MinUn

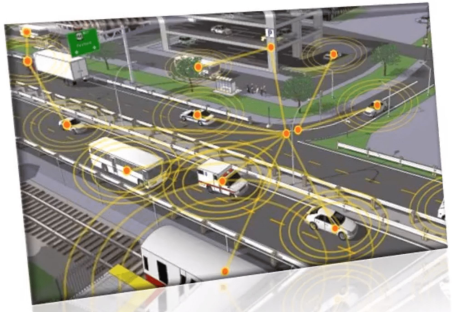
Accurate ML Inference on Microcontrollers

Shikhar Jaiswal, Rahul Goli, Aayan Kumar, Vivek Seshadri, Rahul Sharma

Microsoft Research India

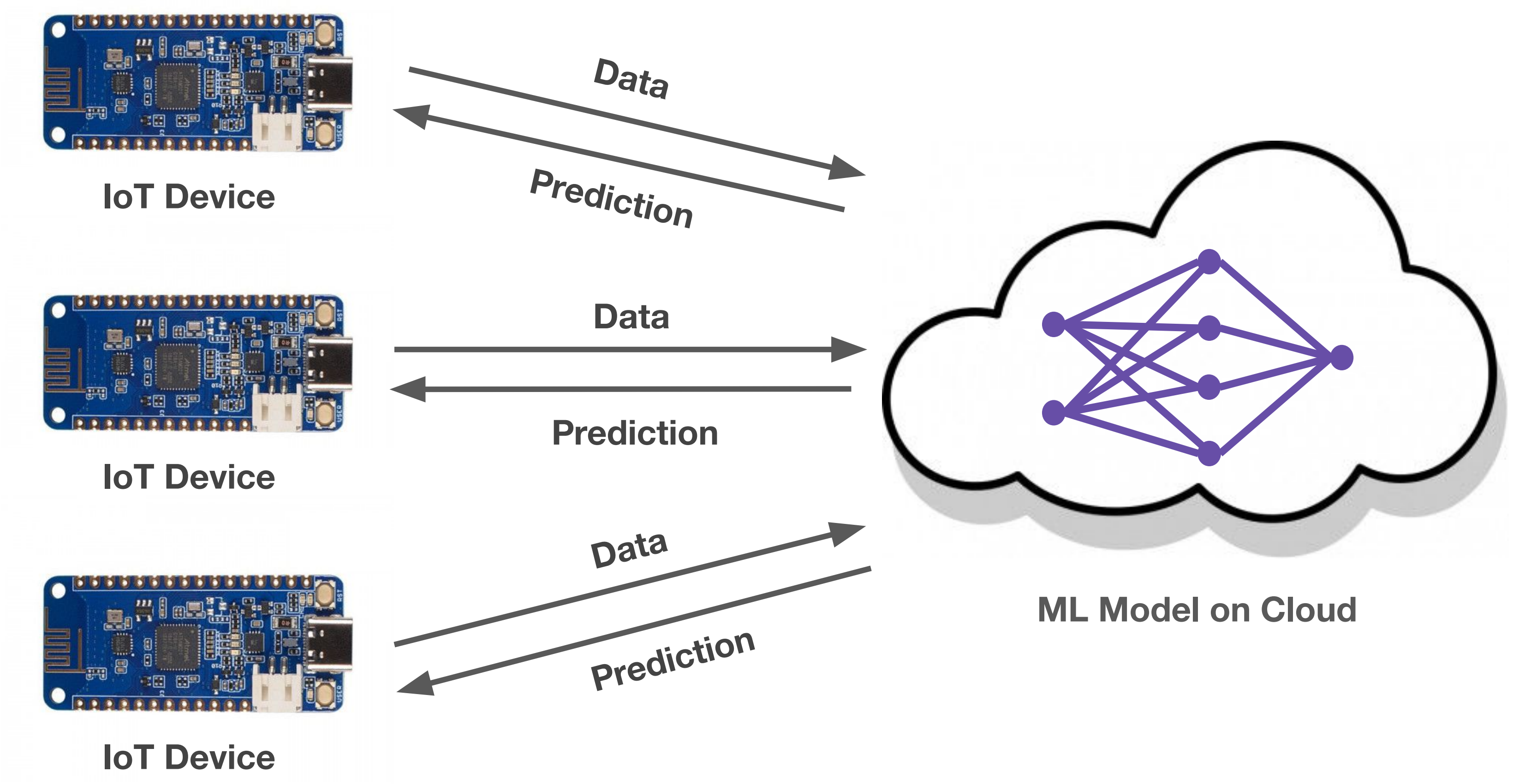
Instructional Repository: **https://github.com/ShikharJ/MinUn**

Microsoft



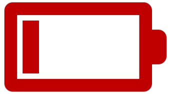
Embedded Devices are Ubiquitous

Previous IoT Approaches: ML-On-Cloud

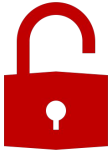


Limitations of ML-On-Cloud

High Communication Latency

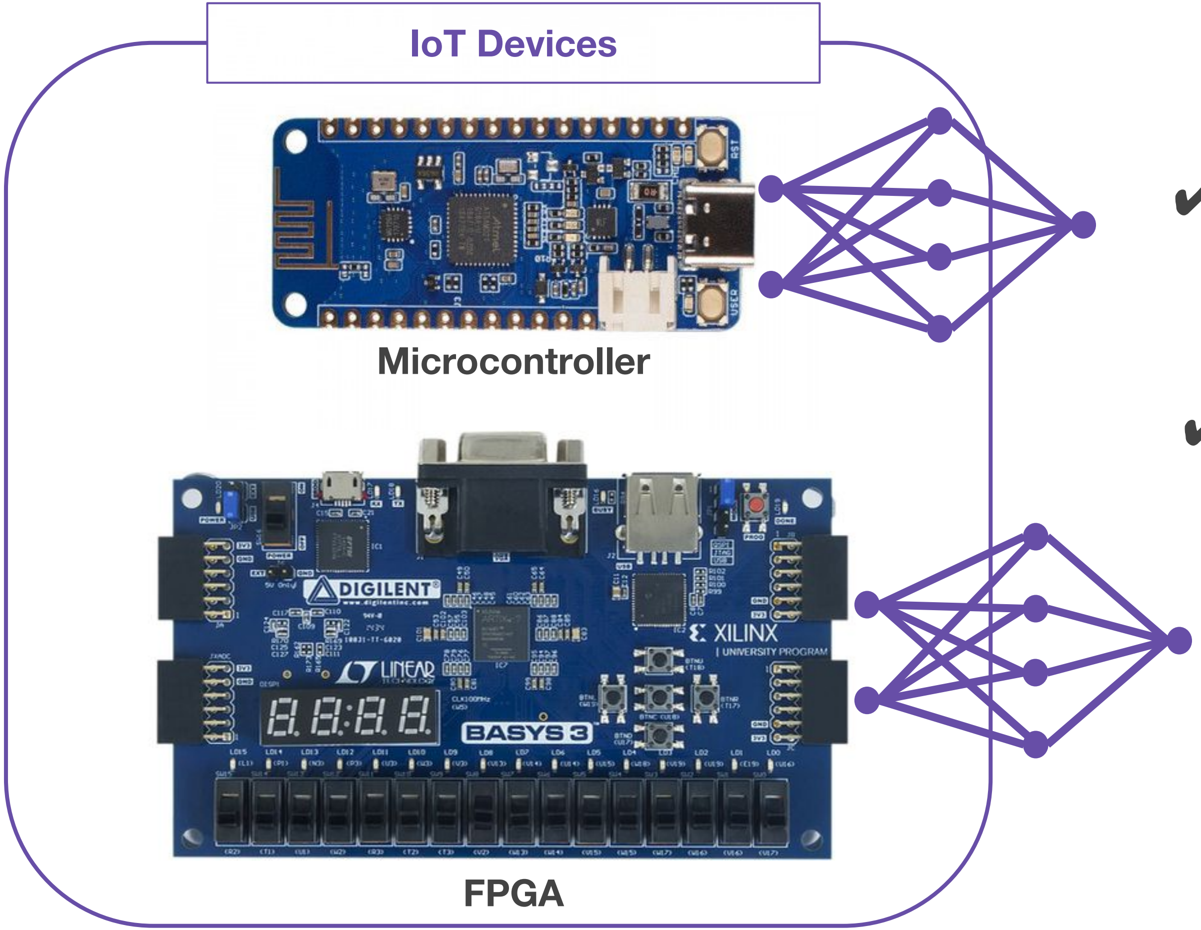


Poor Efficiency in Battery-Operated Scenarios



Data Privacy Considerations

Solution: ML-On-Edge-Devices (TinyML)



✔ No need to communicate data to the cloud for inference.

✔ Suitable for battery-operated scenarios as communication latencies are eliminated.

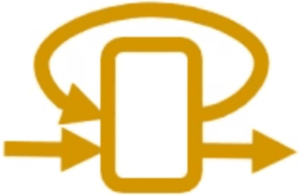
✔ Data doesn't leave the source.

Advances in TinyML Models

Decision Trees

Bonsai
ICML 2017

Recurrent Neural Networks

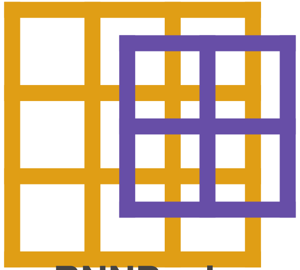


FastGRNN
NeurIPS 2018

Nearest Neighbors

ProtoNN
ICML 2017

Special Pooling Operators



RNNPool
NeurIPS 2020

Frameworks: The Task and the Challenges

Problem Statement: To generate efficient C / C++ codes for TinyML models, which can be executed on tiny microcontrollers with KBs of main memory.

Challenge 1: Which number representation should the program use?

Challenge 2: What bitwidth should the program assign to a variable?

Challenge 3: What about memory management?

Challenge: Representation Independence

Fixed-Point Representation

$$q_b = \lfloor r \times 2^S \rfloor$$

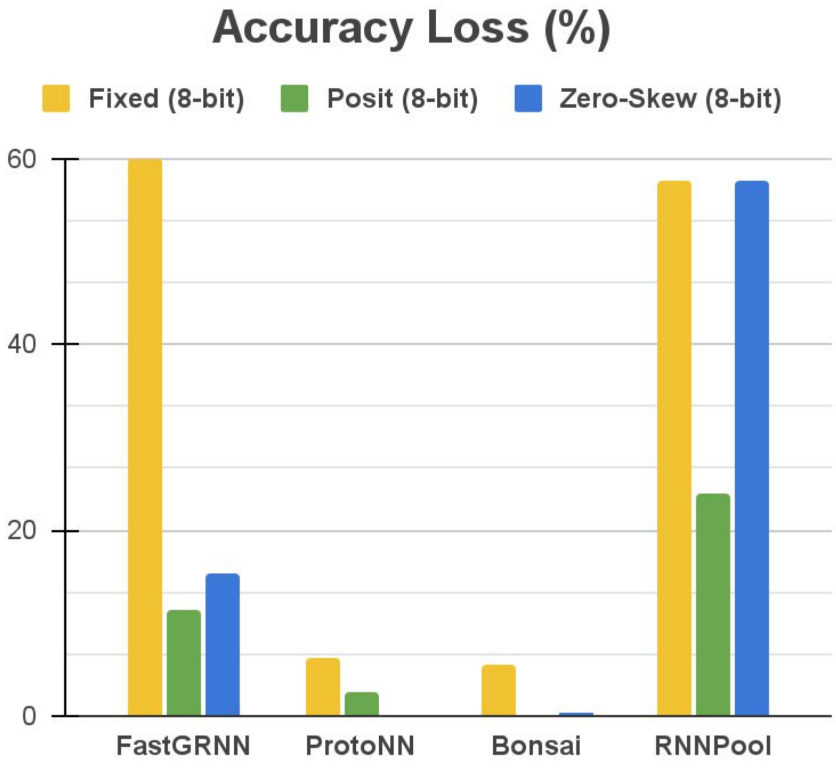
Zero-Skew Representation

$$q_b = \lfloor \frac{r}{S} \rfloor + Z$$

Posit Representation

	s		r				e			f	
q_b	s_1	r_1	r_2	$\ldots$	$\overline{r}$	e_1	e_2	$\ldots$	e_{es}	f_1	$\ldots$

$$r = (-1)^s \times (2^{2^{es}})^k \times 2^E \times 1.F$$



Challenge: Bitwidth Exploration

Linear Classifier

$$W_1 := \begin{pmatrix} -2.139562 & 1.885351 \end{pmatrix}$$

$$X_1 := \begin{pmatrix} 1.185109 \\ -2.206466 \end{pmatrix}$$

$$B_1 := \begin{pmatrix} 0.146048 \end{pmatrix}$$

$$\textbf{return } (W_1 \times X_1) + B_1$$

Fixed-Point Representation $q_b = \lfloor r \times 2^S \rfloor$

Example: $r = \pi = 3.14159$

b	S	q_b	**Interpreted Value**	**Error**
8	5	$\lfloor \pi \times 2^5 \rfloor \approx 101$	$101 / 2^5 \approx 3.156$	10^{-2}
16	9	$\lfloor \pi \times 2^9 \rfloor \approx 1608$	$1608 / 2^9 \approx 3.1406$	10^{-3}

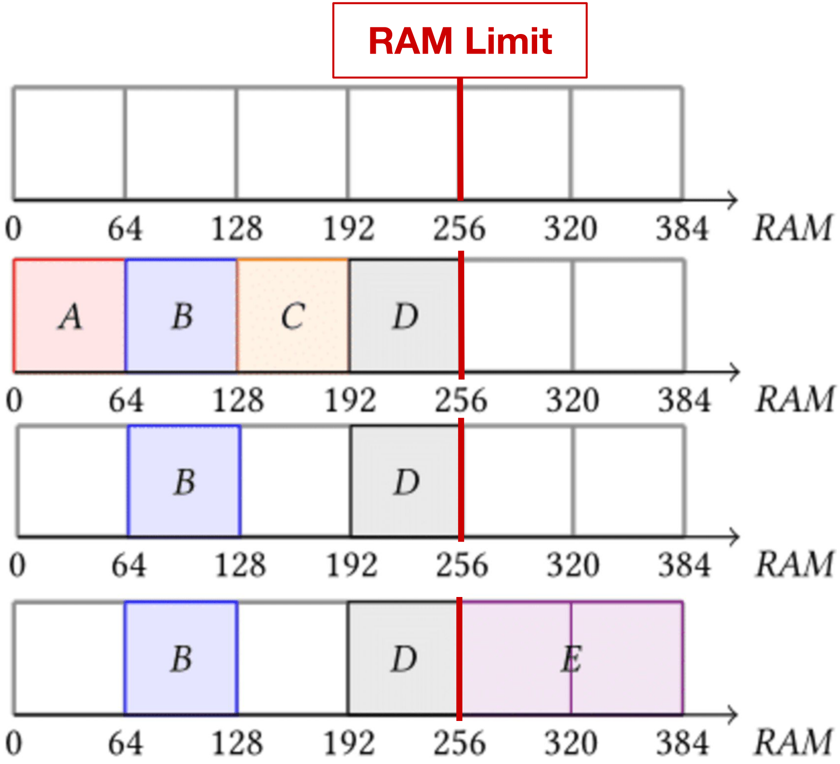
Very Large Exploration Space: For a program with **N** variables and **k** bitwidth options, the total number of possible assignments is k^N.

Challenge: Memory Fragmentation

```
...
a = malloc();
b = malloc();
c = malloc();
d = malloc();

b = MatMul(a, c);
d = MatMul(a, b);
free(a);
free(c);

e = malloc();
e = MatMul(b, d);
...
```



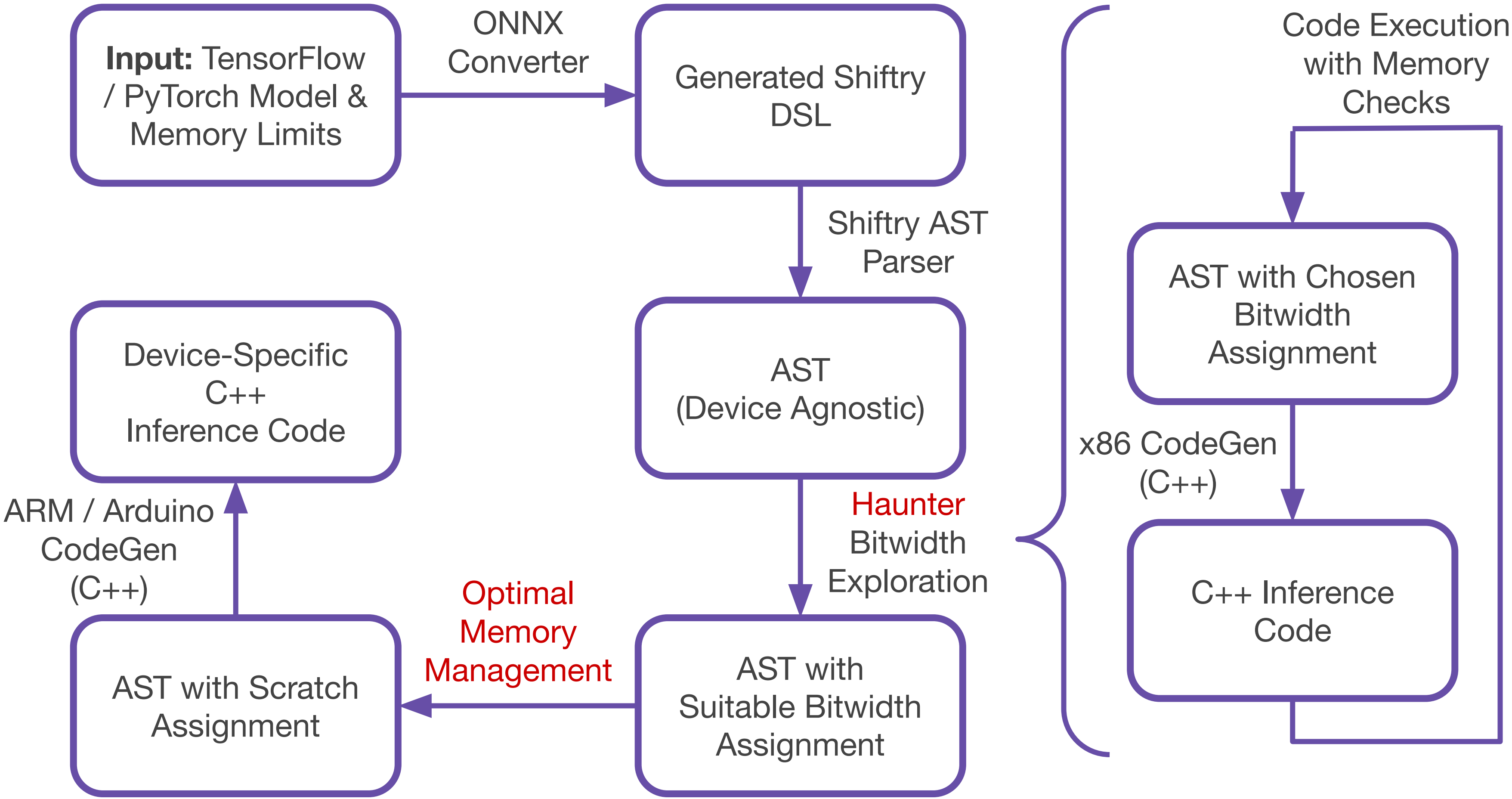
MinUn: Addressing the Challenges

MinUn Compiler: Offers representation and platform-independent design, which is easily integrable with any representation of choice.

Haunter Bitwidth Exploration Algorithm: Makes use of information regarding both the variable size and it's impact on overall program accuracy to generate candidate bitwidth assignments, within a strict memory budget.

Optimum Memory Management: Makes use of Knuth's Algorithm X, for optimally assigning scratch space to variables.

MinUn: Overview



Related TinyML Frameworks

TFLite

CVPR 2018

-> Compiles floating-point code into zero-skew code.

-> Only generates uniform bitwidth (8 / 16-bit weights and 32-bit biases) codes.

-> Uses TFLite interpreter with memory overheads. Leaves memory handling to the end user.

SeeDot

PLDI 2019

-> Compiles floating-point code into fixed-point code.

-> Only generates uniform bitwidth (8 / 16-bit) codes.

-> Ignores the memory fragmentation problem and assumes sufficient RAM is always available.

Shiftry

OOPSLA 2020

-> Compiles floating-point code into fixed-point code.

-> Generates variable bitwidth (8 and 16-bit) codes.

-> Offers a suboptimal greedy heuristic for preventing fragmentation.

Experiments

Models

- FastGRNN
- ProtoNN
- Bonsai
- RNNPool
- SqueezeNet

Datasets

- CIFAR
- CR
- Curet
- Letter
- USPS
- ImageNet
- MNIST
- Ward
- DSA
- Google
- HAR
- SCUT-HEAD

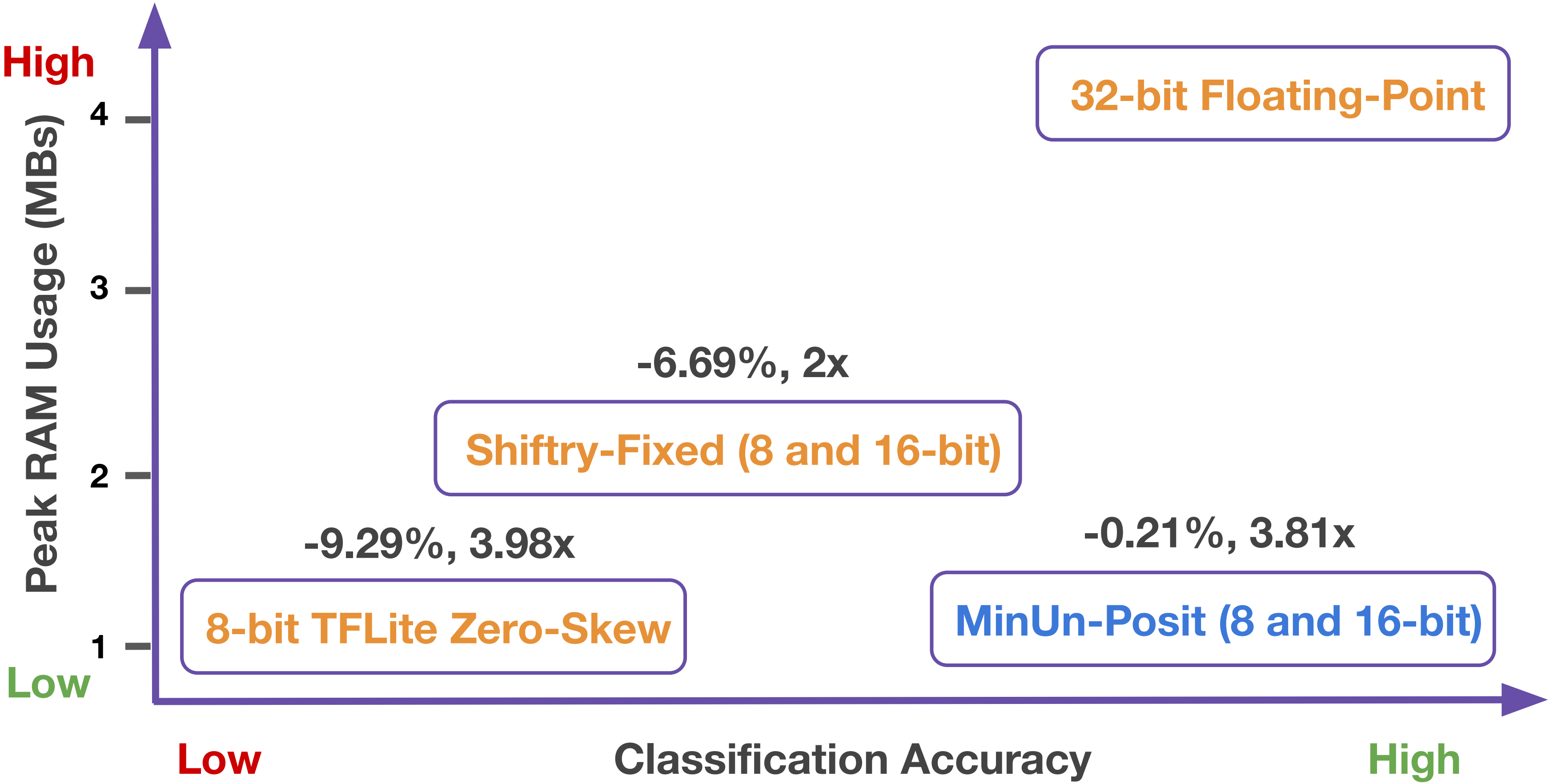
IoT Devices

- Arduino Uno (2KB SRAM, 32KB Flash)
- Arduino Due (96KB SRAM, 512KB Flash)
- STM32H747 (1MB SRAM, 2MB Flash)

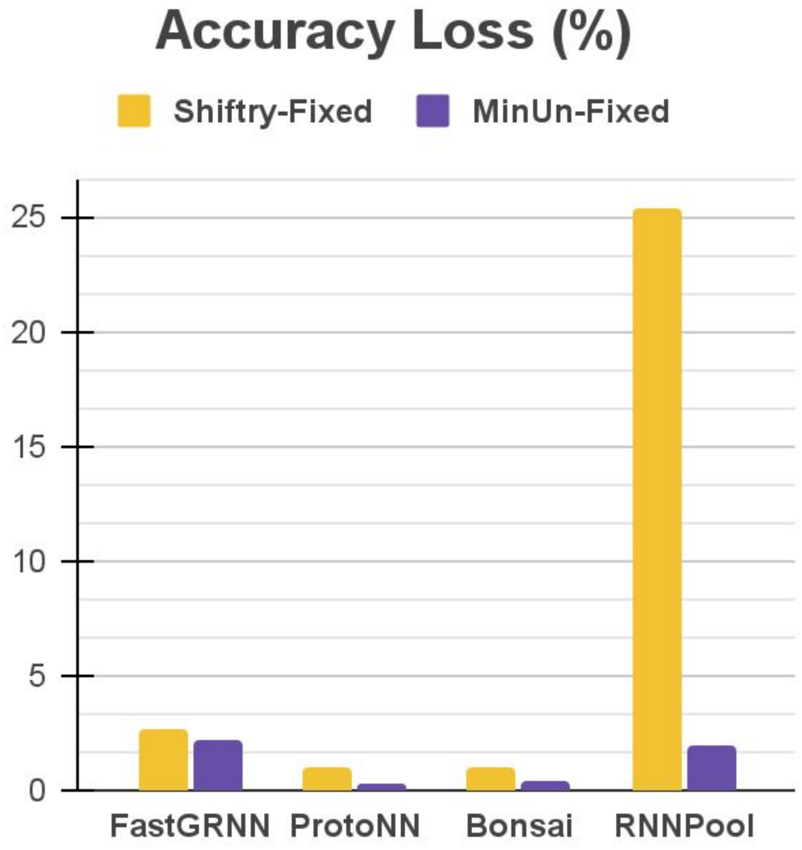
Representations

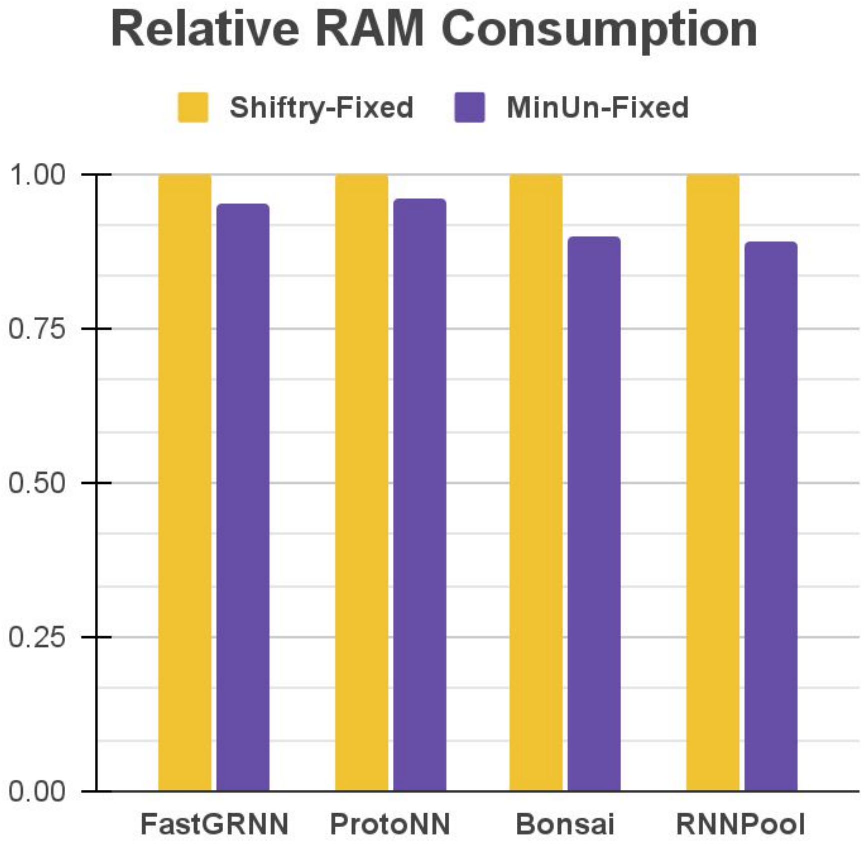
- Floating-Point (32-bit)
- BFloat (16-bit)
- Fixed-Point (8, 16-bit)
- Posit (8, 9, 10, 12, 16-bit)
- TFLite Zero-Skew (8-bit)

Running SqueezeNet on ImageNet-1K

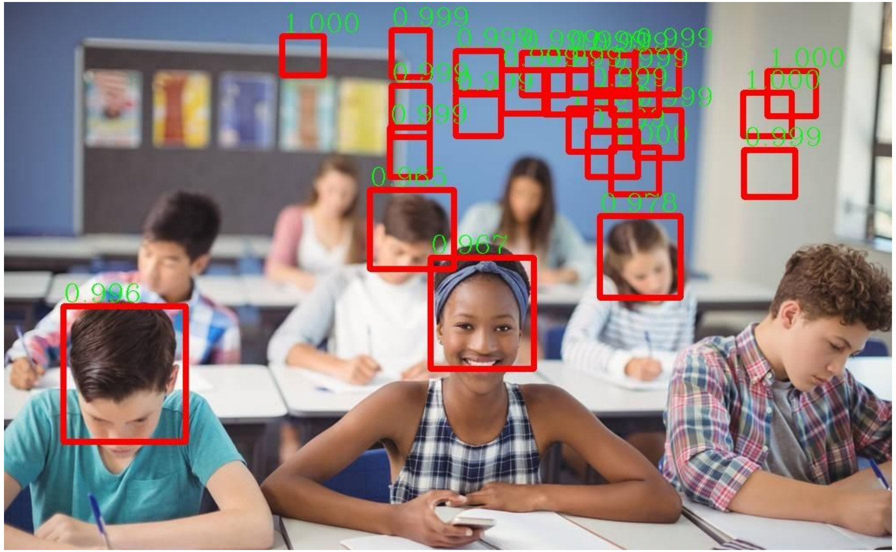


Quantitative Comparison with Shiftry

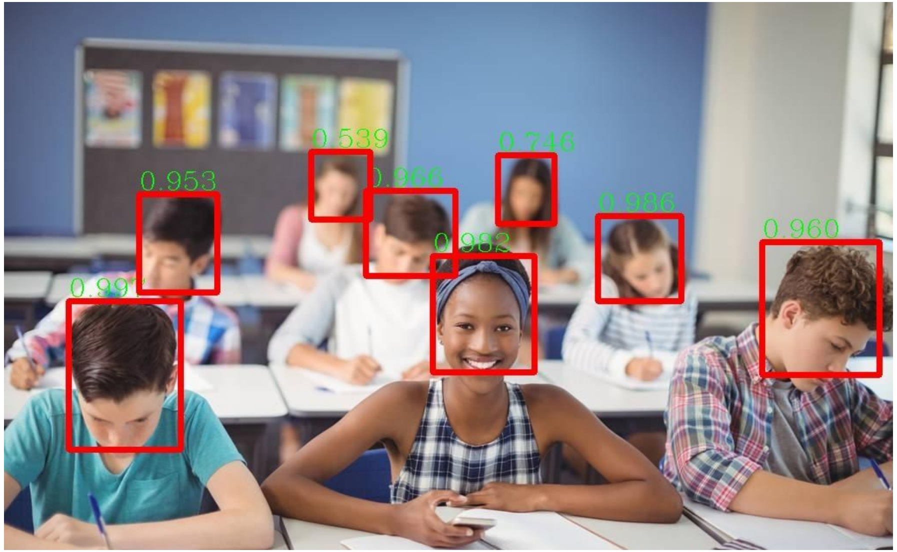




Qualitative Comparison with Shiftry



Shiftry-Fixed on Face-C Model



MinUn-Fixed on Face-C Model

Conclusion

- ➔ MinUn is a new framework for compiling ML models on embedded devices.
- ➔ **MinUn:**
  - ◆ Addresses the representation independence, bitwidth exploration and memory fragmentation challenges.
  - ◆ Generates C / C++ codes for bare-metal environments.
- ➔ Our evaluation shows that MinUn generates ML models which are more accurate and consume less RAM than prior SOTA.

Instructional Repository: https://github.com/ShikharJ/MinUn