Mobile and Sensor Systems

Lecture 6: Mobile Sensing Energy and Systems Considerations

Prof Cecilia Mascolo

[thanks to Prof Nic Lane for material]

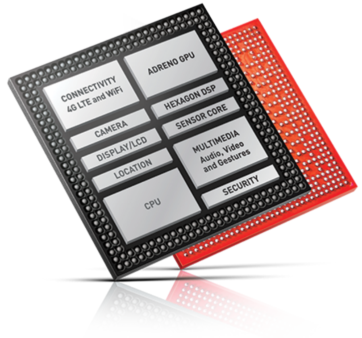
UNIVERSITY OF CAMBRIDGE

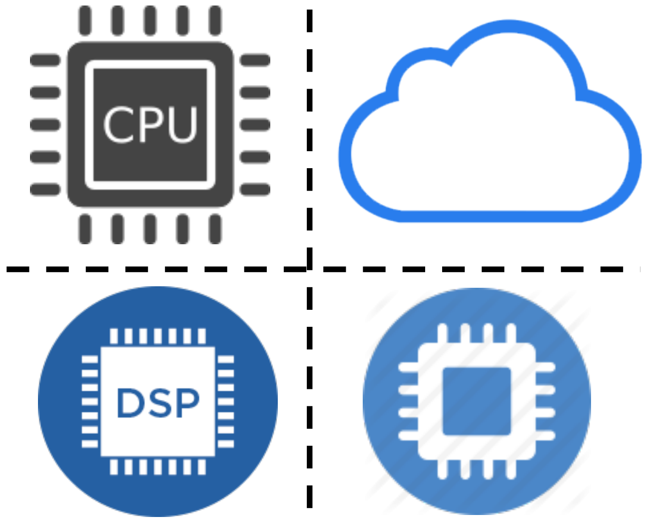
In this Lecture

- We will study approaches to preserve energy in mobile sensing systems
- We will look at aspects of local versus cloud computation
- We will look at how machine learning can be used on mobile/wearable/powered devices

Devices have various processors

- Locally …and remotely (cloud)
- Trading these off vs power is important





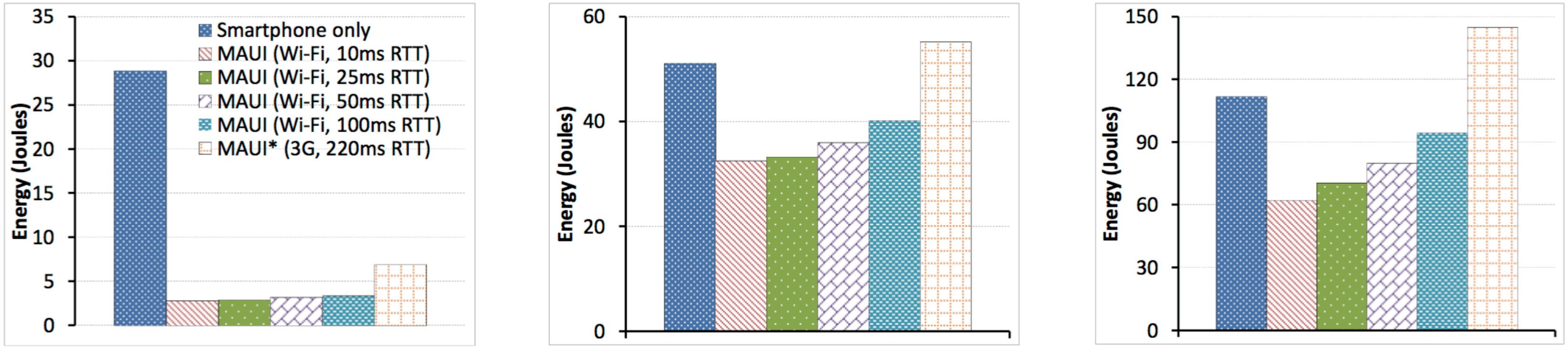
MAUI

- MAUI is a mobile device framework which profiles code components in terms of energy to decide if to run them locally or remotely (considering latency requirements).
  - Costs related to the transfer of code/data
  - Programming framework
  - Dynamic decisions based on network constraints
  - CPU only

UNIVERSITY OF CAMBRIDGE

[Cuervo et al 2010]

MAUI Offloading



ONE RUN FACE RECOGNITION | 400 FRAMES of VIDEO GAME | 30 MOVE CHESS GAME

Figure 9: A comparison of MAUI's energy consumption. *We compare the energy consumption of running three applications standalone on the smartphone versus using MAUI for remote execution to servers that are successively further away (the RTT is listed for each case). The graph on the left shows one run of the face recognition application; the graph in the middle shows running the video game for 400 frames; the graph on the right shows running the chess game for 30 moves. MAUI* is a slight modification to MAUI to bypass the optimizer and to always offload code. Without this modification, MAUI would have not performed code offload in the case of the video game and chess because offload ends up hurting energy performance.*

UNIVERSITY OF CAMBRIDGE

MAUI would not perform offloading with 3G…

Continuous Audio Sensing Applications

Emotion recognition

Speaker count

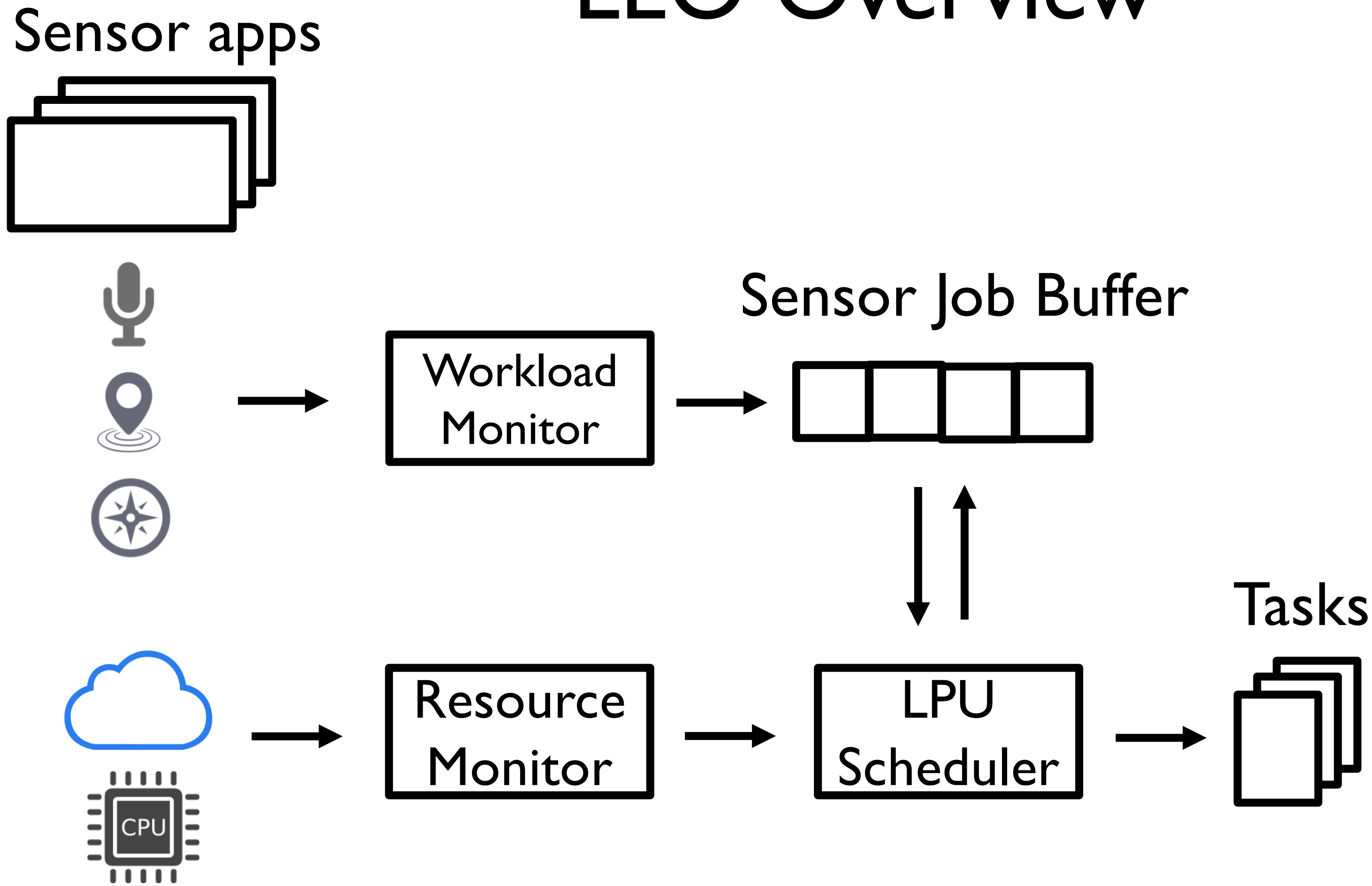
Speaker identification

Gender estimation

Ambient sound detection

UNIVERSITY OF CAMBRIDGE

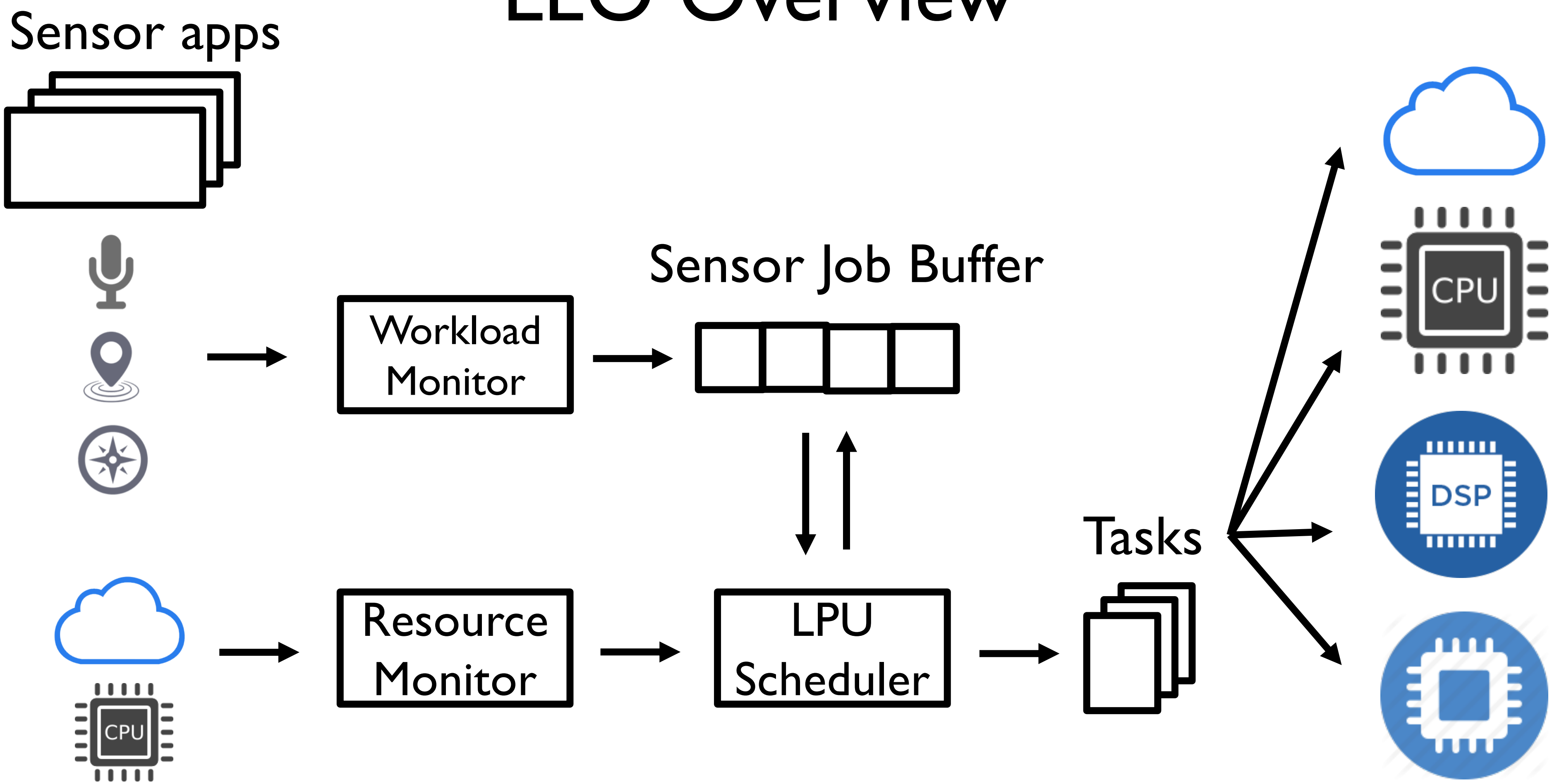
LEO Overview



UNIVERSITY OF CAMBRIDGE

[Georgiev et al 2016]

LEO Overview



UNIVERSITY OF CAMBRIDGE

Low overhead

- uses heuristics (fast runtime)
- runs on the LPU (low energy)

10 app workload

~100 ms

<0.5%

vs.

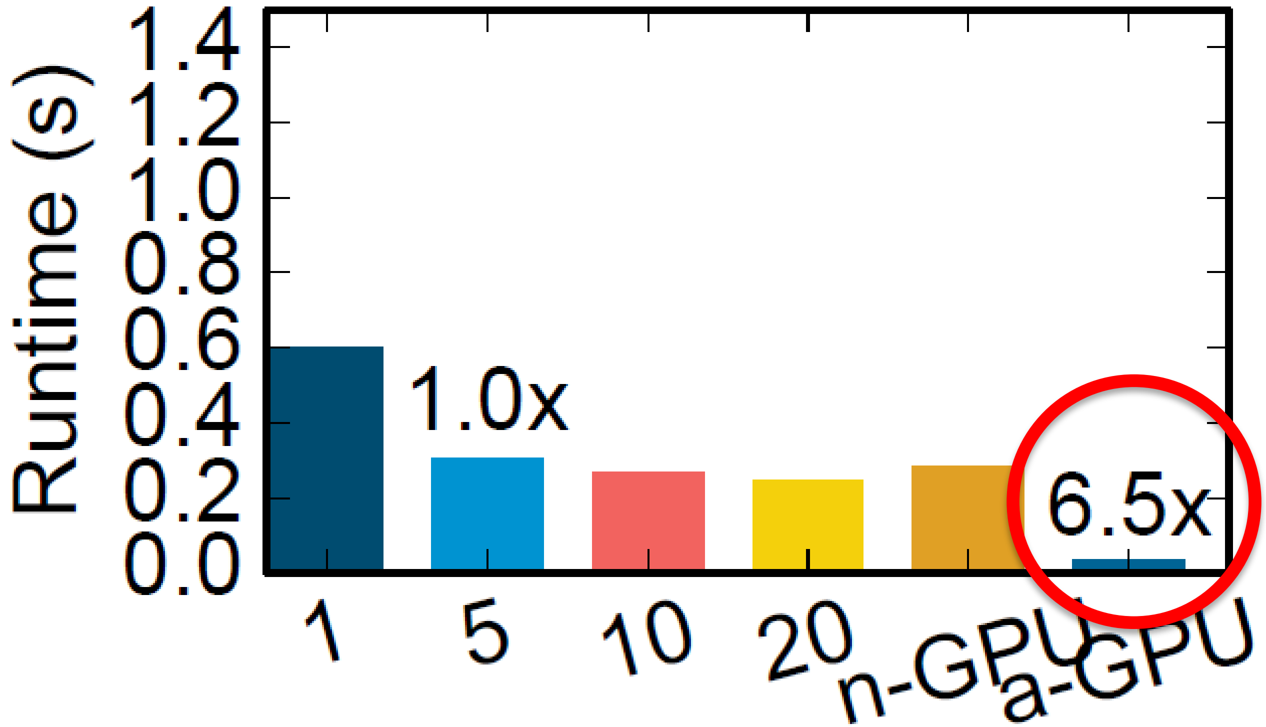
~3.5%

scheduling in cloud (next best alternative)

UNIVERSITY OF CAMBRIDGE

Optimized GPU is Efficient

Optimized GPU is >6x faster than cloud



nGPU= naïve GPU usage
aGPU=optimized use of GPU

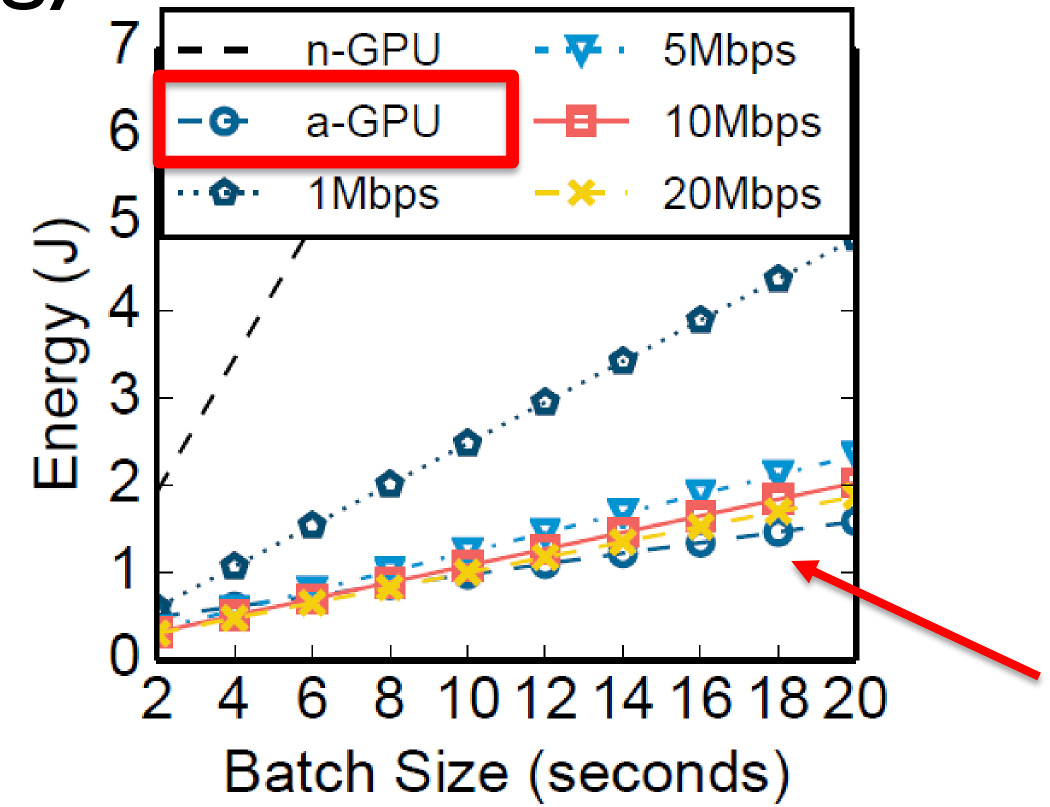
UNIVERSITY OF CAMBRIDGE

Keyword Spotting application

[Georgiev et al 2017]

Optimized GPU is Efficient

Optimized GPU with batching outperforms cloud energy-wise



UNIVERSITY OF CAMBRIDGE Keyword Spotting classification

Machine Learning for Mobiles

- We have seen in the previous lecture that sensor data can be analysed offline with machine learning
- This allows rich applications and understanding of user behaviour

UNIVERSITY OF CAMBRIDGE

Could We Perform Inference On Device?

- Machine Learning models are often built with little consideration of system resources…
- AlphaGo: 1920 CPUs and 280 GPUs, $3000 in electricity per game…mhhhh.

UNIVERSITY OF CAMBRIDGE

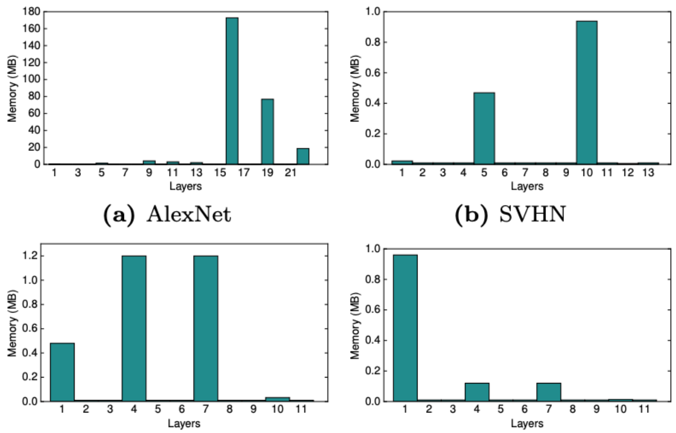
Why Perform Inference On Device

- Performing Inference on device would allow for data not to flow out of devices…(privacy)
- Limit how much bandwidth is used to send data out (at the cost of processing usage for inference)…
- Applications:
  – Video applications on image sensors for traffic characterization (comms costs reduced)
  – Drone/robot navigation local processing for low latency and security
- Thinking of trade offs is essential.

UNIVERSITY OF CAMBRIDGE

Resource requirements

- Tradeoffs:
  - Accuracy per £.
  - Memory / latency.
- Considerations:
  - Memory.
  - Energy.
  - Latency.



(a) AlexNet (b) SVHN (c) DeepEar *(emotion only)* (d) Deep KWS

Figure 6: Memory requirements during inference on a per layer basis; only the layers of the model being operated upon are left in memory to lower requirements. (Execution on Snapdragon CPU).

	Layer type	Tunable parameters	Time (%)
1	Convolution	34, 944	37.20
2	Non-linear	-	0.05
3	Normalization	-	0.12
4	Pooling	-	0.15
5	Convolution	307,456	2.05
6	Non-linear	-	0.05
7	Normalization	-	0.21
8	Pooling	-	1.11
9	Convolution	885,120	30.89
10	Non-linear	-	0.46
11	Convolution	663,936	13.56
12	Non-linear	-	0.08
13	Convolution	442,624	7.45
14	Non-linear	-	0.38
15	Pooling	-	0.74
16	Feed-forward	37,752,832	0.49
17	Non-linear	-	0.15
18	Dropout	-	0.06
19	Feed-forward	16,781,312	0.19
20	Non-linear	-	0.14
21	Dropout	-	0.07
22	Feed-forward	4,097,000	4.34
22	Softmax	-	0.06

Table 5: Layer-by-layer runtime performance of AlexNet.

	Tegra		Snapdragon		Edison
	CPU	GPU	CPU	DSP	CPU
Deep KWS	0.8	1.1	7.1	7.0	63.1
DeepEar	6.7	3.2	71.2	379.2	109.0
AlexNet	600.2	49.1	159,383.1	-	283,038.6
SVHN	15.1	2.8	1,616.5	-	3,562.3

Table 3: Execution Time (msec.)

	Tegra		Snapdragon		Edison
	CPU	GPU	CPU	DSP	CPU
Deep KWS	14.34	9.16	5.00	134.41	27.78
DeepEar	21.74	22.02	16.93	342.47	30.99
AlexNet	3.49	10.36	3.80	-	13.88
SVHN	13.98	14.81	3.97	-	15.38

Table 4: Battery Life Estimate (hrs.)

UNIVERSITY OF CAMBRIDGE

[Lane et al 2015]

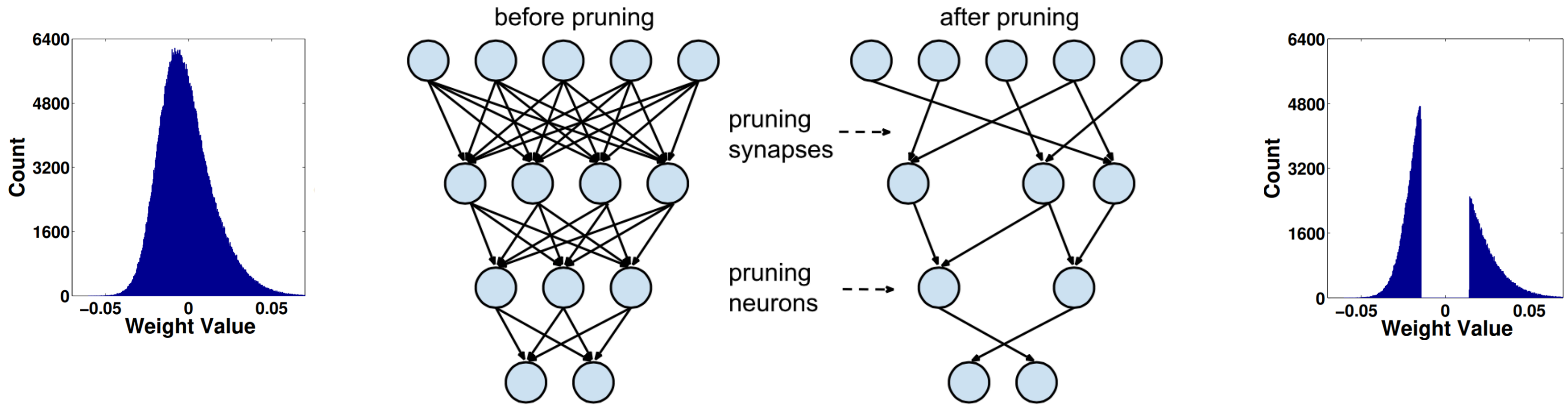
How to improve resource tradeoffs?

- **General methods**
  - Pruning – removing excess parameters.
  - Quantization – decreasing parameter precision.
- Fully connected layers
  - Weight factorization – low rank approximation.
- Convolutional layers
  - Convolution separation – low rank approximation.
- Other paths to resource efficiency.

[Sze et al]

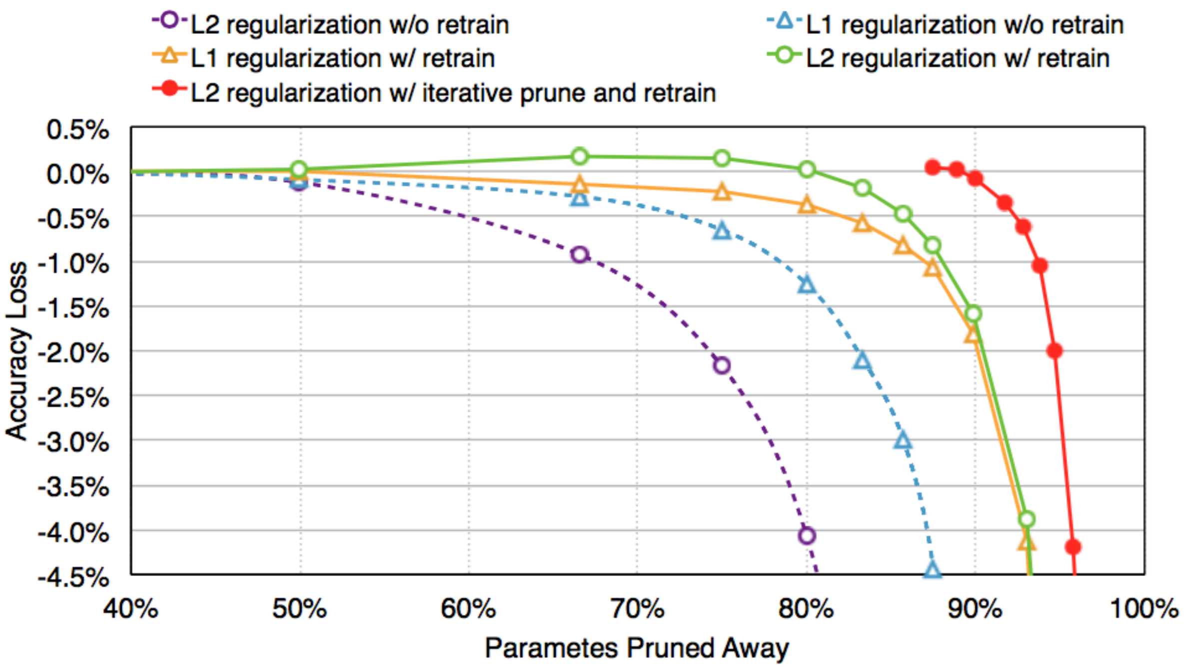
Pruning

- Pruning removes, sets to zero, weights in NN base on a pre-defined heuristic. Magnitude (abs. value) is the most used criterion. It performs as well as a random criterion.



Pruning

- Pruning followed by re-training performs very well and doing it iteratively is best…



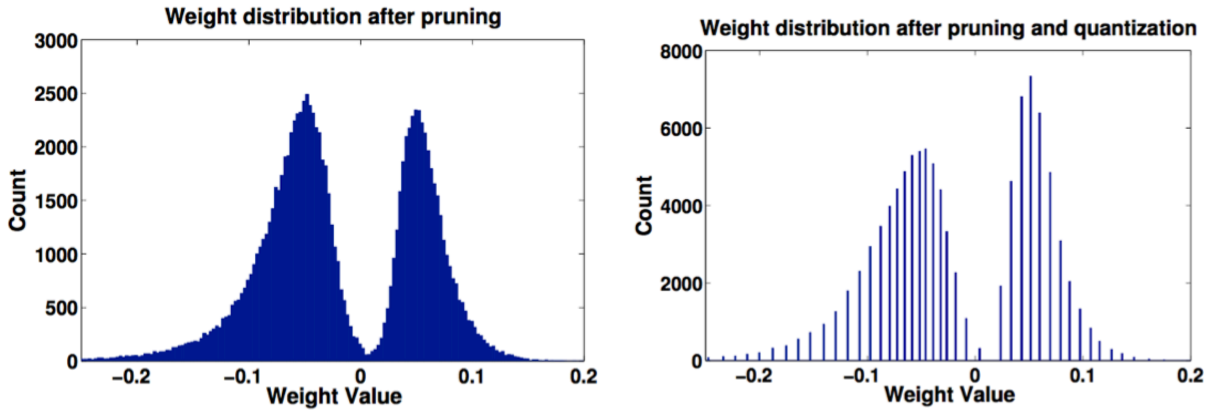
UNIVERSITY OF CAMBRIDGE

[Gupta et al 2015]

Quantization

Is a **lower precision representation** of trained parameters.

- Post-training quantization.
  - Usually applied after pruning.
  - Varied options:
    - K-means
    - Hashing
    - Huffman Coding
    - Weight Sharing

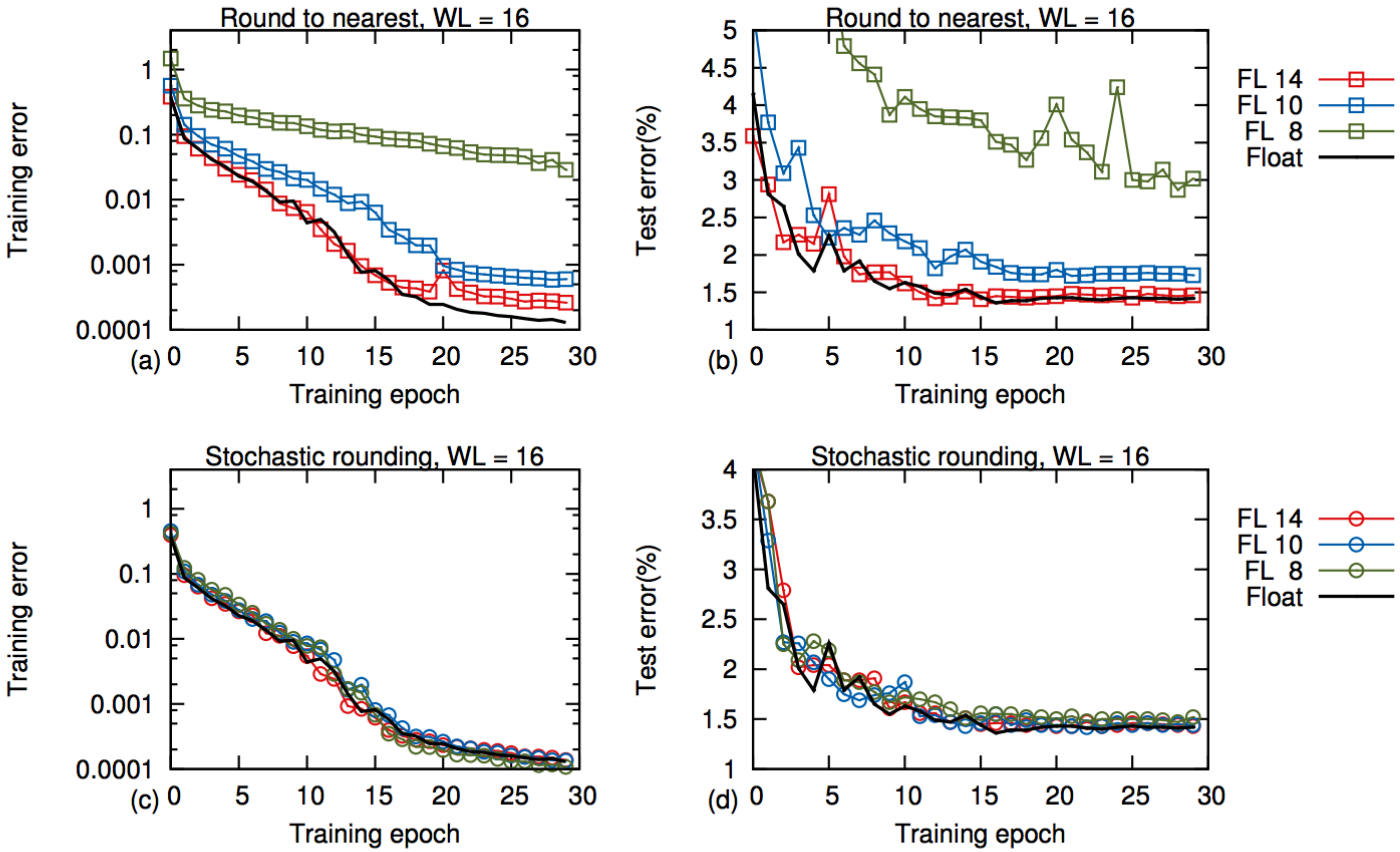


- Training quantized models.
  - Networks are quantized at each step in the training process at the forward pass (but leaving the back propagation parameters in higher precision): this limits accuracy loss.

UNIVERSITY OF CAMBRIDGE

Training quantized models

- At train time quantization is achieved by:
  - Truncation
  - (Stochastic) Rounding



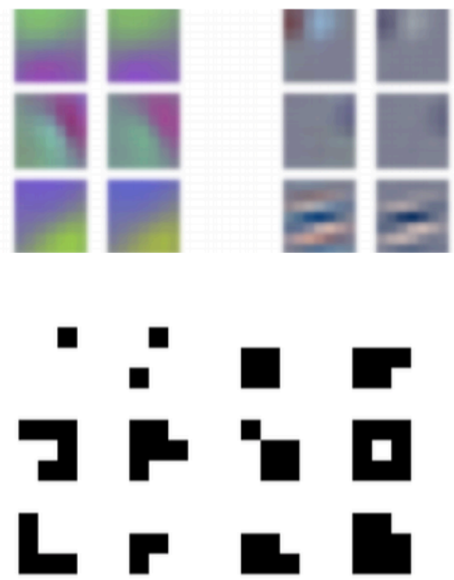
- MNIST dataset

fully connected DNNs

Binary Weight Networks (BWNs)

- Weights set to $\{-\alpha, +\alpha\}$ set based on original layer values.
- Activations and last layer are 32-bit.

	Network Variations	Operations used in Convolution	Memory Saving (Inference)	Computation Saving (Inference)	Accuracy on ImageNet (AlexNet)
Standard Convolution	Real-Value Inputs: 0.11 -0.21 ... -0.34 / -0.25 0.61 ... 0.52; Real-Value Weights: 0.12 -1.2 ... 0.41 / -0.2 0.5 ... 0.68	+ , − , ×	1x	1x	%56.7
Binary Weight	Real-Value Inputs: 0.11 -0.21 ... -0.34 / -0.25 0.61 ... 0.52; Binary Weights: 1 -1 ... 1 / -1 1 ... 1	+ , −	~32x	~2x	%56.8
BinaryWeight Binary Input **(XNOR-Net)**	Binary Inputs: 1 -1 ... -1 / -1 1 ... 1; Binary Weights: 1 -1 ... 1 / -1 1 ... 1	XNOR , bitcount	~32x	~58x	%44.2



UNIVERSITY OF CAMBRIDGE

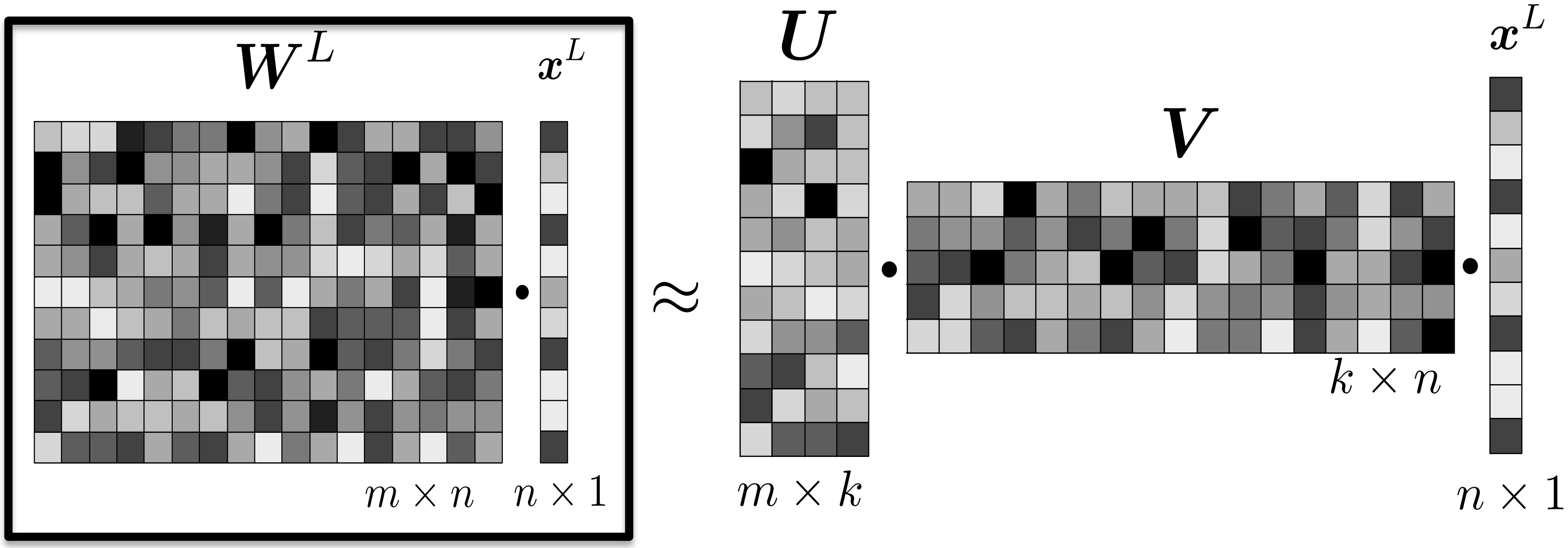
[Rastegari et al 2016]

How to improve resource tradeoffs?

- General methods
  - Pruning – removing excess parameters.
  - Quantization – decreasing parameter precision.
- **Fully connected layers**
  - Weight factorization – low rank approximation.
- Convolutional layers
  - Convolution separation – low rank approximation.
- Other paths to resource efficiency.

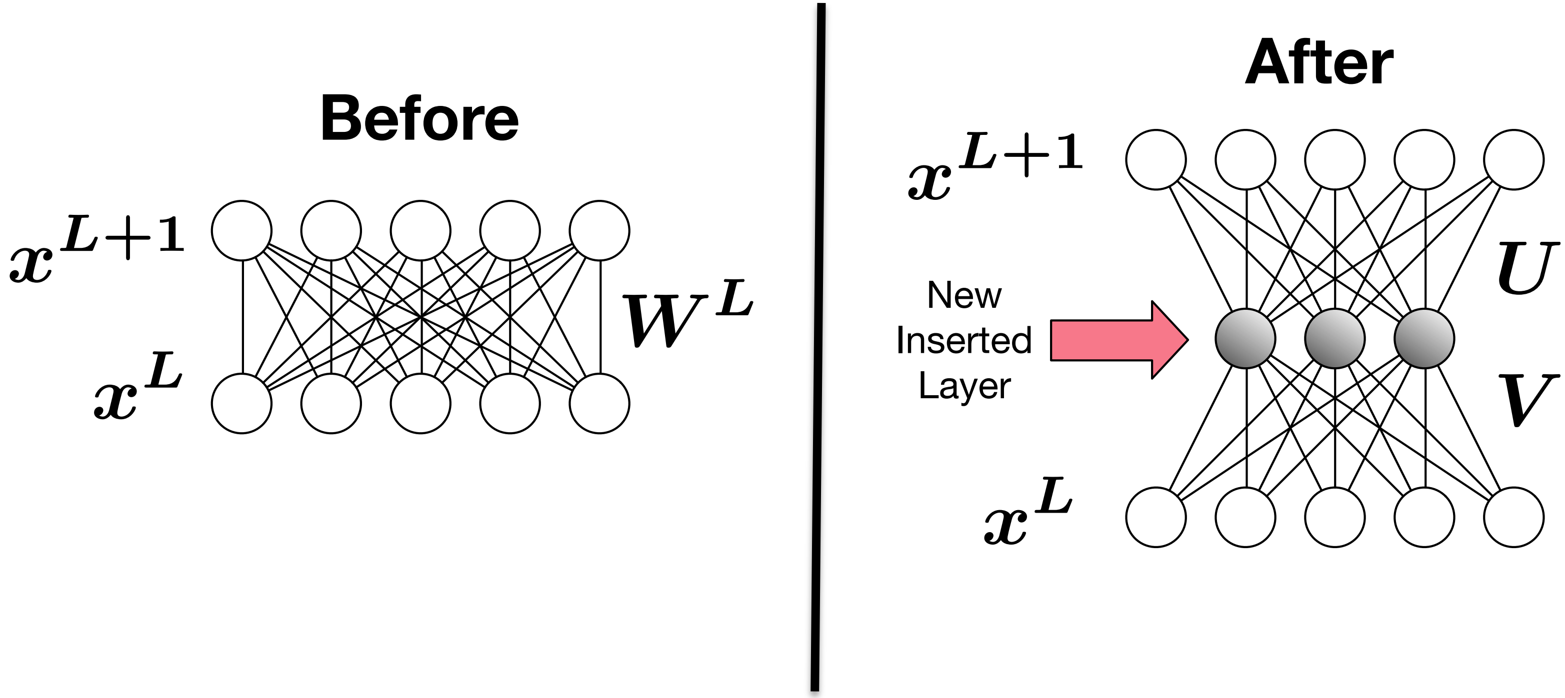
UNIVERSITY OF CAMBRIDGE

SVD weight approximation



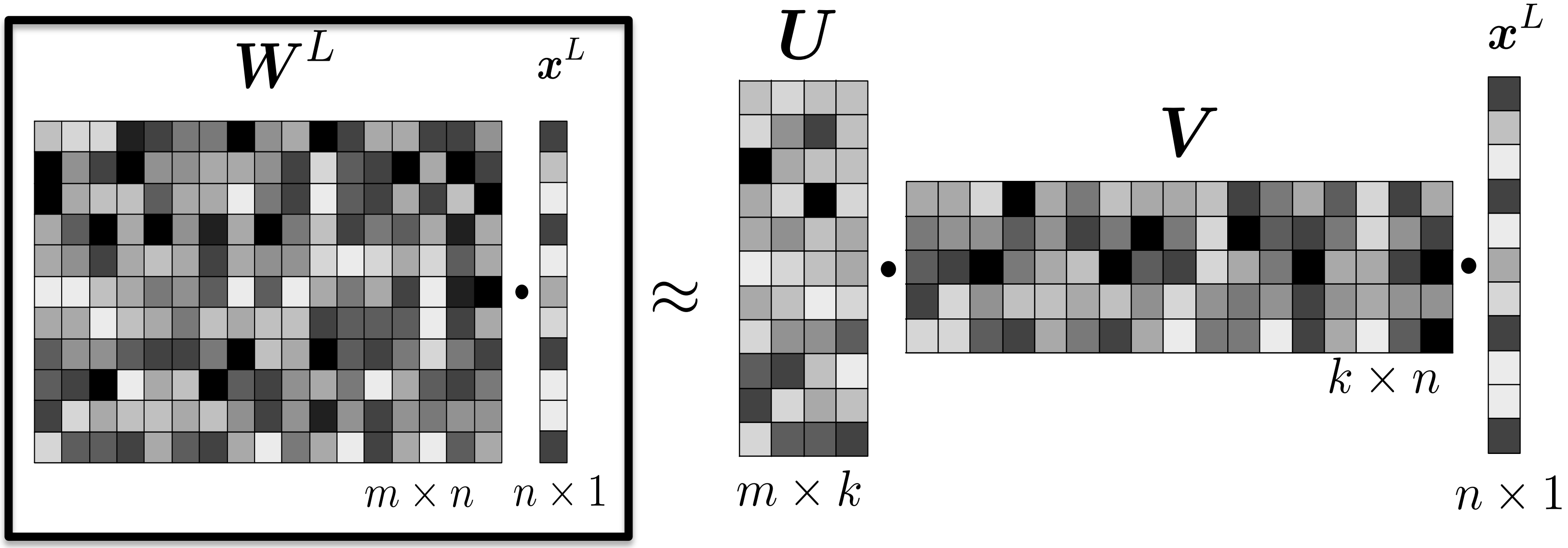
[S. Bhattacharya et al 2016]

SVD weight approximation



UNIVERSITY OF CAMBRIDGE

SVD weight approximation



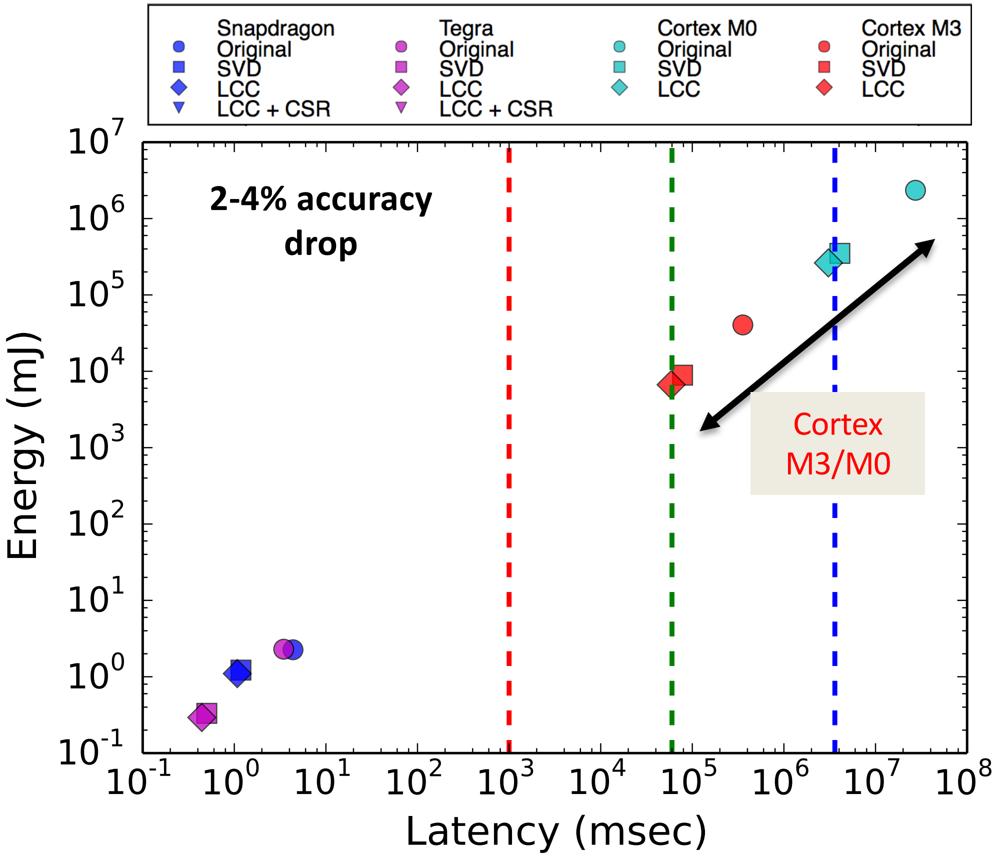
Total Operations: m x n x 1

Total Operations: m x k x 1 + k x n x 1

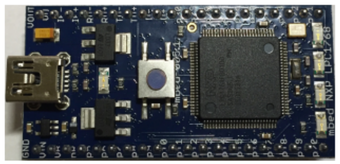
Memory & compute savings if: $k < \dfrac{m \times n}{m+n}$

SVD weight approximation

Ambient scene analysis and speaker detection tasks.

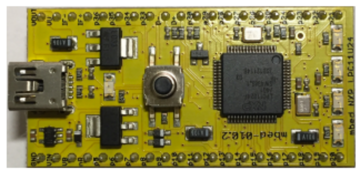


32 KB



ARM Cortex M3

16 KB



ARM Cortex M0

UNIVERSITY OF CAMBRIDGE

How to improve resource tradeoffs?

- General methods
  - Pruning – removing excess parameters.
  - Quantization – decreasing parameter precision.
- Fully connected layers
  - Weight factorization – low rank approximation.
- **Convolutional layers**
  - Convolution separation – low rank approximation.
- Other paths to resource efficiency.

UNIVERSITY OF CAMBRIDGE

Convolution separation

Find an approximation of the kernels that is:

- more computationally efficient,
- faithful to original kernel.

$$\mathcal{K}_n \in \mathbb{R}^{d \times d \times C}$$

$$\mathcal{K}_n \approx \hat{\mathcal{K}}_n$$

$$\hat{\mathcal{K}}_n^c = \sum_{k=1}^{K} \mathcal{H}_n^k (\mathcal{V}_k^c)^T$$

Horizontal Filter

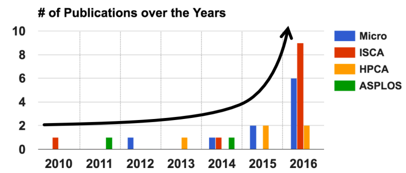
Vertical Filter

$$\begin{bmatrix} 1 & 3 & 1 \\ 2 & 6 & 2 \\ 3 & 9 & 3 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \text{ And } \begin{bmatrix} 1 & 3 & 1 \end{bmatrix}$$

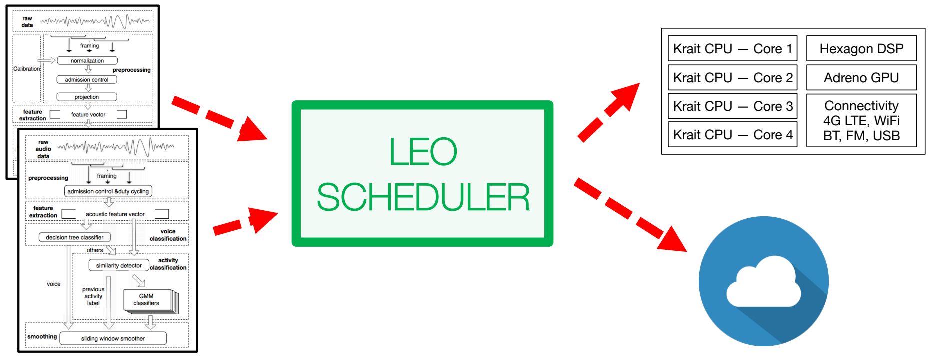
Computational gain when: $K < \dfrac{dCN}{C + N}$

Other parts towards efficiency

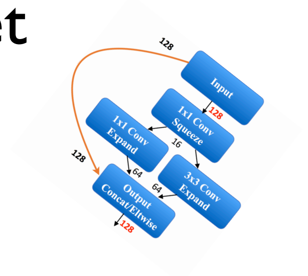
- Commodity processors and accelerators
  - The elephant in the room in this discussion.

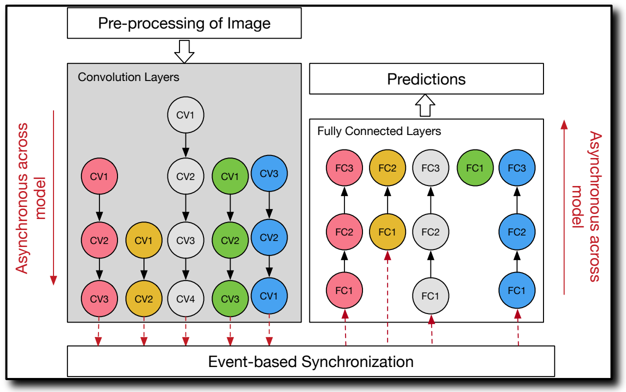


- System-level Solutions
  - Including runtime.
- Cross Models Optimization



- Low-Resource Architectures
  - MobileNet





UNIVERSITY OF CAMBRIDGE

Summary

- We have looked at on device computation vs offloading to cloud/edge
- We have studied how local resources and cloud offloading have an impact on energy efficiency and could be used to improve it.
- We have explored the trade offs of accuracy and energy and the techniques which can improve machine learning on device.

UNIVERSITY OF CAMBRIDGE

References

- E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, P. Bahl. 2010. MAUI: making smartphones last longer with code offload. In Proceedings of MobiSys '10.
- P. Georgiev, N. Lane, K. Rachuri, C. Mascolo. 2016. LEO: scheduling sensor inference algorithms across heterogeneous mobile processors and network resources. In *Proceedings of* MobiCom '16.
- P.. Georgiev, N. Lane, C. Mascolo, D. Chu. Accelerating Mobile Audio Sensing Algorithms through On-Chip GPU Offloading. In Proceedings of 15th ACM International Conference on Mobile Systems, Applications and Services (Mobisys 2017). Niagara Falls, NY. USA. June 2017.
- N. Lane, S. Bhattacharya, P. Georgiev, C. Forlivesi, F. Kawsar. 2015. An Early Resource Characterization of Deep Learning on Wearables, Smartphones and Internet-of-Things Devices. In Workshop on Internet of Things towards Applications 2015.
- S. Bhattacharya, N. Lane. 2016. Sparsification and Separation of Deep Learning Layers for Constrained Resource Inference on Wearables. In Procs of the ACM SenSys '16.
- S. Gupta, A. Agrawal, K. Gopalakrishnan, P. Narayanan. 2015. Deep learning with limited numerical precision. In Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37 (ICML'15),.
- Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. XNOR-Net: Imagenet Classification Using Binary Convolutional Neural Networks. ECCV 2016.
- V. Sze and Y/ Chen and T. Yang, J. Emer. Efficient Processing of Deep Neural Networks: A Tutorial and Survey. https://arxiv.org/abs/1703.09039

UNIVERSITY OF CAMBRIDGE