기계학습개론

- MNIST digit DB 분류기 설계-

|  |  |
| --- | --- |
| 담당 교수 |  |
| 제출 일자 |  |
| 학과 |  |
| 학번 |  |
| 이름 | 안정은 |

목차

[1. 구현 환경 3](#_Toc531550123)

[2. 구현 방법 3](#_Toc531550124)

[3. Parameter 변경 및 성능 시험 결과 4](#_Toc531550125)

[(1) Edge detector의 구조 (size, vertical/horizontal/diagonal) 4](#_Toc531550126)

[(2) CNN의 구조 5](#_Toc531550127)

[(3) Pooling 선택 (MAX, Average) 6](#_Toc531550128)

[(4) Activation function 6](#_Toc531550129)

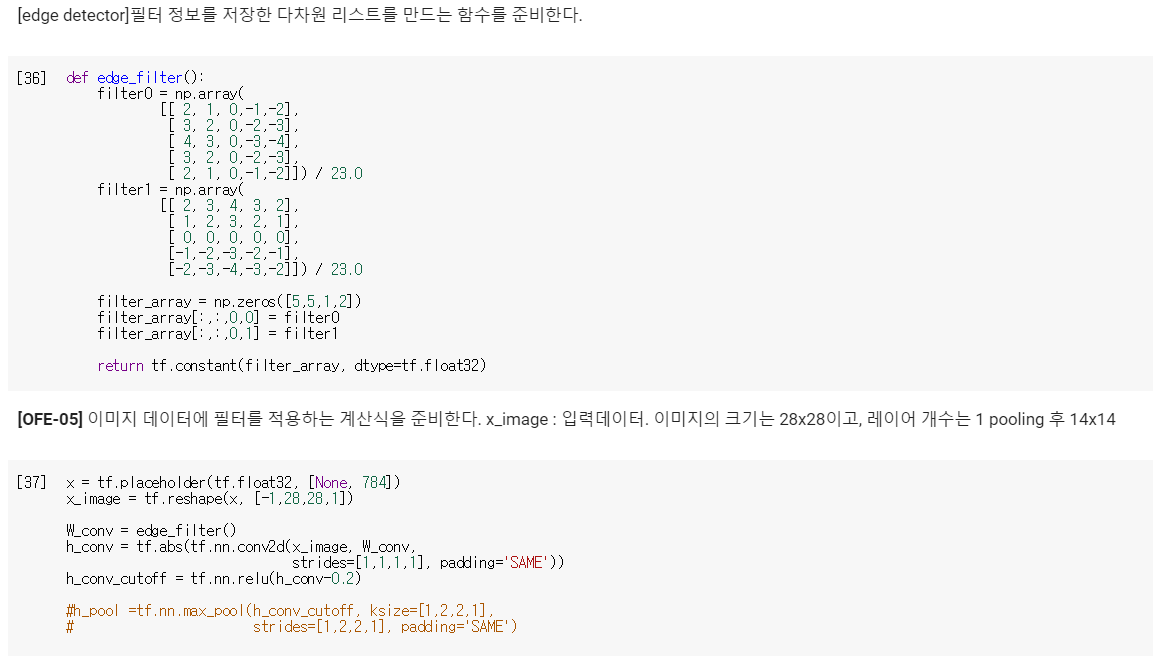
# 구현 환경

Colab-jupter notebook

# 구현 방법

1. Sample 코드인 ‘MNIST double layer CNN classification.ipynb’ 이 클라우드 환경에서 제대로 동작하도록 텐서플로우 최신버전 및 python3에 맞게 코드 수정
2. 초단에 edge detector 추가

부교재인 ‘텐서플로로 시작하는 딥러닝’의 chapter4 코드를 참고하였다.



첨부한 파일은 다음과 같은 edge filter를 사용하였으며, filter의 크기 등을 수정하여 성능을 비교하였다. Filter0은 세로 엣지를 추출하고, filter 1은 가로 엣지를 추출한다.

|  |  |
| --- | --- |
| Filter0만 사용한 경우 | Filter1만 사용한 경우 |
|  |  |
| 필터의 크기를 3 by 3으로 변경 |  |
|  |  |

1. 실행결과

|  |  |
| --- | --- |
| … |  |
|  |

최종적으로 98%정도의 정확도를 보였다.

# Parameter 변경 및 성능 시험 결과

## (1) Edge detector의 구조 (size, vertical/horizontal/diagonal)

- vertical/horizontal/vertical+horizontal

|  |  |
| --- | --- |
| loss | accuracy |
|  |  |

가로엣지와 세로엣지를 모두 검출하는 코드에서 성능이 가장 좋았다.

* Edge detector의 크기

|  |  |
| --- | --- |
| loss | accuracy |
|  |  |

가로엣지와 세로엣지를 둘 다 검출하되 크기를 달리하여 비교하였다. 필터의 크기가 작을 때 성능이 더 좋았다.

## (2) CNN의 구조

* 필터의 개수 변경

|  |  |  |
| --- | --- | --- |
| N1=128  N2=64 | N1=64  N2=32 | N1=32  N2=16 |
| Time(min) = 6.81 | Time(min) = 4.09 | Time(min) = 3.31 |

|  |  |
| --- | --- |
| loss | accuracy |
|  |  |

필터의 개수가 많을수록 시간이 오래 걸렸지만 성능은 더 좋았다.

- 필터의 크기 변경

필터의 개수는 N1=64 N2=32로 설계하였다.

|  |  |  |
| --- | --- | --- |
| 3 by 3 | 5 by 5 | 7 by 7 |
| Time(min) = 2.58 | Time(min) = 3.19 | Time(min) =4.61 |

|  |  |
| --- | --- |
| loss | accuracy |
|  |  |

필터의 크기가 클수록 시간이 오래 걸렸다. 하지만 성능은 5 by 5와 7 by 7사이에서는 크게 차이나지 않았다.

## Pooling 선택 (MAX, Average)

|  |  |
| --- | --- |
| MAX\_Pooling | Average\_Pooling |
| Time(min) = 4.50 | Time(min) = 4.64 |

|  |  |
| --- | --- |
| loss | accuracy |
|  |  |

Max pooling을 사용하였을 때 성능이 좋았다.

## Activation function

Activation function이 쓰이는 곳에 다음 세가지 종류의 함수를 각각 사용하였다.

|  |  |  |
| --- | --- | --- |
| Sigmoid | ReLU | tanh |
| Time(min) = 3.41 | Time(min) = 3.56 | Time(min) =3.48 |

|  |  |
| --- | --- |
| loss | accuracy |
|  |  |

Sigmoid가 현저히 성능이 떨어지는 것을 확인할 수 있다. 위 그래프에서는 ReLU와 tanh의 차이가 크지 않아 보인다. 다음의 그래프를 보자.

|  |  |
| --- | --- |
| loss | accuracy |
|  |  |

두가지 그래프만 놓고 비교한 결과 ReLU를 사용하였을 때 성능이 훨씬 좋은 것을 확인할 수 있다.