

ChatGPT와 함께하는 딥러닝 5일 완성

CNN 모델 심화 및 CIFAR-10 분류 실습

3일차



서울대학교 평생교육원
Extension College Seoul National University

Topics

- 다양한 CNN 모델 구조 및 특징 소개
- 커널 이니셜라이저, 배치 정규화 등
- 오버피팅 문제와 해결 방법 학습
- 이미지 데이터 증강 기법 적용
- CIFAR-10 데이터셋을 활용한 CNN 모델 설계 실습
- ChatGPT로 CNN 구조 개선 연습

PIL 라이브러리

- Stands for Python Imaging Library 파이썬 이미징 라이브러리의 약자
- Image processing libraries for the Python programming language.
Python 프로그래밍 언어를 위한 이미지 처리 라이브러리



메소드, 속성 체크

- `dir()`: List all the attributes (methods and properties) of an object. 개체의 모든 속성 (메서드 및 속성)을 나열

```
import PIL  
  
dir(PIL)
```

```
from PIL import Image  
  
dir(Image)
```

```
['ADAPTIVE', 'AFFINE', 'BICUBIC', 'BILINEAR', 'BOX', 'Callable', 'DECODERS', 'DEFAULT_STRATEGY',
```

```
 'BmpImagePlugin',  
 'ExifTags',  
 'GifImagePlugin',  
 'GimpGradientFile',  
 'GimpPaletteFile',  
 'Image',  
 'ImageChops',  
 'ImageColor',  
 'ImageDraw',  
 'ImageFile',  
 'ImageMode',  
 'ImageOps',  
 'ImagePalette',  
 'ImageSequence',  
 'JpegImagePlugin',  
 'JpegPresets',  
 'PaletteFile',  
 'PngImagePlugin',  
 'PpmImagePlugin',  
 'TiffImagePlugin',  
 'TiffTags',  
 'UnidentifiedImageError',  
 '__builtins__',  
 '__cached__',  
 '__doc__',  
 '__file__',  
 '__loader__',  
 '__name__',  
 '__package__',  
 '__path__',  
 '__spec__',  
 '__version__',  
 '_binary',  
 '_deprecate',  
 '_imaging',  
 '_plugins',  
 '_util']
```

타입 체크

타입 체크

```
from PIL import Image  
print(type(Image)) #module  
print(type(Image.open)) #function  
print(type(Image.open('/content/coffee.jpg')) #PIL 객체
```

```
<class 'module'>  
<class 'function'>  
<class 'PIL.JpegImagePlugin.JpegImageFile'>
```

이미지 열기 및 저장

```
from PIL import Image
```

이미지 열기

```
img = Image.open('example.jpg')
```

이미지 표시

```
import matplotlib.pyplot as plt
```

```
plt.imshow(img)
```

```
plt.show()
```

jupyter notebook에
서는 img만으로도 이미
지가 출력됨

이미지 저장

```
img.save('output.png')
```

프로그램 종료

```
img.close()
```



이미지 확인

이미지 사이즈 확인

```
img.size # (width, height)
```

```
img.width, img.height
```

이미지 포맷 확인

```
img.format #JPG
```

이미지 모드 확인

```
img.mode #RGB
```

이미지 조작

이미지 크기 변경 (확대, 축소)

```
resized_img = img.resize((width, height))
```

이미지 회전

```
rotated_img = img.rotate(90, expand=True)
```

이미지 자르기

```
cropped_img = img.crop((left, top, right, bottom))
```



이미지 위에 박스 표시

```
from PIL import ImageDraw
```

```
draw = ImageDraw.Draw(img)
```

```
draw.rectangle([410, 50, 970, 520], outline='red')
```

```
display(img)
```



상하반전



▪ 이미지를 위에서 아래로 뒤집음

```
img.transpose(Image.FLIP_TOP_BOTTOM)
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
plt.imshow(np.array(im)[:, ::-1, :])
```

```
plt.show()
```

이미지의 가로축을 따라
있는 모든 요소 포함

넘파이 배열 자르기

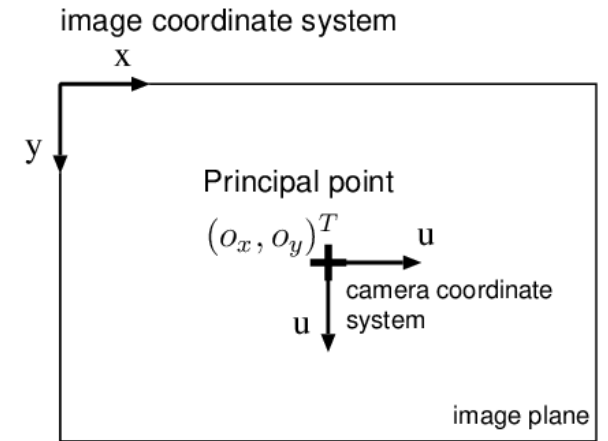
이미지의 세로축을 따
라 요소의 순서를 반대
로 함

색상 채널이 변경되지
않음을 의미

기타 반전

Seven options that you can pass as arguments to `.transpose()`. 반전에 인수로 전달할 수 있는 7가지 옵션

- `Image.FLIP_LEFT_RIGHT`: 이미지를 왼쪽에서 오른쪽으로 뒤집어 거울 이미지를 만듦
- `Image.FLIP_TOP_BOTTOM`: 이미지를 위에서 아래로 뒤집음
- `Image.ROTATE_90`: 이미지를 시계 반대 방향으로 90도 회전
- `Image.ROTATE_180`: 이미지를 180도 회전
- `Image.ROTATE_270`: 이미지를 시계 반대 방향으로 270도 회전. 시계방향으로 90도 회전과 같음
- `Image.TRANSPOSE`: 왼쪽 위 픽셀을 원점으로 행열전치. 전치된 이미지에서 왼쪽 위 픽셀은 원본 이미지와 동일
- `Image.TRANSVERSE`: 왼쪽 하단 픽셀을 원점으로 행열전치. 왼쪽 하단 픽셀은 원본 버전과 수정된 버전 사이에 고정된 상태로 유지



색상 변환

흑백이미지

```
gray_img = img.convert("L") # Grayscale  
gray_img.getbands() # ('L',)  
display(gray_img)
```

RGB 칼라 이미지

```
rgb_img = img.convert("RGB") # 디지털  
디스플레이를 위한 RGB  
rgb_img.getbands() # ('R', 'G', 'B')  
display(rgb_img)
```



컬러 분리

```
red, green, blue = img.split()  
img_merged = Image.merge("RGB", (blue, green, red))  
img_merged
```



이미지 조작

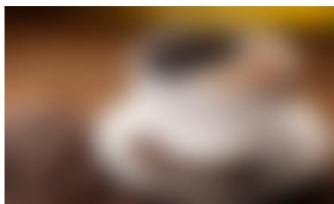
- 밝게/어둡게 (factor>1 더 밝아짐.
factor<1 덜 밝아짐)

```
from PIL import ImageEnhance
enhancer =
ImageEnhance.Brightness(img)
brightened_img =
enhancer.enhance(1.5)  # Increase
brightness by 50%
```



- 블러링

```
from PIL import ImageFilter
img.filter(ImageFilter.BLUR)
img.filter(ImageFilter.BoxBlur(5))
img.filter(ImageFilter.BoxBlur(20))
img.filter(ImageFilter.GaussianBlur(20))
```



- 날카롭게/부드럽게

```
img.filter(ImageFilter.SHARPEN)
img.filter(ImageFilter.SMOOTH)
```



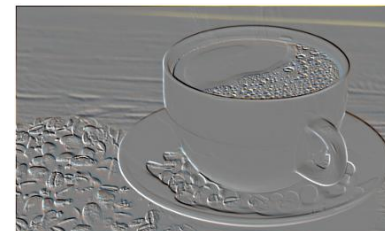
- 경계감지

```
img.filter(ImageFilter.FIND_
EDGES)
img.filter(ImageFilter.EDGE_
ENHANCE)
```



- 엠보싱

```
img_gray_smooth.filter(Image
Filter.EMBOSS)
```



PIL Image to Numpy Array

```
# PIL 이미지를 NumPy 배열로 변환
```

```
x = np.array(img1)
```

```
print(x.shape)
```

```
x_2 = np.asarray(img1)
```

```
print(x_2.shape)
```



```
[[[ 16   2   1]
  [ 15   2   1]
  [ 15   1   0]
  ...
  [168 127   1]
  [167 127   3]
  [167 127   3]]

[[[ 17   3   2]
  [ 17   2   2]
  [ 17   2   2]
  ...
  [168 127   1]
  [167 127   3]
  [167 127   3]]
```


Numpy Array to PIL Image

```
# NumPy 배열을 PIL 이미지로 변환
```

```
img_2 = Image.fromarray(x)
```

```
img_2
```

```
[[[ 16  2  1]
   [ 15  2  1]
   [ 15  1  0]
   ...
   [168 127  1]
   [167 127  3]
   [167 127  3]]

 [[ 17  3  2]
   [ 17  2  2]
   [ 17  2  2]
   ...
   [168 127  1]
   [167 127  3]
   [167 127  3]]
```



Exercise #3

- Predict with test data. 테스트 데이터 예측
 - Crop, resize, FLIP_LEFT_RIGHT
 - Array, negative image, reshape, normalization

테스트 데이터 준비

테스트 데이터 가져오기

```
from PIL import Image

ankleboot =
Image.open('/content/ankleboot.jpg').conve
rt('L')
```

사이즈 확인

```
ankleboot.size
```

이미지 자르고 사이즈 변환, 좌우반전

```
ankleboot = ankleboot.crop((100, 400, 600,
900))

ankleboot = ankleboot.resize((28,28))

ankleboot =
ankleboot.transpose(Image.FLIP_LEFT_RIGHT)
```

배열변환

```
ankleboot =
np.asarray(ankleboot)

plt.imshow(ankleboot)

plt.show()
```

이미지 반전

```
ankleboot = 255-ankleboot

plt.imshow(ankleboot)

plt.show()
```

테스트 데이터 전처리 및 예측

이미지 전처리

```
ankleboot = ankleboot.reshape(1,28,28,1)
```

```
ankleboot = ankleboot/255.0
```

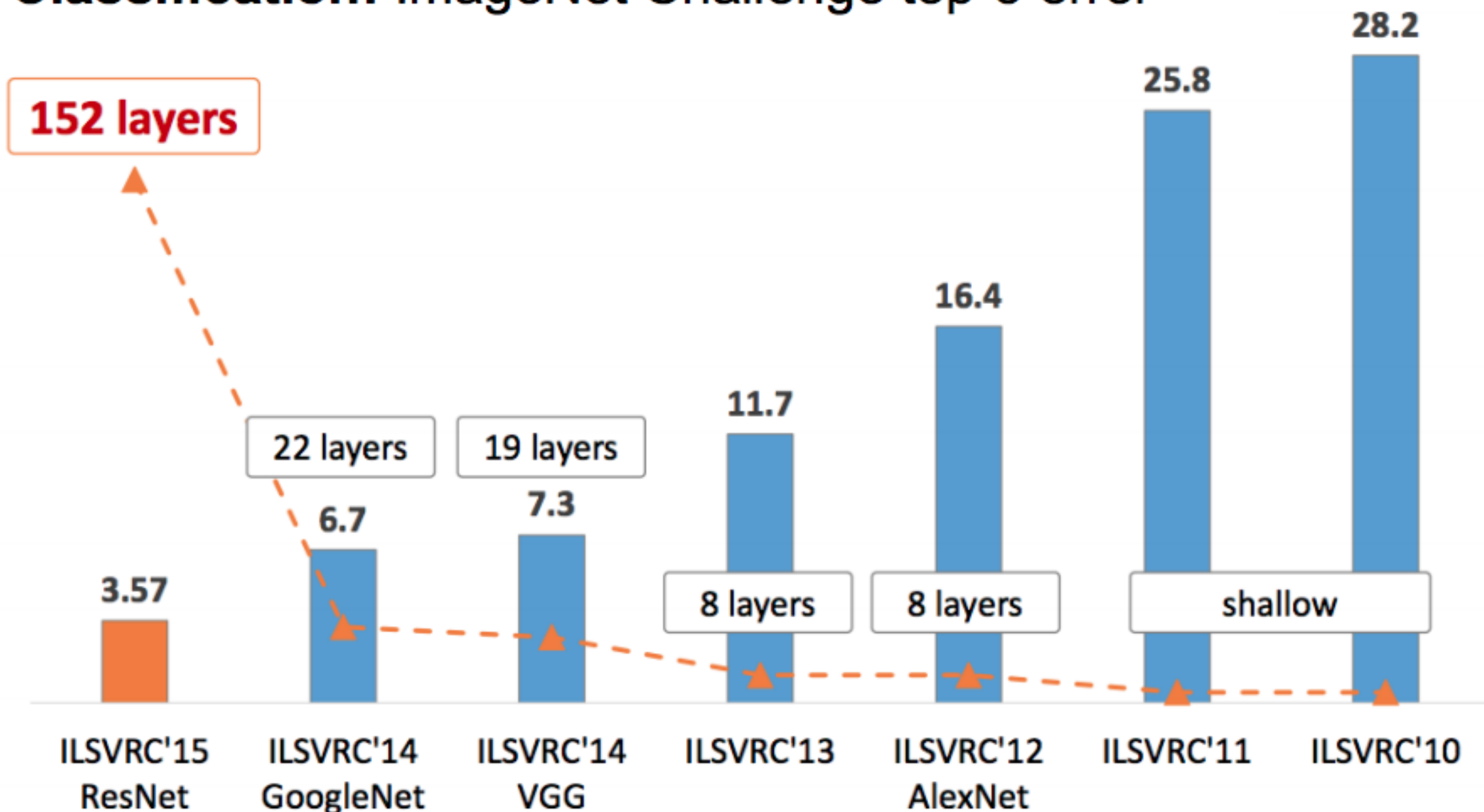
테스트 데이터로 예측

```
label_names[np.argmax(model.predict(ankleboot), axis=1)[0]]
```

CNN Architectures 아키텍처 구조

- Popular CNN architectures won in ILSVRC (ImageNet Large Scale Vision Recognition Challenge) competitions. ILSVRC 대회에서 우승했던 인기 있는 CNN 아키텍처
 - LeNet-5
 - AlexNet
 - VGGNet
 - GoogLeNet
 - ResNet
 - Inception
- Participants are provided with 1.4 millions of images. 참가자에게는 140만 개의 이미지가 제공

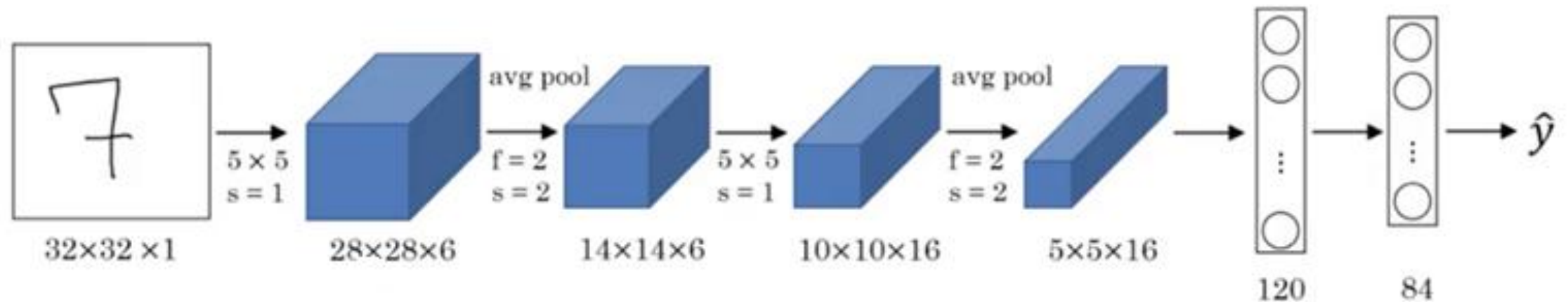
Classification: ImageNet Challenge top-5 error



LeNet-5 리넷

- Classical neural network architecture successfully used on MNIST patterns. 손글씨 문제에 성공적으로 사용됐던 고전적 신경망 아키텍처

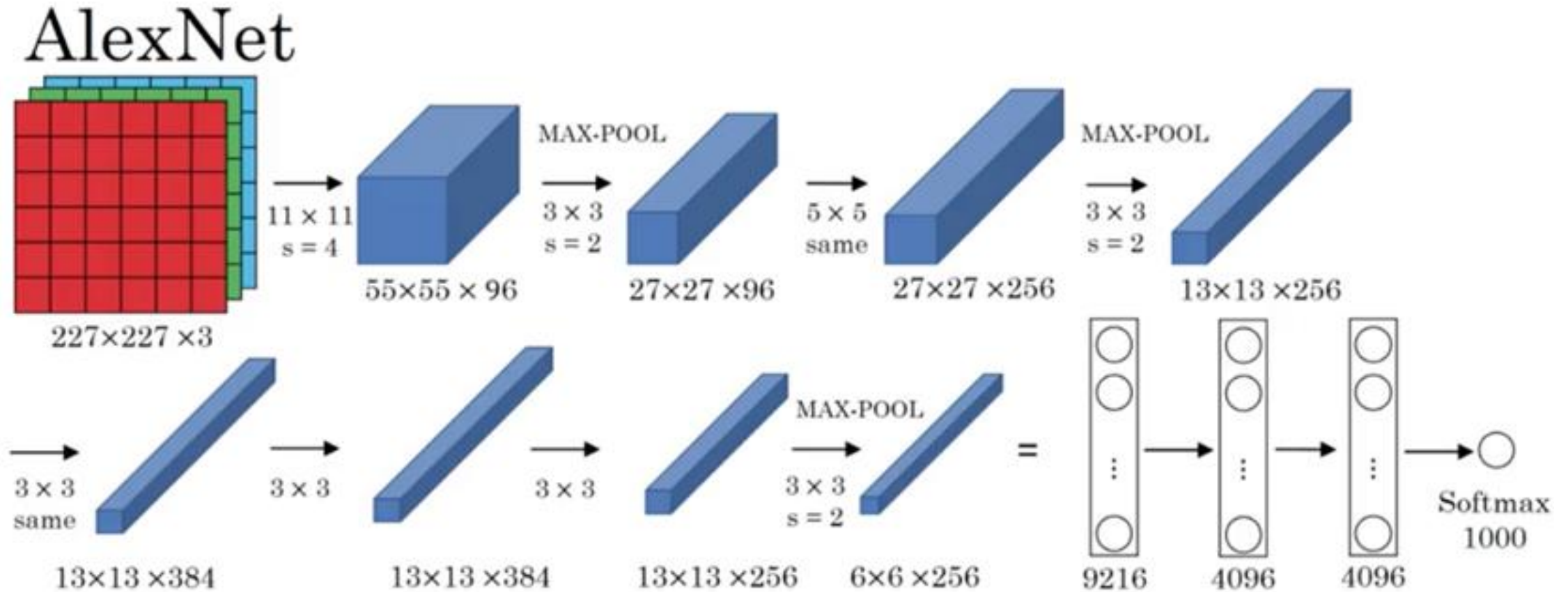
LeNet - 5



By Yann LeCun, Leon Bottou, Yosuha Bengio and Patrick Haffner (1998)

AlexNet 알렉스넷

Error - 16.4
8 layers

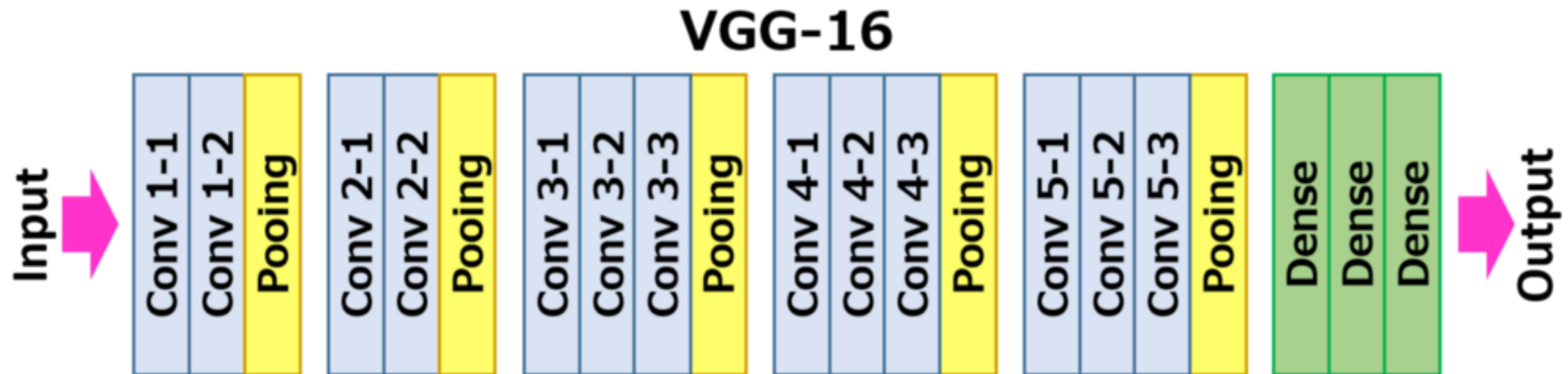


VGG-16

224x224 pixels colored
images 사용

Error - 7.3
19 layers

- A simpler architecture model with less hyper parameters. 하이퍼 파라미터가 적은 단순한 아키텍처 모델
- Trained using more than 1 million images from the ImageNet database and can classify up to 1000 objects. 백만장 이상 이미지를 훈련시켜서 1000개 사물을 분류

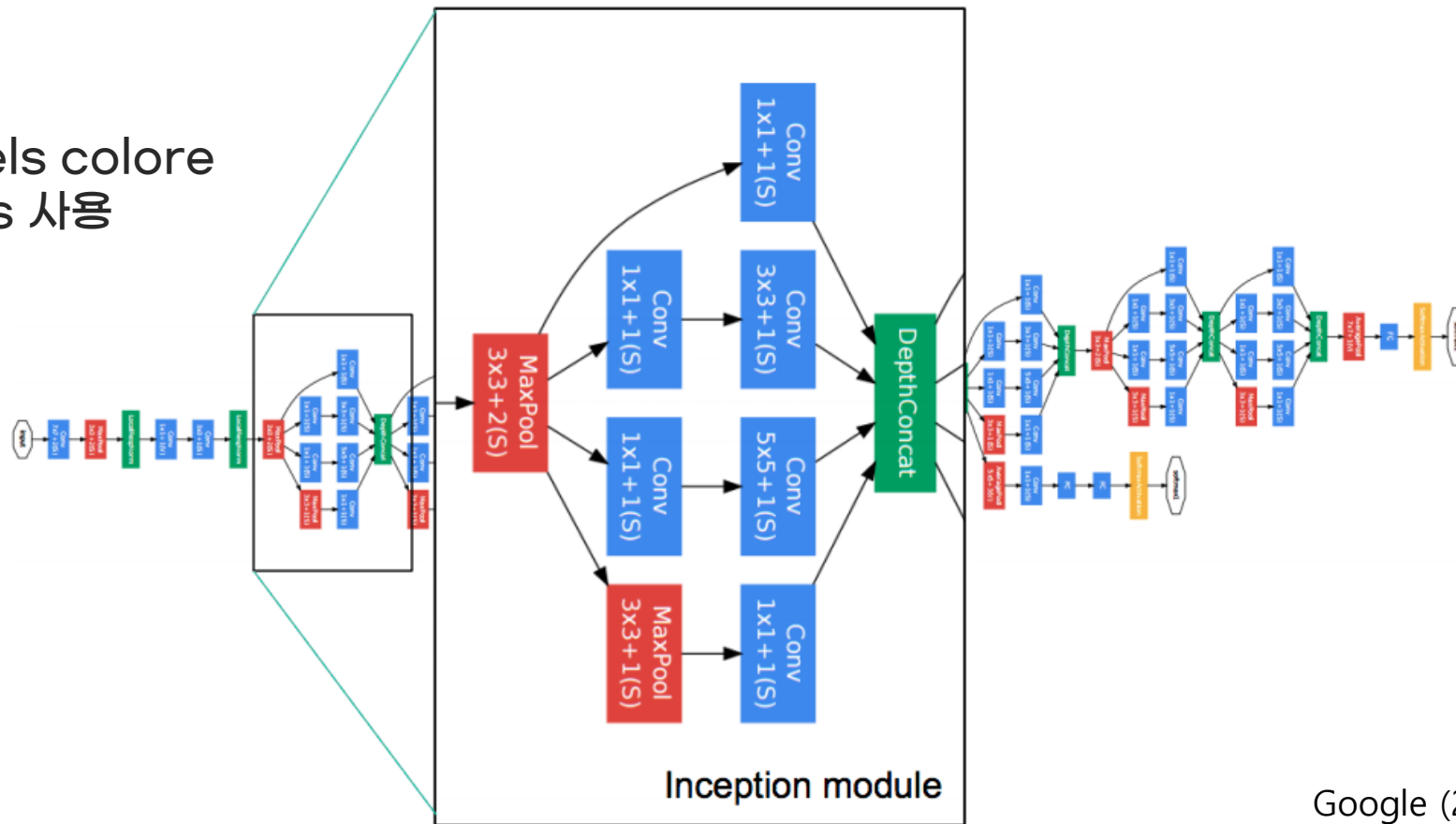


GoogLeNet

Error - 6.7
22 layers

- Also known as Inception Model. 인셉션 모델이라고도 함

299x299 pixels colored images 사용

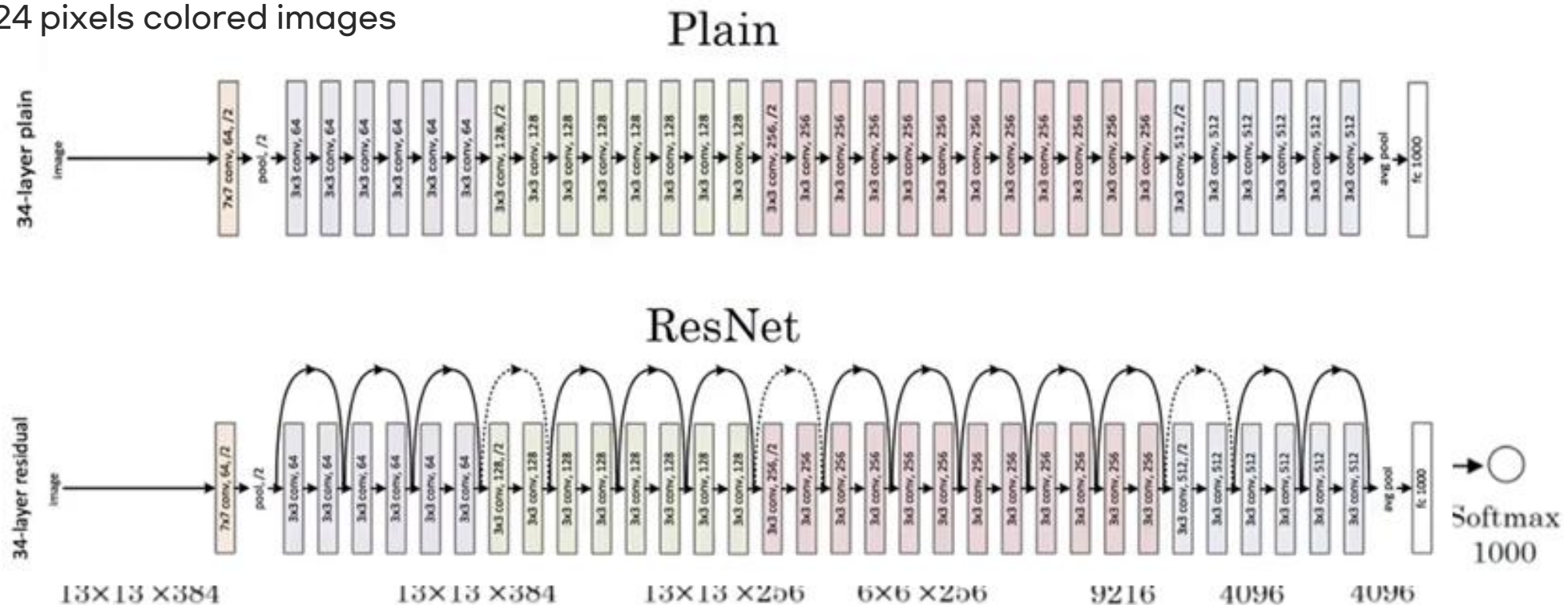


Google (2014)

ResNet (Residual Neural Network) Error - 3.57

- Introduced a concept called “skip connections.” 연결 건너뛰기라는 개념을 도입

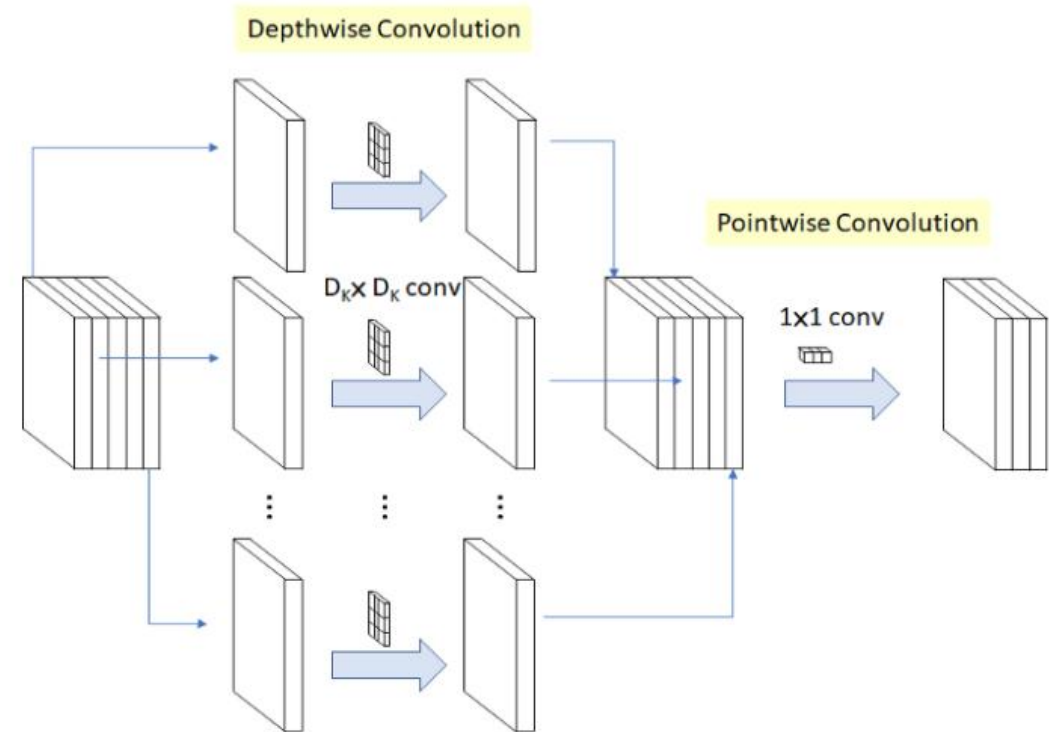
224x224 pixels colored images



MobileNet 모바일넷

- A type of convolutional neural network designed for mobile and embedded vision applications. 핸드폰이나 임베디드 시스템 같이 저용량 메모리환경에 딥러닝을 적용하기 위해 경량화된 합성신경망 모델의 종류
- Uses the idea of Depth convolution and point convolution which is different from the normal convolution as done by normal CNNs. 일반 CNN에서 수행하는 일반 컨볼루션과 다른 depth convolution과 point convolution의 아이디어를 사용

Depthwise separable convolution을 활용하여 모델을 경량화하고, Depthwise Convolution 이후에 Pointwise Convolution을 결합



MobileNet Body Architecture 모바일넷 아키텍처

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32 \text{ dw}$	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64 \text{ dw}$	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128 \text{ dw}$	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128 \text{ dw}$	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256 \text{ dw}$	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256 \text{ dw}$	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
5×	Conv dw / s1	$3 \times 3 \times 512 \text{ dw}$
	Conv / s1	$1 \times 1 \times 512 \times 512$
	Conv dw / s2	$3 \times 3 \times 512 \text{ dw}$
	Conv / s1	$1 \times 1 \times 512 \times 1024$
	Conv dw / s2	$3 \times 3 \times 1024 \text{ dw}$
	Conv / s1	$1 \times 1 \times 1024 \times 1024$
	Avg Pool / s1	Pool 7×7
	FC / s1	1024×1000
	Softmax / s1	Classifier

Table 8. MobileNet Comparison to Popular Models

Model	ImageNet	Million	Million
	Accuracy	Mult-Adds	Parameters
1.0 MobileNet-224	70.6%	569	4.2
GoogleNet	69.8%	1550	6.8
VGG 16	71.5%	15300	138

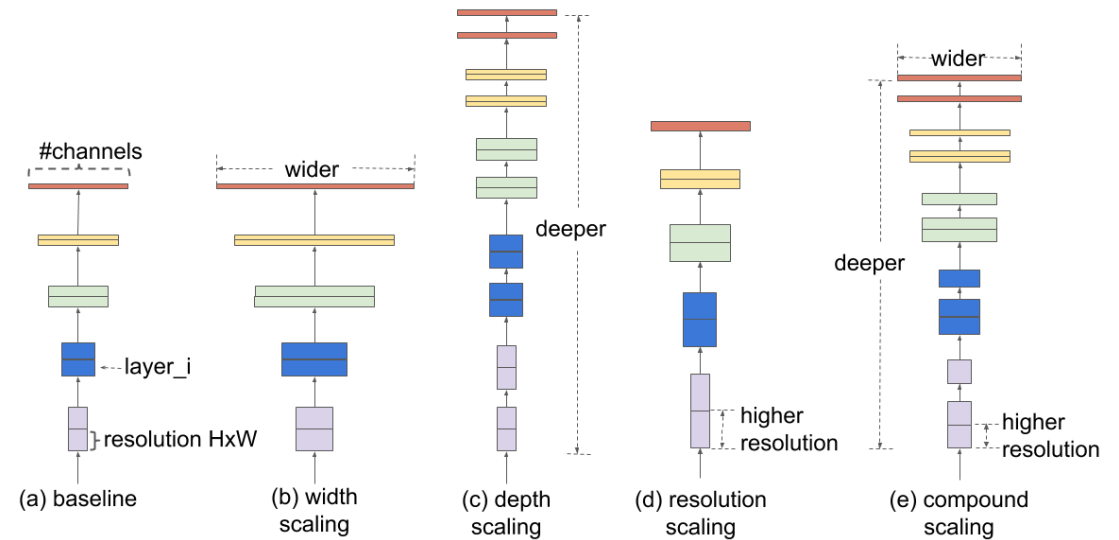
Table 9. Smaller MobileNet Comparison to Popular Models

Model	ImageNet	Million	Million
	Accuracy	Mult-Adds	Parameters
0.50 MobileNet-160	60.2%	76	1.32
Squeezenet	57.5%	1700	1.25
AlexNet	57.2%	720	60

Efficient Net

- A principled method to scale up a CNN to obtain better accuracy and efficiency. 더 나은 정확성과 효율성을 얻기 위해 합성곱 신경망을 스케일업하는 방법
- Uniformly scales width, depth and resolution with a fixed set of scaling coefficients. 고정된 스케일링 계수를 사용하여 폭, 깊이, 해상도를 균일하게 스케일링

Compound Scaling를 사용하여 이전 모델들 보다 훨씬 작으면서도 빠름



Google (2019)

Family of Efficient Net Model

- EfficientNet provides a family of models (B0 to B7) that represents a good combination of efficiency and accuracy on a variety of scales. 다양한 스케일에서 효율성과 정확도의 조합을 나타내는 모델들을 제공

Base model	resolution
EfficientNetB0	224
EfficientNetB1	240
EfficientNetB2	260
EfficientNetB3	300
EfficientNetB4	380
EfficientNetB5	456
EfficientNetB6	528
EfficientNetB7	600

Comparison of Pre-Trained Models

사전 학습된 모델 비교

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.79	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.9	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.76	0.93	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.78	0.942	60,380,648	-
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.75	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
NASNetLarge	343 MB	0.825	0.96	88,949,818	-
EfficientNetB0	29 MB	0.771	0.933	5,330,571	-
EfficientNetB1	31 MB	0.791	0.944	7,856,239	-
EfficientNetB2	36 MB	0.801	0.949	9,177,569	-
EfficientNetB3	48 MB	0.816	0.957	12,320,535	-
EfficientNetB4	75 MB	0.829	0.964	19,466,823	-
EfficientNetB5	118 MB	0.836	0.967	30,562,527	-
EfficientNetB6	166 MB	0.84	0.968	43,265,143	-
EfficientNetB7	256 MB	0.843	0.97	66,658,687	-

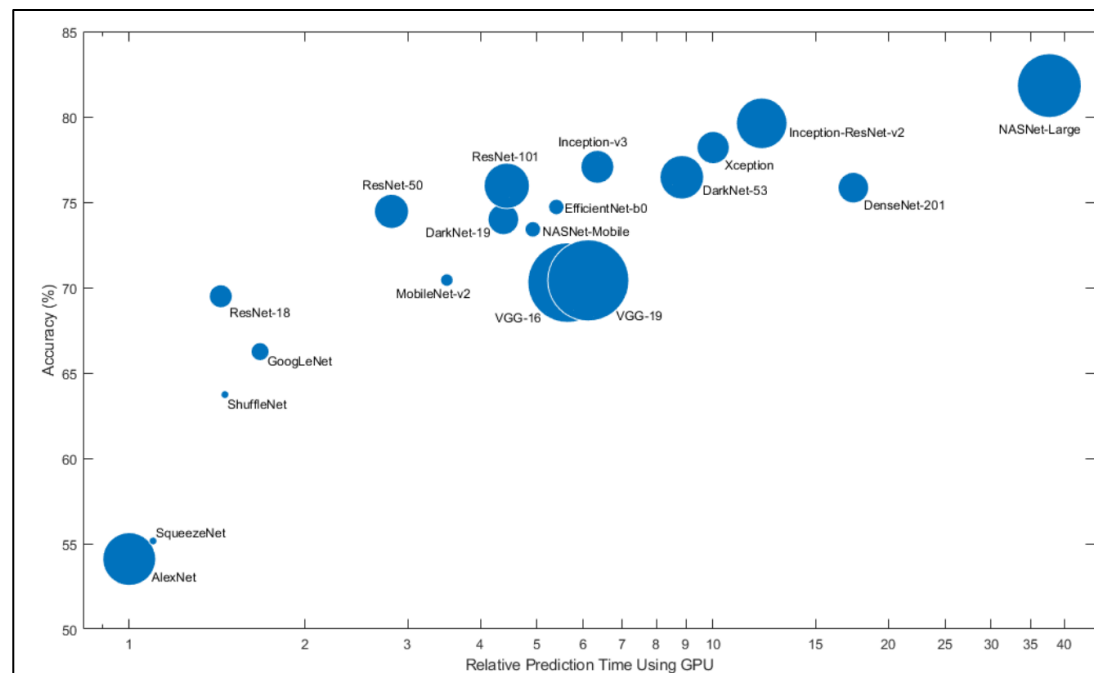
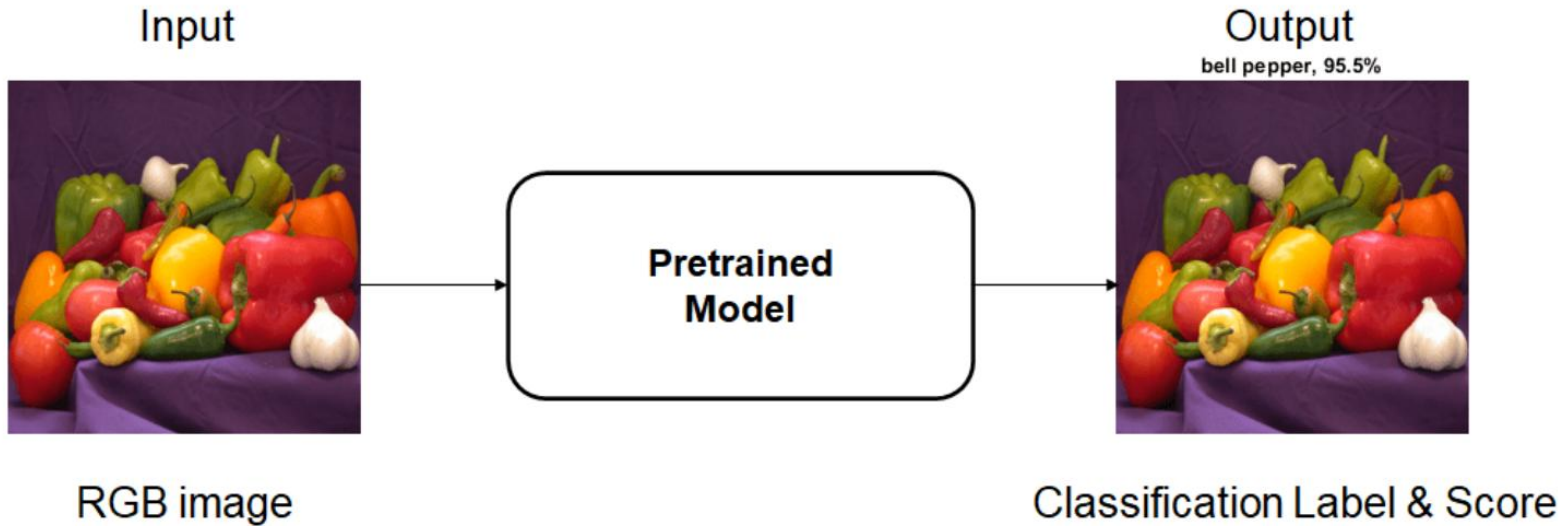


Image Classification 이미지 분류

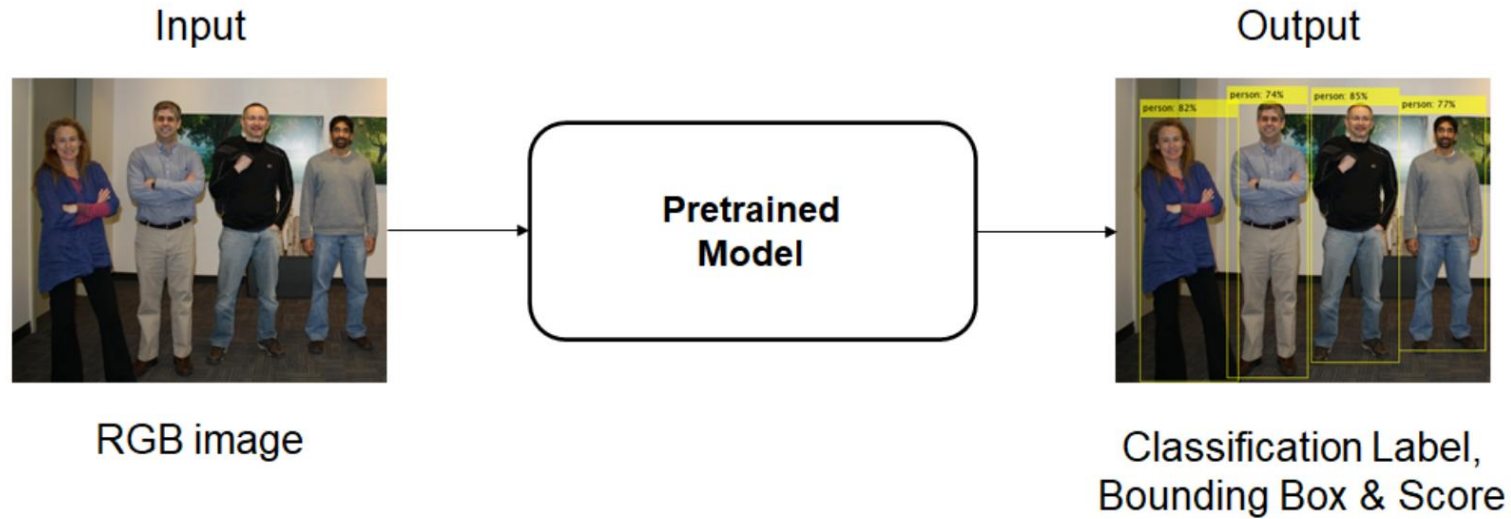


- Extracts features from natural images. 자연 이미지에서 특징을 추출

Network	Size (MB)	Classes	Accuracy %	Location
googlenet¹	27	1000	66.25	Doc GitHub
squeezenet¹	5.2	1000	55.16	Doc
alexnet¹	227	1000	54.10	Doc
resnet18¹	44	1000	69.49	Doc GitHub
resnet50¹	96	1000	74.46	Doc GitHub
resnet101¹	167	1000	75.96	Doc GitHub
mobilenetv2¹	13	1000	70.44	Doc GitHub
vgg16¹	515	1000	70.29	Doc
vgg19¹	535	1000	70.42	Doc
inceptionv3¹	89	1000	77.07	Doc
inceptionresnetv2¹	209	1000	79.62	Doc
xception¹	85	1000	78.20	Doc
darknet19¹	78	1000	74.00	Doc
darknet53¹	155	1000	76.46	Doc
densenet201¹	77	1000	75.85	Doc
shufflenet¹	5.4	1000	63.73	Doc
nasnetmobile¹	20	1000	73.41	Doc
nasnetlarge¹	332	1000	81.83	Doc
efficientnetb0¹	20	1000	74.72	Doc
ConvMixer	7.7	10	-	GitHub
ViT	Large-16 - 1100 Base-16 - 331.4 Small-16 - 84.7 Tiny-16 - 22.2	1000	Large-16 - 85.59 Base-16 - 85.49 Small-16 - 83.73 Tiny-16 - 78.22	Doc

Source: Matlab

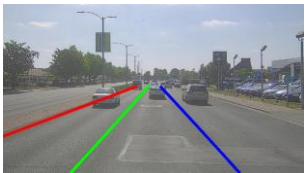



Object Detection 물체 감지



- Locates instances of objects in images or videos. 이미지 또는 비디오에서 개체의 인스턴스를 찾음

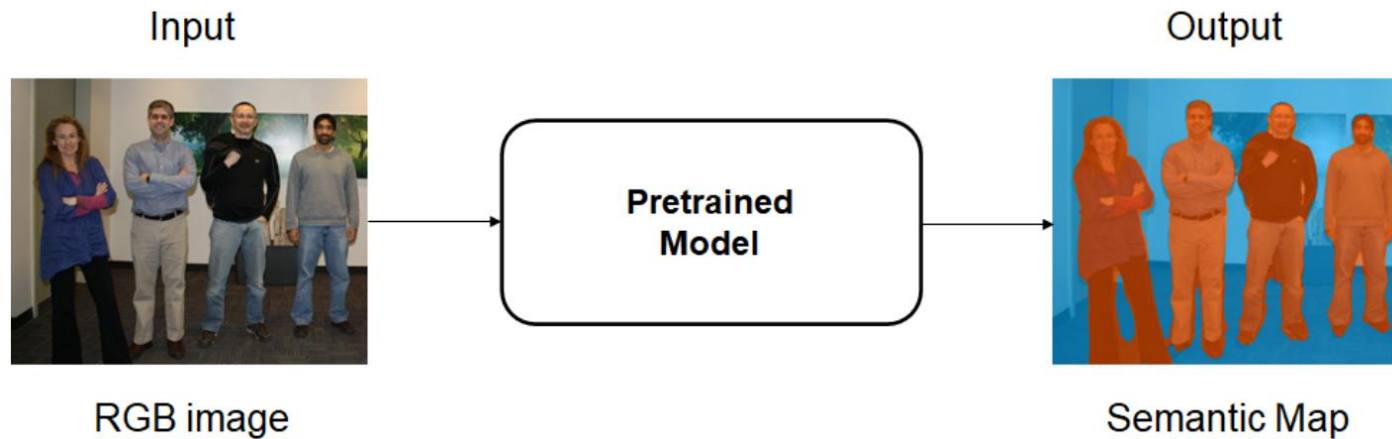
Network	Network variants	Size (MB)	Mean Average Precision (mAP)	Object Classes	Location
EfficientDet-D0	efficientnet	15.9	33.7	80	GitHub
YOLO v8	yolo8n yolo8s yolo8m yolo8l yolo8x	10.7 37.2 85.4 143.3 222.7	37.3 44.9 50.2 52.9 53.9	80	GitHub
YOLOX	YoloX-s YoloX-m YoloX-l	32 90.2 192.9	39.8 45.9 48.6	80	Doc GitHub
YOLO v4	yolov4-coco yolov4-tiny-coco	229 21.5	44.2 19.7	80	Doc GitHub
YOLO v3	darknet53-coco tiny-yolov3-coco	220.4 31.5	34.4 9.3	80	Doc
YOLO v2	darknet19-COCO tiny-yolo_v2-coco	181 40	28.7 10.5	80	Doc GitHub

Application Specific Object Detectors 응용 분야별 물체 감지기

Network	Application	Size (MB)	Location	Example Output
Spatial-CNN	Lane detection	74	GitHub	
RESA	Road Boundary detection	95	GitHub	
Single Shot Detector (SSD)	Vehicle detection	44	Doc	
Faster R-CNN	Vehicle detection	118	Doc	

Semantic Segmentation

의미론적 분할

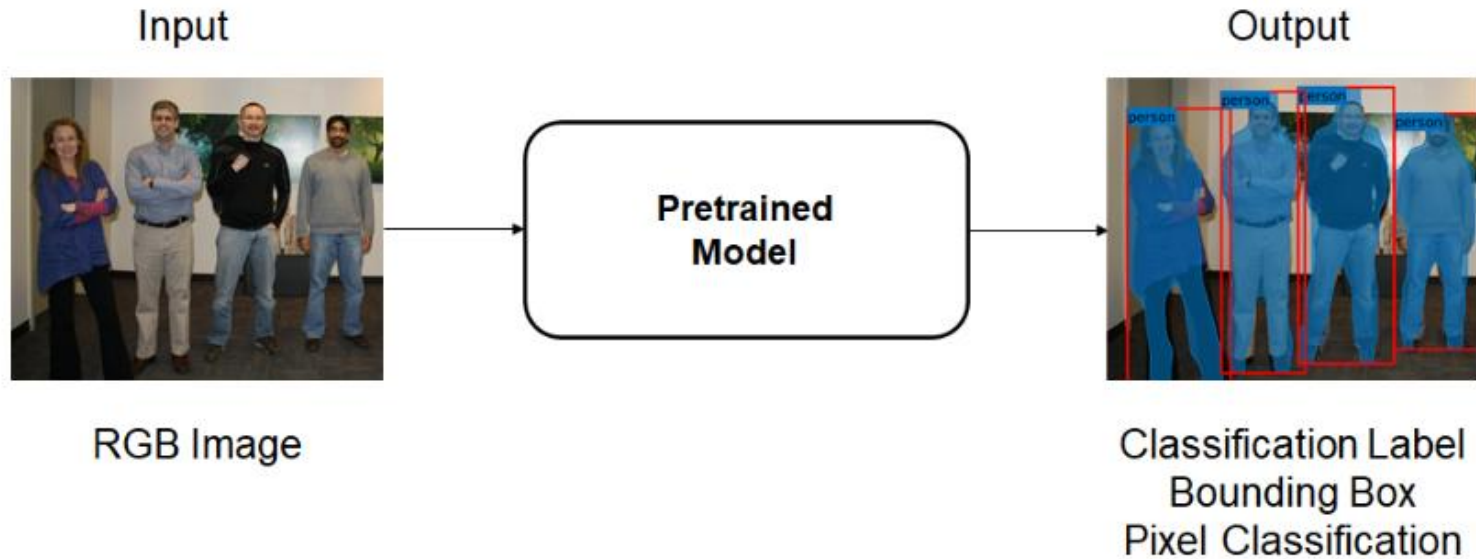


- Describes the process of associating each pixel of an image with a class label, (such as flower, person, road, sky, ocean, or car). 이미지의 각 픽셀을 클래스 레이블 (예: 꽃, 사람, 도로, 하늘, 바다 또는 자동차)과 연결하는 프로세스를 설명

Network	Size (MB)	Mean Accuracy / Application	Object Classes / Example Output	Location
DeepLabv3+	209	0.87	20	GitHub
segmentAnythingModel	358	Zero-shot image segmentation		Doc
U-net	31	Raw Camera Processing		Doc
3-D U-net	56.2	Brain Tumor Segmentation		Doc
AdaptSeg (GAN)	54.4	Model training using 3-D simulation data		Doc

Instance Segmentation

인스턴스 세그멘테이션

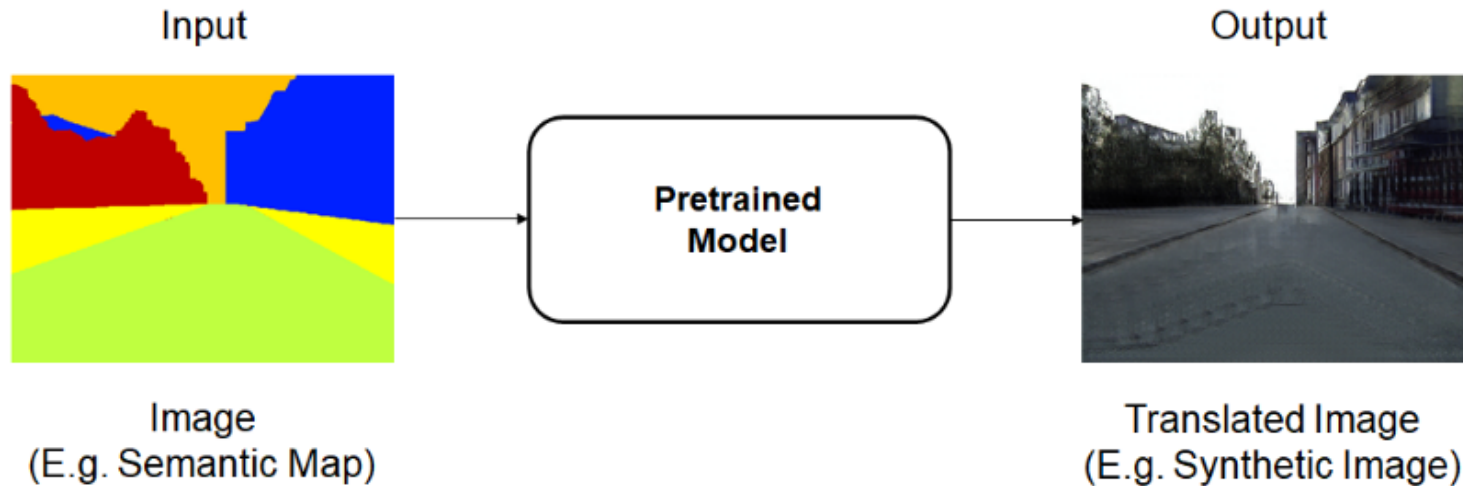


Network	Object Classes	Location
Mask R-CNN	80	Doc Github

Source: Matlab

Generates a segmentation map for each detected instance of an object. 감지된 각 개체 인스턴스에 대한 세그멘테이션 맵을 생성

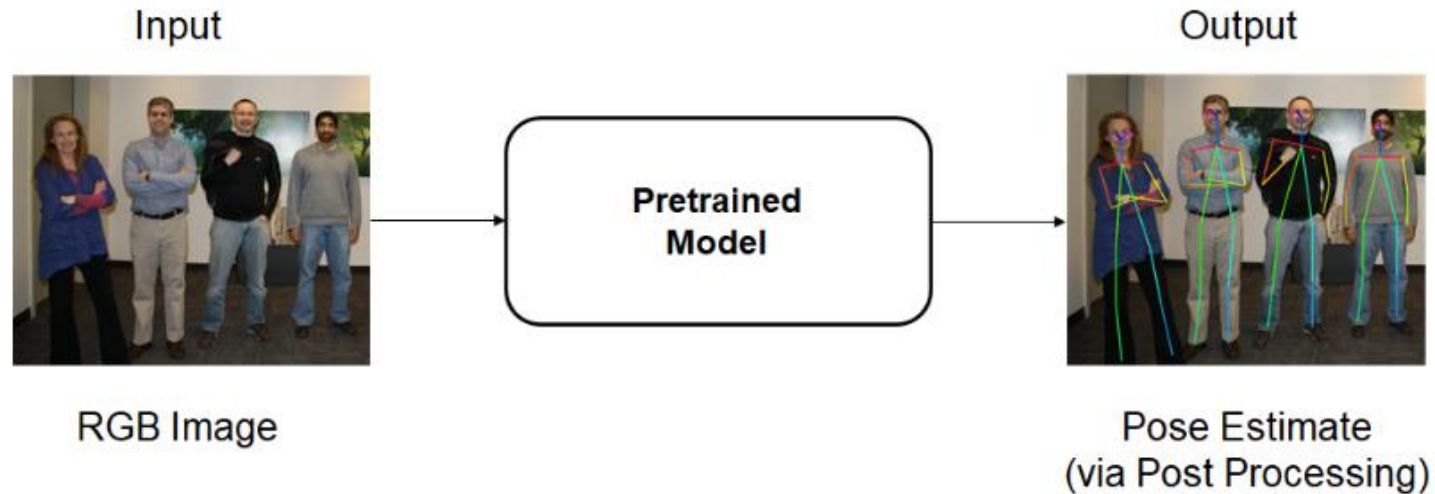
Image Translation 이미지 번역



- Transfers styles and characteristics from one image domain to another (e.g., image enhancement, image colorization, defect generation, and medical image analysis) 한 이미지 도메인에서 다른 이미지 도메인으로 스타일과 특성을 변환 (예: 이미지 향상, 이미지 채색, 결함 생성 및 의료 이미지 분석)

Network	Application	Size (MB)	Location	Example Output
Pix2PixHD(CGAN)	Synthetic Image Translation	648	Doc	
UNIT (GAN)	Day-to-Dusk Dusk-to-Day Image Translation	72.5	Doc	
UNIT (GAN)	Medical Image Denoising	72.4	Doc	
CycleGAN	Medical Image Denoising	75.3	Doc	
VDSR	Super Resolution (estimate a high-resolution image from a low-resolution image)	2.4	Doc	

Pose Estimation 포즈 추정




Network	Backbone Networks	Size (MB)	Location	Network
OpenPose	vgg19	14	Doc	OpenPose
HR Net	human-full-body-w32 human-full-body-w48	106.9 237.7	Doc	HR Net

Source: Matlab

- Localizes the position and orientation of an object using a fixed set of keypoints. 고정된 키포인트 세트를 사용하여 개체의 위치와 방향을 추정

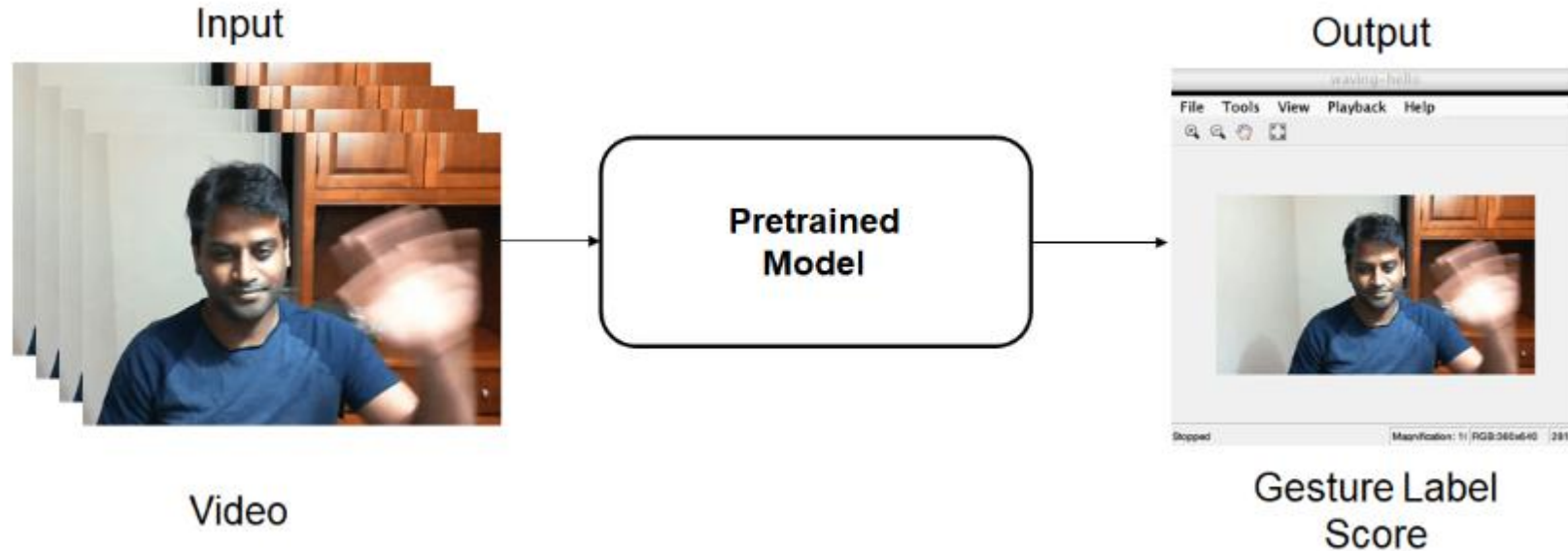
3D Reconstruction 3D 재구성

- Captures the shape and appearance of real objects. 실제 물체의 모양을 캡처

Network	Size (MB)	Location	Example Output
NeRF	3.78	GitHub	

Source: Matlab

Video Classification 비디오 분류

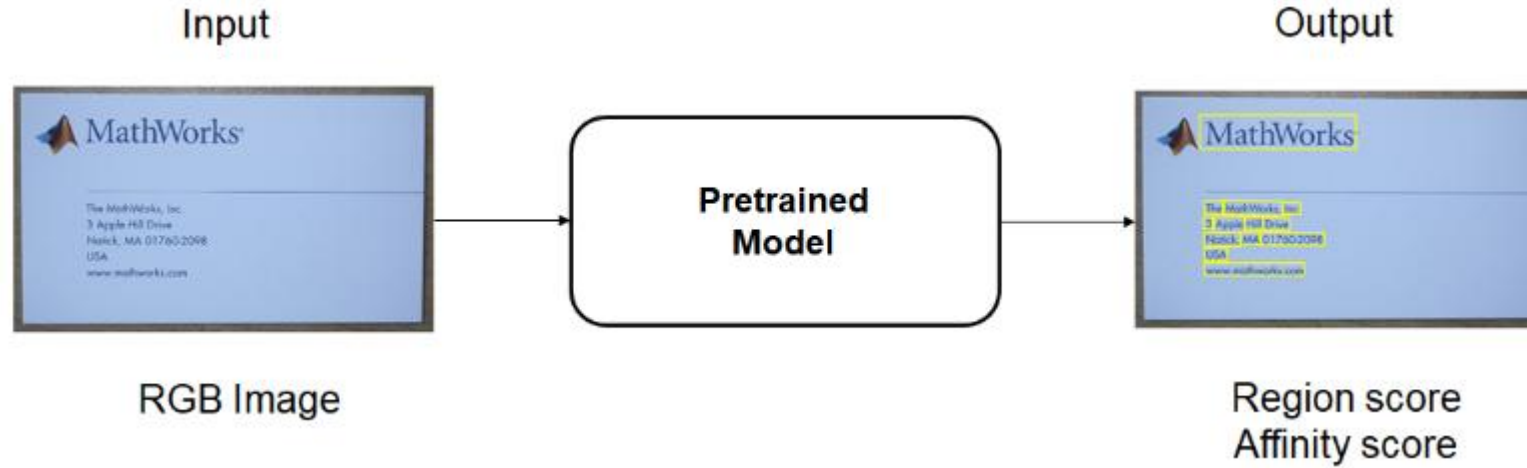


- Classifies the action or content in a sequence of video frames. 일련의 비디오 프레임에서 작업 또는 콘텐츠를 분류


Network	Inputs	Size(MB)	Classifications (Human Actions)	Description	Location
SlowFast	Video	124	400	Faster convergence than Inflated-3D	Doc
R(2+1)D	Video	112	400	Faster convergence than Inflated-3D	Doc
Inflated-3D	Video & Optical Flow data	91	400	Accuracy of the classifier improves when combining optical flow and RGB data.	Doc

Text Detection and Recognition

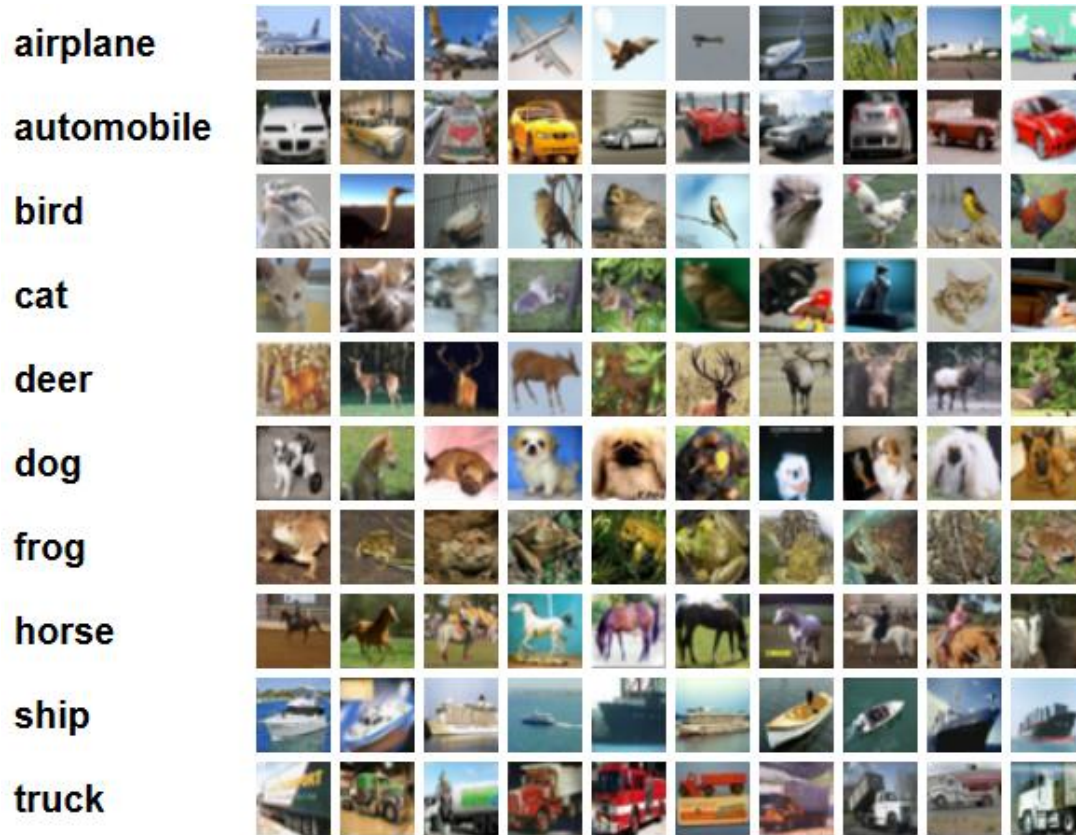
텍스트 감지 및 인식



- Locates instances of text within in images.
이미지에서 텍스트의 인스턴스를 찾음

Network	Application	Size (MB)	Location	Example Output
CRAFT	Trained to detect English, Korean, Italian, French, Arabic , German and Bangla (Indian).	3.8	Doc GitHub	
Seven Segment Digit Recognition	Seven segment digit recognition using deep learning and OCR. This is helpful in industrial automation applications where digital displays are often surrounded with complex background.	3.8	Doc GitHub	 7-segment Text Recognition

CIFAR-10 Dataset



- Consists of 60000 32x32 color images in 10 classes, with 6000 images per class. 10개 클래스의 60000 32x32 컬러 이미지로 구성되며 클래스당 6000개의 이미지가 있음
- There are 50000 training images and 10000 test images. 50000개의 훈련 이미지와 10000개의 테스트 이미지
- Divided into five training batches and one test batch, each with 10000 images. 5개의 훈련 배치와 1개의 테스트 배치로 나뉘어지며, 각각 10000개의 이미지가 있음

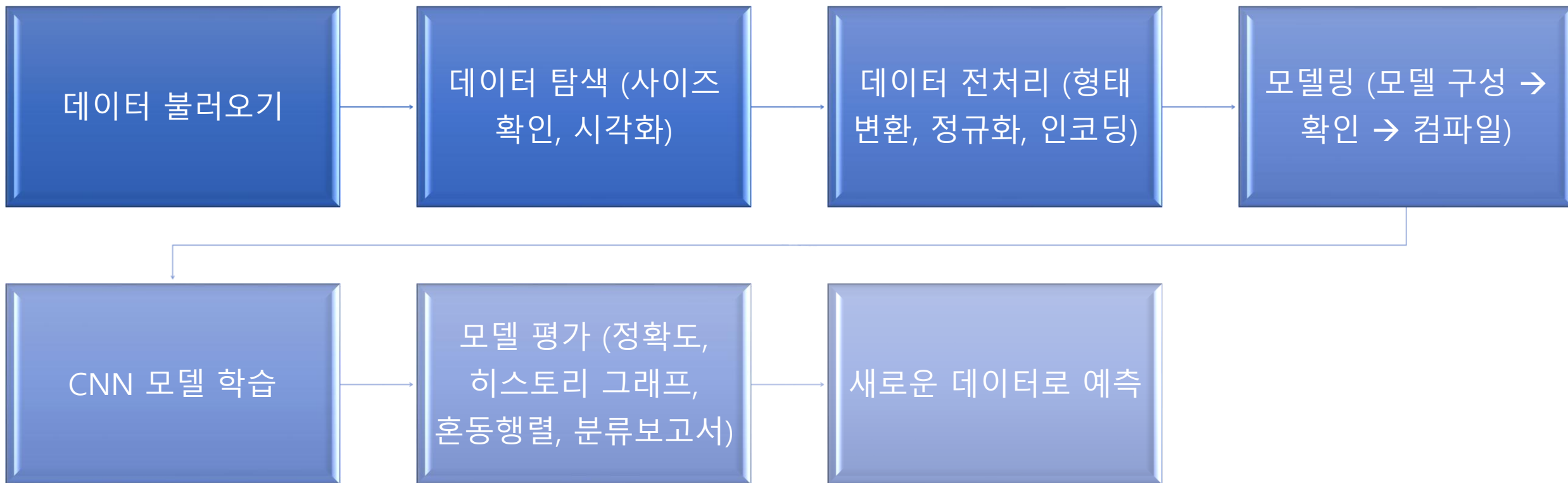
<https://www.cs.toronto.edu/~kriz/cifar.html>

Labels 라벨

- Each training and test example is assigned to one of the following labels. 각 학습 및 테스트 예제는 다음 레이블 중 하나에 할당

Label	Description
0	Airplane
1	Automobile
2	Bird
3	Cat
4	Deer
5	Dog
6	Frog
7	Horse
8	Ship
9	Truck

이미지 분류 작업순서



Exercise #2

- CIFAR-10 Dataset을 이용하여 다음 모델을 구축하시오
 1. 1 VGG block model
 2. 2 VGG blocks model
 3. 3 VGG blocks model
 4. 3 VGG blocks model with Dropout
 5. 3 VGG blocks model with Weight Regularization
 6. 3 VGG blocks model with Data Augmentation
 7. 3 VGG blocks model with Dropout, Weight Regularization
 8. 3 VGG blocks model with Dropout, Weight Regularization, Data Augmentation
 9. 3 VGG blocks model with Dropout and Batch Normalization
 10. 3 VGG Blocks Model with Dropout, Weight Regularization, Batch Normalization

데이터 불러오기

```
# CIFAR-10 데이터 불러오기
```

```
from keras.datasets import cifar10
```

```
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

데이터 탐색

데이터 형태 확인

```
x_train.shape, y_train.shape, x_test.shape,  
y_test.shape
```

훈련데이터 이미지 시각화

```
import matplotlib.pyplot as plt  
class_names = ['airplane', 'automobile', 'bird',  
'cat', 'deer',  
               'dog', 'frog', 'horse', 'ship',  
'truck']  
for i in range(9):  
    plt.subplot(3,3,i+1)  
    plt.imshow(x_train[i])  
    plt.title(class_names[y_train[i][0]])  
    plt.axis('off')  
plt.show()
```



deer



bird



automobile



horse



automobile



ship



데이터 전처리

정규화

```
x_train = x_train / 255.0
```

```
x_test = x_test / 255.0
```


1 VGG Block Model

모델 생성

```
from keras.models import Sequential

from keras.layers import Conv2D, MaxPooling2D, Dense,
Flatten

model = Sequential()

model.add(Conv2D(32, (3, 3), activation='relu',
kernel_initializer='he_uniform', padding='same',
input_shape=(32, 32, 3)))

model.add(Conv2D(32, (3, 3), activation='relu',
kernel_initializer='he_uniform', padding='same'))

model.add(MaxPooling2D((2, 2)))

model.add(Flatten())

model.add(Dense(128, activation='relu',
kernel_initializer='he_uniform'))

model.add(Dense(10, activation='softmax'))

model.summary()
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 32)	896
conv2d_1 (Conv2D)	(None, 32, 32, 32)	9248
max_pooling2d (MaxPooling2D)	(None, 16, 16, 32)	0
flatten (Flatten)	(None, 8192)	0
dense (Dense)	(None, 128)	1048704
dense_1 (Dense)	(None, 10)	1290
Total params: 1060138 (4.04 MB)		
Trainable params: 1060138 (4.04 MB)		
Non-trainable params: 0 (0.00 Byte)		

Kernel Initializer 커널 초기화자

- Initialize the weight matrix at each layer of the neural network. 신경망의 각 층에서 가중치 행렬을 초기화
- A significant impact on the model's training performance and convergence rate. 모델의 학습 성능과 수렴 속도에 큰 영향을 미침
- Choose based on the nature of the initialization method and the activation function you use. 초기화 방법의 특성과 사용하는 활성화 함수에 따라 선택
- Experiment to find the best way to initialize. 실험을 통해 최적의 초기화 방법을 찾음

Initialization Method 초기화 방법

- Random initialization 무작위 초기화

- 균등 분포나 정규 분포에서 랜덤한 값을 선택

```
model.add(Dense(32, activation='relu', kernel_initializer='random_uniform'))
```

- Xavier (Glorot) initialization 제이비아 (글로롯) 초기화

- 입력 뉴런의 개수와 출력 뉴런의 개수를 고려하여 가중치를 초기화. S자 형태의 활성화 함수에서 잘 작동함

```
model.add(Dense(32, activation='relu', kernel_initializer='glorot_normal'))
```

- He initialization 허 초기화

- 입력 뉴런의 개수만을 고려하여 가중치를 초기화. 활성화 함수가 ReLU 함수인 경우 잘 작동함

```
model.add(Dense(32, activation='relu', kernel_initializer='he_uniform'))
```

2 VGG Blocks Model

모델 생성

```
model2 = Sequential()

model2.add(Conv2D(32, (3, 3), activation='relu',
kernel_initializer='he_uniform', padding='same',
input_shape=(32, 32, 3)))

model2.add(Conv2D(32, (3, 3), activation='relu',
kernel_initializer='he_uniform', padding='same'))

model2.add(MaxPooling2D((2, 2)))

model2.add(Conv2D(64, (3, 3), activation='relu',
kernel_initializer='he_uniform', padding='same'))

model2.add(Conv2D(64, (3, 3), activation='relu',
kernel_initializer='he_uniform', padding='same'))

model2.add(MaxPooling2D((2, 2)))

model2.add(Flatten())

model2.add(Dense(128, activation='relu',
kernel_initializer='he_uniform'))

model2.add(Dense(10, activation='softmax'))

model2.summary()
```

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 32, 32, 32)	896
conv2d_3 (Conv2D)	(None, 32, 32, 32)	9248
max_pooling2d_1 (MaxPooling2D)	(None, 16, 16, 32)	0
conv2d_4 (Conv2D)	(None, 16, 16, 64)	18496
conv2d_5 (Conv2D)	(None, 16, 16, 64)	36928
max_pooling2d_2 (MaxPooling2D)	(None, 8, 8, 64)	0
flatten_1 (Flatten)	(None, 4096)	0
dense_2 (Dense)	(None, 128)	524416
dense_3 (Dense)	(None, 10)	1290
Total params: 591274 (2.26 MB)		
Trainable params: 591274 (2.26 MB)		
Non-trainable params: 0 (0.00 Byte)		

3 VGG Blocks Model

모델 생성

```
model3 = Sequential()

model3.add(Conv2D(32, (3, 3), activation='relu',
kernel_initializer='he_uniform', padding='same', input_shape=(32, 32,
3)))

model3.add(Conv2D(32, (3, 3), activation='relu',
kernel_initializer='he_uniform', padding='same'))

model3.add(MaxPooling2D((2, 2)))

model3.add(Conv2D(64, (3, 3), activation='relu',
kernel_initializer='he_uniform', padding='same'))

model3.add(Conv2D(64, (3, 3), activation='relu',
kernel_initializer='he_uniform', padding='same'))

model3.add(MaxPooling2D((2, 2)))

model3.add(Conv2D(128, (3, 3), activation='relu',
kernel_initializer='he_uniform', padding='same'))

model3.add(Conv2D(128, (3, 3), activation='relu',
kernel_initializer='he_uniform', padding='same'))

model3.add(MaxPooling2D((2, 2)))

model3.add(Flatten())

model3.add(Dense(128, activation='relu',
kernel_initializer='he_uniform'))

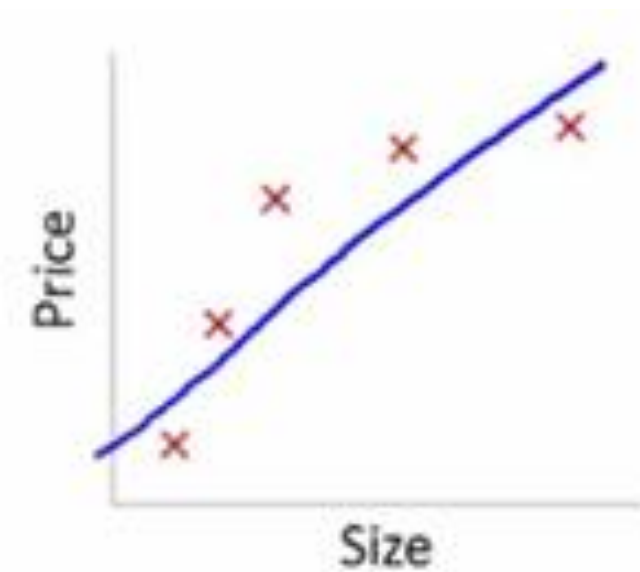
model3.add(Dense(10, activation='softmax'))

model3.summary()
```

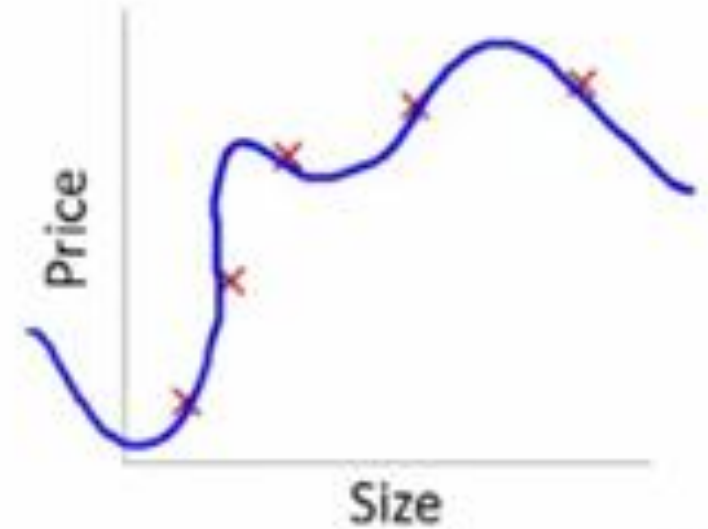
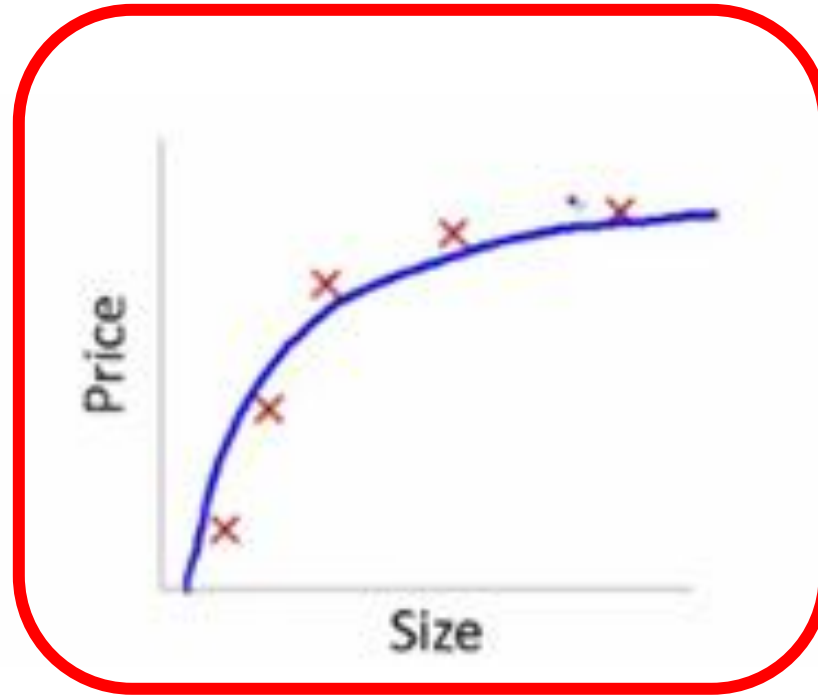
Layer (type)	Output Shape	Param #
=====		
conv2d_6 (Conv2D)	(None, 32, 32, 32)	896
conv2d_7 (Conv2D)	(None, 32, 32, 32)	9248
max_pooling2d_3 (MaxPoolin g2D)	(None, 16, 16, 32)	0
conv2d_8 (Conv2D)	(None, 16, 16, 64)	18496
conv2d_9 (Conv2D)	(None, 16, 16, 64)	36928
max_pooling2d_4 (MaxPoolin g2D)	(None, 8, 8, 64)	0
conv2d_10 (Conv2D)	(None, 8, 8, 128)	73856
conv2d_11 (Conv2D)	(None, 8, 8, 128)	147584
max_pooling2d_5 (MaxPoolin g2D)	(None, 4, 4, 128)	0
flatten_2 (Flatten)	(None, 2048)	0
dense_4 (Dense)	(None, 128)	262272
dense_5 (Dense)	(None, 10)	1290
=====		
Total params: 550570 (2.10 MB)		
Trainable params: 550570 (2.10 MB)		
Non-trainable params: 0 (0.00 Byte)		

Fit of Models 모형적합

모델의 복잡도와 정확도 사이에는 상반된 관계가 있음. 모델이 복잡할 수록 데이터에 과적합되고, 단순화 될수록 데이터와 맞지 않음



- Underfitting 과소적합
- High error from training data points. 훈련데이터에서 오차가 큼



- Overfitting 과적합
- Almost zero training error, but too sensitive and capture random patterns present only in the current data set. 훈련데이터에서는 오차가 거의 없음. 현재 데이터에 너무 민감하고, 무작위패턴까지 반영

오버피팅 해결하는 방법

- Dropout 드롭아웃
- Batch Normalization 배치 정규화
- Weight Regularization 가중치 정규화
- 모델의 복잡도 줄이기, 훈련 데이터 양 늘리기 등

3 VGG Blocks Model with Dropout

모델 생성

```
model4 = Sequential()

model4.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same',
input_shape=(32, 32, 3)))

model4.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))

model4.add(MaxPooling2D((2, 2)))

model4.add(Dropout(0.2))

model4.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))

model4.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))

model4.add(MaxPooling2D((2, 2)))

model4.add(Dropout(0.2))

model4.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform',
padding='same'))

model4.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform',
padding='same'))

model4.add(MaxPooling2D((2, 2)))

model4.add(Dropout(0.2))

model4.add(Flatten())

model4.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))

model4.add(Dropout(0.2))

model4.add(Dense(10, activation='softmax'))

model4.summary()
```

Layer (type)	Output Shape	Param #
=====		
conv2d_24 (Conv2D)	(None, 32, 32, 32)	896
conv2d_25 (Conv2D)	(None, 32, 32, 32)	9248
max_pooling2d_12 (MaxPooling2D)	(None, 16, 16, 32)	0
dropout_8 (Dropout)	(None, 16, 16, 32)	0
conv2d_26 (Conv2D)	(None, 16, 16, 64)	18496
conv2d_27 (Conv2D)	(None, 16, 16, 64)	36928
max_pooling2d_13 (MaxPooling2D)	(None, 8, 8, 64)	0
dropout_9 (Dropout)	(None, 8, 8, 64)	0
conv2d_28 (Conv2D)	(None, 8, 8, 128)	73856
conv2d_29 (Conv2D)	(None, 8, 8, 128)	147584
max_pooling2d_14 (MaxPooling2D)	(None, 4, 4, 128)	0
dropout_10 (Dropout)	(None, 4, 4, 128)	0
flatten_5 (Flatten)	(None, 2048)	0
dense_10 (Dense)	(None, 128)	262272
dropout_11 (Dropout)	(None, 128)	0
dense_11 (Dense)	(None, 10)	1290
=====		
Total params: 550570 (2.10 MB)		
Trainable params: 550570 (2.10 MB)		
Non-trainable params: 0 (0.00 Byte)		

Dropout Layer 드랍아웃 레이어

- Randomly switches off some neurons in the network which forces the data to find new paths. Therefore, this reduces overfitting. 무작위로 뉴론을 꺼버림으로써 데이터가 새로운 경로를 찾게 하여 오버피팅을 방지

```
from keras.layers import Dropout  
model.add(Dropout(0.25))
```

- Dropout은 노드의 일부분을 사용하지 않는 기법
- 네트워크의 각 부분이 독립적으로 유용한 특징을 학습하게 하여 불필요한 협업을 방지하게 함
- 과대적합을 줄임

3 VGG Blocks Model with Weight Regularization

모델 생성

```
model5 = Sequential()

model5.add(Conv2D(32, (3, 3), activation='relu',
kernel_initializer='he_uniform', padding='same',
kernel_regularizer=l2(0.001), input_shape=(32, 32, 3)))

model5.add(Conv2D(32, (3, 3), activation='relu',
kernel_initializer='he_uniform', padding='same',
kernel_regularizer=l2(0.001)))

model5.add(MaxPooling2D((2, 2)))

model5.add(Conv2D(64, (3, 3), activation='relu',
kernel_initializer='he_uniform', padding='same',
kernel_regularizer=l2(0.001)))

model5.add(Conv2D(64, (3, 3), activation='relu',
kernel_initializer='he_uniform', padding='same',
kernel_regularizer=l2(0.001)))

model5.add(MaxPooling2D((2, 2)))

model5.add(Conv2D(128, (3, 3), activation='relu',
kernel_initializer='he_uniform', padding='same',
kernel_regularizer=l2(0.001)))

model5.add(Conv2D(128, (3, 3), activation='relu',
kernel_initializer='he_uniform', padding='same',
kernel_regularizer=l2(0.001)))

model5.add(MaxPooling2D((2, 2)))

model5.add(Flatten())

model5.add(Dense(128, activation='relu', kernel_initializer='he_uniform',
kernel_regularizer=l2(0.001)))

model5.add(Dense(10, activation='softmax'))
```

Layer (type)	Output Shape	Param #
=====		
conv2d_30 (Conv2D)	(None, 32, 32, 32)	896
conv2d_31 (Conv2D)	(None, 32, 32, 32)	9248
max_pooling2d_15 (MaxPooling2D)	(None, 16, 16, 32)	0
conv2d_32 (Conv2D)	(None, 16, 16, 64)	18496
conv2d_33 (Conv2D)	(None, 16, 16, 64)	36928
max_pooling2d_16 (MaxPooling2D)	(None, 8, 8, 64)	0
conv2d_34 (Conv2D)	(None, 8, 8, 128)	73856
conv2d_35 (Conv2D)	(None, 8, 8, 128)	147584
max_pooling2d_17 (MaxPooling2D)	(None, 4, 4, 128)	0
flatten_6 (Flatten)	(None, 2048)	0
dense_12 (Dense)	(None, 128)	262272
dense_13 (Dense)	(None, 10)	1290
=====		
Total params: 550570 (2.10 MB)		
Trainable params: 550570 (2.10 MB)		
Non-trainable params: 0 (0.00 Byte)		

Weight Regularization 가중치 정규화

- Assign additional penalties to the loss function to control the complexity of the model. 손실 함수에 추가적인 페널티를 부여하여 모델의 복잡도를 제어
- Reduce overfitting in deep learning models and improve generalization performance. 딥러닝 모델에서 과적합을 줄이고 일반화 성능을 향상

L1 and L2 Regularization

- L2 (Ridge) Regularization: 가중치 값을 제공한 값에 대한 페널티를 손실 함수에 추가. 가중치의 크기를 작게 유지하면서 모델의 복잡도를 줄임

```
from tensorflow.keras.regularizers import  
l2  
  
kernel_regularizer=l2(0.001) #0.001은 정규화  
강도
```

- L1 (Lasso) Regularization: 가중치의 절대값에 대한 페널티를 손실 함수에 추가. 일부 가중치를 0으로 만들어 희소성을 유도

```
from tensorflow.keras.regularizers import  
l1  
  
kernel_regularizer=l1(0.001) #0.001은 정규화  
강도
```

L1 Regularization

$$\text{Cost} = \sum_{i=0}^N (y_i - \sum_{j=0}^M x_{ij} W_j)^2 + \lambda \sum_{j=0}^M |W_j|$$

L2 Regularization

$$\text{Cost} = \underbrace{\sum_{i=0}^N (y_i - \sum_{j=0}^M x_{ij} W_j)^2}_{\text{Loss function}} + \lambda \underbrace{\sum_{j=0}^M W_j^2}_{\text{Regularization Term}}$$

3 VGG Blocks Model with Dropout and Batch Normalization

모델 생성

```
from keras.layers import BatchNormalization
```

```
model10 = Sequential()
```

```
model10.add(Conv2D(32, (3, 3), activation='relu',  
kernel_initializer='he_uniform', padding='same',  
input_shape=(32, 32, 3)))
```

```
model10.add(BatchNormalization())
```

```
model10.add(Conv2D(32, (3, 3), activation='relu',  
kernel_initializer='he_uniform', padding='same'))
```

```
model10.add(BatchNormalization())
```

```
model10.add(MaxPooling2D((2, 2)))
```

```
model10.add(Dropout(0.2))
```

```
model10.add(Conv2D(64, (3, 3), activation='relu',  
kernel_initializer='he_uniform', padding='same'))
```

```
model10.add(BatchNormalization())
```

```
model10.add(Conv2D(64, (3, 3), activation='relu',  
kernel_initializer='he_uniform', padding='same'))
```

```
model10.add(BatchNormalization())
```

```
model10.add(MaxPooling2D((2, 2)))
```

```
model10.add(Dropout(0.3))
```

```
model10.add(Conv2D(128, (3, 3), activation='relu',  
kernel_initializer='he_uniform', padding='same'))
```

```
model10.add(BatchNormalization())
```

```
model10.add(Conv2D(128, (3, 3), activation='relu',  
kernel_initializer='he_uniform', padding='same'))
```

```
model10.add(BatchNormalization())
```

```
model10.add(MaxPooling2D((2, 2)))
```

```
model10.add(Dropout(0.4))
```

```
model10.add(Flatten())
```

```
model10.add(Dense(128, activation='relu',  
kernel_initializer='he_uniform'))
```

```
model10.add(BatchNormalization())
```

```
model10.add(Dropout(0.5))
```

```
model10.add(Dense(10, activation='softmax'))
```

```
model10.summary()
```

3 VGG Blocks Model with Dropout and Batch Normalization

Model: "sequential_10"		
Layer (type)	Output Shape	Param #

conv2d_54 (Conv2D)	(None, 32, 32, 32)	896
batch_normalization_7 (Batch Normalization)	(None, 32, 32, 32)	128
conv2d_55 (Conv2D)	(None, 32, 32, 32)	9248
batch_normalization_8 (Batch Normalization)	(None, 32, 32, 32)	128
max_pooling2d_27 (MaxPooling2D)	(None, 16, 16, 32)	0
dropout_24 (Dropout)	(None, 16, 16, 32)	0
conv2d_56 (Conv2D)	(None, 16, 16, 64)	18496
batch_normalization_9 (Batch Normalization)	(None, 16, 16, 64)	256
conv2d_57 (Conv2D)	(None, 16, 16, 64)	36928
batch_normalization_10 (Batch Normalization)	(None, 16, 16, 64)	256
max_pooling2d_28 (MaxPooling2D)	(None, 8, 8, 64)	0
dropout_25 (Dropout)	(None, 8, 8, 64)	0

conv2d_58 (Conv2D)	(None, 8, 8, 128)	73856
batch_normalization_11 (Batch Normalization)	(None, 8, 8, 128)	512
conv2d_59 (Conv2D)	(None, 8, 8, 128)	147584
batch_normalization_12 (Batch Normalization)	(None, 8, 8, 128)	512
max_pooling2d_29 (MaxPooling2D)	(None, 4, 4, 128)	0
dropout_26 (Dropout)	(None, 4, 4, 128)	0
flatten_10 (Flatten)	(None, 2048)	0
dense_20 (Dense)	(None, 128)	262272
batch_normalization_13 (Batch Normalization)	(None, 128)	512
dropout_27 (Dropout)	(None, 128)	0
dense_21 (Dense)	(None, 10)	1290

Total params: 552874 (2.11 MB)		
Trainable params: 551722 (2.10 MB)		
Non-trainable params: 1152 (4.50 KB)		

Batch Normalization 배치정규화

- A technique to standardize the distribution of data by adjusting the mean and variance in each minibatch in the training process of a deep learning model. 딥러닝 모델의 학습 과정에서 각 미니배치에서 평균과 분산을 조정하여 데이터의 분포를 표준화하는 기법
- Speed up training, mitigate slope loss and runaway issues, and improve model performance and reliability. 이 방법으로 학습 속도를 높이고, 기울기 소실 및 폭주 문제를 완화하며, 모델의 성능과 안정성을 향상
- Put a normalized layer for each layer and adjust it so that the modified distribution does not appear. 각 레이어마다 정규화 하는 레이어를 두어, 변형된 분포가 나오지 않도록 조절

```
from keras.layers import BatchNormalization  
model.add(BatchNormalization())
```

Early Stopping 조기 종료

- As one of the callbacks of deep learning, you can set it to end early when a certain event occurs in the model training process. 딥러닝의 콜백의 하나로 모델 학습과정에서 특정 이벤트가 발생했을 때 일찍 종료되도록 설정할 수 있음

```
# 체크포인트, 조기종료 설정
```

```
from keras.callbacks import ModelCheckpoint, EarlyStopping
```

```
earlystopping = EarlyStopping(monitor='val_accuracy', patience=5) # 5번의 에포크  
동안 정확도에 개선이 없을 경우 학습을 멈추도록 설정
```

```
# 모델 학습
```

```
history = model.fit(x_train, y_train, epochs=100, batch_size=64, validation_da  
ta=(x_test, y_test), verbose=1, callbacks=[earlystopping])
```


Image Data Augmentation

이미지 데이터 증강

- A technique used to artificially increase the diversity of your training dataset by applying random transformations to the existing images. 기존 이미지에 무작위 변환을 적용하여 훈련 데이터 세트의 다양성을 인위적으로 늘리는 데 사용되는 기술
- Helps improve the generalization and robustness of deep learning models, especially in tasks like image classification, object detection, and segmentation. 딥러닝 모델의 일반화와 견고성을 개선하는 데 도움이 되며, 특히 이미지 분류, 객체 감지 및 세분화와 같은 작업에서 도움이 됨
- Increases data variety, reduces the need for data, and produces the realistic performance. 데이터 다양성을 높이고, 데이터의 필요성을 줄이며, 사실적인 성능을 생성

Image Data Augmentation Techniques 이미지 증강 기법

- Flipping
 - Horizontal Flip: 이미지를 수평으로 뒤집어 거울 이미지를 만듦
 - Vertical Flip: 이미지를 수직으로 뒤집음
- Rotation: 지정된 각도로 이미지를 회전하여 개체 방향에 변형을 적용
- Zooming: 이미지를 확대하거나 축소하여 이미지에 있는 개체의 배율을 변경
- Shifting
 - Width Shift: 이미지를 수평으로 이동
 - Height Shift: 이미지를 수직으로 이동
- Shearing: 이미지의 한 부분을 지정된 각도로 이동하여 이미지를 기울임
- Brightness Adjustment: 이미지의 밝기를 임의로 조정
- Contrast Adjustment: 이미지의 대비를 임의로 조정
- Noise Injection: 이미지에 무작위 노이즈를 추가

Image Data Augmentation Code

이미지 데이터 증강

```
from keras.preprocessing.image import ImageDataGenerator
```

```
train_datagen = ImageDataGenerator(  
    width_shift_range=0.1,  
    height_shift_range=0.1,  
    horizontal_flip=True  
)
```

```
train_flow = train_datagen.flow(x_train, y_train, batch_size=64)
```

```
test_datagen = ImageDataGenerator()
```

```
test_flow = test_datagen.flow(x_test, y_test, batch_size=64)
```

모델 학습

```
history = model6.fit(train_flow, epochs=100, batch_size=64, validation_data=test_flow, verbose=1,  
    callbacks=[checkpoint, earlystopping])
```

ChatGPT로 코드 개선 연습

- 다음 코드에 대한 개선된 코드 받아보기. 해설도 요청하세요

```
# 잘못된 코드
test_img = Image.open('seven.jpg').convert('L')
test_array = np.array(test_img).reshape(28, 28, 1)
prediction = model.predict(test_array)
```

- 출력 결과

! 문제점

- model.predict()는 입력으로 **4차원 텐서(batch 포함)**를 기대합니다.
 - 기대하는 입력 shape: (1, 28, 28, 1) ← 배치가 1장인 이미지
- 하지만 위 코드는 3차원: (28, 28, 1) → 그래서 ValueError 발생
- 또한 **정규화(0255 → 01)**가 되어 있지 않으면 예측 정확도가 떨어질 수 있습니다.

ChatGPT로 코드 개선 연습

- 다음 코드에 대한 개선된 코드 받아보기. 해설도 요청하세요

```
model = Sequential()  
model.add(Dense(128, input_shape=(784,),  
activation='relu'))  
model.add(Dense(10, activation='softmax'))
```

- 출력 결과

! 한계점

- 이미지의 공간 구조(위치, 패턴 등)를 고려하지 못함 → 성능이 CNN보다 떨어짐
- 이미지를 평탄화(Flatten)해야 함 → 전처리가 필요함