CMSC 5743

Efficient Computing of Deep Neural Networks

Implementation 01: GEMM

Bei Yu
CSE Department, CUHK
byu@cse.cuhk.edu.hk

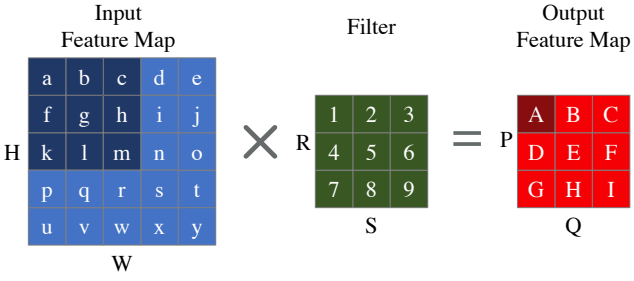
(Latest update: September 11, 2023)

2023 Fall

Overview

Convolution Basis

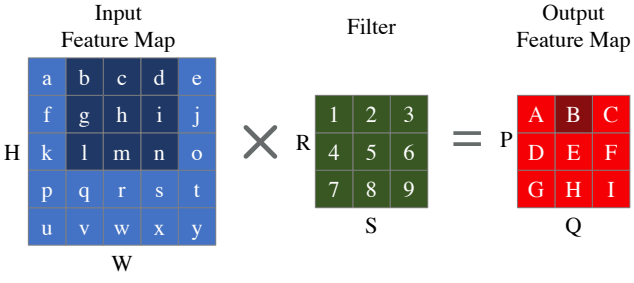
2D-Convolution



$$
\begin{aligned}
A = a \cdot 1 + b \cdot 2 + c \cdot 3 \\
+ f \cdot 4 + g \cdot 5 + h \cdot 6 \\
+ k \cdot 7 + l \cdot 8 + m \cdot 9
\end{aligned}
$$

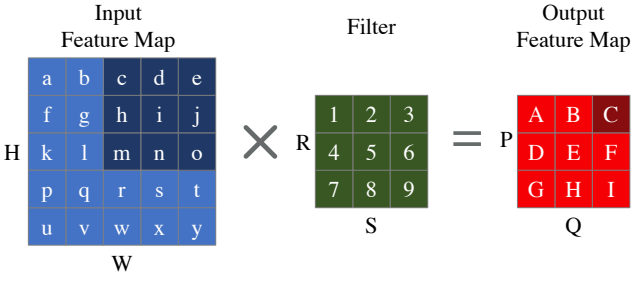
- H: Height of input feature map
- W: Width of input feature map
- R: Height of filter
- S: Width of filter
- P: Height of output feature map
- Q: Width of output feature map

2D-Convolution



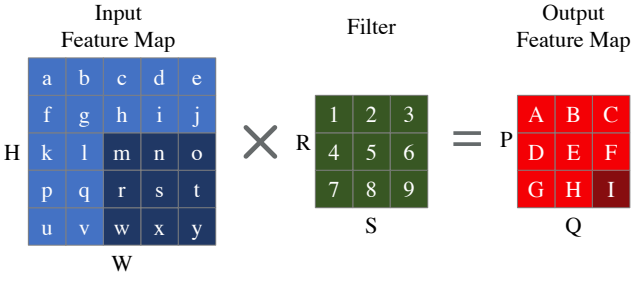
- H: Height of input feature map
- W: Width of input feature map
- R: Height of filter
- S: Width of filter
- P: Height of output feature map
- Q: Width of output feature map
- stride: # of rows/columns traversed per step

2D-Convolution



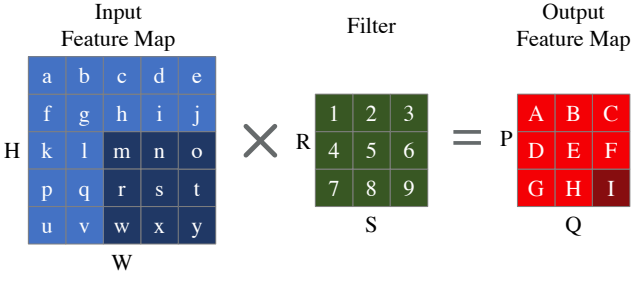
- H: Height of input feature map
- W: Width of input feature map
- R: Height of filter
- S: Width of filter
- P: Height of output feature map
- Q: Width of output feature map
- stride: # of rows/columns traversed per step

2D-Convolution



- H: Height of input feature map
- W: Width of input feature map
- R: Height of filter
- S: Width of filter
- P: Height of output feature map
- Q: Width of output feature map
- stride: # of rows/columns traversed per step

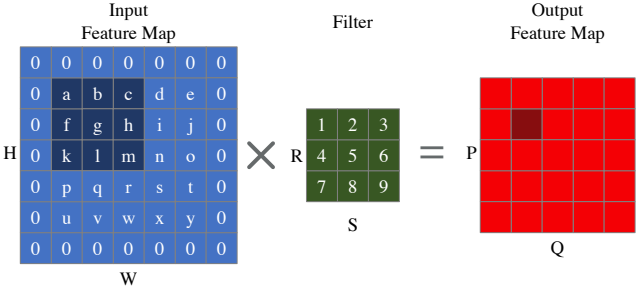
2D-Convolution



$$P = \frac{(H - R)}{\text{stride}} + 1;$$
$$Q = \frac{(W - S)}{\text{stride}} + 1.$$

- H: Height of input feature map
- W: Width of input feature map
- R: Height of filter
- S: Width of filter
- P: Height of output feature map
- Q: Width of output feature map
- stride: # of rows/columns traversed per step

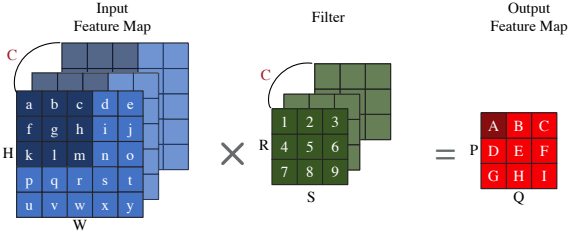
2D-Convolution



$$P = \frac{(H - R + 2 \cdot \text{pad})}{\text{stride}} + 1;$$
$$Q = \frac{(W - S + 2 \cdot \text{pad})}{\text{stride}} + 1.$$

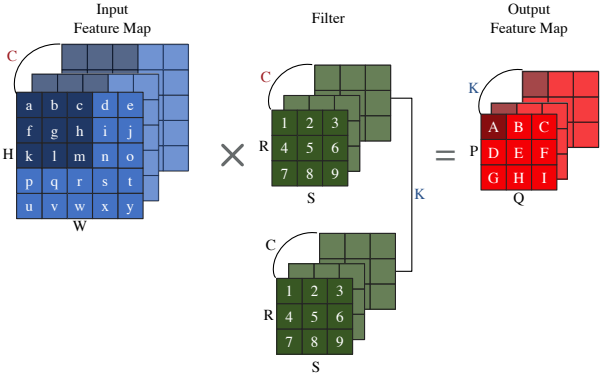
- H: Height of input feature map
- W: Width of input feature map
- R: Height of filter
- S: Width of filter
- P: Height of output feature map
- Q: Width of output feature map
- stride: # of rows/columns traversed per step
- padding: # of zero rows/columns added

3D-Convolution



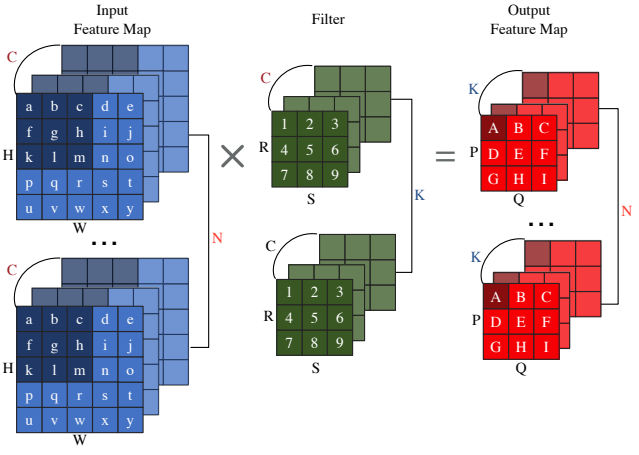
- H: Height of input feature map
- W: Width of input feature map
- R: Height of filter
- S: Width of filter
- P: Height of output feature map
- Q: Width of output feature map
- stride: # of rows/columns traversed per step
- padding: # of zero rows/columns added
- C: # of input channels

3D-Convolution



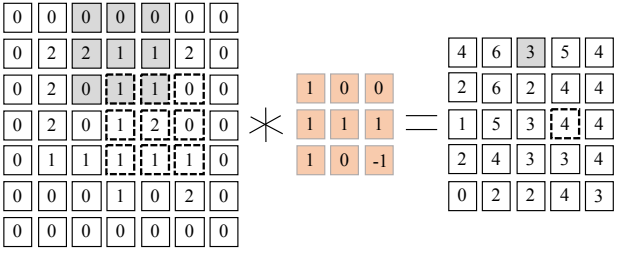
- H: Height of input feature map
- W: Width of input feature map
- R: Height of filter
- S: Width of filter
- P: Height of output feature map
- Q: Width of output feature map
- stride: # of rows/columns traversed per step
- padding: # of zero rows/columns added
- C: # of input channels
- K: # of output channels

3D-Convolution



- H: Height of input feature map
- W: Width of input feature map
- R: Height of filter
- S: Width of filter
- P: Height of output feature map
- Q: Width of output feature map
- stride: # of rows/columns traversed per step
- padding: # of zero rows/columns added
- C: # of input channels
- K: # of output channels
- N: Batch size

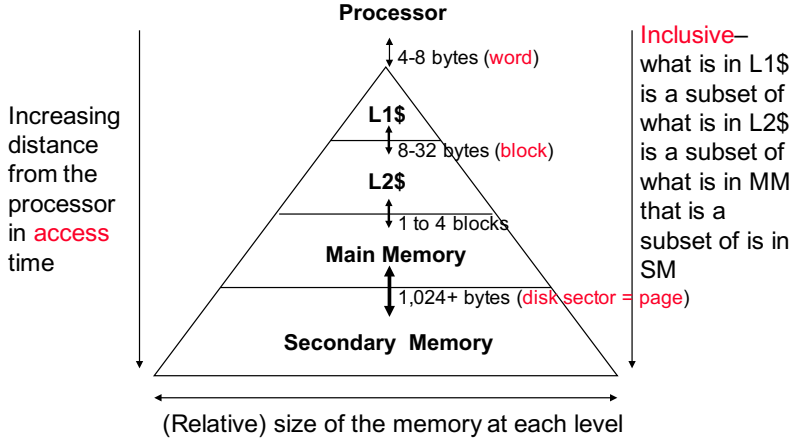
Convolution 101



Direct convolution: No extra memory overhead

- Low performance
- Poor memory access pattern due to geometry-specific constraint
- Relatively short dot product

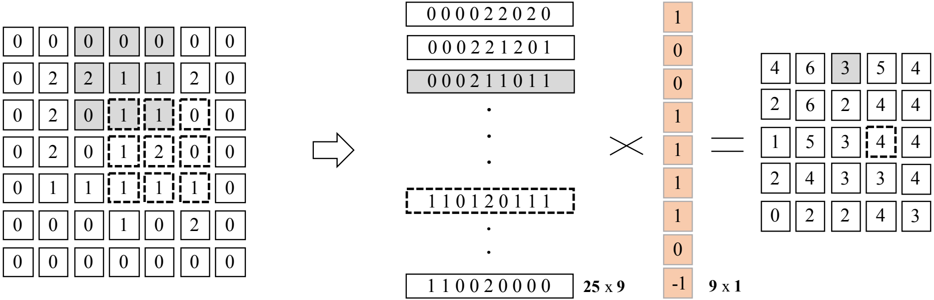
Background: Memory System



- Spatial locality
- Temporal Locality

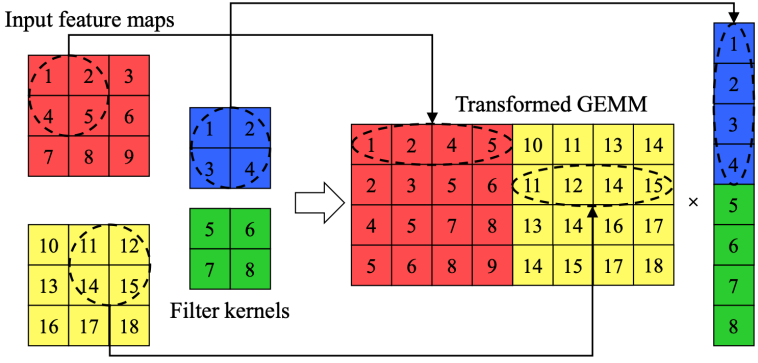
Im2Col

Im2col (Image2Column) Convolution



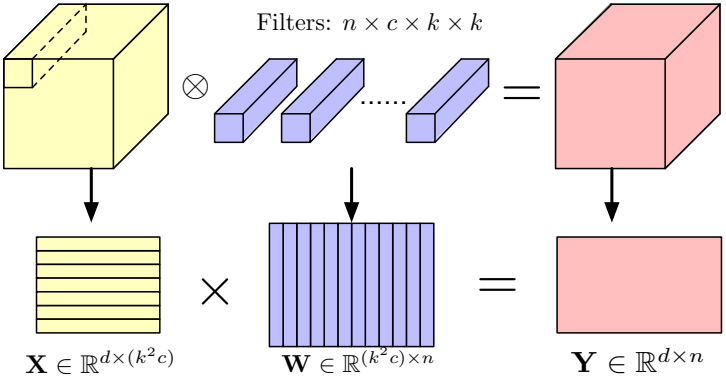
- Large extra memory overhead
- Good performance
- BLAS-friendly memory layout to enjoy SIMD/locality/parallelism
- Applicable for any convolution configuration on any platform

`Im2col` (Image2Column) Convolution



- Input channel #: 2
- Output channel #: 1

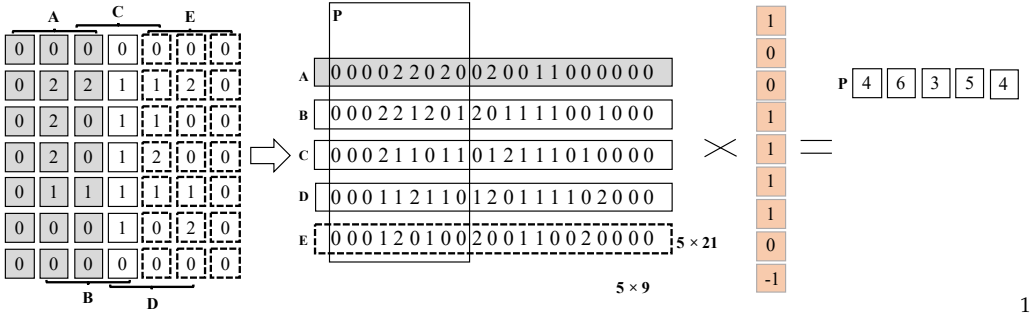
`Im2col` (Image2Column) Convolution



- Transform convolution to matrix multiplication
- Unified calculation for both convolution and fully-connected layers

Memory-efficient Convolution

Memory-efficient Convolution (MEC)

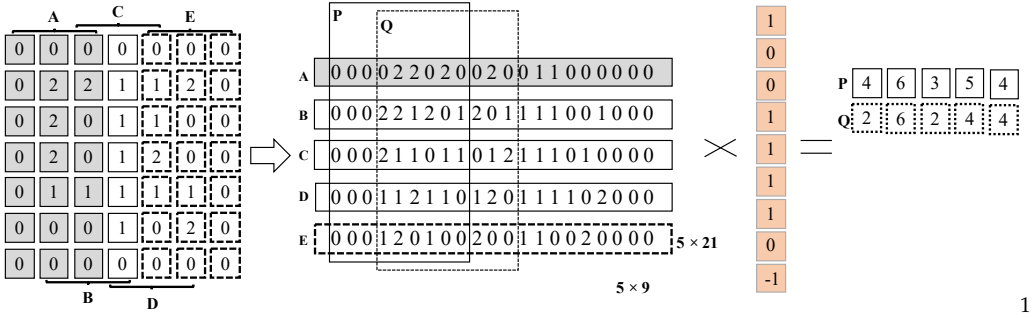


1

- Sub matrices in the lowered matrix will be "sgemm" ed in parallel
- Smaller memory foot print, cache locality, and explicit parallelism

[1] Minsik Cho and Daniel Brand (2017). "MEC: memory-efficient convolution for deep neural network". In: *Proc. ICML*.

Memory-efficient Convolution (MEC)

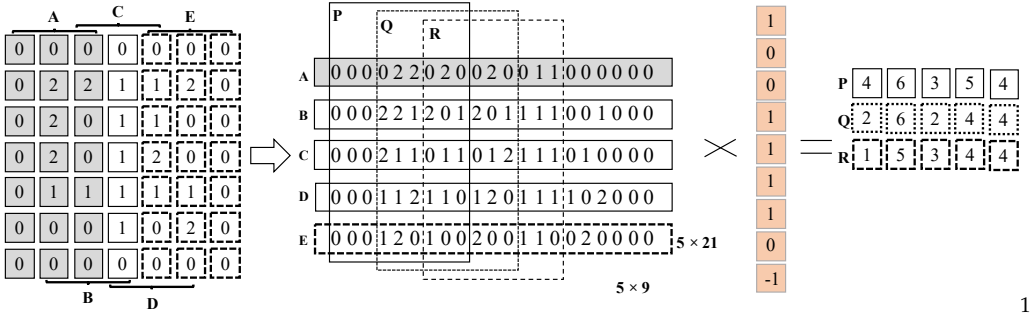


1

- Sub matrices in the lowered matrix will be "sgemm" ed in parallel
- Smaller memory foot print, cache locality, and explicit parallelism

[1] Minsik Cho and Daniel Brand (2017). "MEC: memory-efficient convolution for deep neural network". In: *Proc. ICML*.

Memory-efficient Convolution (MEC)

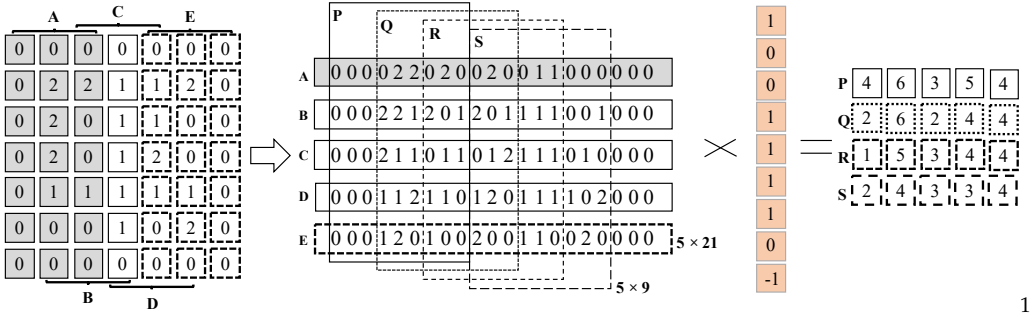


1

- Sub matrices in the lowered matrix will be "sgemm" ed in parallel
- Smaller memory foot print, cache locality, and explicit parallelism

[1] Minsik Cho and Daniel Brand (2017). "MEC: memory-efficient convolution for deep neural network". In: *Proc. ICML*.

Memory-efficient Convolution (MEC)

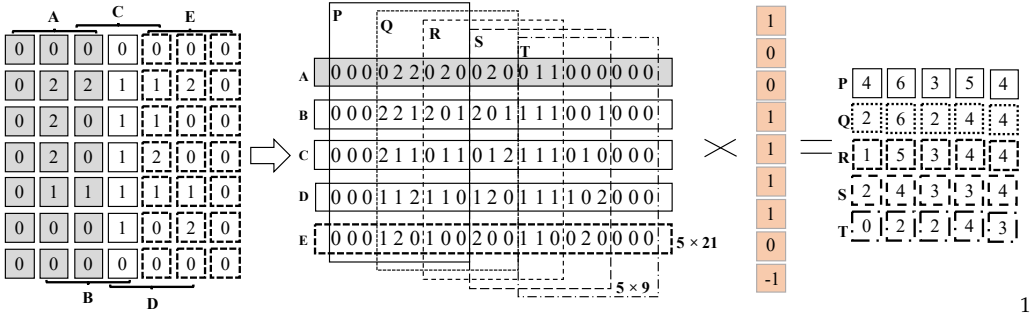


1

- Sub matrices in the lowered matrix will be "sgemm" ed in parallel
- Smaller memory foot print, cache locality, and explicit parallelism

[1]Minsik Cho and Daniel Brand (2017). "MEC: memory-efficient convolution for deep neural network". In: *Proc. ICML*.

Memory-efficient Convolution (MEC)

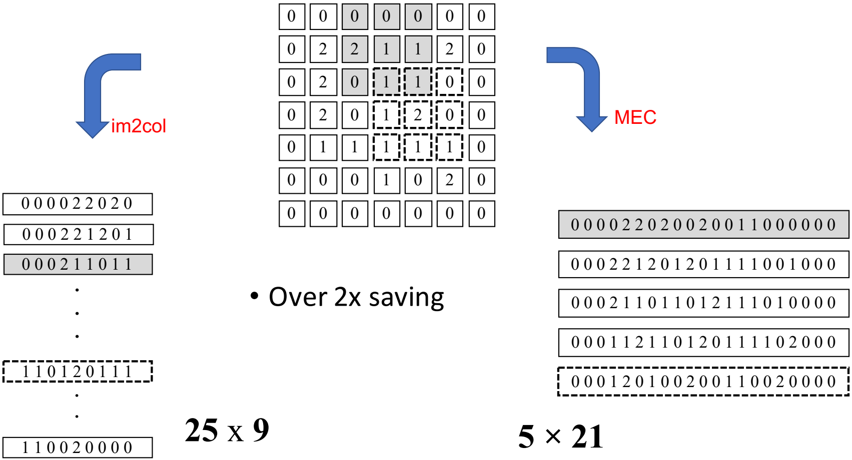


1

- Sub matrices in the lowered matrix will be "sgemm" ed in parallel
- Smaller memory foot print, cache locality, and explicit parallelism

[1]Minsik Cho and Daniel Brand (2017). "MEC: memory-efficient convolution for deep neural network". In: *Proc. ICML*.

Over 2× memory saving[2]:

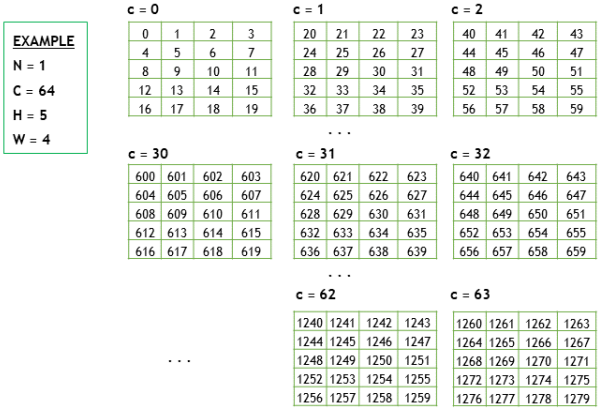


[2] Minsik Cho and Daniel Brand (2017). "MEC: memory-efficient convolution for deep neural network". In: *Proc. ICML*.

Memory Layout

Data Layout Formats[3]

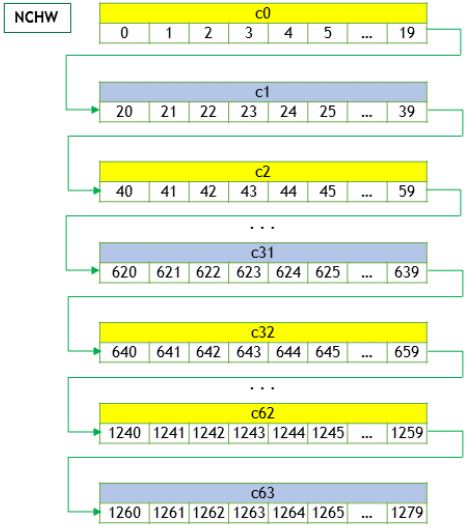
- N is the batch size
- C is the number of feature maps
- H is the image height
- W is the image width



[3] https://docs.nvidia.com/deeplearning/cudnn/developer-guide/index.html

NCHW Memory Layout

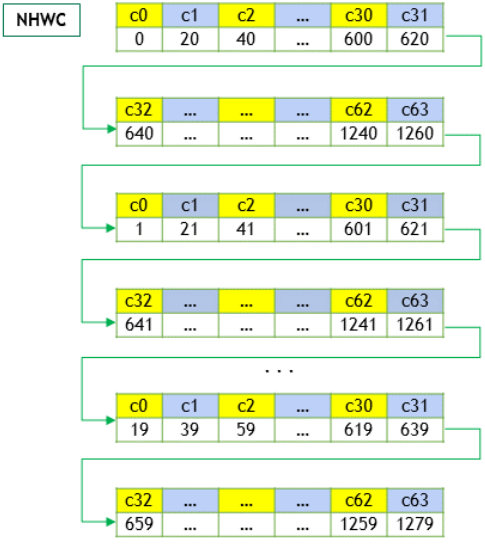
- Begin with first channel (c=0), elements arranged contiguously in row-major order
- Continue with second and subsequent channels until all channels are laid out



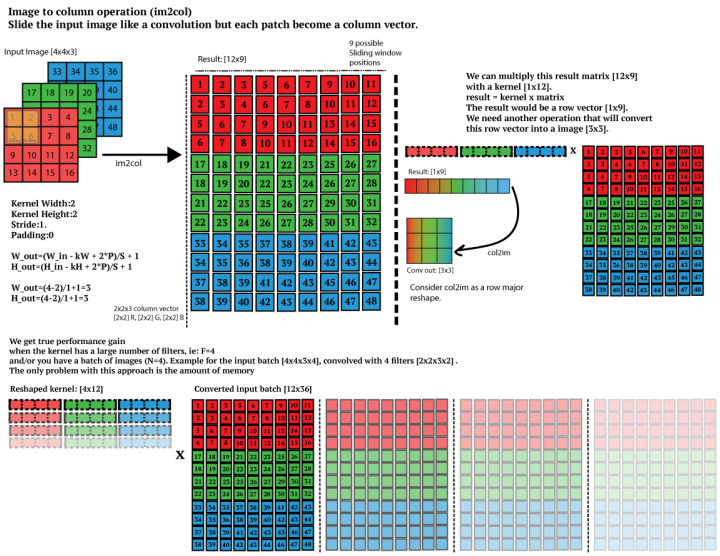
NHWC Memory Layout

- Begin with the first element of channel 0, then proceed to the first element of channel 1, and so on, until the first elements of all the C channels are laid out
- Next, select the second element of channel 0, then proceed to the second element of channel 1, and so on, until the second element of all the channels are laid out
- Follow the row-major order of channel 0 and complete all the elements
- Proceed to the next batch (if N is > 1)

NHWC Memory Layout



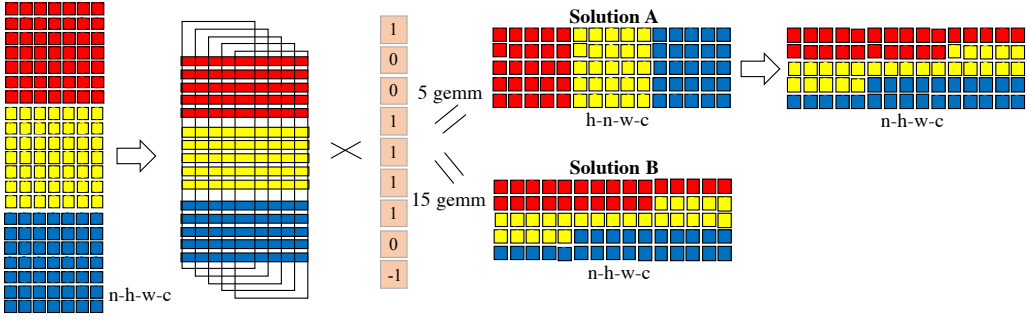
Memory Layout in `Im2col`



[4]

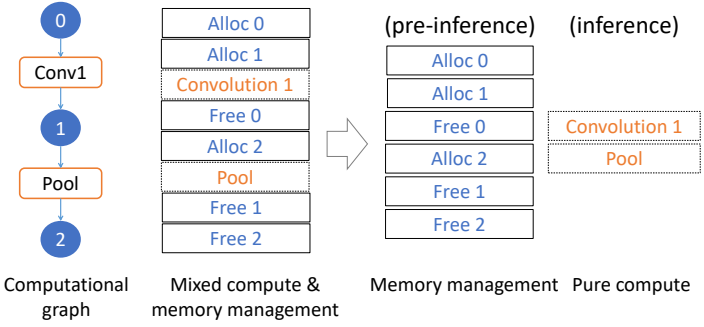
[4] https://leonardoaraujosantos.gitbook.io/artificial-inteligence/machine_learning/deep_learning/convolution_layer/making_faster

Memory Layout in Mini-batch MEC



- MEC w. mini-batch: can use `n-h-w-c` format
- Fusing convolution+pooling can be another solution

Memory optimization of MNN



- MNN can infer the exact required memory for the entire graph:
  - virtually walking through all operations
  - summing up all allocation and freeing