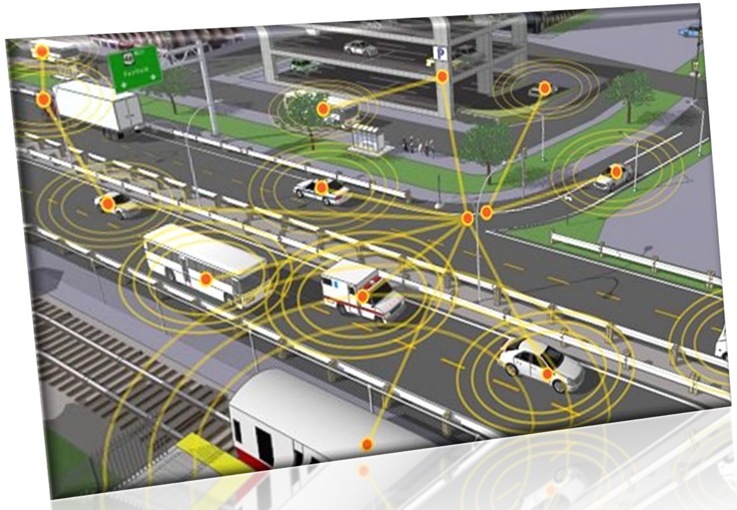
Machine Learning with 2KB RAM

Rahul Sharma

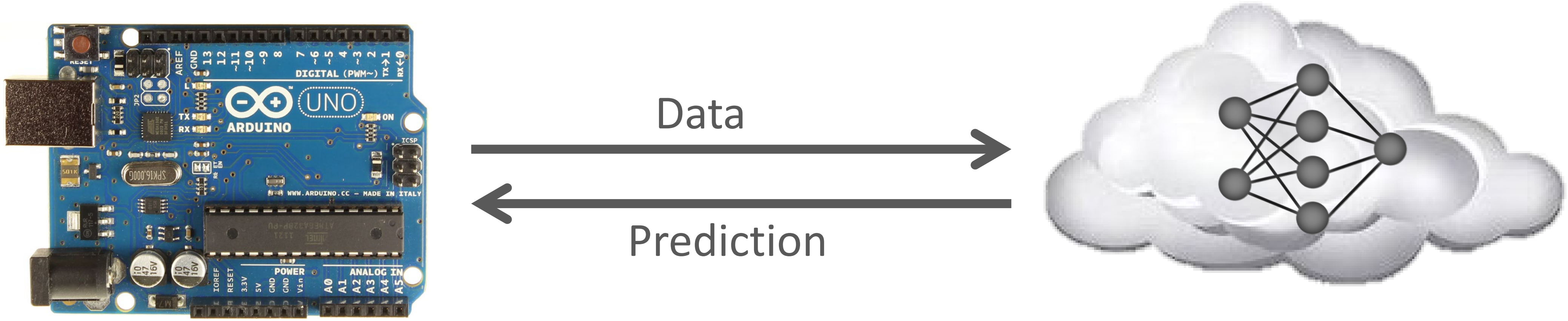
Joint work with Vivek Seshadri, Harsha Vardhan Simhadri, Ajay Manchepalli

RF: Shikhar Jaiswal, Aayan Kumar, Nikhil Pratap Ghanathe, Sridhar Gopinath



IoT is everywhere

Current IoT approach: shortcomings

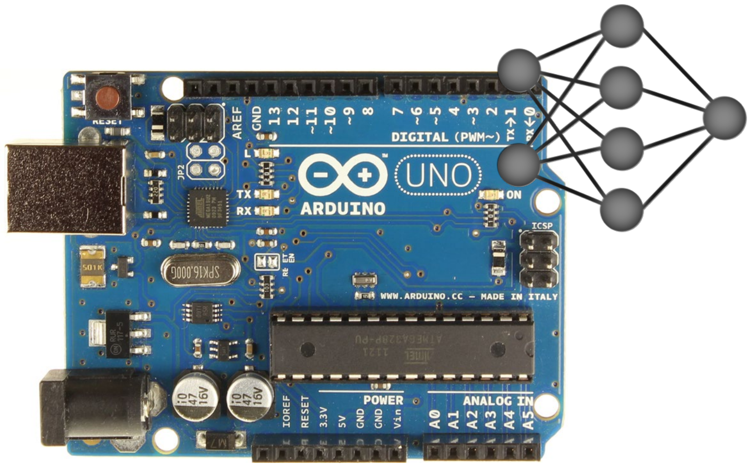


Milli-watt scale microcontroller

High Latency

Energy Efficiency

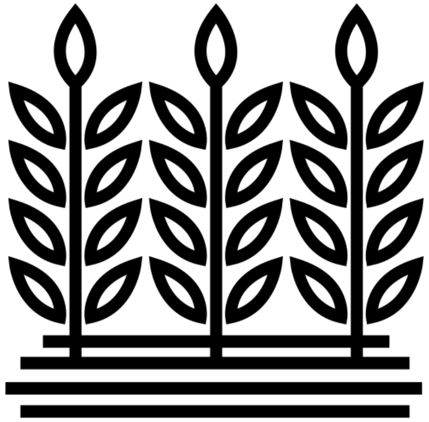
Data Privacy Issues



Arduino Uno

Run ML inference on the IoT device itself

Applications of ML on device



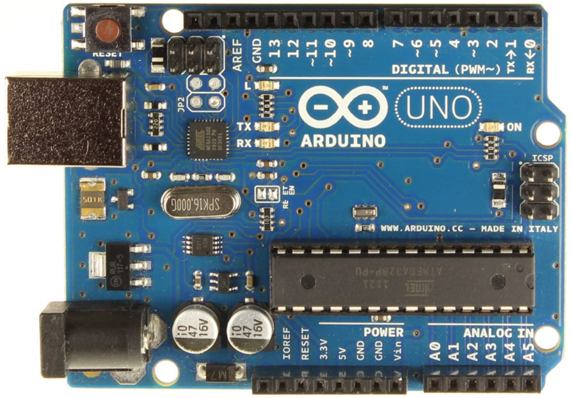
FarmBeats

SenSys’18

GesturePod

UIST’19

IoT devices have limited resources



Arduino Uno

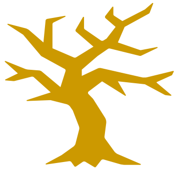
- Arduino Uno
  - Read only memory: **Flash 32 KB**
  - Read/write memory: **RAM 2 KB**
  - No floating-point units

- However, real world ML algorithms are hungry for resources
  - Most require **megabytes** or **gigabytes** of memory

Recent advances in ML

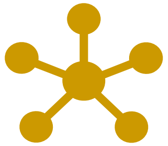
Recent Models have demonstrated high accuracy on real world problems using thousands (instead of millions/billions) of parameters

Decision Trees



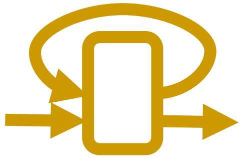
Bonsai
ICML 2017

Nearest Neighbors



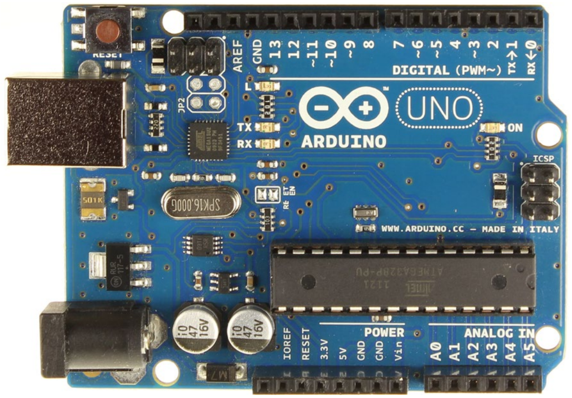
ProtoNN
ICML 2017

Recurrent Neural Networks



FastGRNN, RNNPool
NeurIPS 2018, 2020

These “Edge ML” algorithms are *targeted for IoT devices*



Arduino Uno

- Read only memory: **Flash 32 KB**
- Read/write memory: **RAM 2 KB**
- No floating-point units

Overflow memory

No floating-point units

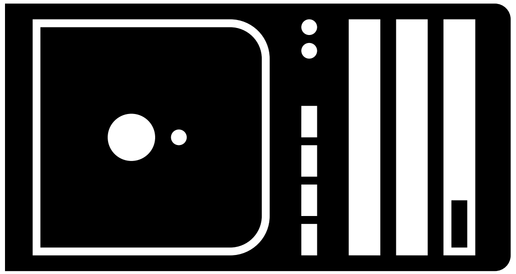
No OS, Virtual memory

Accurate ML models that use few parameters

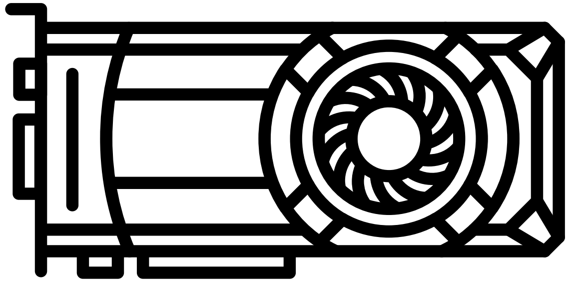
Edge ML

Machine Learning

ONNX K PYTORCH



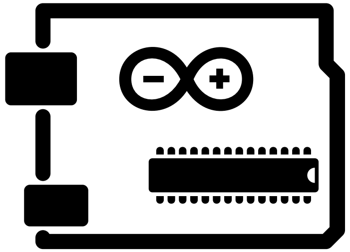
Intel x86



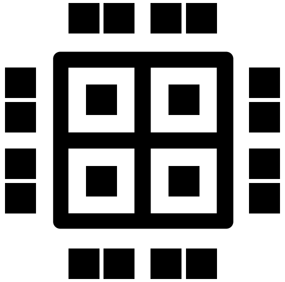
nVIDIA GPU

Machine Learning

IoT
Devices

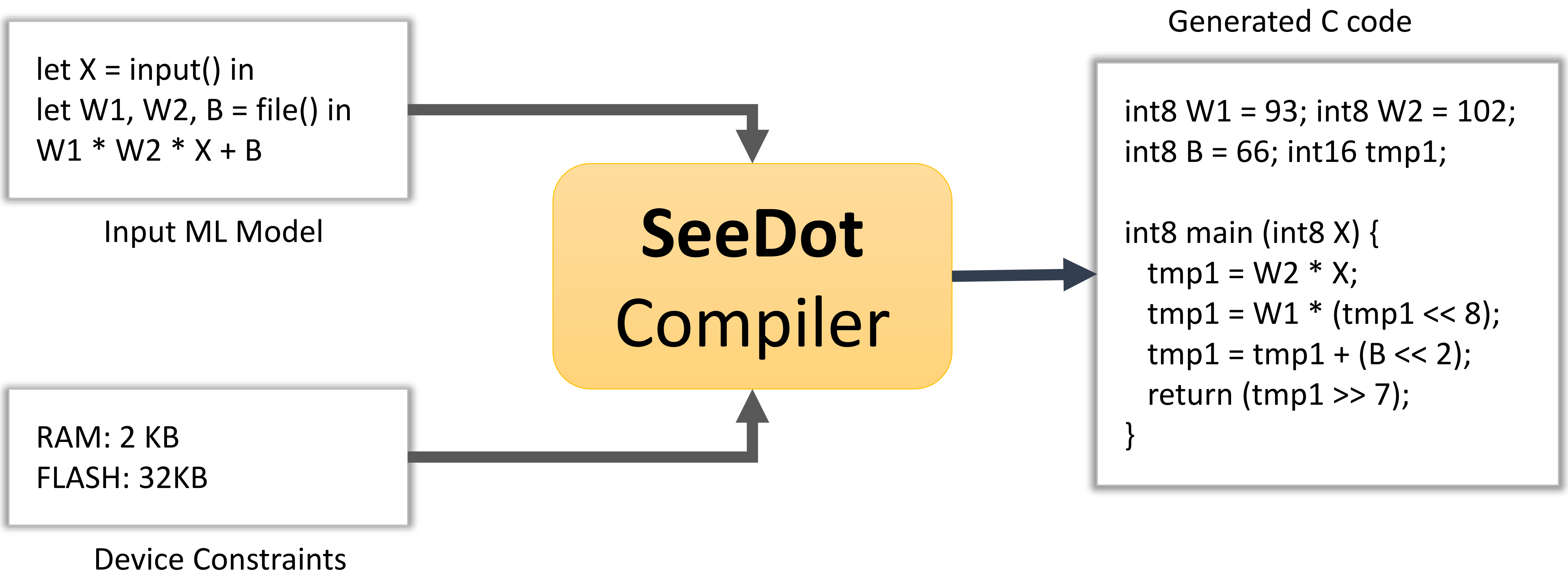


Arduino



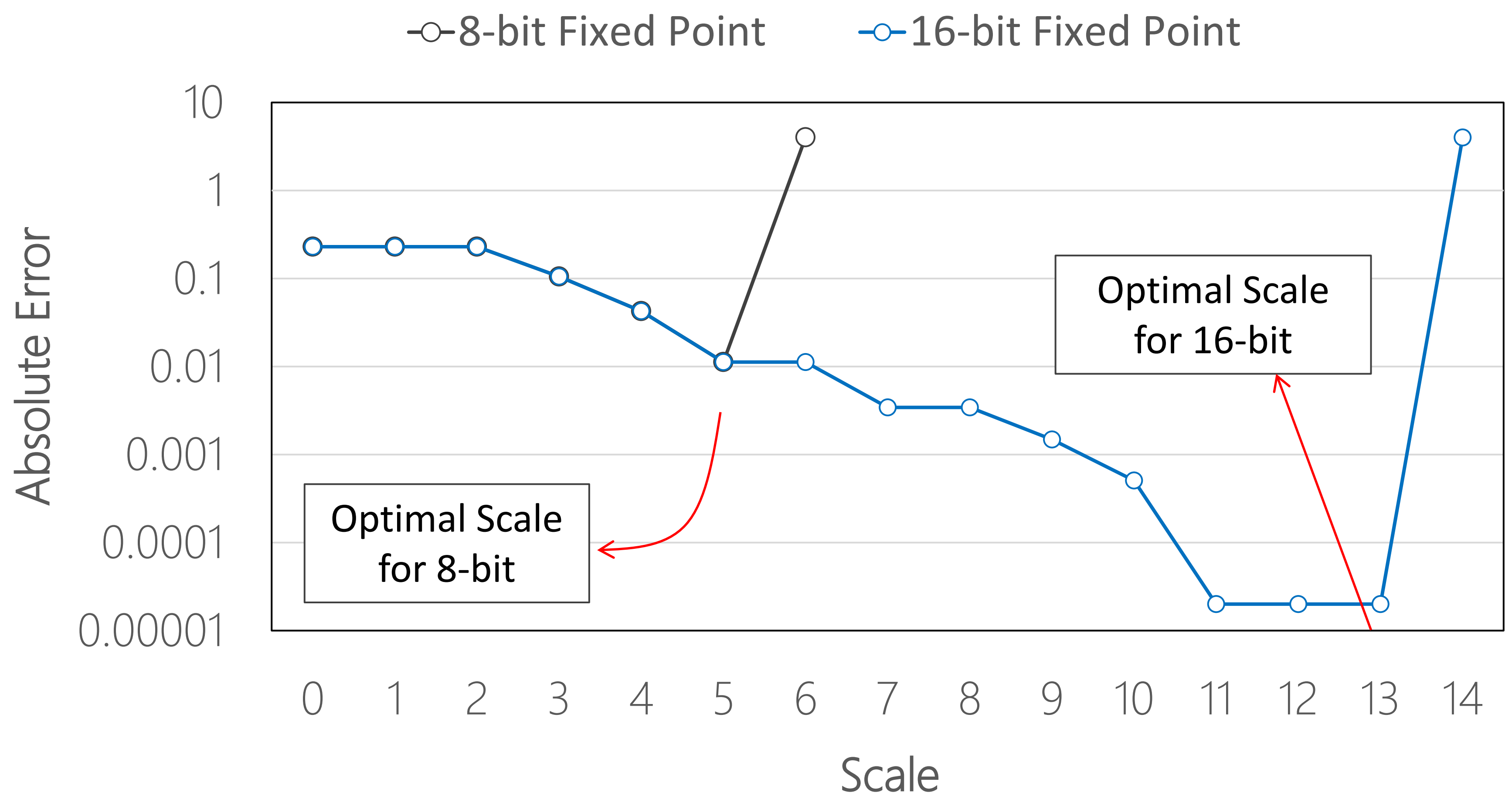
Low-end FPGA

Our approach



No Floating-point | Low Flash | Low RAM

Fixed-point representation: $3.23 \approx 25 / 2^3$ (error = 0.105, scale=3)



No Floating-point | **Low Flash** | Low RAM

- 16-bit homogenous code ⟶ does not fit on Flash
- 8-bit homogenous code ⟶ fits on Flash, but accuracy suffers
  - In some cases, accuracy *equivalent to a random classifier*
  - Models with less parameters need more bits for each parameter
- **Heterogeneous bitwidths:** some variables 16-bits, others 8-bits
  - N-variables lead to 2^N choices
  - Use a heuristic to keep minimal number of variables in 16-bits

No Floating-point | **Low Flash** | Low RAM

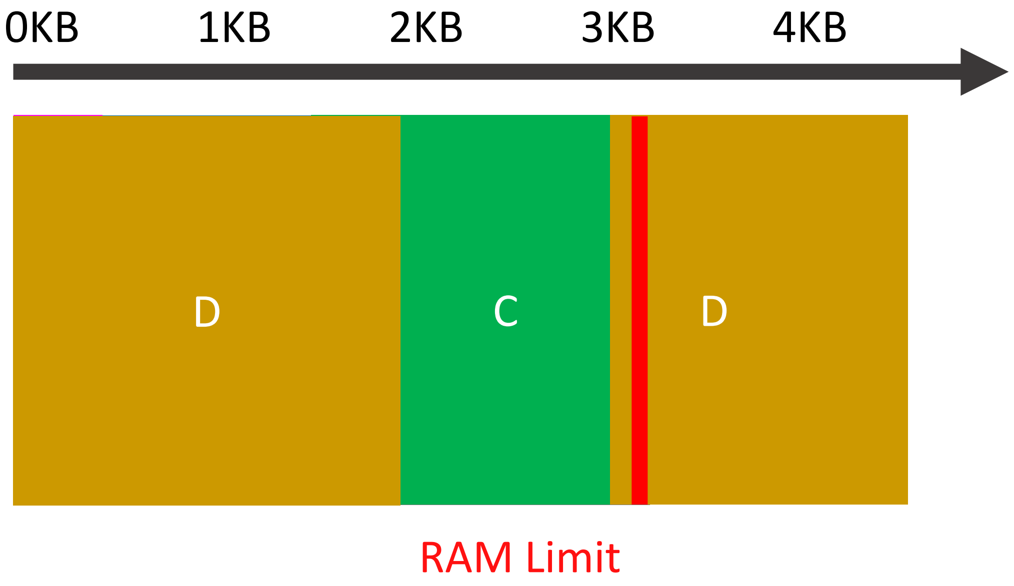
- **Heterogenous bitwidths**

Variables in 8 bits	Variables in 16 bits	Accuracy	Fits within Flash
-	a, b, c, d	97%	NO
a	b, c, d	96%	NO
b	a, c, d	92%	NO
c	a, b, d	67%	NO
d	a, b, c	94%	NO
a	b, c, d	96%	NO
a, d	b, c	93%	YES
a, d, b	c	91%	YES
a, d, b, c	-	55%	YES

No Floating-point | Low Flash | **Low RAM**

- Allocation techniques used by present day compilers do not work
- Dynamic memory allocation also fails

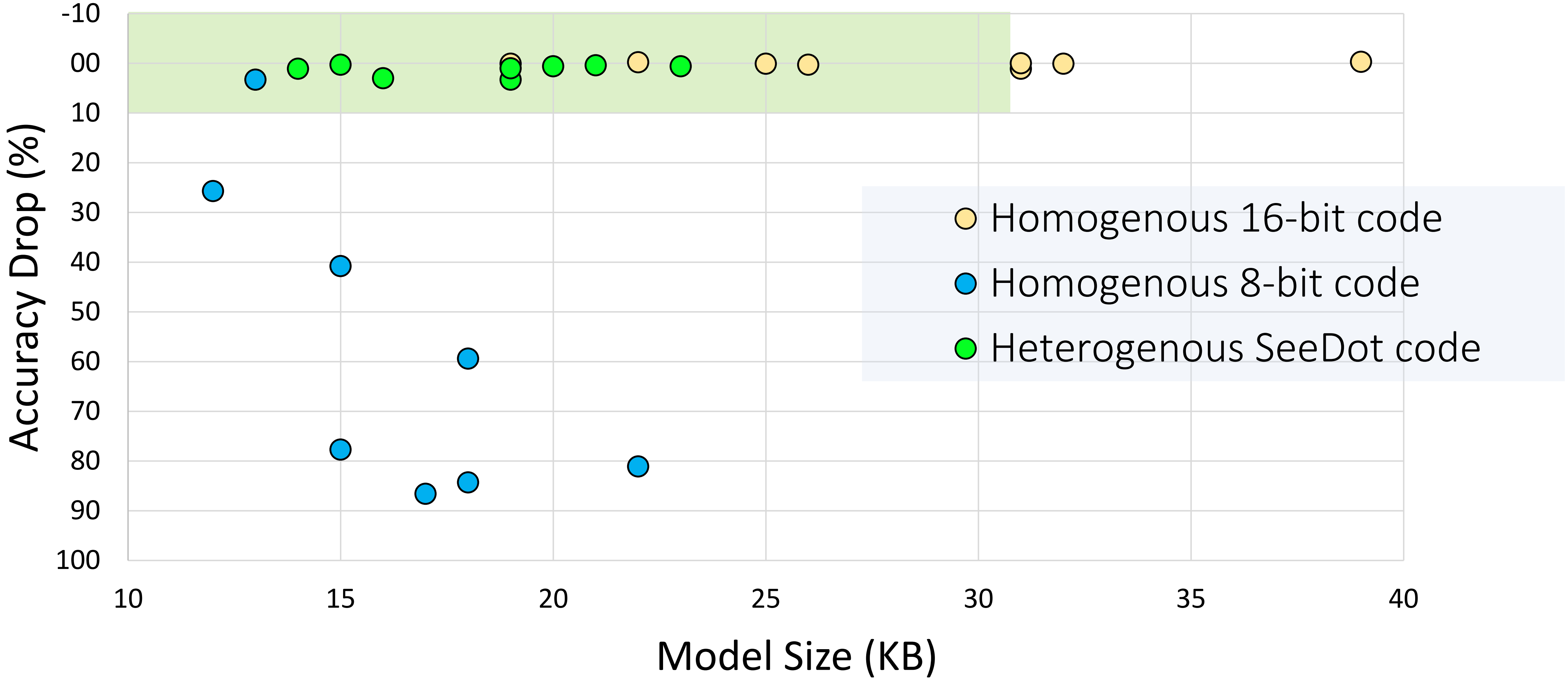
```
      A = malloc(…)
1     A = MatMul(W2, X, …);
      B = malloc(…)
2     B = MatMul(W1, A, …);
      free(A)
      C = malloc(…)
3     C = MatAdd(B, …);
      free(B)
      D = malloc(…)
4     D = MatMul(C, …);
      free(C)
5     return D;
```



No Floating-point | Low Flash | **Low RAM**

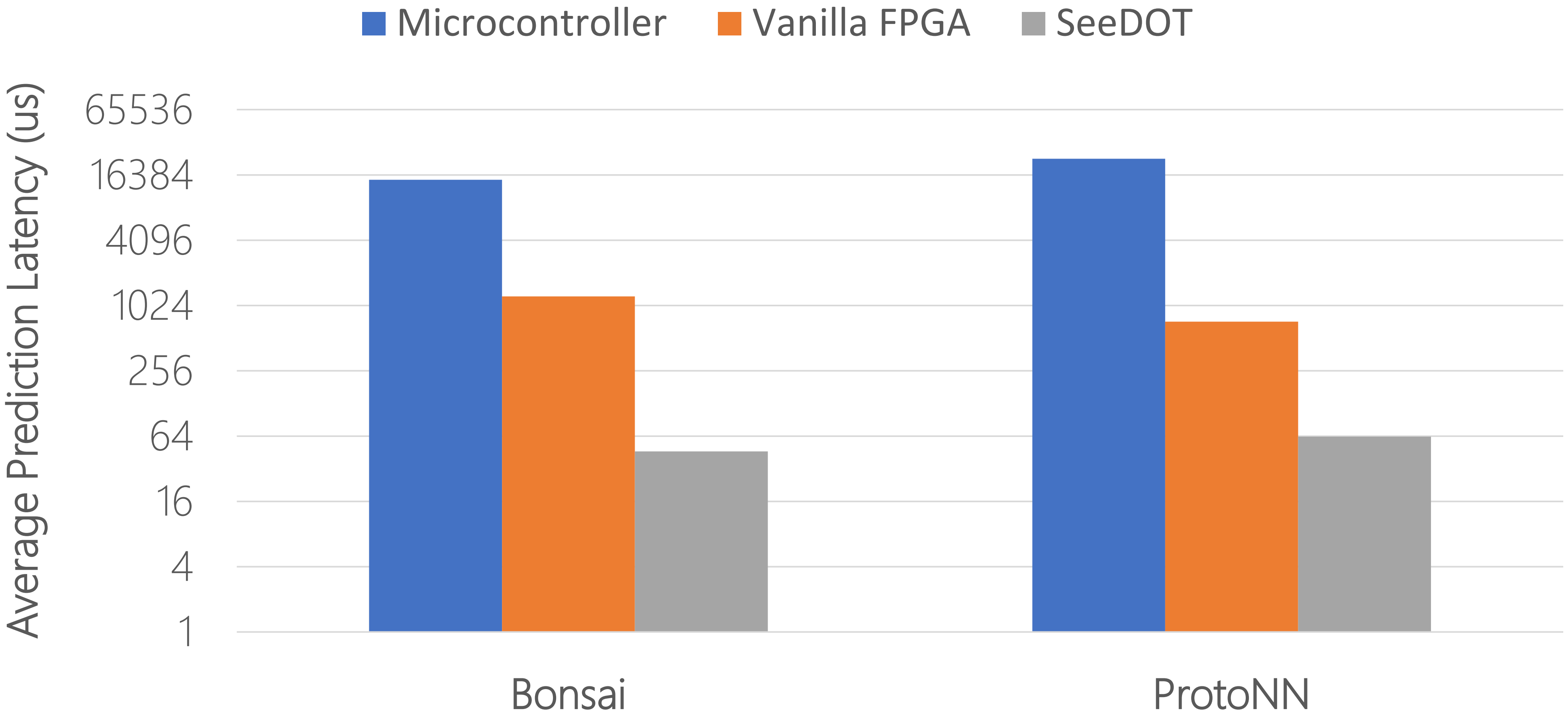
- Dynamic allocation: **fragmentation**
- Sizes of variables and their live ranges known at compile time
- SeeDot simulates dynamic memory allocation, *at compile time*
  - Variable to address mapping computed at compile time
- SeeDot *injects defragmentation code* in the compiler output

FastGRNN on Uno (PLDI'19, OOPSLA'20)



Results on FPGAs (FPL'21)

- Evaluated on Xilinx Arty-board (similar power consumption to Uno)

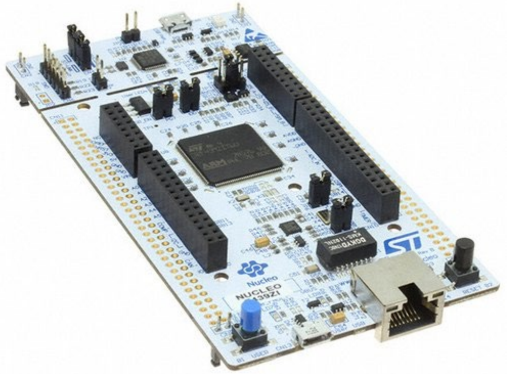


Results with RNNPool (NeurIPS'20 spotlight)

- Using tiny edge devices for room occupancy

- RNNPool+MobileNets for Face Detection problem
  - Provides state-of-the-art accuracy with 150K+ parameters

- ARM Cortex M4 class device
  - 256KB RAM and 512KB Flash



STM32 M4 device

- Comparison with floating-point (default compilation):
  - RAM Usage: **32x reduction** from 7MB to 225KB
  - Flash Usage: **3.3x reduction** from 1.3MB to 405KB

Demo: 1 min video

Conclusion

- SeeDot: given a high-level device-agnostic ML algorithm and device constraints, generates code to run on the device

✓First unified framework for both microcontrollers and FPGAs

✓First evaluation of RNNs on Arduino Uno with 2KB of RAM

✓First demonstration of face detection on ARM Cortex M4

Microsoft

Machine Learning

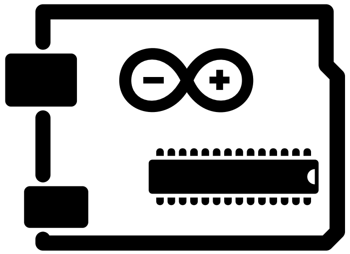
[PLDI’19] [OOPSLA’20] [FPL’21]

[Medium blogpost]

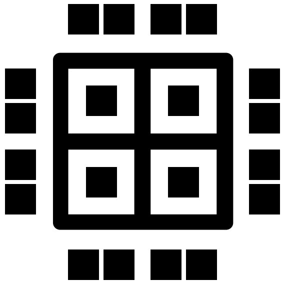
SeeDot

IoT

Devices



Arduino



Low-end FPGA