딥러닝/클라우드

Chapter 13

Keras CNN

(augmentation & workshop)

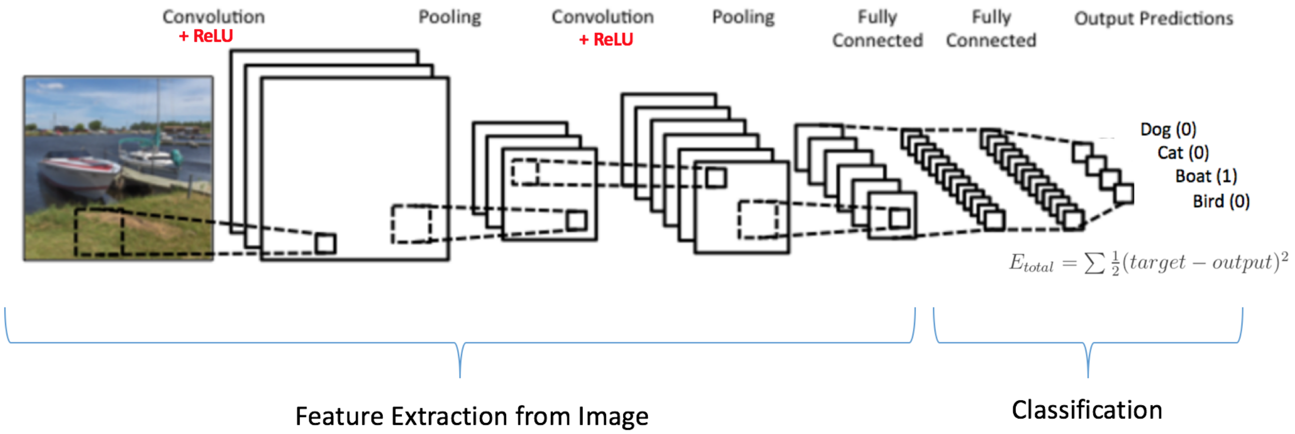
오 세 종

DANKOOK UNIVERSITY

Contents

1. Image augmentation
2. CNN workshop

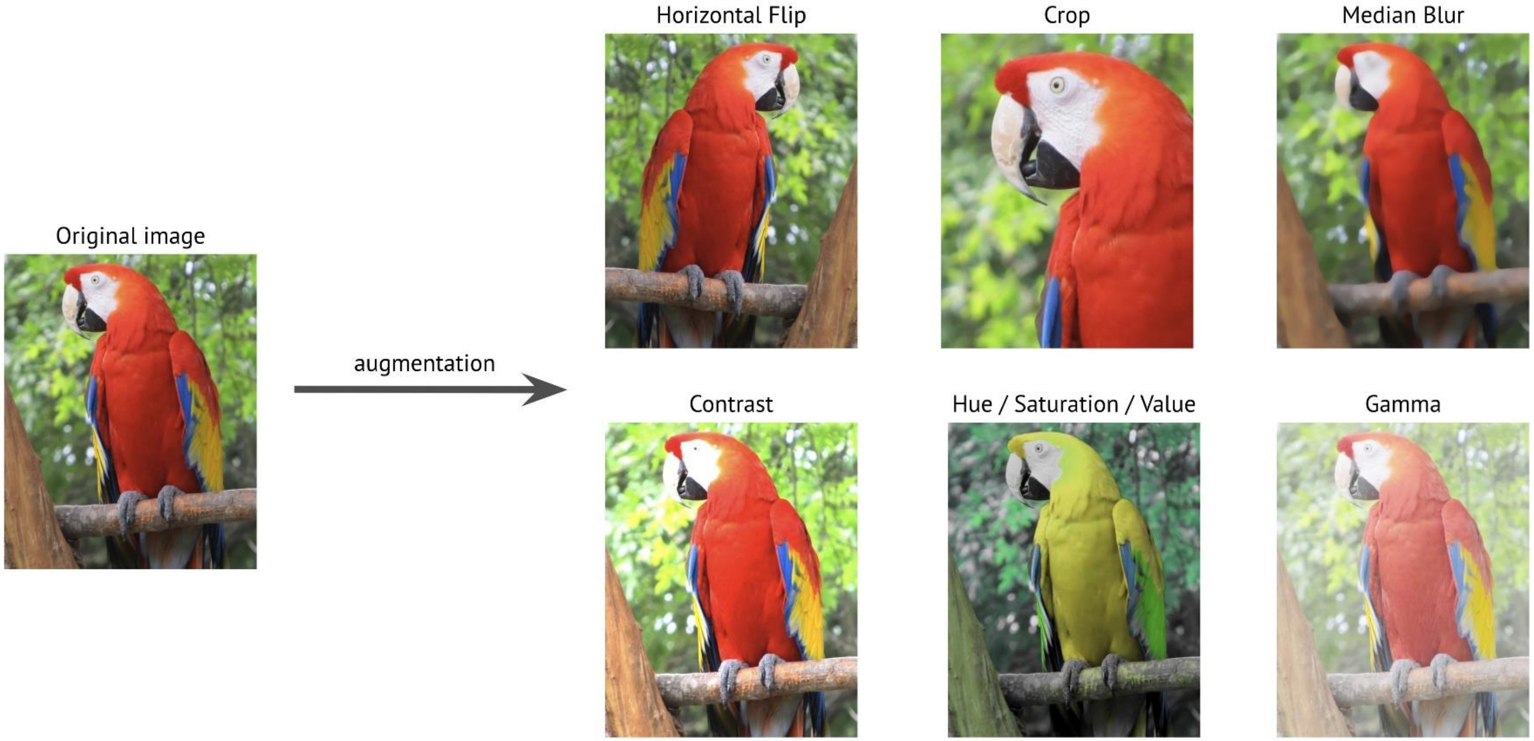
CNN : Convolutional Neural Network



https://www.kdnuggets.com/2016/11/intuitive-explanation-convolutional-neural-networks.html/3

1. Image augmentation

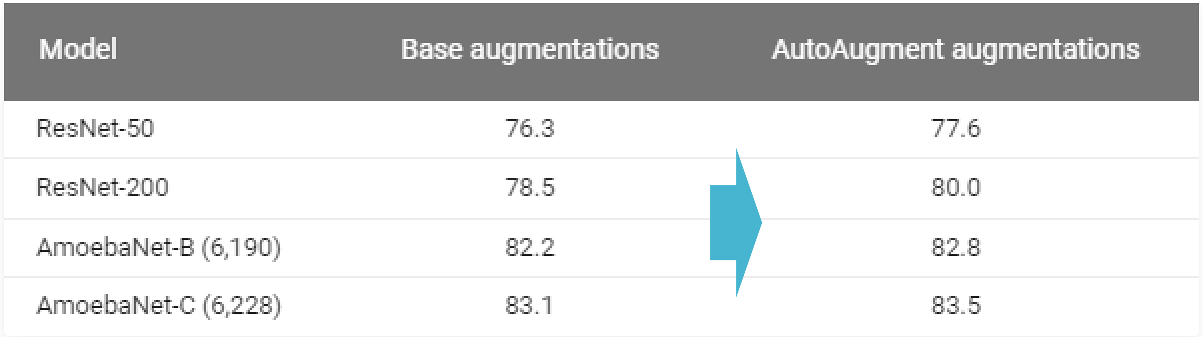
- Image classification 은 많은 양의 학습 데이터를 필요로 함
- 확보한 데이터가 부족할 경우 augmentation 을 통해 데이터를 늘릴 수 있음
- 기존 데이터를 가공, 변형해서 새로운 데이터를 만드는 방식



https://albumentations.ai/docs/introduction/image_augmentation/

1. Image augmentation

- Augmentation을 통해 예측 정확도를 향상 시킬 수 있다.



Model	Base augmentations	AutoAugment augmentations
ResNet-50	76.3	77.6
ResNet-200	78.5	80.0
AmoebaNet-B (6,190)	82.2	82.8
AmoebaNet-C (6,228)	83.1	83.5

https://albumentations.ai/docs/introduction/image_augmentation/

- Keras 에서도 image augmentation 기능을 제공

1. Image augmentation

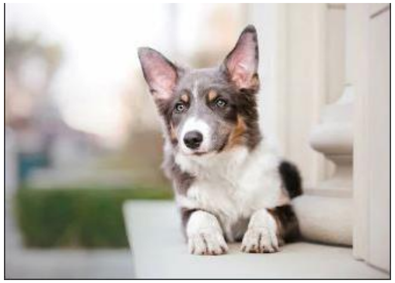
- An example

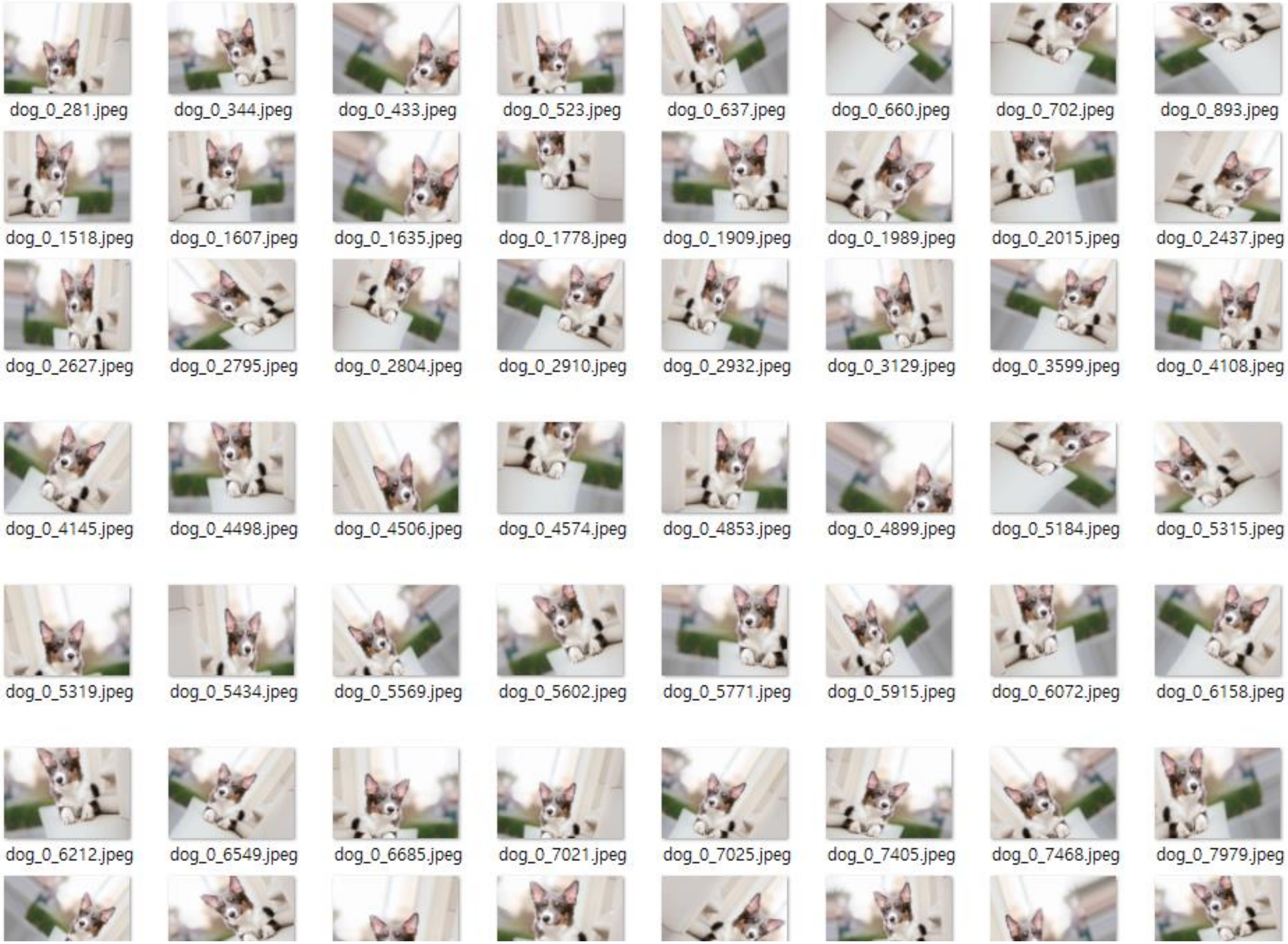
13_augmentation_example.py

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator, array_to_img,
     img_to_array, load_img

datagen = ImageDataGenerator(
        rotation_range=40,                      # 0~180
        width_shift_range=0.2,
        height_shift_range=0.2,
        rescale=1./255,                         # 픽셀값을 0~1 로 변환
        shear_range=0.2,                        # shearing transformations
        zoom_range=0.2,                         # randomly zooming
        horizontal_flip=True,                   # randomly flipping
        fill_mode='nearest')                    # filling in newly created pixels

img = load_img('d:/data/dog.jpg')
x = img_to_array(img)                           # shape (3, 331, 237)
x = x.reshape((1,) + x.shape)                   # shape (1, 3, 331, 237)

# the .flow() command below generates batches of randomly transformed images
# and saves the results to the `d:/data/aug/` directory
i = 0
for batch in datagen.flow(x, batch_size=1,
                          save_to_dir='d:/data/aug/', save_prefix='dog',
save_format='jpeg'):
    i += 1
    if i > 30:
        break                                   # or for working infinitely
```

1. Image augmentation

- An example





2. CNN with augmentation

13_CNN_MNIST_augmentation_example.py

```
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.utils  import to_categorical
import numpy as np
import matplotlib.pyplot as plt

from tensorflow.keras.preprocessing.image import ImageDataGenerator

# define image size
img_rows=28
img_cols=28
```

2. CNN with augmentation

```
# load dataset
(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train, X_test = X_train / 255.0, X_test / 255.0

# reshape
X_train = X_train.reshape(X_train.shape[0], img_rows, img_cols, 1)
X_test = X_test.reshape(X_test.shape[0], img_rows, img_cols, 1)


# one hot encoded
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

# data augmentation
datagen = ImageDataGenerator(
 zoom_range=0.2,
    shear_range=0.2,
    rotation_range=10,
    fill_mode='nearest',
    validation_split = 0.2
    )

datagen.fit(X_train)
```

2. CNN with augmentation

```
train_gen = datagen.flow(X_train, y_train, batch_size=60)

# fix random seed for reproducibility
seed = 100
np.random.seed(seed)
num_classes = 10
```

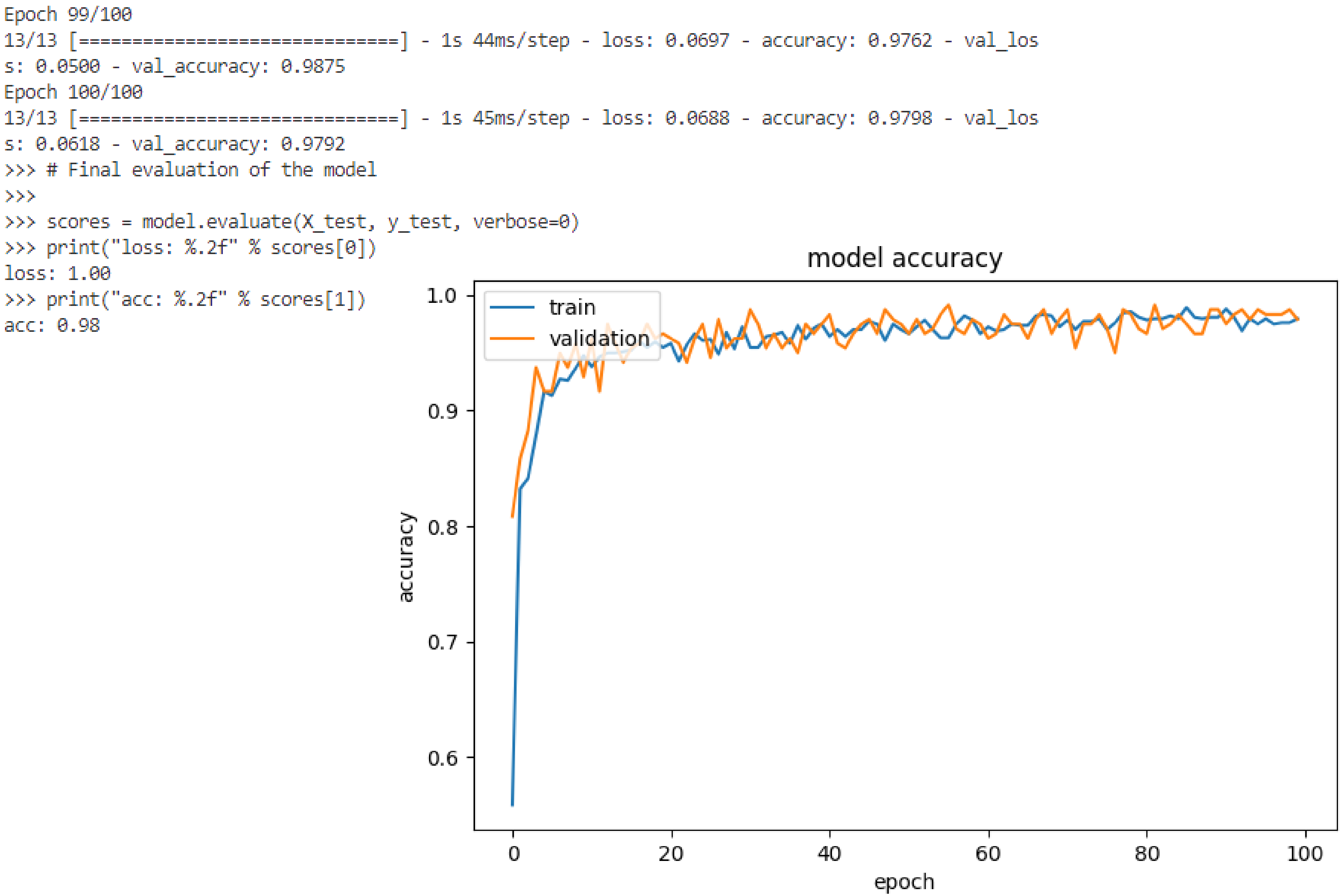
2. CNN with augmentation

```
# create CNN model
def cnn_model():
    # define model
    model = Sequential()
    model.add(Conv2D(64, kernel_size=(5, 5),
                        padding='valid',
                        strides=(1, 1),
                        input_shape=(img_rows, img_cols, 1),
                        activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.5))
    model.add(Flatten())
    model.add(Dense(127, activation='relu'))
    model.add(Dense(num_classes, activation='softmax'))

    # Compile model
    model.compile(loss='categorical_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])
    return model
```

2. CNN with augmentation

```
model = cnn_model()

# Fit the model
disp = model.fit(train_gen,
          validation_data=(X_test, y_test),
          epochs=10,   # 100
          batch_size=200,
          verbose=1)

# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("loss: %.2f" % scores[0])
print("acc: %.2f" % scores[1])

# summarize history for accuracy
plt.plot(disp.history['accuracy'])
plt.plot(disp.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```

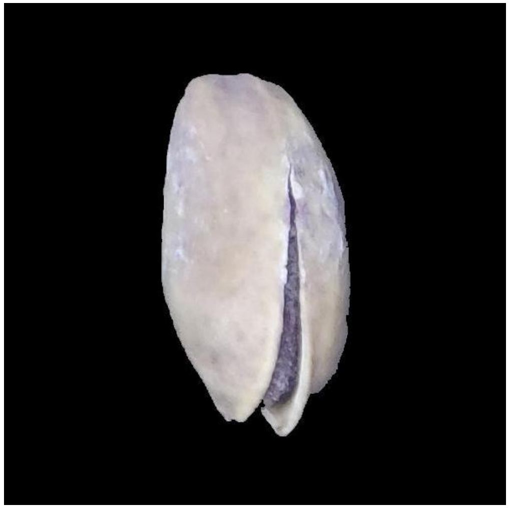
2. CNN with augmentation



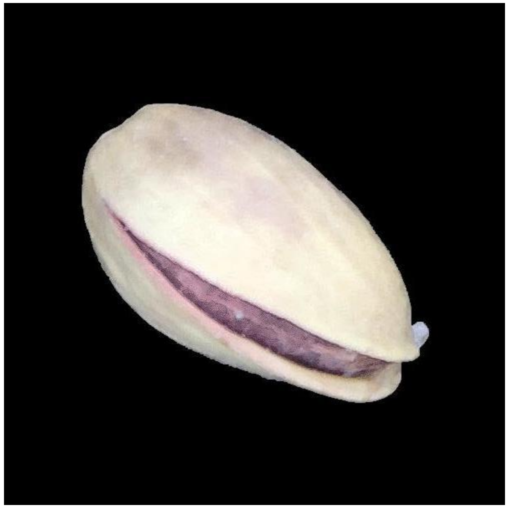
```
Epoch 99/100
13/13 [==============================] - 1s 44ms/step - loss: 0.0697 - accuracy: 0.9762 - val_los
s: 0.0500 - val_accuracy: 0.9875
Epoch 100/100
13/13 [==============================] - 1s 45ms/step - loss: 0.0688 - accuracy: 0.9798 - val_los
s: 0.0618 - val_accuracy: 0.9792
>>> # Final evaluation of the model
>>>
>>> scores = model.evaluate(X_test, y_test, verbose=0)
>>> print("loss: %.2f" % scores[0])
loss: 1.00
>>> print("acc: %.2f" % scores[1])
acc: 0.98
```

3. CNN Workshop

- 과제: 두종류의 피스타치오를 식별할 수 있는 CNN 모델의 개발



kirmizi



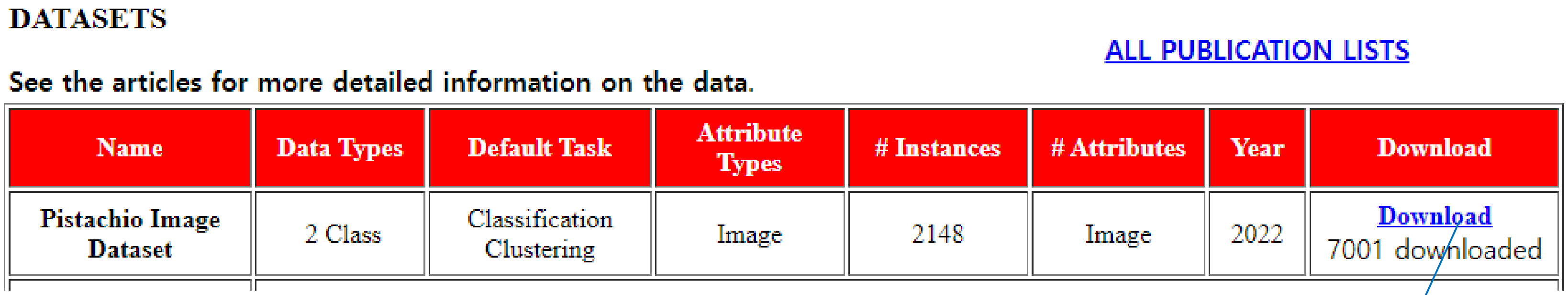
Siirt

Note. 이미지 사이즈가 600x600 이어서 학습에 시간이 많이 걸릴 수 있으므로 모든 이미지를 120x120 으로 축소하여 사용한다.

** chap13 slide 11에 있는 transform.resize() 함수 이용

3. CNN Workshop

- Dataset: Pistachio Image
  - Download URL : https://www.muratkoklu.com/datasets/

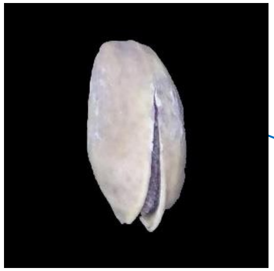


DATASETS

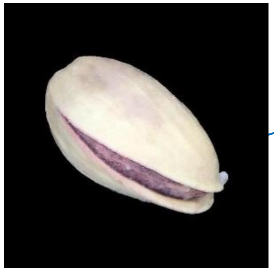
ALL PUBLICATION LISTS

See the articles for more detailed information on the data.

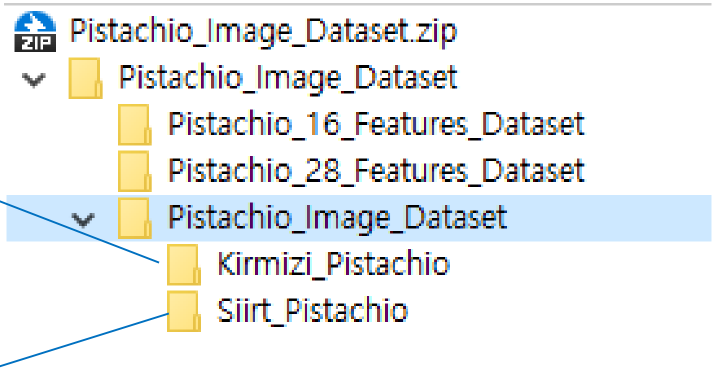
Name	Data Types	Default Task	Attribute Types	# Instances	# Attributes	Year	Download
Pistachio Image Dataset	2 Class	Classification Clustering	Image	2148	Image	2022	Download 7001 downloaded



kirmizi



Siirt



- Pistachio_Image_Dataset.zip
  - Pistachio_Image_Dataset
    - Pistachio_16_Features_Dataset
    - Pistachio_28_Features_Dataset
    - Pistachio_Image_Dataset
      - Kirmizi_Pistachio
      - Siirt_Pistachio

3. CNN Workshop

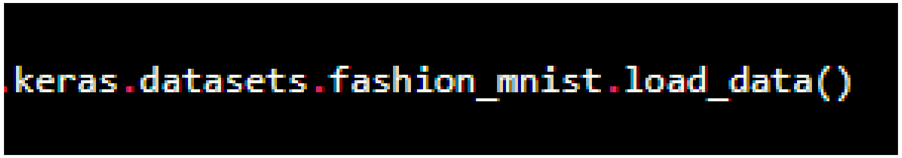
- Hidden layer의 수는 3~5개 정도로 한다.
- 각 레이어의 노드 수는 각자 정한다.
- Data는 train, test 으로 7:3 의 비율로 나눈다. (random state: 123)
- Epoch, batch size 등도 각자 정한다.
- 분류 모델을 학습하고 test set으로 성능을 평가한다 (accuracy)
- 성능 평가 결과 및 학습 곡선 그래프를 보인다

- 이번에는 앞의 과정을 반복하되 data augmentation 을 적용한 모델을 개발하고, 모델의 성능과 학습 곡선 그래프를 보인다.

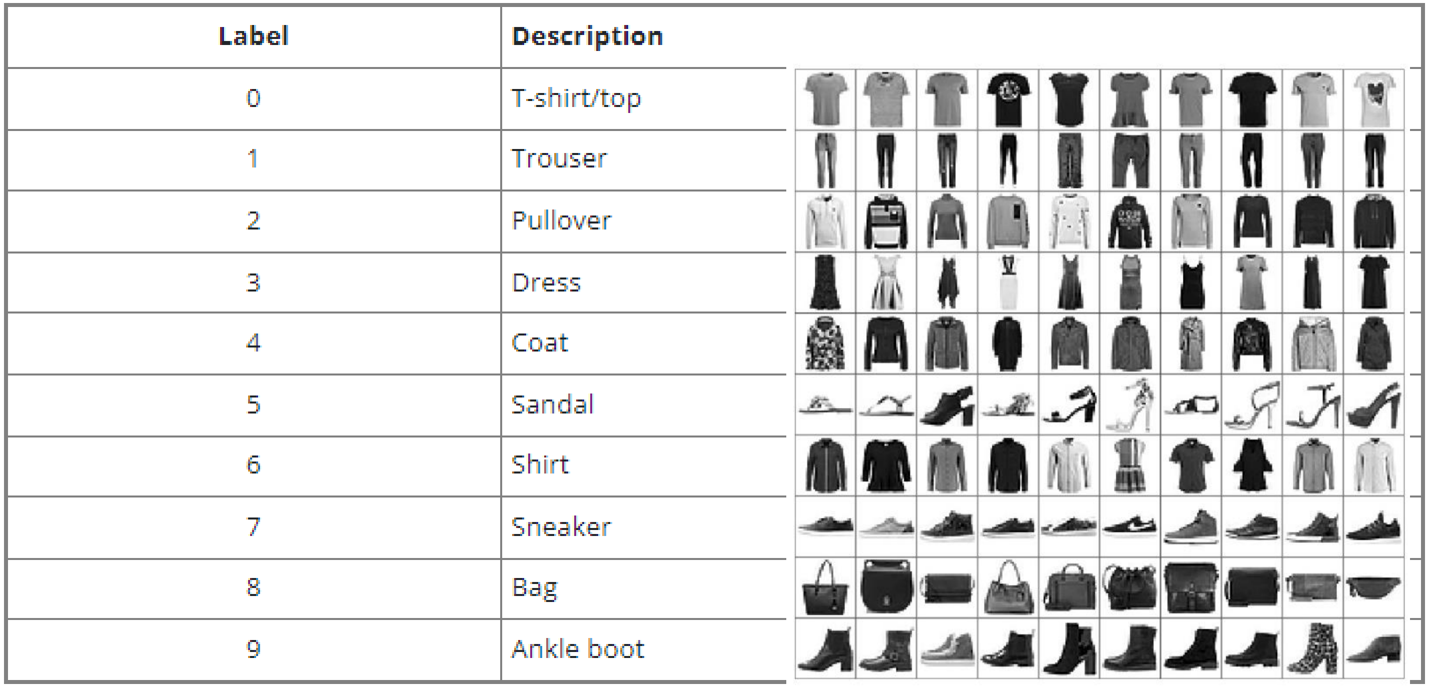
- 수업 자료에 제시된 CNN및 augmentation 예제 코드를 수정하는 식으로 하면 어렵지 않게 할 수 있음

3. CNN Workshop

- 과제: 패션 이미지로 부터 옷 또는 악세서리의 종류를 식별하는 CNN 모델의 개발



```
.keras.datasets.fashion_mnist.load_data()
```



Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot