Memory-Optimal Direct Convolutions for Maximizing Classification Accuracy in Embedded Devices

Albert Gural[1], Boris Murmann[1]

[1]Stanford University

The 36th International Conference on Machine Learning
Long Beach, California
June 11, 2019

Stanford University

Introduction

- Embedded devices are increasingly targets of machine learning for IoT
  - Microsoft EdgeML
    - Bonsai [1]: decision tree achieves 94.38% on MNIST-2 in 2KB
    - ProtoNN [2]: nearest neighbors achieves 93.25% on MNIST-2 in 2KB
    - FastGRNN [3]: RNN achieves 98.20% on MNIST in 6KB
  - Google TensorFlow Lite for MCUs [4]

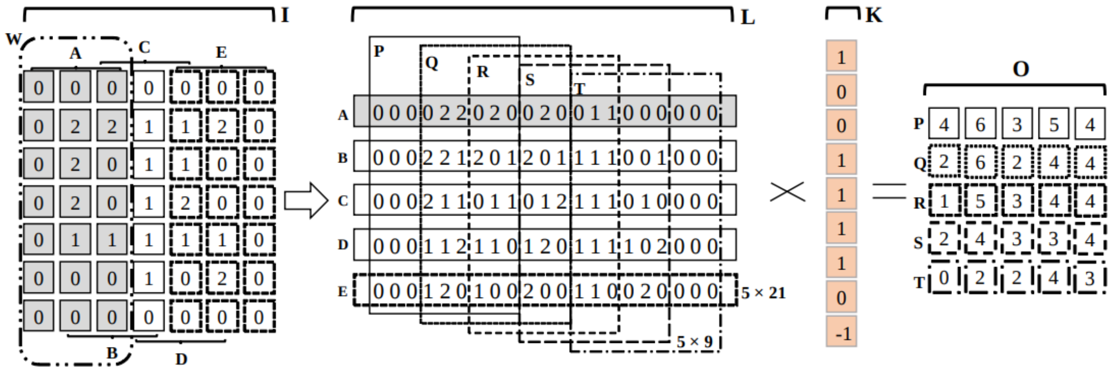
- Hard memory constraints make deep learning difficult
  - "Bonsai is not compared to deep convolutional neural networks as they have not yet been demonstrated to fit on such tiny IoT devices" [1]

- But CNNs typically have SOTA performance for image classification tasks
  - Can we do better with CNNs?
  - Goal: MNIST classifier in 2KB

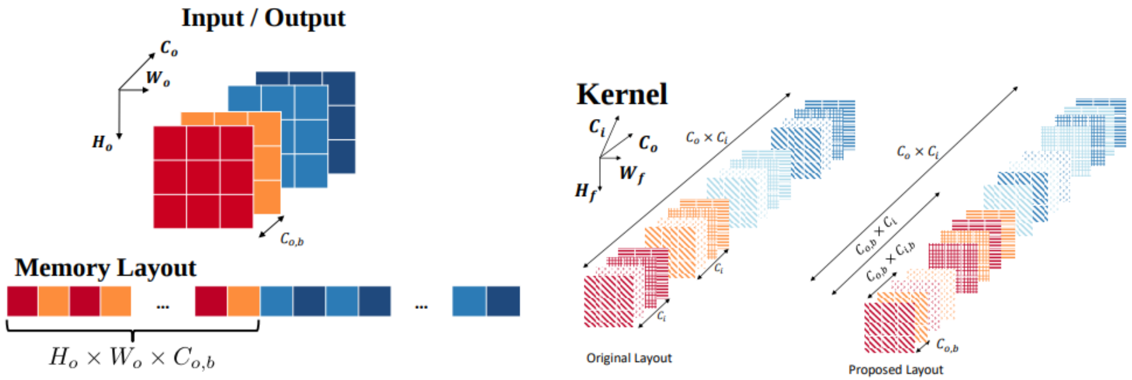
Introduction

- Deep CNN implementation research typically focused on speed
  - FFT, Winograd, *gemm*

- Minimal research prioritizing memory reduction
  - Memory-Efficient Convolution [5] improves memory use of *gemm* methods, but still has overhead
  - Zero-Memory Overhead [6] performs direct convolutions for zero overhead beyond input/output activation storage



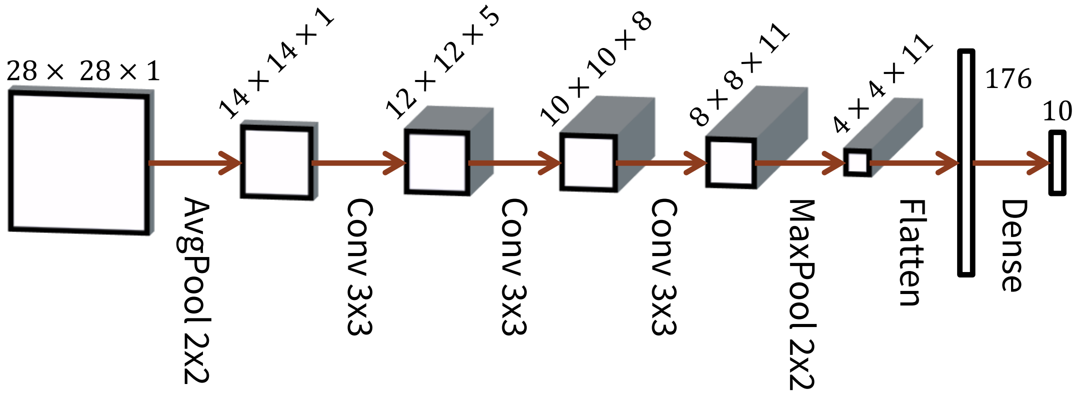
Memory-Efficient Convolution [5]

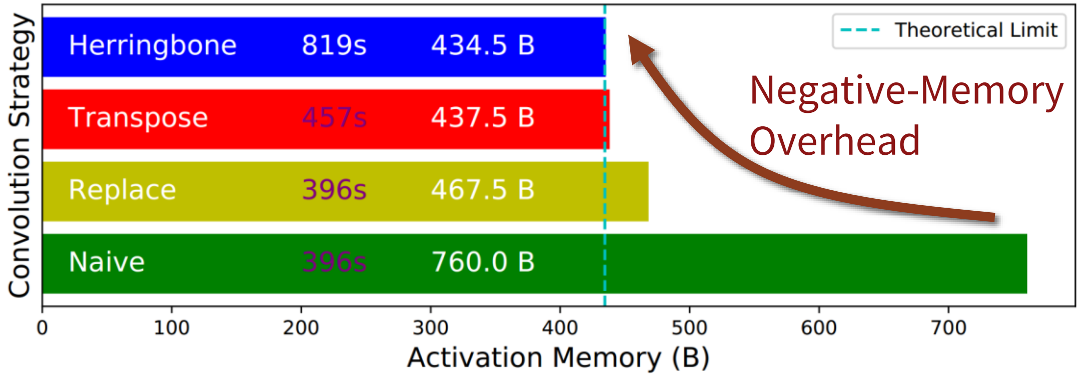


Zero-Memory Overhead [6]

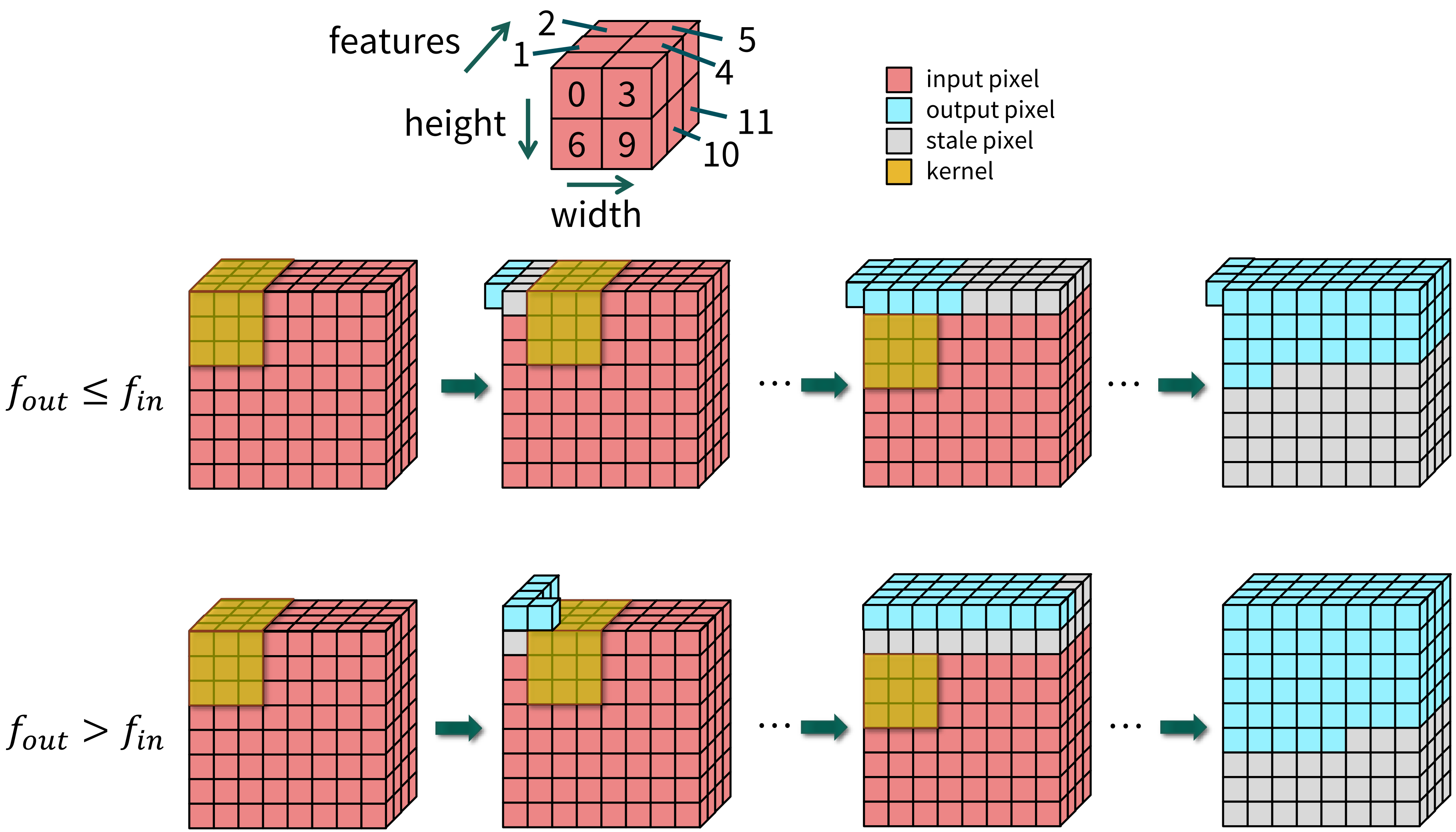
Introduction

- Deep CNN implementation research typically focused on throughput
  - FFT, Winograd, *gemm*
- Minimal research prioritizing memory reduction
  - Memory-Efficient Convolution [5] improves memory use of *gemm* methods, but still has overhead
  - Zero-Memory Overhead [6] performs direct convolutions for zero overhead beyond input/output activation storage
    - **Can do even better by replacing input activations while computing output activations**

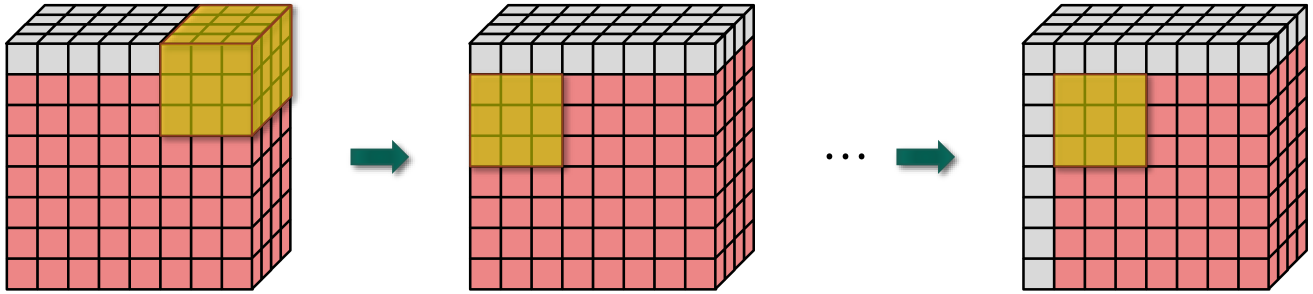




Replace Method



Herringbone Method

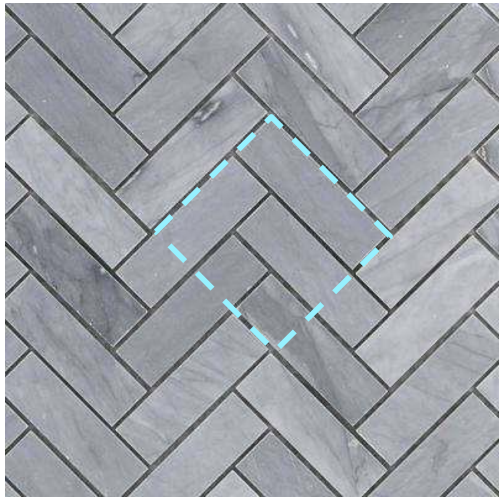


25 cost; 20 free — 30 cost; 32 free — … — 55 cost; 60 free

Order of Convolutions

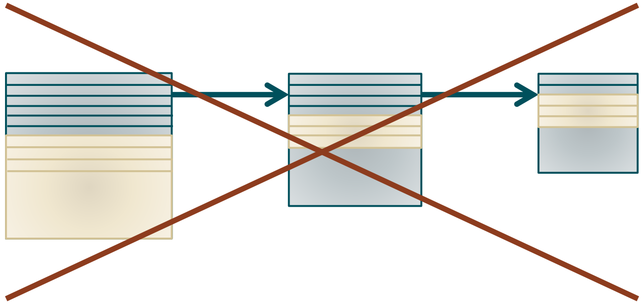
0	1	2	3	4	5	6	7
8	15	16	17	18	19	20	21
9	22	28	29	30	31	32	33
10	23	34	39	40	41	42	43
11	24	35	44	48	49	50	51
12	25	36	45	52	55	56	57
13	26	37	46	53	58	60	61
14	27	38	47	54	59	62	63

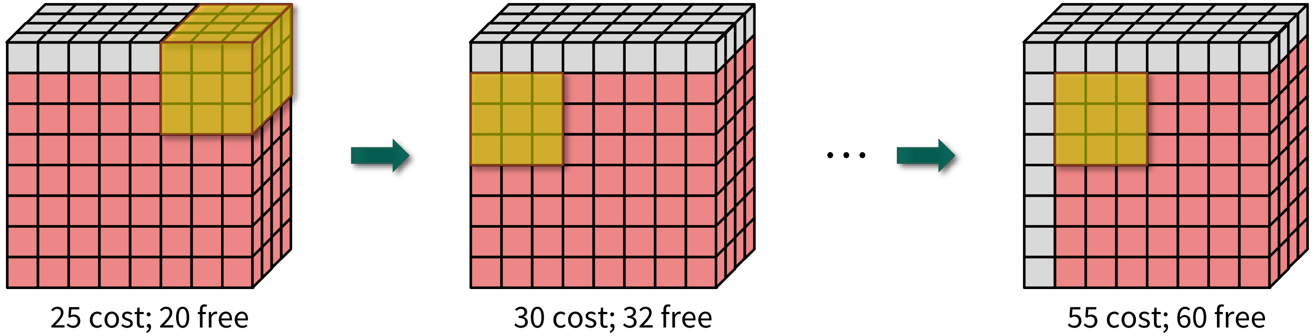
Herringbone tile



Herringbone Method

In paper, we demonstrate optimality for lossless, per-layer, direct convolutions

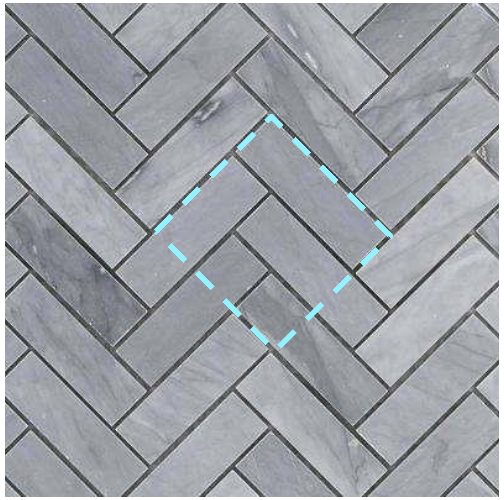




Order of Convolutions

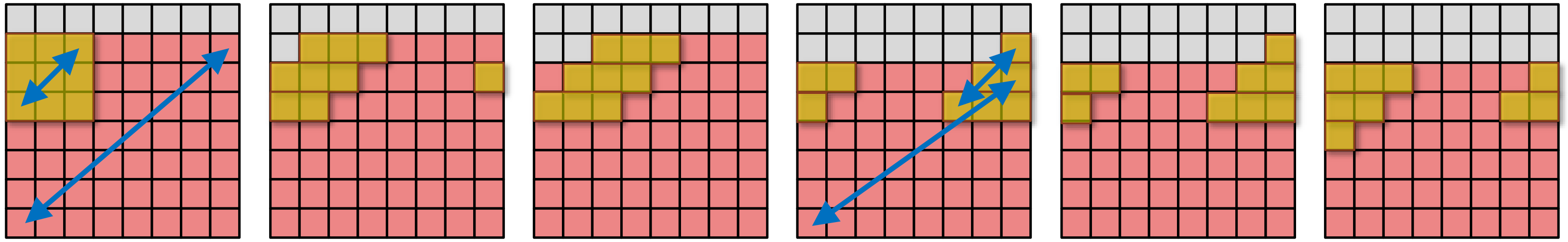
0	1	2	3	4	5	6	7
8	15	16	17	18	19	20	21
9	22	28	29	30	31	32	33
10	23	34	39	40	41	42	43
11	24	35	44	48	49	50	51
12	25	36	45	52	55	56	57
13	26	37	46	53	58	60	61
14	27	38	47	54	59	62	63

Herringbone tile



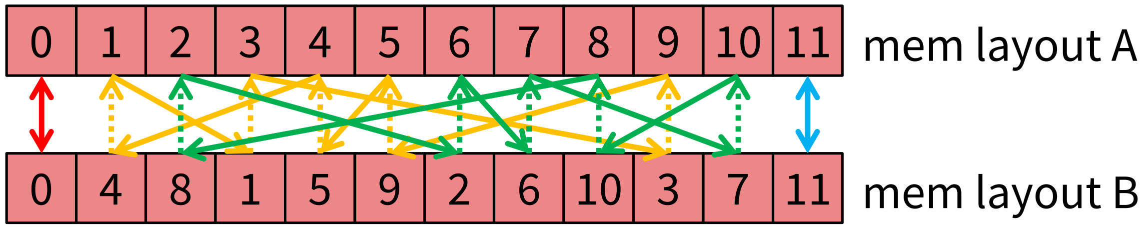
Transpose Implementation

Transpose method: process a row, transpose, process a row, transpose, …



0	1	2	3
4	5	6	7
8	9	10	11

Successor: $j = (i \bmod H) \cdot W + \lfloor i/H \rfloor$



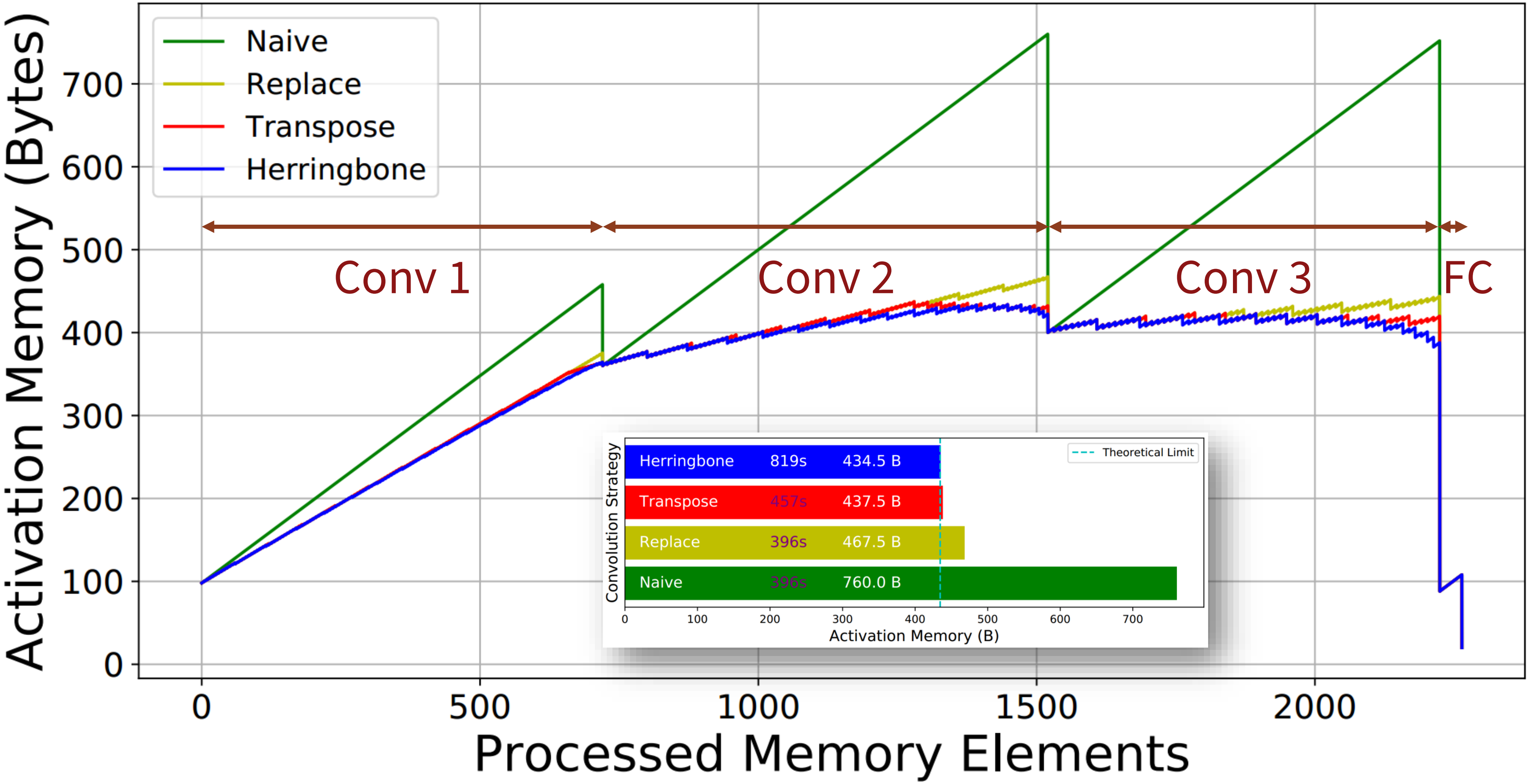
0	4	8
1	5	9
2	6	10
3	7	11

For each start:

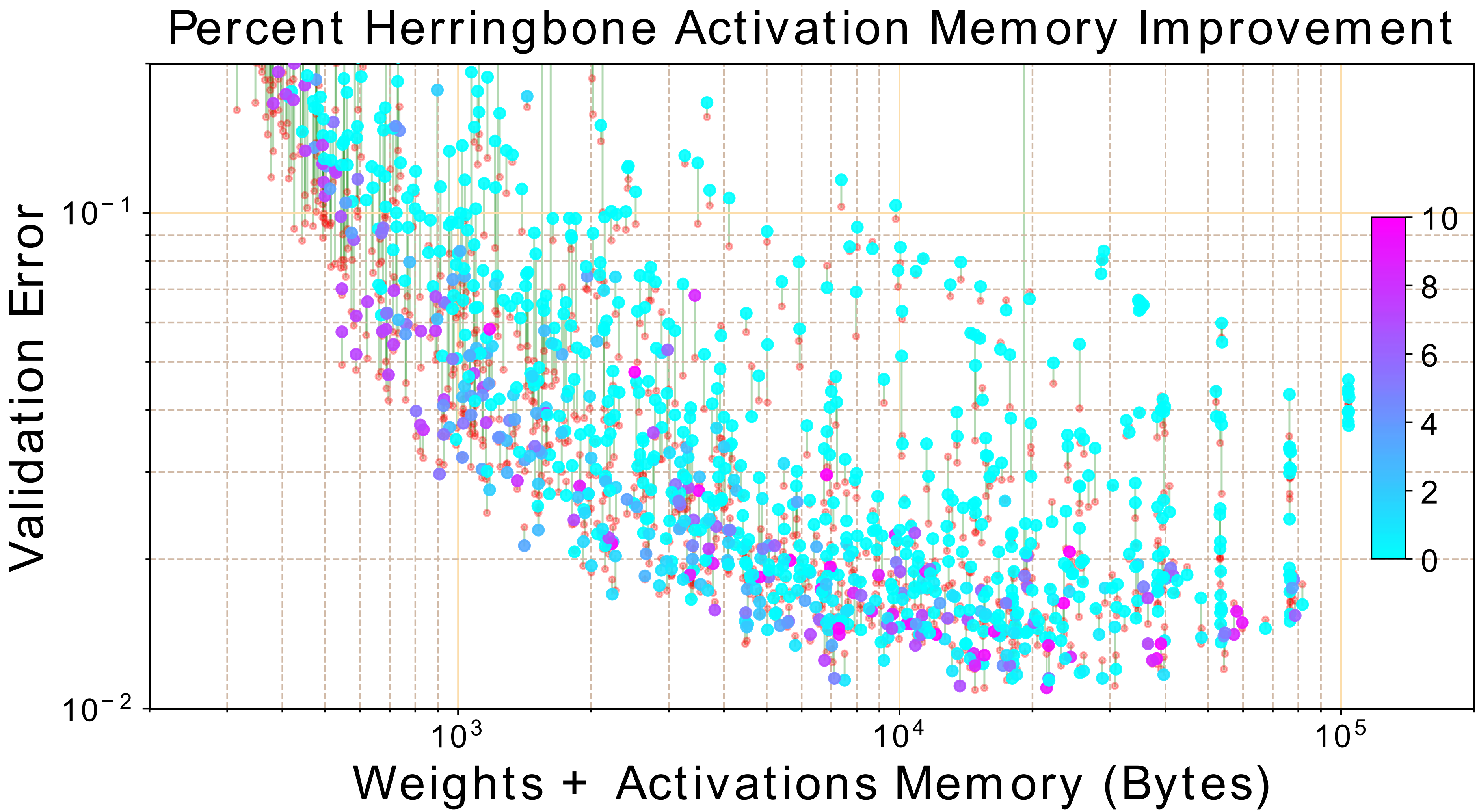
Check if start > any other element in its cycle

If not, rotate elements in the cycle

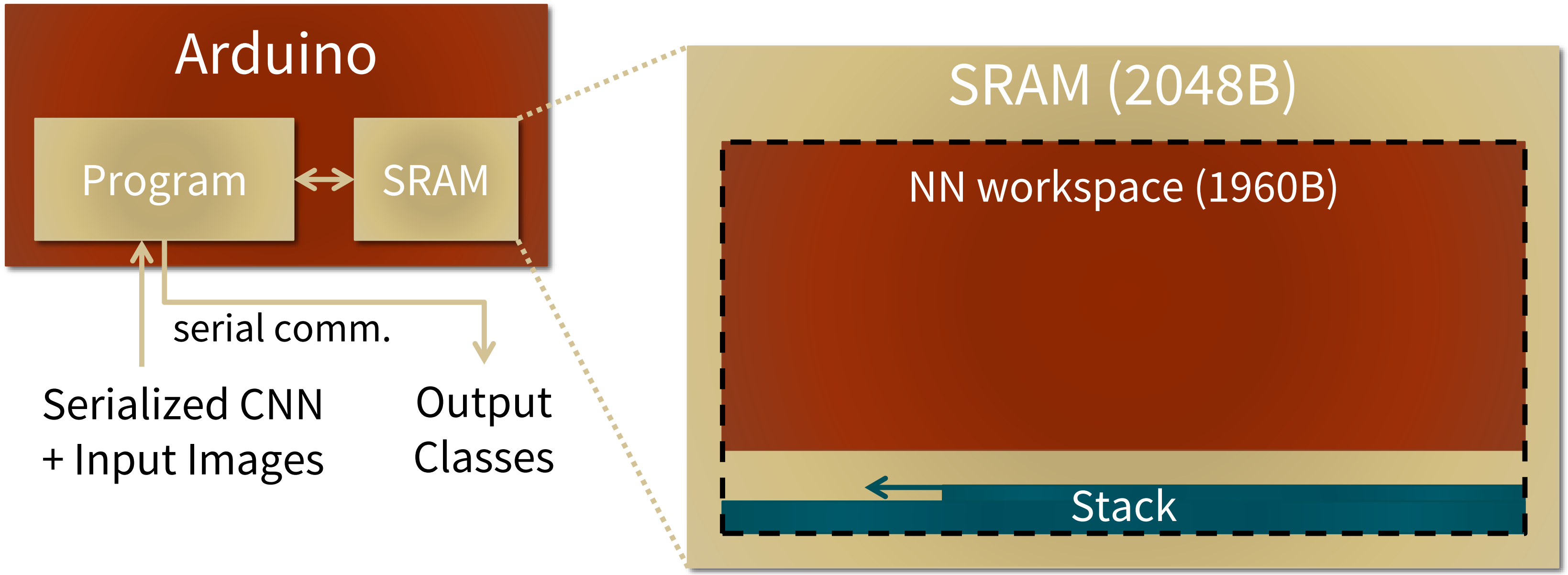
Convolution Strategy Comparison



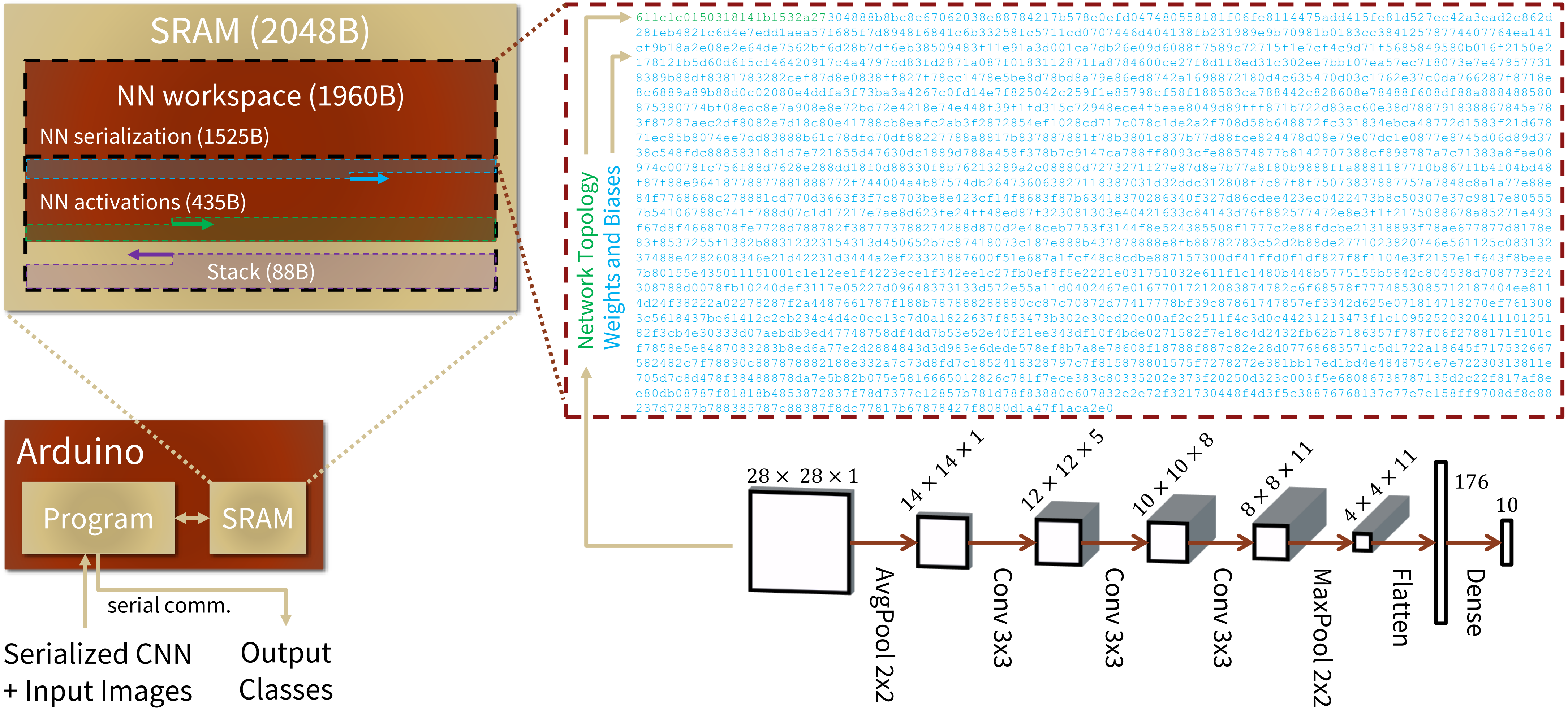
Applicability



Case Study


Arduino
Program
SRAM
serial comm.
Serialized CNN
+ Input Images
Output
Classes
SRAM (2048B)
NN workspace (1960B)
Stack

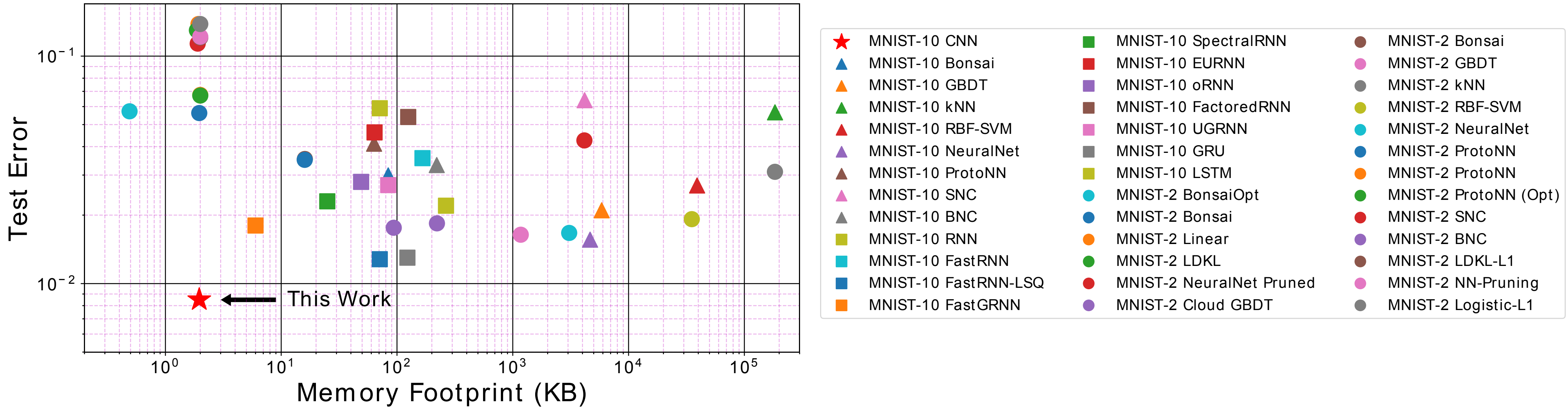
Case Study



SRAM (2048B)
NN workspace (1960B)
NN serialization (1525B)
NN activations (435B)
Stack (88B)
Arduino
Program
SRAM
serial comm.
Serialized CNN
+ Input Images
Output
Classes
Network Topology
Weights and Biases
28 × 28 × 1
14 × 14 × 1
12 × 12 × 5
10 × 10 × 8
8 × 8 × 11
4 × 4 × 11
176
10
AvgPool 2x2
Conv 3x3
Conv 3x3
Conv 3x3
MaxPool 2x2
Flatten
Dense

Results

- Fits in 2KB SRAM
  - Network Topology
  - Weights and Biases
  - Intermediate Activations
- Achieves 99.15% Test Accuracy on MNIST



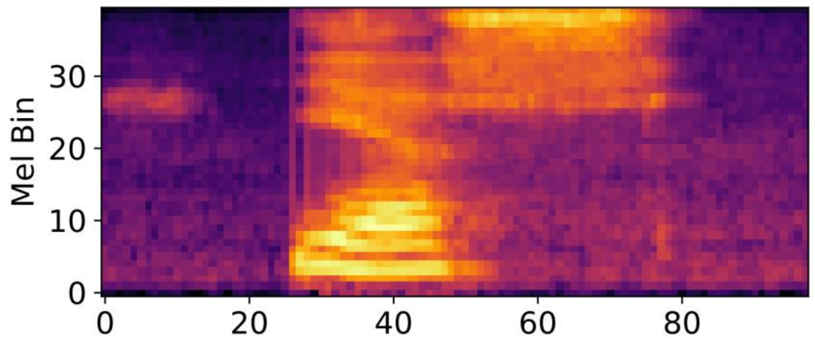
Comparison to MNIST-2 and MNIST-10 results from [1,2,3]

Summary

- Applicability
  - Replace strategy applies to any CNN
  - Herringbone/Transpose strategies apply to many 2D classification CNNs

- Use Scenario
  - Tiny MCUs with negligible caching
  - Maximize accuracy given memory constraint
  - Maximize free memory given fixed NN

- Applications
  - Microrobotic vision
  - Touchpad input classification
  - Spectrogram classification of 1D signals
    - Voice, gesture recognition
    - Activity tracking
    - Biometric security
    - Other sensors



Spectrogram of “yes” keyword from [7]

References

1. Kumar, Ashish, Saurabh Goyal, and Manik Varma. "Resource-efficient machine learning in 2 KB RAM for the internet of things." Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR.org, 2017.
2. Gupta, Chirag, et al. "Protonn: Compressed and accurate knn for resource-scarce devices." Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR.org, 2017.
3. Kusupati, Aditya, et al. "Fastgrnn: A fast, accurate, stable and tiny kilobyte sized gated recurrent neural network." Advances in Neural Information Processing Systems. 2018.
4. TensorFlow Lite for Microcontrollers. URL: https://github.com/tensorflow/tensorflow/tree/master/tensorflow/lite/experimental/micro
5. Cho, Minsik, and Daniel Brand. "MEC: memory-efficient convolution for deep neural network." Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR.org, 2017.
6. Zhang, Jiyuan, Franz Franchetti, and Tze Meng Low. "High performance zero-memory overhead direct convolutions." arXiv preprint arXiv:1809.10170 (2018).
7. Warden, Pete. "Speech commands: A dataset for limited-vocabulary speech recognition." arXiv preprint arXiv:1804.03209 (2018).

Code: https://github.com/agural/memory-optimal-direct-convolutions

Poster: Pacific Ballroom #89