Designing Bespoke CNNs for Target Hardware

Woonhyun Nam
Director, Algorithms
StradVision
September, 2020

StradVision

- We develop deep learning-based perception software for Advanced Driver Assistance Systems (ADAS) and Autonomous Vehicles

For more detailed information, please refer to https://youtu.be/tK7F8yvpiGA and https://stradvision.com/

STRADVISION

Agenda

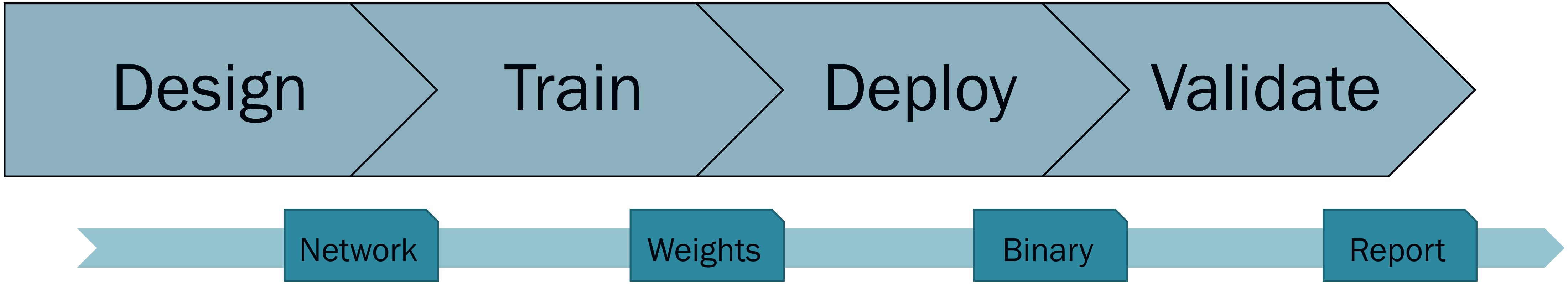
- Development Process of Deep Learning-based Software
- Scalability Challenge
- Layer Transformations
- Case Studies
- Conclusions

Development Process of Deep Learning-based Software

STRADVISION

Development Process

Repeated for each task (e.g., object detection) and target platform



(Pros)

- Reuse existing CNN designs
- Tweak some layers

(Cons)

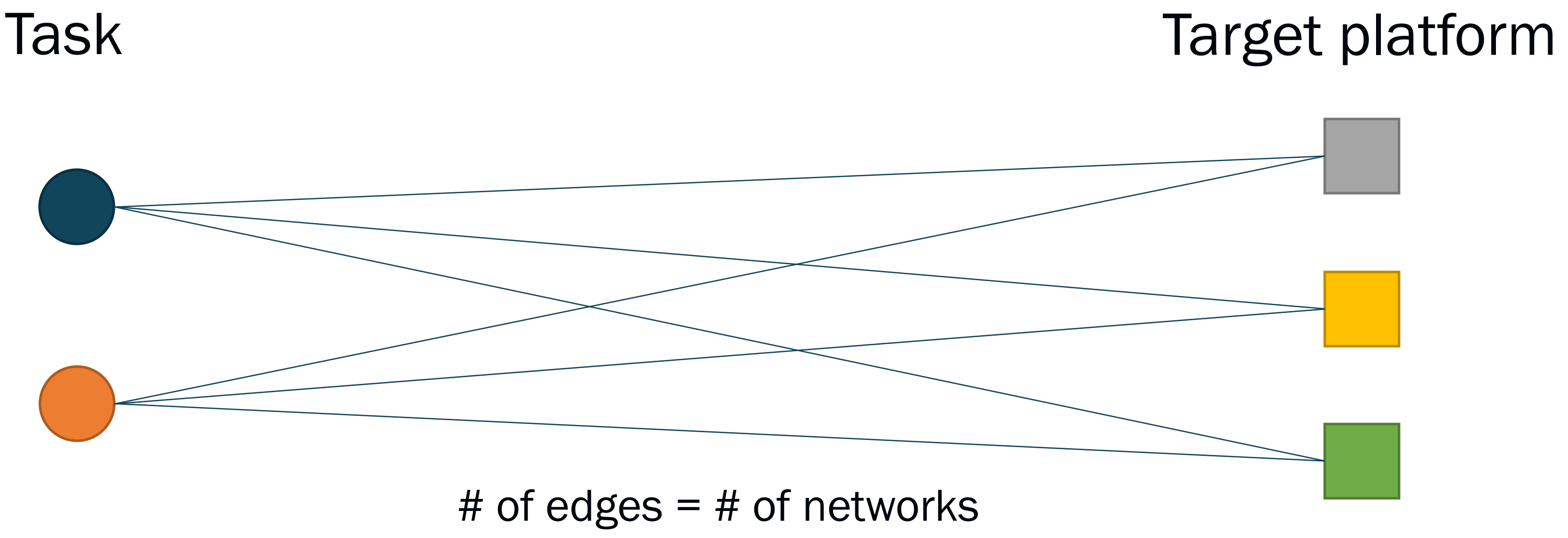
- Retrain from scratch even for small changes
- Revalidate from scratch if output changes

Scalability Challenge

Development Process

Scalability challenge

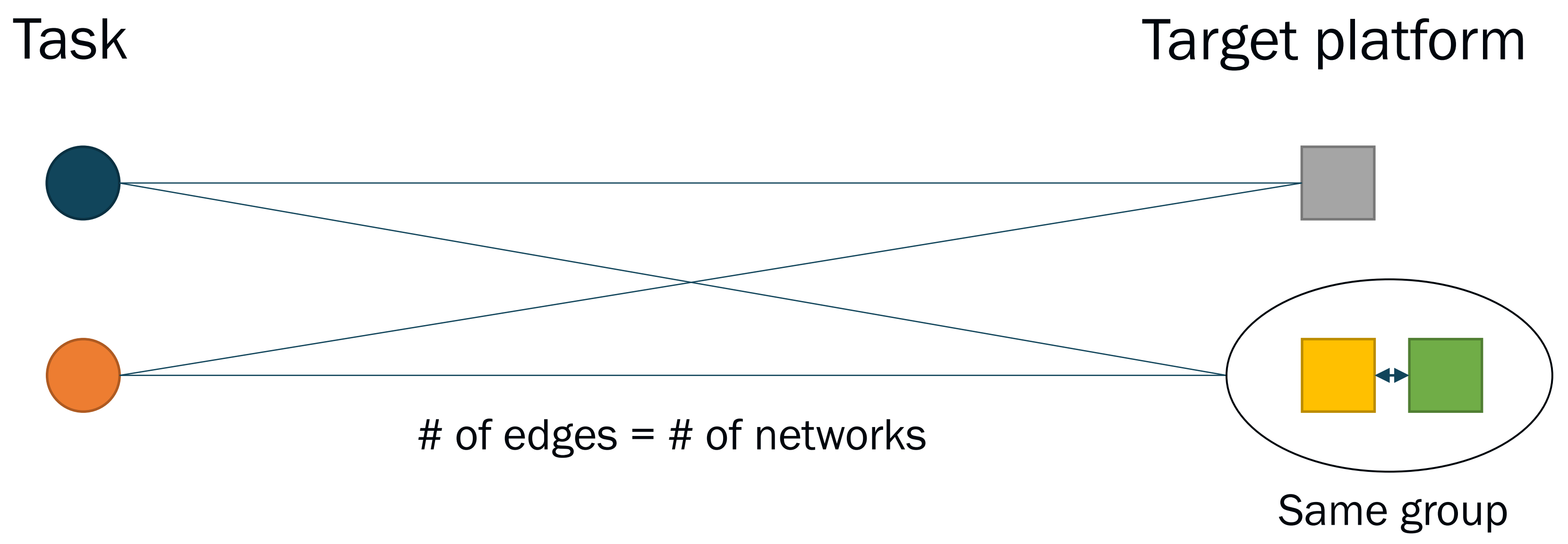
- Number of networks to maintain grows rapidly
- More than 100 networks become difficult to manage



Scalable Development Process

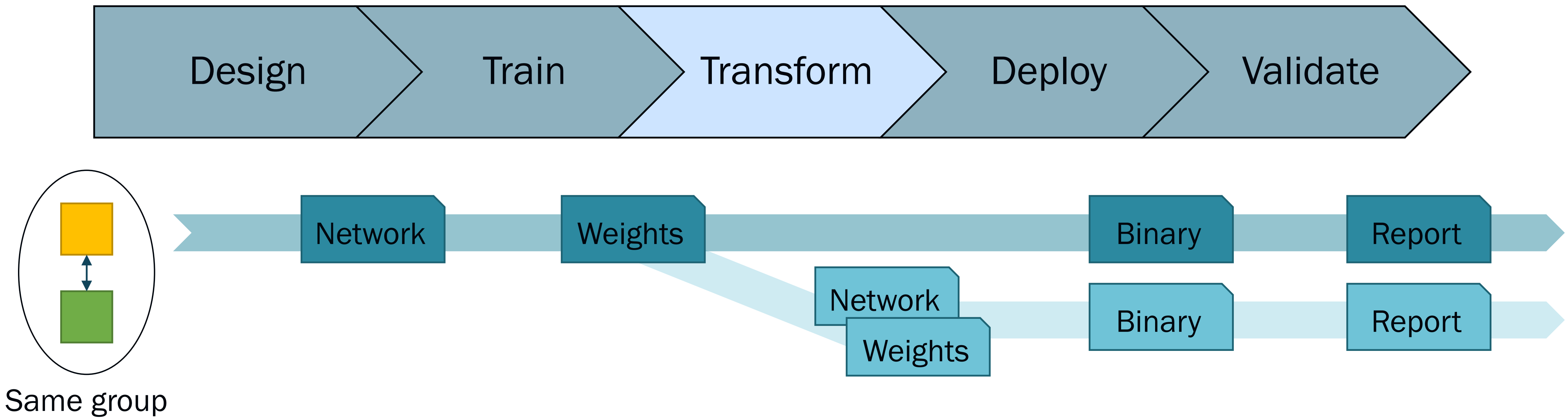
Main idea

- Grouping target platforms based on similar computational characteristics (or capabilities) with respect to a given CNN transformation



Scalable Development Process

- Design/Train CNNs for each group
- Transform CNNs without retraining within same group



Scalable Development Process

Cost-effective method

- Increasing computational efficiency
  - E.g., lower latency, higher throughput, cheaper hardware
- Increasing engineering efficiency
  - E.g., shorter development time, less engineers, more projects
- Increasing economic efficiency
  - E.g., lower usage of cloud services for training, lower production-cost

Layer Transformations

Layer Transformations

- Basic operations
- Case studies
    - YUV Input
    - Fixed-size Kernel Acceleration in Convolution
    - No FC Support
    - No Custom Layer Support in Quantization
    - Block-level Accumulation in Convolution

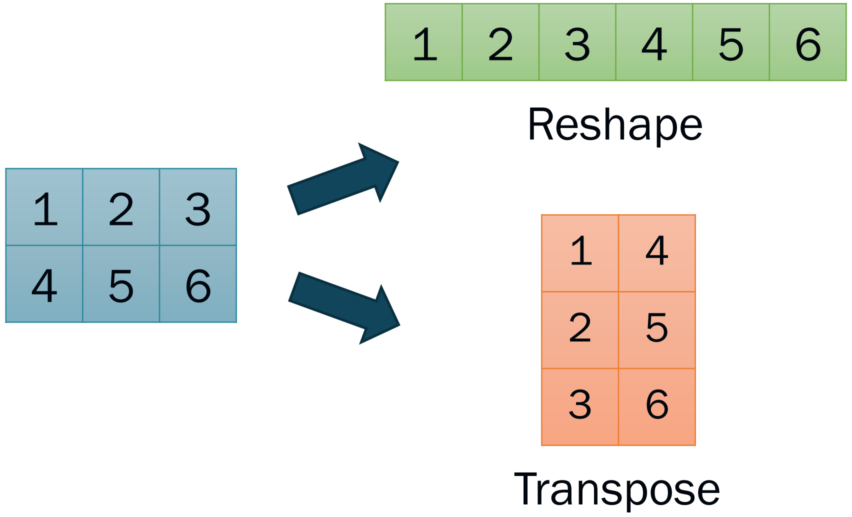
Layer Transformations

Basic operations

- (Linear) layer fusion
  - Conv & Conv → Conv
- Memory layout change
  - For Input, Weight, and Output tensors
  - Reshape, Transpose
- Zero-filling

$y = ax + b$ and $z = cy + d$

$\rightarrow \quad z = \underbrace{(ac)}_{\text{New weight}}x + \underbrace{(bc + d)}_{\text{New bias}}$



Case Studies

Case Study – YUV Input

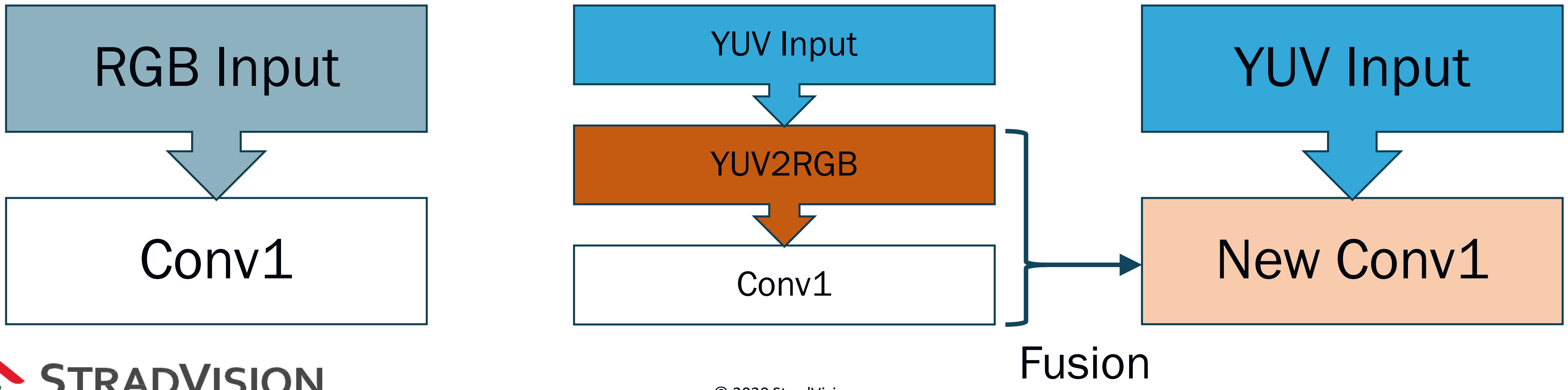
Hypothetical target hardware

- YUV input, instead of RGB
- YUV2RGB equation is given

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} w_{RY} & w_{RU} & w_{RV} \\ w_{GY} & w_{GU} & w_{GV} \\ w_{BY} & w_{BU} & w_{BV} \end{bmatrix} \begin{bmatrix} Y + b_Y \\ U + b_U \\ V + b_V \end{bmatrix}$$

YUV2RGB equation

→ Fusion of YUV2RGB and the first Conv



Case Study – YUV Input

Equations for new weights and new bias

$$Z = \begin{bmatrix} w_R & w_G & w_B \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + b$$

Conv1 equation

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} w_{RY} & w_{RU} & w_{RV} \\ w_{GY} & w_{GU} & w_{GV} \\ w_{BY} & w_{BU} & w_{BV} \end{bmatrix} \begin{bmatrix} Y + b_Y \\ U + b_U \\ V + b_V \end{bmatrix}$$

YUV2RGB equation

Fusion

$$Z = \underbrace{\begin{bmatrix} w_R & w_G & w_B \end{bmatrix} \begin{bmatrix} w_{RY} & w_{RU} & w_{RV} \\ w_{GY} & w_{GU} & w_{GV} \\ w_{BY} & w_{BU} & w_{BV} \end{bmatrix}}_{\text{New weight}} \begin{bmatrix} Y \\ U \\ V \end{bmatrix} + \underbrace{b + \begin{bmatrix} w_R & w_G & w_B \end{bmatrix} \begin{bmatrix} w_{RY} & w_{RU} & w_{RV} \\ w_{GY} & w_{GU} & w_{GV} \\ w_{BY} & w_{BU} & w_{BV} \end{bmatrix} \begin{bmatrix} b_Y \\ b_U \\ b_V \end{bmatrix}}_{\text{New bias}}$$

New Conv1 equation

Case Study – Fixed-size Kernel Acceleration in Convolution

Hypothetical target hardware

- Only 5x5 kernel acceleration supported
- Applying zero-filling for smaller kernels
- Computational utilization
  - 1x1 Conv ~ 4%, 3x3 Conv ~ 36%

w_{11}	w_{12}	w_{13}
w_{21}	w_{22}	w_{23}
w_{31}	w_{32}	w_{33}

Zero filling

0	0	0	0	0
0	w_{11}	w_{12}	w_{13}	0
0	w_{21}	w_{22}	w_{23}	0
0	w_{31}	w_{32}	w_{33}	0
0	0	0	0	0

Case Study – Fixed-size Kernel Acceleration in Convolution

→ Convert 1x1 Conv into 4(kh)x1(kw) Conv

iw=input width	kw=kernel width	ow=output width
ih=input height	kh=kernel height	oh=output height
ic=input channel		oc=output channel

Input	Weight	Output.reshape(oc,oh,ow)
(ic)x(ih)x(iw)	(oc)x(ic)x(1)x(1)	(oc)x(oh)x(ow)
	1x1 Convolution →	
Reshape ↓	Reshape ↓	↑ Reshape
Input.reshape(ic/4,4,ih*iw)	Weight.reshape(oc,ic/4,4,1)	Output
(ic/4)x(4)x(ih*iw)	(oc)x(ic/4)x(4)x(1)	(oc)x(1)x(oh*ow)
	4x1 Convolution →	

STRADVISION

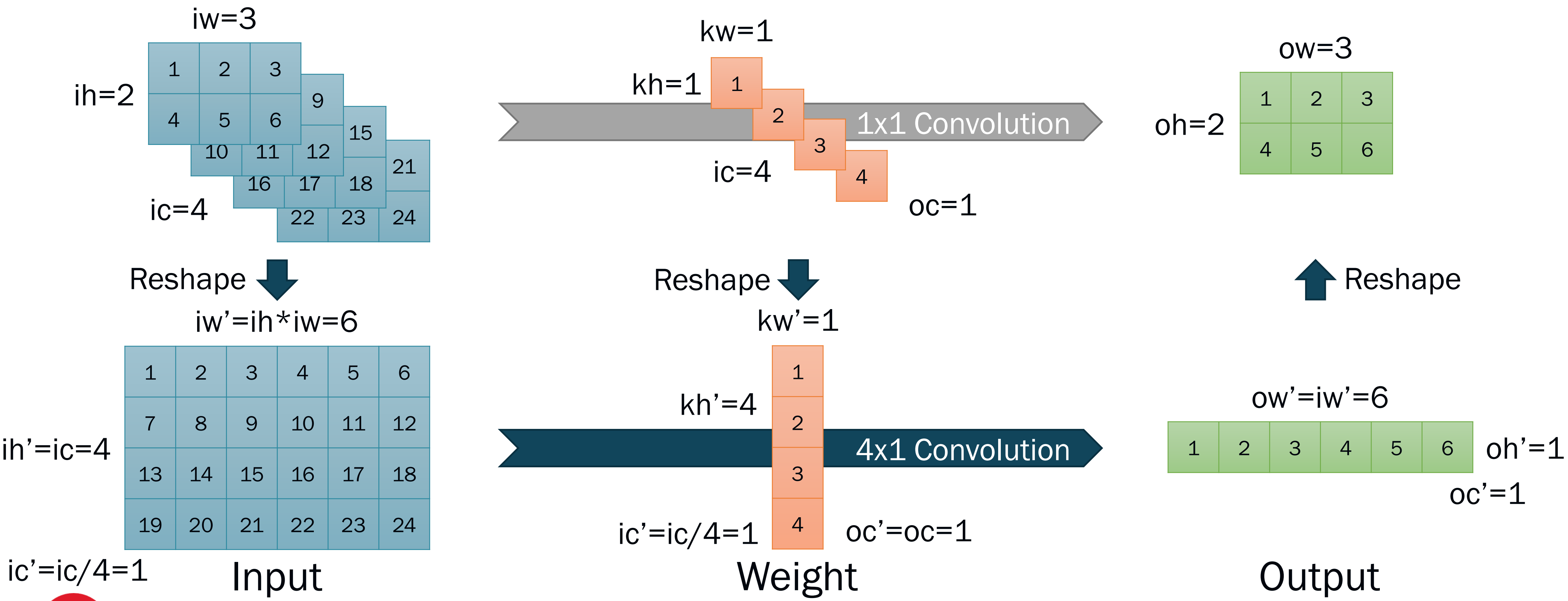
Case Study – Fixed-size Kernel Acceleration in Convolution

→ Convert 1x1 Conv into 4(kh)x1(kw) Conv

iw=input width
ih=input height
ic=input channel

kw=kernel width
kh=kernel height

ow=output width
oh=output height
oc=output channel



STRADVISION

Case Study – No FC Support

Hypothetical target hardware

- No support for FC layers, or
- Only single-batched FC layers supported

$$\underset{\text{Input}}{\begin{array}{cc} & ic=3 \\ batch=2 & \begin{bmatrix}1&2&3\\4&5&6\end{bmatrix}\end{array}} * \underset{\text{Weight}}{\begin{array}{cc} & oc=2 \\ ic=3 & \begin{bmatrix}1&4\\2&5\\3&6\end{bmatrix}\end{array}} = \underset{\text{Output}}{\begin{array}{cc} & oc=2 \\ batch=2 & \begin{bmatrix}1&2\\3&4\end{bmatrix}\end{array}}$$

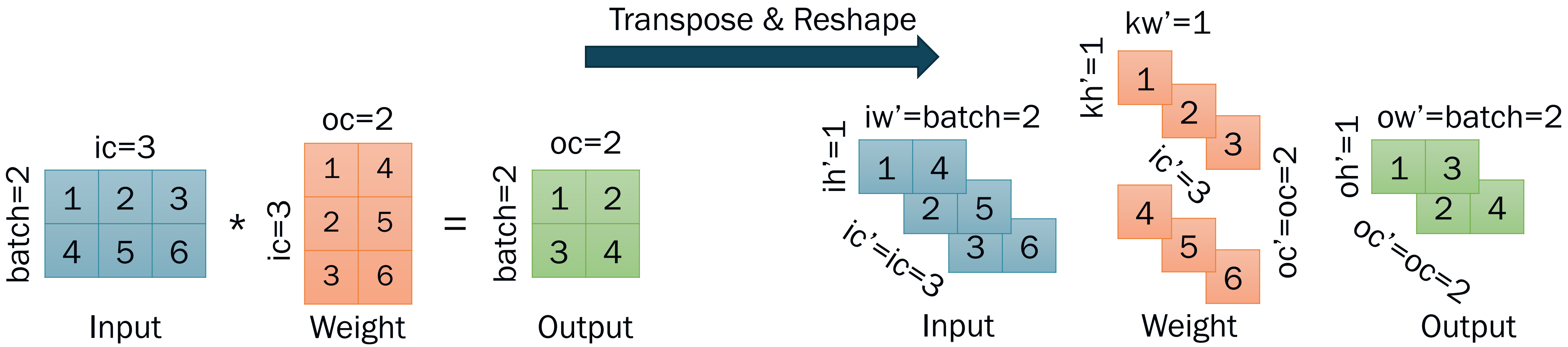
Matrix Multiplication in FC layer

Case Study – No FC Support

→ Convert FC into 1x1 Conv

Input (batch)x(ic)	*	Weight (ic)x(oc)	=	Output (batch)x(oc)

transpose(Input) .reshape(ic,1,batch) (ic)x(1)x(batch)	*	transpose(Weight) .reshape(oc,ic,1,1) (oc)x(ic)x(1)x(1)	=	transpose(Output) .reshape(oc,1,batch) (oc)x(1)x(batch)



FC

1x1 Conv

Case Study – No Custom Layer Support in Quantization

Hypothetical target hardware

- Quantization should be done by a tool provided by hardware vendor
- No support for custom layers
- The tool accepts only image files as quantization inputs

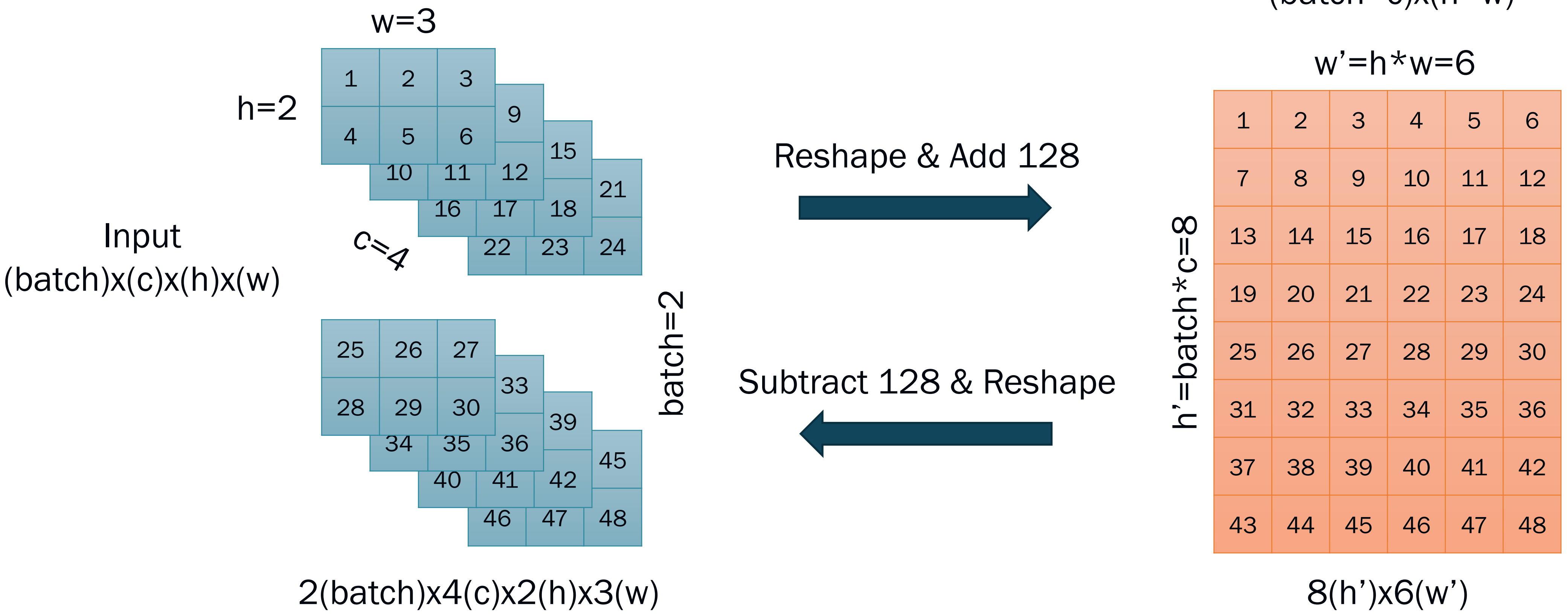
→ Convert input tensors of custom layers into image files

- Useful for two-stage detection networks (E.g., Faster RCNN)

Case Study – No Custom Layer Support in Quantization

Convert 4D tensors into image files

Input.reshape(batch*c,h*w) + 128
(batch*c)x(h*w)



Tensor in Int8 format

Gray image in Uint8 format

Case Study – Block-level Accumulation in Convolution

Hypothetical target hardware

- Accumulating intermediate results of Conv in 8(oc)x8(ic) channels
- For example,

$$\mathrm{Conv}\left(\underset{24(ic)x32(ih)x32(iw)}{\text{Input}},\ \underset{8(oc)x24(ic)x1(kh)x1(kw)}{\text{Weight}}\right)$$

$$= \mathrm{Conv}_{8(oc)x8(ic)}\left(\underset{8(ic)x32(ih)x32(iw)}{\text{Input[0:8,:,:]}},\ \underset{8(oc)x8(ic)x1(kh)x1(kw)}{\text{Weight[:,0:8,:,:]}}\right)$$

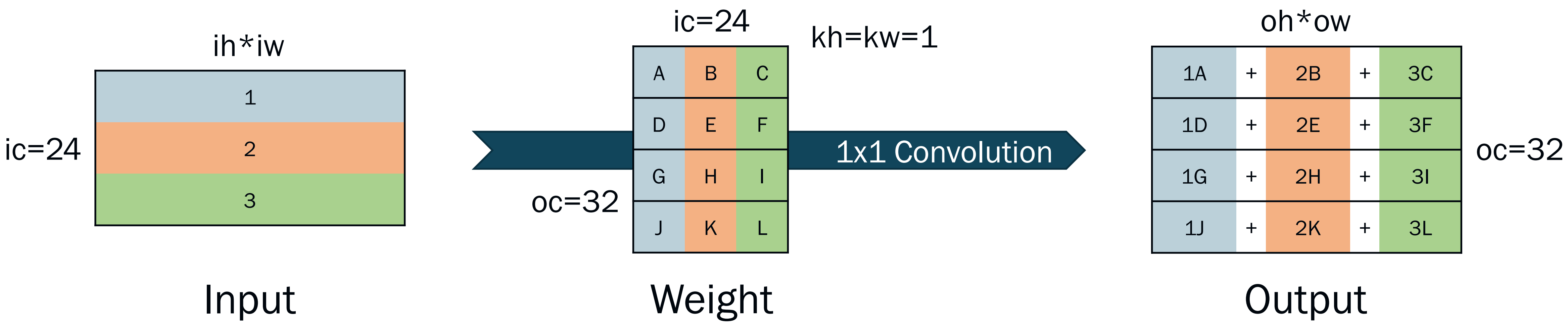
$$+ \mathrm{Conv}_{8(oc)x8(ic)}\left(\underset{8(ic)x32(ih)x32(iw)}{\text{Input[8:16,:,:]}},\ \underset{8(oc)x8(ic)x1(kh)x1(kw)}{\text{Weight[:,8:16,:,:]}}\right)$$

$$+ \mathrm{Conv}_{8(oc)x8(ic)}\left(\underset{8(ic)x32(ih)x32(iw)}{\text{Input[16:24,:,:]}},\ \underset{8(oc)x8(ic)x1(kh)x1(kw)}{\text{Weight[:,16:24,:,:]}}\right)$$

Case Study – Block-level Accumulation in Convolution

Hypothetical target hardware

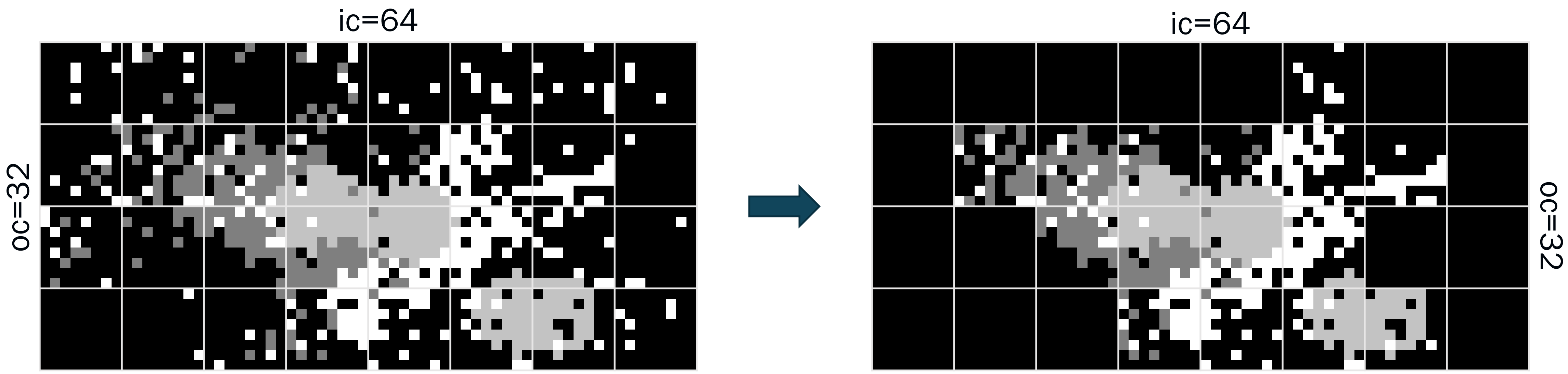
- Accumulating intermediate results of Conv in 8(oc)x8(ic) channels
- For example,



Case Study – Block-level Accumulation in Convolution

→ 8(oc)x8(ic) block-level sparsification

$$\text{Weight}_{sparse}(\text{i-th block}) = \begin{cases} 0 & \text{if } \Sigma|\text{Weight}_{dense}(\text{i-th block})| < \theta \\ \text{Weight}_{dense}(\text{i-th block}) & \text{otherwise} \end{cases}$$



1x1 Conv Weight (dense)

1x1 Conv Weight (53% sparse)

Conclusions

StradVision

Conclusions

- Layer transformation techniques can reduce production-cost especially for retraining of networks while supporting a diverse array of target platforms
- It may also be beneficial for deep learning hardware manufacturers to support a wide variety of networks not yet natively supported by their hardware

Resources

- Sparsification
  - “Exploring the Granularity of Sparsity in Convolution Neural Networks”, CVPR2017 https://openaccess.thecvf.com/content_cvpr_2017_workshops/w29/html/Mao_Exploring_the_Granularity_CVPR_2017_paper.html