AI VIETNAM
AI Insight Course

Naïve Object Detection

Quang-Vinh Dinh
Ph.D. in Computer Science

Year 2020

Hệ số tương quan (correlation coefficient)

Công thức: Gọi x,y là hai biến ngẫu nhiên

$$\rho_{xy} = \frac{E[(x-\mu_x)[(y-\mu_y)]}{\sqrt{var(x)}\sqrt{var(y)}}$$

$$= \frac{n(\sum_i x_i y_i) - (\sum_i x_i)(\sum_i y_i)}{\sqrt{n\sum_i x_i^2 - (\sum_i x_i)^2}\sqrt{n\sum_i y_i^2 - (\sum_i y_i)^2}}$$

Tính chất 1

$$-1 \leq \rho_{xy} \leq 1$$

Tương quan nghịch ← | → **Tương quan thuận**

Tính chất 2

$$\rho_{xy} = \rho_{uv}$$

trong đó

$$u = ax + b$$

$$v = cy + d$$

Ví dụ 1

x = [7, 18, 29, 2, 10, 9, 9]

y = [1, 6, 12, 8, 6, 21, 10]

$$\rho_{xy} = \frac{E[(x-\mu_x)[(y-\mu_y)]}{\sqrt{var(x)}\sqrt{var(y)}}$$

$$= \frac{n*818 - 84*64}{\sqrt{n*1480 - 7056}\sqrt{n*822 - 4096}} = 0.149$$

Ví dụ 2

u=2*x-14 = [0, 22, 44, -10, 6, 4, 4]

v=y+2 = [3, 8, 14, 10, 8, 23, 12]

$$\rho_{uv} = \frac{E[(u-\mu_u)[(v-\mu_v)]}{\sqrt{var(u)}\sqrt{var(v)}}$$

$$= \frac{n*880 - 70*78}{\sqrt{n*2588 - 4900}\sqrt{n*1106 - 6084}} = 0.149$$

Ứng dụng cho patch matching

P_1 P_2 P_3 P_4

$\rho_{P_1P_2} = 0.55$

$\rho_{P_1P_3} = 0.23$ ➧ Ảnh P_2 giống với ảnh P_1 hơn so với P_3 và P_4

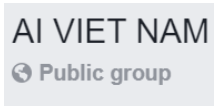
$\rho_{P_1P_4} = 0.30$

P_1 $P_2 = P_1+50$ $P_3 = 1.2P_1+10$

$\rho_{P_1P_2} = 0.9970$

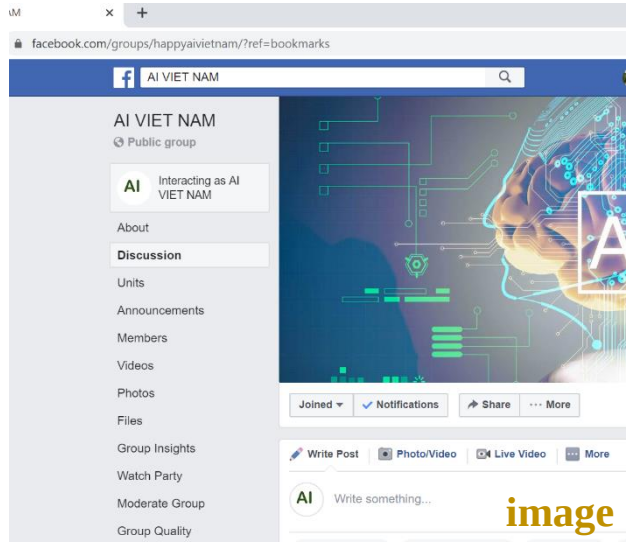
$\rho_{P_1P_3} = 0.9979$ ➧ ρ hoạt động tốt dưới sự thay đổi tuyến tính

Ứng dụng vào template matching



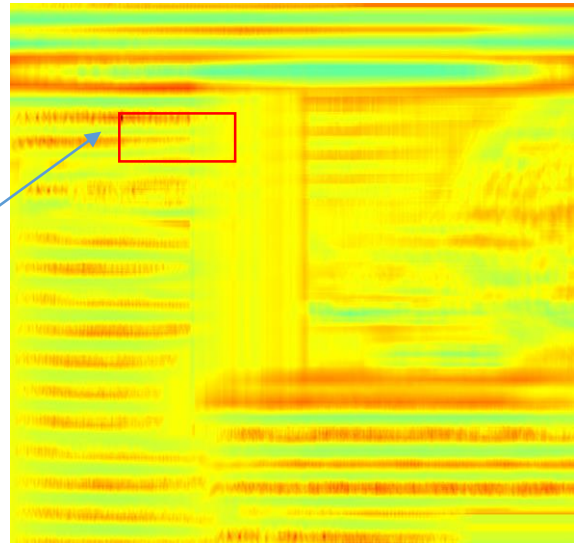
template

Tìm template có trong hình image

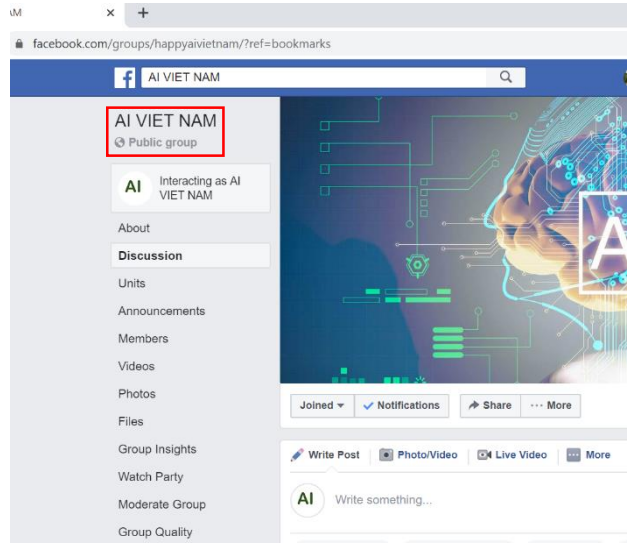


image

max



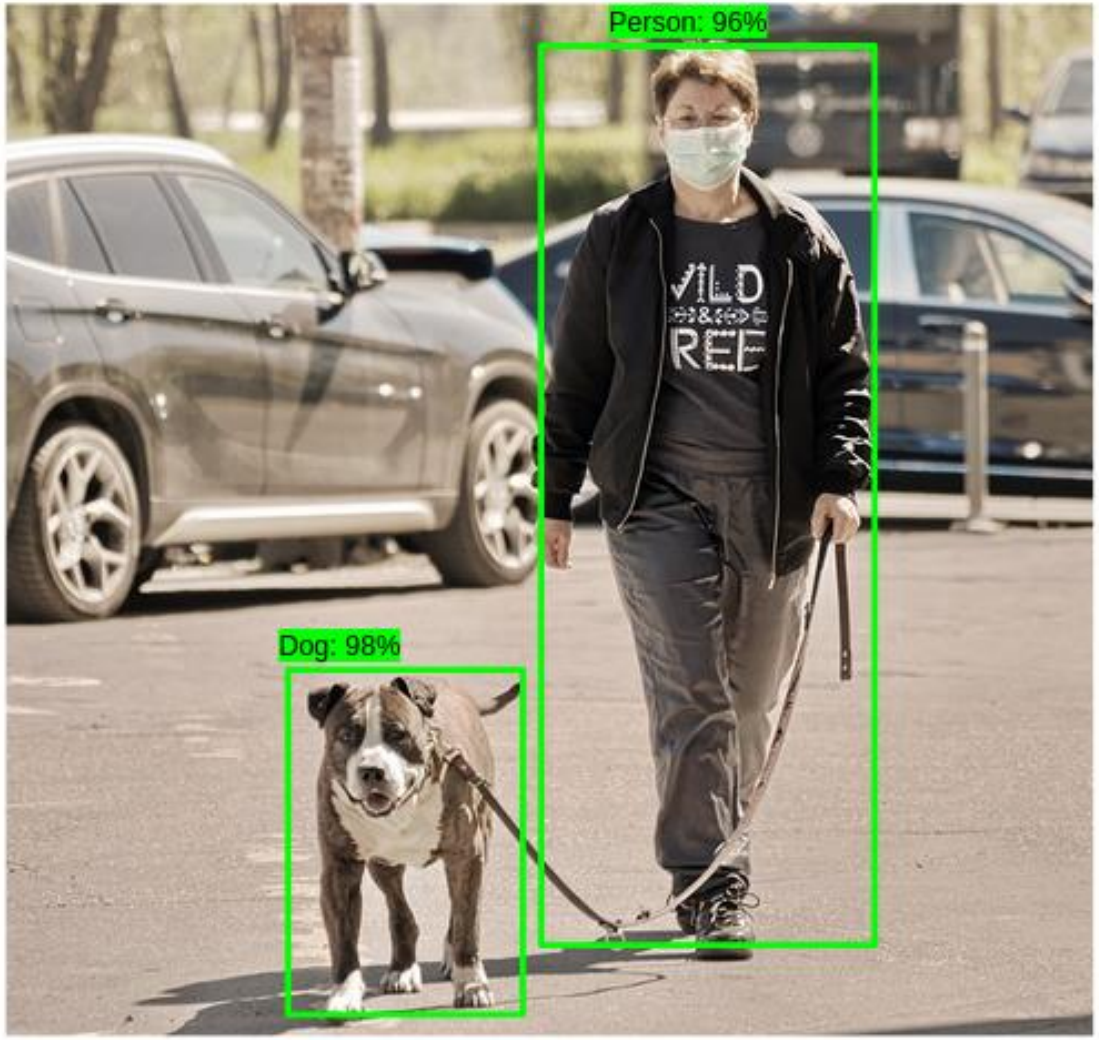
Map ρ cho từng pixel trong ảnh image



Kết quả

Naïve Object Detection

❖ Idea

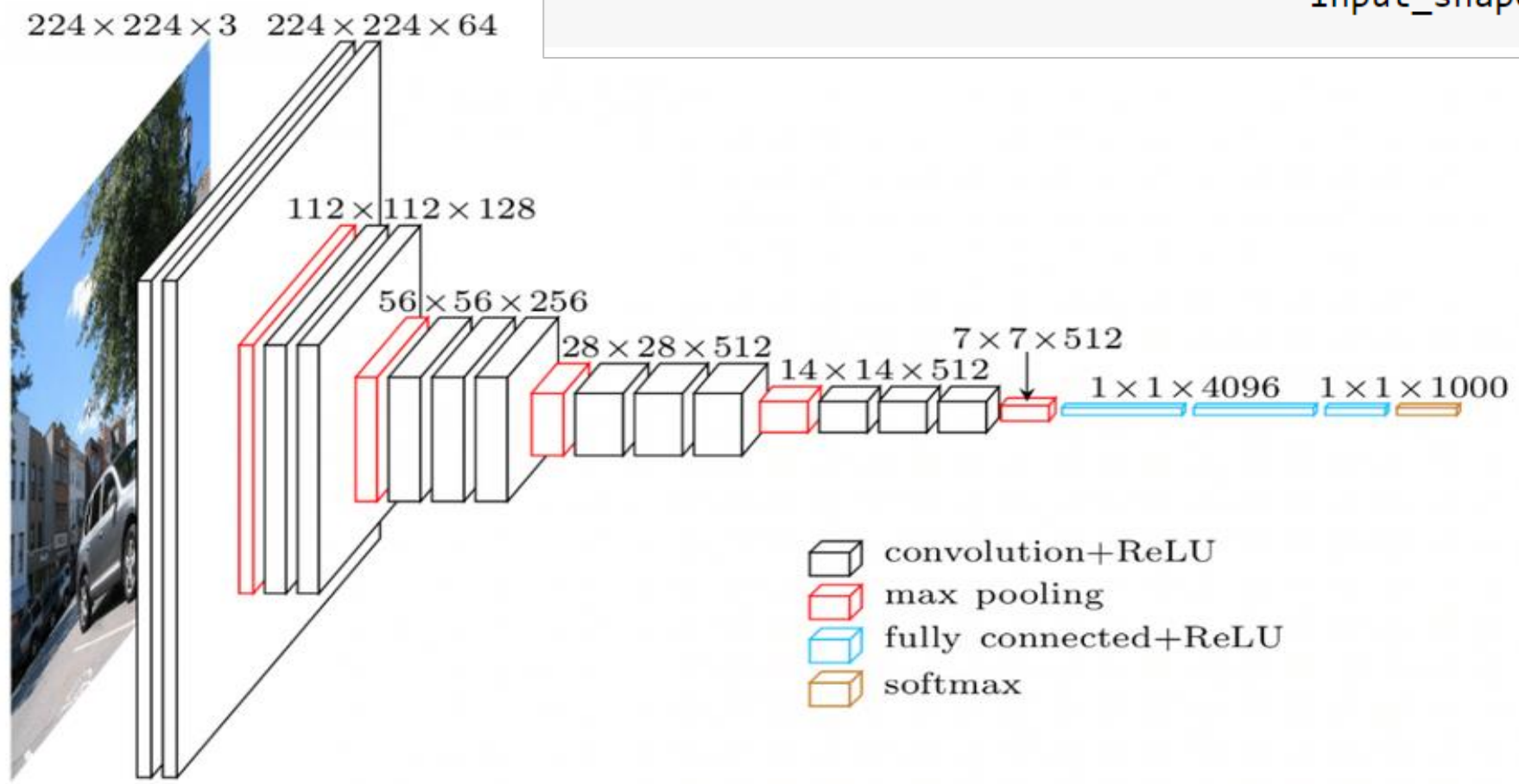


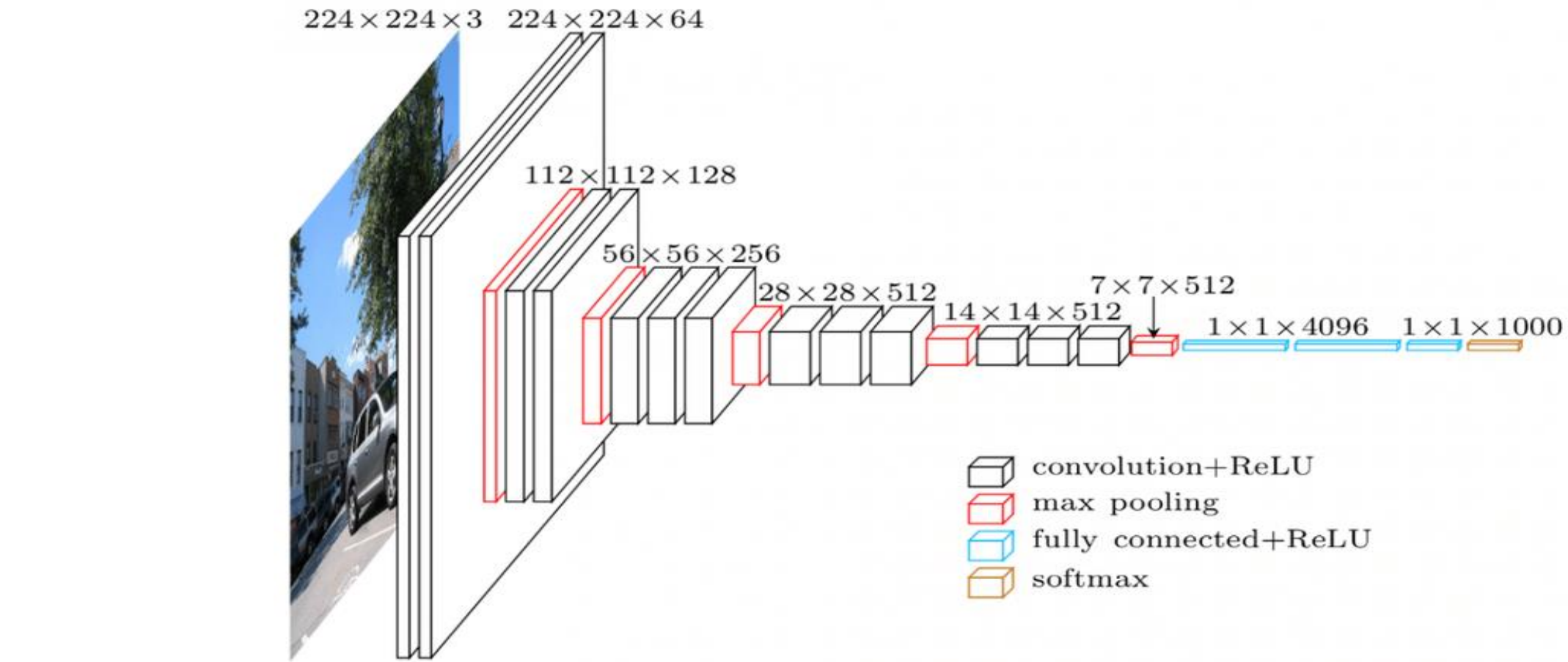
https://www.analyticsvidhya.com/blog/2020/08/selecting-the-right-bounding-box-using-non-max-suppression-with-implementation/

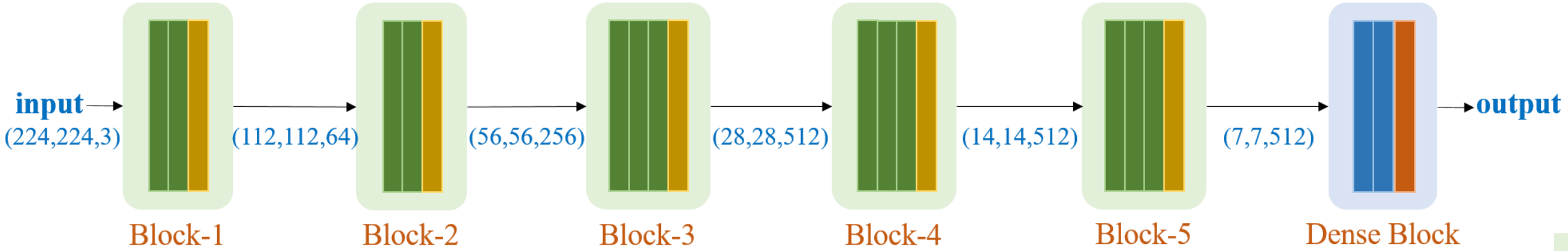
Naïve Object Detection

❖ Get the VGG16 model

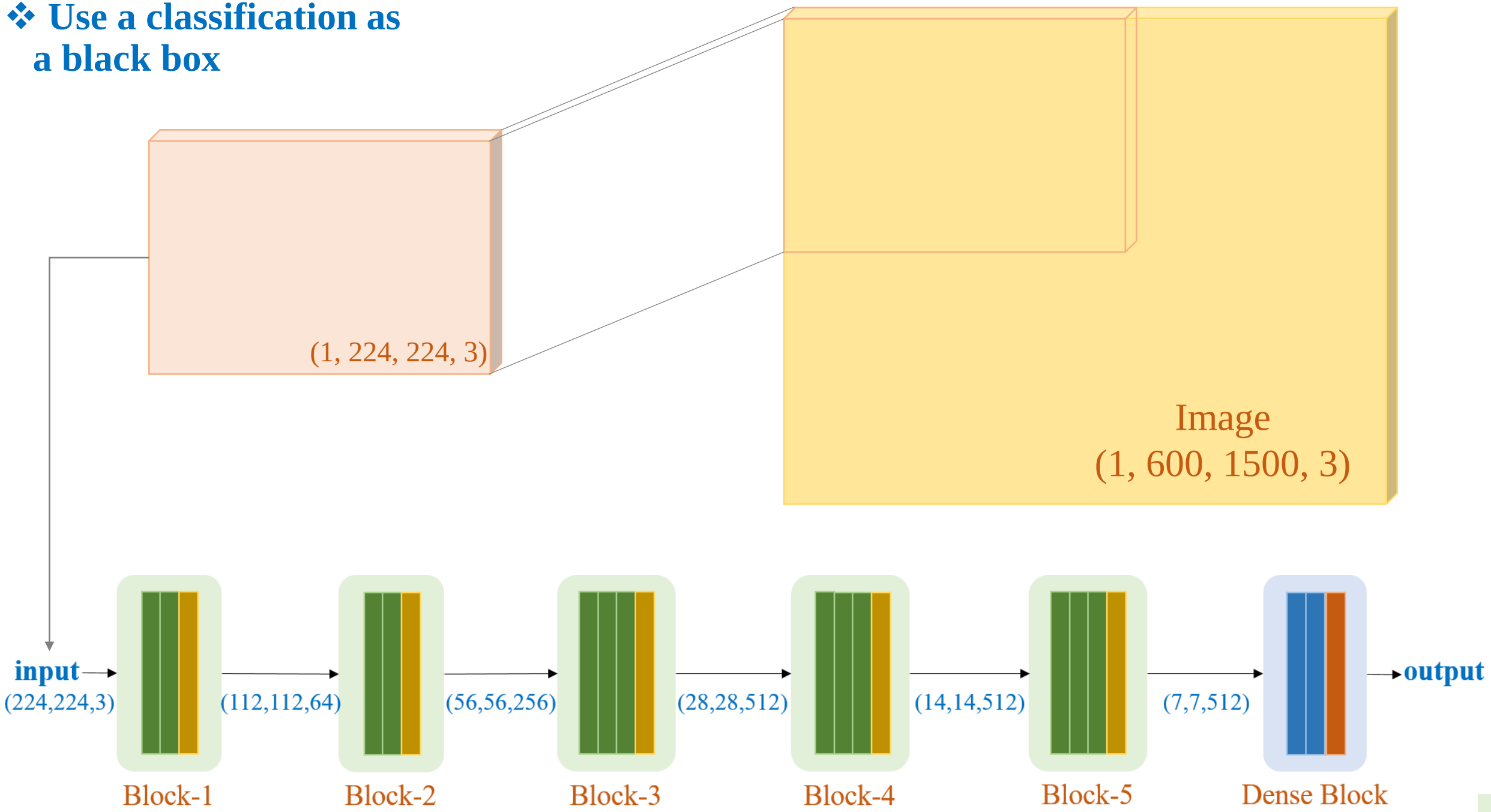
```
# get model
model = tf.keras.applications.VGG16(include_top=True,
                                    weights='imagenet',
                                    input_shape=(224, 224, 3))
```



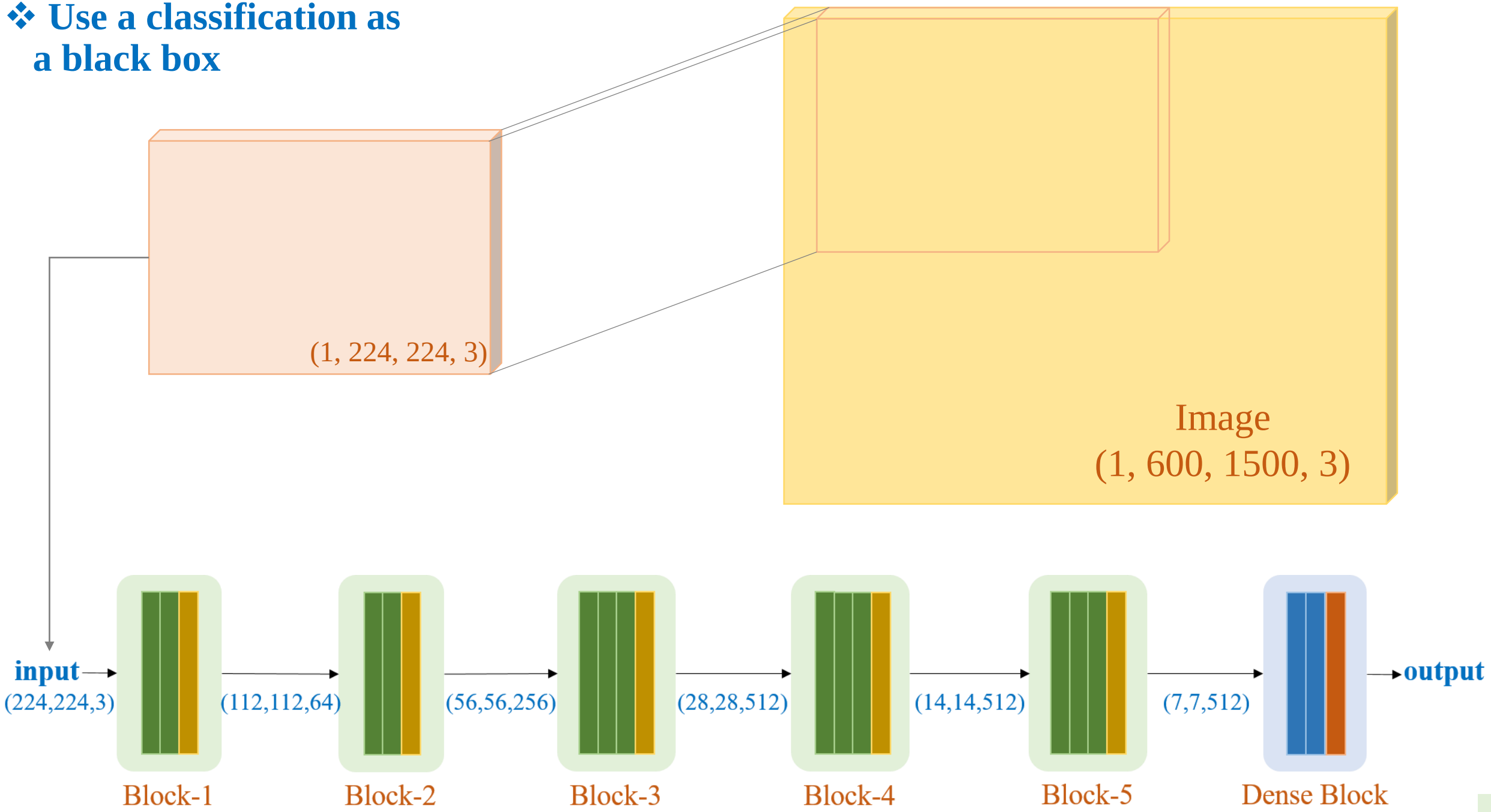

224 × 224 × 3
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
7 × 7 × 512
1 × 1 × 4096
1 × 1 × 1000
convolution+ReLU
max pooling
fully connected+ReLU
softmax


input
(224,224,3)
(112,112,64)
(56,56,256)
(28,28,512)
(14,14,512)
(7,7,512)
output
Block-1
Block-2
Block-3
Block-4
Block-5
Dense Block

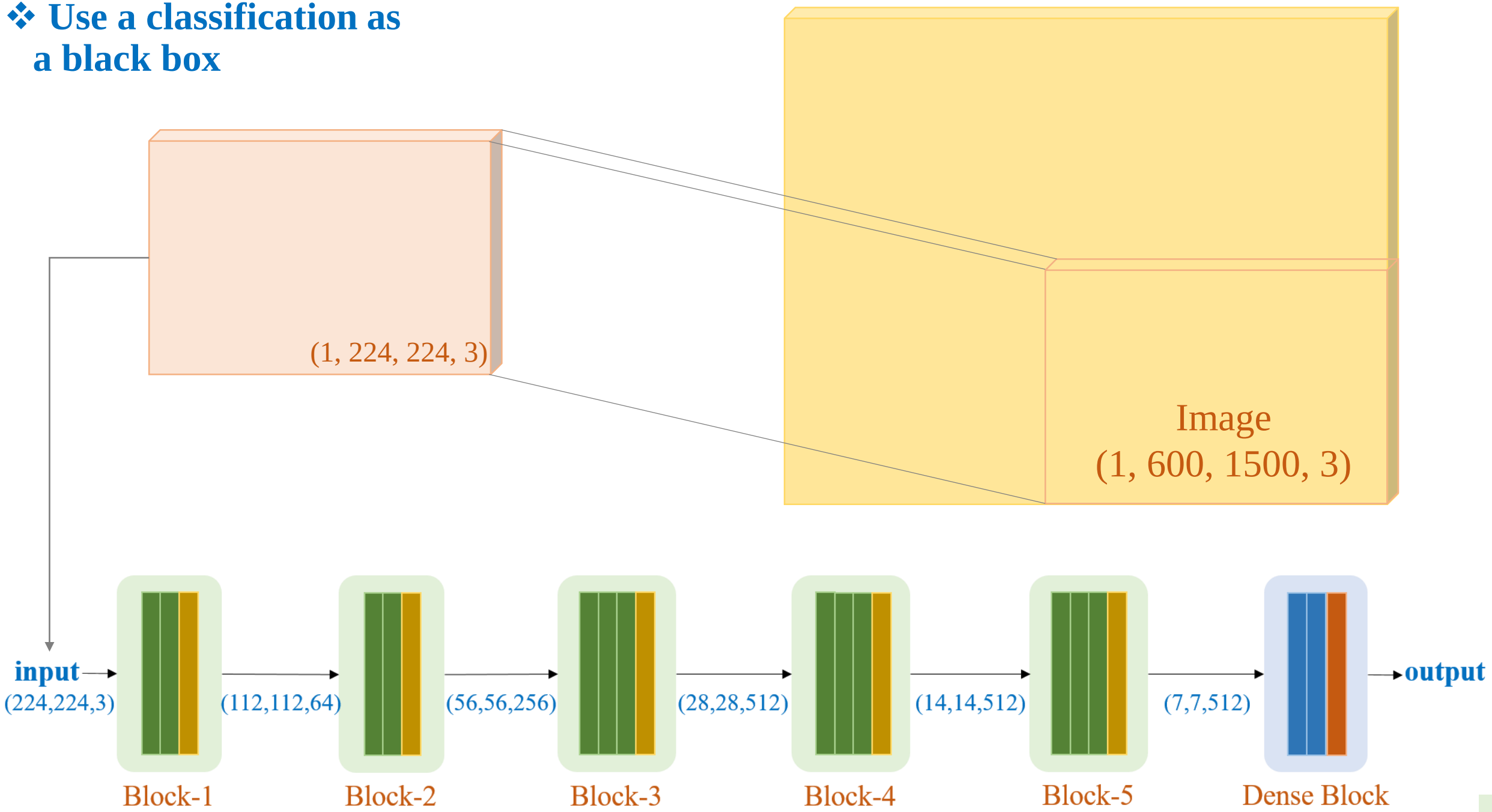
❖ Use a classification as a black box



❖ Use a classification as a black box

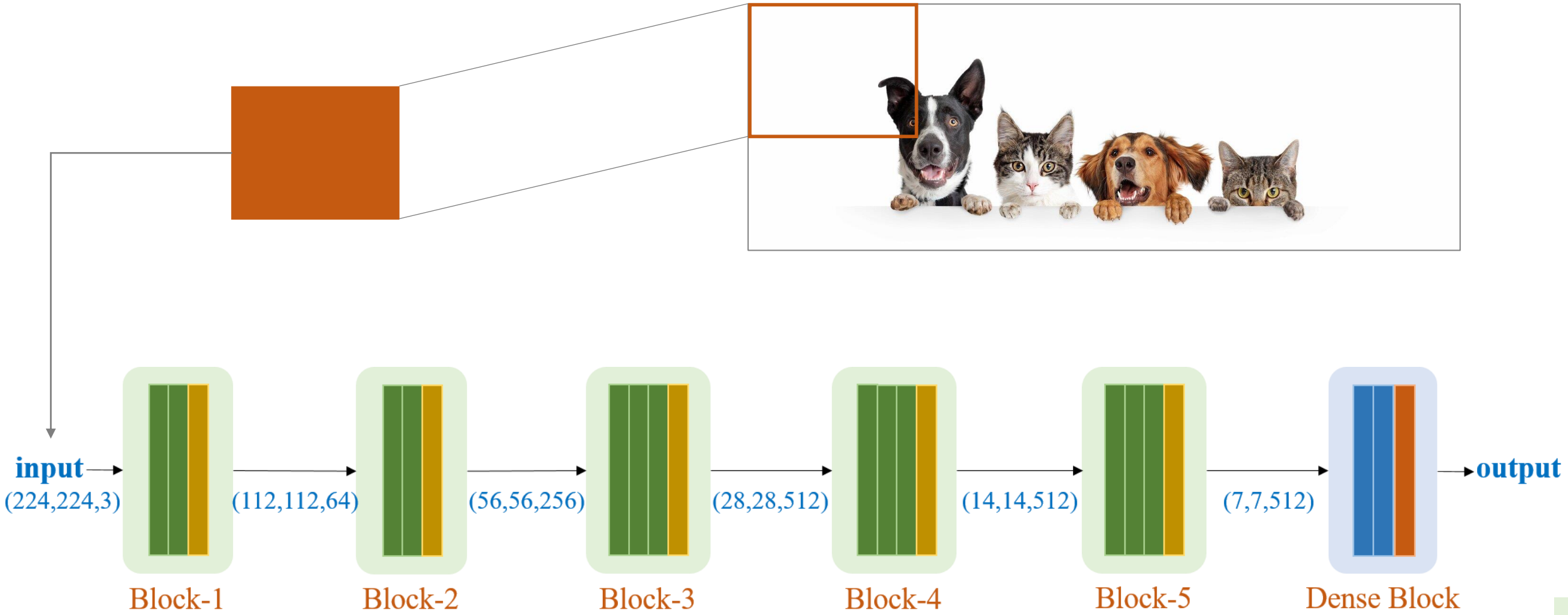


❖ Use a classification as a black box



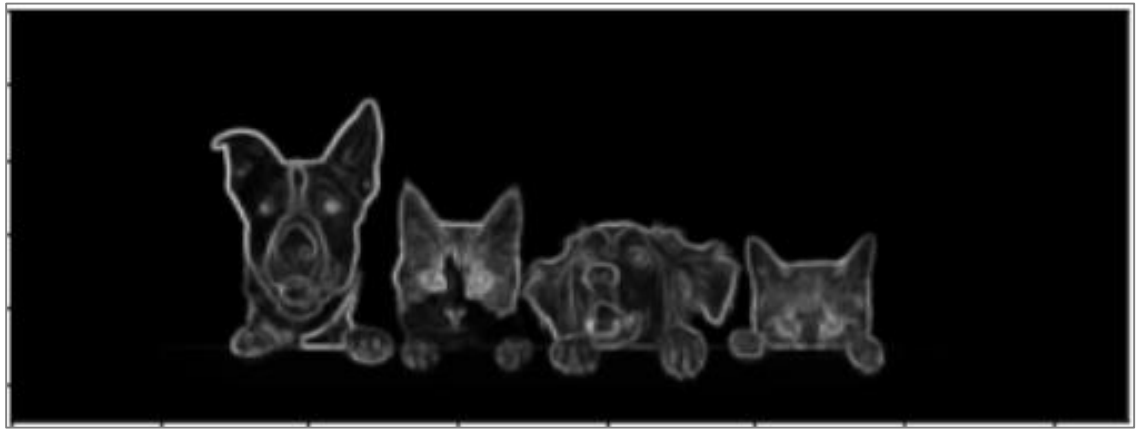
Naïve Object Detection

❖ Use a classification as a black box



Naïve Object Detection

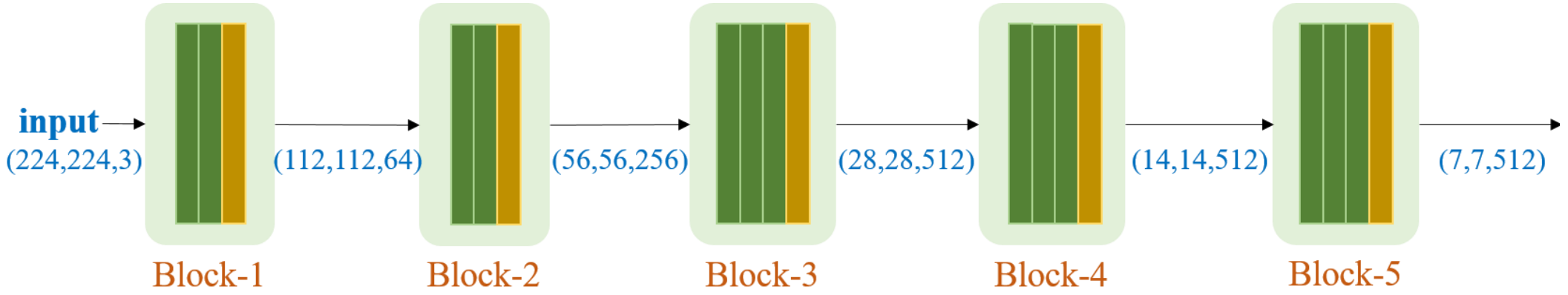
❖ **Use a classification as a black box**

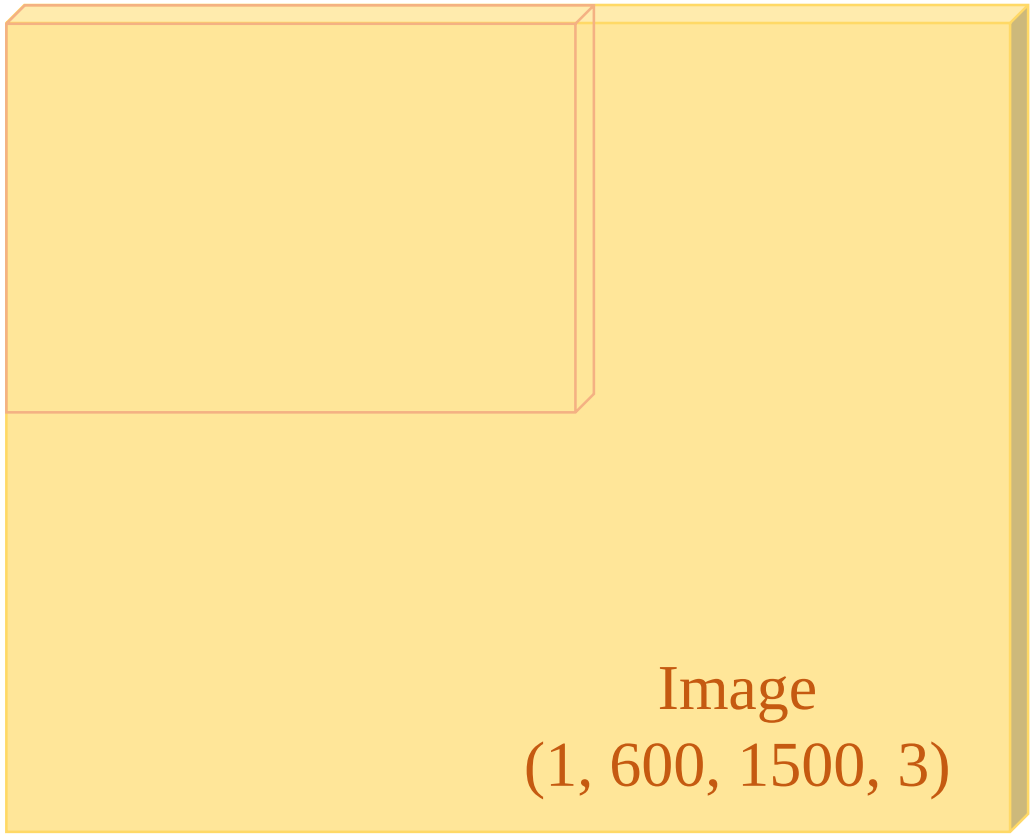


```python
import numpy as np
import cv2
import math
from scipy.ndimage.filters import generic_filter

img  = cv2.imread(PATH+'image7.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
gray = cv2.resize(gray, (1500, 600))
gray = gray.astype('float')
gray_std = generic_filter(gray, np.std, size=7)
```

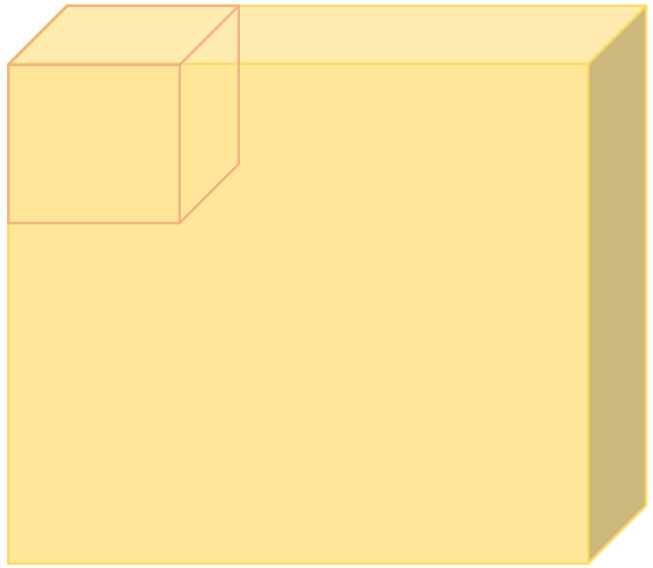
```python
# compute predictions - 30mins
prediction_data = []
for i in range(height-model_height+1):
  for j in range(width-model_width+1):
    flag = gray_std[i:i+model_height, j:j+model_width]
    flag = np.sum(flag) / (model_height*model_width*1.0)
    if (flag > 45.0):
      patch = image_np[:, i:i+model_height, j:j+model_width, :]
      patch = preprocess_input(patch)
      patch = model.predict(patch)
      prediction_data.append( (tf.math.reduce_max(patch[0]).numpy(),
                               i, j, tf.math.argmax(patch[0]).numpy()) )
```

❖ Method 2: Open the black box





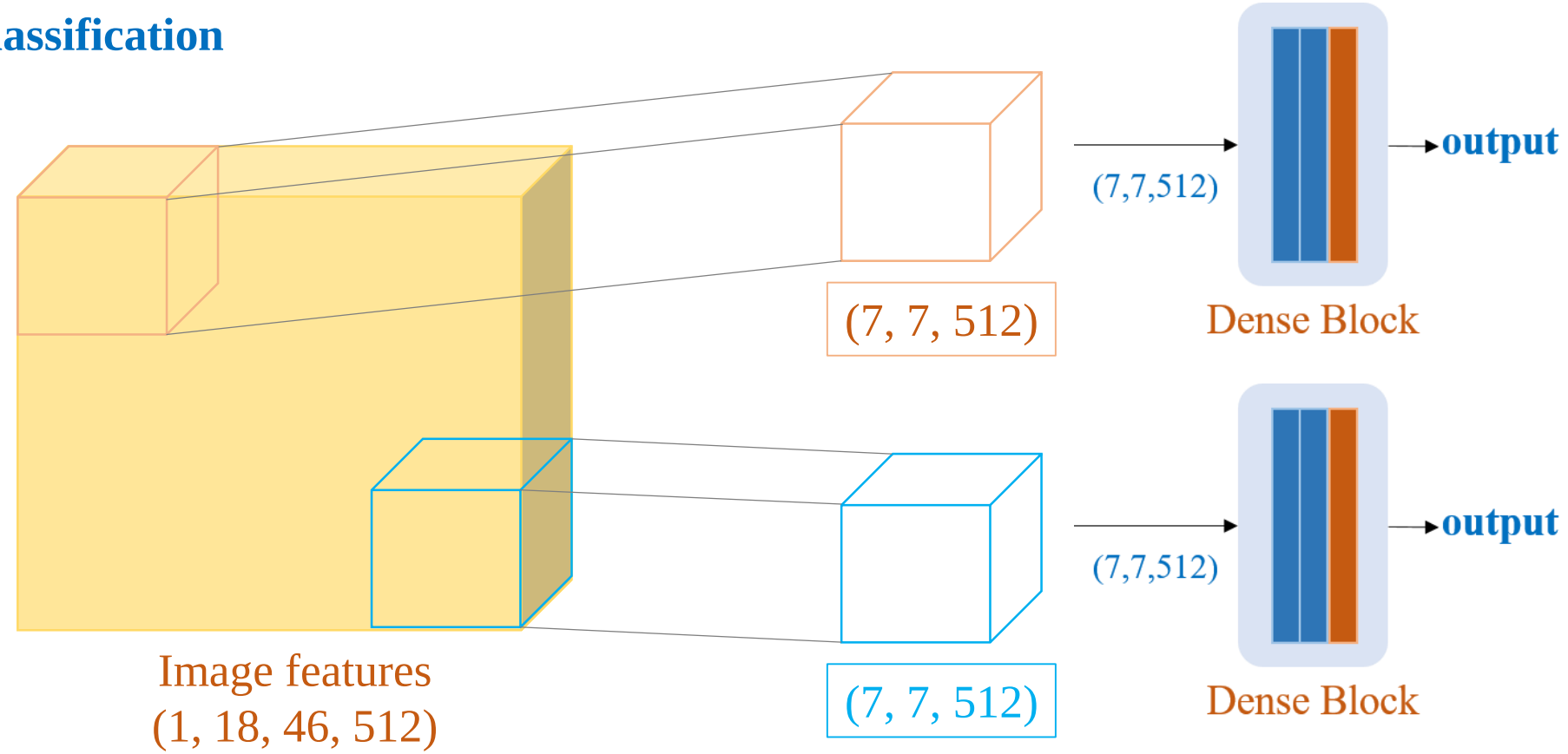
Feature Extraction



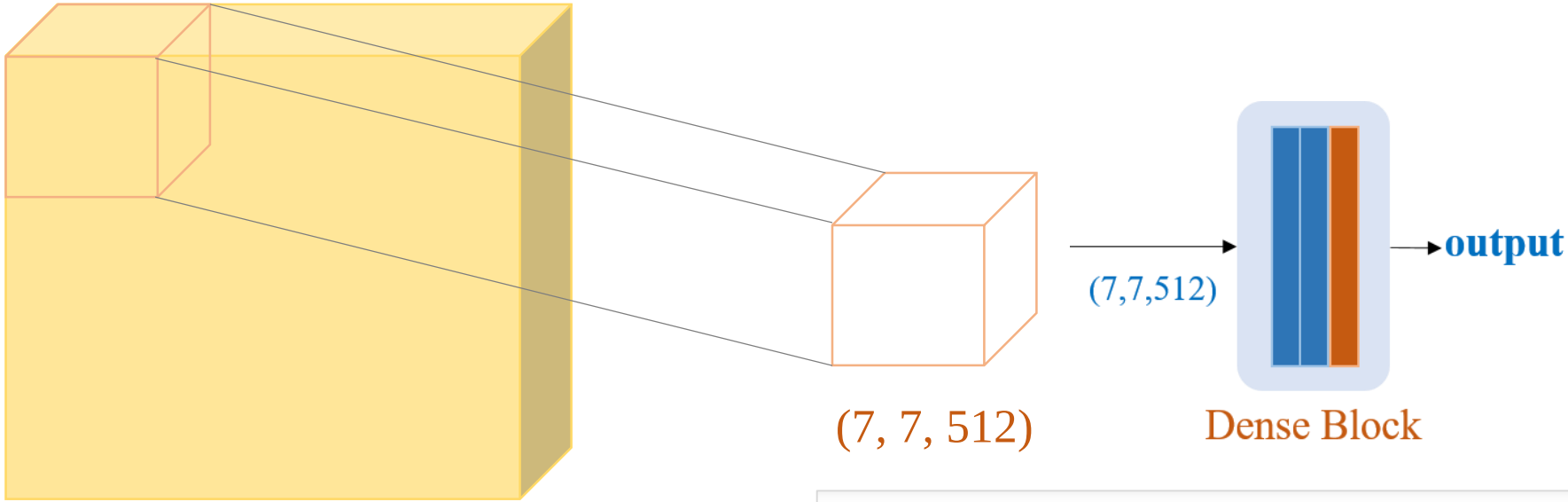
Naïve Object Detection

❖ Method 2: Open the black box

❖ Classification



Naïve Object Detection



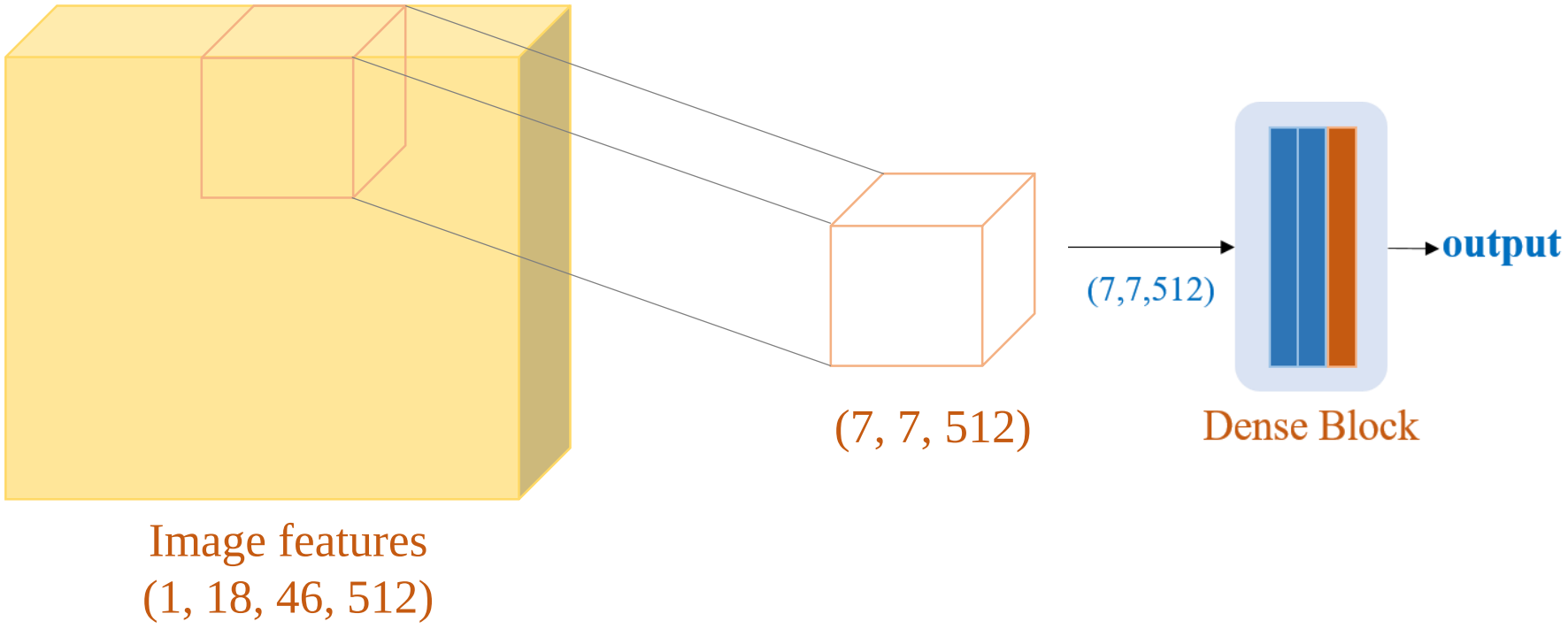

```
def compute_features(data):
  for i in range(1,19):
    data = model.layers[i](data)
  return data
```

```
def compute_prediction(data):
  for i in range(19,23):
    data = model.layers[i](data)
  return data
```

Naïve Object Detection

❖ Method 2: Open the black box

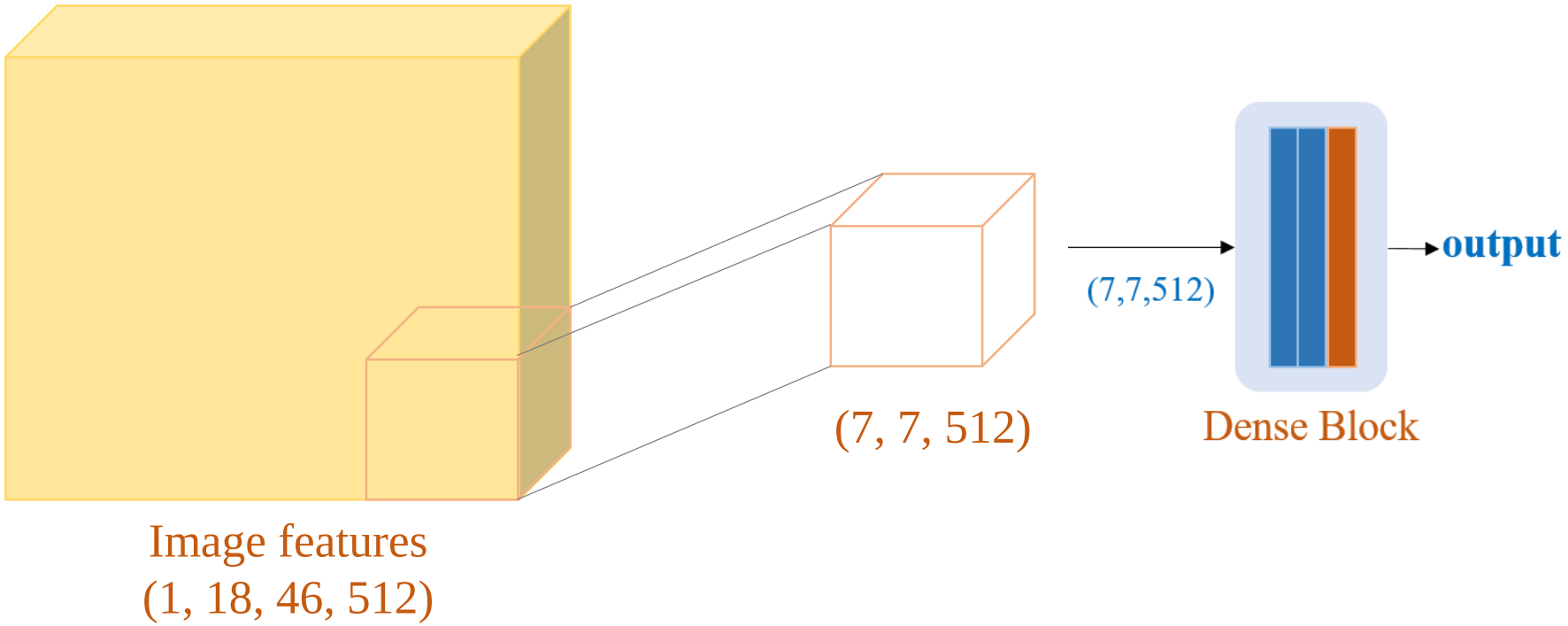
❖ Classification



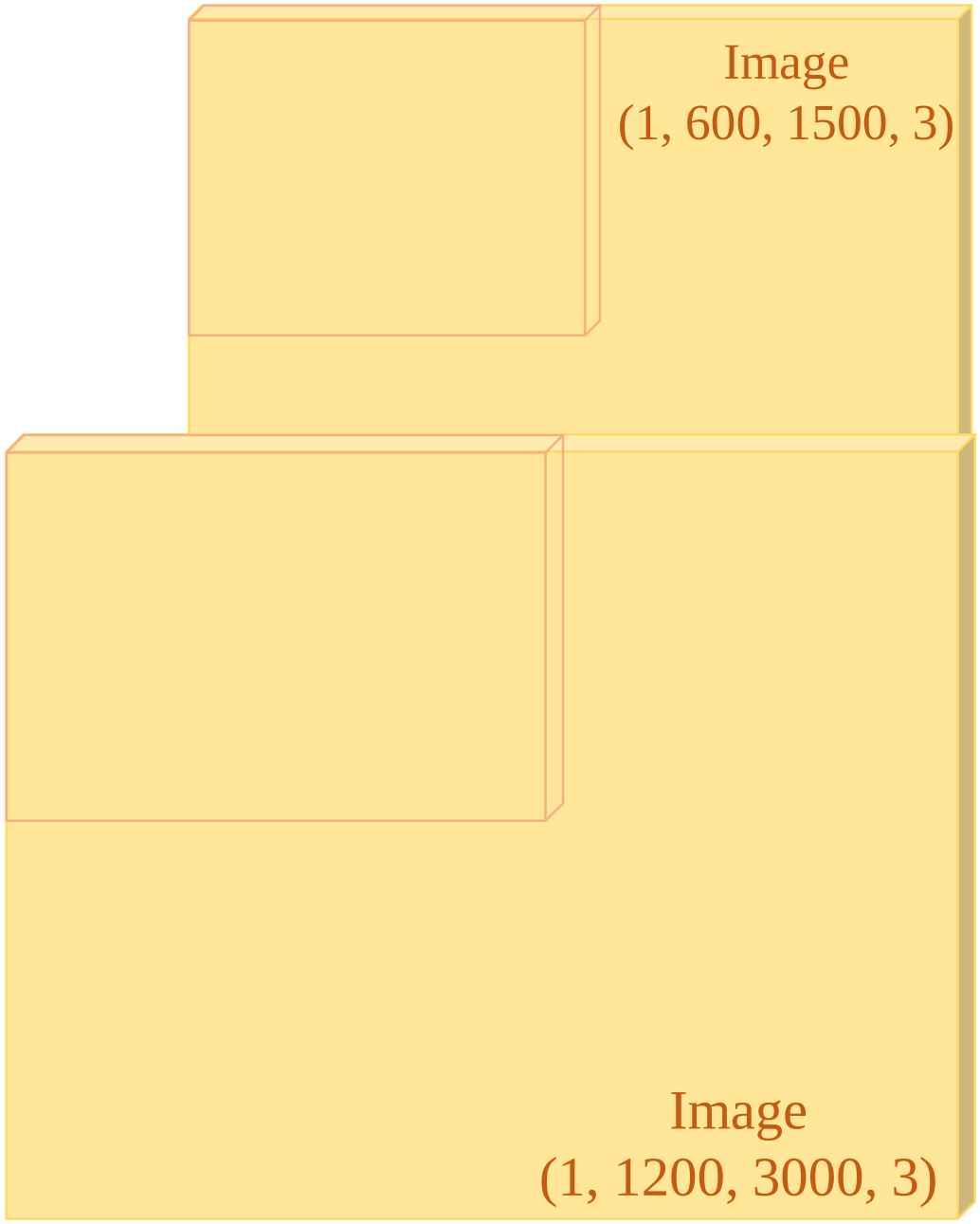
Naïve Object Detection

❖ Method 2: Open the black box

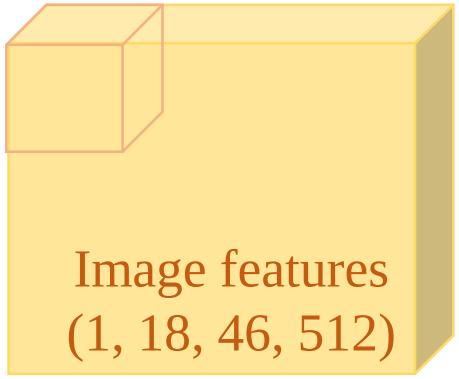
❖ Classification

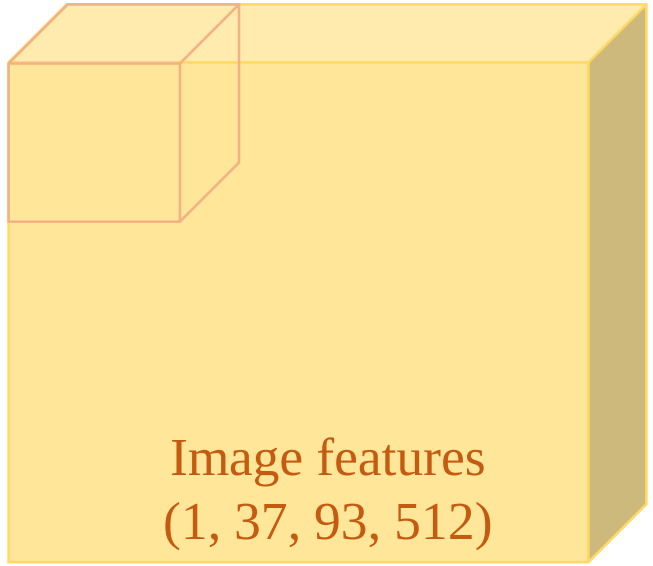


❖ Method 3: Multi-scale Inputs



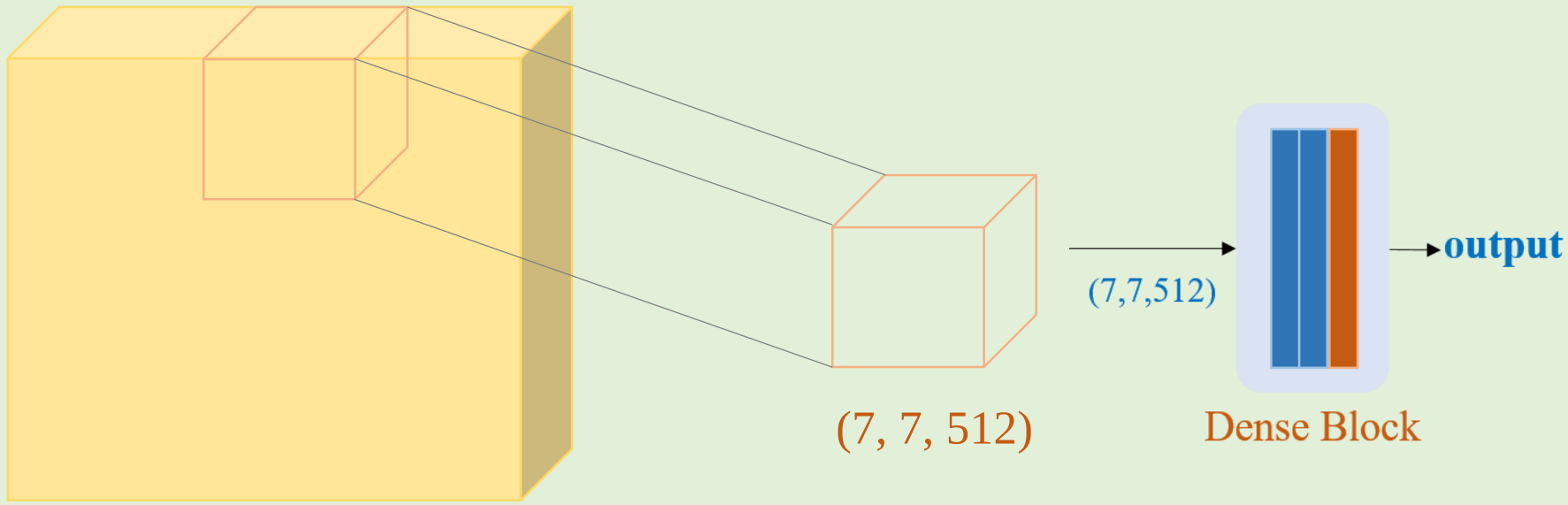
Feature Extraction





Naïve Object Detection

❖ Case study 1: Single Object Detection



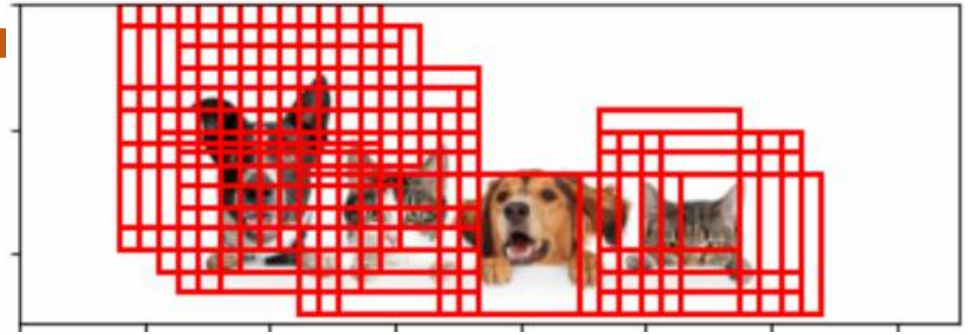
Naïve Object Detection

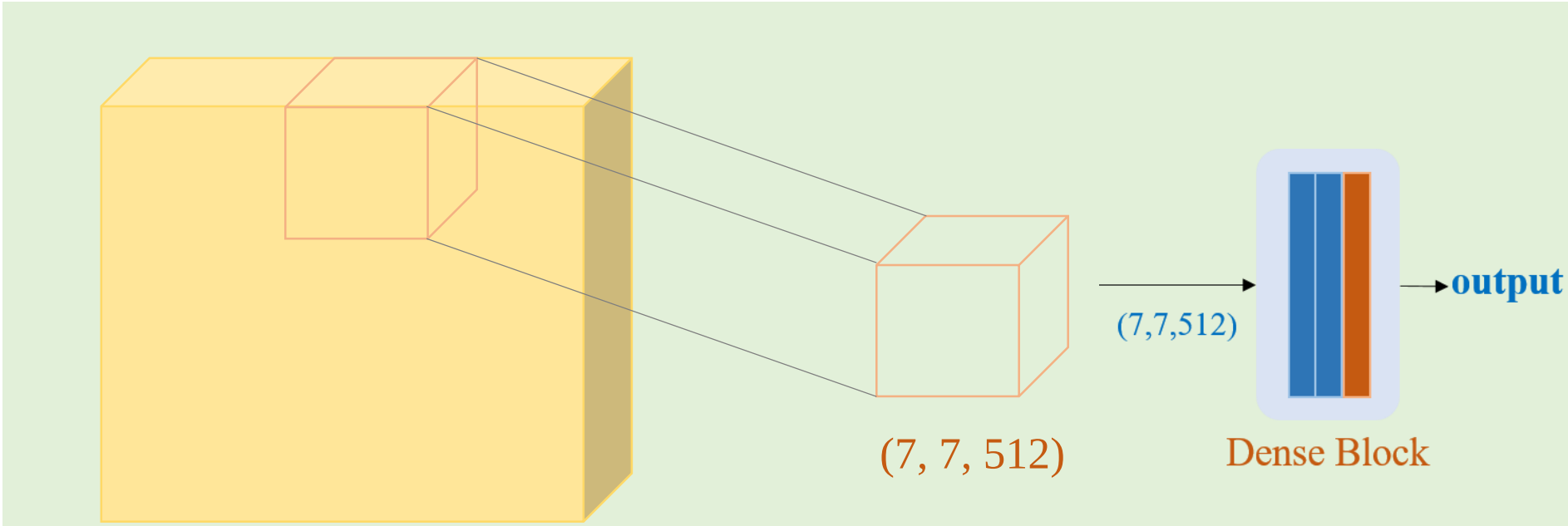
❖ Case study 1: Single Object Detection


```
# compute predictions
height = pred_query.shape[1]
width  = pred_query.shape[2]
depth  = pred_query.shape[3]

side = 7
prediction_data = []
for i in range(height-side+1):
  for j in range(width-side+1):
    patch = pred_query[:,i:i+side,j:j+side,:]
    patch = compute_prediction(patch)
    prediction_data.append( (tf.math.reduce_max(patch[0]).numpy(),
                             i, j, tf.math.argmax(patch[0]).numpy()) )
```

Naïve Object Detection

❖ Case study 2: Multi-Object Detection

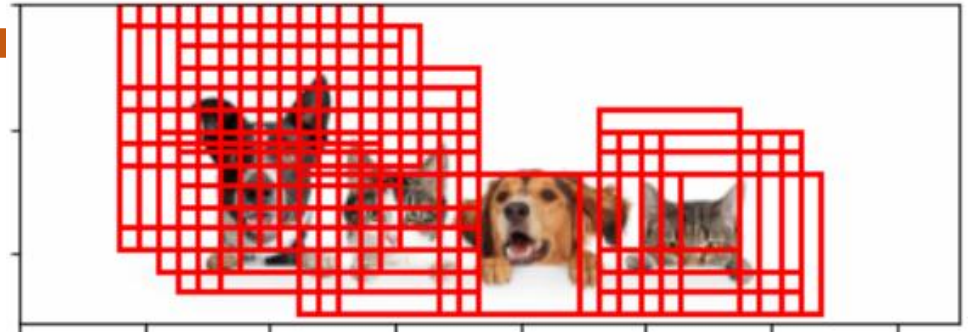




Get the K most similar patch

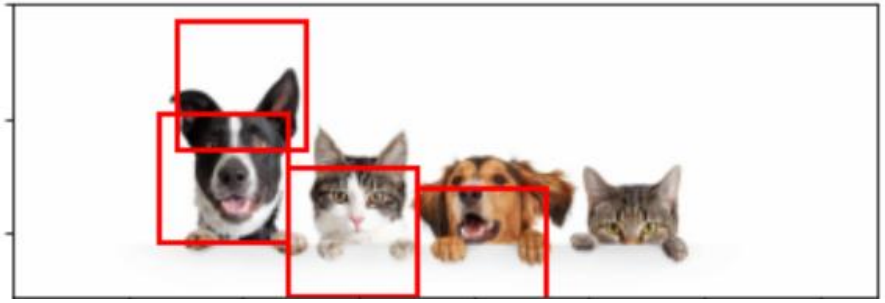
Naïve Object Detection

❖ Case study 2: Multi-Object Detection

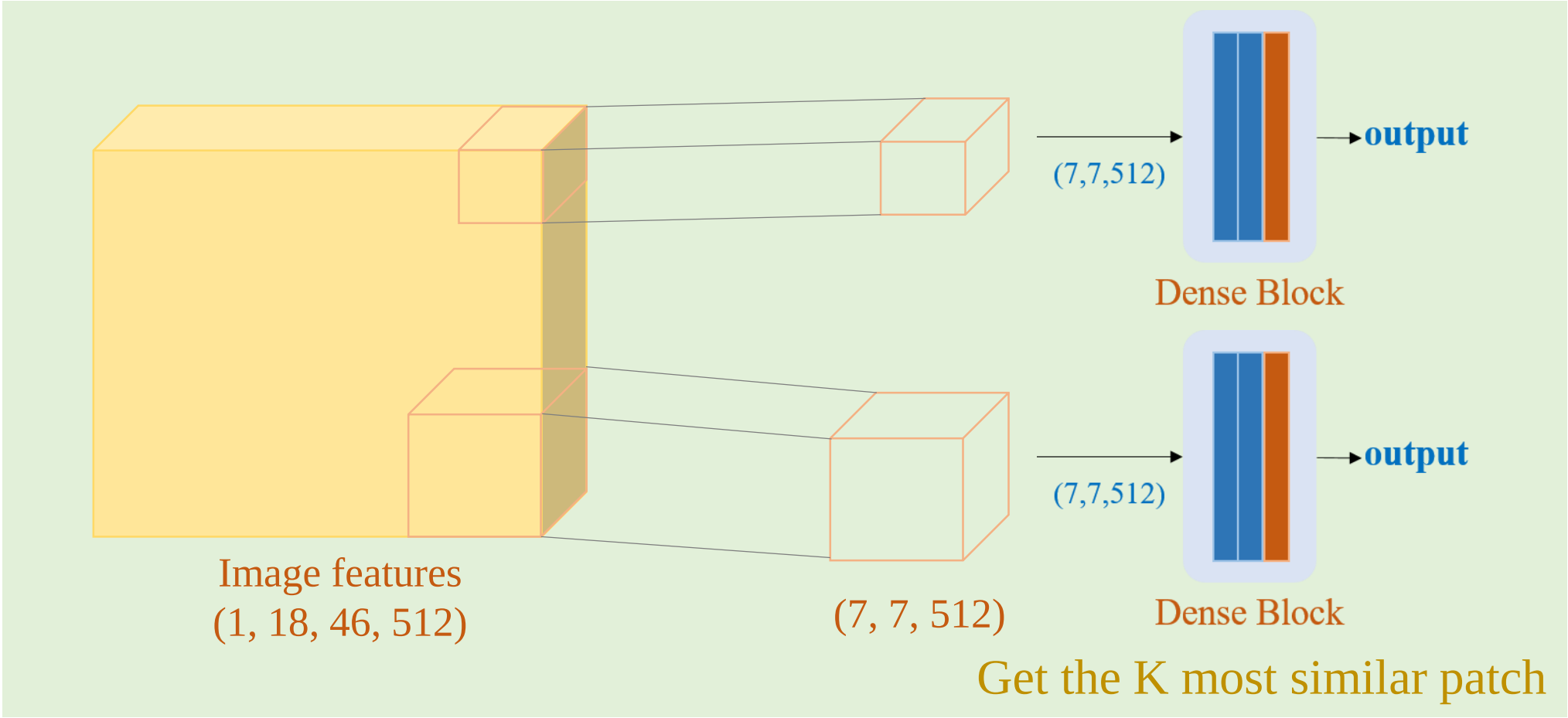


```
from scipy.spatial import distance

# remove duplication
def remove_duplication(data):
  result = []

  length = len(data)
  for k in range(length-1):
    duplicated = check_duplication(data[k][1], data[k][2], result)

    if (duplicated==False and data[k][0]>0.5):
      result.append( data[k] )


  return result
```

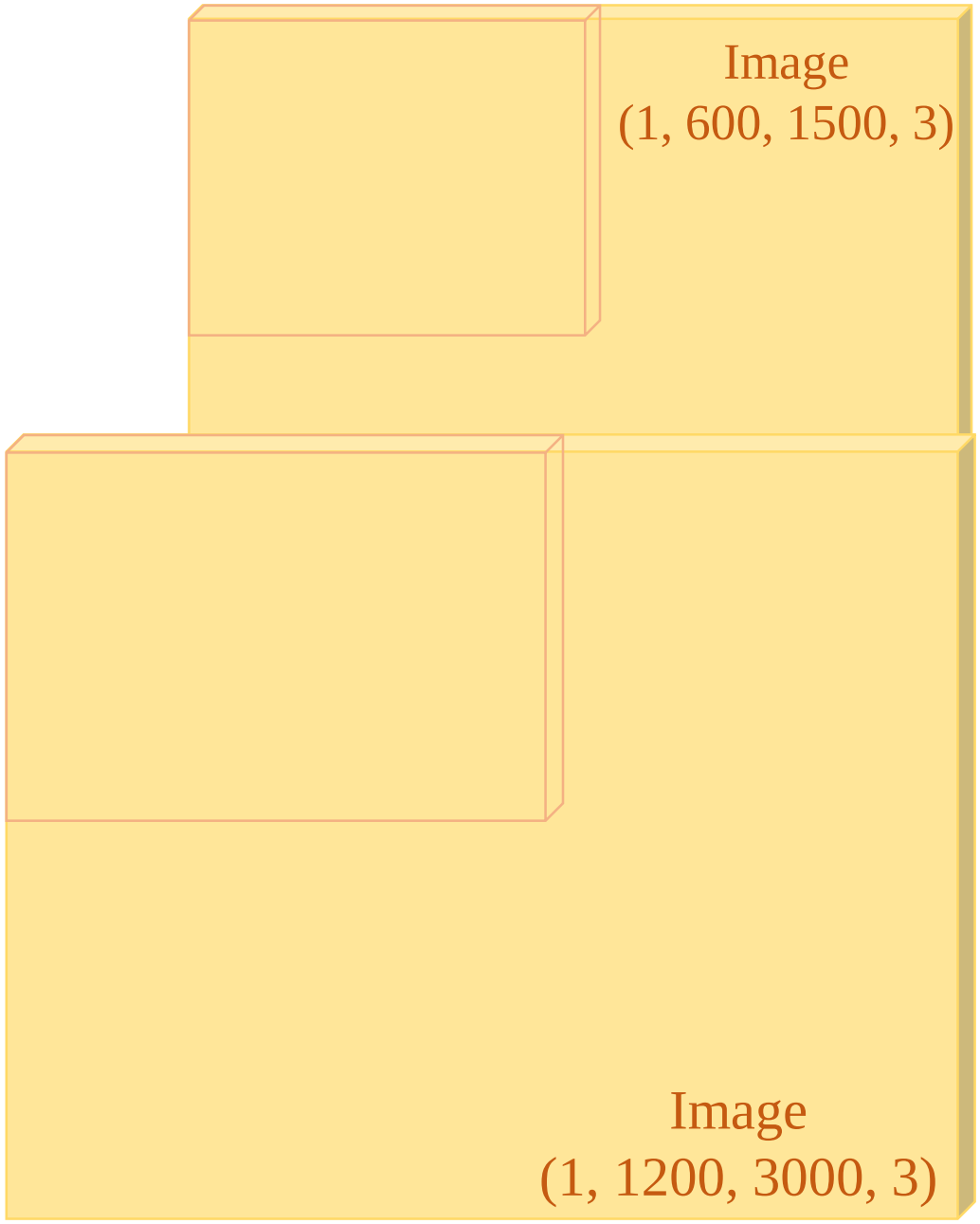
Naïve Object Detection



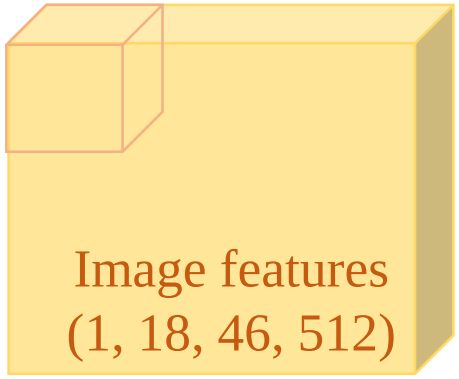
❖ Case study 3: Multi-scale Object Detection

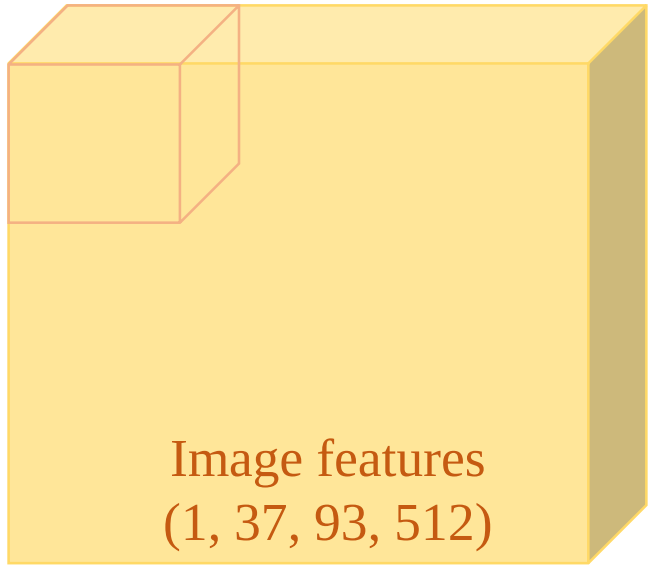


❖ Case study 3: Multi-scale Object Detection

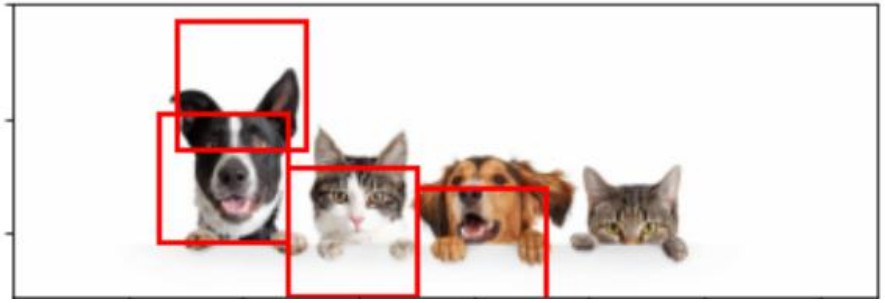


Feature Extraction





Naïve Object Detection



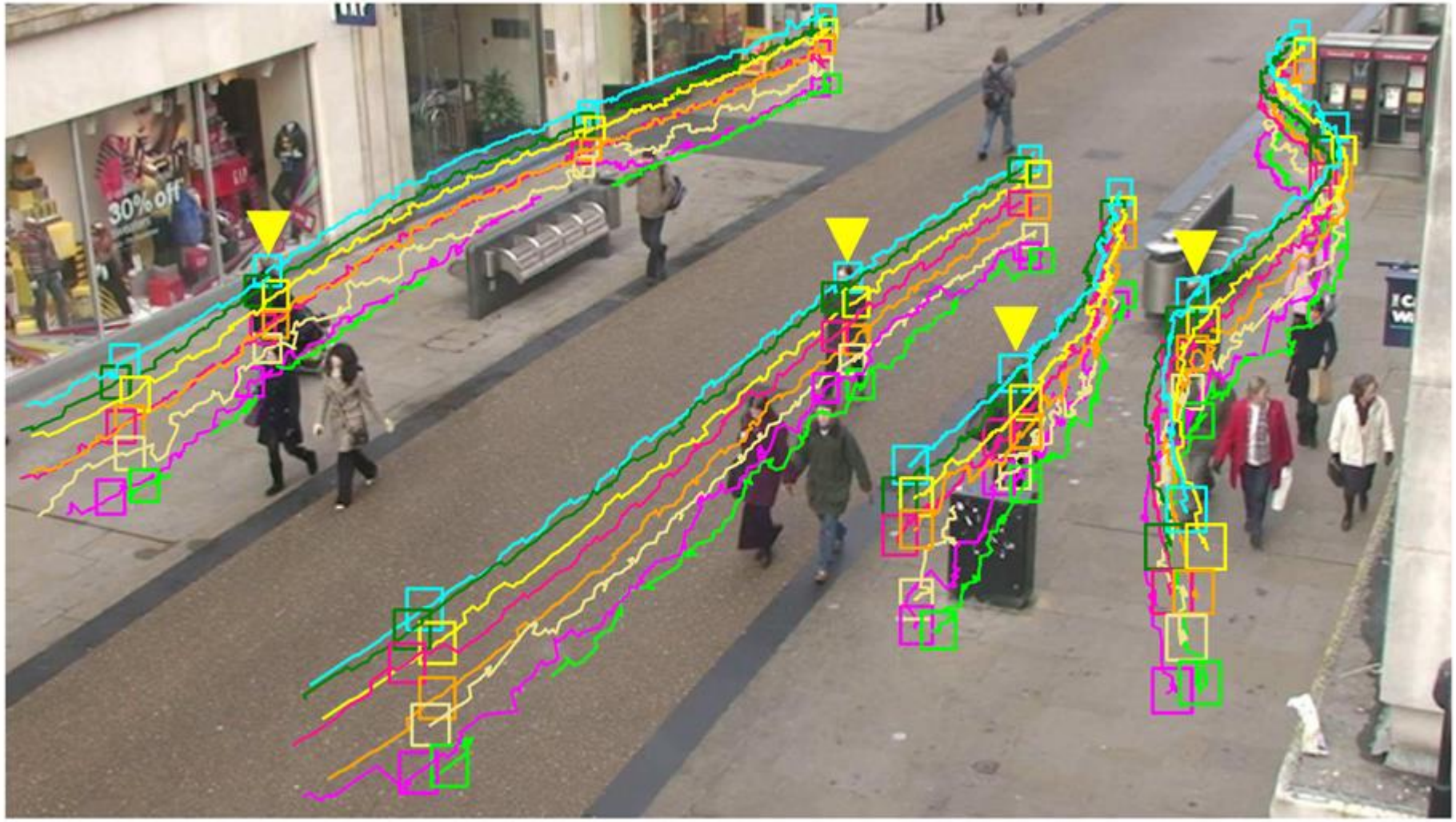
❖ Case study 3: Multi-scale Object Detection

```
# compute predictions
side = 7
prediction_data = []

for scale in range(scale_level):
  height_fm = list_of_features[scale].shape[1]
  width_fm  = list_of_features[scale].shape[2]
  depth_fm  = list_of_features[scale].shape[3]

  for i in range(height_fm-side+1):
    for j in range(width_fm-side+1):
      patch = list_of_features[scale][:,i:i+side,j:j+side,:]
      patch = compute_prediction(patch)
      prediction_data.append( (tf.math.reduce_max(patch[0]).numpy(),
                               i, j, tf.math.argmax(patch[0]).numpy(), scale) )

print(len(prediction_data))
```

Object Tracking Using Pretrained Model

Object Tracking

❖ **Objective**



http://deepmachinelearningai.com/object-tracking-in-deep-learning/

Object Tracking

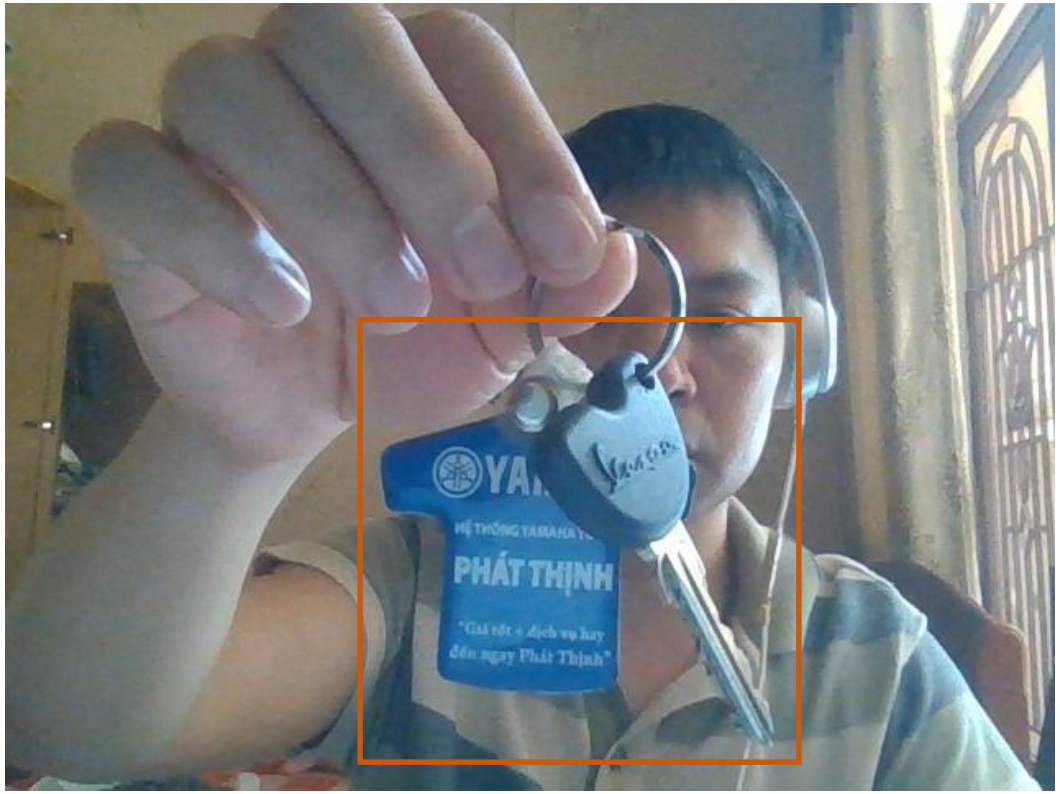
❖ Objective

Frame at time t

Frame at time k

Object Tracking

❖ Objective

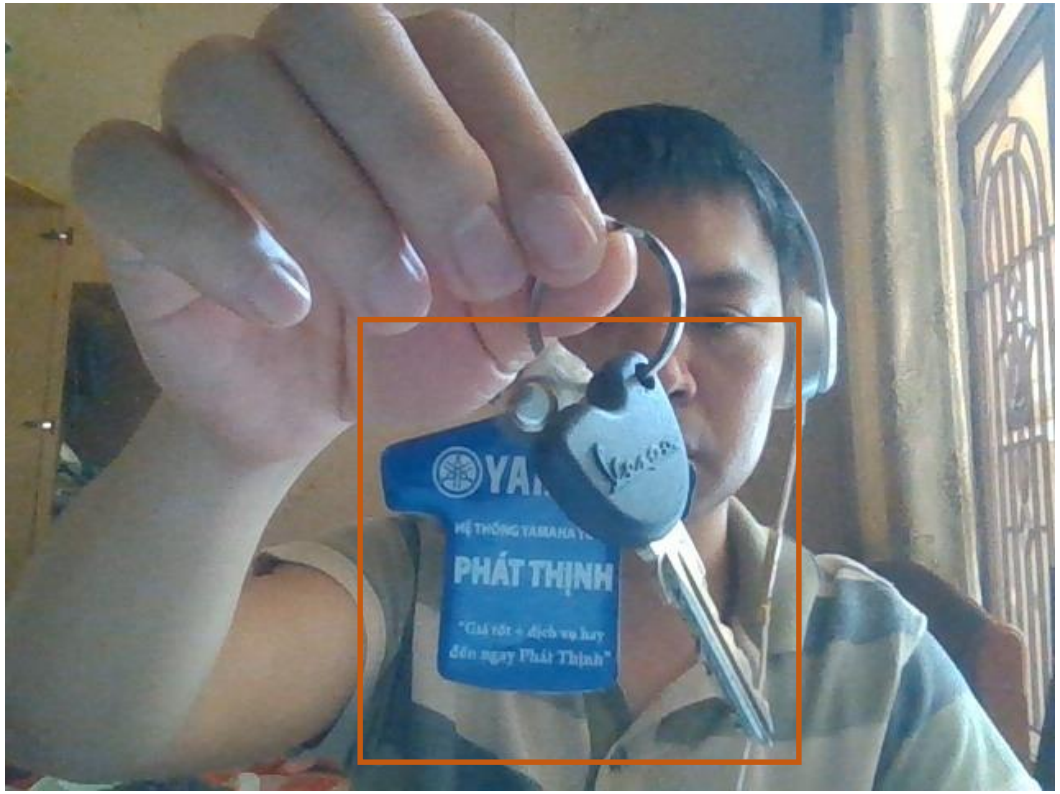


Frame at time t

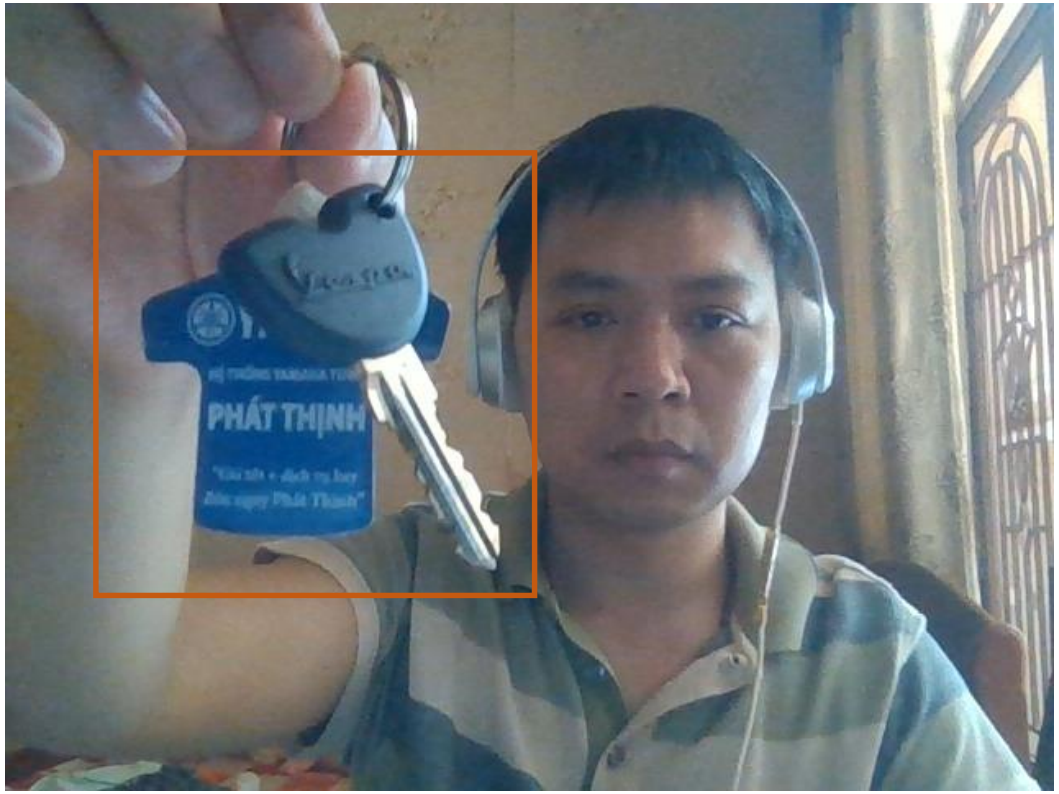
Frame at time k

Object Tracking

❖ Objective



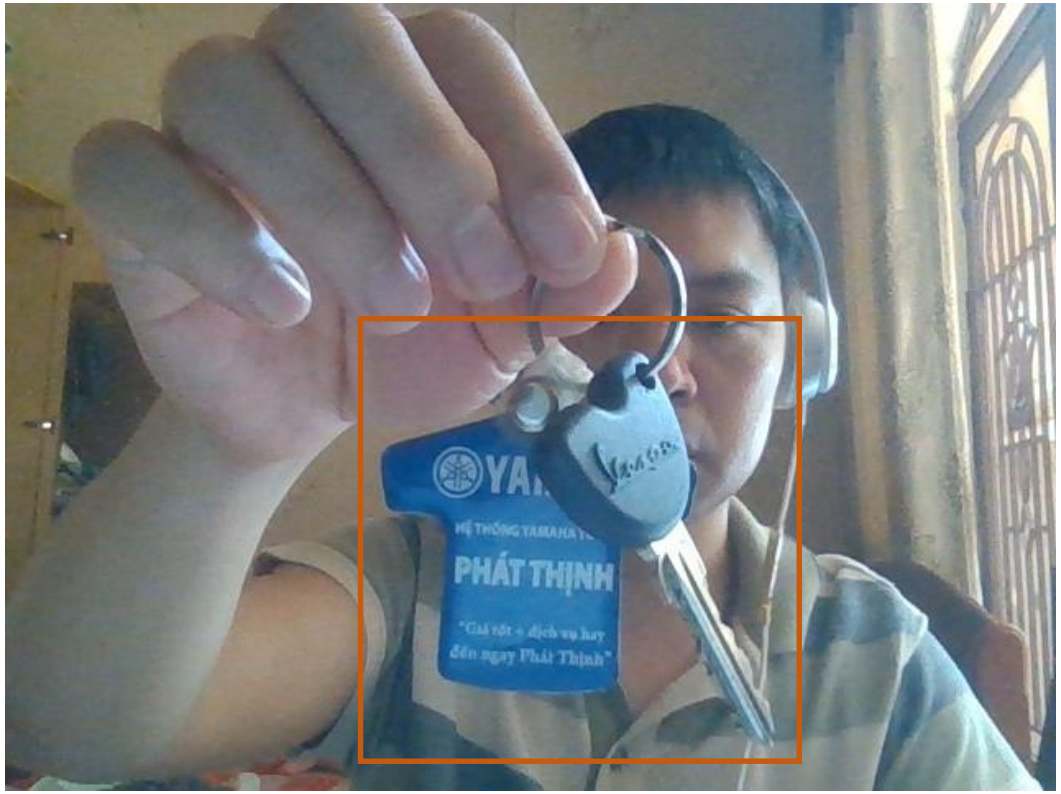
Frame at time t



Frame at time k

Object Tracking

❖ Idea



Frame at time t

Object Tracking

❖ Idea

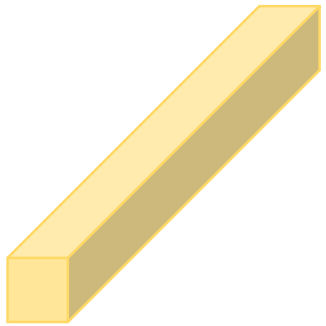
Frame at time k

Object Tracking

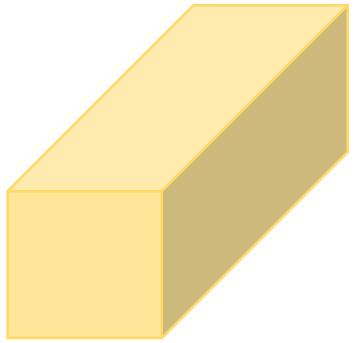
Template

Frame at time k

Feature Extraction

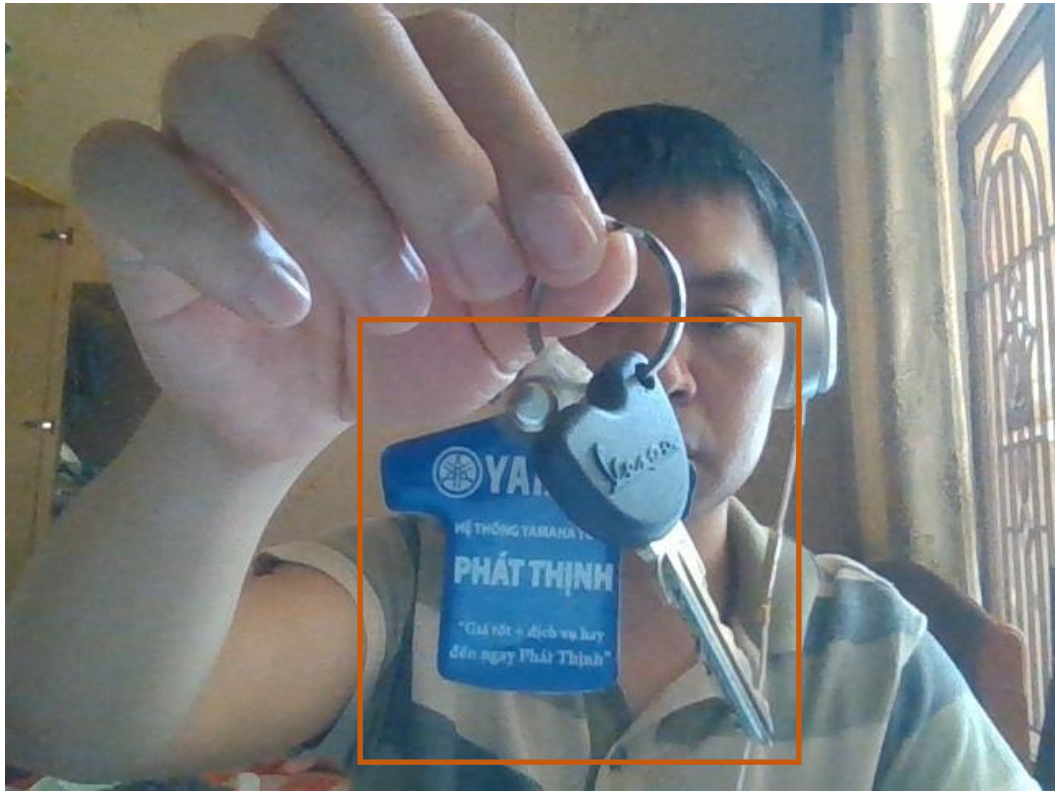


Feature Extraction



Object Tracking

❖ Case Study



Frame at time t

Frame at time k


Thank you!