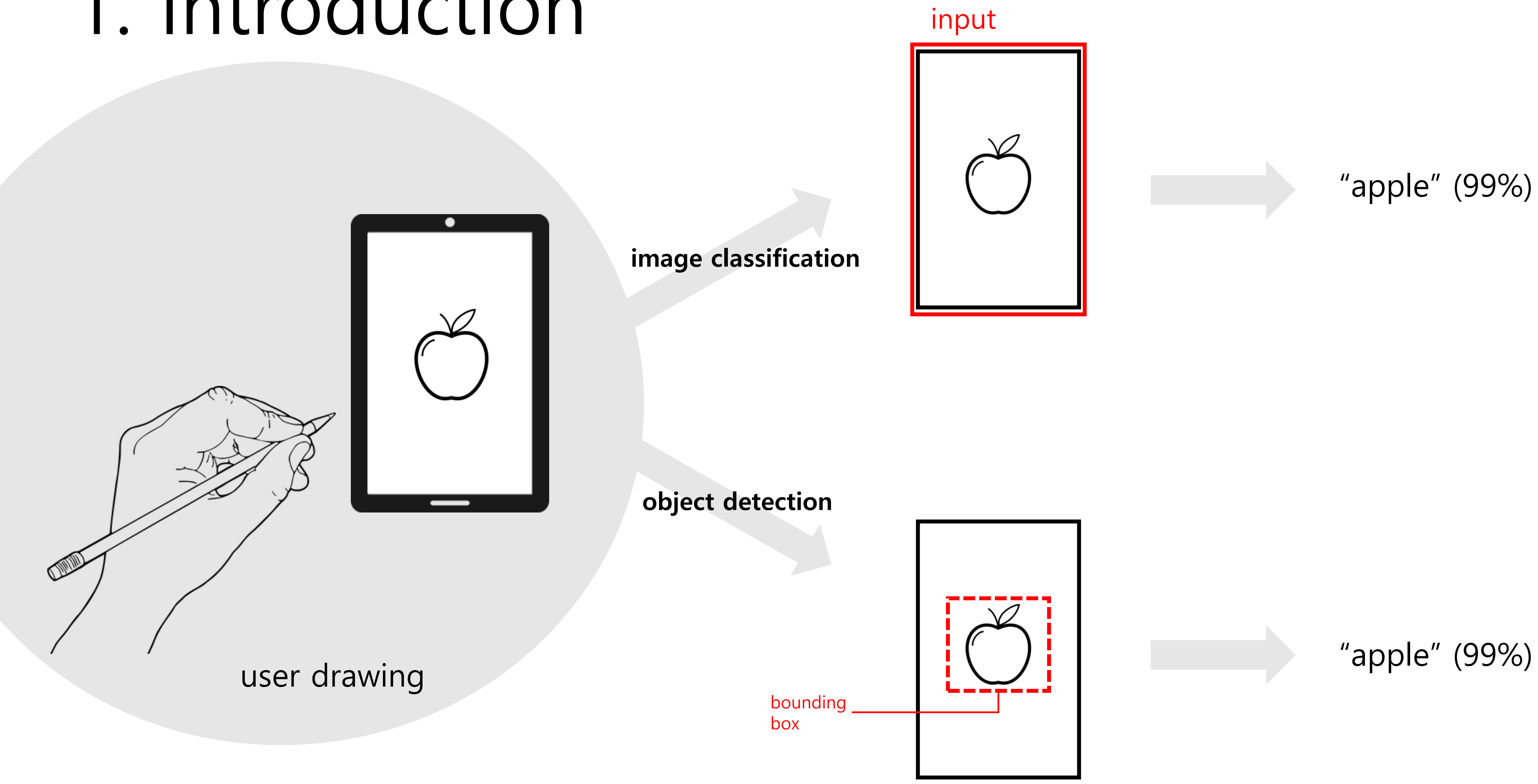
Capstone design

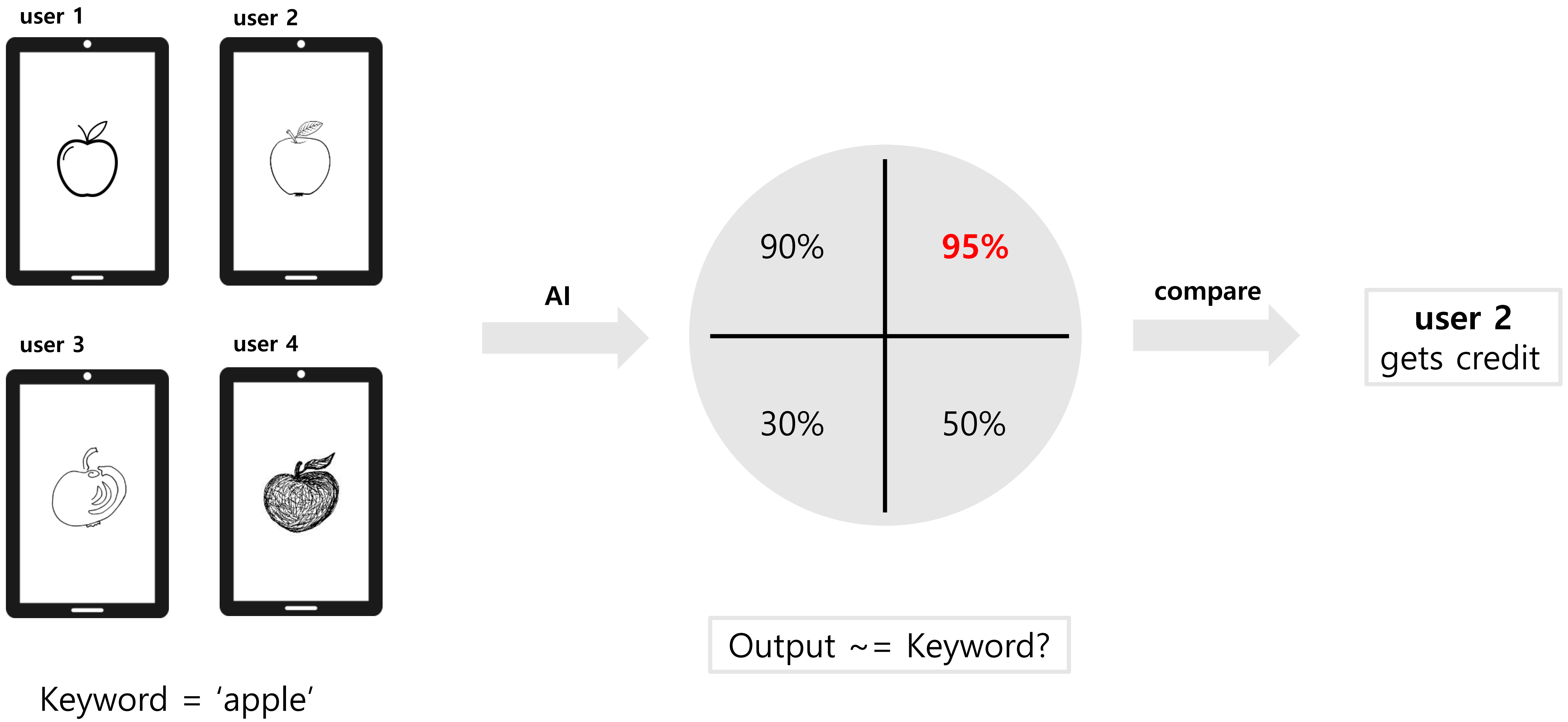
Team F

2016310237 김동우
2016314577 김동한
2016311033 김영현
2017312329 최형규

Contents

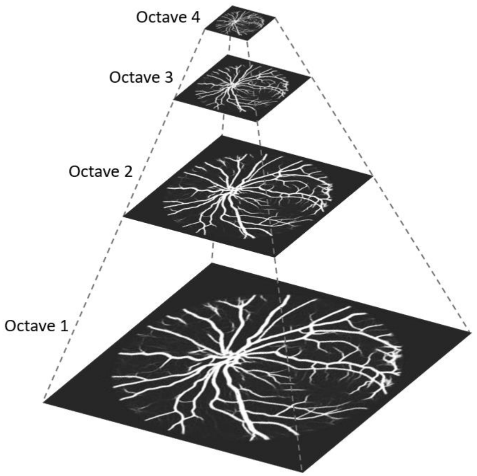
1. Introduction

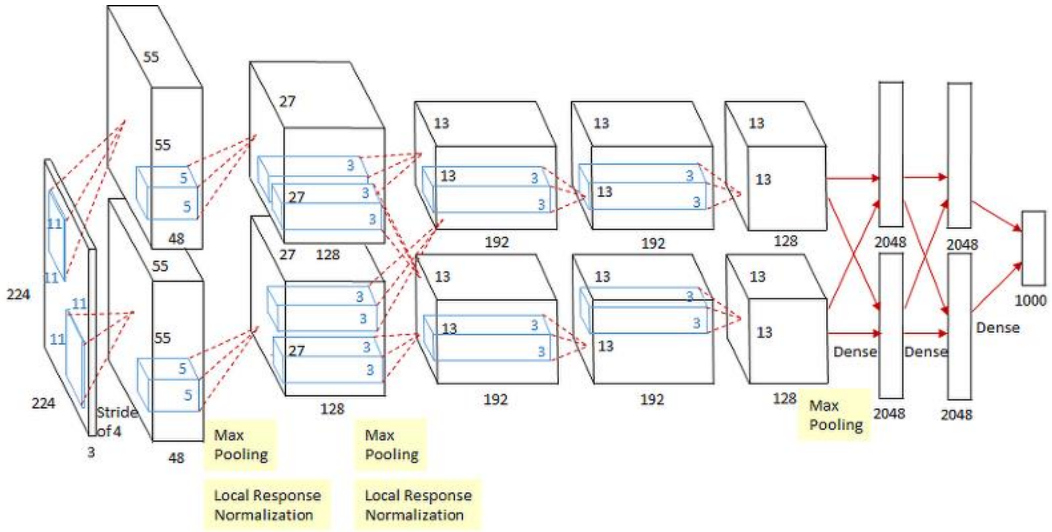



user 1
user 2
user 3
user 4
Keyword = 'apple'
AI
90%
95%
30%
50%
Output ~= Keyword?
compare
user 2
gets credit

2. Image Classification

- ImageNet Large Scale Visual Recognition Challenge (ILSVRC)
- (1000) classes + (1.35M) dataset

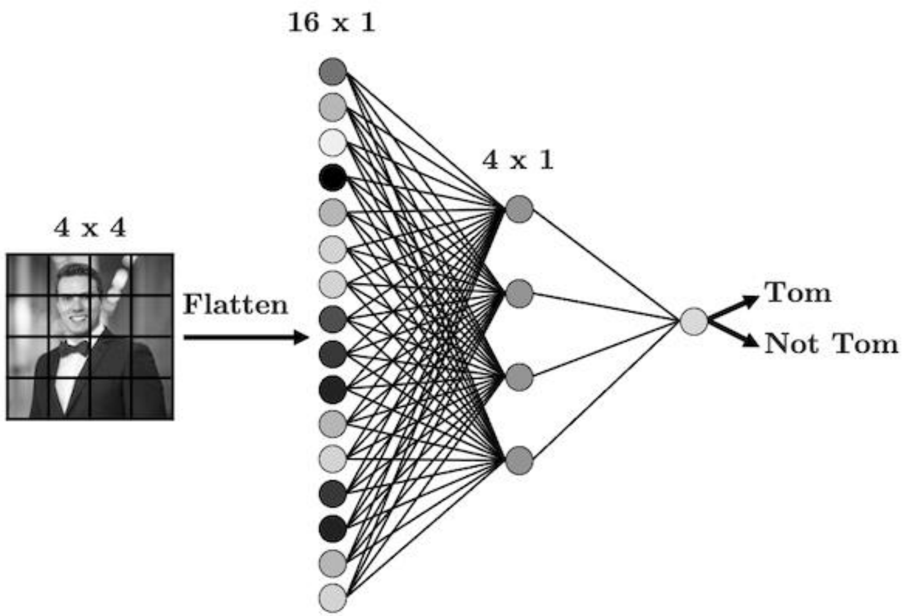




(~2011) sift/surf descriptor based algorithm

(2012~) DNN algorithm

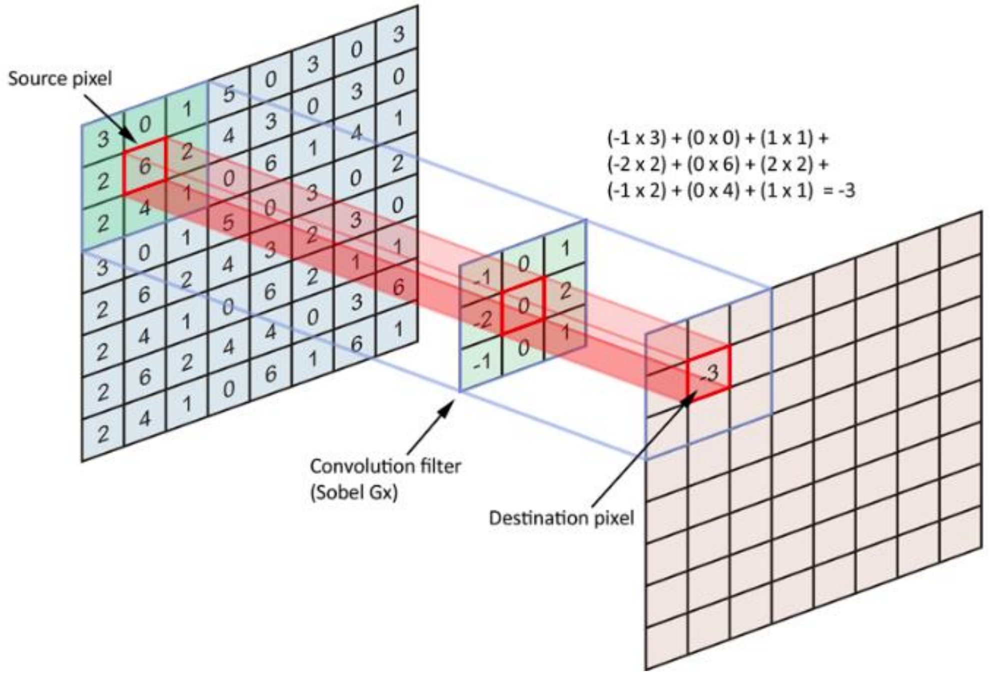
Multi-Layerd Perceptron (MLP)



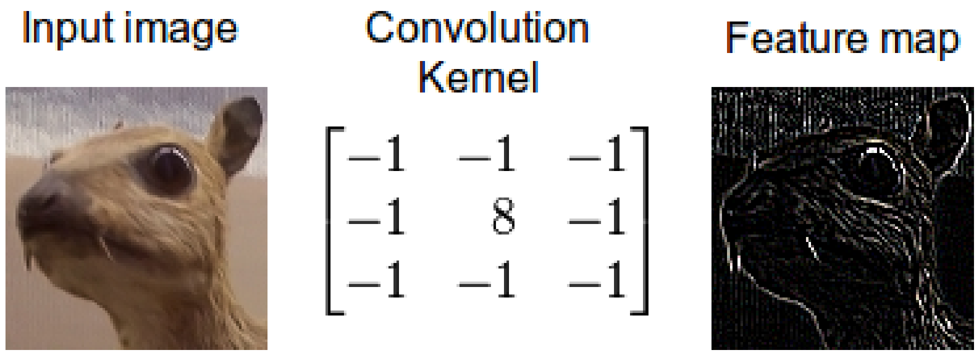
- flatten the input into 1-dimensional and inject into the fully connected layer
- important spatial informations are inherently lost

VS

convolutional Neural Network (CNN)



- more appropriate for images and video frames
- kernel slides through the entire input grid and creates a feature map
- keep spatial informations

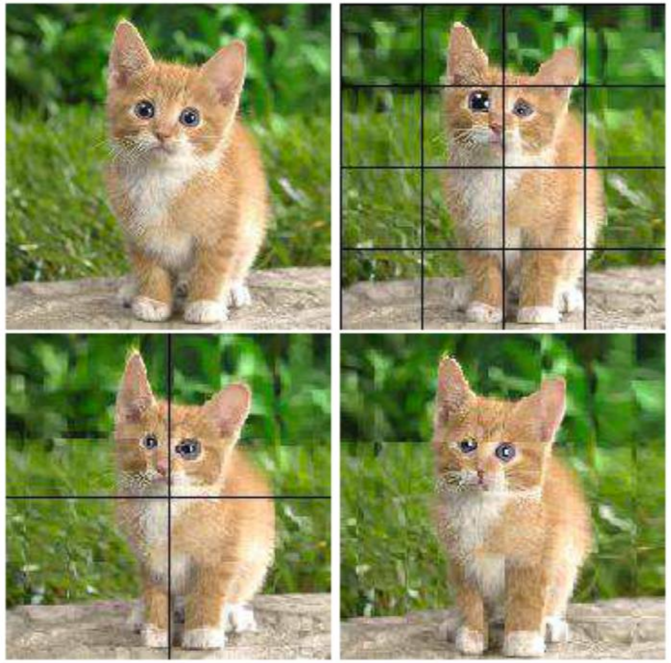


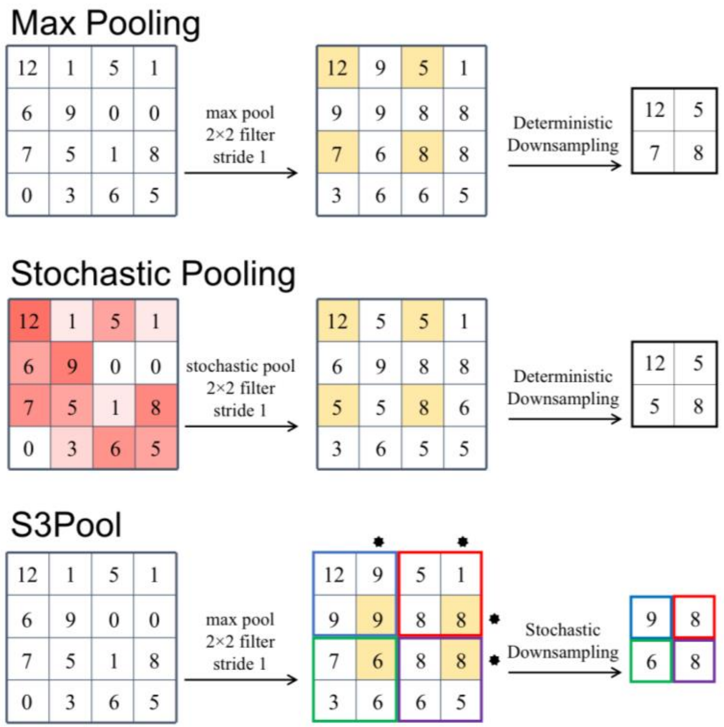
Convolution layer)

- map the input into the feature space to obtain important features of the image
- kernel size, num of kernel, stride, padding

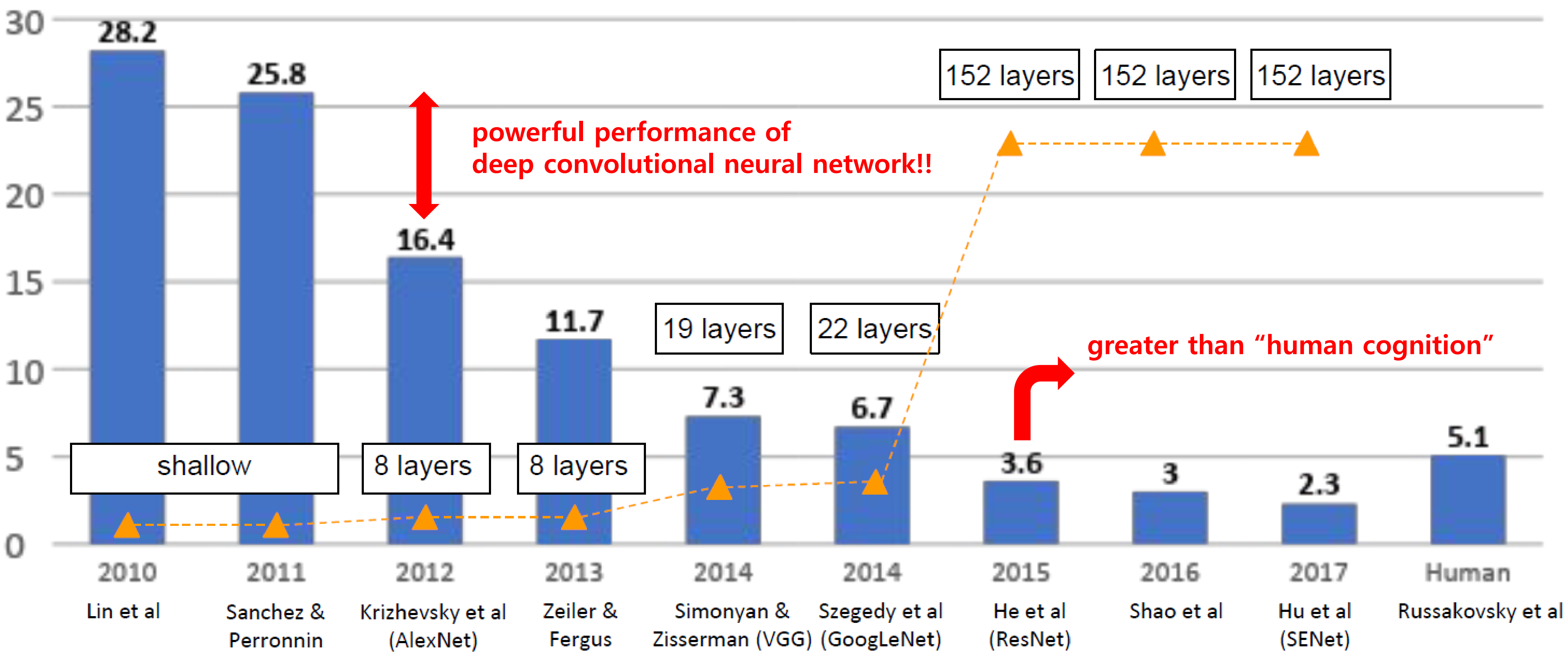
Pooling layer)

- reduce the size of the feature map
- no params for pooling layer itself
- lower params, suppress overfitting, lower computation, lower hardware resources
- Max Pooling, Stochastic Pooling..
- pooling size, stride, padding

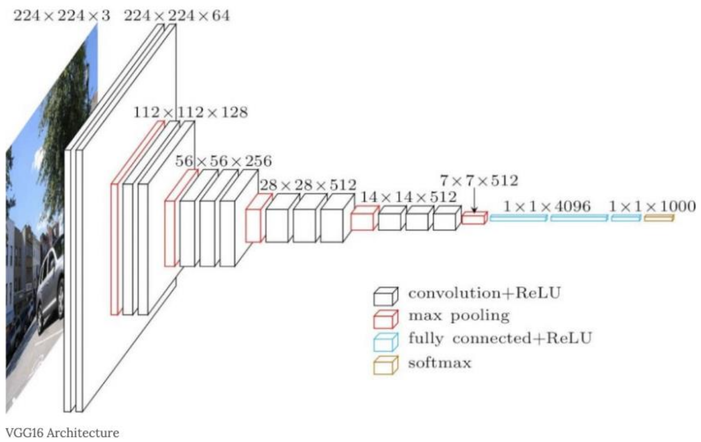




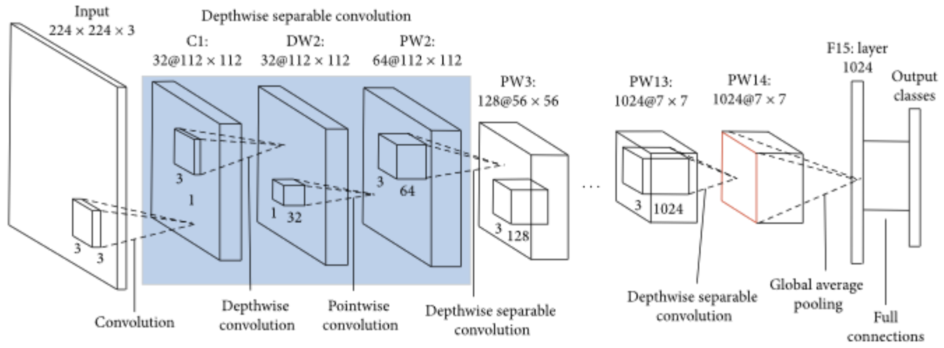
ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



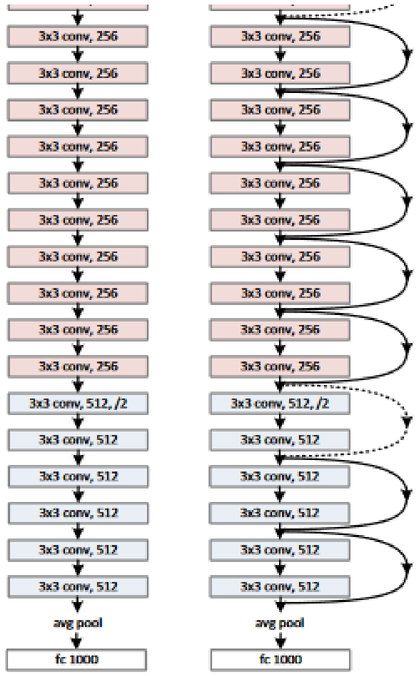
Our proposed candidate models)



VGGNet-16/19

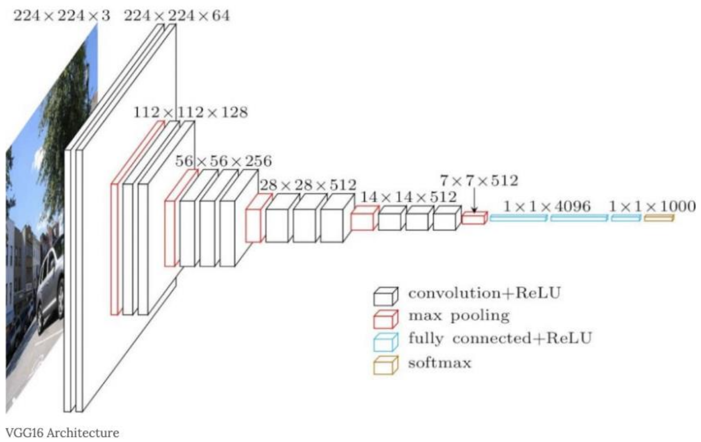


MobileNet

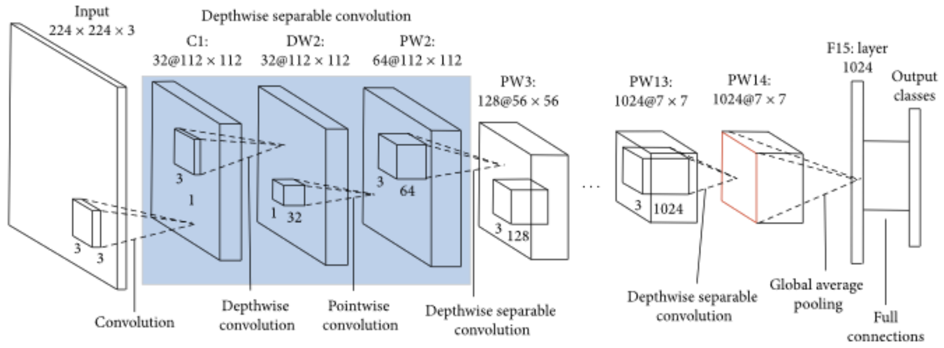


ResNet-50

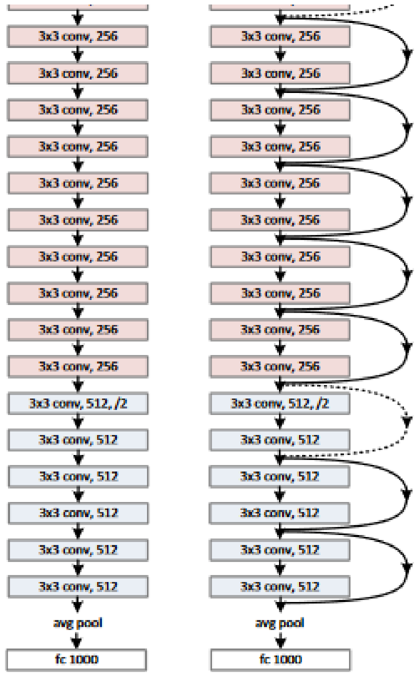
Our proposed candidate models)



VGGNet-16/19



MobileNet



ResNet-50

MobileNet v1

<u>Previous models…</u>
- have tremendous number of parameters(weights)
- require high computational power & memories
- e.g. ALPHAGO used 1202 CPUs and 176 GPUs

<u>However in real world…</u>
- numerous environments s.t…
    - no GPUs
    - only one CPU, lack of memories

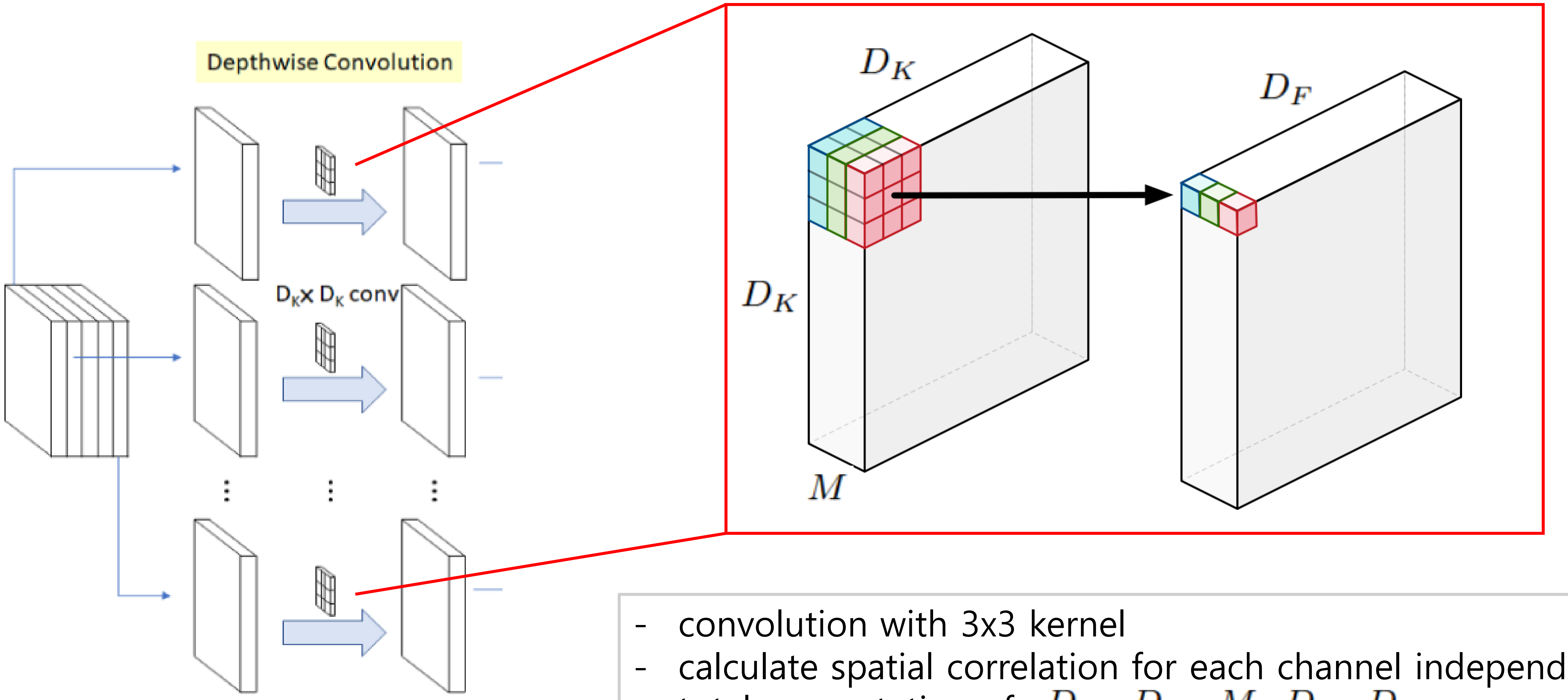
Figure 1. MobileNet models can be applied to various recognition tasks for efficient on device intelligence.

<u>MobileNet is exactly for these situations</u>
- small num of parameters
- model with less complexity
- low latency & high speed

(CVPR, 2017) "MobileNet: Efficient Convolutional Neural Network for Mobile Vision Application"

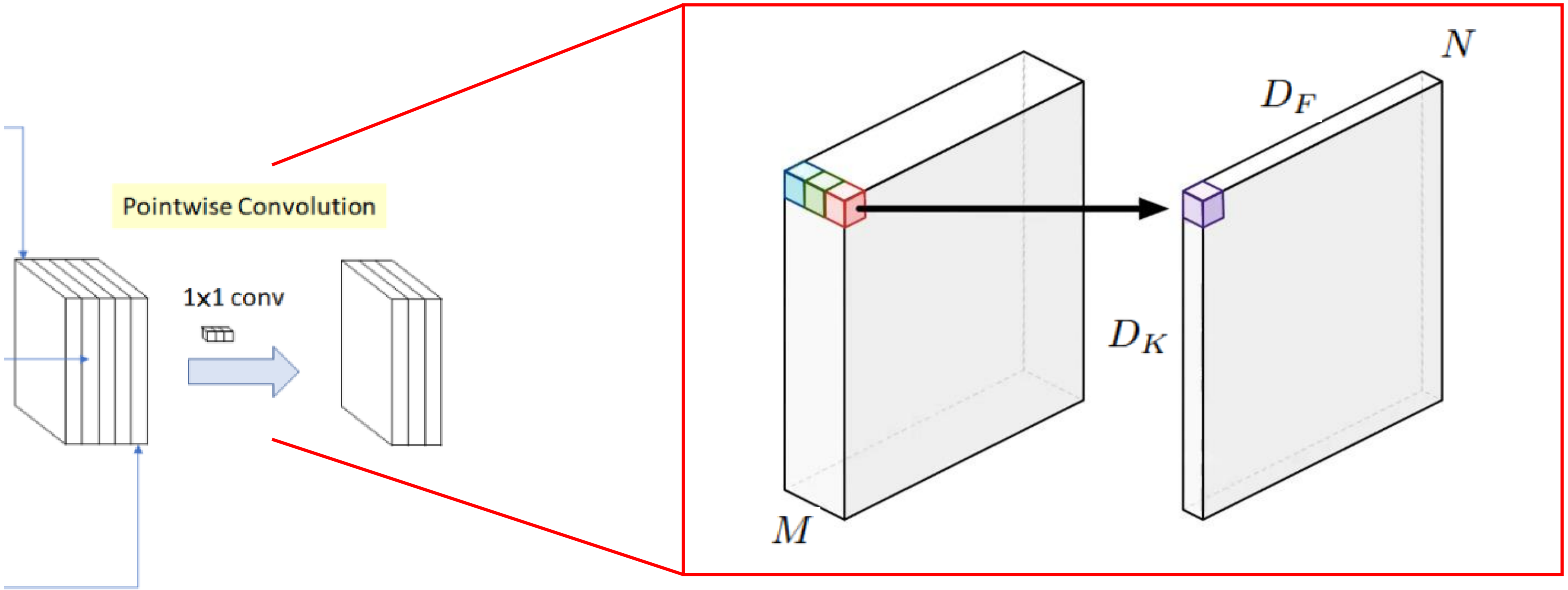
1) Depthwise convolution

$$M \cdot N \cdot D_F \cdot D_F$$



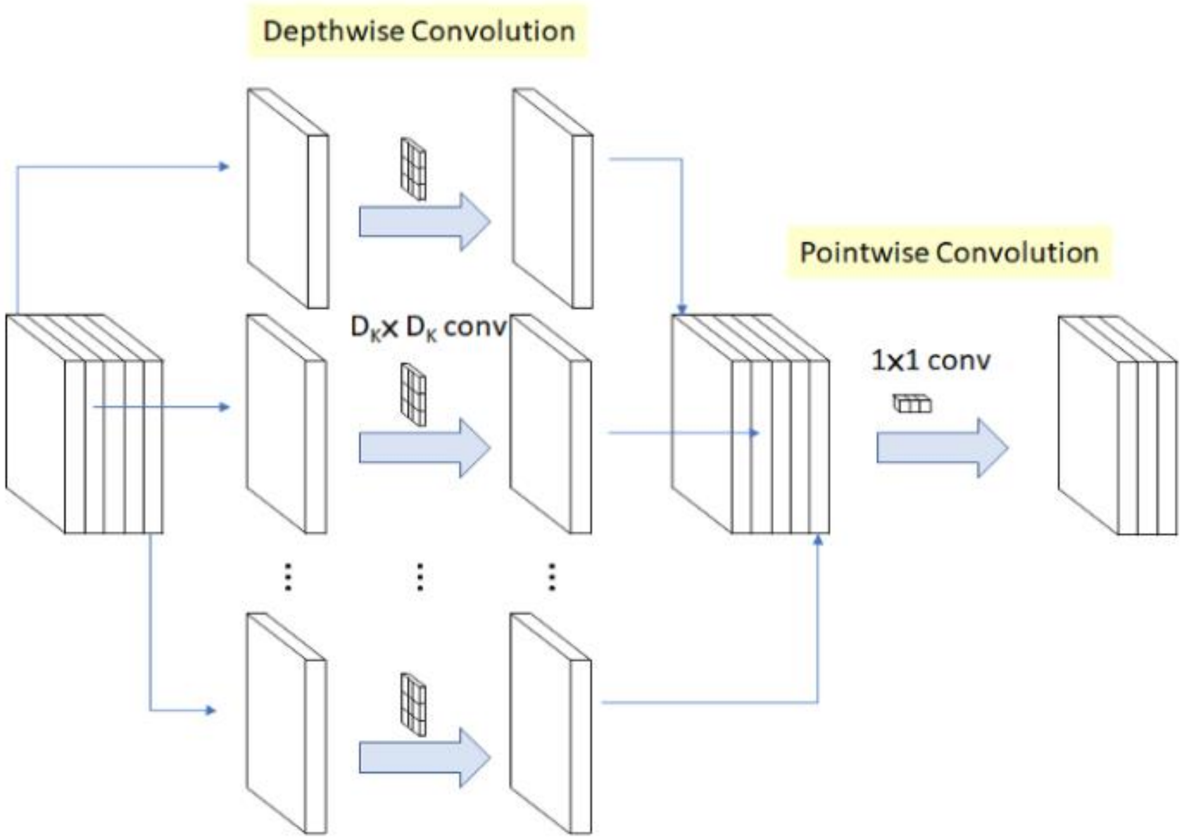
- convolution with 3x3 kernel
- calculate spatial correlation for each channel independently
- total computation of $D_K \cdot D_K \cdot M \cdot D_F \cdot D_F$

2) Pointwise convolution



- convolution with 1x1 kernel => control the size of out-channel
- calculate cross-channel correlation
- total computation of $M \cdot N \cdot D_F \cdot D_F$

3) Depthwise separable convolution



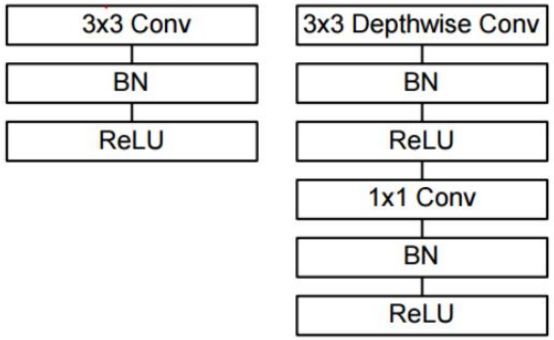


Figure 3. Left: Standard convolutional layer with batchnorm and ReLU. Right: Depthwise Separable convolutions with Depthwise and Pointwise layers followed by batchnorm and ReLU.

- total calculation for depthwise separable convolution:

$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F$$

- total calculation for traditional convolution:

$$D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F$$

- more than 8~9 times less computation

4) parameters for latency and accuracy

1) width multiplier: latency(α) (default=1.0, range=0.0~1.0)
- control the 'width' of the layer
- total computation of $D_K \cdot D_K \cdot \alpha M \cdot D_F \cdot D_F + \alpha M \cdot \alpha N \cdot D_F \cdot D_F$

Table 6. MobileNet Width Multiplier

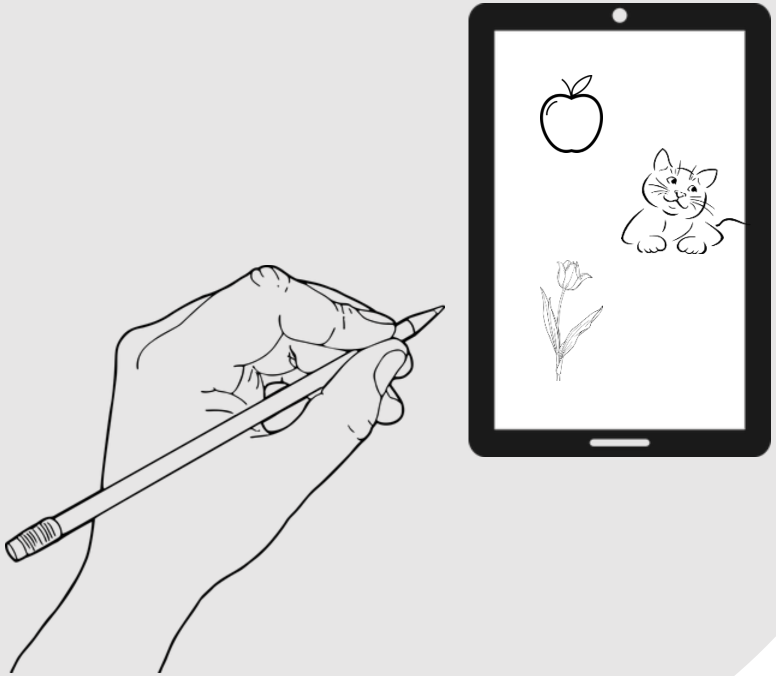
Width Multiplier	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
0.75 MobileNet-224	68.4%	325	2.6
0.5 MobileNet-224	63.7%	149	1.3
0.25 MobileNet-224	50.6%	41	0.5

Table 7. MobileNet Resolution

Resolution	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
1.0 MobileNet-192	69.1%	418	4.2
1.0 MobileNet-160	67.2%	290	4.2
1.0 MobileNet-128	64.4%	186	4.2

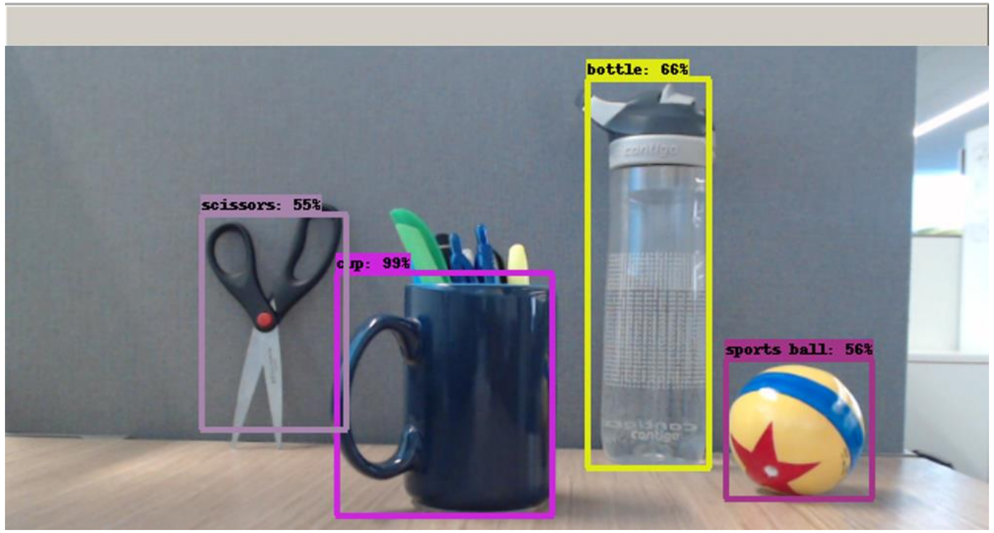
2) resolution multiplier: accuracy(ρ) (default=1.0, range=0.0~1.0)
- control the 'resolution' of the image
- in paper, author tests for image size of 224, 192, 169, 128

3. Object Detection

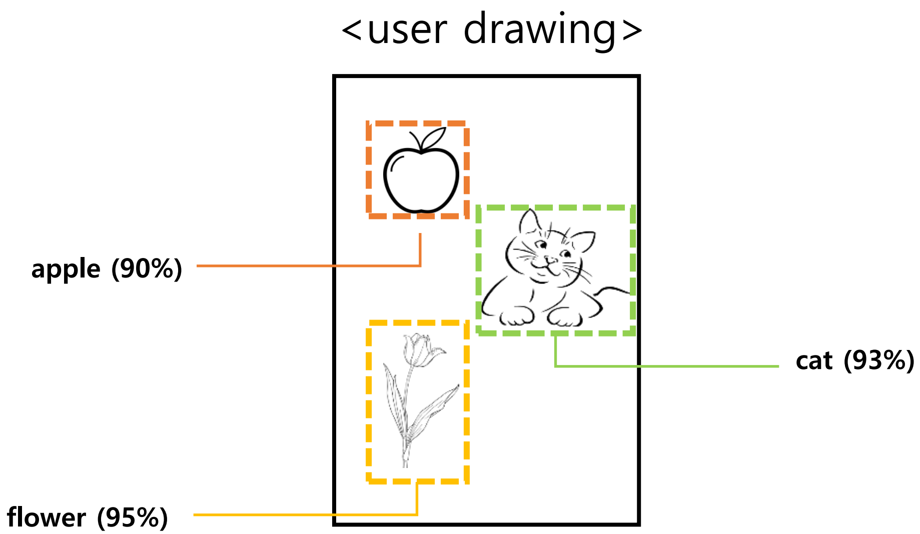


user drawing

- Image classification : one class per one image
- *what if multiple classes in one image?*



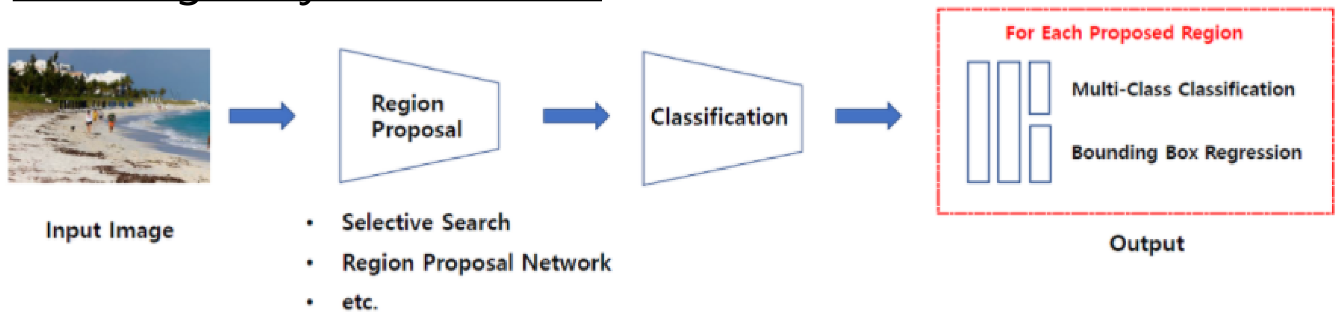
(object detection)



(our usage of object detection)

- Object detection: localization + image classification
- Use Object detection for input with multiple classes of doodling

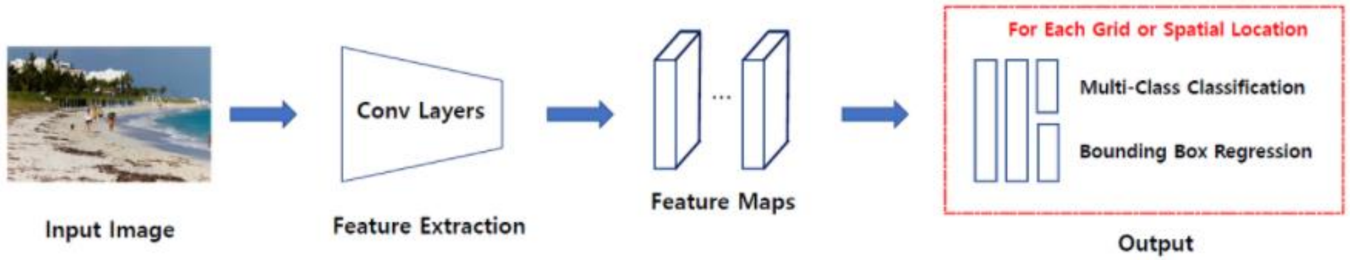
two-stage object detection



- localization & classification in different step
- relatively high accuracy, low speed
- e.g. R-CNN, Fast R-CNN…

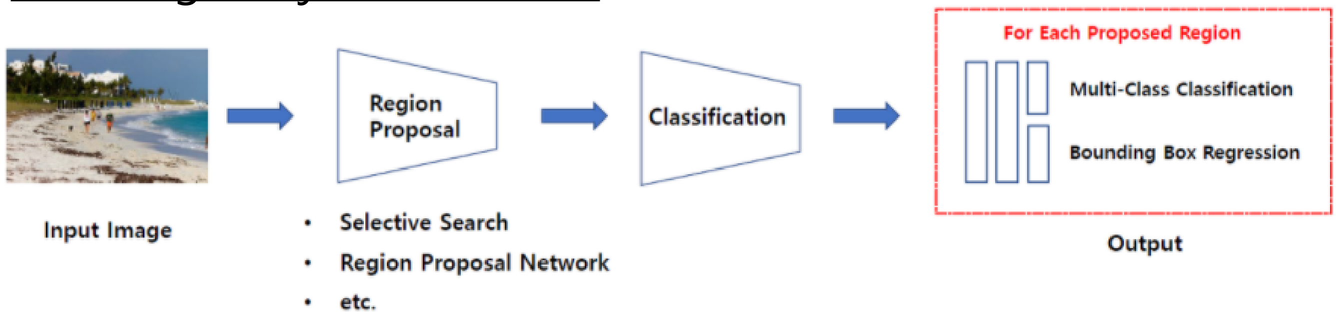
vs

one-stage object detection



- localization & classification in same step
- relatively low accuracy, high speed
- e.g. Yolo, SSD…

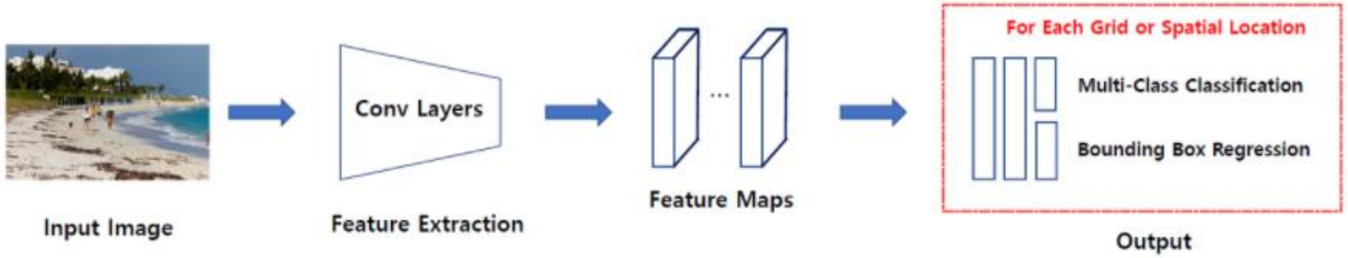
two-stage object detection



- localization & classification in different step
- relatively high accuracy, low speed
- e.g. R-CNN, Fast R-CNN…

vs

one-stage object detection



- localization & classification in same step
- relatively low accuracy, high speed
- e.g. Yolo, SSD…

today's topic

Yolo v1

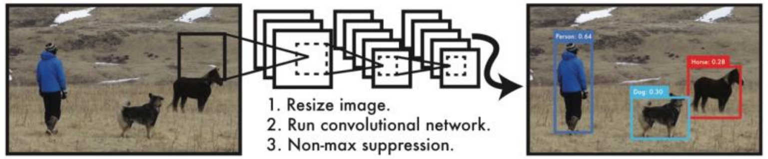
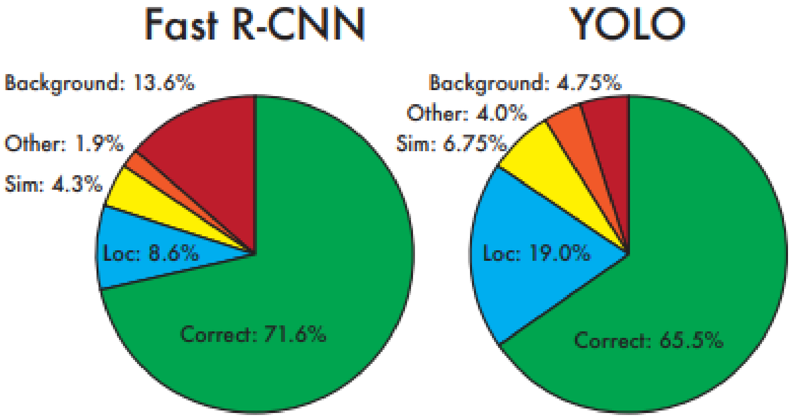


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448×448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

(one-stage object detection)



(low background err)

(performance on artwork)

<u>Yolo)</u>
- One-stage detection

<u>Strength of Yolo)</u>
- Extremely fast : real-time detection (45 fps)
- Low background error than state-of-the-art (fast r-cnn)
- Learns generalizable representations of object

(CVPR, 2016) "You Only Look Once: Unified, Real-Time Object Detection"

S x S grid on input

- divides the input image into an S x S grid
- If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object
- Each grid cell predicts B bounding boxes & confidence score($\Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}$)

bounding boxes+confidence

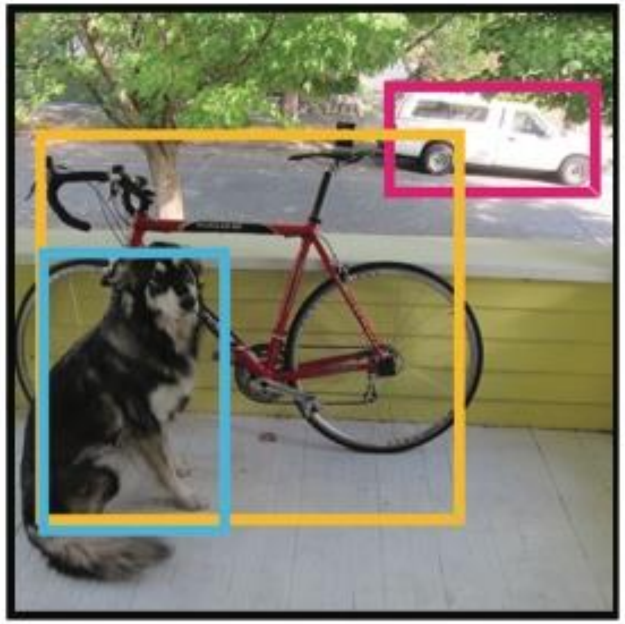
class probability map

- each bounding box predicts
x, y, w, h, confidence($\Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$)

- each grid cell predicts C conditional probabilies $\Pr(\text{Class}_i|\text{Object})$

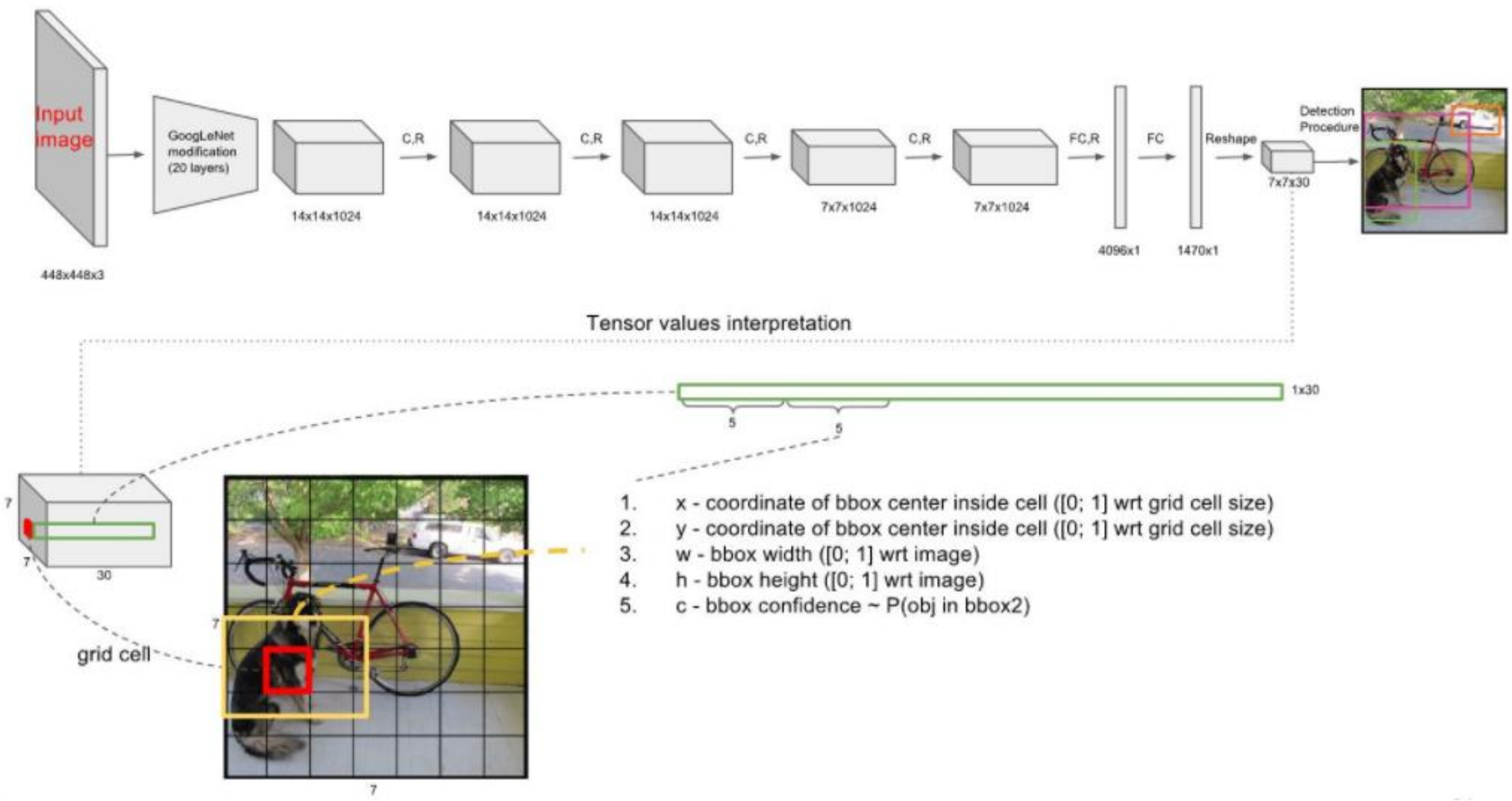
<class-specific confidence score>

$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}$$



final detections

- obtain class-specific confidence scores from each box
- class-specific confidence score encode both the probability of that class appearing in the box
  & how well the predicted box fits the object


Input image
448x448x3
GoogLeNet modification (20 layers)
14x14x1024
C,R
14x14x1024
C,R
14x14x1024
C,R
7x7x1024
C,R
7x7x1024
FC,R
4096x1
FC
1470x1
Reshape
7x7x30
Detection Procedure
Tensor values interpretation
1x30
5
5
7
7
30
grid cell
7
7
1. x - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
2. y - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
3. w - bbox width ([0; 1] wrt image)
4. h - bbox height ([0; 1] wrt image)
5. c - bbox confidence ~ P(obj in bbox2)

Thank you