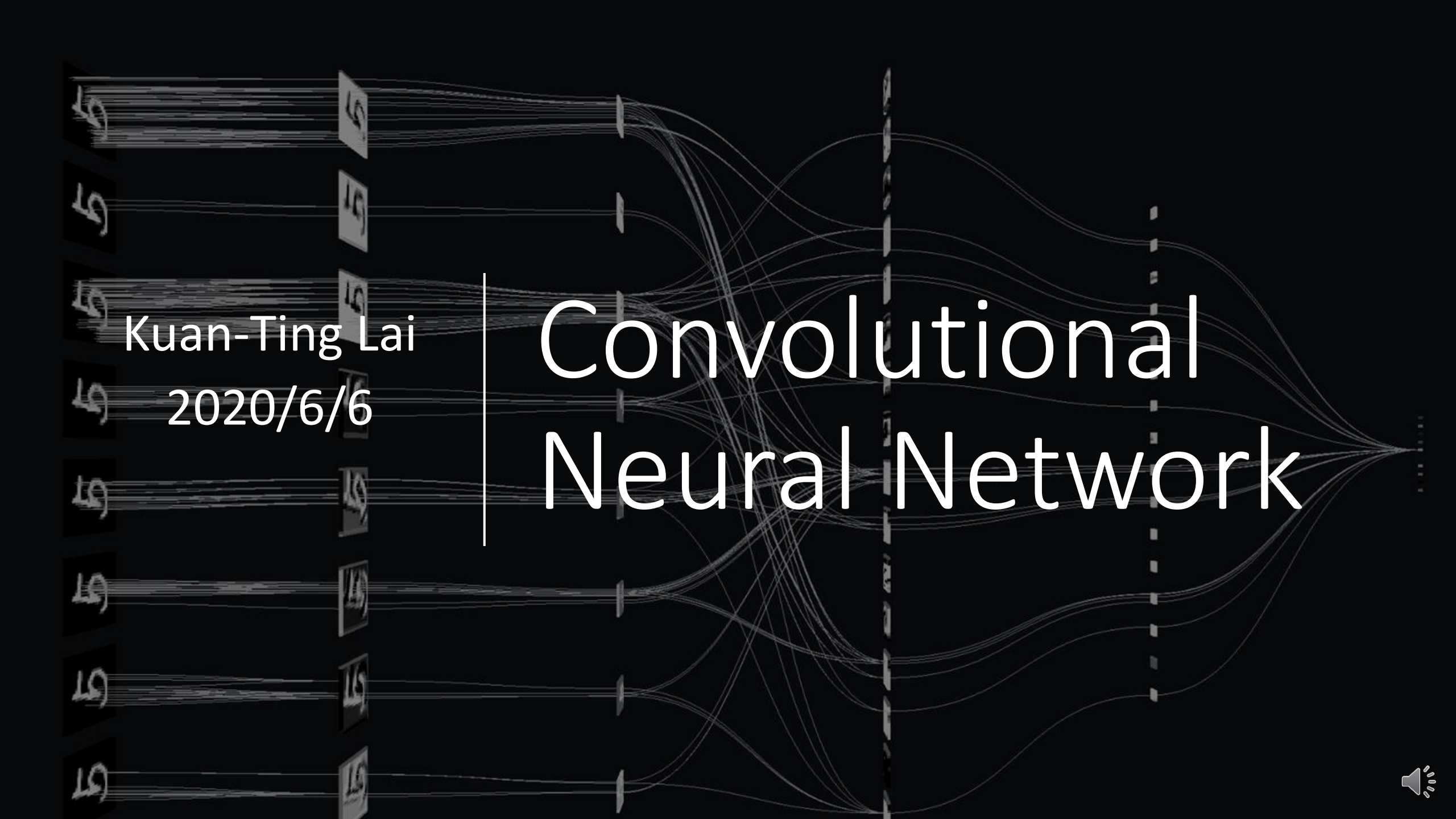
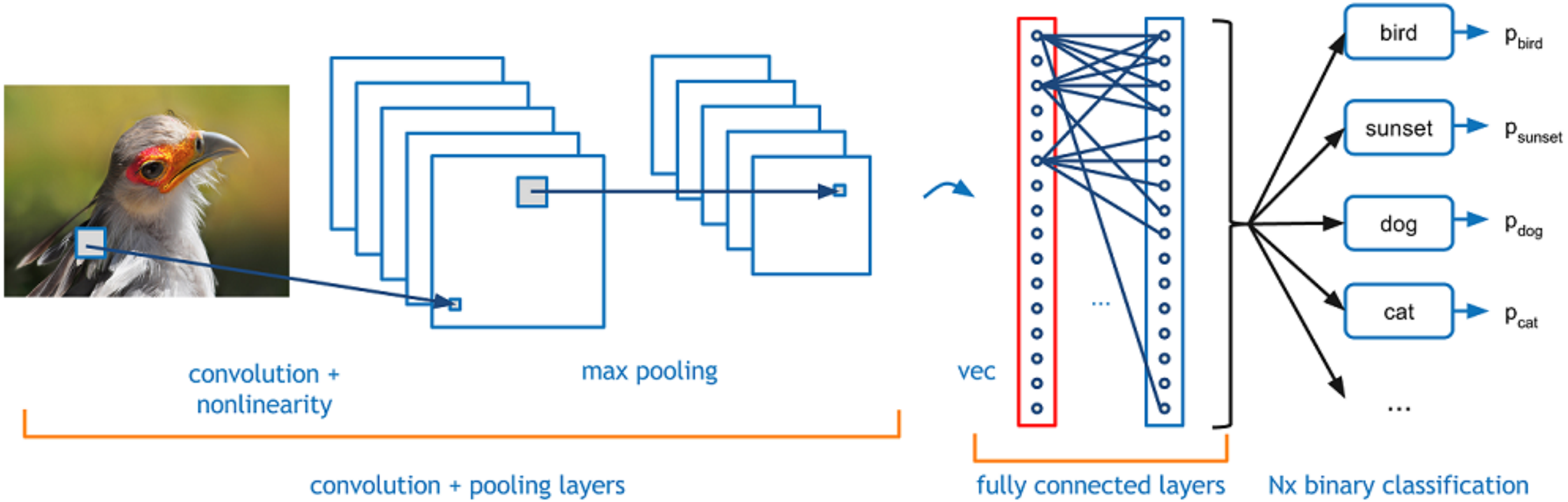


Kuan-Ting Lai
2020/6/6

Convolutional Neural Network

Convolutional Neural Networks (CNN)

- A.k.a. CNN or ConvNet



Adit Deshpande, [A Beginner's Guide To Understanding Convolutional Neural Networks](#).

Digital Images

- Input array: an image's height × width × 3 (RGB)
- Value of each pixel: 0 - 255

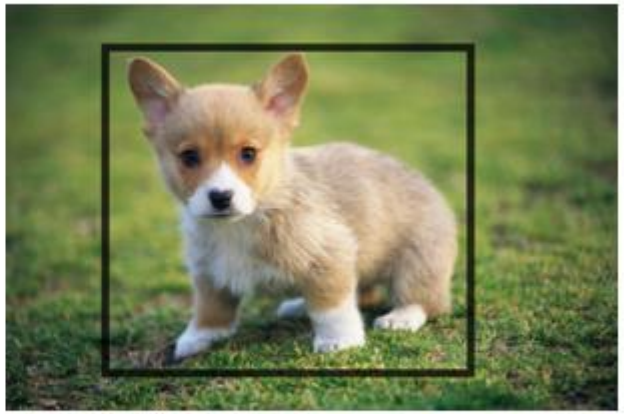
What We See

```
08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 91 08
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 48 04 56 62 00
81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 91
22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80
24 47 32 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50
32 98 81 28 64 23 67 10 26 38 40 67 59 54 70 66 18 38 64 70
67 26 20 68 02 62 12 20 95 63 94 39 63 08 40 91 66 49 94 21
24 55 58 05 66 73 99 26 97 17 78 78 96 83 14 88 34 89 63 72
21 36 23 09 75 00 76 44 20 45 35 14 00 61 33 97 34 31 33 95
78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92
16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57
86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58
19 80 81 68 05 94 47 69 28 73 92 13 86 52 17 77 04 89 55 40
04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66
88 36 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69
04 42 16 73 38 25 39 11 24 94 72 18 08 46 29 32 40 62 76 36
20 69 36 41 72 30 23 88 34 62 99 69 82 67 59 85 74 04 36 16
20 73 35 29 78 31 90 01 74 31 49 71 48 86 81 16 23 57 05 54
01 70 54 71 83 51 54 69 16 92 33 48 61 43 52 01 89 19 67 48
```

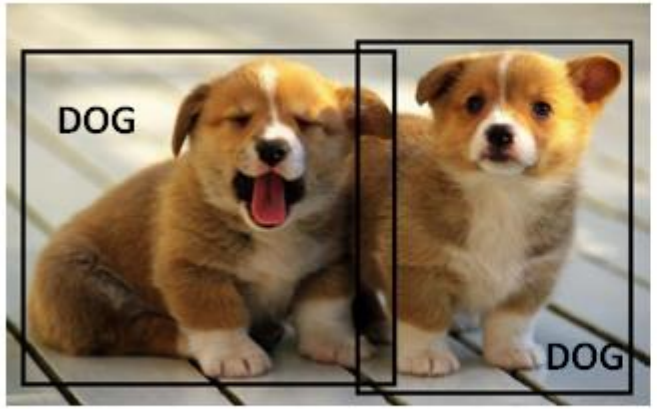
What Computers See

Classification, Localization, Detection, Segmentation

Object Classification is the task of identifying that picture is a dog



Object Localization involves the class label as well as a bounding box to show where the object is located.

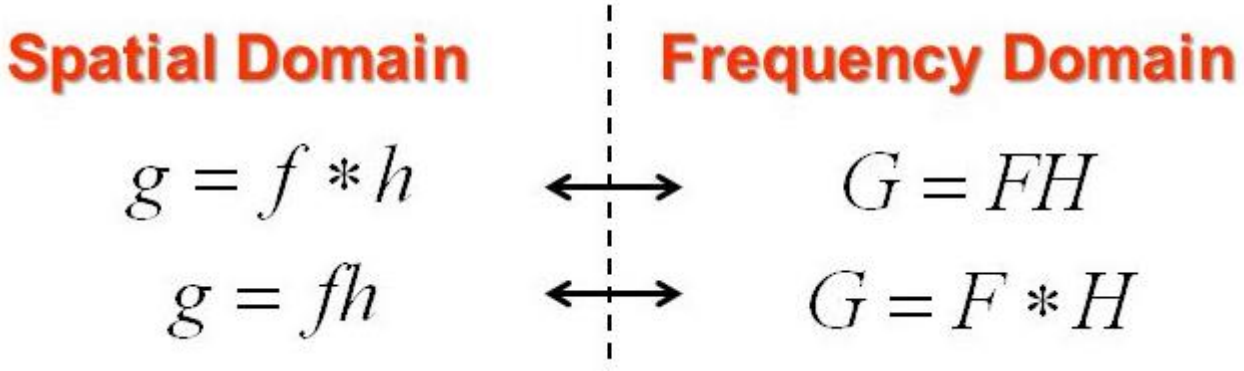


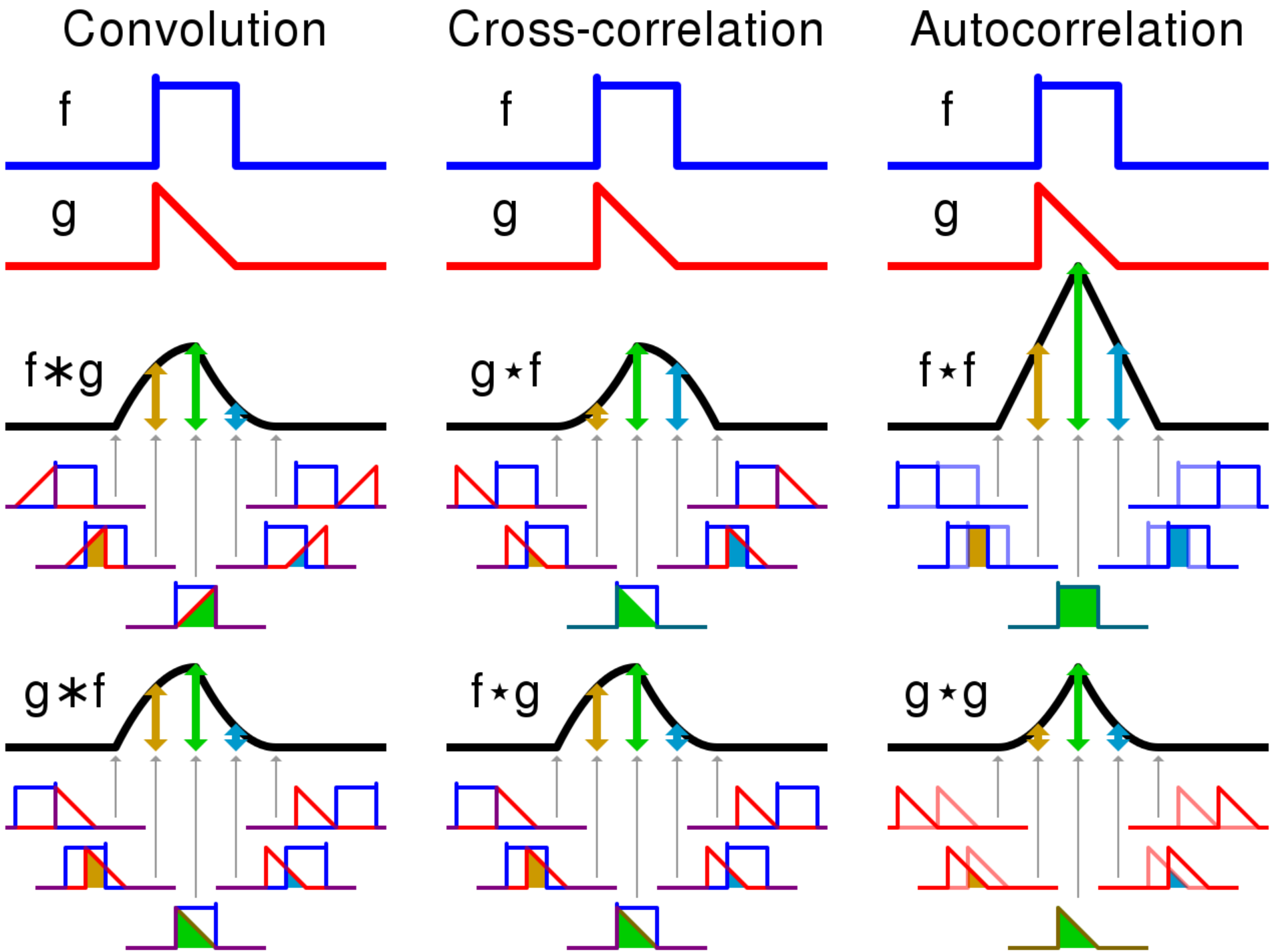
Object Detection involves localization of multiple objects (doesn't have to be the same class).

Object Segmentation involves the class label as well as an outline of the object in interest.

Convolution Theorem

- Fourier transform of a convolution of two signals is the pointwise product of their Fourier transforms



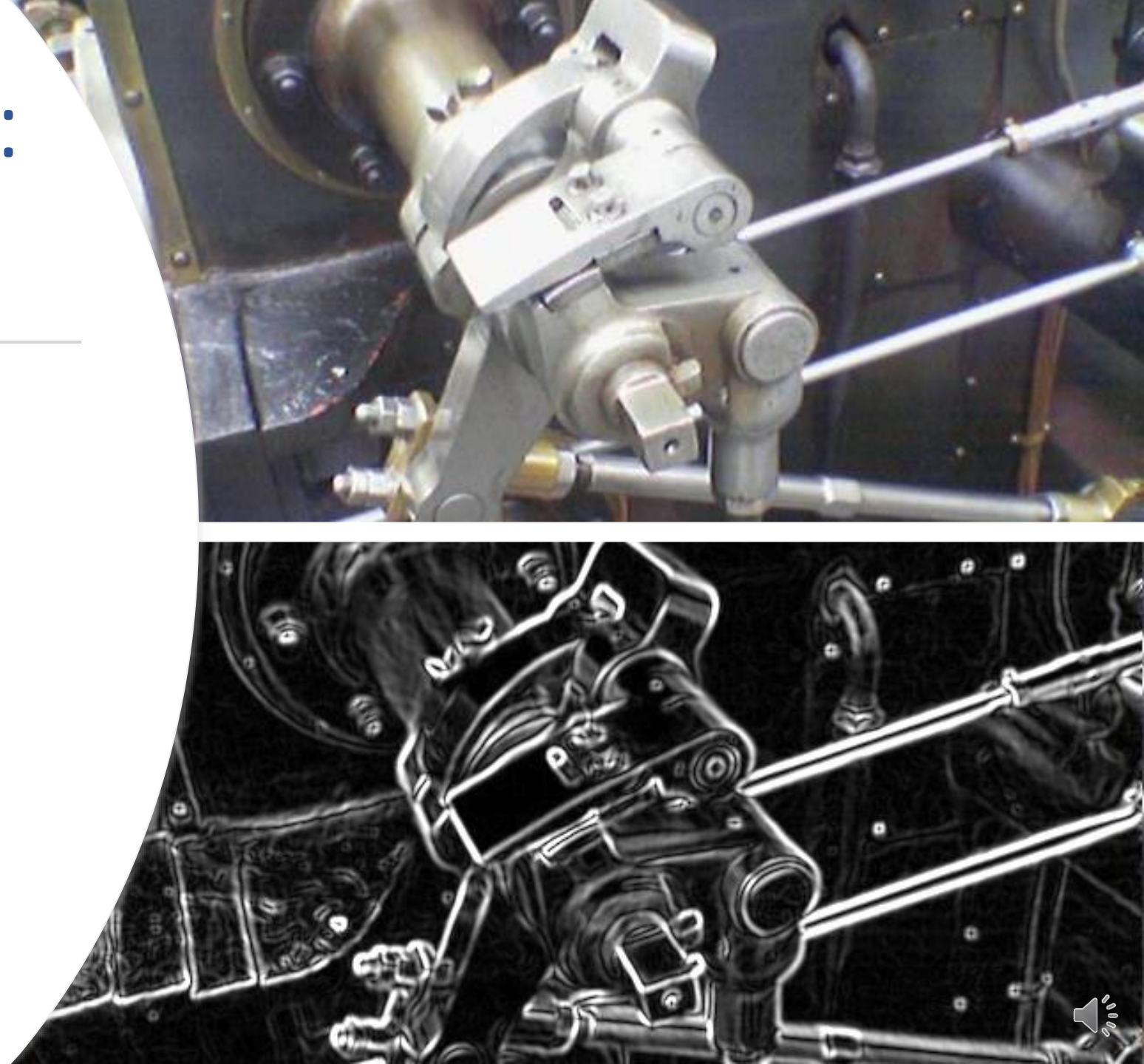

Convolution
Cross-correlation
Autocorrelation
f
g
f∗g
g⋆f
f⋆f
g∗f
f⋆g
g⋆g

2D Convolution: Sobel Filter

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A}$$

$$\mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

https://en.wikipedia.org/wiki/Sobel_operator



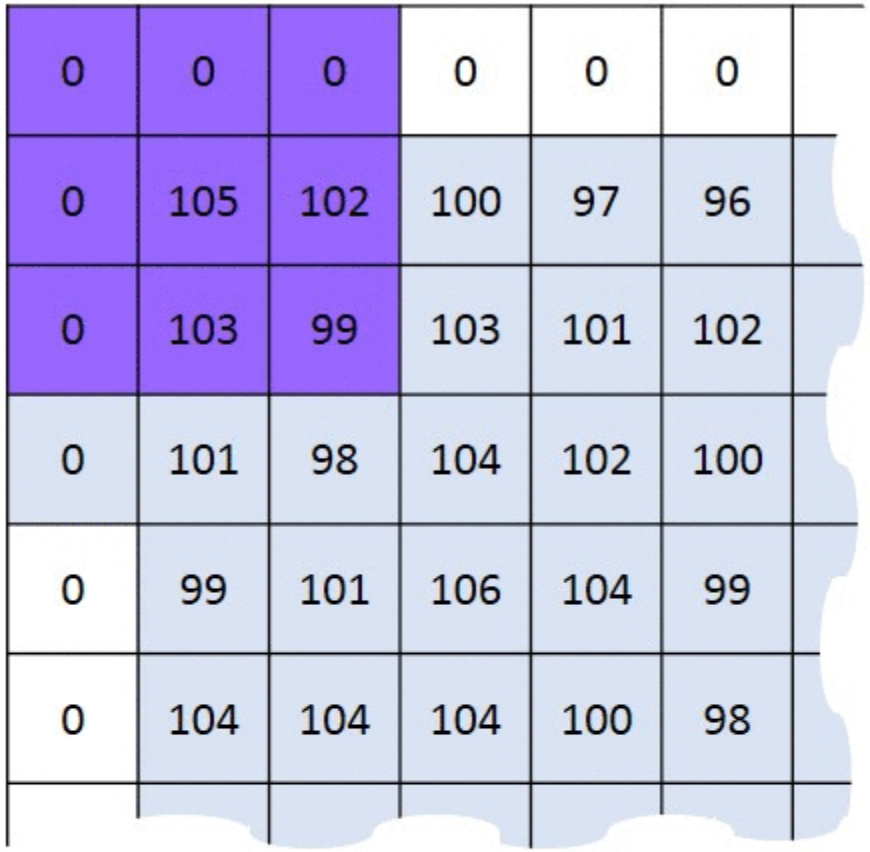
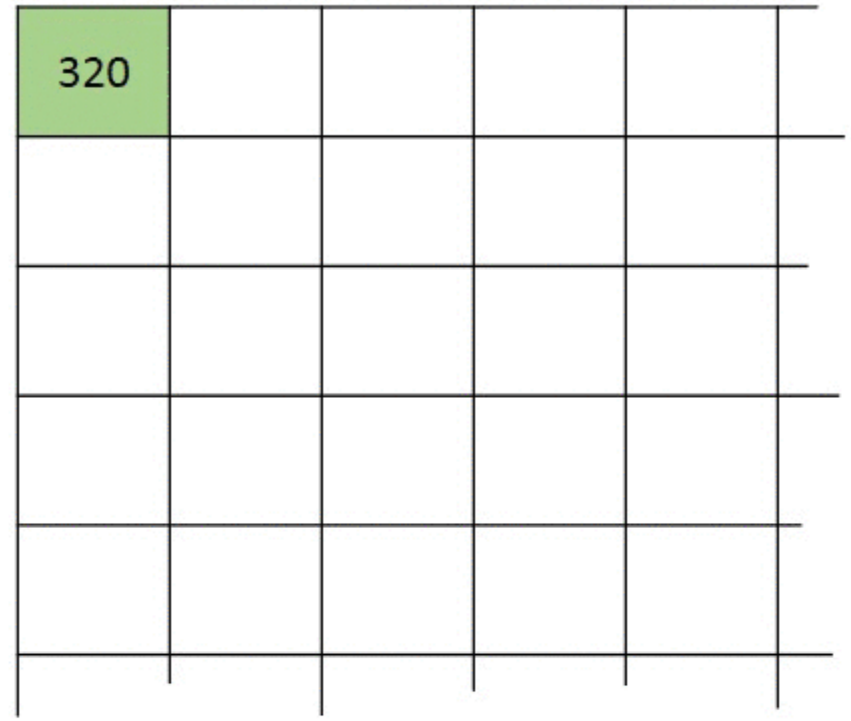


Image Matrix

Kernel Matrix

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

$$0*0+0*-1+0*0$$
$$+0*-1+105*5+102*-1$$
$$+0*0+103*-1+99*0 = 320$$



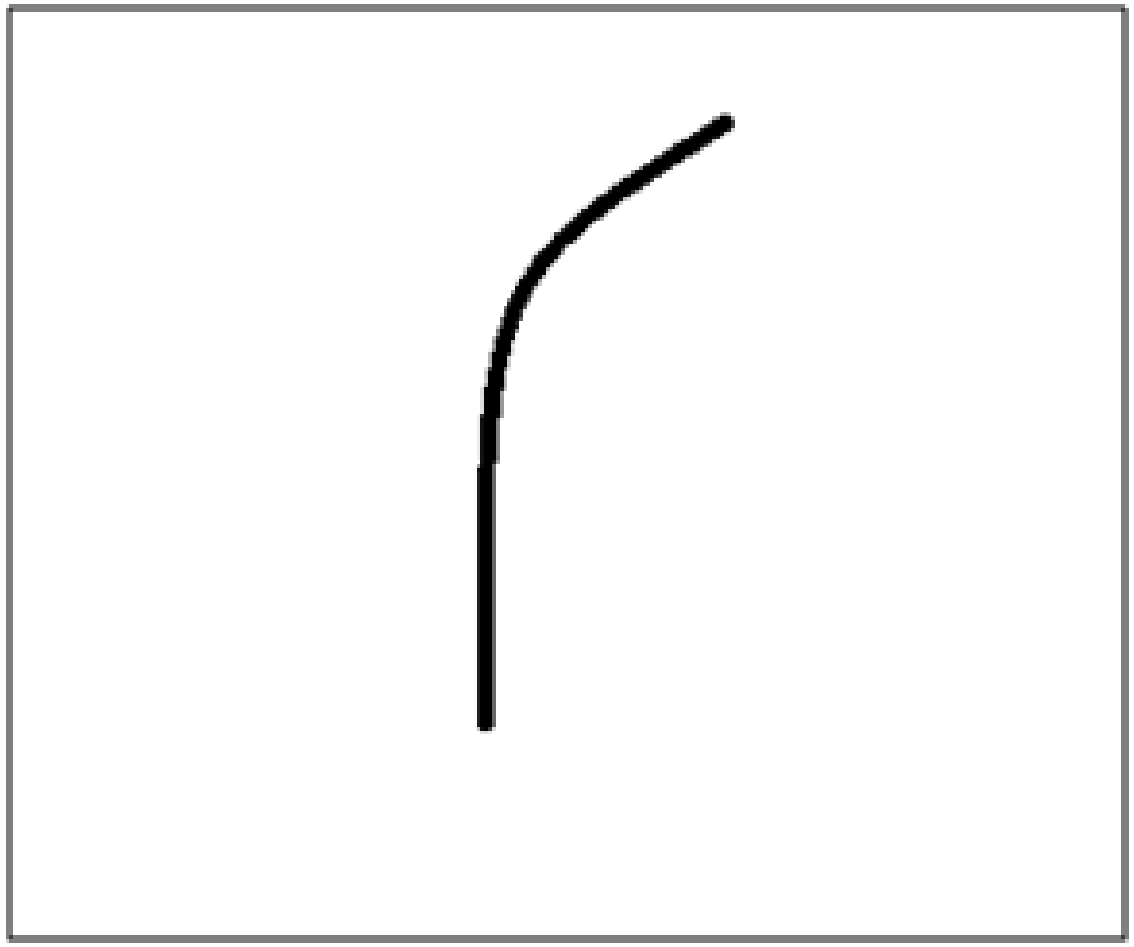
Output Matrix

Convolution with horizontal and vertical strides = 1

Example: A Curve Filter

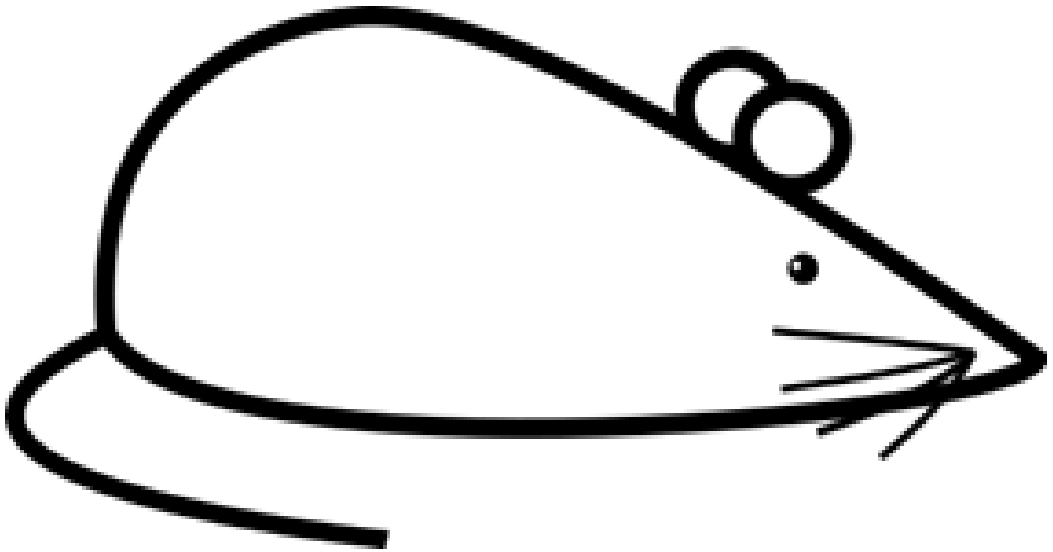
0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

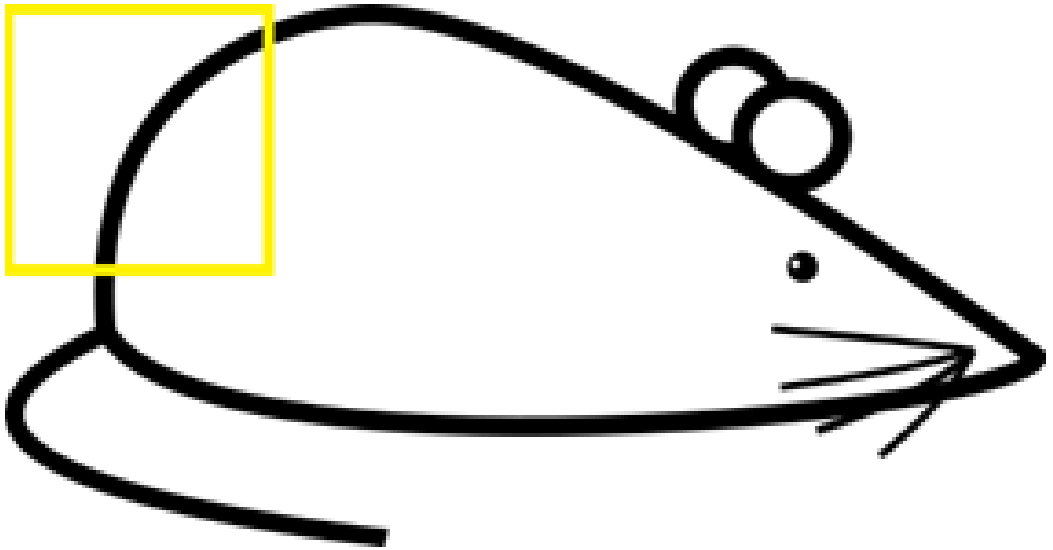


Visualization of a curve detector filter

Scan the Image to Detect an Edge

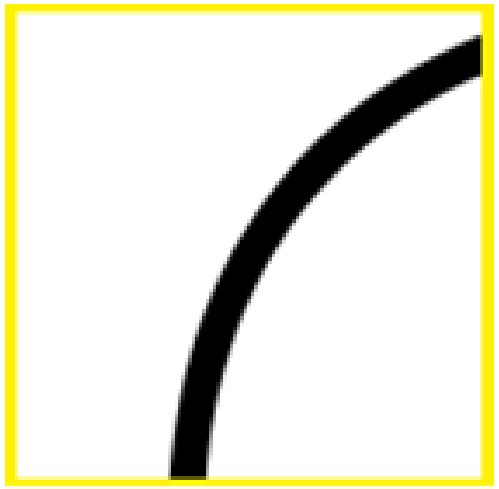


Original image



Visualization of the filter on the image

Edge Detected!



Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

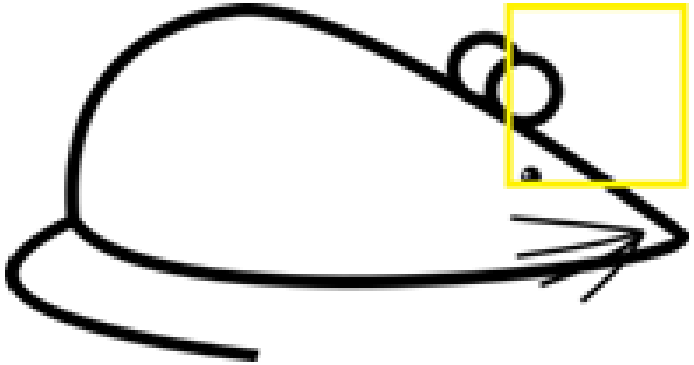
*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

Continue Scanning (No edge)



Visualization of the filter on the image

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

Pixel representation of receptive field

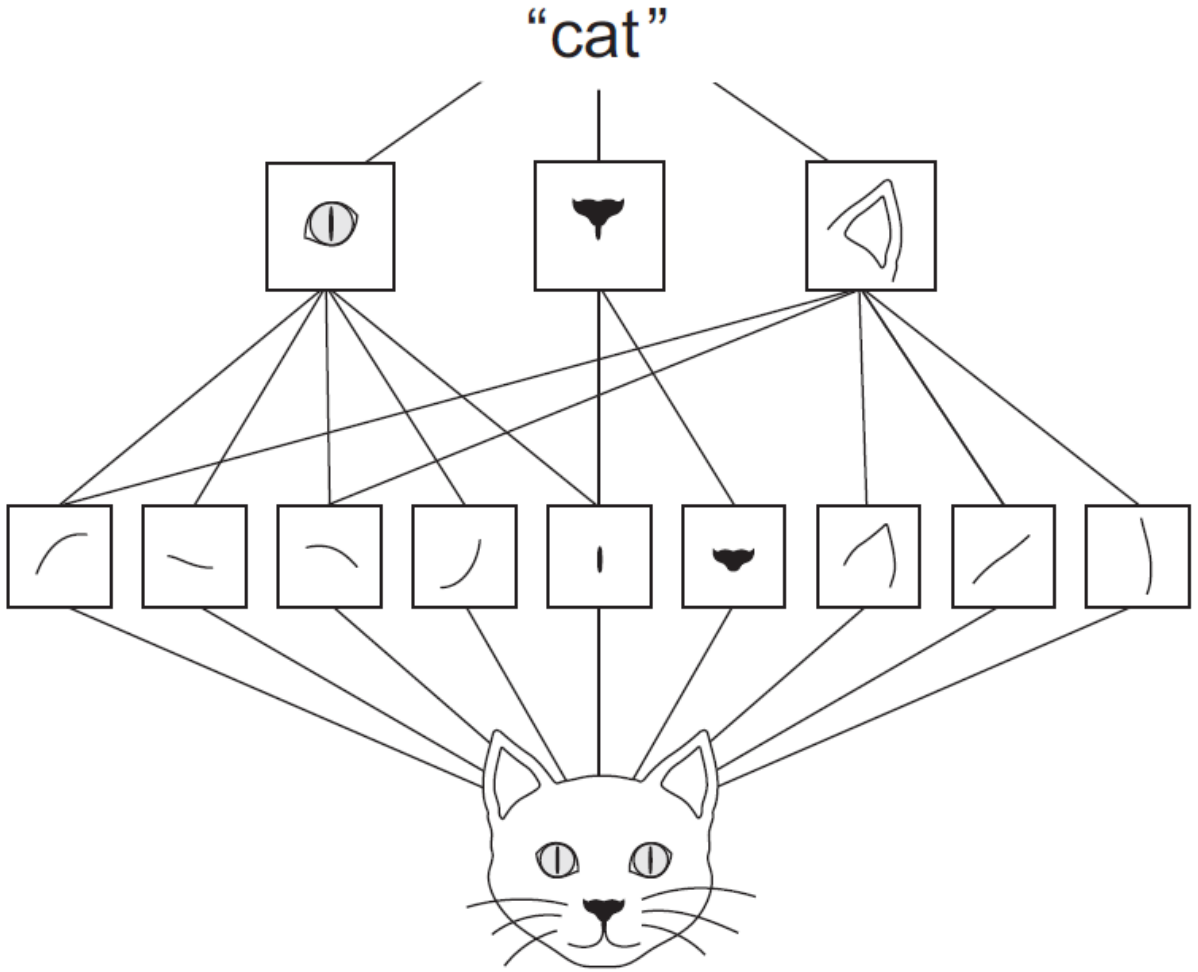
*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = 0

Spatial Hierarchy of Features



Create First ConvNet

- Create a CNN to classify MNIST digits

```
from keras import layers
from keras import models

model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
```

Model Summary

- model.summary()

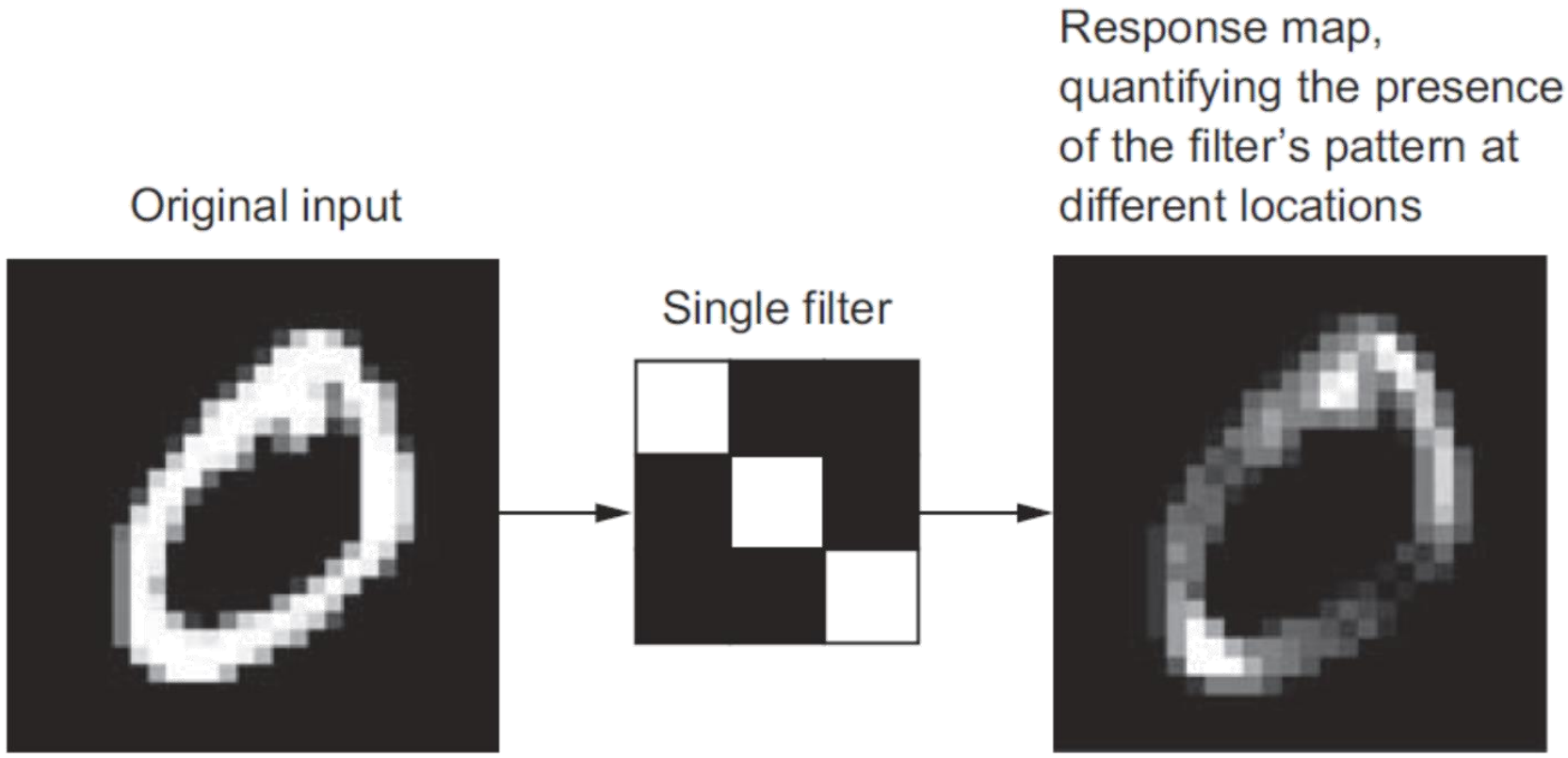
```
_________________________________________________________________
Layer (type) Output Shape Param #
=================================================================
conv2d_1 (Conv2D) (None, 26, 26, 32) 320
_________________________________________________________________
maxpooling2d_1 (MaxPooling2D) (None, 13, 13, 32) 0
_________________________________________________________________
conv2d_2 (Conv2D) (None, 11, 11, 64) 18496 = (3×3×32+1)×64
_________________________________________________________________
maxpooling2d_2 (MaxPooling2D) (None, 5, 5, 64) 0
_________________________________________________________________
conv2d_3 (Conv2D) (None, 3, 3, 64) 36928
=================================================================
```

Feature Map

- Outputs of a Convolution Layer is also called as Feature Map

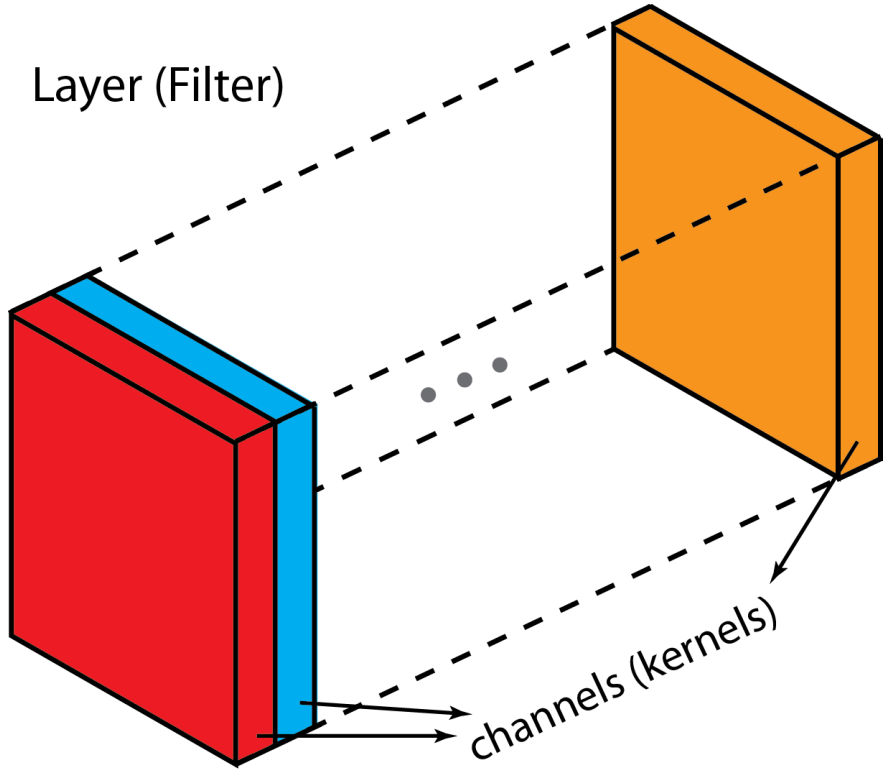
```
=>layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1))
```

  - Receive a 28x28 input image and computes 32 filters over it
  - Each filter has size 3x3

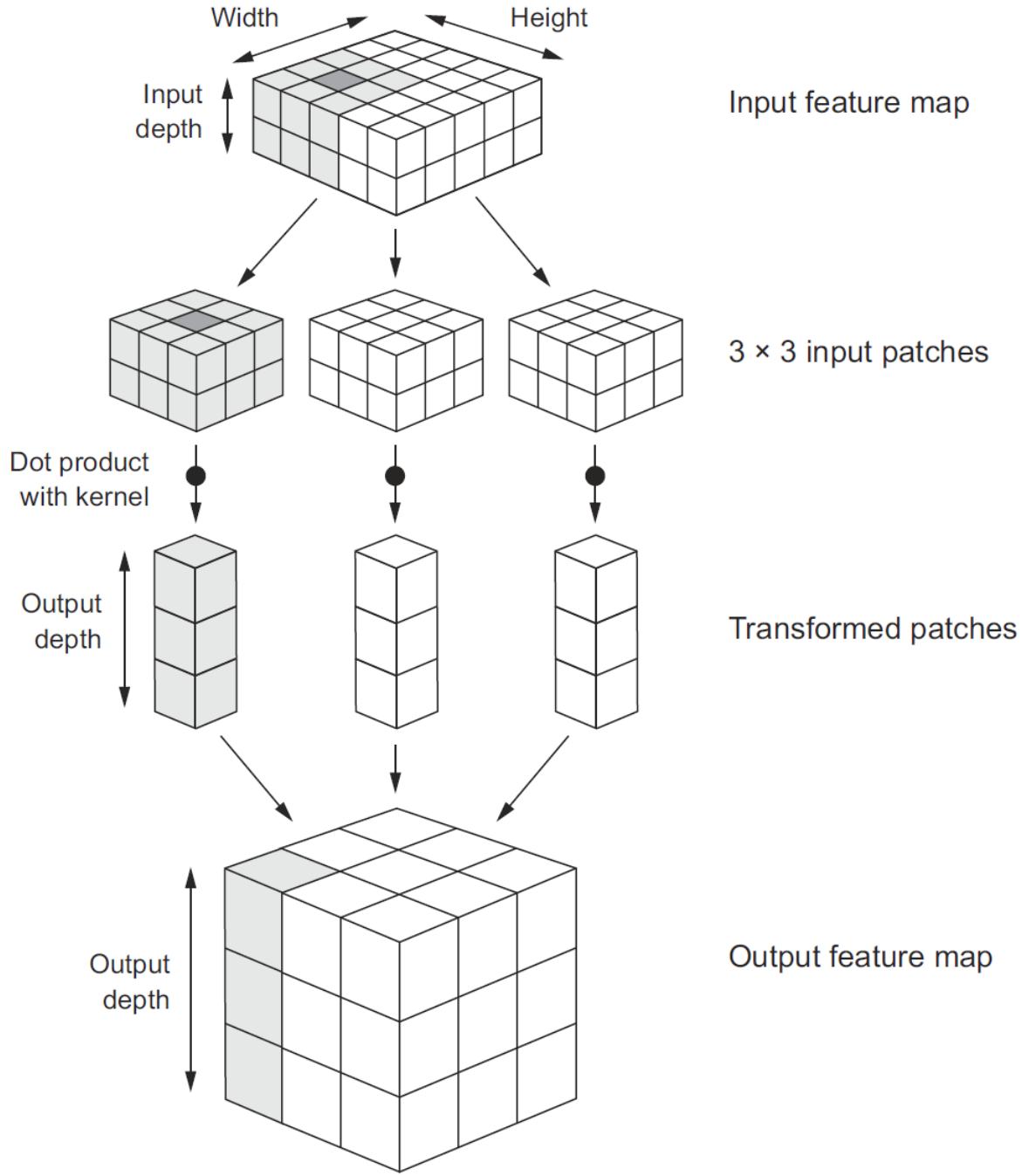


Kernel and Filter in Deep Learning

- “Kernel” refers to a 2D array of weights.
- “filter” is for 3D structures of multiple kernels stacked together.



https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215


Width
Height
Input
depth
Input feature map
3 × 3 input patches
Dot product
with kernel
Output
depth
Transformed patches
Output
depth
Output feature map

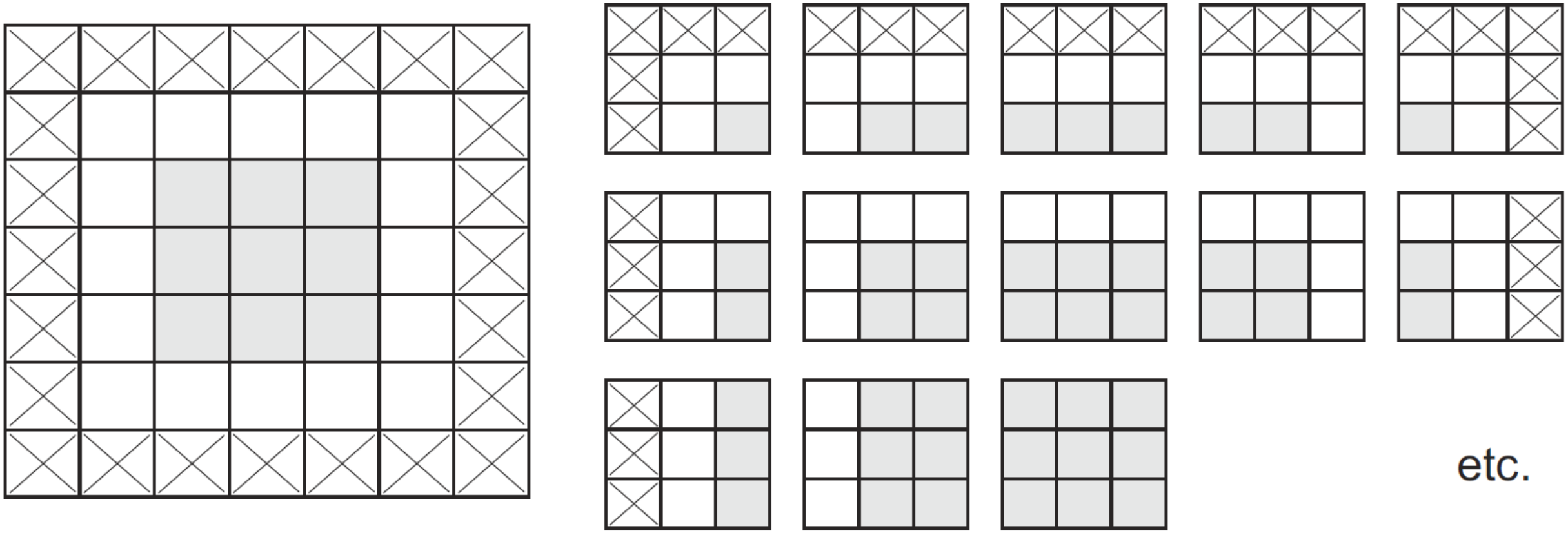
Add a Classifier on Top of ConvNet

```
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
```

```
Layer (type) Output Shape Param #
=================================================================
conv2d_1 (Conv2D) (None, 26, 26, 32) 320
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 32) 0
_________________________________________________________________
conv2d_2 (Conv2D) (None, 11, 11, 64) 18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 5, 5, 64) 0
_________________________________________________________________
conv2d_3 (Conv2D) (None, 3, 3, 64) 36928
_________________________________________________________________
flatten_1 (Flatten) (None, 576) 0
_________________________________________________________________
dense_1 (Dense) (None, 64) 36928
_________________________________________________________________
dense_2 (Dense) (None, 10) 650
=================================================================
Total params: 93,322 Trainable params: 93,322 Non-trainable params: 0
```

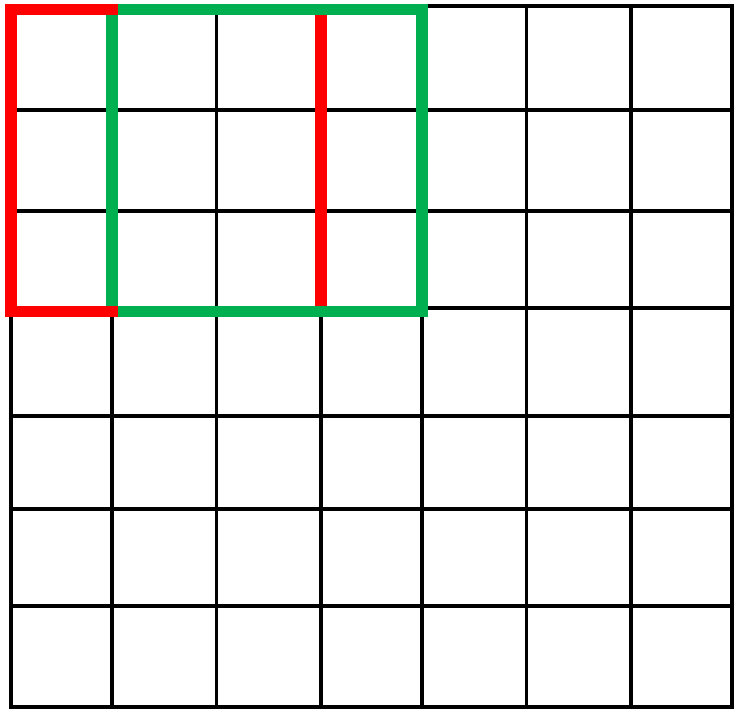
Padding

- Padding a 5x5 input to extract 25 3x3 patches

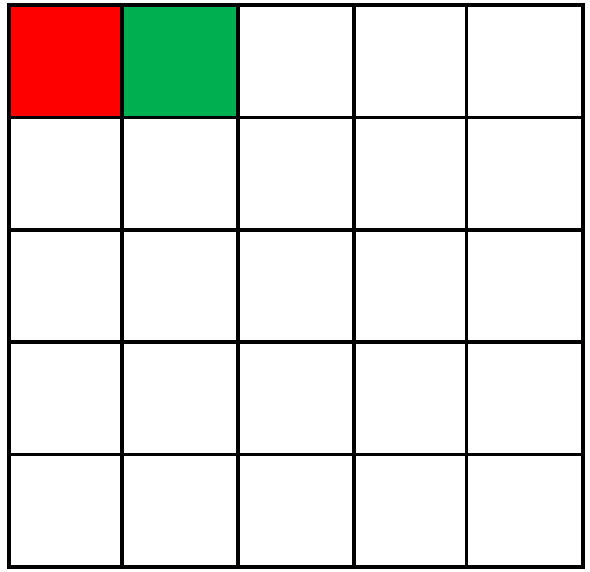


Stride=1

7 x 7 Input Volume

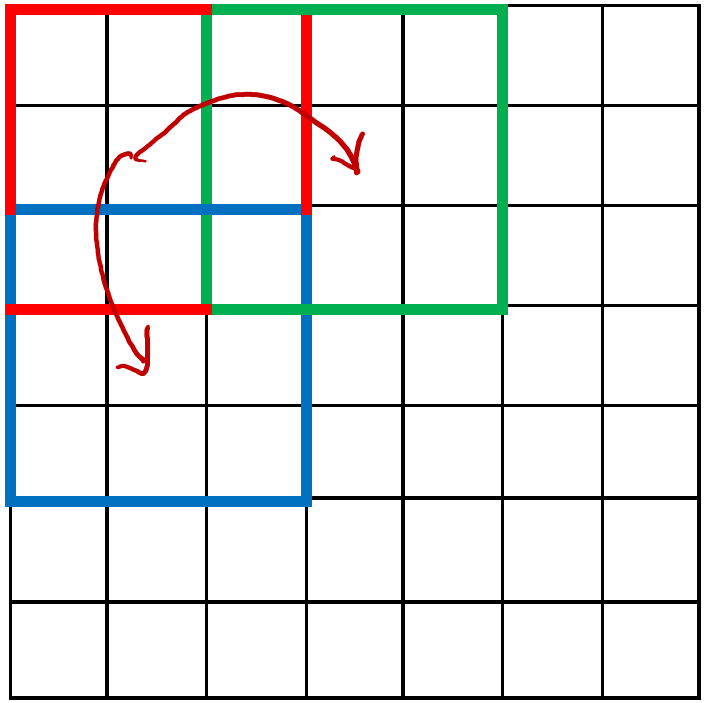


5 x 5 Output Volume

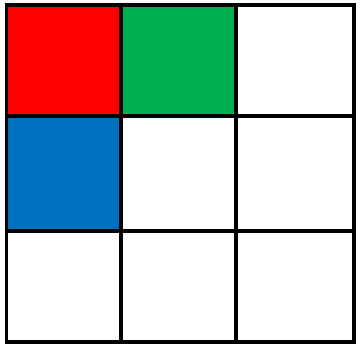


Stride=2

7 x 7 Input Volume

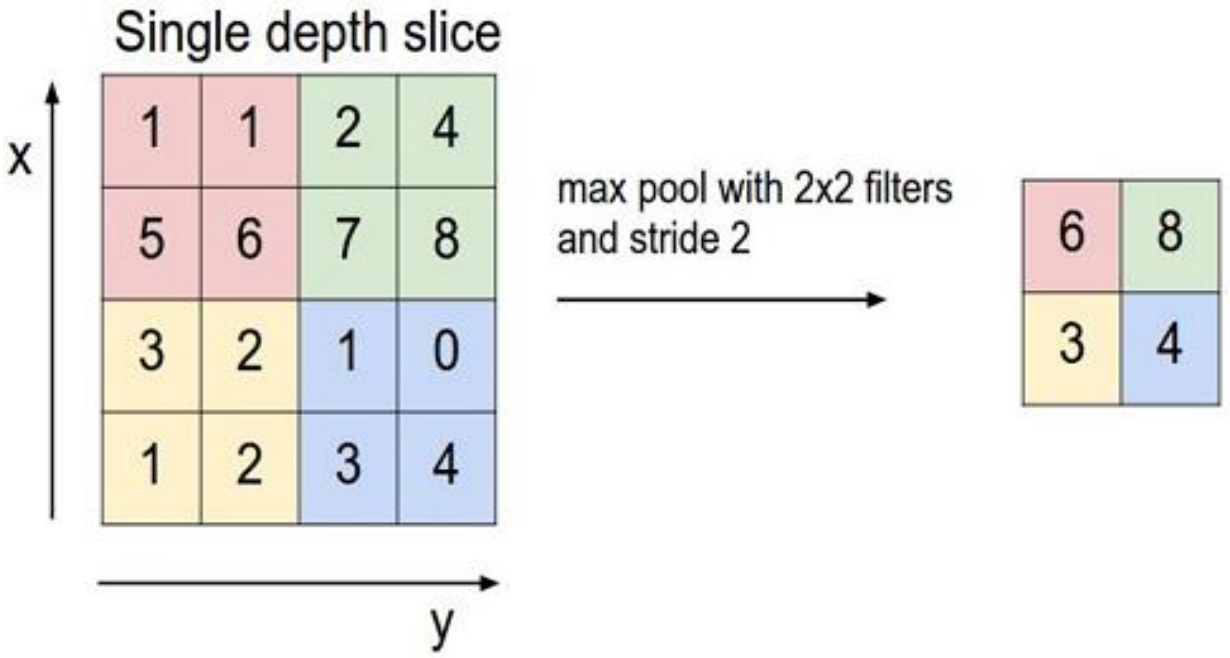


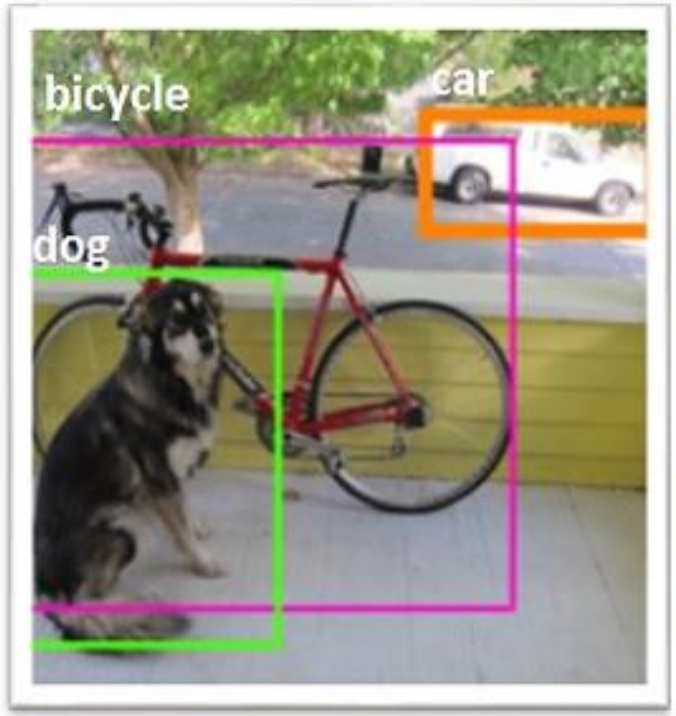
3 x 3 Output Volume

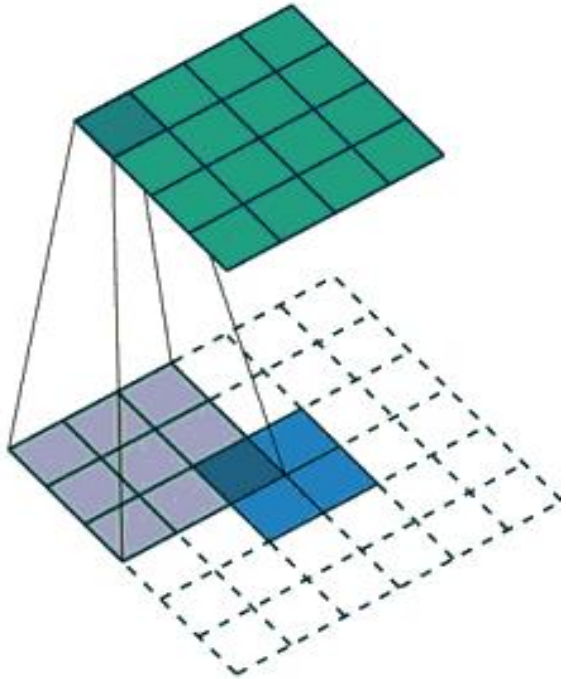


Max Pooling

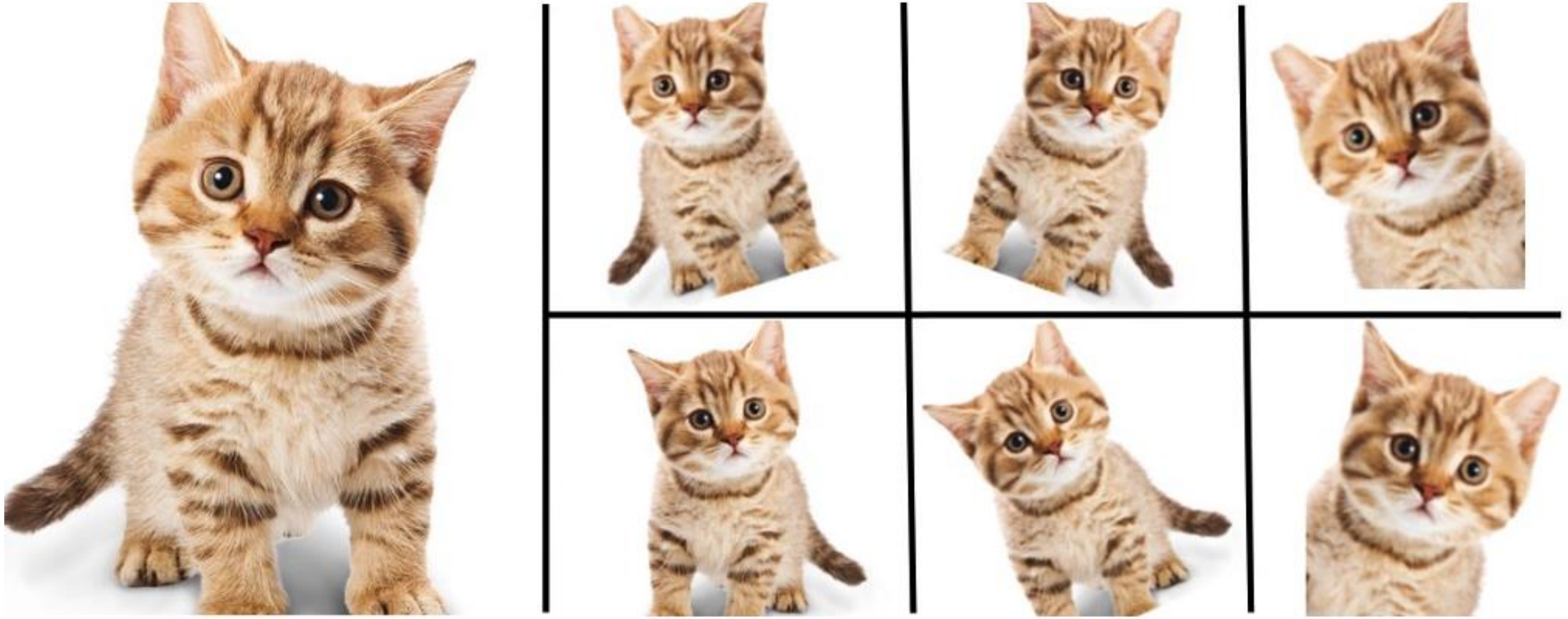
- Downsampling an image
- Better than average pooling and strides





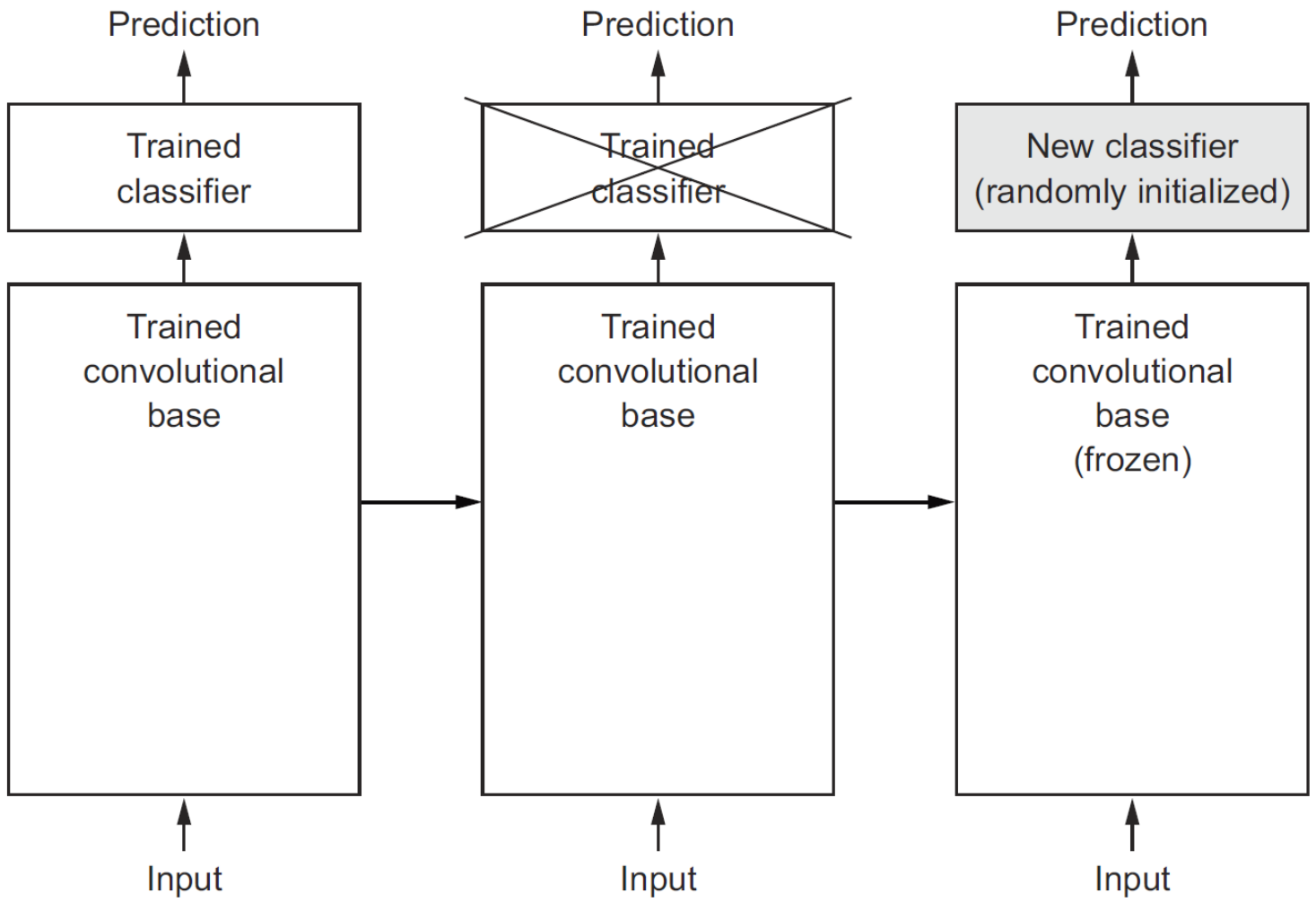


Data Augmentation



Using Pre-trained Models

- Xception
- VGG16
- VGG19
- ResNet, ResNetV2, ResNeXt
- InceptionV3
- InceptionResNetV2
- MobileNet
- MobileNetV2
- DenseNet
- NASNet



Summary

- Convnets are the best for Computer Vision (but may be replaced by attention modules)
- Data augmentation is a powerful way to fight overfitting
- We can use pre-trained model for feature extraction
- We can further improve the pre-trained model on our dataset by fine-tuning