Energy-Efficient Processing at the Edge: From Compressing to Understanding Pixels

Vivienne Sze
Massachusetts Institute of Technology

In collaboration with Luca Carlone, Anantha Chandrakasan, Yu-Hsin Chen, Joel Emer, Sertac Karaman, Tushar Krishna, Amr Suleiman, Mehul Tikekar, Tien-Ju Yang, Zhengdong Zhang

Contact Info
email: sze@mit.edu
website: www.rle.mit.edu/eems
Video is the Biggest Big Data

Over 70% of today’s Internet traffic is video
Over 300 hours of video uploaded to YouTube every minute
Over 500 million hours of video surveillance collected every day

Energy limited due to battery capacity
Power limited due to heat dissipation

Need energy-efficient pixel processing!
Energy-Efficient Pixel Processing

Next-Generation Video Coding (Compress Pixels)

Goal: Increase coding efficiency, speed and energy-efficiency

Ultra-HD

Energy-Efficient Computer Vision & Deep Learning (Understand Pixels)

Goal: Make computer vision as ubiquitous as video coding
Energy-Efficient Video Compression
High Efficiency Video Coding (HEVC)

- HEVC achieves ~2x higher coding efficiency than H.264/AVC
- High throughput (Ultra-HD 8K @ 120fps) & low power
  - Implementation-friendly features (e.g. built-in parallelism)

<table>
<thead>
<tr>
<th></th>
<th>Coding Efficiency</th>
<th>Efficient Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Larger and Flexible Coding Block Size</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>More Sophisticated Intra Prediction</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Larger Interpolation for Motion Comp.</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Larger Transform Size</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Parallel Deblocking Filter</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Sample Adaptive Offset</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>High-Throughput CABAC</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>High Level Parallel Tools</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

Joint algorithm and hardware design is required to address coding efficiency, throughput and power challenges
**Efficient Hardware for HEVC**

### Energy-Efficient HEVC Transform

- Large transforms give coding gain: 7-9%
- Adapt to sparsity of coefficients
- Enable constant energy/pixel for all transform sizes

### High-Throughput HEVC CABAC

- CABAC bottleneck in H.264/AVC
- HEVC 4x faster than H.264/AVC
- Ultra-HD 8K @ 120fps
- Also gives coding gain: 3-5%

---

**[M. Tikekar et al., ICIP 2014]**  
**[Y.-H. Chen et al., TCSVT 2015]**
Low Power HEVC Decoder for Wearable Devices

Reduce On-chip and Off-chip Data Movement

- Memory consumes 2.8x to 6x more energy than video decoder
- Use low cost compression to reduce storage cost of frame buffer
- Total decoder system power of 25mW

[12, 5, 2, 3] - [2] = [10, 3, 0, 1]

floor(log₂(x + 1))

minimum (M : 8b)
delta (D : 16 × 4b)
range (R : 4b)

4 × 4 block of pixels (16 × 8b)

[Tikekar et al., VLSI 2017]
Compression Inspired by Computer Vision
SIFT features are widely used to establish correspondence between two similar images

[DG Lowe, IJCV 2004]
Before the matching, SIFT descriptors are normalized to the canonical pose (dominant gradient) so that patches of different orientation can be matched.
Intra Block Copy for Still Image Coding

Use one block to **predict** repetitive blocks. Only encode the **difference (residual)**.

**NOT** rotate invariant. Limited to screen content.

[Yu et al., JVT-C151], [Budagavi et al., JCTVC-M0350], [Peng et al., JCTVC-N0256]
Repetitive structures with rotation
In both screen content and camera captured images
Reduction of Residual Energy

HEVC  
HEVC + Block Copy  
HEVC + Rotate Block Copy

40% reduction of residual energy over HEVC  
27% reduction of residual energy over HEVC + Block Copy

However, there is overhead in signaling the rotate angle and motion vector

First frame of ParkScene Sequence

[Z. Zhang et al., ICIP 2015]
Motion vectors need to be on the same rotated coordinate system.

\[ \text{Reduce average bit rate of motion vector difference by 25\%} \]

\[ \text{mv}_{\theta_2}^{(2)} \text{ is encoded as } \text{mv}_{\theta_2}^{(2)} - \text{round} \left( R_{\theta_2 - \theta_1} \text{mv}_{\theta_1}^{(1)} \right) \]

Where 
\[ R_{\theta_2 - \theta_1} = \begin{bmatrix} \cos(\theta_2 - \theta_1) & -\sin(\theta_2 - \theta_1) \\ \sin(\theta_2 - \theta_1) & \cos(\theta_2 - \theta_1) \end{bmatrix} \]

[Z. Zhang et al., ICIP 2015]
## HEVC + Intra Block Copy vs. HEVC + Rotate Intra Block Copy

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Residual reduction</th>
<th>BD-rate reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Class C</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RaceHorse</td>
<td>23.66%</td>
<td>4.75</td>
</tr>
<tr>
<td>PartyScene</td>
<td>27.64%</td>
<td>4.65</td>
</tr>
<tr>
<td>BQMall</td>
<td>17.92%</td>
<td>2.70</td>
</tr>
<tr>
<td>BasketballDrill</td>
<td>22.12%</td>
<td>3.52</td>
</tr>
<tr>
<td><strong>Class D</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BQSquare</td>
<td>30.82%</td>
<td>5.25</td>
</tr>
<tr>
<td>BasketballPass</td>
<td>15.44%</td>
<td>1.87</td>
</tr>
<tr>
<td>BlowingBubbles</td>
<td>7.59%</td>
<td>2.88</td>
</tr>
<tr>
<td>RaceHorse</td>
<td>28.97%</td>
<td>4.62</td>
</tr>
<tr>
<td><strong>Class E</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FourPeople</td>
<td>18.09%</td>
<td>2.60</td>
</tr>
<tr>
<td>Johnny</td>
<td>12.79%</td>
<td>2.41</td>
</tr>
<tr>
<td>KristenAndSara</td>
<td>15.67%</td>
<td>2.48</td>
</tr>
<tr>
<td><strong>Class F</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BasketballDrillText</td>
<td>21.15%</td>
<td>3.78</td>
</tr>
<tr>
<td>SlideShow</td>
<td>29.01%</td>
<td>8.03</td>
</tr>
<tr>
<td>SlideEditing</td>
<td>19.12%</td>
<td>0.74</td>
</tr>
<tr>
<td><strong>Class C Average</strong></td>
<td>22.83%</td>
<td>3.91</td>
</tr>
<tr>
<td><strong>Class D Average</strong></td>
<td>20.70%</td>
<td>3.66</td>
</tr>
<tr>
<td><strong>Class E Average</strong></td>
<td>15.52%</td>
<td>2.50</td>
</tr>
<tr>
<td><strong>Class F Average</strong></td>
<td>23.09%</td>
<td>4.18</td>
</tr>
<tr>
<td><strong>Overall Average</strong></td>
<td>20.71%</td>
<td>3.60</td>
</tr>
</tbody>
</table>

Evaluate on First Frame of JCT-VC test sequences

- Residual Energy reduction of 20.7%
- BD-rate savings of 3.6%

[Z. Zhang et al., ICIP 2015]
Energy-Efficient Deep Learning
Example Applications of Deep Learning

- **Computer Vision**
- **Speech Recognition**
- **Game Play**
- **Medical**
Using Deep Learning for Compression

Coding Tool in Video Codec

Proposed for VVC: prediction, loop filtering, upsampling, etc.

**JVET AHG: Neural Networks in Video Coding**

Intra Prediction (Upsampling) [Li et al., TCSVT 2017]

**End to End Auto Encoder**

[Google, ICLR 2016, CVPR 2017]

**Challenge:** Computation complexity higher than typical image processing
Deep Convolutional Neural Networks

Modern *deep* CNN: up to 1000 CONV layers
Deep Convolutional Neural Networks

1 – 3 layers

Classes
Deep Convolutional Neural Networks

Convolutions account for more than 90% of overall computation, dominating runtime and energy consumption.
High-Dimensional CNN Convolution

Input Image (Feature Map)

Filter

H

R

H

R
High-Dimensional CNN Convolution

Input Image (Feature Map)

Filter

Element-wise Multiplication

R

R

R

H
High-Dimensional CNN Convolution

Input Image (Feature Map)  Output Image

Filter

Element-wise Multiplication  Partial Sum (\(\text{psum}\)) Accumulation

\(R \rightarrow R\)  \(H \rightarrow H\)  \(E \rightarrow E\)

\(\times\)  \(\oplus\)

\(a\) pixel
High-Dimensional CNN Convolution

Input Image (Feature Map)  Output Image

Filter

Sliding Window Processing

a pixel
High-Dimensional CNN Convolution

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

Many Filters (M)

Input Image

Many Output Channels (M)

Output Image

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

Many Input Images (N)

Many Output Images (N)

Filters

Image batch size: 1 – 256 (N)
# Large Sizes with Varying Shapes

## AlexNet\(^1\) Convolutional Layer Configurations

<table>
<thead>
<tr>
<th>Layer</th>
<th>Filter Size (R)</th>
<th># Filters (M)</th>
<th># Channels (C)</th>
<th>Stride</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11x11</td>
<td>96</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>5x5</td>
<td>256</td>
<td>48</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3x3</td>
<td>384</td>
<td>256</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3x3</td>
<td>384</td>
<td>192</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>3x3</td>
<td>256</td>
<td>192</td>
<td>1</td>
</tr>
</tbody>
</table>

1. [Krizhevsky, NIPS 2012]
Properties We Can Leverage

- Operations exhibit **high parallelism** → **high throughput** possible
- Memory Access is the Bottleneck

**Memory Read**
- DRAM

**MAC**
- ALU
  - multiply-and-accumulate

**Memory Write**
- DRAM

\[ 200x \quad \text{to} \quad 1x \quad \text{updated partial sum} \]

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

- Example: AlexNet [NIPS 2012] 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 **low-cost memory**
Highly-Parallel Compute Paradigms

Temporal Architecture
(SIMD/SIMT)

Spatial Architecture
(Dataflow Processing)
Advantages of Spatial Architecture

Temporal Architecture (SIMD/SIMT)

- Efficient Data Reuse
  Distributed local storage (RF)

- Inter-PE Communication
  Sharing among regions of PEs

Spatial Architecture (Dataflow Processing)

- Memory Hierarchy

Processing Element (PE)

- Reg File

- Control

0.5 – 1.0 kB
Data Movement is Expensive

Maximize data reuse at lower levels of hierarchy
How to Map the Dataflow?

Goal: Increase reuse of input data (weights and pixels) and local partial sums accumulation.
Weight Stationary (WS)

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

- Examples:
  [Chakradhar, ISCA 2010]  [nn-X (NeuFlow), CVPRW 2014]
  [Park, ISSCC 2015]        [Google’s TPU, ISCA 2017]
Output Stationary (OS)

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

- Examples:
  - [Gupta, *ICML* 2015]
  - [ShiDianNao, *ISCA* 2015]
  - [Peemen, *ICCD* 2013]
Row Stationary Dataflow

Optimize for **overall energy efficiency** instead for only a certain data type for up to 2.5x energy savings

[Chen et al., Eyeriss, ISCA 2016]
Sparsity in Data

Many **zeros** in output fmaps after **ReLU**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>-1</td>
<td>-3</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>-5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>-2</td>
<td>6</td>
<td>-1</td>
<td></td>
</tr>
</tbody>
</table>

**ReLU**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>6</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

---

**# of activations**  
**# of non-zero activations**

(Normalized)
Exploit Sparsity

Method 1. Skip memory access and computation

Method 2. Compress data to reduce storage and data movement

[Chen et al., ISSCC 2016]
Eyeriss: Energy-Efficient Deep Learning

278mW for AlexNet @ 30fps (batch size 4) in 65nm LP CMOS

> 10x more energy-efficient than mobile GPUs
Features: Energy vs. Accuracy

Energy/ Pixel (nJ)

Accuracy (Average Precision)

Measured in 65nm*
1. [Suleiman, VLSI 2016]
2. [Chen, ISSCC 2016]

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

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

[Suleiman et al., ISCAS 2017]
Design of Efficient DNN Algorithms

- Popular efficient DNN algorithm approaches

**Network Pruning**

before pruning

after pruning

pruning synapses

pruning neurons

**Compact Network Architectures**

Examples: SqueezeNet, MobileNet

... also reduced precision

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

[Chen et al., SysML 2018]
Energy-Evaluation Methodology

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

CNN 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\_comp

E\_data

Energy

Energy estimation tool available at [http://eyeriss.mit.edu](http://eyeriss.mit.edu)
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 et al., CVPR 2017]
Deeper CNNs with fewer weights do not necessarily consume less energy than shallower CNNs with more weights

[Yang et al., CVPR 2017]
Reduce number of weights by removing small magnitude weights

[Han et al., NIPS 2015]

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

[Yang et al., CVPR 2017]
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)

[Yang et al., arXiv 2018]
NetAdapt boosts the real inference speed of MobileNet by 1.7x with higher accuracy

Reference:

*Tested on the ImageNet dataset and a Google Pixel 1 CPU
Hardware Architectures for Deep Neural Networks

ISCA Tutorial
June 24, 2017

Website: http://eyeriss.mit.edu/tutorial.html

Efficient Computer Vision using Compression
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*)
Complexity of Super Resolution Algorithms

**SRCNN** (Dong et al. ECCV 14)

State-of-the-art super resolution algorithms use CNNs → **computationally expensive**, especially at high resolutions (HD or 4K)

8032 MACs/pixel → ~500 GMAC/s for HD @ 30 fps
A framework that accelerates any SR algorithm by up to 15x when running on compressed videos.

[Zhang et al., CVPRW 2017]
Free Information in Compressed Videos

Video as a stack of pixels

Representation in compressed video

This representation can help accelerate super-resolution
Transfer the Super-Resolution Results

High Res Frame 1 + Bicubic = High Res Frame 2

Low Res Frame 1 + motion vector = Low Res Frame 2

Super resolution $f_{sr}$

Paste
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 $N$
Challenge 1: Scene Transition

Transfer will NOT work if there is a transition of scenes

Group-of-Picture (GoP) Structure

GoP structure in the compressed video provides video segmentation for free
Challenge 2: Prediction Error

Ground-truth

Non-Adaptive
Artifacts when missing high frequency components of residual

Adaptive
Use lightweight metric to identify occurrence and skip transfer

FAST skips transfer on blocks with large residual
Challenge 3: Blocking Artifacts

**SRCNN**

*No deblocking*

**Overlapped Block Processing**

- No blocking artifacts
- Process each pixel multiple times

**Non-Overlapped Block Processing**

- Process each pixel once
- Blocking artifacts
Challenge 3: Blocking Artifacts

FAST applies the **deblocking filter** to alleviate the blocking effect caused by non-overlapping block division.
Challenge 4: Accumulated Error

When a SR result gets transferred multiple times, the error **accumulates**

FAST estimates the accumulated error as the **accumulated Laplacian of the residual**, and stops the transfer when it exceeds a threshold.
Evaluation: Accelerating SRCNN

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

<table>
<thead>
<tr>
<th>Video</th>
<th>PSNR with 4x acceleration (GOP = 4)</th>
<th>PSNR with 16x acceleration (GOP = 16)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PartyScene</td>
<td>SRCNN: 31.04</td>
<td>SRCNN with FAST: 30.89</td>
</tr>
<tr>
<td></td>
<td>SRCNN with FAST: 31.04</td>
<td>SRCNN with FAST: 30.65</td>
</tr>
<tr>
<td></td>
<td>Bicubic: 29.87</td>
<td>Bicubic: 29.77</td>
</tr>
</tbody>
</table>

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

Bicubic  SRCNN  SRCNN with FAST  Ground-truth
Visual Evaluation

SRCNN

FAST with SRCNN

Bicubic
Summary of FAST

- **Transfer** the SR results guided by **motion vectors**
- **Adaptively** perform the transfer by thresholding on the **residue**, and **accumulated Laplacian**
- **Accelerates** SR algorithms by up to **15x** with **minimal PSNR loss**

[Image of SR algorithm flowchart]

Compressed video → **FAST** → SR 15x faster → Real-time

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

[Zhang et al., CVPRW 2017]
Enable Real-time Navigation on nano drone

Enable energy-efficient navigation for Search and Rescue

[Zhang et al., RSS 2017]

http://navion.mit.edu

In collaboration with Sertac Karaman and Luca Carlone (AeroAstro)
**Localization with Visual Inertial Odometry**

**VIO** determines location/orientation of drone from images and IMU (also used by headset in Augmented Reality and Virtual Reality).

**Legend**
- **KF_j**: Keyframe (j)
- **x**: Drone’s state
- **R**: Orientation
- **P**: Position
- **f_i**: 2D feature (i)
- **L_i**: 3D Landmark (i)

**Navion**: Fully integrated VIO system on-chip consuming < 30mW

**http://navion.mit.edu**
Visual Inertial Odometry Demo

http://navion.mit.edu
Compression to Reduce Energy and Cost

Memory dominates energy and chip area.

Navion Architecture
Processes high dimensional data

http://navion.mit.edu

[ Suleiman et al., VLSI 2018]

Apply various compression techniques to reduce on-chip storage cost by 4.1x. Entire VIO system is fully integrated on chip (20mm²).

http://navion.mit.edu
• Video is perhaps the biggest of the ‘big data’ being collected and transmitted.

• Moving from compressing to understanding pixels at the edge increasingly desirable due to privacy/security and latency constraints. However, energy significantly limited at edge.

• Co-design of algorithms and hardware can enable energy-efficient video coding, computer vision and deep learning such that they can efficiently operate on edge devices such as smartphones and autonomous vehicles/drones.

• Bridging the gap between video coding, computer vision and deep learning plays an important role in overcoming many of the challenges faced by next generation of edge devices.
Research conducted in the MIT Energy-Efficient Multimedia Systems Group would not be possible without the support of the following organizations:

[Logos of various organizations]

For updates on our research Follow @eems_mit
References

• Energy-Efficient Video Coding

• Compression Inspired by Computer Vision
• Energy-Efficient Deep Learning


References

- **Efficient Computer Vision using Compression**
  
  
  