Smart cities

Smart homes

Smart factories

World of smart devices

Smart healthcare
ML on the cloud

Sensor/IoT devices

Data

Cloud
Limitations of ML in the cloud

- Connectivity
- Battery life
- Privacy

FarmBeats
GesturePod
Limitations of ML in the cloud

- Connectivity
- Battery life
- Privacy
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 \rightarrow \text{SeeDot compiler} \rightarrow \text{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

Classification accuracy

Performance

Low

High

Low-bitwidth fixed-point

SeeDot (low-bitwidth fixed-point)

High-bitwidth fixed-point

Floating-point emulation
Fixed-point Representation

Floating Point  8-bit Fixed Point

\( x \quad (y, k) \text{ where } y = \lfloor x \times 2^k \rfloor \)

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

\[
\begin{array}{c|c|c}
\text{Overflow} & \text{Ideal} & \text{Low precision} \\
\hline
\pi = 3.1415... & (-55,6) & (100,5) & (50,4) \\
e = 2.7182... & (-83,6) & (86,5) & (43,4) \\
\pi + e & (100,5) + (86,5) & (-70,5) & (93,4) \\
\end{array}
\]
Standard Fixed-point Arithmetic

\[ a = (x, k); \quad b = (y, k) \]

8-bit Fixed-point Addition:
\[ a + b = (x \ll 1 + y \ll 1, \; k-1) \]

8-bit Fixed-point Multiplication:
\[ a \times b = (x \ll 4 \times y \ll 4, \; 2k-8) \]

Smaller scale than original numbers

Scale down operation
Naïve fixed-point program

Using standard fixed-point rules

ML algorithm

\[
\begin{align*}
u &= a \times b \\
v &= c + d \\
w &= \ldots \\
x &= u \times w \\
y &= x + v
\end{align*}
\]

Generated code

\[
\begin{align*}
u &= a\times4 \times b\times4 \\
v &= c\times1 + d\times1 \\
w &= \ldots \\
x &= u\times4 \times w\times4 \\
y &= x\times1 + v\times1
\end{align*}
\]

Equivalent to a random classifier due to imprecision
Our insight – 1 of 2

Avoid scaling down towards the end of the program

ML algorithm

\[ u = a \times b \]
\[ v = c + d \]
\[ w = \ldots \]
\[ x = u \times w \]
\[ y = x + v \]

Generated code

\[ u = a\times4 \times b\times4 \]
\[ v = c\times1 + d\times1 \]
\[ w = \ldots \]
\[ x = u \times w \]
\[ y = x + v \]

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

<table>
<thead>
<tr>
<th>Device</th>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arduino Uno</td>
<td>• 2 KB RAM&lt;br&gt;• 32 KB flash&lt;br&gt;• 16-bit MCU</td>
</tr>
<tr>
<td>Arduino MKR1000</td>
<td>• 32 KB RAM&lt;br&gt;• 256 KB flash&lt;br&gt;• 32-bit MCU</td>
</tr>
<tr>
<td>Xilinx Arty FPGA</td>
<td>• 20 KB LUT&lt;br&gt;• 225 KB memory&lt;br&gt;• 450 MHz freq.</td>
</tr>
</tbody>
</table>

### ML models

- Bonsai
- ProtoNN
- Lenet

### Datasets

- Cifar
- Character recognition
- Curet
- Letter
- Mnist
- Usps
- Ward
Experimental results

- **Low-bitwidth fixed-point**: 46%, ~4.8x speedup.
- **High-bitwidth fixed-point**: 8.2%, 0.1x speedup.
- **Floating-point emulation**: 0.8%, 4.8x speedup.

SeeDot (low-bitwidth fixed-point)
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