Google Neural Network Models for Edge Devices:
Analyzing and Mitigating Machine Learning Inference Bottlenecks

Amirali Boroumand[1,2], Saugata Ghose[3], Berkin Akin[4], Ravi Narayanaswami[4],
Geraldo F. Oliveira[5], Xiaoyu Ma[4], Eric Shiu[4], Onur Mutlu[5,1]

[1] Carnegie Mellon Univ., [2] Stanford Univ. ,
[3] Univ. of Illinois Urbana-Champaign, [4] Google, [5] ETH Zürich

PACT 2021

Presented by Lotte Seifert

Executive Summary

Context: *Edge ML accelerators* have to execute *inference efficiently* across a *wide variety of NN models*

- Extensive analysis of state-of-the-art edge ML accelerator (Google Edge TPU) using 24 diverse Google edge models

Problem: ML inference computations on the *Google Edge TPU* suffer from *three shortcomings*:

- The TPU operates significantly below its peak throughput
- The TPU operates significantly below its theoretical energy efficiency
- The TPU inefficiently accesses memory

Key Insight: *Customizing* all accelerator *key components* to layer *heterogeneity* is crucial for good performance

- The layer characteristics significantly vary across and within the state-of-the-art Google edge models
- The monolithic design of the Edge TPU is the root cause of its shortcomings and the resulting large inefficiency

Key Mechanism: Mensa - a new acceleration framework for edge NN inference

- Mensa consists of heterogeneous accelerators whose dataflow and hardware are specialized for specific families of layers

Key Results: We design a version of *Mensa* for *Google edge ML models*

- Mensa improves performance and energy by 3.0x and 3.1x
- Mensa reduces cost and improves area efficiency

Outline

Context
- Edge Computing
- Neural Network Models
- Machine Learning Accelerators

Problem
- Edge TPU Shortcomings

Key Insight
- NN Model Characterization
- Sources of Edge TPU Shortcomings

Key Mechanism
- Mensa Framework
- Mensa Runtime Scheduler

Key Results
- Identifying Layer Families
- Mensa-G: Mensa for Google Edge Models
- Evaluation

Outline of Edge Computing

Why deploy ML on Edge Devices?

Consumer Demand

- Privacy
- Connectivity
- Low Latency
- Bandwidth

Technology

- Internet of Things
- 5G

Edge Computing

"Computation, storage, and analysis of data close to its creation and consumption"

- Constrained power budget
- Limited computational resources

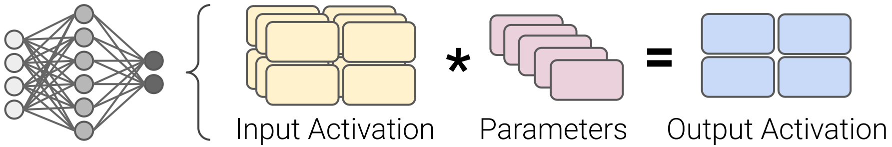
Accelerator Deployment

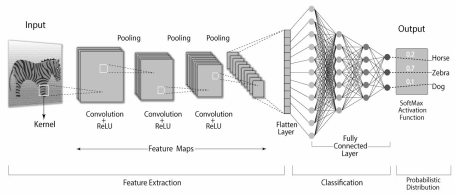
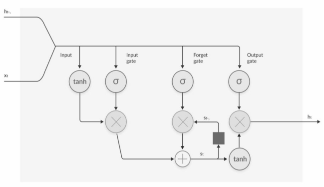
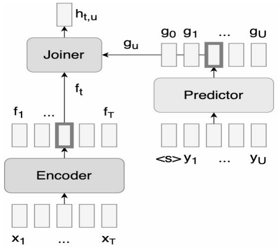
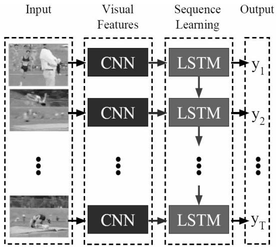
- Google Edge TPU
- NVIDIA Jetson
- Intel Movidius
- Apple Neural Engine

Algorithm Development

- Neural Network (NN) Models

NN Models

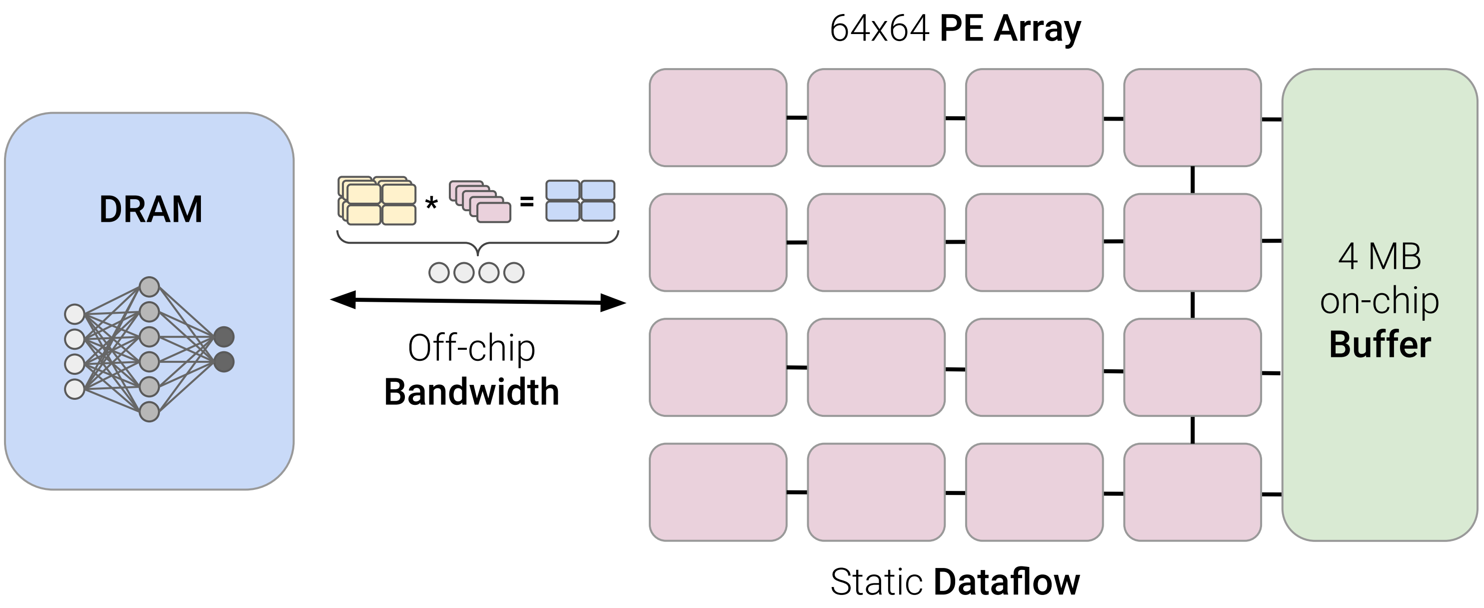


Convolutional Neural Networks (CNN)	Long Short-Term Memory Networks (LSTM)	Transducers	Recurrent Convolutional Neural Networks (RCNN)
• Feed-forward multi-layer model • Captures and classifies spatial features ○ Image classification ○ Object detection	• Multi-layer models with recurrent connections • Classfies and predicts future data sequences ○ Traffic forecasting ○ Text reply prediction	• Typically implemented by stacking LSTM layers • Classfies sequences with distortions in input data ○ Automatic speech recognition	• Hybrid multi-layer recurrent NNs • Captures spatio-temporal information ○ Image captioning ○ Video scene labeling
			
13 CNNs	2 LSTMs	6 RNN Transducers	3 RCNNs

= 24 Google Edge Models

1. https://developersbreach.com/convolution-neural-network-deep-learning/, 2. https://indiantechwarrior.com/all-about-deep-learning-long-short-term-memory-lstm-networks/
3. https://lorenlugosch.github.io/posts/2020/11/transducer/, 4. https://www.researchgate.net/figure/We-propose-Long-term-Recurrent-Convolutional-Networks-LRCNs-a-class-of-architectures_fig1_308034527

Edge TPU: Baseline Accelerator



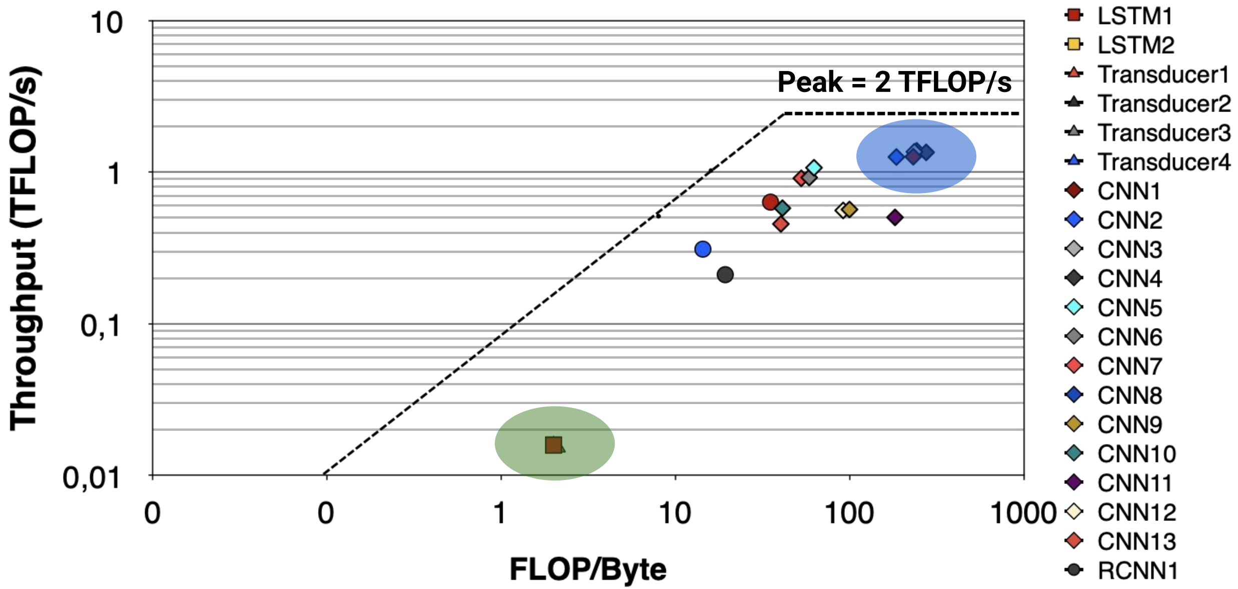
Take Away

Context: *Edge ML accelerators* have to execute *inference efficiently* across a *wide variety of NN models*

- Extensive analysis of state-of-the-art edge ML accelerator (Google Edge TPU) using 24 diverse Google edge models

Edge TPU Shortcomings

1. High Resource Underutilization



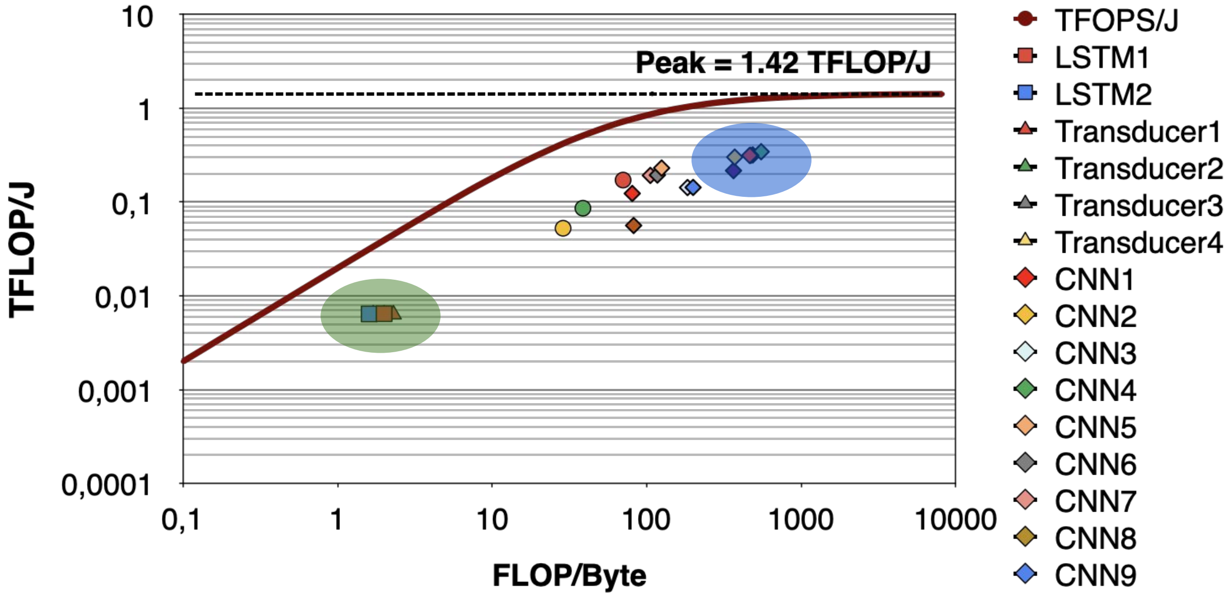
< 1% of peak throughput for **Transducers** and **LSTM** on average

40.7% of peak throughput for **CNN** and **RCNN** on average

The Edge TPU utilizes **only 24%** of its **peak throughput**, averaged across all models.

Edge TPU Shortcomings

2. Low Energy Efficiency



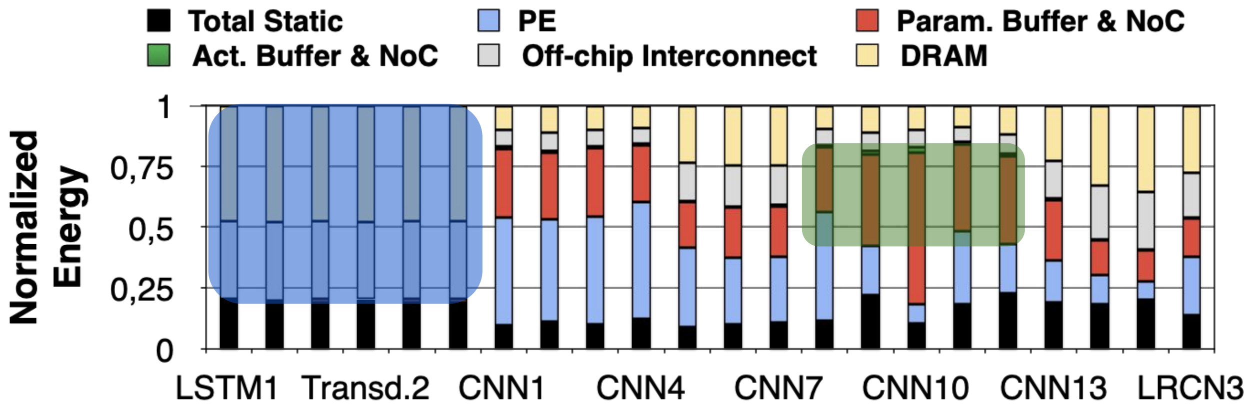
Upper bound at **33.8%** of peak efficiency for **Transducers** and **LSTM**

Upper bound at **50.7%** of peak efficiency for **CNN** and **RCNN**

The Edge TPU provides **only 37%** of its **peak energy efficiency**, averaged across all models.

Edge TPU Shortcomings

3. Inefficient Memory Access Handling



High energy cost of **large on-chip buffers**

High energy cost of **off-chip memory accesses**

The Edge TPU's **memory system** is often a **large bottleneck**.

Take Away

Context: *Edge ML accelerators* have to execute *inference efficiently* across a *wide variety of NN models*

- Extensive analysis of state-of-the-art edge ML accelerator (Google Edge TPU) using 24 diverse Google edge models

Problem: ML inference computations on the *Google Edge TPU* suffer from *three shortcomings*:

- The TPU operates significantly below its peak throughput
- The TPU operates significantly below its theoretical energy efficiency
- The TPU inefficiently accesses memory

NN Model Characterization

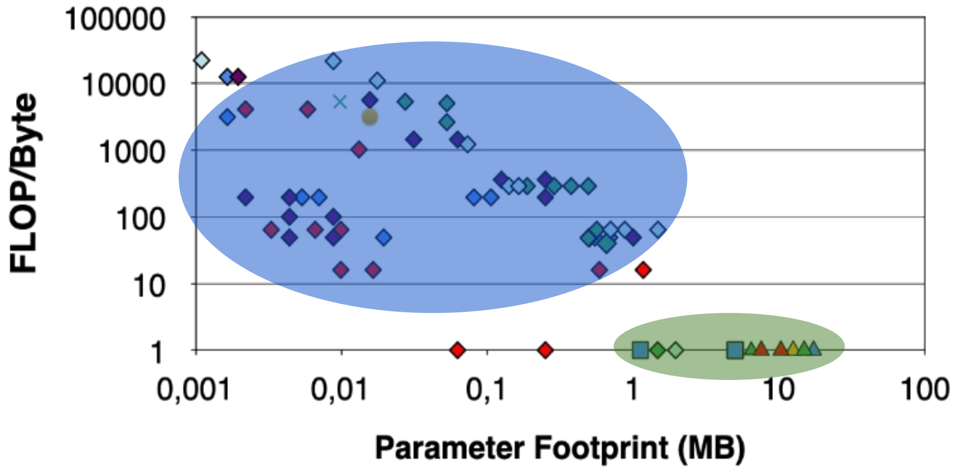
1. Layer Heterogeneity across Models

Memory Footprints

- Layer Composition

FLOP/B ratio

- Reuse Patterns
- Computational Complexity
- Intra-and Inter-cell Dependencies



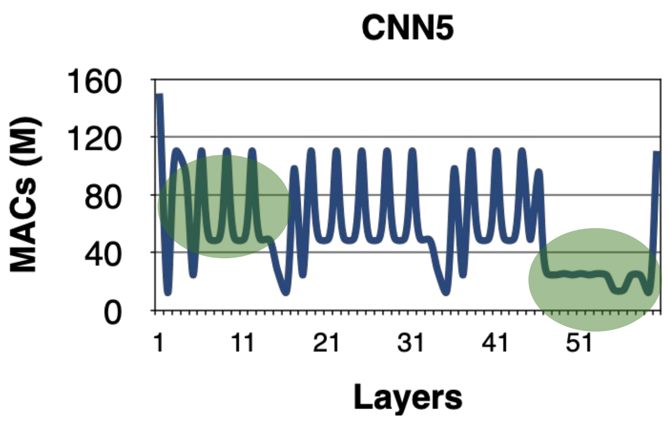
Layers within **CNN** and **RCNN**

Layers within **LSTM** and **Transducer**

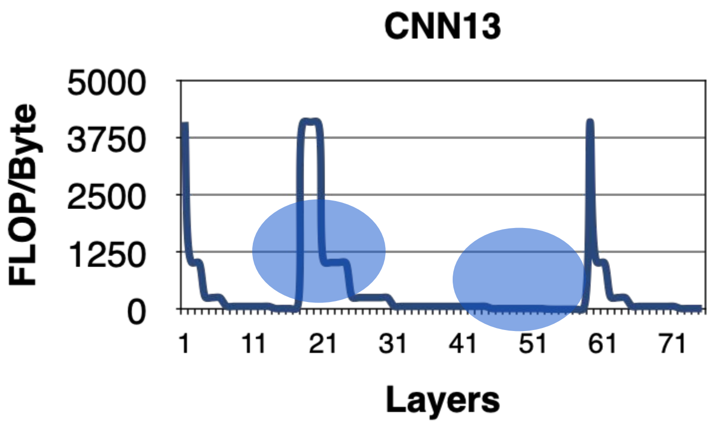
Significant variations exist with regards to **layer characteristics across** the different models.

NN Model Characterization

2. Layer Heterogeneity within Models



Variation in **MAC intensity**:
up to **200x** across layers



Variation in **FLOP/Byte**:
up to **244x** across layers

Significant variations exist with regards to **layer characteristics within** each model.

Sources of Edge TPU Shortcomings

PE Underutilization

- *Memory bandwidth bottleneck* slows performance
- *Static dataflow* fails to exploit diverse data reuse patterns
- *Fixed size PE* unfit for efficient execution of layers with diverse shapes and dependencies

Poor Energy Efficiency

- *Large on-chip buffer* results in high energy costs
- *Underutilized PEs* result in high energy costs
- *Frequent off-chip traffic* results in high energy costs

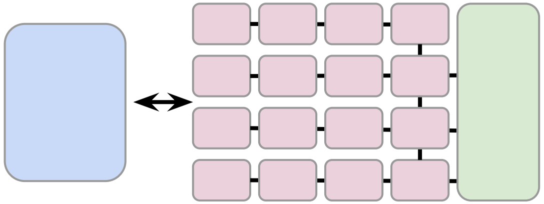
Memory System Issues

- *Unnecessary buffer* for layers with little or no data reuse
- *Over-sized buffer* compared to average parameter footprint of layers with large data reuse

1. Key Insight:
Accelerator's key components fail to account for layer heterogeneity

2. Key Insight:
Monolithic approach performs inefficiently over range of models

Monolithic designed Accelerators



- Over-provisioned PE array
- Over-provisioned on-chip buffer
- Rigid dataflow
- Fixed off-chip bandwidth

The Edge TPU's **monolithic design** is the **root cause** of its **shortcomings**.

Take Away

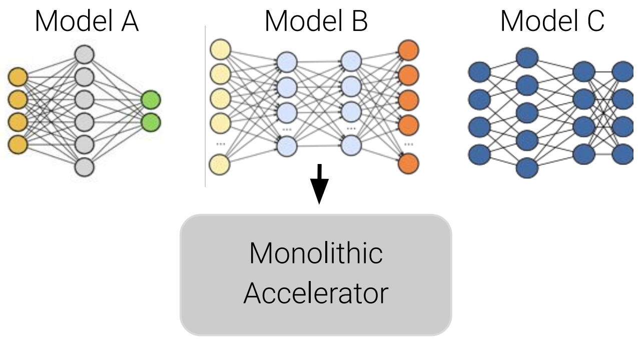
Context: *Edge ML accelerators* have to execute *inference efficiently* across a *wide variety of NN models*
- Extensive analysis of state-of-the-art edge ML accelerator (Google Edge TPU) using 24 diverse Google edge models

Problem: ML inference computations on the *Google Edge TPU* suffer from *three shortcomings*:
- The TPU operates significantly below its peak throughput
- The TPU operates significantly below its theoretical energy efficiency
- The TPU inefficiently accesses memory

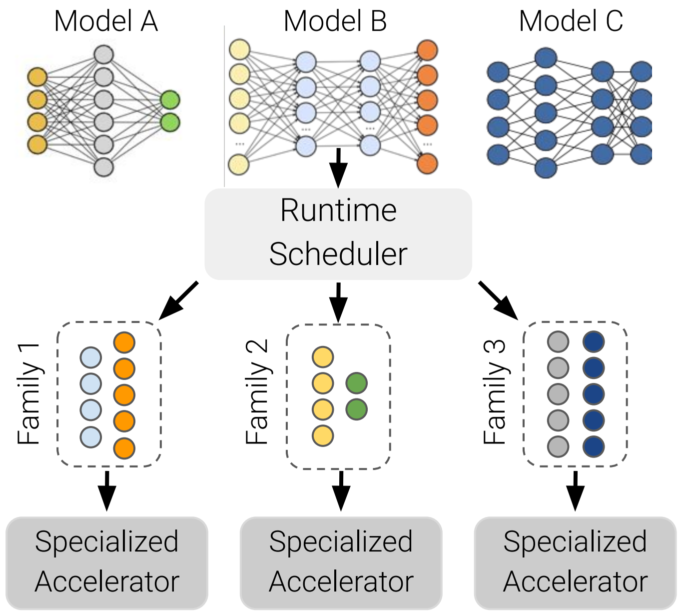
Key Insight: *Customizing* all accelerator *key components* to layer *heterogeneity* is crucial for good performance
- The layer characteristics significantly vary across and within the state-of-the-art Google edge models
- The monolithic design of the Edge TPU is the root cause of its shortcomings and the resulting large inefficiency

Mensa Framework

Current Mechanism: Run entire NN model on monolithic Edge TPU accelerator



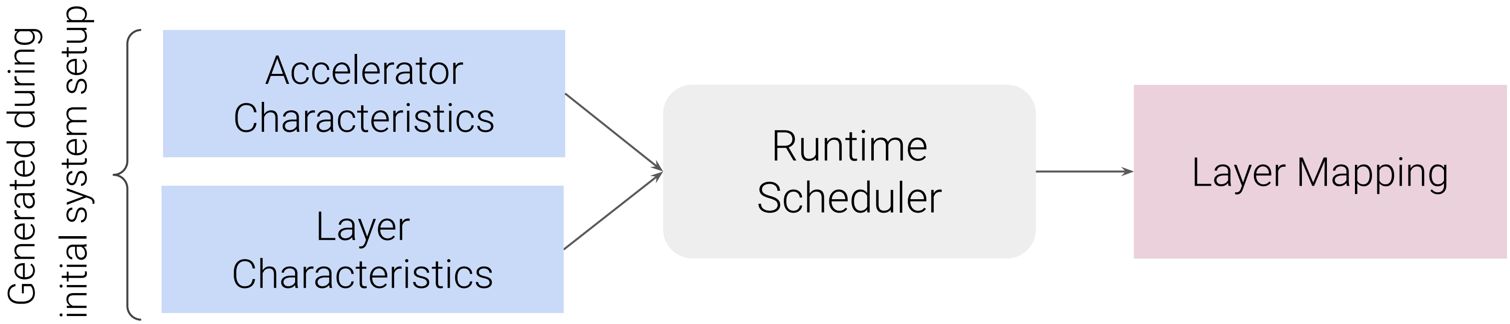
New Mechanism: Distribute NN model layers across multiple specialized smaller accelerators



Heterogeneous accelerators with specific dataflow and hardware optimized for subset of layer characteristics

Mensa **exploits** the **variations** between and within layers for **high efficiency** and **high performance**.

Mensa Runtime Scheduler



Mensa's **software runtime scheduler** determines on which **accelerator each layer** should run.

Take Away

Context: *Edge ML accelerators* have to execute *inference efficiently* across a *wide variety of NN models*

- Extensive analysis of state-of-the-art edge ML accelerator (Google Edge TPU) using 24 diverse Google edge models

Problem: ML inference computations on the *Google Edge TPU* suffer from *three shortcomings*:

- The TPU operates significantly below its peak throughput
- The TPU operates significantly below its theoretical energy efficiency
- The TPU inefficiently accesses memory

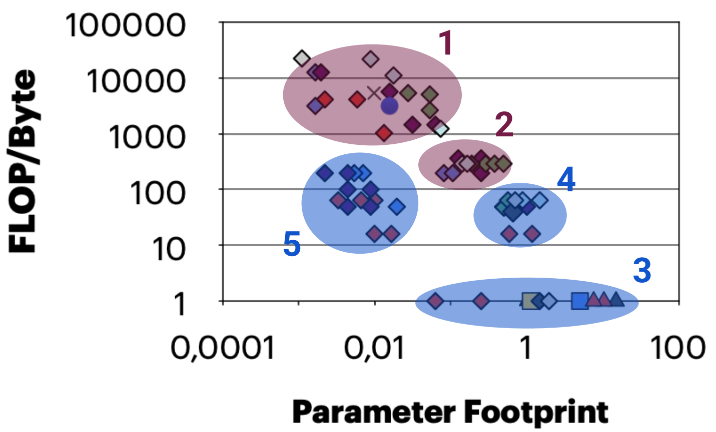
Key Insight: *Customizing* all accelerator *key components* to layer *heterogeneity* is crucial for good performance

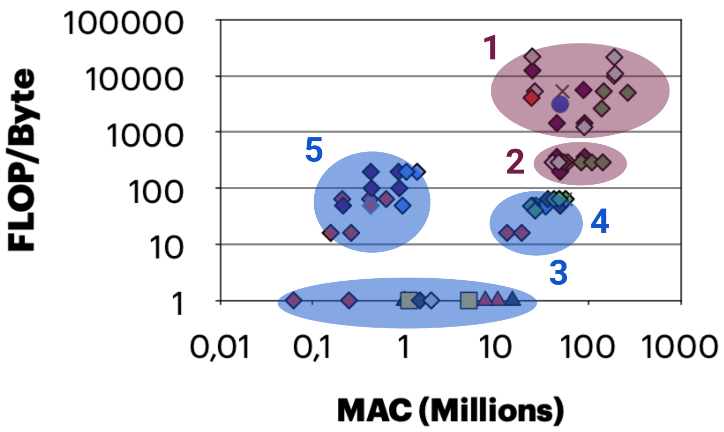
- The layer characteristics significantly vary across and within the state-of-the-art Google edge models
- The monolithic design of the Edge TPU is the root cause of its shortcomings and the resulting large inefficiency

Key Mechanism: Mensa - a new acceleration framework for edge NN inference

- Mensa consists of heterogeneous accelerators whose dataflow and hardware are specialized for specific families of layers

Identifying Layer Families





Compute-centric layers: Families 1 & 2

- Small parameter footprint
- High data reuse
- High MAC intensity

⇨ High PE utilization

Data-centric layers: Families 3, 4 & 5

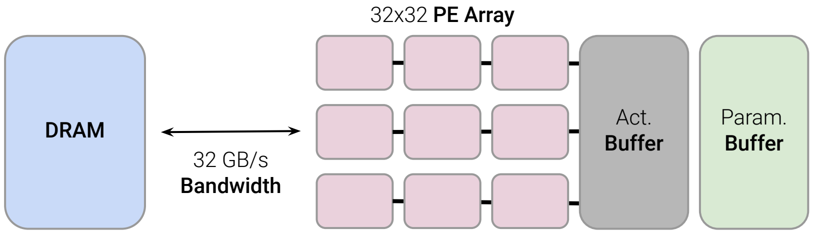
- Large parameter footprint
- Low data reuse
- Low MAC intensity

⇨ Low PE utilization

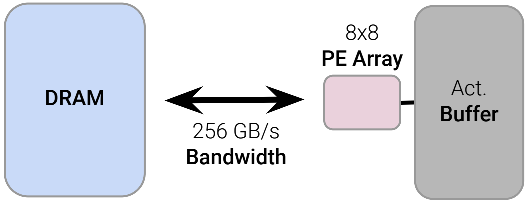
The majority of **layers group** into a **small number** of **layer families** with specific characteristics.

Mensa-G

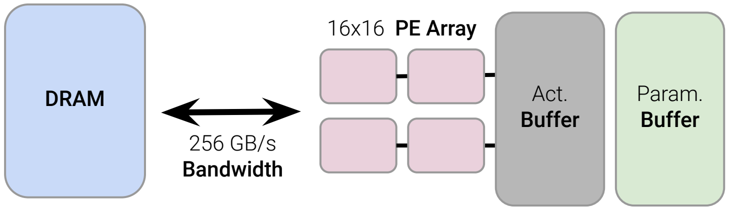
Mensa for Google Edge Models



Pascal: Families 1 & 2: compute-centric layers
- 32x32 PE Array (2 TFLOP/s)
- 256 KB Act. Buffer (8x Reduction)
- 128 KB Param. Buffer (32x Reduction)
- On-chip accelerator

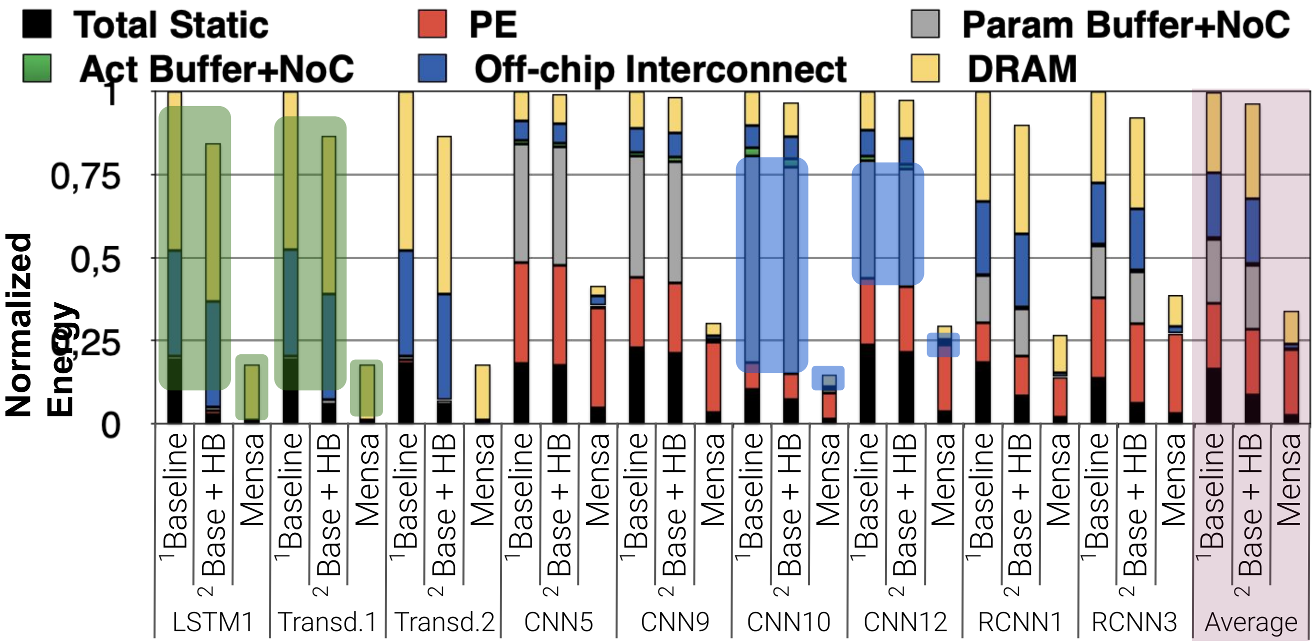


Pavlov: Family 3: LSTM data-centric layers
- 8x8 PE Array (128 GFLOP/s)
- 128 KB Act. Buffer (16x Reduction)
- No Param. Buffer (4MB in Baseline)
- Near-data accelerator



Jacquard: Families 4 & 5: non-LSTM data-centric layers
- 16x16 PE Array (256 GFLOP/s)
- 128 KB Act. Buffer (16x Reduction)
- 128 KB Param. Buffer (32x Reduction)
- Near-data accelerator

Energy Analysis



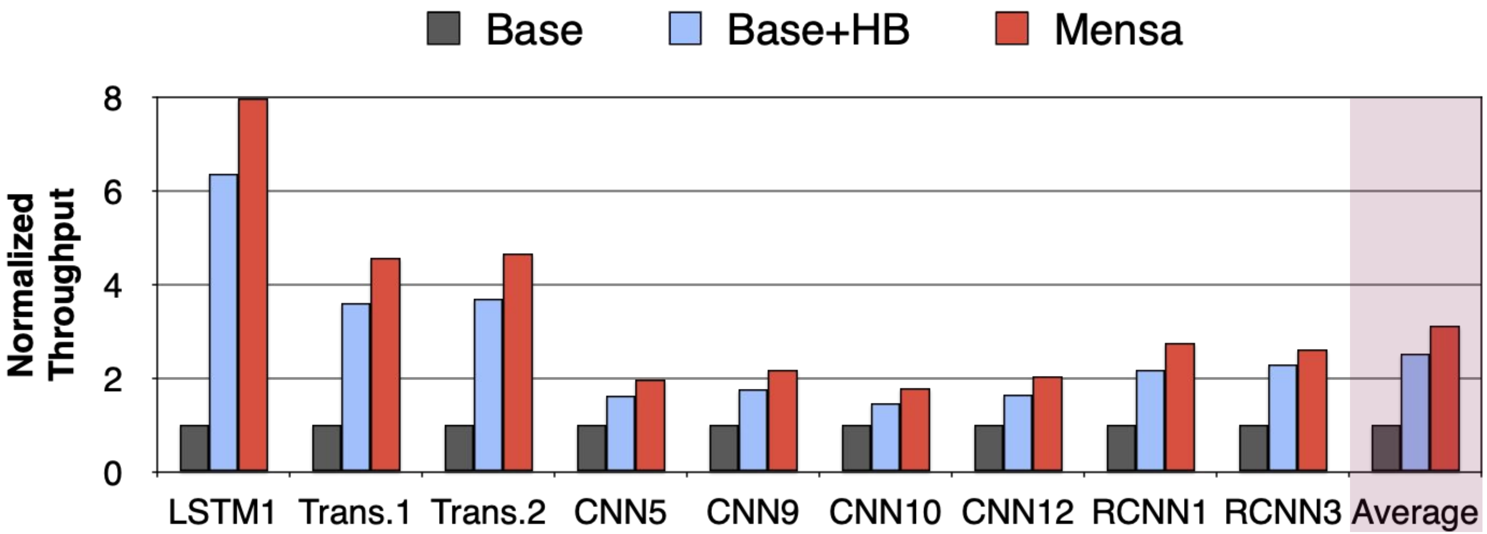
[1] Baseline Google Edge TPU accelerator
[2] Baseline Google Edge TPU accelerator with high-bandwidth off-chip memory

15.3x lower **on-chip/off-chip parameter traffic energy** by scheduling on accelerator with appropriate **dataflow** and **memory bandwidth**

49.8x lower **on-chip buffer dynamic energy** by avoiding **overprovisioning** and catering to **specialized dataflows**

Mensa-G **improves energy efficiency** by **3.0x** compared to the Baseline.

Throughput Analysis



Mensa-G **improves throughout** by **3.1x** compared to the Baseline.

Take Away

Context: *Edge ML accelerators* have to execute *inference efficiently* across a *wide variety of NN models*

- Extensive analysis of state-of-the-art edge ML accelerator (Google Edge TPU) using 24 diverse Google edge models

Problem: ML inference computations on the *Google Edge TPU* suffer from *three shortcomings*:

- The TPU operates significantly below its peak throughput
- The TPU operates significantly below its theoretical energy efficiency
- The TPU inefficiently accesses memory

Key Insight: *Customizing* all accelerator *key components* to layer *heterogeneity* is crucial for good performance

- The layer characteristics significantly vary across and within the state-of-the-art Google edge models
- The monolithic design of the Edge TPU is the root cause of its shortcomings and the resulting large inefficiency

Key Mechanism: Mensa - a new acceleration framework for edge NN inference

- Mensa consists of heterogeneous accelerators whose dataflow and hardware are specialized for specific families of layers

Key Results: We design a version of *Mensa* for *Google edge ML models*

- Mensa improves performance and energy by 3.0x and 3.1x
- Mensa reduces cost and improves area efficiency

More in the Paper

- Details about Mensa Runtime Scheduler
- Hardware Design Principles and Decisions
- Details about Pascal, Pavlov, and Jacquard's dataflows
- Energy comparison with Eyeriss v2
- Mensa-G's utilization results
- Mensa-G's inference latency results

Conclusion

Context: *Edge ML accelerators* have to execute *inference efficiently* across a *wide variety of NN models*
- Extensive analysis of state-of-the-art edge ML accelerator (Google Edge TPU) using 24 diverse Google edge models

Problem: ML inference computations on the *Google Edge TPU* suffer from *three shortcomings*:
- The TPU operates significantly below its peak throughput
- The TPU operates significantly below its theoretical energy efficiency
- The TPU inefficiently accesses memory

Key Insight: *Customizing* all accelerator *key components* to layer *heterogeneity* is crucial for good performance
- The layer characteristics significantly vary across and within the state-of-the-art Google edge models
- The monolithic design of the Edge TPU is the root cause of its shortcomings and the resulting large inefficiency

Key Mechanism: Mensa - a new acceleration framework for edge NN inference
- Mensa consists of heterogeneous accelerators whose dataflow and hardware are specialized for specific families of layers

Key Results: We design a version of *Mensa* for *Google edge ML models*
- Mensa improves performance and energy by 3.0x and 3.1x
- Mensa reduces cost and improves area efficiency

Paper Discussion:

Google Neural Network Models for Edge Devices:
Analyzing and Mitigating Machine Learning Inference Bottlenecks

Amirali Boroumand[1,2], Saugata Ghose[3], Berkin Akin[4], Ravi Narayanaswami[4],
Geraldo F. Oliveira[5], Xiaoyu Ma[4], Eric Shiu[4], Onur Mutlu[5,1]

[1] Carnegie Mellon Univ., [2] Stanford Univ. ,
[3] Univ. of Illinois Urbana-Champaign, [4] Google, [5] ETH Zürich

PACT 2021

Presented by Lotte Seifert

Overview

Strengths

Weaknesses

Outlook

Discussion

Strengths

1. Layer-Level Study of NN Models

- **Novelty:**
  - First quantification of **intra-model variation** within edge models compared to traditional ones
- **Mechanism:**
  - Investigation at the level of **layer granularity** generated relevant insights
- **Evaluation:**
  - Extraction of **layer clusters** with high degree of validity
  - Demonstration of monolithic design as a **root cause** for TPU inefficiencies

Strengths

1. Layer-Level Study of NN Models

2. Mensa Multi-Accelerator Framework

- **Novelty:**
  - First ML accelerator to **exploit computational and memory heterogeneity** of edge NN models
- **Mechanism:**
  - Well-designed mechanism to **overcome** the **shortcomings** of monolithic design
  - Processing in memory is an **active area of research**
- **Evaluation:**
  - **Practical** through its **integration** into the existing architecture stack
  - **Application potential** of multi-accelerator framework **beyond the edge devices**
    - Within Data Centers?
    - Processing in memory? Processing in storage?

Strengths

1. Layer-Level Study of NN Models

2. Mensa Multi-Accelerator Framework

3. Mensa G

- **Novelty:**
  - First **implementation** of **Mensa accelerator** framework for 24 Google Edge NN models
- **Mechanism:**
  - Mapping of layer features into family clusters effectively **limits number** of **heterogeneous accelerators**
  - Well-explained **design choices**
- **Evaluation:**
  - Significantly **higher energy efficiency and performance** than Edge TPU and Eyeriss v2

Strengths

1. Layer-Level Study of NN Models

2. Mensa Multi-Accelerator Framework

3. Mensa G

4. Performance analysis of Google Edge TPU

- **Novelty:**
  - First in-depth, **well-crafted performance analysis** of Google Edge TPU
- **Mechanism:**
  - Straightforward application of **standard analysis procedures**
- **Evaluation:**
  - Clear identification of **key shortcoming**

Weaknesses

1. Performance analysis of Google Edge TPU

- **Mechanism:**
  - **Reproducibility** and **transferability** of results due to proprietary models and architecture
    - Anticipation of results for popular public models
- **Evaluation:**
  - Weighting of various NN models according to their **importance** and **frequency distribution**
  - Deployment of Google Edge TPUs and the **significance of their inefficiencies**
  - **Trade-off design decisions** during Google Edge TPU development

Weaknesses

1. Performance analysis of Google Edge TPU

2. Mensa Multi-Accelerator Framework

- **Mechanism:**
  - **Future proofness** in light of new families / accelerators through NN model development
- **Evaluation:**
  - Runtime **scheduler overhead**

Weaknesses

1. Performance analysis of Google Edge TPU
2. Mensa Multi-Accelerator Framework
3. Layer-Level Study of NN Models

- **Evaluation:**
  - **Applicability** of **layer clusters** to other edge NN models

Weaknesses

1. Performance analysis of Google Edge TPU
2. Mensa Multi-Accelerator Framework
3. Layer-Level Study of NN Models
4. Mensa G

- **Mechanism:**
  - Development neglects **frequency considerations** of different layer families
- **Evaluation:**
  - **Suitability** of Google Edge TPU as **evaluation baseline**
    - Google Edge TPU with better scheduling as evaluation baseline
    - CPU performance as evaluation baseline
  - Assessment based on **simulated results** that disregard frequency considerations

Outlook

Will Edge ML Accelerators remain important?

Consumer Demand

- Privacy
- Connectivity
- Low Latency
- Bandwidth

Technology

- Internet of Things
- 5G

Edge Computing

"Computation, storage, and analysis of data close to its creation and consumption"

- Constrained power budget
- Limited computational resources

Accelerator Deployment

- Google Edge TPU
- NVIDIA Jetson
- Intel Movidius
- Apple Neural Engine

Algorithm Development

- Neural Network (NN) Models

Alternative Ideas / Discussion

- Is a Multi-Accelerator Framework the best solution?
  - Address issues through better scheduling?
  - Address issues through better memory footprint (i.e. smaller buffer and/or better bandwidth)?
  - Address issues through heterogeneous PE’s?
  - Address issues through model / layer aware prefetching?
  - Address issues through a combination of the above?
- Design Multi-Accelerator Framework with NN model developments in mind?
  - Recommender systems
- Optimize Edge NN model compilation with hardware in mind?
  - Which optimization criteria govern the current tradeoff?