IMPLEMENTATION MISC.

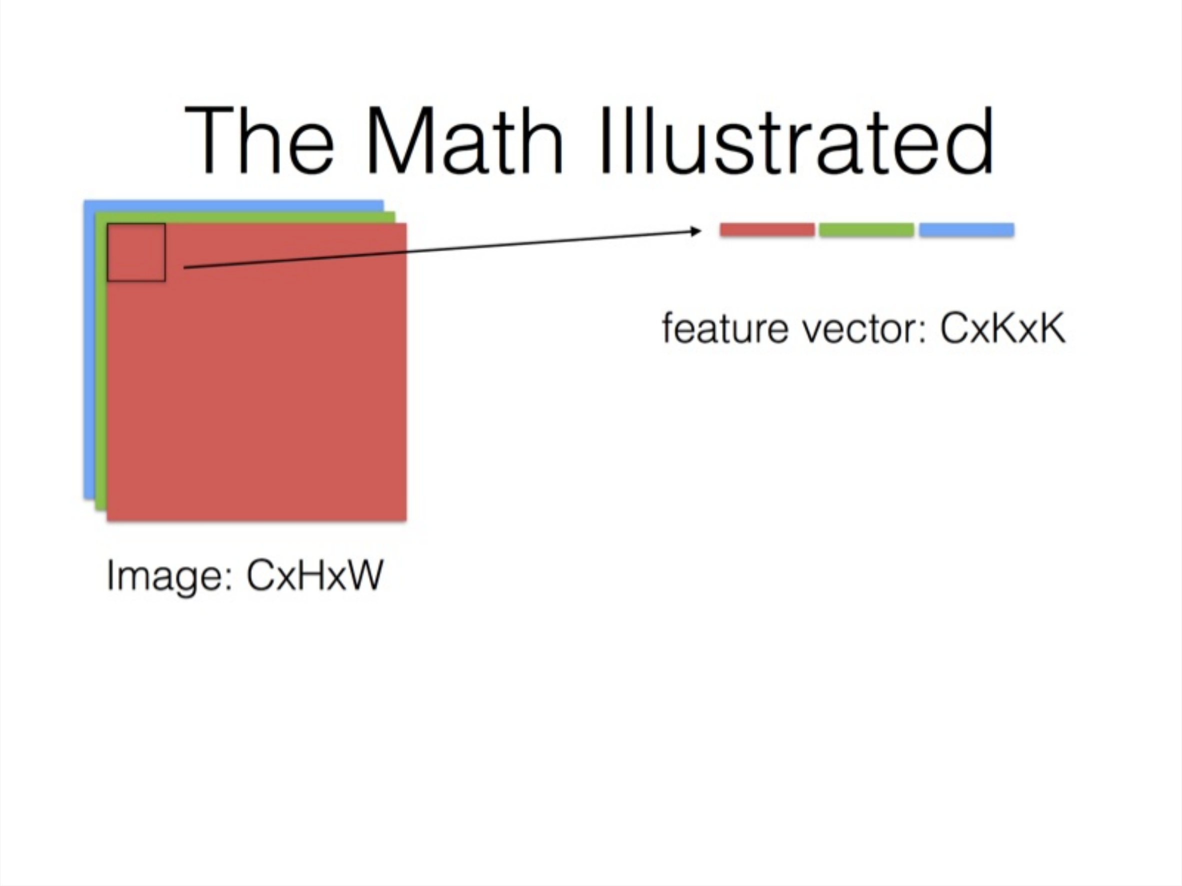
Rundong Li

April 9th, 2019

IMPLEMENTATION MISC.

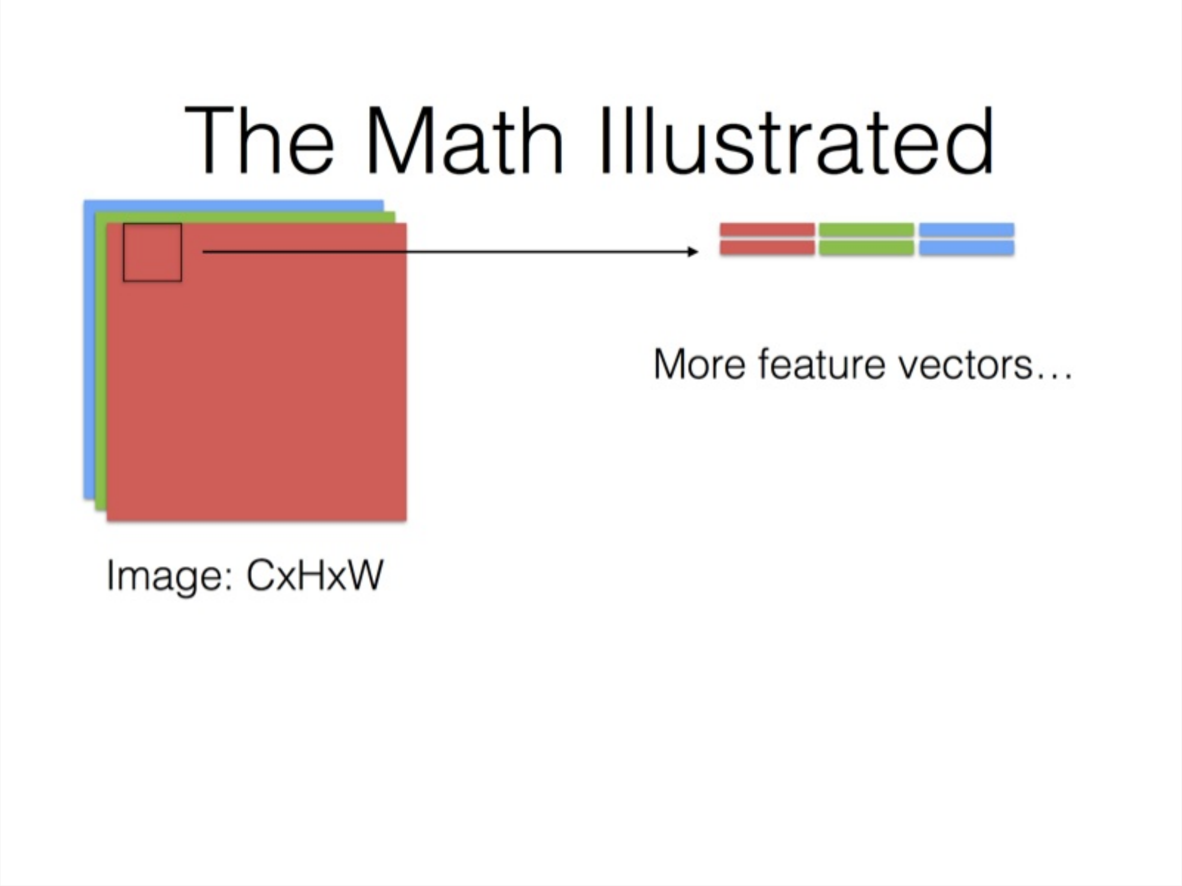
- Implement Convolution:
  - Caffe Style: img2col -> GEMM -> col2img
  - Small Kernels: Winograd
- Topic: QNNPACK
- Topic*: PyTorch Architecture
- Topic*: Caffe Architecture

CAFFE STYLE: IMG2COL -> GEMM -> COL2IMG

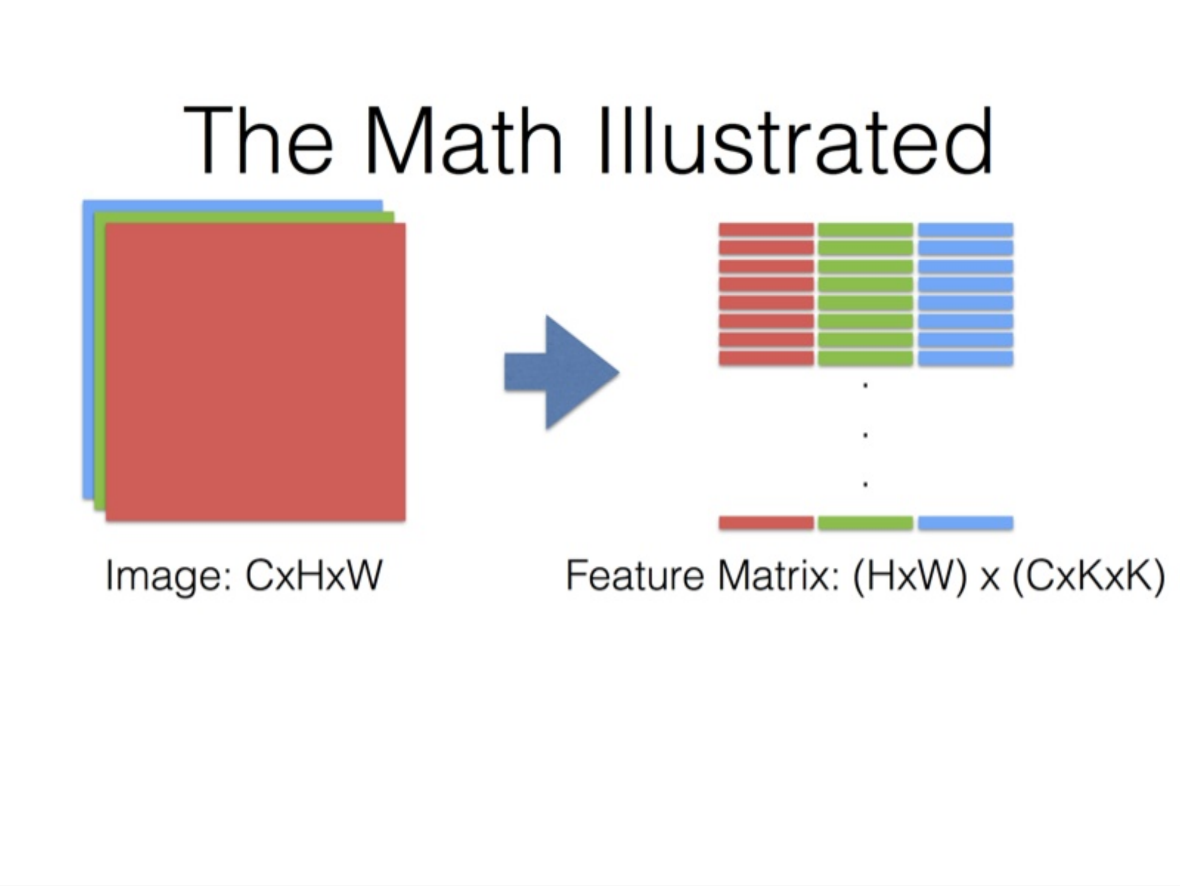


https://www.zhihu.com/question/28385679/answer/44297845

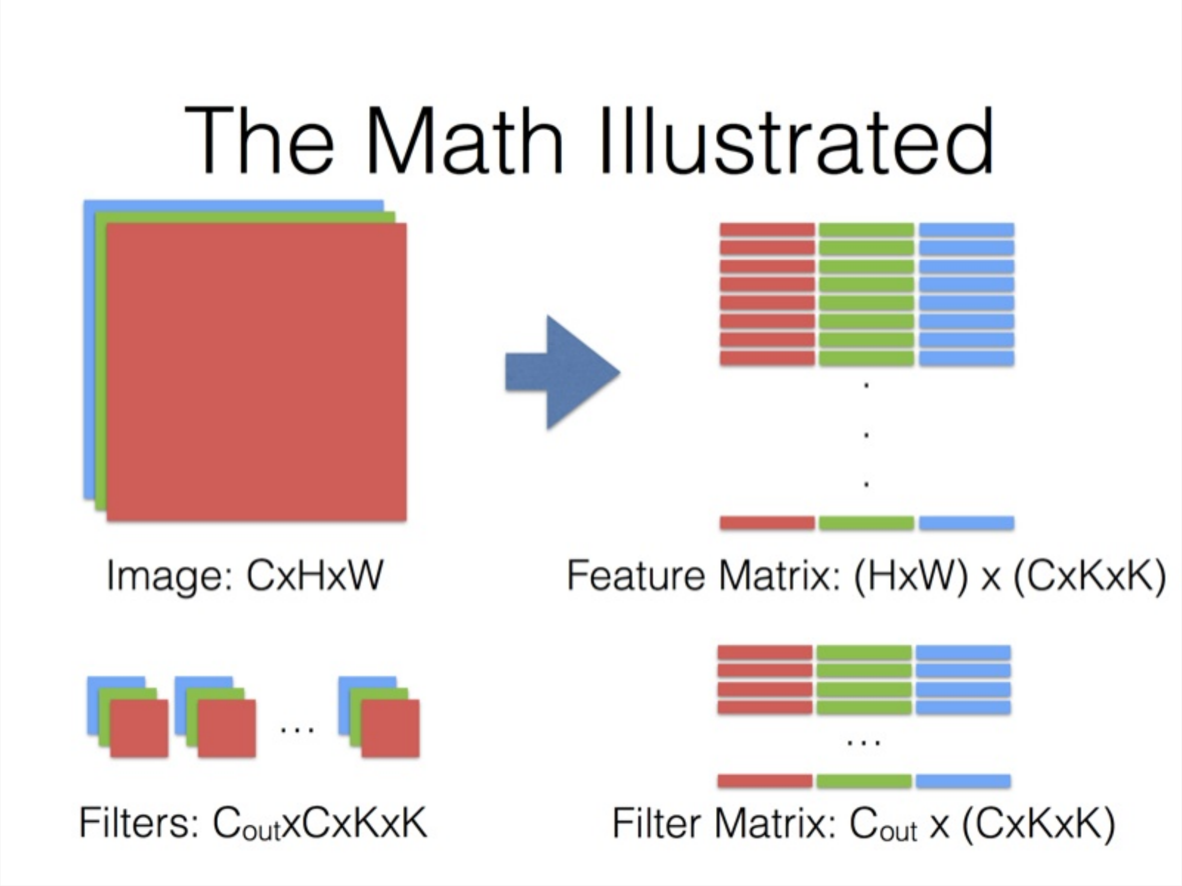
CAFFE STYLE: IMG2COL -> GEMM -> COL2IMG



CAFFE STYLE: IMG2COL -> GEMM -> COL2IMG



CAFFE STYLE:
IMG2COL -> GEMM -> COL2IMG



CAFFE STYLE:
IMG2COL -> GEMM -> COL2IMG

- Caffe implementation: img2col, forward_gemm
- Issues when deployed to mobile devices:
  - Runtime overhead: img2col & col2img should be performed on both weights and **activations**;
  - Memory consumption: large col-buffers;
  - Not efficient on depth-wise convolution (#groups = #channels);
- From Yangqing's memo, a developer's joke:
  - *`// somedev1 - 6/7/02 Adding temporary tracking of Login screen`*
  - *`// somedev2 - 5/22/07 Temporary my ass`*

SMALL KERNELS: WINOGRAD

- Core idea:
  - reuse intermediate production;
  - #multiplications = #inputs

$$[x_1, x_2, x_3, x_4] \star \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}$$

$$= \begin{bmatrix} x_1, x_2, x_3 \\ x_2, x_3, x_4 \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}$$

$$= \begin{bmatrix} m_1 + m_2 + m_3 \\ m_2 - m_3 - m_4 \end{bmatrix}$$

- Where

$$m_1 = (x_1 - x_3)w_1$$

$$m_2 = \frac{(x_2 + x_3)(w_1 + w_2 + w_3)}{2}$$

$$m_3 = \frac{(-x_2 + x_3)(w_1 - w_2 + w_3)}{2}$$

$$m_4 = (x_2 - x_4)w_3$$

https://arxiv.org/abs/1509.09308

SMALL KERNELS: WINOGRAD

- Formulation: when convolve $\boldsymbol{x}$ with $\boldsymbol{w}$,
  $y = A^T[(G\boldsymbol{w}) \odot (B^T\boldsymbol{x})]$
- For F(2, 3) Winograd filter:
  - 2: output size (m)
  - 3: kernel size (r)
  - Stride is fixed to 1
  - Input $\boldsymbol{x}$ size: $m + r - 1 = 4$
  - Other terms:

$$B^T = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}$$

$$G = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

$$A^T = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & -1 & -1 \end{bmatrix}$$

SMALL KERNELS: WINOGRAD

- Similar to FFT:
  - $y = A^T[(G\boldsymbol{w}) \odot (B^T\boldsymbol{x})]$
  - Project to "transform domain";
  - Project back to "real domain";
  - Convolution in real domain $\star$ becomes element-wise multiplication $\odot$ (not really);
- However, computations in "transform domain" (e.g. $\odot$) are performed on $\mathbb{R}$, (note that FFT is on $\mathbb{C}$) which gives 4x less real multiplications;

SMALL KERNELS: WINOGRAD

- Two nested F(m, r)'s form a 2D filter F(m×m, r×r), with input size $(m+r-1, m+r-1)$. When applying this kernel to (H, W) sized input $\boldsymbol{x}$:
  - Split $\boldsymbol{x}$ into $(m+r-1, m+r-1)$ sized tiles, *with* $r-1$ *overlap between neighbor tiles (see the notes)*;
  - Note that W can not be split: only works for small W;
  - Handel channels, tiles and spatial locations:

$$
\begin{aligned}
Y_{i,k,\widetilde{x},\widetilde{y}} &= \sum_{c=1}^{C} D_{i,c,\widetilde{x},\widetilde{y}} * G_{k,c} \\
&= \sum_{c=1}^{C} A^T \left[U_{k,c} \odot V_{c,i,\widetilde{x},\widetilde{y}} \right] A \\
&= A^T \left[\sum_{c=1}^{C} U_{k,c} \odot V_{c,i,\widetilde{x},\widetilde{y}} \right] A
\end{aligned}
$$

$$M_{k,i,\widetilde{x},\widetilde{y}} = \sum_{c=1}^{C} U_{k,c} \odot V_{c,i,\widetilde{x},\widetilde{y}}$$

$$M_{k,b}^{(\xi,\nu)} = \sum_{c=1}^{C} U_{k,c}^{(\xi,\nu)} V_{c,b}^{(\xi,\nu)}$$

$$M^{(\xi,\nu)} = U^{(\xi,\nu)} V^{(\xi,\nu)}$$

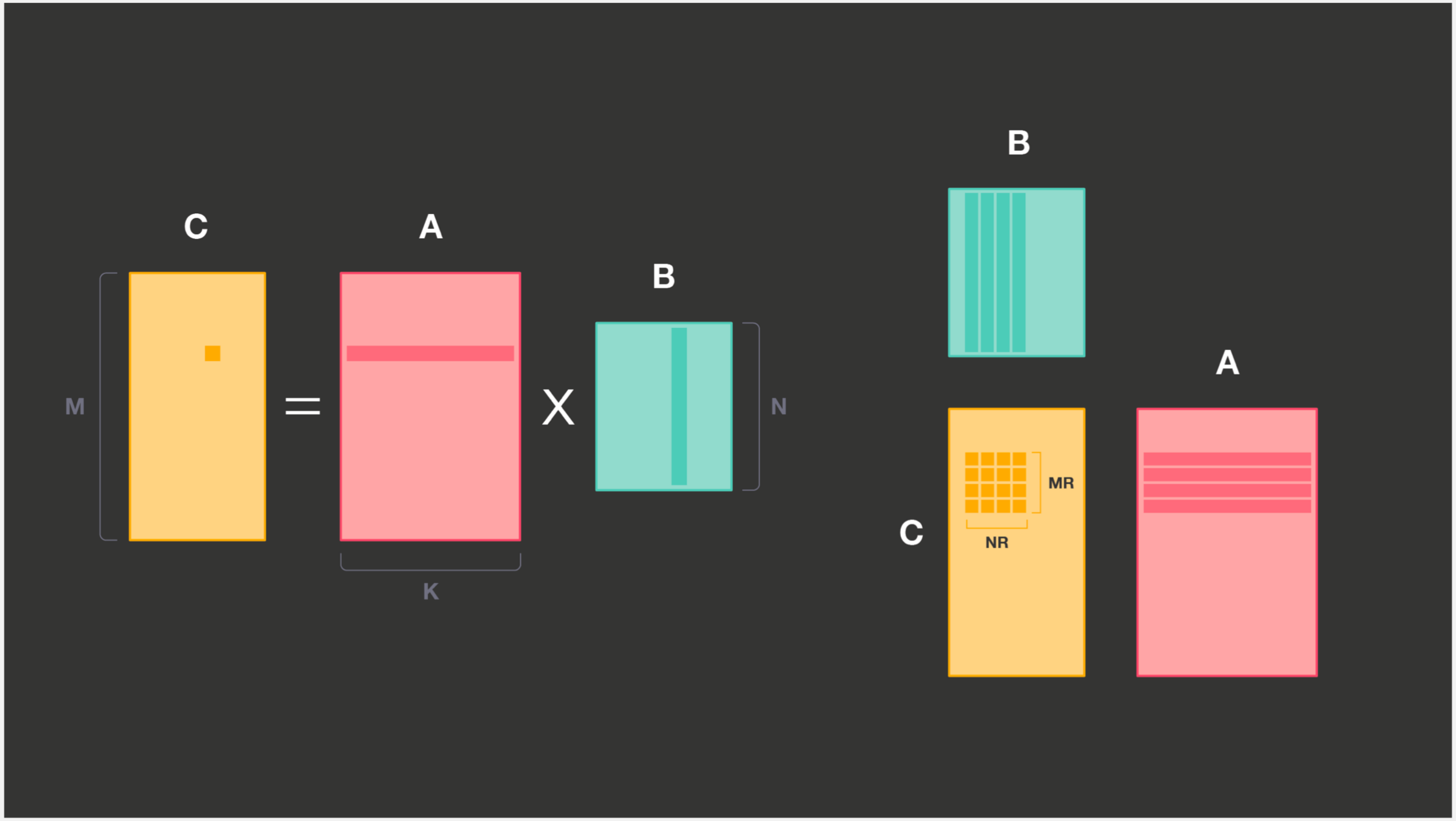
TOPIC: QNNPACK

- Integrated in PyTorch 1.0 / Caffe2;
- Optimized for mobile deployment;
- Optimized FP32 / INT8 GEMM, Convolution, Depth-wise convolution, etc.;

- Major optimizations:
  - GEMM (with small #channels)
  - INT8 Convolution
  - Depth-wise Convolution

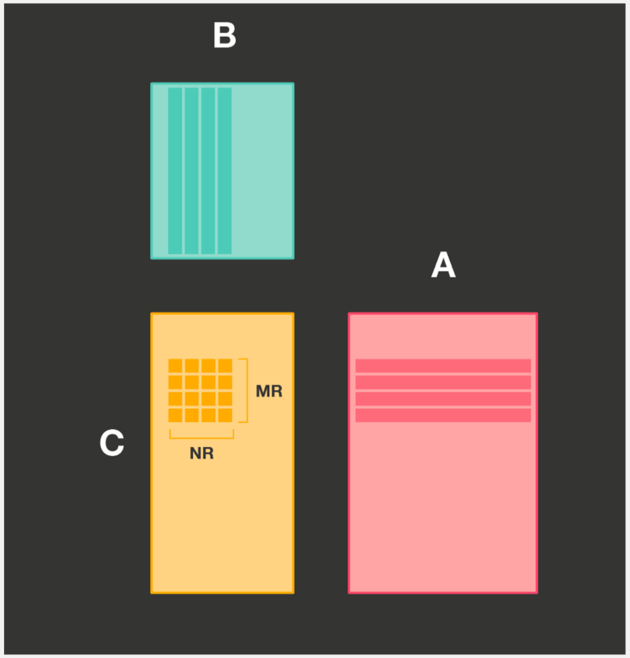
https://code.fb.com/ml-applications/qnnpack/

QNNPACK: GEMM

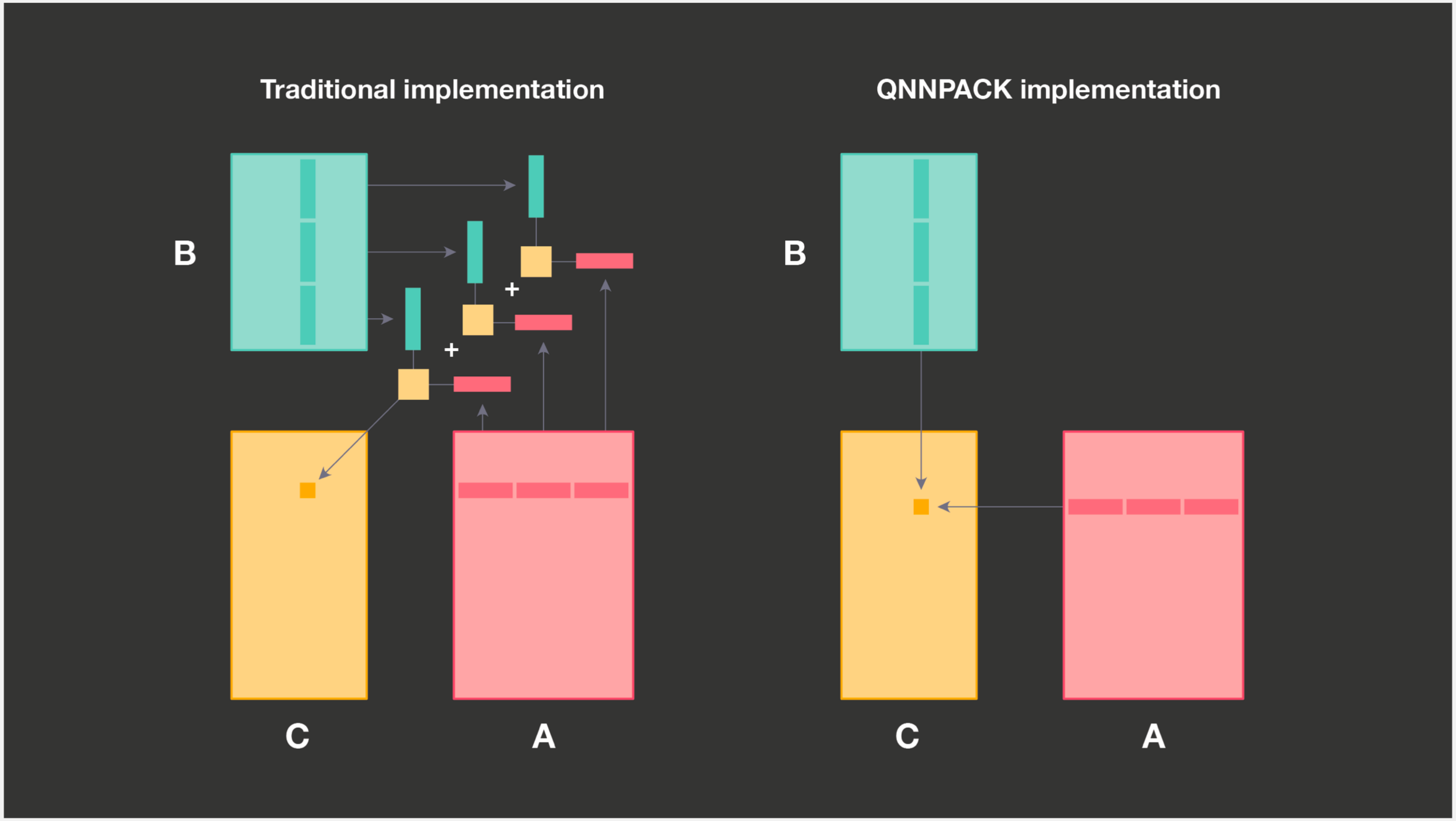


QNNPACK: GEMM

- One dot per loop: memory bounded;
- PDOT (panel dot product): load R row/col from A/B simultaneously;
- MR / NR is constrained by:
  - #registers
  - Cache size

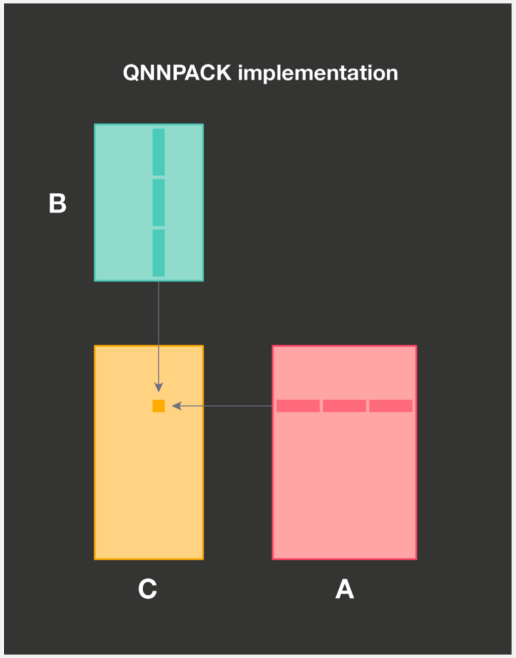


QNNPACK: GEMM

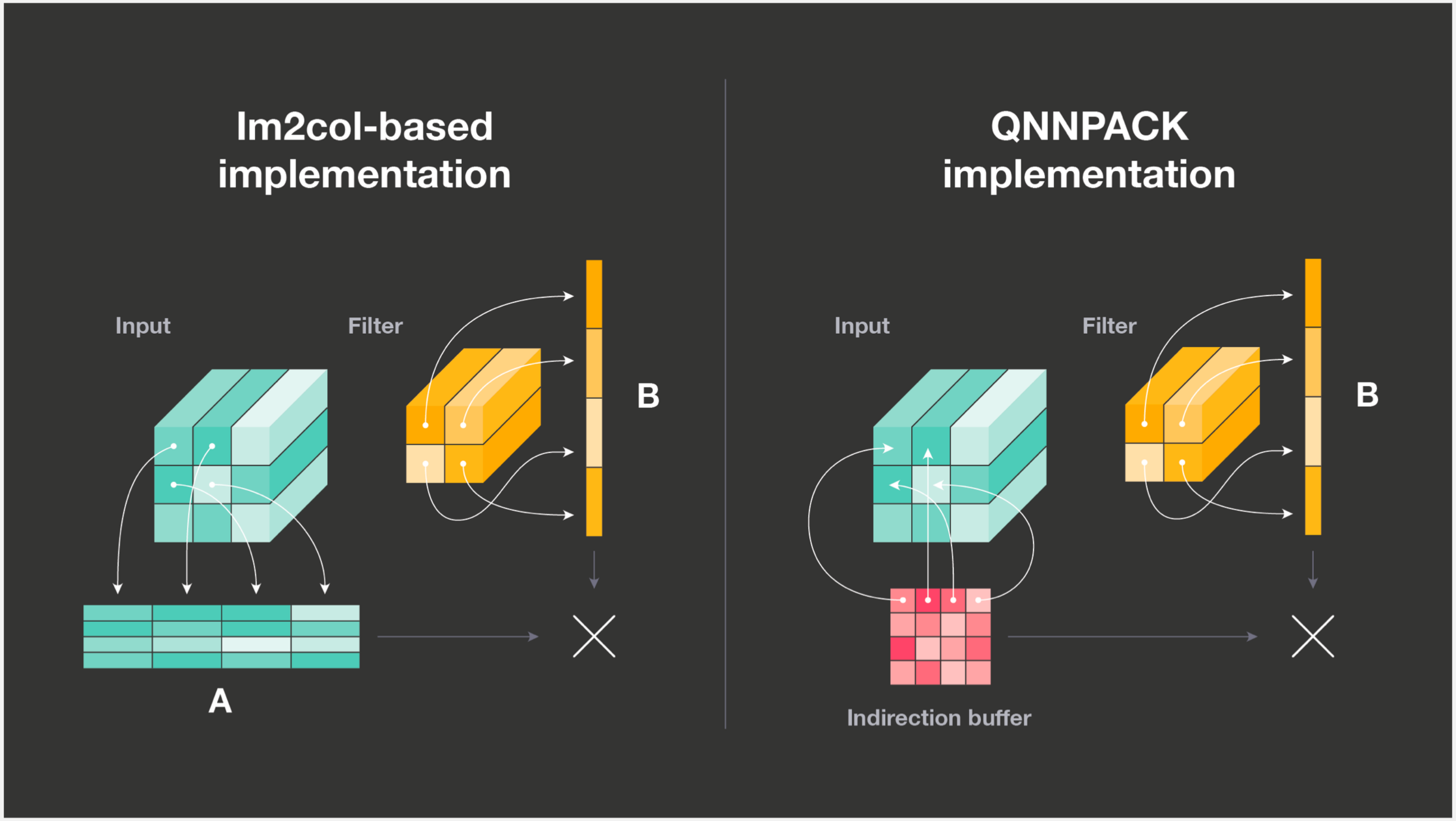


QNNPACK: GEMM

- MR / NR constrained to 8, K is no larger than 1024 (channels) thus size of memory blocks on each PDOT call is no larger than 16KB (2 * 1024 * 8 * 8bits), which can be easily fit into L1 on all ARM arch;
- Thus no panel spliting is required;

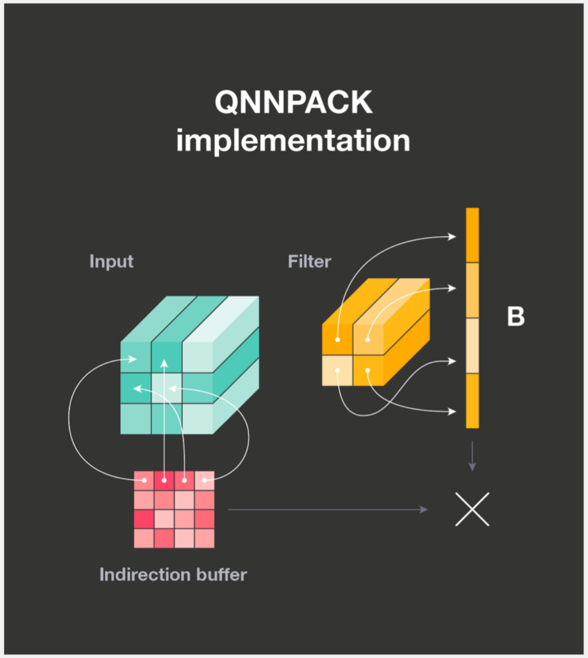


QNNPACK: CONVOLUTION



QNNPACK: CONVOLUTION

- Much less runtime overhead for activation A: no more copies;
- “Virtual” col-buffer: only store *pointers* to each spatial location of A, less memory footprint;
- A and B should be in *NHWC* format: QNNPACK, SNPE, etc.



QNNPACK: DEPTH CONV.