Efficient Computing for
Deep Learning, AI and Robotics

Vivienne Sze (@eems_mit)
Massachusetts Institute of Technology


Slides available at
https://tinyurl.com/SzeMITDL2020
Compute Demands for Deep Neural Networks

AlexNet to AlphaGo Zero: A 300,000x Increase in Compute

Petaflop/s-days (exponential)

Source: Open AI (https://openai.com/blog/ai-and-compute/)

Vivienne Sze (✈️@eems_mit)
Compute Demands for Deep Neural Networks

Common carbon footprint benchmarks
in lbs of CO2 equivalent

- Roundtrip flight b/w NY and SF (1 passenger) 1,984
- Human life (avg. 1 year) 11,023
- American life (avg. 1 year) 36,156
- US car including fuel (avg. 1 lifetime) 126,000
- Transformer (213M parameters) w/ neural architecture search 626,155

[Strubell, ACL 2019]
Processing at “Edge” instead of the “Cloud”

Communication  Privacy  Latency
Computing Challenge for Self-Driving Cars

Cameras and radar generate ~6 gigabytes of data every 30 seconds.

Self-driving car prototypes use approximately 2,500 Watts of computing power.

Generates wasted heat and some prototypes need water-cooling!

Vivienne Sze (＠eems_mit)
Existing Processors Consume Too Much Power

- < 1 Watt
- > 10 Watts
Transistors are NOT Getting More Efficient

Slow down of Moore’s Law and Dennard Scaling

General purpose microprocessors not getting faster or more efficient

- Need **specialized hardware** for significant improvement in speed and energy efficiency
- Redesign computing hardware from the ground up!
“Today, at least 45 start-ups are working on chips that can power tasks like speech and self-driving cars, and at least five of them have raised more than $100 million from investors. Venture capitalists invested more than $1.5 billion in chip start-ups last year, nearly doubling the investments made two years ago, according to the research firm CB Insights.”
### Power Dominated by Data Movement

<table>
<thead>
<tr>
<th>Operation</th>
<th>Energy (pJ)</th>
<th>Relative Energy Cost</th>
<th>Area (µm²)</th>
<th>Relative Area Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>8b Add</td>
<td>0.03</td>
<td></td>
<td>36</td>
<td></td>
</tr>
<tr>
<td>16b Add</td>
<td>0.05</td>
<td></td>
<td>67</td>
<td></td>
</tr>
<tr>
<td>32b Add</td>
<td>0.1</td>
<td></td>
<td>137</td>
<td></td>
</tr>
<tr>
<td>16b FP Add</td>
<td>0.4</td>
<td></td>
<td>1360</td>
<td></td>
</tr>
<tr>
<td>32b FP Add</td>
<td>0.9</td>
<td></td>
<td>4184</td>
<td></td>
</tr>
<tr>
<td>8b Mult</td>
<td>0.2</td>
<td></td>
<td>282</td>
<td></td>
</tr>
<tr>
<td>32b Mult</td>
<td>3.1</td>
<td></td>
<td>3495</td>
<td></td>
</tr>
<tr>
<td>16b FP Mult</td>
<td>1.1</td>
<td></td>
<td>1640</td>
<td></td>
</tr>
<tr>
<td>32b FP Mult</td>
<td>3.7</td>
<td></td>
<td>7700</td>
<td></td>
</tr>
<tr>
<td>32b SRAM Read (8KB)</td>
<td>5</td>
<td></td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>32b DRAM Read</td>
<td>640</td>
<td></td>
<td>N/A</td>
<td></td>
</tr>
</tbody>
</table>

Memory access is **orders of magnitude** higher energy than compute

[Horowitz, ISSCC 2014]
Autonomous Navigation Uses a Lot of Data

• **Semantic Understanding**
  - High frame rate
  - Large resolutions
  - Data expansion

  ![Image of people and building](image1)
  2 million pixels
  \[\text{[Pire, RAS 2017]}\]

• **Geometric Understanding**
  - Growing map size

  ![Image of map and diagram](image2)
  10x-100x more pixels
Understanding the Environment

Depth Estimation

State-of-the-art approaches use Deep Neural Networks, which require up to several hundred millions of operations and weights to compute! >100x more complex than video compression.
Deep Neural Networks (DNNs) have become a cornerstone of AI.
**What Are Deep Neural Networks?**

Input: Image

Low Level Features

High Level Features

Output: “Volvo XC90”

Modified Image Source: [Lee, CACM 2011]
Weighted Sum

\[ Y_j = \text{Nonlinear Activation Function} \left( \sum_{i=1}^{3} W_{ij} \times X_i \right) \]

Key operation is **multiply and accumulate (MAC)**
Accounts for > 90% of computation

Image source: Caffe tutorial
Popular Types of Layers in DNNs

- **Fully Connected Layer**
  - Feed forward, fully connected
  - Multilayer Perceptron (MLP)

- **Convolutional Layer**
  - Feed forward, sparsely-connected with weight sharing
  - Convolutional Neural Network (CNN)
  - Typically used for images

- **Recurrent Layer**
  - Feedback
  - Recurrent Neural Network (RNN)
  - Typically used for sequential data (e.g., speech, language)

- **Attention Layer/Mechanism**
  - Attention (matrix multiply) + feed forward, fully connected
  - Transformer [Vaswani, NeurIPS 2017]
High-Dimensional Convolution in CNN

a plane of input activations
a.k.a. input feature map (fmap)

filter (weights)
High-Dimensional Convolution in CNN

Element-wise Multiplication

Partial Sum ($psum$) Accumulation

input fmap

output fmap

filter (weights)
High-Dimensional Convolution in CNN

Filter (weights)

Input fmap

Output fmap

Sliding Window Processing

Vivienne Sze (@eems_mit)
High-Dimensional Convolution in CNN

Many Input Channels (C)

AlexNet: 3 – 192 Channels (C)
High-Dimensional Convolution in CNN

Many filters (M)

Input fmap

Output fmap

Many output channels (M)

AlexNet: 96 – 384 filters (M)
High-Dimensional Convolution in CNN

Many Input fmaps (N)

Many Output fmaps (N)

Image batch size: 1 – 256 (N)
Define Shape for Each Layer

Shape varies across layers

- **H** – Height of input fmap (activations)
- **W** – Width of input fmap (activations)
- **C** – Number of 2-D input fmaps/filters (channels)
- **R** – Height of 2-D filter (weights)
- **S** – Width of 2-D filter (weights)
- **M** – Number of 2-D output fmaps (channels)
- **E** – Height of output fmap (activations)
- **F** – Width of output fmap (activations)
- **N** – Number of input fmaps/output fmaps (batch size)
## MobileNetV3-Large Convolutional Layer Configurations

<table>
<thead>
<tr>
<th>Block</th>
<th>Filter Size (RxS)</th>
<th># Filters (M)</th>
<th># Channels (C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3x3</td>
<td>16</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1x1</td>
<td>64</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>3x3</td>
<td>64</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1x1</td>
<td>24</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1x1</td>
<td>120</td>
<td>40</td>
</tr>
<tr>
<td>6</td>
<td>5x5</td>
<td>120</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1x1</td>
<td>40</td>
<td>120</td>
</tr>
</tbody>
</table>

[Howard, ICCV 2019]
## Popular DNN Models

<table>
<thead>
<tr>
<th>Metrics</th>
<th>LeNet-5</th>
<th>AlexNet</th>
<th>VGG-16</th>
<th>GoogLeNet (v1)</th>
<th>ResNet-50</th>
<th>EfficientNet-B4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top-5 error (ImageNet)</td>
<td>n/a</td>
<td>16.4</td>
<td>7.4</td>
<td>6.7</td>
<td>5.3</td>
<td>3.7*</td>
</tr>
<tr>
<td>Input Size</td>
<td>28x28</td>
<td>227x227</td>
<td>224x224</td>
<td>224x224</td>
<td>224x224</td>
<td>380x380</td>
</tr>
<tr>
<td># of CONV Layers</td>
<td>2</td>
<td>5</td>
<td>16</td>
<td>21 (depth)</td>
<td>49</td>
<td>96</td>
</tr>
<tr>
<td># of Weights</td>
<td>2.6k</td>
<td>2.3M</td>
<td>14.7M</td>
<td>6.0M</td>
<td>23.5M</td>
<td>14M</td>
</tr>
<tr>
<td># of MACs</td>
<td>283k</td>
<td>666M</td>
<td>15.3G</td>
<td>1.43G</td>
<td>3.86G</td>
<td>4.4G</td>
</tr>
<tr>
<td># of FC layers</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>65**</td>
</tr>
<tr>
<td># of Weights</td>
<td>58k</td>
<td>58.6M</td>
<td>124M</td>
<td>1M</td>
<td>2M</td>
<td>4.9M</td>
</tr>
<tr>
<td># of MACs</td>
<td>58k</td>
<td>58.6M</td>
<td>124M</td>
<td>1M</td>
<td>2M</td>
<td>4.9M</td>
</tr>
<tr>
<td>Total Weights</td>
<td>60k</td>
<td>61M</td>
<td>138M</td>
<td>7M</td>
<td>25.5M</td>
<td>19M</td>
</tr>
<tr>
<td>Total MACs</td>
<td>341k</td>
<td>724M</td>
<td>15.5G</td>
<td>1.43G</td>
<td>3.9G</td>
<td>4.4G</td>
</tr>
</tbody>
</table>

*DNN models getting larger and deeper*

* Does not include multi-crop and ensemble
** Increase in FC layers due to squeeze-and-excitation layers (much smaller than FC layers for classification)
Efficient Hardware Acceleration for Deep Neural Networks
Properties We Can Leverage

• Operations exhibit **high parallelism** → **high throughput** possible

• Memory Access is the Bottleneck

**Memory Read** | **MAC** \(^*\) | **Memory Write**

DRAM | ALU | DRAM

filter weight | image pixel | updated partial sum

partial sum | \(200x\) | partial sum

* multiply-and-accumulate

**Worst Case**: all memory R/W are **DRAM** accesses

• Example: AlexNet has **724M** MACs → **2896M** DRAM accesses required
Properties We Can Leverage

- Operations exhibit **high parallelism** → **high throughput** possible

- **Input data reuse** opportunities (**up to 500x**)
Exploit Data Reuse at Low-Cost Memories

Normalized Energy Cost*

- 1× (Reference)
- 1×
- 2×
- 6×
- 200×

* measured from a commercial 65nm process

Farther and larger memories consume more power

Drum

Global Buffer

PE

PE

PE

ALU

Reg File

Control

NoC: 200 – 1000 PEs

0.5 – 1.0 kB

100 – 500 kB

DRAM

RF

Control

Farther and larger memories consume more power

Specialized hardware with small (<1kB) low cost memory near compute
Weight Stationary (WS)

- Minimize weight read energy consumption
  - maximize convolutional and filter reuse of weights

- Broadcast activations and accumulate partial sums spatially across the PE array

- Examples: TPU [Jouppi, ISCA 2017], NVDLA

[Chen, ISCA 2016]
Output Stationary (OS)

- Minimize partial sum R/W energy consumption
  - maximize local accumulation

- Broadcast/Multicast filter weights and reuse activations spatially across the PE array

- Examples: [Moons, VLSI 2016], [Thinker, VLSI 2017]
• Minimize activation read energy consumption
  – maximize convolutional and fmap reuse of activations

• Unicast weights and accumulate partial sums spatially across the PE array

• Example: [SCNN, ISCA 2017]
Row Stationary Dataflow

- Maximize row convolutional reuse in RF
  - Keep a filter row and fmap sliding window in RF

- Maximize row psum accumulation in RF

$\text{Row 1} \times \text{Row 1} = \text{Result}$

[Chen, ISCA 2016] Select for Micro Top Picks
Optimize for **overall energy efficiency** instead for only a certain data type

[Chen, ISCA 2016] Select for Micro Top Picks
Dataflow Comparison: CONV Layers

RS optimizes for the best **overall** energy efficiency

---

Vivienne Sze (@eems_mit)

[Chen, ISCA 2016]
Exploit Sparsity

**Method 1. Skip memory access and computation**

Method 1. Skip memory access and computation

No R/W

No Switching

**Method 2. Compress data to reduce storage and data movement**

Method 2. Compress data to reduce storage and data movement

45% power reduction

Uncompressed Fmaps + Weights

RLE Compressed Fmaps + Weights

[Chen, ISSCC 2016]
**Eyeriss: Deep Neural Network Accelerator**

Exploits data reuse for **100x** reduction in memory accesses from global buffer and **1400x** reduction in memory accesses from off-chip DRAM

Overall **>10x energy reduction** compared to a mobile GPU (Nvidia TK1)

Eyeriss Project Website: [http://eyeriss.mit.edu](http://eyeriss.mit.edu)

Results for AlexNet

Vivienne Sze (@eems_mit)  

[Joint work with Joel Emer]
Features: Energy vs. Accuracy

Accuracy (Average Precision)

- **VGG16**
- **AlexNet**
- **HOG**

Energy/ Pixel (nJ)

Measured in 65nm*

* Only feature extraction. Does not include data, classification energy, augmentation and ensemble, etc.

- [Suleiman, VLSI 2016]
- [Chen, ISSCC 2016]

Measured in on VOC 2007 Dataset
1. DPM v5 [Girshick, 2012]

Vivienne Sze (@eems_mit)

[Suleiman*, Chen*, ISCAS 2017]
Energy-Efficient Processing of DNNs

A significant amount of algorithm and hardware research on energy-efficient processing of DNNs

We identified various limitations to existing approaches


Book Coming
Spring 2020!
Design of Efficient DNN Algorithms

• Popular efficient DNN algorithm approaches

Network Pruning

before pruning

pruning synapses

pruning neurons

after pruning

Efficient Network Architectures

Examples: SqueezeNet, MobileNet

... also reduced precision

• Focus on reducing **number of MACs and weights**
• Does it translate to energy savings and reduced latency?

Vivienne Sze (@eems_mit)

[Chen*, Yang*, SysML 2018]
Data Movement is Expensive

Energy of weight depends on memory hierarchy and dataflow

Normalized Energy Cost*

- 1× (Reference)
- 1×
- 2×
- 6×
- 200×

* measured from a commercial 65nm process
Energy-Evaluation Methodology

DNN Shape Configuration
(# of channels, # of filters, etc.)

DNN Weights and Input Data
[0.3, 0, -0.4, 0.7, 0, 0, 0.1, …]

Hardware Energy Costs of each
MAC and Memory Access

Memory Accesses Optimization

# of MACs Calculation

# acc. at mem. level 1
# acc. at mem. level 2
⋯
# acc. at mem. level n

# of MACs

E\text{comp}

E\text{data}

Energy

DNN Energy Consumption

Tool available at: https://energyestimation.mit.edu/

[Yang, CVPR 2017]
Key Observations

- Number of weights *alone* is not a good metric for energy
- **All data types** should be considered

Energy Consumption of GoogLeNet

- Output Feature Map: 43%
- Input Feature Map: 25%
- Weights: 22%
- Computation: 10%

[Yang, CVPR 2017]
Directly target energy and incorporate it into the optimization of DNNs to provide greater energy savings.

- Sort layers based on energy and prune layers that consume most energy first.
- EAP reduces AlexNet energy by 3.7x and outperforms the previous work that uses magnitude-based pruning by 1.7x.

Pruned models available at http://eyeriss.mit.edu/energy.html

[Yang, CVPR 2017]
# of Operations vs. Latency

- # of operations (MACs) does not approximate latency well


Vivienne Sze (✈️@eems_mit)
NetAdapt: Platform-Aware DNN Adaptation

- Automatically adapt DNN to a mobile platform to reach a target latency or energy budget
- Use empirical measurements to guide optimization (avoid modeling of tool chain or platform architecture)

![Diagram of NetAdapt process]

- Pretrained Network
- NetAdapt
- Empirical Measurements
- Platform
- Measure
- Adapted Network

<table>
<thead>
<tr>
<th>Metric</th>
<th>Budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency</td>
<td>3.8</td>
</tr>
<tr>
<td>Energy</td>
<td>10.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metric</th>
<th>Proposal A</th>
<th>...</th>
<th>Proposal Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency</td>
<td>15.6</td>
<td>...</td>
<td>14.3</td>
</tr>
<tr>
<td>Energy</td>
<td>41</td>
<td>...</td>
<td>46</td>
</tr>
</tbody>
</table>

Code available at [http://netadapt.mit.edu](http://netadapt.mit.edu) [Yang, ECCV 2018]

In collaboration with Google’s Mobile Vision Team
Simplified Example of One Iteration

1. Input

Network from Previous Iteration

Latency: 100ms
Budget: 80ms

2. Meet Budget

Layer 1

100ms 90ms 80ms

Selected

Layer 4

100ms 80ms

Selected

3. Maximize Accuracy

Acc: 60%

4. Output

Network for Next Iteration

Latency: 80ms
Budget: 60ms

[Vivienne Sze (@eems_mit)]

[ Yang, ECCV 2018 ]
Improved Latency vs. Accuracy Tradeoff

- NetAdapt boosts the real inference speed of MobileNet by up to 1.7x with higher accuracy

Reference:

*Tested on the ImageNet dataset and a Google Pixel 1 CPU
Depth estimation from a single RGB image desirable, due to the relatively low cost and size of monocular cameras.

FastDepth: Fast Monocular Depth Estimation

Auto Encoder DNN Architecture (Dense Output)

[Joint work with Sertac Karaman]
FastDepth: Fast Monocular Depth Estimation

Apply NetAdapt, compact network design, and depth wise decomposition to decoder layer to enable depth estimation at high frame rates on an embedded platform while still maintaining accuracy.

Configuration: Batch size of one (32-bit float)

Models available at [http://fastdepth.mit.edu](http://fastdepth.mit.edu)

[Wofk*, Ma*, ICRA 2019]
Many Efficient DNN Design Approaches

Network Pruning

Efficient Network Architectures

Reduce Precision

32-bit float: $101001010000000000001010000000000100$

8-bit fixed: $01100110$

Binary: $0$

No guarantee that DNN algorithm designer will use a given approach.

Need flexible hardware!

[Chen*, Yang*, SysML 2018]
• Specialized DNN hardware often rely on certain properties of DNN in order to achieve high energy-efficiency

• **Example:** Reduce memory access by amortizing across MAC array
Limitation of Existing DNN Architectures

**Example:** Reuse and array utilization depends on # of channels, feature map/batch size
- Not efficient across all network architectures (e.g., compact DNNs)
Limitation of Existing DNN Architectures

- **Example**: Reuse and array utilization depends on # of channels, feature map/batch size
  - Not efficient across all network architectures (e.g., compact DNNs)

Example mapping for depth wise layer

- Number of input channels
- Number of filters (output channels)
- MAC array (spatial accumulation)
- MAC array (temporal accumulation)
- Number of filters (output channels)
- Feature map or batch size
**Limitation of Existing DNN Architectures**

- **Example:** Reuse and array utilization depends on # of channels, feature map/batch size
  - Not efficient across all network architectures (e.g., compact DNNs)
  - Less efficient as array scales up in size
  - Can be challenging to exploit sparsity

![Diagram showing MAC array and its parameters](image-url)
Need Flexible Dataflow

• Use flexible dataflow (Row Stationary) to exploit reuse in any dimension of DNN to increase energy efficiency and array utilization

Example: Depth-wise layer
Need Flexible NoC for Varying Reuse

- When reuse available, need **multicast** to exploit spatial data reuse for energy efficiency and high array utilization
- When reuse not available, need **unicast** for high BW for weights for FC and weights & activations for high PE utilization
- An all-to-all satisfies above but too expensive and not scalable

---

[Vivienne Sze (@eems_mit)]

[Chen, *JETCAS* 2019]
Hierarchical Mesh

Mesh

GLB Cluster

Router Cluster

PE Cluster

Mesh Network

All-to-All

High Bandwidth

High Reuse

Grouped Multicast

Interleaved Multicast

[Chen, JETCAS 2019]
Eyeriss v2: Balancing Flexibility and Efficiency

Efficiently supports
- Wide range of filter shapes  
  - Large and Compact
- Different Layers  
  - CONV, FC, depth wise, etc.
- Wide range of sparsity  
  - Dense and Sparse
- Scalable architecture

Over an order of magnitude faster and more energy efficient than Eyeriss v1

Speed up over Eyeriss v1 scales with number of PEs

<table>
<thead>
<tr>
<th># of PEs</th>
<th>256</th>
<th>1024</th>
<th>16384</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet</td>
<td>17.9x</td>
<td>71.5x</td>
<td>1086.7x</td>
</tr>
<tr>
<td>GoogLeNet</td>
<td>10.4x</td>
<td>37.8x</td>
<td>448.8x</td>
</tr>
<tr>
<td>MobileNet</td>
<td>15.7x</td>
<td>57.9x</td>
<td>873.0x</td>
</tr>
</tbody>
</table>

[Chen, JETCAS 2019]
Looking Beyond the DNN Accelerator for Acceleration
Super-Resolution on Mobile Devices

Transmit low resolution for lower bandwidth

Screens are getting larger

Use **super-resolution** to improve the viewing experience of lower-resolution content (*reduce communication bandwidth*)
FAST: A Framework to Accelerate SuperRes

A framework that accelerates any SR algorithm by up to 15x when running on compressed videos

[Zhang, CVPRW 2017]
Free Information in Compressed Videos

Compressed video

Video as a stack of pixels

Pixels

Block-structure

Motion-compensation

This representation can help **accelerate** super-resolution

Vivienne Sze (🐦@eems_mit)

[Zhang, CVPRW 2017]
**Transfer is Lightweight**

Transfer allows SR to run on only a **subset of frames**

The complexity of the transfer is comparable to bicubic interpolation.

Transfer $N$ frames, accelerate by $N$

---

Vivienne Sze (@eems.mit)  

[Zhang, CVPRW 2017]
Evaluation: Accelerating SRCNN

Examples of videos in the test set (20 videos for HEVC development)

4 × acceleration with NO PSNR LOSS. 16 × acceleration with 0.2 dB loss of PSNR

[Zhang, CVPRW 2017]
Look **beyond** the DNN accelerator for opportunities to accelerate DNN processing (e.g., structure of data and temporal correlation)

Code released at [www.rle.mit.edu/eems/fast](http://www.rle.mit.edu/eems/fast)

[Vivienne Sze (@eems_mit)](https://twitter.com/eems_mit)

[Zhang, CVPRW 2017]
Beyond Deep Neural Networks
Visual-Inertial Localization

Determines location/orientation of robot from images and IMU

*Subset of SLAM algorithm
(Simultaneous Localization And Mapping)
Localization at Under 25 mW

*First chip* that performs *complete* Visual-Inertial Odometry

**Front-End for camera**  
(*Feature detection, tracking, and outlier elimination*)

**Front-End for IMU**  
(*pre-integration of accelerometer and gyroscope data*)

**Back-End Optimization of Pose Graph**

Consumes $684 \times$ and $1582 \times$ less energy than mobile and desktop CPUs, respectively

Navion Project Website: [http://navion.mit.edu](http://navion.mit.edu)  
[Zhang et al., RSS 2017], [Suleiman et al., VLSI 2018]

Vivienne Sze (🐦@eems.mit)  
[Joint work with Sertac Karaman]
Key Methods to Reduce Data Size

**Navion**: Fully integrated system – no off-chip processing or storage

![Diagram showing flow of data and control between Vision Frontend (VFE) and Backend (BE).]

- **Vision Frontend (VFE)**
  - Line Buffers
  - Feature Detection (FD)
  - Undistort & Rectify (UR)
  - Feature Tracking (FT)
  - Left Frame
  - Right Frame
  - Sparse Stereo (SS)
  - Vision Frontend Control
  - Data & Control Bus
  - RANSAC
  - Fixed Point Arithmetic
  - Point Cloud

- **Backend (BE)**
  - Backend Control
  - Data & Control Bus
  - Build Graph
  - Linearize
  - Linear Solver
  - Horizon States
  - Shared Memory
  - Register File
  - Graph
  - Linear Solver
  - Rodrigues Operations
  - Back Substitute
  - Marginal
  - Retract
  - Register File

Exploit Sparsity in Graph and Linear Solver

**Use compression and exploit sparsity** to reduce memory down to 854kB

---

Vivienne Sze (@eems_mit)  
[Suleiman, VLSI-C 2018]  
Best Student Paper Award
**Where to Go Next: Planning and Mapping**

**Robot Exploration:** Decide where to go by computing Shannon Mutual Information

1. Select candidate scan locations
2. Compute Shannon MI and choose best location
3. Move to location and scan
4. Update Occupancy Map

**Where to scan?**

- Occupancy map
- Mutual Information map
- MI surface

**Updated Map**

**Usage:**

- Exploration with a mini race car using motion capture for localization
- Occupancy map with planned path

**References:**

- [Zhang, ICRA 2019]
Challenge is Data Delivery to All Cores

Process multiple beams in parallel

Data delivery from memory is limited
Specialized Memory Architecture

Break up map into separate memory banks and novel storage pattern to minimize read conflicts when processing different beams in parallel.

Compute the mutual information for an entire map of 20m x 20m at 0.1m resolution in under a second → a 100x speed up versus CPU for 1/10th of the power.

[Joint work with Sertac Karaman]
Monitoring Neurodegenerative Disorders

Dementia affects 50 million people worldwide today (75 million in 10 years) [World Alzheimer’s Report]

Mini-Mental State Examination (MMSE)

Q1. What is the year? Season? Date?
Q2. Where are you now? State? Floor?
Q3. Could you count backward from 100 by sevens? (93, 86, ...)

Clock-drawing test


- Neuropsychological assessments are **time consuming** and **require a trained specialist**
- Repeat **medical assessments** are **sparse**, mostly **qualitative**, and suffer from **high retest variability**
Use Eye Movements for *Quantitative* Evaluation

Eye movements can be used to quantitatively evaluate severity, progression or regression of neurodegenerative diseases.

- High-speed camera: *Phantom v25-11*
- Substantial head support: *SR EYELINK 1000 PLUS*
- IR illumination


Clinical measurements of saccade latency are done in constrained environments that rely on specialized, costly equipment.
Measure Eye Movements Using Phone

Eye movements → Smartphone → Count

Eye movement feature

Develop algorithm to measure eye movement using a consumer-grade camera rather than high-cost research-grade camera.

Enable low-cost in-home longitudinal measurements.

iPhone 6 (< $1k)

Phantom ($100k)

Reaction Time (milliseconds)

[iPhone - Subject 001]

\[ \mu = 136 \text{ ms}, \sigma = 29 \text{ ms} \]

\[ n = 111 \]

[Phantom - Subject 001]

\[ \mu = 136 \text{ ms}, \sigma = 29 \text{ ms} \]

\[ n = 103 \]

Vivienne Sze (@eems_mit)

[Saavedra Peña, EMBC 2018] [Lai, ICIP 2018]
Looking For Volunteers for Eye Reaction Time

If you are near or on MIT Campus and interested in volunteering your eye movements for this study, please contact us at volunteer-eye-movement@mit.edu
Low Power 3D Time of Flight Imaging

- Pulsed Time of Flight: Measure distance using round trip time of laser light for each image pixel
  - Illumination + Imager Power: 2.5 – 20 W for range from 1 - 8 m

- Use computer vision techniques and passive images to estimate changes in depth without turning on laser
  - CMOS Imaging Sensor Power: < 350 mW

Real-time Performance on Embedded Processor
VGA @ 30 fps on Cortex-A7 (< 0.5W active power)

[Noraky, ICIP 2017]
Results of Low Power Depth ToF Imaging

Mean Relative Error: 0.7%
Duty Cycle (on-time of laser): 11%

Noraky, ICIP 2017
Summary

• Efficient computing extends the reach of AI beyond the cloud by **reducing communication requirements**, **enabling privacy**, and **providing low latency** so that AI can be used in wide range of applications ranging from robotics to health care.

• **Cross-layer design with specialized hardware** enables energy-efficient AI, and will be critical to the progress of AI over the next decade.

Today’s slides available at https://tinyurl.com/SzeMITDL2020
Additional Resources

Overview Paper

Book Coming Spring 2020!

More info about Tutorial on DNN Architectures
http://eyeriss.mit.edu/tutorial.html

For updates
EEMS Mailing List
Follow @eems_mit
Additional Resources

MIT Professional Education Course on “Designing Efficient Deep Learning Systems”

http://shortprograms.mit.edu/dls

Next Offering: July 20-21, 2020 on MIT Campus
Additional Resources

Talks and Tutorial Available Online
https://www.rle.mit.edu/eems/publications/tutorials/

YouTube Channel
EEMS Group – PI: Vivienne Sze

Vivienne Sze (@eems_mit)
Acknowledgements

Research conducted in the MIT Energy-Efficient Multimedia Systems Group would not be possible without the support of the following organizations:

Vivienne Sze (@eems.mit)  Mailing List: http://mailman.mit.edu/mailman/listinfo/eems-news
References

• Energy-Efficient Hardware for Deep Neural Networks
  – Project website: http://eyeriss.mit.edu

• Limitations of Existing Efficient DNN Approaches
  – Hardware Architecture for Deep Neural Networks: http://eyeriss.mit.edu/tutorial.html
References

• Co-Design of Algorithms and Hardware for Deep Neural Networks

• Energy-Efficient Visual Inertial Localization
  – Project website: http://navion.mit.edu
References

- **Fast Shannon Mutual Information for Robot Exploration**

- **Low Power Time of Flight Imaging**

- **Monitoring Neurodegenerative Disorders Using a Phone**

Vivienne Sze (🐦@eems_mit)