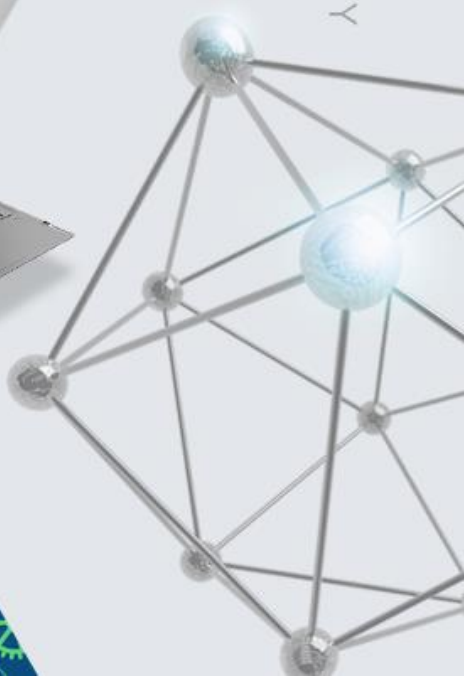
한국기술교육대학교
온라인평생교육원

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

CIFAR-10 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.

CIFAR-10 데이터셋을 활용한 딥러닝 실습

학/습/목/표

1. 텐서플로우 모듈의 여러가지 활용법에 대해 학습하고, 직접 코딩할 수 있다.
2. 딥러닝 실습을 통해, 직접 딥러닝 모델을 구성하고 학습할 수 있다.

학/습/내/용

1. 텐서플로우 모듈 활용
2. 딥러닝 실습

1. 텐서플로우 모듈 활용

1) 조기종료를 위한 콜백 함수

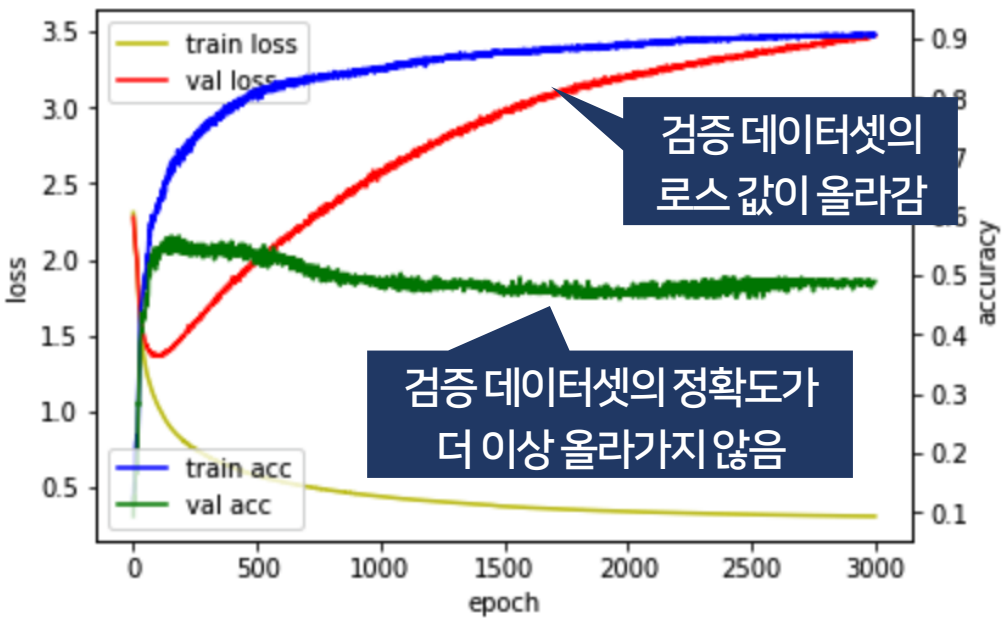
(1) 텐서플로우 데이터셋

- 텐서플로우 모듈

  - 딥러닝 학습을 위해 다양한 기능을 제공

  - 그중 사용하기 편리한 기능 중 하나가 조기종료를 위한 콜백 함수

- 실제 딥러닝 모델을 학습할 때 반복학습을 많이하면 훈련 데이터의 과적합이 됨



- 검증 데이터를 활용해 학습 종료 시점을 알 수 있지만 결과는 계속 달라질 수 있음

1. 텐서플로우 모듈 활용

1) 조기종료를 위한 콜백 함수

(1) 텐서플로우 데이터셋

- 텐서플로우 모듈

  - 과적합을 방지하기 위한 조기종료 함수 제공

- EarlyStopping

  - 모델이 더 이상 개선의 여지가 없다고 판단할 때 학습을 종료하는 콜백 함수

- 학습이 잘 되고 있다고 판단하는 기준

  - loss(손실) 값이 반복마다 점점 떨어지는 것

- 과적합 되었다고 판단하는 기준

  - loss(손실) 값이 다음 반복에 오히려 올라감

1) 조기종료를 위한 콜백 함수

(2) 콜백 함수

- 특정 함수를 수행할 때 해당 함수에서 내가 지정한 함수를 호출하는 것
- EarlyStopping 함수
  - loss 값이 오히려 올라가게 되면 자동으로 학습 종료

```
from tensorflow.keras.callbacks import EarlyStopping
early_stopping = EarlyStopping() # 조기종료 콜백함수 정의
hist = model.fit(X_train, Y_train,
                 epochs=3000,
                 batch_size=500,
                 validation_data=(X_val, Y_val),
                 callbacks=[early_stopping])
```

```
Epoch 1/3000
100/100 [==============================] - 1s 8ms/step - loss: 0.0040 - accuracy: 0.9985 - val_loss: 0.1562 - val_acc
uracy: 0.9801
Epoch 2/3000
100/100 [==============================] - 1s 8ms/step - loss: 0.0039 - accuracy: 0.9985 - val_loss: 0.1532 - val_acc
uracy: 0.9801
Epoch 3/3000
100/100 [==============================] - 1s 8ms/step - loss: 0.0037 - accuracy: 0.9986 - val_loss: 0.1603 - val_acc
uracy: 0.9796
```

- 어렵고 복잡한 문제의 경우 loss 값이 불안정하게 감소할 수 있음 (특히 학습 초반)
  - 학습이 덜 됐지만 우연히 정답을 맞춘 경우 학습이 종료되는 상황 발생

1. 텐서플로우 모듈 활용

1) 조기종료를 위한 콜백 함수

(2) 콜백 함수

- patience 옵션을 활용

  - 몇 번 연속 올라가지 않으면 일시적인 경우로 보고 다시 학습 진행

```
from tensorflow.keras.callbacks import EarlyStopping
early_stopping = EarlyStopping(patience = 10)
hist = model.fit(X_train, Y_train,
                 epochs=1000,
                 batch_size=100,
                 validation_data=(X_val, Y_val),
                 callbacks=[early_stopping])
```

```
Epoch 1/1000
500/500 [==============================] - 2s 4ms/step - loss: 0.0224 - accuracy: 0.9945 - val_loss: 0.1877 - val_acc
uracy: 0.9728
Epoch 2/1000
500/500 [==============================] - 2s 4ms/step - loss: 0.0376 - accuracy: 0.9910 - val_loss: 0.1143 - val_acc
uracy: 0.9791
Epoch 3/1000
500/500 [==============================] - 2s 4ms/step - loss: 0.0202 - accuracy: 0.9943 - val_loss: 0.1387 - val_acc
uracy: 0.9787
Epoch 4/1000
500/500 [==============================] - 2s 4ms/step - loss: 0.0202 - accuracy: 0.9950 - val_loss: 0.1293 - val_acc
uracy: 0.9769
Epoch 5/1000
500/500 [==============================] - 2s 4ms/step - loss: 0.0196 - accuracy: 0.9947 - val_loss: 0.1475 - val_acc
uracy: 0.9782
Epoch 6/1000
500/500 [==============================] - 2s 4ms/step - loss: 0.0187 - accuracy: 0.9954 - val_loss: 0.1436 - val_acc
uracy: 0.9782
Epoch 7/1000
500/500 [==============================] - 2s 4ms/step - loss: 0.0158 - accuracy: 0.9958 - val_loss: 0.1324 - val_acc
uracy: 0.9797
Epoch 8/1000
500/500 [==============================] - 2s 5ms/step - loss: 0.0201 - accuracy: 0.9954 - val_loss: 0.1520 - val_acc
uracy: 0.9774
Epoch 9/1000
500/500 [==============================] - 2s 5ms/step - loss: 0.0173 - accuracy: 0.9961 - val_loss: 0.1456 - val_acc
uracy: 0.9781
Epoch 10/1000
500/500 [==============================] - 2s 4ms/step - loss: 0.0121 - accuracy: 0.9970 - val_loss: 0.1987 - val_acc
uracy: 0.9748
Epoch 11/1000
500/500 [==============================] - 2s 5ms/step - loss: 0.0144 - accuracy: 0.9964 - val_loss: 0.2368 - val_acc
uracy: 0.9768
Epoch 12/1000
500/500 [==============================] - 2s 4ms/step - loss: 0.0198 - accuracy: 0.9959 - val_loss: 0.1703 - val_acc
uracy: 0.9784
```

1. 텐서플로우 모듈 활용

1) 조기종료를 위한 콜백 함수

(3) 텐서플로우 모듈의 종류

- 인공 신경망의 기본 모델인 다층 퍼셉트론
- CNN 모델 : 이미지 처리에 적합
- RNN 모델 : 자연어 처리 등 연속 데이터에 적합

(4) 텐서플로우 모듈 사용법

```
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import SimpleRNN
```

텐서플로우 모듈에서 여러 가지 모델을
손쉽게 사용할 수 있도록 제공

1) CIFAR-10 데이터셋을 활용한 딥러닝 실습

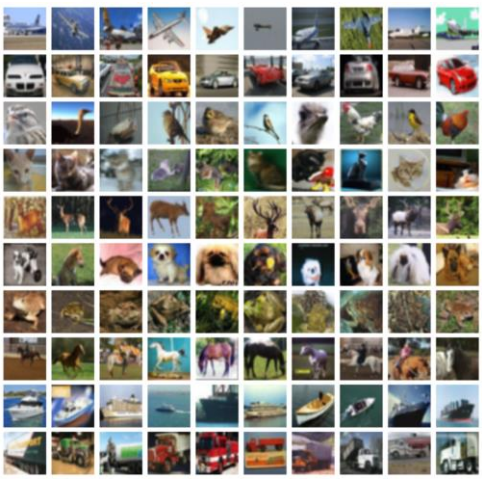
(1) CIFAR-10 데이터셋

- 인공지능 모델 학습용으로 제공되는 데이터셋
- MNIST 데이터셋보다 난이도가 높음
- 10종류의 컬러 이미지 제공(CIFAR-10)
  - 100종류의 이미지 데이터를 제공하는 CIFAR-10 데이터셋도 있음
- 텐서플로우 모듈에서 손쉽게 다운로드 및 활용 가능

```
cifar10_data = cifar10.load_data()
((X_train, Y_train), (X_test, Y_test)) = cifar10_data
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170500096/170498071 [==============================] - 40s 0us/step
```

- 이미지의 크기는 32X32X3이며, 10가지 종류의 이미지 제공



훈련 데이터 50,000개

테스트 데이터 10,000개

총 60,000개의 이미지 데이터를 제공

1) CIFAR-10 데이터셋을 활용한 딥러닝 실습

(2) CIFAR-10 데이터셋을 활용한 딥러닝 실습

- 텐서플로우 및 각종 딥러닝 학습에 필요한 모듈 호출 + 학습 데이터 저장

```
import tensorflow as tf
import numpy as np
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
from tensorflow.keras.datasets import cifar10
import matplotlib.pyplot as plt
```

```
cifar10_data = cifar10.load_data()
((X_train, Y_train), (X_test, Y_test)) = cifar10_data
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170500096/170498071 [==============================] - 40s 0us/step
```

- 다층 퍼셉트론 모델에 적용시키기 위한 데이터 전처리 (데이터셋 분리, 입,출력 데이터 수정)

```
X_val = X_train[40000:]
Y_val = Y_train[40000:]
X_train = X_train[:40000]
Y_train = Y_train[:40000]

X_train = X_train.reshape(40000, 32*32*3).astype('float32')
X_val = X_val.reshape(10000, 32*32*3).astype('float32')
X_test = X_test.reshape(10000, 32*32*3).astype('float32')
```

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

1) CIFAR-10 데이터셋을 활용한 딥러닝 실습

(2) CIFAR-10 데이터셋을 활용한 딥러닝 실습

- 학습 모델 구성(입력 데이터, 출력 데이터의 크기 중요)

  - 정답이 정해져 있는 것이 아니고 MNIST 데이터보다는 CIFAR-10 데이터가 훨씬 어렵고 복잡한 문제이기 때문에 임의로 많은 출력층의 개수를 작성함

```
model = Sequential()
model.add(Dense(units=512, input_dim=32*32*3, activation='relu'))
model.add(Dense(units=256,  activation='relu'))
model.add(Dense(units=256,  activation='relu'))
model.add(Dense(units=128,  activation='relu'))
model.add(Dense(units=128,  activation='relu'))
model.add(Dense(units=64,  activation='relu'))
model.add(Dense(units=64,  activation='relu'))
model.add(Dense(units=10, activation='softmax'))
```

여러 가지 정답이 있는 카테고리컬 분류 문제이기 때문

- 모델 엮기 및 모델 학습

  - 조기 종료를 위한 콜백 함수 사용 : 학습이 과적합 되는 것을 미리 판단하여 조기종료 할수 있는 옵션

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
hist = model.fit(X_train, Y_train, epochs=30, batch_size=32, validation_data=(X_val, Y_val))
```

```
Epoch 1/30
1250/1250 [==============================] - 9s 7ms/step - loss: 1.3543 - accuracy: 0.5188 - v
ccuracy: 0.4663
Epoch 2/30
```

1) CIFAR-10 데이터셋을 활용한 딥러닝 실습

(2) CIFAR-10 데이터셋을 활용한 딥러닝 실습

- 모델 평가

  - 모델 구성, 반복 횟수 등을 조정하여 성능을 조금 높일 수 있음

```
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]))
```

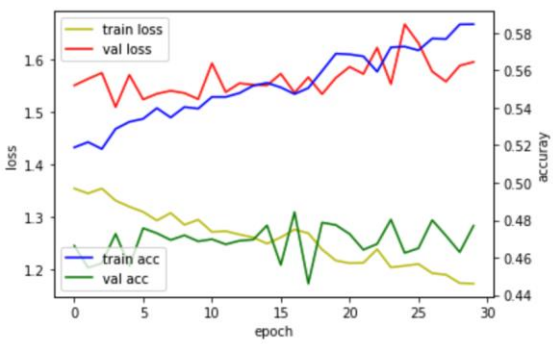
```
313/313 [==============================] - 1s 2ms/step - loss: 1.

loss : 1.5759062767028809
accuray : 0.47940000891685486
```

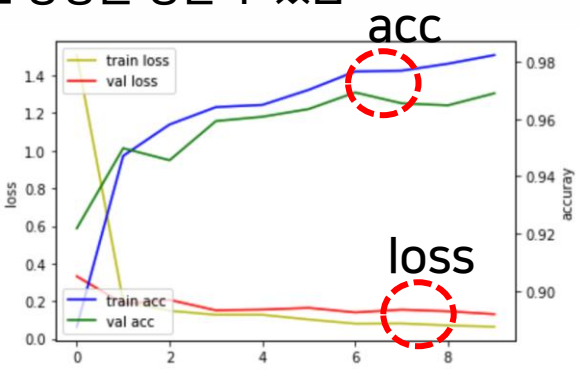
MNIST 데이터셋(28*28) 보다 훨씬 어려운 데이터셋(32*32*3)으로 아직 학습이 덜 되었고 성능이 좋지 못함

- 시각화

  - 시각화한 결과를 보고 모델 개선 방향을 정할 수 있음



CIFAR-10 데이터셋 결과



MNIST 데이터셋 결과

Summary

1. 텐서플로우 모듈 활용

- 텐서플로우 2.0버전부터는 케라스를 함께 사용할 수 있어 딥러닝 모델 설계 및 학습이 매우 편리한 장점이 있음
- 특히, 학습에 필요한 다양한 데이터셋이나 복잡한 수식을 통해 계산되던 딥러닝의 어려운 모델들을 간단한 함수 호출로 사용할 수 있음
- 딥러닝 학습 시 자주 겪는 과적합 문제 또한 텐서플로우 모듈의 조기종료 콜백 함수를 사용하면 어렵지 않게 해결 가능

2. 딥러닝 실습

- CIFAR-10 데이터셋은 MNIST 데이터셋보다 복잡한 이미지 데이터셋임
- 10가지 종류의 총 6만개 이미지 데이터를 제공하며, 이미지의 크기는 32X32X3임
- 딥러닝 학습을 위해 텐서플로우와 필요한 각종 모듈을 불러온 뒤, 데이터셋을 저장함
- 불러온 데이터는 훈련데이터, 검증데이터, 테스트데이터로 다시 한 번 분리하여 사용함
- 모델에 적용하기 위해 2D 입력데이터를 한 줄로 변환 하는 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