

ChatGPT와 함께하는 딥러닝 5일 완성

CNN 모델 심화 및 CIFAR-10 분류 실습

4일차



서울대학교 평생교육원
Extension College Seoul National University

Topics

- 전이 학습 개념 및 활용 목적 이해
- 사전 학습된 대표 모델(VGG16, ResNet50 등)의 구조
- 전이 학습 기반 이미지 분류 모델 구축
- 일부 레이어 미세 조정 및 재학습
- 허깅페이스 모델 활용 이미지 분류 실습

이미지 분류 작업순서



CNN 모델 성능향상 기법

- Dropout Regularization (20%, 30%, 50%) 드롭아웃
- Weight Regularization 가중치 정규화
- Data Augmentation 데이터 증강
- Dropout and Data Augmentation 드롭아웃 + 데이터증강
- Dropout and Data Augmentation and Batch Normalization 드롭아웃 + 데이터증강 + 배치정규화

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)
```

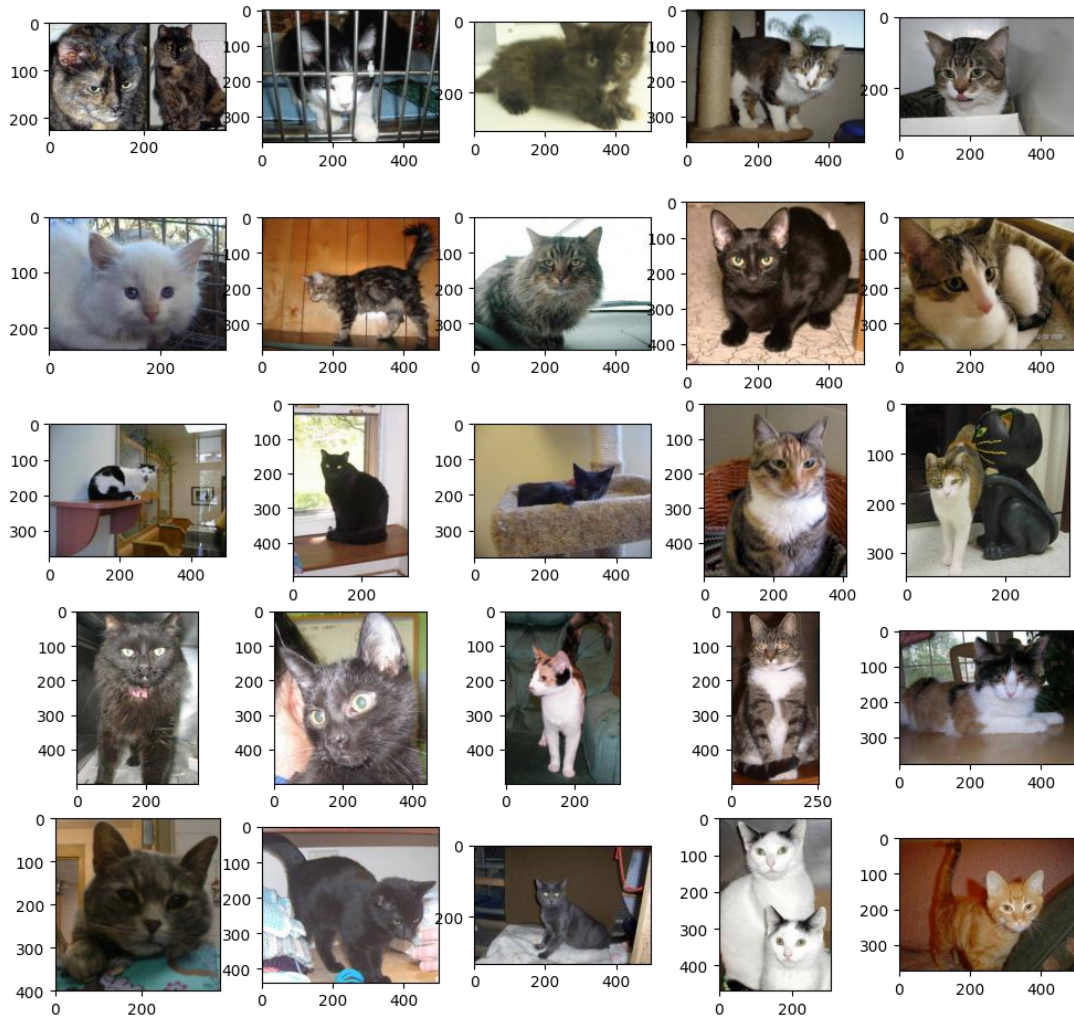
```
test_datagen = ImageDataGenerator()
```

```
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])
```

Cats and Dogs Dataset



- 마이크로 소프트 사이트

- <https://www.microsoft.com/en-us/download/details.aspx?id=54765>

- 캐글

- <https://www.kaggle.com/competitions/dogs-vs-cats/data>

Exercise #1

- 구글에서 cats_and_dogs_filtered.zip 파일 다운로드 받고 압축풀기
- Train and validation directory에 있는 이미지 데이터 증강 생성 (150x150)
- CNN 모델 생성 및 훈련, 히스토리 그래프
- 테스트 이미지 분류

Dogs Vs. Cats 데이터 다운로드

다운로드 (리눅스)

```
!wget --no-check-certificate \ #SSL 인증서 검사를 비활성화
      https://storage.googleapis.com/mledu-
datasets/cats_and_dogs_filtered.zip \
      -O /tmp/cats_and_dogs_filtered.zip #저장이름
```

다운로드 (윈도우)

```
!curl -o /tmp/cats_and_dogs_filtered.zip -k
https://storage.googleapis.com/mledu-
datasets/cats_and_dogs_filtered.zip
```

폴더에 압축 풀기

압축풀기

```
import os
```

```
import zipfile
```

```
local_zip = '/content/cats_and_dogs_filtered.zip'
```

```
zip_ref = zipfile.ZipFile(local_zip, 'r')
```

```
zip_ref.extractall('/content')
```

```
zip_ref.close()
```

훈련 및 검증 디렉토리 설정

패쓰설정

```
base_dir = '/content/cats_and_dogs_filtered'
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'validation')
train_dir, validation_dir

('/content/cats_and_dogs_filtered/train',
 '/content/cats_and_dogs_filtered/validation')
```

훈련 고양이, 강아지 디렉토리 설정

훈련 고양이, 훈련 강아지 디렉토리 설정

```
train_cats_dir = os.path.join(train_dir, 'cats')
```

```
train_dogs_dir = os.path.join(train_dir, 'dogs')
```

```
train_cats_dir, train_dogs_dir
```

```
( '/content/cats_and_dogs_filtered/train/cats',  
  '/content/cats_and_dogs_filtered/train/dogs')
```

폴더안에 파일이름 읽기

jpg 파일 리스트 저장

```
import glob
```

```
train_cat_fnames = glob.glob(train_cats_dir+'/*.jpg')
```

```
train_dog_fnames = glob.glob(train_dogs_dir+'/*.jpg')
```

이미지 갯수 세기

```
len(train_cat_fnames), len(train_dog_fnames)
```

```
(1000, 1000)
```

시각화

훈련 데이터 시각화

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

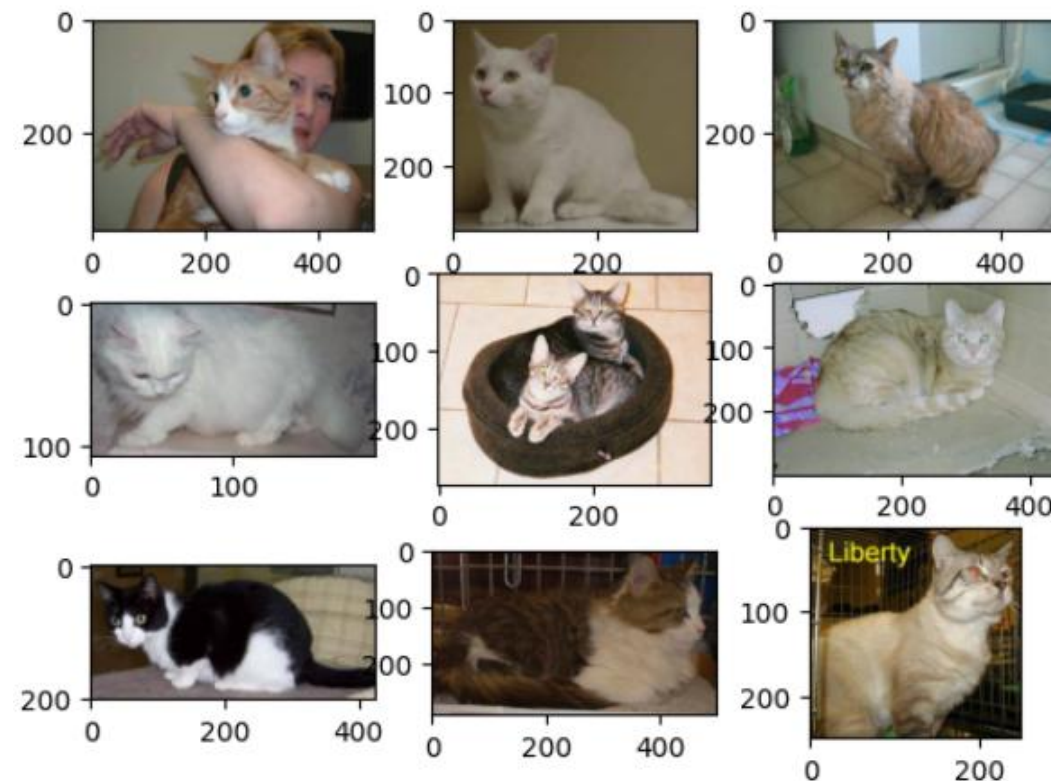
```
for i in range(9):
```

```
    plt.subplot(3,3,i+1)
```

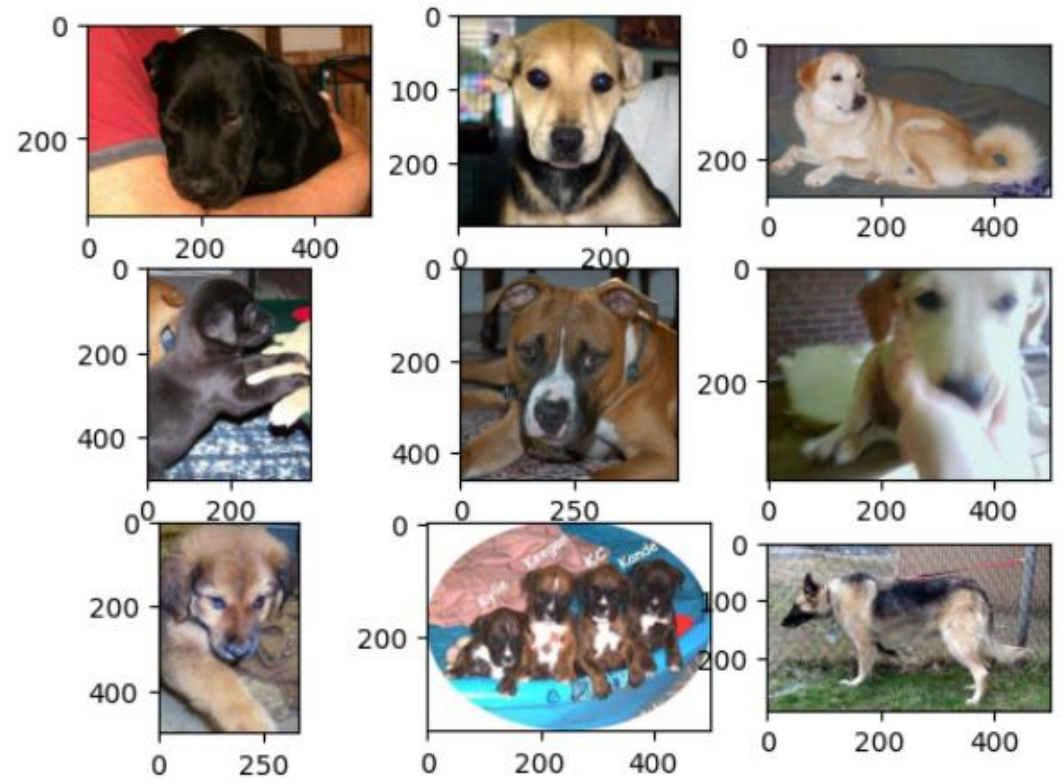
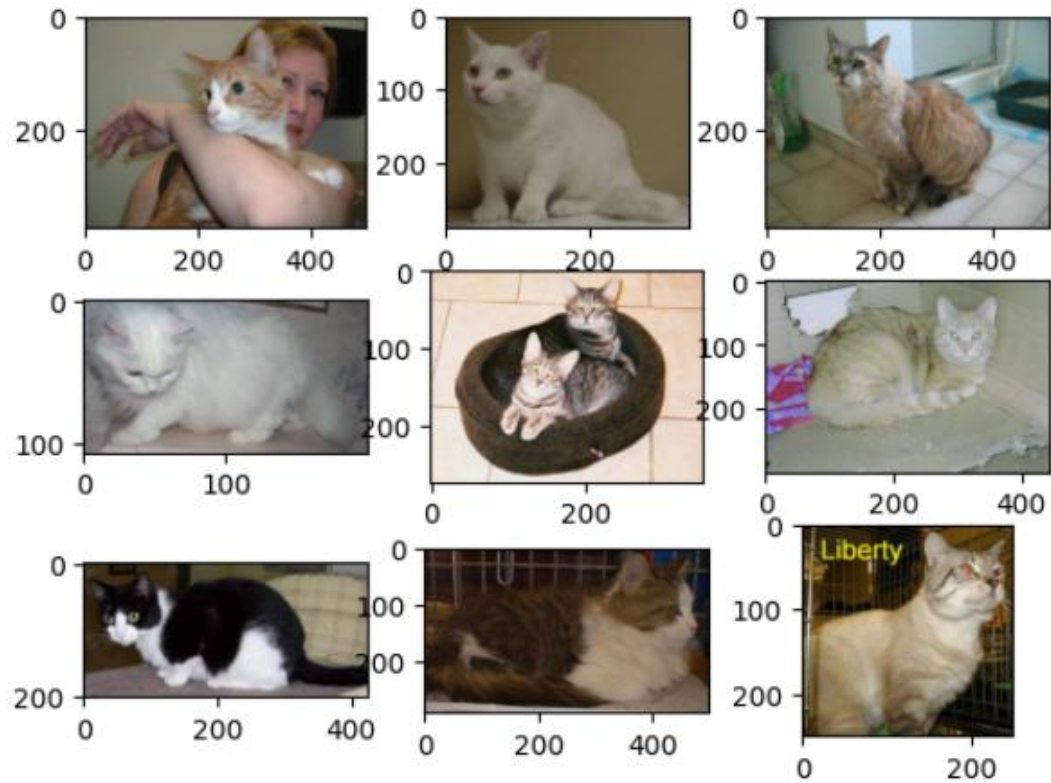
```
    img = load_img(train_cat_fnames[i])
```

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

```
plt.show()
```

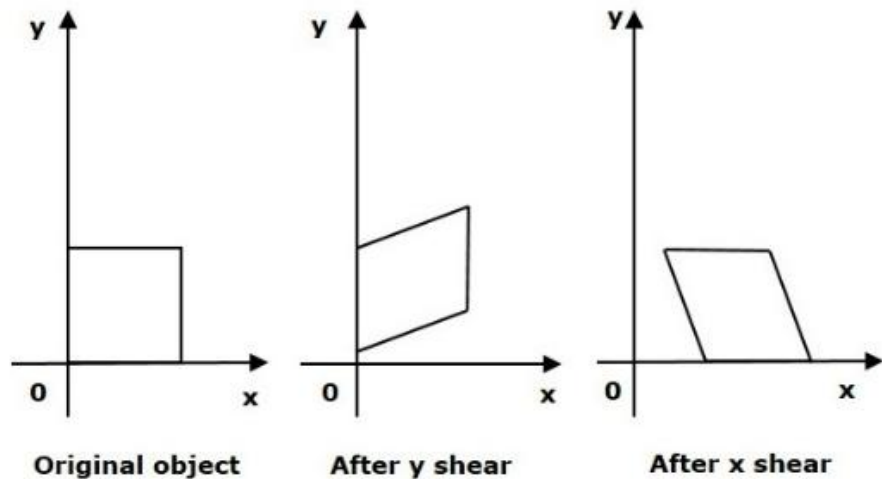


훈련 데이터 시각화



이미지 데이터 증강 설정

```
train_datagen = ImageDataGenerator(  
    rescale = 1.0/255., # 픽셀값 스케일 범위 [0,1]  
    rotation_range=40, # 40 도까지 임의 회전  
    width_shift_range=0.2, # 수평이동  
    height_shift_range=0.2, # 수직이동  
    shear_range=0.2, # 전단변환  
    zoom_range=0.2, # 20%까지 임의 확대  
    horizontal_flip=True) #수평플립  
test_datagen = ImageDataGenerator(  
    rescale = 1.0/255.) # 정규화
```



Flow from Directory

- 이미지 파일을 로딩하여 라벨을 가진 증강 이미지의 배치를 생성

```
train = train_datagen.flow_from_directory(  
    train_dir, #훈련이미지가 있는 폴더  
    target_size=(150,150), #이미지 사이즈 변경  
    class_mode='binary' #이진 레이블  
)  
  
test = test_datagen.flow_from_directory(  
    validation_dir, #검증이미지가 있는 폴더  
    target_size=(150,150), #이미지 사이즈 변경  
    class_mode='binary' #이진 레이블  
)
```

기본 CNN 모델

모델생성

```
model = Sequential()  
model.add(Conv2D(32, 3, activation='relu', input_shape=(150,150,3)))  
model.add(MaxPool2D(2,2))  
model.add(Conv2D(64, 3, activation='relu'))  
model.add(MaxPool2D(2,2))  
model.add(Conv2D(128, 3, activation='relu'))  
model.add(MaxPool2D(2,2))  
model.add(Conv2D(128, 3, activation='relu'))  
model.add(MaxPool2D(2,2))  
model.add(Flatten())  
model.add(Dropout(.5))  
model.add(Dense(512, activation='relu'))  
model.add(Dense(1, activation='sigmoid'))  
model.summary()
```

Model: "sequential"		
Layer (type)	Output Shape	Param #
=====		
conv2d (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d (MaxPooling2D)	(None, 74, 74, 32)	0
conv2d_1 (Conv2D)	(None, 72, 72, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 36, 36, 64)	0
conv2d_2 (Conv2D)	(None, 34, 34, 128)	73856
max_pooling2d_2 (MaxPooling2D)	(None, 17, 17, 128)	0
conv2d_3 (Conv2D)	(None, 15, 15, 128)	147584
max_pooling2d_3 (MaxPooling2D)	(None, 7, 7, 128)	0
flatten (Flatten)	(None, 6272)	0
dropout (Dropout)	(None, 6272)	0
dense (Dense)	(None, 512)	3211776
dense_1 (Dense)	(None, 1)	513
=====		
Total params: 3453121 (13.17 MB)		
Trainable params: 3453121 (13.17 MB)		
Non-trainable params: 0 (0.00 Byte)		

컴파일 및 학습

컴파일

```
model.compile(loss='binary_crossentropy', optimizer=Adam(learning_rate=1e-4),  
metrics=['accuracy'])
```

체크 포인트, 조기 종료 설정

```
from keras.callbacks import ModelCheckpoint, EarlyStopping  
  
save_file_name = 'cats_and_dogs_cnn_model.h5'  
checkpoint = ModelCheckpoint(save_file_name, monitor='val_accuracy',  
                             verbose=1, save_best_only=True, mode='auto')  
earlystopping = EarlyStopping(monitor='val_accuracy', patience=5)
```

모델 학습

```
history = model.fit(train_flow, epochs=30, validation_data=test_flow, verbose=2,  
callbacks=[checkpoint, earlystopping])
```

히스토리 그래프

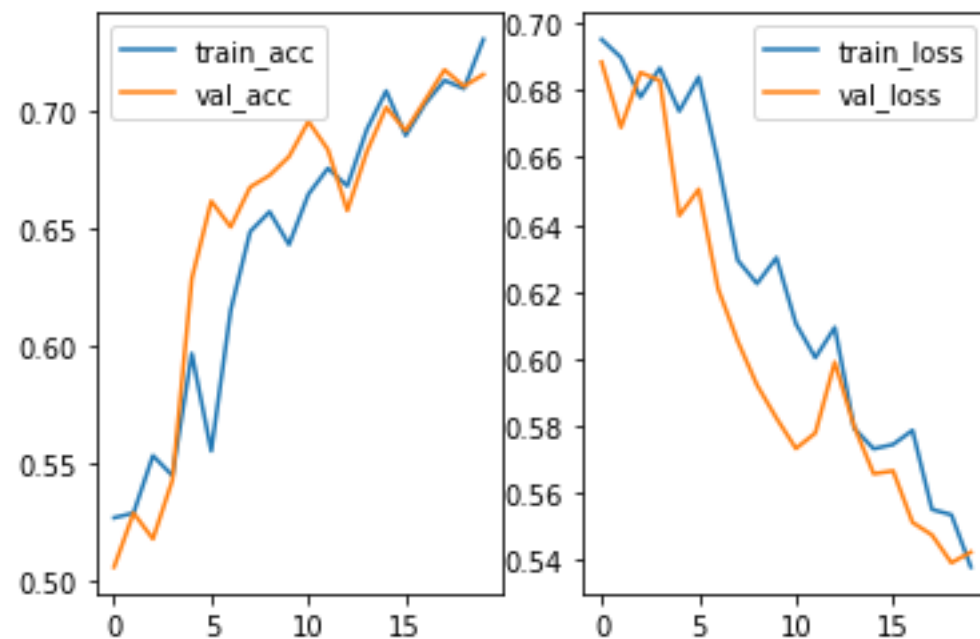
```
import matplotlib.pyplot as plt

plt.figure(figsize=(15,5))

plt.subplot(121)
plt.title('Loss')
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='test')
plt.legend()

plt.subplot(122)
plt.title('Accuracy')
plt.plot(history.history['accuracy'], label='train')
plt.plot(history.history['val_accuracy'], label='test')
plt.legend()

plt.show()
```



테스트 이미지 분류

```
# 테스트 이미지 분류
```

```
import numpy as np
```

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

```
path = '/content/chihuahua.jpg'
```

```
img=image.load_img(path, target_size=(150, 150))
```

```
img.size # (150, 150)
```

```
x=image.img_to_array(img)
```

```
x.shape # (150, 150, 3)
```

```
x=np.expand_dims(x, axis=0)
```

```
x.shape # (1, 150, 150, 3)
```

```
classes = model.predict(x)
```

```
print(classes[0]) #[1.]
```

```
if classes[0]>0:
```

```
    print(path + " is a dog")
```

```
else:
```

```
    print(path + " is a cat")
```

```
    /content/chihuahua.jpg is a dog
```



```
array([[227., 227., 229.],
       [229., 229., 231.],
       [227., 227., 229.],
       ...,
       [ 36.,  35.,  31.],
       [ 39.,  38.,  34.],
       [ 37.,  33.,  30.]])
```

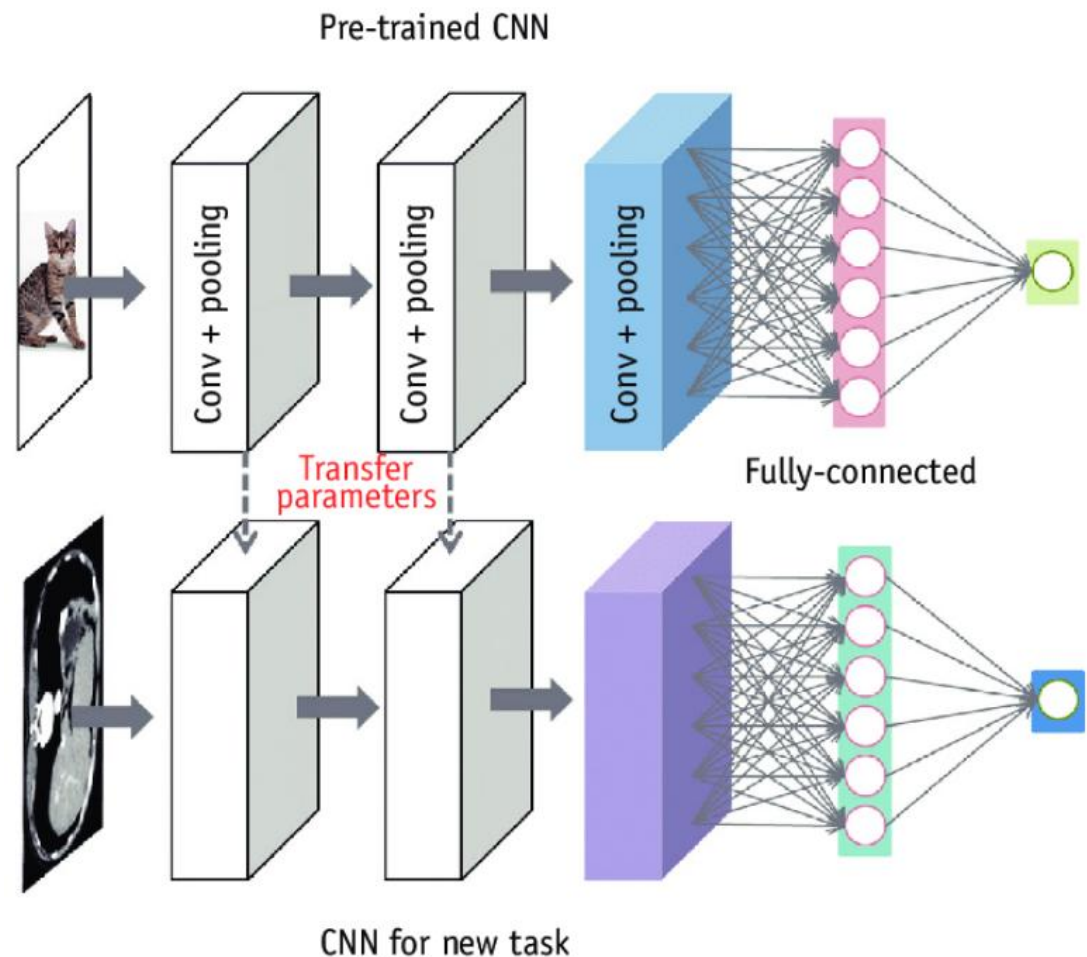
```
[[227., 227., 229.],
 [222., 222., 222.],
 [220., 220., 220.],
```

```
array([[[[227., 227., 229.],
         [229., 229., 231.],
         [227., 227., 229.],
         ...,
         [ 36.,  35.,  31.],
         [ 39.,  38.,  34.],
         [ 37.,  33.,  30.]])])
```

```
[[227., 227., 229.],
 [222., 222., 222.],
 [220., 220., 220.],
```

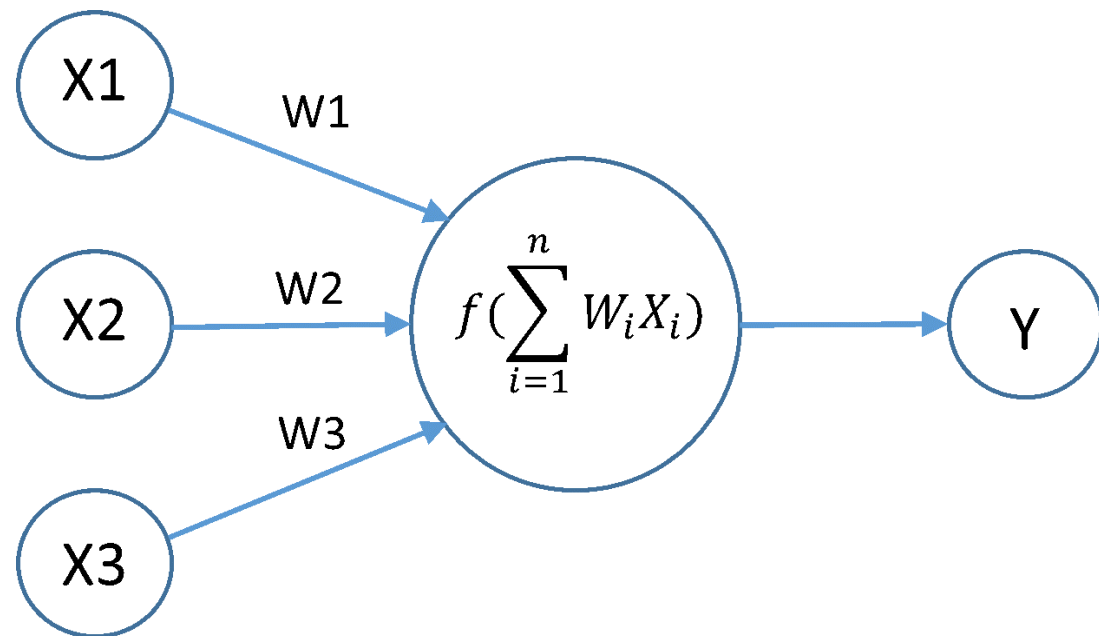
Transfer Learning 전이학습

- Process of taking pretrained model (usually trained on large dataset, such as ImageNet) and “fine-tuning” model with new dataset. 사전 훈련된 모델 (일반적으로 ImageNet과 같은 대규모 데이터 세트에서 미리 훈련)을 가지고 새로운 데이터 세트에 “미세 조정”을 하는 프로세스



Weights and Bias 가중치와 바이어스

- A set of weights and biases between each layer, W and b. 레이어 사이에 가중치와 바이어스가 있음
- The accuracy of the prediction is determined by the value of the weight and bias. 가중치와 바이어스의 값에 따라 예측의 정확도가 결정됨



The circles are neurons or nodes, with their functions on the data and the lines/edges connecting them are the weights/information being passed along. 동그라미는 함수를 가진 노드들. 노드를 연결시키는 선은 가중치 임!

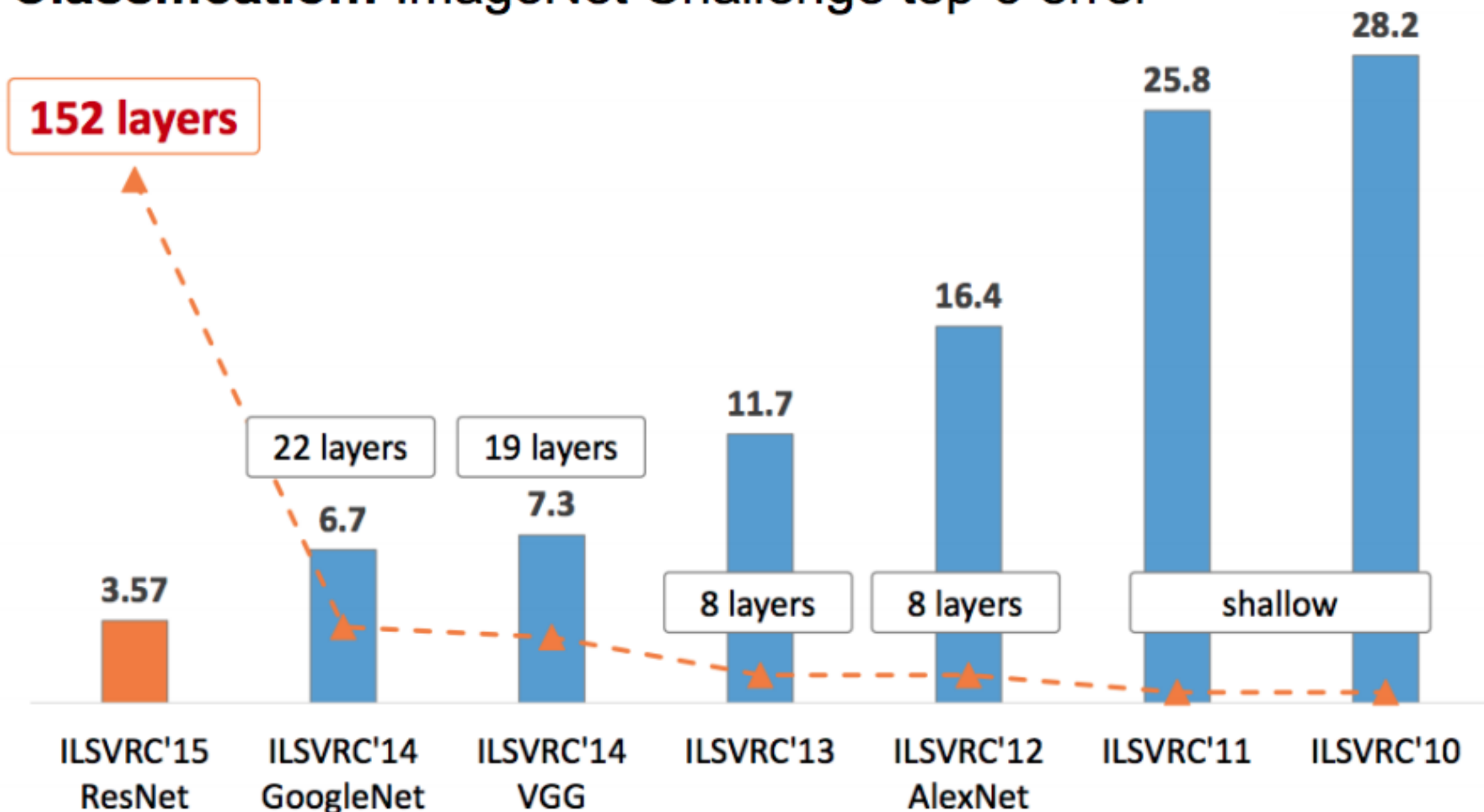
Pre-Trained Model 사전학습 모델

- A saved network that was previously trained on a large dataset, typically on a large-scale image-classification task. You either use the pretrained model as is or use transfer learning to customize this model to a given task. 대규모 이미지 분류 작업에 대해 미리 훈련되고 저장된 네트워크. 사전 학습된 모델을 그대로 사용하거나 전이 학습을 사용하여 이 모델을 지정된 작업에 맞게 바꿀 수 있음
- Used for new and similar problem. 새로운 문제나 비슷한 문제에 사용

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

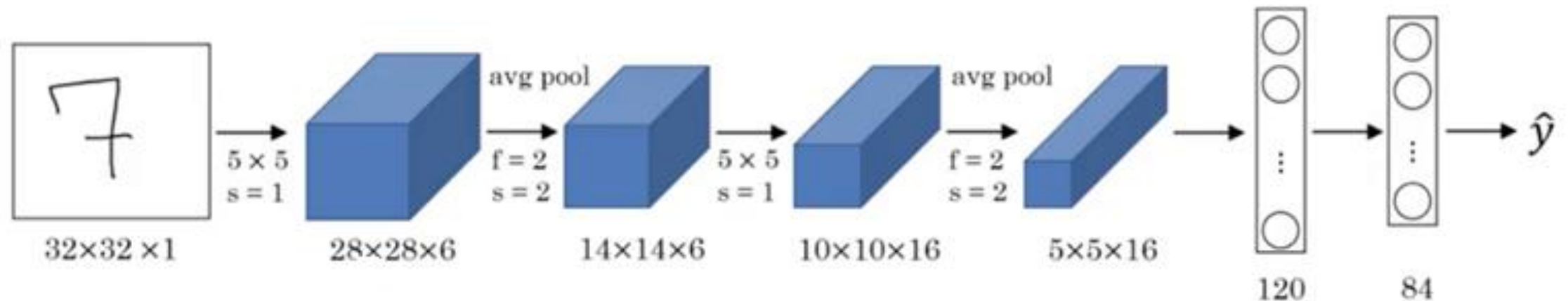
Classification: ImageNet Challenge top-5 error



LeNet-5 리넷

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

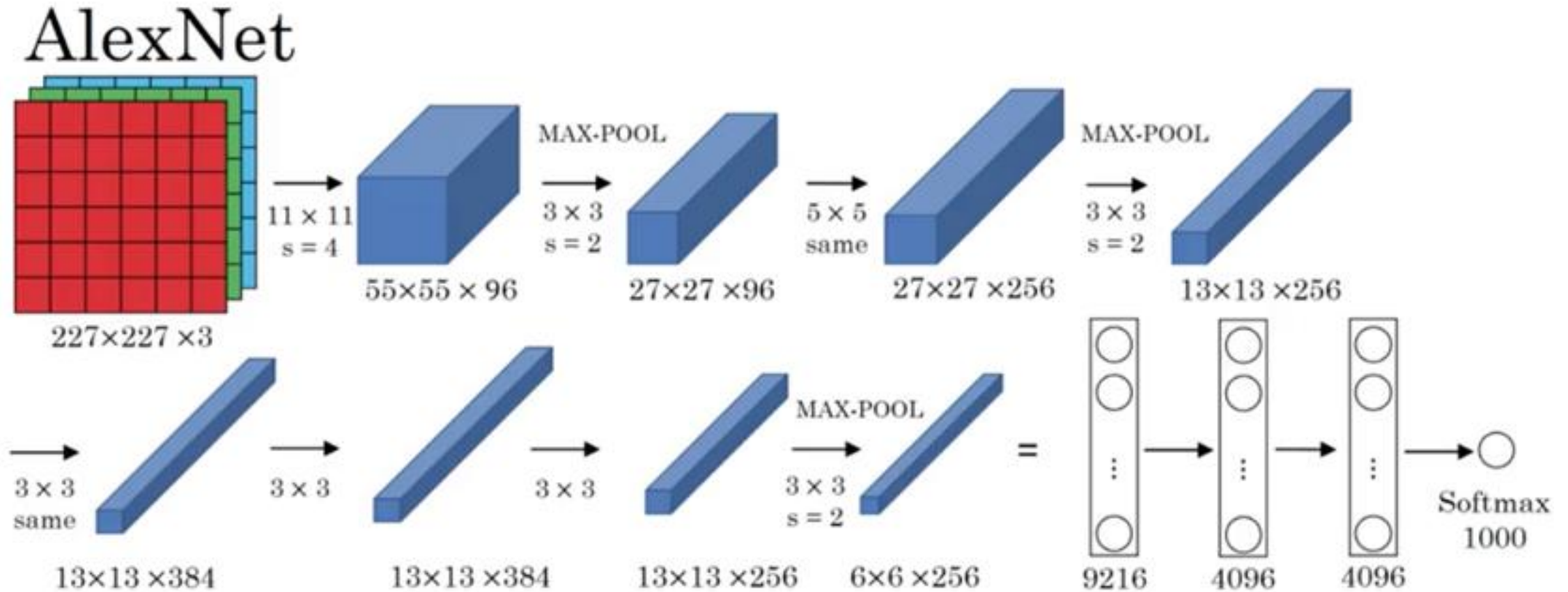
LeNet - 5



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

AlexNet 알렉스넷

Error: - 16.4
8 layers

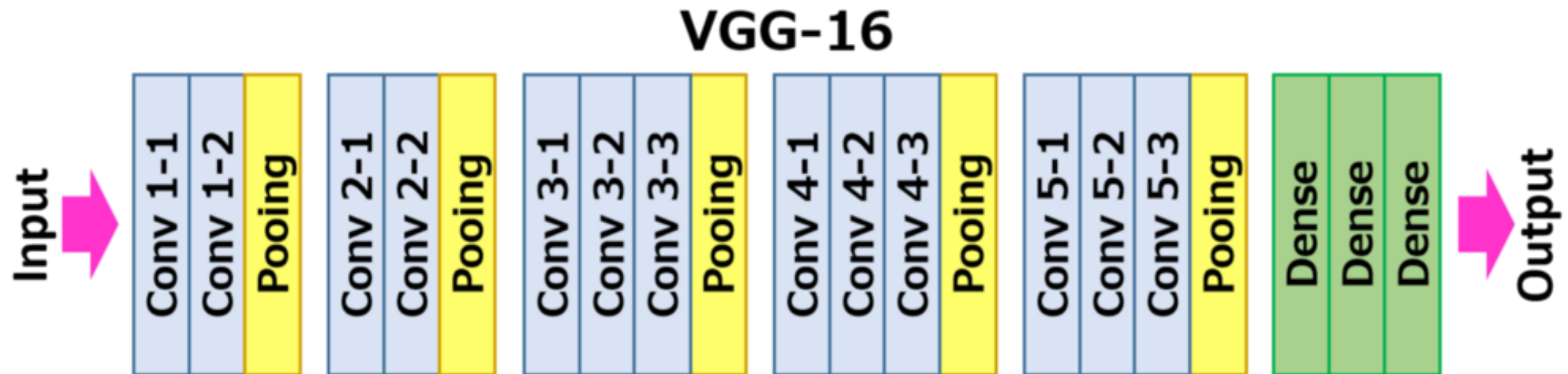


VGG-16

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

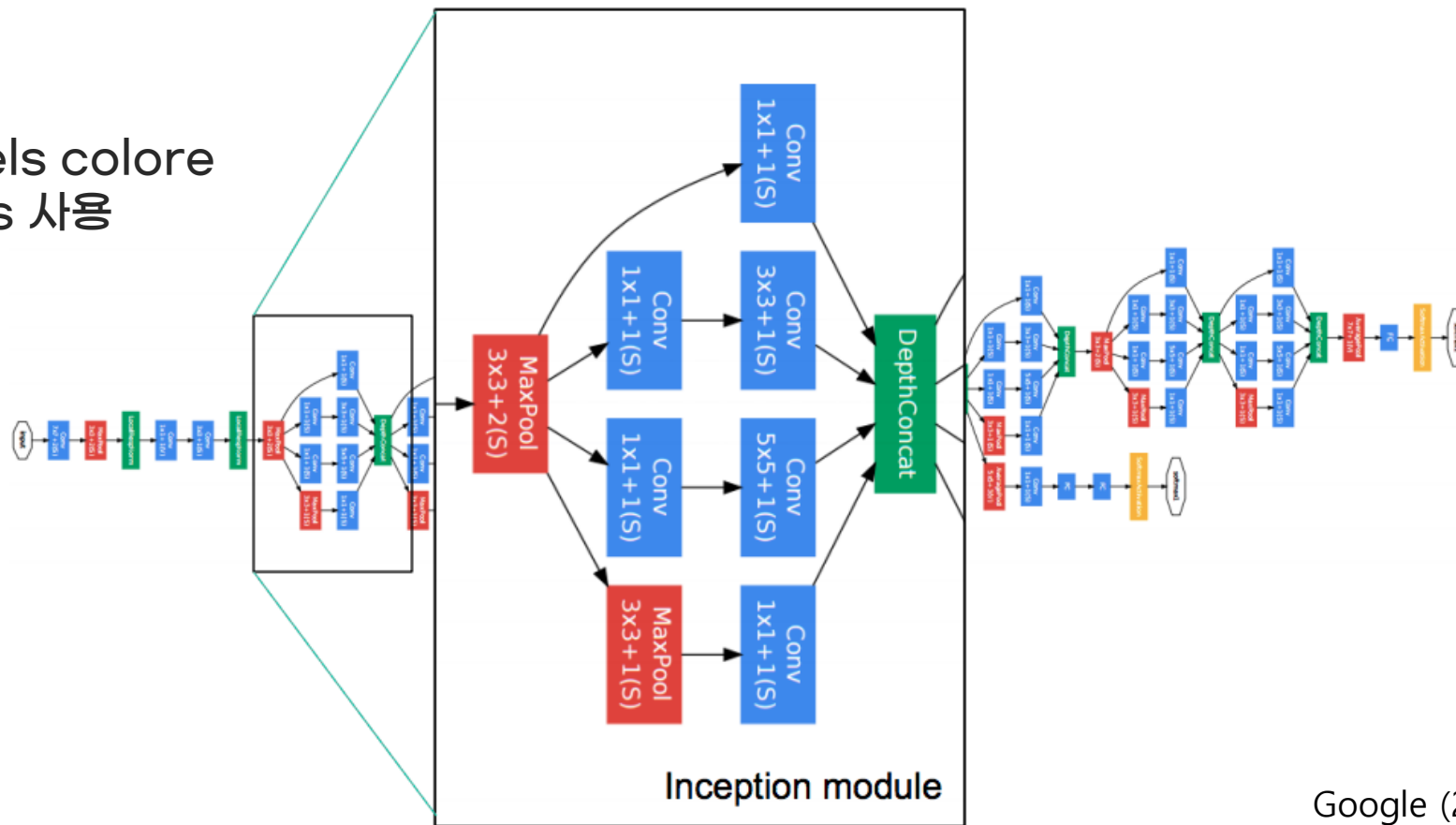


GoogLeNet

Error - 6.7
22 layers

- Also known as Inception Model. 인셉션 모델이라고도 함

299x299 pixels colored images 사용



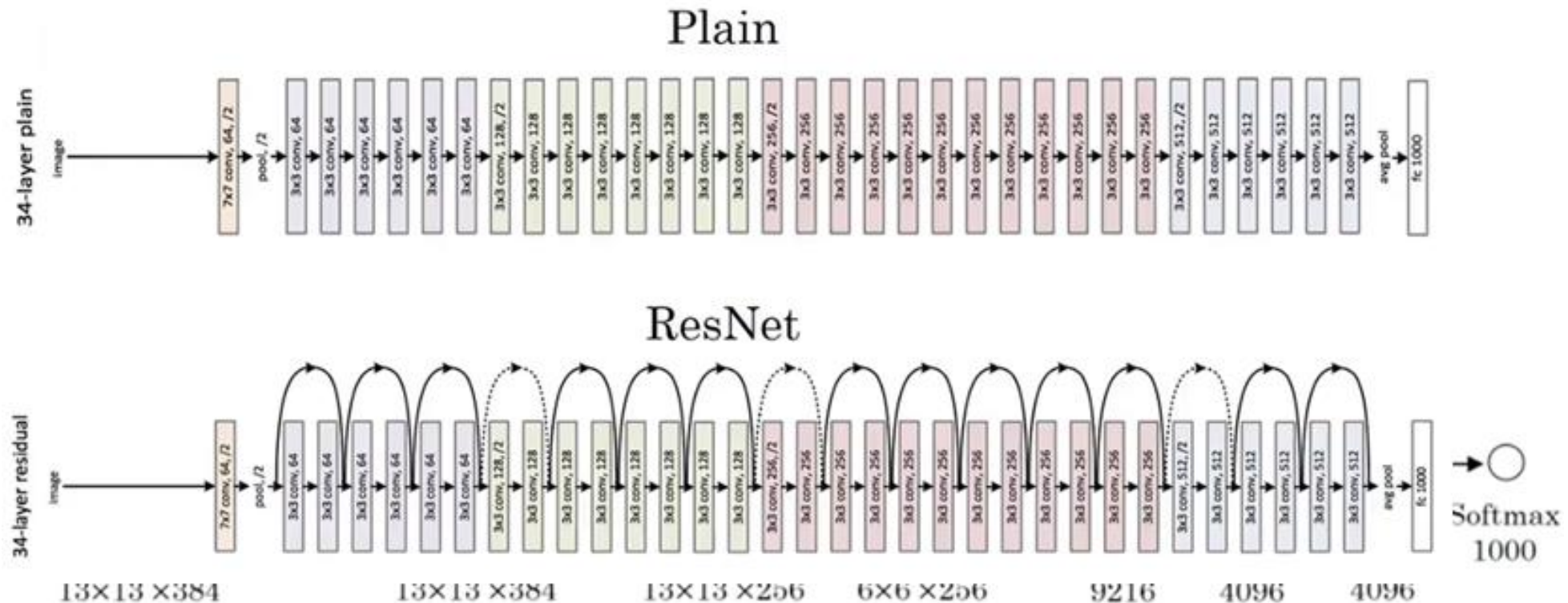
Google (2014)

ResNet (Residual Neural Network)

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

Error - 3.57

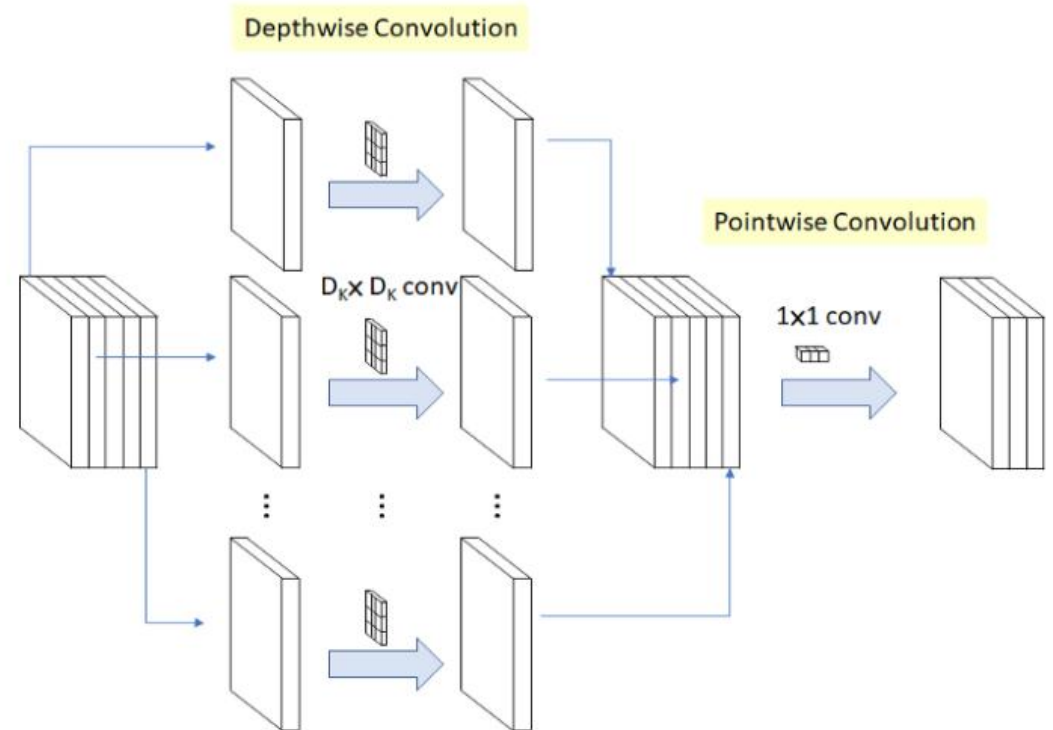
224x224 pixels colored images



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 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

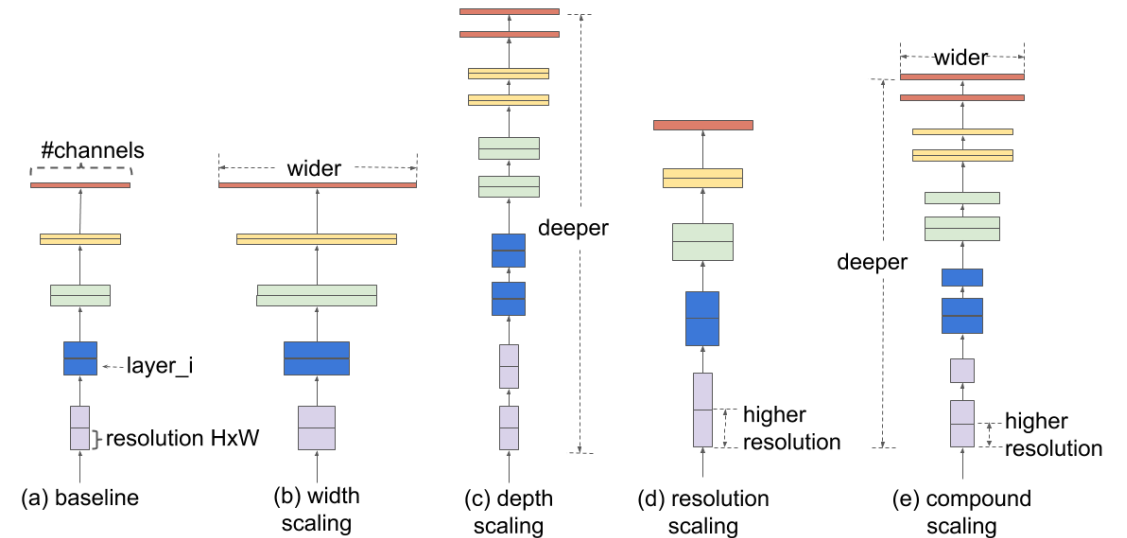
Table 9. Smaller MobileNet Comparison to Popular Models

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	-

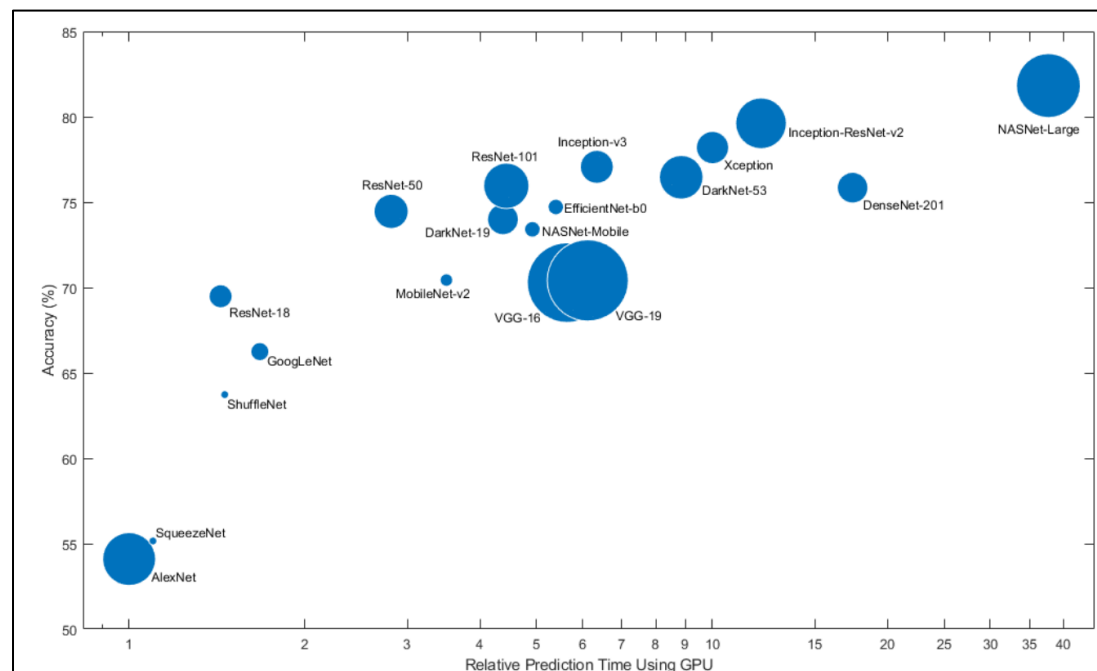
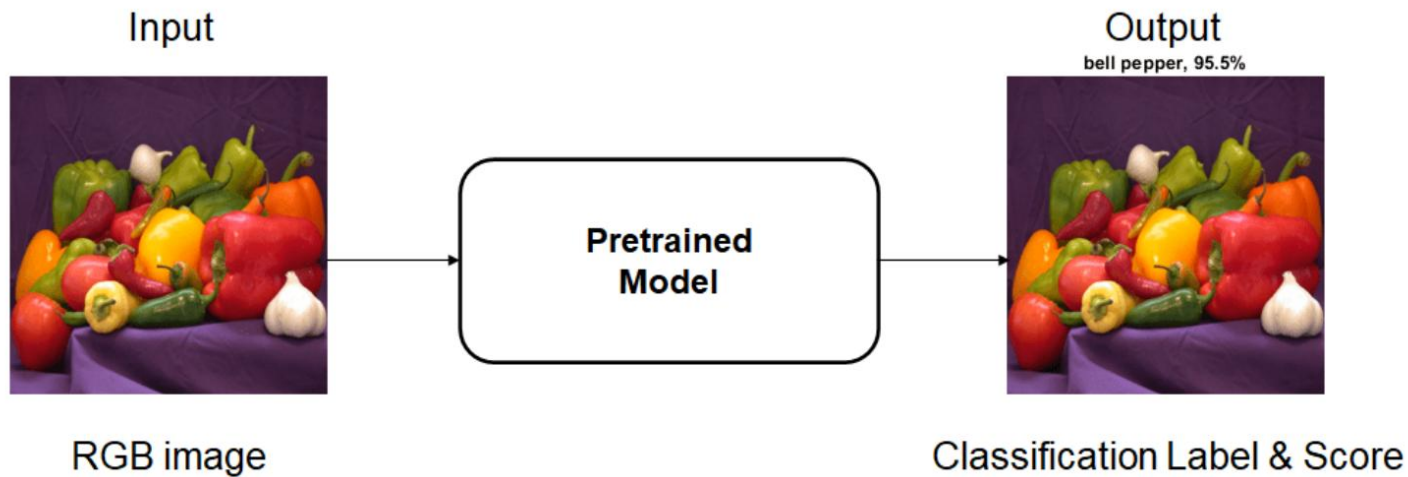


Image Classification 이미지 분류

- Extracts features from natural images. 자연 이미지에서 특징을 추출

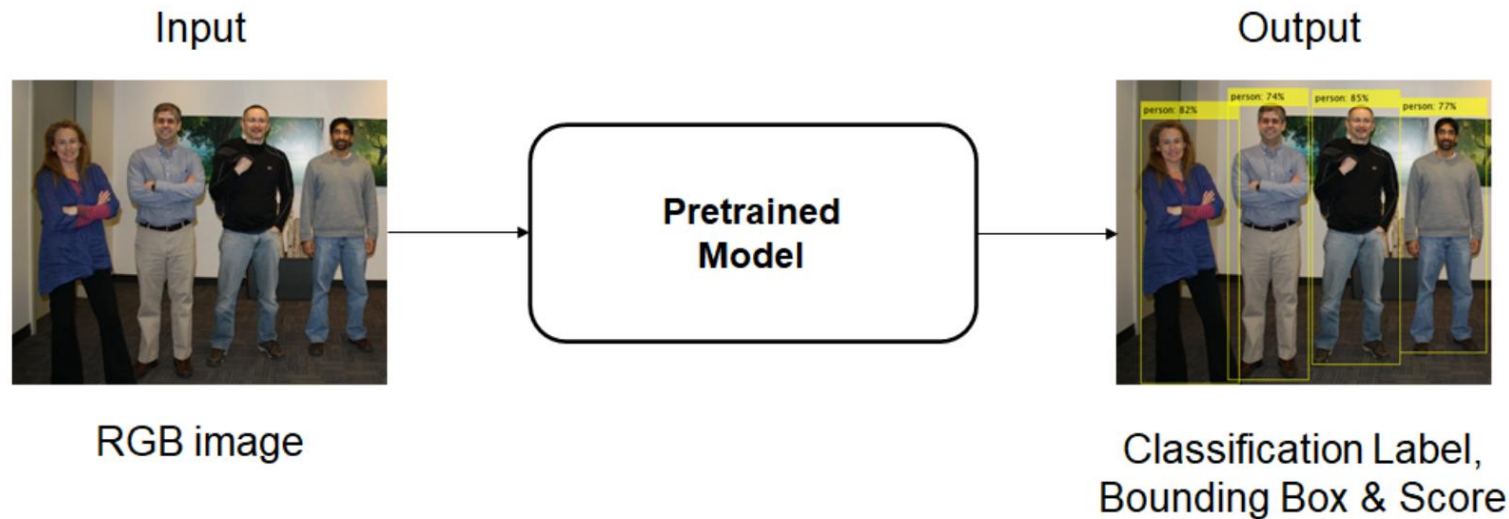


Network	Size (MB)	Classes	Accuracy %	Location
googlenet¹	27	1000	66.25	Doc GitHub
squeezenet¹	5.2	1000	55.16	Doc
alexnet¹	227	1000	54.10	Doc
resnet18¹	44	1000	69.49	Doc GitHub
resnet50¹	96	1000	74.46	Doc GitHub
resnet101¹	167	1000	75.96	Doc GitHub
mobilenetv2¹	13	1000	70.44	Doc GitHub
vgg16¹	515	1000	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
Vision Transformer	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

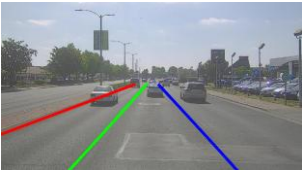
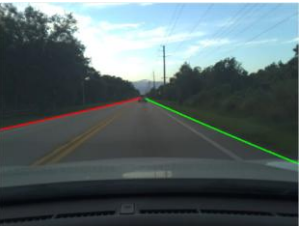


Object Detection 물체 감지

- 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

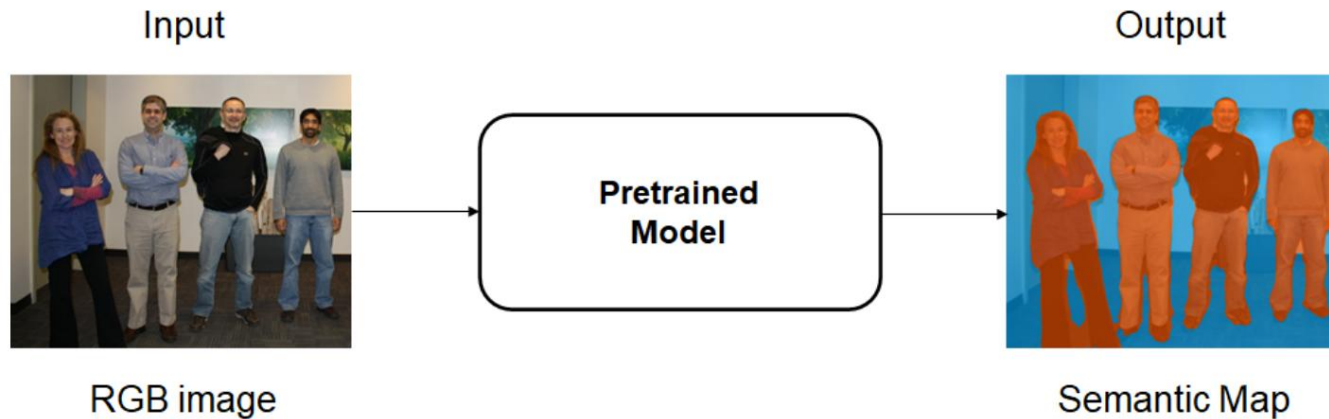
Application Specific Object Detectors 응용 분야별 물체 감지기




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

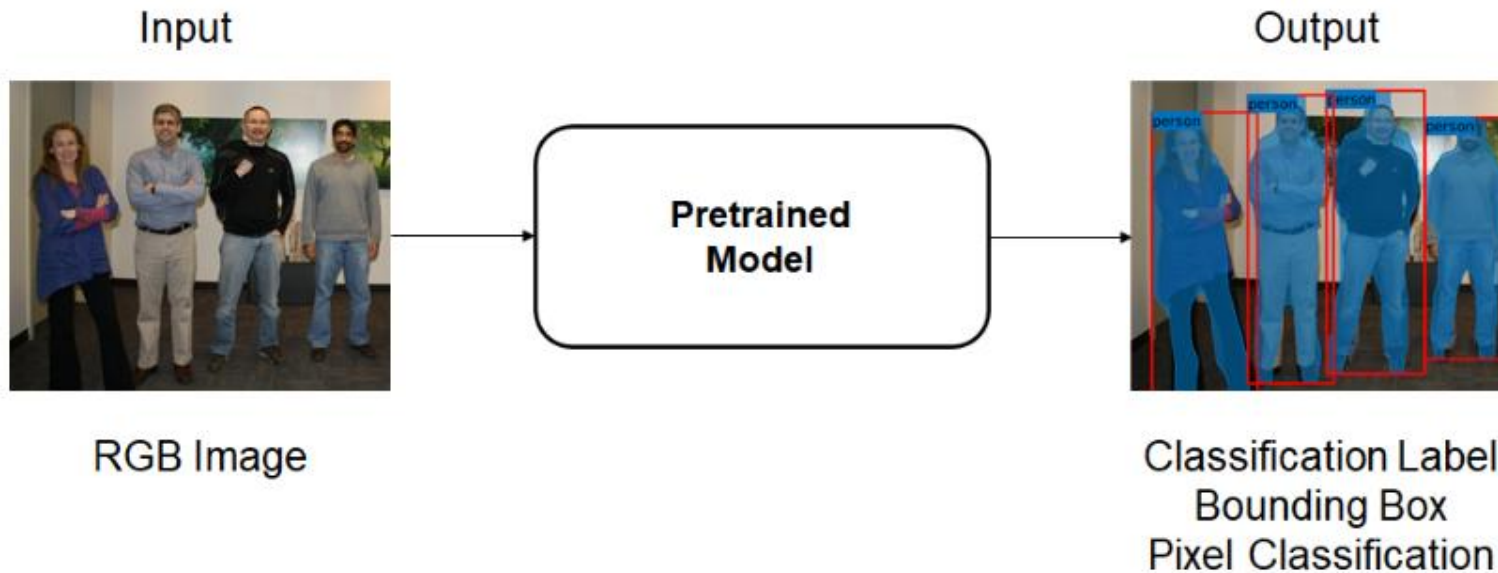
- Describes the process of associating each pixel of an image with a class label, (such as flower, person, road, sky, ocean, or car).
 이미지의 각 픽셀을 클래스 레이블 (예: 꽃, 사람, 도로, 하늘, 바다 또는 자동차)과 연결하는 프로세스를 설명



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

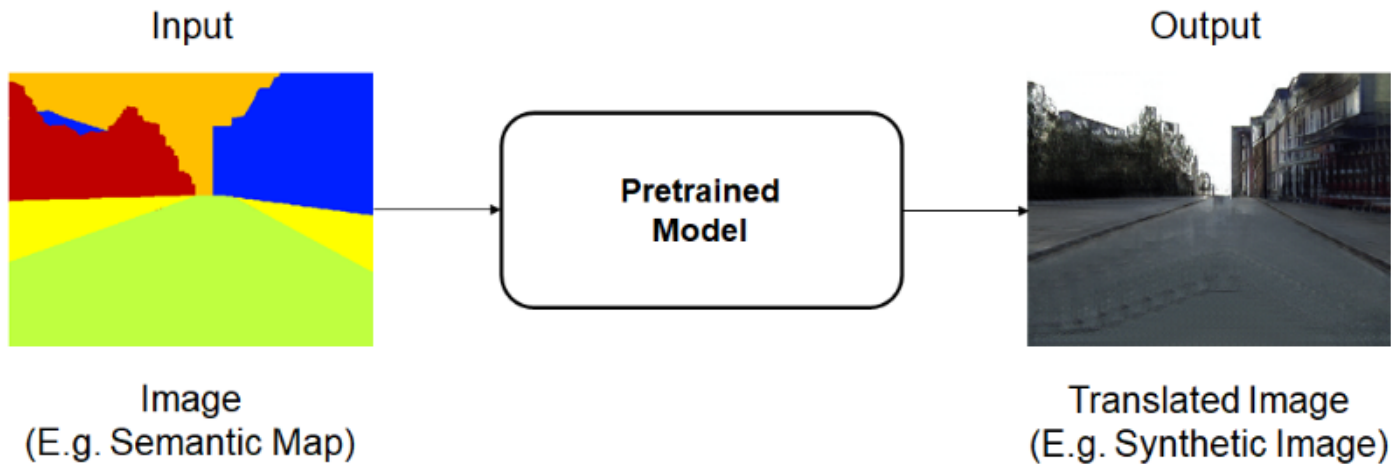


Network	Object Classes	Location
Mask R-CNN	80	Doc Github

Source: Matlab

Image Translation 이미지 번역

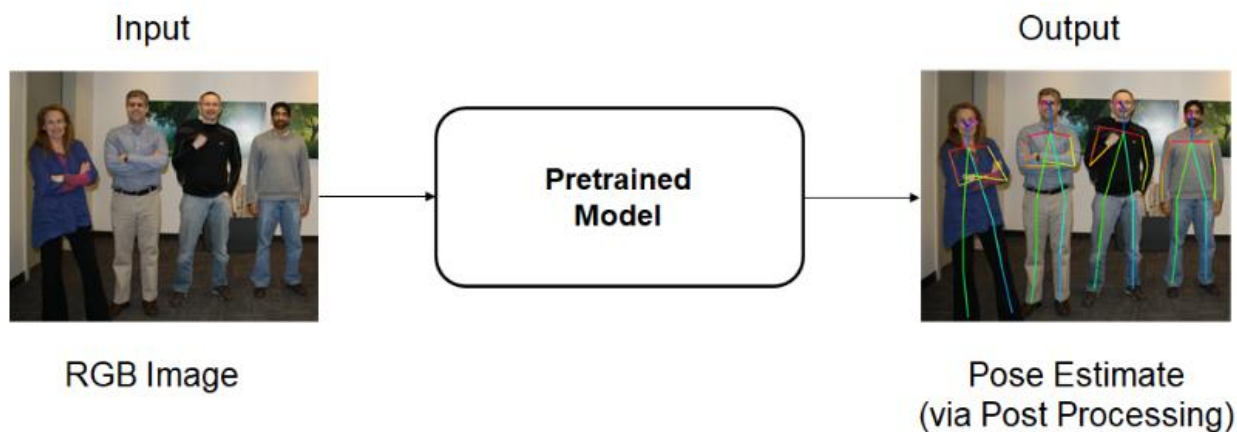
- Transfers styles and characteristics from one image domain to another (e.g., image enhancement, image colorization, defect generation, and medical image analysis). 한 이미지 도메인에서 다른 이미지 도메인으로 스타일과 특성을 변환 (예: 이미지 향상, 이미지 채색, 결함 생성 및 의료 이미지 분석)



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

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

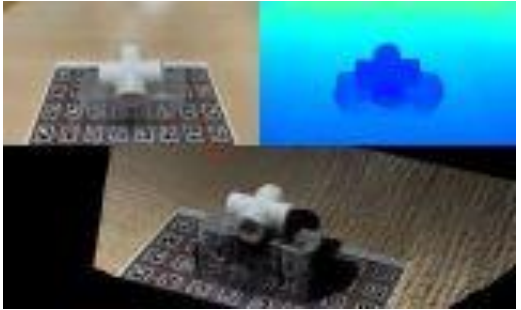


Network	Backbone Networks	Size (MB)	Location	Network
OpenPose	vgg19	14	Doc	OpenPose
HR Net	human-ful l-body-w3 2 human-ful l-body-w4 8	106.9 237.7	Doc	HR Net

Source: Matlab

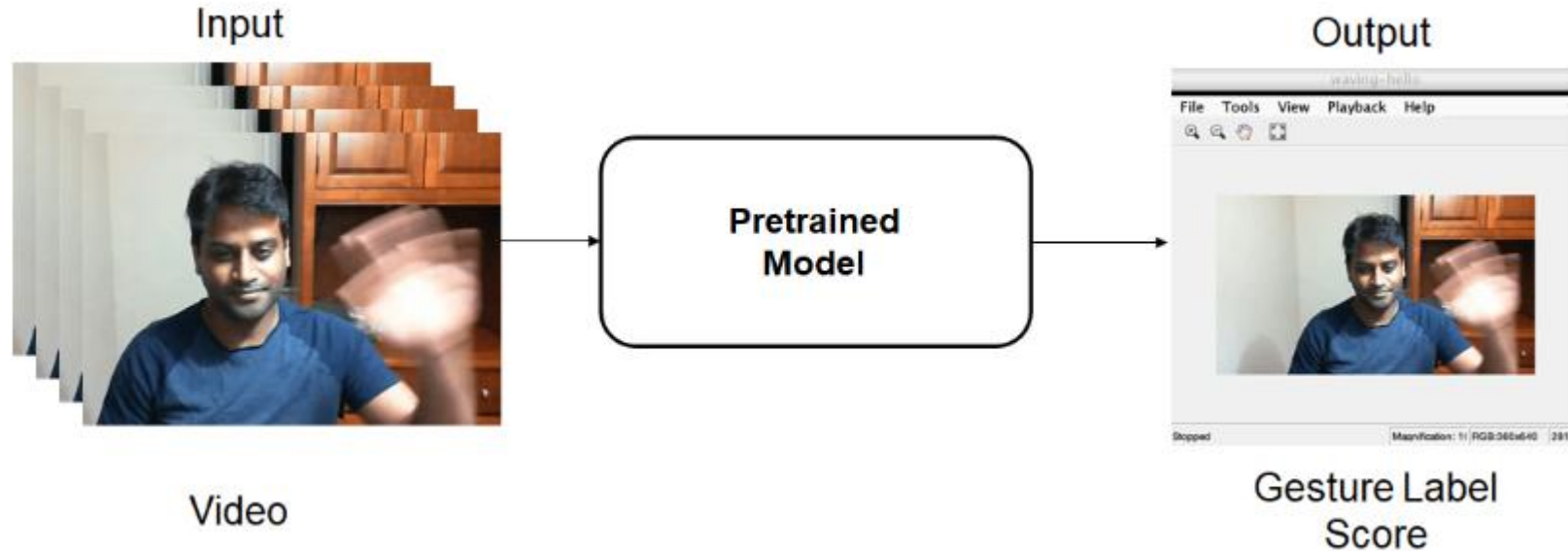
3D Reconstruction 3D 재구성

- 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

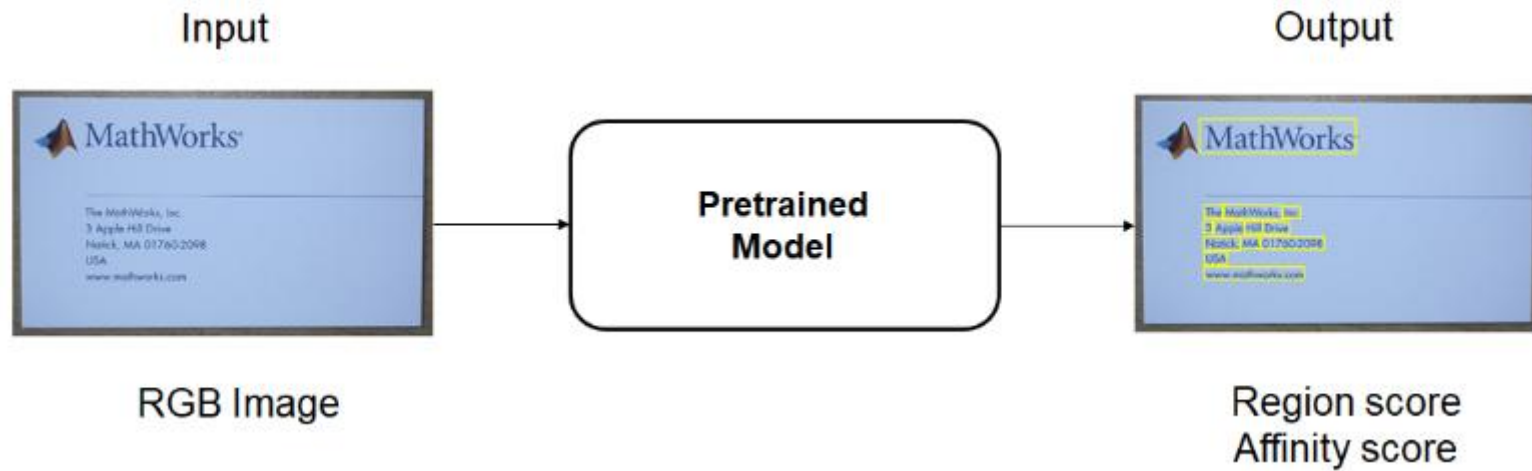


- Classifies the action or content in a sequence of video frames.
일련의 비디오 프레임에서 작업 또는 콘텐츠를 분류


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

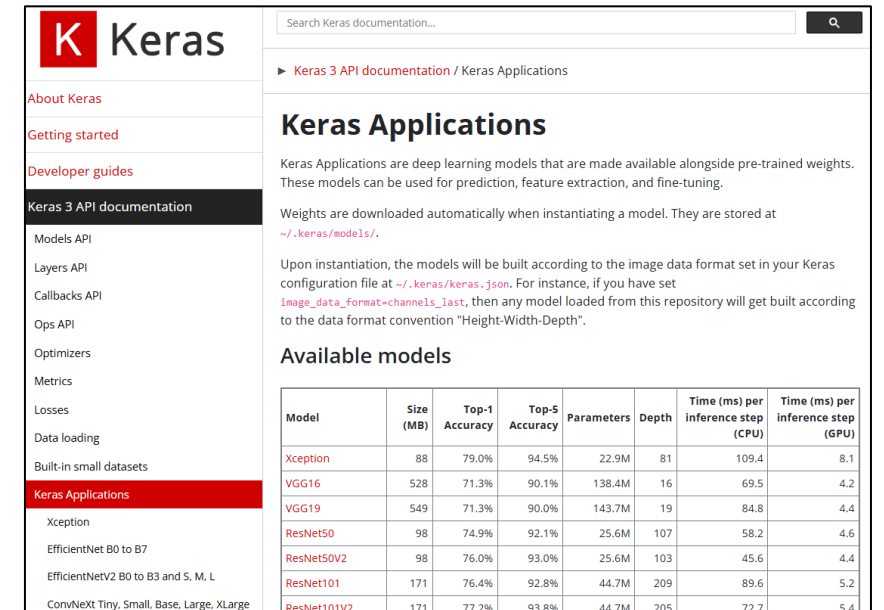


- Locates instances of text within in images. 이미지에서 텍스트의 인스턴스를 찾음

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

Pre-Trained Models in Keras

- Keras Applications
 - Xception
 - EfficientNet B0 to B7
 - VGG16 and VGG19
 - ResNet and ResNetV2
 - MobileNet and MobileNetV2
 - DenseNet
 - NasNetLarge and NasNetMobile
 - InceptionV3
 - InceptionResNetV2



The screenshot shows the Keras Applications documentation page. The left sidebar contains a navigation menu with links to 'About Keras', 'Getting started', 'Developer guides', 'Keras 3 API documentation', 'Models API', 'Layers API', 'Callbacks API', 'Ops API', 'Optimizers', 'Metrics', 'Losses', 'Data loading', 'Built-in small datasets', 'Keras Applications' (highlighted in red), 'Xception', 'EfficientNet B0 to B7', 'EfficientNetV2 B0 to B3 and S, M, L', and 'ConvNext Tiny, Small, Base, Large, XLarge'. The main content area is titled 'Keras Applications' and includes a search bar, a breadcrumb 'Keras 3 API documentation / Keras Applications', and a description of Keras Applications. Below the description is a table titled 'Available models' with columns for Model, Size (MB), Top-1 Accuracy, Top-5 Accuracy, Parameters, Depth, Time (ms) per inference step (CPU), and Time (ms) per inference step (GPU). The table lists models such as Xception, VGG16, VGG19, ResNet50, ResNet50V2, ResNet101, and ResNet101V2.

Keras Applications

Keras Applications are deep learning models that are made available alongside pre-trained weights. These models can be used for prediction, feature extraction, and fine-tuning.

Weights are downloaded automatically when instantiating a model. They are stored at `~/.keras/models/`.

Upon instantiation, the models will be built according to the image data format set in your Keras configuration file at `~/.keras/keras.json`. For instance, if you have set `image_data_format=channels_last`, then any model loaded from this repository will get built according to the data format convention "Height-Width-Depth".

Available models

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	Time (ms) per inference step (CPU)	Time (ms) per inference step (GPU)
Xception	88	79.0%	94.5%	22.9M	81	109.4	8.1
VGG16	528	71.3%	90.1%	138.4M	16	69.5	4.2
VGG19	549	71.3%	90.0%	143.7M	19	84.8	4.4
ResNet50	98	74.9%	92.1%	25.6M	107	58.2	4.6
ResNet50V2	98	76.0%	93.0%	25.6M	103	45.6	4.4
ResNet101	171	76.4%	92.8%	44.7M	209	89.6	5.2
ResNet101V2	171	77.2%	93.8%	44.7M	205	72.7	5.4

<https://keras.io/api/applications/>

Exercise #2

- Complete the tutorial on <https://keras.io/api/applications/>
 - Classify ImageNet classes with ResNet50

```
import keras
from keras.applications.resnet50 import ResNet50
from keras.applications.resnet50 import preprocess_input,
import numpy as np

model = ResNet50(weights='imagenet')

img_path = 'elephant.jpg'
img = keras.utils.load_img(img_path, target_size=(224, 224))
x = keras.utils.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

preds = model.predict(x)
# decode the results into a list of tuples (class, description)
# (one such list for each sample in the batch)
print('Predicted:', decode_predictions(preds, top=3)[0])
# Predicted: [(u'n02504013', u'Indian_elephant', 0.826582),
```

ResNet 모델 구축

```
# 필요한 라이브러리 import
import keras

from keras.applications.resnet50 import ResNet50 #
ResNet50 모델

from keras.applications.resnet50 import preprocess_input,
decode_predictions # 전처리 및 예측 결과 디코딩

import numpy as np # 수치
계산용 라이브러리

# ImageNet 데이터셋으로 사전 학습된 ResNet50 모델 불러오기
model = ResNet50(weights='imagenet')
```

이미지 전처리

예측에 사용할 이미지 파일 경로 지정

```
img_path = 'elephant.jpg'
```

이미지를 불러오고 ResNet50 입력 크기인 (224, 224)로 리사이즈

```
img = keras.utils.load_img(img_path, target_size=(224, 224))
```

이미지를 NumPy 배열로 변환

```
x = keras.utils.img_to_array(img)
```

배열의 차원을 (1, 224, 224, 3)으로 확장 → 배치 크기 포함

```
x = np.expand_dims(x, axis=0)
```

ResNet50에 맞는 전처리 수행 (픽셀 정규화 등)

```
x = preprocess_input(x)
```

테스트 데이터로 예측

모델에 입력 데이터를 넣고 예측 수행

```
preds = model.predict(x)
```

예측 결과를 해석 가능한 형태로 디코딩 (클래스 ID, 설명, 확률)

top=3 → 확률이 높은 상위 3개의 클래스만 추출

```
print('Predicted:', decode_predictions(preds, top=3)[0])
```

예시 출력:

```
# Predicted: [('n02504013', 'Indian_elephant', 0.82658225),
```

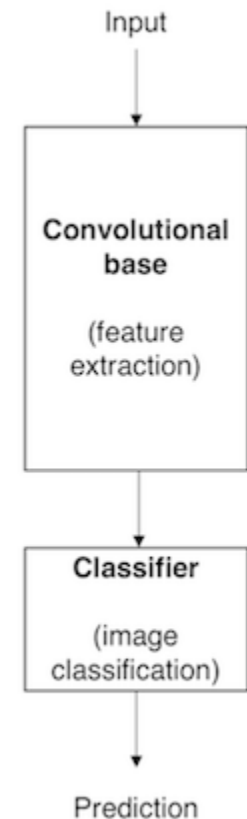
```
#           ('n01871265', 'tusker', 0.1122357),
```

```
#           ('n02504458', 'African_elephant', 0.061040461)]
```

Feature Extractor and Classifier

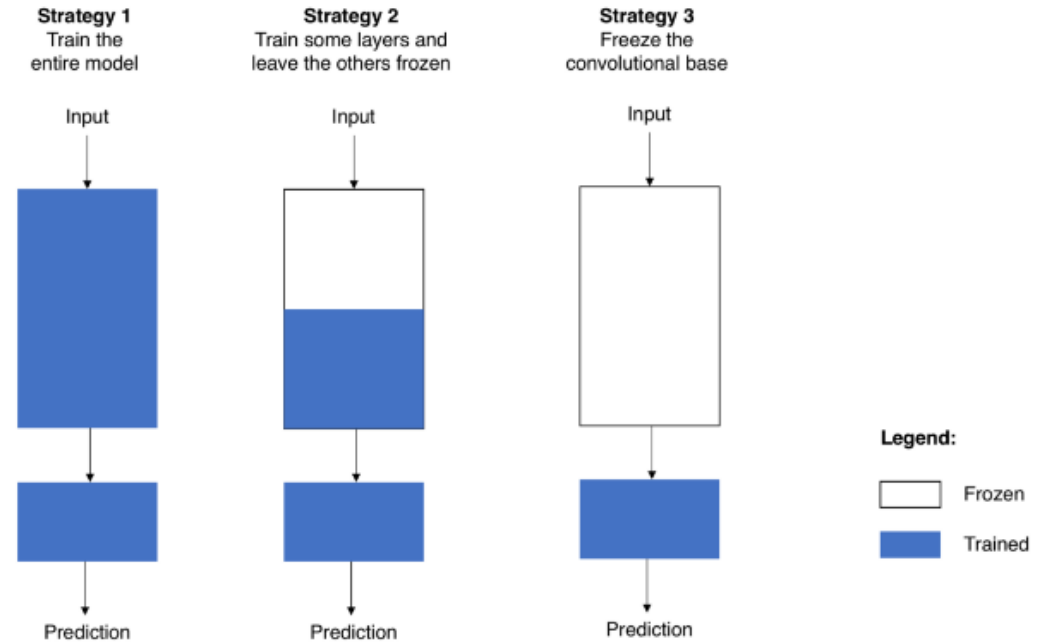
특성추출기와 분류기

- Feature Extractor 특성 추출기
 - 컨볼루션 층과 풀링 층으로 구성되며 ImageNet 데이터로 이미 학습되어 있음
 - 출력데이터: bottleneck 또는 feature vector
- Classifier 분류기
 - Fully connected layer (완전 연결층)으로 구성되며 추출된 특성을 입력 받아 분류를 진행
 - 오버피팅을 줄이기 위해 Dropout, BatchNormalization 레이어 등을 추가



Fine-Tuning 파인 튜닝

- 사전학습 모델의 가중치를 미세하게 조정하는 방법
- 분류하는 데이터의 종류와 갯수에 따라 사전학습 모델의 가중치 일부를 재학습 시키거나 모든 가중치를 처음부터 다시 학습시킬 수 있음



전체를 다 훈련할 수 있게 함

```
base_model.trainable = True
```

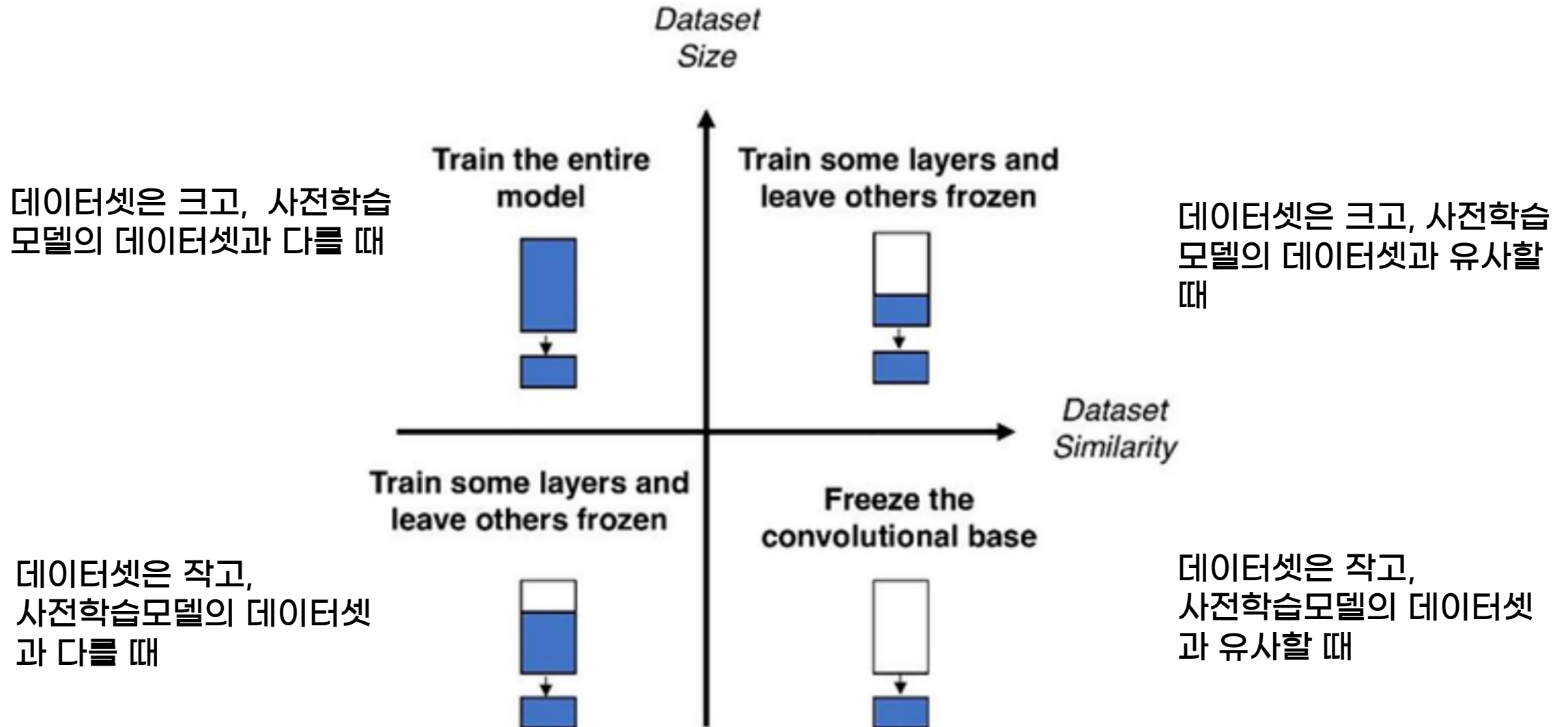
일부 특성추출기를 동결

```
for layer in base_model.layers[143:]:  
    layer.trainable = True
```

특성추출기를 동결

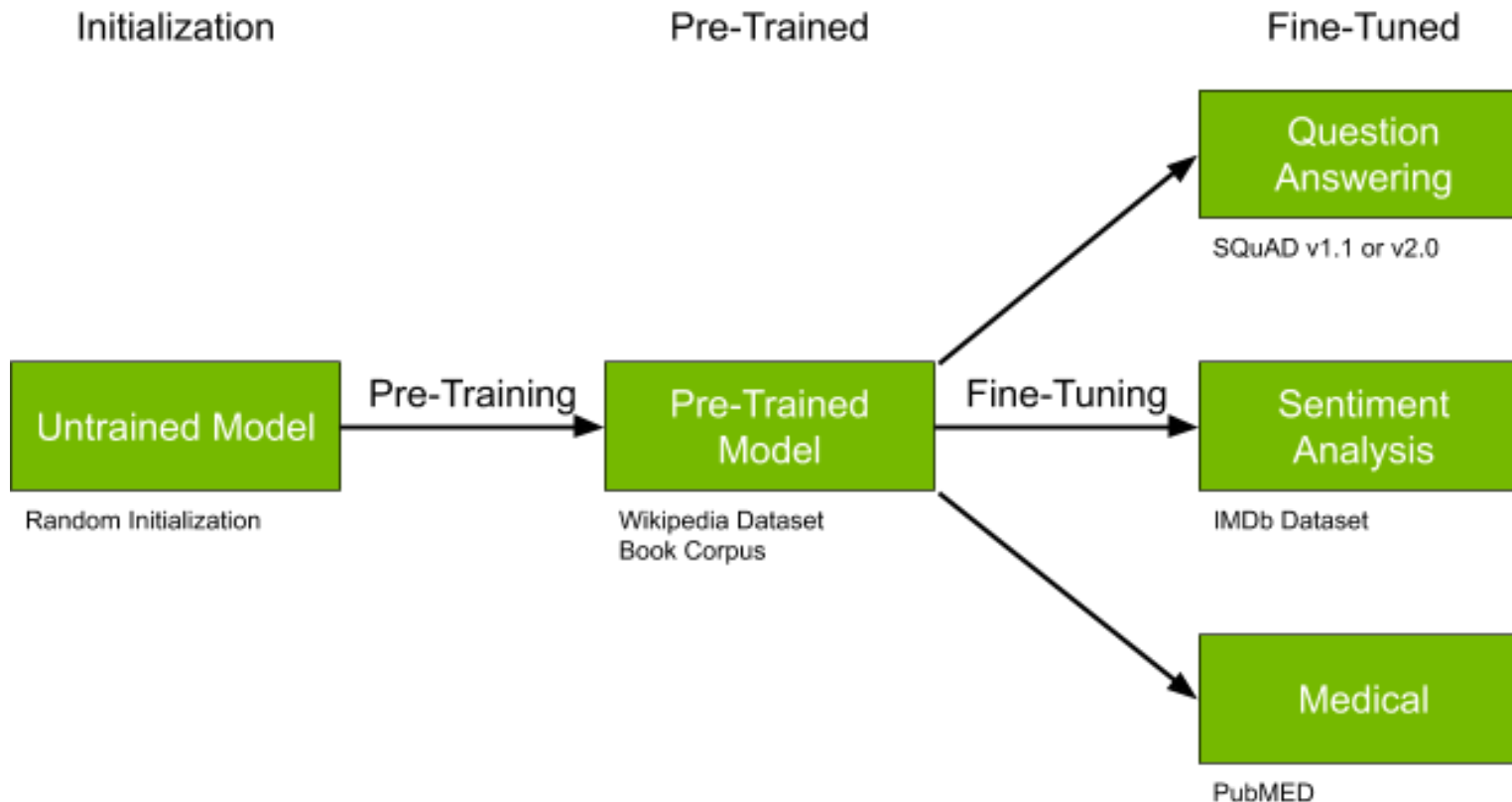
```
base_model.trainable = False
```

데이터 크기와 유사도에 따른 전이학습의 사용



Fine-Tuning Example

- BERT (Bidirectional Encoder Representations from Transformers)의 핵심 모델은 대규모의 일반 데이터 세트에서 사전 훈련한 다음 질문/답변, 감정 분석 또는 명명된 엔터티 인식과 같은 다양한 작업을 수행하도록 미세 조정할 수 있음



Typical Fine-Tuning Workflow

1. 기본 모델을 개체화하고 사전 훈련된 가중치를 로드
2. `trainable = False`를 설정하여 기본 모델의 모든 레이어를 동결
3. 기본 모델의 하나 또는 여러 레이어의 출력 위에 새 모델을 생성
4. 새로운 데이터 세트에 대한 새로운 모델을 훈련
5. 모델이 새 데이터에 수렴되면 기본 모델의 전체 또는 일부를 해제하고 매우 낮은 학습률로 전체 모델을 다시 훈련할 수 있음

특성 추출기만 가져오기

- Instantiate a base model and load pre-trained weights into it. 기본 모델을 개체화하고 사전 훈련된 가중치를 로드

```
from tensorflow.keras.applications import *  
  
base_model = VGG16(weights='imagenet',  
                    input_shape=(240, 240, 3),  
                    include_top=False) # 특성추출기만 사용  
base_model.summary()
```

특성 추출기, 분류기 가져오기

```
from tensorflow.keras.applications.inception_v3 import InceptionV3
base_model = InceptionV3(input_shape=(150, 150, 3),
                          include_top=True, #사전학습 모델의 특성추출기와
                                             분류기 둘다 가져옴
                          weights=None) #적용할 가중치 지정
base_model.summary()
```

Weights

- None: 임의로 초기화 (random initialization)된 가중치를 적용
- imagenet: ImageNet에 대해 사전 훈련된 가중치를 적용 (Default 값)

Base Model 구조 확인

```
from tensorflow.keras.applications import *
```

```
base_model = VGG16(weights='imagenet',  
                    input_shape=(240, 240, 3),  
                    include_top=False) # 특성추출기만 가져옴
```

```
base_model.input # 입력 텐서
```

```
<KerasTensor: shape=(None, 240, 240, 3) dtype=float32 (created by layer 'input_1')>
```

```
base_model.output # 출력 텐서
```

```
<KerasTensor: shape=(None, 7, 7, 512) dtype=float32 (created by layer 'block5_pool')>
```

```
print(base_model.layers[-1]) # 마지막 레이어
```

```
<keras.src.layers.pooling.max_pooling2d.MaxPooling2D at 0x7802582870a0>
```

```
print(base_model.layers.pop()) #마지막 레이어 제거
```

```
<keras.src.layers.pooling.max_pooling2d.MaxPooling2D at 0x7802582870a0>
```

전이학습 모델 위에 새로운 분류기를 추가

- Create a new model on top of the output of one (or several) layers from the base model. 기본 모델의 하나 또는 여러 레이어의 출력 위에 새 모델을 생성

```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout
from keras.optimizers import Adam

model = Sequential()
model.add(base_model)
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer=Adam(learning_rate=1e-4), metrics=['accuracy'])
model.summary()
```

Fine-tuning시 학습
률을 낮게 설정해
pre-trained
weights를 조금씩
업데이트 해줌

모델 컴파일

- Train your new model on your new dataset. 새로운 데이터 세트에 대한 새로운 모델을 훈련

모델 컴파일

```
model.compile(optimizer = 'adam',  
              loss = 'sparse_categorical_crossentropy',  
              metrics=['accuracy'])
```

가중치의 훈련가능 여부

- 훈련하기 전 base model 의 가중치가 바뀌는 것을 막기 위해 이를 동결

- 베이스 모델 전체를 동결하는 경우

```
base_model.trainable=False
```

- 각 레이어를 순차적으로 동결하는 경우

```
for layer in base_model.layers:  
    layer.trainable = False
```

모든 레이어 동결

- Freeze all layers in the base model by setting trainable = False.
훈련 가능 = False를 설정하여 기본 모델의 모든 레이어를 동결

```
# 전체 모델의 가중치 동결
base_model.trainable = False

# 훈련 가능한 가중치의 갯수 확인
len(model.trainable_weights) #4
```


동결 해제

- Once your model has converged on the new data, you can try to unfreeze all or part of the base model and retrain the whole model end-to-end with a very low learning rate. 모델이 새 데이터에 수렴되면 기본 모델의 전체 또는 일부를 해제하고 매우 낮은 학습률로 전체 모델을 다시 훈련할 수 있음

모든 층의 동결 해제

```
base_model.trainable = True
```

훈련할 수 있는 가중치 갯수

```
len(model.trainable_weights) #30
```

일부 레이어 해제

```
for layer in base_model.layers[143:]:  
    layer.trainable = True
```

가중치 저장 및 조기종료 설정

```
from keras.callbacks import ModelCheckpoint, EarlyStopping

save_file_name = 'cifar10_transfer_model.h5'
checkpoint = ModelCheckpoint(save_file_name, monitor='val_loss',
                             verbose=1, save_best_only=True,
                             mode='auto')
earlystopping = EarlyStopping(monitor='val_loss', patience=5)
```

모델 훈련

- Train your new model on your new dataset. 새로운 데이터 세트에 대한 새로운 모델을 훈련

```
# 모델 훈련
```

```
history = model.fit(x_train, y_train, epochs = 100, validation_data=(x_test, y_test), batch_size= 256, callbacks=[checkpoint, earlystopping])  
model.save("cifar10_resnet_model.h5")
```

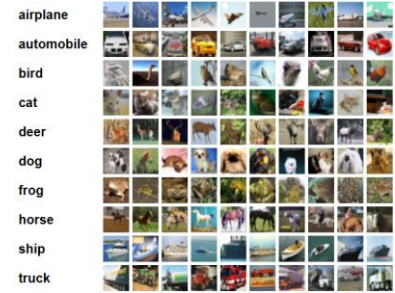
전이학습 모델 튜닝기법

- 옵티마이저 변경
- BatchNormalization 추가
- Dropout 추가
- 사전학습 모델 전체 학습
 - `base_model.trainable = True`
- 데이터 전처리 및 증강
- 분류기에 은닉층 및 노드 추가
- Learning decay를 이용한 유동적인 학습 진행

Learning decay:

- 학습률(learning rate)을 점진적으로 감소
- 초반엔 빠르게 학습. 후반엔 작게 조정하여 안정적으로 수렴

Exercise #3



- CIFAR-10 데이터를 이용하여 전이학습을 수행하시오
- ResNet50의 특성추출기와 분류기를 그대로 이용하여 데이터를 예측하시오 (컴파일 필요, 필요한 경우 이미지 차원 및 사이즈 변경)
- ResNet50의 특성추출기만 가져와서 모델을 만든 후 다음 학습을 수행하시오
 - 특성추출기 동결 후 학습
 - 특성추출기 143층부터 해제한 후 학습 (0.001 학습률)
- ResNet50의 특성추출기만 가져와서 모델을 만든 후 데이터를 증강한 후 다음 학습을 수행하시오
 - 특성추출기 동결 후 학습

사전학습 모델 가져오기 (ResNet50)

- ResNet50의 특성추출기와 분류기를 그대로 이용

```
from tensorflow.keras.applications.resnet50 import  
ResNet50
```

베이스 모델 가져오기

```
base_model = ResNet50(input_shape=(224, 224, 3),  
weights='imagenet', include_top=True) # 특성추출기, 분류기 다  
가져옴
```

- ResNet50의 특성추출기만 가져오기

베이스 모델 가져오기

```
base_model = ResNet50(input_shape = (32, 32, 3), weights =  
'imagenet', include_top=False) # 특성추출기만 가져옴
```

사전학습 모델 가져오기 (기타)

```
from tensorflow.keras.applications import *
```

```
mobilenet = MobileNet(weights=None, input_shape=None,  
include_top=True)
```

```
efficientnet = EfficientNetB0(weights=None, input_shape=None,  
include_top=True)
```

```
resnet50 = ResNet50(weights=None, input_shape=None,  
include_top=True)
```

```
xception = Xception(weights=None, input_shape=None,  
include_top=True)
```

이미지 차원 및 사이즈 변경

```
from tensorflow.keras.applications.resnet50 import preprocess_input

inputs = tf.expand_dims(x_test[0], axis=0) #차원추가 (None,224,224,3)
x = tf.keras.layers.experimental.preprocessing.Resizing(224,
224)(inputs) #사이즈 변경
x = preprocess_input(x) #정규화
```


예측 및 시각화

```
from tensorflow.keras.applications.resnet50 import  
decode_predictions
```

예측

```
preds = base_model.predict(x)
```

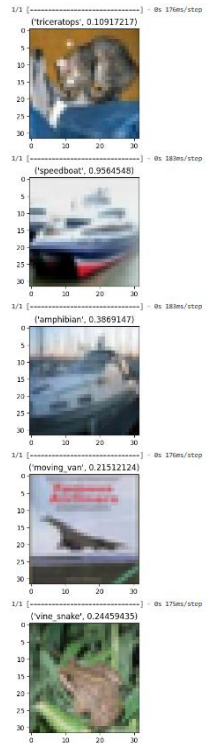
시각화

```
fig= plt.figure(figsize=(3,3))
```

```
plt.title(decode_predictions(preds, top=1)[0][0][1:3]) # 가장 확률이  
높은 예측 이름 추출해서 타이틀로 넣음
```

```
plt.imshow(x_test[0])
```

```
plt.show()
```



전이학습 모델 + 전처리

사이즈를 바꿔서 모델 생성

```
import tensorflow as tf

from keras import Input, Model

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


inputs = Input(shape=(32, 32, 3))

x = tf.keras.layers.experimental.preprocessing.Resizing(224, 224)(inputs)

x = tf.keras.applications.resnet50.preprocess_input(inputs)

x = base_model(x, training = False)

x = Flatten()(x)

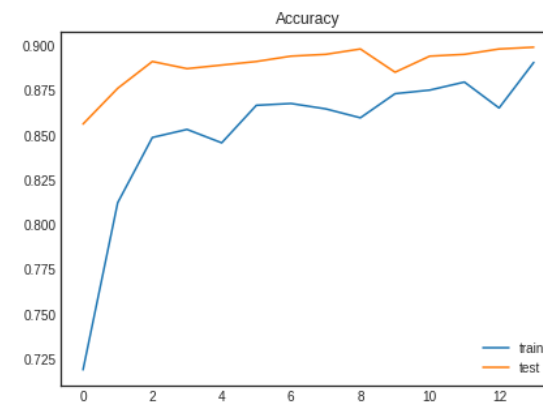
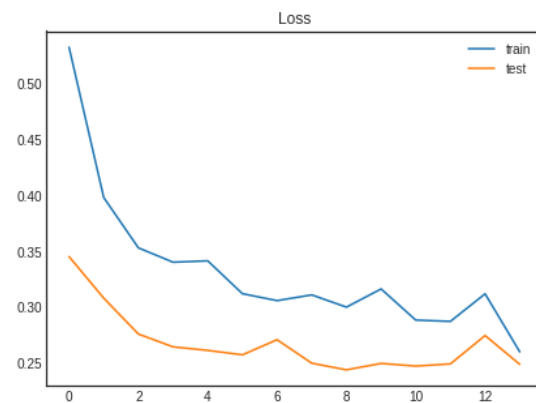
outputs = Dense(10, activation = 'softmax')(x)

model = Model(inputs, outputs)

model.summary()
```

히스토리 그래프

```
plt.figure(figsize=(15,5))
plt.subplot(121)
plt.title('Loss')
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='test')
plt.legend()
plt.subplot(122)
plt.title('Accuracy')
plt.plot(history.history['acc'], label='train')
plt.plot(history.history['val_acc'], label='test')
plt.legend()
plt.show()
```



학습률 조정

```
# 모델 컴파일 (학습률 0.001)
```

```
from tensorflow.keras.optimizers import Adam
```

```
model.compile(optimizer = Adam(learning_rate= 0.001),  
              loss = 'sparse_categorical_crossentropy',  
              metrics=['accuracy'])
```

ChatGPT로 성능 개선 전략 탐색

- 모델 구조 개선 프롬프트 예

"ResNet50 기반 CNN 모델의 구조를 개선하고 싶은데, 어떤 레이어를 추가하면 좋을까요?"

- 하이퍼파라미터 튜닝 프롬프트 예

"Adam 옵티마이저를 사용할 때, 학습률을 어떻게 설정하는 게 좋을까요?"

- 전이학습 세부 설정 프롬프트 예

"ResNet50에서 어느 layer부터 훈련을 다시 시작하는 것이 좋을까요?"

- 데이터 증강 전략 프롬프트 예

"과적합이 있는 모델에 효과적인 이미지 증강 전략을 추천해줘."