

Convolutional neural network (CNN) for computer vision

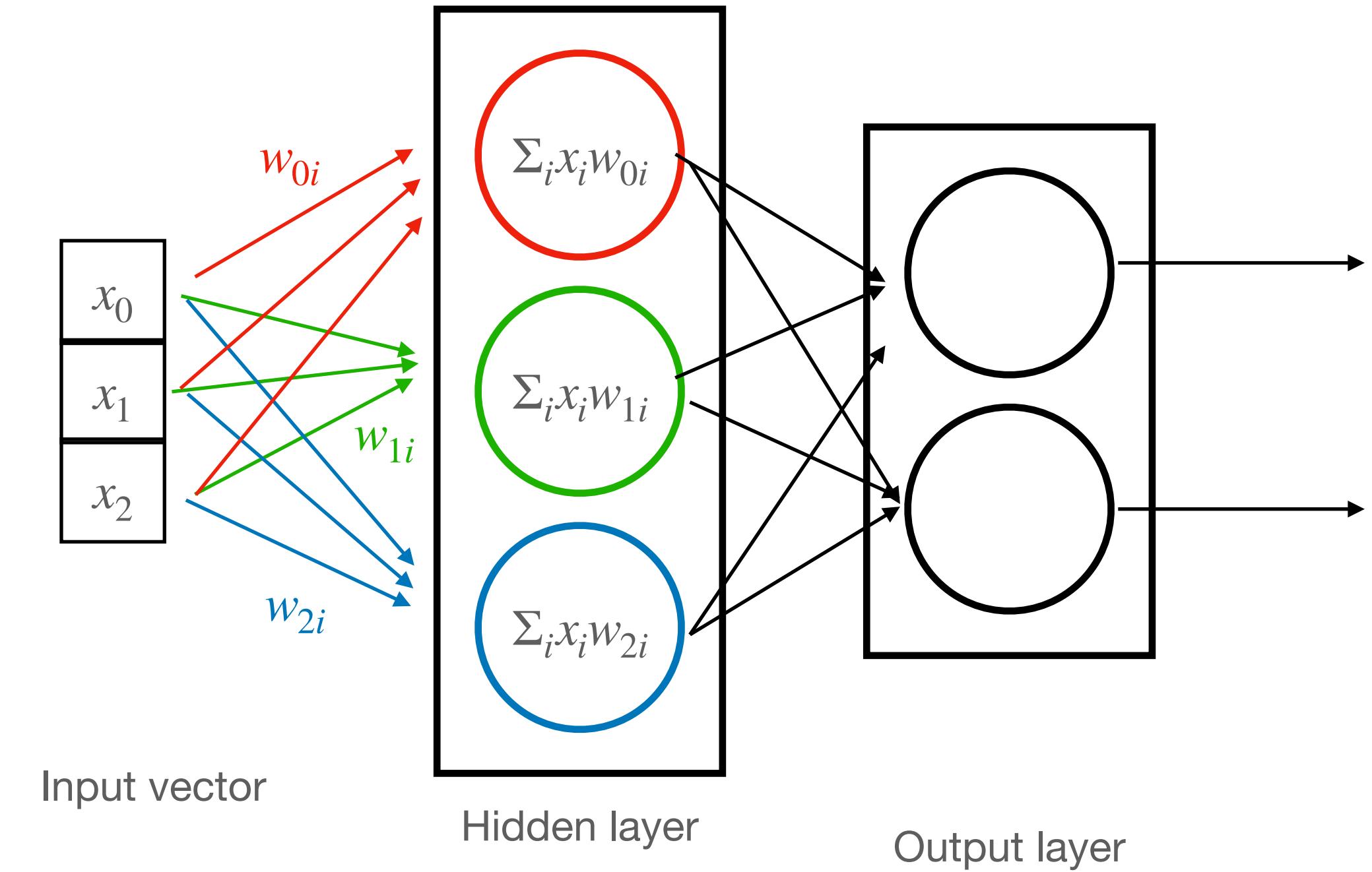
Insop

Agenda

1. Review: Neural Network
2. Image classification, motivation
3. Convolutional neural network
4. CNN architecture

Neural network

- Each neuron: sum of weighted input passed to **non-linear** function
- **Layer**: multiple neurons
- **Multiple layers**: connecting many layers
- Neurons in each layers are connected

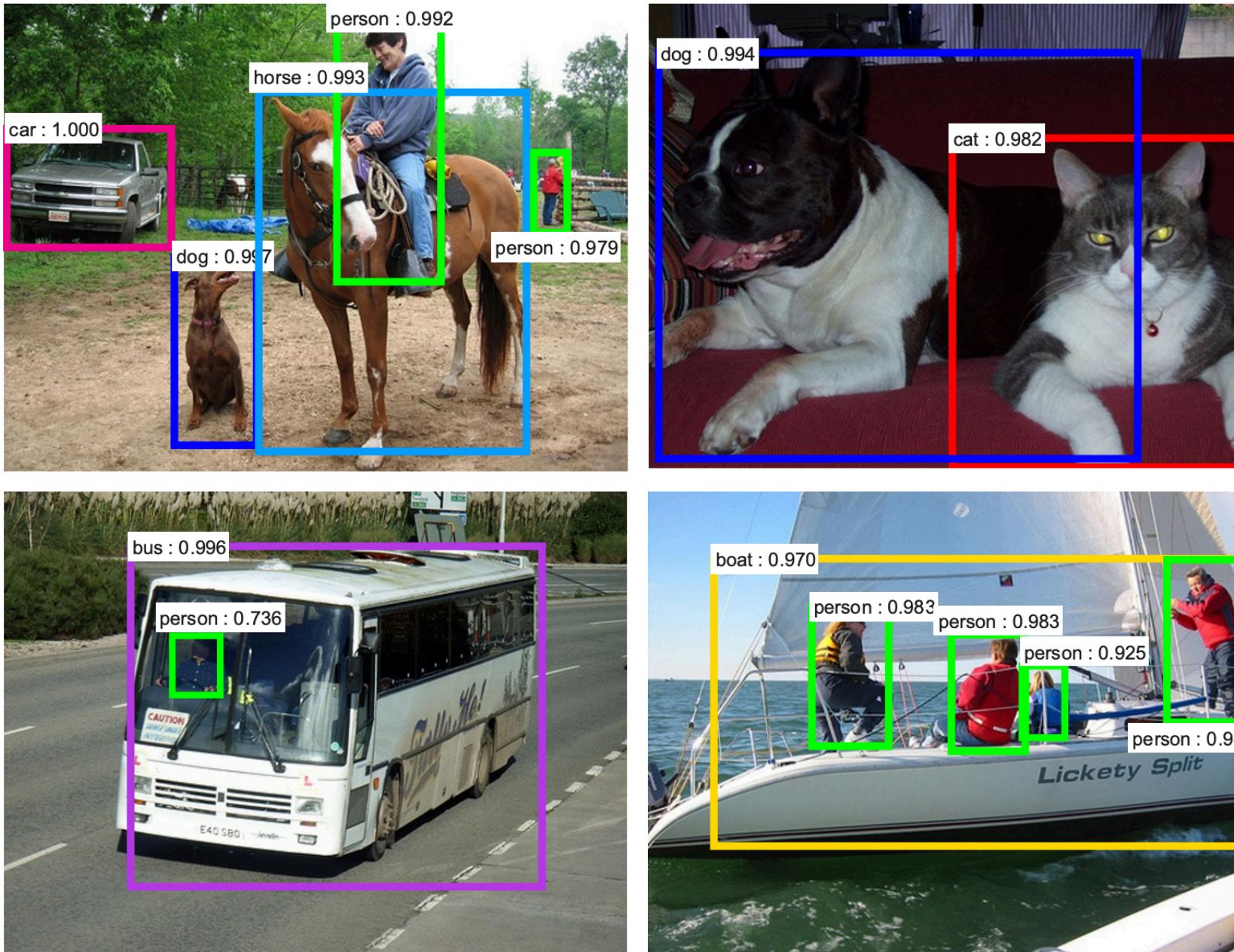


$$\begin{bmatrix} w_{00}x_0 + w_{01}x_1 + w_{02}x_2 \\ w_{10}x_0 + w_{11}x_1 + w_{12}x_2 \\ w_{20}x_0 + w_{21}x_1 + w_{22}x_2 \end{bmatrix} = \begin{bmatrix} w_{00} & w_{01} & w_{02} \\ w_{10} & w_{11} & w_{12} \\ w_{20} & w_{21} & w_{22} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$$

Deep learning breakthrough

- Alexnet started deep learning breakthrough using CNN in Imagenet competition in 2012
- Since then CNN is used every where in computer vision applications

Object detection



Source: <https://arxiv.org/pdf/1506.01497.pdf>

Image segmentation



Source: <https://arxiv.org/pdf/1703.06870.pdf>

Image classification

Fully connected neural network?

- Image classification with neural network
- Flatten image and use it as input
- example>



32x32 with single color

- Input: 32x32 image
- Output: 10 class

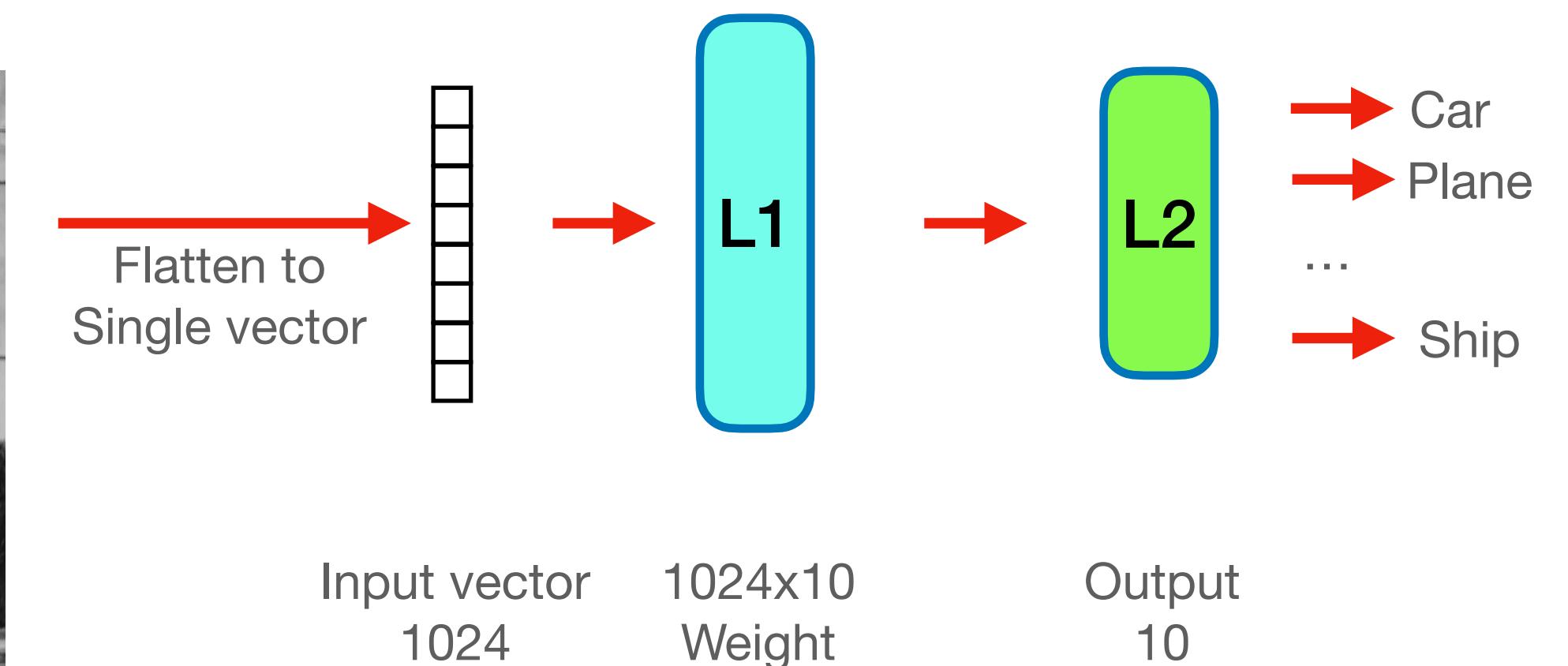
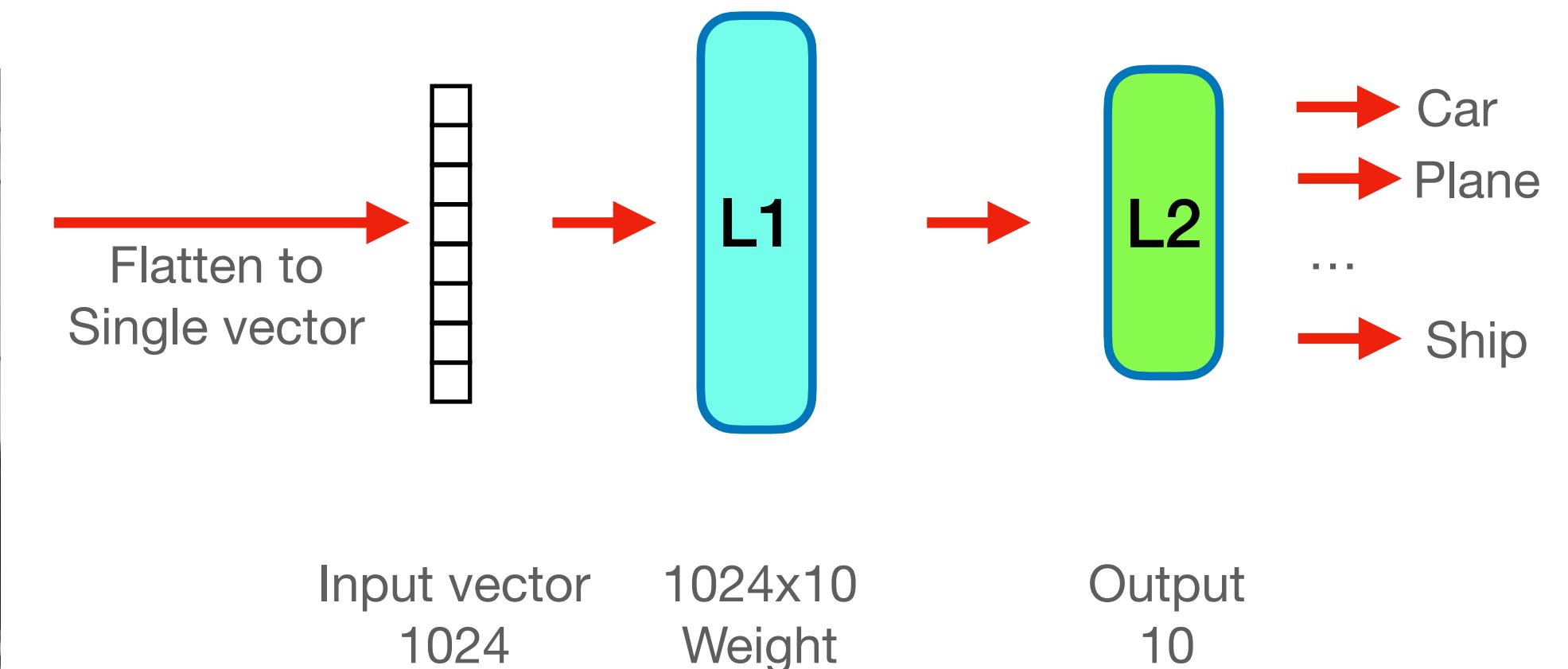


Image classification

Fully connected neural network is not efficient for image.

- Spatial information is hard to capture
- Parameter only learn pixel information in specific position
- Number of parameter can get large



Kernel in image processing

- Kernel (3x3 or 5x5 matrix) is applied to images by sliding
- Based on the kernel, it can detect edges, blur, sharpen images

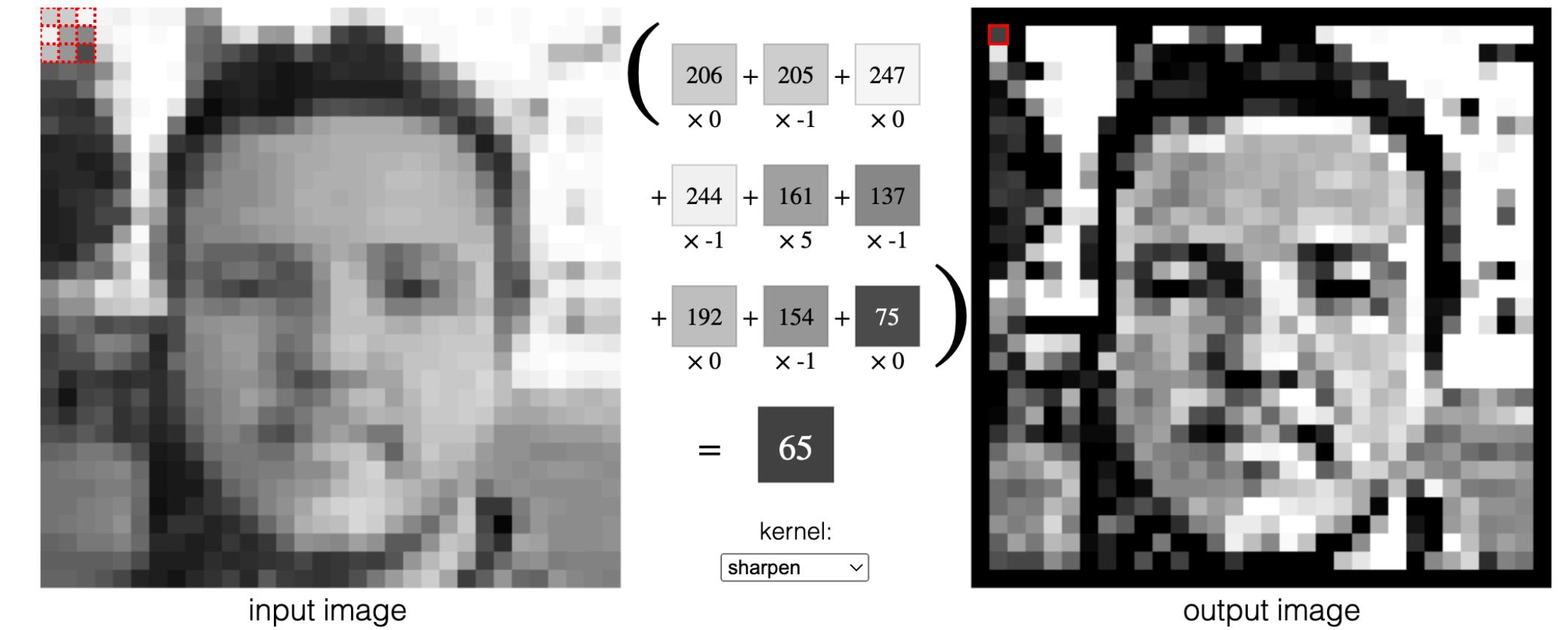
0	-1	0
-1	5	-1
0	-1	0

sharpen



$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

Below, for each 3x3 block of pixels in the image on the left, we multiply each pixel by the corresponding entry of the kernel and then take the sum. That sum becomes a new pixel in the image on the right. Hover over a pixel on either image to see how its value is computed.



Source: <https://setosa.io/ev/image-kernels/>

Image classification

Convolutional neural network

- Convolve kernel over the image
- i.e. slide the kernel over the image and do the dot product (multiply each element and sum them)
- Keeping the spatial information
- Kernel is reused for all inputs

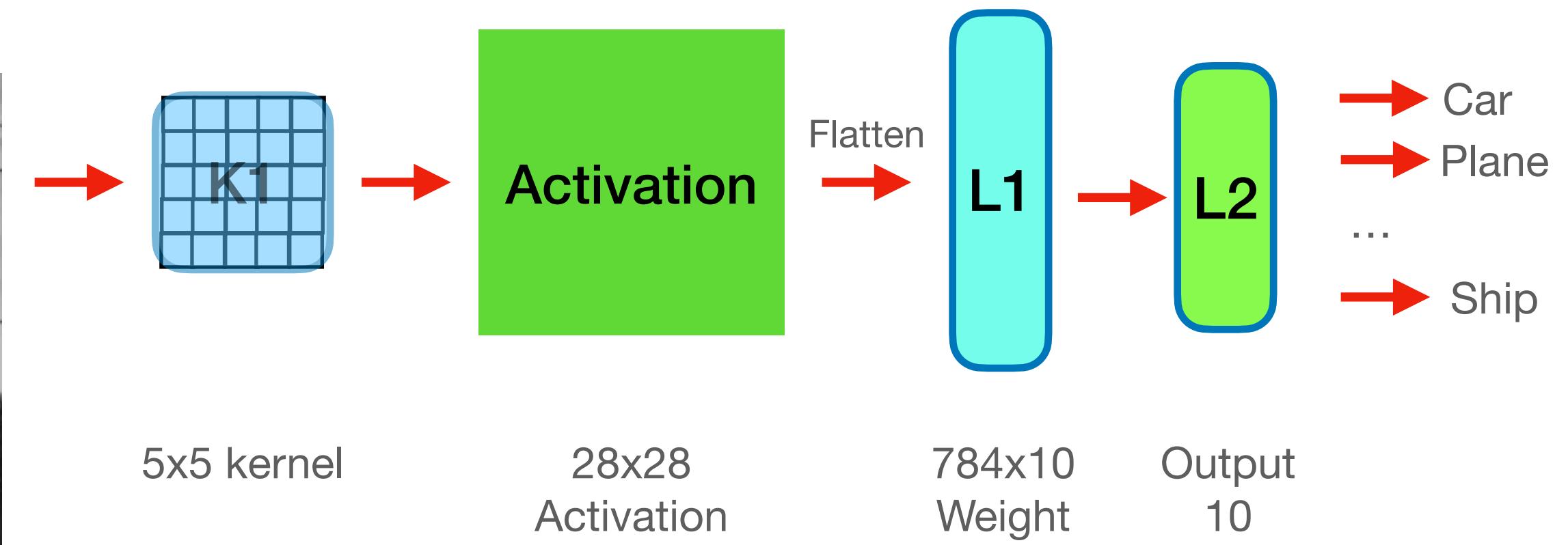
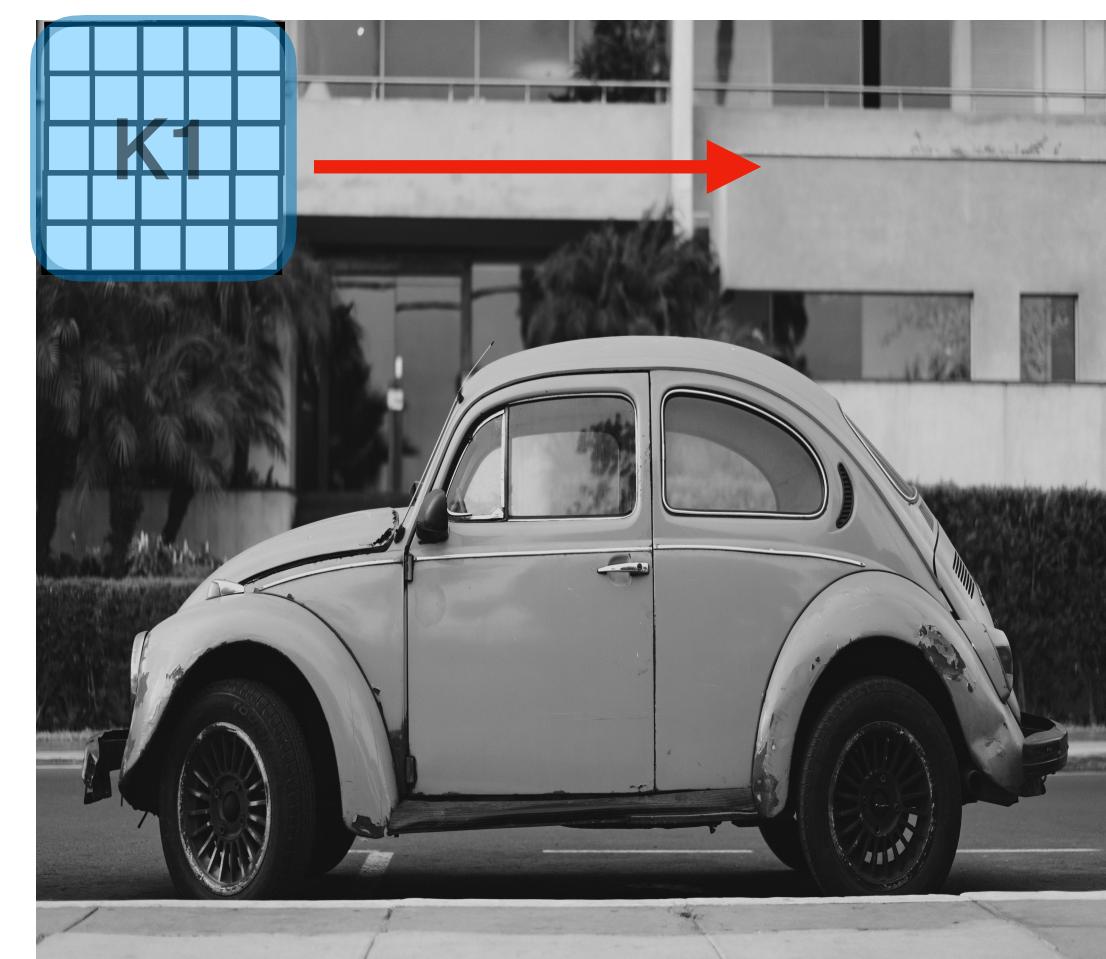


Image classification

Convolutional neural network

- Color image, rgb:
defined as **3 channels**
- Number kernel is always
based on the input
channel
- i.e. 3 in kernel in this
example

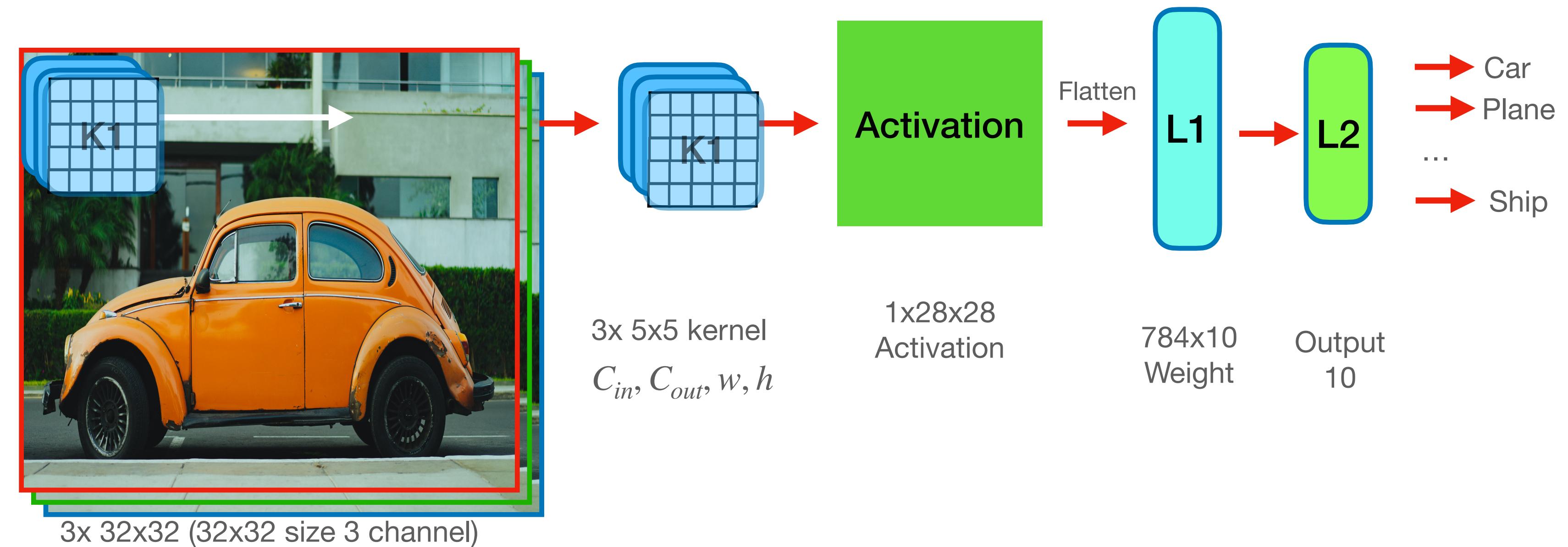
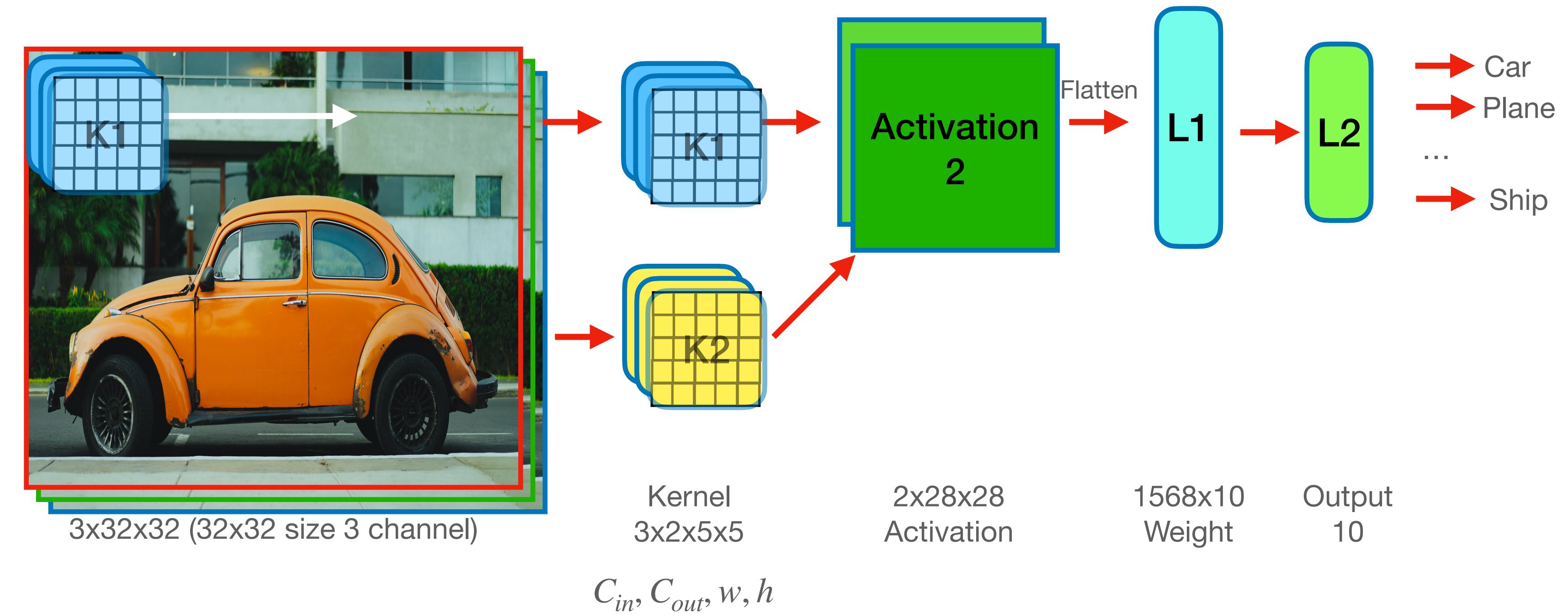


Image classification

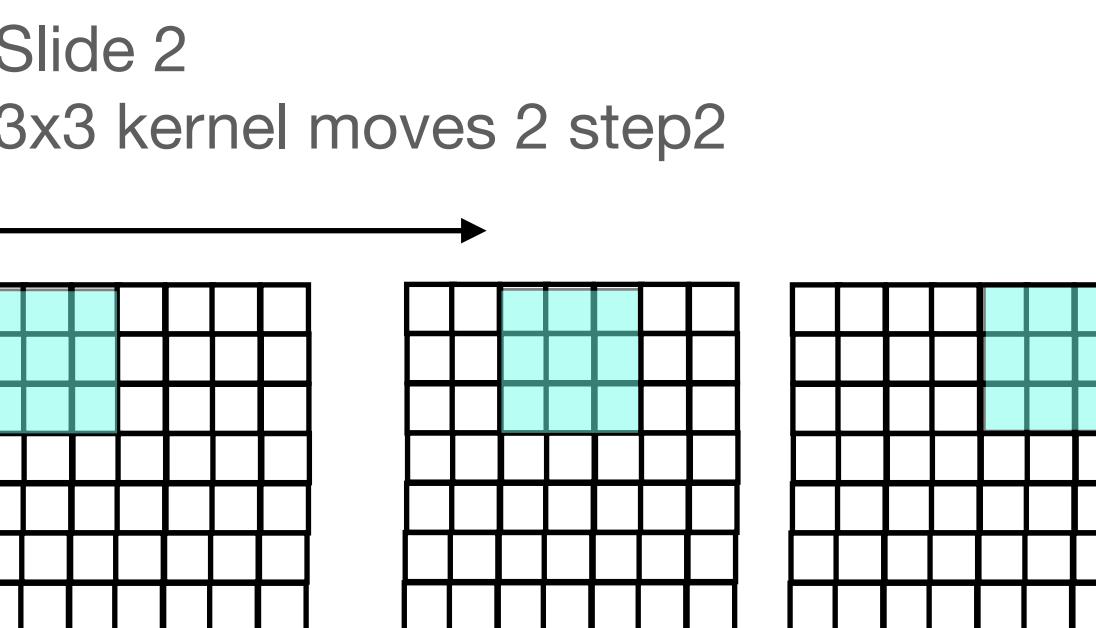
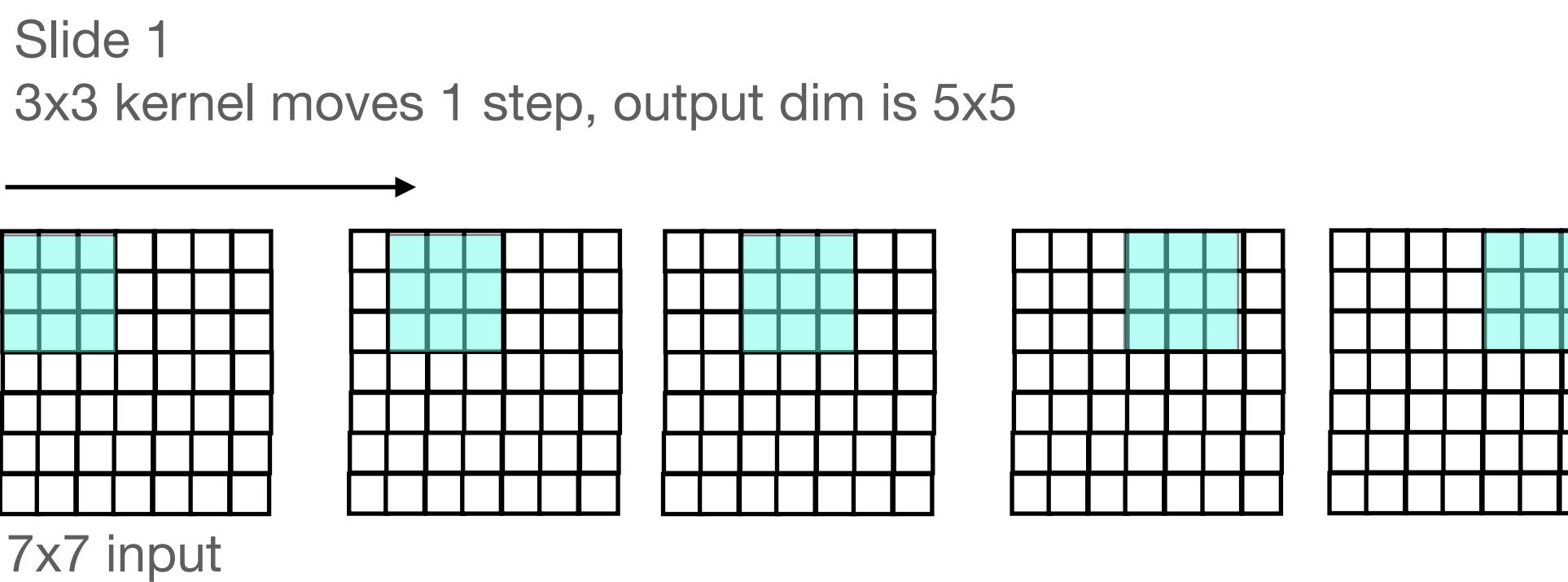
Convolutional neural network

- Multiple kernel can be used



Stride

- Define how many steps to move
- Stride 3? Won't fit
- Output = $\frac{(N - F)}{S} + 1$
- N: input size (assume same height&width)
- F: kernel size
- S: stride

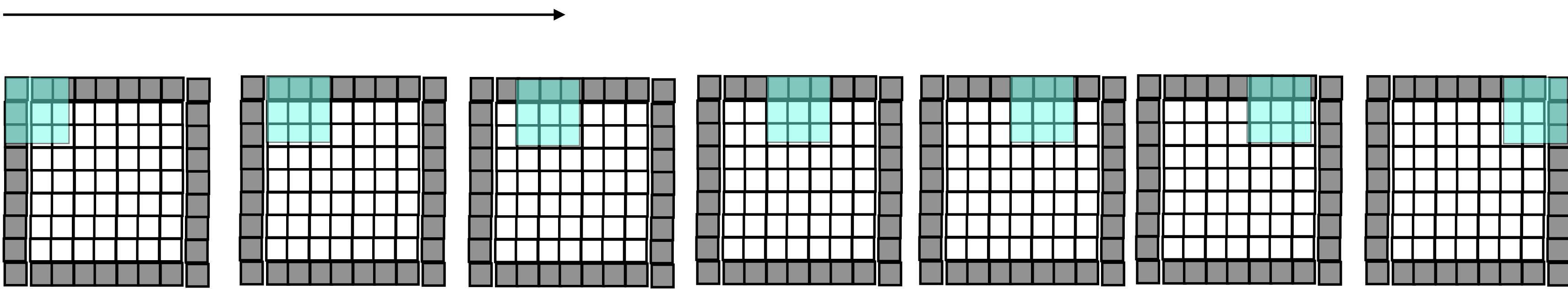


Padding

- To preserve the same dimension for output
- Add paddings (0) to the input
- Output :
 - $\frac{(N + 2P - F)}{S} + 1$
- N: input size (assume same height&width)
- F: kernel size
- S: stride
- P: padding
- To preserve the size, add padding size
 $P = (F - 1)/2$
 - When $F=3 \rightarrow P=1$, $F=5 \rightarrow P=2$

Slide 1, **padding 1**

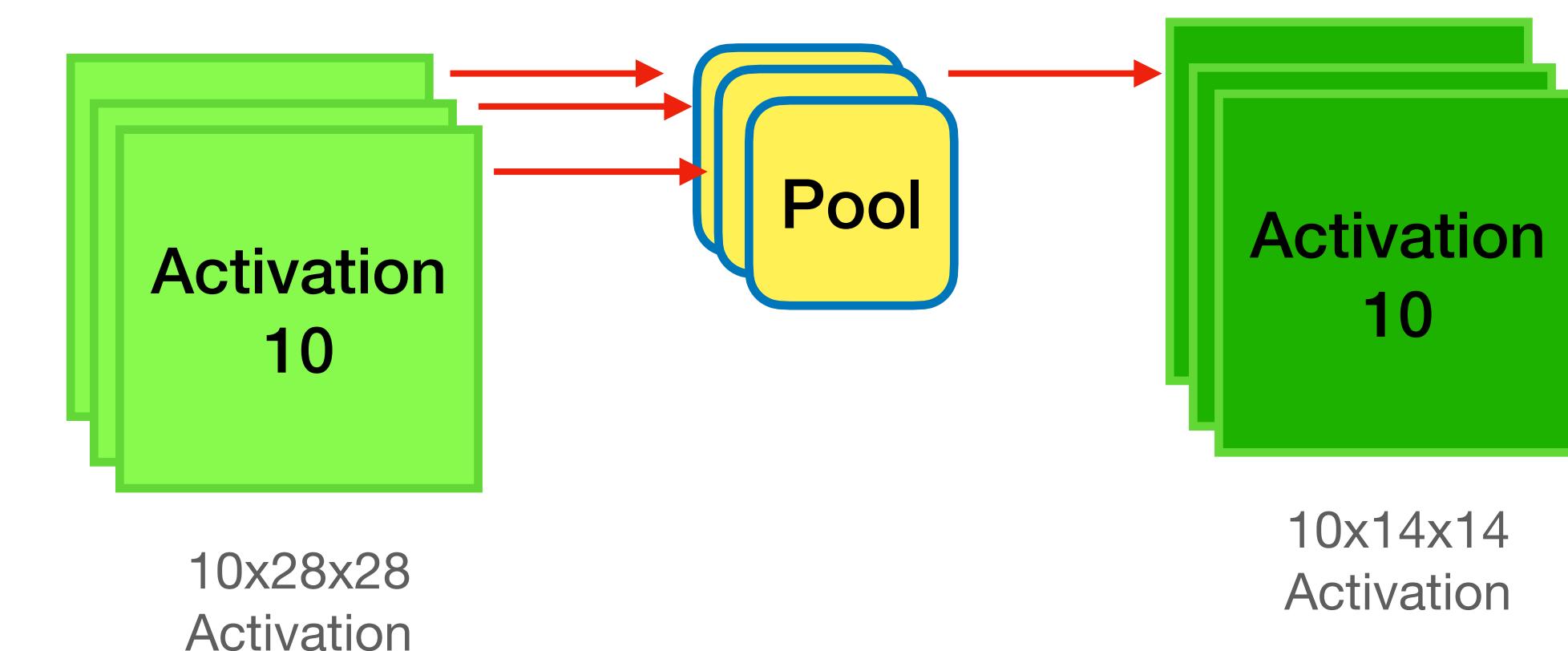
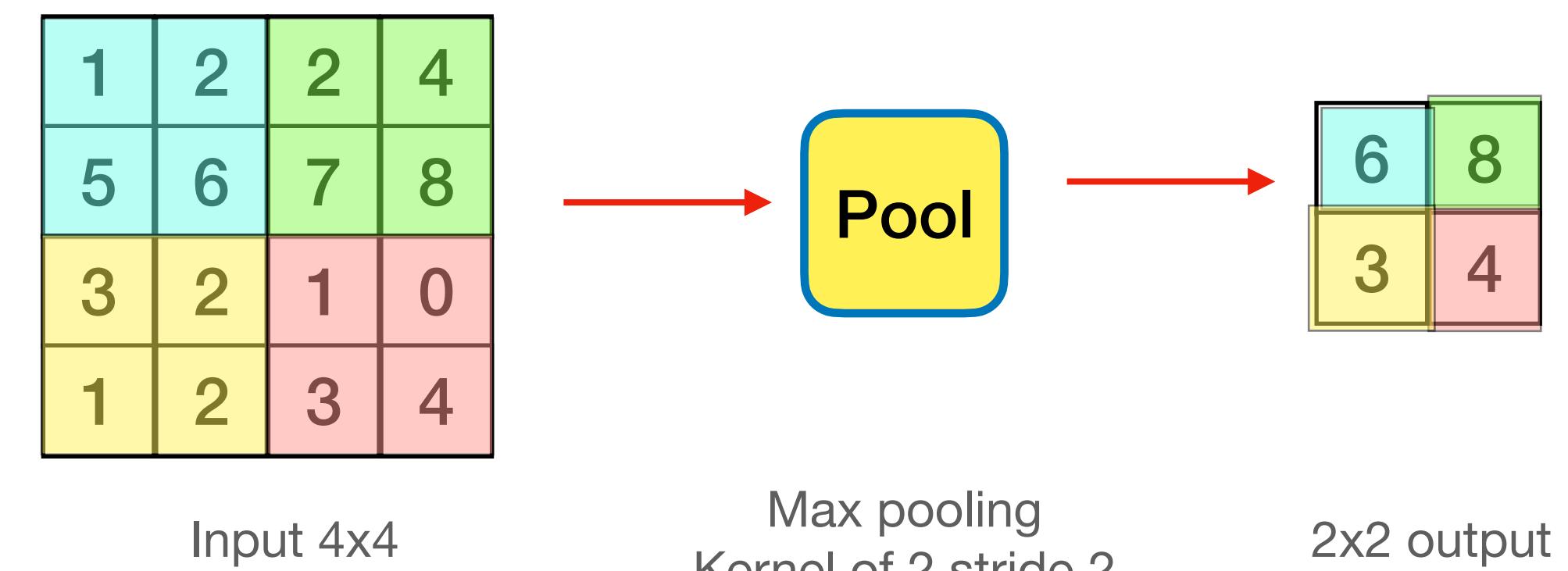
3x3 kernel moves 1 step, output dim is 7x7



7x7 input
Gray area is zero padding

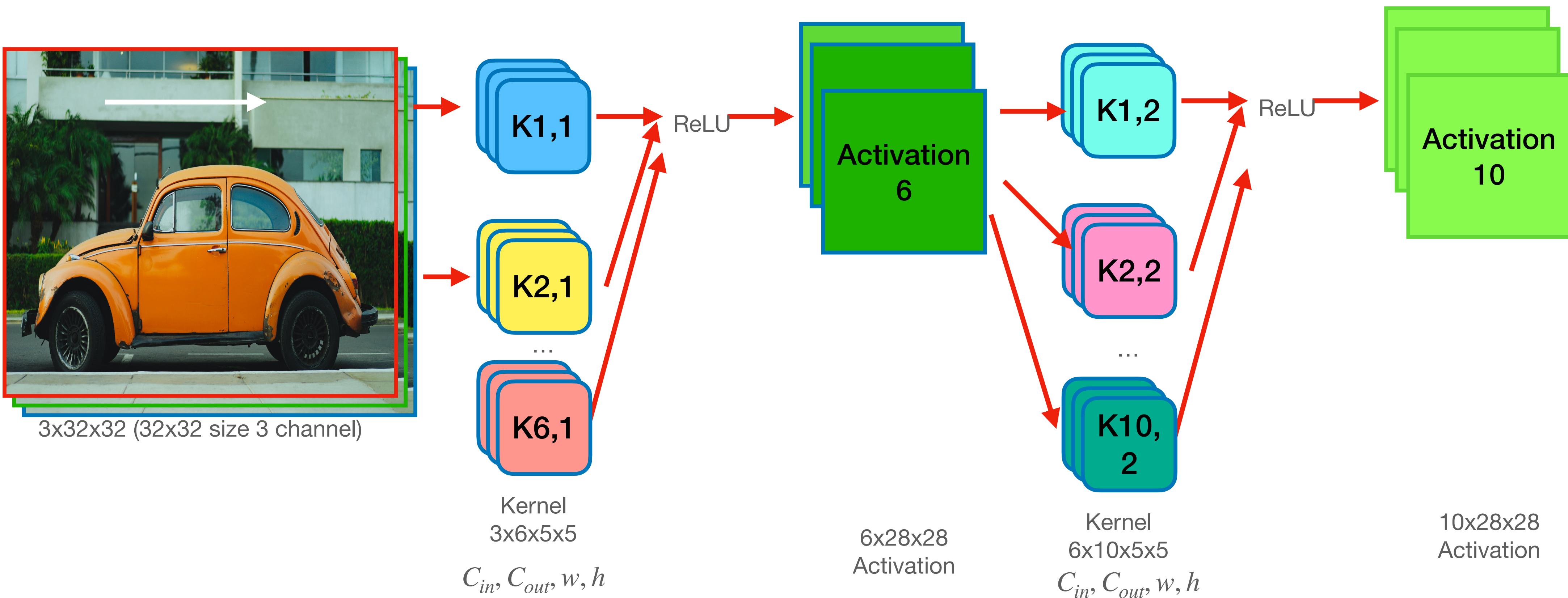
Pooling

- Reduce the activation size
- Applied to each activation layers independently
- Downsample the input
- Max, average pool are used
- No learnable parameters



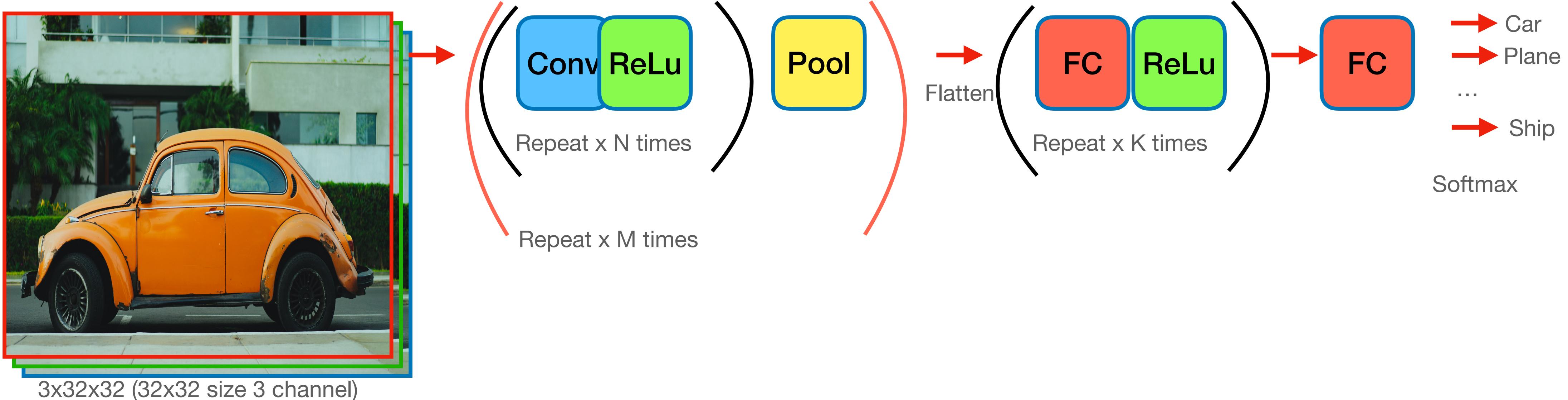
CNN structure

- Multiple Conv kernel layer with non-linear layer (ReLU)



CNN structure

- Repeats with conv, relu, and pooling layer, and add fully connected layer at the end



Credits

The following materials are referenced for this slide.

- Sample CNN code from Pytorch tutorial
- Full Stack Deep Learning
- Deep Learning for Computer Vision