AMLNN Convolution Acceleration Tips

1. Padding has no software overhead

Convolution padding is supported by NN hardware with no software overhead. The padding attribute can be specified individually for all four sides [Top, Bottom, Left, Right]. The maximum pad size is 7.

2. Use pooling instead of stride

In order to simplify hardware design, the NN convolution core does not support stride natively, except for stride of 2; instead it uses TP to perform a reshuffle operation before feeding data into NN. While reshuffle by TP is extremely fast, it still adds a little overhead to the overall compute time. Whenever possible for strides not of 2, design the convolution kernels without stride and use pooling operations to achieve feature map shrinking.

3. Use Depth convolution

We recommends that the convolution output channel number be a multiple of the convolution core number to ensure all the convolution cores are occupied, and thereby achieving higher hardware utilization. This is especially important for configurations with large number of cores, such as -Q (Quad 4 conv core).

4. Optimal Kernel Size is 3x3

Convolution cores can support a large range of kernel sizes. The minimum supported kernel size is [1] and maximum is [11 * stride - 1].

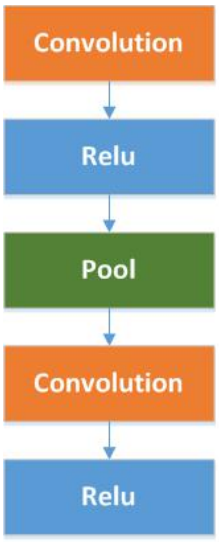
The NN Engine performs most optimally when the Convolution kernel size is 3x3, under which the highest MAC utilization can be achieved.

Non-square kernels are also supported, but with some computation overhead.

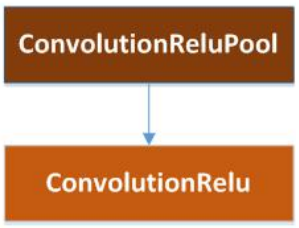
5. Fused Operations Reduce Overhead

The Convolution core can fuse ReLU and MAX Pooling operations on the fly to further reduce computation and bandwidth overhead. A ReLU layer following a Convolution layer will always be fused, while MAX pooling layer fusion has the following restrictions, Max pooling must

- have a pool size of 3x3 or 2x2, and stride of 2
- 2x2 pooling must have an even input size and no padding
- 3x3 pooling must have odd input size which is not one and no padding
- Horizontal input size must be less than 64 (8-bit mode) or 32 (16-bit mode) if pool size is 3x3



Original Graph

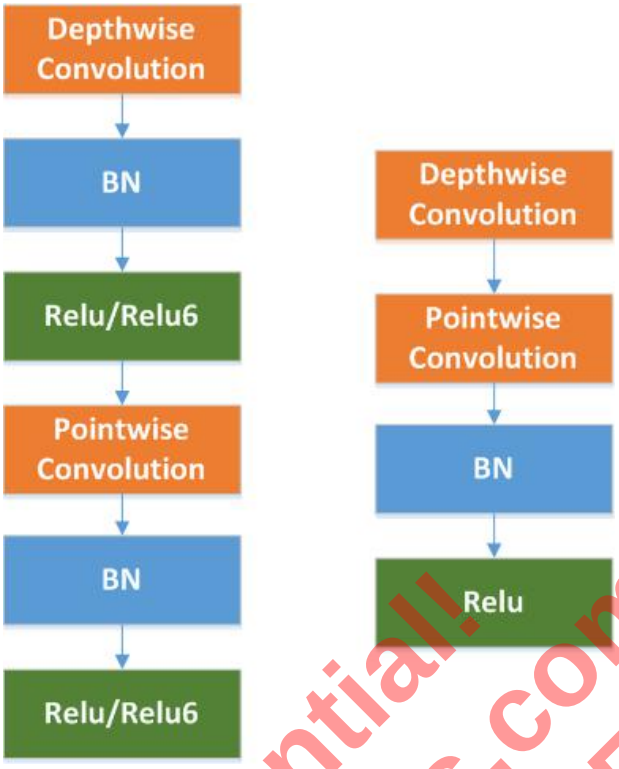


Fused Graph

6. Depthwise Convolutions

Both regular 2D and Depthwise convolutions are supported, while 2D convolutions perform more optimally. Since Depthiwise Convolution-specific structure makes it less friendly to quantized model. We recommend using 2D convolution whenever possible when designing your network.

If you must use a Depthwise convolution, we recommend following the rules below that can improve the accuracy of the quantized model:



- Change the activation function RELU6 to RELU
- When training, for the Depthwise convolution, use L2 normalize on its weight
- Remove the BN layer and activation layer of the Depthwise convolution layer

7. Take advantage of Hardware's Sparse Matrix Support

Modern Neural-Networks are known to be over parameterized and have much redundancy in their design. Pruning a network to be sparse has been proven to reduce computation overhead while maintaining accuracy.

AMLNN hardware is designed to support sparse matrix operations efficiently by skipping computations and memory fetches on zero values. The sparsity level can be fine grain down to individual weights. Designing a sparse network to take advantage of this technology could further improve performance on AMLNN.