Generating Fast Operators for Binarizable Networks

Meghan Cowan
Running Binarizable Networks?
Running Binarizable Networks?

Training in frameworks with no binarizable operators.
Running Binarizable Networks?

Training in frameworks with no binarizable operators.

Can’t evaluate performance gains.
Running Binarizable Networks?

Training in frameworks with no binarizable operators.

Can’t evaluate performance gains

Easy to introduce bugs
Running Binarizable Networks?

- Training in frameworks with no binarizable operators.
- Can’t evaluate performance gains
- Easy to introduce bugs

Need to generate binarizable operators ourselves!
Baselines are incredibly well optimized.

Without optimizations low precision can’t compete.
Baselines are incredibly well optimized

Without optimizations low precision can't compete
Baselines are incredibly well optimized
Without optimizations low precision can’t compete

Want operators that are fast

Need optimized operators for all workloads
Performance portability across different CPUs
Generating Fast Operators for Binarizable Networks
Generating Fast Operators for Binarizable Networks

Declare bitserial computation and CPU schedule describing an optimization space
Generating Fast Operators for Binarizable Networks

Declare bitserial computation and CPU schedule describing an optimization space

Use AutoTVM use to find schedule parameters for different operators and backends
Generating Fast Operators for Binarizable Networks

- Declare bitserial computation and CPU schedule describing an optimization space
- Use AutoTVM use to find schedule parameters for different operators and backends

- Overrule LLVM code generation with custom microkernel
- Use tensorize primitive to replace inner-most loop of computation
Convolutions on Raspberry Pi

Can generate low precision convolutions
5.5x to 15.2x faster than optimized 16-bit integer