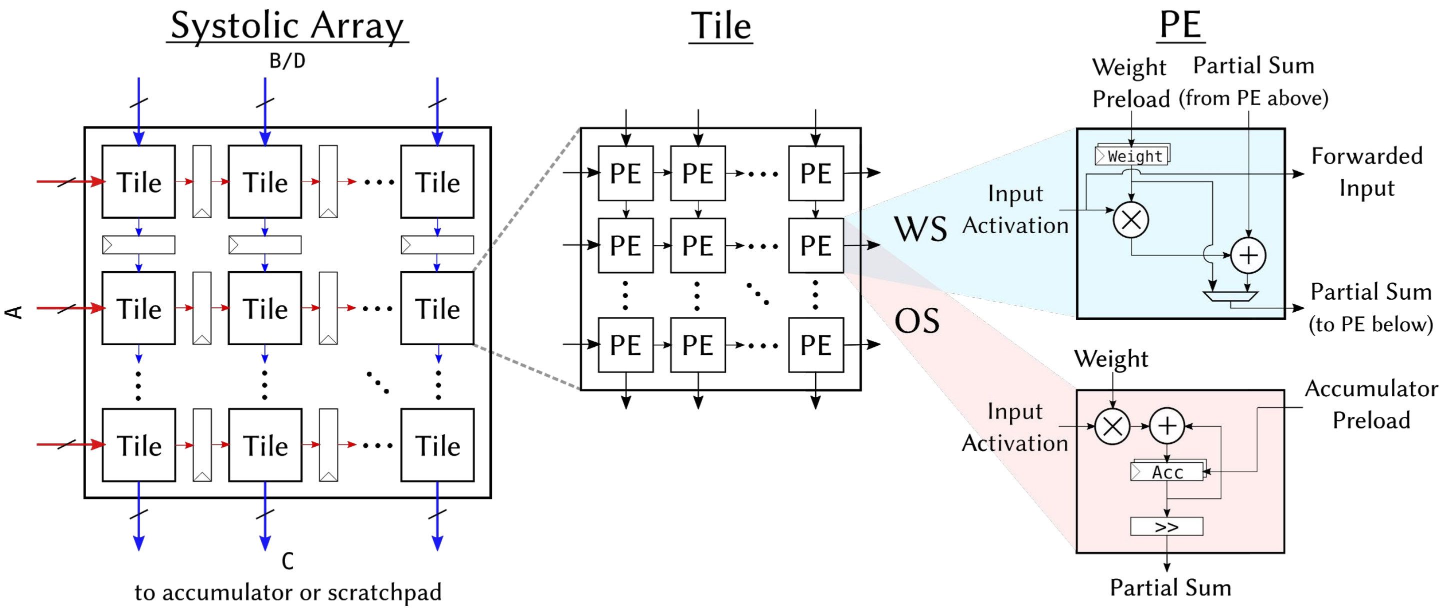
Comparing Reduced Bitwidth Formats for Machine Learning: Quantization vs Brain Float

Anson Tsai, Chufan Liang, Ryan Lund

Background

- Machine Learning (ML) is a massive industry
  - Spans across a wide range of applications from edge devices to warehouse scale computers
- Both ends of the ML spectrum face similar (yet subtly different) requirements with regards to power, area, accuracy, and latency.
- Research in acceleration has spanned from network architecture to domain specific accelerators
- A recent strategy that has received a lot of focus is reduced bitwidth representation for weight and activation storage as well as computation.

Gemmini Systolic Array Generator



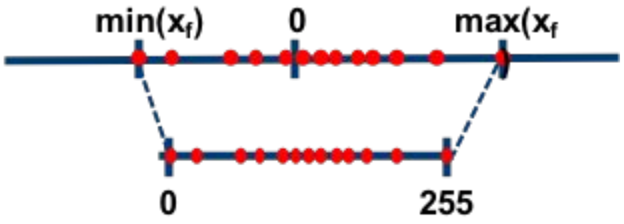
Reducing Bitwidth

- Using values that have a smaller footprint to minimally impact performance while decreasing resource usage
    - Higher throughput with the same bandwidth
    - Decreased memory usage (smaller model size)
    - Specialized Hardware
        - Smaller area
        - Less power
- Some ways to reduce bitwidth
    - Integer quantization
    - Novel numerical formats (e.g. Brain Float)

Integer Quantization

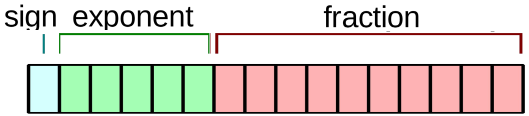
- Taking a set of floating point values and converting them into a lower bitwidth integer representation
  - Can be thought of generating step functions that map a range of fp32 to a range of smaller integer values
  - Requires the storage of scaling factors and algorithmic tweaks during the training process

FP32	Int8
0.0	0
0.5	1
1.0	2
1.5	3
2.0	4

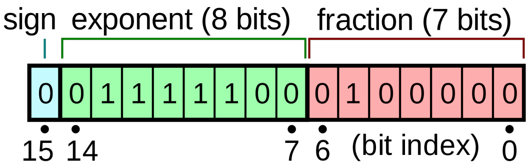


Brain Float (bFloat16)

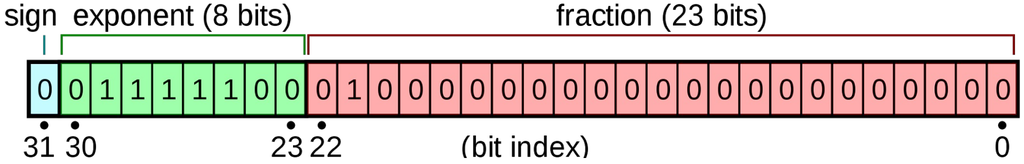
- Novel float format that has the footprint of float16 but dynamic range of a float32 (at cost of precision)
- Popularized by Google’s TPU accelerator
- Float 16



- bFloat 16



- Float 32



Images courtesy of Wikipedia

Methodology

- Modified Gemmini to generate systolic arrays of varying data types
  - Int8 (default)
  - Float16 and Float32 (thanks to help from Hasan Genc)
  - bFloat16 (required SoftFloat extension, modifications to Gemmini-Rocc-Tests)
    - Challenge, correctness checking (also seen with Float32)
- Calculated accuracies via modified LeNet-5 networks trained with Tensorflow
  - Inference is done by matrix multiplication, which we verified in our simulation tests
- Power and area statistics were gathered from Hammer post-par designs of the custom Rocket configs

Hypothesis

- Expect that both bf16 and int8 will have a lower power and area footprint than the fp32 baseline, at the cost of some accuracy
- Bf16 would have similar area and power usage to fp16
- Int8 will have the smallest overall footprint, and the lowest latency, making it will suited to highly constrained environments
- Bf16 will have a slightly larger overall footprint than int8, but less accuracy degradation, making it suited to accuracy sensitive environments

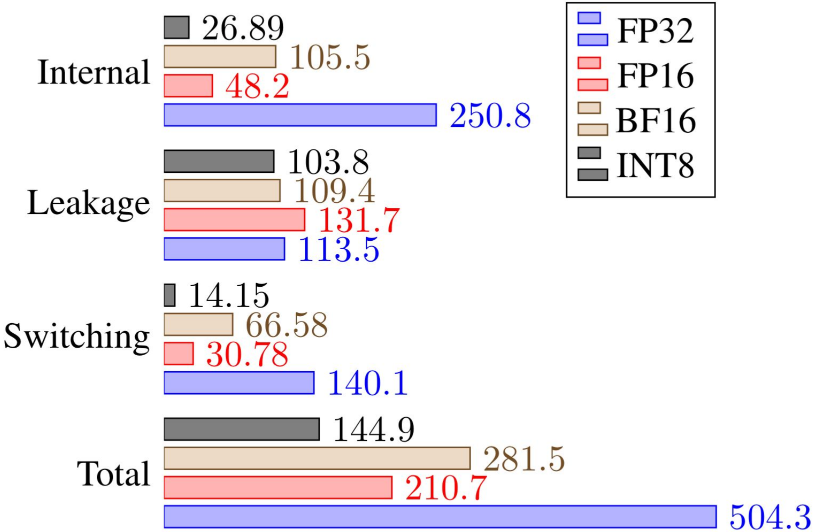
Results - Accuracy

Table 1: Accuracy and Latency Metrics

	FP32	**FP16**	**BF16**	**INT8**
Accuracy	53.11%	54.75%	53.38%	52.21%
Cycles / Inference (Total)	3328271	N/A	3339487	3325983
Cycles / Inference (Matmul)	741768	N/A	749682	737942

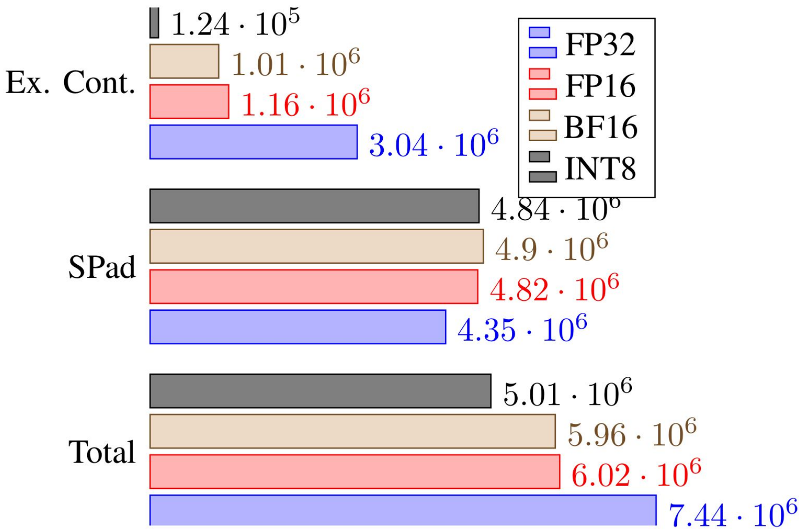
Results - Power

Fig. 1: Power (mW)



Results - Area

Fig. 2: Area (um^2)



Discussion

- Results largely confirm hypotheses: both reduced bitwidth methods resulted in lower area/power compared to baseline fp32
  - Int8 resulted in the greatest decrease of area/power, but had the lowest accuracy
    - Int8 is optimal for situations where resources are very limited and high accuracy is not required (e.g. mobile devices)
  - bf16 resulted in a considerable decrease of area/power, and had the highest accuracy
    - bf16 is suitable for situations where accuracy is more important and resources are not as constrained (e.g. warehouse computing servers)
    - Higher power usage when compared to fp16 indicates inefficiencies of b16 within Gemmini
- No meaningful comparisons of accuracy vs latency trade-offs due to similar values across all methods

Conclusion

- Reduced bitwidth techniques promise to further the optimization of ML hardware
- Significant reduction in resource utilization
- Comparable accuracy
- Enables more diverse applications
  - Expansion into smaller and more portable devices
- Trend of increased specialization
  - No longer one processor fits all