ECE408/CS483/CSE408 Spring 2020
Optimizing Convolution Layers

Contributed by Carl Pearson
pearson@Illinois.edu

Objective
- To understand how unrolling input X can improve performance for convolution layers on GPUs.

Reorganize Input X for Convolution

Consider one image b. One thread computes one value of Y for one feature.

Reorganize Input X for Convolution

Each thread needs a K×K block of X for each channel.

Note: Likely to be larger than X, since values are replicated.
Implementing a Convolution Layer with Matrix Multiplication

Simple Matrix Multiplication

Each product matrix element is an output feature map pixel.

This inner product generates element 0 of output feature map 0.

Tiled Matrix Multiplication 2x2 Example

Each block calculates one output tile – 2 elements from each output map
Each input element is reused 2 times in the shared memory

Tiled Matrix Multiplication 2x4 Example

Each block calculates one output tile – 4 elements from each output map
Each input element is reused 2 times in the shared memory
**Efficiency Analysis: Total Input Replication**

- Replicated input features are shared among output maps
  - There are $H_{out} \times W_{out}$ output feature map elements
  - Each requires $K \times K$ elements from the input feature maps
  - So, the total number of input element after replication is $H_{out} \times W_{out} \times K \times K$ times for each input feature map
  - The reduction factor of the tiled algorithm is $K$
  - The total number of input feature map element accesses was $(TILE\_WIDTH)^2$
  - Each input tile has $(TILE\_WIDTH+K-1)^2$
  - Each output tile has $TILE\_WIDTH$
  - The total number of elements in each original input feature map is $(H_{out}+K-1) \times (W_{out}+K-1)$

**Properties of the Unrolled Matrix**

- Each unrolled column correspond to an output feature map element
- For an output feature element $(h,w)$, the index for the unrolled column is $h \times W_{out} + w$ (linearized index of the output feature map element)

**Analysis of a Small Example**

- $H_{out} = 2$
- $W_{out} = 2$
- $K = 2$
- There are 3 input maps (channels)
- The total number of input elements in the replicated (“unrolled”) input matrix is $3 \times 2 \times 2 \times 2$
- The replicating factor is $(3 \times 2 \times 2 \times 2)/(3 \times 3) = 1.78$
Properties of the Unrolled Matrix (cont.)

- Each section of the unrolled column corresponds to an input feature map.
- Each section of the unrolled column has \( k^2 \) elements (convolution mask size).
- For an input feature map \( c \), the vertical index of its section in the unrolled column is \( c^2 k^2 \) (linearized index of the output feature map element).

To Find the Input Elements

- For output element \((h, w)\), the base index for the upper left corner of the input feature map \( c \) is \((c, h, w)\).
- The input element index for multiplication with the convolution mask element \((p, q)\) is \((c+h+p, w+q)\).

Function to generate “unrolled” \( X \)

```c
void unroll(int B, int C, int H, int W, int K, float* X, float* X_unroll) {
    int H_out = H - K + 1;
    int W_out = W - K + 1;
    for (int b = 0; b < B; ++b) {
        for (int p = 0; p < K; ++p) {
            for (int q = 0; q < K; ++q) {
                for (int h = 0; h < H_out; ++h) {
                    for (int w = 0; w < W_out; ++w) {
                        int w_base = h * W_out + w;
                        int h_unroll = w_base + p * K + q;
                        X_unroll[b, h_unroll, w_unroll] = X[b, c, h + p, w + q];
                    }
                }
            }
        }
    }
}
```
### Implementation Strategies for a Convolution Layer

- **Baseline**
  - Tiled 2D convolution implementation, use constant memory for convolution masks

- **Matrix-Multiplication Baseline**
  - Input feature map unrolling kernel, constant memory for convolution masks as an optimization
  - Tiled matrix multiplication kernel

- **Matrix-Multiplication with built-in unrolling**
  - Perform unrolling only when loading a tile for matrix multiplication
  - The unrolled matrix is only conceptual
  - When loading a tile element of the conceptual unrolled matrix into the shared memory, use the properties in the lecture to load from the input feature map

- **More advanced Matrix-Multiplication**
  - Use joint register-shared memory tiling