휴스타 ICT 설계 팀프로젝트1
10장 영상인식

1조
[윤상현 이새봄 임채홍]

생능출판, 천인국 저「딥러닝 EXPRESS」

2022.05.03

목차

- 영상 인식
- 전통적인 영상 인식
- 심층 신경망을 이용한 영상 인식
- 데이터 증대
- 가중치 저장과 전이 학습
- 사전 훈련 모델 사용

영상 인식

- 영상 인식이란 영상 안의 물체를 인식하거나 분류하는 것이다.

- 이번 장에서는 컨벌루션 신경망을 이용한 영상 인식 방법을 살펴본다.

그림 10–2 영상 인식 시스템

전통적인 영상 인식

- 강아지와 고양이를 분류하는 문제에서, 우리는 카메라를 통하여 영상을 찍어서 강아지와 고양이의 특징을 파악한 후에 특징을 이용하여 분류한다.

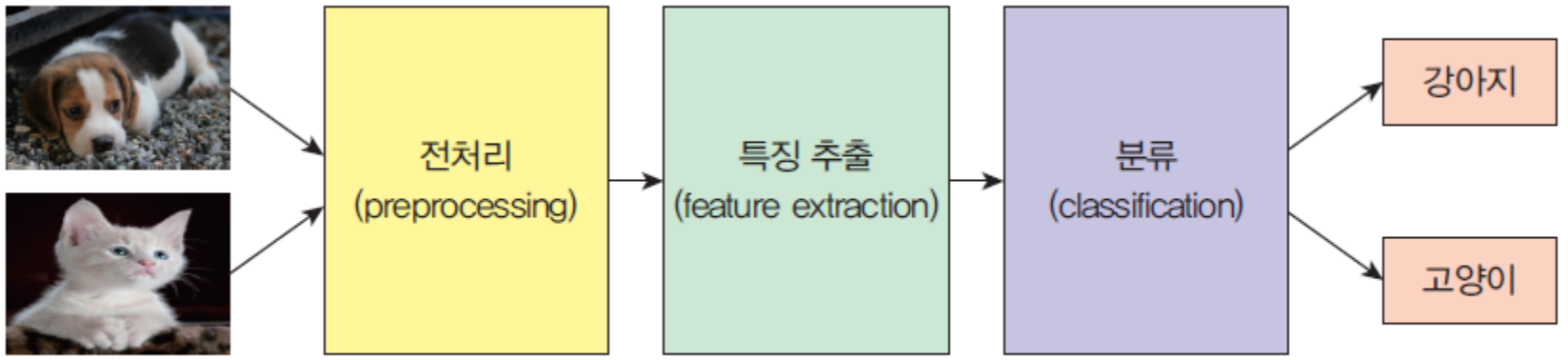


그림 10-3 전통적인 영상 인식 시스템

전통적인 영상 인식

- 특징을 사용한 분류

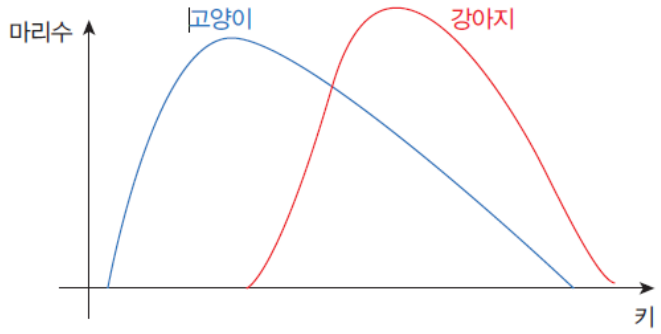


그림 10-4 특징

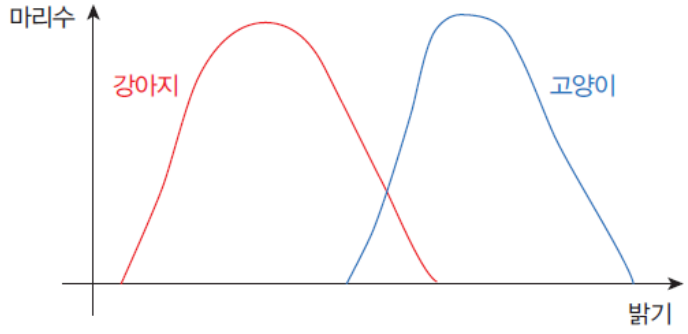


그림 10-5 밝기를 특징으로 사용한다.

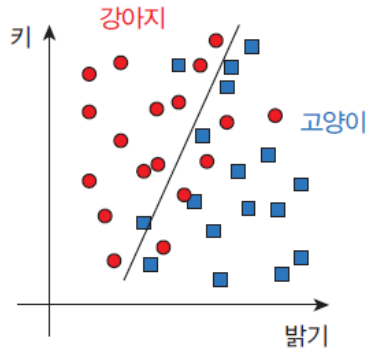


그림 10-6 키와 밝기를 동시에 특징으로 사용한다.

전통적인 영상 인식

- 과잉 반응을 할 수 있기에 성능과 단순성을 적절하게 조합한 판단 경계선이 바람직할 수도 있다.

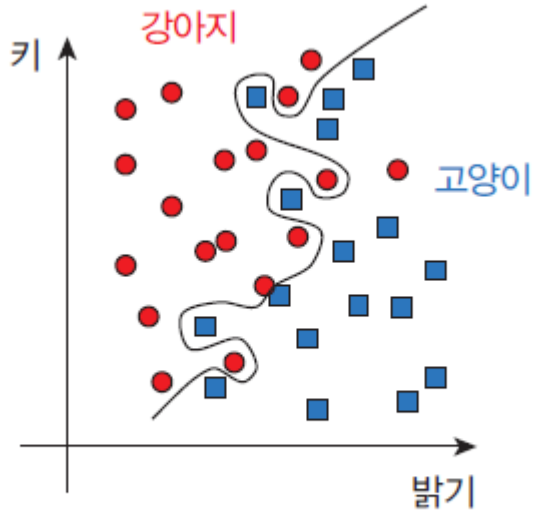


그림 10-7 과잉 적합된 분류기

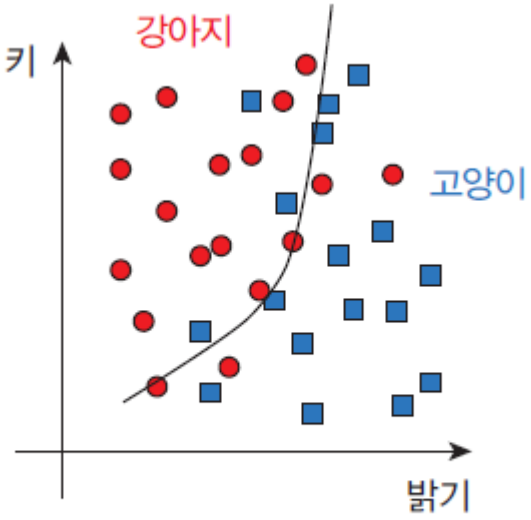


그림 10-8 적절한 판단 경계선

심층 신경망을 이용한 영상 인식

- 신경망을 사용하여 특징 추출과 분류를 동시에 한다.

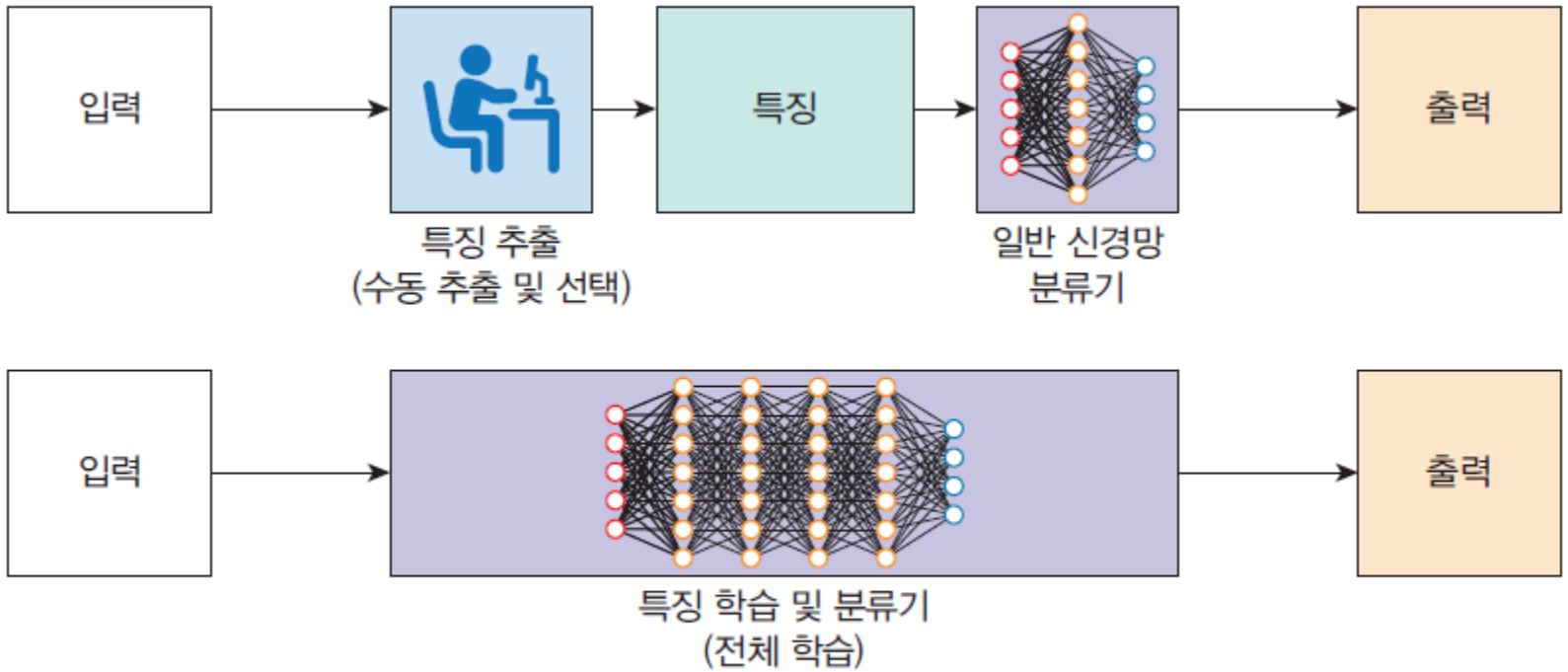


그림 10-9 전통적인 영상 인식과 심층 신경망을 이용한 영상 인식

심층 신경망을 이용한 영상 인식

- 영상을 처리할 때 적합한 신경망 구조가 있는데, 앞장에서 학습했던 컨벌루션 신경망이다.

- 컨벌루션 신경망을 이용한 영상 인식은 다음과 같다.

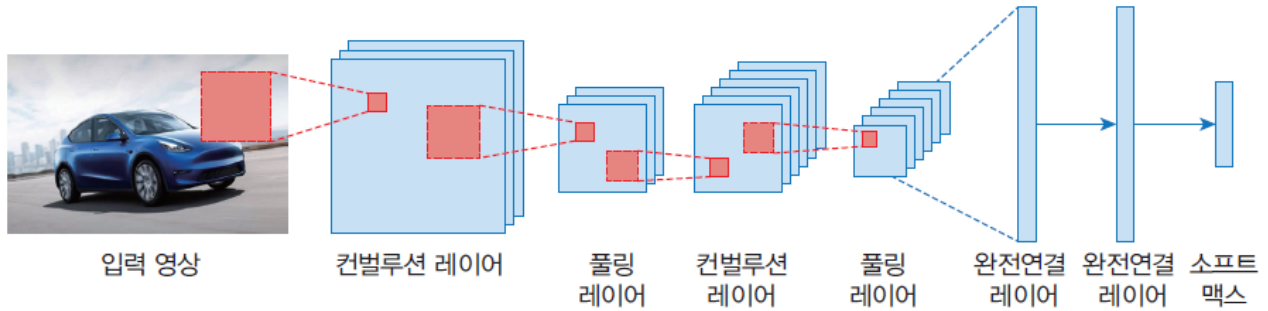
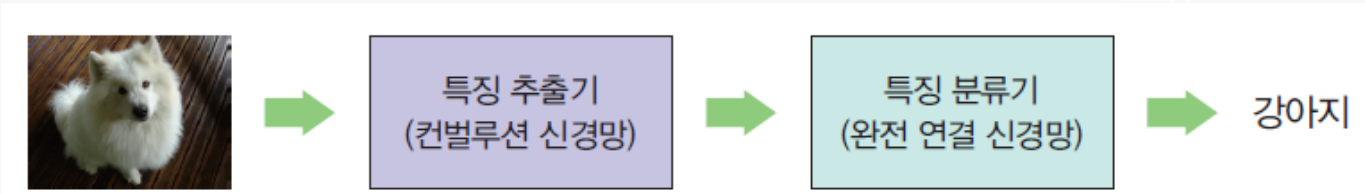


그림 10-10 컨벌루션 신경망을 이용한 영상 인식



CIFAR-10 영상 분류하기

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import *

# CFAR-10 데이터 세트를 적재한다.
# 훈련 세트와 테스트 세트를 반환받는다.
(X_train, y_train), (X_test, y_test) = keras.datasets.cifar10.load_data()

# 두 번째 영상(트럭)을 화면에 표시한다.
plt.figure()
plt.imshow(X_train[1])
plt.colorbar()

# 영상의 픽셀 값을 0에서 1 사이로 변환한다.
X_train = X_train/255.0
X_test = X_test/255.0
```

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

# 모델을 컴파일한다.
model.compile(optimizer = 'adam', loss = 'sparse_categorical_crossentropy',
              metrics = ['accuracy'])
# 모델을 훈련한다.
history = model.fit(X_train, y_train, epochs=10, verbose=1,
                    validation_split=0.3)
```

앞서 배웠던 Conv2D 레이어와 MaxPooling2D 레이어를 번갈아서 추가

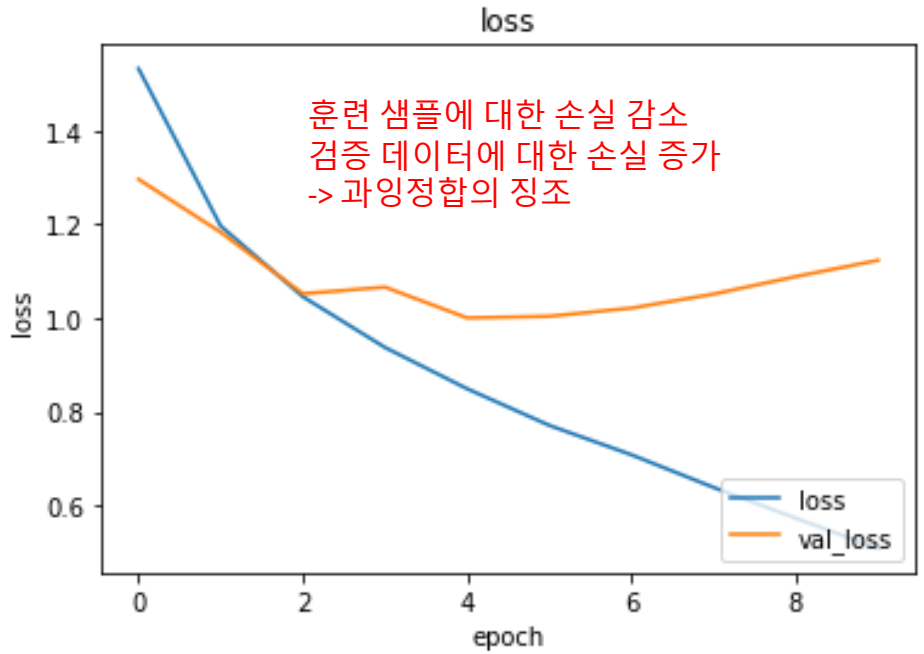
2차원 행렬을 1차원 행렬로

훈련 데이터의 30%를 검증 데이터로 사용

```
Epoch 8/10
1094/1094 [==============================] - 88s 80ms/step - loss: 0.6274 - accuracy: 0.7785 - val_loss: 1.0766 - val_accuracy: 0.654
6
Epoch 9/10
1094/1094 [==============================] - 90s 82ms/step - loss: 0.5564 - accuracy: 0.8030 - val_loss: 1.1267 - val_accuracy: 0.648
6
Epoch 10/10
1094/1094 [==============================] - 83s 76ms/step - loss: 0.4935 - accuracy: 0.8259 - val_loss: 1.1600 - val_accuracy: 0.657
5
```

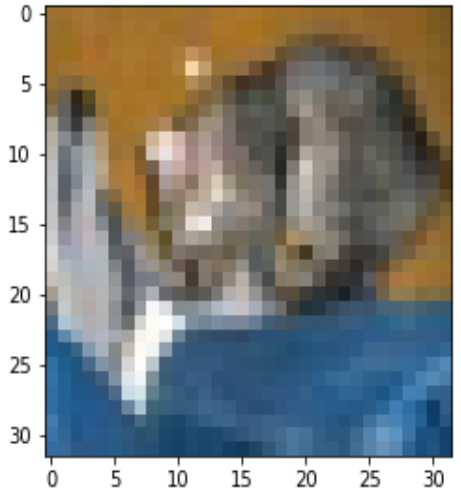
CIFAR-10 영상 분류하기

```
# 손실값을 그래프로 그린다.
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['loss', 'val_loss'], loc = 'lower right')
plt.show()
```




```
plt.figure()
plt.imshow(X_test[0])
y_pred = model.predict(X_test)
print("정답=", y_test[0])
print("예측값=", y_pred[0])
```

```
정답= [3]
예측값= [1.0941911e-03 4.7506925e-05 7.3310710e-03 8.6607480e-01 7.4933792e-05
 8.5330464e-02 2.8897187e-02 4.7681537e-05 1.0639723e-02 4.6242727e-04]
```



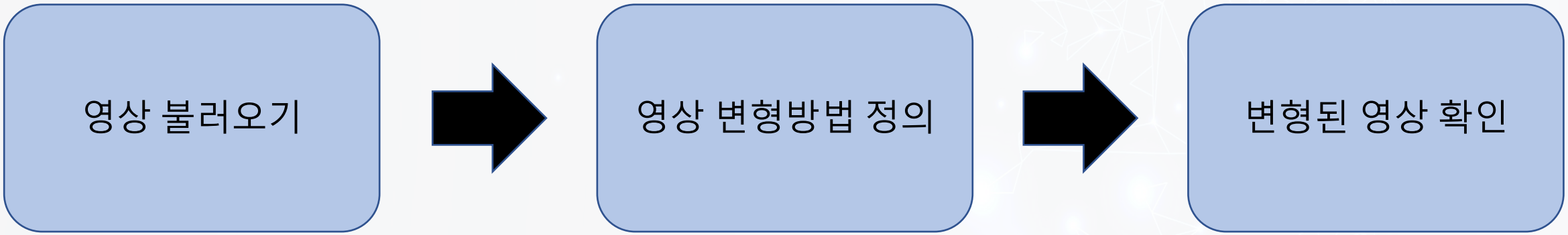
원핫 인코딩으로 출력
인덱스 3이 제일 큰 값이므로 잘 예측함

데이터 증대

- 학습에 사용할 만한 충분한 데이터를 구하기, 만들기가 어려움
- 데이터가 불충분한 경우 과대적합(overfitting)이 생길 가능성이 높음, 이를 막는 기법 중 하나
- 한정된 데이터를 이용해 변형된 데이터를 만드는 기법
- 영상데이터 -> 좌우반전, 밝기조절, 좌표이동, 회전, 확대 등을 통해 데이터 증대

데이터 증대

코드실행과정



데이터 증대

영상 불러오기

```
import tensorflow as tf
import matplotlib.pyplot as plt
from numpy import expand_dims
from tensorflow.keras.preprocessing.image import load_img,
img_to_array

image = load_img("dog.jpg")
array = img_to_array(image)
sample = expand_dims(array, 0)

from tensorflow.keras.preprocessing.image import
ImageDataGenerator
datagen = ImageDataGenerator(rescale = 1./255,
    rotation_range=90, brightness_range=[0.8, 1.0],
    width_shift_range=0.2, zoom_range=[0.8, 1.2],
    height_shift_range=0.2)

obj = datagen.flow(sample, batch_size=1)
fig = plt.figure(figsize=(20,5))

for i in range(8):
    plt.subplot(1,8,i+1)
    image = obj.next()
    plt.imshow(image[0])
```

dog.jpg

expand_dims 함수 활용하여
0번째 축 추가 -> 훈련데이터를 4차원으로 변환
(데이터가 항상 4차원으로 있어야함)

데이터 증대

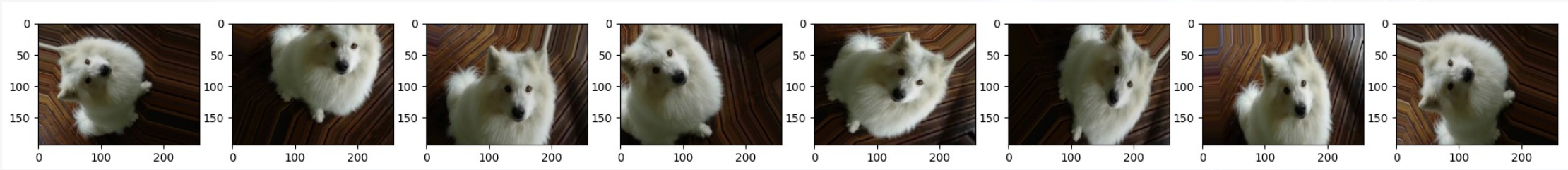
영상 변형방법 정의

```
import tensorflow as tf
import matplotlib.pyplot as plt
from numpy import expand_dims
from tensorflow.keras.preprocessing.image import load_img,
img_to_array

image = load_img("dog.jpg")
array = img_to_array(image)
sample = expand_dims(array, 0)

from tensorflow.keras.preprocessing.image import
ImageDataGenerator
datagen = ImageDataGenerator(rescale = 1./255,
    rotation_range=90, brightness_range=[0.8, 1.0],
    width_shift_range=0.2, zoom_range=[0.8, 1.2],
    height_shift_range=0.2)

obj = datagen.flow(sample, batch_size=1)
fig = plt.figure(figsize=(20,5))

for i in range(8):
    plt.subplot(1,8,i+1)
    image = obj.next()
    plt.imshow(image[0])
```

rotation_range : 회전 한도
brightness_range : 밝기 변형 비율
width_shift_range : 좌우 이동 한도
height_shift_range : 상하 이동 한도
zoom_range : 확대 한도

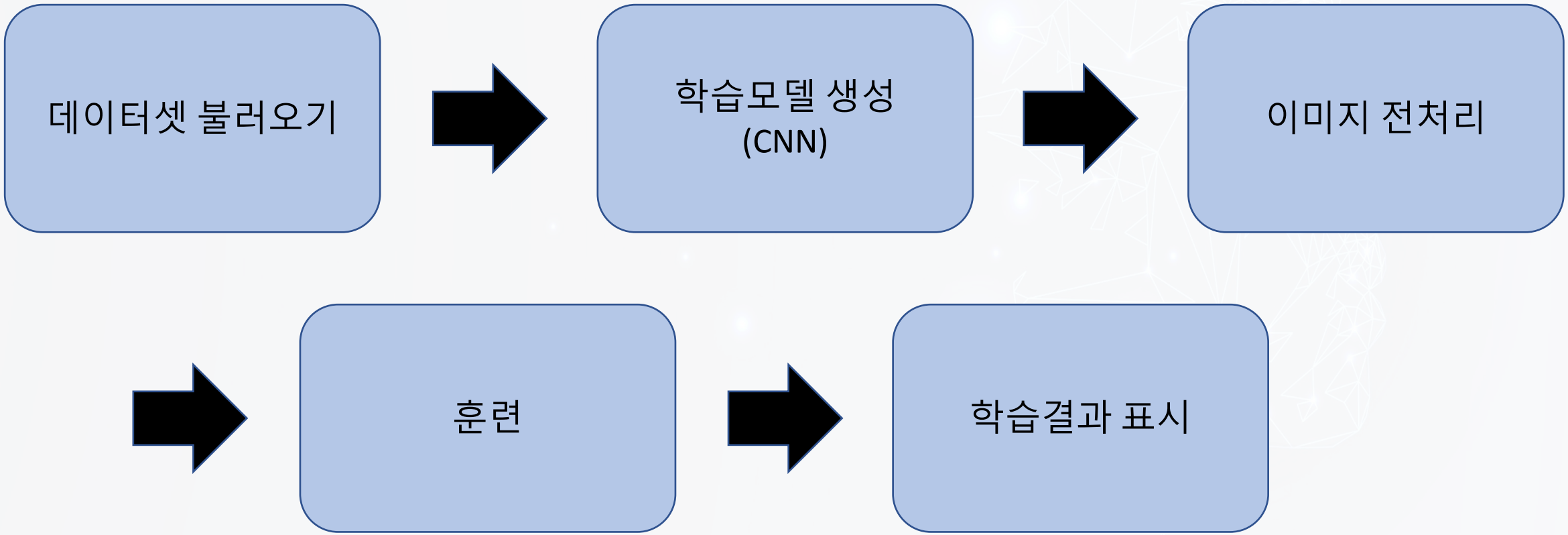
데이터 증대

변형된 영상 확인



강아지와 고양이 구별하기

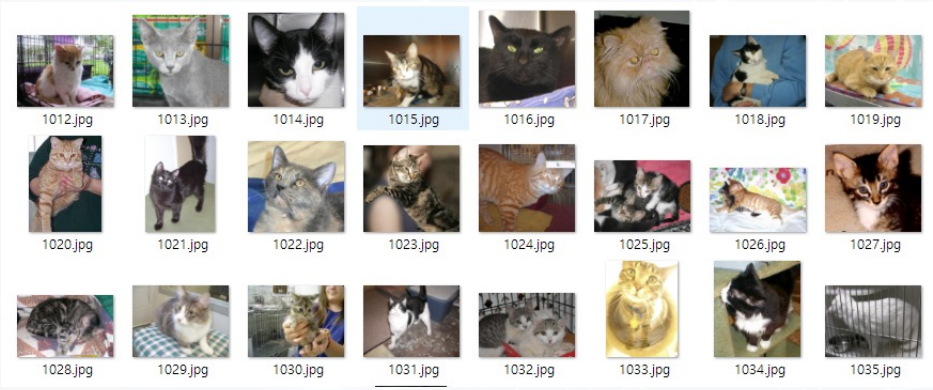
코드실행



강아지와 고양이 구별하기

데이터셋 불러오기

```
from matplotlib import pyplot
from matplotlib.image import imread
image = imread('./Petimages/train/dog/1.jpg')
pyplot.imshow(image)
pyplot.show()

from tensorflow.keras import models, layers

train_dir = './Petimages/train'
test_dir = './Petimages/test'

model = models.Sequential()
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'))

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



고양이 데이터 : 학습 데이터 1000장, 테스트 데이터 100장

강아지 데이터 : 학습 데이터 1000장, 테스트 데이터 100장

강아지와 고양이 구별하기

학습모델 생성 (CNN) 및 이미지 전처리

```
from matplotlib import pyplot
from matplotlib.image import imread
image = imread('./Petimages/train/dog/1.jpg')
pyplot.imshow(image)
pyplot.show()

from tensorflow.keras import models, layers

train_dir = './Petimages/train'
test_dir = './Petimages/test'
```

```
model = models.Sequential()
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'))

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

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

train_datagen = ImageDataGenerator(rescale = 1./255, shear_range = 0.2,
  zoom_range = 0.2, horizontal_flip = True)


test_datagen = ImageDataGenerator(rescale = 1./255)
```

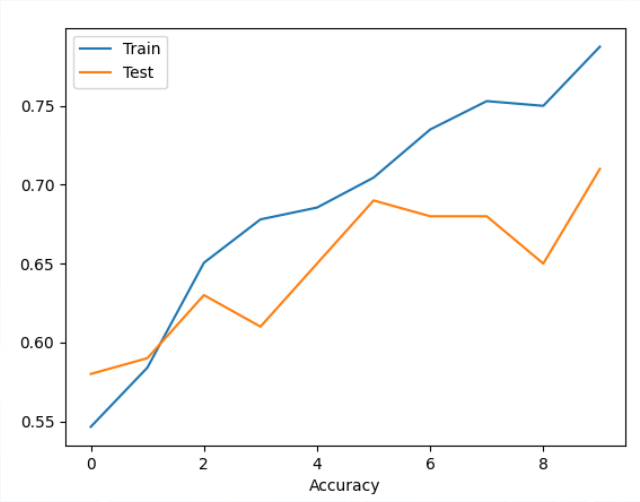
0 ~ 255(RGB값) -> 0 ~ 1 스케일링

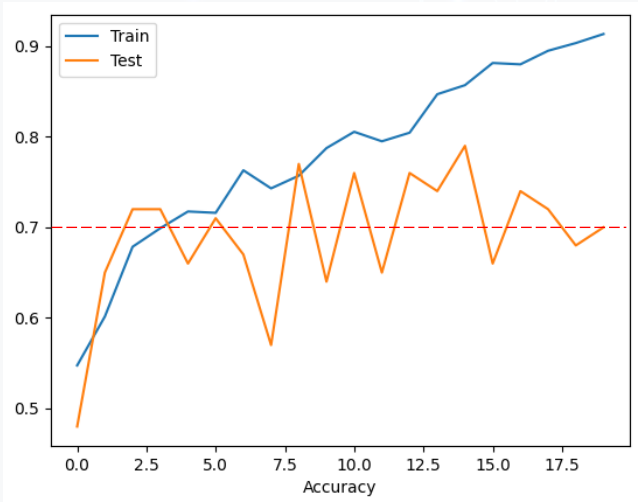
3개의 컨볼루션 레이어와 2개의 완전연결 레이어

강아지와 고양이 구별하기

훈련 및 학습결과

```
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(128, 128),
    batch_size=20,
    class_mode = 'binary')

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(128, 128),
    batch_size=20,
    class_mode = 'binary')

history = model.fit_generator(
    train_generator, steps_per_epoch = 100, epochs=10,
    validation_data=test_generator, validation_steps=5)

import matplotlib.pyplot as plt
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.xlabel('Epoch')
plt.xlabel('Accuracy')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```





강아지와 고양이 구별하기

- Epoch 1/10
- 100/100 [==============================] - 27s 262ms/step - loss: 0.8388 - accuracy: 0.5465 - val_loss: 0.6857 - val_accuracy: 0.5800
- Epoch 2/10
- 100/100 [==============================] - 25s 246ms/step - loss: 0.6740 - accuracy: 0.5840 - val_loss: 0.6711 - val_accuracy: 0.5900
- Epoch 3/10
- 100/100 [==============================] - 25s 246ms/step - loss: 0.6339 - accuracy: 0.6505 - val_loss: 0.6497 - val_accuracy: 0.6300
- Epoch 4/10
- 100/100 [==============================] - 25s 249ms/step - loss: 0.5983 - accuracy: 0.6780 - val_loss: 0.6288 - val_accuracy: 0.6100
- Epoch 5/10
- 100/100 [==============================] - 25s 252ms/step - loss: 0.5875 - accuracy: 0.6855 - val_loss: 0.6679 - val_accuracy: 0.6500
- Epoch 6/10
- 100/100 [==============================] - 25s 250ms/step - loss: 0.5551 - accuracy: 0.7045 - val_loss: 0.6322 - val_accuracy: 0.6900
- Epoch 7/10
- 100/100 [==============================] - 25s 248ms/step - loss: 0.5410 - accuracy: 0.7350 - val_loss: 0.6503 - val_accuracy: 0.6800
- Epoch 8/10
- 100/100 [==============================] - 24s 244ms/step - loss: 0.5048 - accuracy: 0.7530 - val_loss: 0.6000 - val_accuracy: 0.6800
- Epoch 9/10
- 100/100 [==============================] - 25s 246ms/step - loss: 0.5041 - accuracy: 0.7500 - val_loss: 0.7311 - val_accuracy: 0.6500
- Epoch 10/10
- 100/100 [==============================] - 25s 245ms/step - loss: 0.4597 - accuracy: 0.7875 - val_loss: 0.5443 - val_accuracy: 0.7100

가중치 저장과 전이 학습

학습된 가중치의 저장 및 적재

학습에 많은 시간 소요

→

학습된 모델의 가중치 저장

```
model.save('mymodel')
```

→

필요할 때마다 가중치를 불러와서 사용

```
model = load_model('mymodel')
```

- 신경망 모델의 아키텍처 및 구성
- 훈련 중에 학습된 모델의 가중치 값
- 신경망 모델의 컴파일 정보
- 옵티마이저와 현재 상태(훈련을 중단한 곳에서 다시 시작하기 위해)

knu

가중치 저장과 전이 학습

예제

```python
import numpy as np
import tensorflow as tf

test_input = np.random.random((128, 32))
test_target = np.random.random((128, 1))

inputs = tf.keras.Input(shape=(32,))
outputs = tf.keras.layers.Dense(1)(inputs)
model = tf.keras.Model(inputs, outputs)
model.compile(optimizer="adam", loss="mean_squared_error")
model.fit(test_input, test_target, epochs=3)

model.save("my_model")

saved_model = tf.keras.models.load_model("my_model")
saved_model.fit(test_input, test_target, epochs=3)
```

학습

모델을 저장

전이 학습

```
Epoch 1/3
4/4 [==============================] - 0s 2ms/step - loss: 0.4620
Epoch 2/3
4/4 [==============================] - 0s 2ms/step - loss: 0.4063
Epoch 3/3
4/4 [==============================] - 0s 1ms/step - loss: 0.3597
2022-05-03 08:44:54.437291: W tensorflow/python/util/util.cc:368] Sets are not
Epoch 1/3
4/4 [==============================] - 0s 997us/step - loss: 0.3231
Epoch 2/3
4/4 [==============================] - 0s 1ms/step - loss: 0.2944
Epoch 3/3
4/4 [==============================] - 0s 2ms/step - loss: 0.2743
```

학습재개 시작

가중치 저장과 전이 학습

전이학습

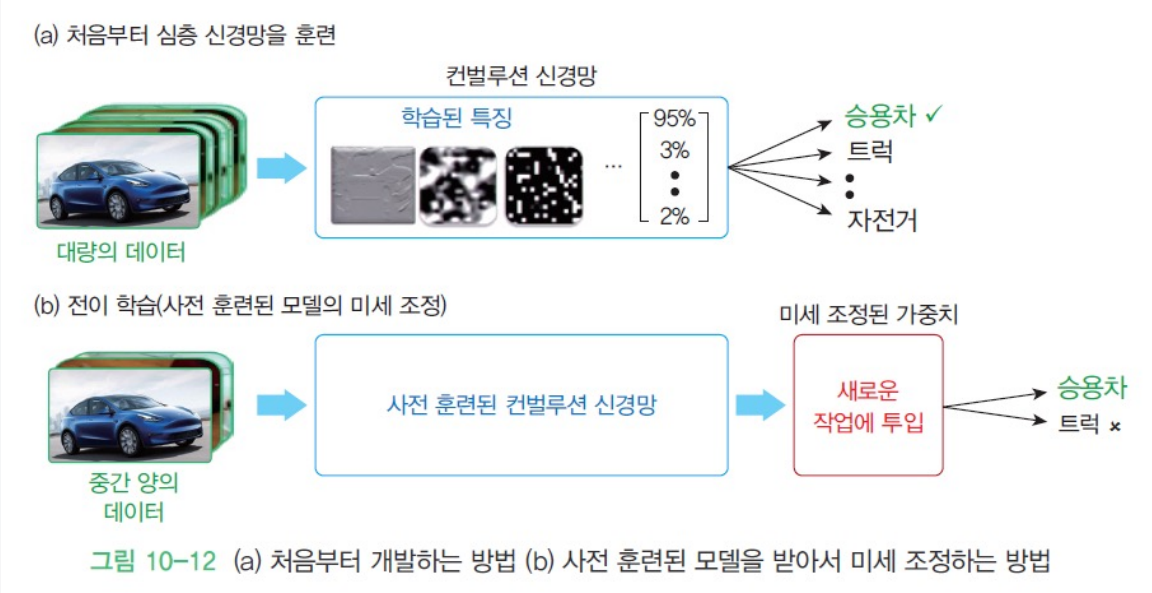


그림 10-12 (a) 처음부터 개발하는 방법 (b) 사전 훈련된 모델을 받아서 미세 조정하는 방법

하나의 문제에 대해
학습한 신경망의 모델과 가중치를,
새로운 문제에 적용하는 것

↓

적은학습 데이터의 수
빠른 학습 속도
높은 정확도

가중치 저장과 전이 학습

사전 훈련된 모델을 내 프로젝트에 맞게 재정의하기

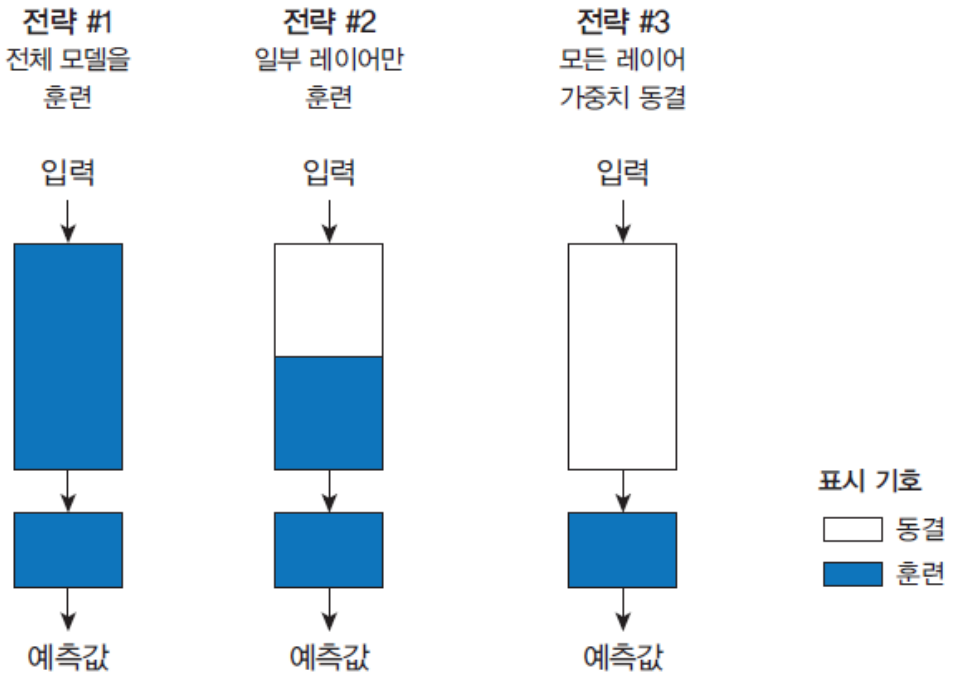


그림 10-13 사전 훈련 모델을 사용하는 3가지 방법

전략#1 -> 학습을 전부 새로, 좋은 자원 필요

전략#2 -> 일부는 고정, 일부는 학습

전략#3 -> 특징을 추출하는 레이어들은 학습시키지 않음
분류기 레이어만 => fine-tuing

knu

가중치 저장과 전이 학습

예제

```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Dense,GlobalAveragePooling2D
from tensorflow.keras.applications import MobileNet
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.mobilenet import preprocess_input
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam

base_model=MobileNet(weights='imagenet',include_top=False) #imports the

x=base_model.output
x=GlobalAveragePooling2D()(x)
x=Dense(1024,activation='relu')(x)
x=Dense(1024,activation='relu')(x)
x=Dense(512,activation='relu')(x)
preds=Dense(2,activation='softmax')(x)

model=Model(inputs=base_model.input,outputs=preds)
```

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

케라스 제공 사전 훈련 모델

사전 훈련된 모델에 분류기 레이어 추가

가중치 저장과 전이 학습

예제

```
for layer in model.layers[:20]:
    layer.trainable=False
for layer in model.layers[20:]:
    layer.trainable=True


train_datagen=ImageDataGenerator(preprocessing_function=preprocess_input)

train_generator=train_datagen.flow_from_directory('./Petimages/',
                                                  target_size=(128,128),
                                                  color_mode='rgb',
                                                  batch_size=32,
                                                  class_mode='categorical',
                                                  shuffle=True)


model.compile(optimizer='Adam',loss='categorical_crossentropy',metrics=['accuracy'])

step_size_train=train_generator.n//train_generator.batch_size
model.fit_generator(generator=train_generator,
                    steps_per_epoch=step_size_train,
                    epochs=5)
```

20번째 레이어부터 마지막 레이어까지는
변경이 가능하도록 설정

```
Epoch 1/5
68/68 [==============================] - 18s 254ms/step - loss: 0.4081 - accuracy: 0.8939
Epoch 2/5
68/68 [==============================] - 17s 251ms/step - loss: 0.2761 - accuracy: 0.9087
Epoch 3/5
68/68 [==============================] - 17s 256ms/step - loss: 0.2121 - accuracy: 0.9142
Epoch 4/5
68/68 [==============================] - 17s 249ms/step - loss: 0.1641 - accuracy: 0.9400
Epoch 5/5
68/68 [==============================] - 17s 248ms/step - loss: 0.1358 - accuracy: 0.9553
```

요약

- 영상 인식(image recognition)란 영상 안의 물체를 인식하거나 분류하는 것이다.

- 영상 인식에 많이 사용되는 신경망은 컨볼루션 인공신경망(CNN)이다. 컨볼루션 신경망(CNN)은 동물의 조직에서 영감을 얻어서 만들어진 신경망이다. CNN에서 뉴런의 수용 공간은 다른 뉴론들과 약간 겹치게 된다. 컨볼루션 신경망은 영상 및 비디오 인식, 추천 시스템 및 자연 언어 처리 분야에서 폭넓게 응용되고 있다.

- 풀링(Pooling)이란 서브 샘플링이라고도 하는 것으로 입력 데이터의 크기를 줄이는 것이다.

- 데이터 증대(data augmentation)는 한정된 데이터에서 여러 가지로 변형된 데이터를 만들어내는 기법이다.
-
- 케라스에서는 save(), load_model() 메소드를 이용하여 가중치를 저장하거나 불러올 수 있다.

- **전이 학습(transfer learning)**은 하나의 문제에 대해 학습한 신경망의 모델과 가중치를, 새로운 문제에 적용하는 것이다.

참고자료

- 딥 러닝 강의자료 코드