Flynn's Taxonomy

SISD Single instruction stream Single data stream	(SIMD) Single instruction stream Multiple data stream
MISD Multiple instruction stream Single data stream	(MIMD) Multiple instruction stream Multiple data stream

classic von Neumann

Does it make sense?

Yes, systolic array.

1

Outline

- What are systolic arrays?
- How systolic arrays work?
- Where are they used?
- How people optimize the algorithm to be mapped?
- How are they related with CNN/DNN?

2

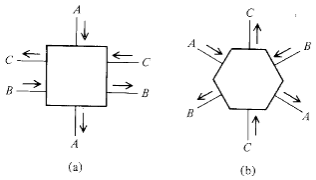
What are systolic arrays?

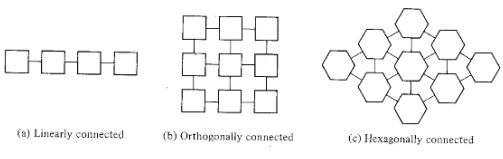
- Parallelism: A parallel computer architecture (a VLSI design prototype)
- Homogeneity: A homogeneous network of tightly coupled data processing units (DPUs).
- Pipelining: A systolic algorithm relies on data from different directions arriving at cells in the array at regular intervals

3

What are systolic arrays?

- Cells
  - Computation (e.g. C ← C+A*B)
  - Local memory (e.g. register Ra, Rb, Rc)
- Connections
  - Linear
  - Mesh
  - Hex





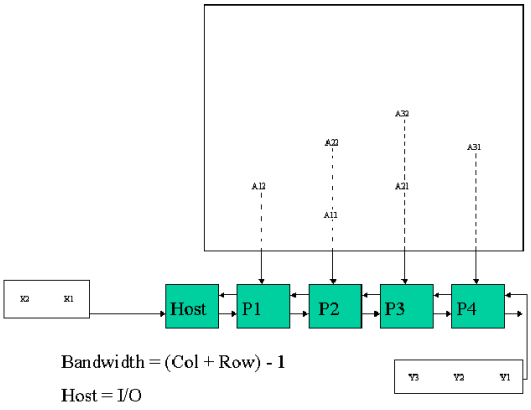
4

How systolic arrays work?

- A simple example

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

$$y_1 = (a_{11} * x_1) + (a_{12} * x_2)$$
$$y_2 = (a_{21} * x_1) + (a_{22} * x_2)$$
$$y_3 = (a_{31} * x_1) + (a_{32} * x_2)$$

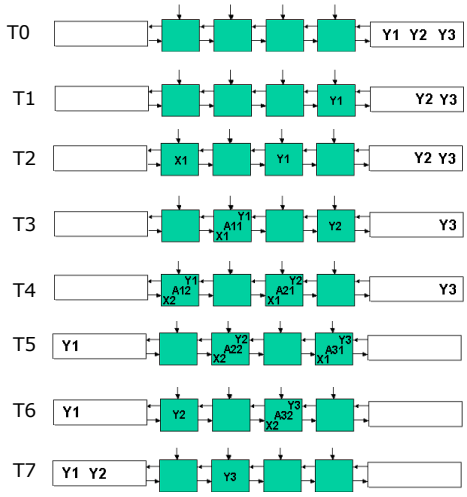


Bandwidth = (Col + Row) - 1

Host = I/O

5

$$y_1 = (a_{11} * x_1) + (a_{12} * x_2)$$
$$y_2 = (a_{21} * x_1) + (a_{22} * x_2)$$
$$y_3 = (a_{31} * x_1) + (a_{32} * x_2)$$



6

Matrix-vector multiplication

- $A = a_{ij}$
- $x = (x1, x2, \ldots, xn)$
- Suppose the band width is $w = p+q-1$

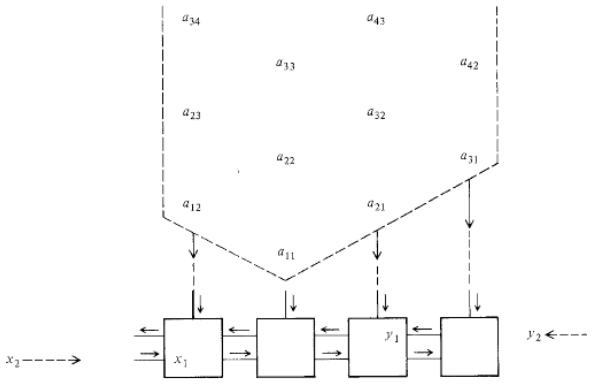
$$
\begin{aligned}
y_i^{(1)} &= 0, \\
y_i^{(k+1)} &= y_i^{(k)} + a_{ik}x_k, \\
y_i &= y_i^{(n+1)}.
\end{aligned}
$$

$$
\begin{bmatrix}
a_{11} & a_{12} & & & & \\
a_{21} & a_{22} & a_{23} & & & 0 \\
a_{31} & a_{32} & a_{33} & a_{34} & & \\
 & a_{42} & a_{43} & a_{44} & a_{45} & \\
 & & a_{53} & & \ddots & \\
 & 0 & & \ddots & &
\end{bmatrix}
\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \end{bmatrix}
=
\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ \vdots \end{bmatrix}
$$

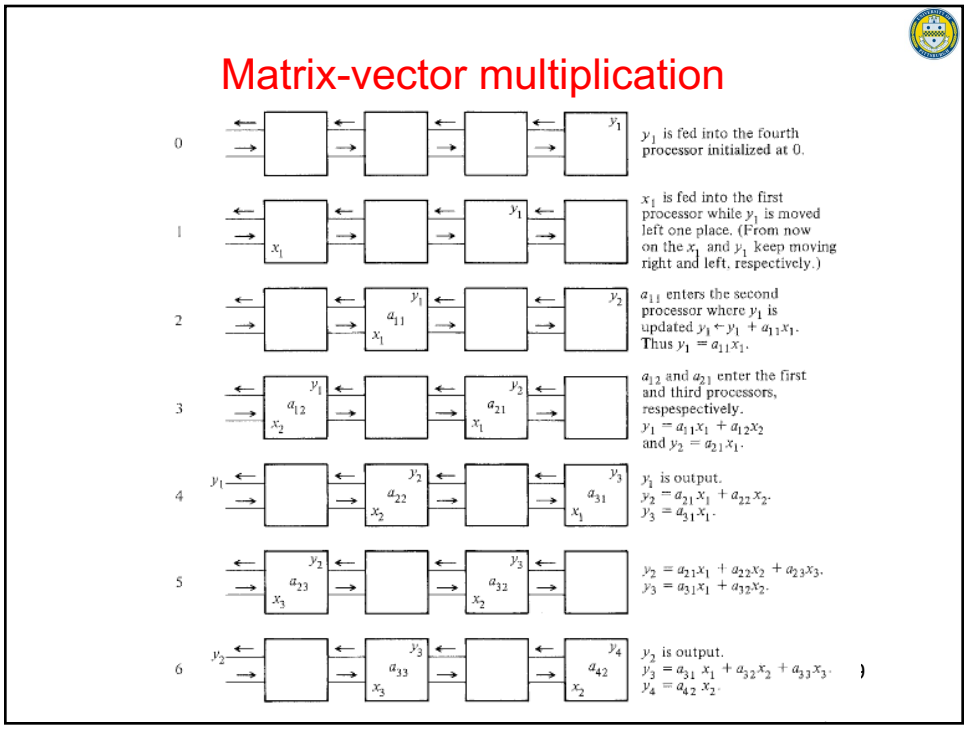
(p, q, A, x, y)

7

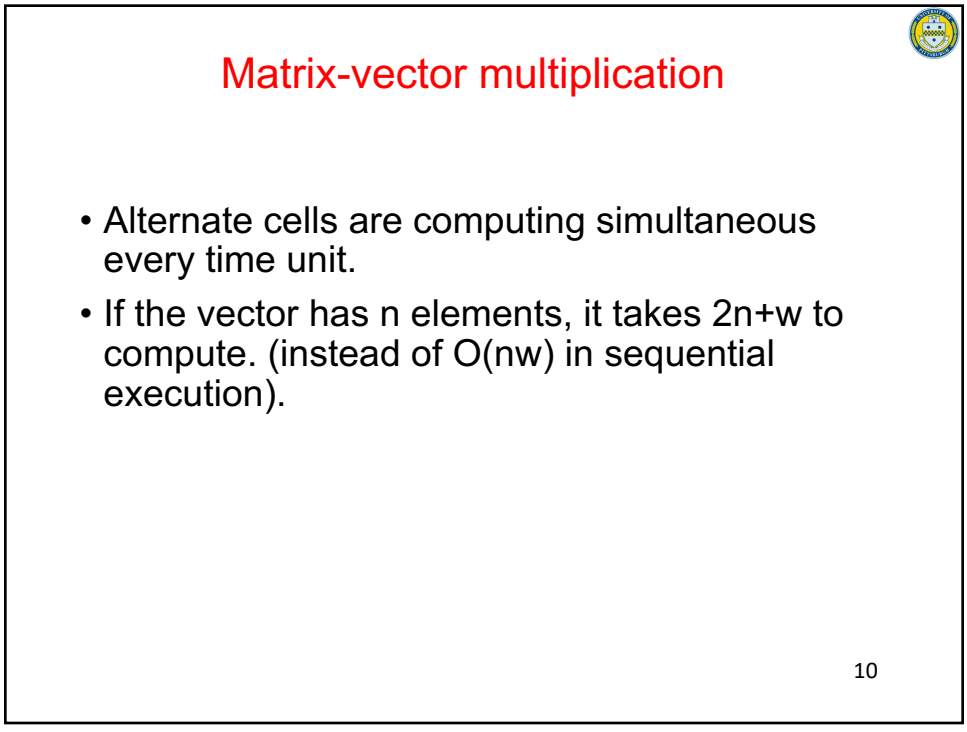
Matrix-vector multiplication



8



Matrix-vector multiplication



Matrix-vector multiplication

- Alternate cells are computing simultaneous every time unit.
- If the vector has n elements, it takes 2n+w to compute. (instead of O(nw) in sequential execution).

10

Matrix multiplication

$$\begin{bmatrix} a_{11} & a_{12} & & & 0 \\ a_{21} & a_{22} & a_{23} & & \\ a_{31} & a_{32} & a_{33} & a_{34} & \\ & a_{42} & \ddots & & \\ 0 & & & & \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} & & 0 \\ b_{21} & b_{22} & b_{23} & b_{24} & \\ & b_{32} & b_{33} & b_{34} & b_{35} \\ & & b_{43} & \ddots & \\ 0 & & & & \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} & 0 \\ c_{21} & c_{22} & c_{23} & c_{24} & \\ c_{31} & c_{32} & c_{33} & c_{34} & \\ c_{41} & c_{42} & & \ddots & \\ 0 & & & & \end{bmatrix}$$

A B C

11

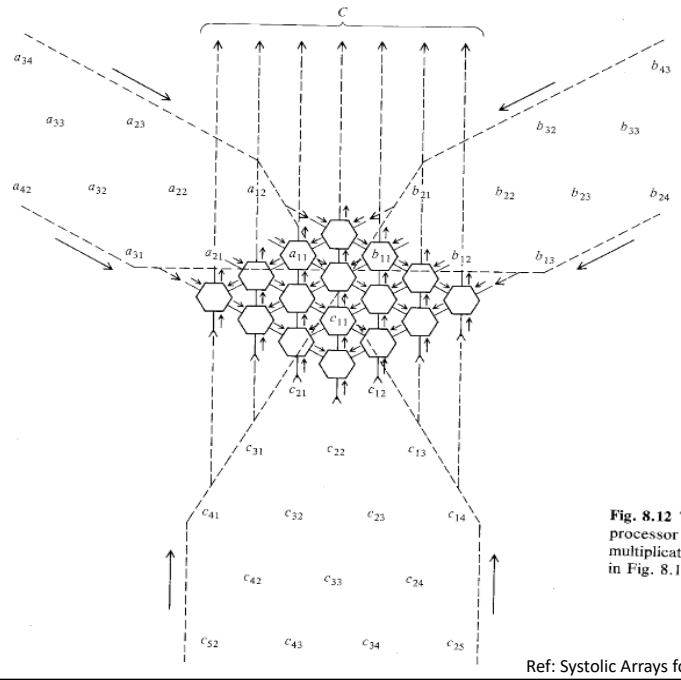


Fig. 8.12 The hex-connected processor array for the matrix multiplication problem shown in Fig. 8.11.

Ref: Systolic Arrays for VLSI. By Kung. 1978

Matrix multiplication

- If A and B are n*n band matrices of band w1 and w2, the network of w1*w2 can pipeline the matrix multiplication in 3n+min(w1,w2) units of time.

13

Where are they used?

- Matrix-vector multiplication
- Matrix multiplication
- LU decomposition
- Convolution
- Filter
- Discrete Fourier Transform (DFT).

14

Optimizations

- Not all the algorithms are naturally suited to systolic arrays.
- Simple optimization potential: observing idle PEs during execution
- Answer???
- We need help from compilers.

15

Optimizations

- Synthesizing systolic arrays from recurrence equations with linear dependences. – Sanjay et.al. 1986
- Recurrence equation:
  - e.g., A[i,j,k] = 2*B[i, j+2, k-1]
  - p=[i,j,k], q=[i,j+2, k-1], p-q = [0,-2,1]
  - With another dimension t (time), the distance is constant.
- What if p-q is relied on i (or, j, k).
- Transformation and then map to systolic arrays.

16

Optimizations

- A Systolic Array Parallelizing Compiler – 1990
- Problem: cells has limited memory, cannot accommodate huge data arrays.
- Solutions: distributed arrays based on data relations.
  - Given a loop, slice x is related with y if x and y are referenced in the same loop iteration.
  - Example,

```
DARRAY float[500] A[500];
float row[500];
DO(k = 0, n){
  DO(i = k+1, n) A[k][i] = -A[k][i]/A[k][k];
  DO(i = k+1, n) row[i]=A[k][i];
  DO(j = k+1, n){
    DO(i = k+1, n){
      A[j][i] = A[j][i]+A[j][k]*row[i];
    }
  }
}
```

17

Optimizations

- Loop distribution
  - Intra-loop parallelism
  - Inter-loop parallelism
  - Delayed synchronizations
- Data distribution
  - Blocking
  - Interleaving

19

How are they related with CNN/DNN?

A: First we need to understand DNN computation pattern and memory access pattern

20

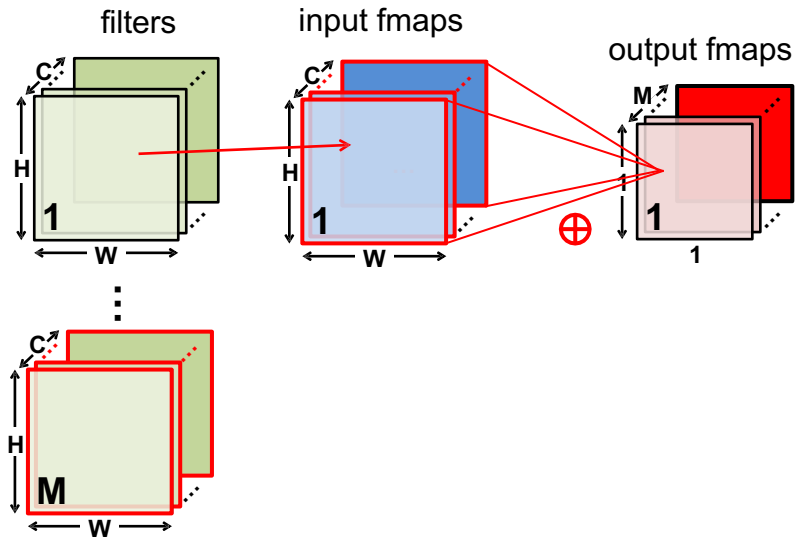
For DNN/CNN basics: https://eyeriss.mit.edu/
Slides credits to MIT eyeriss project.

DNN Basics

- Convolution layer
- Fully connected layer

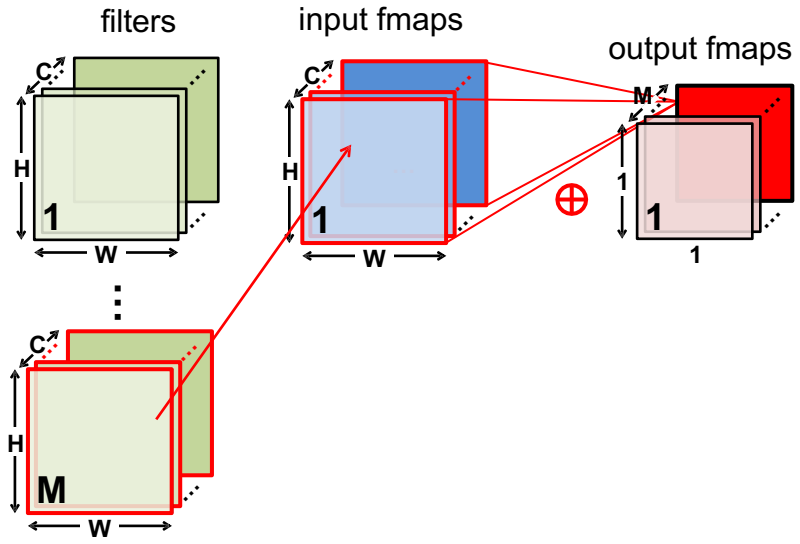
21

Fully-Connected (FC) Layer



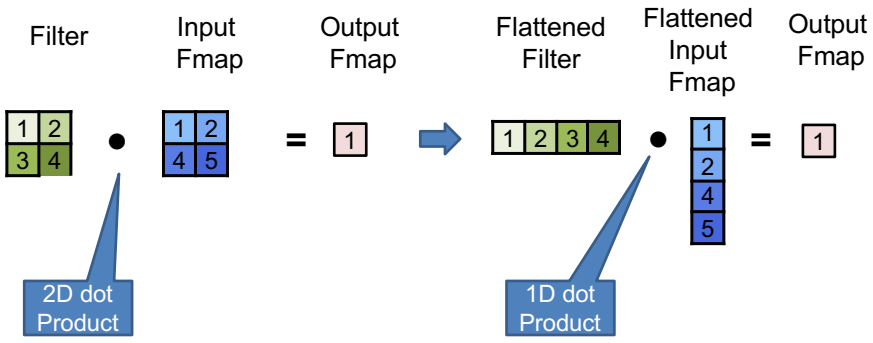
22

Fully-Connected (FC) Layer



23

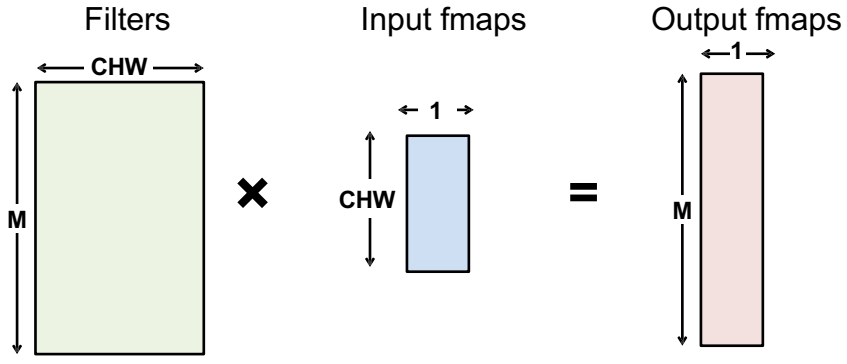
Flattened 2D Dot Product



24

Fully-Connected (FC) Layer

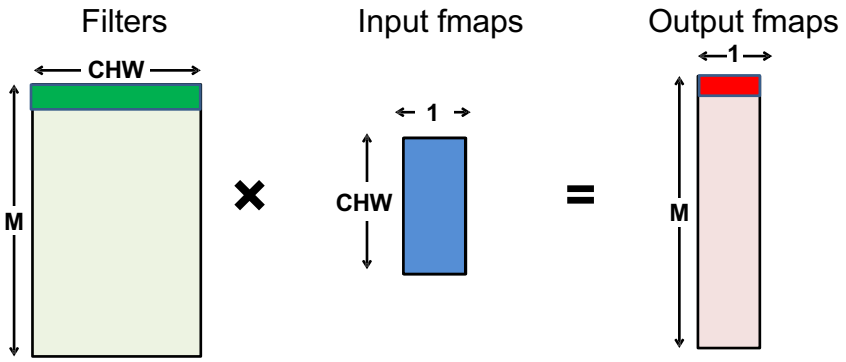
- Matrix–Vector Multiply:
  - Multiply all inputs in all channels by a weight and sum



25

Fully-Connected (FC) Layer

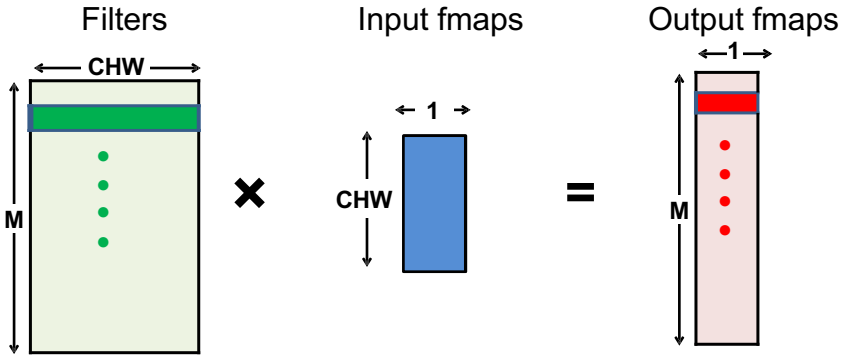
- Matrix–Vector Multiply:
  - Multiply all inputs in all channels by a weight and sum



26

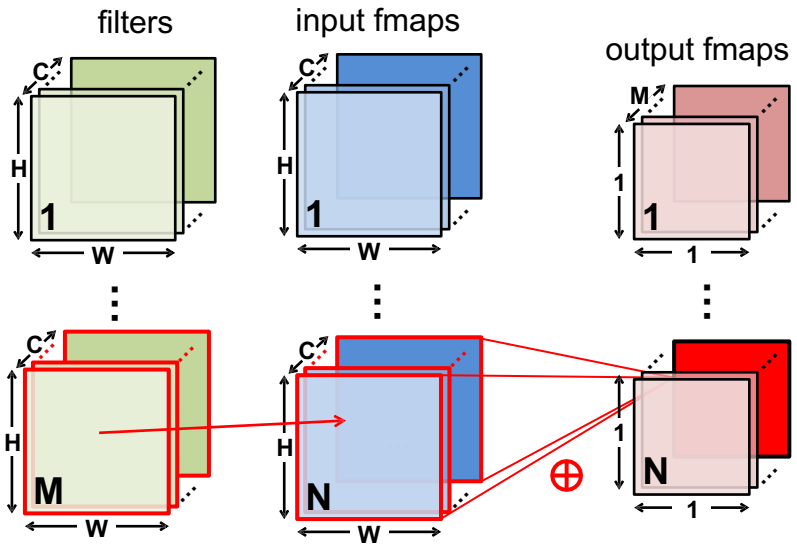
Fully-Connected (FC) Layer

- Matrix–Vector Multiply:
  - Multiply all inputs in all channels by a weight and sum



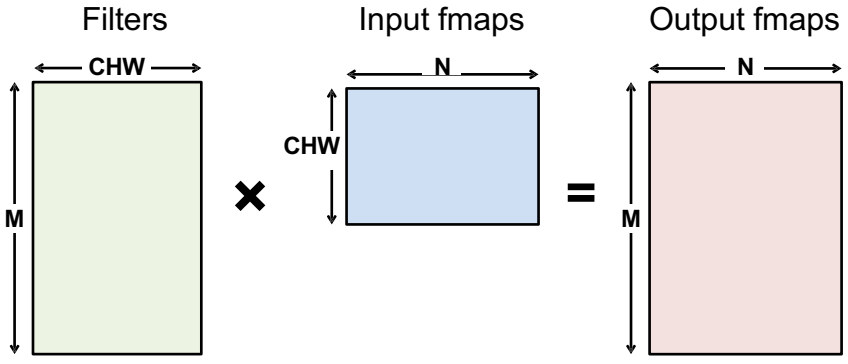
27

Fully-Connected (FC) Layer



28

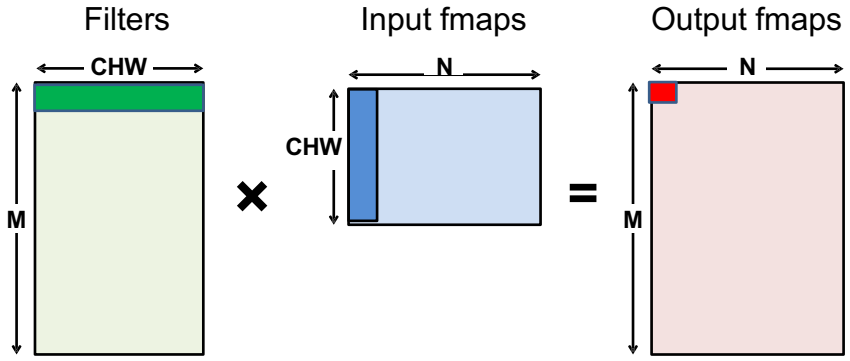
Flattened Fully-Connected Layer



- After flattening, having a batch size of N turns the matrix-vector operation into a matrix-matrix multiply

29

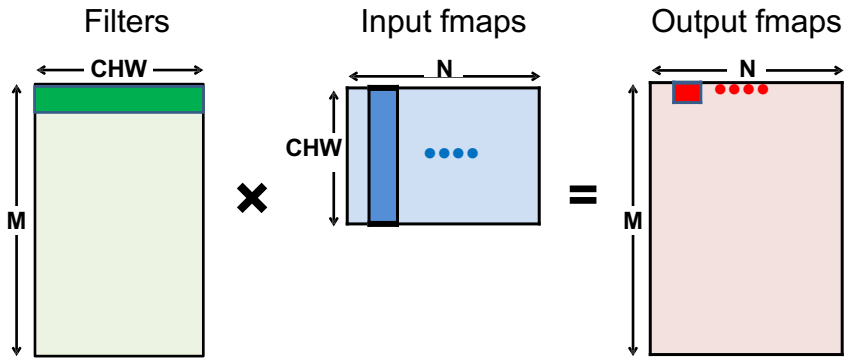
Fully-Connected (FC) Layer



- After flattening, having a batch size of N turns the matrix-vector operation into a matrix-matrix multiply

30

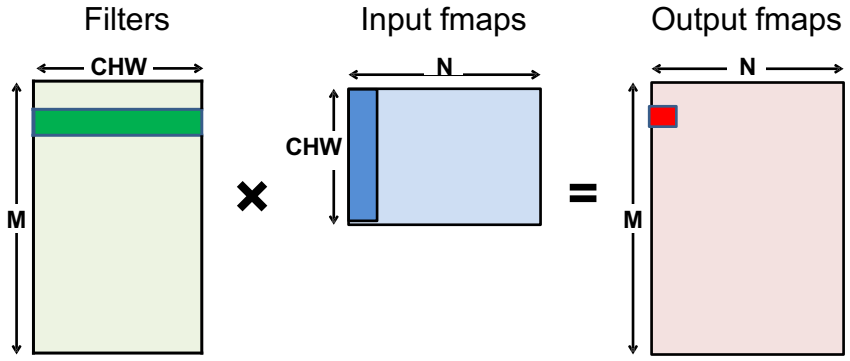
Fully-Connected (FC) Layer



- After flattening, having a batch size of N turns the matrix-vector operation into a matrix-matrix multiply

31

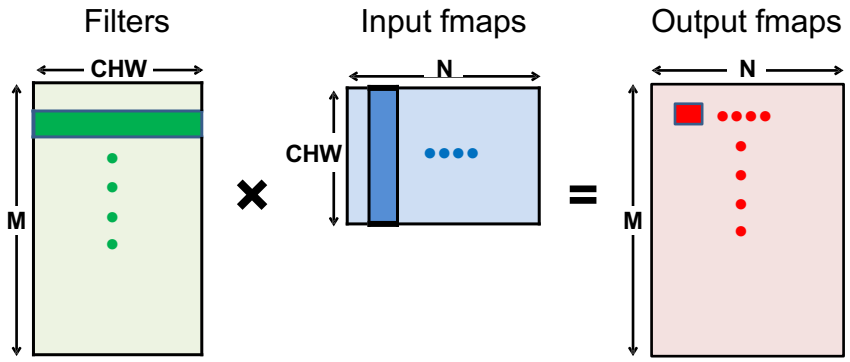
Fully-Connected (FC) Layer



- After flattening, having a batch size of N turns the matrix-vector operation into a matrix-matrix multiply

32

Fully-Connected (FC) Layer

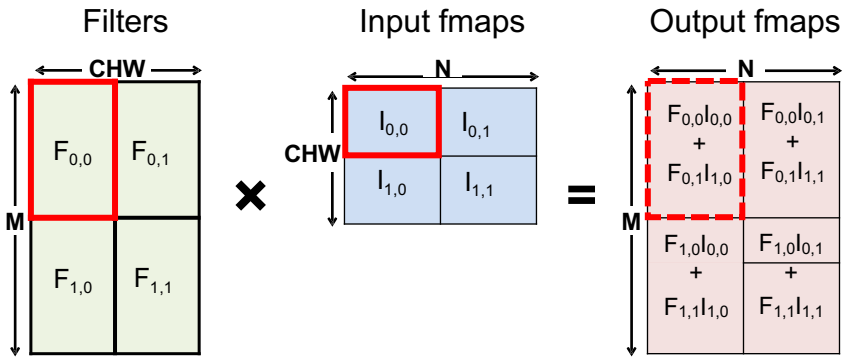


- After flattening, having a batch size of N turns the matrix-vector operation into a matrix-matrix multiply

How much temporal locality for naïve implementation? None

33

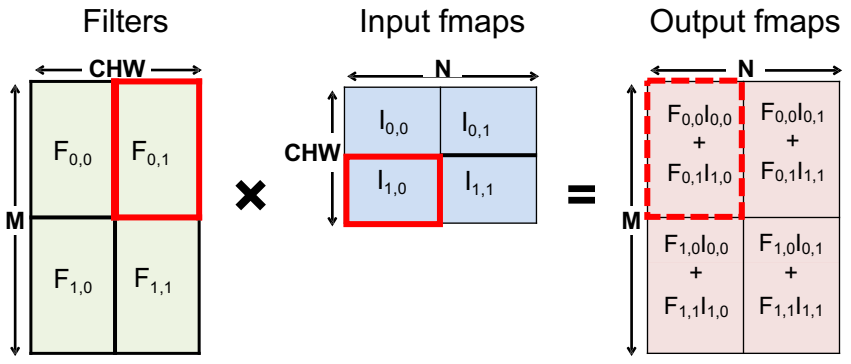
Tiled Fully-Connected (FC) Layer



Matrix multiply tiled to fit in cache
and computation ordered to maximize reuse of data in cache

34

Tiled Fully-Connected (FC) Layer



Matrix multiply tiled to fit in cache
and computation ordered to maximize reuse of data in cache

35