머신러닝 기반 이미지 분류

붓꽃 분류하기

머신러닝 기반 이미지 분류

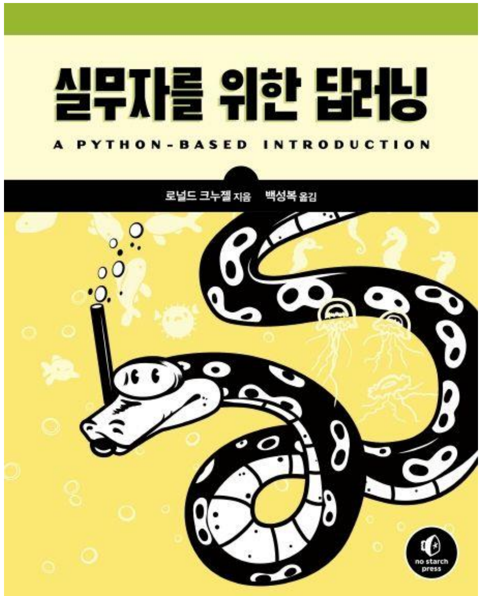
[1] 머신러닝을 이용한 이미지 분류 실습

- 아이리스 꽃 학습 데이터 세트 준비
- 데이터 전처리 및 피쳐 분석
- 머신러닝 모델 적용 및 실행
- 머신러닝 모델 별 성능 분석

[2] 딥러닝을 이용한 이미지 분류 실습

- MNIST 학습 데이터 세트 준비
- MNIST 데이터 피쳐 분석
- 딥러닝 모델(CNN) 적용 및 실행
- CNN 모델 구조 별 성능 분석

참고 자료



[1] 참고 교재: "실무자를 위한 딥러닝"

- 5장: 데이터 세트 만들기
- 7장: 고전 모델 실습
- 13장: 케라스와 MNIST를 활용한 CNN 분석

<Original Text>

- Practical Deep Learning: A Python-Based Introduction

[2] 소스 코드 및 데이터 세트

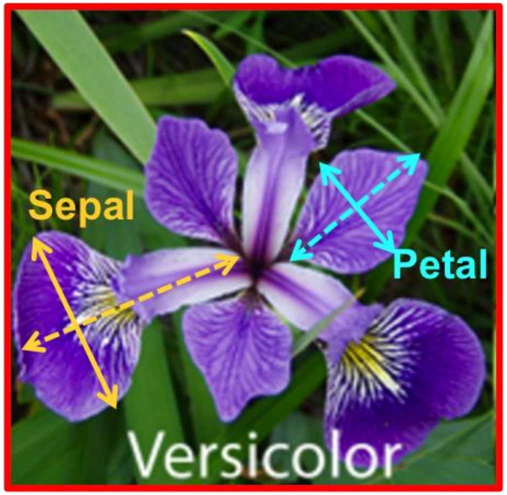
- 소스

https://github.com/rkneusel9/PracticalDeepLearningPython/

- 데이터

https://archive.ics.uci.edu/dataset/53/iris

Iris 꽃 종류

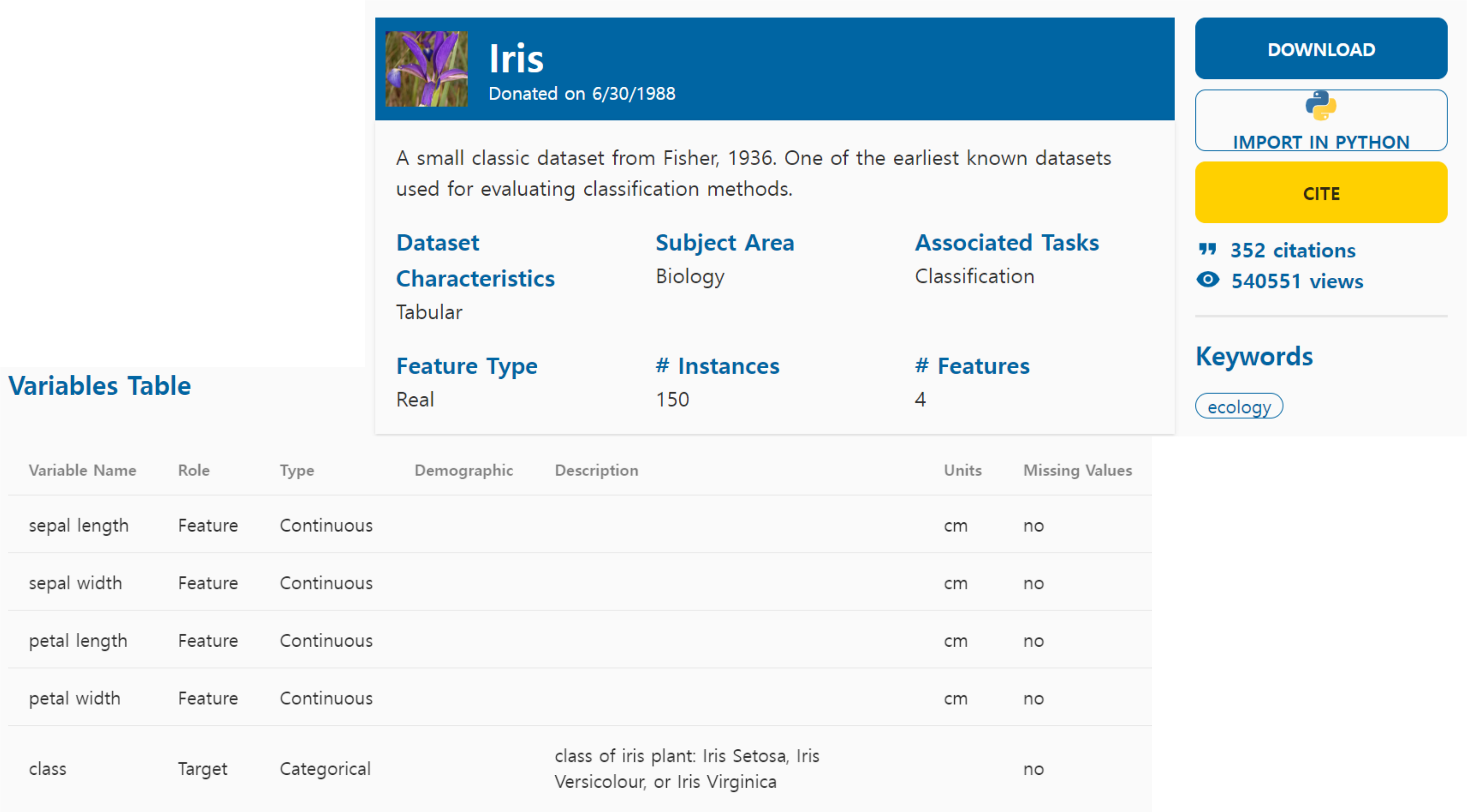


- Setosa: 부채 붓꽃
- Versicolour: 베르시 붓꽃
- Virginica: 버지니카 붓꽃

Iris 꽃 데이터

UC Irvine ML Repository

https://archive.ics.uci.edu/dataset/53/iris



Iris

Donated on 6/30/1988

A small classic dataset from Fisher, 1936. One of the earliest known datasets used for evaluating classification methods.

Dataset Characteristics	Subject Area	Associated Tasks
Tabular	Biology	Classification

Feature Type	# Instances	# Features
Real	150	4

352 citations

540551 views

Keywords

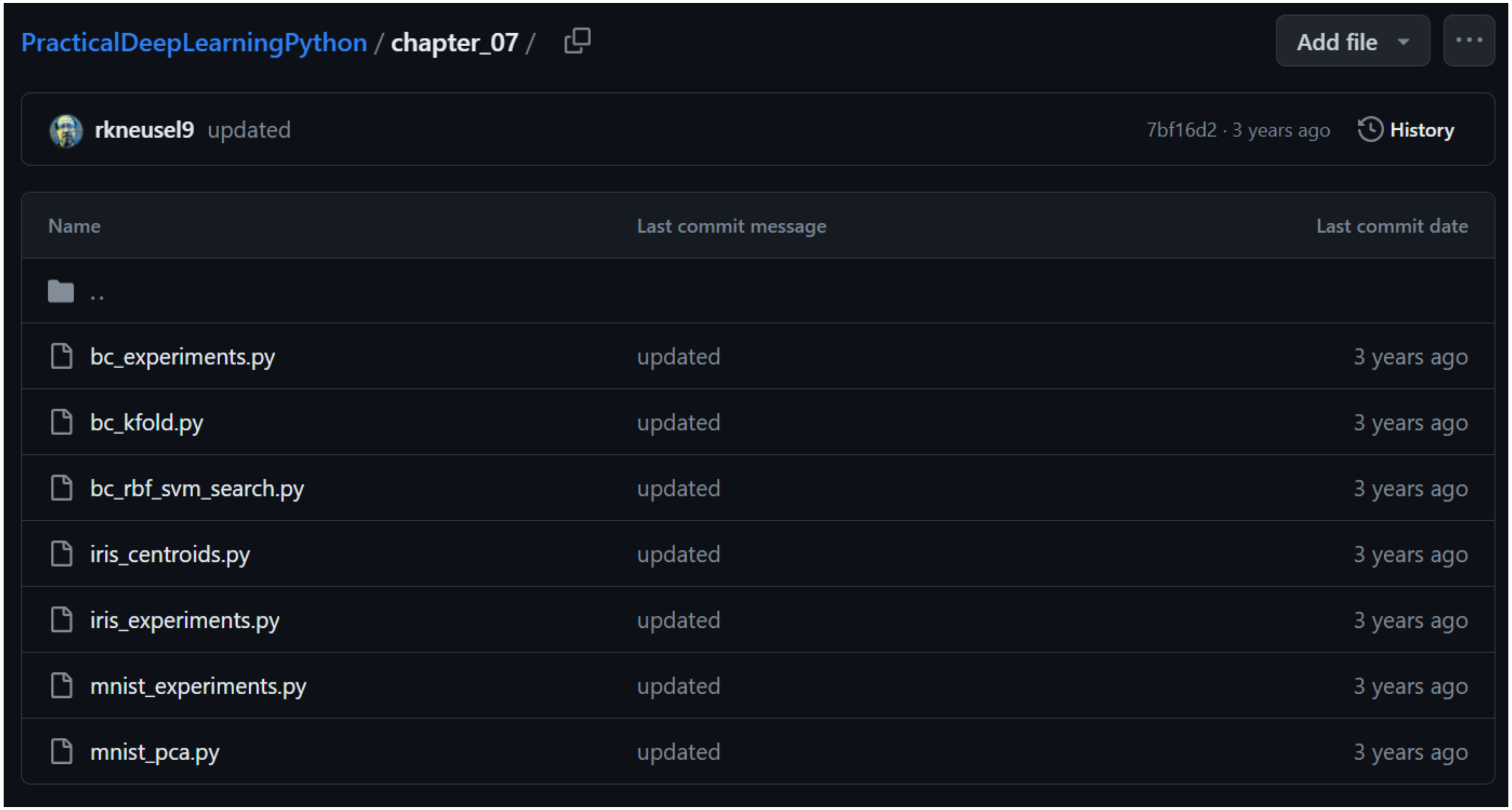
ecology

Variables Table

Variable Name	Role	Type	Demographic	Description	Units	Missing Values
sepal length	Feature	Continuous			cm	no
sepal width	Feature	Continuous			cm	no
petal length	Feature	Continuous			cm	no
petal width	Feature	Continuous			cm	no
class	Target	Categorical		class of iris plant: Iris Setosa, Iris Versicolour, or Iris Virginica		no

Iris 꽃 분류 소스 코드

https://github.com/rkneusel9/PracticalDeepLearningPython/tree/main/chapter_07



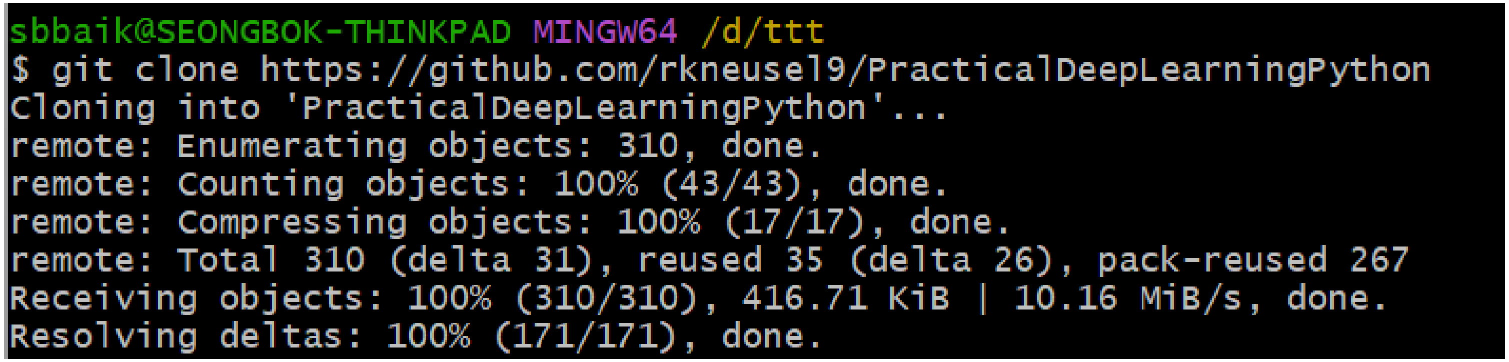
PracticalDeepLearningPython / chapter_07 /

rkneusel9 updated 7bf16d2 · 3 years ago

Name	Last commit message	Last commit date
..		
bc_experiments.py	updated	3 years ago
bc_kfold.py	updated	3 years ago
bc_rbf_svm_search.py	updated	3 years ago
iris_centroids.py	updated	3 years ago
iris_experiments.py	updated	3 years ago
mnist_experiments.py	updated	3 years ago
mnist_pca.py	updated	3 years ago

소스 코드 준비

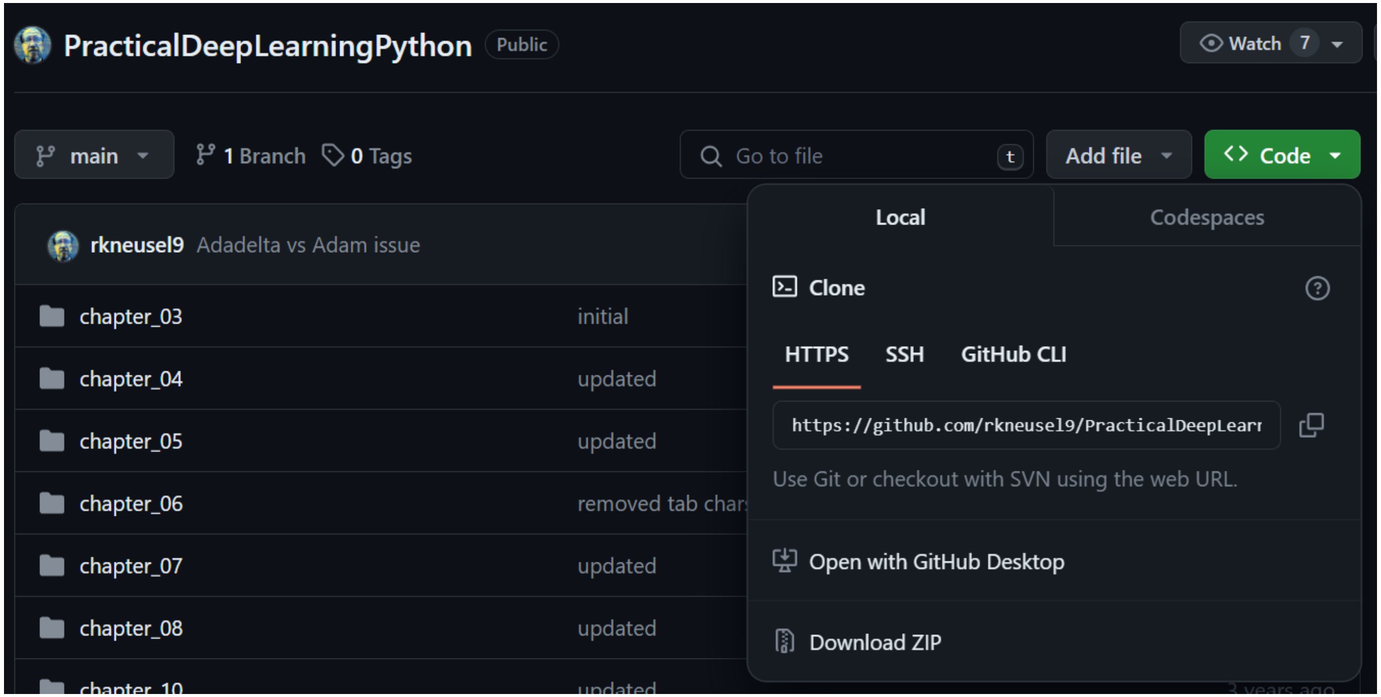
% git clone https://github.com/rkneusel9/PracticalDeepLearningPython



```
sbbaik@SEONGBOK-THINKPAD MINGW64 /d/ttt
$ git clone https://github.com/rkneusel9/PracticalDeepLearningPython
Cloning into 'PracticalDeepLearningPython'...
remote: Enumerating objects: 310, done.
remote: Counting objects: 100% (43/43), done.
remote: Compressing objects: 100% (17/17), done.
remote: Total 310 (delta 31), reused 35 (delta 26), pack-reused 267
Receiving objects: 100% (310/310), 416.71 KiB | 10.16 MiB/s, done.
Resolving deltas: 100% (171/171), done.
```

또는, zip 파일

다운로드



PracticalDeepLearningPython Public

rkneusel9 Adadelta vs Adam issue

chapter_03	initial
chapter_04	updated
chapter_05	updated
chapter_06	removed tab char
chapter_07	updated
chapter_08	updated
chapter_10	updated

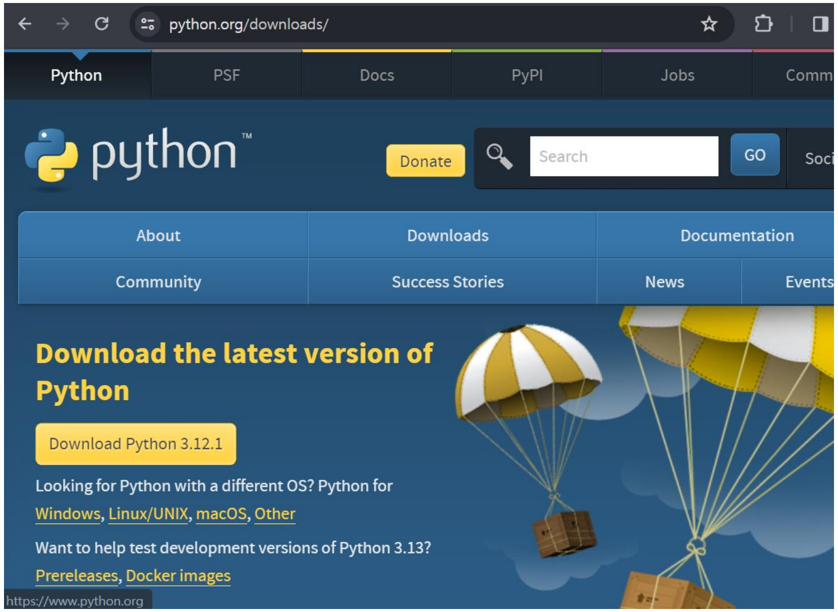
Clone

https://github.com/rkneusel9/PracticalDeepLearn

Use Git or checkout with SVN using the web URL.

실행 환경 구성

- 파이썬 설치 ==> python.org/downloads



Download the latest version of Python

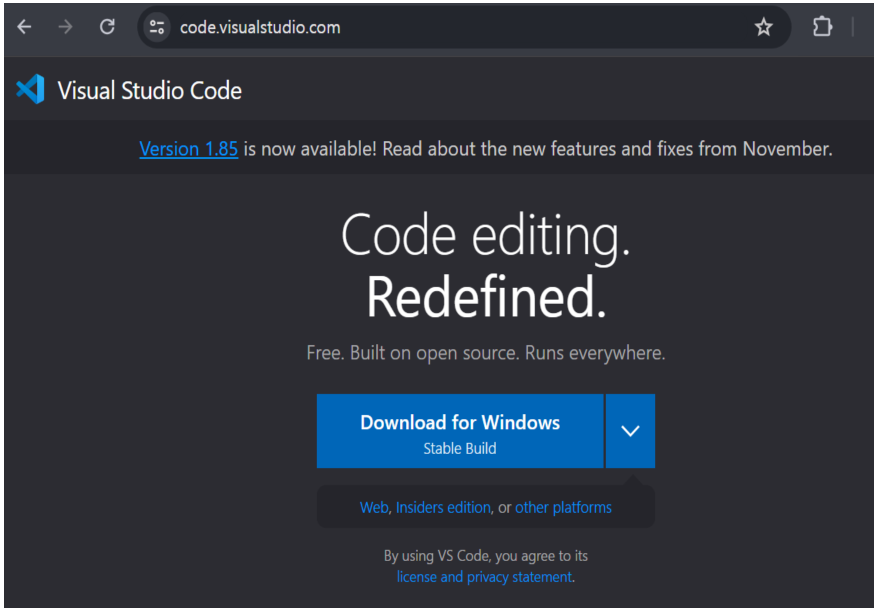
Looking for Python with a different OS? Python for Windows, Linux/UNIX, macOS, Other

Want to help test development versions of Python 3.13? Prereleases, Docker images

- 파이썬 패키지 설치

```
- pip install numpy         # 넘파이
- pip install scikit-learn  # 싸이킷-런 패키지
```

- 개발 환경 설치 ==> vs code



Version 1.85 is now available! Read about the new features and fixes from November.

Code editing.
Redefined.

Free. Built on open source. Runs everywhere.

Web, Insiders edition, or other platforms

By using VS Code, you agree to its license and privacy statement.

- 또는, 다른 개발 환경 ==> 파이참(PyCharm)

Iris 꽃 데이터 로딩
load_iris.py

```
with open("data/iris/iris.data") as f:
    lines = [i[:-1] for i in f.readlines()]

n = ["Iris-setosa","Iris-versicolor","Iris-virginica"]
x = [n.index(i.split(",")[-1]) for i in lines if i != ""]
x = np.array(x, dtype="uint8")

y = [[float(j) for j in i.split(",")[:-1]] for i in lines if i != ""]
y = np.array(y)

i = np.argsort(np.random.random(x.shape[0]))
x = x[i]
y = y[i]

np.save("data/iris/iris_features.npy", y)
np.save("data/iris/iris_labels.npy", x)
```

'₩n' 문자 지우기

붓꽃 종류별 레이블 0,1,2로 달아주기

붓꽃 데이터 값 2차원 배열로 저장 (레코드, 4가지 특징값)

무작위로 섞어 주기

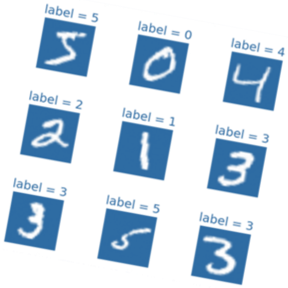
Iris 꽃 분류 학습

iris_experiments.py

```
...
from sklearn.neighbors import NearestCentroid
...

def run(x_train, y_train, x_test, y_test, clf):
    clf.fit(x_train, y_train)
    print("    predictions  :", clf.predict(x_test))
    print("    actual labels:", y_test)
    print("    score = %0.4f" % clf.score(x_test, y_test))

def main():
    x = np.load("../data/iris/iris_features.npy")
    y = np.load("../data/iris/iris_labels.npy")
    N = 120
    x_train = x[:N]; x_test = x[N:]
    y_train = y[:N]; y_test = y[N:]
    xa_train=np.load
("../data/iris/iris_train_features_augmented.npy")
    ya_train=np.load
("../data/iris/iris_train_labels_augmented.npy")
    xa_test =np.load
("../data/iris/iris_test_features_augmented.npy")
    ya_test =np.load
("../data/iris/iris_test_labels_augmented.npy")
```

```
    print("Nearest centroid:")
    run(x_train, y_train, x_test, y_test, NearestCentroid())
    print("k-NN classifier (k=3):")
    run(x_train, y_train, x_test, y_test,
KNeighborsClassifier(n_neighbors=3))
    print("Naive Bayes classifier (Gaussian):")
    run(x_train, y_train, x_test, y_test, GaussianNB())
    print("Naive Bayes classifier (Multinomial):")
    run(x_train, y_train, x_test, y_test, MultinomialNB())
    print("Decision Tree classifier:")
    run(x_train, y_train, x_test, y_test, DecisionTreeClassifier())
    print("Random Forest classifier (estimators=5):")
    run(xa_train, ya_train, xa_test, ya_test,
RandomForestClassifier(n_estimators=5))

    print("SVM (linear, C=1.0):")
    run(xa_train, ya_train, xa_test, ya_test, SVC(kernel="linear",
C=1.0))
    print("SVM (RBF, C=1.0, gamma=0.25):")
    run(xa_train, ya_train, xa_test, ya_test, SVC(kernel="rbf",
C=1.0, gamma=0.25))
    print("SVM (RBF, C=1.0, gamma=0.001, augmented)")
    run(xa_train, ya_train, xa_test, ya_test, SVC(kernel="rbf",
C=1.0, gamma=0.001))
    print("SVM (RBF, C=1.0, gamma=0.001, original)")
    run(x_train, y_train, x_test, y_test, SVC(kernel="rbf", C=1.0,
gamma=0.001))
```

딥러닝 기반 이미지 분류



손글씨(MNIST) 분류하기

머신러닝 기반 이미지 분류

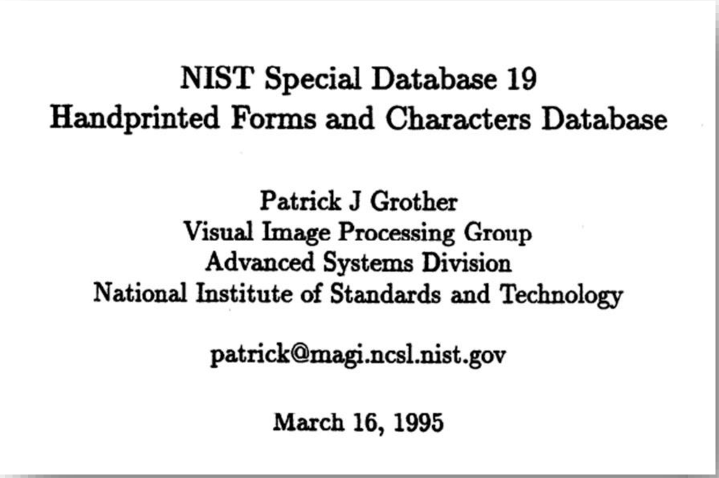
[1] 머신러닝을 이용한 이미지 분류 실습

- 아이리스 꽃 학습 데이터 세트 준비
- 데이터 전처리 및 피쳐 분석
- 머신러닝 모델 적용 및 실행
- 머신러닝 모델 별 성능 분석

[2] 딥러닝을 이용한 이미지 분류 실습

- MNIST 학습 데이터 세트 준비
- MNIST 데이터 피쳐 분석
- 딥러닝 모델(CNN) 적용 및 실행
- CNN 모델 구조 별 성능 분석

MNIST 데이터 세트

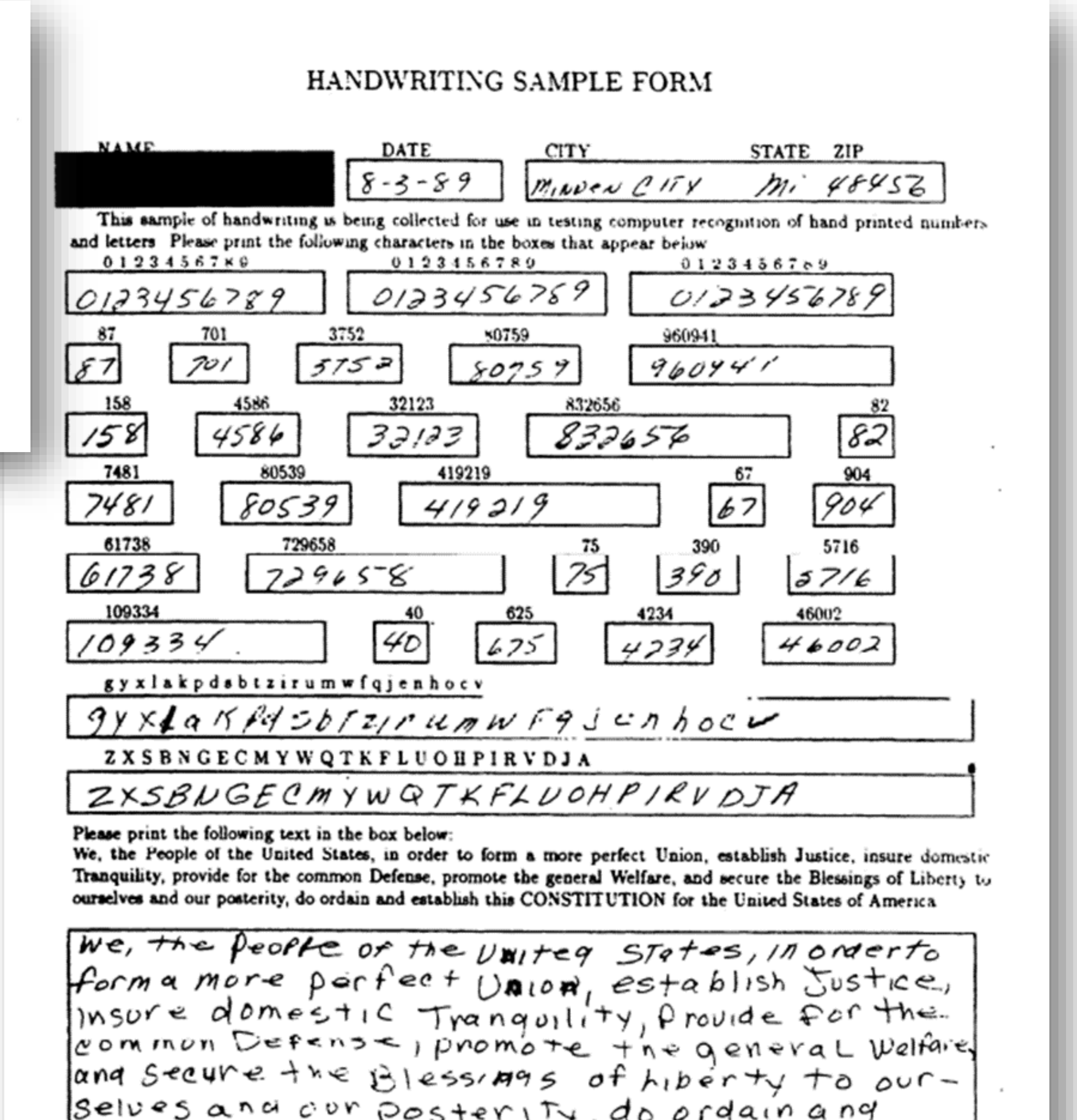


NIST Special Database 19
Handprinted Forms and Characters Database

Patrick J Grother
Visual Image Processing Group
Advanced Systems Division
National Institute of Standards and Technology

patrick@magi.ncsl.nist.gov

March 16, 1995



HANDWRITING SAMPLE FORM

NAME	DATE	CITY	STATE	ZIP
[redacted]	8-3-89	Minden City	MI	48456

This sample of handwriting is being collected for use in testing computer recognition of hand printed numbers and letters. Please print the following characters in the boxes that appear below.

0 1 2 3 4 5 6 7 8 9	0 1 2 3 4 5 6 7 8 9	0 1 2 3 4 5 6 7 8 9
0123456789	0123456789	0123456789

87	701	3752	80759	960941
87	701	3752	80759	960941
158	4586	32123	832656	82
158	4586	32123	832656	82
7481	80539	419219	67	904
7481	80539	419219	67	904
61738	729658	75	390	5716
61738	729658	75	390	5716
109334	40	625	4234	46002
109334	40	625	4234	46002

g y x l a k p d s b t z i r u m w f q j e n h o c v

gyxlakpdsbtzirumwfqjenhocv

Z X S B N G E C M Y W Q T K F L U O H P I R V D J A

ZXSBNGECMYWQTKFLUOHPIRVDJA

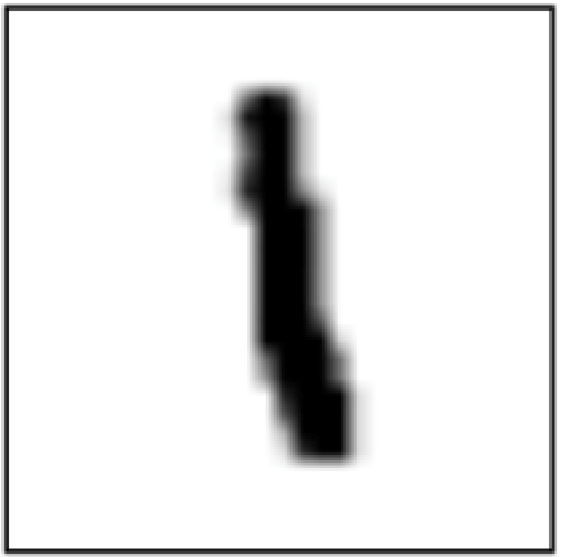
Please print the following text in the box below:
We, the People of the United States, in order to form a more perfect Union, establish Justice, insure domestic Tranquility, provide for the common Defense, promote the general Welfare, and secure the Blessings of Liberty to ourselves and our posterity, do ordain and establish this CONSTITUTION for the United States of America

We, the PeoPle of the Uniteq States, in orderto
form a more perfect Union, establish Justice,
insure domestic Tranquility, Provide for the
common Defense, promote the general Welfare,
anq Secure the Blessings of Liberty to our-
selves and our posterity, do ordain and

Modified **NIST**
데이터베이스

MNIST 데이터 세트

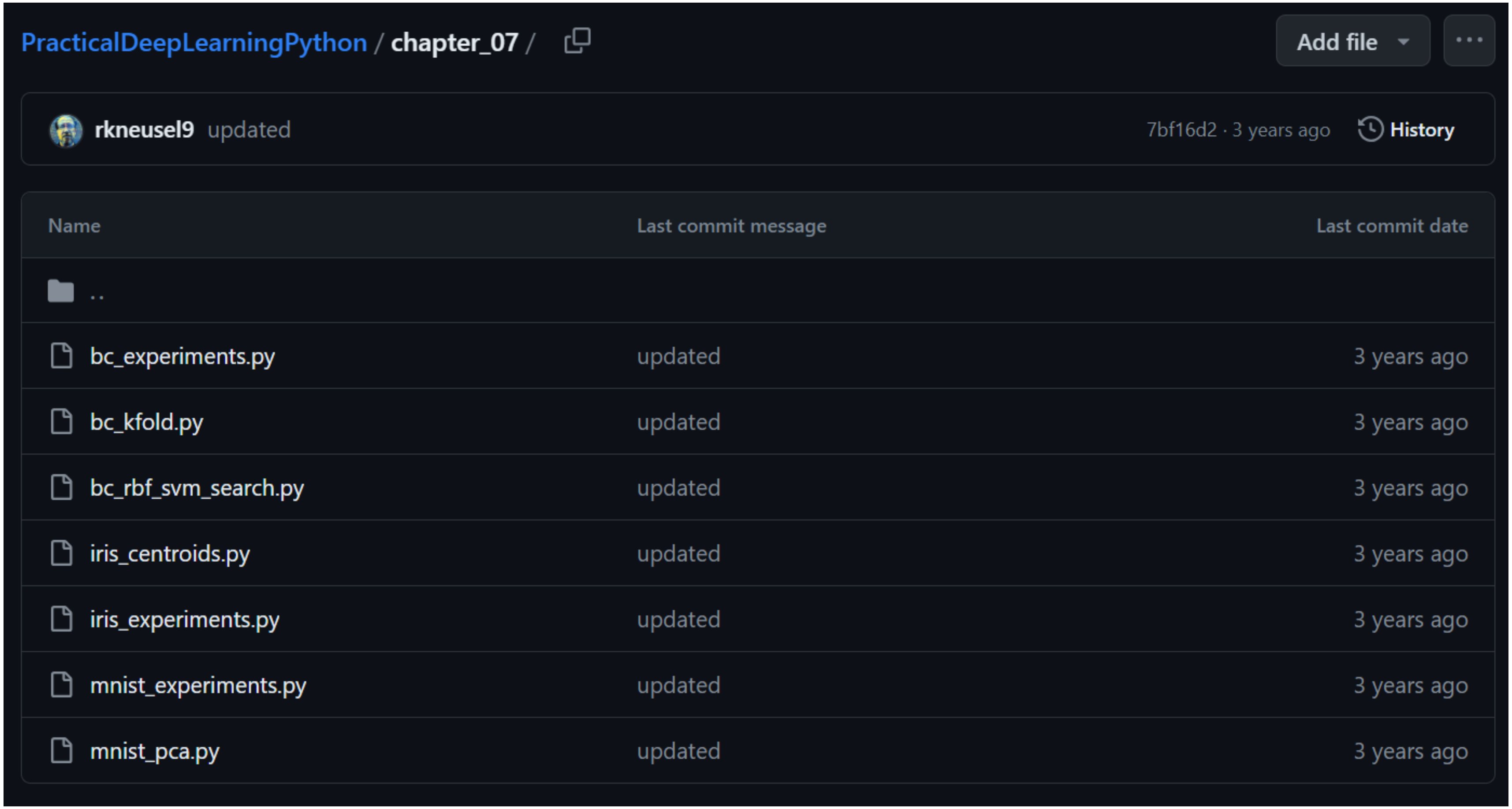
- 딥러닝 분야의 Hello, World
- 60,000개의 training set / 10,000개의 test set
- 28x28 흑백 이미지



$$\approx \begin{bmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&.6&.8&0&0&0&0&0&0\\
0&0&0&0&0&0&.7&1&0&0&0&0&0&0\\
0&0&0&0&0&0&.7&1&0&0&0&0&0&0\\
0&0&0&0&0&0&.5&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.7&0&0&0&0&0\\
0&0&0&0&0&0&0&1&1&0&0&0&0&0\\
0&0&0&0&0&0&0&.9&1&.1&0&0&0&0\\
0&0&0&0&0&0&0&.3&1&.1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix}$$

ML기반 MNIST 소스 코드

https://github.com/rkneusel9/PracticalDeepLearningPython/tree/main/chapter_07



rkneusel9 updated 7bf16d2 · 3 years ago

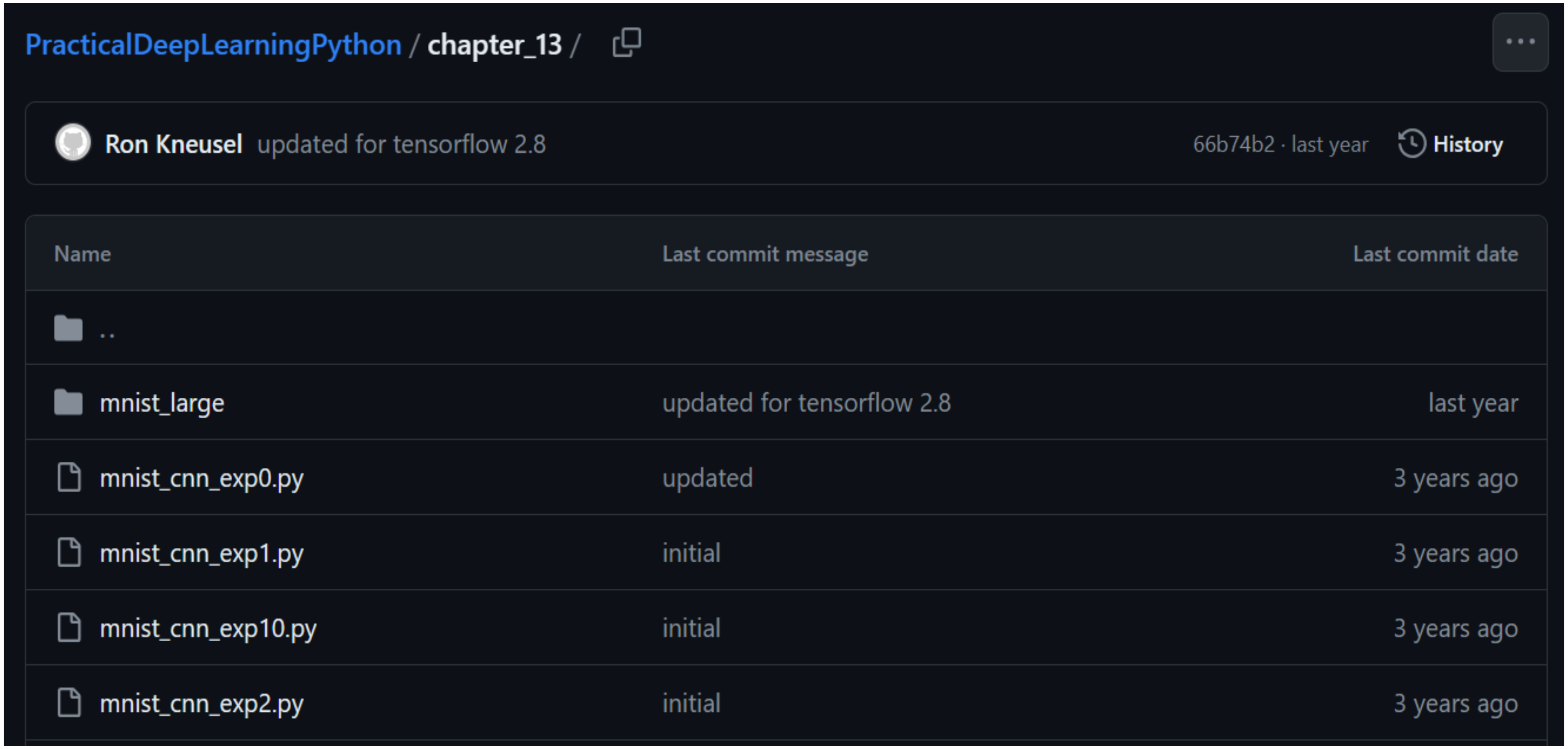
Name	Last commit message	Last commit date
..		
bc_experiments.py	updated	3 years ago
bc_kfold.py	updated	3 years ago
bc_rbf_svm_search.py	updated	3 years ago
iris_centroids.py	updated	3 years ago
iris_experiments.py	updated	3 years ago
mnist_experiments.py	updated	3 years ago
mnist_pca.py	updated	3 years ago

ML기반 MNIST 분류 성능

- LinearSVM (C=10.0)   : score = 0.8784 (time, train= 880.605, test=   0.035)
- Nearest centroid   : score = 0.7523 (time, train=   0.024, test=   0.005)
- k-NN classifier (k=3)   : score = 0.9360 (time, train=   0.294, test=   4.541)
- k-NN classifier (k=7)   : score = 0.9372 (time, train=   0.170, test=   5.206)
- Naive Bayes (Gaussian)   : score = 0.7999 (time, train=   0.032, test=   0.025)
- Decision Tree   : score = 0.8422 (time, train=   5.981, test=   0.006)
- Random Forest (trees= 5) : score = 0.8816 (time, train=   4.082, test=   0.016)
- Random Forest (trees= 50) : score = 0.9252 (time, train= 41.157, test=   0.138)
- Random Forest (trees=500) : score = 0.9270 (time, train= 472.070, test=   1.372)
- Random Forest (trees=1000): score = 0.9269 (time, train= 820.068, test=   2.689)

DL기반 MNIST 소스 코드

https://github.com/rkneusel9/PracticalDeepLearningPython/tree/main/chapter_13



PracticalDeepLearningPython / chapter_13 /

Ron Kneusel updated for tensorflow 2.8

66b74b2 · last year

Name	Last commit message	Last commit date
..		
mnist_large	updated for tensorflow 2.8	last year
mnist_cnn_exp0.py	updated	3 years ago
mnist_cnn_exp1.py	initial	3 years ago
mnist_cnn_exp10.py	initial	3 years ago
mnist_cnn_exp2.py	initial	3 years ago

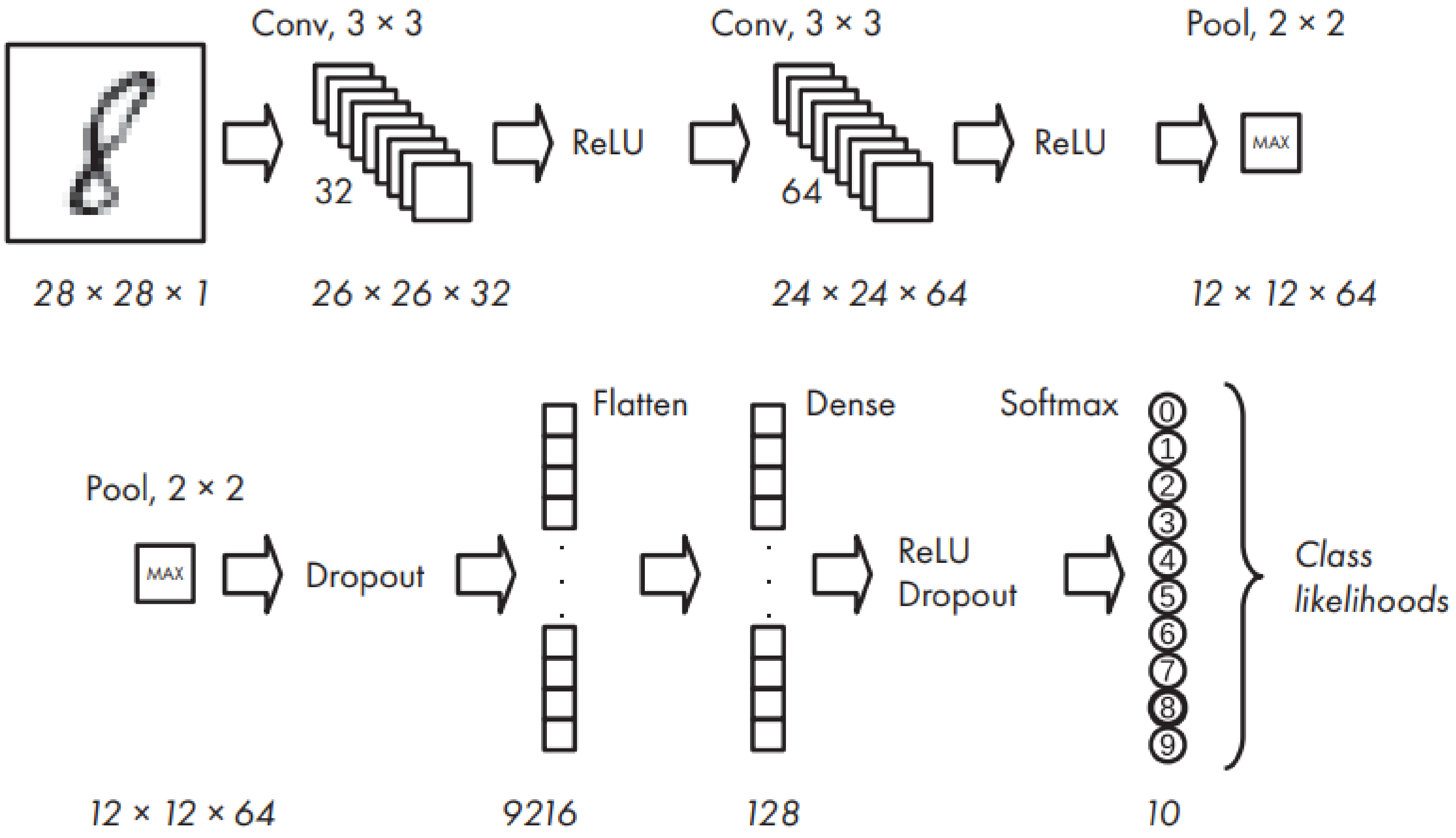
mnist_cnn_exp0.py

```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K

(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255

y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

history = model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
```

mnist_cnn_exp0.py

CNN기반 모델 구조



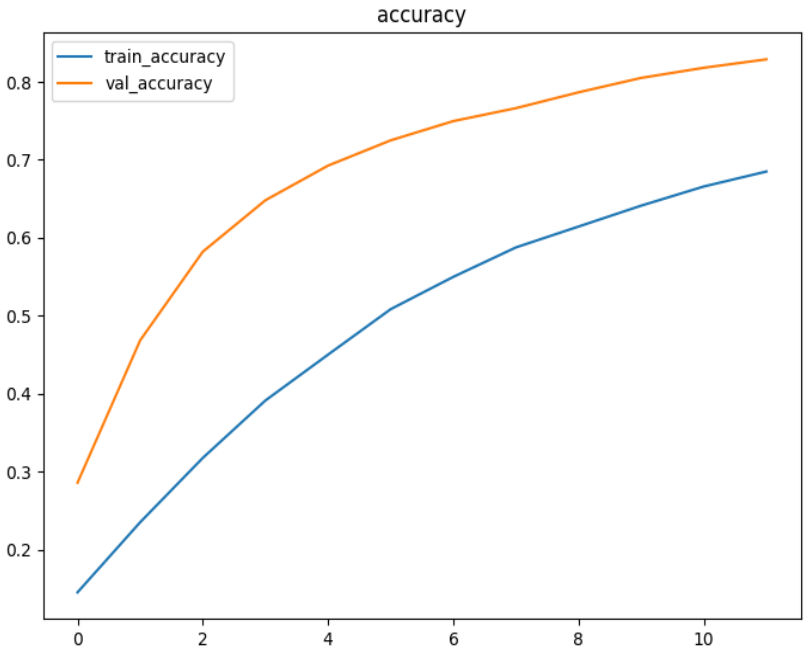
mnist_cnn_exp0.py

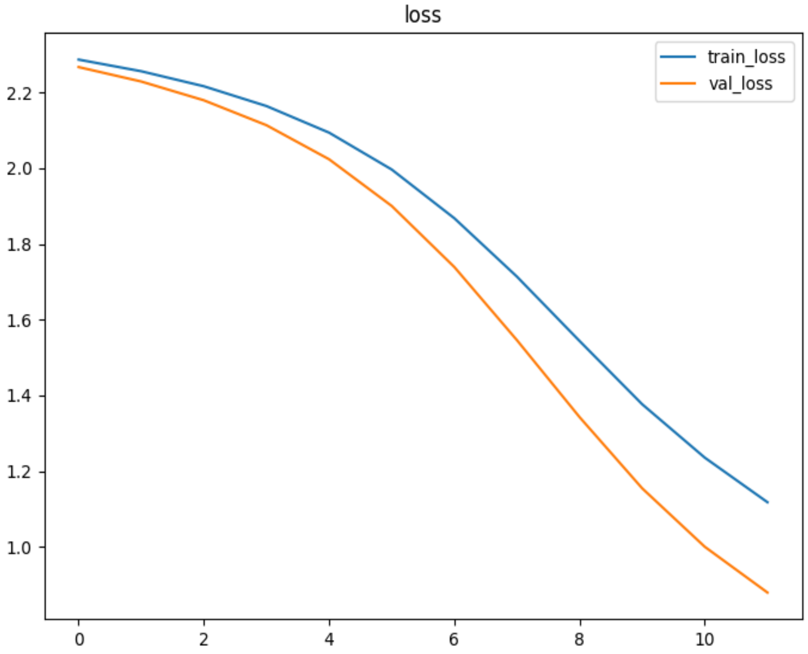
model summary

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 conv2d_1 (Conv2D)           (None, 24, 24, 64)        18496

 max_pooling2d(MaxPooling2D) (None, 12, 12, 64)        0

 dropout (Dropout)           (None, 12, 12, 64)        0

 flatten (Flatten)           (None, 9216)              0

 dense (Dense)               (None, 128)               1179776

 dropout_1 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 1199882 (4.58 MB)
Trainable params: 1199882 (4.58 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

mnist_cnn_exp0.py
분류 성능

- Epoch 12/12 469/469 [==============================]
  - 4s 9ms/step - loss: 1.1179 - accuracy: 0.6851 - val_loss: 0.8793 - val_accuracy: 0.8289





전처리 추가

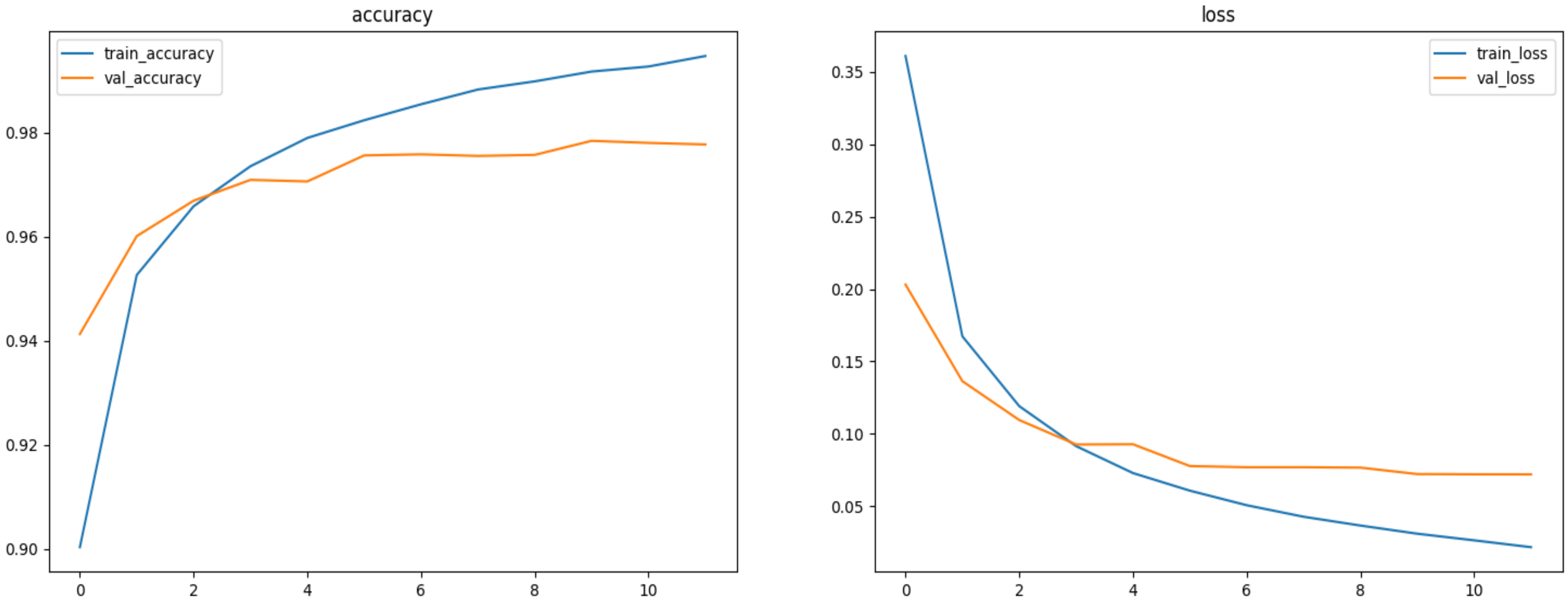
https://colab.research.google.com/github/tensorflow/datasets/blob/master/docs/keras_example.ipynb?hl=ko

```
import tensorflow as tf
import tensorflow_datasets as tfds

(ds_train, ds_test), ds_info = tfds.load(
    'mnist',
    split=['train', 'test'],
    shuffle_files=True,
    as_supervised=True,
    with_info=True,
)

def normalize_img(image, label):
  """Normalizes images: `uint8` -> `float32`."""
  return tf.cast(image, tf.float32) / 255., label

ds_train = ds_train.map(
    normalize_img, num_parallel_calls=tf.data.AUTOTUNE)
ds_train = ds_train.cache()
ds_train =
ds_train.shuffle(ds_info.splits['train'].num_examples)
ds_train = ds_train.batch(128)
ds_train = ds_train.prefetch(tf.data.AUTOTUNE)
```

```
ds_test = ds_test.map(
    normalize_img, num_parallel_calls=tf.data.AUTOTUNE)
ds_test = ds_test.batch(128)
ds_test = ds_test.cache()
ds_test = ds_test.prefetch(tf.data.AUTOTUNE)

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(10)
])
model.compile(
    optimizer=tf.keras.optimizers.Adam(0.001),

loss=tf.keras.losses.SparseCategoricalCrossentropy(from_l
ogits=True),
    metrics=[tf.keras.metrics.SparseCategoricalAccuracy()],
)

history = model.fit(
    ds_train,
    epochs=6+6,
    validation_data=ds_test,
)
```

전처리 추가 모델 분류 성능

-
  ```
  Epoch 12/12 469/469 [==============================]
  - 2s 3ms/step - loss: 0.0218 - sparse_categorical_accuracy: 0.9947
  - val_loss: 0.0721 - val_sparse_categorical_accuracy: 0.9777
  ```



CNN 기반 전처리 추가 모델

https://colab.research.google.com/github/tensorflow/datasets/blob/master/docs/keras_example.ipynb?hl=ko

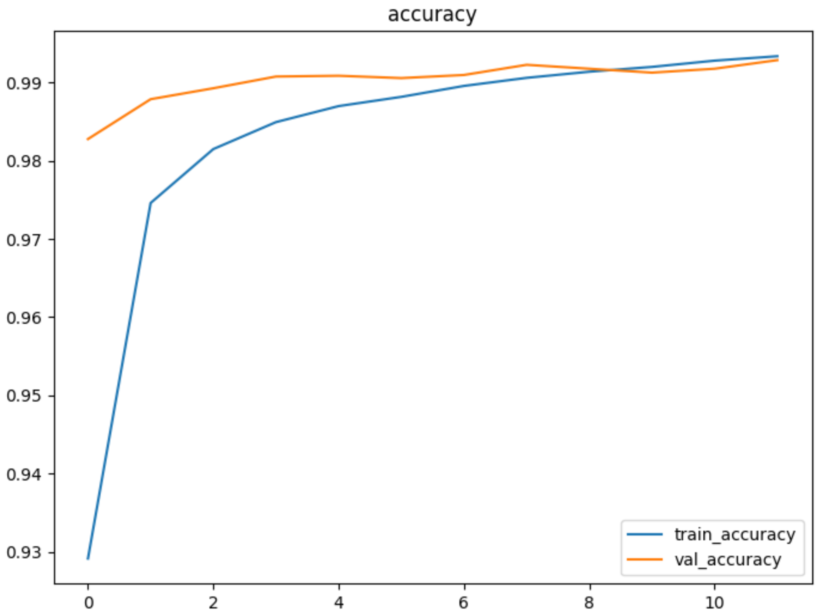
```
(ds_train, ds_test), ds_info = tfds.load(
    'mnist',
    split=['train', 'test'],
    shuffle_files=True,
    as_supervised=True,
    with_info=True,
)
def normalize_img(image, label):
    """Normalizes images: `uint8` -> `float32`."""
    return tf.cast(image, tf.float32) / 255., label
ds_train = ds_train.map(normalize_img,
num_parallel_calls=tf.data.AUTOTUNE)
ds_train = ds_train.cache()
ds_train = ds_train.shuffle(ds_info.splits['train'].num_examples)
ds_train = ds_train.batch(128)
ds_train = ds_train.prefetch(tf.data.AUTOTUNE)

ds_test = ds_test.map(normalize_img,
num_parallel_calls=tf.data.AUTOTUNE)
ds_test = ds_test.batch(128)
ds_test = ds_test.cache()
ds_test = ds_test.prefetch(tf.data.AUTOTUNE)
```

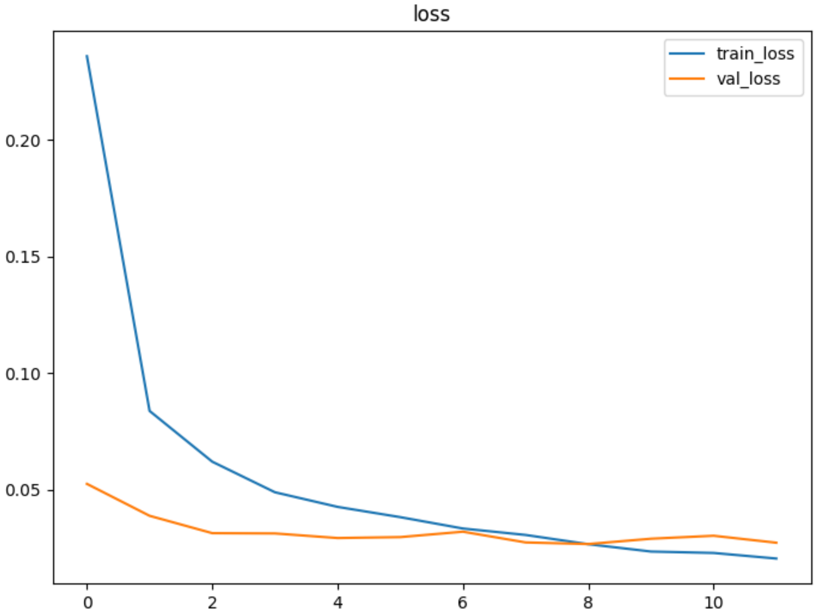
```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
input_shape=(28, 28, 1)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

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

print("Model parameters = %d" % model.count_params())
print(model.summary())

history = model.fit(
    ds_train,
    epochs=12,
    validation_data=ds_test,
)
```

CNN 기반 전처리 추가 모델 성능

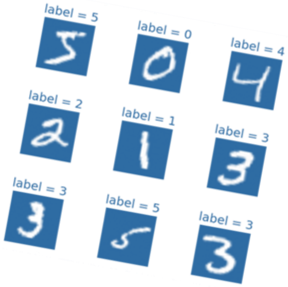
- Epoch 12/12 469/469 [==============================]

  - 5s 10ms/step - loss: 0.0204 - accuracy: 0.9934 - val_loss: 0.0272 - val_accuracy: 0.9929





Q&A


label = 5
label = 0
label = 4
label = 2
label = 1
label = 3
label = 3
label = 5
label = 3