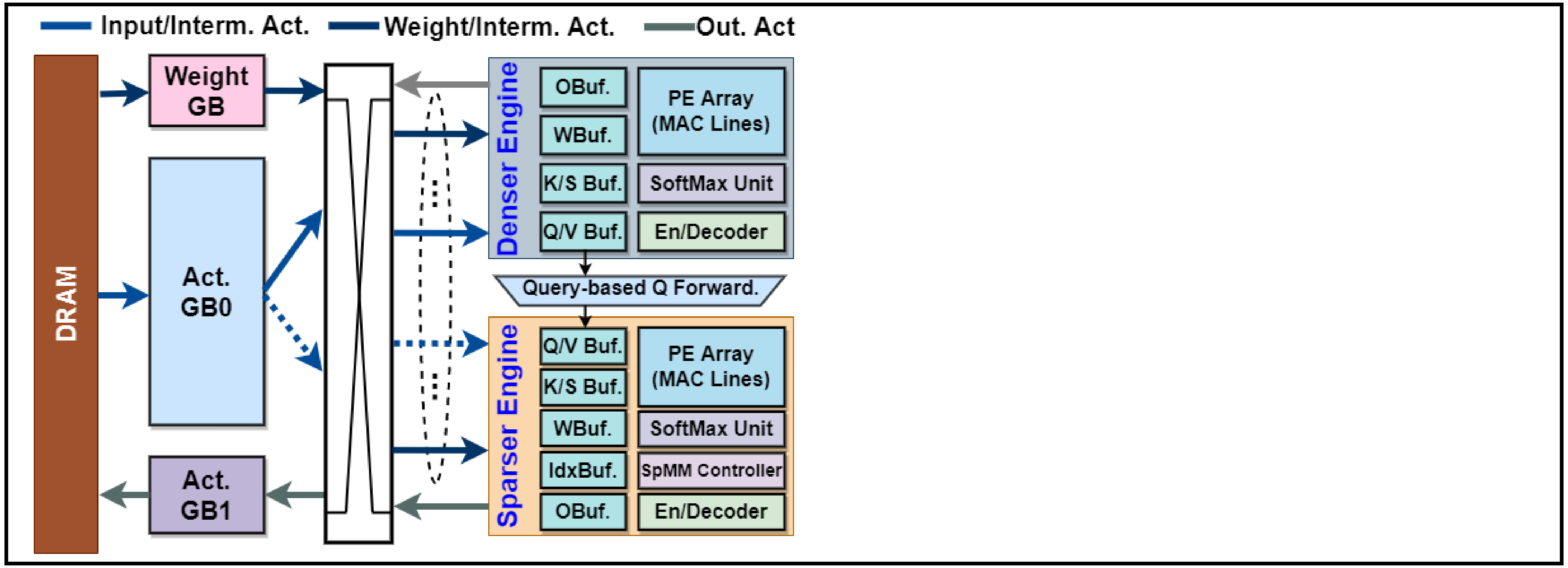


S-stationary

K-stationary

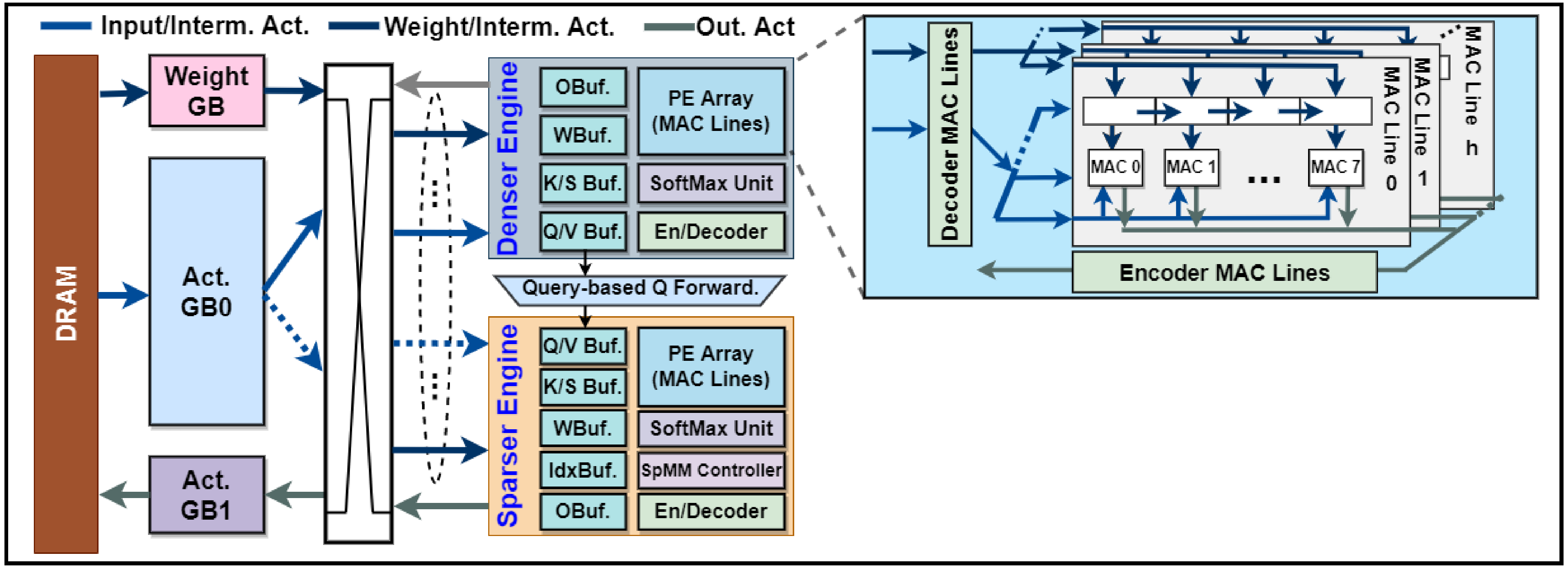
ViTCoD Accelerator: Micro-Architecture

- Our micro-architecture design features
  - Two-pronged architecture
  - Encoder and decoder engines



ViTCoD Accelerator: Micro-Architecture

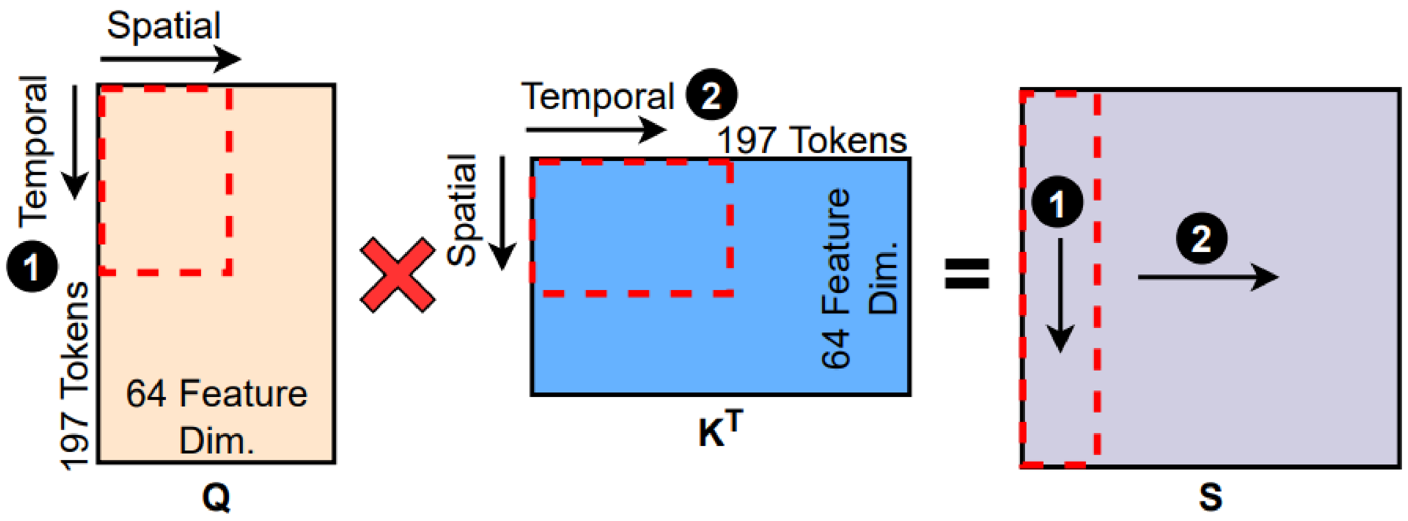
- **Our micro-architecture design features**
  - **Two-pronged architecture**
  - **Encoder and decoder engines**



ViTCoD Accelerator: Micro-Architecture

- **Our micro-architecture design features**
  - **Two-pronged architecture**
  - **Encoder and decoder engines**

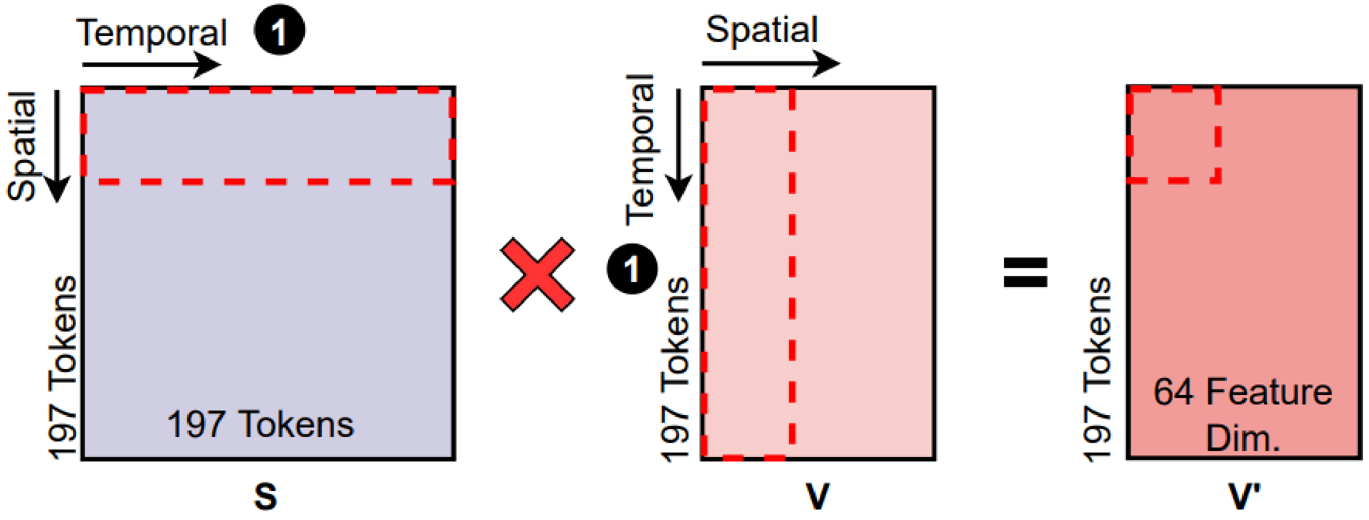
- **Tiling and spatial or temporal mappings**
  - **$Q * K^T$**



ViTCoD Accelerator: Micro-Architecture

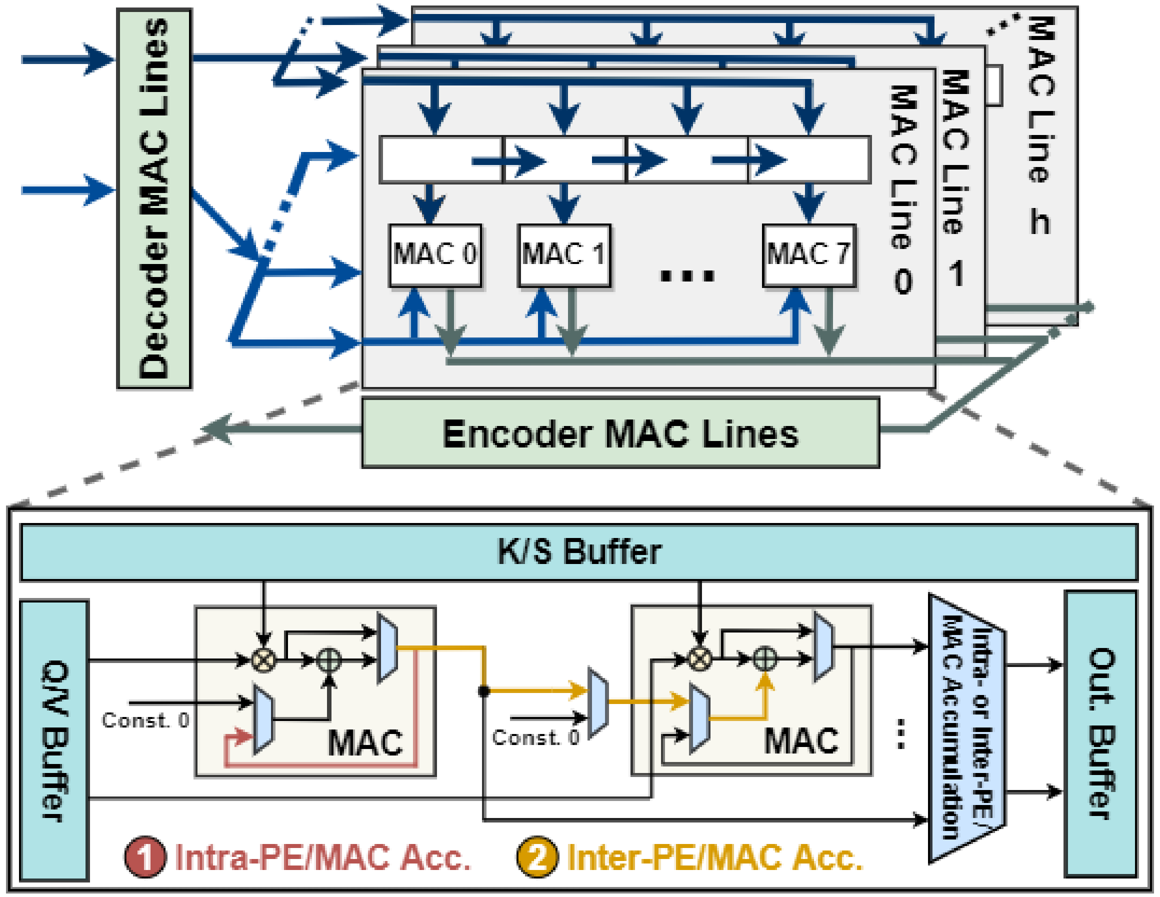
- **Our micro-architecture design features**
  - **Two-pronged architecture**
  - **Encoder and decoder engines**

- **Tiling and spatial or temporal mappings**
  - **$Q * K^T$**
  - **S * V**



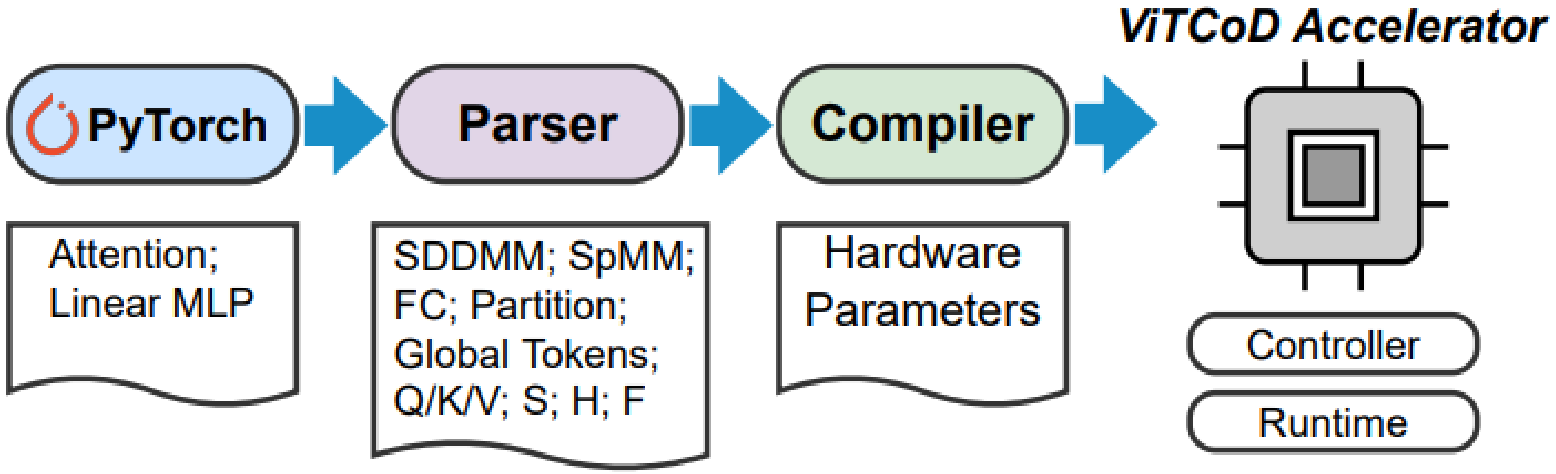
ViTCoD Accelerator: Micro-Architecture

- **Our micro-architecture design features**
  - **Two-pronged architecture**
  - **Encoder and decoder engines**
  - **Inter- or Intra-MAC accumulation**



ViTCoD Accelerator: Micro-Architecture

- **Our micro-architecture design features**
  - **Two-pronged architecture**
  - **Encoder and decoder engines**
  - **Inter- or Intra-MAC accumulation**
  - **Reconfigurability**



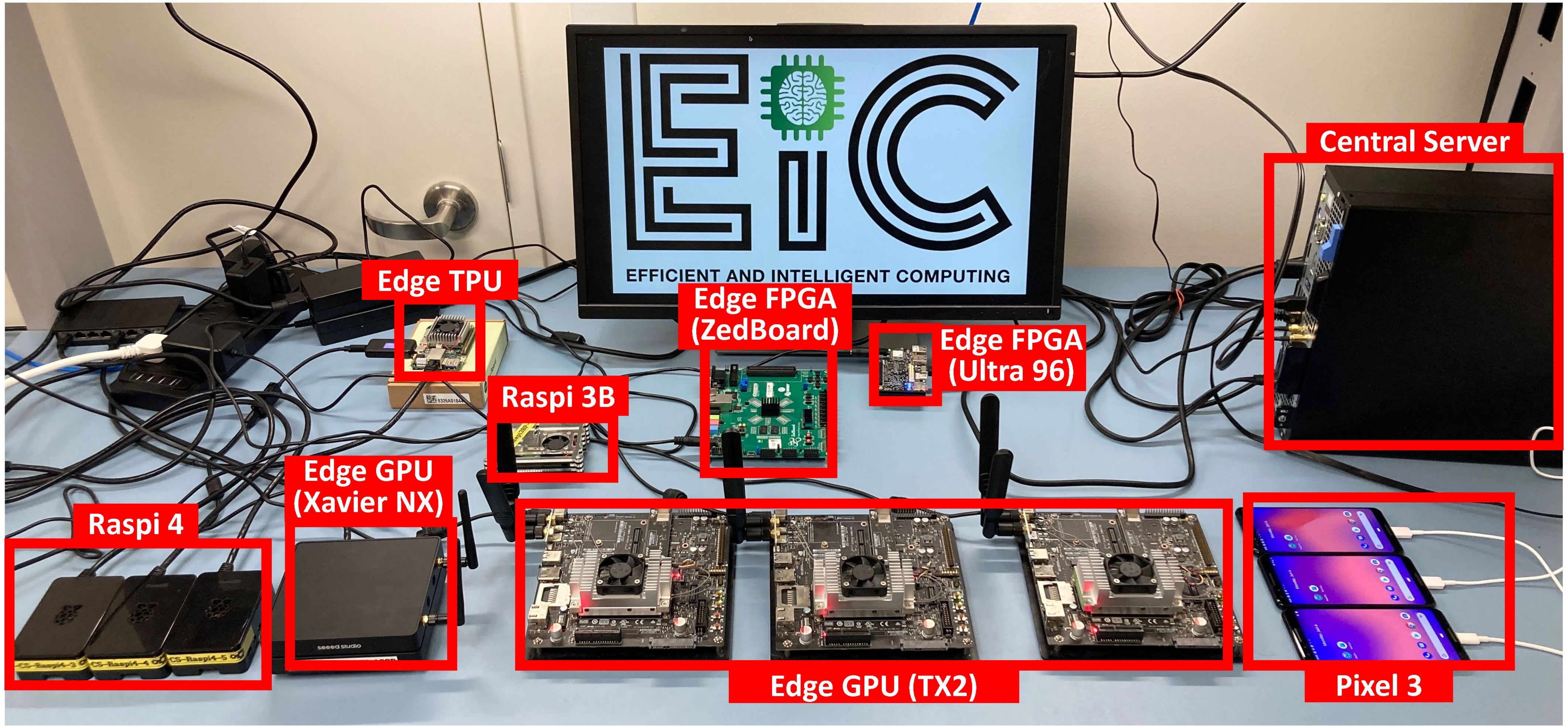
Evaluation Setup and Baselines

- Evaluation Setup
  - **Seven ViT Models**:
    - DeiT-Base/Small/Tiny, LeViT-128/192/256 for image classification
    - Strided Transformer for human pose estimation
  - **Datasets**:
    - ImageNet and Human3.6M
  - **Metrics**:
    - Accuracy, Latency speedups

- **Benchmark Baselines**
  - Commercial devices
    - CPU, GPU, EdgeGPU
  - Customized accelerators
    - SpAtten, Sanger

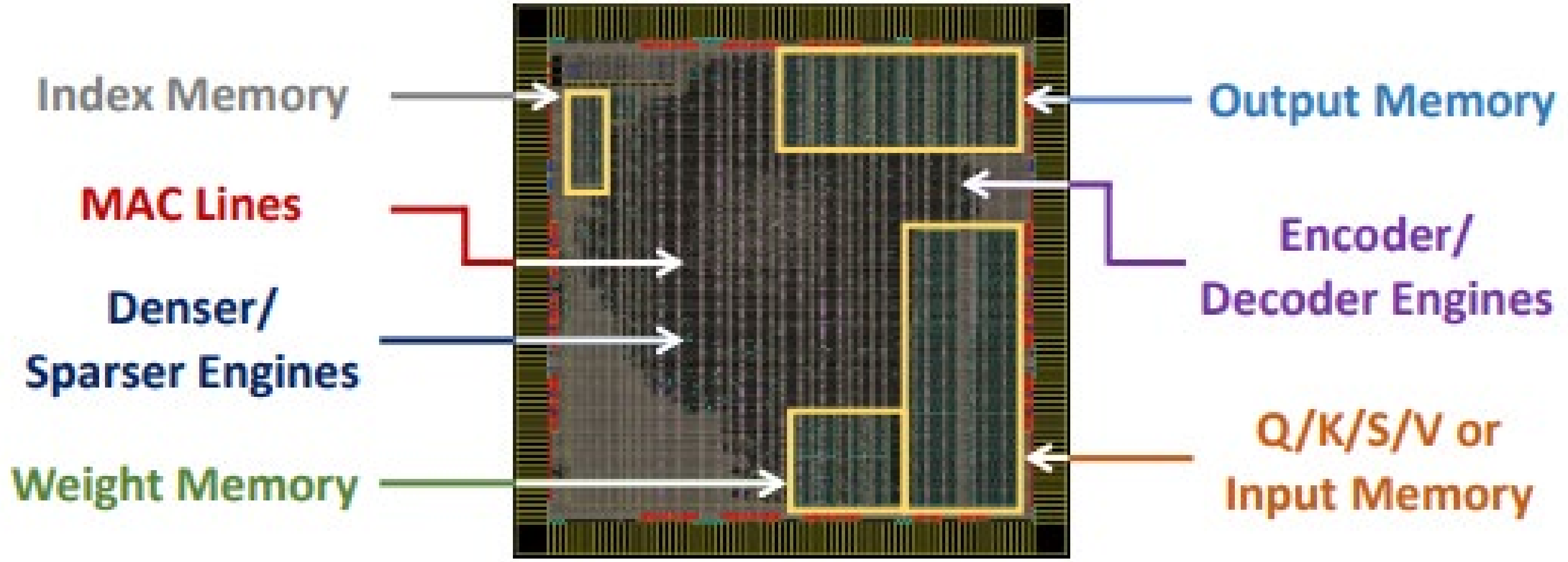
Evaluation Setup and Baselines

- **Benchmark Baselines**:
  - Commercial devices
    - CPU, GPU, EdgeGPU

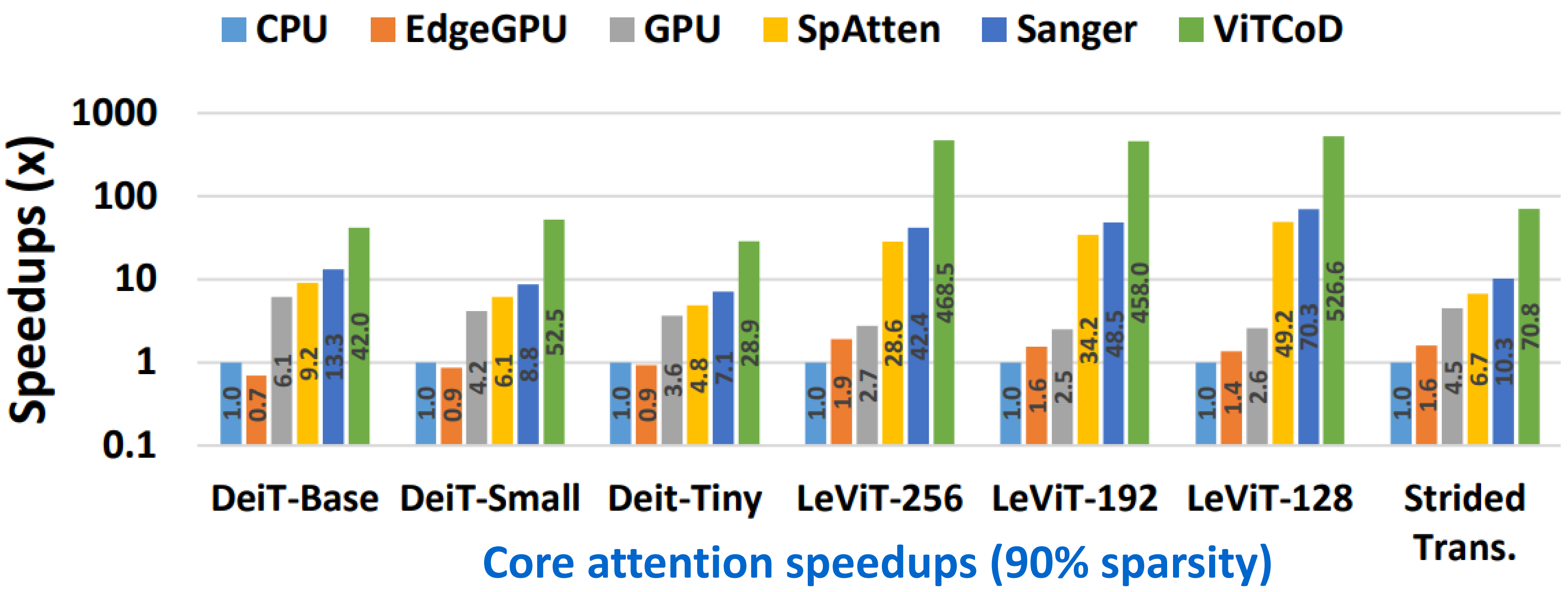


Evaluation Setup and Baselines

- **Evaluation Setup**
  - Layout floorplan

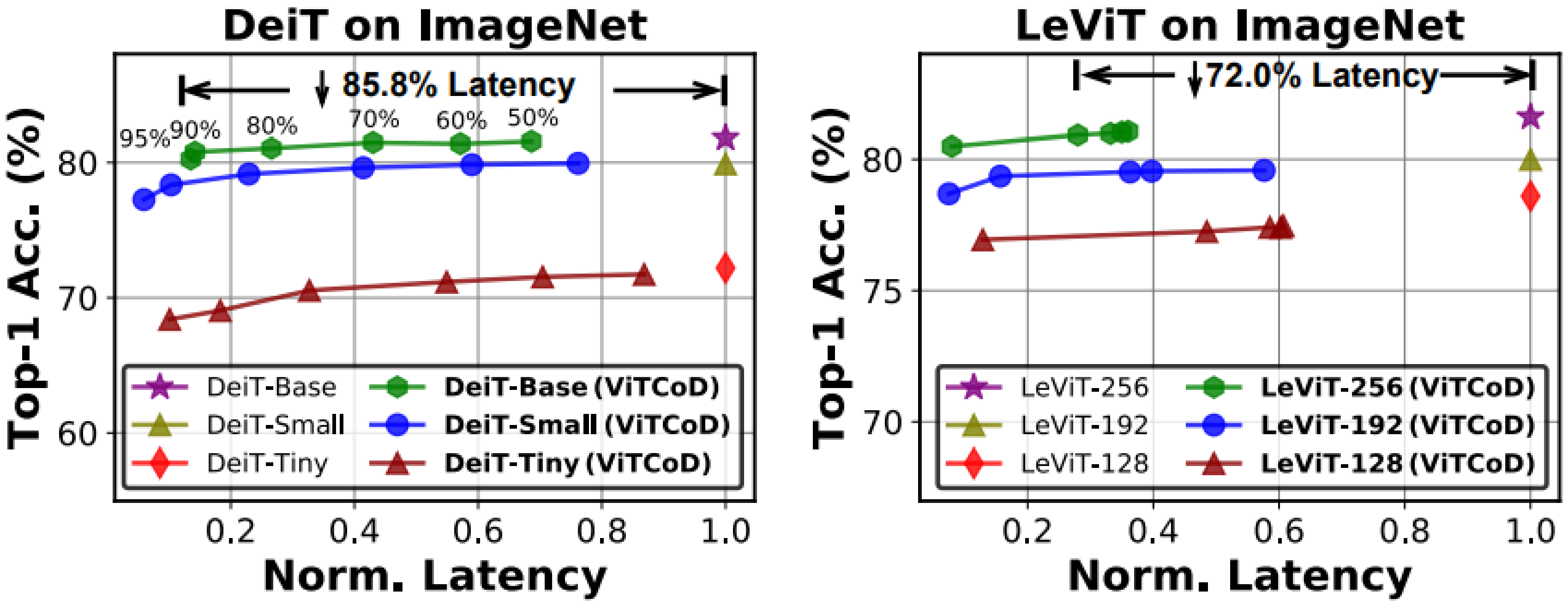


Evaluation: ViTCoD over SOTA Accelerators



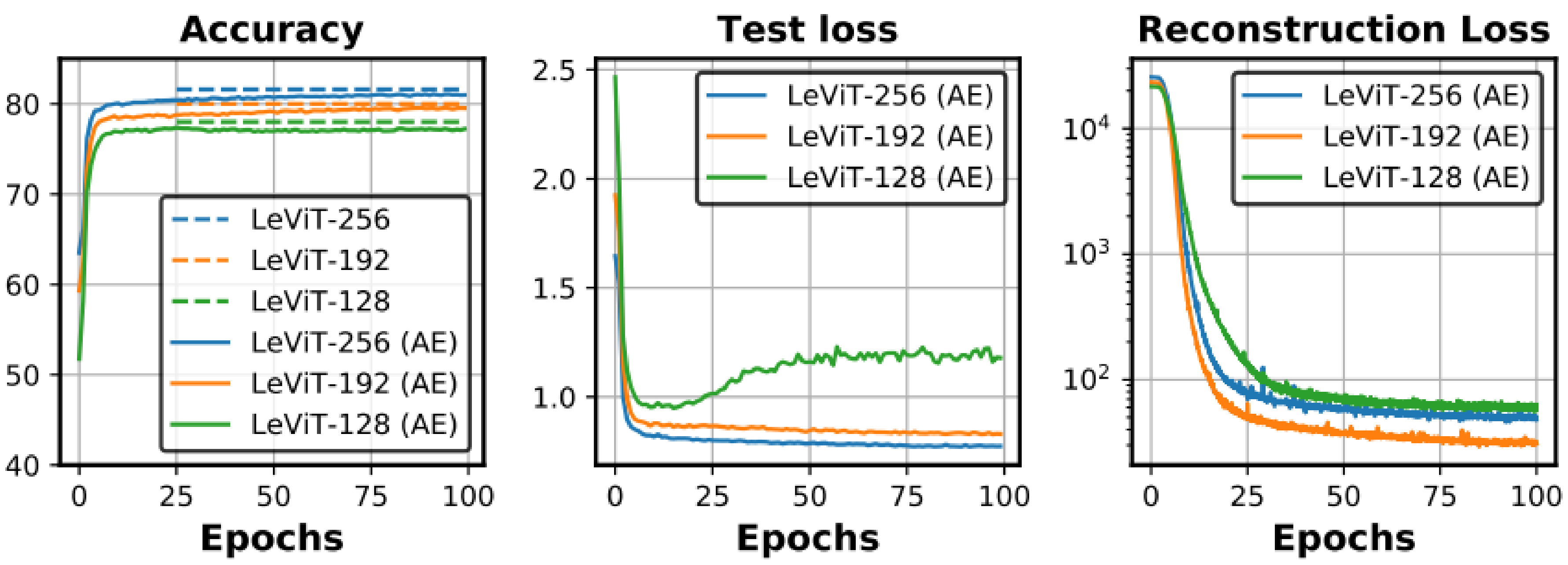
- **ViTCoD over CPU/GPU platforms**
  - ViTCoD achieves up to 235.3x, 160.6x, and 86x speedups over CPU, EdgeGPU and GPU
- **ViTCoD over SOTA attention accelerators**
  - ViTCoD achieves 10.1x and 6.8x speedups over SpAtten and Sanger

Evaluation of ViTCoD Algorithm



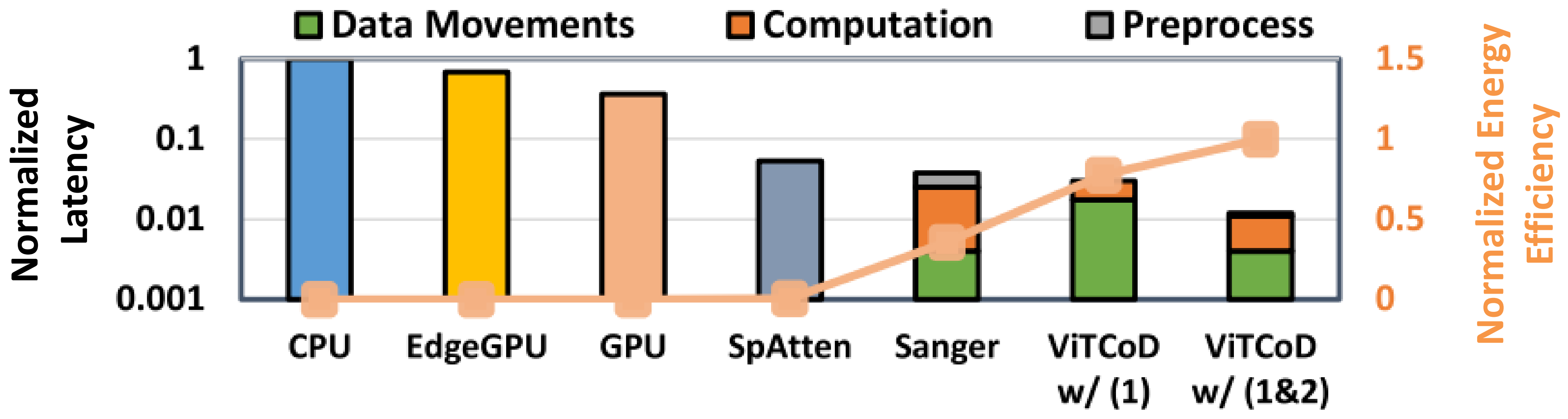
- **Evaluate ViTCoD's split and conquer algorithm**
  - ViTCoD reduce 45.1% ~ 85.8% and 72.0% ~ 84.3% latency of attention layers for DeiT and LeViT, respectively, while leading to a comparable model accuracy (i.e., < 1% accuracy drop)

Evaluation of ViTCoD Algorithm



- **Evaluate ViTCoD's auto-encoder module**
  - ViTCoD compress 50% Q/K vectors, e.g., 12 heads → 6 heads, with < 0.5% accuracy drops

Evaluation of ViTCoD Accelerators



- **Averaged across 60% ~ 90% sparsity**
  - ViTCoD achieves 6.8x and 4.3x speedups over SpAtten and Sanger
  - ViTCoD achieves 9.8x energy efficiency over the most competitive baseline Sanger

Summary

In this work, we

- **Propose the first Vision Transformer algorithm & accelerator co-design framework, dubbed ViTCoD**
- **On the algorithm level, ViTCoD integrates a split and conquer training and an auto-encoder module without compromising the accuracy**
- **On the hardware level, GCoD further develop a dedicated two-pronged accelerator with encoder/decoder modules**

Acknowledge: NSF EPCN & RTML programs

EyeCoD: Eye Tracking System Acceleration via FlatCam-based Algorithm & Accelerator Co-Design

Haoran You*[1], Cheng Wan*[1], Yang Zhao*[1], Zhongzhi Yu*[1], Yonggan Fu[1], Jiayi Yuan[1], Shang Wu[1], Shunyao Zhang[1], Yongan Zhang[1], Chaojian Li[1], Vivek Boominathan[1], Ashok Veeraraghavan[1], Ziyun Li[2], and Yingyan Lin[1]

[1]Rice University

[2]Meta Reality Labs

The 49th International Symposium on Computer Architecture (ISCA 2022)

Efficient and Intelligent Computing Lab

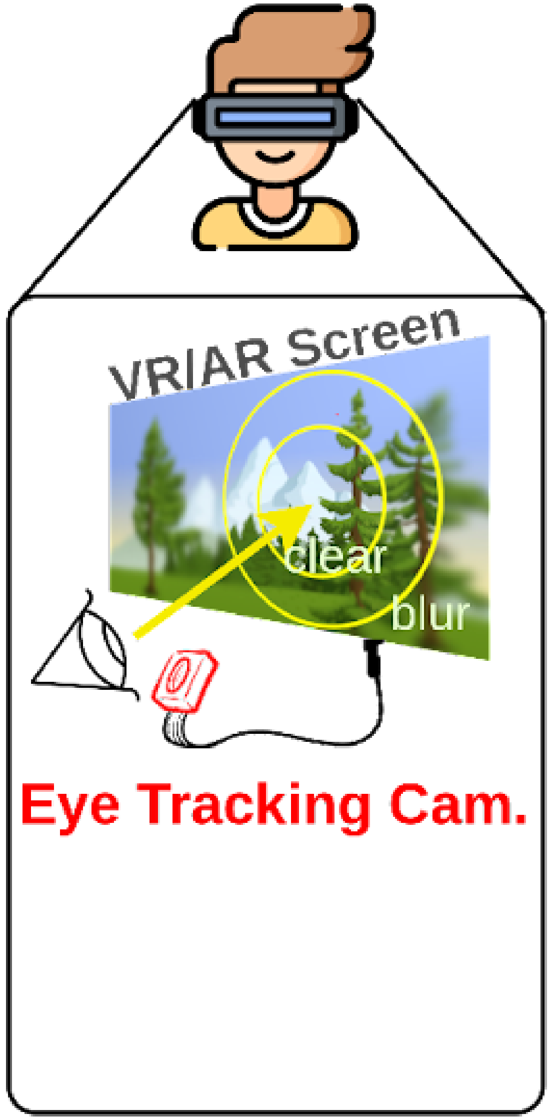
Background: Tremendously Growing AR/VR Market

- **Augmented and virtual reality (VR/AR) market is blooming**
  - **$766 billion by 2025**
  - **Compound annual growth rate (CAGR) of 73.7% [1]**

[1] Market Research Future (MRFR), 2021

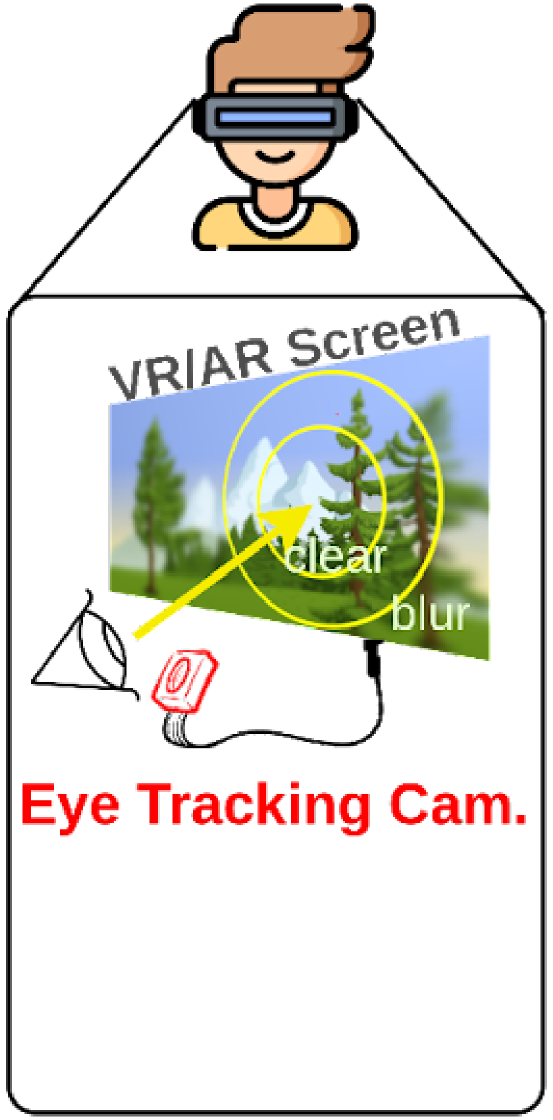
Background: Eye Tracking in AR/VR

- Eye tracking is an essential human-machine interface in AR/VR



Background: Eye Tracking in AR/VR

- Eye tracking is an essential human-machine interface in AR/VR



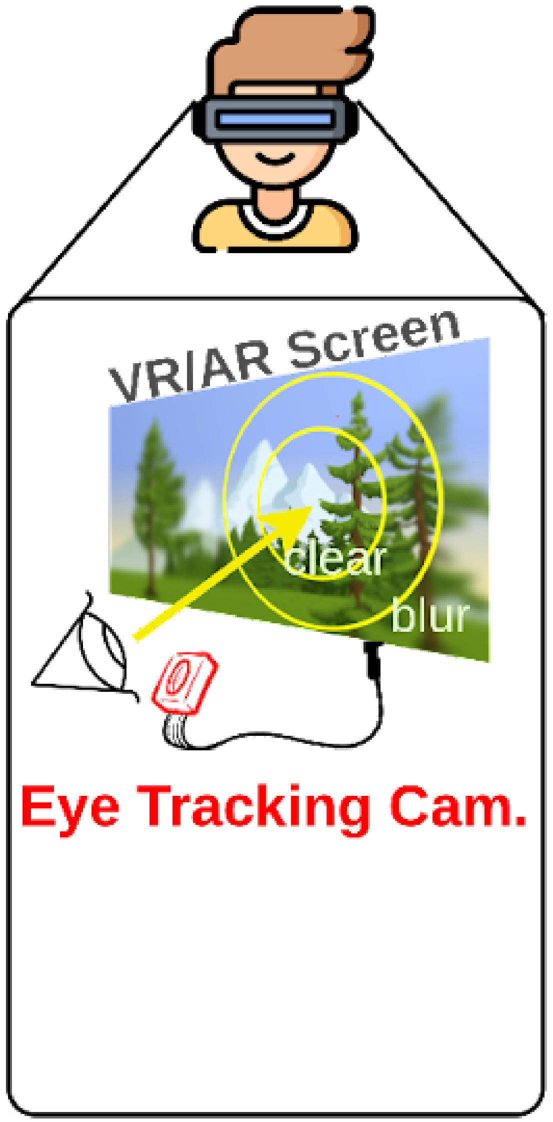
- AR/VR devices with eye tracking modalities

PlayStation VR2

VIVE PRO EYE

Background: Eye Tracking in AR/VR

- Eye tracking is an essential human-machine interface in AR/VR

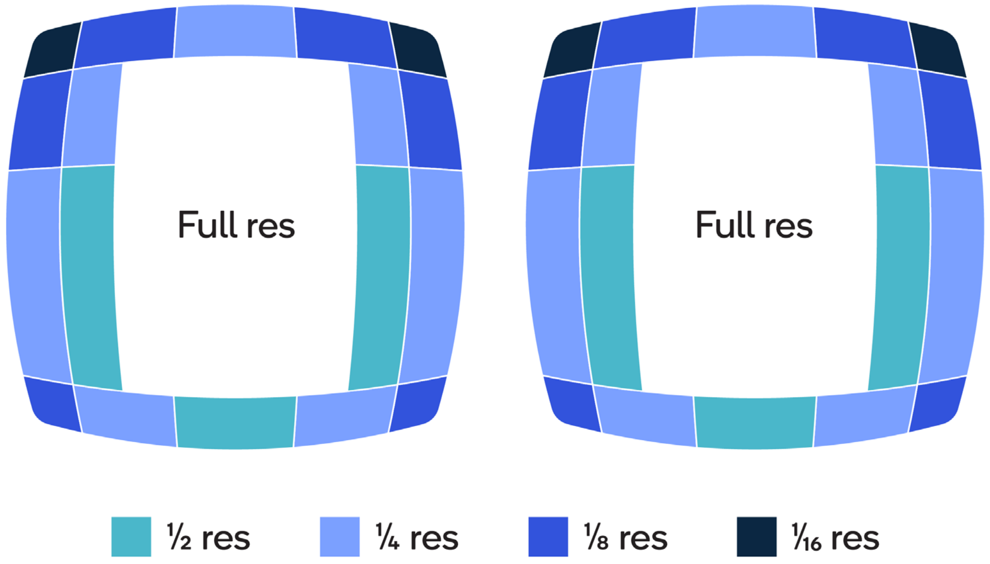


- AR/VR devices with eye tracking modalities

PlayStation VR2

VIVE PRO EYE

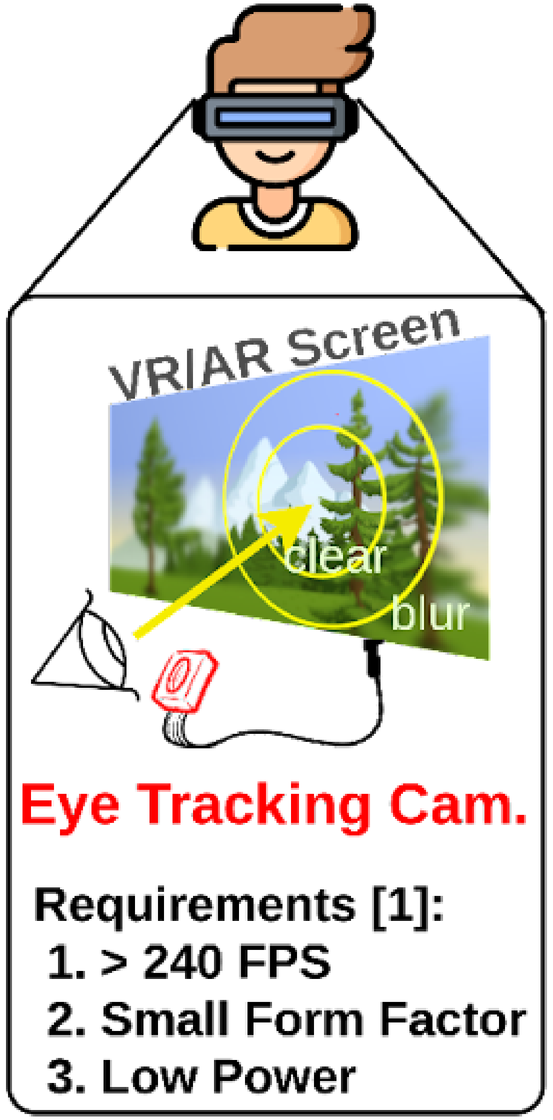
- Foveated rendering application [2]



[2] The Evolution of High Performance Foveated Rendering, Qualcomm 2021

Motivation: Eye Tracking in AR/VR

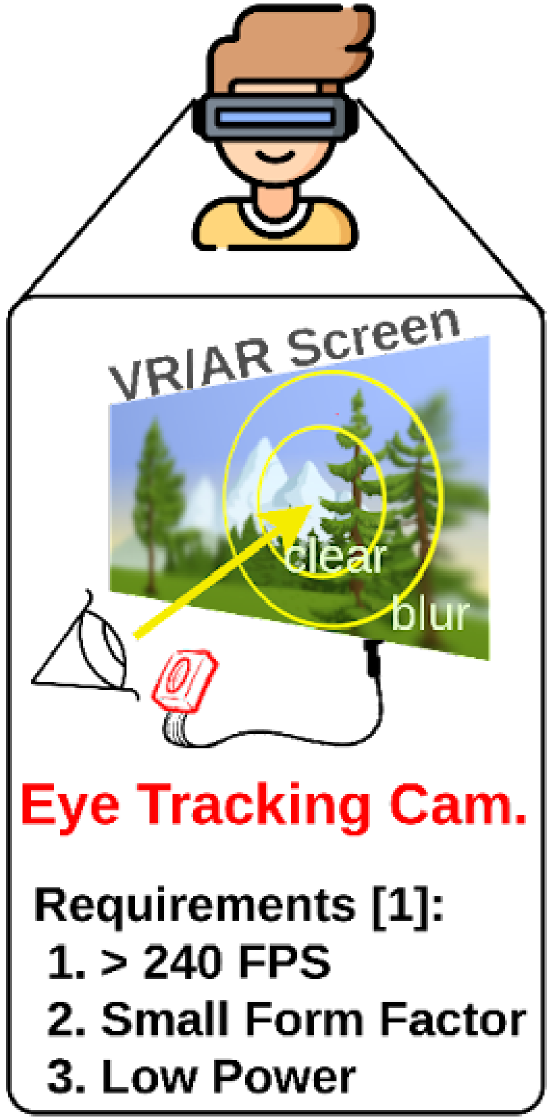
- Eye tracking is an essential human-machine interface in AR/VR



- Challenges for eye tracking in AR/VR [3]
  - >240 FPS
  - Small form factor
  - Power consumption in mW
  - Visual privacy

Motivation: Eye Tracking in AR/VR

- **Eye tracking is an essential human-machine interface in AR/VR**



- **Challenges for eye tracking in AR/VR [3]**
  - **>240 FPS**
  - **Small form factor**
  - **Power consumption in mW**
  - **Visual privacy**

- **Existing works [4,5]**
  - **An order of magnitude slower (i.e., 30 FPS)**
  - **Large form factor and low visual privacy due to the adopted lens-based cameras**

→ Fail to meet real-time application requirements

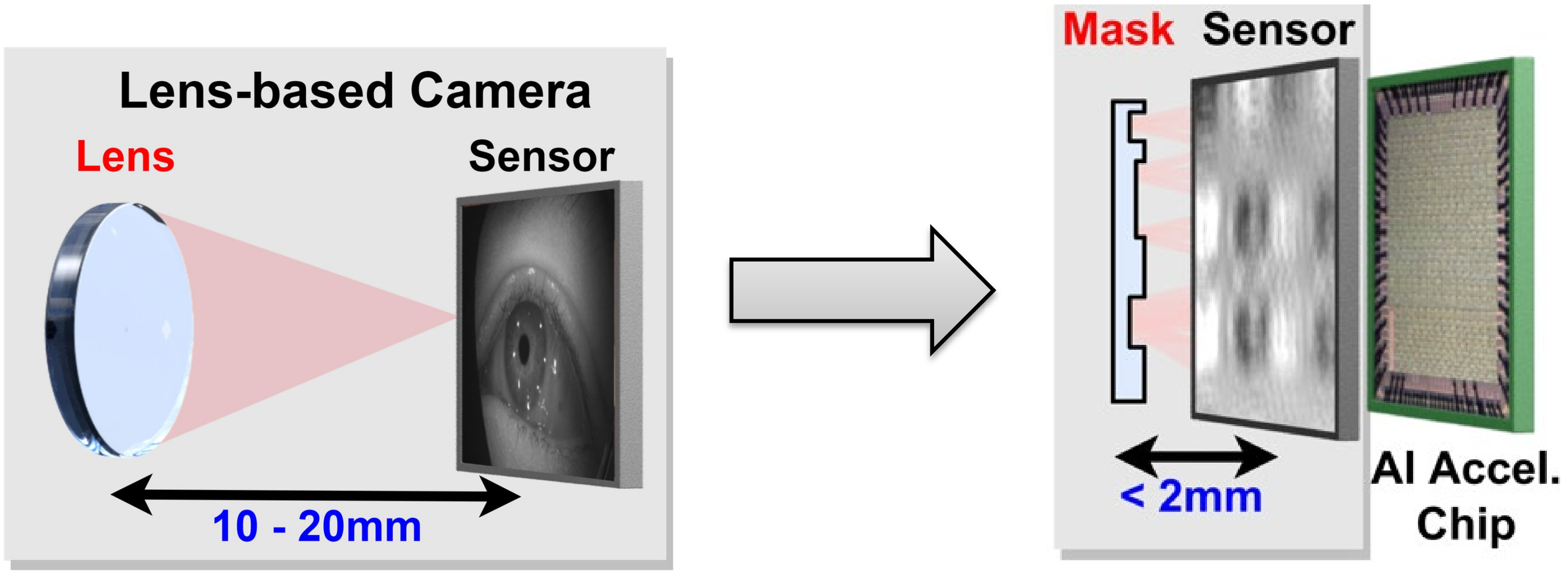
[3] C. Liu, et. al., IDEM'21
[4] Y. Feng, et. al., IEEE VR'22
[5] K Bong, et. al., VLSI'15

Limitations of Existing Solutions

- **Why existing eye tracking can not satisfy the requirements?**
  - **Rely on lens-based cameras → Limitations**
    - Large form factor
    - High communication cost between camera and backend processor
    - Low visual privacy

Unexplored Opportunities for Eye Tracking?

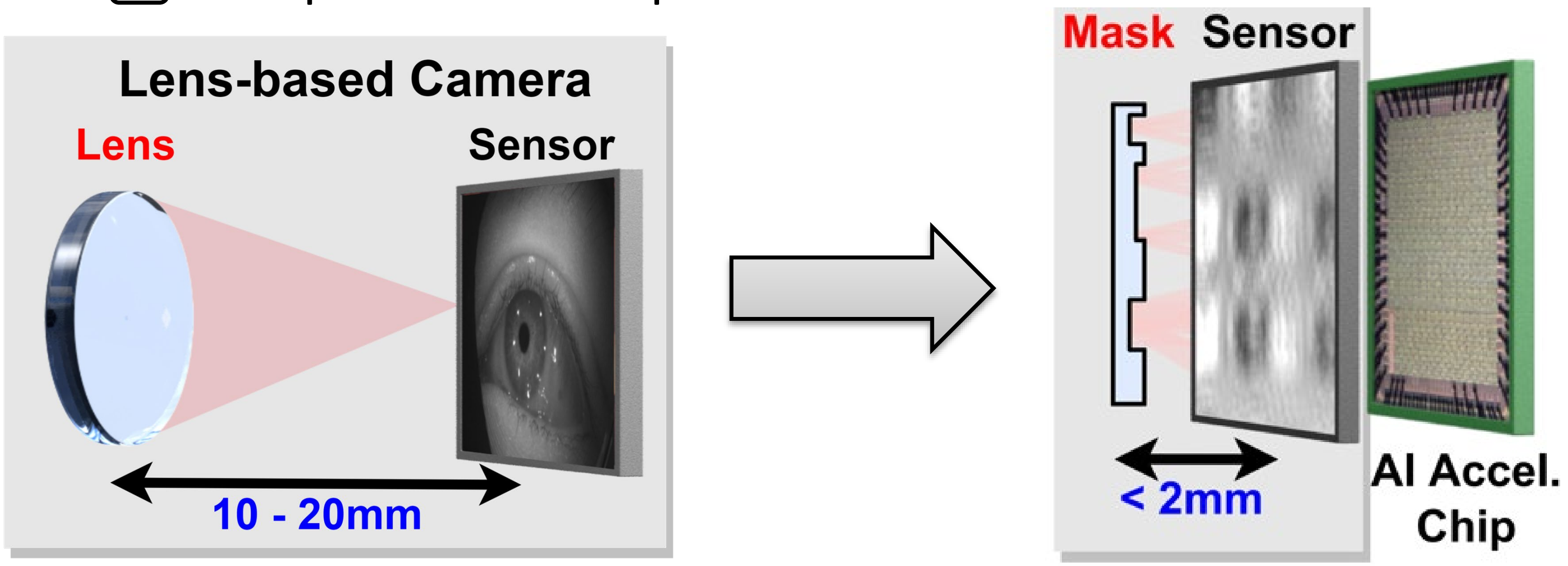
- Opportunity 1: Can we build a lensless eye tracking system?
  - A lensless camera, i.e., FlatCam [6]
    - ☑ Small form factor, i.e., 5-10× thinner
    - ☑ Visual privacy



[6] M. Asif, et. al., TCI'17

Unexplored Opportunities for Eye Tracking?

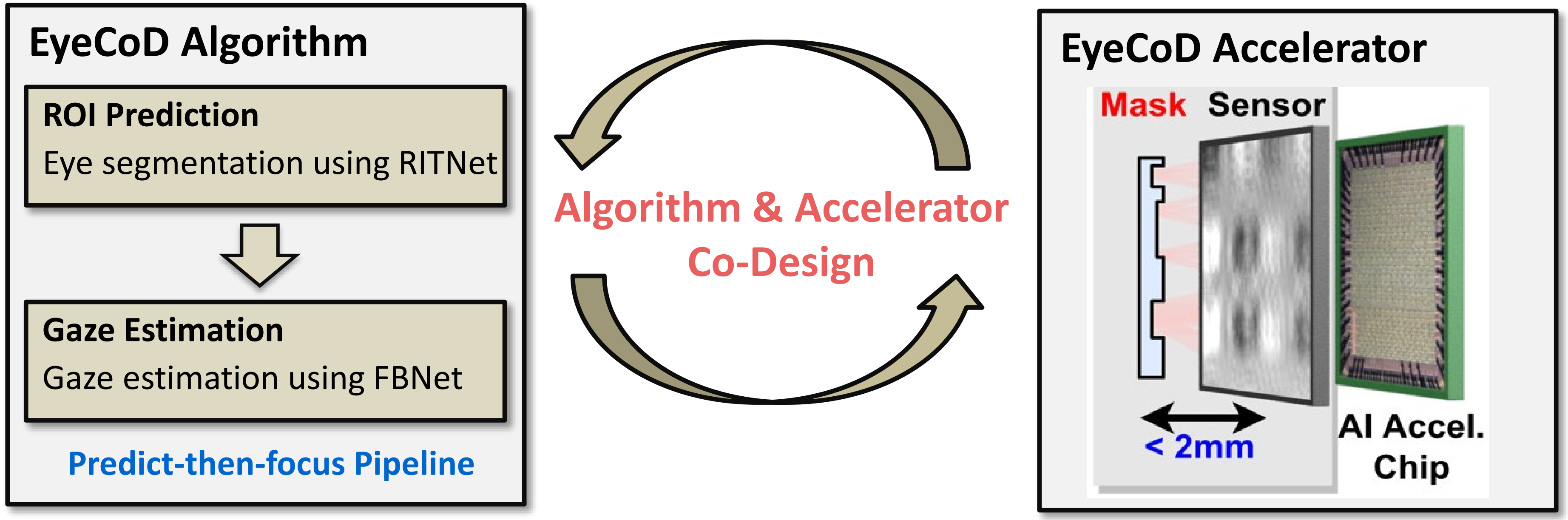
- **Opportunity 1: Can we build a lensless eye tracking system?**
  - **A lensless camera, i.e., FlatCam [6]**
    - ☑ Small form factor, i.e., 5-10× thinner
    - ☑ Visual privacy
- **Opportunity 2: Leverage end-to-end co-design?**
  - **An AI acceleration chip featuring algorithm and accelerator co-design**
    - ☑ >240 FPS
    - ☑ mW power consumption



[6] M. Asif, et. al., TCI'17

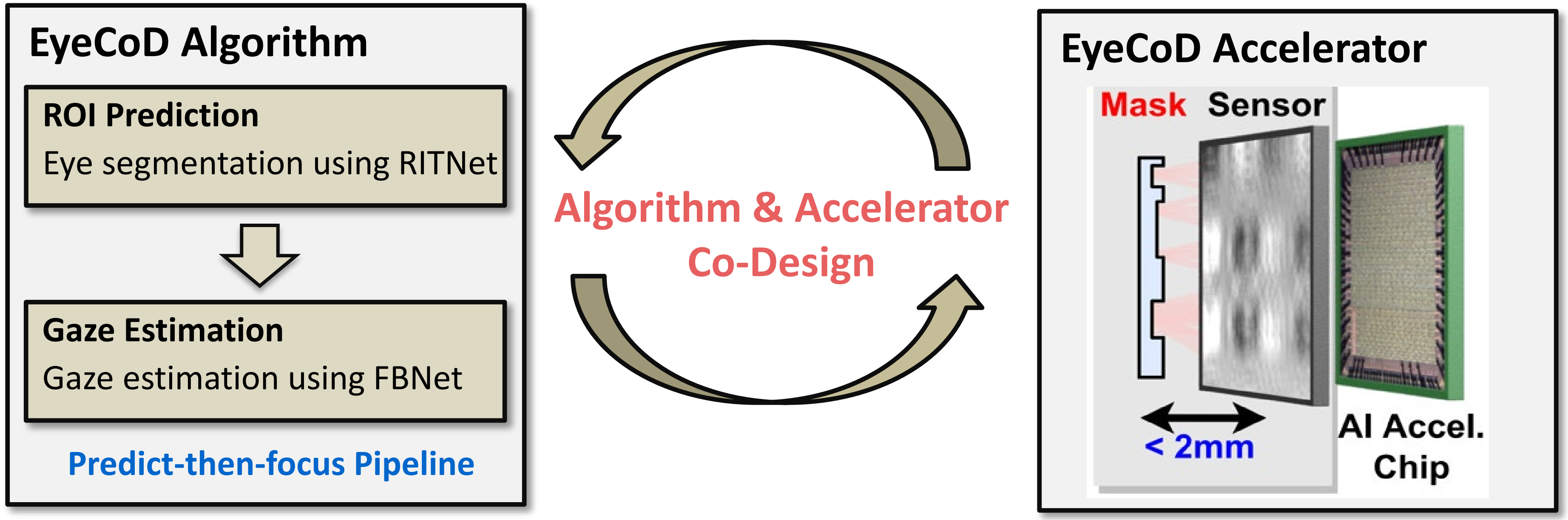
Proposed EyeCoD: Algorithm & Accel. Co-Design

- **Proposed FlatCam-based algorithm & accelerator co-design (EyeCoD) for accelerating eye tracking in AR/VR devices**
  - Incorporating three features:
    - Sensing-processing interface
    - Predict-then-focus algorithm pipeline
    - Dedicated accelerator attached to FlatCam



Proposed EyeCoD: Algorithm & Accel. Co-Design

- Proposed FlatCam-based algorithm & accelerator co-design (EyeCoD) for accelerating eye tracking in AR/VR devices



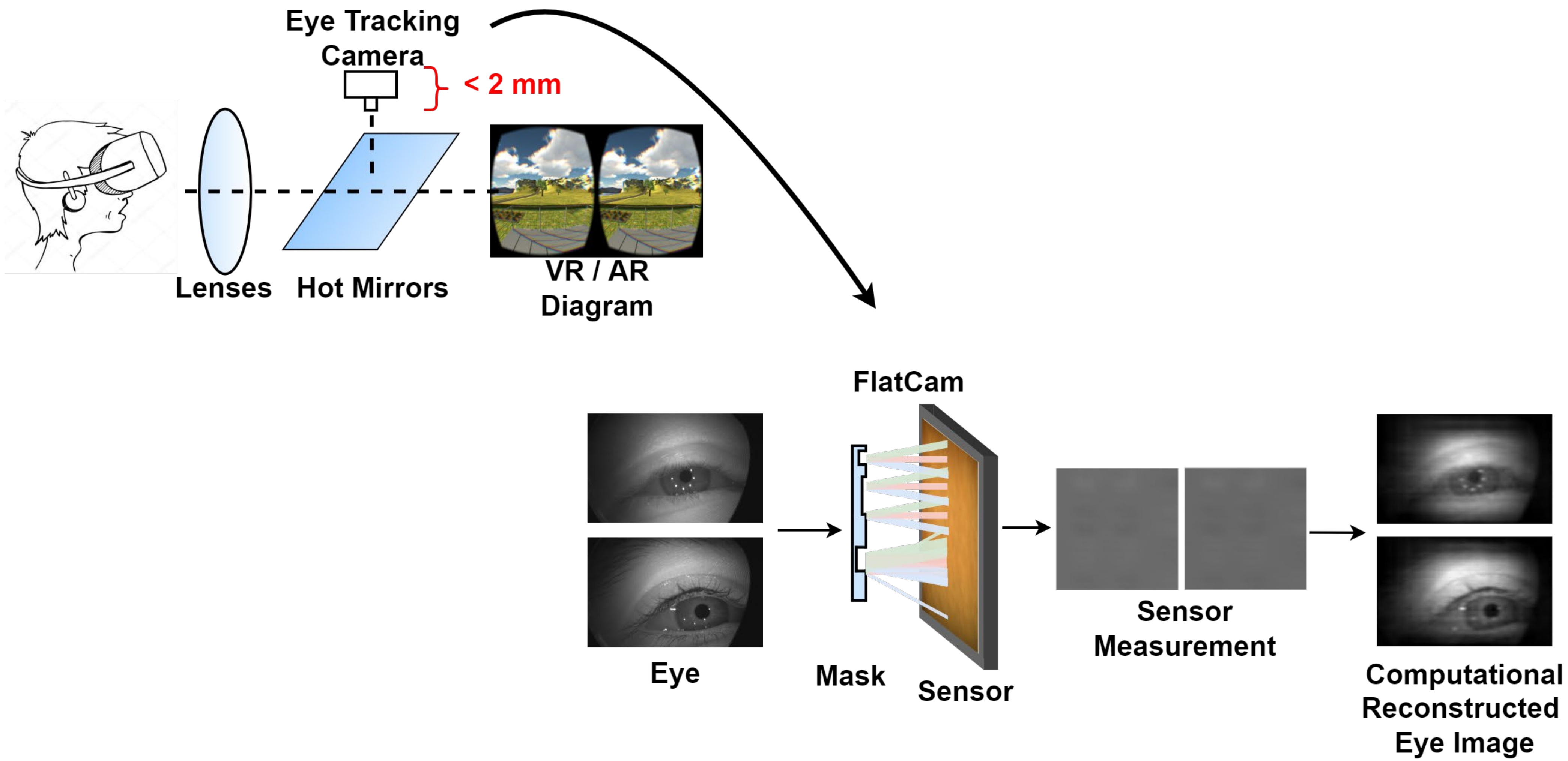
- Challenges to achieve EyeCoD: small form factor vs. large DNNs
  - On the algorithm level, how to track FlatCam captured eye images efficiently without compromising task accuracy?
  - On the hardware level, how to leverage and support EyeCoD algorithm for further boosting the acceleration efficiency?

Our Overall Contributions in EyeCoD

In this work, we

- **Propose the first lensless FlatCam-based eye tracking algorithm & accelerator co-design framework, dubbed EyeCoD**
- **On the system level,** EyeCoD advocates lensless FlatCams instead of lens-based cameras to facilitate small form factor in mobile VR devices
- **On the algorithm level,** EyeCoD integrates a predict-then-focus pipeline to first predict ROIs and then estimate gazes merely based on ROIs, without compromising task accuracy
- **On the hardware level,** EyeCoD further develops a dedicated accelerator attached to FlatCams for accelerating EyeCoD algorithm

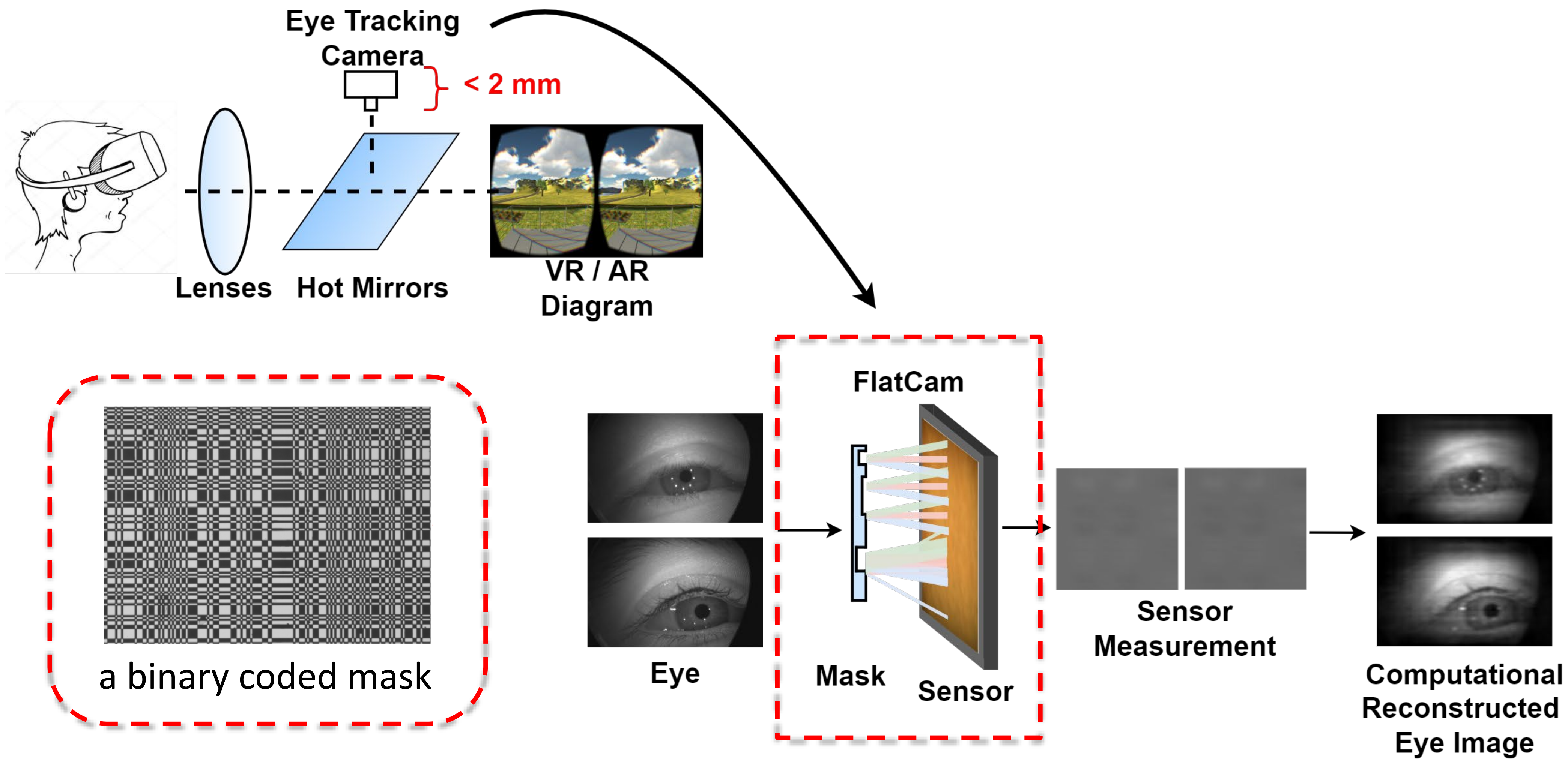
EyeCoD Overview: Eye Tracking System



EyeCoD Overall System:

The core idea on the system level is to replace lens-based cameras with lensless FlatCams → thinner + reduced distance btw cameras and processors

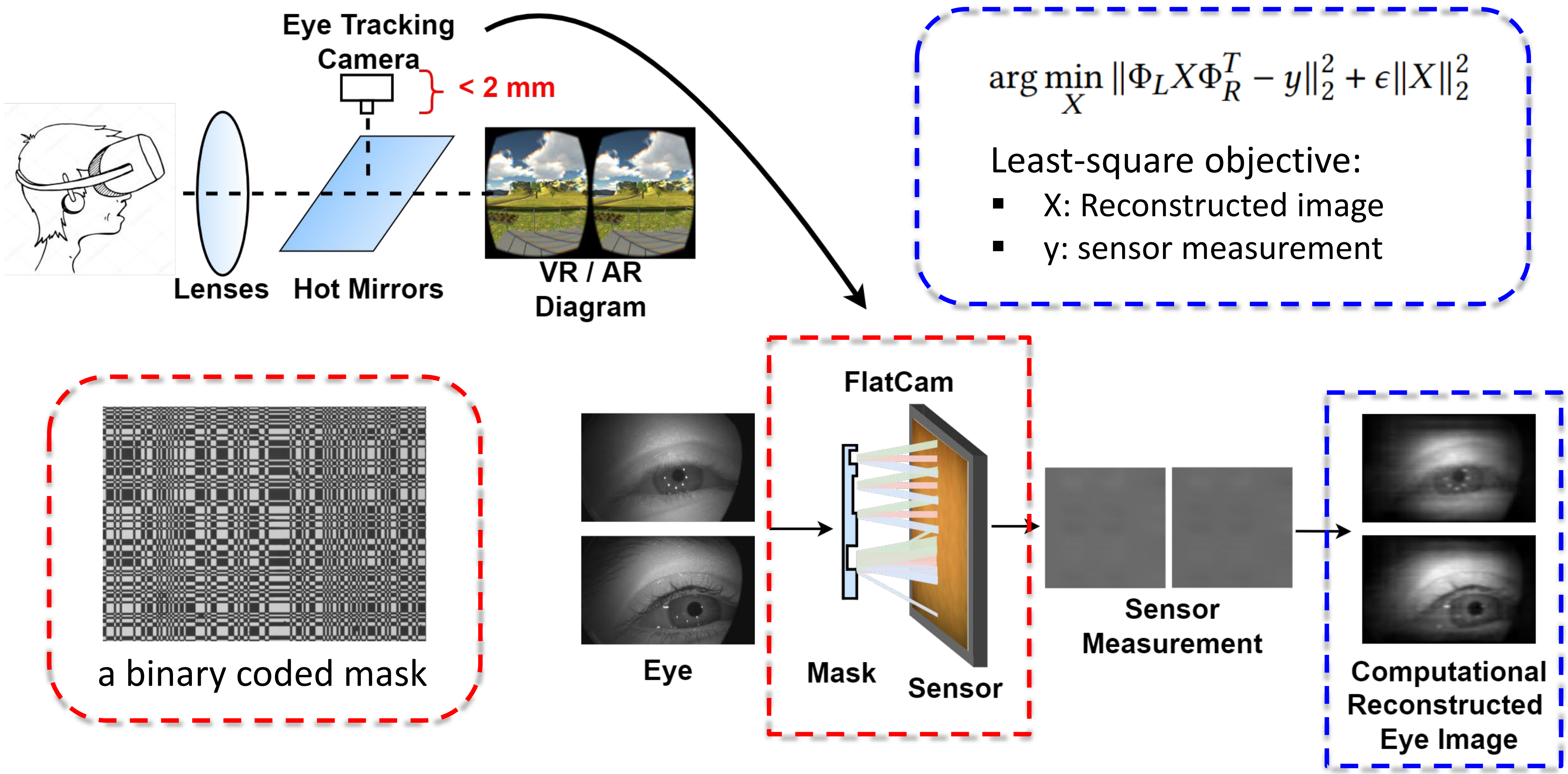
Proposed EyeCoD System for Eye Tracking: Overview



EyeCoD Overall System:

The core idea on the system level is to replace lens-based cameras with lensless FlatCams → thinner + reduced distance btw cameras and processors

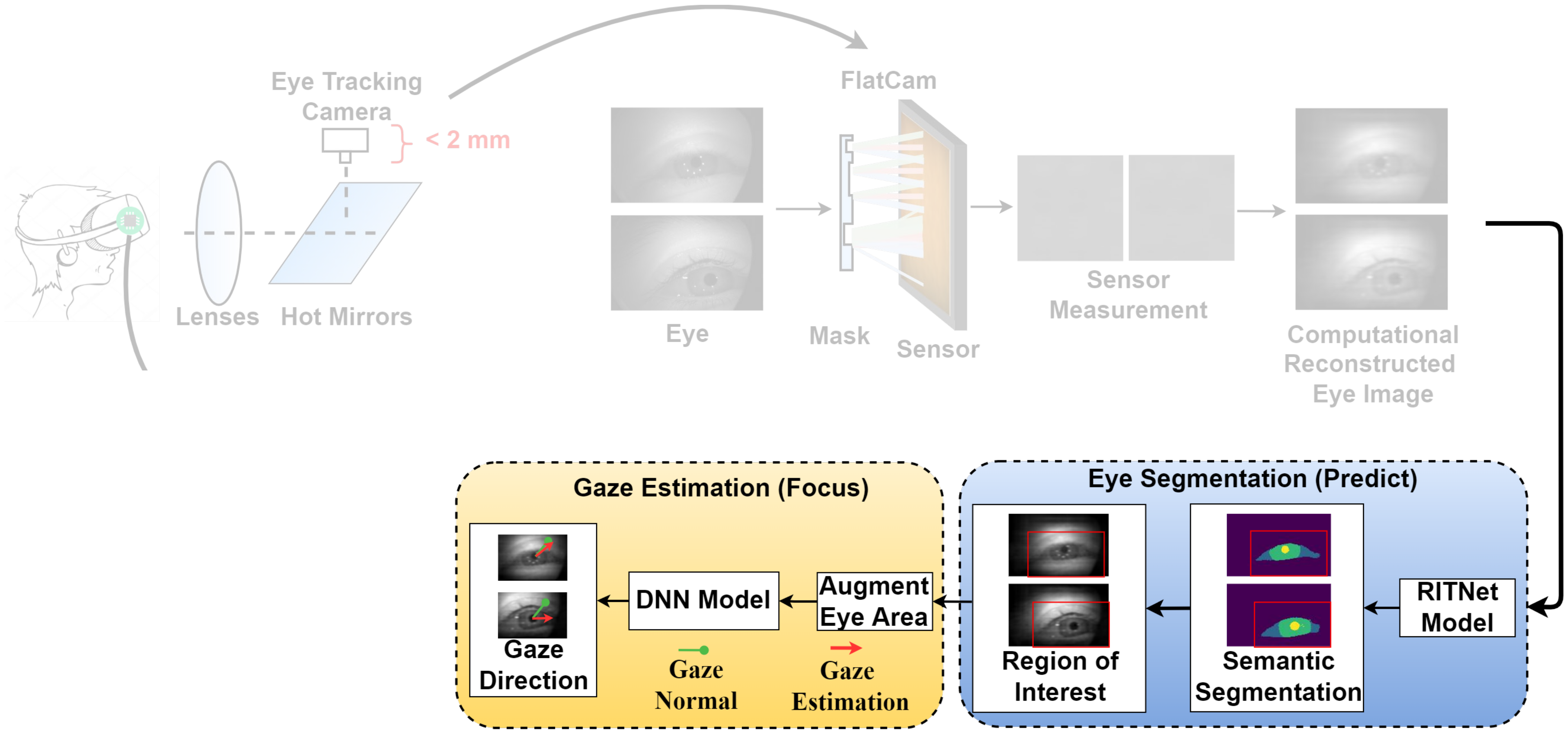
Proposed EyeCoD System for Eye Tracking: Overview



EyeCoD System:

The core idea on the system level is to replace lens-based cameras with lensless FlatCams → thinner + reduced distance btw cameras and processors

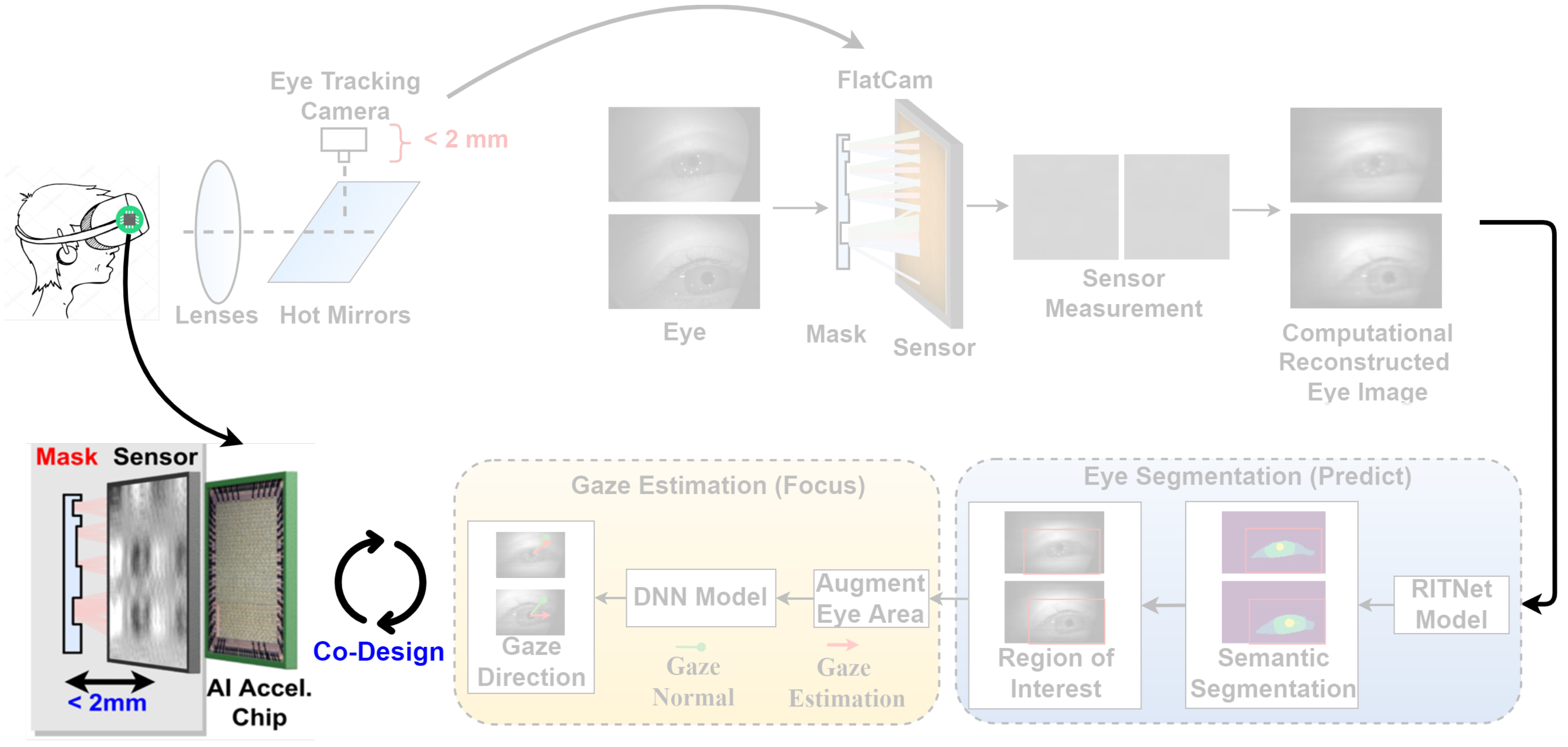
Proposed EyeCoD System for Eye Tracking: Overview



EyeCoD Algorithm:

The core idea on the algorithm level is to first predict the ROIs before estimating the gaze direction → reduced the required computational cost

Proposed EyeCoD System for Eye Tracking: Overview



EyeCoD Accelerator:

The core idea on the accelerator level is to develop a dedicated accelerator attached to FlatCams → accelerated computations and data movements

EyeCoD Algorithm: Predict-then-focus Pipeline

- *Challenge:* How to aggressively reduce the model complexity?
  - **Design insight:**
    - **Perform gaze estimation after extracting ROIs**
  - **EyeCoD leverages the following fact:**
    - **The movement of eyes is much slower than that of gaze direction [7]**

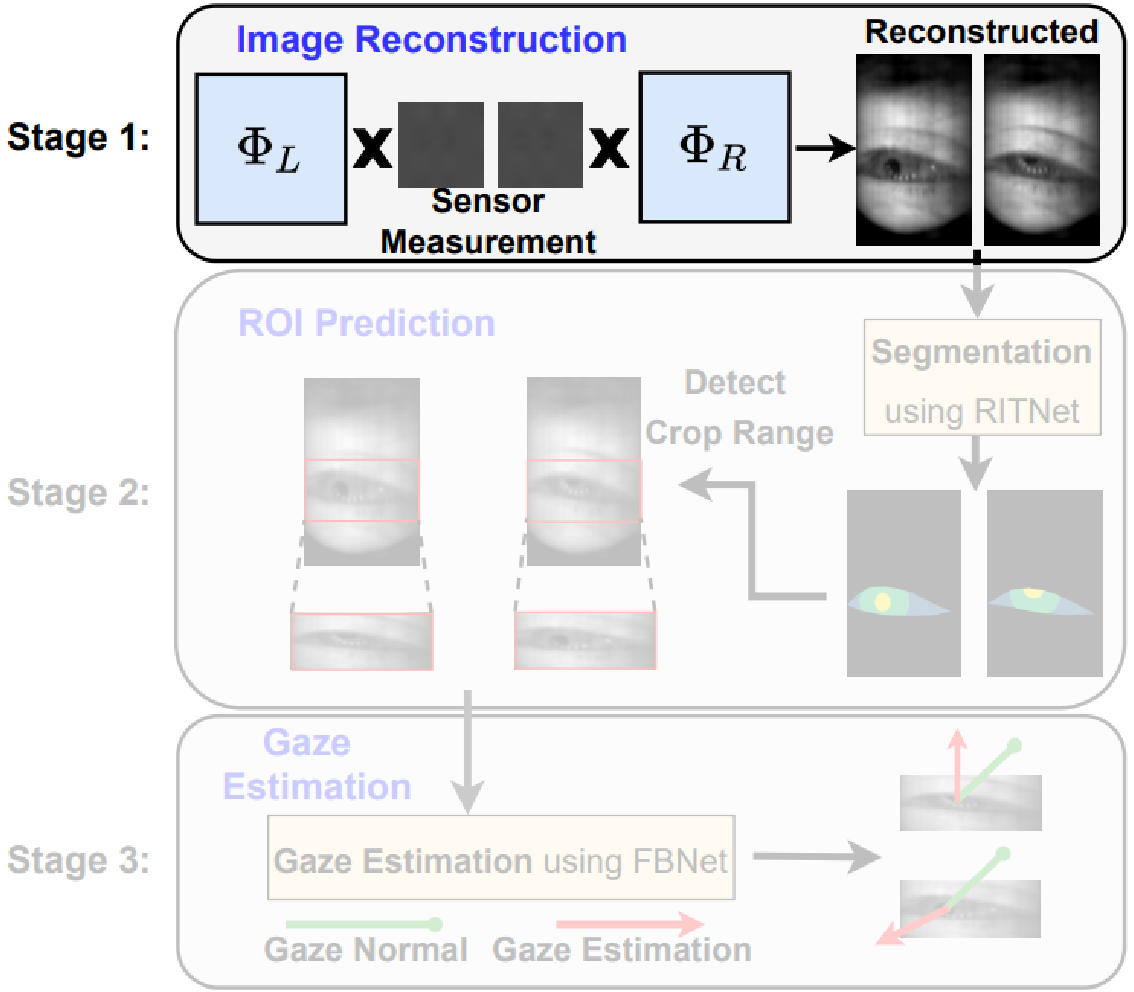
      → ROI prediction is only needed once for every 50 frames

      → Gaze estimation need to be computed every frame

[7] C. Palmero, et. al., Sensor'21

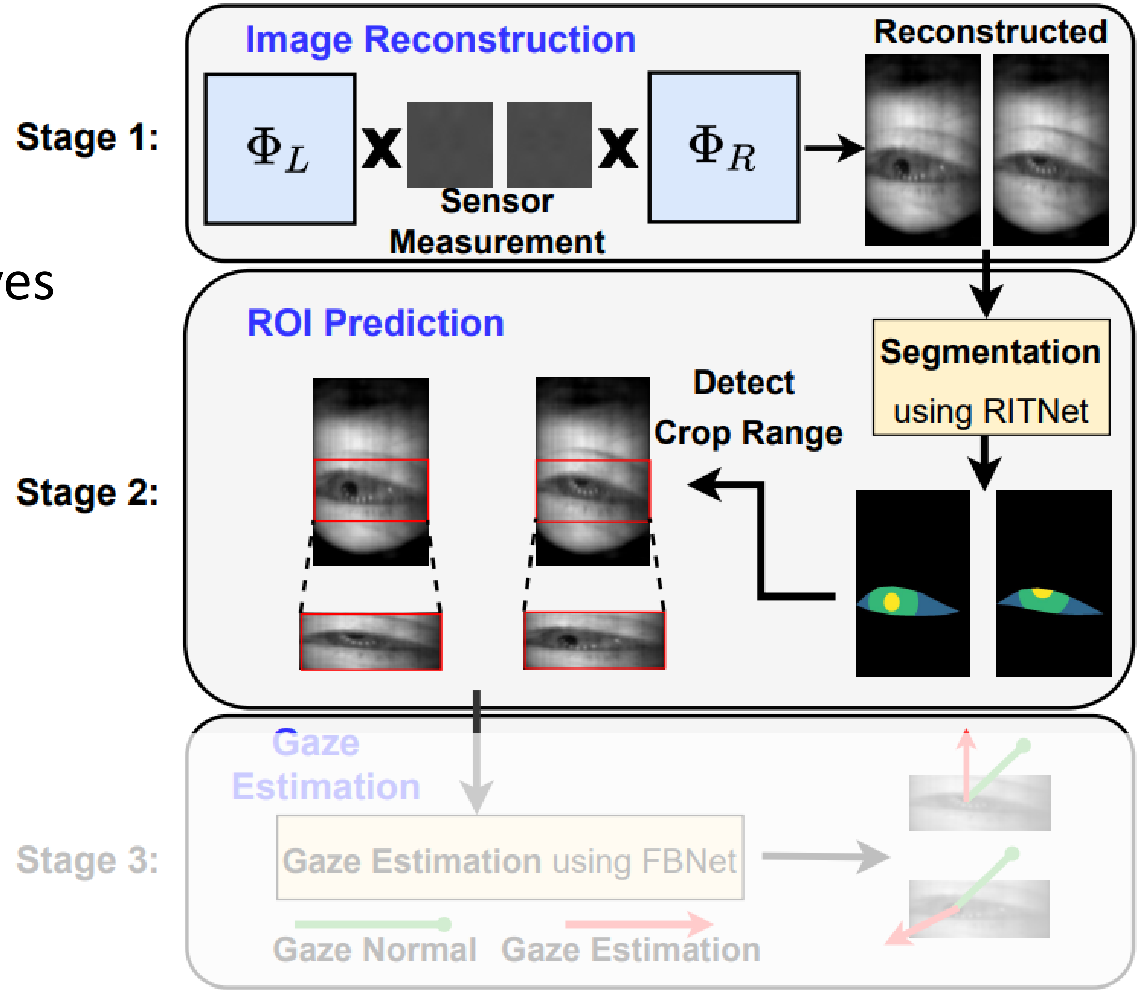
EyeCoD Algorithm: Predict-then-focus Pipeline

- **The proposed predict-then-focus pipeline**
  - **Stage 1: Image reconstruction after FlatCam**
    - Sensing-processing interface: replaces both camera sensors and the first layer of the eye tracking model → FlatCam's coded masks



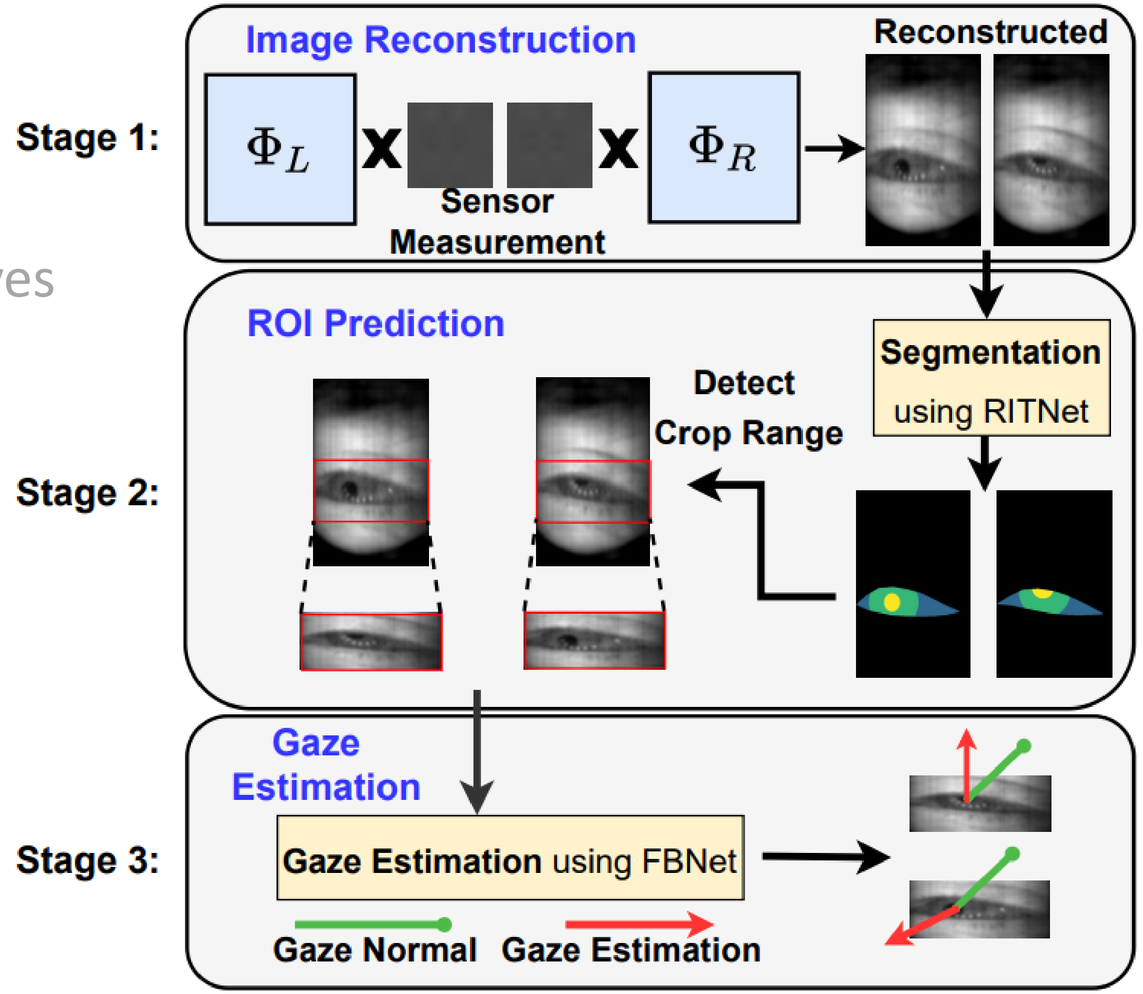
EyeCoD Algorithm: Predict-then-focus Pipeline

- **The proposed predict-then-focus pipeline**
  - Stage 1: Image reconstruction after FlatCam
    - Sensing-processing interface: replaces both camera sensors and the first layer of the eye tracking model → FlatCam's coded masks
  - **Stage 2: ROI prediction**
    - Predict and crop the most informative area of eyes (i.e., pupil, iris, and sclera)
    - Once per 50 frames



EyeCoD Algorithm: Predict-then-focus Pipeline

- **The proposed predict-then-focus pipeline**
  - Stage 1: Image reconstruction after FlatCam
    - Sensing-processing interface replaces both FlatCam sensing and the first layer of following eye tracking models → FlatCam's coded masks
  - Stage 2: ROI prediction
    - Predict and crop the most informative area of eyes (i.e., pupil, iris, and sclera)
    - Once per 50 frames
  - **Stage 3: Gaze estimation**
    - Estimate the gaze direction based on extracted ROIs
    - Perform for each frame

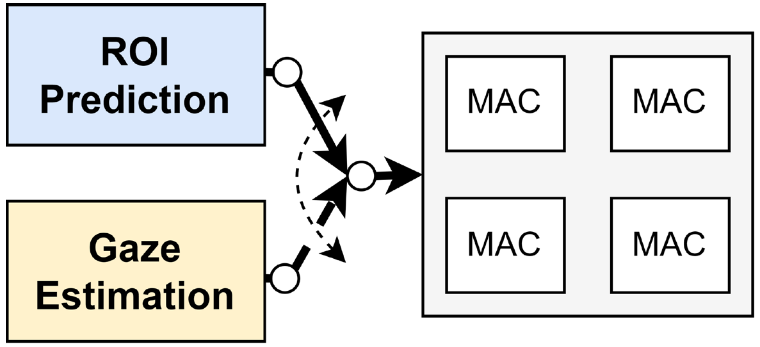


EyeCoD Accelerator

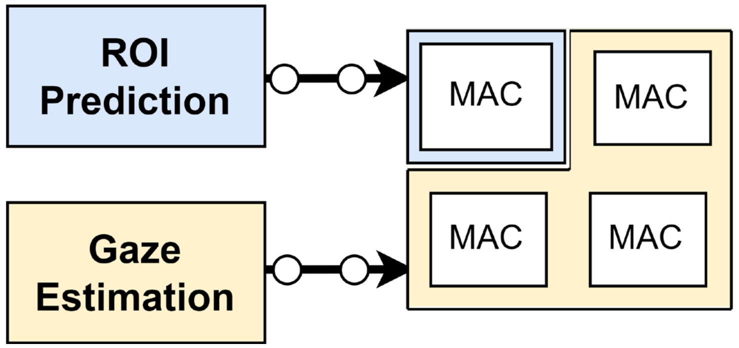
- *Challenge:* How to fully exploit the potential of EyeCoD algorithm?
- Design challenges and considerations
- Our proposed EyeCoD accelerator features:
  - Partial time-multiplexing mode for *workload orchestration*
  - Intra-channel reuse for *depth-wise conv layers' hardware utilization*
  - Dedicated support for *activation partition and cross layer processing*

EyeCoD Accelerator: Design Challenge 1

- *Challenge*: How to fully exploit the potential of EyeCoD algorithm?
- Design challenges and considerations
  - Workload orchestration
    - ✗ Time-multiplexing mode
    - ✗ Concurrent mode



Illustrating Time-multiplexing Mode



Illustrating Concurrent Mode

☺ High reuse opportunity

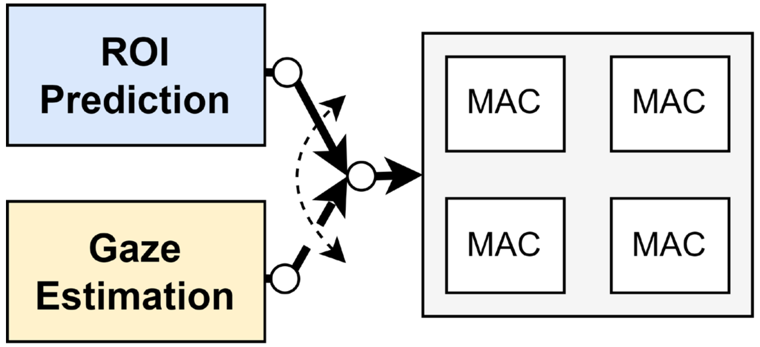
☹ Peak resource usage for ROI prediction

☺ Amortizing ROI prediction workload

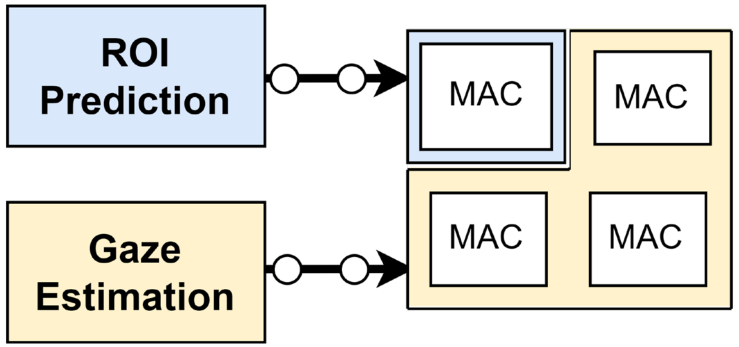
☹ Low reuse opportunity

EyeCoD Accelerator: Design Challenge 1

- ***Challenge*: How to fully exploit the potential of EyeCoD algorithm?**
- **Design challenges and considerations**
  - **Workload orchestration**
    - ✗ **Time-multiplexing mode**
    - ✗ **Concurrent mode**



Illustrating Time-multiplexing Mode



Illustrating Concurrent Mode

☺ **High reuse opportunity**

☹ **Peak resources usage for ROI prediction**

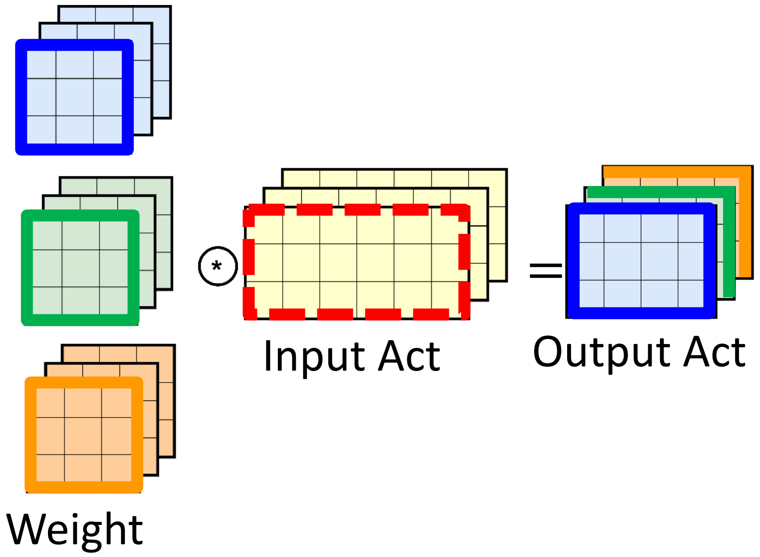
☺ **Amortizing ROI prediction workload**

☹ **Low reuse opportunity**

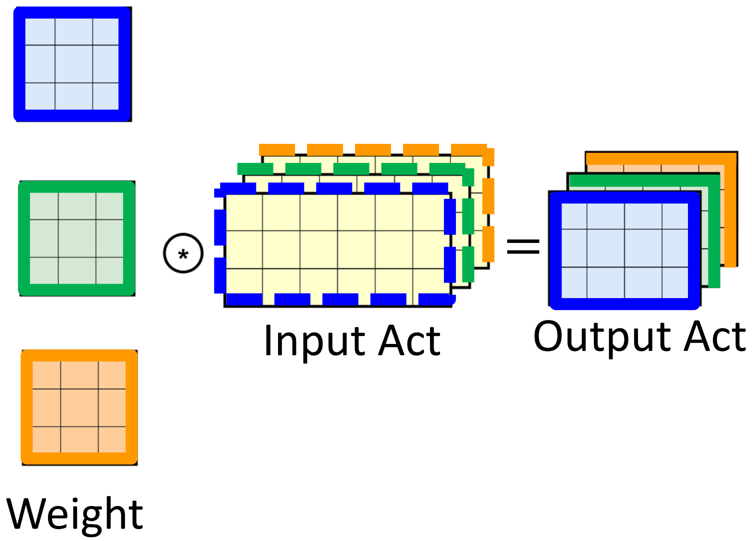
➢ **Can we marry the best of both modes?**

EyeCoD Accelerator: Design Challenge 2

- *Challenge*: How to fully exploit the potential of EyeCoD algorithm?
- **Design challenges and considerations**
  - Workload orchestration
  - **Depthwise conv layers (DW): Reduced mode size yet low utilization**
    - ✓ 7.9% FLOPs of the whole workload
    - ✗ yet 34% overall processing time



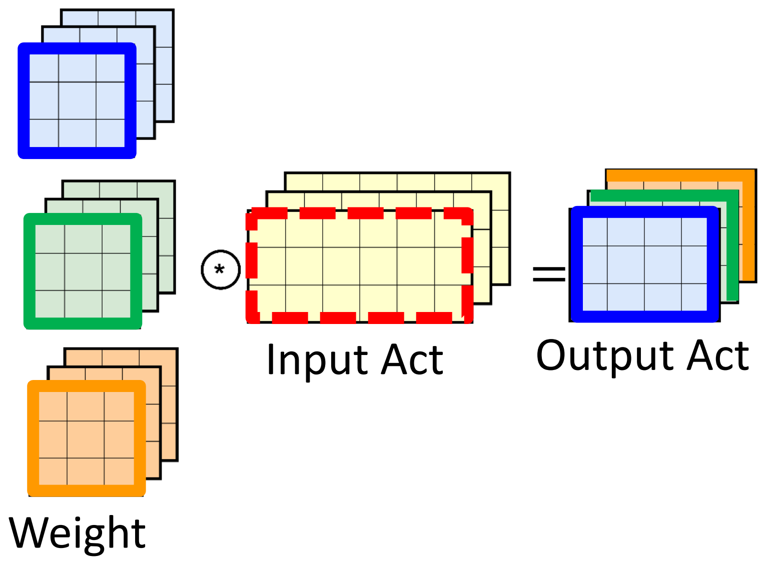
Generic/Point-wise Conv Layer



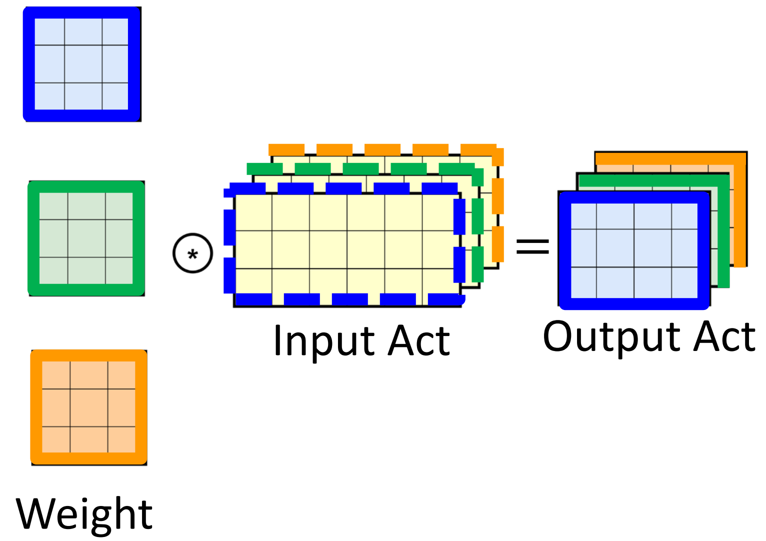
Depth-wise Conv Layer (DW)

EyeCoD Accelerator: Design Challenge 2

- *Challenge*: How to fully exploit the potential of EyeCoD algorithm?
- **Design challenges and considerations**
  - Workload orchestration
  - **Depthwise conv layers (DW): Reduced mode size yet low utilization**
    - ✓ 7.9% FLOPs of the whole workload
    - ✗ yet 34% overall processing time



Generic/Point-wise Conv Layer

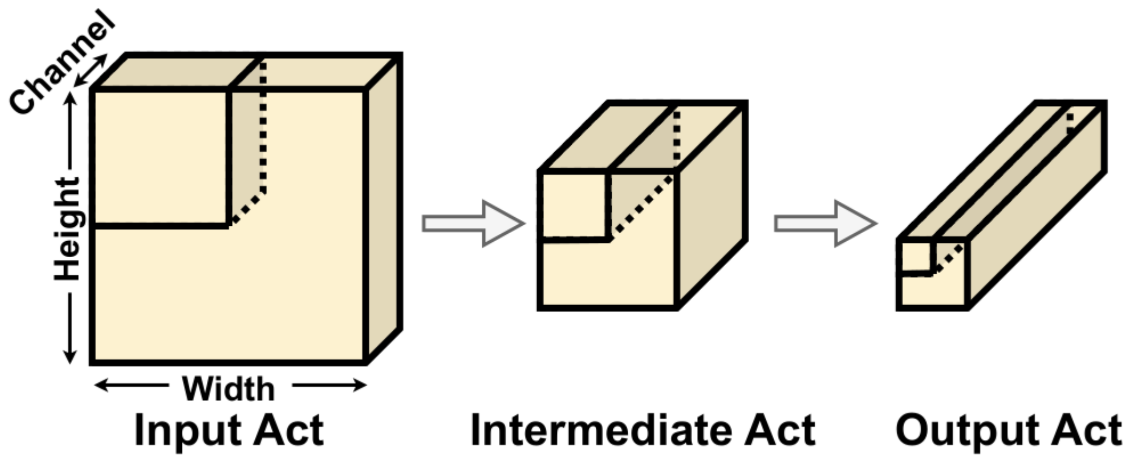


Depth-wise Conv Layer (DW)

➢ **Can we improve the input activation reuses → high MAC utilization?**

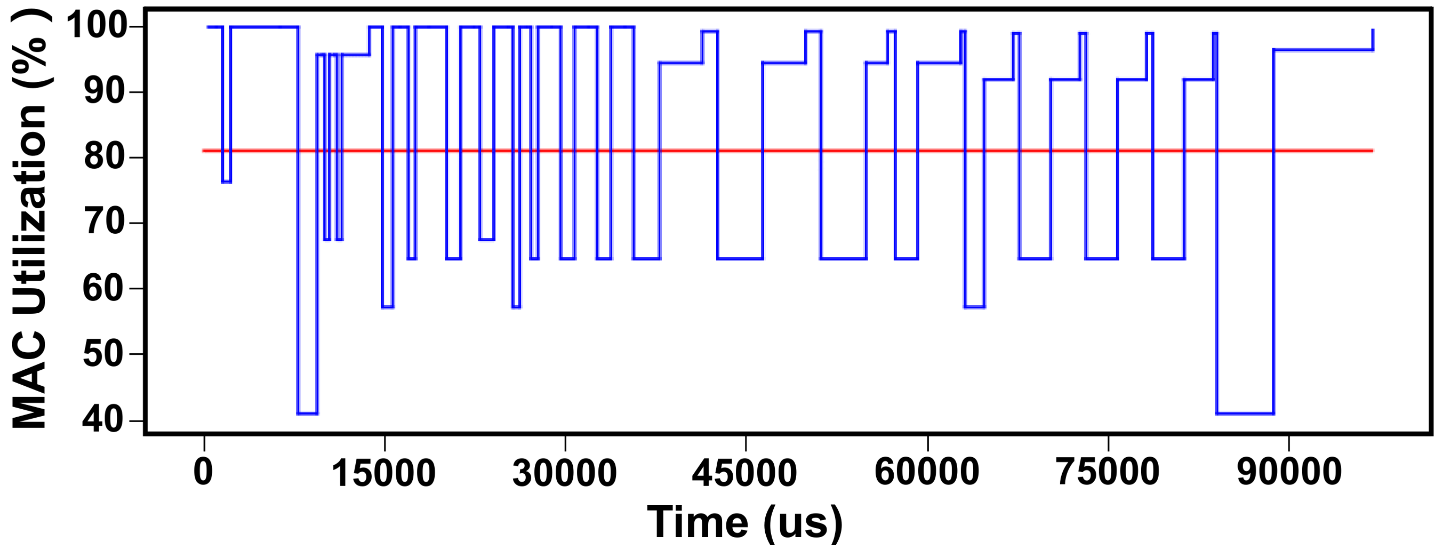
EyeCoD Accelerator: Design Challenge 3

- *Challenge*: How to fully exploit the potential of EyeCoD algorithm?
- **Design challenges and considerations**
  - Workload orchestration
  - Depthwise conv layers (DW): Reduced mode size yet low utilization
  - **Dedicated support for activation partition and cross layer processing**



EyeCoD Accelerator: Feature 1

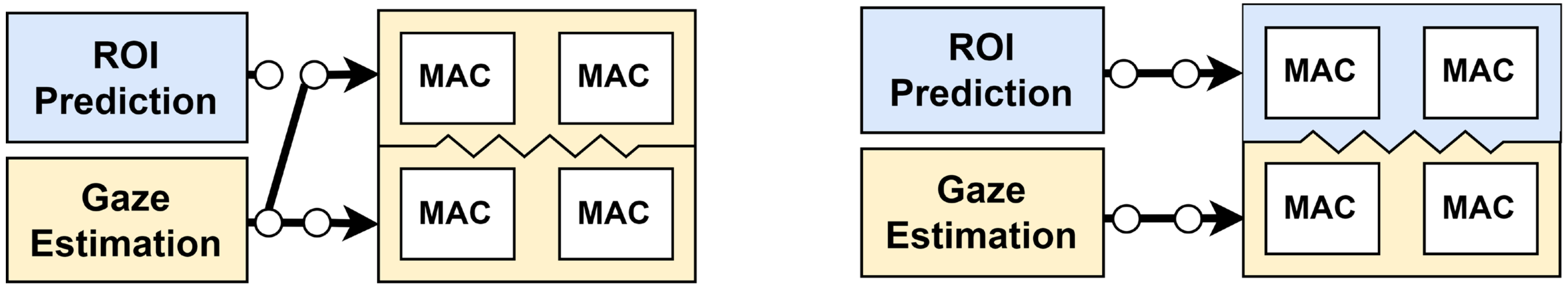
- Design challenges and considerations
- **Our proposed EyeCoD accelerator features:**
  - **Partial time-multiplexing mode for workload orchestration**
    - **Observation: Fluctuated utilization for gaze estimation**



Visualizing the temporal MAC utilization of the gaze estimation

EyeCoD Accelerator: Feature 1

- Design challenges and considerations
- **Our proposed EyeCoD accelerator features:**
  - **Partial time-multiplexing mode for workload orchestration**
    - **Observation: The utilization for gaze estimation fluctuate**
    - **Proposed: Amortize ROI prediction workload to underutilized MACs**



Gaze estimation only

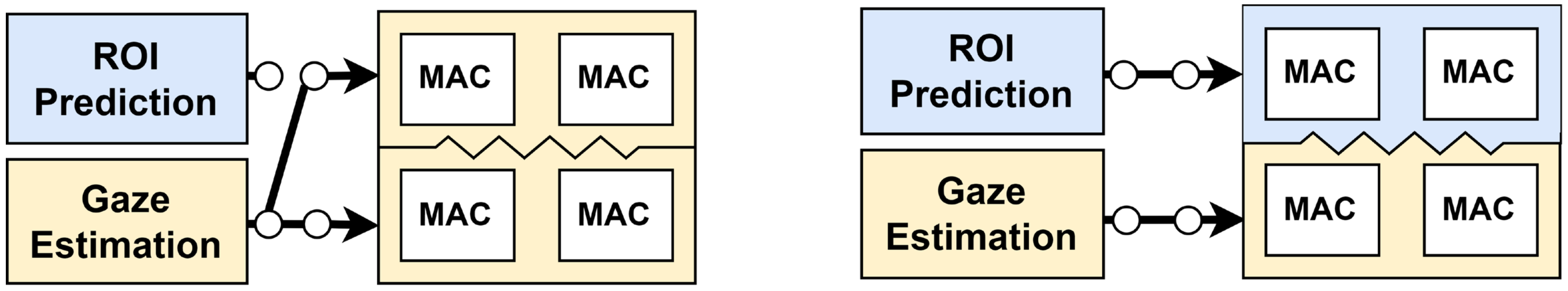
Concurrent ROI prediction and gaze estimation

☺ **Amortize ROI prediction workload**

☺ **Higher reuse opportunity**

EyeCoD Accelerator: Feature 1

- Design challenges and considerations
- **Our proposed EyeCoD accelerator features:**
  - **Partial time-multiplexing mode for workload orchestration**
    - **Observation: The utilization for gaze estimation fluctuate**
    - **Proposed: Amortize ROI prediction workload to underutilized MACs**



Gaze estimation only

Concurrent ROI prediction and gaze estimation

☺ **Amortize ROI prediction workload**

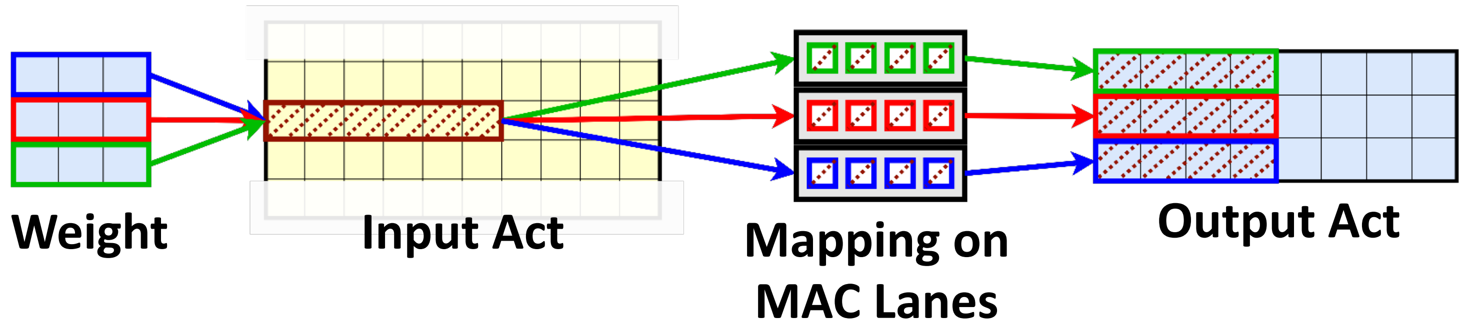
→ **2.31× speed up over the time-multiplexing mode**

☺ **Higher reuse opportunity**

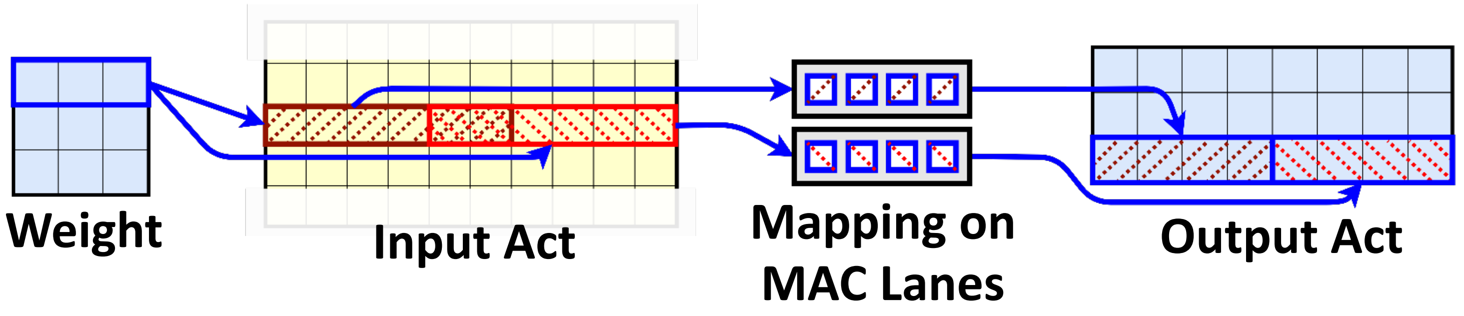
→ **1.6× higher energy efficiency over the concurrent mode**

EyeCoD Accelerator: Feature 2

- Design challenges and considerations
- **Our proposed EyeCoD accelerator features:**
  - Partial time-multiplexing mode for workload orchestration
  - Intra-channel reuse for boosting depth-wise conv layers' ultilization
    - Column-wise intra-channel reuse → 3× utilizaiotn
    - Deeper row-wise intra-channel reuse → 2× utilization



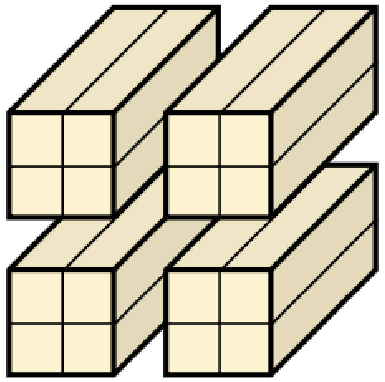
Column-wise
Intra-channel Reuses

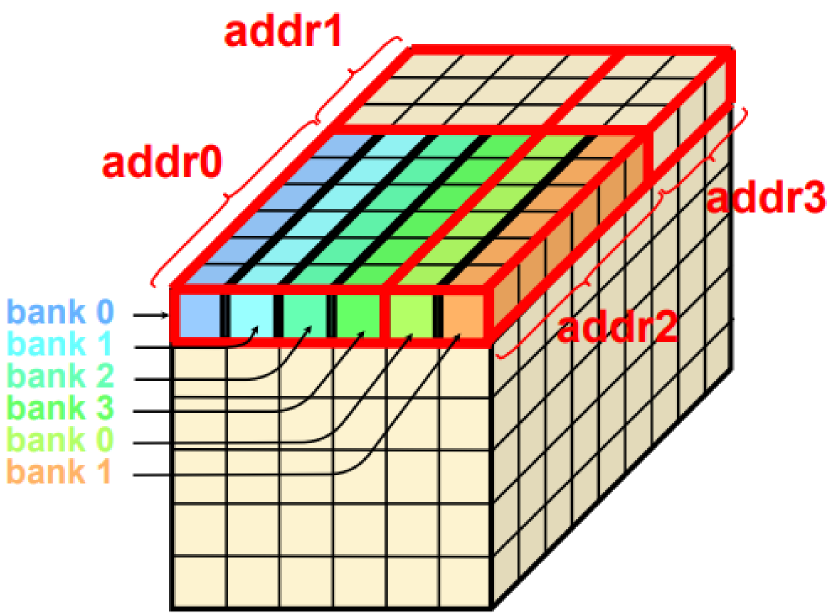


Deeper row-wise
Intra-channel Reuses

EyeCoD Accelerator: Feature 3

- Design challenges and considerations
- **Our proposed EyeCoD accelerator features:**
  - Partial time-multiplexing mode for workload orchestration
  - Intra-channel reuse for boosting depth-wise conv layers' utilization
  - Dedicated support for activation partition and cross layer processing
    - Support versatile operations:
      - Partition operation
      - Concatenation operation
      - Up/Down-sampling operation

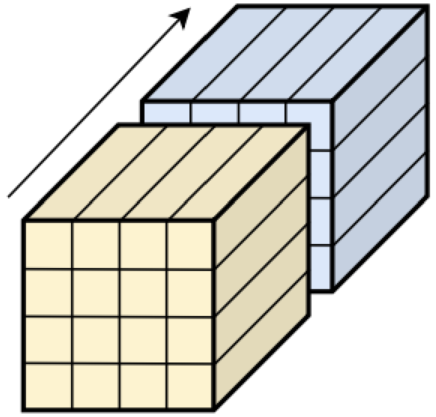


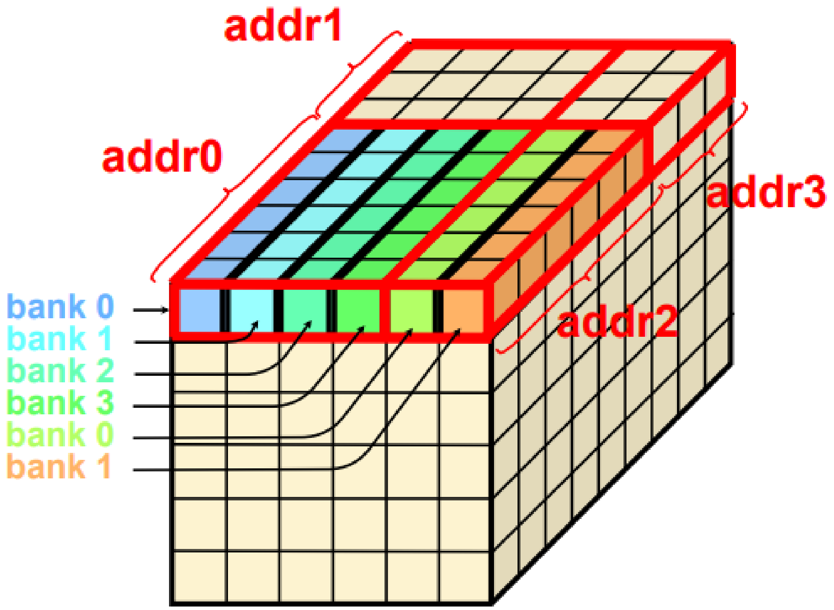


Proposed Activation Memory Storage Layout (i.e., Address) (An Example for a 6×6×24 Act Tensor)

EyeCoD Accelerator: Feature 3

- Design challenges and considerations
- **Our proposed EyeCoD accelerator features:**
  - Partial time-multiplexing mode for workload orchestration
  - Intra-channel reuse for boosting depth-wise conv layers' utilization
  - Dedicated support for activation partition and cross layer processing
    - Support versatile operations:
      - Partition operation
      - Concatenation operation
      - Up/Down-sampling operation

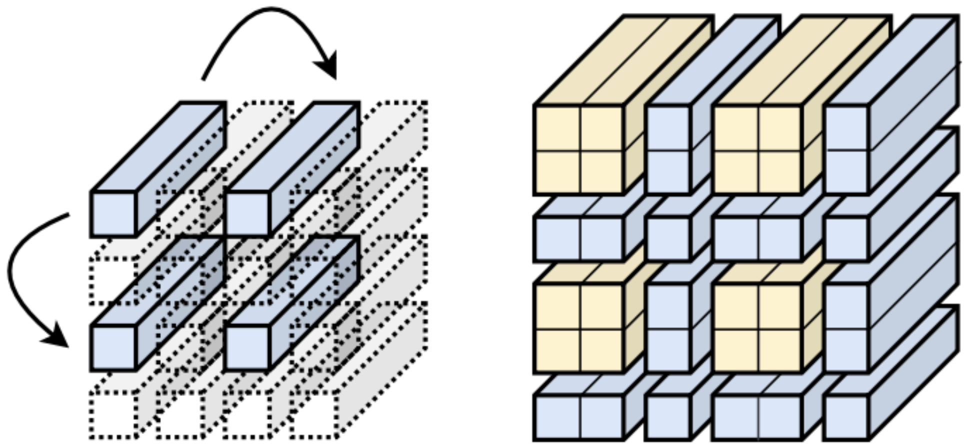


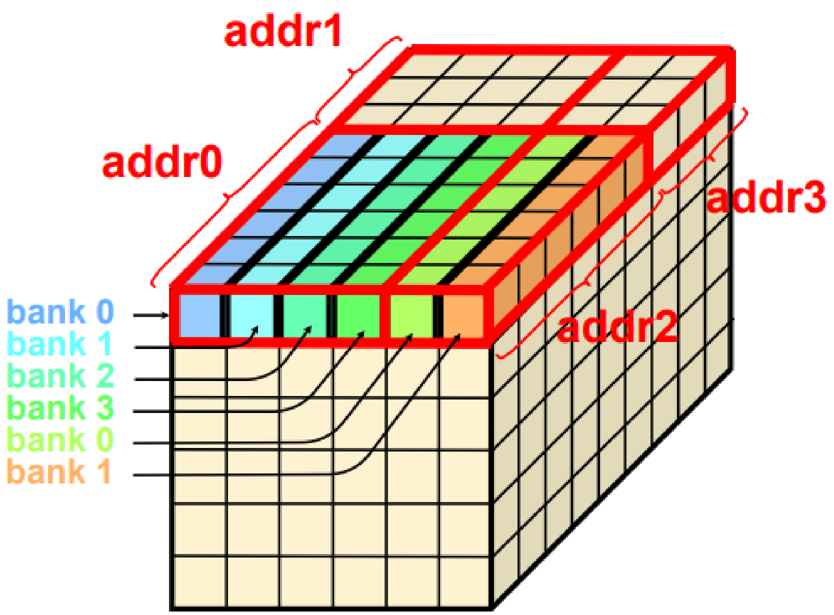


Proposed Activation Memory Storage Layout (i.e., Address) (An Example for a 6×6×24 Act Tensor)

EyeCoD Accelerator: Feature 3

- Design challenges and considerations
- **Our proposed EyeCoD accelerator features:**
  - Partial time-multiplexing mode for workload orchestration
  - Intra-channel reuse for boosting depth-wise conv layers' utilization
  - Dedicated support for activation partition and cross layer processing
    - Support versatile operations:
      - Partition operation
      - Concatenation operation
      - Up/Down-sampling operation

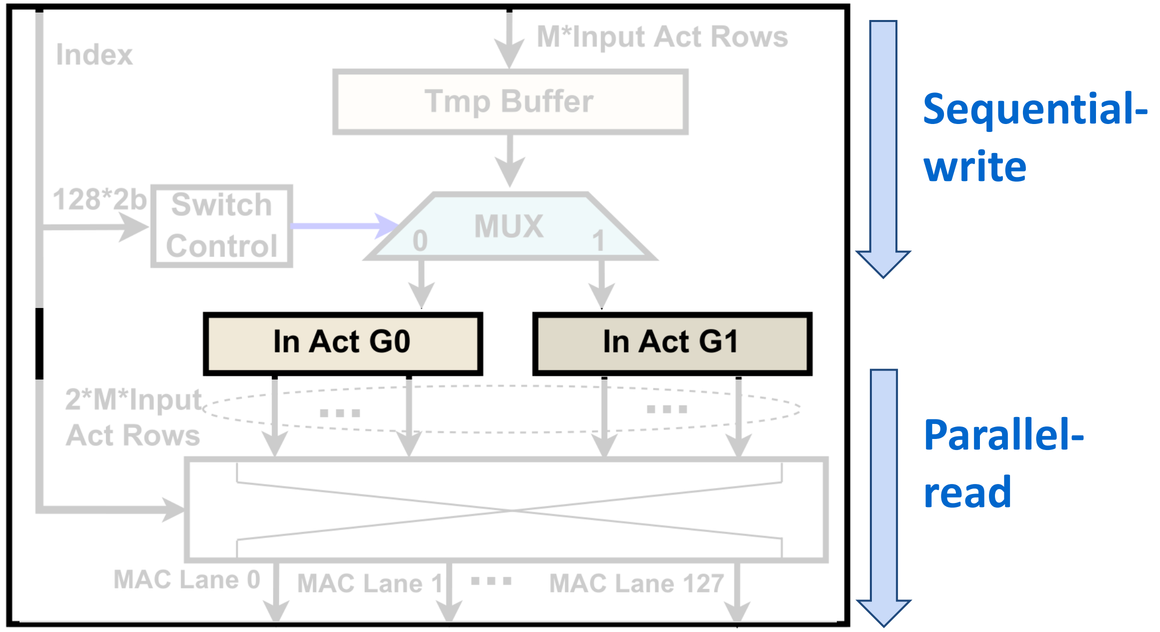




Proposed Activation Memory Storage Layout (i.e., Address) (An Example for a 6×6×24 Act Tensor)

EyeCoD Accelerator: Feature 3

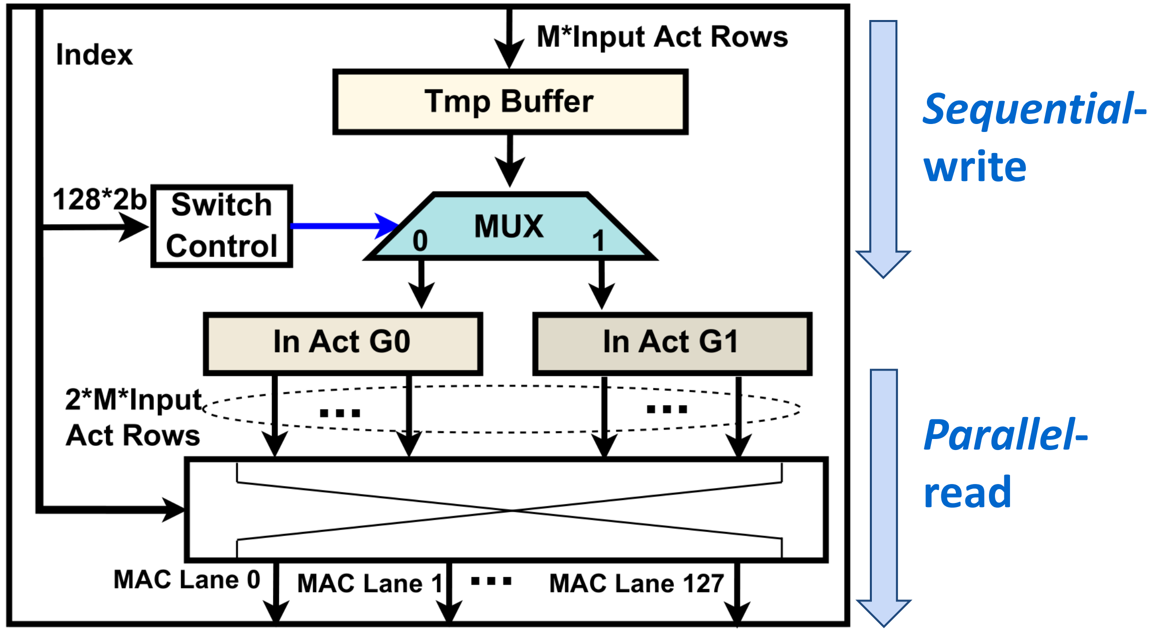
- Design challenges and considerations
- **Our proposed EyeCoD accelerator features:**
  - Partial time-multiplexing mode for workload orchestration
  - Intra-channel reuse for depth-wise convolution layers
  - Dedicated support for activation partition and cross layer processing



Proposed Sequential-write-parallel-read Activation Buffer for 2× Higher Activation Bandwidth

EyeCoD Accelerator: Feature 3

- Design challenges and considerations
- **Our proposed EyeCoD accelerator features:**
  - Partial time-multiplexing mode for workload orchestration
  - Intra-channel reuse for depth-wise convolution layers
  - Dedicated support for activation partition and cross layer processing



Proposed Sequential-write-parallel-read Activation Buffer for 2× Higher Activation Bandwidth

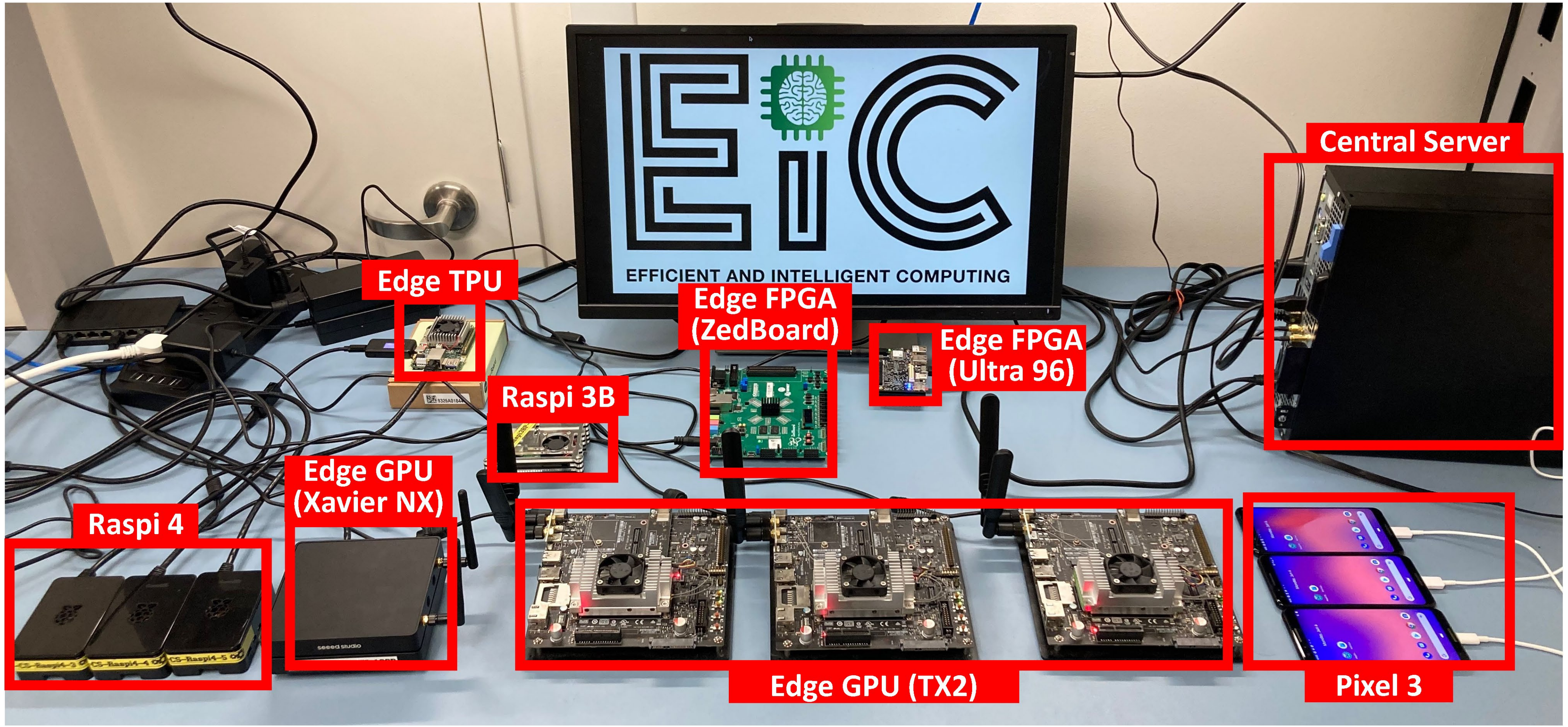
Evaluation Setup and Baselines

- Evaluation Setup
  - **Considered Models**:
    - RITNet for eye segmentation
    - FBNet-C100 for gaze estimation
  - **Eye Tracking Datasets**:
    - OpenEDS 2019 for eye segmentation
    - OpenEDS 2020 for gaze estimation
  - **Evaluation Metrics**:
    - Gaze estimation accuracy
    - Model FLOPs, and task throughput and energy efficiency
  - **Benchmark Baselines**:
    - EdgeCPU (Raspberry Pi) and CPU (AMD EPYC 7742)
    - EdgeGPU (Nvidia Jetson TX2) and GPU (Nvidia 2080Ti)
    - Prior eye tracking accelerator: CIS-GEP [8]

[8] K. Bong, et. al., JSSC'16

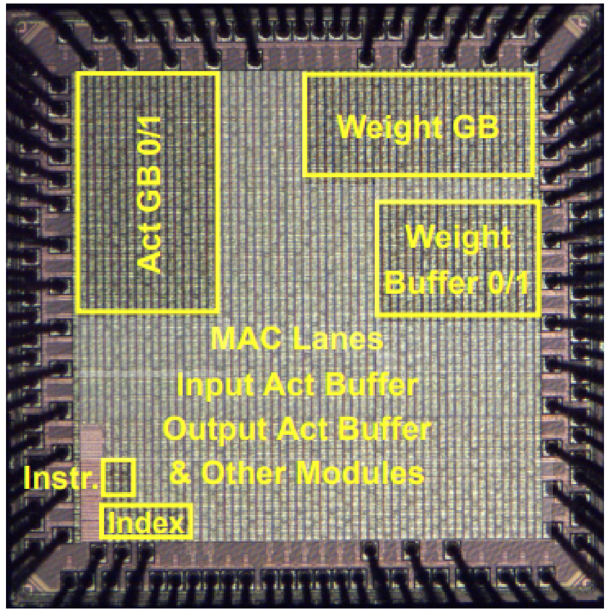
Evaluation Setup and Baselines

- Evaluation Setup
  - Benchmark Baselines:
    - EdgeCPU (Raspberry Pi) and CPU (AMD EPYC 7742)
    - EdgeGPU (Nvidia Jetson TX2) and GPU (Nvidia 2080Ti)
    - Eye tracking processor: CIS-GEP [8]



Evaluation Setup and Baselines

- **Evaluation Setup**
  - **EyeCoD AI Chip and Configurations:**
    - **Silicon prototype:**



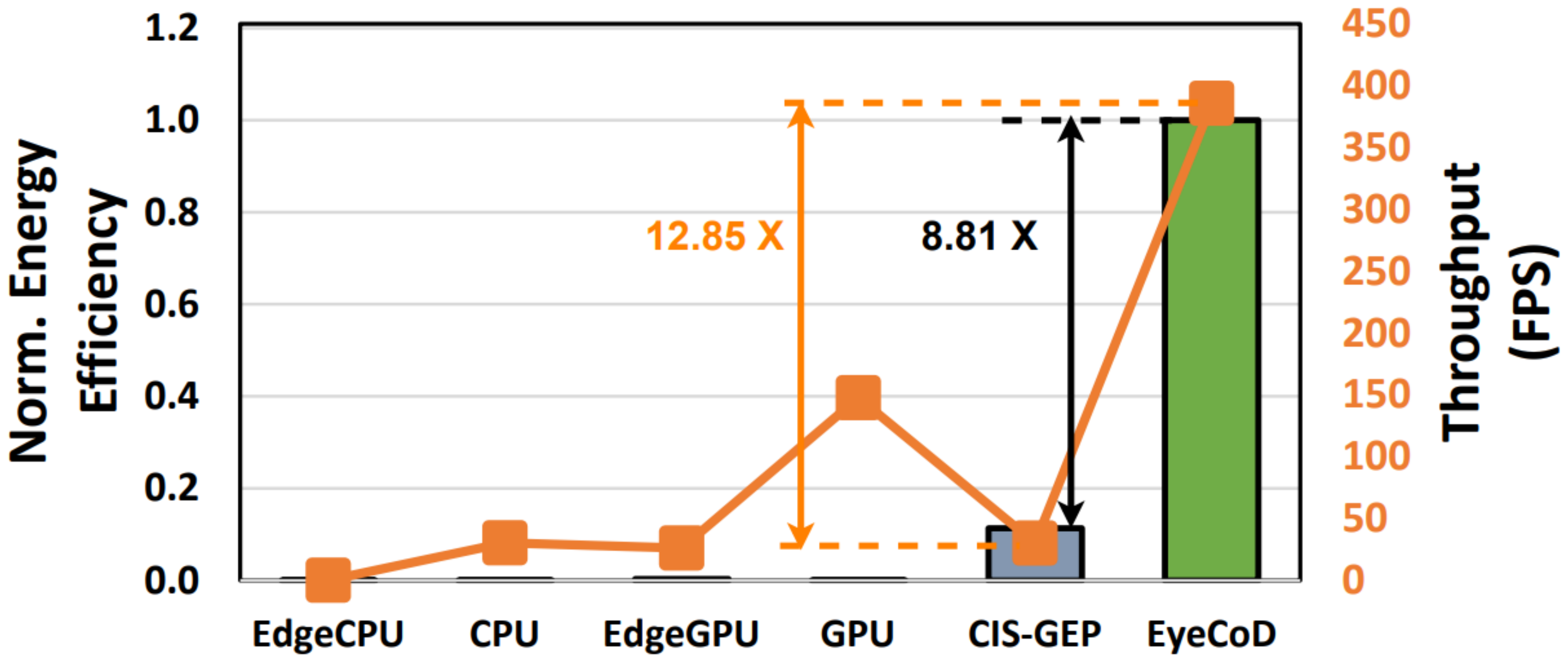
Technology	28nm
Chip Area	3.00 mm^2
Supply Voltage	0.51-0.8 V (Core) 0.59-0.88 V (Mem)
Core Frequency	370 MHz @ (0.8V, 0.88V)
Total SRAM	316KB
# of MACs	512
Power	154.32 mW @ (0.8V. 0.88V), 370 MHz

    - **Accelerator configurations:**

Act GB0/GB1	Weight Buffer0/1	Weight GB	Index SRAM	Instr. SRAM
512KB * 2	64KB * 2	512KB	20KB	4KB

MAC Lanes	MACs/MAC Lane	Area	Clock frequency	Power
128	8	8 mm^2	370MHz	335mW

Evaluation: EyeCoD over SOTA Accelerators



- **EyeCoD over CPU/GPU platforms:**
  - EyeCoD achieves up to 2966x, 12.7x, 14.8x, and 2.61x throughput improvements over EdgeCPU, CPU, EdgeGPU, and GPU
- **EyeCoD over SOTA eye tracking accelerators:**
  - EyeCoD achieves on average 12.8x throughput improvement and 8.1x higher energy efficiency over CIS-GEP, respectively.

Evaluation of EyeCoD Algorithm Pipeline

Model	Resolution	Eye Segmentation mIOU		FLOPs
		Origin Image	FlatCam Image	
U-net	512×512	93.3	92.5	14.1G
RITNet	512×512	95.1	93.6	17.0G
RITNet	256×256	94.7	93.8	4.1G
RITNet (8-bit)	256×256	94.0	92.8	0.3G
RITNet	128×128	94.1	93.5	1.0G
RITNet (8-bit)	128×128	93.3	**92.7**	**0.1G**

- **ROI prediction based on eye segmentation model**
  - EyeCoD achieves up to 16x FLOPs reduction over the SOTA RITNet with a comparable (~93%) mIOU on FlatCam captured images

  → Validate the effectiveness of EyeCoD's ROI prediction

Evaluation of EyeCoD Algorithm Pipeline

Model	Camera	Resolution	Error	Parameter	FLOPs
ResNet18	Lens	224×224	3.17	11.18M	1.82 G
ResNet18	FlatCam	96×160	3.27	11.18M	0.56G
MobileNet			3.43	2.23M	0.10G
FBNet-C100			3.23	3.59M	0.12G
FBNet-C100 (8-bit)			**3.23**	**3.59M**	**0.01G**

- **Gaze estimation on top of the predicted ROIs**
  - EyeCod with FBNet-C100 (8-bit) achieves 0.04 error reduction while reducing 78.2% FLOPs, compared with the award winner using ResNet18

→ Validate the effectiveness of EyeCoD algorithm pipeline

Evaluation of Our EyeCoD Accelerator

System	Throughput (FPS)	Norm. Energy Eff.
Lens-based System*	96.34	1.00
EyeCoD w/ P.F.*	191.94	1.99
EyeCoD w/ P.F. & Input.	233.64	2.43
EyeCoD w/ P.F. & Input. & Partial.	299.04	3.10
EyeCoD w/ P.F. & Input. & Partial. & Depth.	385.66	4.00

- * : Using time-multiplexing mode
- **P.F.** : EyeCoD w/ predict-then-focus pipeline
- **Input.** : Sequential-write-parallel-read input activation buffer design
- **Partial.** : Partial time-multiplexing workload orchestration
- **Depth.** : Intra-channel reuse for depth-wise layers

- **Overall throughput or energy efficiency improvements:**
  - EyeCoD achieves 4x over lens-based eye tracking system

- **Breakdown analysis:**
  - **P.F.** leads to 1.99x improvements, **Input., Partial., and Depth.** further offers 1.22x, 1.28x, and 1.29x improvements, respectively.

Summary

EyeCoD integrates *system-*, *algorithm-*, and *accelerator*-level innovations:

- **The first FlatCam based algorithm & accelerator co-design framework for eye tracking that can simultaneously meet all three requirements for next-generation AR/VR devices**
- **On the algorithm level,** EyeCoD integrates a predict-then-focus pipeline to largely reduce the computational cost without compromising the task accuracy;
- **On the hardware level,** EyeCoD further develops a dedicated accelerator attached to FlatCams for accelerating both computations and data movements.

Acknowledge: NSF RTML & EPCN programs

Demonstration

ShiftAddNAS: Hardware-Inspired Search for More Accurate and Efficient Neural Networks

Haoran You, Baopu Li, Huihong Shi, Yonggan Fu, Yingyan Lin

ICML 2022

NASA: Neural Architecture Search and Acceleration for Hardware Inspired Hybrid Networks

Huihong Shi, **Haoran You**, Yang Zhao, Zhongfeng Wang, Yingyan Lin

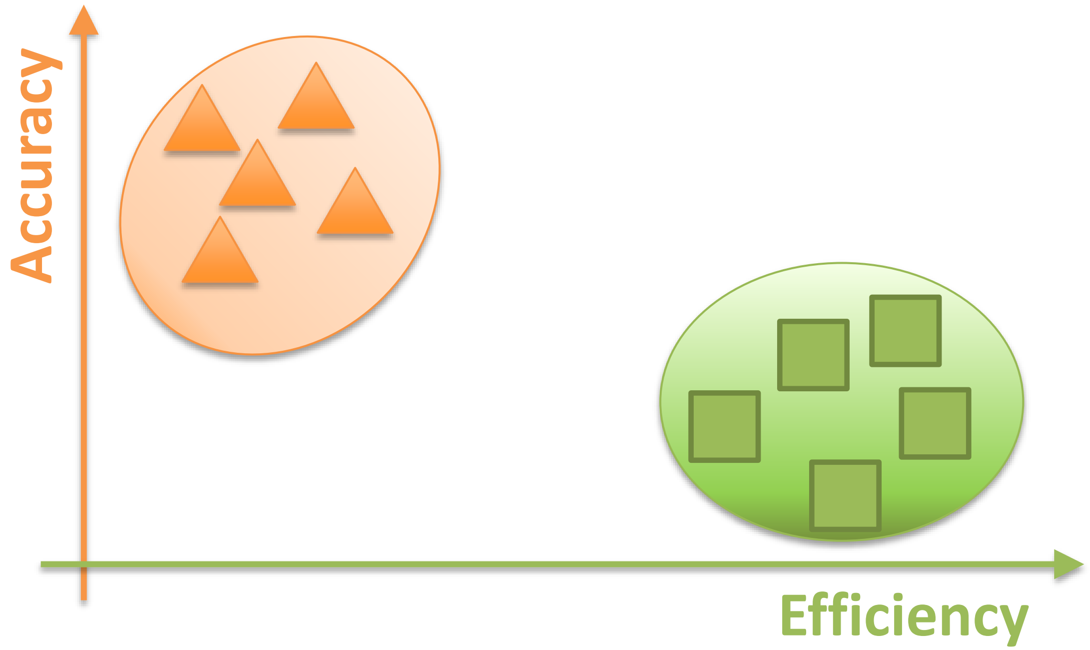
ICCAD 2022

ShiftAddNAS: Background and Motivation

- **Two branches of SOTA DNN design: Trade off accuracy and efficiency**
  - ***Multiplication-based* DNNs, e.g., CNNs, Transformers**
    - ☺ **Achieve unprecedented task accuracy**
    - ☹ **Power hungry → Challenge their deployment to edge devices**

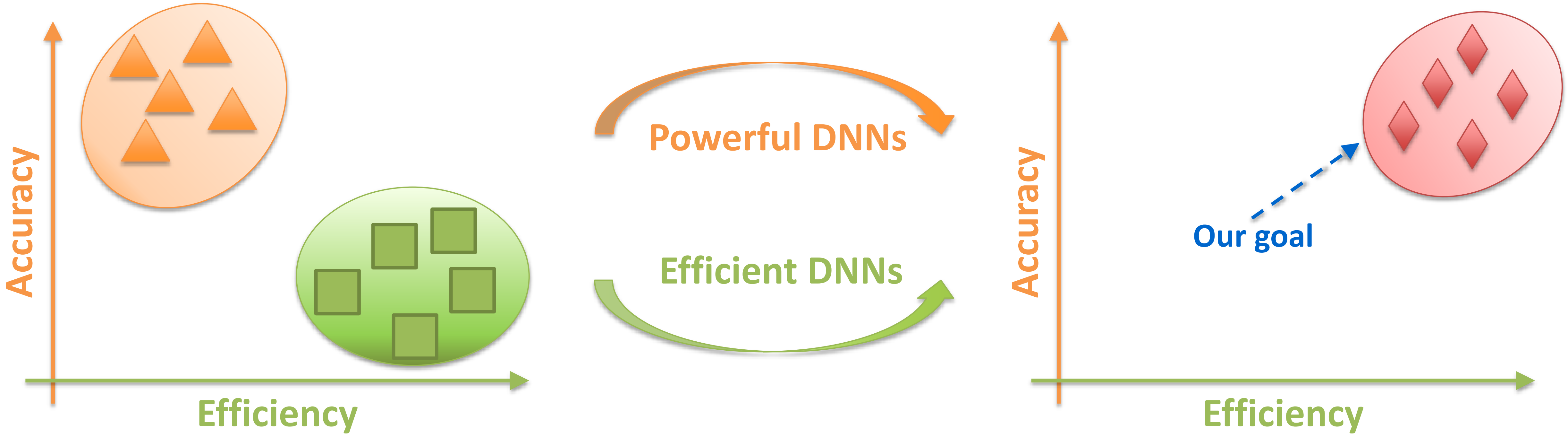
ShiftAddNAS: Background and Motivation

- Two branches of SOTA DNN design: Trade off accuracy and efficiency
  - *Multiplication-based* ***DNNs***, e.g., CNNs, Transformers
    - ☺ Achieve unprecedented task accuracy
    - ☹ Power hungry → Challenge their deployment to edge devices
  - *Multiplication-free* ***DNNs***, e.g., ShiftNet, AdderNet, ShiftAddNet
    - ☺ Efficient and favor their deployment to edge devices
    - ☹ Under-perform their multiplication-based counterparts in terms of task accuracy



ShiftAddNAS: Background and Motivation

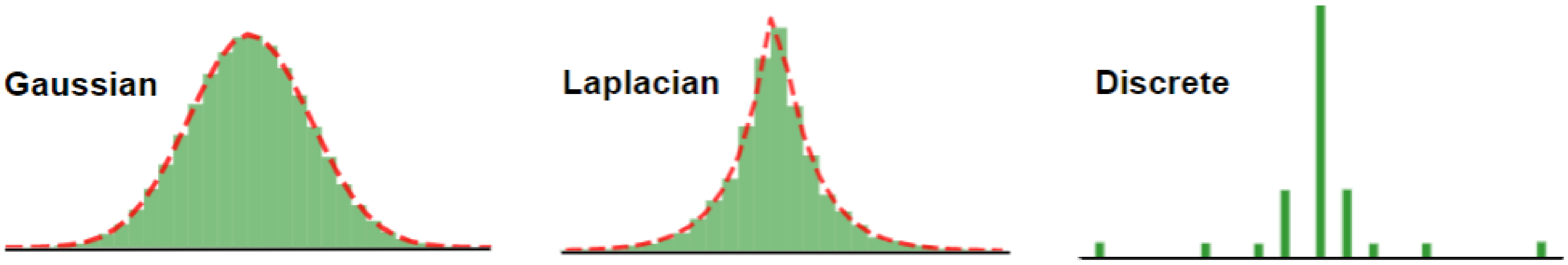
- **Motivation of ShiftAddNAS**
  - Enable automated search for hybrid network architecture to marry the best of both worlds
    - ☺ Multiplication-based operators (e.g., Conv & Attention) → High accuracy
    - ☺ Multiplication-free operators (e.g., Shift & Add) → High efficiency



ShiftAddNAS: Tackled Challenges

- Motivation of ShiftAddNAS
  - Enable automated search for hybrid network architecture to marry the best of both worlds
    - Multiplication-based operators (e.g., Conv & Attention) → High accuracy
    - Multiplication-free operators (e.g., Shift & Add) → High efficiency

- Associated Challenges
  - How to construct an effective hybrid search space?
  - More operators → larger SuperNets, but SOTA weight sharing strategy is not applicable



(a) Weights in Conv (b) Weights in Add (c) Weights in Shift

ShiftAddNAS: Our Contributions

For the first time, we

- Develop ShiftAddNAS, featuring a hybrid search space that incorporates both *multiplication-based* and *multiplication-free* operators

- Propose a new heterogeneous weight sharing strategy that enables automated search for hybrid operators with heterogeneous weight distributions

- Conduct extensive experiments on both CV and NLP tasks to validate the effectiveness of our proposed ShiftAddNAS framework

Contribution 1: Hybrid Search Space and SuperNet

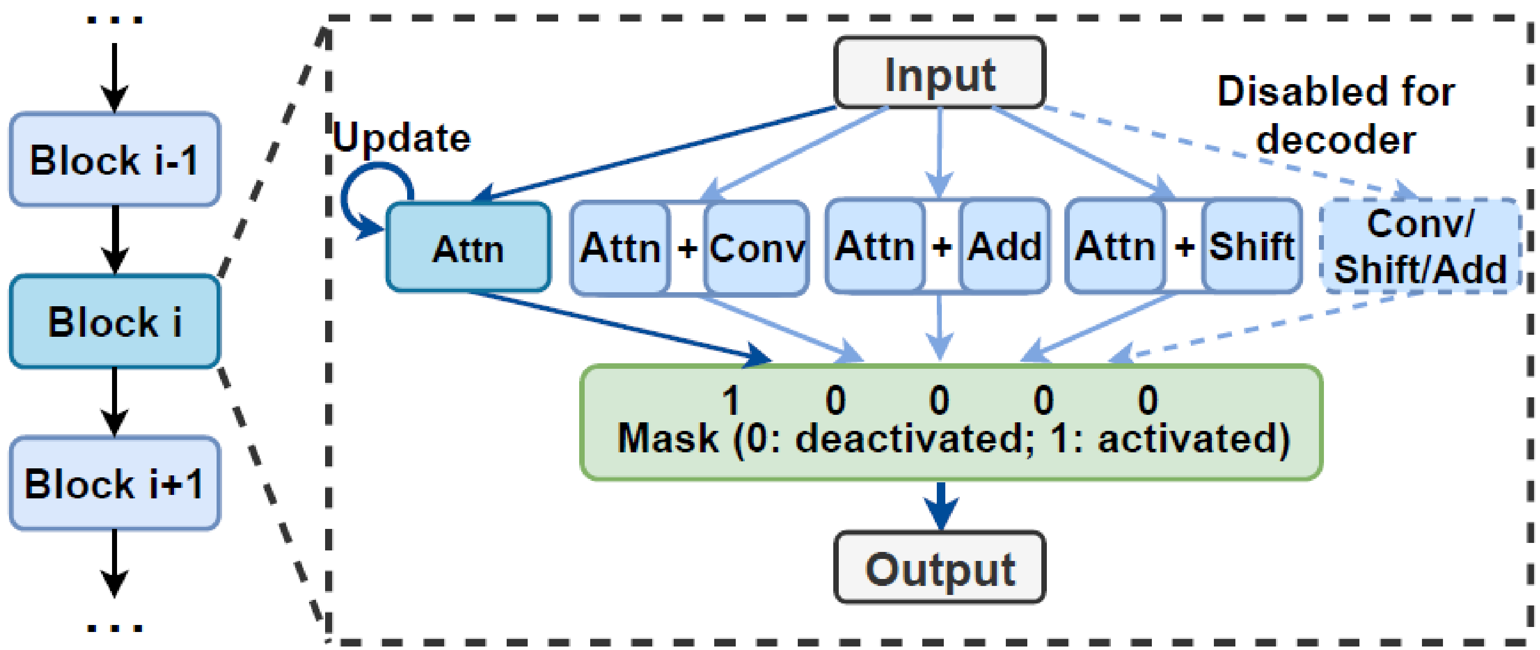
- **Search space for NLP tasks**
  - **Seven different blocks**
    - **Attn, Conv, Shift, and Add**
    - **Attn+Conv, Attn+Add, and Attn+Shift**
  - **Elastic dimensions for MLPs, embeddings, and heads**

Encoder block types	[Attn, Attn+Conv, Attn+Shift] [Attn+Add, Conv, Shift, Add]
Decoder block types	[Attn, Attn+Conv] [Attn+Shift, Attn+Add]
Num. of decoder blocks	[6, 5, 4, 3, 2, 1]
Elastic embed. Dim.	[1024, 768, 512]
Elastic head number	[16, 8, 4]
Elastic MLP dim.	[4096, 3072, 2048, 1024]
Arbitrary Attn	[3, 2, 1]

The Search Space for NLP Tasks

Contribution 1: Hybrid Search Space and SuperNet

- Search space for NLP tasks
  - Seven different blocks
    - Attn, Conv, Shift, and Add
    - Attn+Conv, Attn+Add, and Attn+Shift
  - Elastic dimensions for MLPs, embeddings, and heads



The SuperNet for NLP Tasks

Contribution 1: Hybrid Search Space and SuperNet

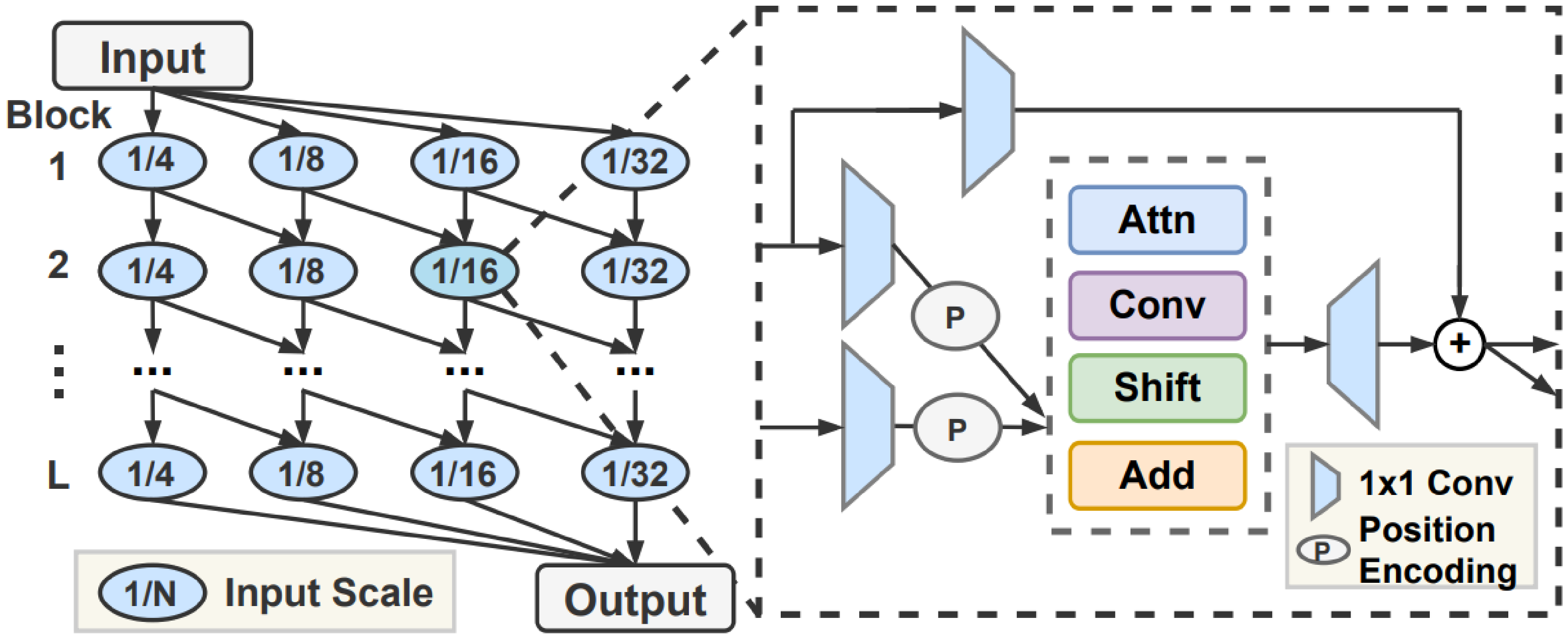
- **Search space for NLP tasks**
- **Search space for CV tasks**
  - **Multi-resolution**
    - **Various spatial resolutions or scales are essential for CV tasks**

Block types	[Attn, Conv, Shift, Add]
Num. of $56^2 \times 128$ blocks	[1, 2, 3, 4]
Num. of $28^2 \times 256$ blocks	[1, 2, 3, 4]
Num. of $14^2 \times 512$ blocks	[3, 4, 5, 6, 7]
Num. of $7^2 \times 1024$ blocks	[4, 5, 6, 7, 8, 9]

The Search Space for CV Tasks

Contribution 1: Hybrid Search Space and SuperNet

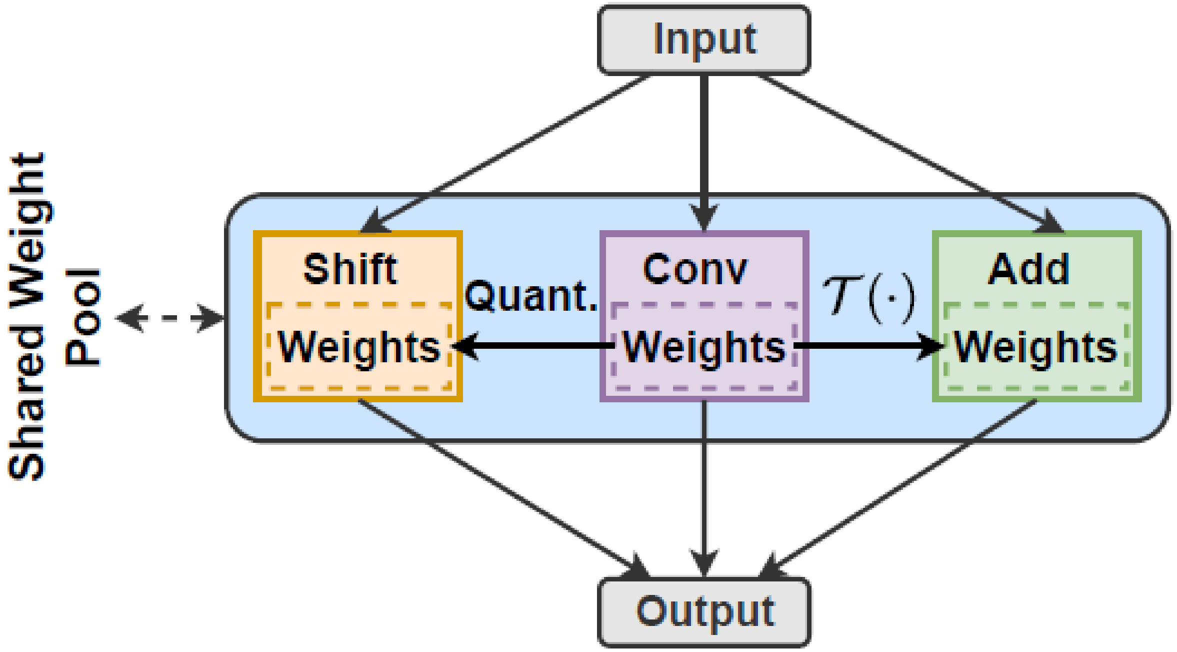
- Search space for NLP tasks
- Search space for CV tasks
  - Multi-resolution
    - Various spatial resolutions or scales are essential for CV tasks



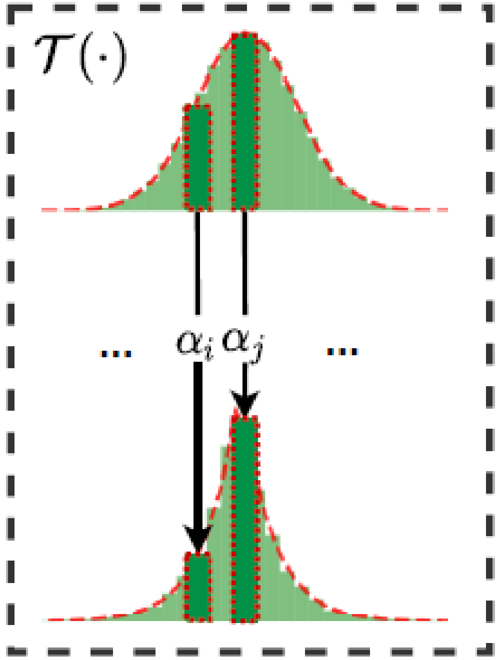
The SuperNet for CV Tasks

Contribution 2: Heterogenous Weight Sharing Strategy

- **One-shot NAS with heterogeneous weight sharing**
  - **Weight sharing among Conv, Add, and Shift blocks**



(a) Heterogenous Weight Sharing Strategy

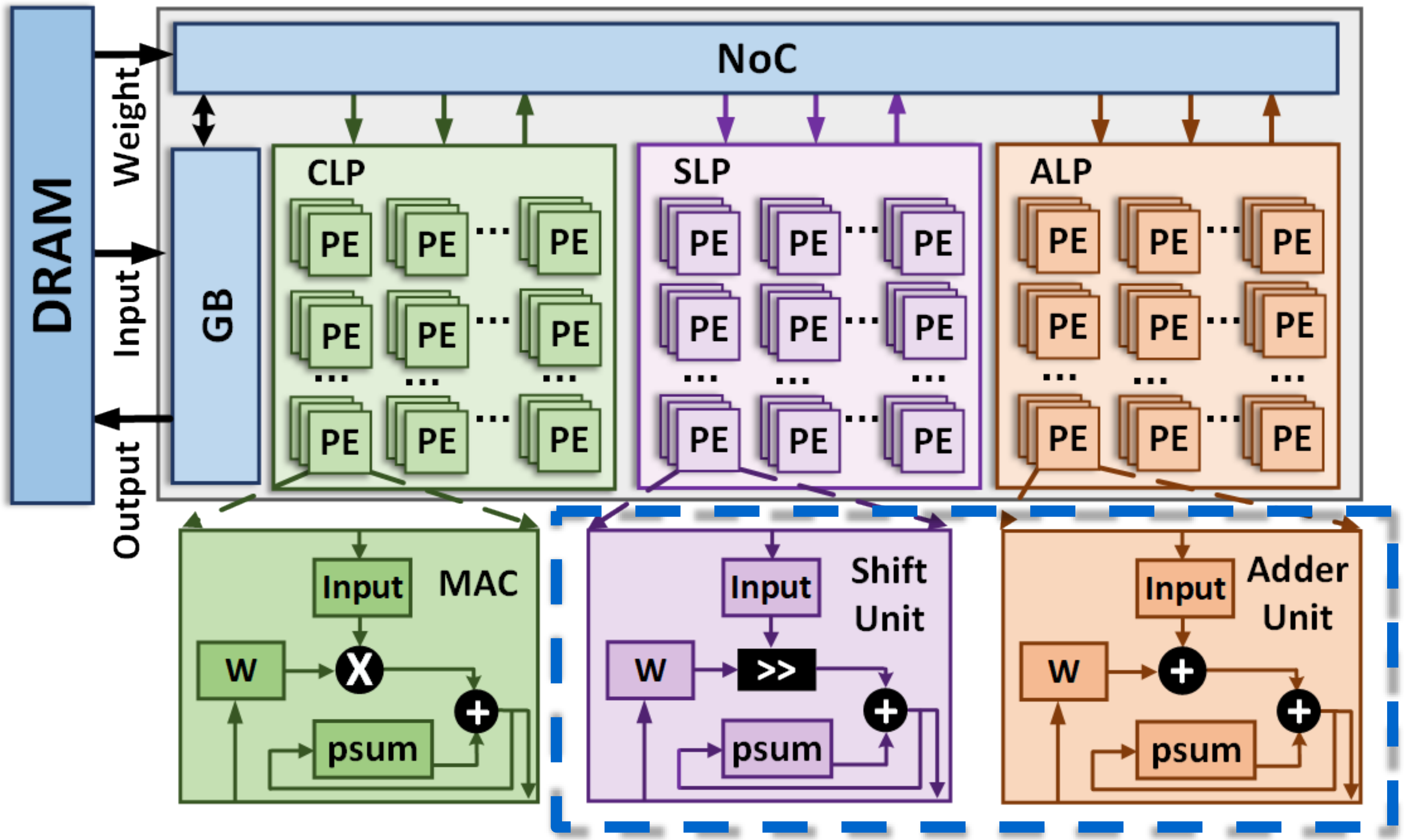


(b) Transformation Kernel

$$\mathcal{L}_{\mathcal{S}} = \mathcal{L}_{CE} + \mathcal{L}_{KL} = -\frac{1}{N}\sum_{i=1}^{N} P(y_i|x_i)\log(P(\hat{y}_i|x_i))$$
$$+ \mathcal{D}_{KL}(P_{\text{Conv}}(\mathbf{W}_{\mathcal{S}}) \mid\mid \mathcal{N}(0, I)) + \mathcal{D}_{KL}(P_{\text{Add}}(\mathcal{T}(\mathbf{W}_{\mathcal{S}})) \mid\mid \mathcal{L}_p(0, \lambda)),$$

NASA: Dedicated Accelerator for Hybrid Networks

- **Micro-architecture**
  - Multi-chunk design with customized PEs → Support heterogeneous layers
  - Four-level memory hierarchy → Enhance data locality



Micro-Architecture

NASA:PE Allocation Strategy

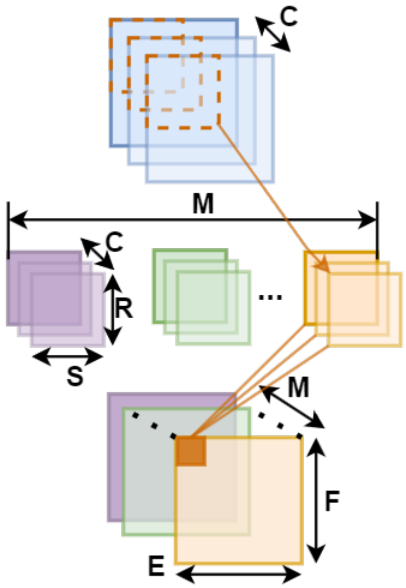
- ***<u>Challenge 1</u>***
  - **How to partition and then allocate limited hardware resources to multiple chunks?**
- **Proposed PE allocation strategy**
  - **Balance the throughput of multiple chunks → Minimize the overall latency**
  - **Formally, allocated PEs in chunks are proportional to the corresponding operations under the area budget**

$$\frac{N_C}{O_{Conv}} = \frac{N_S}{O_{Shift}} = \frac{N_A}{O_{Adder}},$$

$$s.t. A_C + A_S + A_A = Area\ Constraint.$$

NASA: Auto-Mapper

- ***Challenge 2***
  - Our bigger design space → Nontrivial to manually identify the optimal dataflow
- **Proposed Auto-Mapper**
  - Enable automated search for the optimal dataflow
  - Nested for-loop description:
    - Loop ordering factors: Determine the data reuse patterns
    - Loop tiling factors: Determine how to store data within each memory hierarchy



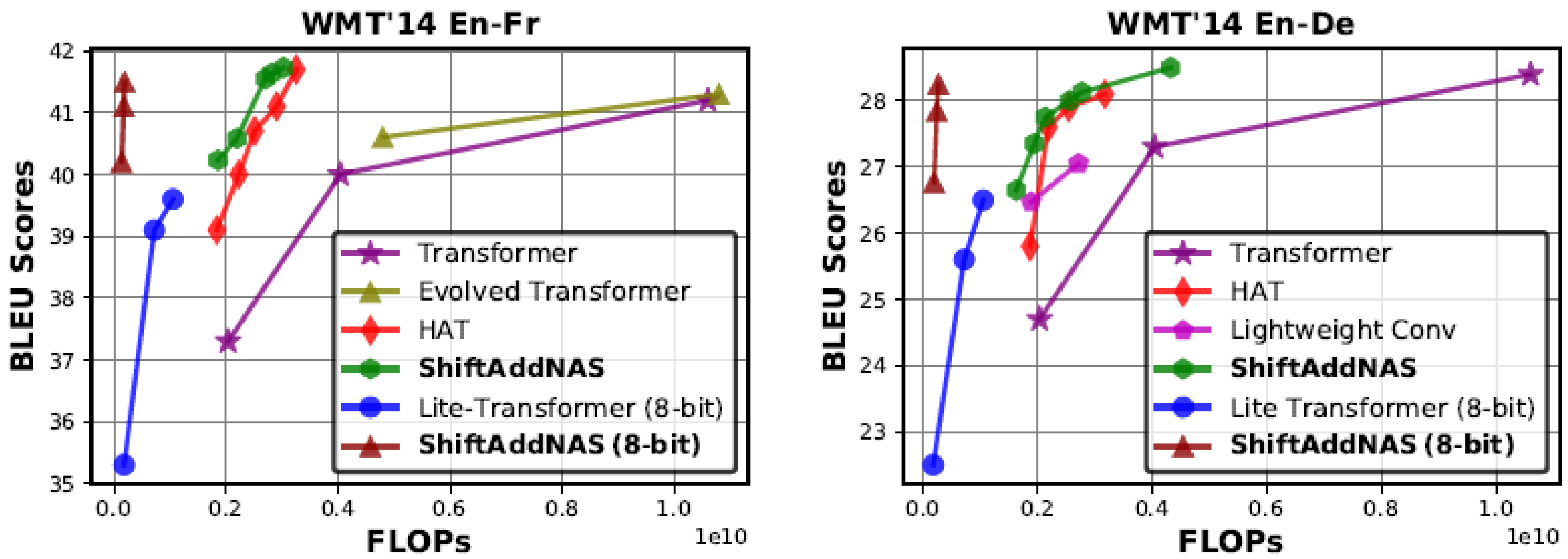

```
for (m3=0; m3<M3; m3++)                      DRAM
  for (e3=0; e3<E3; e3++)
    for (c3=0; c3<C3; c3++)
      for (m2=0; m2<M2; m2++)                GB
        for (c2=0; c2<C2; c2++)
NoC       pfor (m1=0; m1<M1; m1++)
          pfor (e1=0; e1<E1; e1++)
          pfor (c1=0; c1<C1; c1++)
          pfor (r=0; r<R; r++)
RF          for (m0=0; m0<M0; m0++)
              for (f=0; f<F; f++)
                for (c0=0; c0<C0; c0++)
                  for (s=0; s<S; s++)
                    basic computations
```

ShiftAddNAS: Experimental Setting

- NLP tasks
  - Two datasets
    - WMT’14 English to French (En-Fr)
    - WMT’14 English to German (En-De)
  - Five evaluation metrics
    - BLEU score
    - Number of parameters/FLOPs
    - Hardware energy and latency
  - Four baselines
    - Transformer
    - Lightweight Conv
    - Lite Transformer
    - HAT

- CV tasks
  - One dataset: ImageNet
  - Five evaluation metrics
    - Accuracy
    - Number of parameters/MACs
    - Hardware energy and latency
  - Four categories of baselines
    - Multiplication-free NNs
      - AdderNet, DeepShift, BNN
    - CNNs
      - ResNet, SENet
    - Transformer
      - ViT, DeiT, VITAS, Autoformer
    - CNN-Transformer
      - BoT, HR-NAS, BossNAS

ShiftAddNAS: Experimental Results for NLP Tasks



BLEU scores vs. FLOPs of ShiftAddNAS over SOTA baselines on NLP tasks.

ShiftAddNAS vs. SOTA baselines in terms of accuracy and efficiency on NLP tasks.

	WMT'14 En-Fr					WMT'14 En-De				
	Params	FLOPs	BLEU	Latency	Energy	Params	FLOPs	BLEU	Latency	Energy
Transformer	176M	10.6G	41.2	130ms	214mJ	176M	10.6G	28.4	130ms	214mJ
Evolved Trans.	175M	10.8G	41.3	-	-	47M	2.9G	28.2	-	-
HAT	48M	3.4G	41.4	49ms	81mJ	44M	2.7G	28.2	42ms	69mJ
ShiftAddNAS	**46M**	**3.0G**	**41.8**	**43ms**	**71mJ**	**43M**	**2.7G**	**28.2**	**40ms**	**66mJ**
HAT	46M	2.9G	41.1	42ms	69mJ	36M	2.2G	27.6	34ms	56mJ
ShiftAddNAS	**41M**	**2.7G**	**41.6**	**39ms**	**64mJ**	**33M**	**2.1G**	**27.8**	**31ms**	**52mJ**
HAT	30M	1.8G	39.1	29ms	48mJ	25M	**1.5G**	25.8	24ms	40mJ
ShiftAddNAS	**29M**	**1.8G**	**40.2**	**16ms**	**45mJ**	**25M**	1.6G	**26.7**	**24ms**	**40mJ**
Lite Trans. (8-bit)	17M	1G	39.6	19ms	31mJ	17M	1G	26.5	19ms	31mJ
ShiftAddNAS (8-bit)	**11M**	**0.2G**	**41.5**	**11ms**	**16mJ**	**17M**	**0.3G**	**28.3**	**16ms**	**24mJ**
Lite Trans. (8-bit)	12M	0.7G	39.1	14ms	24mJ	12M	0.7G	25.6	14ms	24mJ
ShiftAddNAS (8-bit)	**10M**	**0.2G**	**41.1**	**10ms**	**15mJ**	**12M**	**0.2G**	**26.8**	**9.2ms**	**14mJ**

- **Overall Improvement on NLP**
  - **ShiftAddNAS achieves up to +2 BLEU scores improvement and 69.1% and 69.2% energy and latency savings**

ShiftAddNAS: Experimental Results for CV Tasks

Comparison with SOTA baselines on ImageNet classification task.

Model	Top-1 Acc.	Top-5 Acc.	Params	Res.	MACs	#Mult.	#Add	#Shift	Model Type
BNN	55.8%	78.4%	26M	224^2	3.9G	0.1G	3.9G	3.8G	Mult.-free
AdderNet	74.9%	91.7%	26M	224^2	3.9G	0.1G	7.6G	0	Mult.-free
AdderNet-PKKD	76.8%	93.3%	26M	224^2	3.9G	0.1G	7.6G	0	Mult.-free
DeepShift-Q	70.7%	90.2%	26M	224^2	3.9G	0.1G	3.9G	3.8G	Mult.-free
DeepShift-PS	71.9%	90.2%	52M	224^2	3.9G	0.1G	3.9G	3.8G	Mult.-free
ResNet-50	76.1%	92.9%	26M	224^2	3.9G	3.9G	3.9G	0	CNN
ResNet-101	77.4%	94.2%	45M	224^2	7.6G	7.6G	7.6G	0	CNN
SENet-50	79.4%	94.6%	26M	224^2	3.9G	3.9G	3.9G	0	CNN
SENet-101	81.4%	95.7%	45M	224^2	7.6G	7.6G	7.6G	0	CNN
ViT-B/16	77.9%	-	86M	384^2	18G	18G	17G	0	Transformer
ViT-L/16	76.5%	-	304M	384^2	64G	64G	63G	0	Transformer
DeiT-T	74.5%	-	6M	224^2	1.3G	1.3G	1.3G	0	Transformer
DeiT-S	81.2%	-	22M	224^2	4.6G	4.6G	4.6G	0	Transformer
VITAS	77.4%	93.8%	13M	224^2	2.7G	2.7G	2.7G	0	Transformer
Autoformer-S	81.7%	95.7%	23M	224^2	5.1G	5.1G	5.1G	0	Transformer
BoT-50	78.3%	94.2%	21M	224^2	4.0G	4.0G	4.0G	0	CNN + Trans.
BoT-50 + SE	79.6%	94.6%	21M	224^2	4.0G	4.0G	4.0G	0	CNN + Trans.
HR-NAS	77.3%	-	6.4M	224^2	0.4G	0.4G	0.4G	0	CNN + Trans.
BossNAS-T0	80.5%	95.0%	38M	224^2	3.5G	3.5G	3.5G	0	CNN + Trans.
BossNAS-T0 + SE	80.8%	95.2%	38M	224^2	3.5G	3.5G	3.5G	0	CNN + Trans.
ShiftAddNAS-T0	**82.1%**	**95.8%**	**30M**	224^2	**3.7G**	**2.7G**	**3.8G**	**1.0G**	Hybrid
ShiftAddNAS-T0↑	**82.6%**	**96.2%**	**30M**	256^2	**4.9G**	**3.6G**	**4.9G**	**1.4G**	Hybrid
T2T-ViT-19	81.9%	-	39M	224^2	8.9G	8.9G	8.9G	0	Transformer
TNT-S	81.3%	95.6%	24M	224^2	5.2G	5.2G	5.2G	0	Transformer
Autoformer-B	82.4%	95.7%	54M	224^2	11G	11G	11G	0	Transformer
BoTNet-S1-59	81.7%	95.8%	28M	224^2	7.3G	7.3G	7.3G	0	CNN + Trans.
BossNAS-T1	82.2%	95.8%	38M	224^2	8.0G	8.0G	8.0G	0	CNN + Trans.
ShiftAddNAS-T1	**82.7%**	**96.1%**	**30M**	224^2	**6.4G**	**5.4G**	**6.4G**	**1.0G**	Hybrid
ShiftAddNAS-T1↑	**83.0%**	**96.4%**	**30M**	256^2	**8.5G**	**7.1G**	**8.5G**	**1.4G**	Hybrid

- **Overall Improvement on CV**
  - **ShiftAddNAS on average offers a +0.8% ~ +7.7% higher accuracy and 24% ~ 93% energy savings**

Summary

For the first time, we

- **Develop ShiftAddNAS, featuring a hybrid search space that incorporates both *multiplication-based* and *multiplication-free* operators**
- **Propose a new heterogeneous weight sharing strategy that enables automated search for hybrid operators with heterogeneous weight distributions**
- **Conduct extensive experiments on both CV and NLP tasks to validate the effectiveness of our proposed ShiftAddNAS framework**

Open-source Code:
https://github.com/RICE-EIC/ShiftAddNAS

NSF
NIH National Institute of Mental Health

Q & A

Thank you for your listening!