# SCNN: An Accelerator for Compressed-sparse Convolutional Neural Networks

Angshuman Parashar, **Minsoo Rhu***, Anurag Mukkara, Antonio Puglielli, Rangharajan Venkatesan, Brucek Khailany, Joel Emer, Stephen W. Keckler, and William J. Dally

NVIDIA

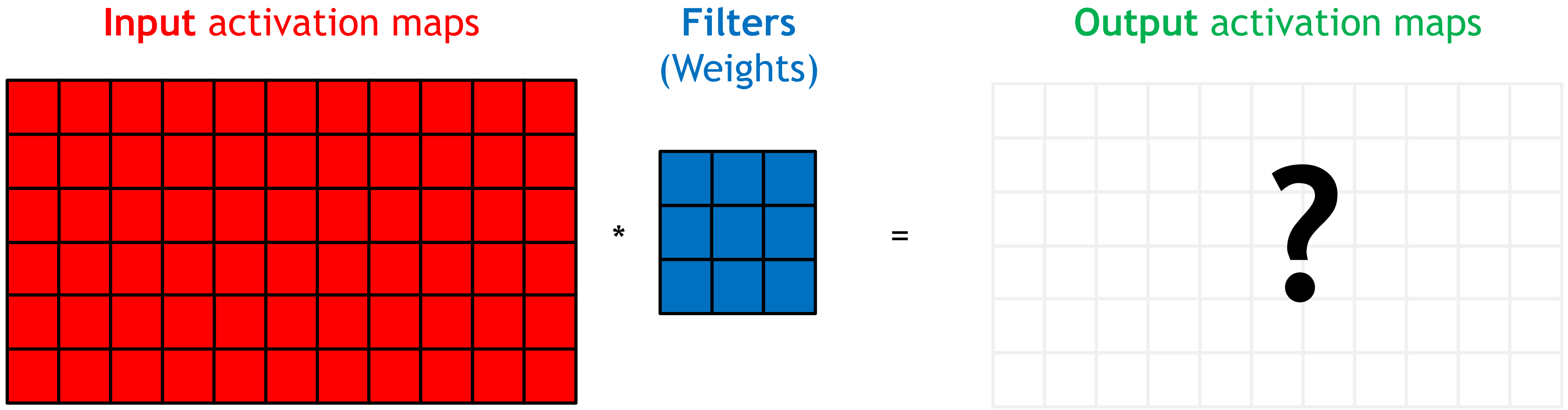
* Now at POSTECH (Pohang University of Science and Technology), South Korea

# Motivation

# Convolution (dense)

## Inner product between every input pixel and every filter coefficient

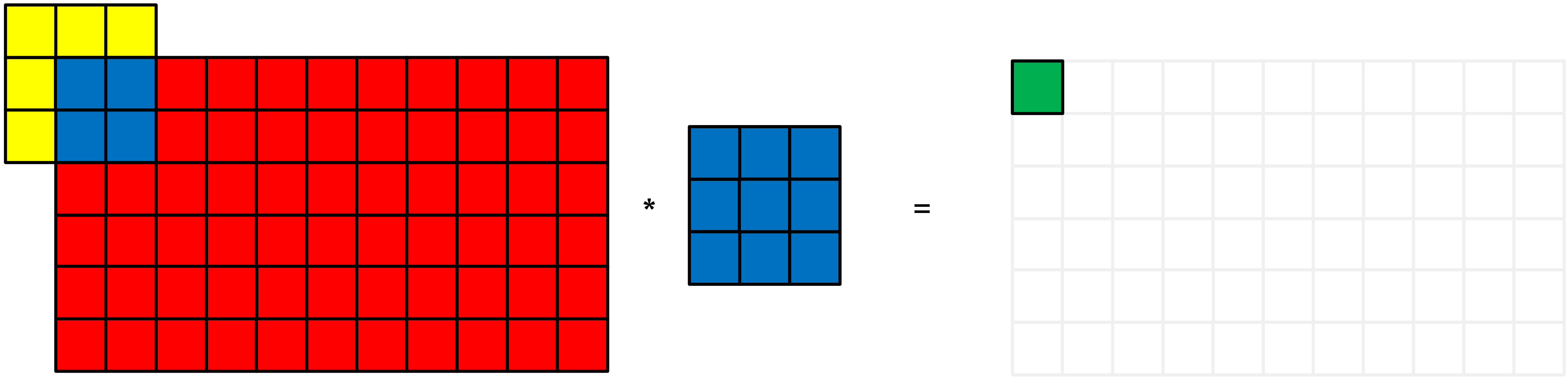
Sliding window is intuitive and maps to reasonable hardware implementation

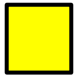


# Convolution (dense)

## Inner product between every input pixel and every filter coefficient

Sliding window is intuitive and maps to reasonable hardware implementation

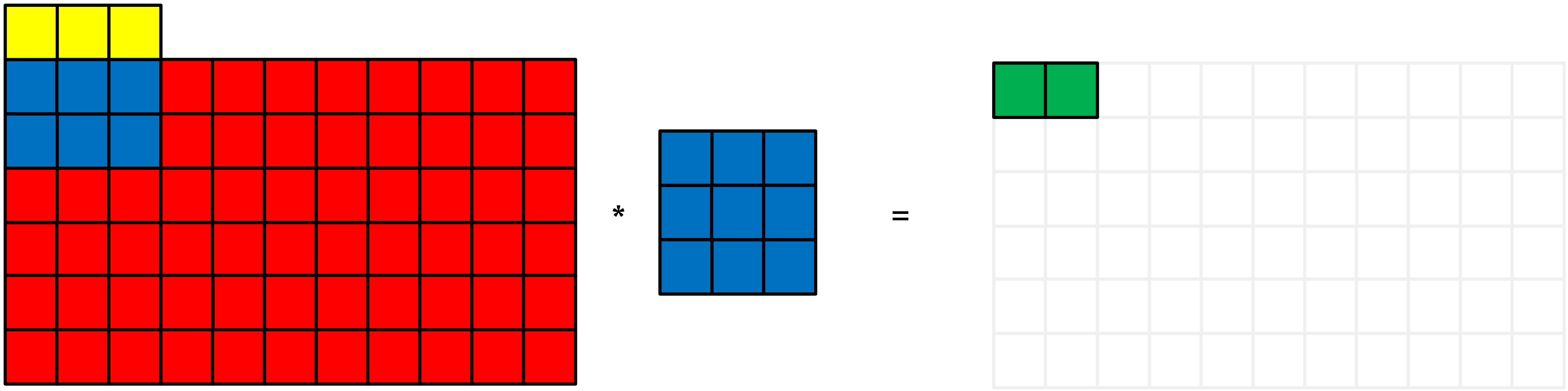



 : useless MUL ops

# Convolution (dense)

## Inner product between every input pixel and every filter coefficient

Sliding window is intuitive and maps to reasonable hardware implementation

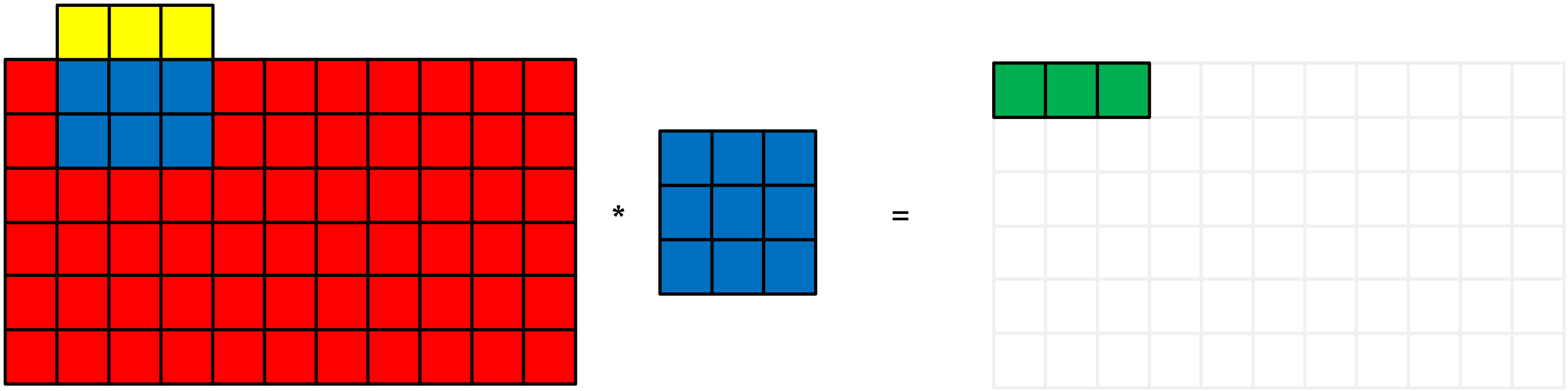


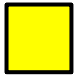

: useless MUL ops

# Convolution (dense)

## Inner product between every input pixel and every filter coefficient

Sliding window is intuitive and maps to reasonable hardware implementation

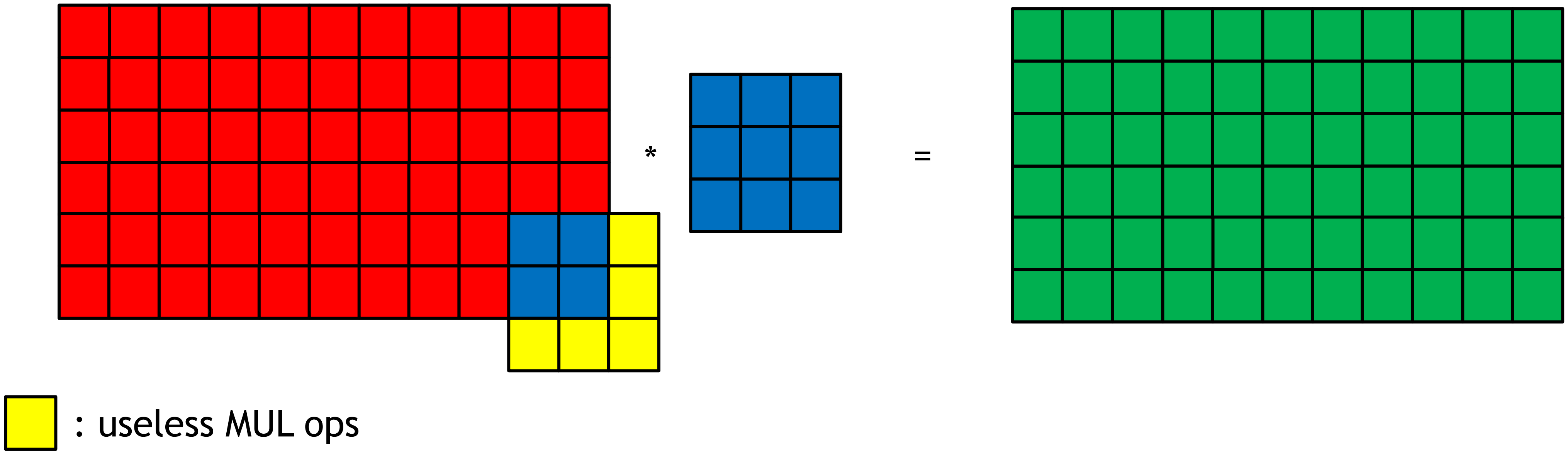



 : useless MUL ops

# Convolution (dense)

## Inner product between every input pixel and every filter coefficient

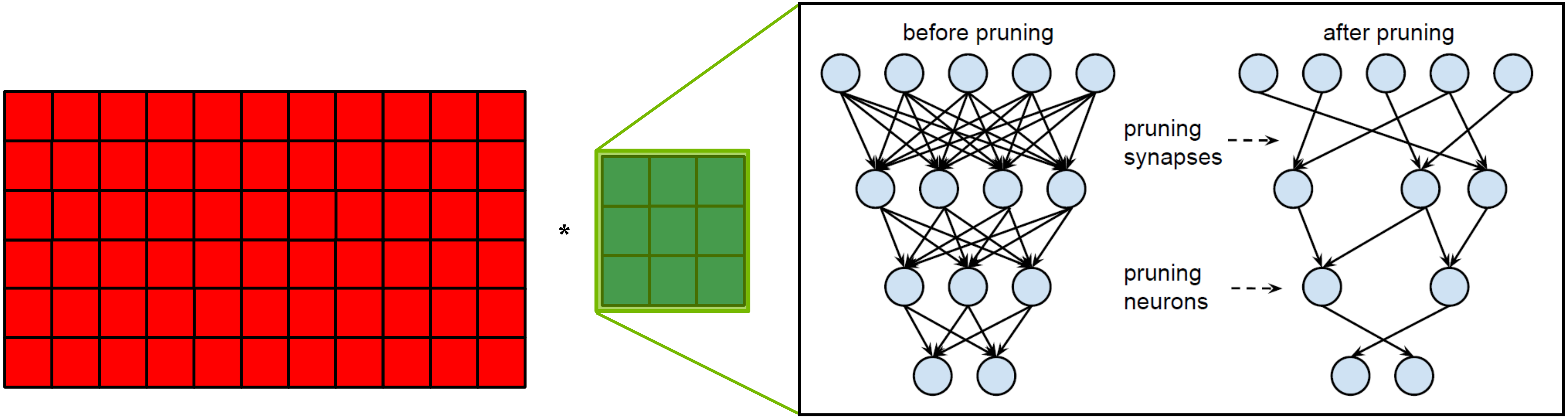
Sliding window is intuitive and maps to reasonable hardware implementation



# Convolution with Sparsity

## Most operand values are zero

**Static** sparsity: pruned network weights set to ‘0’ during training

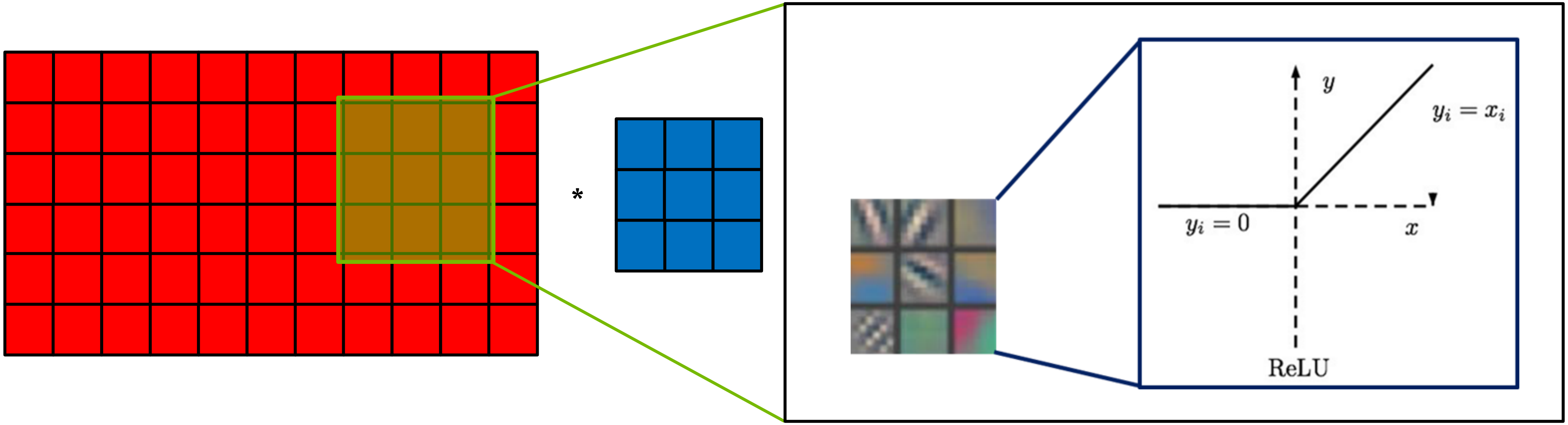



* Han et al., “Learning Both Weights and Connections for Efficient Neural Network”, NIPS-2015

# Convolution with Sparsity

## Most operand values are zero

**Dynamic** sparsity: negative-valued activations clamped to ‘0’ during inference



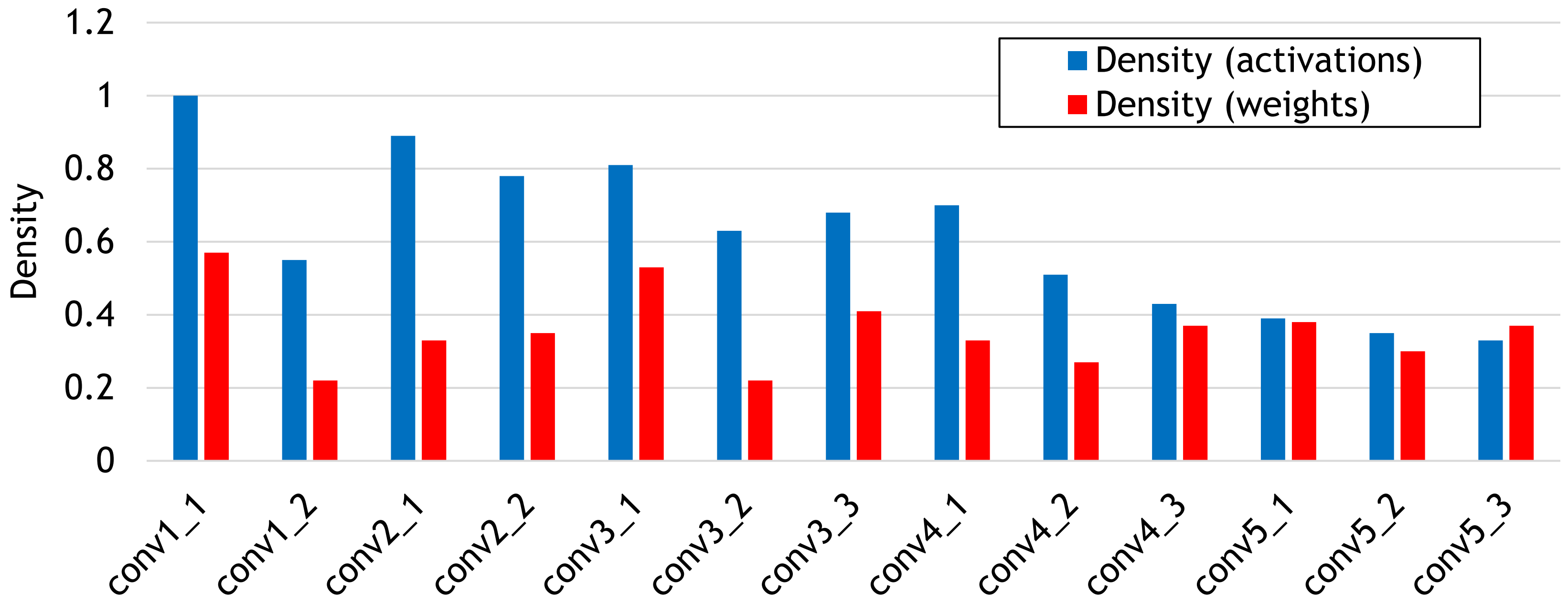

* Albericio et al., “CNVLUTIN: Ineffectual-Neuron-Free Deep Neural Network Computing”, ISCA-2016

# Convolution with Sparsity

## Most operand values are zero

Sliding window based convolution is wasteful

Fraction of non-zero (NZ) activations & weights is roughly 20~50% per layer



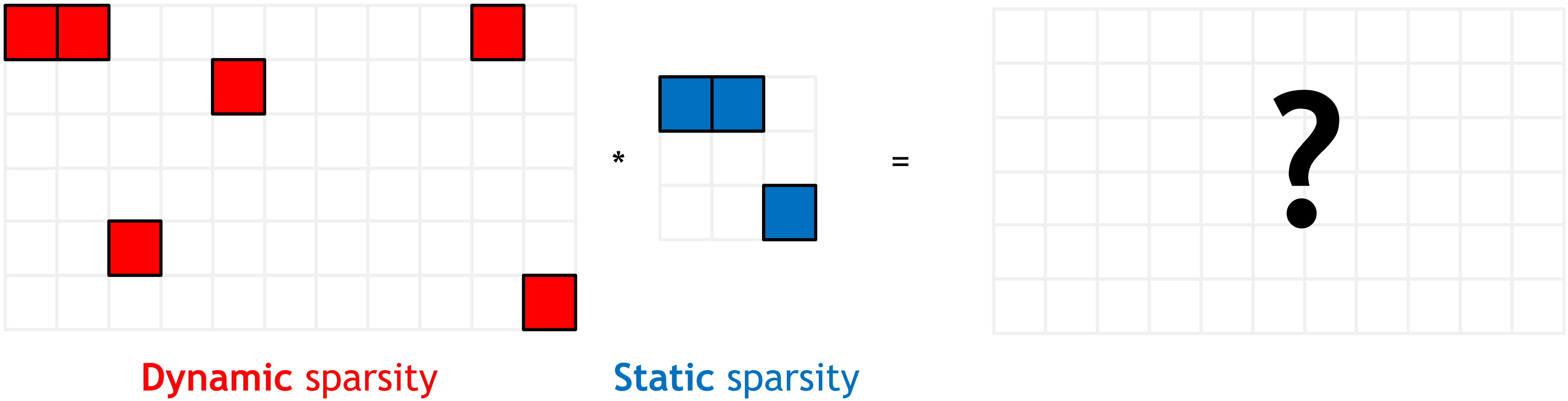

< VGGNet >

# Convolution with Sparsity

## Most operand values are zero

Sliding window based convolution is wasteful

Fraction of non-zero (NZ) activations & weights is roughly 20~50% per layer

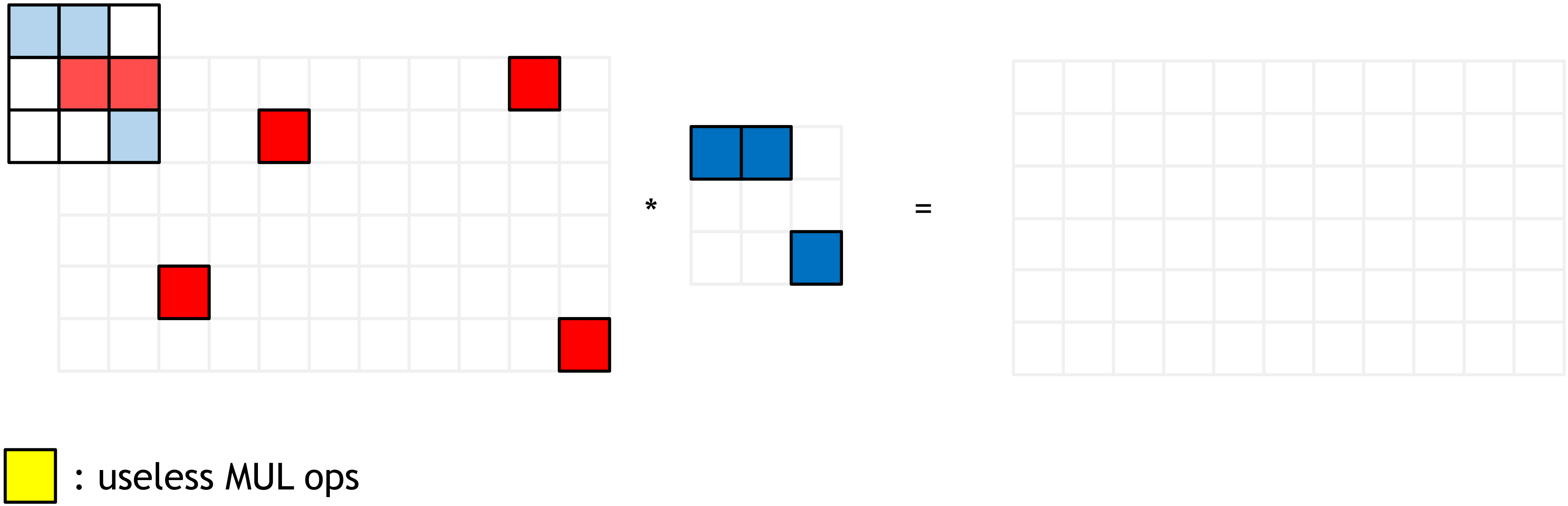


# Convolution with Sparsity

## Most operand values are zero

Sliding window based convolution is wasteful

Fraction of non-zero (NZ) activations & weights is roughly 20~50% per layer

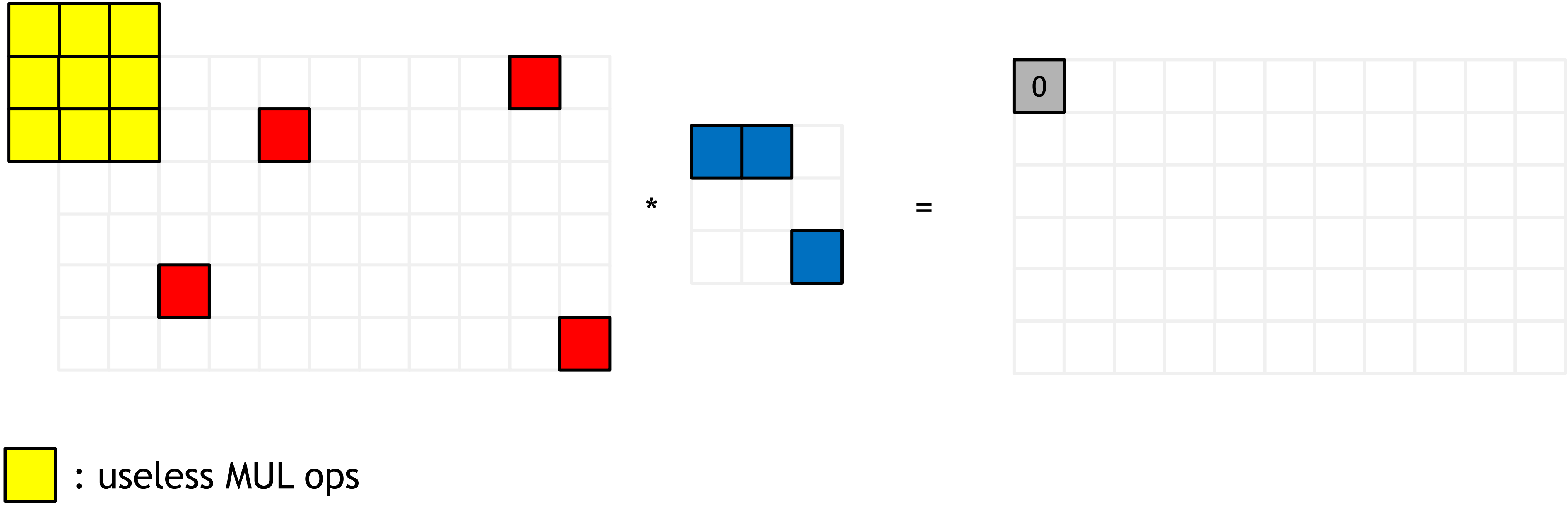


# Convolution with Sparsity

## Most operand values are zero

Sliding window based convolution is wasteful

Fraction of non-zero (NZ) activations & weights is roughly 20~50% per layer

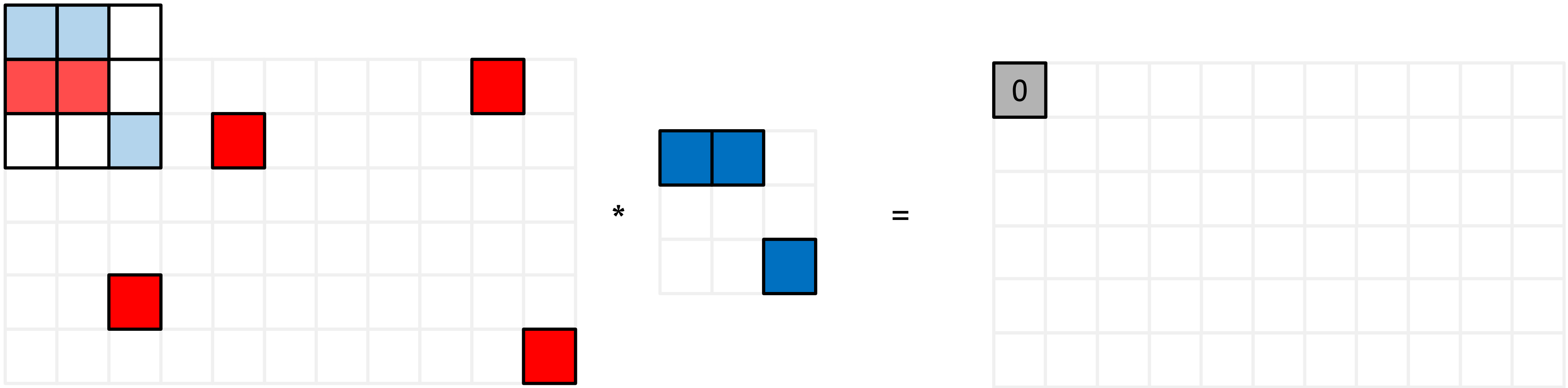


# Convolution with Sparsity

## Most operand values are zero

Sliding window based convolution is wasteful

Fraction of non-zero (NZ) activations & weights is roughly 20~50% per layer



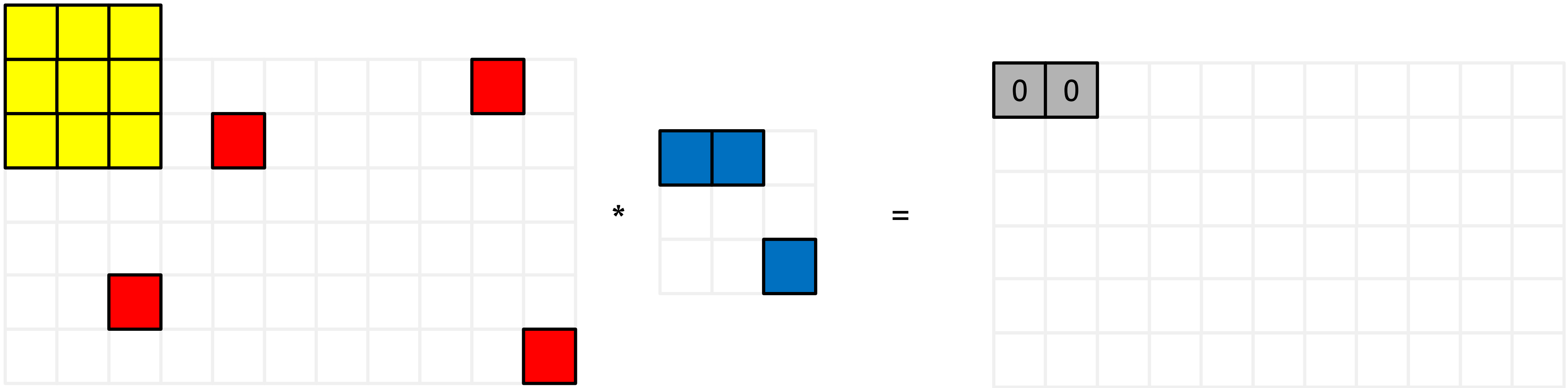

: useless MUL ops

# Convolution with Sparsity

## Most operand values are zero

Sliding window based convolution is wasteful

Fraction of non-zero (NZ) activations & weights is roughly 20~50% per layer




: useless MUL ops

# Motivation

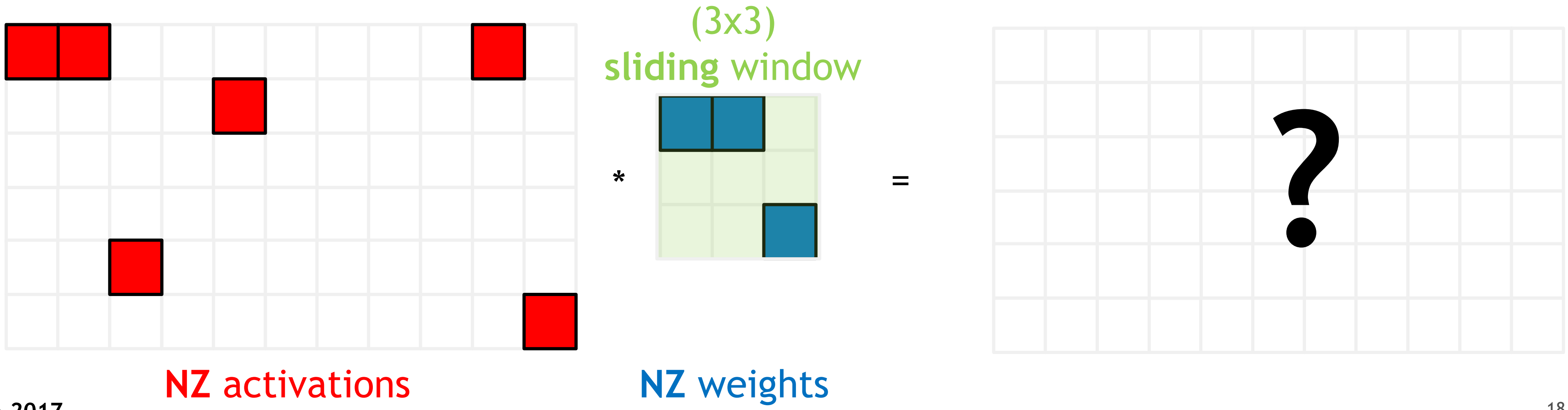
## CNN inference often performed in power-limited environments

Our goal: sparsity-optimized CNN accelerator for high energy-efficiency

# Possible Solutions

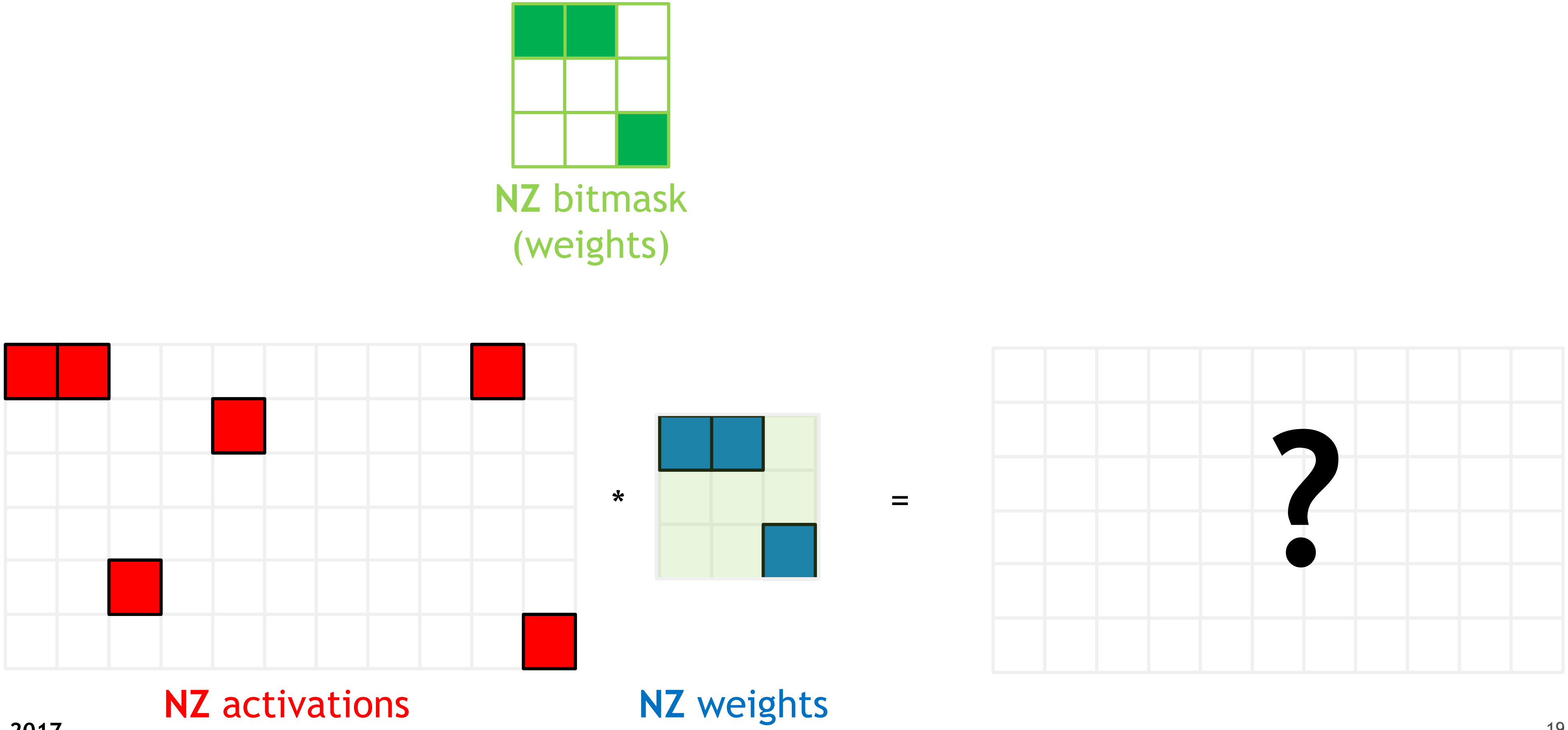
# Option 1: Leverage Dense CNN Design

## Employ pair of bit-masks to track non-zero weights and/or activations



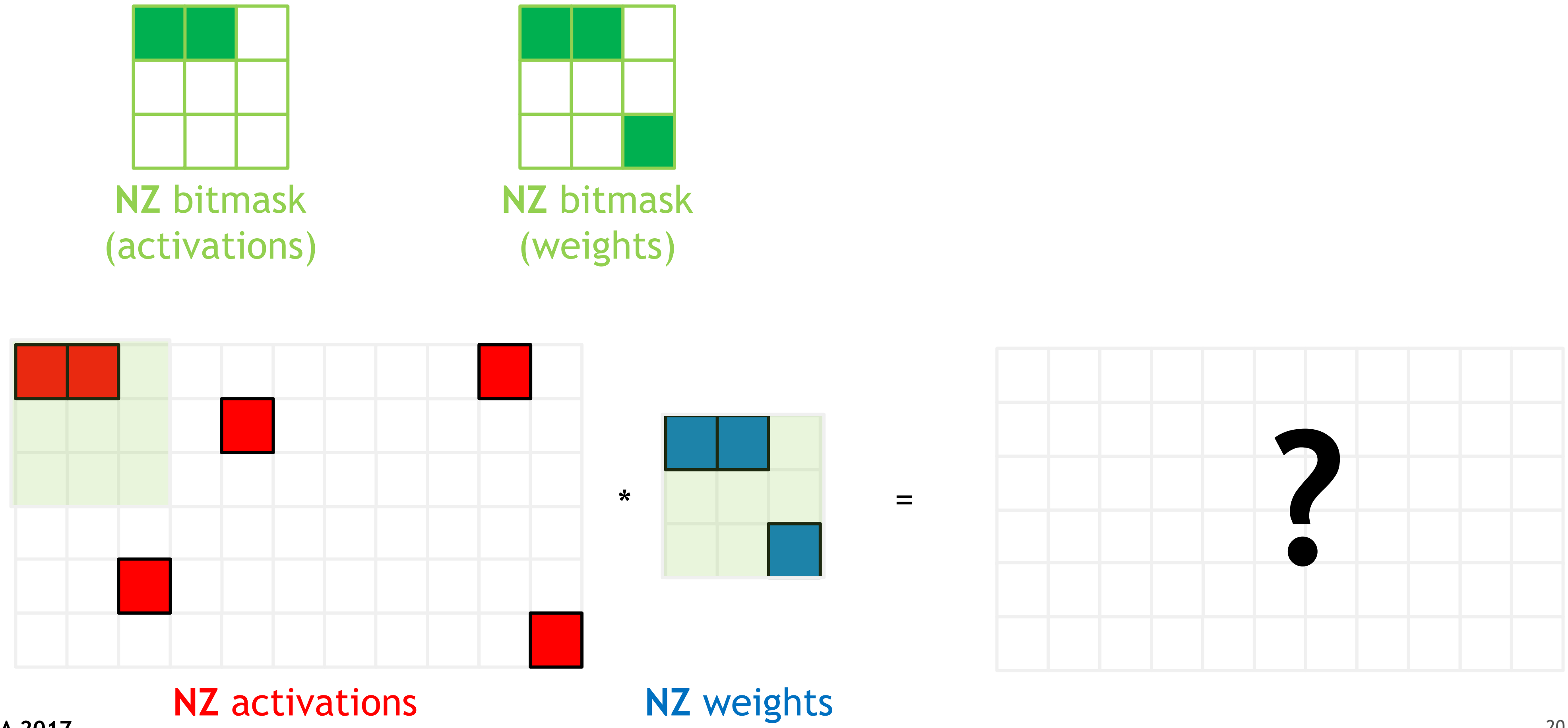
# Option 1: Leverage Dense CNN Design

Employ pair of bit-masks to track non-zero weights and/or activations



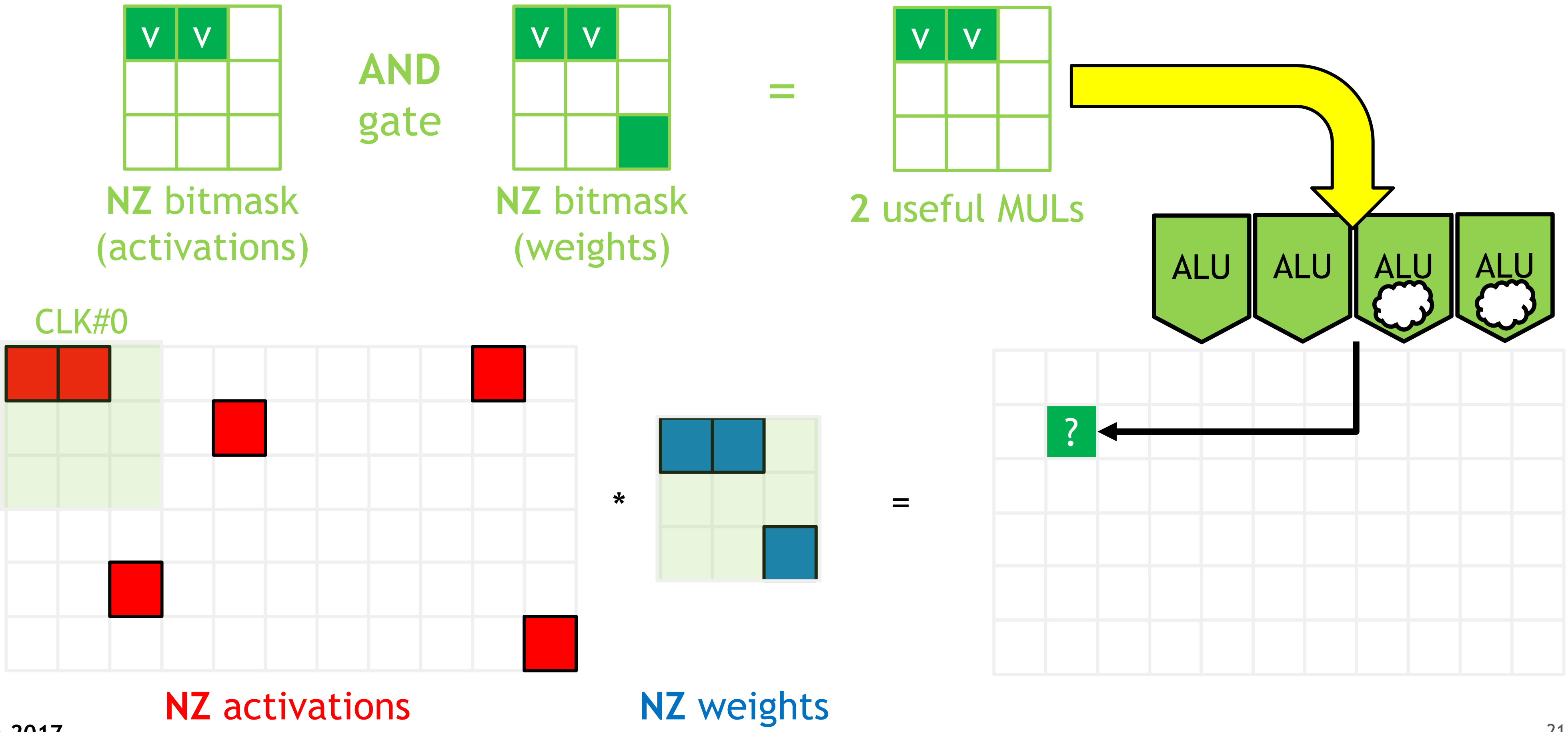
# Option 1: Leverage Dense CNN Design

## Employ pair of bit-masks to track non-zero weights and/or activations



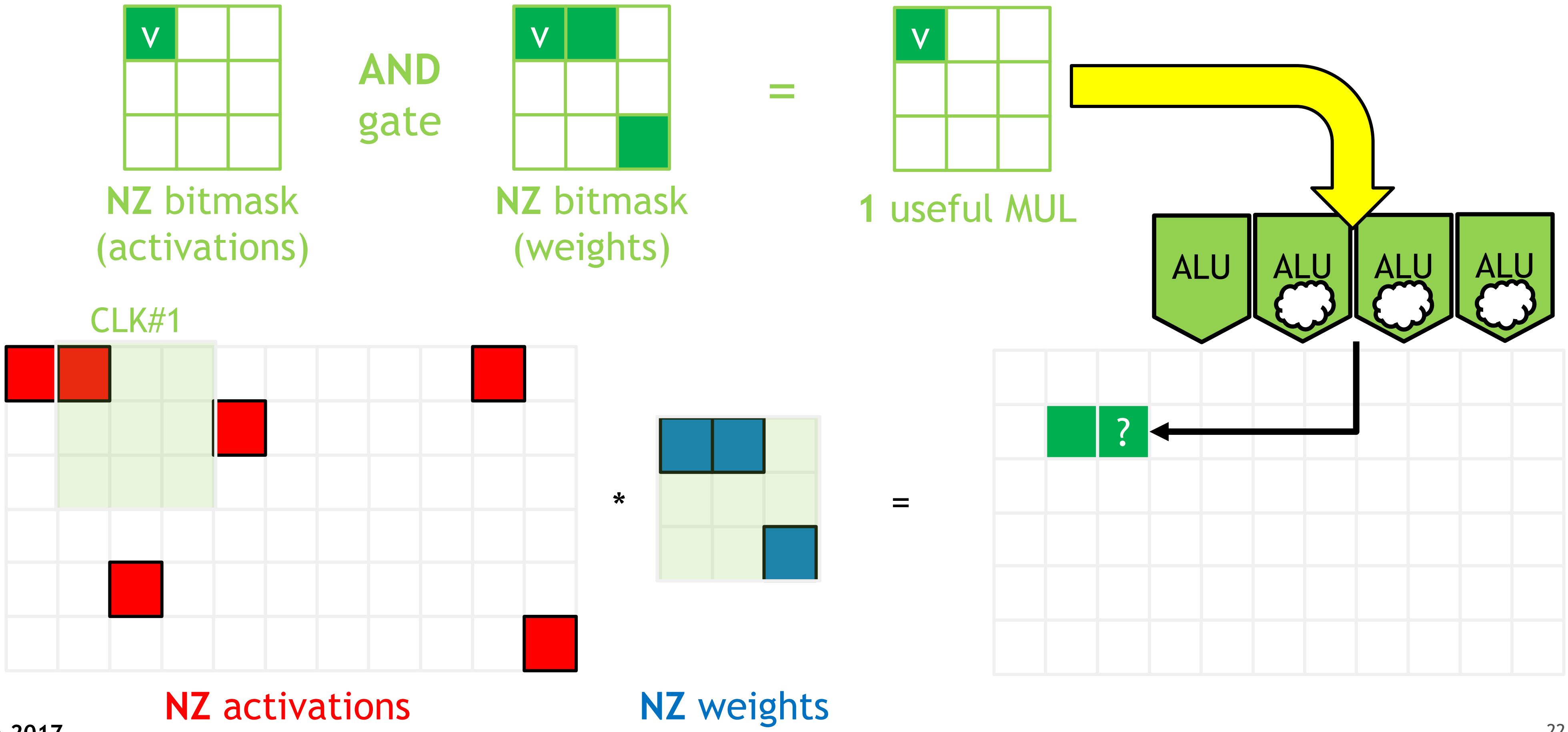
# Option 1: Leverage Dense CNN Design

## Employ pair of bit-masks to track non-zero weights and/or activations



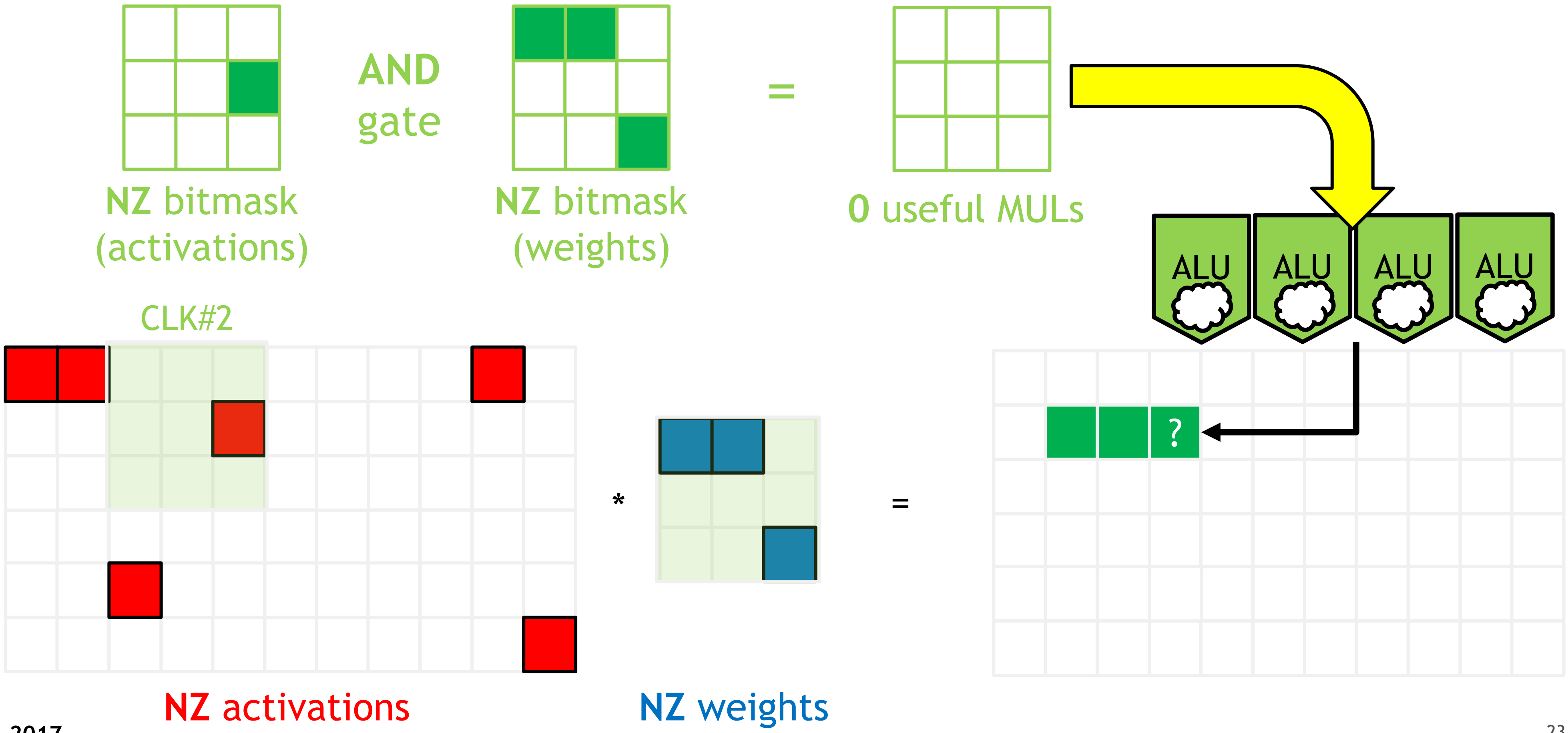
# Option 1: Leverage Dense CNN Design

## Employ pair of bit-masks to track non-zero weights and/or activations



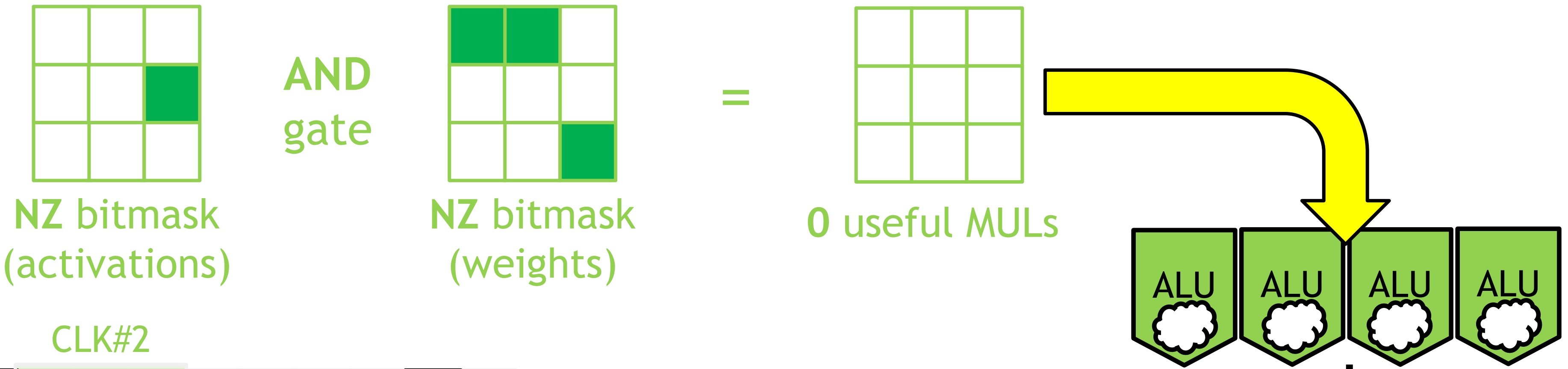
# Option 1: Leverage Dense CNN Design

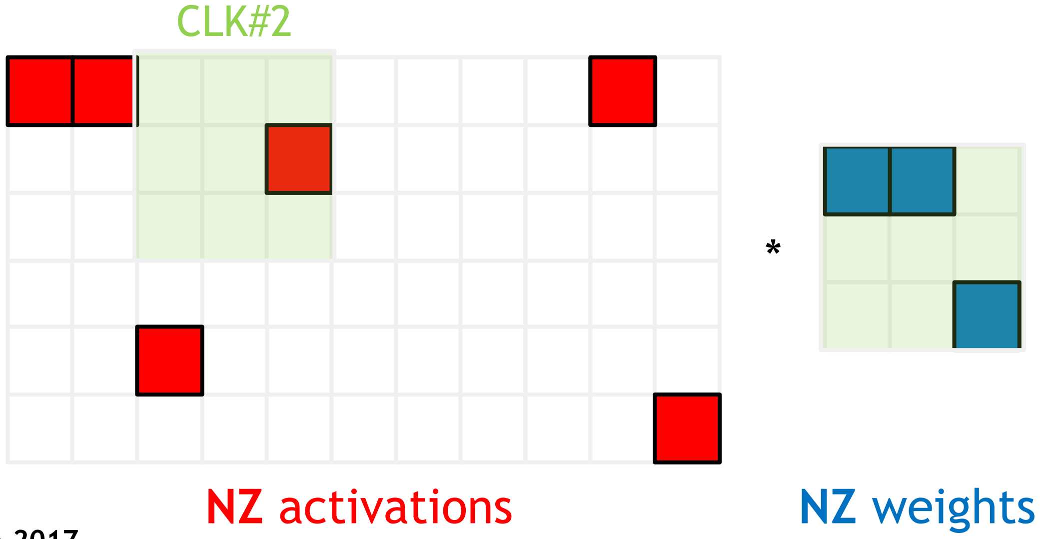
## Employ pair of bit-masks to track non-zero weights and/or activations

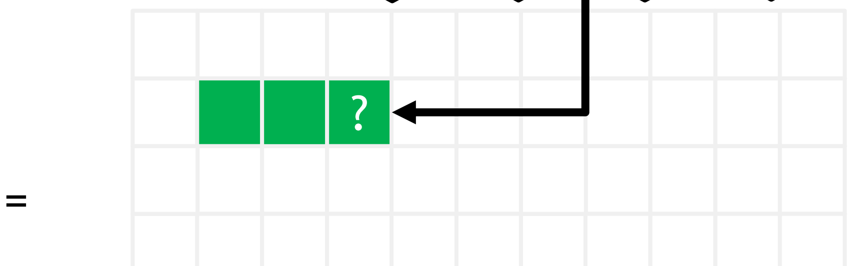


# Option 1: Leverage Dense CNN Design

## Employ pair of bit-masks to track non-zero weights and/or activations




**Challenge:** find enough useful MULs to fully populate the vector ALUs

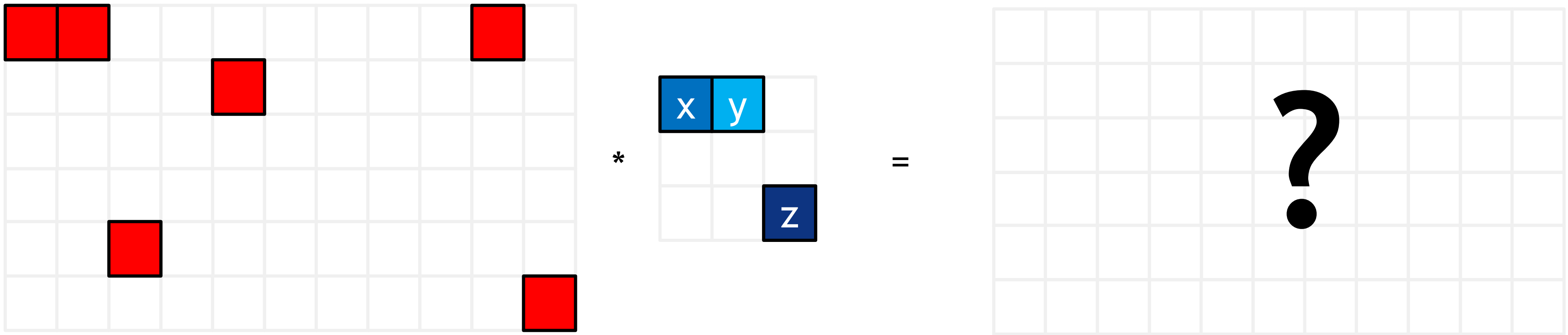
# SCNN Intuition & Approach

# Intuition behind SCNN

## Forget the sliding windows based convolution

All NZ activations must (at some point in time) be multiplied by all NZ weights

Holds true for convolution stride ‘1’

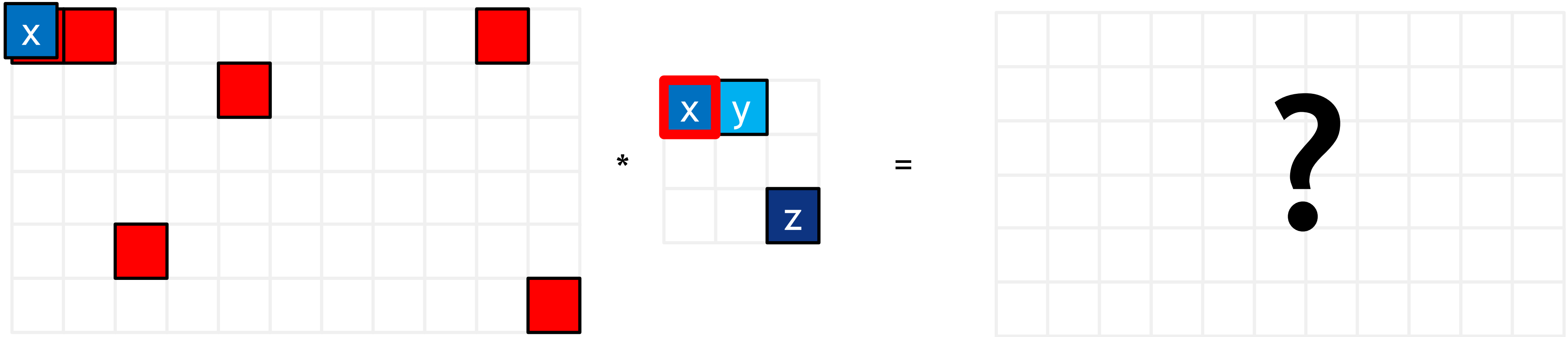


# Intuition behind SCNN

## Forget the sliding windows based convolution

All NZ activations must (at some point in time) be multiplied by all NZ weights

Holds true for convolution stride ‘1’

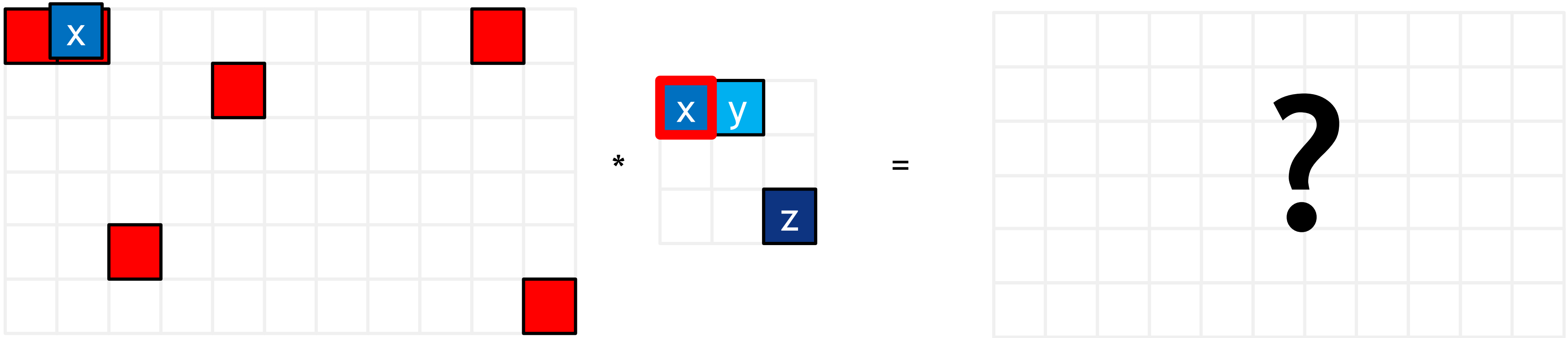


# Intuition behind SCNN

## Forget the sliding windows based convolution

All NZ activations must (at some point in time) be multiplied by all NZ weights

Holds true for convolution stride ‘1’

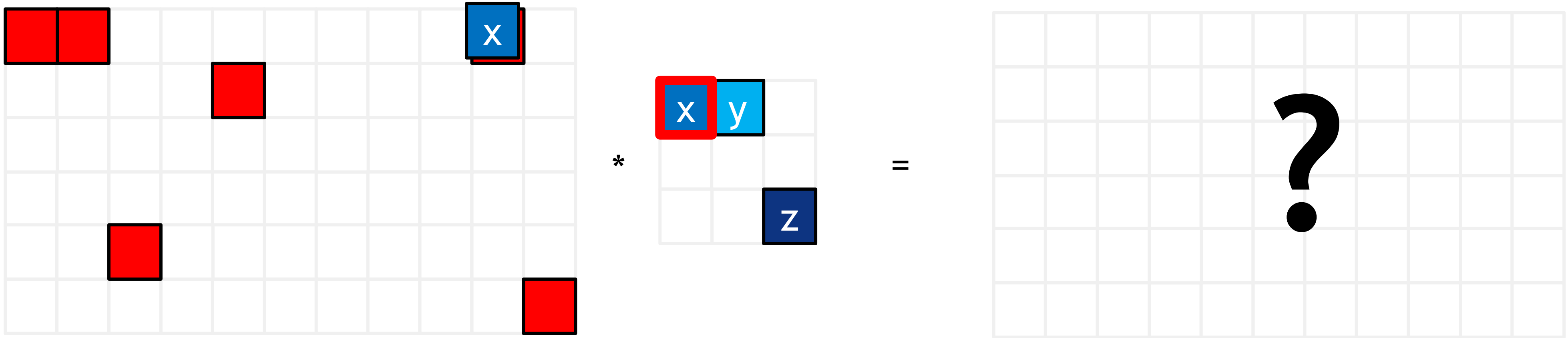


# Intuition behind SCNN

## Forget the sliding windows based convolution

All NZ activations must (at some point in time) be multiplied by all NZ weights

Holds true for convolution stride '1'

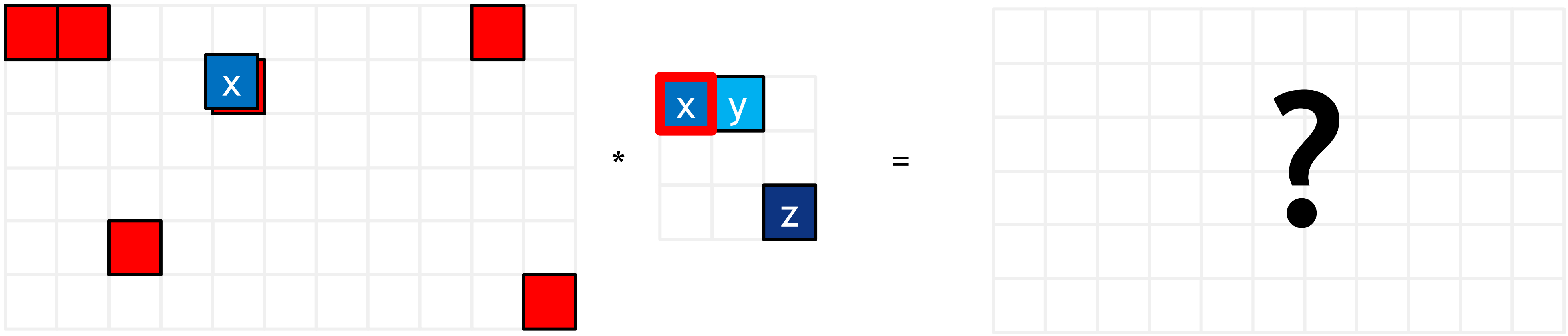


# Intuition behind SCNN

## Forget the sliding windows based convolution

All NZ activations must (at some point in time) be multiplied by all NZ weights

Holds true for convolution stride '1'

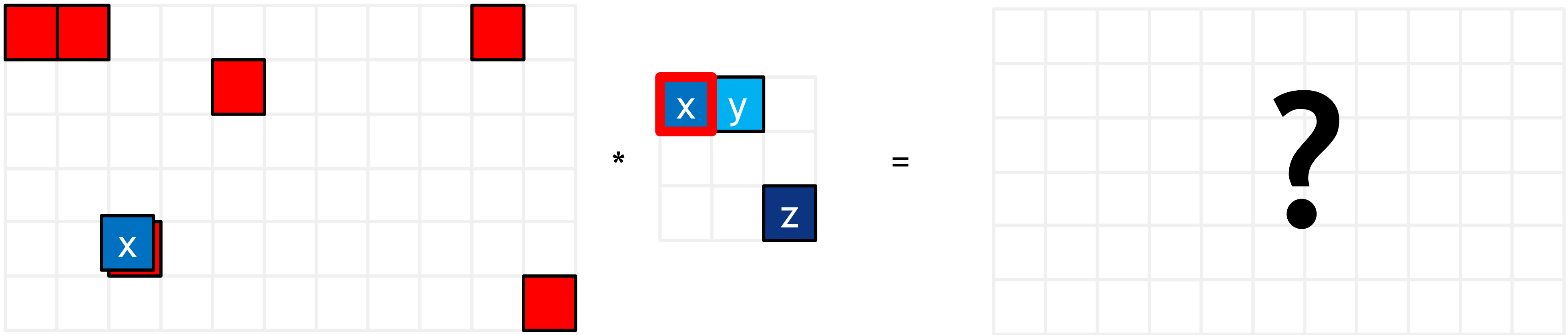


# Intuition behind SCNN

## Forget the sliding windows based convolution

All NZ activations must (at some point in time) be multiplied by all NZ weights

Holds true for convolution stride ‘1’

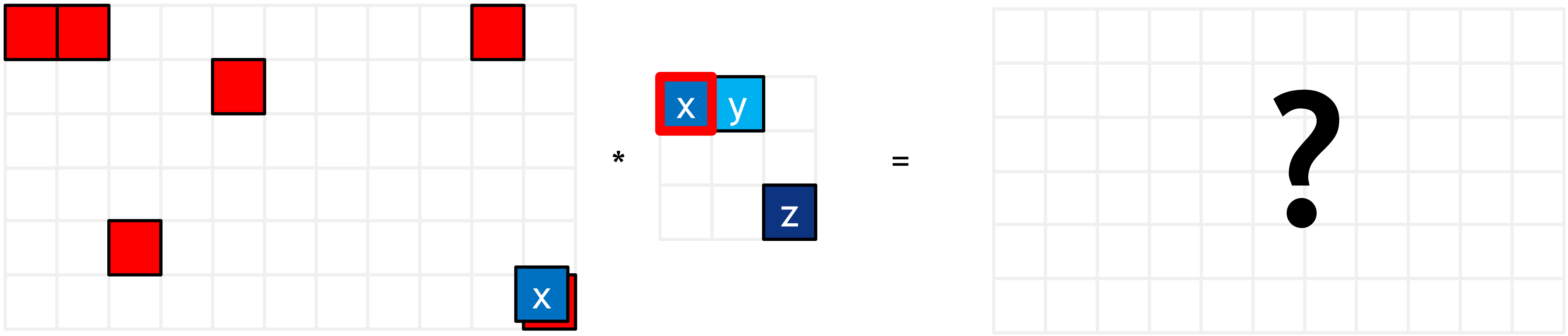


# Intuition behind SCNN

## Forget the sliding windows based convolution

All NZ activations must (at some point in time) be multiplied by all NZ weights

Holds true for convolution stride '1'

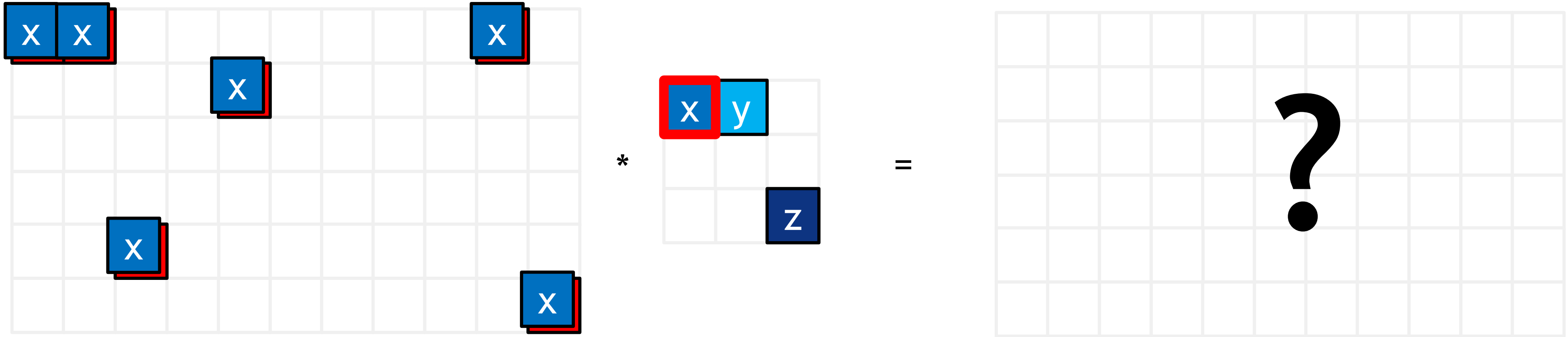


# Intuition behind SCNN

## Forget the sliding windows based convolution

All NZ activations must (at some point in time) be multiplied by all NZ weights

Holds true for convolution stride '1'

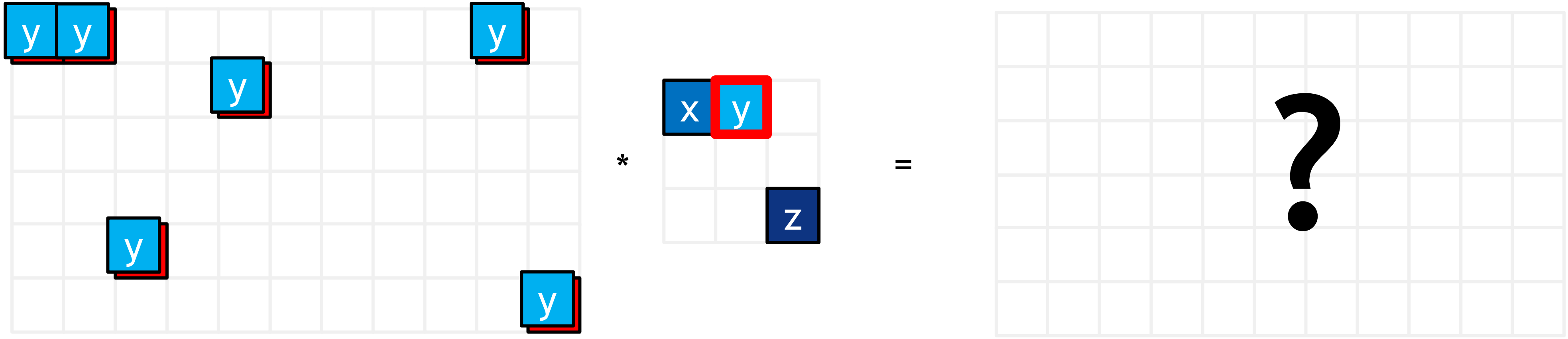


# Intuition behind SCNN

## Forget the sliding windows based convolution

All NZ activations must (at some point in time) be multiplied by all NZ weights

Holds true for convolution stride '1'

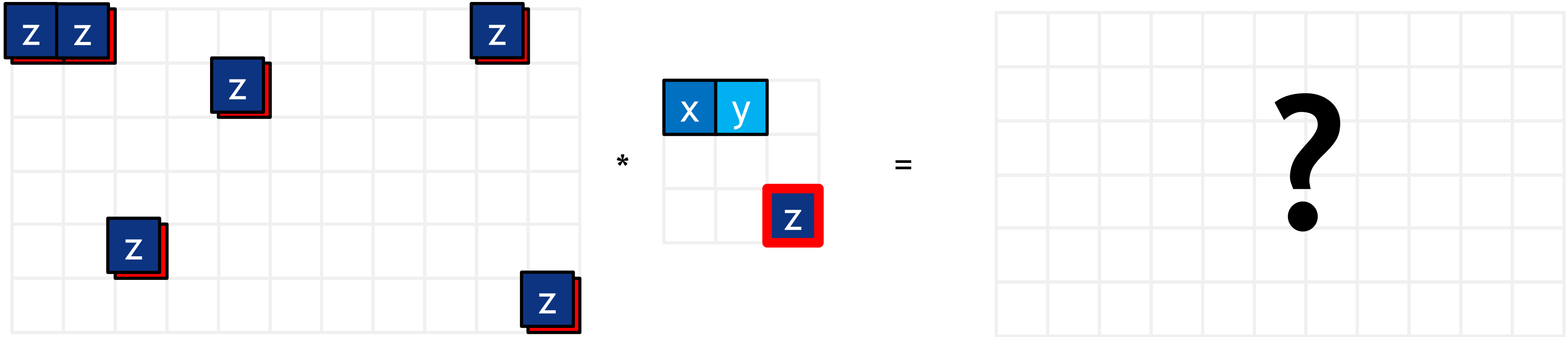


# Intuition behind SCNN

## Forget the sliding windows based convolution

All NZ activations must (at some point in time) be multiplied by all NZ weights

Holds true for convolution stride ‘1’

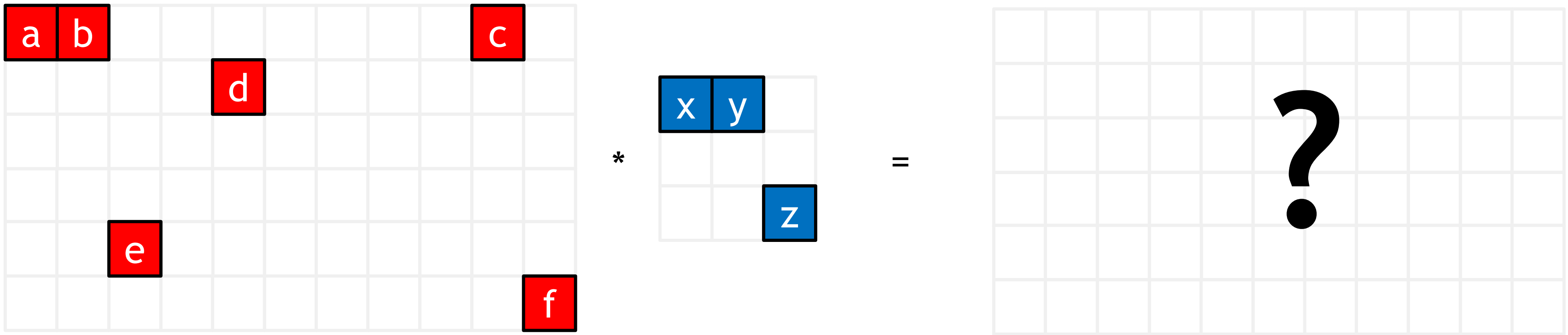


# The SCNN approach

## The *Cartesian* product (i.e., all-to-all) based convolution operation

Assuming a convolution stride of ‘1’:

Minimum # MULs: Cartesian product of the NZ activations and the NZ weights

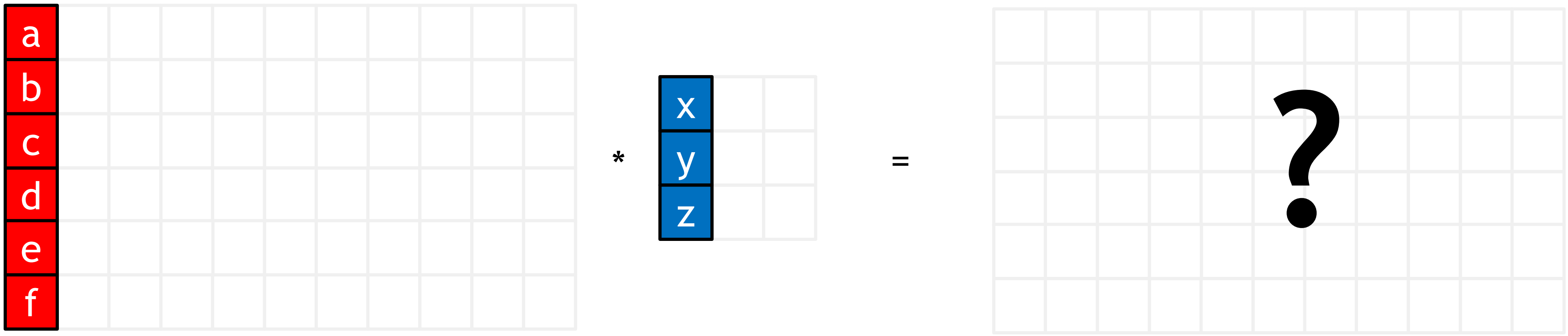


# The SCNN approach

## The *Cartesian* product (i.e., all-to-all) based convolution operation

Assuming a convolution stride of ‘1’:

Minimum # MULs: Cartesian product of the NZ activations and the NZ weights




Compress
NZ activations

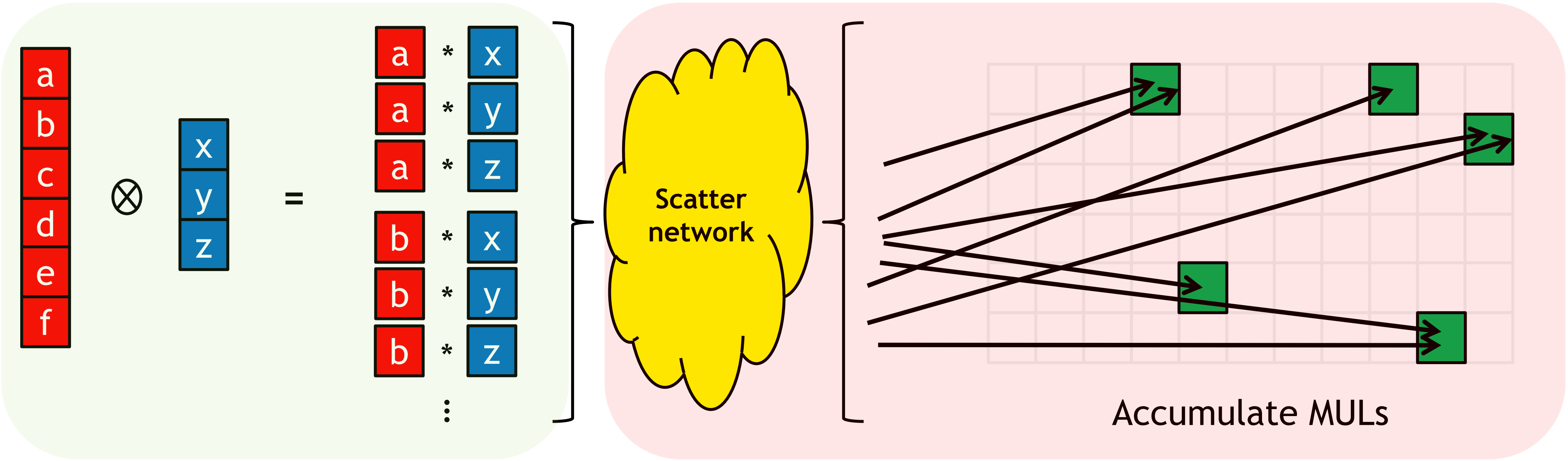
Compress
NZ weights

# The SCNN approach

## The *Cartesian* product (i.e., all-to-all) based convolution operation

Assuming a convolution stride of '1':

Minimum # MULs: Cartesian product of the NZ activations and the NZ weights



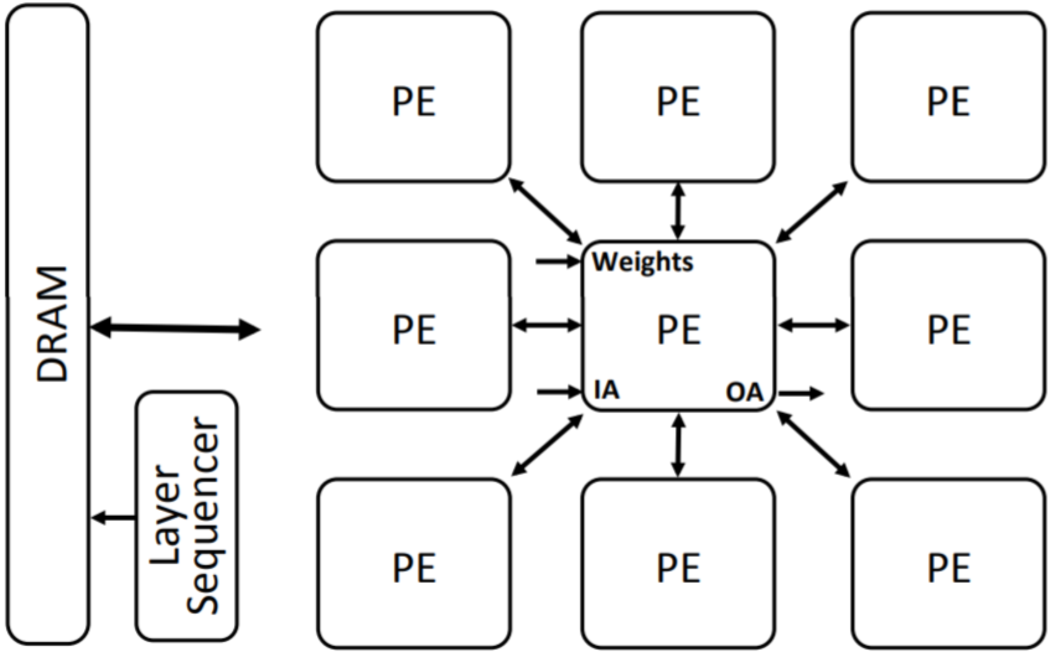

PE frontend

PE backend

# SCNN Architecture

# SCNN Architecture

## 2D spatially arranged processing elements (PEs)

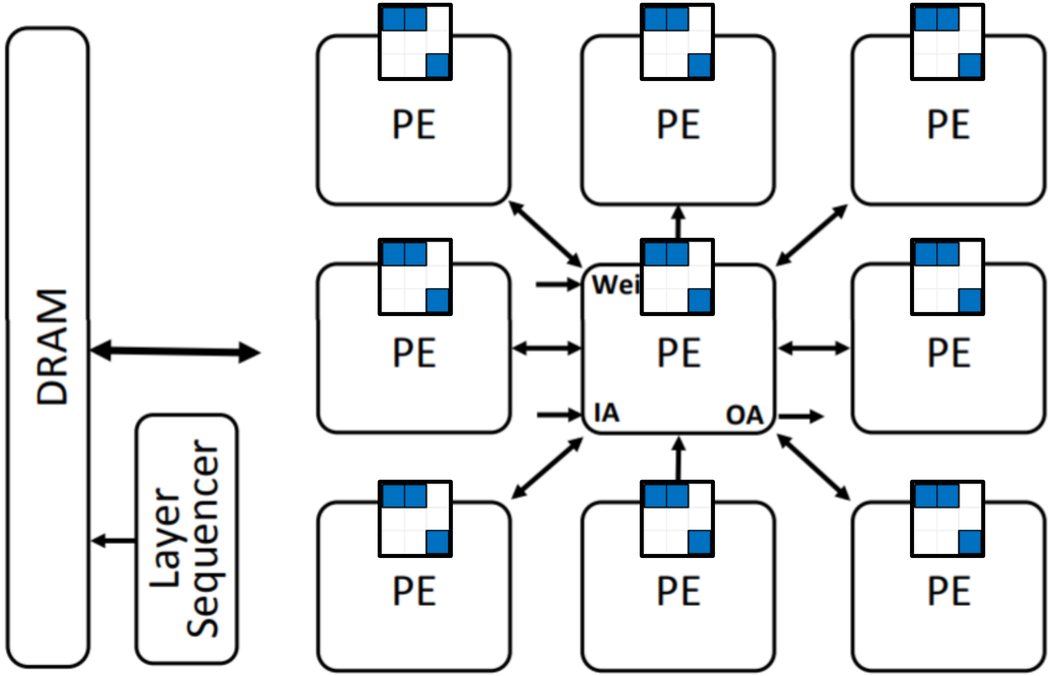


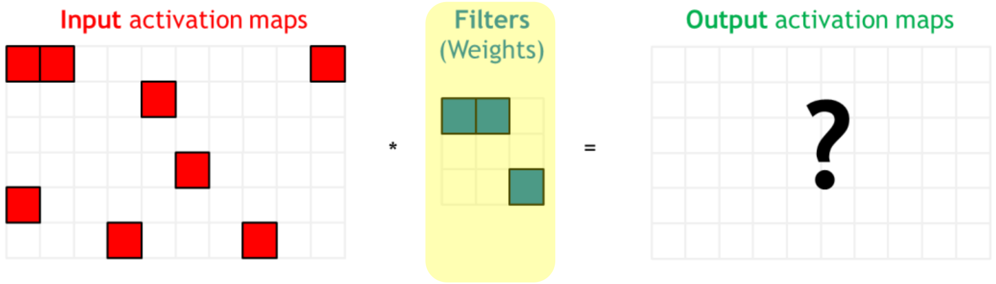
# SCNN Architecture

## Workload distribution

Weights *broadcast* to all PEs

All PEs have a copy of all the NZ weights of the CNN model



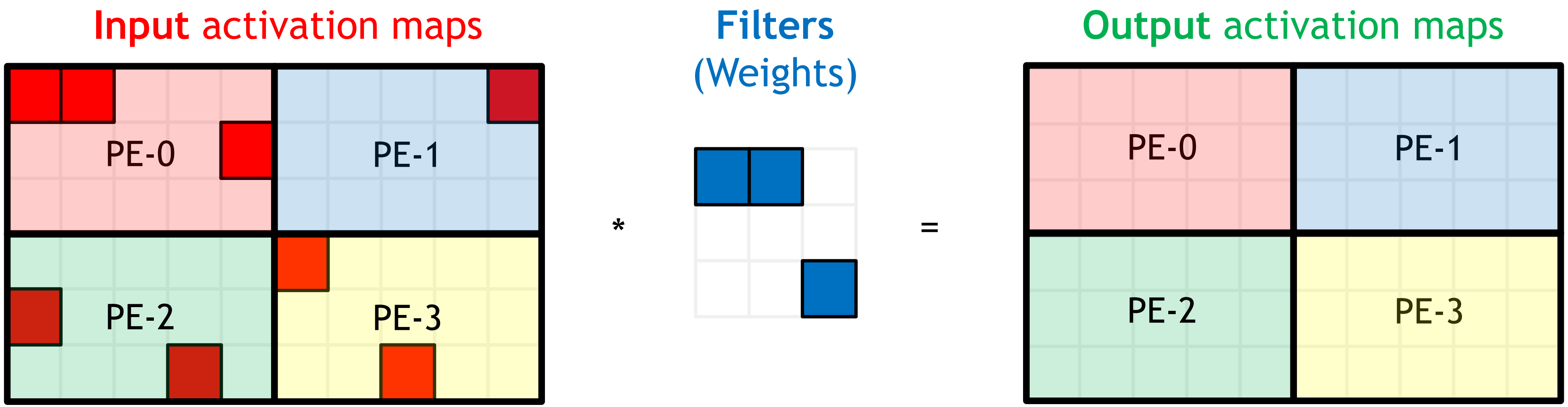



# SCNN Architecture

## Workload distribution

Each PE is allocated with a *partial* volume of input and output activations

Input and output activations stay local to PE

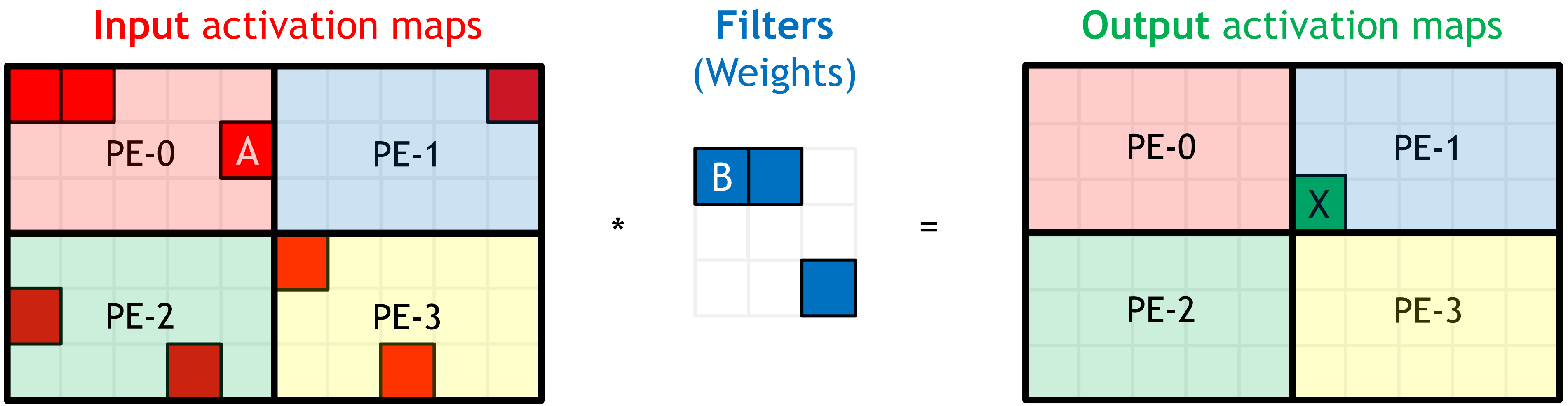


# SCNN Architecture

## Halo resolution?

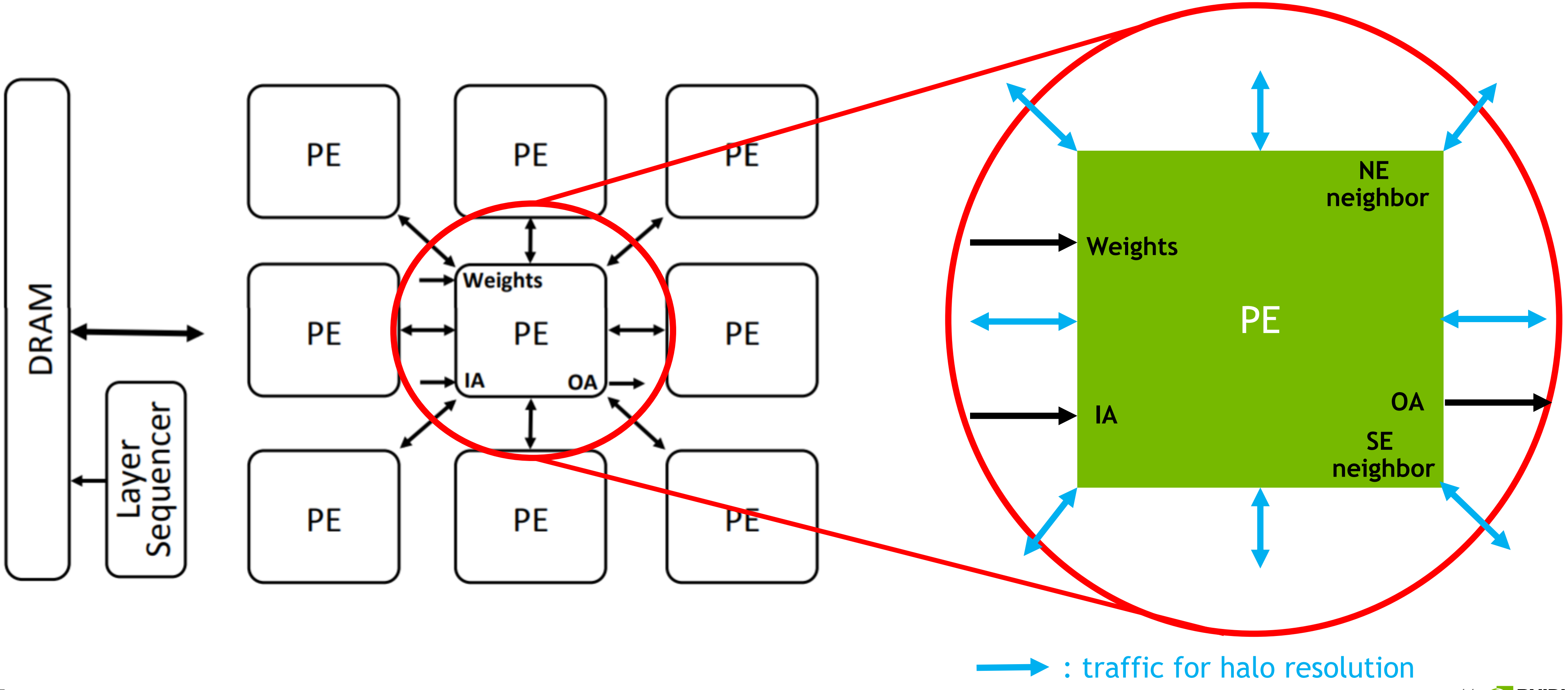
Output halos:

PE-0 calculates (A x B), but the result should be accumulated in PE-1 (X)



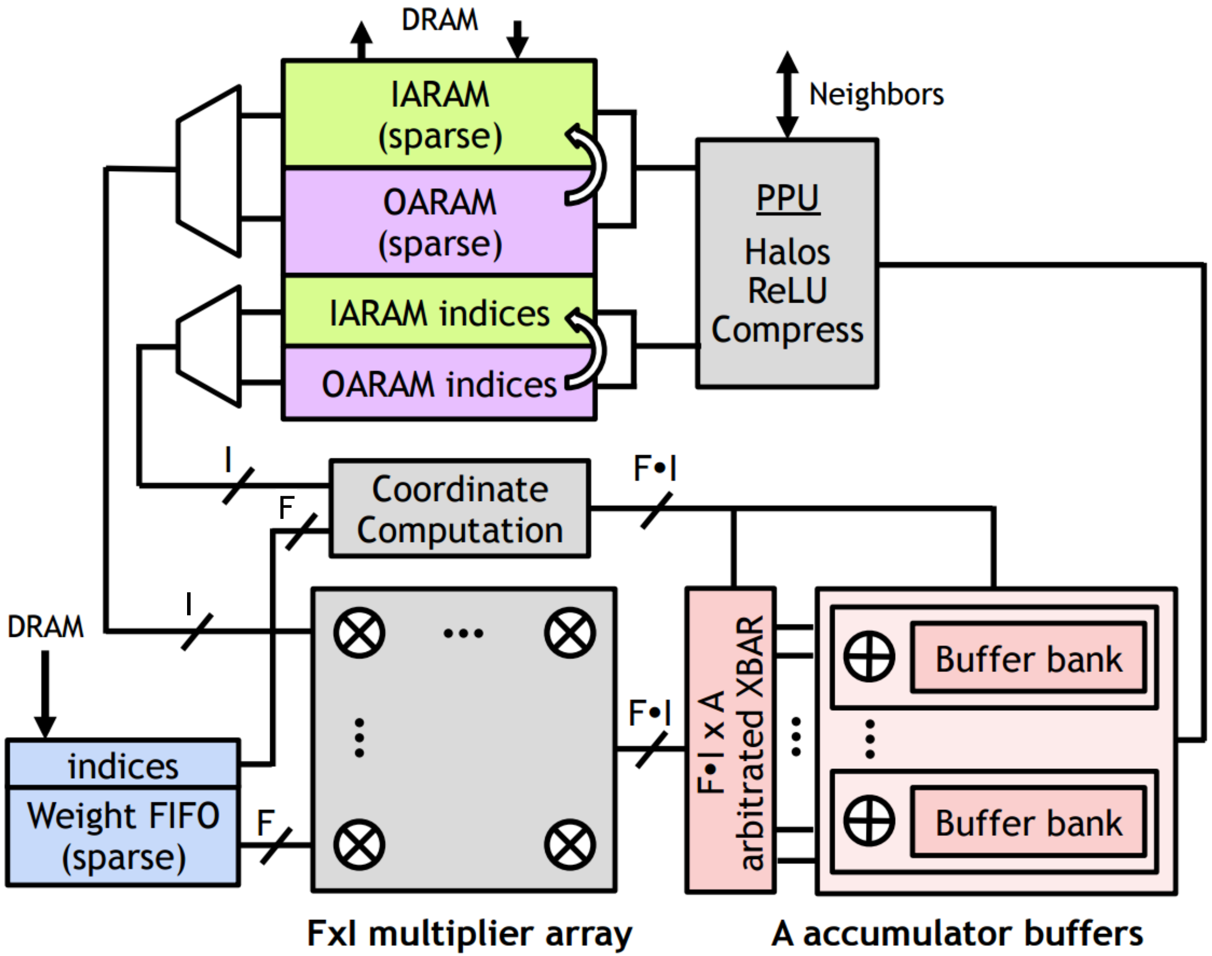
# SCNN Architecture

## Inter-PE communication channel for halo resolution



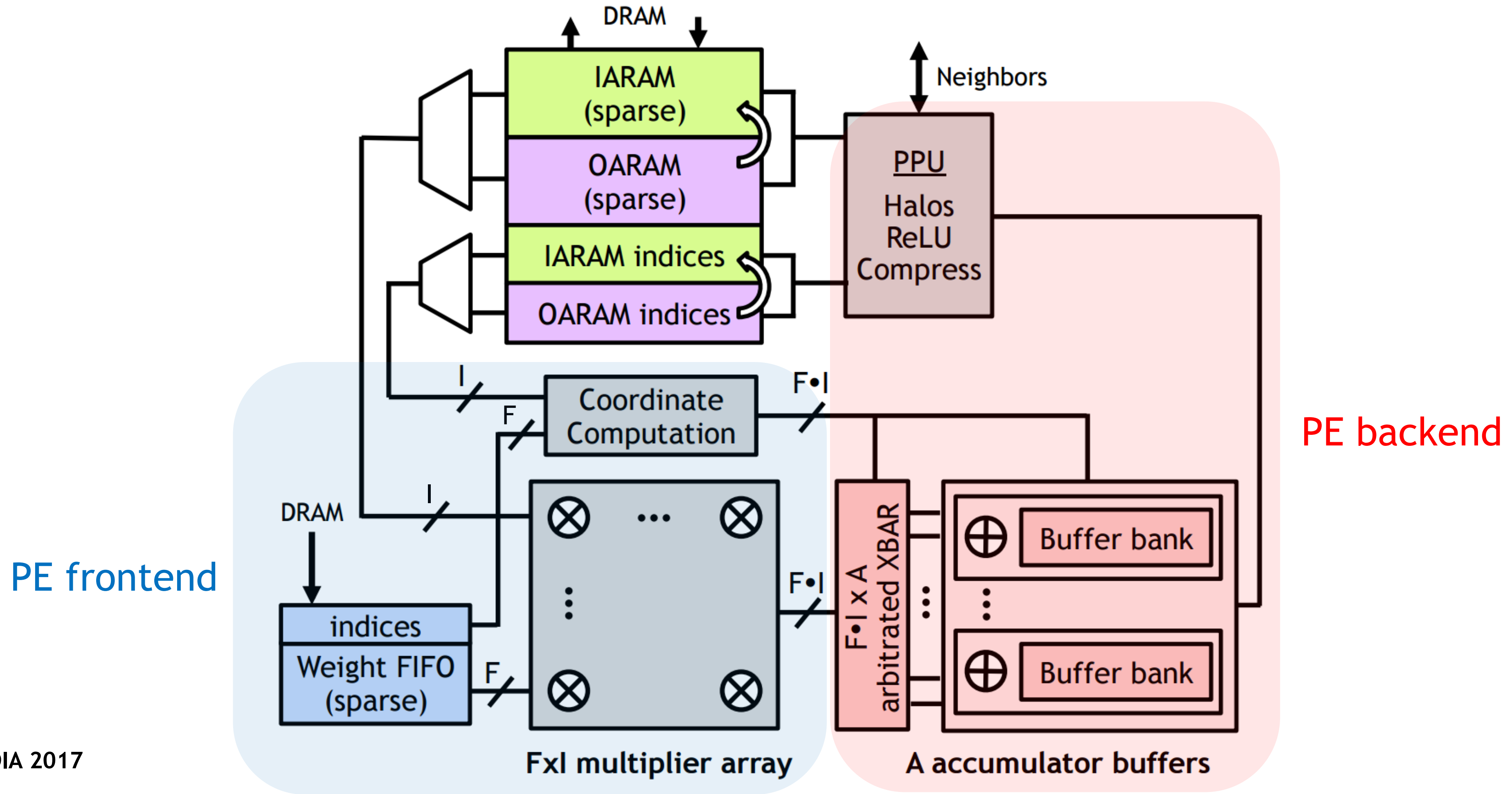
# SCNN PE microarchitecture

## (Compressed-sparse frontend) + (Scattered-dense backend)



# SCNN PE microarchitecture

## (Compressed-sparse frontend) + (Scattered-dense backend)



# Evaluation

# Evaluation

## Methodology

Network models and input activations

trained → pruned → retrained models using Caffe

Area & power

System-C → Catapult HLS → Verilog RTL → Synthesis of an SCNN PE

Performance & energy

Performance model for cycle-level simulation of SCNN

Analytical model for design space exploration (dataflows, sparse vs. dense)

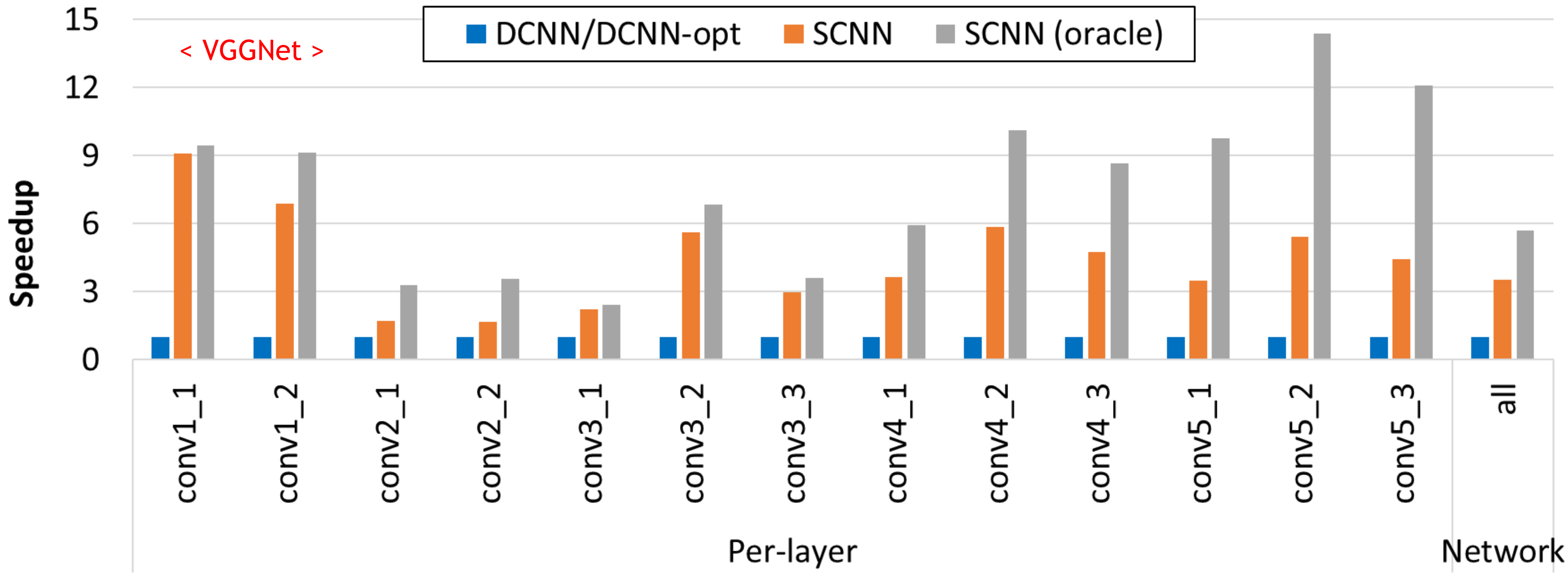
# Evaluation

## Architecture configurations

| | |
|---|---|
| DCNN: | operates solely on dense weights and activations |
| DCNN-opt: | DCNN with (de)compression of activations + ALU power-gating |
| SCNN: | sparse-optimized CNN accelerator |

# Performance

## Dense vs. sparse

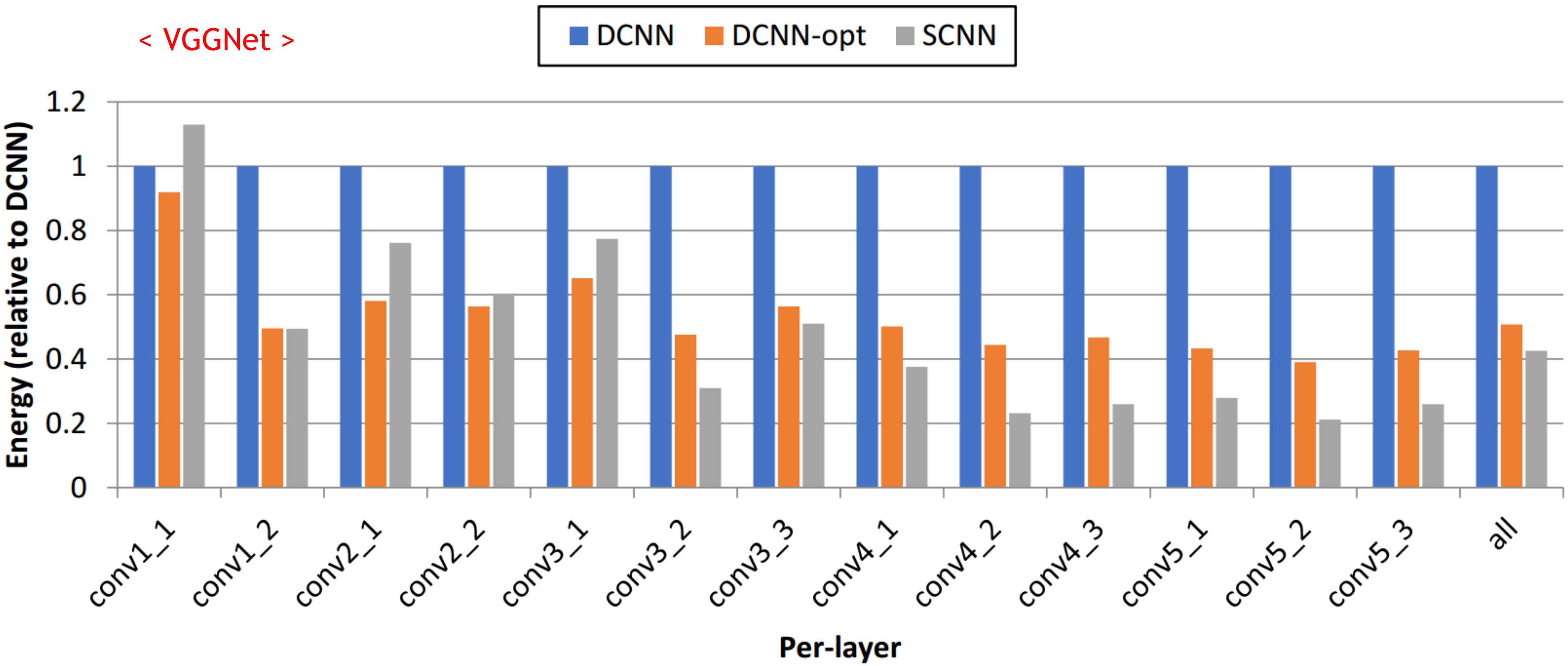





* Network-wide improvement over DCNN: AlexNet (2.37x), GoogLeNet (2.19x)

# Energy consumption

## Dense vs. sparse




* Network-wide improvement over DCNN: AlexNet (2.13x), GoogLeNet (2.51x)

# Related Work

## Qualitative comparison to prior work

| Architecture | Sparse optimizations | | Convolution dataflow |
|---|---|---|---|
| | Weights | Activations | |
| DaDianNao [ASPLOS ‘14] | - | - | (Variant of) Sliding-window |
| Eyeriss [ISCA ‘16] | - | Power-gating | |
| CNVLUTIN [ISCA ‘16] | - | Zero-skipping | |
| Cambricon-X [MICRO ’16] | Zero-skipping | - | |
| SCNN | Zero-skipping | | Cartesian-product |

# Follow-up questions?

## Contacts the authors

Technical leads:

| | |
|---|---|
| SCNN architecture & sparse models: | Minsoo Rhu |
| TimeLoop (CNN analytical model): | Angshuman Parashar |
| Power & area modeling: | Rangharajan Venkatesan |

# Conclusion

## SCNN: a compressed-sparse CNN accelerator

Novel Cartesian-product based convolution operation

- Simple/compressed/sparse PE frontend
- Scatter/dense PE backend

Superior performance and energy-efficiency

- Average 2.7x higher performance than dense CNN architecture
- Average 2.3x higher energy-efficiency than dense CNN architecture