CS5120 VLSI System Design, Spring 2025

Deep Neural Networks (DNNs) Part 2

黃稚存
Chih-Tsun Huang
cthuang@cs.nthu.edu.tw

國立清華大學
NATIONAL TSING HUA UNIVERSITY

資訊工程學系
Computer Science

Lecture 03

聲明

- 本課程之內容 (包括但不限於教材、影片、圖片、檔案資料等)，僅供修課學生個人合理使用，非經授課教師同意，不得以任何形式轉載、重製、散布、公開播送、出版或發行本影片內容(例如將課程內容放置公開平台上，如 Facebook, Instagram, YouTube, Twitter, Google Drive, Dropbox 等等)。如有侵權行為，需自負法律責任。

Outline

- Convolutional Neural Networks
- Convolutional Layer
- Pointwise Convolutional Layer
- Depth-wise Convolutional Layer
- Pooling Layer
- Batch Normalization Layer
- Fully Connected Layer
- Compute of Convolution
- Classic CNNs and Datasets

Convolutional Neural Networks

Image Classification as Target

(assume given set of discrete labels)
{dog, cat, truck, plane, ...}

→ cat

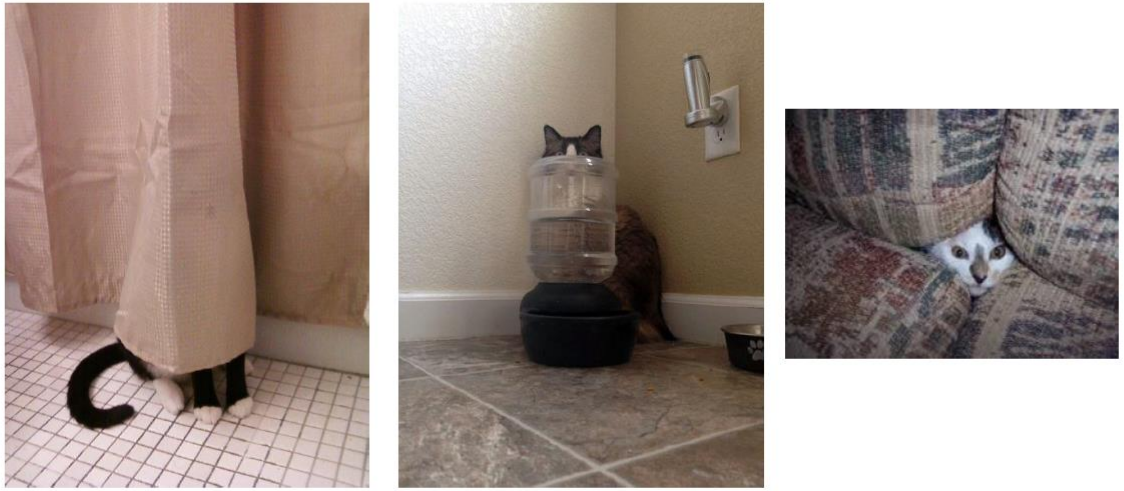
[Source: Stanford CS231n]

Challenges

Deformation

Background Clutter

Occlusion

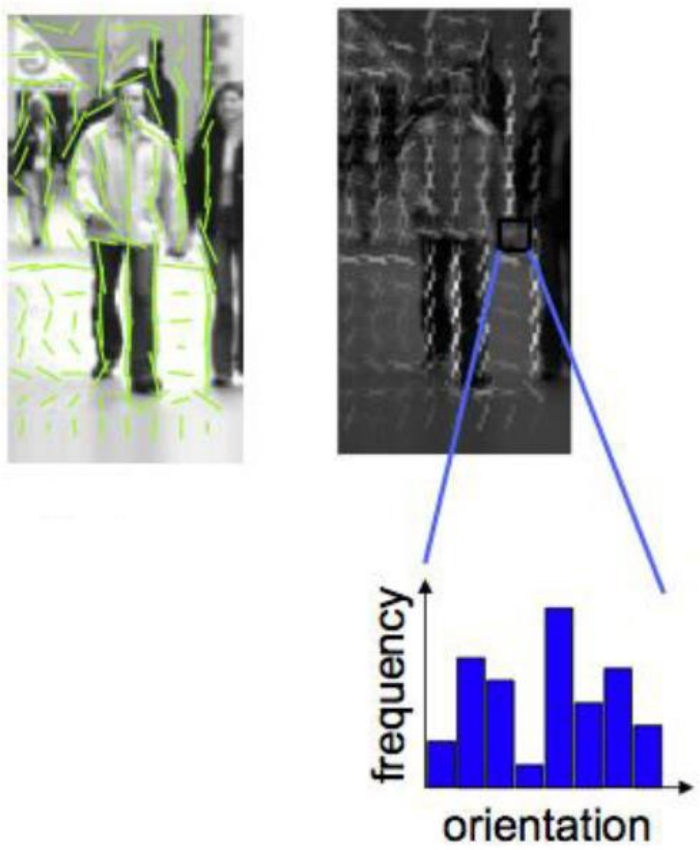


Intraclass Variation

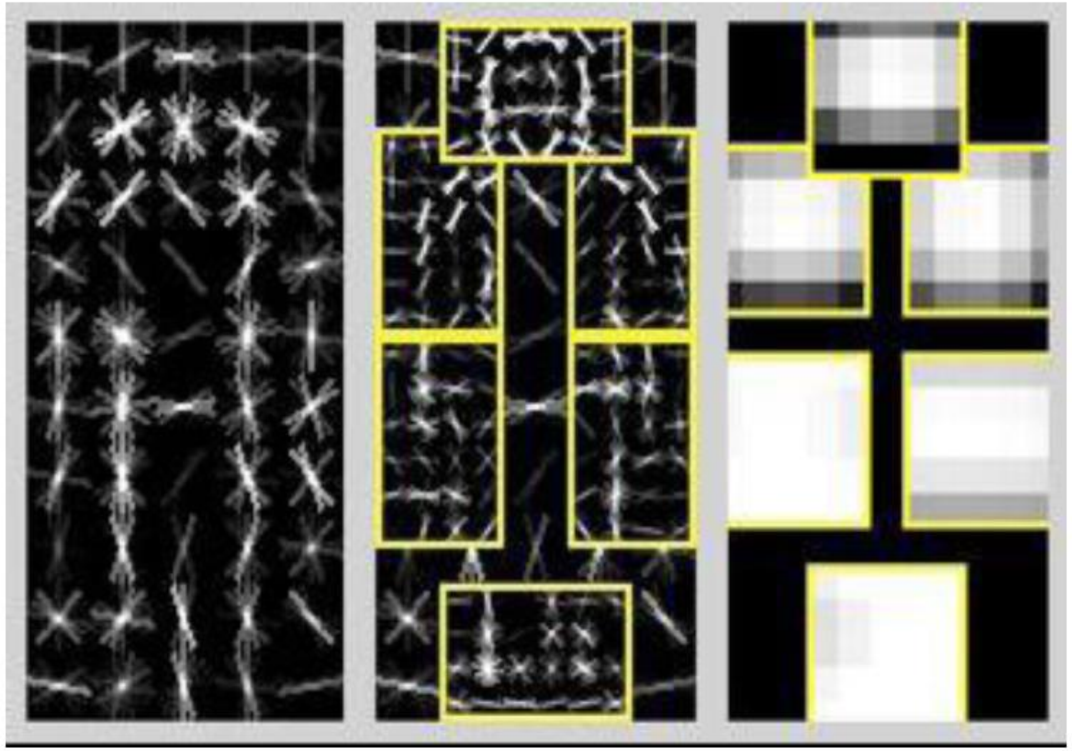
[Source: Stanford CS231n]

Traditional Rule-Based Approaches

Histogram of Gradients (HoG)
Dalal & Triggs, 2005

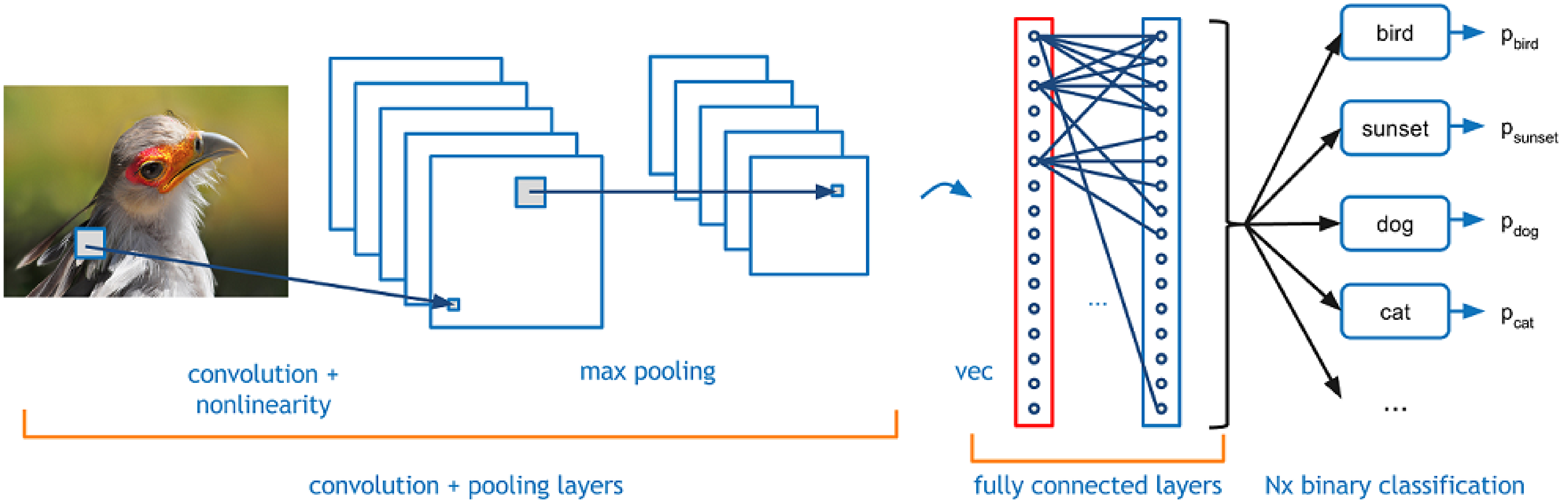


Deformable Part Model
Felzenswalb, McAllester, Ramanan, 2009



[Source: Stanford CS231n]

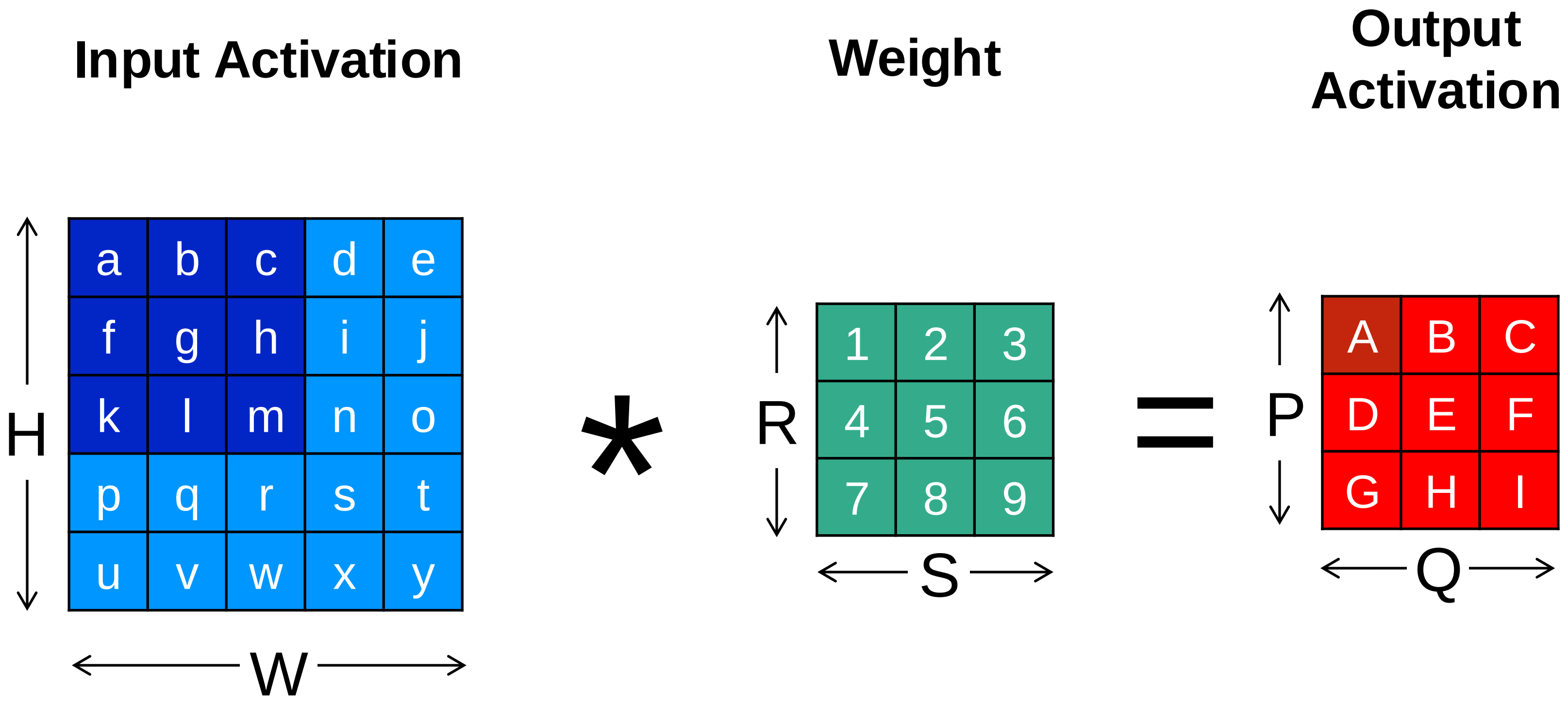
Convolutional Neural Networks



https://github.com/vdumoulin/conv_arithmetic

Convolution Layer

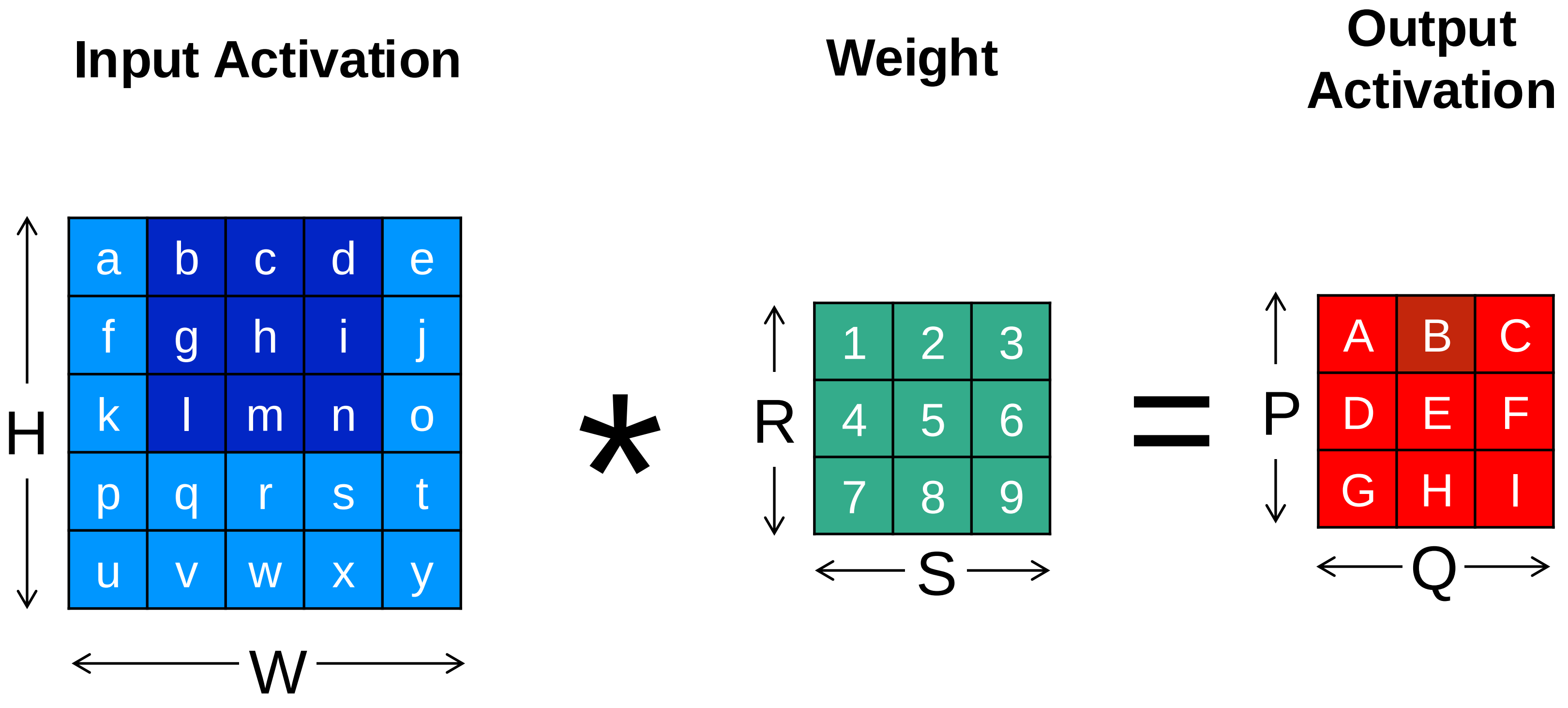
2-D Convolution



H: Height of Input Activation
W: Width of Input Activation
R: Height of Weight
S: Width of Weight
P: Height of Output Activation
Q: Width of Output Activation

$$A = a \times 1 + b \times 2 + c \times 3$$
$$+ f \times 4 + g \times 5 + h \times 6$$
$$+ k \times 7 + l \times 8 + m \times 9$$

2-D Convolution: Stride U=1

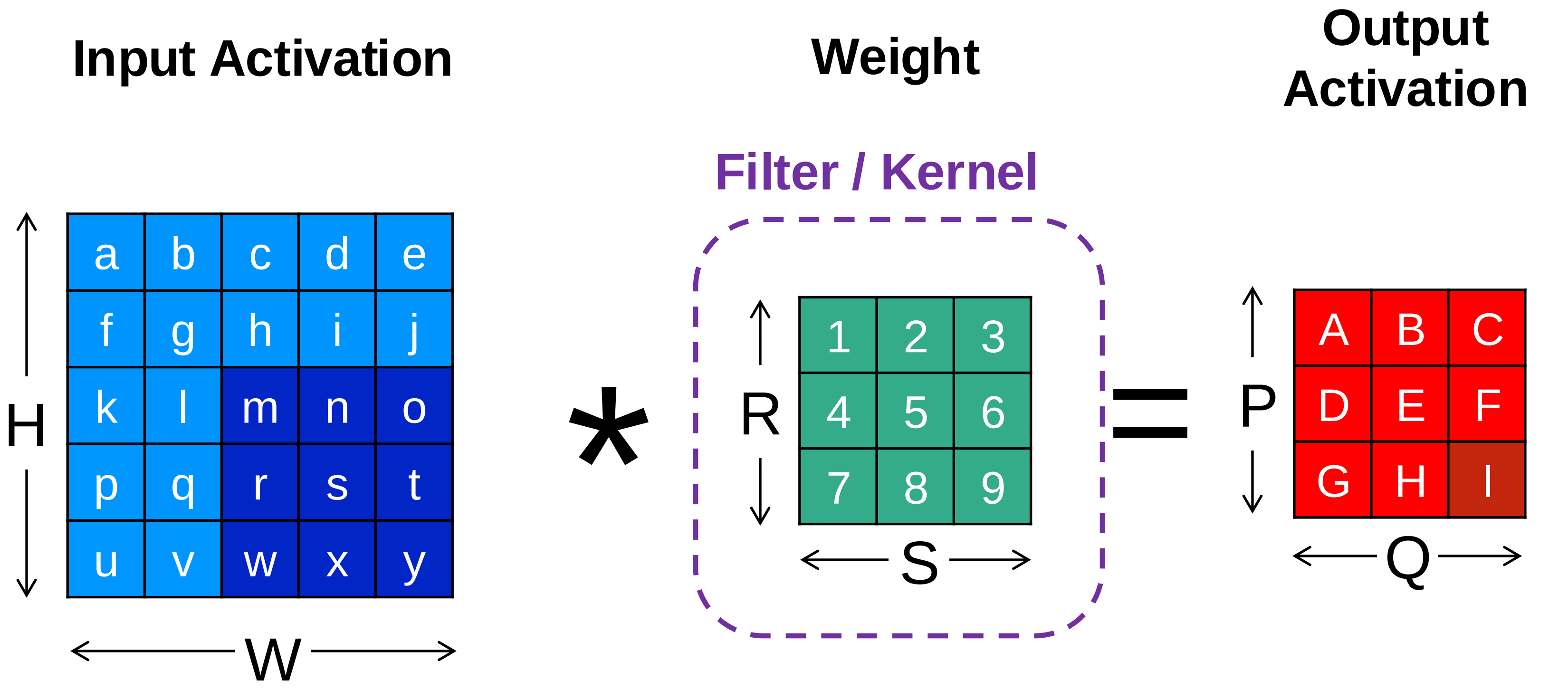


$$B = b \times 1 + c \times 2 + d \times 3$$
$$+ g \times 4 + h \times 5 + i \times 6$$
$$+ l \times 7 + m \times 8 + n \times 9$$

H: Height of Input Activation
W: Width of Input Activation
R: Height of Weight
S: Width of Weight
P: Height of Output Activation
Q: Width of Output Activation

U (Stride): # of rows/columns traversed per step

2-D Convolution: Stride U=1

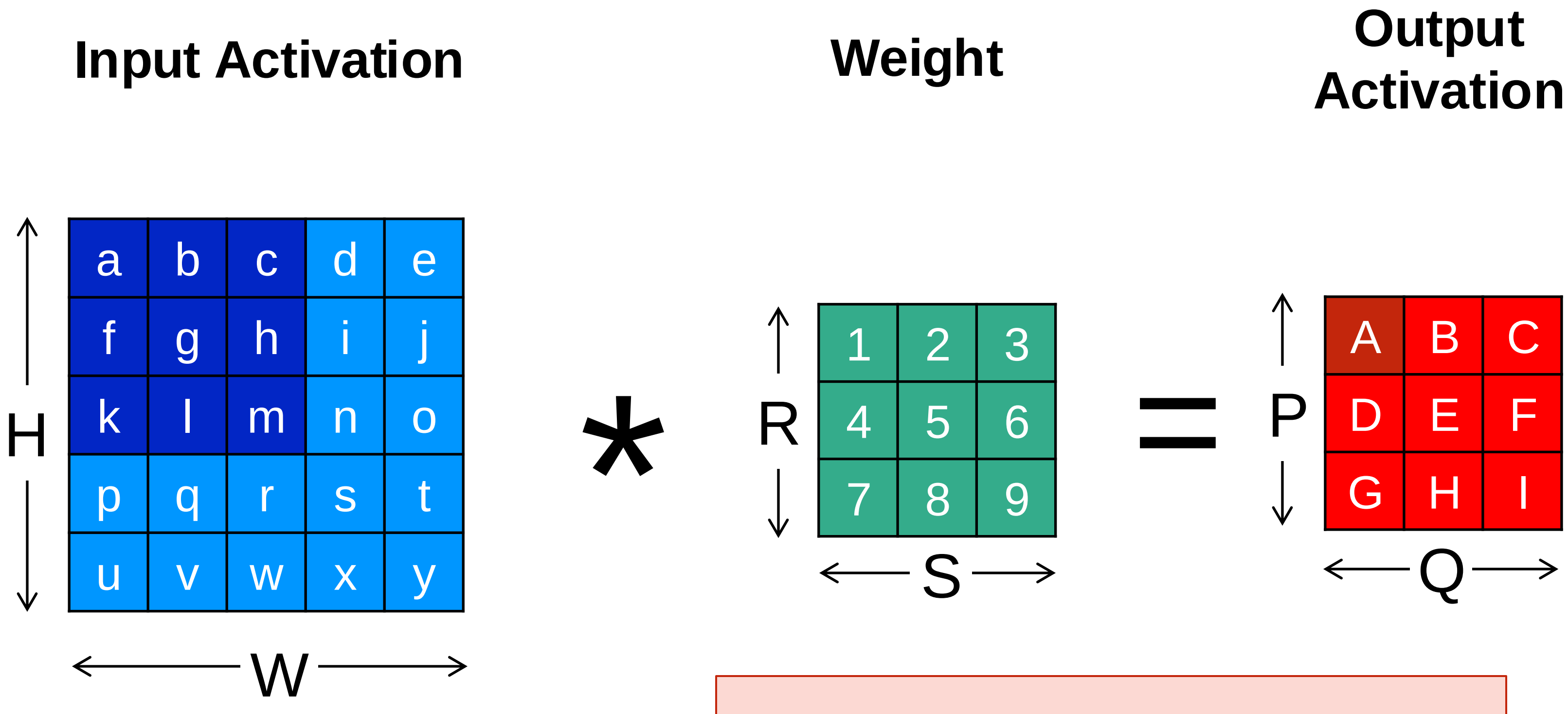


$$I = m \times 1 + n \times 2 + o \times 3$$
$$+ r \times 4 + s \times 5 + t \times 6$$
$$+ w \times 7 + x \times 8 + y \times 9$$

H: Height of Input Activation
W: Width of Input Activation
R: Height of Weight
S: Width of Weight
P: Height of Output Activation
Q: Width of Output Activation

U (Stride): # of rows/columns traversed per step

2-D Convolution: Stride U=1, Valid Convolution



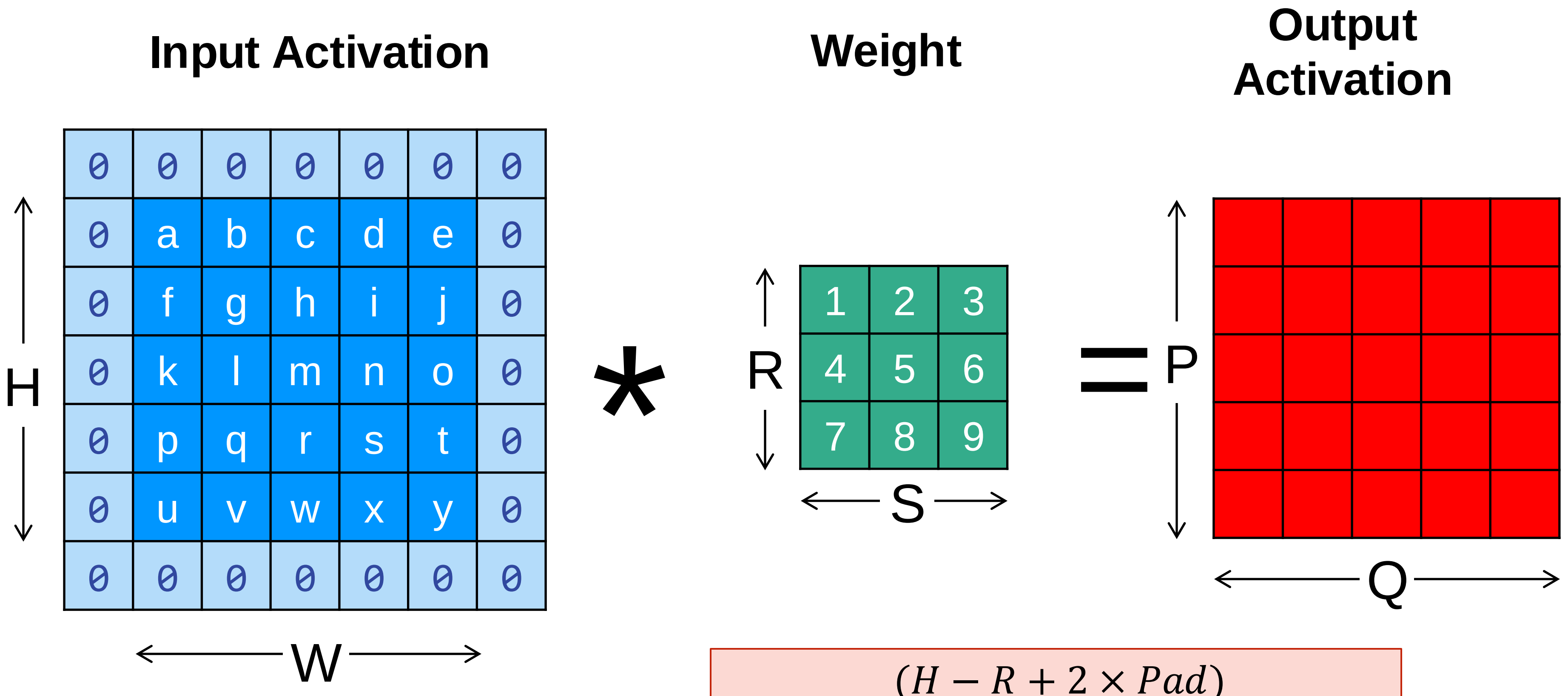
$$P = \frac{(H - R)}{U} + 1$$

$$Q = \frac{(W - S)}{U} + 1$$

H: Height of Input Activation
W: Width of Input Activation
R: Height of Weight
S: Width of Weight
P: Height of Output Activation
Q: Width of Output Activation

U (Stride): # of rows/columns traversed per step

2-D Convolution: Stride U=1, Padding = 1



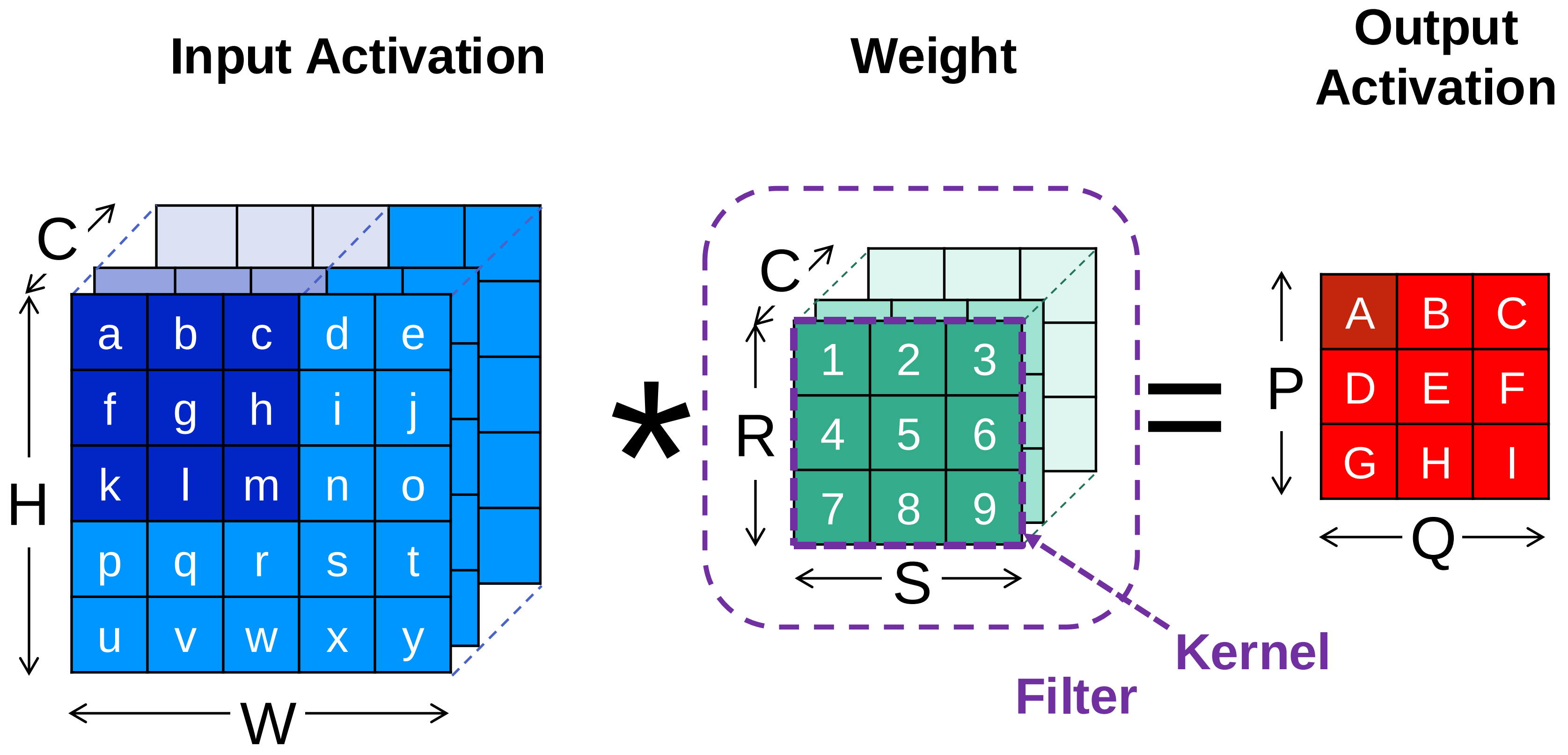
$$P = \frac{(H - R + 2 \times Pad)}{U} + 1$$

$$Q = \frac{(W - S + 2 \times Pad)}{U} + 1$$

H: Height of Input Activation
W: Width of Input Activation
R: Height of Weight
S: Width of Weight
P: Height of Output Activation
Q: Width of Output Activation

U (Stride): # of rows/columns traversed per step
Pad (Padding): # of zero rows/columns added

3-D Convolution: C Input Channels

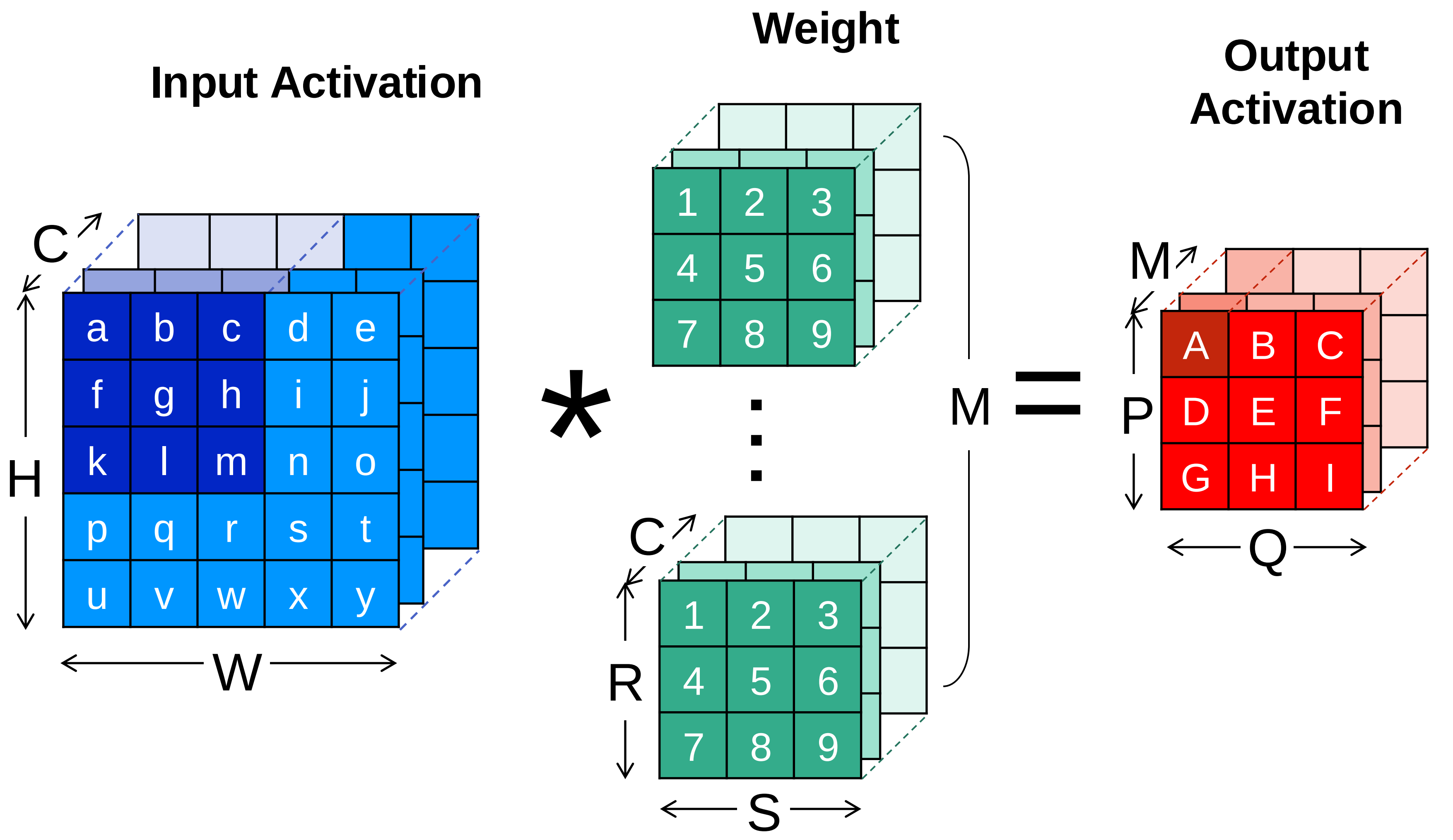


H: Height of Input Activation
W: Width of Input Activation
R: Height of Weight
S: Width of Weight
P: Height of Output Activation
Q: Width of Output Activation

U (Stride): # of rows/columns traversed per step
Pad (Padding): # of zero rows/columns added

C: # of Input Channels

3-D Convolution: K Output Channels

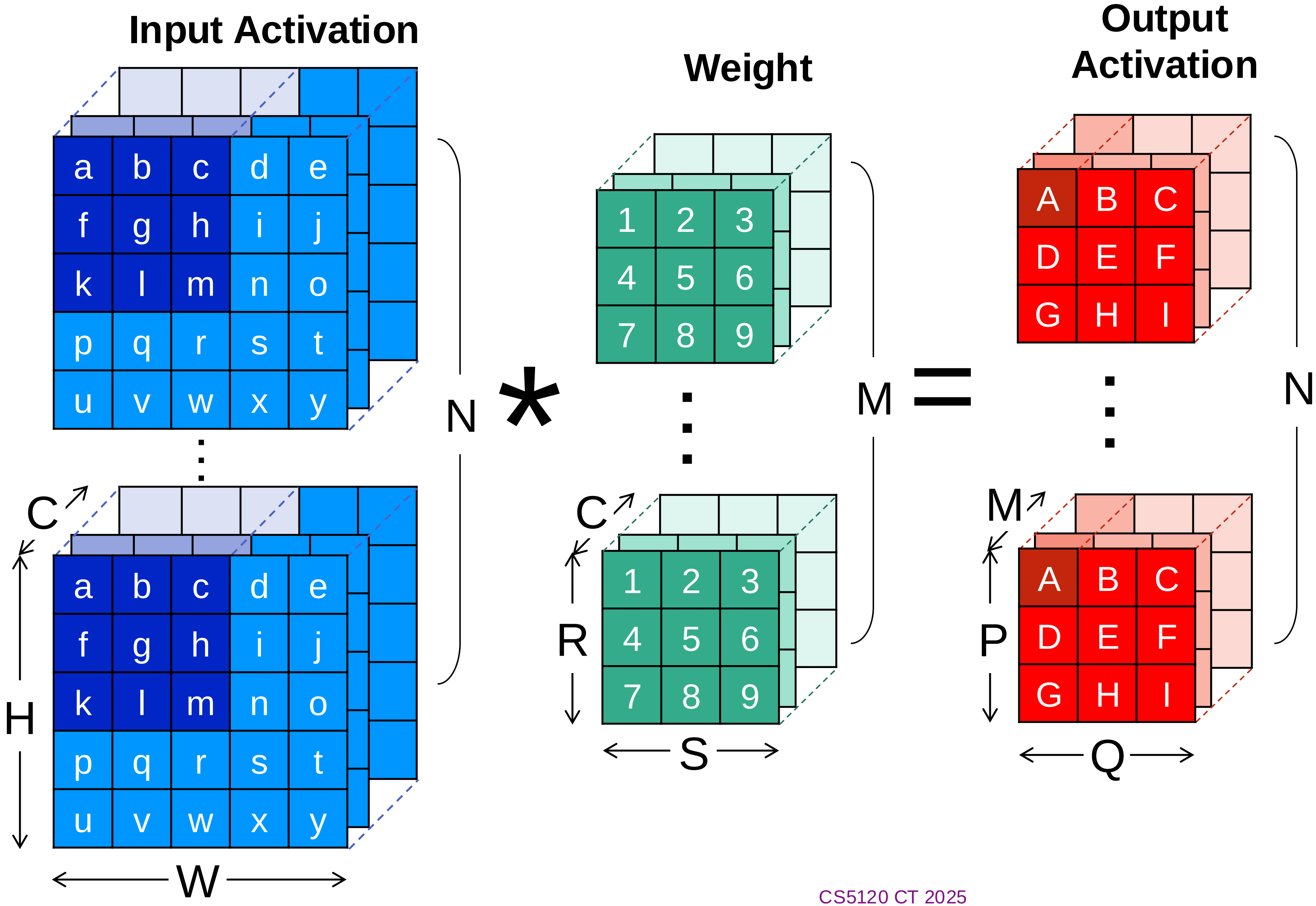


H: Height of Input Activation
W: Width of Input Activation
R: Height of Weight
S: Width of Weight
P: Height of Output Activation
Q: Width of Output Activation

U (Stride): # of rows/columns traversed per step
Pad (Padding): # of zero rows/columns added

C: # of Input Channels
M: # of Output Channels

3-D Convolution: N Batches



H: Height of Input Activation
W: Width of Input Activation
R: Height of Weight
S: Width of Weight
P: Height of Output Activation
Q: Width of Output Activation

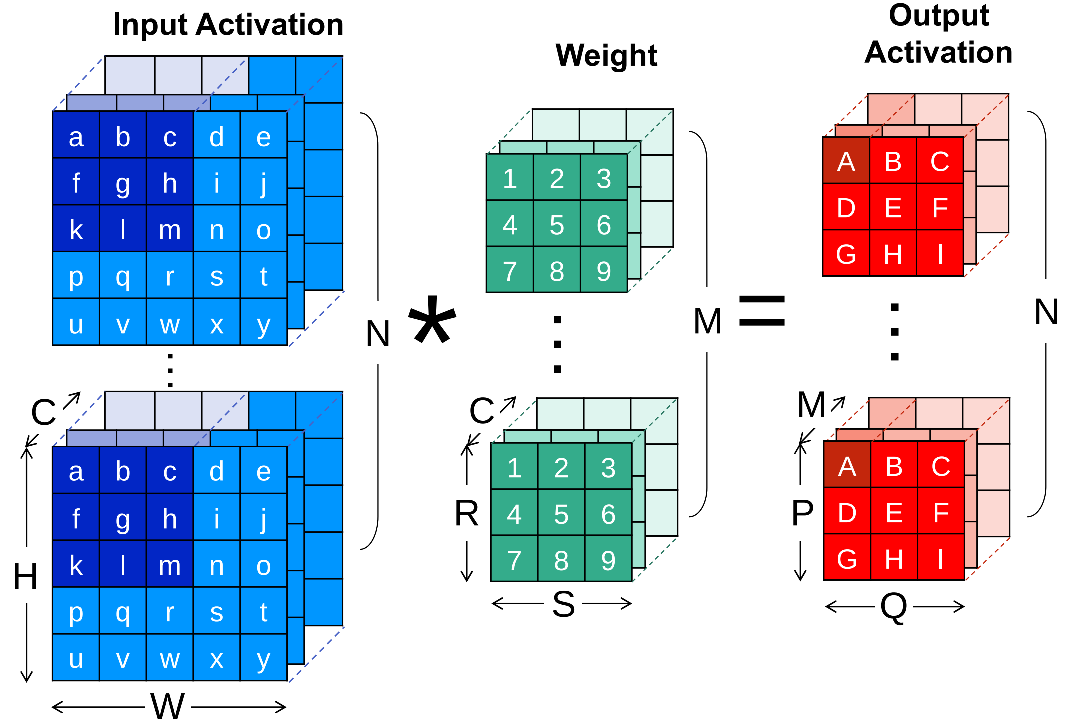
U (Stride): # of rows/columns traversed per step
Pad (Padding): # of zero rows/columns added

C: # of Input Channels
M: # of Output Channels
N: Batch size

Convolution Layer: 7 Nested Loops

Input Feature Map (ifmap)

Output Feature Map (ofmap)




```
for (n=0; n<N; n++) {
  for (m=0; m<M; m++) {
    for (p=0; p<P; p++) {
      for (q=0; q<Q; q++) {
        OA[n][m][p][q] = 0;
        for (r=0; r<R; r++) {
          for (s=0; s<S; s++) {
            for (c=0; c<C; c++) {
              h = p * U - Pad + r;
              w = q * U - Pad + s;
              OA[n][m][p][q] +=
                IA[n][c][h][w] * W[m][c][r][s];
        }}}
        OA[n][m][p][q] = Activation(OA[n][m][p][q]);
}}}}
```

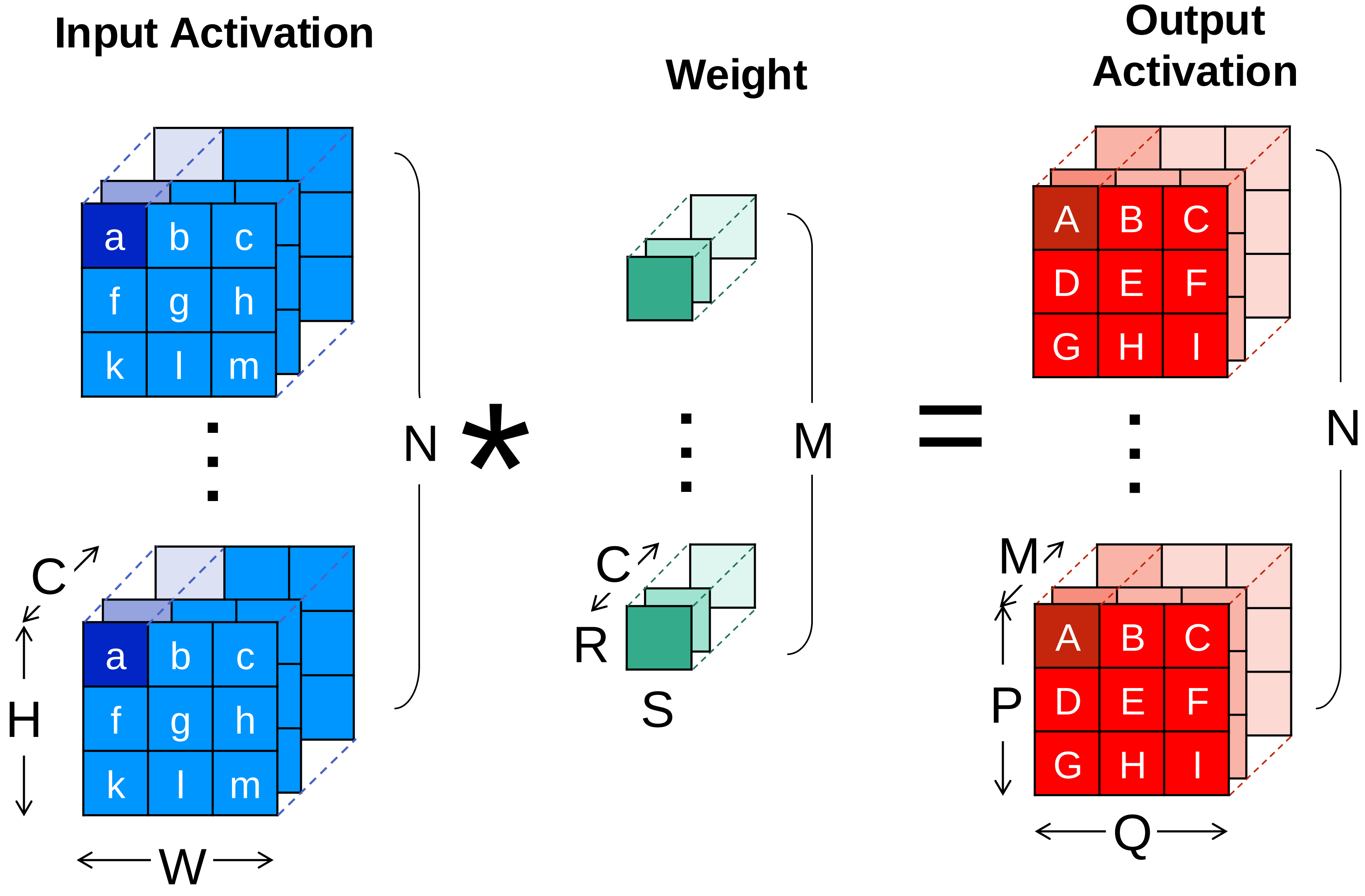
For each output activation

Convolution window

Partial Sum (psum)

Pointwise Convolution Layer

Pointwise Convolution = 1x1 Convolution



H: Height of Input Activation
W: Width of Input Activation
R = 1
S = 1
P = H
Q = W

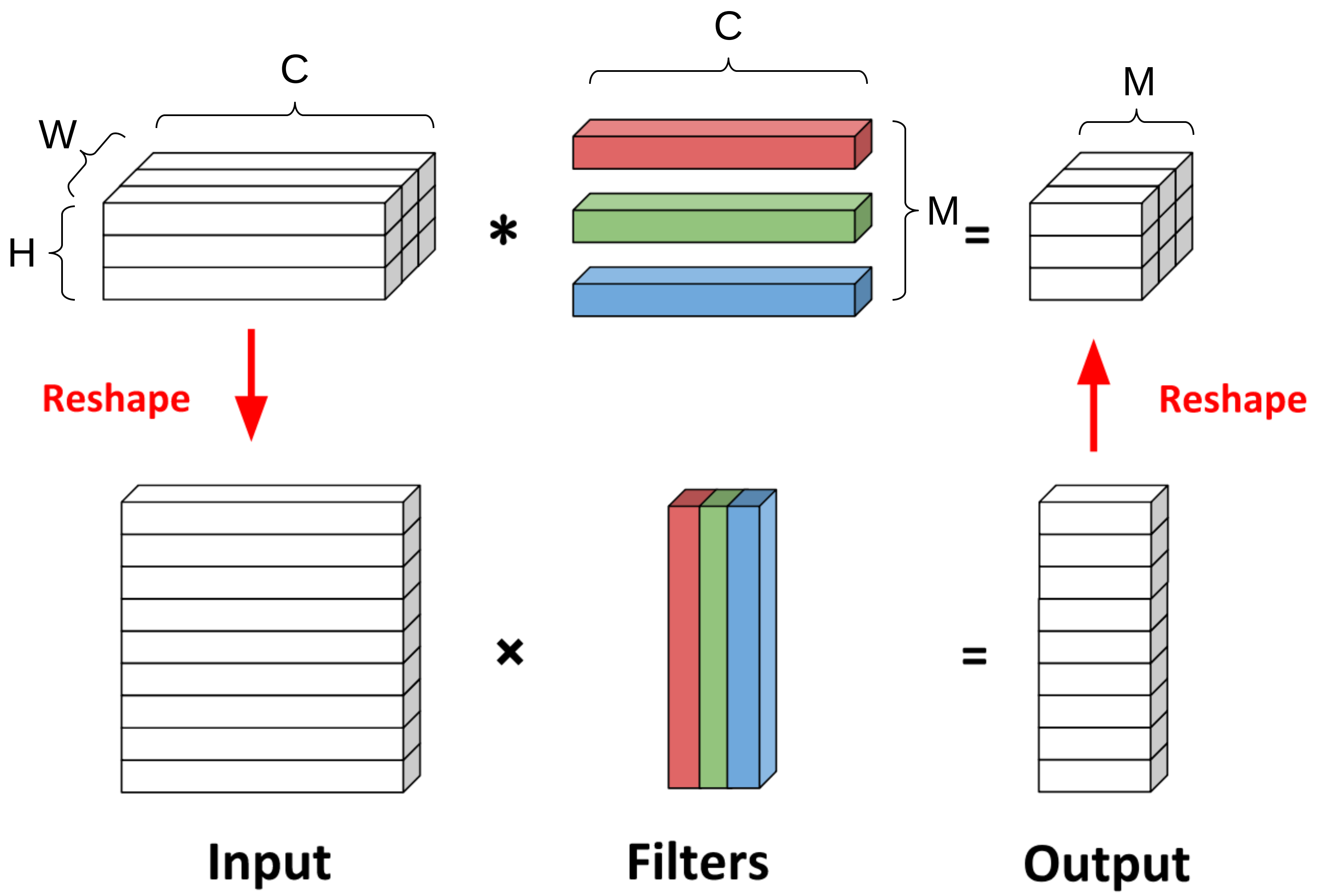
U (Stride): # of rows/columns traversed per step
Pad (Padding) = 0

C: # of Input Channels
M: # of Output Channels
N: Batch size

Pointwise Convolution as Matrix Multiplication

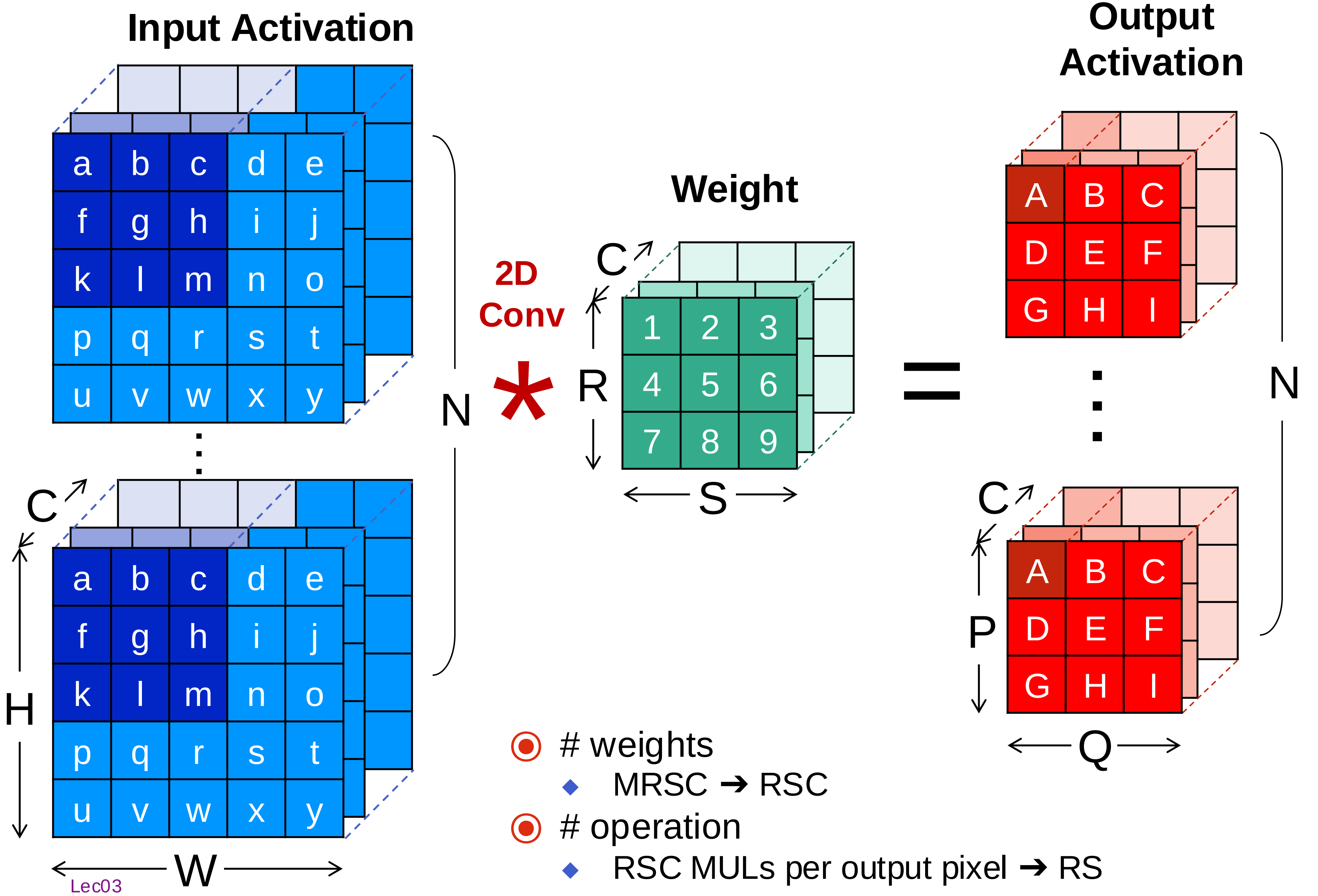
Use 1×1 filters

= Matrix Multiplication



Depth-wise Convolution Layer

Depth-wise Convolution



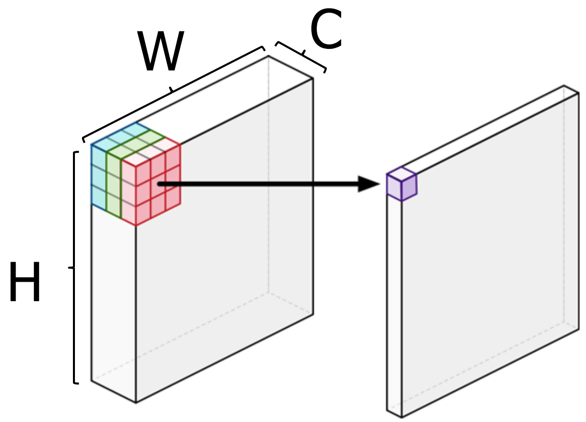
H: Height of Input Activation
W: Width of Input Activation
R: Height of Weight
S: Width of Weight
P: Height of Output Activation
Q: Width of Output Activation

U (Stride): # of rows/columns traversed per step
Pad (Padding): # of zero rows/columns added

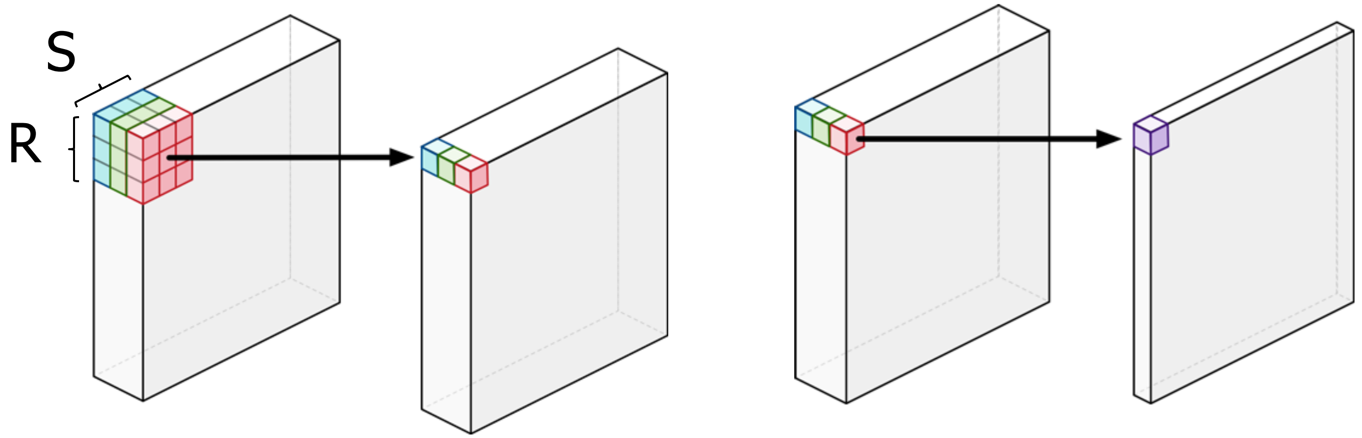
C: # of Input Channels
M = C
N: Batch size

- # weights
  - MRSC ➔ RSC
- # operation
  - RSC MULs per output pixel ➔ RS

Depth-wise Separable Convolution: Depth-wise + Pointwise Convolution



Standard convolution



Depth-wise convolution

Pointwise convolution

$$H \times W \times C \times M \times R \times S$$

$$H \times W \times C \times R \times S \; + \; H \times W \times C \times M$$

- Reduction in computation

$$\frac{H \times W \times C \times R \times S + H \times W \times C \times M}{H \times W \times C \times M \times R \times S}$$

- Parameters↓, Computational Cost↓

Pooling Layer

Pooling (Subsampling)

- Nonlinear down-sampling
  - Reducing the spatial size (# parameters; amount of computation) of the representation
  - Also control overfitting
- Parameters:
  - Type: MAX or AVG
  - Pooling kernel size
  - Stride

Ex: pooling with 2x2 filter and stride 2

2	0	9	1
4	6	3	7
3	1	3	0
2	2	5	8

Max Pooling

6	9
3	8

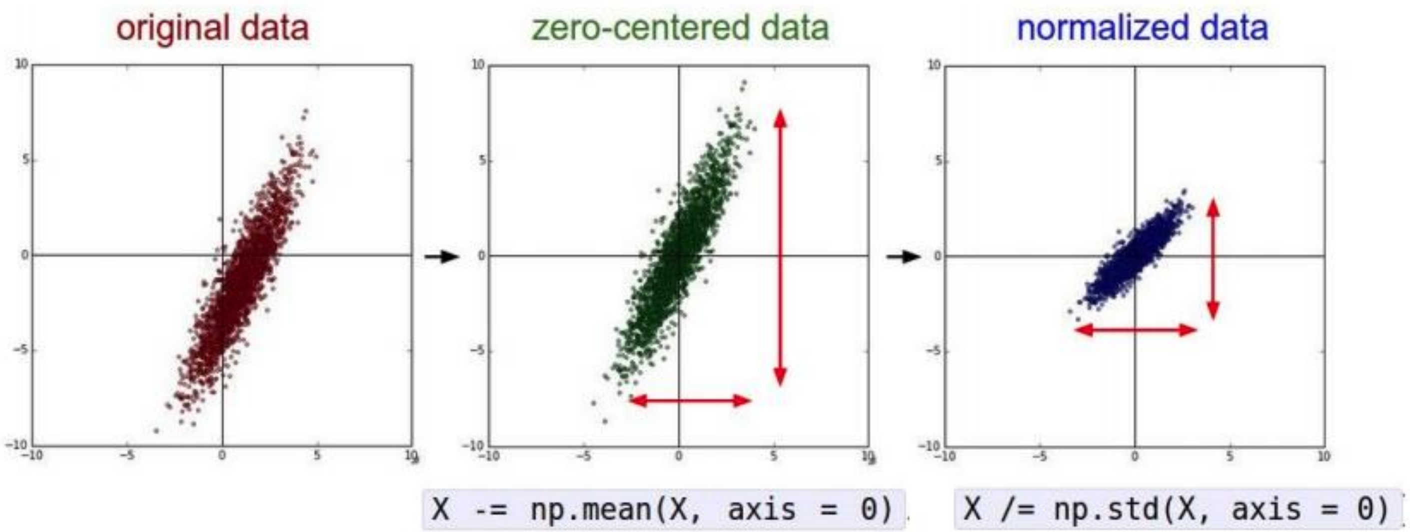
Average Pooling

3	5
2	4

Batch Normalization (BatchNorm) Layer

BatchNorm Layer

- To ease the training with zero-mean, unit-variance activations
  - The training is complicated by the fact that the inputs to each layer are affected by the parameters of all preceding layers – so that small changes to the network parameters amplify as the network becomes deeper



Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

BatchNorm Layer

Input: $x : N \times D$

Learnable scale and shift parameters:

$\gamma, \beta : D$

Learning $\gamma = \sigma$, $\beta = \mu$ will recover the identity function!

Training Phase

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j}$$ Per-channel mean, shape is D

$$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N}(x_{i,j} - \mu_j)^2$$ Per-channel var, shape is D

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$ Normalized x, Shape is N x D

$$y_{i,j} = \gamma_j \hat{x}_{i,j} + \beta_j$$ Output, Shape is N x D

Test Phase

$\mu_j =$ (Running) average of values seen during training — Per-channel mean, shape is D

$\sigma_j^2 =$ (Running) average of values seen during training — Per-channel var, shape is D

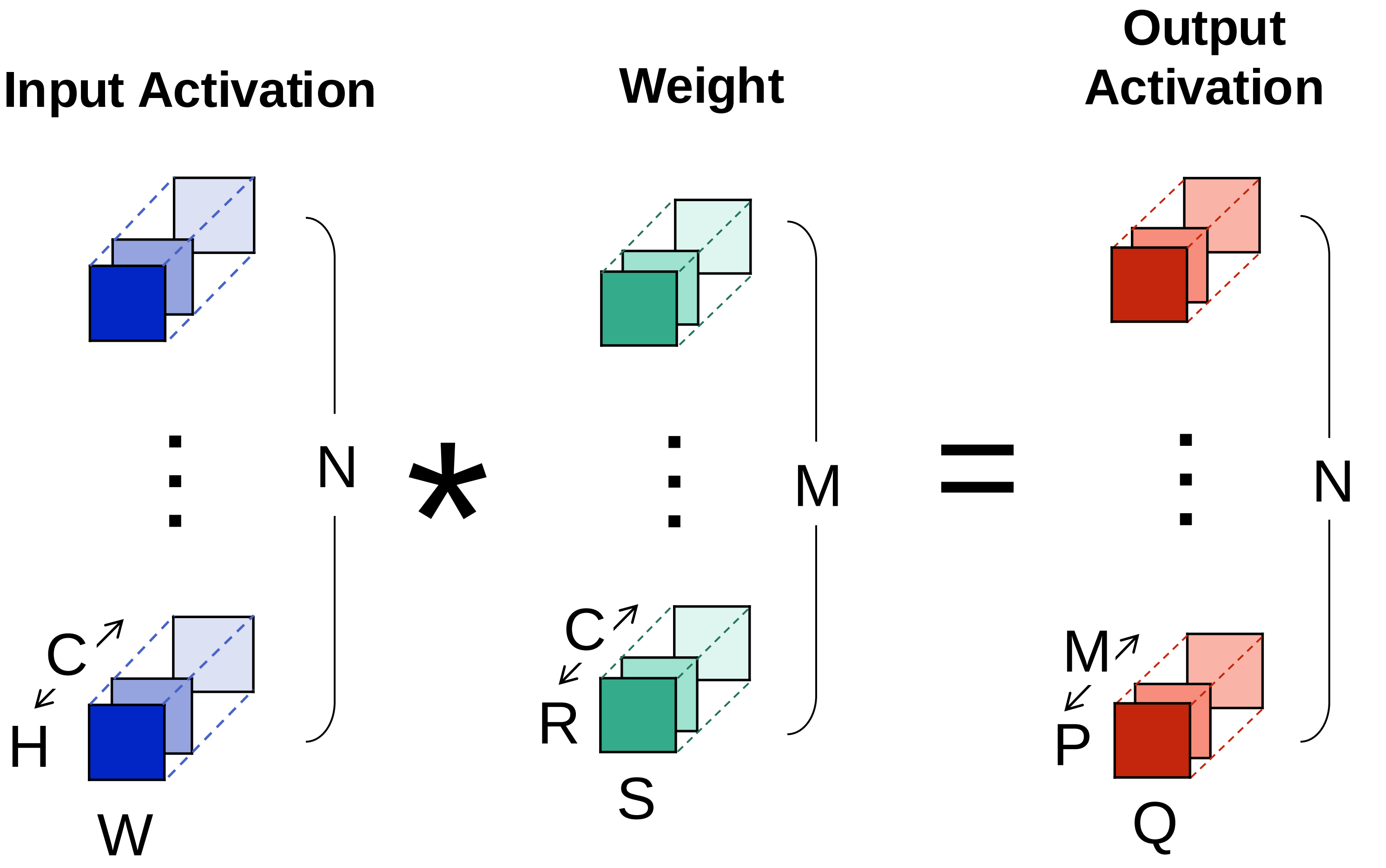
$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$ Normalized x, Shape is N x D

$$y_{i,j} = \gamma_j \hat{x}_{i,j} + \beta_j$$ Output, Shape is N x D

[Source: Prof. Sophia Shao, EE290-2, Berkeley]

Fully-Connected Layer

Fully-Connected Layer



H = 1
W = 1
R = 1
S = 1
P = 1
Q = 1

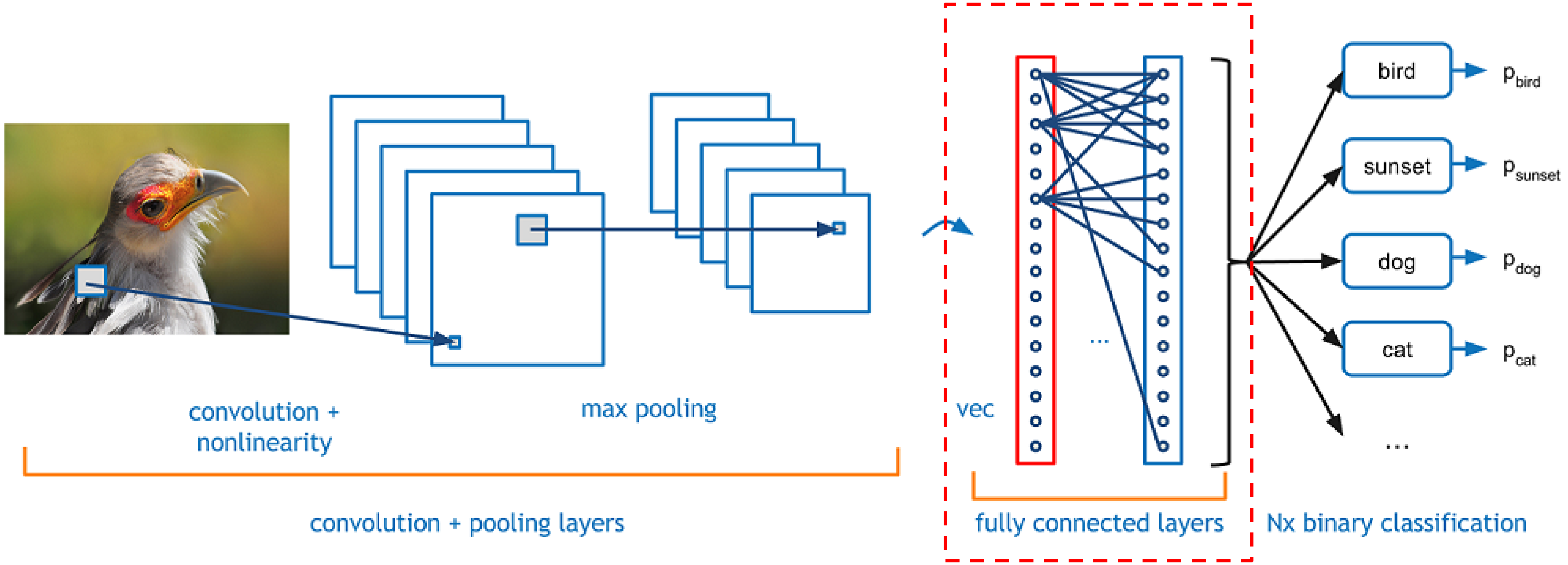
U (Stride)= 1

Pad (Padding) = 0

C: # of Input Channels
M: # of Output Channels
N: Batch size

Fully Connected (FC) Layer

- The output from the convolutional and pooling layers represents high-level features
- FC layer uses these features for classifying the input image into various classes



Compute of Convolution

Computation of a CONV Layer

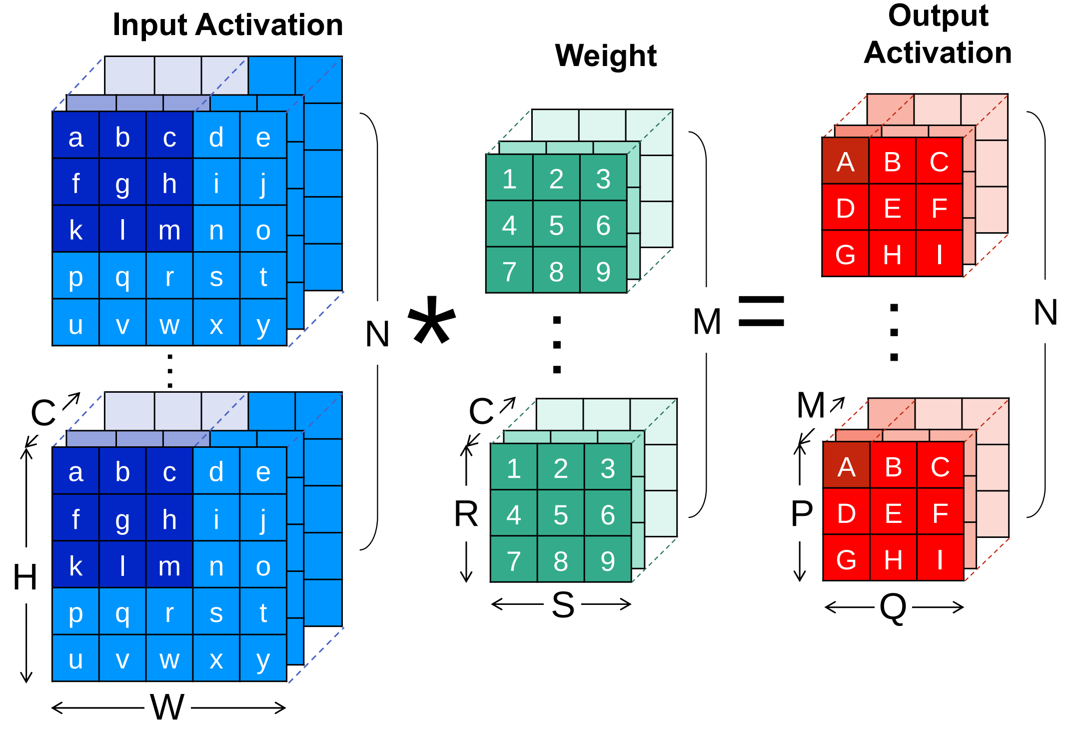
$$\mathbf{o}[n][m][p][q] = \left(\sum_{c=0}^{C-1}\sum_{r=0}^{R-1}\sum_{s=0}^{S-1} \mathbf{i}[n][c][Up+r][Uq+s] \times \mathbf{f}[m][c][r][s]\right) + \mathbf{b}[m],$$

$$0 \le n < N, 0 \le m < M, 0 \le p < P, 0 \le q < Q,$$

$$P = (H - R + U)/U, Q = (W - S + U)/U.$$

o: tensor of the ofmaps
i: tensor of the ifmaps
f: tensor of filters
b: tensor of biases

Method 1: Naïve 7-level for-loop Implementation



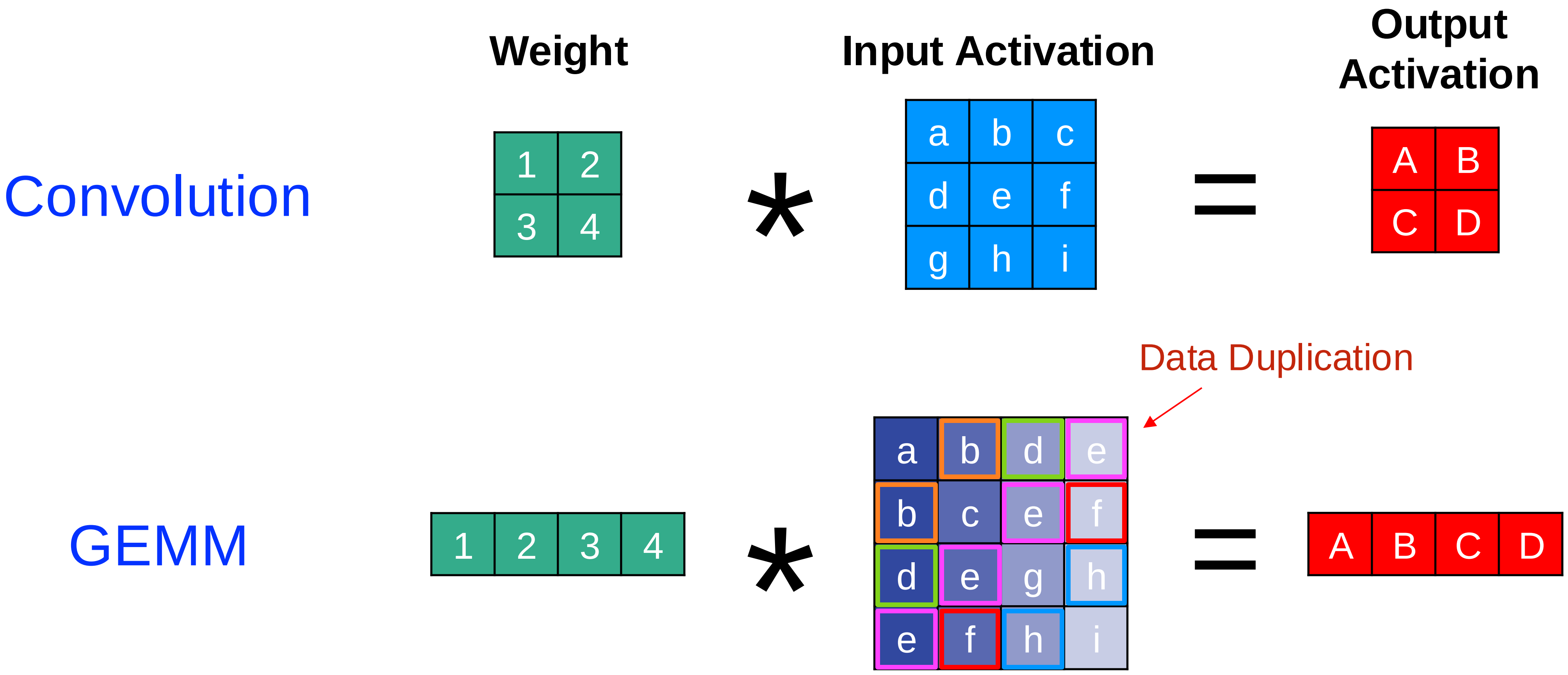

```
for (n=0; n<N; n++) {
  for (m=0; m<M; m++) {
    for (p=0; p<P; p++) {
      for (q=0; q<Q; q++) {
        OA[n][m][p][q] = 0;
        for (r=0; r<R; r++) {
          for (s=0; s<S; s++) {
            for (c=0; c<C; c++) {
              h = p * U - Pad + r;
              w = q * U - Pad + s;
              OA[n][m][p][q] +=
                IA[n][c][h][w] * W[m][c][r][s];


        }}}
        OA[n][m][p][q] = Activation(OA[n][m][p][q]);
}}}}
```

For each output activation (n, m, p, q loops)

Convolution window (r, s, c loops)

Method 2: GEMM (GEneral Matrix Multiply)

- Converting convolution to GEMM via **im2col**



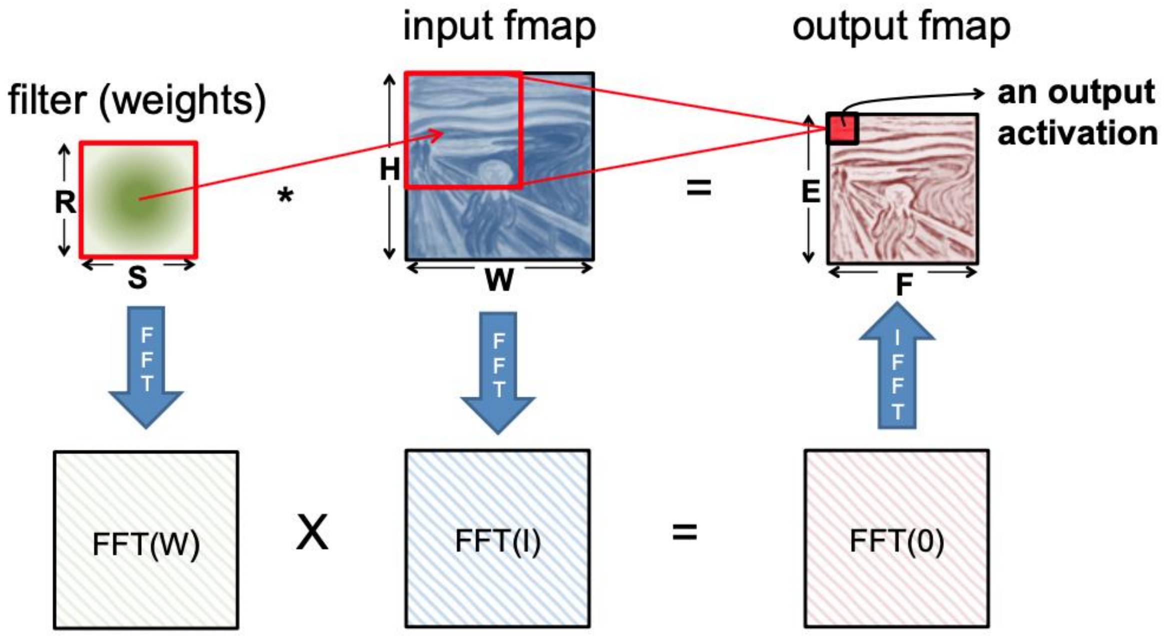
Method 3: FFT-based Convolution

- **Convolution theorem**: convolution in the time domain is equivalent to point-wise multiply in the frequency domain

$$f * g = \mathcal{F}^{-1}\{\mathcal{F}\{f\} \cdot \mathcal{F}\{g\}\}$$

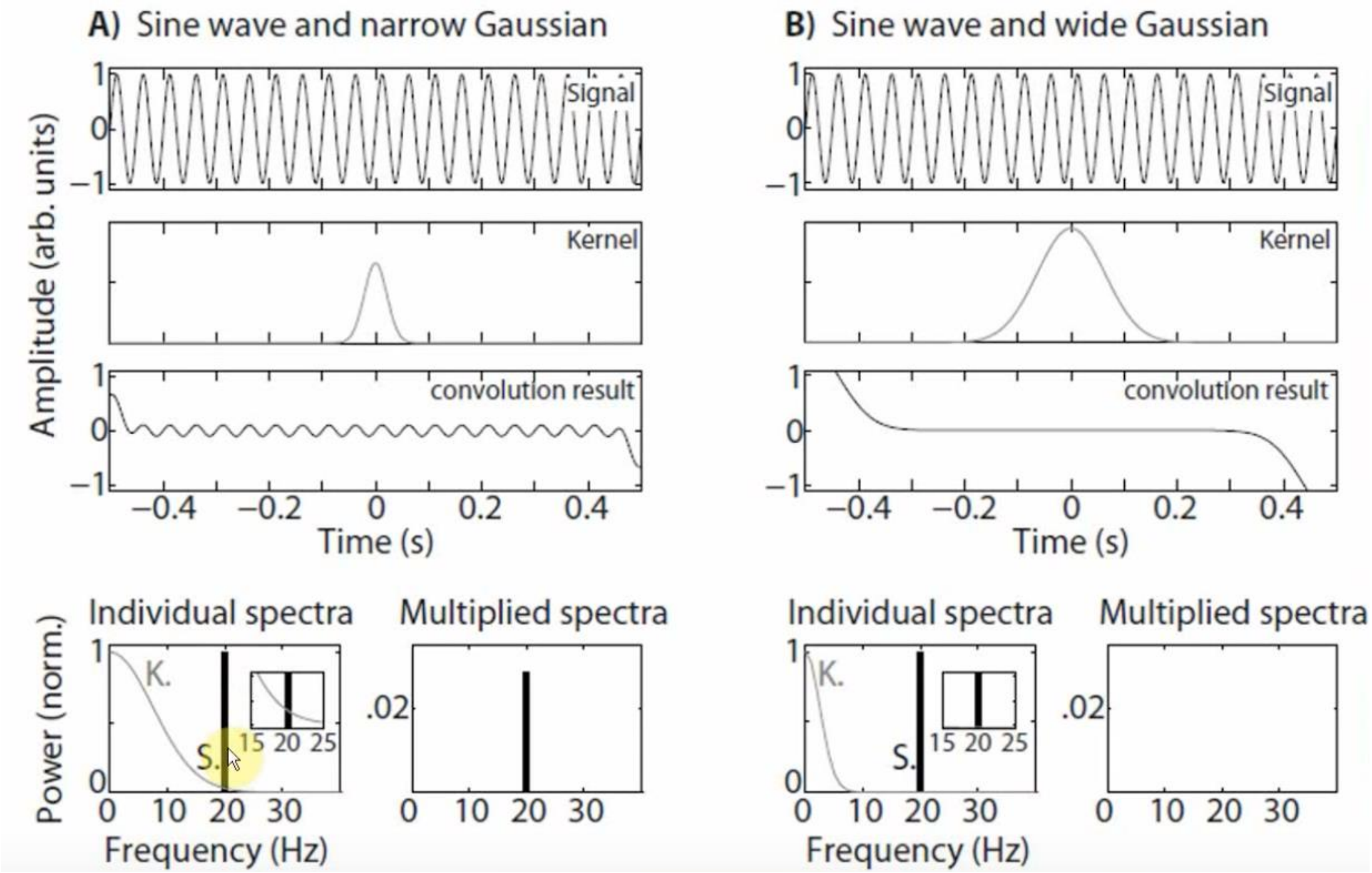
$\mathcal{F}\{f\}$ and $\mathcal{F}\{g\}$ are the Fourier transforms of f and g

The asterisk denotes convolution, not multiplication.



[Source: MIT Eyeriss Tutorial]

Method 3: FFT-based Convolution



[Source: Prof. Sophia Shao, EE290-2, Berkeley]

Method 3: FFT-based Convolution

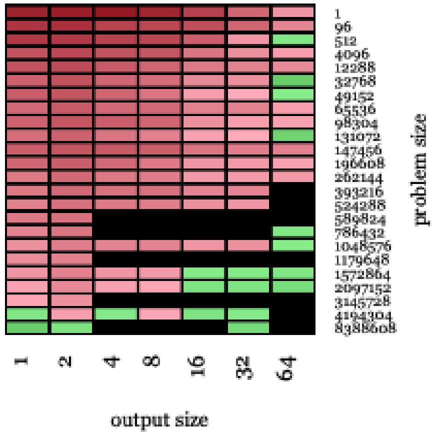


Figure 1: 3 × 3 kernel (K40m)

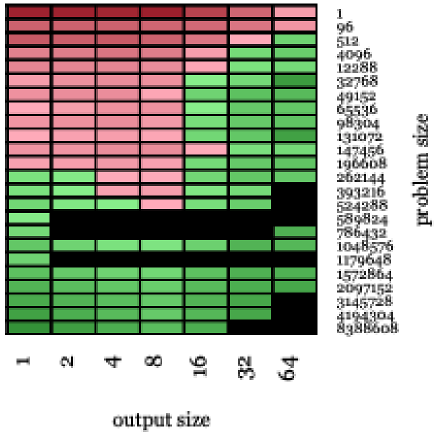


Figure 2: 5 × 5 kernel (K40m)

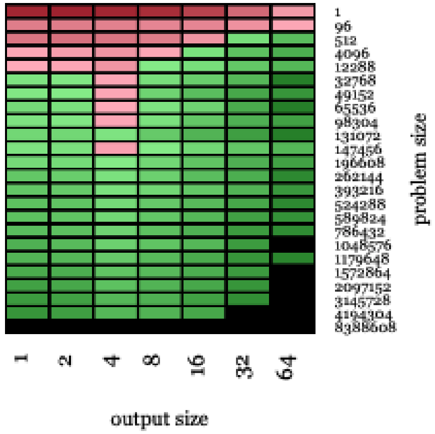


Figure 3: 7 × 7 kernel (K40m)

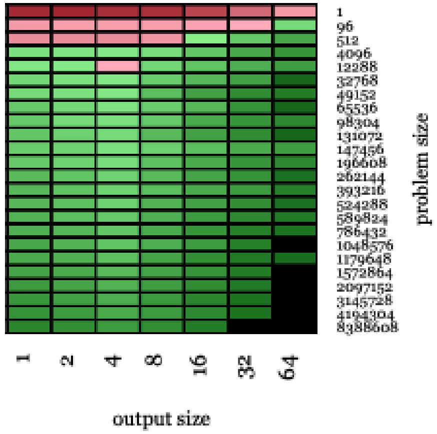


Figure 4: 9 × 9 kernel (K40m)

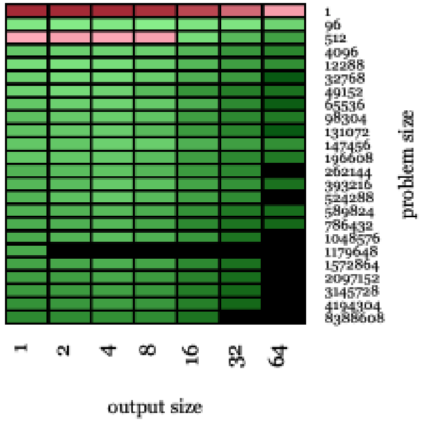


Figure 5: 11 × 11 kernel (K40m)

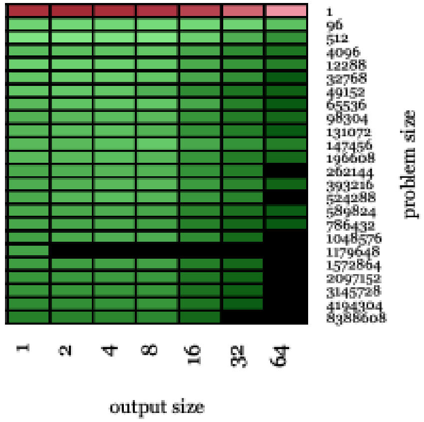
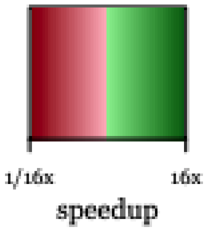


Figure 6: 13 × 13 kernel (K40m)



Fast Convolutional Nets with fbfft: A GPU Performance Evaluation

Method 4: Winograd Transform

- Re-association of intermediate values to reduce # of multiplications
- Works well for 3x3 convolution

$$F(2,3) = \begin{bmatrix} d_0 & d_1 & d_2 \\ d_1 & d_2 & d_3 \end{bmatrix} \begin{bmatrix} g_0 \\ g_1 \\ g_2 \end{bmatrix} = \begin{bmatrix} m_1 + m_2 + m_3 \\ m_2 - m_3 - m_4 \end{bmatrix} \tag{5}$$

where

$$m_1 = (d_0 - d_2)g_0 \qquad m_2 = (d_1 + d_2)\frac{g_0 + g_1 + g_2}{2}$$
$$m_4 = (d_1 - d_3)g_2 \qquad m_3 = (d_2 - d_1)\frac{g_0 - g_1 + g_2}{2}$$

$$Y = A^T\left[(Gg) \odot (B^T d)\right] \tag{6}$$

$$B^T = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}$$
$$G = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$
$$A^T = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & -1 & -1 \end{bmatrix}$$
$$g = \begin{bmatrix} g_0 & g_1 & g_2 \end{bmatrix}^T$$
$$d = \begin{bmatrix} d_0 & d_1 & d_2 & d_3 \end{bmatrix}^T$$

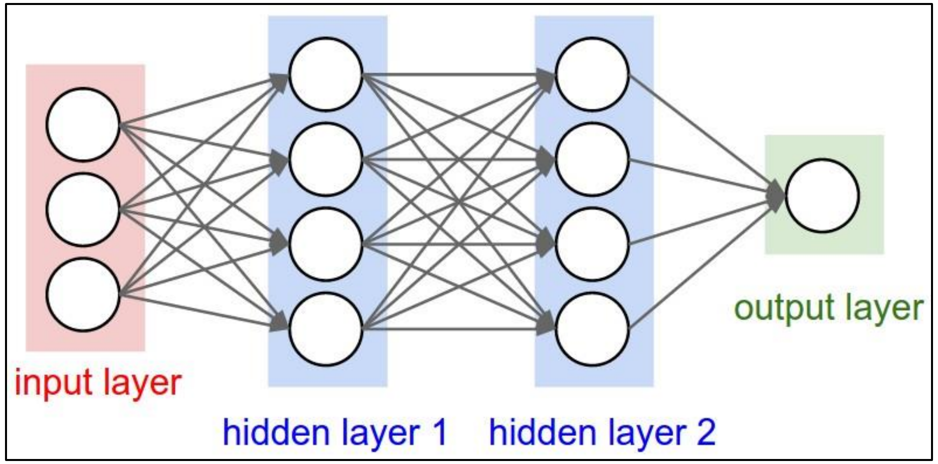
- Original convolution
  - 6 MULs, 4 ADDs
- Winograd
  - IA (d): 4 ADDs
  - W (g): 3 ADDs, 2 MULs
  - OA (m): 4 MULs, 4 ADDs

Classic CNNs and Datasets

Why Convolutional Neural Network Instead of Fully Connected Network?

- Image filtering
  - Spatial relationship

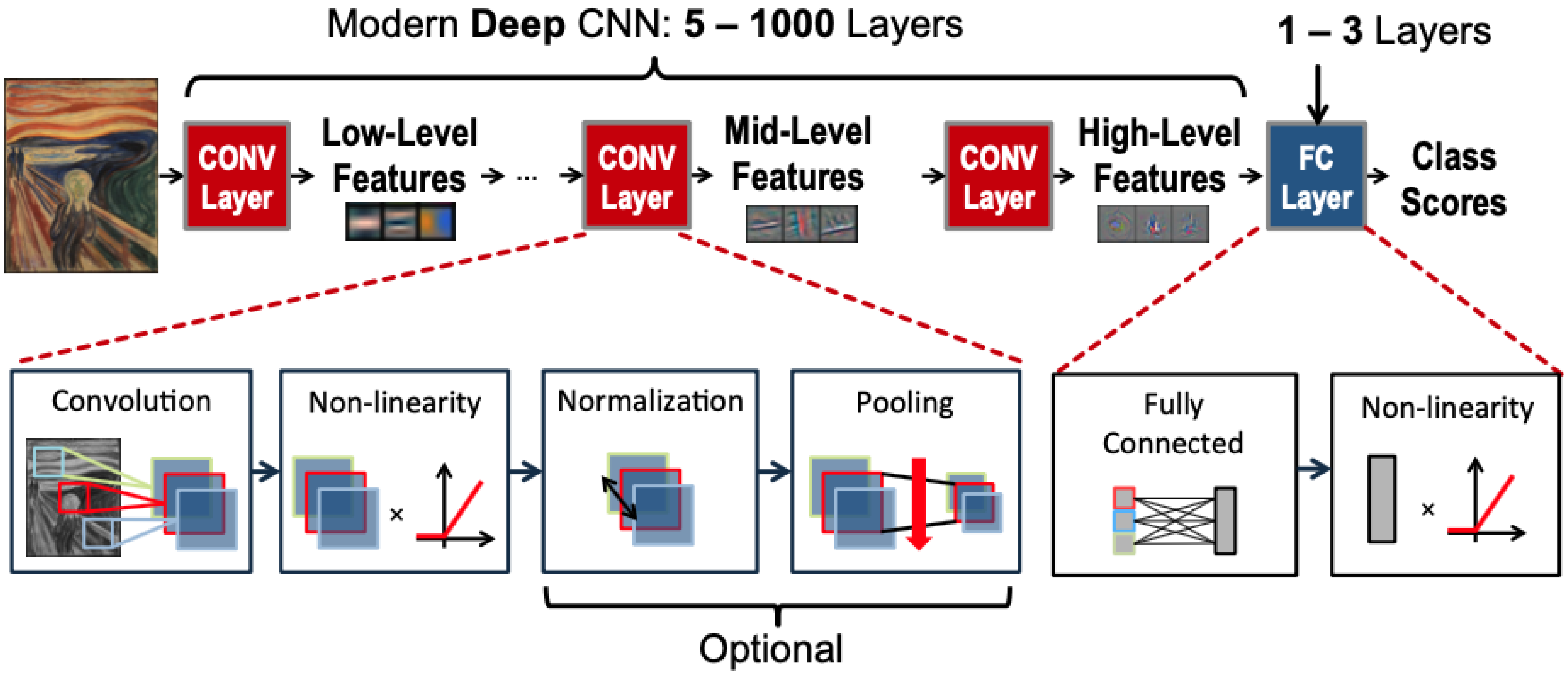
- Fully-connected layer
  is impractical for images
  Ex: for an image of 200x200x3
  - Fully connected layer:
    200 * 200 * 3 = 120,000 weights
  - Convolutional layer with a 5x5 filter:
    5 * 5 * 3 = 75 weights



MLP, Fully Connected Network

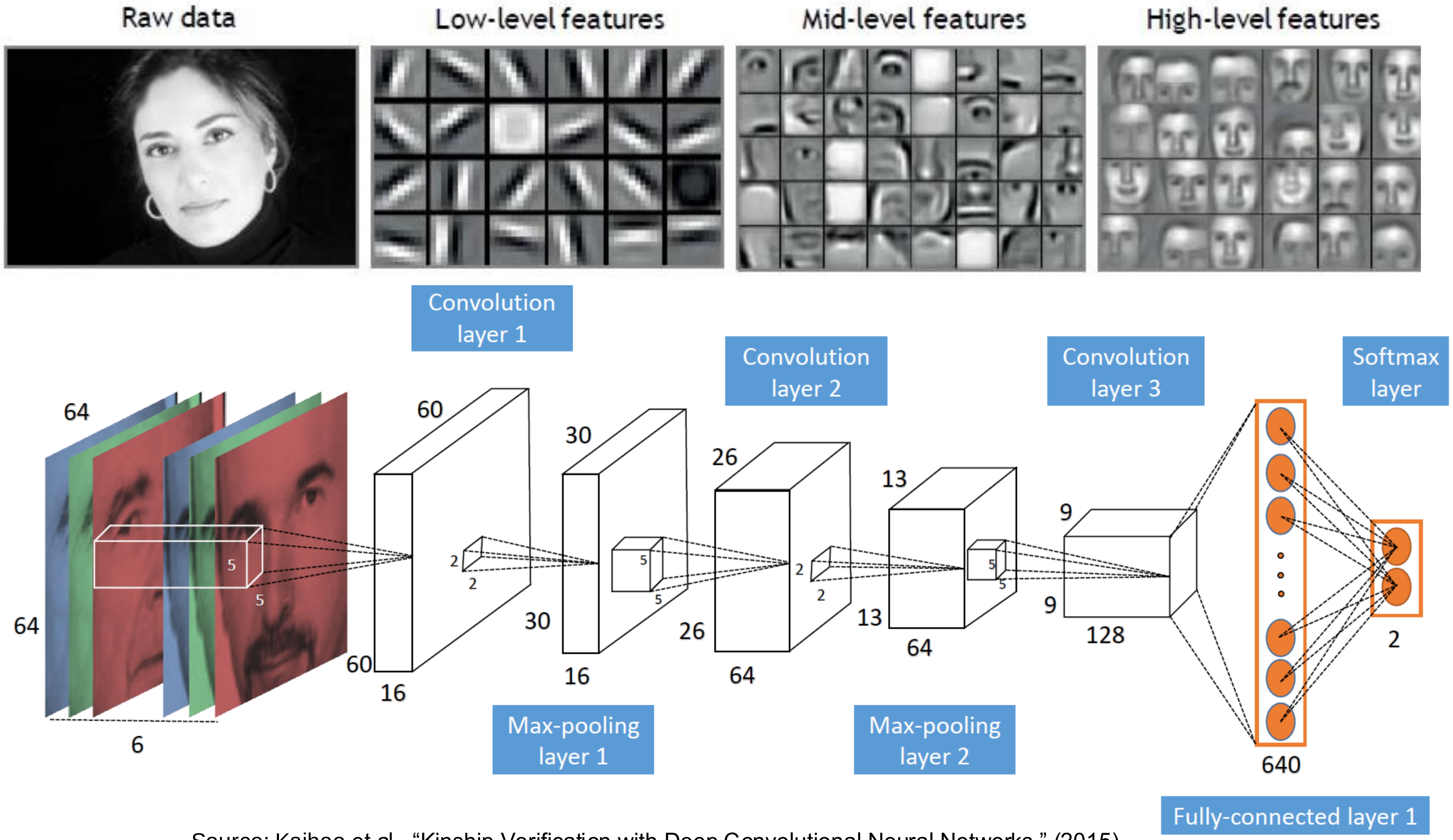
[Source: Stanford CS231n]

Deep Convolutional Neural Networks (DCNNs)



Architecture of CNN

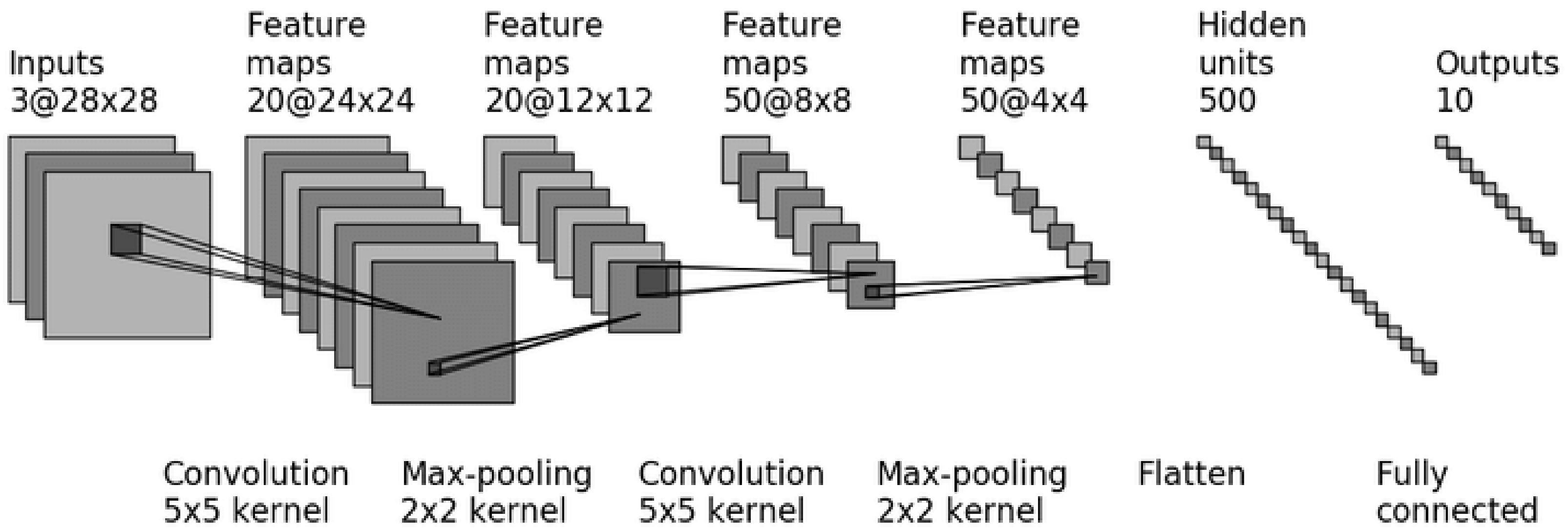
- A type of feed-forward artificial neural network where the response of an individual neuron to stimuli is approximated with a convolution operation

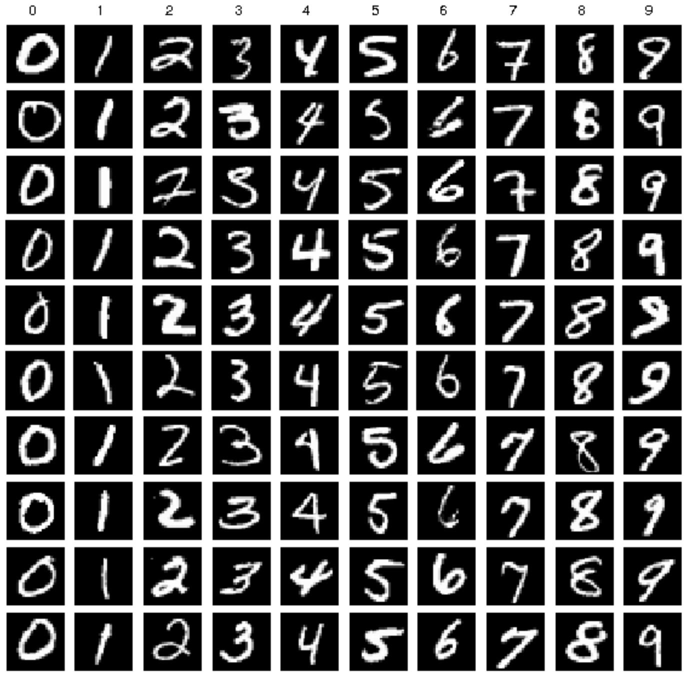


Source: Kaihao et al., “Kinship Verification with Deep Convolutional Neural Networks,” (2015)

LeNet-5 with MNIST Dataset

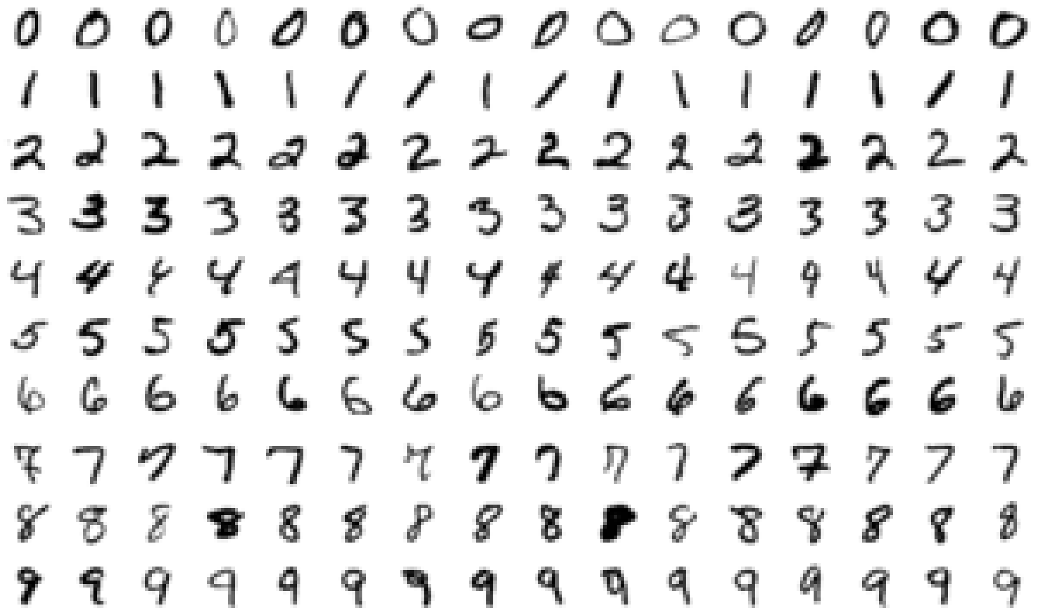
- Proposed by Yann LeCun et al. with Bell Labs in 1989
  - With backpropagation





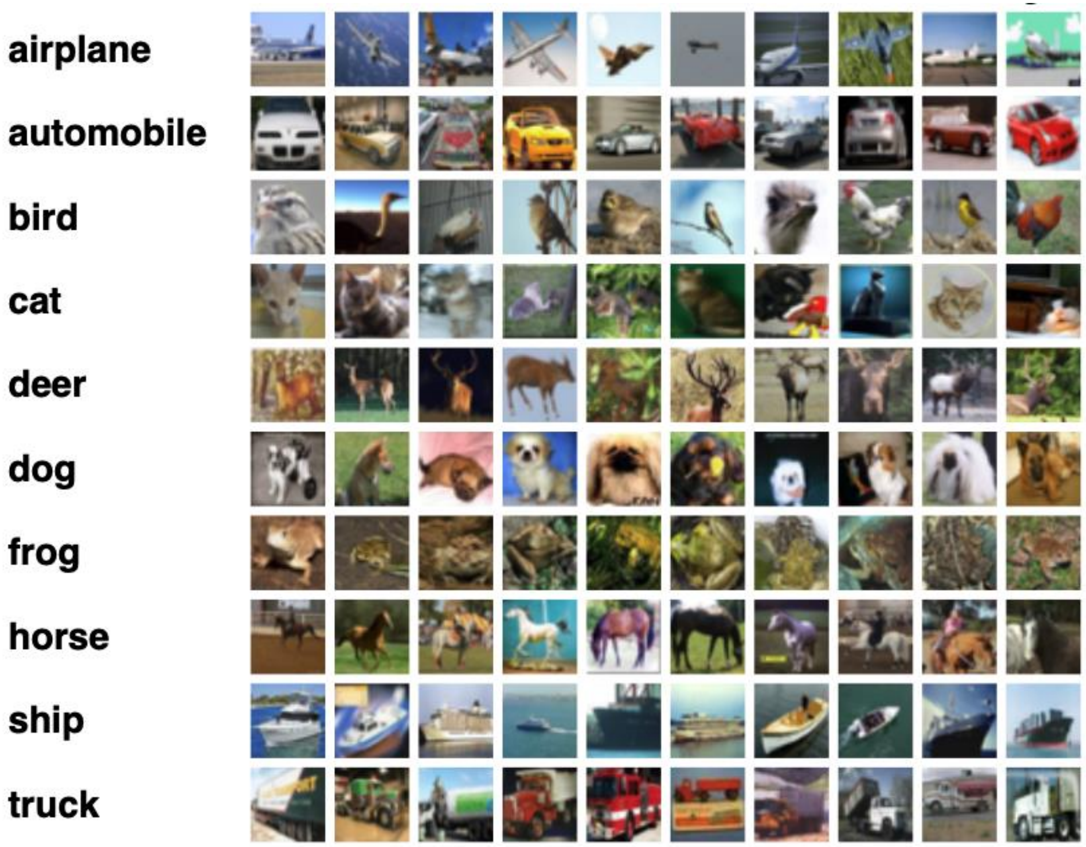
MNIST (Modified National Institute of Standards and Technology) Dataset

- 28x28 grayscale handwritten digits
- 60,000 training images
- 10,000 testing images
- URL: http://yann.lecun.com/exdb/mnist/



CIFAR10 (Canadian Institute for Advanced Research)

- 60,000 32x32 color images in 10 different classes
  - 10 classes:
    Airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck
  - 6,000 images of each class
- URL:
  https://www.cs.toronto.edu/~kriz/cifar.html

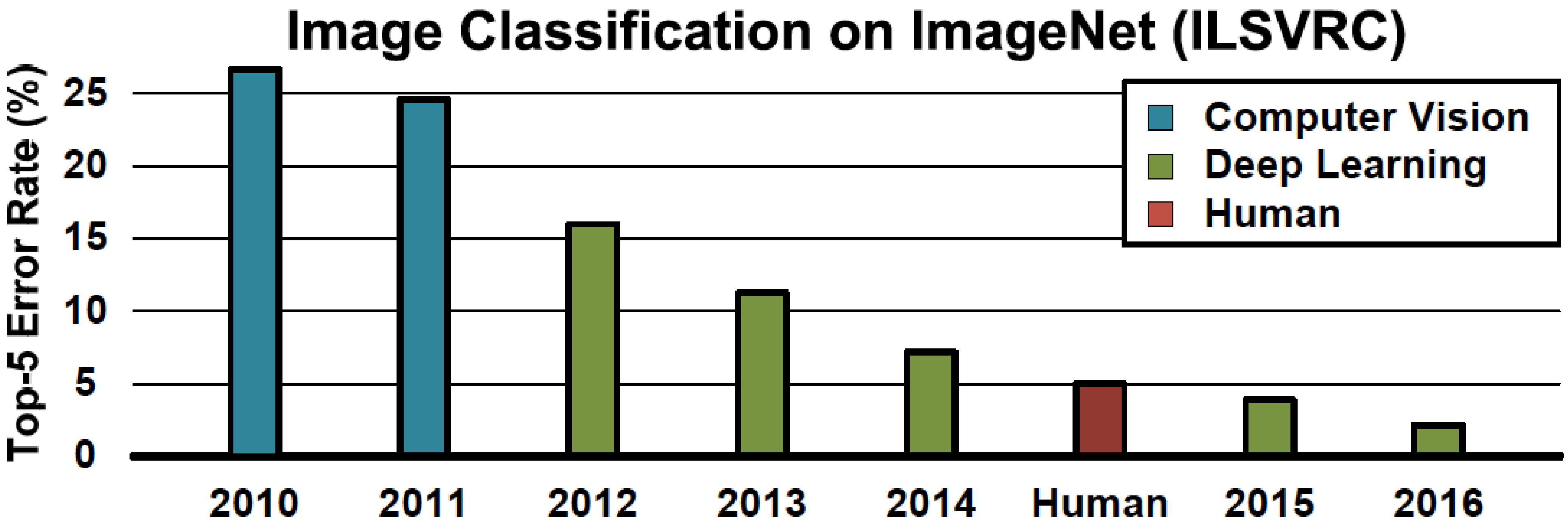


ImageNet

- A large crowdsourced, hand-annotated object annotation
- ImageNet Project runs the annual the ImageNet Large Scale Visual Recognition Challenge (ILSVRC)
- Typical input image size: 224x224 color pixels
- ImageNet LSVRC-2010 with 1.2 million pictures and 1000 classes
- Top-1 (top-5) error rate:
  - The fraction of images for which the correct label is not among the first (first 5) label(s) considered most probable by the model
- URL: http://www.image-net.org/

ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

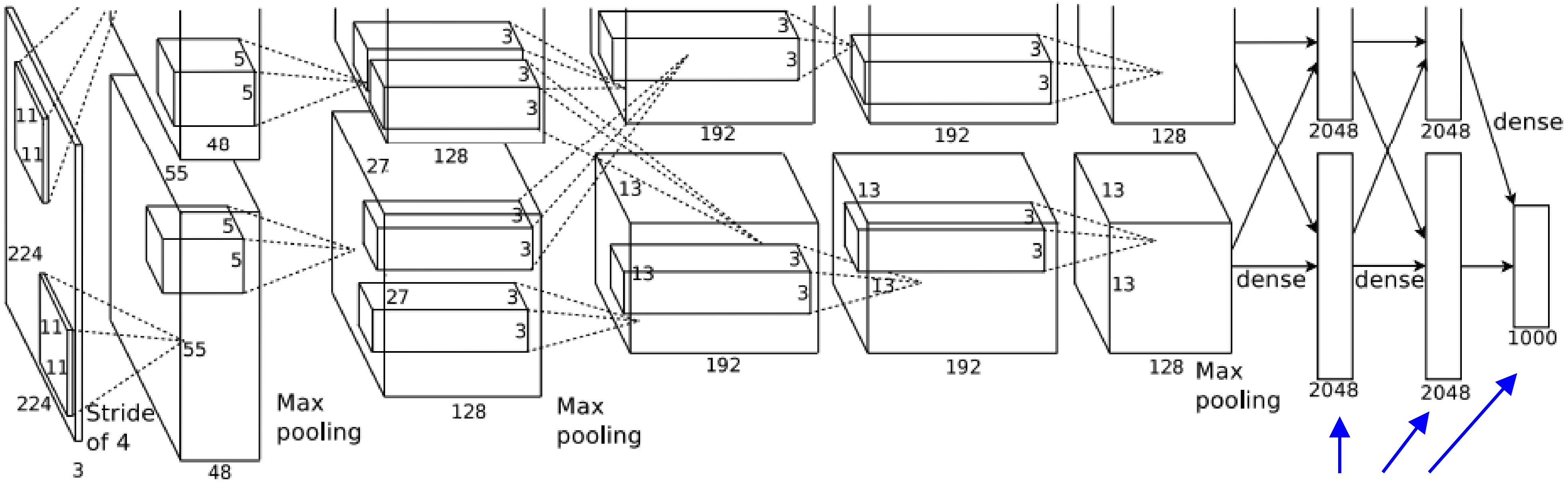
Top 5 Classification Error (%)



[Source: 14.2, ISSCC'17]

AlexNet

5 Convolutional Layers

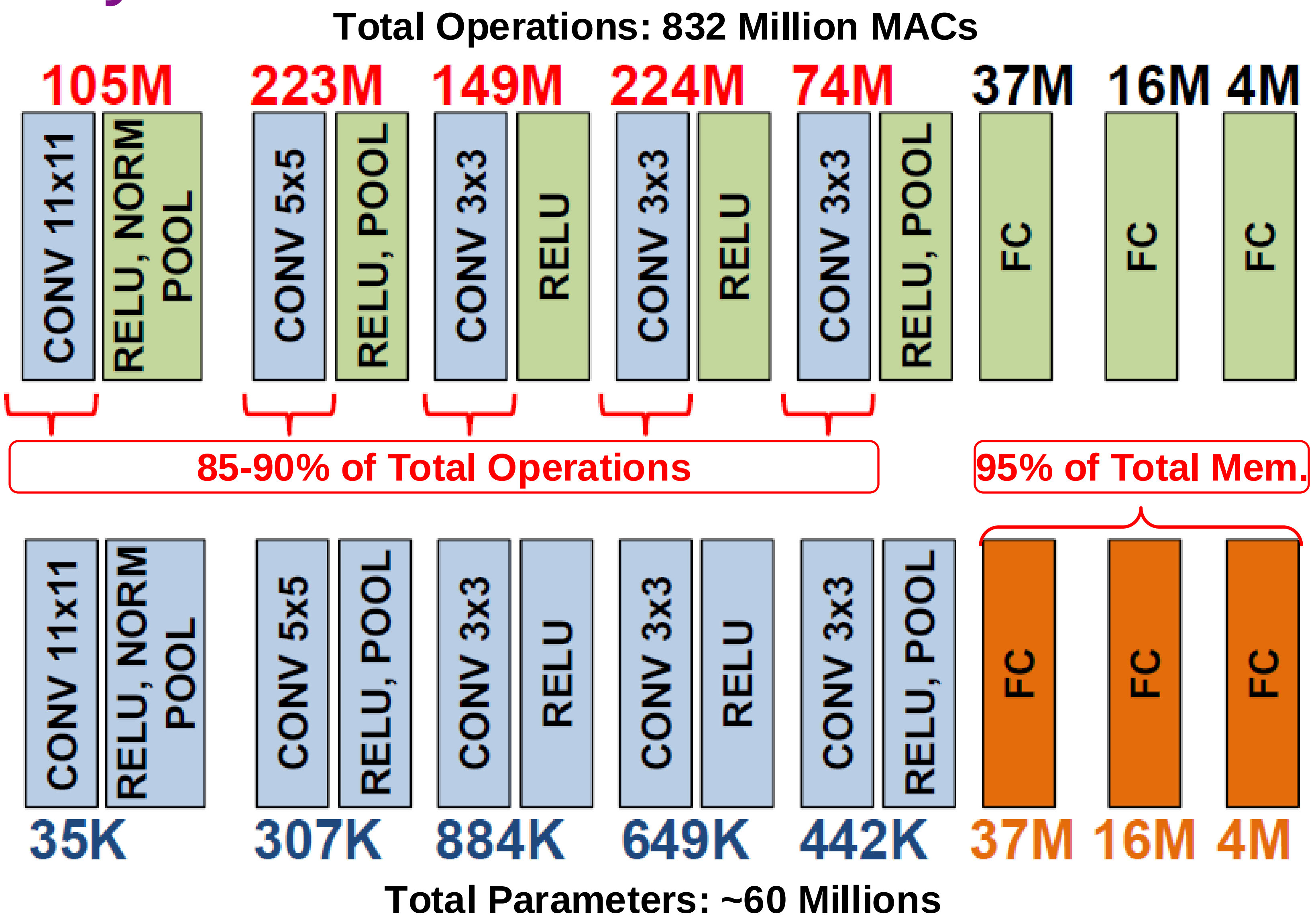


3 Fully Connected Layers

Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

ImageNet Classification with Deep Convolutional Neural Networks

Complexity of AlexNet

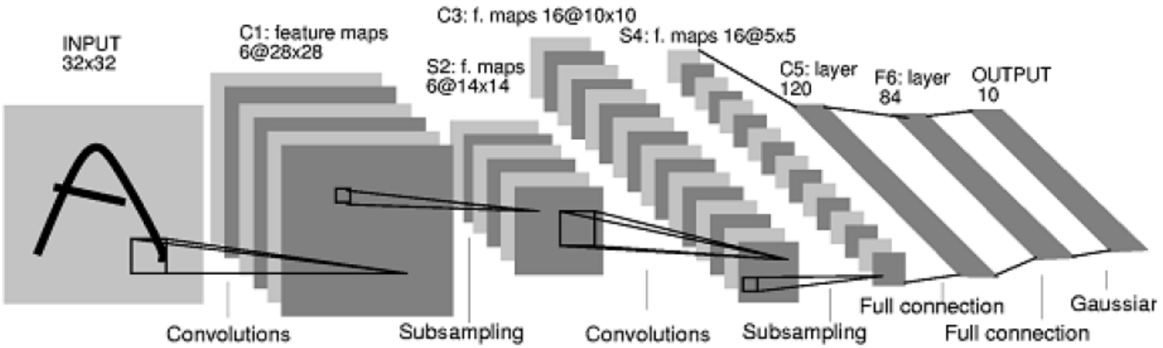


Source: Alex Krizhevsky et al., "ImageNet Classification with Deep Convolutional Neural Networks," (2012)

LeNet vs. AlexNet

1989

LeCun et al.



# of transistors

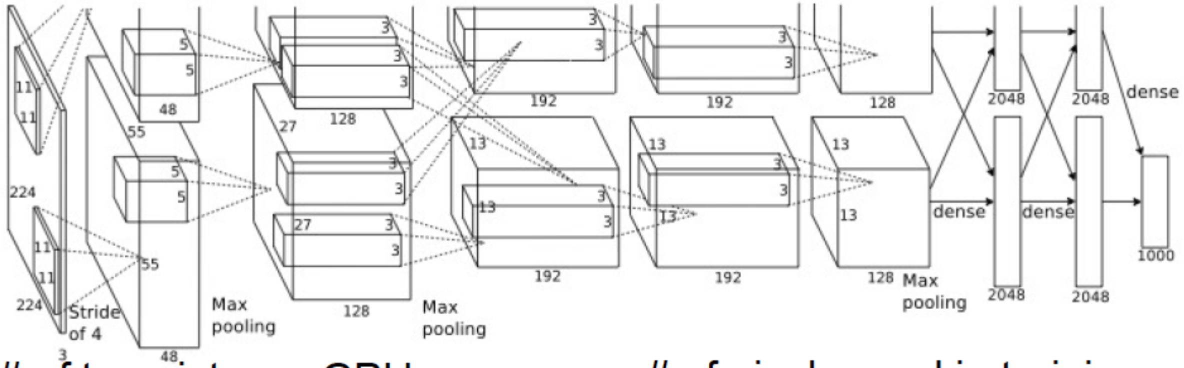
10^6

# of pixels used in training

10^7 NIST

2012

Krizhevsky et al.



# of transistors

10^9

GPUs

# of pixels used in training

10^{14} IMAGENET

[Source: Stanford CS231n]

“It’s not who has the best algorithm that wins. It’s who has the most data.”

- Banko and Brill, 2001

“Datasets—not algorithms—might be the key limiting factor to development of human-level artificial intelligence”

- Alexander Wissner-Gross (edge.org, 2016)