

# 인공지능

## 08. Transfer Learning

Dept. of Digital Contents



# 1. 전이 학습

---

## ○ 전이학습

- 어떤 도메인에서 수집한 데이터로 학습한 모델을 다른 도메인의 데이터를 인식하는데 활용
- 데이터가 부족하여 높은 성능을 달성하기 어려운 상황에 활용
- 예) cub200-2011 데이터셋
  - 200 종의 새영상, 부류별 훈련 집합은 30장 → 과잉 적합이 일어날 가능성이 큼
  - ImageNet으로 예비 학습된 CNN을 cub200-2011 데이터로 전이
    - ImageNet : 21,841 부류에 대해 1,400만장이상의 데이터

# 1. 전이 학습

---

## ○ 전이학습

### ■ 텐서플로가 제공하는 예비학습 CNN

모델이름	파일크기	1순위 정확률	5순위 정확률	매개변수 개수	총개수(깊이)
Xception	88MB	0.790	0.945	22,910,480	126
VGG16	528MB	0.713	0.901	138,357,544	23
VGG19	549MB	0.713	0.900	143,667,240	26
ResNet50	98MB	0.749	0.921	25,636,712	-
ResNet101	171MB	0.764	0.928	44,707,176	-
ResNet152	232MB	0.766	0.931	60,419,944	-
ResNet50V2	89MB	0.760	0.930	25,613,800	-
ResNet101V2	171MB	0.772	0.938	44,675,560	-
ResNet152V2	232MB	0.780	0.942	60,380,648	-
InceptionV3	92MB	0.779	0.937	23,851,784	159

# 1. 전이 학습

---

## ○ 전이학습

### ■ 텐서플로가 제공하는 예비학습 CNN

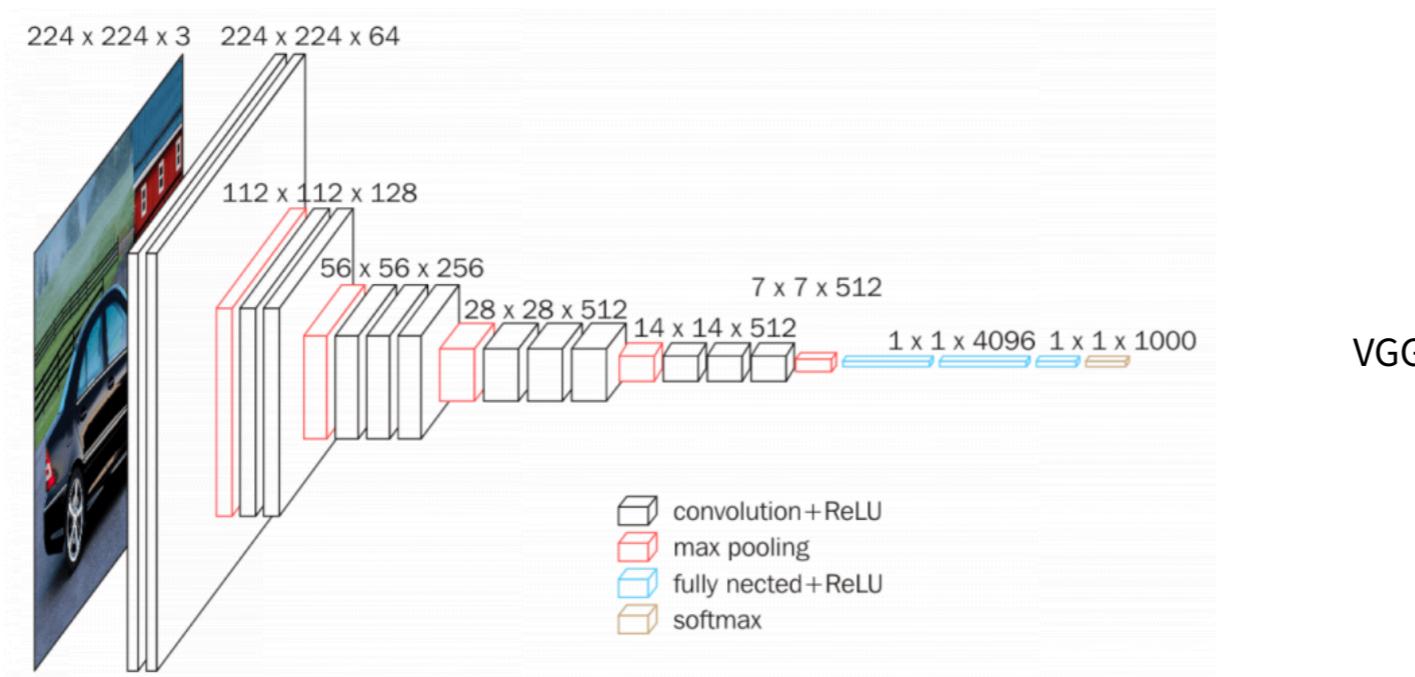
모델이름	파일크기	1순위 정확률	5순위 정확률	매개변수 개수	총개수(깊이)
InceptionResNetV2	215MB	0.803	0.953	55,873,736	572
MobileNet	16MB	0.704	0.895	4,253,864	88
MobileNetV2	14MB	0.713	0.901	3,538,984	88
DenseNet121	33MB	0.750	0.923	8,062,504	121
DenseNet169	57MB	0.762	0.932	14,307,880	169
DenseNet201	80MB	0.773	0.936	20,242,984	201
NASNetMobile	23MB	0.744	0.919	5,326,716	-
NasNetLarge	343MB	0.825	0.960	88,949,818	-

# 1. 전이 학습

## ○ 전이학습

### ■ 텐서플로가 제공하는 예비학습 CNN

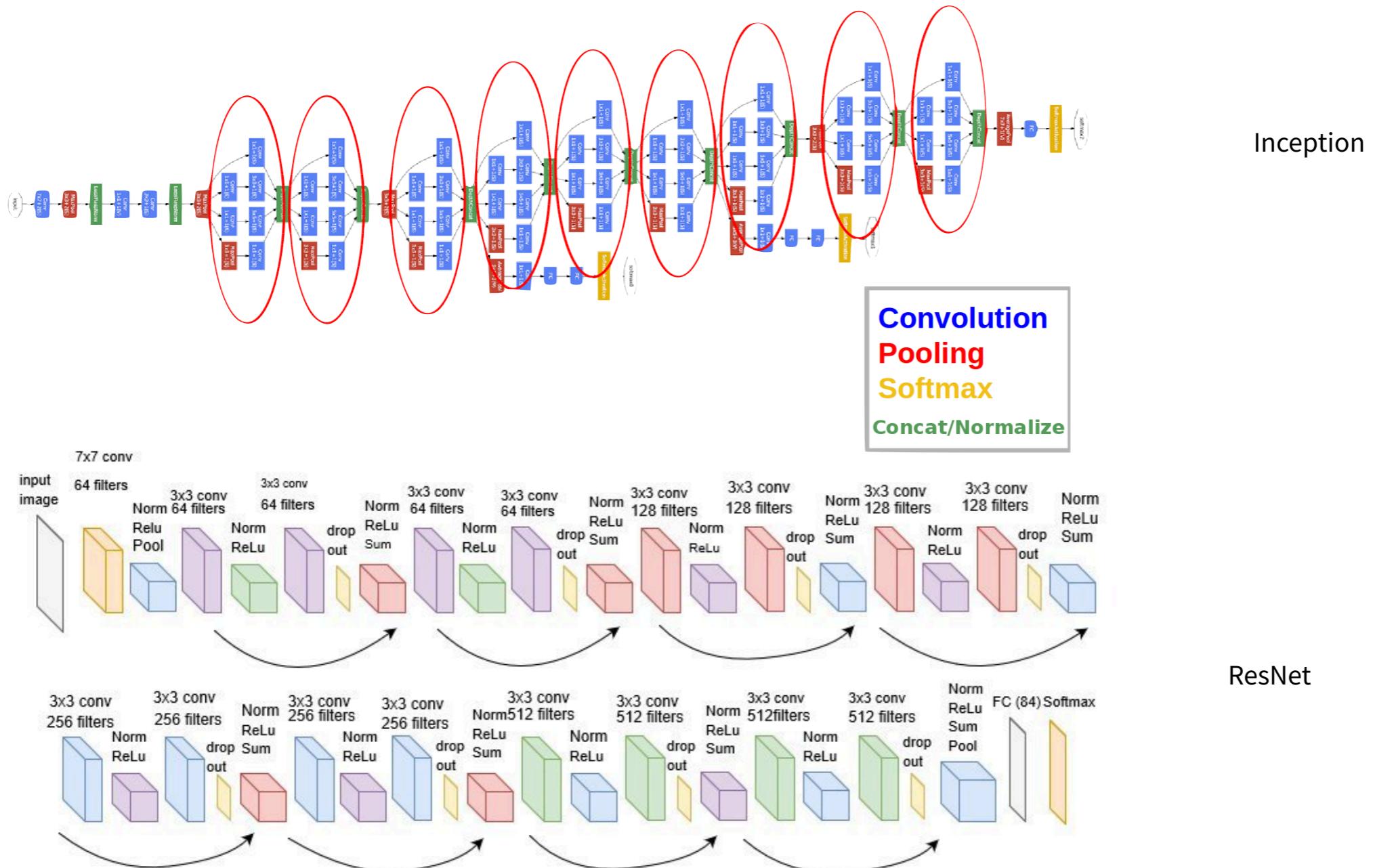
- VGG : 옥스포드 대학교, 2014년 ILSVRC 준우승
- GoogLeNet(Inception) : 2014년 ILSVRC 대회 우승
- ResNet : 마이크로 소프트, 2015년 ILSVRC 대회 3.5% 5순위 오류율(Top-5 error)로 우승



# 1. 전이 학습

## ○ 전이학습

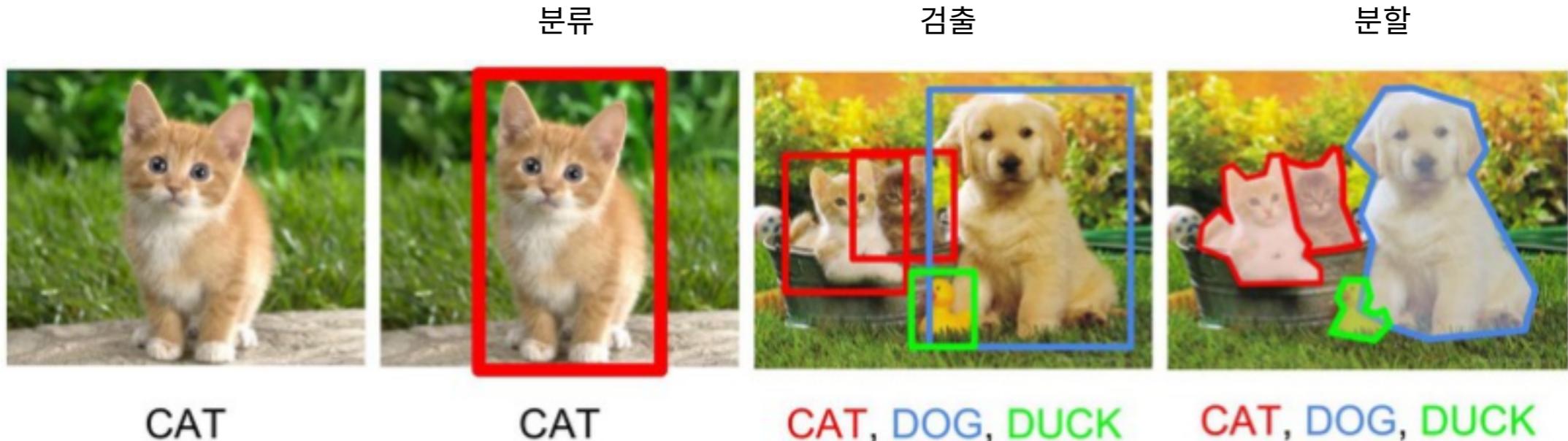
### ■ 텐서플로가 제공하는 예비학습 CNN



## 2. 물체 검출

### ○ 분류, 검출 분할

- 분류(Classification) : 물체의 부류를 알아내는 것
- 검출(Detection) : 물체의 위치를 알아내는 것
- 분할(Segmentation) : 물체가 속하는 화소 수준의 영역을 지정



## 2. 물체 검출

---

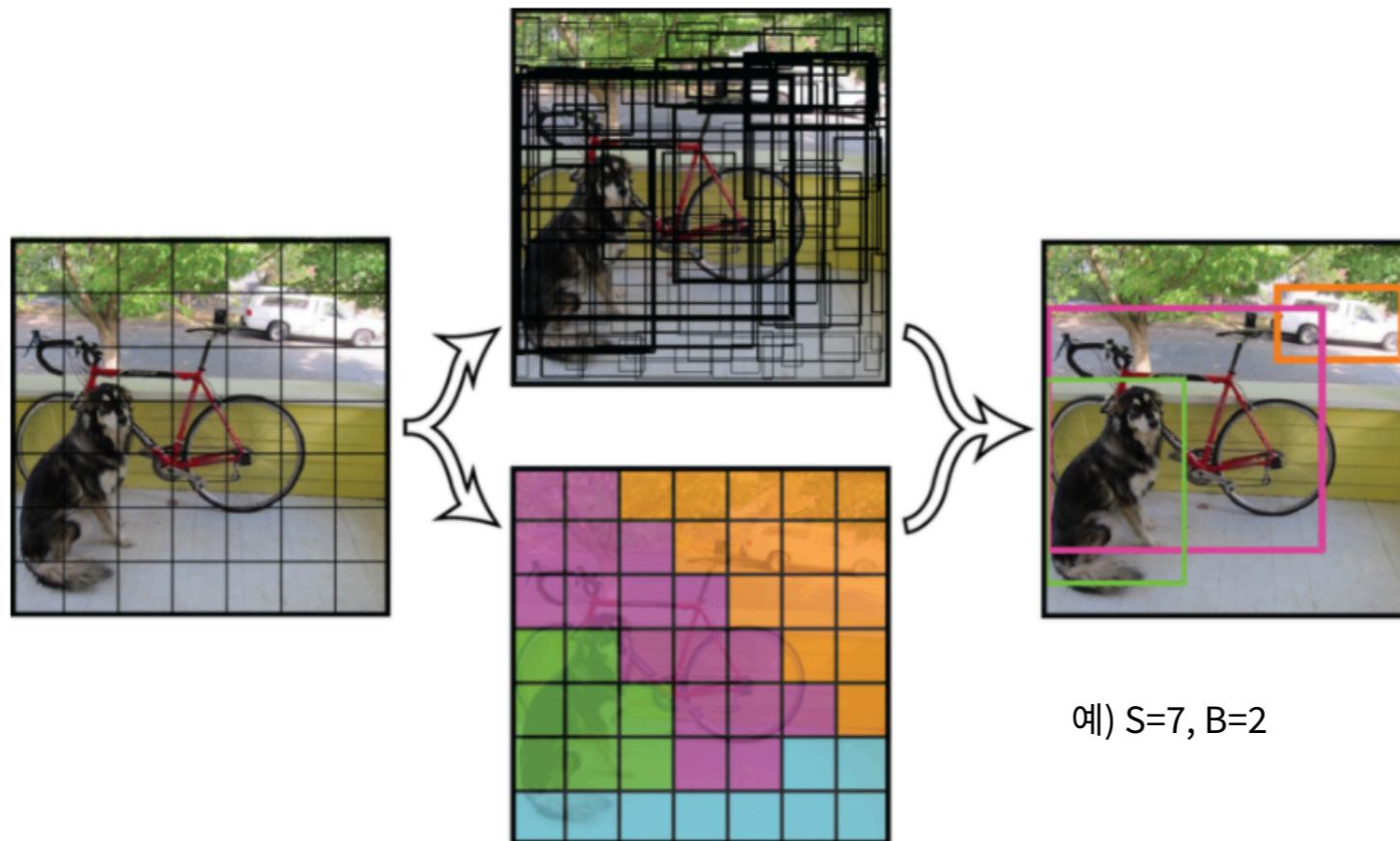
### ○ 물체 검출 모델

- R-CNN(Region with CNN)
  - 설정한 영역을 CNN의 feature 값으로 활용
- Fast R-CNN
- Faster R-CNN
- YOLO(You Only Look Once)
  - R-CNN 보다 정확도는 낮음
  - 초당 30 프레임 이상을 처리하여 실시간 물체 검출에 사용

## 2. 물체 검출

### ○ YOLO

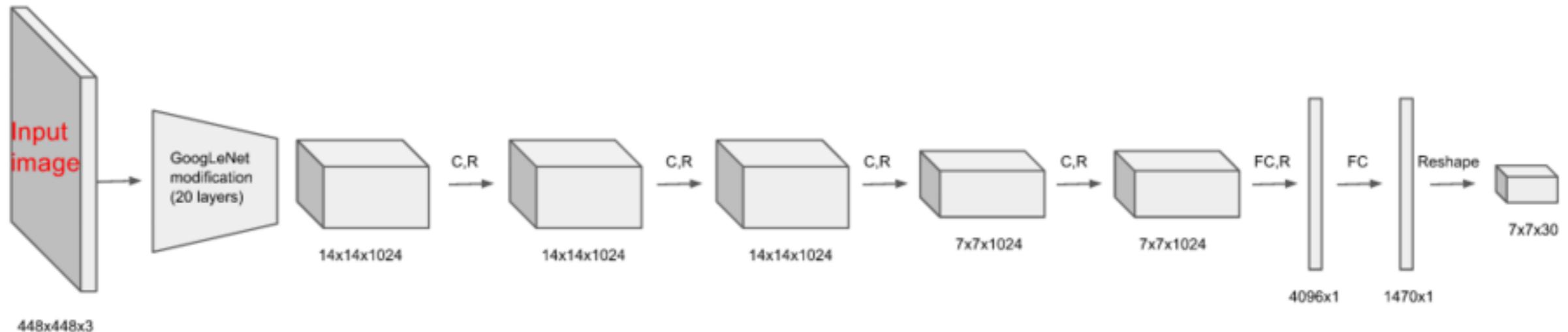
- 경계 박스 위치 찾기와 클래스 분류가 동시에 이루어짐
  - 입력 영상  $\rightarrow S \times S$  격자로 나눔  $\rightarrow$  각 grid에 B개의 경계 박스 생성
  - bounding box ( $x, y, w, h, o$ ) :  $x, y$  중심점,  $w, h$  너비와 높이,  $o$ 는 물체일 가능성



## 2. 물체 검출

### ○ YOLO

- 네트워크 구조 : GoogLeNet의 변형을 특징 추출기로 사용



- YOLOv3

- 2018년 YOLO 성능 개선
- $14 \times 14, 28 \times 28, 56 \times 56$ 의 3개 스케일로 물체 검출 능력 향상
- 각 grid는 3개의 bounding box를 추정

## 2. 물체 검출

---

### ○ YOLO Programming

- MS COCO 데이터셋
  - <https://github.com/pjreddie/darknet/blob/master/data/coco.names> 다운로드
- yolov3.weights, yolov3.cfg 다운로드
  - yolo3.weights : <https://pjreddie.com/darknet/yolo>
  - yolo3.cfg : <https://github.com/pjreddie/darknet/blob/master/cfg>
  - 다운로드 파일들을 구글 드라이브로 업로드

## 2. 물체 검출

### ○ YOLO Programming

The screenshot shows a Google Colab notebook interface. The title bar says 'YOLO.ipynb'. The menu bar includes '파일', '수정', '보기', '삽입', '런타임', '도구', '도움말', and '모든 변경사항이 저장됨'. On the left, there are icons for file operations: a list icon, a search icon, a copy/paste icon, and a folder icon. The main code cell contains the following Python code:

```
[ ] import numpy as np
import cv2
from google.colab.patches import cv2_imshow

classes = []

f = open('drive/MyDrive/Colab Notebooks/coco.names')
classes = [line.strip() for line in f.readlines()]
colors = np.random.uniform(0, 255, size=(len(classes), 3))
img = cv2.imread('drive/MyDrive/Colab Notebooks/yolo_test.jpg')
height, width, channels = img.shape
```

## 2. 물체 검출

---

### ○ YOLO Programming

```
[ ] # OpenCV의 dnn(deep neural network)를 사용
# blob: opencv dnn에 입력하는 이미지 형식
blob = cv2.dnn.blobFromImage(img, 1.0/255, (448, 448), (0, 0, 0),
                             swapRB=True, crop=False)

yolo_model = cv2.dnn.readNet('drive/MyDrive/Colab Notebooks/yolov3.cfg',
                            'drive/MyDrive/Colab Notebooks/yolov3.weights')
layer_names = yolo_model.getLayerNames()

# yolo_83, yolo_94, yolo_106번의 특징 맵 레이어 선택
out_layers = [layer_names[i-1] for i in yolo_model.getUnconnectedOutLayers()]

# 순방향 추론으로 3개의 특징 맵 획득
yolo_model.setInput(blob)
output3 = yolo_model.forward(out_layers)
```

## 2. 물체 검출

### ○ YOLO Programming

```
[ ] class_ids, confidences, boxes = [], [], []
for output in output3:
    for vec85 in output: # output은 85D Tensor (x, y, w, h, o, p1, p2, ..., p80)
        scores = vec85[5:]
        class_id = np.argmax(scores)
        confidence = scores[class_id]
        if confidence > 0.5: # 신뢰도가 50%이상일 때
            centerx, centery = int(vec85[0]*width), int(vec85[1]*height)
            w, h = int(vec85[2]*width), int(vec85[3]*height)
            x, y = int(centerx - w/2), int(centery - h/2)
            boxes.append([x, y, w, h])

        confidences.append(float(confidence))
        class_ids.append(class_id)

# 비최대 억제(Non-Maximum Supresion) 알고리즘
# confidence가 50%이상인 box들 중 40%이상 겹치는 박스는 가장 큰 것을 제외하고 삭제
indices = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)
```

## 2. 물체 검출

---

### ○ YOLO Programming

```
▶ for i in range(len(boxes)):
    if i in indices:
        x, y, w, h = boxes[i]
        text = str(classes[class_ids[i]]) + '%.3f' % confidences[i]
        cv2.rectangle(img, (x, y), (x+w, y+h), colors[class_ids[i]], 2)
        cv2.putText(img, text, (x, y-5), cv2.FONT_HERSHEY_PLAIN, 1.2,
                   colors[class_ids[i]], 1)

cv2_imshow(img)
```

## 2. 물체 검출

### ○ YOLO Programming

