Paralle Algorithm for CNN Inference and its Automatic Synthesis

Yukio Miyasaka, Xinpei Zhang, Takashi Matsumoto, and Masahiro Fujita
Systems Design Lab (d.lab), School of Engineering, The University of Tokyo, Tokyo, Japan

Abstract
Recently, Convolutional Neural Network (CNN) has surpassed conventional methods in the field of image processing. This demonstration shows a new algorithm to calculate CNN inference using processing elements arranged and connected based on the topology of the convolution. They are connected in mesh and calculate CNN inference in a systolic way. The algorithm performs the convolution of all elements with the same output feature in parallel. We demonstrate a method to automatically synthesize an algorithm, which simultaneously performs the convolution and the communication of pixels for the computation of the next layer. We show with several sizes of input layers, kernels, and strides and confirmed that the correct algorithms were synthesized. The synthesis method is extended to the sparse kernel. The synthesized algorithm requires fewer cycles than the original algorithm. There were the more chances to reduce the number of cycles with the sparser kernel.

Booth Demonstration
Figure 1 shows two rectangular parallelepipeds representing the input layer and the kernel for a CNN calculation. A pixel at \((i, j, k)\) of the input layer is set at the coordinate \((i, j, k)\) and the size of the parallelepiped of the input layer is \((I_X + K_X - 1) \times (I_Y + K_Y - 1) \times I_Z\). To calculate pixels near the edge in the output layer, the size of input layer is extended from \(I_X \times I_Y \times I_Z\) to \((I_X + K_X - 1) \times (I_Y + K_Y - 1) \times I_Z\) and the new pixels where the row is more than \(I_X\) or the column is more than \(I_Y\) are set to 0. It uses processing elements (PEs) as many as the pixels of the extended input layer. They are aligned in the same topology as pixels of the input layer. A PE has some registers, a MAC operation unit and a communication unit, which can send one data to an adjacent PE and receive one data from an adjacent PE at once. We propose to transform the 3-dimensional computation into 2-dimensional computation. The \((I_X + K_X - 1) \times (I_Y + K_Y - 1) \times I_Z\) rectangular parallelepiped of PEs is converted into the \((I_X + K_X - 1) \times (I_Y + K_Y - 1)\) rectangular of PEs by merging the PEs at \((i, j, k)\) for all \(k\) to be a PE at \((i, j)\). We define a cycle as the time required to perform one MAC operation. If the communication is not completely hidden in the multiplication, the excess time of communication is also included in a cycle.

In the original algorithm with the rectangular architecture, a partial product of each pixel of the output layer is communicated among PEs, while each pixel of the input layer is fixed to a PE. We modify this algorithm such that pixels of the input layer are communicated among PEs and each partial product is kept in a PE [1]. The goal of the synthesis is to generate the algorithm which takes the fewest cycles. If the stride size is 1, it takes just \(K_X \times K_Y\) cycles for each partial product is kept in a PE [1].

was reduced around 25 % in the best case. The synthesis for 8 \(\times\) 8 input layer, 4 \(\times\) 4 kernel, and 2 stride timed out of 1 day.

References