Hasso Plattner Institut
Digital Engineering · Universität Potsdam

Binary Neural Networks

PD Dr. Haojin Yang

Multimedia and Machine Learning Group

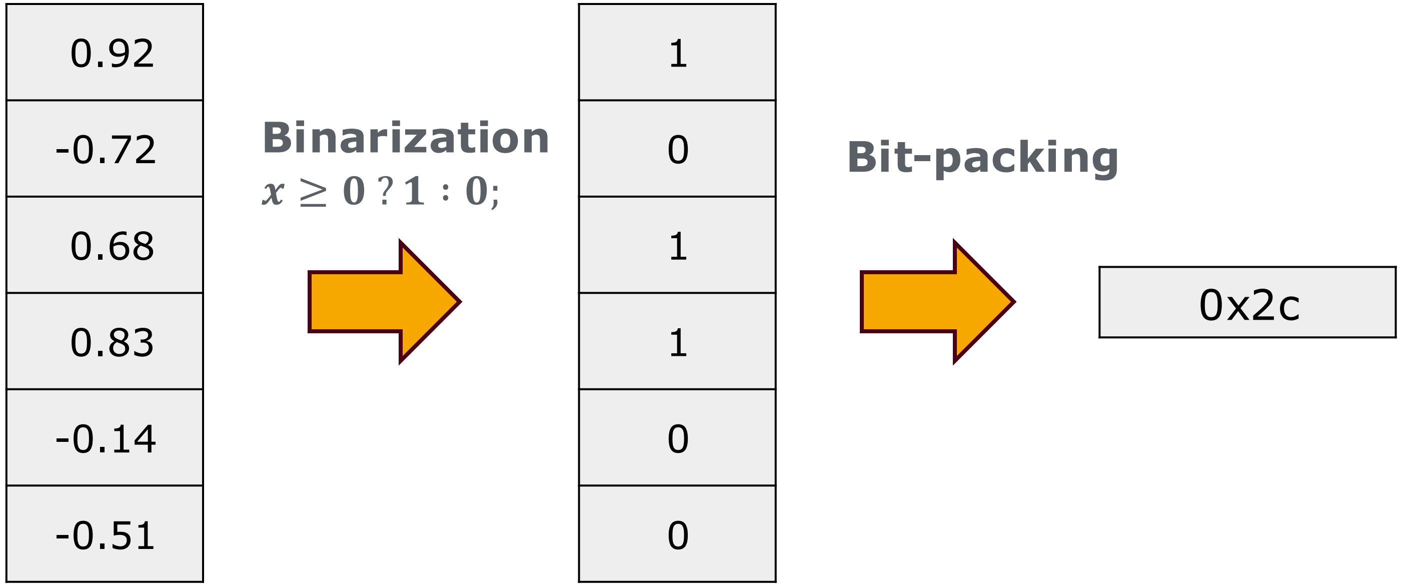
Hasso Plattner Institute

Design IT.
Create Knowledge.

www.hpi.de

Deep Learning with Binary Neural Networks

- The extreme case Binary Neural Networks only use 1-bit information (1 and 0) for weights and inputs instead of 32-bit floating point numbers



Benefits

- 32x smaller model size
- 58x speedup on CPU using bit-wise operators instead of arithmetic operations [1]
- up-to 1000x energy saving on FPGA, ASIC etc. [2]

[1] Rastegari, Mohammad, et al. "Xnor-net: Imagenet Classification using Binary Convolutional Neural Networks." European conference on computer vision. Springer, Cham, 2016.

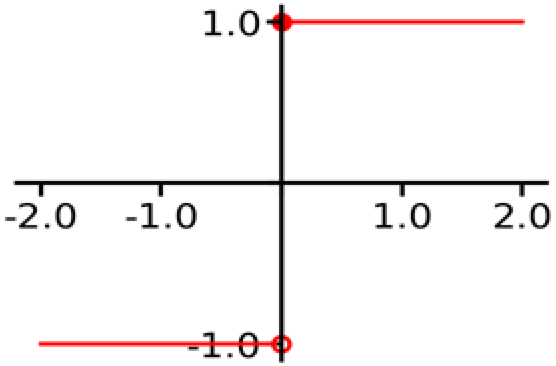
[2] Mishra, Asit, et al. "WRPN: Wide Reduced-Precision Networks." International Conference on Learning Representations. 2018.

How are Binary Neural Networks Trained?

- Sign function binarizes weights and inputs before convolution:

$$\mathrm{sign}(x) = \begin{cases} +1 & \text{if } x \geq 0, \\ -1 & \text{otherwise.} \end{cases}$$

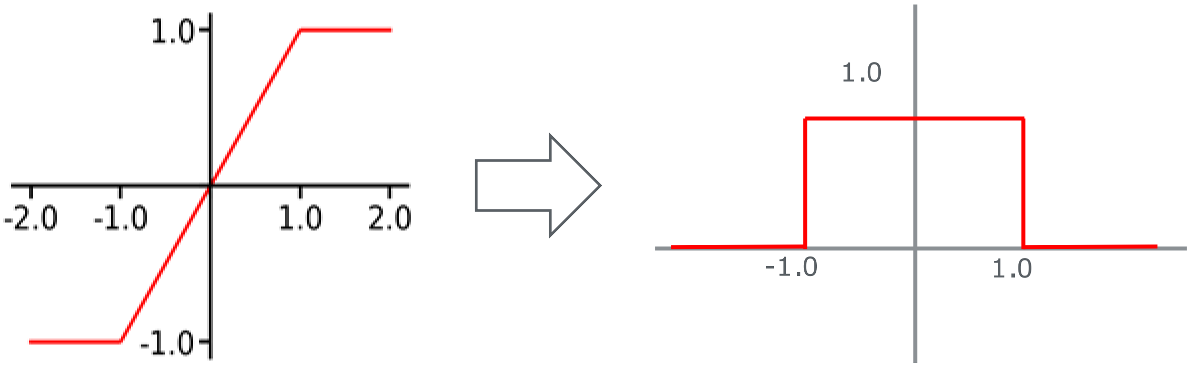
$$\text{Forward: } r_o = \mathrm{sign}(r_i)$$



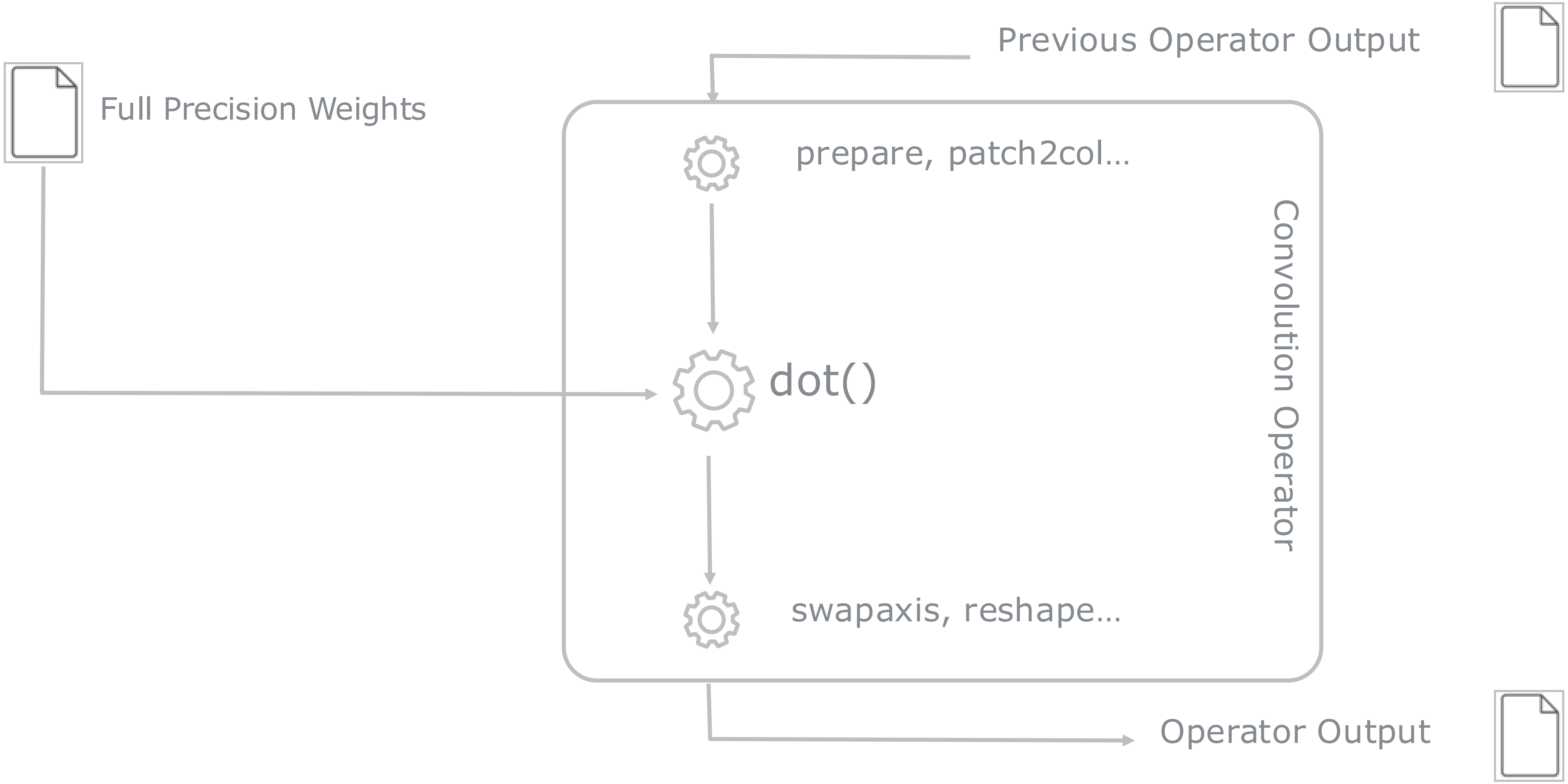
- Differentiating the sign function does not help for backpropagation! → **zero gradient**

$$\text{Backward: } \frac{\partial c}{\partial r_i} = \frac{\partial c}{\partial r_o} 1_{|r_i| \leq t_{cli}}$$

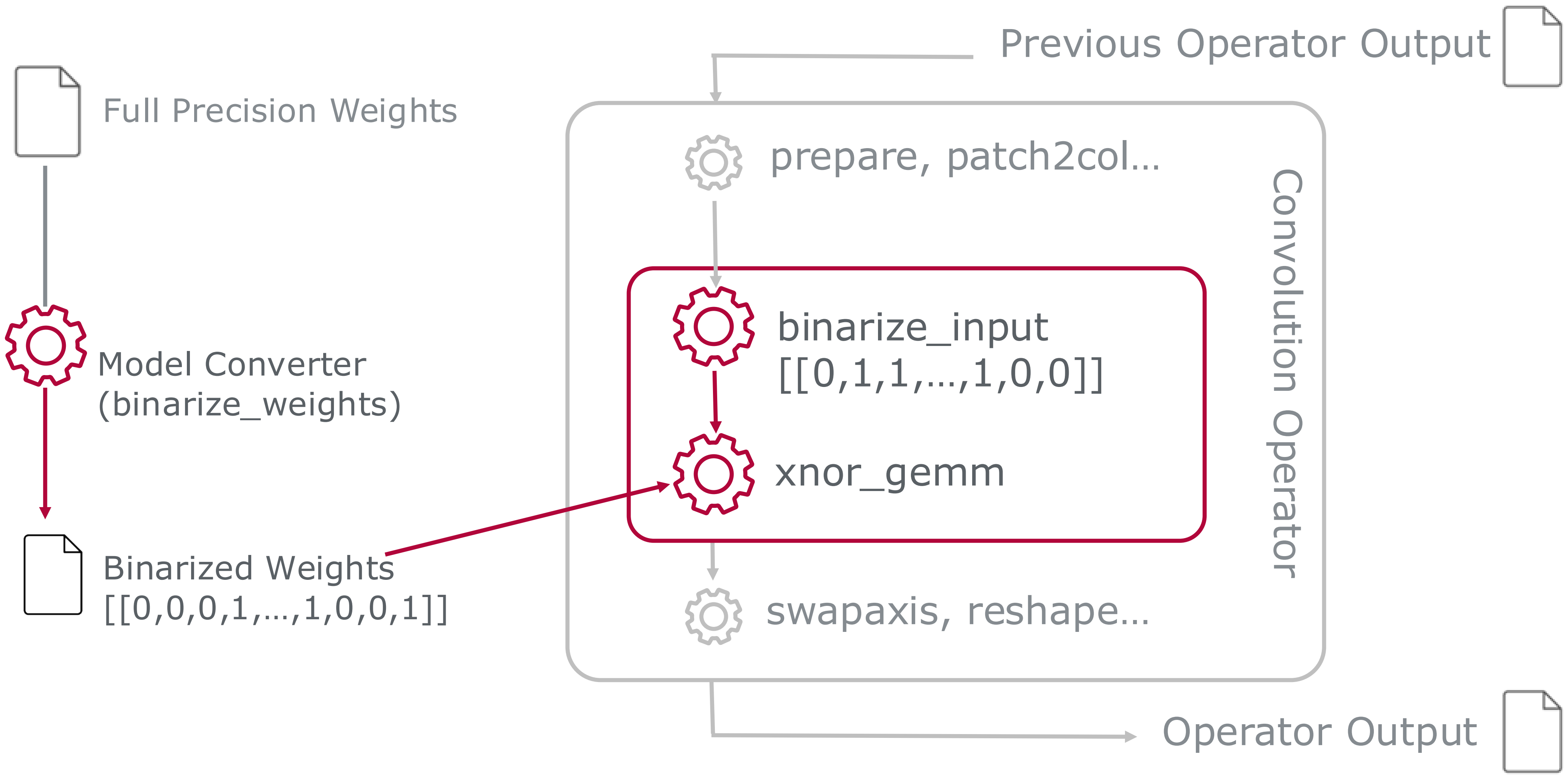
Straight Through Estimator



BNN Inference

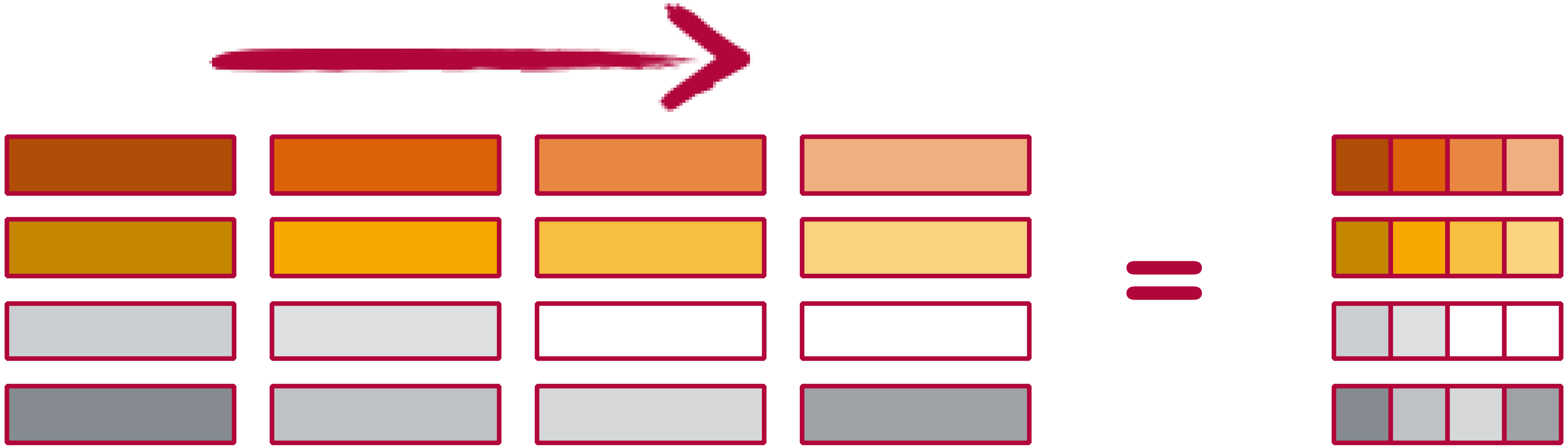


BNN Inference



BNN Inference

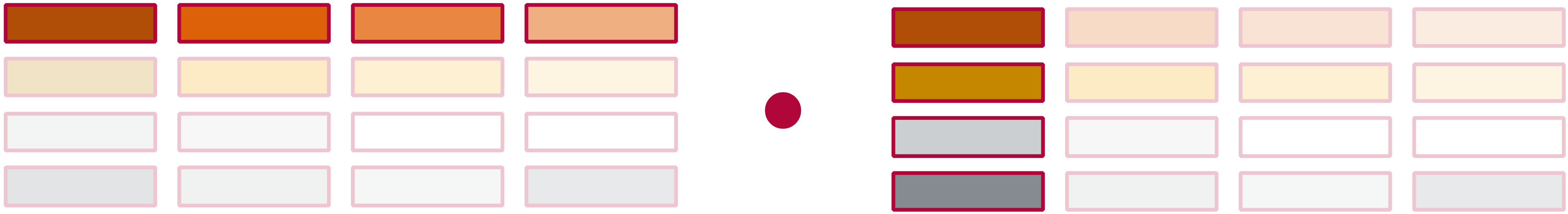
- Binarizing activations row-wise

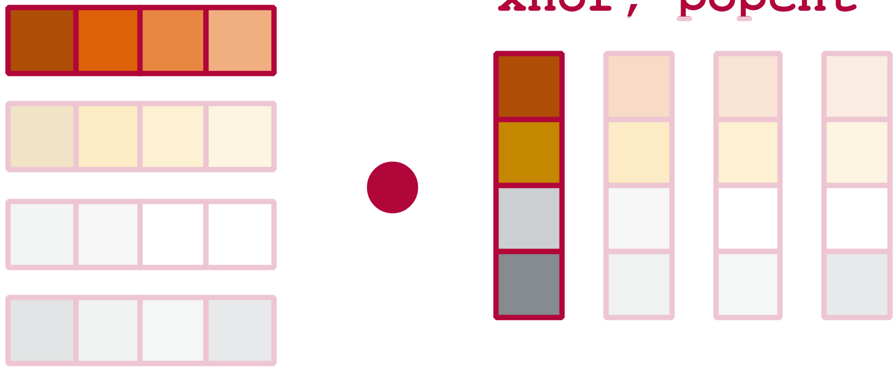


BNN Inference

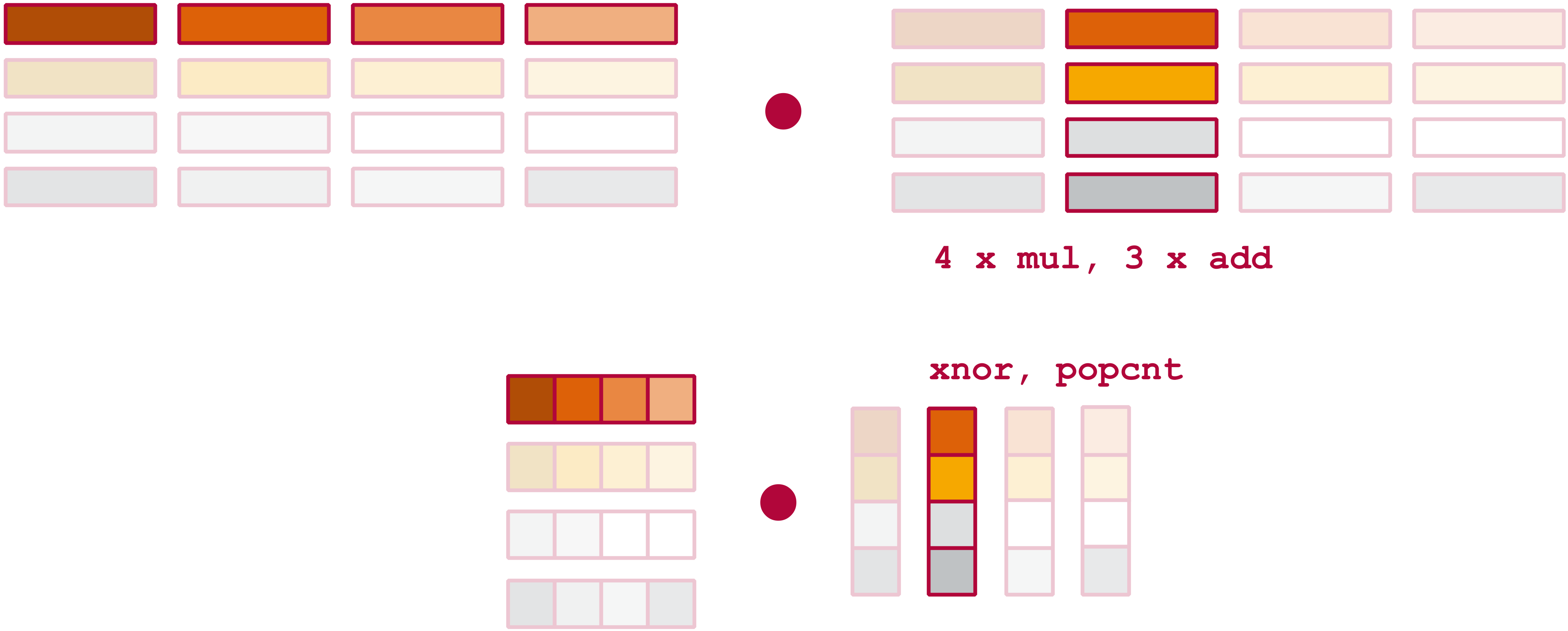
- Binarizing weights column-wise

BNN Inference

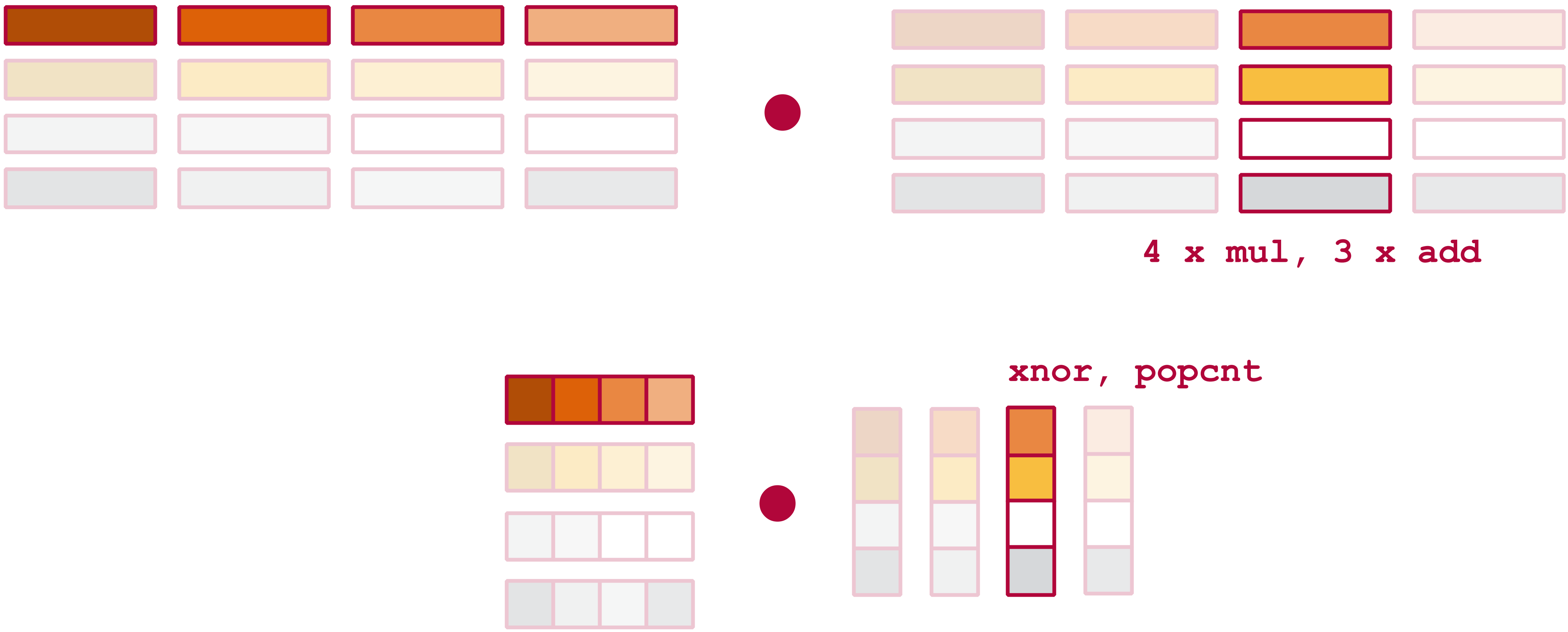




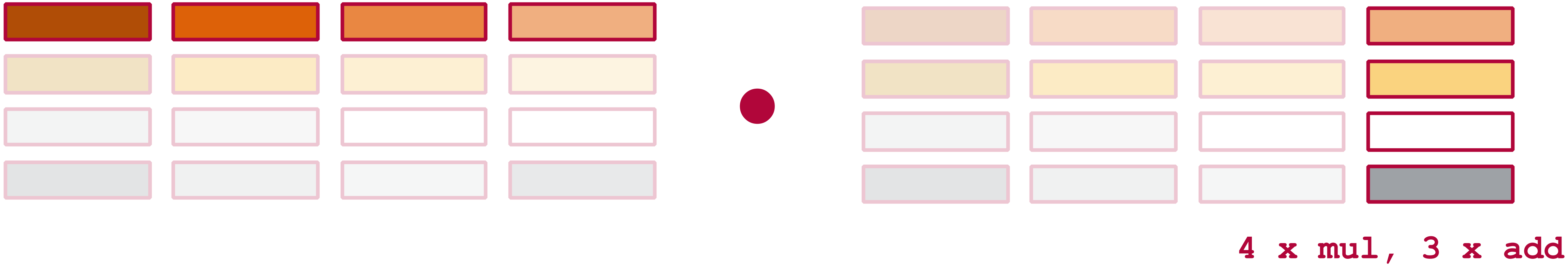
BNN Inference

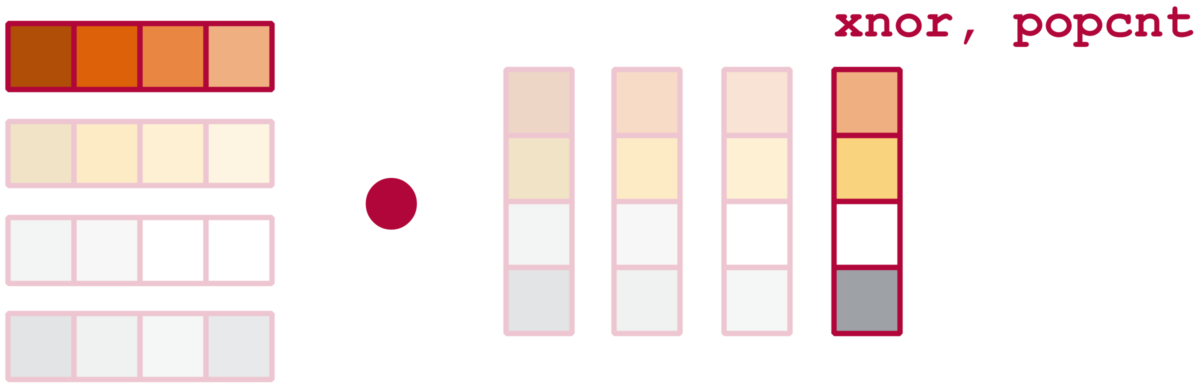


BNN Inference



BNN Inference





```
void xnor_gemm_baseline(int M, int N, int K,
                        BINARY_WORD *A, int lda,
                        BINARY_WORD *B, int ldb,
                        float *C, int ldc){
  for (int m = 0; m < M; ++m) {
    for (int k = 0; k < K; k++) {
      BINARY_WORD A_PART = A[m*lda+k];
      for (int n = 0; n < N; ++n) {
        C[m*ldc+n] += __builtin_popcountll(~(A_PART ^ B[k*ldb+n]));
      }
    }
  }
}
```

XOR

XNOR

POPCOUNT

```
64bit BINARY_WORD, using intrinsic __builtin_popcountll()

  9e961e:  jle    9e9659 <_ZN5mxnet2op8xnor_cpu25xnor_gemm_baseli
  9e9620:  movslq %r11d,%rax
  9e9623:  xor    %ecx,%ecx
  9e9625:  lea    (%r12,%rax,8),%r9
  9e9629:  nopl   0x0(%rax)
  9e9630:  mov    %r8,%rax
  9e9633:  xor    (%r9,%rcx,8),%rax
  9e9637:  vxorps %xmm0,%xmm0,%xmm0
  9e963b:  not    %rax
  9e963e:  popcnt %rax,%rax
```

Summary

- Quantization aware neural network training
- Fundamentals of binary neural network training and inference