SeeDot: Compiling ML to IoT Devices

Sridhar Gopinath, Nikhil Ghanathe, Vivek Seshadri, Rahul Sharma

aka.ms/SeeDot

Microsoft Research

World of smart devices

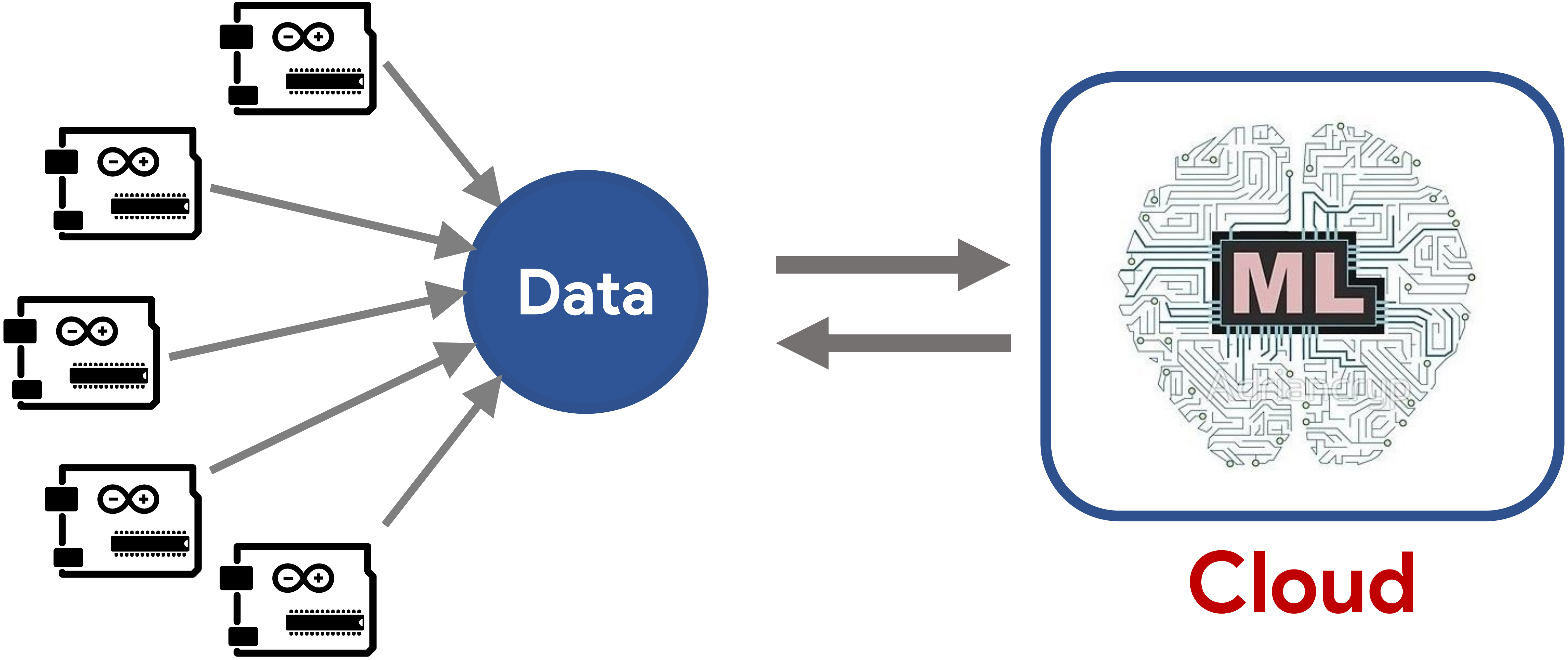
Smart cities

Smart homes

Smart factories

Smart healthcare

ML on the cloud



Sensor/IoT devices

Limitations of ML in the cloud

FarmBeats

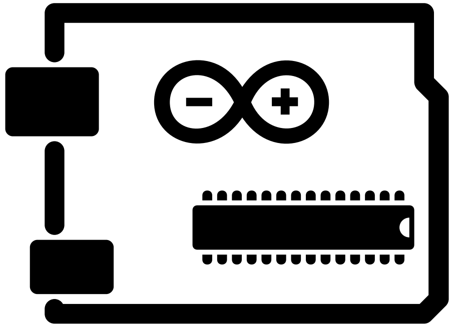
GesturePod

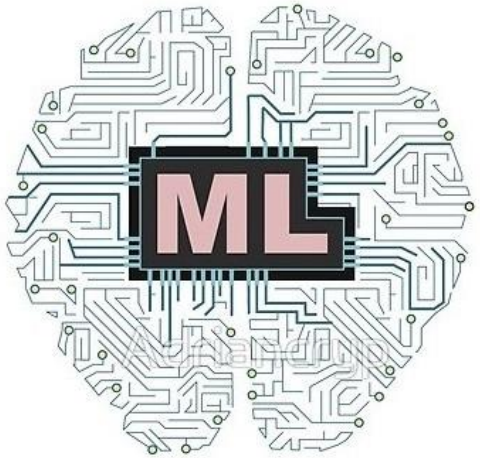
Limitations

- Connectivity
- Battery life
- Privacy

Limitations of ML in the cloud

IoT device





Intelligent edge

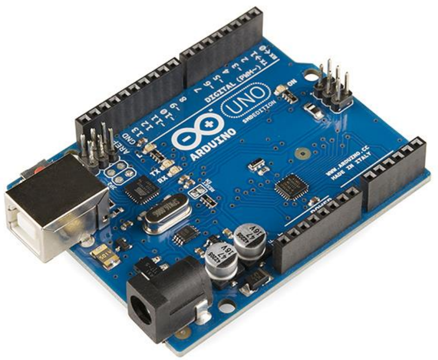
Limitations

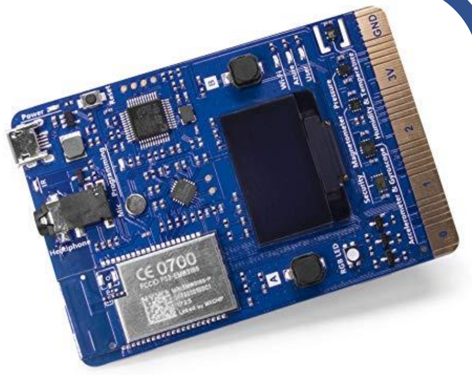
~~Connectivity~~

~~Battery life~~

~~Privacy~~

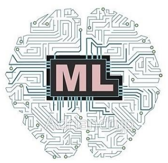
IoT devices





Microcontrollers, FPGAs

1. Low memory/compute resources



New ML algorithms with low memory/compute requirement

Expressed in floating-point

2. No floating-point unit

Translate to integer code

SeeDot overview

ML inference algorithm → **SeeDot compiler** → **Efficient integer program**

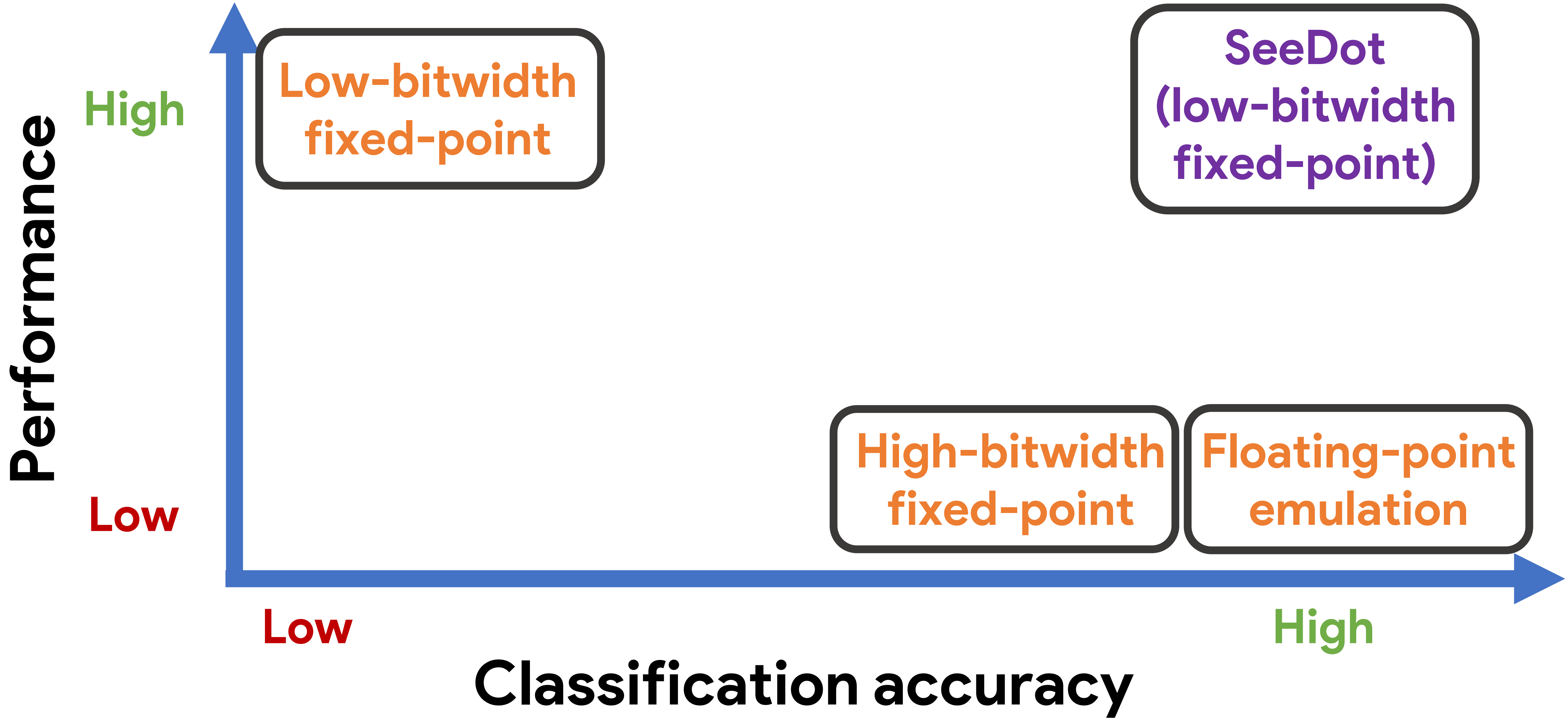
Language

- Mathematical syntax
- Linear algebra operations
- Supports ML operators like conv, maxpool, relu

Compiler

- Automatic floating-point to fixed-point compiler

Related work



Fixed-point Representation

Floating Point → 8-bit Fixed Point

x $\qquad$ (y, k) where $y = \lfloor x * 2^k \rfloor$

y is an 8-bit signed integer, higher k implies better precision

	Overflow	Ideal	Low precision
pi = 3.1415…	(-55,6)	(100,5)	(50,4)
e = 2.7182…	(-83,6)	(86,5)	(43,4)

pi + e $\qquad$ (100,5) + (86,5) → (-70,5) ✗ / (93,4) ✓

Standard Fixed-point Arithmetic

```
a = (x,k); b = (y,k)
```

8-bit Fixed-point Addition:

```
a + b = (x»1 + y»1, k-1)
```

8-bit Fixed-point Multiplication:

```
a * b = (x»4 * y»4, 2k-8)
```

Smaller scale than original numbers

Scale down operation

Naïve fixed-point program

Using standard fixed-point rules

ML algorithm

```
u = a * b
v = c + d
w = ...
x = u * w
y = x + v
```

Generated code

```
u = a»4 * b»4
v = c»1 + d»1
w = ...
x = u»4 * w»4
y = x»1 + v»1
```

Equivalent to a random classifier due to imprecision

Our insight – 1 of 2

Avoid scaling down towards the end of the program

ML algorithm

```
u = a * b
v = c + d
w = ...
x = u * w
y = x + v
```

Generated code

```
u = a»4 * b»4
v = c»1 + d»1
w = ...
x = u * w
y = x + v
```

Prefix

Standard fixed point

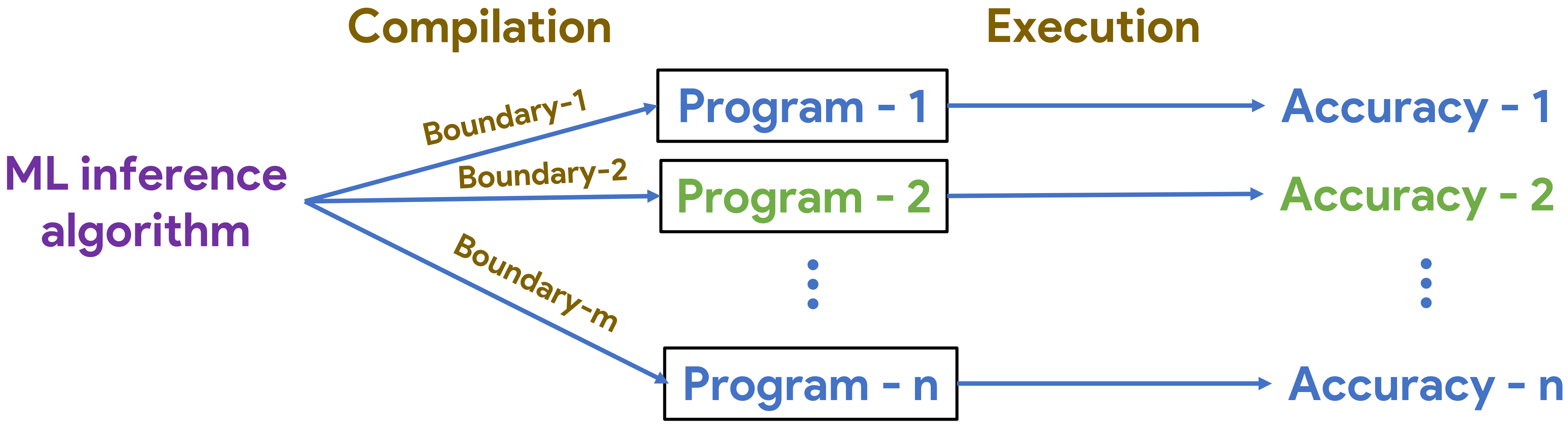
Suffix

No scaling down

Improves precision of the generated program

Our insight – 2 of 2

Measure goodness of the program using classification accuracy



Program with best classification accuracy is selected

Experiments

IoT devices

Arduino Uno

- 2 KB RAM
- 32 KB flash
- 16-bit MCU

Arduino MKR1000

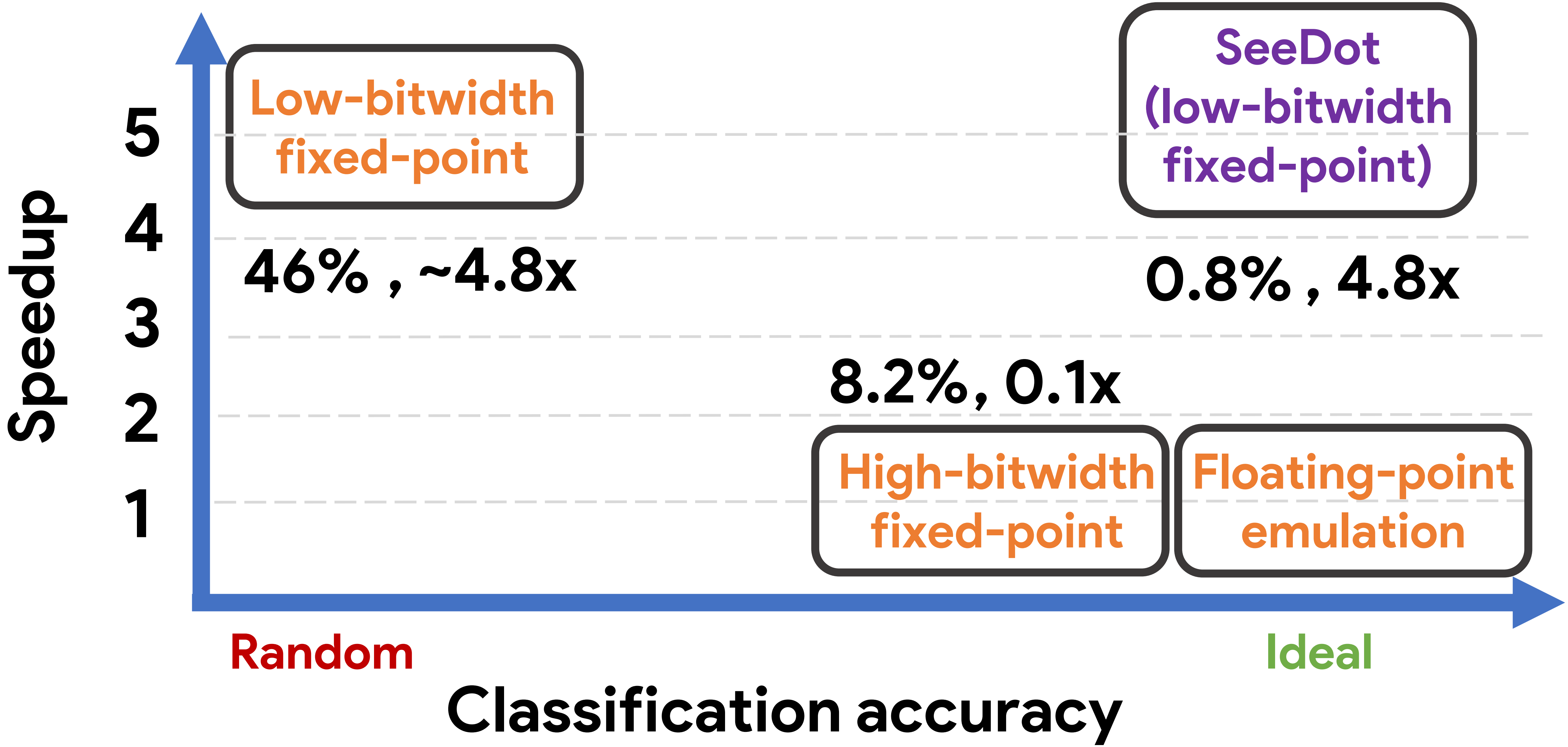
- 32 KB RAM
- 256 KB flash
- 32-bit MCU

Xilinx Arty FPGA

- 20 KB LUT
- 225 KB memory
- 450 MHz freq.

- **ML models**
  - Bonsai
  - ProtoNN
  - Lenet
- **Datasets**
  - Cifar
  - Character recognition
  - Curet
  - Letter
  - Mnist
  - Usps
  - Ward

Experimental results



Other contributions

- Optimized exponentiation
  - Two table look-ups and one fixed-point multiplication
  - Performs **23.3x** faster than **math.h**
- FPGA backend
  - Generates Verilog code
    - Custom SpMV implementation is **13.6x** faster than HLS
    - Generates parallelization hints for HLS
  - SeeDot performs **7.1x** better
  - SeeDot improves FPGA programmability

Conclusion

- Running ML on IoT devices is an emerging domain

- **SeeDot**
  - Language can express ML algorithms succinctly
  - Float-to-fixed compiler to run ML efficiently on IoT devices

- Results
  - Improved performance on **microcontrollers** by **4.8x**
  - Improved performance on **FPGAs** by **7.1x**
  - Implementation available on GitHub:
    github.com/Microsoft/EdgeML