1. Appendix

1.1. Cost of Elementwise Operations

It is usually assumed that elementwise operations such as residual connections, BatchNorm layers, or DPReLU activation functions have negligible computation compared to Conv or linear layers. These elementwise layers are therefore excluded when calculating the compute cost with CPU64 or ACE. BNNs in the literature also adopted the same assumption [**?**, **?**, **?**].

However, the elementwise operations gradually become the cost bottleneck after the expensive Conv and linear layers are either binarized or quantized to ultra-low precision. Using PokeBNN-1.0x as a case study, we show the breakdown of the operation count in the table below.

There are 81.9 million elementwise additions and 38.4 million elementwise multiplications in PokeBNN-1.0x. These operations remain unquantized and are executed in bfloat16. If there is no optimization, these operations will have 30.8×10^9 ACE and will completely dominate the cost. Prior BNN works also suffer from the same issue [**?**, **?**, **?**]. Described as follows, there are several ways in which we can alleviate the problem.

Use integer arithmetic for additions. Most of the elementwise operations are additions. According to the energy numbers in Table 1 in the paper, additions in float16 are about 4 to 5 more expensive than additions in int32, and 13 to 23 times more expensive than int8. It would be challenging to represent all the ops in int8 without an automatic calibration algorithm, but int16 should have wide enough dynamic range.

Assume that no activation needs an absolute value bigger than 2^7, add the following operation can be applied on all the operations in the network:

$$Q_{16}(x) = \text{clip}(\text{round}(x \cdot 2^8) \cdot 2^{-8}, -(2^7 - 1), (2^7 - 1))$$

This implements a low-cost fixed-point arithmetic that has int16 additions. Only 16 adders are needed so the ACE metric is 16 for each addition.

Use narrow integer arithmetic for multiplication. High precision integer multiplication is costly. All of the multiplications in PokeBNN inference are fixed-point. The multiplication constants in `avg_ch` are 0.5 or 0.25, these can be implemented by shifts with negligible energy cost. In `avg_pool_3x3` the constant is $\frac{1}{9}$, but we could approximate it by $\frac{1}{8}$ and use a shift as well. It is the multiplications in BatchNorm, DPReLU and the final multiplication in `SE_4b` that represent the biggest challenge. In this appendix we assume that we assume that their 8-bit fixed-point representation of the fixed multipliers is accurate enough. The cost of int16 multiplied by int8 is ACE=$16 \cdot 8 = 128$.

Operation fusion. All the unquantized operations in our network are linear as they consist of multiplications and additions. The one exception is the slope selection in DPReLU. Nevertheless the size of the channelwise learnable parameters in DPReLU and BatchNorm is small. It is only a vector of length of the number of channels as they are shared between the spatial pixels. The same holds for the output of the SE layer. One can check that the consecutive operations of DPReLU and SE output multiplications can be always fused with the neighboring BatchNorm as all of them are affine functions. Only a separate addition for slope selection needs to be preserved. Both scaling multiplication in the quantization operators can be fused as well. The cost of fused operations can be ignored for the inference. In the case of fusion of SE final multiplication scaling, computation of the coefficients for final affine transformation has to be dynamic. Nevertheless, the cost of it is also negligible as it is shared between pixels and proportional only to the number of channels.

Smaller Pooling Kernel. Current average pooling layers for spatial downsampling uses 3×3 kernels. One could also replace `avg_pool_3x3` with `avg_pool_2x2`, a transformation similar to the first layer in PokeInit. It reduces the cost by 55%.

ACE estimation for pointwise operations. While the additions can be fused as described above, even without that the cost of them will be relatively low $ACE \leq 81.9 \cdot 10^6 \cdot 16 \approx 1.3 \cdot 10^9$.

Importantly, all the multiplications not in BatchNorms can be either replaced by shifting or fused into BatchNorm. The cost of the MULs in BatchNorm is $ACE = 17.9 \cdot 10^6 \cdot 128 = 2.3 \cdot 10^9$.

Overall the cost of elementwise operations is estimated to be $ACE = 3.6 \cdot 10^9$. While this is less than the cost of convolutions and the final classifier in PokeBNN-1.0x ($ACE = 4.2 \cdot 10^9$), it is not negligible and should be taken into account in future research.

1.2. ACE limitations

Memory reads and writes are potentially major energy sinks. The energy cost of writing or reading data to DRAM is 50-150 higher than to SRAM, but in case of the inference both model and activations usually can (or have to) fit in SRAM. We also note that if systolic array matrix multiplication circuit is big enough, for an inference, all inputs to the matmuls and convolutions have to be read exactly once. ACE does not estimate the cost of SRAM reads and writes into account. Also, as discussed in [**?**] the SRAM energy cost is not reducing as fast as the arithmetic cost between 45 nm to 7 nm chip manufacture process, thus making it incompatible with the goals of ACE.

Data movement are potentially major energy sinks. Energy-intensive chips are limited to 2D due to a need of

Layer Type	ADDs ($\times 10^6$)	MULs ($\times 10^6$)
`BatchNorm` addition	17.9	17.9
`DPReLU`	3×9.1	9.1
`ReshapeAdd: avg_ch`	5.4	1.7
`ReshapeAdd: avg_pool_3x3`	7.9	0.87
`ReshapeAdd:` residual addition (local + block)	8.8 + 5.5	0
`SE_4b: SpatialMean`	8.9	< 0.01
`SE_4b:` activation functions	< 0.01	< 0.01
`SE_4b:` final multiplication	0	8.8
Global pooling before classifier	0.1	< 0.01
Sum	81.9	38.4

Number of elementwise operations from unquantized layers.

high area-to-volume ratio for the sake of power delivery and heat removal. Inability to co-locate various circuits and memories may force existence of long wires, so called buses. This problem is not fundamental as tiled chip designs are being explored and well suited for 2D (also 3D and higher) image processing as each tile may be responsible for part of image and communicate only with its neighbors. Also energy use of chip's clock-tree can be thought of a variant of the same problem. Taking such design considerations into account is far beyond the scope of ACE.

ACE does not take into account "vector" operations. Activations, multiplication in BN, residual additions, bound scaling and clipping. All these operations are not taken into account neither by our application of ACE nor by most of the papers using CPU64. The analysis on the these operations is described in Appendix: Cost of Elementwise Operations.

Memory layout and padding. DNN operations that reshape or pad large patches of data are not modeled for energy. Networks like ShuffleNet[cite] might not be modeled well enough by ACE.

Analog and in-memory computing. It is unclear to the authors how to model hardware and networks that perform arithmetic operations using analog, in-memory computing circuits.