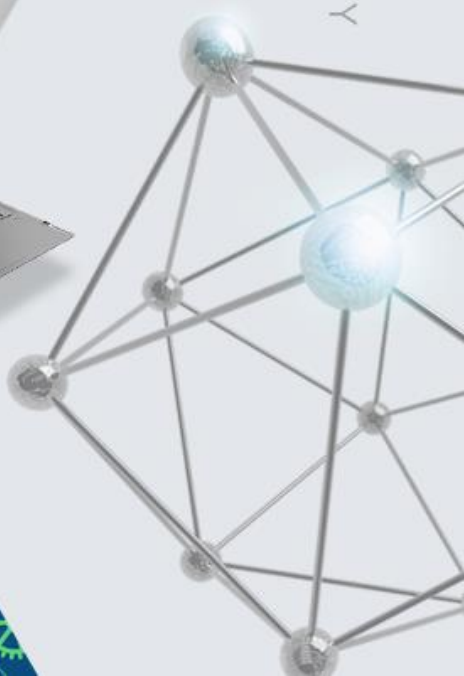
한국기술교육대학교
온라인평생교육원

The 4th Industrial Revolution is characterized by super connectivity and super intelligence, where various products and services are connected to the network, and artificial intelligence and information communication technologies are used in 3D printing, unmanned transportation, robotics, Of the world's most advanced technologies.

TECHNOLOGY

딥러닝 입문

MNIST 데이터셋을 활용한 딥러닝 실습



The 4th Industrial Revolution is characterized by super connectivity and super intelligence, where various products and services are connected to the network, and artificial intelligence and information communication technologies are used in 3D printing, unmanned transportation, robotics, Of the world's most advanced technologies.

MNIST 데이터셋을 활용한 딥러닝 실습

학/습/목/표

1. MNIST 데이터셋에 대해 이해하고, 직접 데이터를 사용할 수 있다.
2. 딥러닝 실습을 통해, 직접 딥러닝 모델을 구성하고 학습할 수 있다.

학/습/내/용

1. MNIST 데이터셋
2. 딥러닝 실습

1. MNIST 데이터셋

1) 텐서플로우 데이터셋

(1) 텐서플로우 데이터셋

- 인공지능은 많은 양의 데이터를 기반으로 특정한 패턴을 찾기를 기대하는 것
  - 데이터는 많을수록 좋지만 수집하기가 어려움
- 안좋은 데이터로 학습하게 되면 해당 모델의 성능 또한 기대하기 어려움

(2) 안좋은 데이터 예시

- 중복된 데이터가 많을 경우
  - 다양한 데이터를 볼 수 없어 컴퓨터가 제대로 학습되지 않음
- 정답 데이터의 분포가 한쪽에 치우쳐진 경우
  - 한쪽의 특징만을 잘 찾고 해결함
- 일반적이지 않은 데이터들인 경우
  - 특정 경우에만 문제를 잘 해결함
- 데이터의 상태(이미지 크기, 음질, 저장 방식 등)가 고르지 않은 경우

1. MNIST 데이터셋

1) 텐서플로우 데이터셋

(3) 텐서플로우

- 딥러닝 학습에 활용할 수 있는 데이터셋 공유
- 약 200여 개
- 음성 이미지 텍스트 등 각종 분야별로 제공

  - https://www.tensorflow.org/datasets/catalog/overview

  - https://github.com/tensorflow/datasets

- 텐서플로우 모듈을 설치하여 간단한 명령어로 원하는 데이터를 불러와 사용할 수 있음

```
from tensorflow.keras.datasets import mnist
(X_train, Y_train), (X_test, Y_test) = mnist.load_data()
```

```
print(X_train)
```

```
[[[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 0 0]]
```

1. MNIST 데이터셋

2) MNIST 데이터셋 이해

(1) MNIST 데이터셋

- 0부터 9까지의 손글씨 이미지 제공
  - 딥러닝 학습 시 가장 기초 실습 예제 데이터로 활용
- 훈련 데이터 6만개, 테스트 데이터 1만개로 총 7만개의 손글씨 데이터 제공
- 텐서플로우 모듈을 불러온 뒤에 mnist 로드데이터 함수 사용

```
from tensorflow.keras.datasets import mnist
(X_train, Y_train), (X_test, Y_test) = mnist.load_data()
```

```
print(len(X_train))
print(len(X_test))
```

```
60000
10000
```

```
print(len(Y_train))
print(len(Y_test))
```

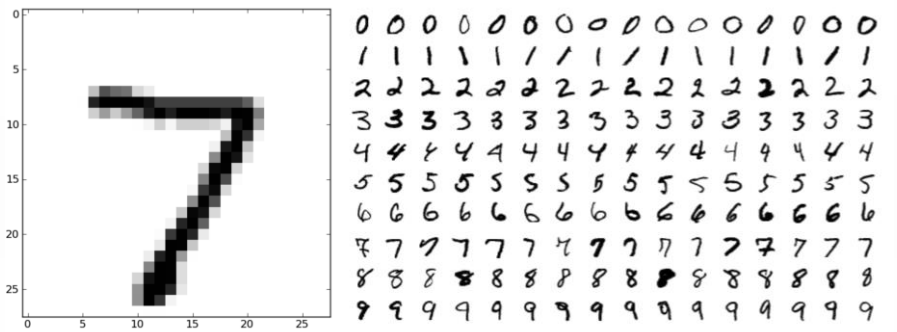
```
60000
10000
```

1. MNIST 데이터셋

2) MNIST 데이터셋 이해

(2) MNIST 데이터셋의 장점

- 데이터의 크기가 비교적 크지 않음
- 모델의 학습 결과를 눈으로 쉽게 확인하기 쉬움
- 가로(28) X 세로(28) 크기
- 기본적인 모델로도 성능이 95% 이상 나올 만큼 쉽고 편리한 예제
- 글씨가 써져 있는 부분은 값이 있고, 나머지는 0으로 구성

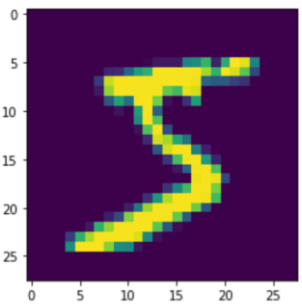


실제 이미지 예시

(3) matplotlib 모듈 사용

```
import matplotlib.pyplot as plt
plt.imshow(X_train[0])
```

```
<matplotlib.image.AxesImage at 0x7fa505436c90>
```



```
print(Y_train[0])
```

```
5
```

1) MNIST 데이터셋을 활용한 딥러닝 실습

(1) 필수 모듈 불러오기

- 모듈명을 쉽게 사용할 수 있도록 여러가지를 직접 명시적으로 적어줌

```
import tensorflow.keras.utils as utils
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
```

(2) 데이터셋 불러오기 & 훈련, 검증, 테스트 데이터셋 분리

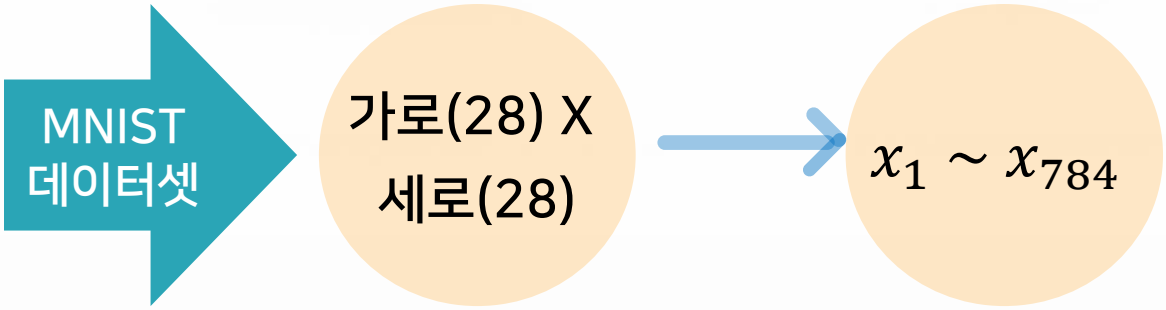
- 입력 데이터를 $x_1 \sim x_{784}$ 형태로 변경

```
(X_train, Y_train), (X_test, Y_test) = mnist.load_data()

X_val = X_train[50000:]
Y_val = Y_train[50000:]
X_train = X_train[:50000]
Y_train = Y_train[:50000]
```

1) MNIST 데이터셋을 활용한 딥러닝 실습

(3) 입력 데이터 변환



(4) 정답 데이터 라벨링 변환

- 기존 정답 데이터
  - 0, 1, 2 … 9 처럼 하나의 숫자
- 딥러닝 모델의 예측 결과
  - 0부터 9 각각 숫자마다 확률을 돌려주기 때문에 정답 개수만큼 배열형태로 변환
  - loss 손실값을 줄이는 것을 학습목표로 하기 대문에 여러 개 중에 한 개로 나올 수 있도록 이상적인 정답 값으로 변환해주어야 함

```
Y_train = utils.to_categorical(Y_train)
Y_val = utils.to_categorical(Y_val)
Y_test = utils.to_categorical(Y_test)
```

```
print(Y_train[0])
```

```
[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
```

0	1	2	3	4	5	6	7	8	9

1) MNIST 데이터셋을 활용한 딥러닝 실습

(5) 딥러닝 학습을 위한 모델 구성

MNIST 데이터기 때문에
28*28(784)로 고정

4개의 층을 쌓아 올리고 각각 출력의
개수, 활성화 함수 등을 설정

해당 설정들은 어느 정도 좋다고 알려진
방법들은 있지만 **확실한 정답은 없음!**

```
model = Sequential()
model.add(Dense(units=512, input_dim=28*28, activation='relu'))
model.add(Dense(units=256,  activation='relu'))
model.add(Dense(units=128,  activation='relu'))
model.add(Dense(units=10, activation='softmax'))
```

1) MNIST 데이터셋을 활용한 딥러닝 실습

(5) 딥러닝 학습을 위한 모델 구성

- Softmax
  - 여러 개 중에 하나를 분류를 해야 되는 문제에서 전체 출력의 데이터를 0과 1사이의 값으로 조정해주는 함수
- Softmax 함수를 사용한 이유
  - 이상적인 정답 데이터를 위해 사용
    (softmax는 입력받은 값을 출력으로 0~1사이의 값으로 모두 정규화하며 출력 값들의 총합은 항상 1이 되는 특성을 가진 함수)
  - 여러가지 정답이 있는 카테고리컬 분류 문제이기 때문
  - 만약 둘 중 하나를 나타내는 바이너리 문제면 sigmond 사용

1) MNIST 데이터셋을 활용한 딥러닝 실습

(6) 모델 엮기 및 모델 학습

- 학습 전 모델의 손실 함수 설정, 옵티마이저 설정, 평가 지표 설정
- 학습은 훈련 데이터와 검증 데이터를 적용, 총 반복 횟수와 배치 사이즈를 적고 학습 시작

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
hist = model.fit(X_train, Y_train, epochs=10, batch_size=50, validation_data=(X_val, Y_val))
```

```
Epoch 1/10
1000/1000 [==============================] - 3s 3ms/step - loss: 1.5085 - accuracy: 0.8874 -
```

(7) 모델 평가

- 테스트 데이터셋을 활용하여 학습 완료된 모델 평가
- loss 값은 낮을 수록, accuracy 값은 높을수록 좋은 모델

```
loss_and_metrics = model.evaluate(X_test, Y_test, batch_size=32)

print('')
print('loss : ' + str(loss_and_metrics[0]))
print('accuray : ' + str(loss_and_metrics[1]))
```

```
313/313 [==============================] - 0s 909us/step - loss:

loss : 0.12749293446540833
accuray : 0.9678000211715698
```

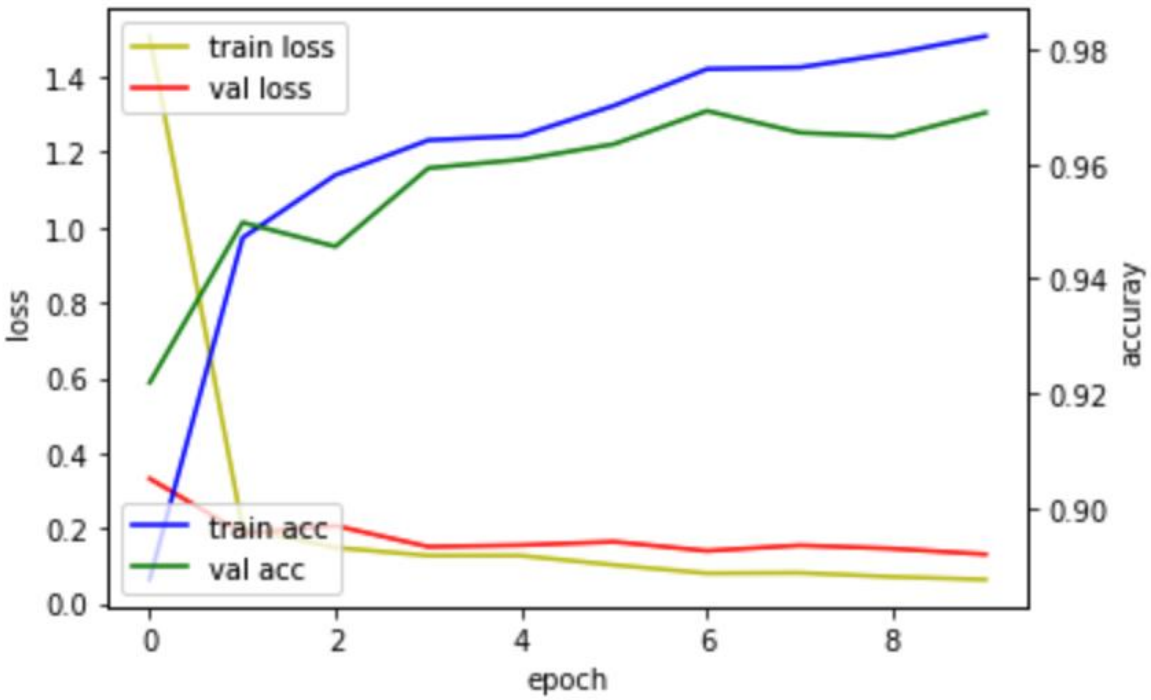
다양한 데이터, 다양한 방법으로 학습이 잘 되었음을 의미

1) MNIST 데이터셋을 활용한 딥러닝 실습

(8) 시각화

- 모델의 학습 과정을 한눈에 살펴볼 수 있도록 그래프로 시각화

  - 학습 정도 파악, 과적합 등 판단




```
fig, loss_ax = plt.subplots()

acc_ax = loss_ax.twinx()

loss_ax.plot(hist.history['loss'], 'y', label='train loss')
loss_ax.plot(hist.history['val_loss'], 'r', label='val loss')

acc_ax.plot(hist.history['accuracy'], 'b', label='train acc')
acc_ax.plot(hist.history['val_accuracy'], 'g', label='val acc')

loss_ax.set_xlabel('epoch')
loss_ax.set_ylabel('loss')
acc_ax.set_ylabel('accuray')

loss_ax.legend(loc='upper left')
acc_ax.legend(loc='lower left')

plt.show()
```

1. MNIST 데이터셋

- 딥러닝을 학습하기 위해서는 많은 양의 데이터가 필요하지만 일상생활에서 입력과 정답까지 모두 있는 데이터를 쉽게 구하기는 어려움
- 텐서플로우는 음성, 영상, 이미지, 텍스트 등 여러 타입의 데이터를 손쉽게 다운받아 사용할 수 있도록 제공
- MNIST 데이터셋 역시 텐서플로우에서 쉽게 다운로드 받아 활용할 수 있으며, 7만개의 손글씨 데이터를 제공
- MNIST 손글씨 하나의 데이터 크기는 28*28로 0부터 9까지의 손글씨 숫자와 해당하는 정답 데이터를 가지고 있음

2. 딥러닝 실습

- 딥러닝 학습을 위해 텐서플로우와 필요한 각종 모듈을 불러온 뒤, 데이터셋을 저장
- 불러온 데이터는 훈련데이터, 검증데이터, 테스트데이터로 다시 한 번 분리하여 사용
- 모델에 적용하기 위해 2D 입력데이터를 한 줄로 변환하는 작업과 한 개의 숫자인 정답 데이터를 배열 형태로 변환
- 여러 개의 층을 쌓아 모델을 구성하고, 옵티마이저 등을 정해 모델을 엮어 학습을 시작할 수 있음
- 학습의 정도는 테스트 데이터셋을 활용한 평가와 시각화 모듈을 활용한 그래프로 보다 쉽게 파악할 수 있음