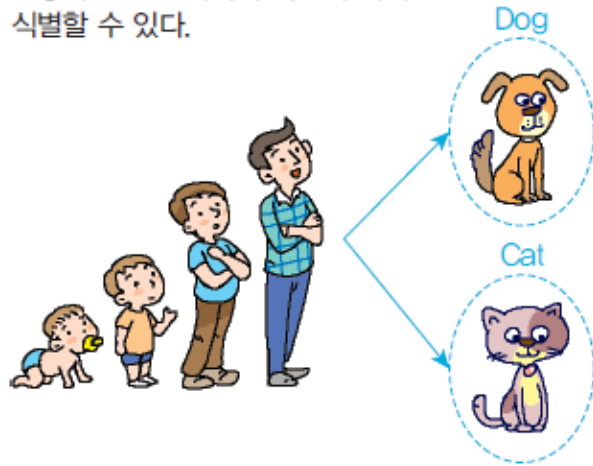


10장 영상인식

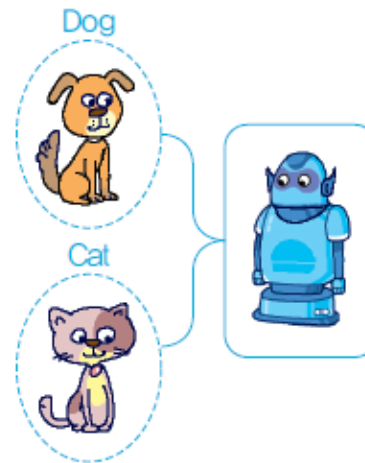
영상 인식이란?

- 영상 인식(image recognition) : 영상 안의 물체를 인식하거나 분류하는 것

인간은 어렸을 때부터 수많은 강아지와 고양이를 보고 자라기 때문에 차이를 식별할 수 있다.



어떻게 하면 컴퓨터가 식별할 수 있을까?





영상인식 신경망 체험하기

- <https://transcranial.github.io/keras-js/#/>

Keras.js - Run Keras models in t x +

transcranial.github.io/keras-js/#/inception-v3

enter image url
https://farm4.staticflickr.com/3852/14447183450_2d0ff0802b...

select image
fox

use GPU

Keras.js

DEMOS

- Basic Convnet MNIST
- Convolutional VAE MNIST
- AC-GAN MNIST
- ResNet-50 ImageNet
- Inception v3 ImageNet
- DenseNet-121 ImageNet
- SqueezeNet v1.1 ImageNet
- Bidirectional LSTM IMDB
- Image Super-Resolution

LINKS

- GitHub repo
- MD.ai


CONTACT

- Leon Chen
- @transcranial

visualization
Class Activation Mapping

colormap
Transparency

(hover over image to view)



inference time: 551.6 ms (1.8 fps)

red fox	67%
kit fox	13%
grey fox	6%
dhole	0%
lesser panda	0%

InputLayer
shape: [299,299,3]

Conv2D
32 3x3 filters, 2x2
striding, no border
padding

BatchNormalization

Activation
relu



전통적인 영상 인식 시스템의 구조

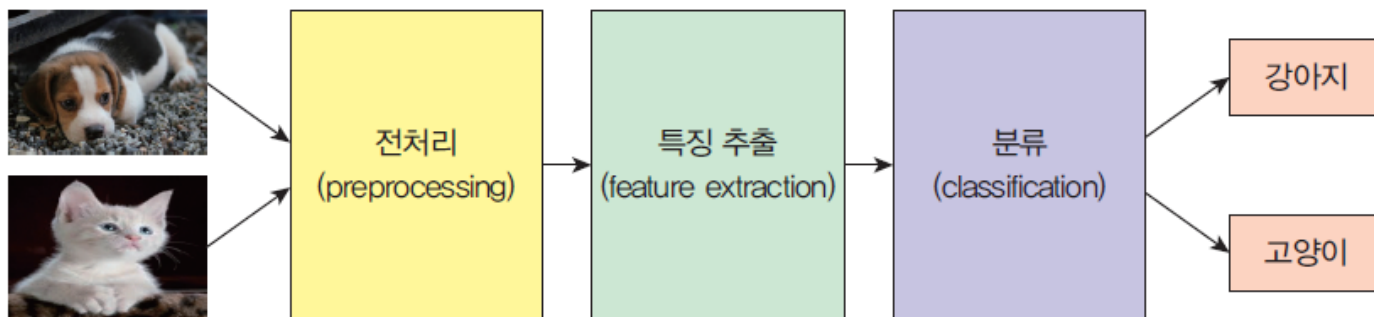


그림 10-3 전통적인 영상 인식 시스템



심층 신경망을 이용한 영상 인식

- 전통적인 영상 인식 방법 - 인간이 특징을 추출하고 특징을 선택하여 분류기로 보낸다.
- 신경망을 이용한 방법 - 특징 추출도 신경망을 통해 수행한다.

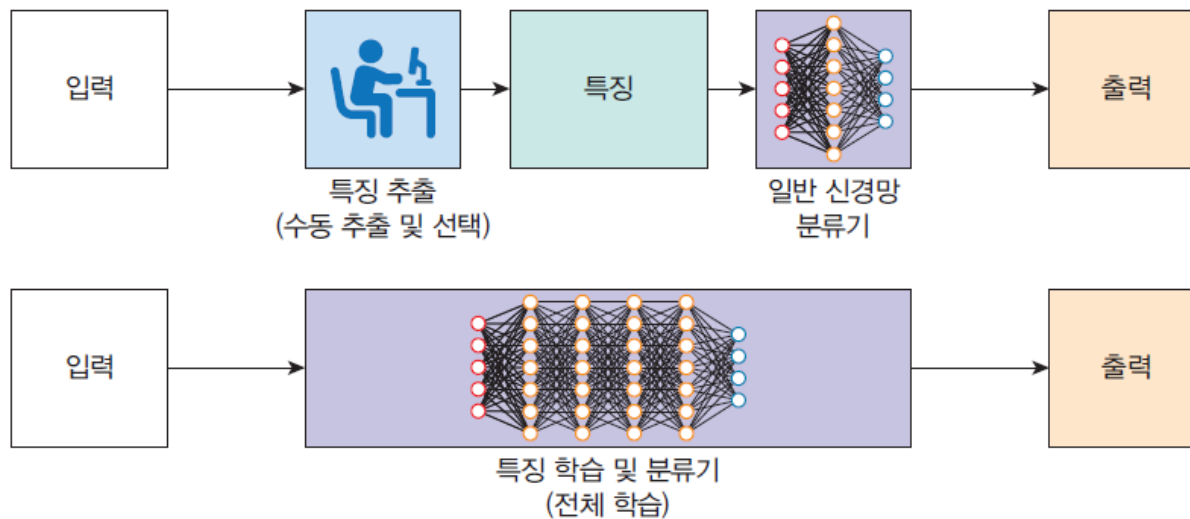


그림 10-9 전통적인 영상 인식과 심층 신경망을 이용한 영상 인식



영상 인식과 컨벌루션 신경망

- 이미지 데이터 처리에 적합
- 2차원 배열을 입력으로 사용. 각 레이어에 일련의 필터를 영상에 적용하여 특징을 추출

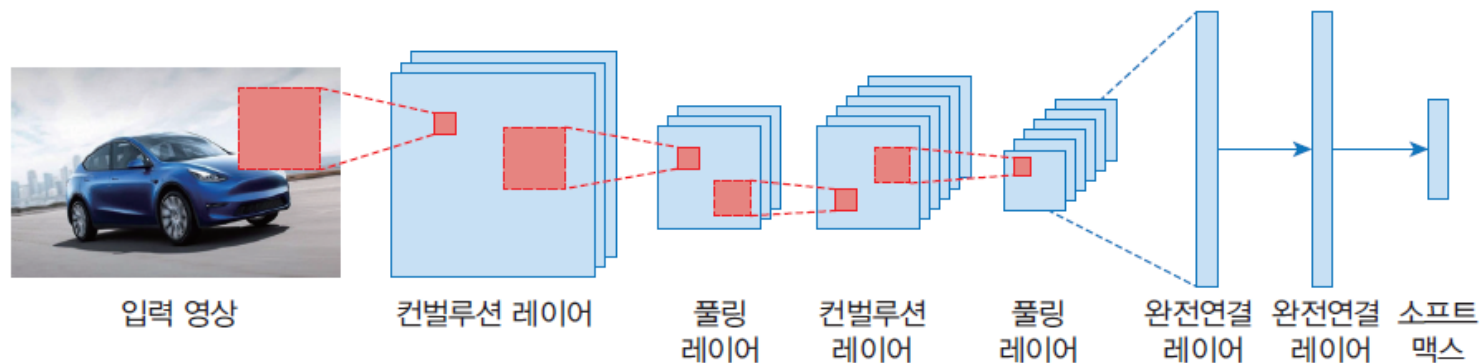
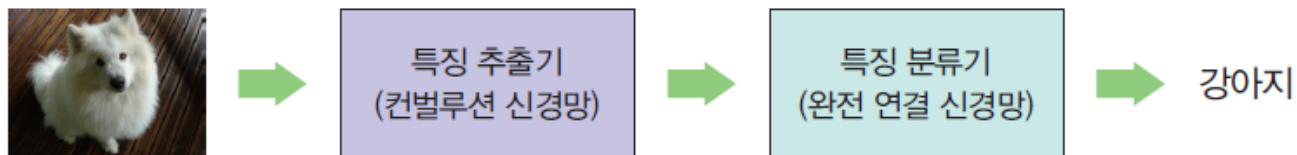


그림 10-10 컨벌루션 신경망을 이용한 영상 인식





컴퓨터 시각 분야 응용의 예



영상 분할



객체 감지

Prediction: Military Uniform



sunflowers



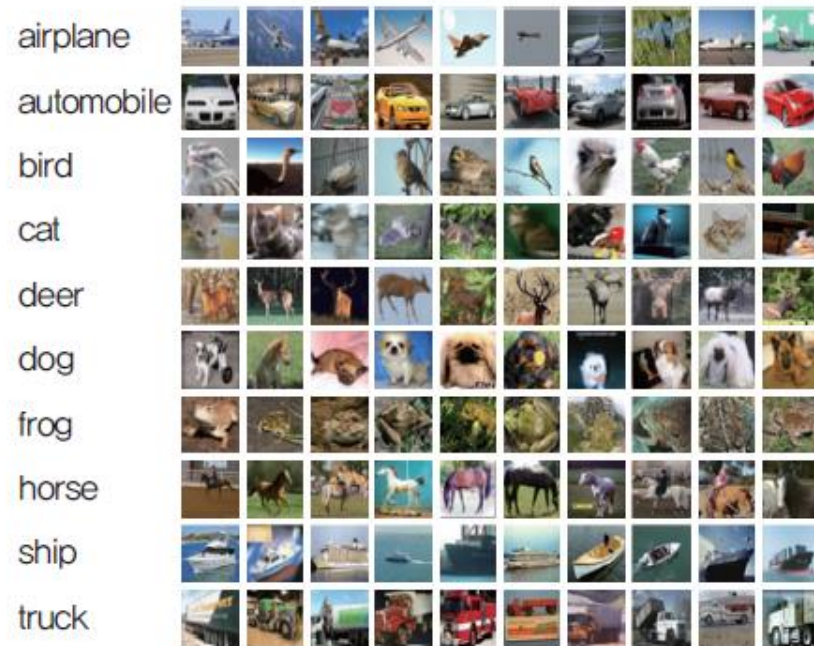
dandelion



영상 분류



- 영상 크기 = 32×32





커버블루션 신경망을 이용한 CIFAR-10 분류 프로그램





컨벌루션 신경망을 이용한 CIFAR-10 분류 프로그램

```
model.add(Conv2D(64, activation = 'relu', kernel_size = (3,3 ), input_shape = (32, 32, 3)))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Conv2D(32, activation = 'relu', kernel_size = (3,3 )))
model.add(Flatten())
model.add(Dense(80, activation = 'relu'))
model.add(Dense(10, activation = 'softmax'))
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 30, 30, 64)	1792
max_pooling2d (MaxPooling2D)	(None, 15, 15, 64)	0
conv2d_1 (Conv2D)	(None, 13, 13, 32)	18464
flatten (Flatten)	(None, 5408)	0
dense (Dense)	(None, 80)	432720
dense_1 (Dense)	(None, 10)	810

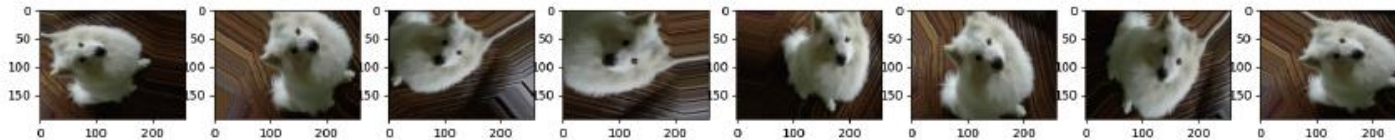
```
=====  
Total params: 453786 (1.73 MB)  
Trainable params: 453786 (1.73 MB)  
Non-trainable params: 0 (0.00 Byte)
```

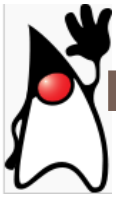
데이터 증대

- 훈련 데이터가 충분하지 않으면 과잉 적합 발생
- 데이터 증대(data augmentation) : 한정된 데이터에서 여러 가지로 변형된 데이터를 만들어내는 기법. 좌우반전, 밝기조절, 좌표이동, 회전, 확대 등
- ImageDataGenerator() 함수 사용

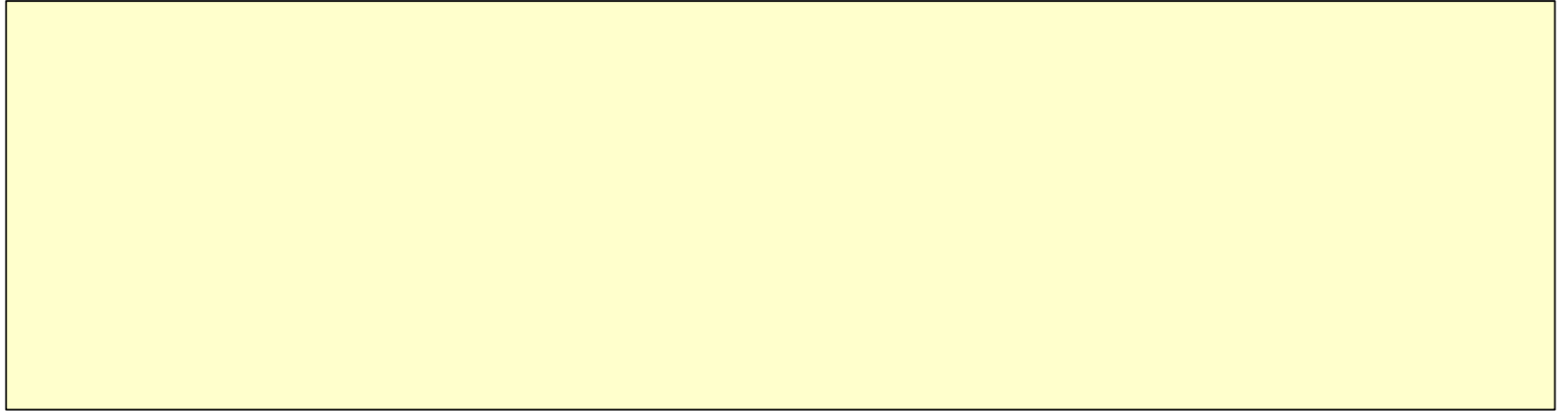


데이터 증대





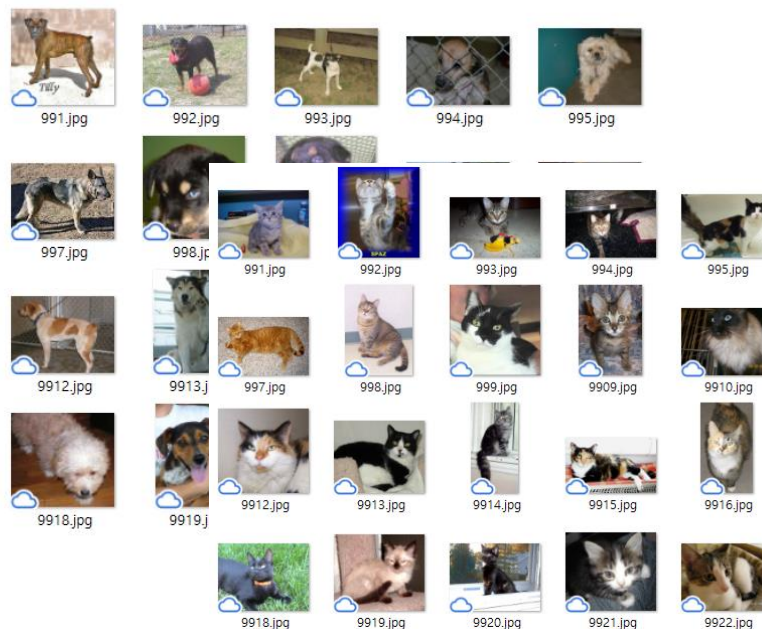
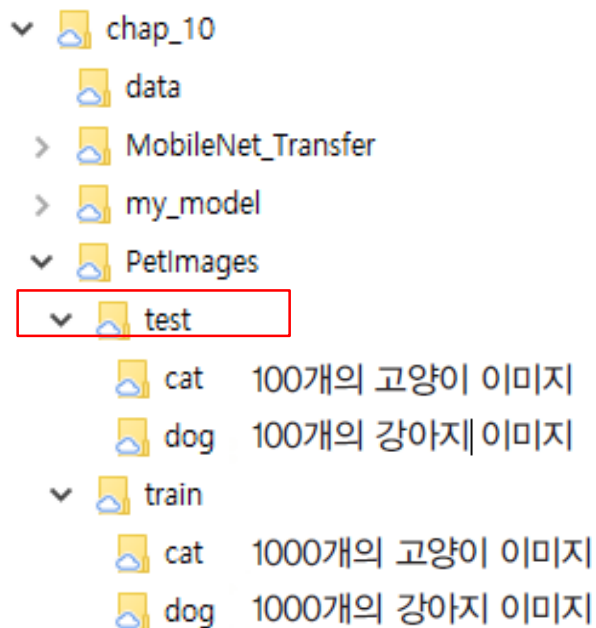
데이터 증대





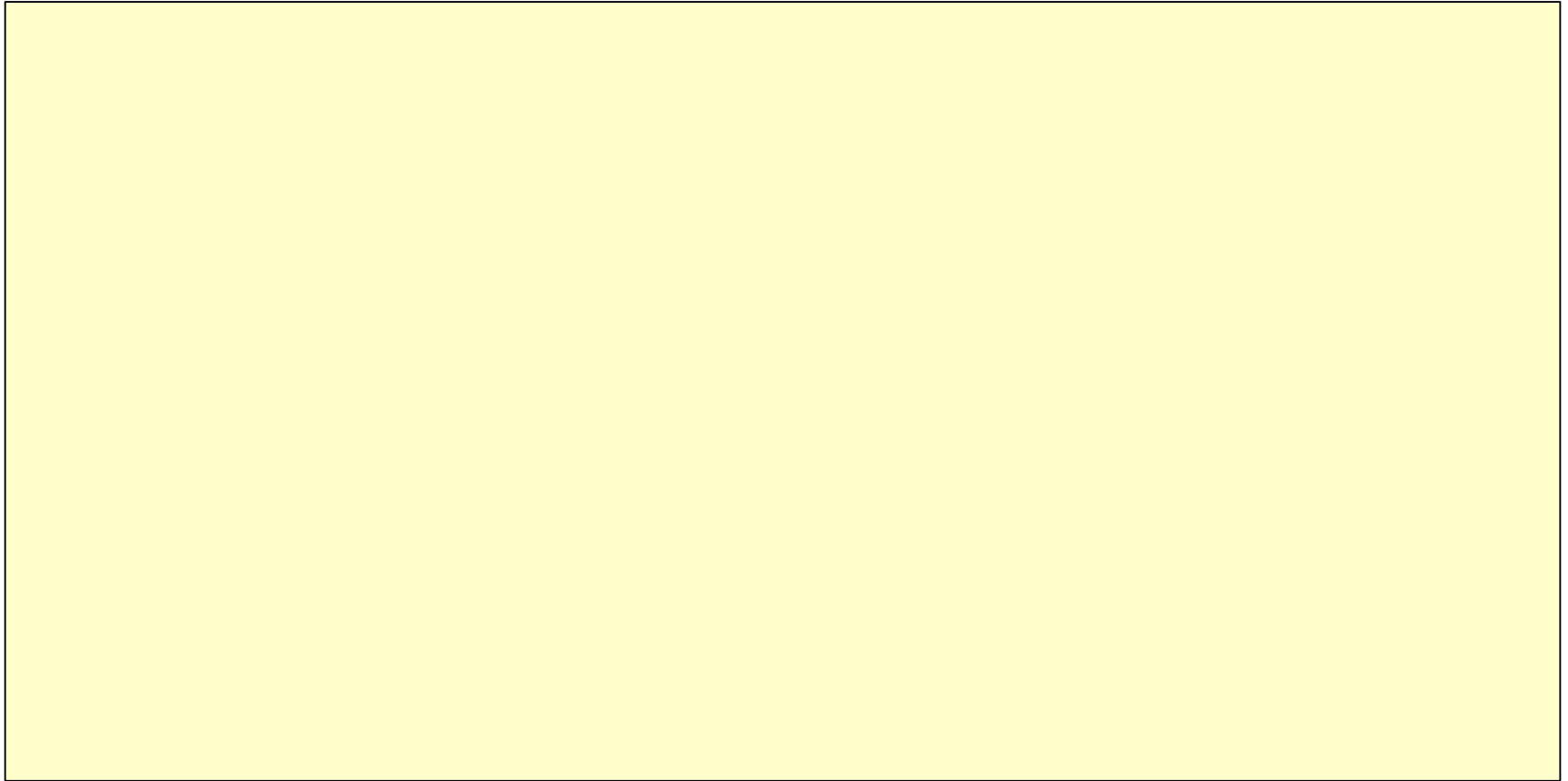
예제: 강아지와 고양이 구별하기

- 데이터 세트 : Kaggle 또는 MS 사이트에서 다운로드
- 25,000장의 이미지들 중에서 약 2200장만 사용(2000장은 훈련 데이터, 200장은 테스트용으로 사용)





예제: 강아지와 고양이 구별하기





예제: 강아지와 고양이 구별하기

```
model.add(layers.Conv2D(32,(3,3), activation='relu', input_shape=(128,128,3)))
model.add(layers.MaxPooling2D(2,2))
model.add(layers.Conv2D(64,(3,3), activation='relu'))
model.add(layers.MaxPooling2D(2,2))
model.add(layers.Flatten())
model.add(layers.Dense(units=512, activation='relu'))
model.add(layers.Dense(units=1, activation='sigmoid'))
```

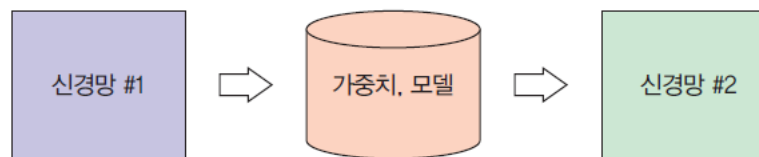
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 126, 126, 32)	896
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	0
conv2d_1 (Conv2D)	(None, 61, 61, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 30, 30, 64)	0
flatten (Flatten)	(None, 57600)	0
dense (Dense)	(None, 512)	29491712
dense_1 (Dense)	(None, 1)	513

```
=====  
Total params: 29511617 (112.58 MB)  
Trainable params: 29511617 (112.58 MB)  
Non-trainable params: 0 (0.00 Byte)
```



가중치 저장과 전이 학습

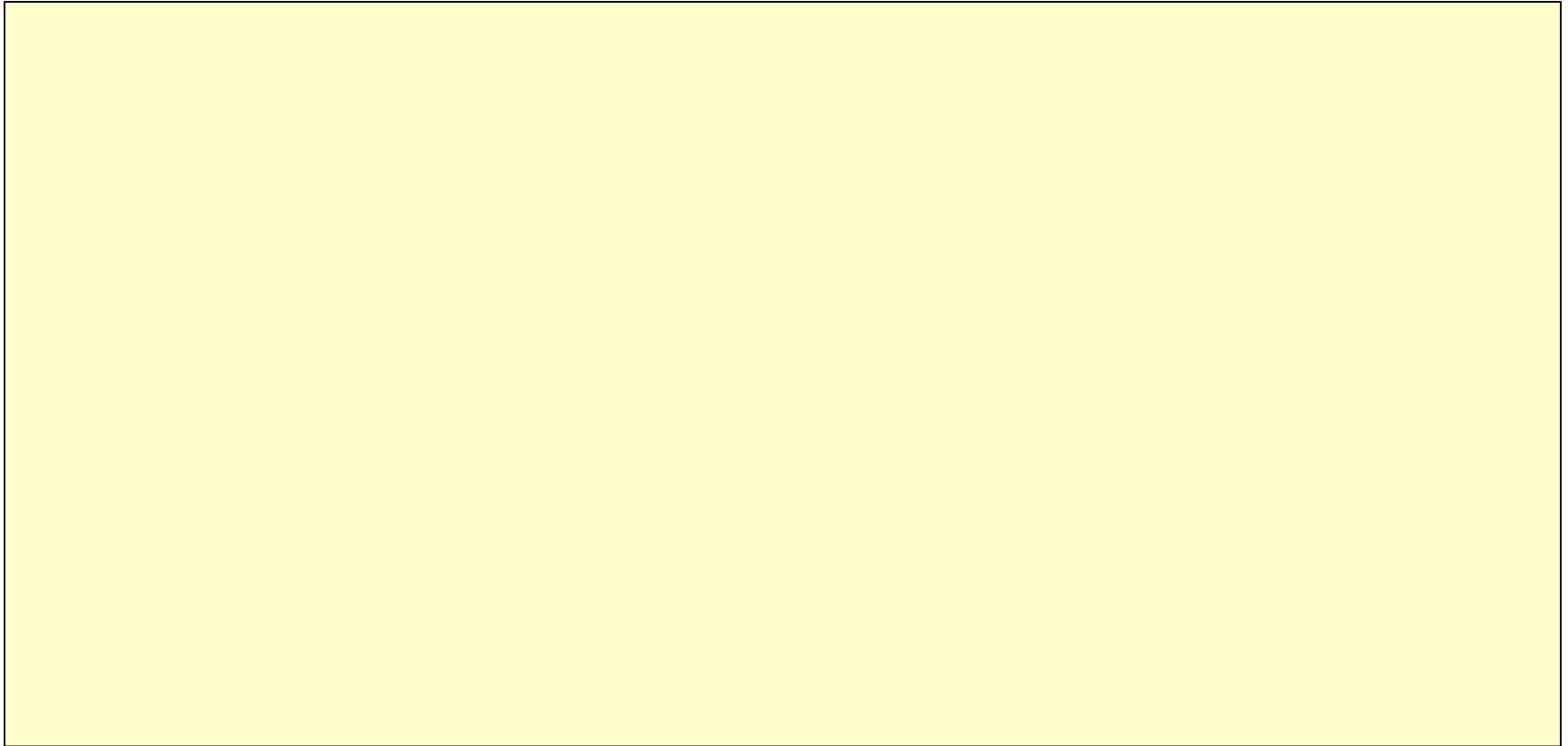
- 이미 학습된 모델의 가중치를 저장할 수 있고, 필요할 때마다 가중치를 불러와서 예측을 할 수 있게 할 수 있다.



- 저장: `model.save('mymodel')`
- 저장되는 정보
 - 신경망 모델의 아키텍처 및 구성
 - 훈련 중에 학습된 모델의 가중치 값
 - 신경망 모델의 컴파일 정보
 - 옵티마이저와 현재 상태
- 복원: `model = load_model('mymodel')`



가정 치 저장



- 전이 학습(transfer learning) : 하나의 문제에 대해 학습한 신경망의 모델과 가중치를 새로운 문제에 적용하는 것

(a) 처음부터 심층 신경망을 훈련



(b) 전이 학습(사전 훈련된 모델의 미세 조정)

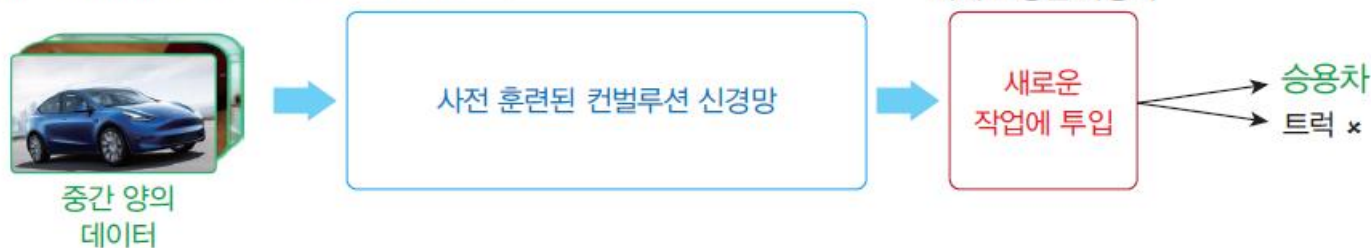


그림 10-12 (a) 처음부터 개발하는 방법 (b) 사전 훈련된 모델을 받아서 미세 조정하는 방법



사전 훈련된 신경망 모델

- 케라스는 사전 훈련된 딥러닝 모델들을 제공한다. 이것을 케라스에서는 “케라스 애플리케이션(keras applications)”이라고 부른다

모델	크기	Top-1 정확도	Top-5 정확도	매개 변수	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	98MB	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
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

사전 훈련된 모델을 내 프로젝트에 맞게 재정의하기

- 전략 1 : 사전 훈련 모델의 구조만 사용하고 학습은 전부 새로 한다.
- 전략 2 : 특징 추출하는 레이어 중에서 일부는 고정시키고 일부는 새로 학습. 분류는 전부 다시 학습.
- 전략 3 : 오직 분류기 레이어만 새로 학습한다.

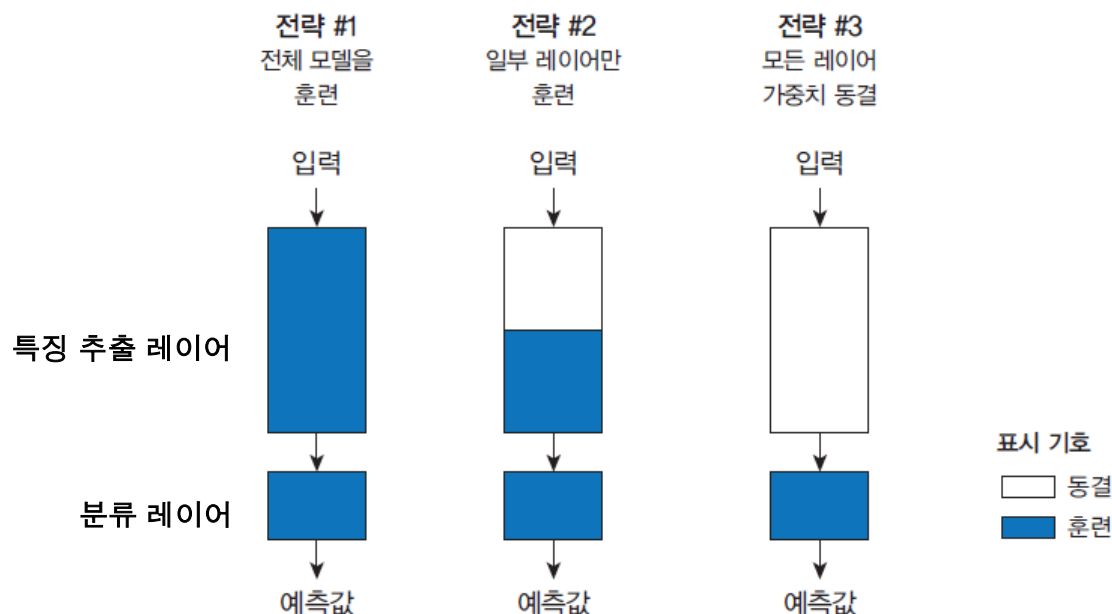
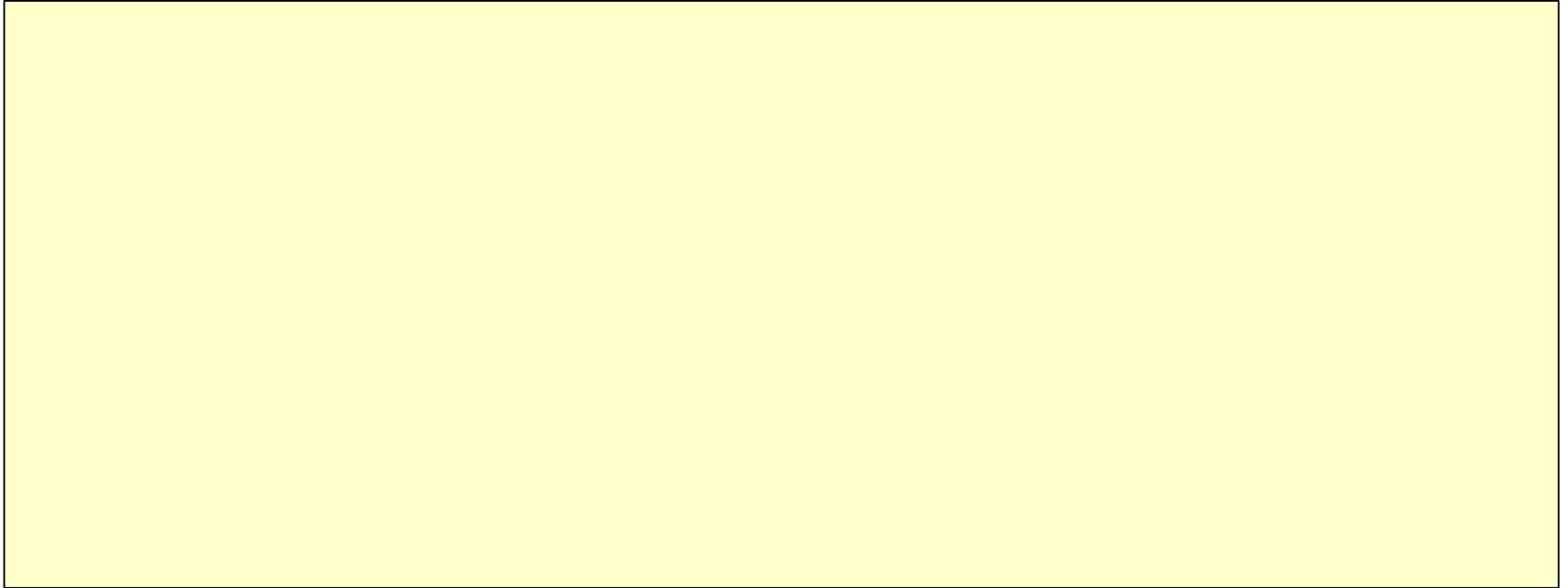


그림 10-13 사전 훈련 모델을 사용하는 3가지 방법



예제 # 1

- ResNet50을 다운로드받아서 변경하지 않고 그대로 사용. 인터넷에서 강아지 사진을 다운받아서 올바르게 인식하는지를 테스트





예제 #2: 사전 훈련된 모델을 특징 추출기 전처리기로 사용

- 케라스가 제공하는 사전 훈련된 모델 중에서 MobileNet을 다운로드받고 여기에 우리가 만든 분류기 레이어를 붙여서 새로운 신경망을 만든다.

