종합설계
최종보고서

손글씨 숫자 인식
인공지능 구현하기

학번	이름
2017010698	오서영
2018080048	이정
2017010709	조지수
2017010702	이지수

목차

Ⅰ. 수행개요

1. 목적 및 필요성

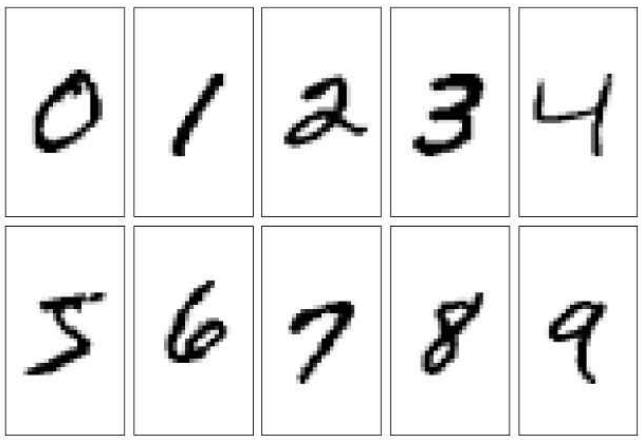
4차 산업혁명 시대를 맞아 인공지능의 역할이 커지고 있지만 아직 수학은 학문으로만 취급받고 있다. 그러나 수학은 인공지능 시대의 혁신 방법이면서 동시에 개인과 국가 경쟁력의 원천이다.

인공지능 연구는 응용 수학이라고 할 만큼 수학이 매우 광범위하게 활용되고 있다. 특히 행렬은 데이터의 공간변환, 인공지능망 최적 설계, 확률의 추출과정에서 필수적인 도구이다. 또한 딥러닝 학습은 데이터가 제시하는 문제를 잘 해결하는 방향으로 시냅스 연결강도를 변경하면서 이루어지는데, 이때 어떻게 변경해야 할지를 편미분을 사용하여 구하게 된다. 그리고 확률이론과 최적화 문제, 수렴속도 분석에 다양한 수학적 아이디어가 많이 활용된다.

우리는 수학의 또 다른 측면을 발견하기 위해 수학이 많이 사용되는 인공지능 모델을 구현해보고자 한다.

2. 목표

최종 구현한 모델이 손으로 쓴 0부터 9까지의 숫자 이미지를 잘 분류 하는 것이 목표이다.



< 손글씨 숫자 이미지 예>

1) 인공지능 구현을 위한 기초 언어와 수학적 개념을 학습한다.

인공지능을 잘 이해하여 구현하기 위해서는 수학적 도구를 선행 학습할 필요가 있고 구현을 하면서 생기는 여러 오류와 현상을 잘 이해하기 위해서 코딩 공부 또한 미리 할 필요가 있다.

2) 기본적인 Neural Network 모델을 구현하고 테스트한다.

우리가 풀어보려고 하는 문제는 이미지 분류 문제이다. 이미지 분류 모델의 가장 기본적인 구조는 Neural Network로 알려져 있기 때문에, 먼저 이를 구현해 볼 것이다.

3) 위 결과를 보완하기 위한 새 모델이나 기법을 학습한다.
모델의 정확도를 더 높이기 위해 새 모델이나 기법을 다시 공부 할 필요가 있다.

4) 최종 모델을 구현하고 여러 사람의 손글씨를 수집하여 테스트한다.
구현한 모델은 수집한 데이터로 학습한 것이므로, 실제 사람의 손글씨 데이터가 모델을 통해 잘 분류가 되는지 테스트 해볼 필요가 있다.

Ⅱ. 수행계획

1. 선행학습

Neural Network 모델 구현에 앞서 컴퓨터 언어와 수학적 개념의 선행학습이 필요하다.
학습방법 : 구글의 학습 자료를 참고하여 일주일에 한번씩 각자 맡은 내용을 다른 조원에게 설명한다.

1) 컴퓨터 언어

(1) 파이썬(Python)
: 파이썬은 데이터 과학 분야를 위한 표준 프로그래밍 언어이다. 머신러닝은 근본적으로 반복 작업인데 파이썬은 이를 빠르게 처리하고 손쉽게 조작가능하다.

2) 수학적 개념

(1) Matrix : 4차 산업혁명 시대의 디지털 데이터는 묶음 형태로 존재한다. 그래서 인공지능이 다루는 입출력 데이터는 다차원 '행렬' 형태를 갖게 된다. 인공지능을 구현하는 과정을 보면, 행렬 데이터를 곱하고 더하기를 반복한다고 볼 수 있다.

(2) Loss Function (손실 함수) : 데이터에 대한 예측 값과 실제 값을 비교하는 함수이다. 가장 좋은 모델은 예측한 값과 실제 값이 거의 차이가 없어야 한다. 즉 손실함수가 최소화 돼야 하는데, 이때 아래와 같은 최적화 방법을 사용한다.

(3) Gradient Descent (경사 하강법) : 손실함수의 최솟값 위치를 찾기 위해 gradient 반대 방향으로 조금씩 움직이면서 최적 해를 찾는 방법이다.

2. 모델 구현하기

아래와 같은 단계를 거쳐 모델을 구현한다.

1) 모델 구상 및 데이터 수집

Neural Network는 목적에 따라 사용되는 함수와 기법이 다르다. 모델을 구현하기에 앞서, 어떤 활성화 함수와 손실함수, 최적화 방법을 사용 할 것인지 세부적인 내용을 먼저 구상한다. 그리고 목적에 맞는 좋은 데이터를 다양한 매체를 통해 수집한다.

2) Neural Network 코드 초안 작성

선행 학습을 통해 배운 파이썬을 활용하여 모델에 사용되는 수학식들을 모델링하고, 앞서 수집한 데이터를 코딩을 통해 불러와 행렬의 형태로 변환시킨다.

3) 모델 구현 및 수정

각각 따로 만들어낸 파이썬 함수들을 한곳으로 불러와 모델을 구현하고 실행시킨다. 실행 후 오류를 분석하여 잘못된 코드를 수정한다.

4) 정확도 확인 및 분석

train/test 정확도를 확인하고 분석한다. train 정확도가 낮다면 underfitting 된 것이고, train 정확도는 높지만 상대적으로 test 정확도가 낮다면 overfitting 된 것이다. underfitting과 overfitting 모두 없도록 모델을 구현해야 한다.
만약 새로운 기법이나 모델이 필요하다고 판단되면 또 다시 개념을 학습한다.

5) 최종 모델 구현 및 점검

최종적으로 모델을 구현하고 오류가 없도록 다시 한 번 점검한다.

6) 제작완료

점검에 문제가 없다면 다음 단계를 수행한다.

3. 모델 평가

1) 모델 분석

최종 테스트 세트의 정확도가 97% 이상이 되면, 모델 구현이 적절하게 이루어졌다고 판단한다.

2) 실제 데이터 테스트

구현한 모델은 수집한 데이터로 학습한 것이므로, 실제 사람의 손글씨 데이터가 모델을 통해 잘 분류가 되는지 테스트 해볼 필요가 있다. 수집한 데이터에 대한 학습이 완료되었으므로, 최종 모델이 실제 사람 손글씨에도 잘 적용이 되는지 평가해야한다.

(1) 실제 손글씨 데이터 수집

0부터 9까지 숫자의 손글씨를 여러 사람으로부터 수집한다.

(2) 최종 모델 테스트

학습용 데이터는 이미지 전처리가 되어있으므로, 수집한 데이터 또한 간단하게 전처리한 후 최종 모델에 테스트 해야한다. 최종 모델이 실제 손글씨 이미지를 잘 분류하는지 확인한다. 50명에게 각각 0부터 9까지 10개의 데이터를 수집하여 총 500개의 데이터로 테스트를 수행한다. 그 결과로 정확도가 90% 이상이 나오면 성능 좋은 모델이라고 판단한다.

Ⅲ. 수행일정

일시	내용
1주차	주제 설정 및 기본계획서 작성
2주차	최종 계획서 작성
3주차	파이썬, 인공지능 기초 학습
4주차	수학적 개념 학습
5주차	모델 구상 및 데이터 수집
6주차	Neural Network 모델 구현
7주차	중간 보고서 작성 및 발표
8주차	모델 정확도 확인 및 분석
9주차	새 모델 및 기법 학습 1
10주차	새 모델 및 기법 학습 2
11주차	최종 모델 구현 및 점검
12주차	모델 정확도 확인 및 분석
13주차	모델 테스트
14주차	기말 보고서 작성 및 검토
15주차	기말 보고서 발표

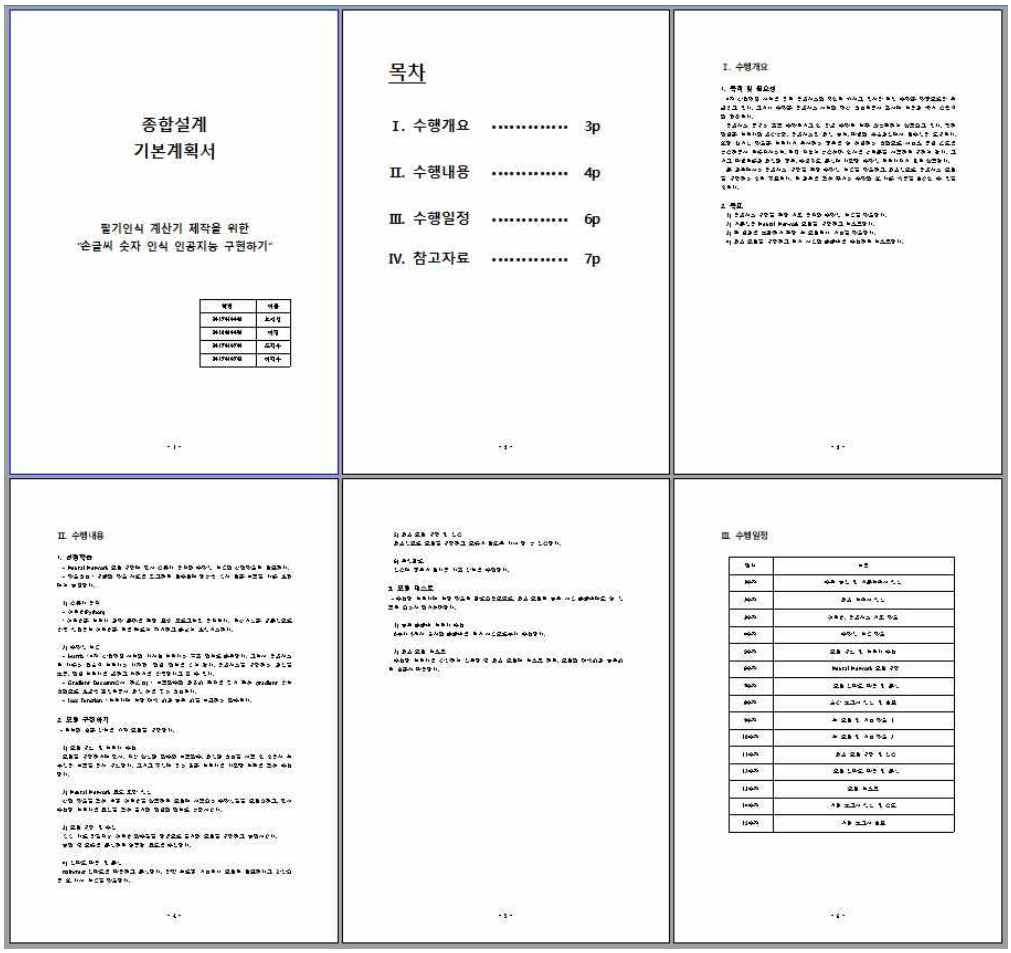
[참고] 주차별 역할 분담표

일시	내용	역할
1주차	주제 설정 및 기본계획서 작성	전체 내용 구상 : 조지수, 이지수 내용검토 : 이정 계획서 작성 : 오서영
2주차	최종 계획서 작성	
3주차	파이썬, 인공지능 기초 학습	파이썬 ppt 제작 및 발표 : 오서영 AI 기초 ppt 제작 및 발표 : 조지수
4주차	수학적 개념 학습	수학적 개념 ppt 제작 및 발표 : 이정, 이지수
5주차	모델 구상 및 데이터 수집	데이터 수집 : 이정, 이지수 프로그램 설치 : 조지수 모델 구상 : 오서영
6주차	Neural Network 모델 구현	코드 자료 수집 : 이정, 이지수 모델 구현 : 오서영, 조지수 오류 확인 및 수정 : 오서영
7주차	중간보고서 작성 및 발표	중간 발표 : 오서영 발표 ppt 제작 : 조지수 중간 보고서 작성 : 이정, 이지수
8주차	모델 정확도 확인 및 분석	정확도 확인 코드 작성 : 오서영, 조지수 정확도 확인 및 분석 : 이정, 이지수
9주차	새 모델 및 기법 학습 1	ppt 제작 및 발표 : 오서영 ppt 제작 및 발표 : 이정
10주차	새 모델 및 기법 학습 2	ppt 제작 및 발표 : 이지수 ppt 제작 및 발표 : 조지수
11주차	최종 모델 구현 및 점검	코드 자료 수집 : 이정, 이지수 모델 구현 : 오서영, 조지수 오류 확인 및 수정 : 오서영
12주차	모델 정확도 확인 및 분석	정확도 확인 코드 작성 : 오서영, 조지수 정확도 확인 및 분석 : 이정, 이지수
13주차	모델 테스트	테스트 자료 수집 : 이정, 이지수 코드 작성 : 오서영, 조지수
14주차	기말 보고서 작성 및 검토	결과 분석 : 오서영 보고서 작성 : 이정, 이지수 발표 : 조지수
15주차	기말 보고서 발표	최종 발표 : 조지수

Ⅳ. 수행내용

1. 1~2주차 : 주제 설정 및 계획서 작성

1주차에서 정한 주제인 "손글씨 숫자 인식 인공지능 구현하기" 에 대한 기본 계획서를 작성하고, 전체적인 이미지 및 내용을 구상하여 최종 보고서에 추가했다. 작성 및 맞춤법 검사는 오서영 학생이 맡았으며, 내용 구상은 조지수, 이지수 학생이, 내용 검토는 이정 학생이 맡았다.


종합설계
기본계획서

필기인식 계산기 제작을 위한
"손글씨 숫자 인식 인공지능 구현하기"

목차

Ⅰ. 수행개요 ····· 3p

Ⅱ. 수행내용 ····· 4p

Ⅲ. 수행일정 ····· 6p

Ⅳ. 참고자료 ····· 7p

Ⅰ. 수행개요

Ⅱ. 수행내용

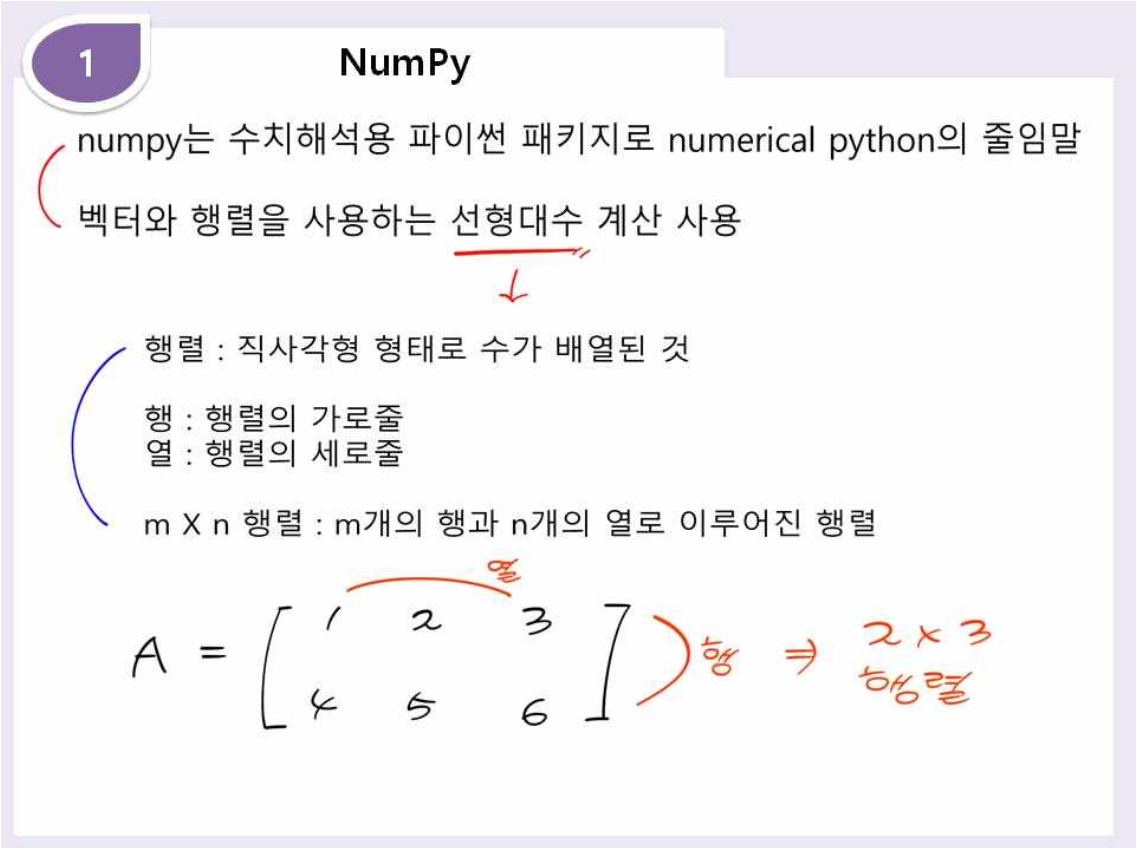
Ⅲ. 수행일정

Ⅳ. 참고자료

< 그림 1. 1주차 기본 계획서>

2. 3주차 : 파이썬, 인공지능 기초 학습

2020년 3월 30일, ZOOM을 이용하여 비대면 회의 및 발표를 진행했다.
기본적인 Neural Network 모델 구현을 위해서 파이썬 문법과 인공지능 기초 학습을 진행했다. 모델 구현에 쓰게 될 파이썬 및 라이브러리(tensorflow) 등을 오서영 학생이, 인공지능, 딥러닝 및 신경망 기본개념을 조지수 학생이 발표했다.



In [1]:
```
import numpy as np
```

In [2]:
```
array = np.array([1, 2, 3])
print(array.size)
print(array.dtype)
print(array[1])
```
배열 원소 개수
원소 타입

```
3
int32
2
```

In [4]:
```
A = np.arange(5)
print(A)
```
0 ~ 4 배열

```
[0 1 2 3 4]
```

In [5]:
```
B = np.zeros((4,3))
print(B)
```
영행렬

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

In [6]:
```
C = np.random.randint(0, 10, (3, 3))
print(C)
```
0 ~ 9 난수

```
[[2 2 7]
 [5 2 7]
 [2 3 2]]
```

In [10]:
```
array1 = np.array([1, 2, 3, 4])
print(array1.shape)
```

```
(4,)
```
← 1차원 배열 (4개의 원소)

In [11]:
```
array2 = array1.reshape((2, 2))
print(array2.shape)
print(array2)
```

```
(2, 2)
[[1 2]
 [3 4]]
```
← 2차원 배열 (2행 2열) $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$

In [12]:
```
a = np.array([[1,2],[3,4]])
b = np.array([[10,20],[30,40]])
```
$a = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ $b = \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix}$

In [19]:
```
c = a + b
c
```
$c = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix} = \begin{bmatrix} 11 & 22 \\ 33 & 44 \end{bmatrix}$

Out[19]:
```
array([[11, 22],
       [33, 44]])
```

In [21]:
```
d = b - a
d
```
a와 b는 동일한 크기의 행렬

Out[21]:
```
array([[ 9, 18],
       [27, 36]])
```

```
In [22]: e = a * b
         e
Out[22]: array([[ 10,  40],
                [ 90, 160]])
```

$$e = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} * \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix} = \begin{bmatrix} 10 & 40 \\ 90 & 160 \end{bmatrix}$$

원소끼리

```
In [23]: f = a @ b
         f
Out[23]: array([[ 70, 100],
                [150, 220]])
```

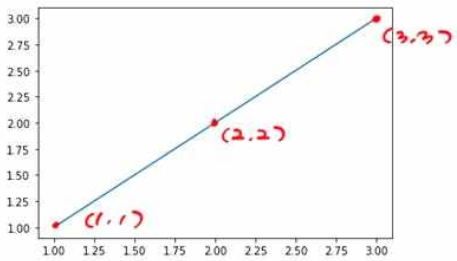
행렬 곱

: 행 성분과 열 성분을 곱하여 더해줌

$$f = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix} = \begin{bmatrix} 1\cdot10+2\cdot30 & 1\cdot20+2\cdot40 \\ 3\cdot10+4\cdot30 & 3\cdot20+4\cdot40 \end{bmatrix}$$

```
In [1]: import matplotlib.pyplot as plt
```

```
In [3]: x = [1,2,3]
        y = [1,2,3]
        plt.plot(x , y)
        plt.show()
```



3 Tensorflow

```
import tensorflow as tf
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
```

```
# 값을 입력할 자리 만들기 (placeholder)
x = tf.placeholder(tf.float32, [None, 784])
```

```
# 수정가능한 텐서 (Variable)
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))
```

```
# Forward Propagation
y = tf.nn.softmax(tf.matmul(x, W) + b)
```

```
# Answer
y_ = tf.placeholder(tf.float32, [None, 10])
```

```
# Cross_Entropy
cross_entropy = tf.reduce_mean(-tf.reduce_sum(y_ * tf.log(y), reduction_indices=[1]))
```

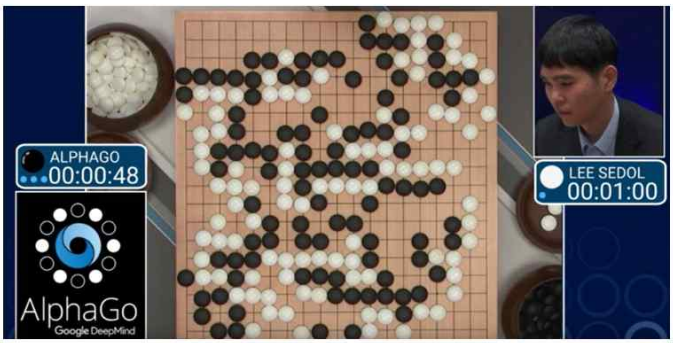
```
# Back Propagation
train_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)
```

```
# 변수 초기화
init = tf.global_variables_initializer()

# 모델 실행
sess = tf.Session()
sess.run(init)

# 학습
for i in range(1000):
    batch_xs, batch_ys = mnist.train.next_batch(100)
    sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})
```

< 그림 2-1. 오서영 학생의 파이썬 설명 일부 ppt >

인공지능

▶ 인간의 지능적인 행동들을 컴퓨터 프로그램으로 실현한 기술



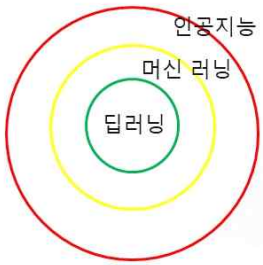
▶ 정보기술의 여러 분야에서 인공지능적 요소를 도입
→ 그 분야의 문제 풀이에 활용하려는 시도가 매우 활발

머신 러닝

▶ 컴퓨터에게 데이터들을 제공하여 학습하게 함으로써 새로운 지식을 얻어내게 하는 분야

▶ ex) 사진데이터 학습을 통한 개와 고양이 구분

▶ 대량의 데이터, 알고리즘을 통해 컴퓨터 그 자체 학습시킴
→ 학습 내용 기반 판단이나 예측 가능

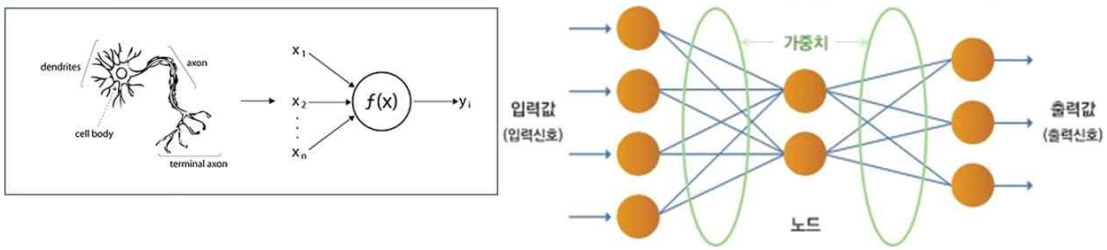


딥러닝

▶ 인공신경망에 기반을 둔 머신 러닝의 한 종류

인공신경망

▶ 사람 또는 동물 두뇌의 신경망에 착안하여 구현된 컴퓨팅 시스템



▶ 어떠한 입력 값이 있으면 입력 값 별로 가중치를 매기고 변환함수(f(x))로 잘 섞어 넣어서 출력 값을 도출해내는 개념

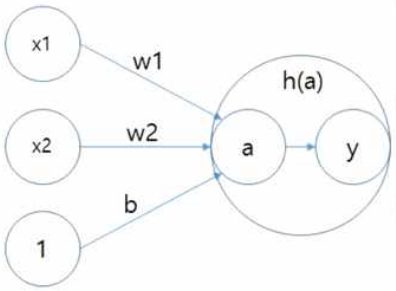
< 그림 2-2. 조지수 학생의 인공지능 설명 일부 ppt >

3. 4주차 : 수학적 개념 학습

2020년 4월 10일, ZOOM을 이용하여 비대면 회의 및 발표를 진행했다.
기본적인 Neural Network 모델 구현을 위해서 모델 내 사용되는 수학적 기법을 학습했다. 손실함수와 경사 하강법을 이지수 학생이, Neural Network 구조와 Softmax 함수 등을 이정 학생이 발표했다.

01. 활성화 함수

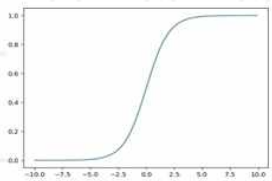
활성화 함수는 출력 값을 활성화를 일으키게 할 것이냐를 결정하고, 그 값을 부여하는 함수이다. 즉, 선형함수를 비선형함수로 바꾸기 위해서 사용한다.



→Input x_1 , x_2 에 가중치 w_1, w_2를 곱하고, bias를 더했을 때, 그 값이 a이고 이 값을 활성화 함수 input 에 넣었을 때의 출력을 y로 하는 것을 의미한다.
여기서 활성화 함수는 h(a)가 된다.

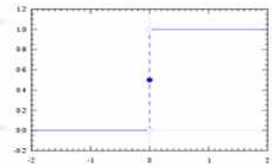
02. 활성화 함수의 종류

- 시그모이드(Sigmoid) 함수 : x 값이 작아질수록 0에 수렴, 커질수록 1에 수렴



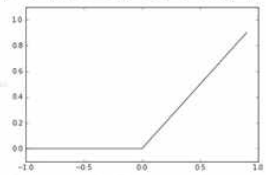
$h(x) = \frac{1}{1+e^{-x}}$ 미분 → h'(x)=h(x)(1-h(x))

- 계단(step) 함수: 특정 값 이하는 0, 이하는 1을 출력하는 함수



$y = \begin{cases} 0 & (b + w_1x_1 + w_2x_2 \le 0) \\ 1 & (b + w_1x_1 + w_2x_2 > 0) \end{cases}$ 활성화 $h(x) = \begin{cases} 0 & (x \le 0) \\ 1 & (x > 0) \end{cases}$

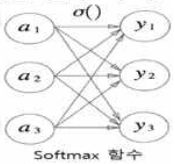
- 렐루(ReLU) 함수 (Recitified Linear Unit): 함수입력이 특정 값을 넘으면 그대로 0을 넘지 않으면 0을 반환 즉, x>0 이면 기울기가 1인 직선이되고 x<0 이면 함수 값이 0.



f(x)= max(0, x)

03. Softmax 함수

Softmax 함수는 분류에 사용되는 함수로 모든 입력신호로 부터 영향을 받는 함수이다.



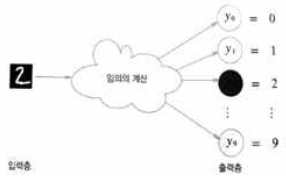
→입력 받은 값을 출력으로 0~1 사이의 값으로 모두 정규화하며
'확률적 해석'을 가능하게 하는 것이 목적이다.

$$p_j = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}$$

- 분자는 지수함수로, 분모는 지수함수의 합으로 구성됨.
- 지수함수로 되어있어 무한대 값이 발생 할 수 있다.

$$= \frac{e^{x_j}}{e^{x_1}+e^{x_2}+\cdots+e^{x_K}} \quad for\ j = 1,\ldots,K$$ 개선식 $$\frac{ce^{x_j}}{c\sum_{k=1}^{k} e^{x_k}} = \frac{e^{x_j+c^*}}{\sum_{k=1}^{k} e^{x_k+c^*}}$$

- 각 값들이 확률로 나오며, 출력 값들의 총합은 항상 "1"이 되는 특성을 가짐.
- 지수함수를 사용하면 가중치를 더 커지게 하는 효과를 얻을 수 있다.
- 다중분류 로지스틱이라는(분류를 여러 개 하는 것)것을 나타낼 때 유용하다.

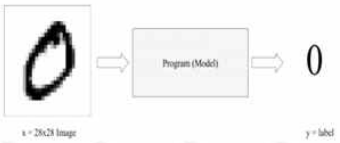


04. MNIST 이미지 인식

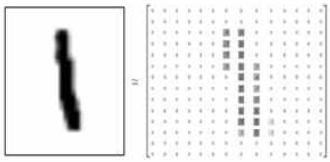
MNIST 함수는 0~9까지 숫자들을 적은 손 글씨로 적은 데이터이다. 즉, 이미지 데이터에는 그 이미지가 어떤 숫자인지를 나타내는 라벨 정보가 함께 들어가 있다.

-MNIST 데이터베이스는 0부터 255 사이의 값을 가지는 흑백 이미지(28×28 사이즈) , 학습데이터 55,000개와 테스트데이터 10,000개 그리고 5,000개의 검증데이터를 포함한다.

예) MNIST 이미지 x가 입력으로 들어오면 그 이미지가 무슨 숫자인지를 해석해서 y로 출력해주는 가장 기본적인 이미지 인식 프로그램



<이미지>



<라벨>

: 이미지가 나타내는 숫자가 어떤 숫자인지를 나타내는 라벨 데이터로 10개의 숫자로 이루어진 1행 행렬이다.

< 그림 3-1. 이정 학생의 NN 구조, Softmax 함수 설명 일부 ppt >

손실함수(loss function)

손실함수란, 예상한 값과 실제 타깃 값의 차이를 함수로 정의한 것

•평균 제곱오차(MSE): 회귀에서 항등함수의 손실함수

$E = \frac{1}{n}\sum_k^n (y_k - t_k)^2$ (y_k=신경망의 출력, t_k= 정답레이블)

• 교차 엔트로피 오차(CEE): 분류에서 소프트맥스의 손실함수

$E = -\sum_k^n t_k \log y_k$ (y_k=신경망의 출력, t_k= 정답레이블)
⇒ 신경망 출력값이 클수록 오차는 0에 가까워 진다.

※ 평균 제곱 오차는 모든 출력층의 값이 계산에 들어간다.
교차 엔트로피 오차는 정답에 해당하는 위치의 값만 계산에 들어간다.

미분(differential), 기울기(gradient)

$$\frac{d}{dx}f(x) = \lim_{\Delta x \to 0}\frac{f(x+\Delta x) - f(x)}{\Delta x}$$

x+h 와 x 사이의 기울기를 얻는 것(전방 차분) 보다 x-h와 x+h 사이의 기울기를 얻는 것(중심 차분, 중앙 차분)이 오차가 더 적다.

•수치 미분: 실제 미분 값이 아니라 실제 값에 대한 근사값
•해석적 미분: 실제로 수식을 미분해 도함수를 구하는 것

$$\nabla f = (\frac{\partial f}{\partial x_1}, \cdots, \frac{\partial f}{\partial x_n})$$

기울기는 벡터 미적분학에서 스칼라장의 최대 증가율을 나타내는 벡터를 뜻한다.
화살표 방향: 증가율이 최대가 되는 방향
화살표 크기: 증가율이 최대일 때 증가율의 크기
기울기 음수: 함수의 출력을 가장 줄이는 방향

경사법(gradient method)

경사법이란, 기울기를 이용해 손실 함수의 최솟값을 찾는 방법

주의할점: 기울기가 가리키는 곳을 따라가도 함수의 최솟값(global minimum)이 있으리라는 보장은 없다. 극댓값이나 안장점도 기울기가 0이다. 극솟값이어도 함수의 최솟값이라고 장담할 수 는 없다.

$$x_i = x_i - \eta\frac{\partial f}{\partial x_i}$$ (η=학습률: 매개변수 값을 갱신하는 양)

※학습률이 너무 크거나 작으면 손실 함수가 최소가 되는 위치를 찾아갈 수 없다. 사람이 경험적으로 직접 설정 해야하며 이런 매개변수를 하이퍼파라미터(hyper parameter)라고 한다.

신경망의 기울기

F는 손실 함수(L), x는 가중치(W)

$W = \begin{bmatrix} w_{11} & w_{21} & w_{31} \\ w_{12} & w_{22} & w_{32} \end{bmatrix}$ 일때,

$$\nabla L = \frac{\partial L}{\partial W} = \begin{bmatrix} \frac{\partial L}{\partial w_{11}} & \frac{\partial L}{\partial w_{12}} & \frac{\partial L}{\partial w_{13}} \\ \frac{\partial L}{\partial w_{21}} & \frac{\partial L}{\partial w_{22}} & \frac{\partial L}{\partial w_{23}} \end{bmatrix}$$

수치 미분을 통한 기울기 계산의 문제점

1. 정확한 미분값이 아니라, 근사값이다.
2. 속도가 너무 느리다. $w_1, \cdots, w_n$까지 있으면, 기울기를 계산하기 위해 이 각각을 대상으로 $\frac{loss(x+h)-loss(x-h)}{2h}$를 구해야 한다.

Sol1. 해석적 미분을 사용한다.
Sol2. 중복 계산을 피하기 위해 오차역전파를 사용한다.

오차역전파법(backpropagation)

역전파: 역방향으로 해당 함수의 국소적 미분을 곱해 나가는 것

※핵심: 국소적 계산, 연쇄법칙

*국소적 계산: 현재 계산이 이전 계산이나 다음 계산의 영향을 받지 않는다.

연쇄법칙(chain rule)

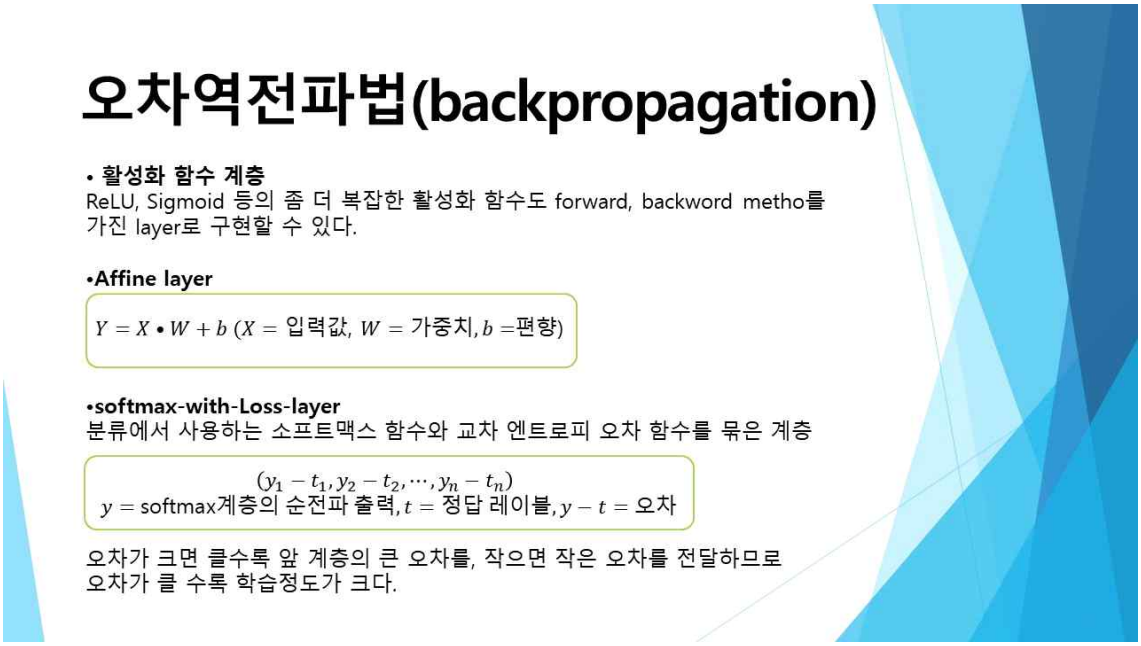
$$h\left(g(f(x))\right)' = h'(g(f(x))) \cdot g'(f(x)) \cdot f'(x)$$

• 덧셈 노드의 역전파

z=x+y를 미분해보면 x에 대해서 미분하든, y에 대해서 미분하든 1이 나오기 때문에 입력을 그대로 흘려 보낸다고 생각하면 된다.

•곱셈 노드의 역전파

z=xy를 미분해보면 x에 대해 미분하면 y, y에 대해 미분하면 x가 나오기 때문에 입력을 서로 바꾼 값을 역전파하게 된다.



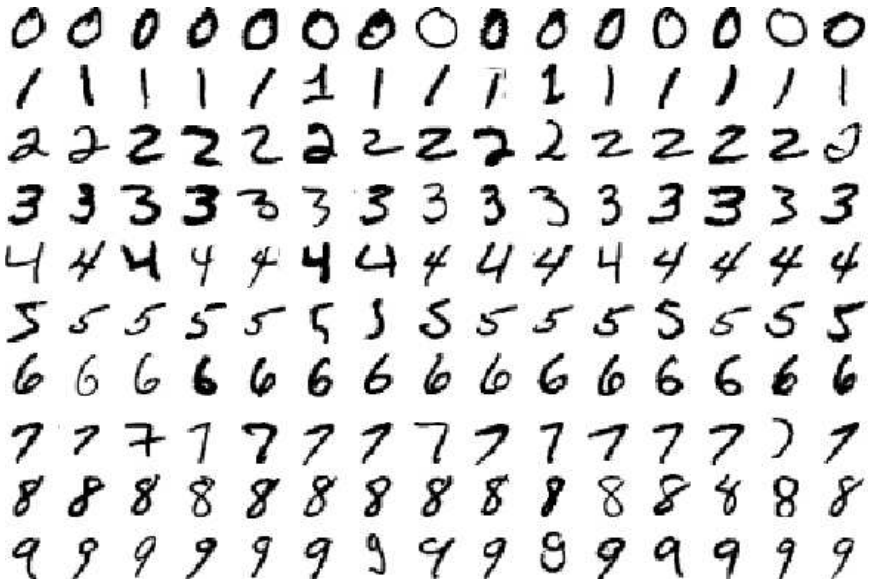
< 그림 3-2. 이지수 학생의 손실함수, 경사법 설명 일부 ppt >

4. 5주차 - 모델 구상 및 데이터 수집

2020년 4월 13일, ZOOM을 이용하여 비대면 회의를 진행했다.

1) 데이터 수집

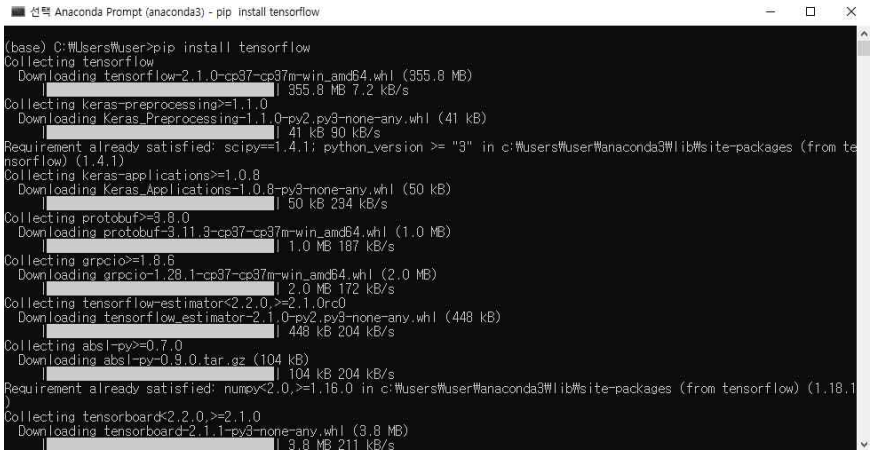
우리가 만들고자 하는 것은 0부터 9까지의 손글씨 숫자를 분류해주는 모델이므로, Yann LeCun의 웹사이트에 호스팅되어 있는 'MNIST 데이터셋'을 사용하기로 했다. 다운로드한 데이터는 55,000개의 학습 데이터, 10,000개의 테스트 데이터, 그리고 5,000개의 검증 데이터이다. 이지수와 이정 학생이 각각 데이터 조사와 수집을 맡았다.



< 그림 4-1. MNIST 데이터셋 >

2) 프로그램 설치

파이썬과 라이브러리를 설치하기 위해 'Anaconda' 프로그램에 내장되어있는 'Anaconda Prompt' 와 'Pip'을 사용하여 텐서플로우를 설치했다. 조지수 학생이 파이썬 및 라이브러리 설치 방법을 ZOOM을 통해 나머지 조원들에게 발표했다.



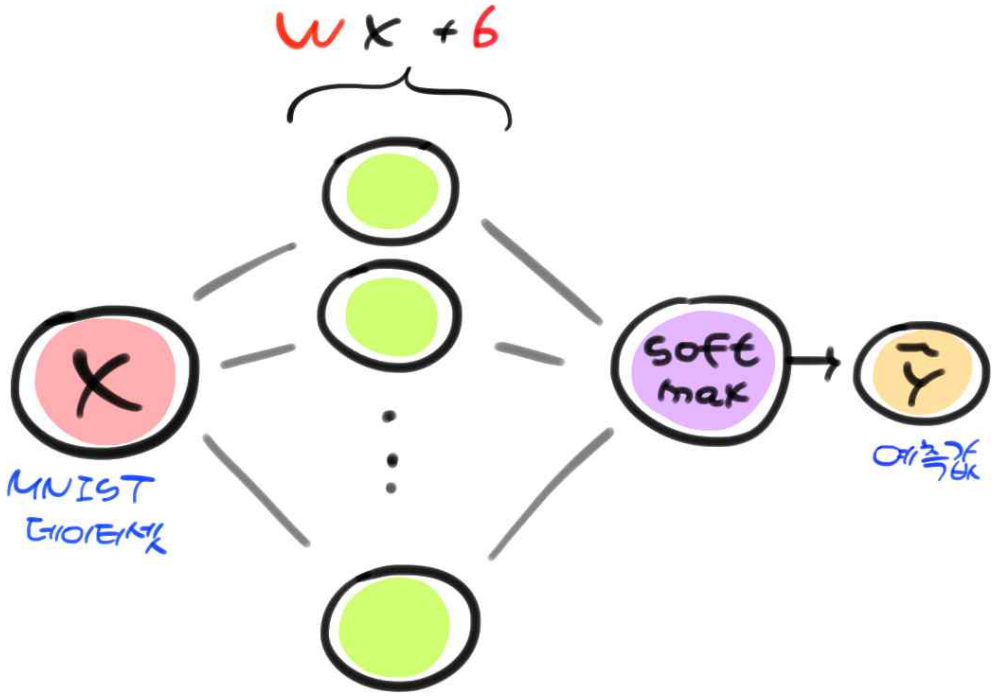
< 그림 4-2. Tensorflow 설치 >

3) 모델 구상

기본적인 Single-Layer Neural Network 모델을 만들기로 정했다. 10가지 종류를 분류해야

하므로 다중분류 회귀인 Softmax 함수를 활성화 함수로 사용할 것이다.

모델 구조는 아래와 같다. 모델 구상 및 스케치는 오서영 학생이 맡았다.



< 그림 4-3. 모델 구조 스케치 >

5. 6주차 - Neural Network 모델 구현

2020년 4월 23일, ZOOM을 이용하여 비대면 회의를 진행했다.

본격적인 모델을 구현하기 전에 이지수, 이정 학생이 참고할만한 코드를 수집했다. 'CodeOnWeb'에 작성된 '머신러닝 초보를 위한 MNIST'를 참고하여 모델을 구현하기로 했다.모델 구현을 위한 코드 작성은 오서영, 조지수 학생이 맡았다.

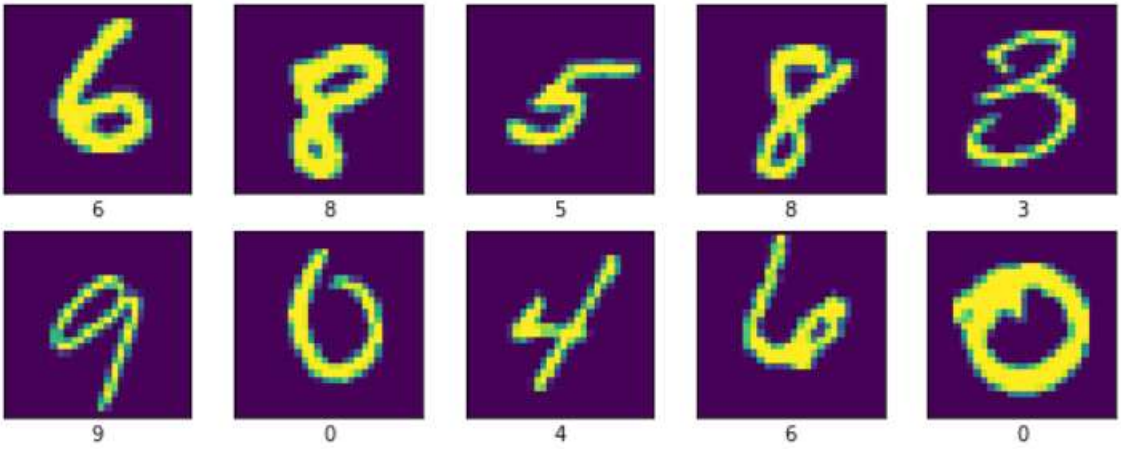
1) 패키지 불러오기

1. Import Packages

```
import input_data
import numpy as np
import matplotlib.pylab as plt
import tensorflow as tf
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
```

데이터를 불러오기 위한 input_data, 행렬 계산을 위한 numpy, 시각화를 위한 matplotlib, 모델 구현을 위한 tensorflow 패키지를 불러왔다.

2) 데이터 확인하기



matplotlib 패키지를 활용하여 이미지와 실제 정답 라벨을 확인했다.

3) 데이터셋 만들기

```
mnist = input_data.read_data_sets("./samples/MNIST_data/", one_hot=True)
```

```
Extracting ./samples/MNIST_data/train-images-idx3-ubyte.gz
Extracting ./samples/MNIST_data/train-labels-idx1-ubyte.gz
Extracting ./samples/MNIST_data/t10k-images-idx3-ubyte.gz
Extracting ./samples/MNIST_data/t10k-labels-idx1-ubyte.gz
```

```
print("the number of train examples :" , mnist.train.num_examples)
print("the number of test examples :" , mnist.test.num_examples)
```

```
the number of train examples : 55000
the number of test examples : 10000
```

55000개의 훈련데이터, 1000개의 테스트데이터를 불러와 데이터 셋을 만들었다.

4) 모델 구현하기

```
# Create placeholders
x = tf.placeholder(tf.float32, [None, 784])

# Initialize parameters
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))
```

```
# Forward Propagation
y = tf.nn.softmax(tf.matmul(x, W) + b)

y_ = tf.placeholder(tf.float32, [None, 10])

# Compute cost
cross_entropy = - tf.reduce_sum(y_*tf.log(y))

# Backward Propagation
learning_rate = 0.01
train_step = tf.train.GradientDescentOptimizer(learning_rate).minimize(cross_entropy)
```

placeholder : 행렬크기를 지정해서 빈공간을 만드는 작업이다. 예로 y_는 실제라벨 값이 들어올 빈공간이므로 (none, 10)으로 행렬 크기를 지정한다. 나중에 none에는 데이터의 개수가 들어온다.

y : W*X + b를 계산해서 softmax함수에 넣어 계산하는 일이다.

cross_entropy : 손실함수를 계산하는 일이다.

train_step : 경사하강법(gradient descent)을 0.01의 학습률을 사용하여 앞에 지정한 cross_entropy 손실함수를 최소화 시키는데 사용한다.

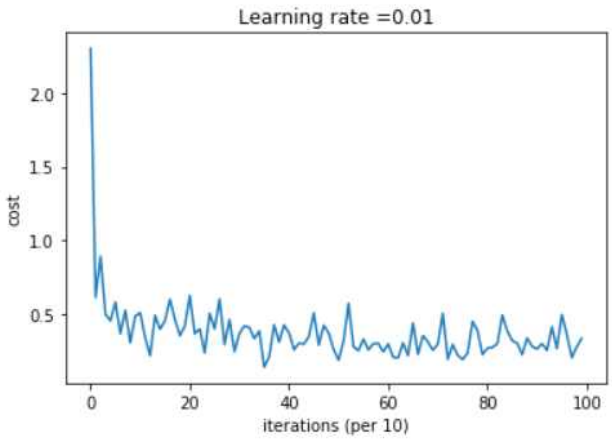
```
# Initialize all the variables
init = tf.global_variables_initializer()

# Start the session to compute the tensorflow graph
sess = tf.Session()
sess.run(init)
```

위에서 지정한 변수들의 초기조건을 세팅하고, 위에서 지정한 일을 수행한다.

```
# Do the training loop - Stochastic training
batch_size = 100
epoch_cost = 0
costs = []

for i in range(1000):
    batch_xs, batch_ys = mnist.train.next_batch(batch_size)    # 100 random data
    _ , minibatch_cost = sess.run([train_step, cross_entropy], feed_dict={x: batch_xs, y_: batch_ys})
    epoch_cost = minibatch_cost / batch_size
```

55000개의 데이터 중 랜덤으로 100개를 뽑아 학습하는데, 이를 1000번(epoch) 반복한다.

```
Cost after epoch 0: 2.302585
Cost after epoch 100: 0.508410
Cost after epoch 200: 0.625304
Cost after epoch 300: 0.366150
Cost after epoch 400: 0.370434
Cost after epoch 500: 0.184827
Cost after epoch 600: 0.295910
Cost after epoch 700: 0.294585
Cost after epoch 800: 0.267045
Cost after epoch 900: 0.260601
```



코드를 작성하고 실행한 결과, 비용(cost)이 100번의 epoch마다 잘 줄어드는 것을 확인할 수 있다.

6. 7주차 - 중간보고서 작성 및 발표

비대면 강의 연장으로 중간보고서 작성만 진행하고, 보고서 첨삭으로 진행했다.

7. 8주차 - 모델 정확도 확인 및 분석

2020년 5월 6일, ZOOM을 이용하여 비대면 회의를 진행했다.

완성된 모델이 손글씨 데이터를 얼마나 잘 분류하는지 확인하기 위해 정확도를 계산하고, 잘못 예측된 데이터를 확인했다. 오서영, 조지수 학생이 코드를 작성하고 이지수, 이정학생이 결과를 분석했다.

6. Check wrong prediction

```
w = []
for r in range(1000):
    if sess.run(tf.argmax(mnist.test.labels[r:r+1], 1)) != sess.run (tf.argmax(y, 1), feed_dict={x: mnist.test.images[r:r+1]}):
        w.append(r)
```

```
print("wrong label : " ,w)
```

```
wrong label :  [8, 33, 63, 80, 92, 124, 149, 195, 211, 233, 241, 245, 247, 259, 290, 300, 307, 320, 321, 340, 352, 359, 362, 381, 406,
412, 435, 445, 448, 449, 464, 468, 469, 478, 479, 502, 507, 511, 528, 530, 531, 536, 542, 543, 550, 551, 565, 569, 578, 582, 583, 588,
591, 613, 619, 624, 627, 628, 629, 658, 659, 684, 689, 691, 692, 707, 717, 720, 728, 740, 741, 760, 766, 791, 810, 839, 844, 857, 866,
877, 881, 890, 898, 924, 938, 939, 947, 950, 951, 955, 956, 959, 965, 982]
```

'wrong label'은 몇 번째 데이터가 잘못 예측된 데이터인지 저장한 리스트이다.
8번째, 33번째, 63번째, ... 데이터가 잘못 예측된 데이터이다.

5. Calculate Accuracy

```
# Validation
# Calculate the correct predictions
correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_,1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

# Accuracy
print("Train Accuracy : " , sess.run(accuracy, feed_dict={x: mnist.train.images, y_: mnist.train.labels}))
print("Test Accuracy : " , sess.run(accuracy, feed_dict={x: mnist.test.images, y_: mnist.test.labels}))
```

```
Train Accuracy :  0.91096365
Test Accuracy :  0.9138
```

훈련데이터에 대한 정확도는 약 91%, 테스트데이터에 대한 정확도 또한 약 91%이다.
정확도는 (잘 예측된 데이터의 개수) * 100 / (전체 데이터의 개수) 으로 계산된다.
우리가 설정한 목표 정확도는 97% 이므로 새로운 기법이나 모델을 통해 정확도를 올릴 필요가 있다고 생각한다.

```
wrong_pred = []
for i in range(len(w)) :
    wrong_pred.append(sess.run (tf.argmax(y, 1), feed_dict={x: mnist.test.images[w[i]:w[i]+1]}))
```

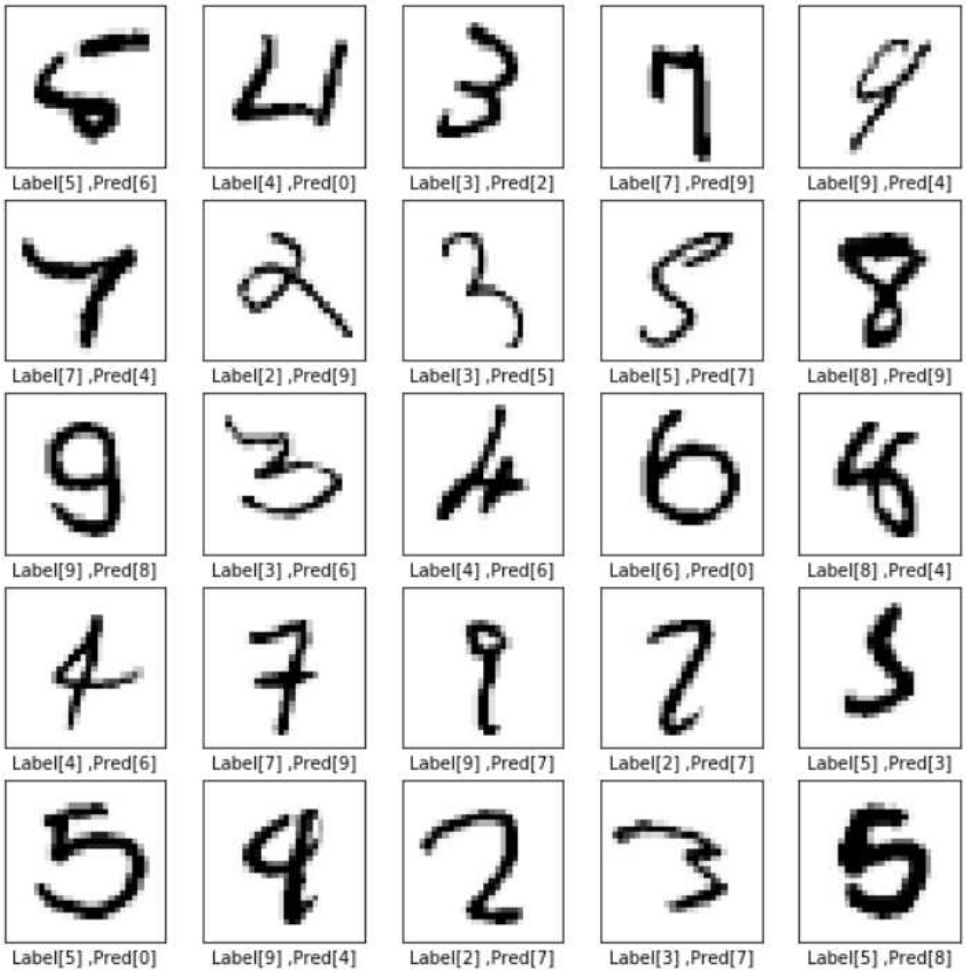
```
wrong_label = []
for i in range(len(w)) :
    wrong_label.append(sess.run(tf.argmax(mnist.test.labels[w[i]:w[i]+1], 1)))
```

wrong_pred : 반복문을 통해 틀린 예측을 모은 행렬이다.

wrong_label : 반복문을 통해 예측이 틀린 이미지의 정답라벨을 모은 행렬이다.

```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(mnist.test.images[w[i]].reshape(28, 28), cmap='Greys', interpolation='nearest')
    plt.xlabel("Label" + str(wrong_label[i]) + " ,Pred" +str(wrong_pred[i]))
plt.show()
```

위에서 만든 wrong_pred와 wrong_label을 25개만 짝지어서 출력했다.



위 그림은 정답과 예측이 다른 이미지들만 나타낸 것이다.
Label은 실제 정답 값, Pred는 모델이 예측한 값이다.
1행 1열, 2행 2열, 4행 4열 처럼 사람이 봐도 잘 구별할 수 없는 데이터까지
학습하려면, 새로운 모델이나 기법을 적용 할 필요가 있다.

8. 9주차 : 새 모델 및 기법 학습 1

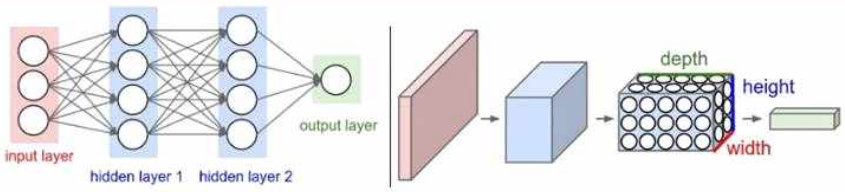
2020년 5월 13일, ZOOM을 이용하여 비대면 회의 및 발표를 진행했다.
더 나은 모델 구현을 위해 새 모델을 찾던 도중, 'CNN(Convolutional Neural Network)'이 이미지 분류에 많이 쓰이는 딥러닝 모델 이라는 사실을 알아냈다. 내용이 쉽지 않았기에, Convolution의 개념과 CNN의 구조 등 여러 내용을 나눠 각자 발표하여 학습했다.
CNN의 용어와 구조 등을 오서영 학생이, CNN 출력 데이터 크기 산정 및 구성을 이정 학생이 발표했다.

FC layer의 문제점

FC : 1차원 데이터만 입력 받을 수 있기 때문에,
3차원 데이터를 평탄화 해서 입력해야 한다.
여기서 3차원 데이터의 공간적 정보가 소실된다는 문제가 발생한다

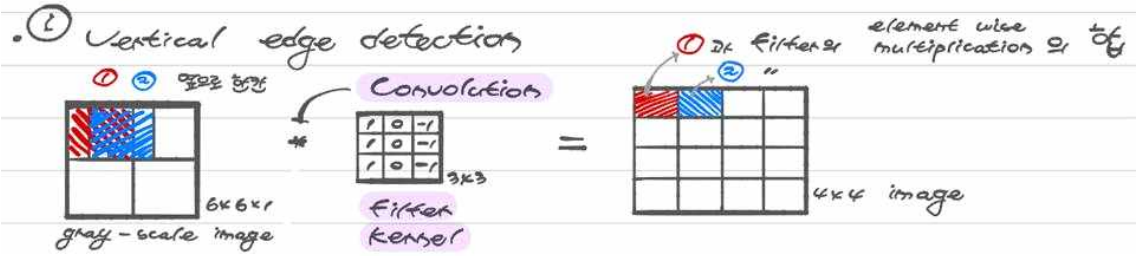
MNIST 이미지는 형상이 (1채널, 가로 28픽셀, 세로 28픽셀)인 3차원
이 3차원 데이터에는 공간적으로 가까운 픽셀은 값이 비슷하다거나,
RGB의 각 채널은 서로 밀접하게 관련되어 있다든가 하는
공간적 정보가 들어있다.

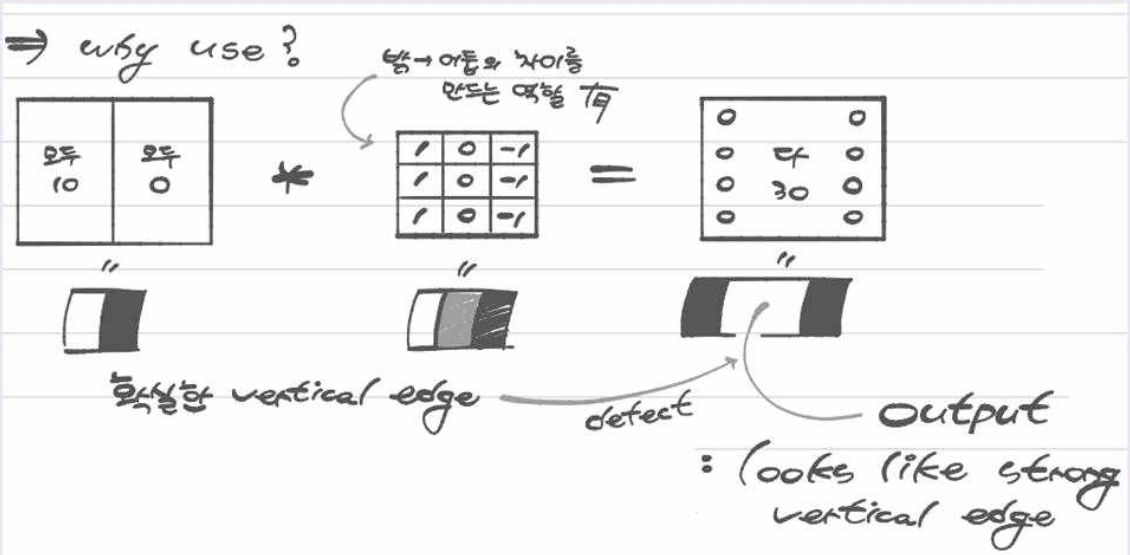
이를 Affine layer에 입력할 때,(1, 784)의 1차원 데이터로
평탄화 해서 넘기기 때문에 이런 공간적 정보가 소실된다.
반면 CONV layer는 형상을 유지한다. 입/출력 모두 3차원
데이터로 처리하기 때문에 공간적 정보를 유지할 수 있다.



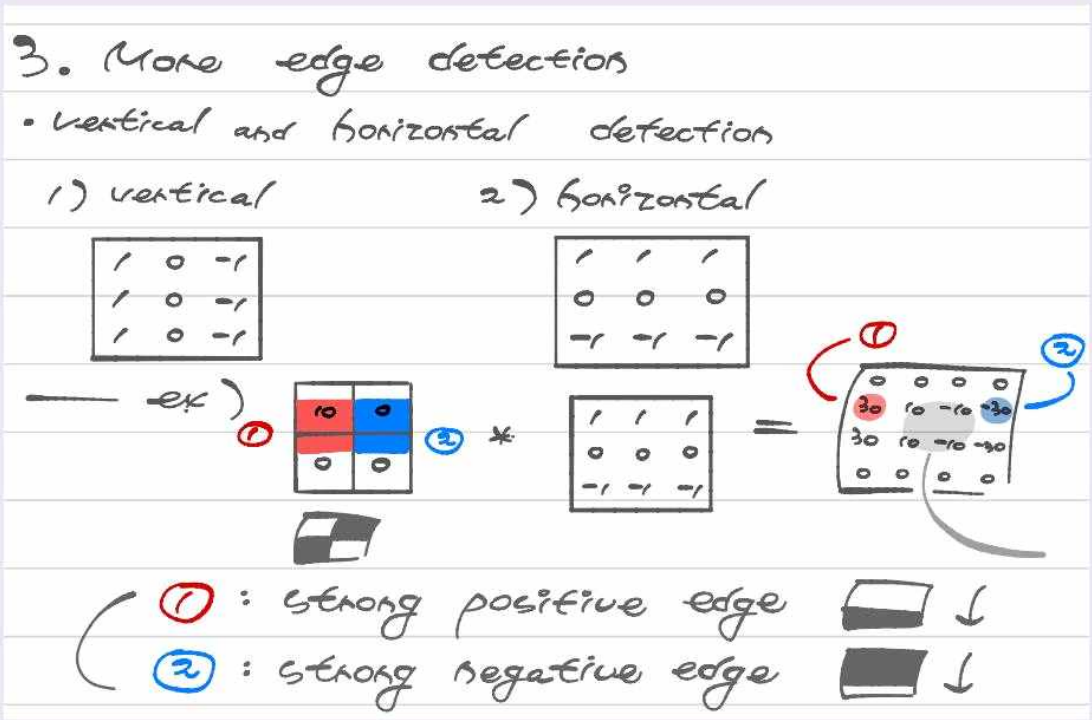
1 Convolution (합성곱)

2. Edge detection example
- Computer Vision problem
— first, detect edges in image
→ how ?) ① detect vertical edges
② detect horizontal edges

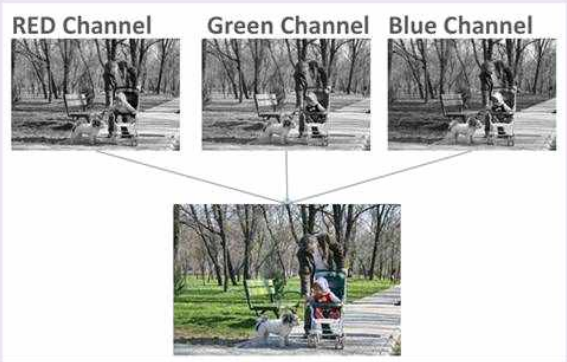


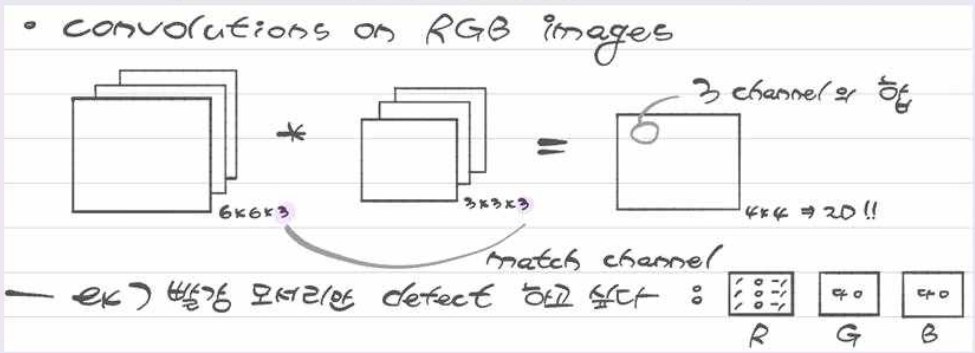


필터 : 이미지의 특징을 찾아내기 위한 공용 파라미터



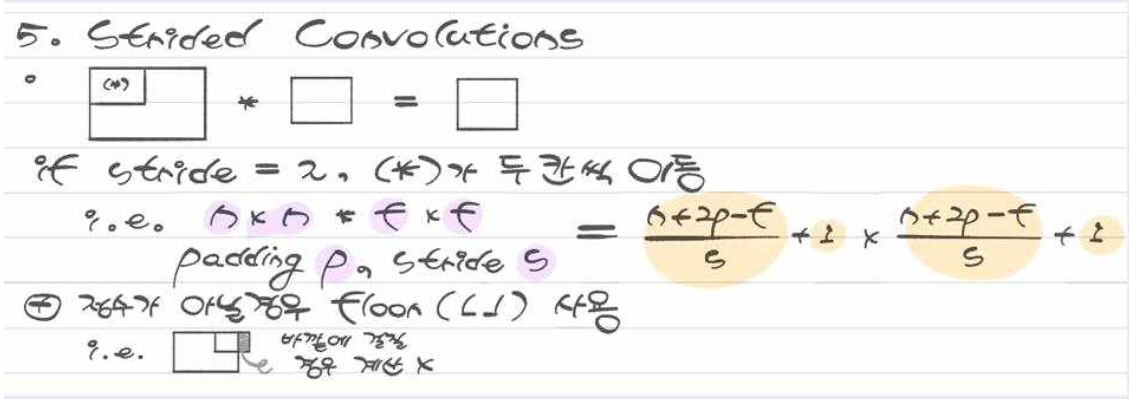
2 Channel (채널)

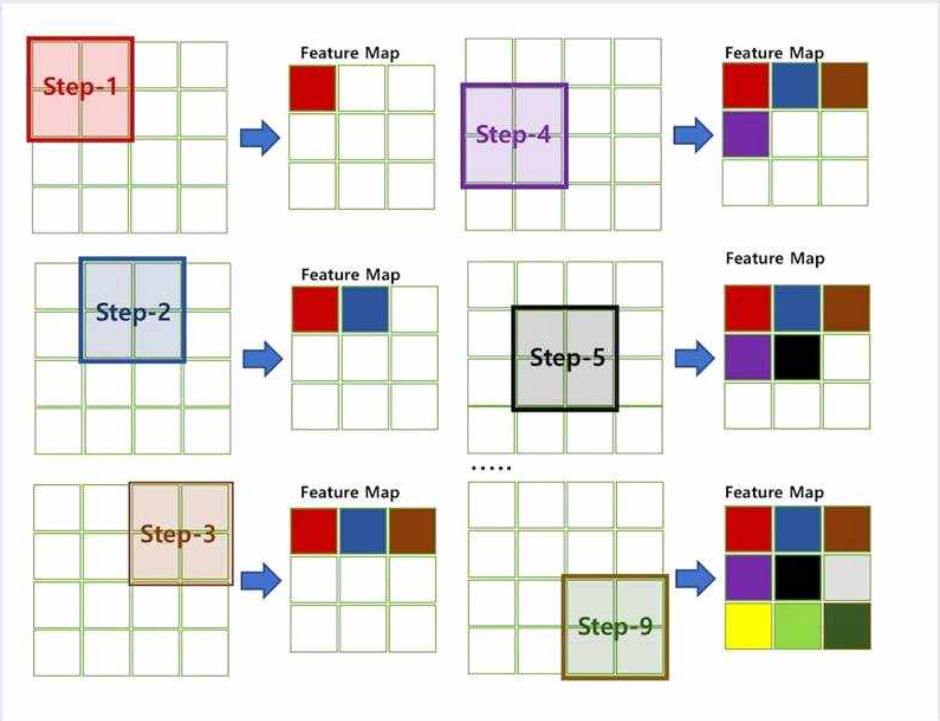




3 Stride

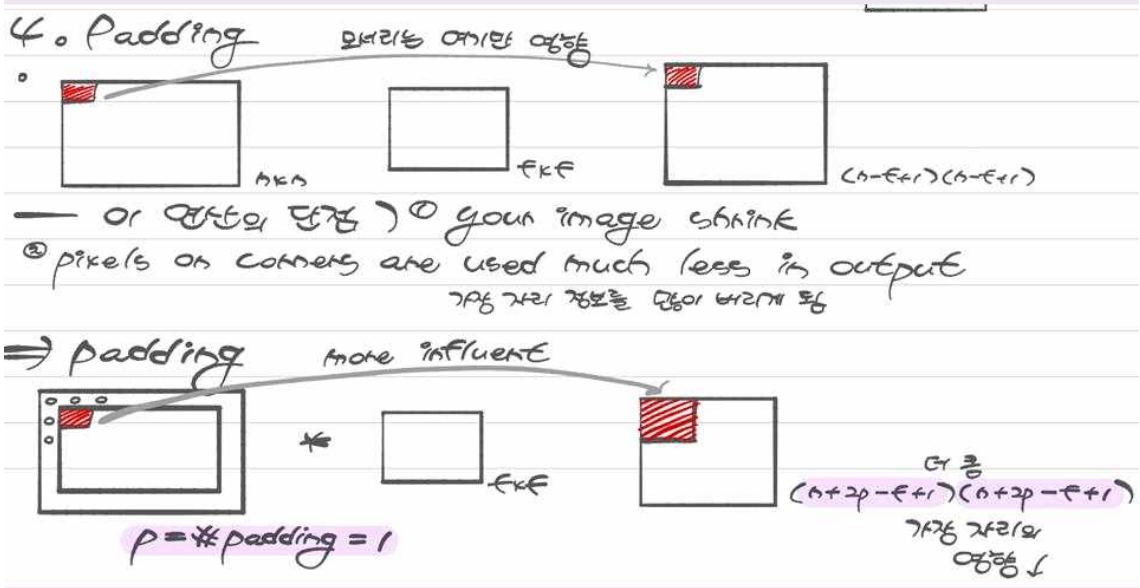
Stride : 지정된 간격으로 필터를 순회하는 간격





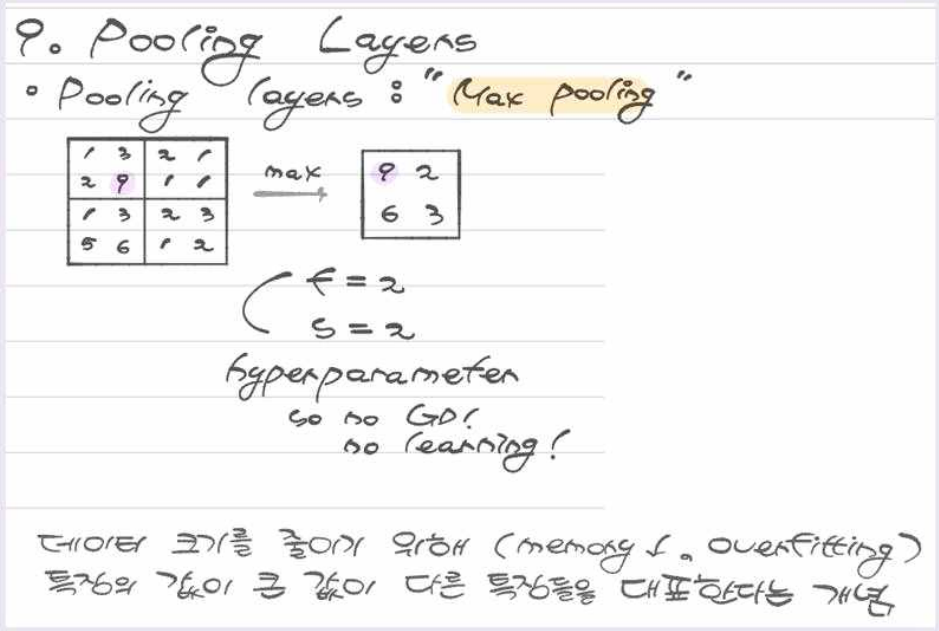
4 Padding

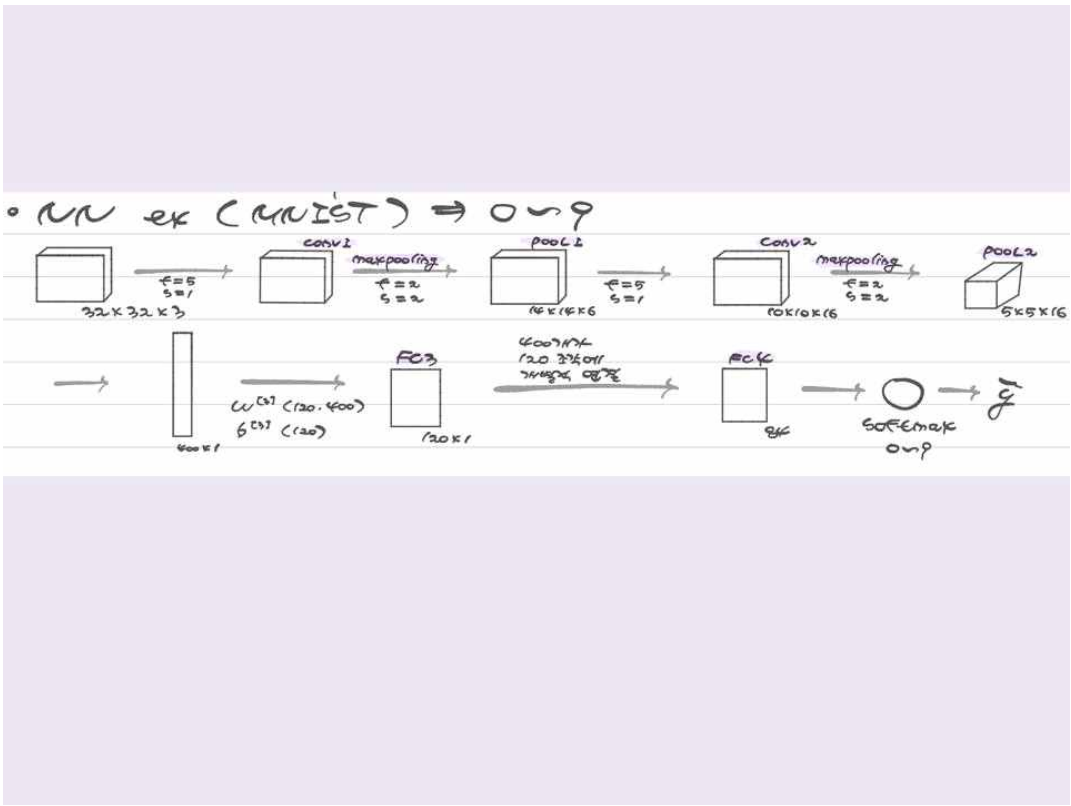
패딩 : 입력 데이터의 외각에 지정된 픽셀만큼 특정 값을 채워 넣는 것



4 Pooling

풀링 : 데이터의 크기를 줄이거나 특정 데이터를 강조하는 용도





< 그림 8-1. 오서영 학생의 CNN의 용어와 구조 설명 일부 ppt >

1. Convolution layer 출력데이터 크기 산정

► 입력 데이터에 대한 필터의 크기와 Stride 크기에 따라서 Feature Map 크기가 결정 된다.

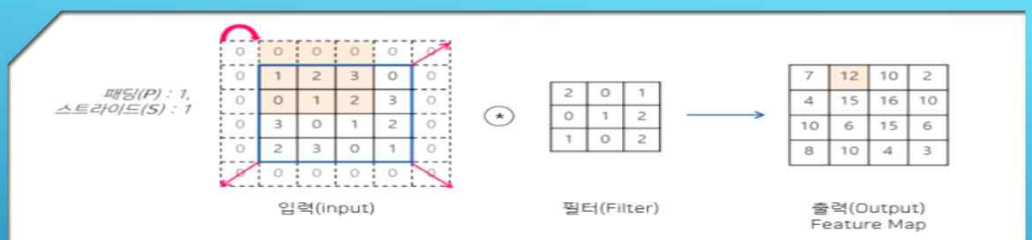
Output Height = OH = $\frac{(H+2P-FH)}{S}+1$

Output Weight = OW= $\frac{(W+2P-FW)}{S}+1$

* H: 입력된 데이터 높이, W: 입력된 데이터 폭, FH: 필터 높이, FW: 필터 폭, S: Stride 크기, P: 패딩 크기

· 입력결과는 자연수가 되어야 한다.

· Convolution 레이어 다음에 Pooling 레이어가 온다면, Feature Map의 행과 열 크기는 Pooling 크기의 배수여야 한다.
ex) Pooling 사이즈가 (3,3)이라면 위 식의 결과는 자연수이고 3의 배수여야 한다.



OH= $\frac{4+2\times1-3}{1}+1=4$

OW= $\frac{4+2\times1-3}{1}+1=4$

· 출력크기가 정수가 아닌 경우에는 에러가 발생할 수 있는데, 보통 딥러닝에서는 반올림을 통해 에러없이 작동한다.

2. Pooling layer 출력데이터 크기 산정

► Pooling 레이어의 출력 데이터의 크기는 행과 열의 크기를 Pooling 사이즈로 나눈 몫이다.

$$\text{Output RowSize} = \frac{InputRow\ Size}{Pooling\ Size}$$

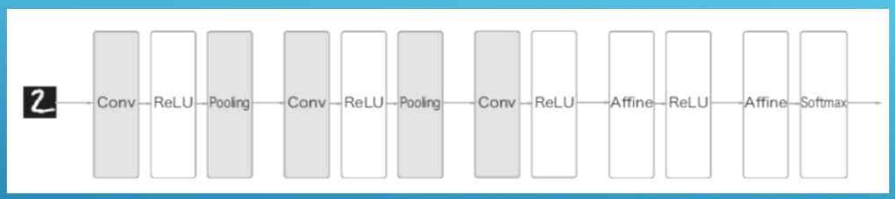
$$\text{Output ColumnSize} = \frac{Input\ Column\ Size}{Pooling\ Size}$$

· Pooling 크기가 (2,2)라면 출력 데이터 크기는 입력 데이터의 행과 열을 2로 나눈 몫이다.
ex) 입력데이터(8,8)이며,Pooling 사이즈(2,2)라면 출력 데이터 크기는 (4,4)이다.

· Pooling 사이즈를 Stride와 같은 크기로 만들어서, 모든 요소가 한번씩 Pooling되도록 만든다.

3. CNN 구성

► CNN은 Convolution Layer 와 Max Pooling 레이어를 반복적으로 stack을 쌓는 특징 추출(Feature Extraction) 부분과 Fully Connected Layer를 구성하고, 마지막 출력층에 softmax를 적용한 분류 부분으로 나뉜다.



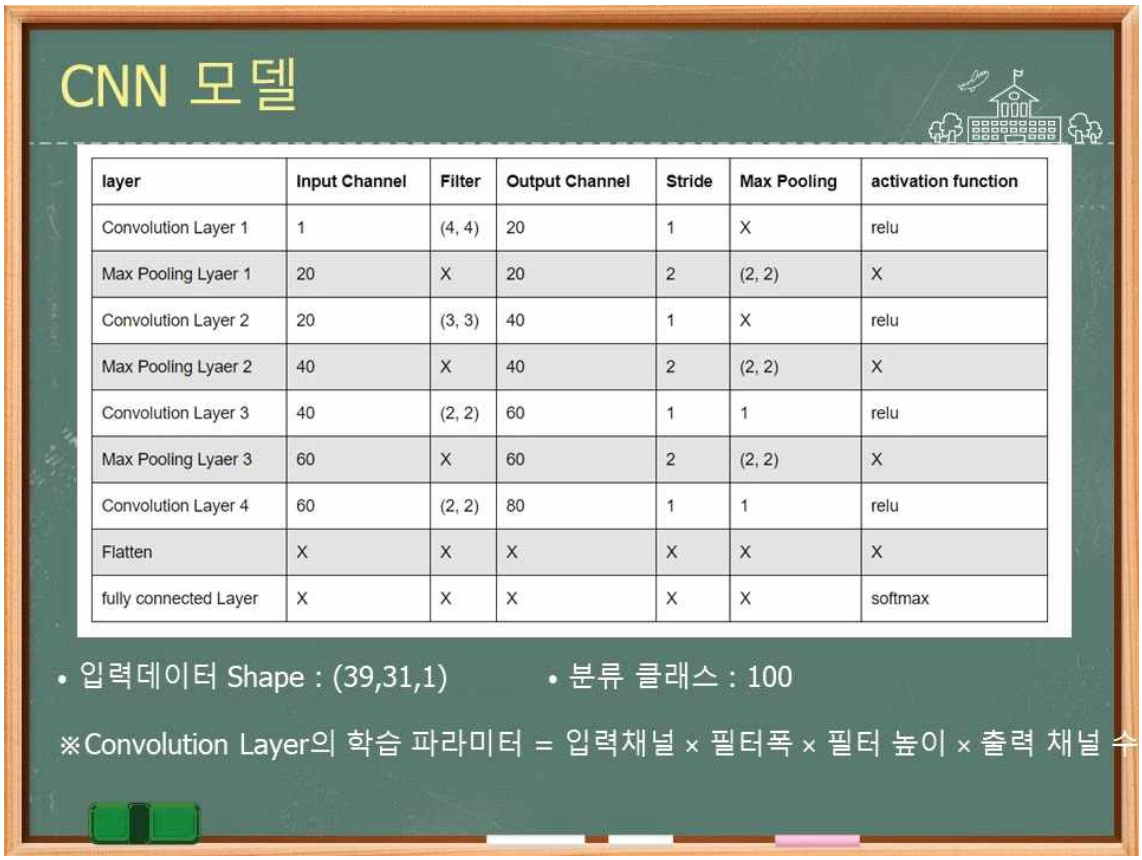
► CNN의 구조 : 합성곱 계층과 풀링 계층이 추가됨

· Conv-ReLU-(Pooling) 흐름, 풀링 계층은 생략되기도 함
· 출력에 가까운 층에서는 Affine-ReLU 구성을 사용할 수 있음
· 마지막 출력 계층에서는 Affine-Softmax 조합을 그대로 사용

< 그림 8-2. 이정 학생의 CNN 출력 데이터 크기 산정 및 구성 설명 일부 ppt >

9. 10주차 : 새 모델 및 기법 학습 2

2020년 5월 20일, ZOOM을 이용하여 비대면 회의 및 발표를 진행했다.
CNN 입출력 파라미터 계산을 이지수 학생이, CNN과 FC Neural Network 파라미터 비교를 조지수 학생이 발표했다.



CNN 모델

layer	Input Channel	Filter	Output Channel	Stride	Max Pooling	activation function
Convolution Layer 1	1	(4, 4)	20	1	X	relu
Max Pooling Lyaer 1	20	X	20	2	(2, 2)	X
Convolution Layer 2	20	(3, 3)	40	1	X	relu
Max Pooling Lyaer 2	40	X	40	2	(2, 2)	X
Convolution Layer 3	40	(2, 2)	60	1	1	relu
Max Pooling Lyaer 3	60	X	60	2	(2, 2)	X
Convolution Layer 4	60	(2, 2)	80	1	1	relu
Flatten	X	X	X	X	X	X
fully connected Layer	X	X	X	X	X	softmax

• 입력데이터 Shape : (39,31,1) • 분류 클래스 : 100

※Convolution Layer의 학습 파라미터 = 입력채널 x 필터폭 x 필터 높이 x 출력 채널 수

4.1 Layer 1의 Shape와 파라미터

4.1.1 Convolution Layer 1

- 입력 데이터 shape : (39,31,1)
- 입력 채널 : 1
- 필터 : (4,4)
- 출력 채널 : 20
- Stride : 1

출력 데이터(Activation Map)의 Shape

Row size $= \frac{N-F}{strid} + 1 = \frac{39-4}{1} + 1 = 36$

Column size $= \frac{N-F}{strid} + 1 = \frac{31-4}{1} + 1 = 28$

- 입력채널 : 1
- 출력 데이터(Activation Map) Shape : (36,28,20)
- 학습 파라미터 : 320개(1×4×4×20)

4.1 Layer 1의 Shape와 파라미터

4.1.2 Max Pooling Layer 1

- 입력 데이터 Shape : (36,28,20)
- Max Pooling의 크기 : (2,2)

출력 데이터(Activation Map)의 Shape

Row size $= \frac{36}{2} = 18$

Column size $= \frac{28}{2} = 14$

- 입력 채널 : 20
- 출력 데이터 Shape : (18,14,20)
- 학습 파라미터 : 0

4.2 Layer 2의 Shape과 파라미터

4.2.1 Convolution Layer 2

- 입력 데이터 shape : (18,14,20)
- 입력 채널 : 20
- 필터 : (3,3,40)
- 출력 채널 : 40
- Stride : 1

출력 데이터(Activation Map)의 Shape

Row size = $\frac{N-F}{strid}+1=\frac{18-3}{1}+1=16$

Column size= $\frac{N-F}{strid}+1=\frac{14-3}{1}+1=12$

- 입력채널 : 20
- 출력 데이터(Activation Map) Shape : (16,12,40)
- 학습 파라미터 : 7,200개(20×3×3×40)

4.2 Layer 2의 Shape과 파라미터

4.2.1 Convolution Layer 2

- 입력 데이터 shape : (18,14,20)
- 입력 채널 : 20
- 필터 : (3,3,40)
- 출력 채널 : 40
- Stride : 1

출력 데이터(Activation Map)의 Shape

Row size = $\frac{N-F}{strid}+1=\frac{18-3}{1}+1=16$

Column size= $\frac{N-F}{strid}+1=\frac{14-3}{1}+1=12$

- 입력채널 : 20
- 출력 데이터(Activation Map) Shape : (16,12,40)
- 학습 파라미터 : 7,200개(20×3×3×40)

4.2 Layer 2의 Shape와 파라미터

4.2.2 Max Pooling Layer 2

- 입력 데이터 Shape : (16,12,40)
- Max Pooling의 크기 : (2,2)

출력 데이터(Activation Map)의 Shape

Row size $= \frac{16}{2} = 8$

Column size $= \frac{12}{2} = 6$

- 입력 채널 : 40
- 출력 데이터 Shape : (8,6,40)
- 학습 파라미터 : 0

4.3 Layer 3의 Shape과 파라미터

4.3.1 Convolution Layer 3

- 입력 데이터 shape : (8,6,40)
- 입력 채널 : 40
- 필터 : (3,3)
- 출력 채널 : 60
- Stride : 1

출력 데이터(Activation Map)의 Shape

Row size $= \frac{N-F}{strid} + 1 = \frac{8-3}{1} + 1 = 6$

Column size $= \frac{N-F}{strid} + 1 = \frac{6-3}{1} + 1 = 4$

- 입력채널 : 40
- 출력 데이터(Activation Map) Shape : (6,4,60)
- 학습 파라미터 : 21,600개(40×3×3×60)

4.3 Layer 3의 Shape와 파라미터

4.3.2 Max Pooling Layer 3

- 입력 데이터 Shape : (6,4,60)
- Max Pooling의 크기 : (2,2)

출력 데이터(Activation Map)의 Shape

Row size = $\frac{6}{2}$ = 3

Column size= $\frac{4}{2}$ = 2

- 입력 채널 : 60
- 출력 데이터 Shape : (3,2,60)
- 학습 파라미터 : 0

4.4 Layer 4의 Shape과 파라미터

4.4.1 Convolution Layer 4

- 입력 데이터 shape : (3,2,60)
- 입력 채널 : 60
- 필터 : (2,2)
- 출력 채널 : 80
- Stride : 1

출력 데이터(Activation Map)의 Shape

Row size = $\frac{N-F}{strid}+1=\frac{3-2}{1}+1=2$

Column size= $\frac{N-F}{strid}+1=\frac{2-2}{1}+1=1$

- 입력채널 : 60
- 출력 데이터(Activation Map) Shape : (2,1,80)
- 학습 파라미터 : 19,200개(60×2×2×80)

4.5 Flatten Layer의 Shape

Flatten Layer : CNN 데이터 타입
→ Fully Connected Neural Network로 변경

※파라미터가 존재하지 않음
입력 데이터의 Shape 변경만 수행

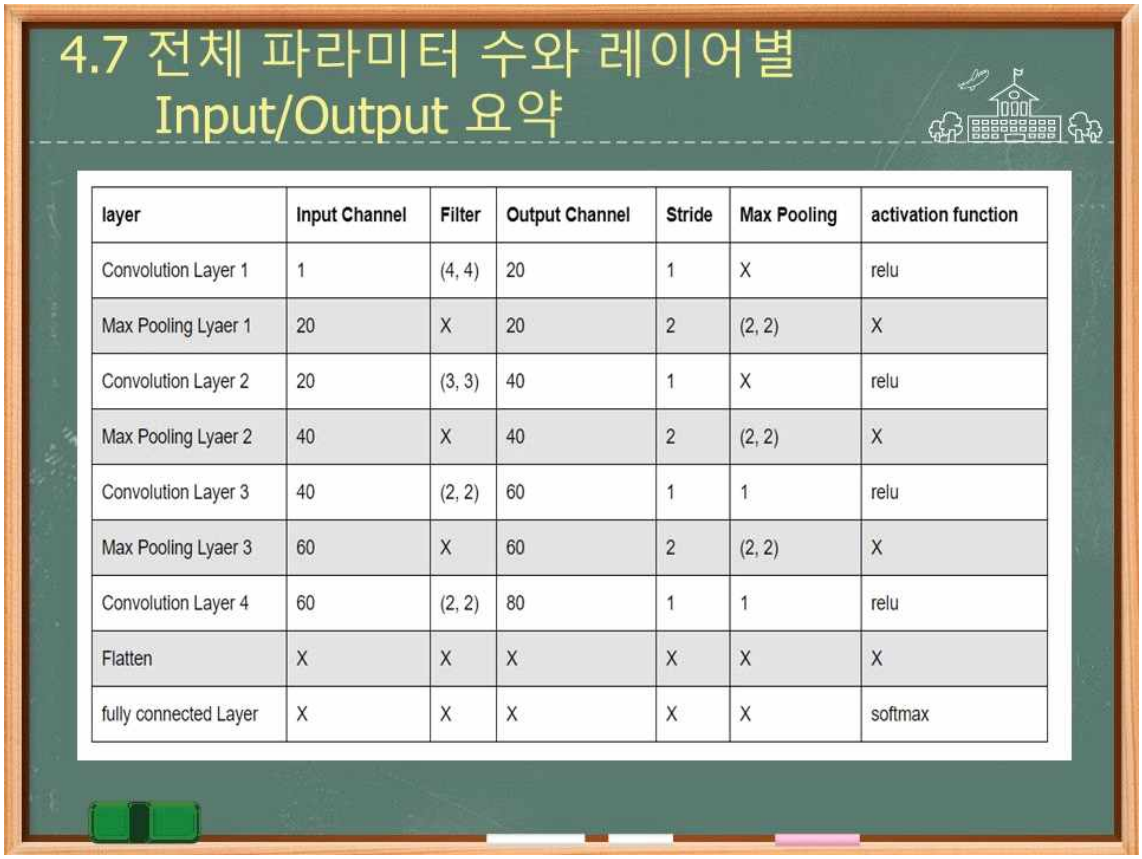
- 입력 데이터 Shape : (2,1,80)
- 출력 데이터 Shape : (160,1)

4.6 Softmax Layer

- 입력 데이터 Shape : (160,1)
- 분류 클래스 : 100

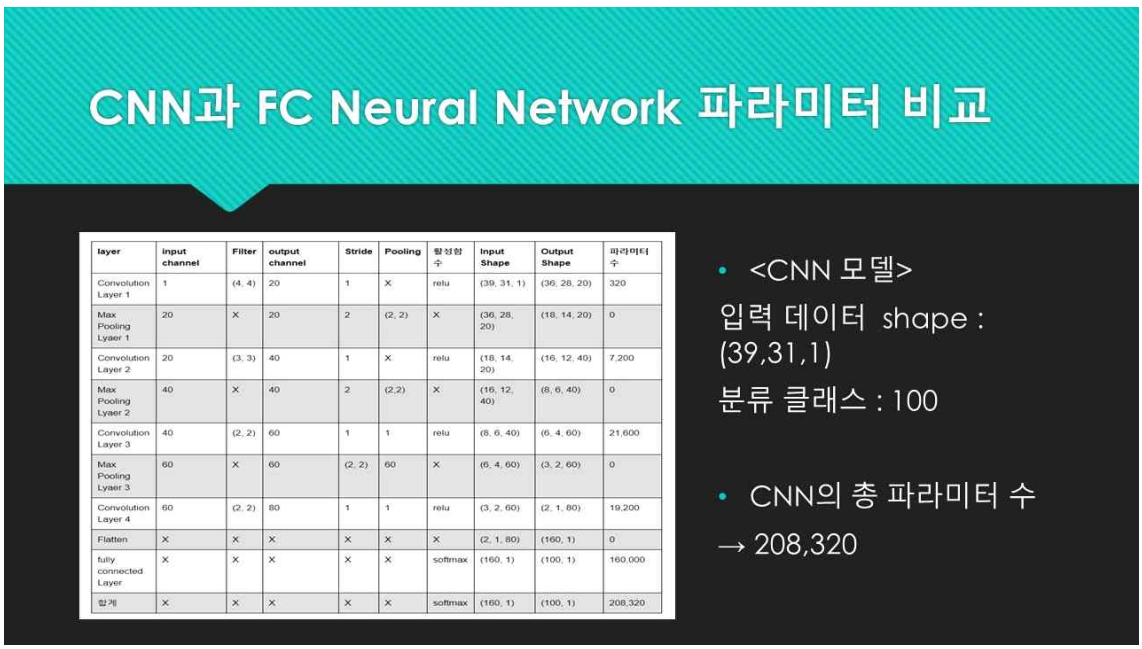
- 출력 데이터 Shape : (100,1)

- Weight Shape (100,160)
- Softmax Layer의 파라미터 : 160,000개(100×160)



layer	Input Channel	Filter	Output Channel	Stride	Max Pooling	activation function
Convolution Layer 1	1	(4, 4)	20	1	X	relu
Max Pooling Lyaer 1	20	X	20	2	(2, 2)	X
Convolution Layer 2	20	(3, 3)	40	1	X	relu
Max Pooling Lyaer 2	40	X	40	2	(2, 2)	X
Convolution Layer 3	40	(2, 2)	60	1	1	relu
Max Pooling Lyaer 3	60	X	60	2	(2, 2)	X
Convolution Layer 4	60	(2, 2)	80	1	1	relu
Flatten	X	X	X	X	X	X
fully connected Layer	X	X	X	X	X	softmax

< 그림 9-1. 이지수 학생의 CNN 입출력 파라미터 계산 설명 일부 ppt >



layer	input channel	Filter	output channel	Stride	Pooling	활성함수	Input Shape	Output Shape	파라미터수
Convolution Layer 1	1	(4, 4)	20	1	X	relu	(39, 31, 1)	(36, 28, 20)	320
Max Pooling Lyaer 1	20	X	20	2	(2, 2)	X	(36, 28, 20)	(18, 14, 20)	0
Convolution Layer 2	20	(3, 3)	40	1	X	relu	(18, 14, 20)	(16, 12, 40)	7,200
Max Pooling Lyaer 2	40	X	40	2	(2,2)	X	(16, 12, 40)	(8, 6, 40)	0
Convolution Layer 3	40	(2, 2)	60	1	1	relu	(8, 6, 40)	(6, 4, 60)	21,600
Max Pooling Lyaer 3	60	X	60	(2, 2)	60	X	(6, 4, 60)	(3, 2, 60)	0
Convolution Layer 4	60	(2, 2)	80	1	1	relu	(3, 2, 60)	(2, 1, 80)	19,200
Flatten	X	X	X	X	X	X	(2, 1, 80)	(160, 1)	0
fully connected Layer	X	X	X	X	X	softmax	(160, 1)	(100, 1)	160,000
합계	X	X	X	X	X	softmax	(160, 1)	(100, 1)	208,320

CNN과 FC Neural Network 파라미터 비교

- 이 CNN과 유사한 4개의 은닉 레이어를 갖는 FC Neural Network

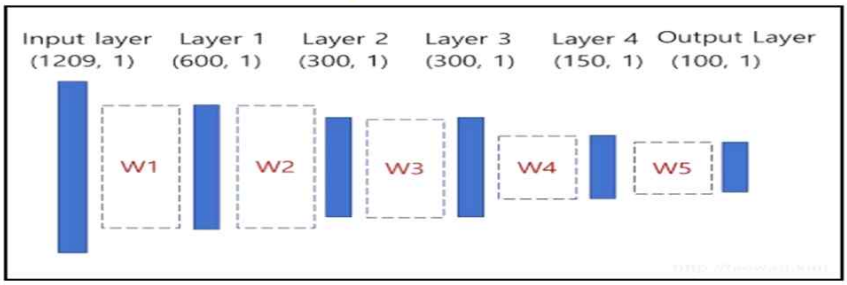


그림 10: FC 신경망 - 비교

CNN과 FC Neural Network 파라미터 비교

- 총 파라미터 → 1,055,400
- CNN 파라미터와 비교하면 10배 이상의 학습 파라미터

레이어	입력 노드	출력 노드	Weight Shape	파라미터 수
Layer 1	1209	600	(1209,600)	725,400
Layer 2	600	300	(600,300)	180,000
Layer 3	300	300	(300,300)	90,000
Layer 4	300	150	(300,150)	45,000
Output	150	100	(150,100)	15,000
합계				1,055,400

<은닉층 더 깊게 만들 경우>

CNN vs FC Neural Network

- CNN은 학습 파라미터 수가 매우 작음
- 학습 파라미터가 작고, 학습이 쉽고 네트워크 처리 속도가 빠름

합성곱 신경망 요약 – CNN(Convolution Neural Network)

- 이미지의 공간 정보 유지하면서 인접 이미지와의 특징을 효과적으로 인식 & 강조하는 방법
- 이미지의 특징을 추출하는 부분 + 이미지를 분류하는 부분
- 특징 추출 영역은 Fiilter 사용

→ 공유 파라미터 수 최소화 & 이미지의 특징을 찾는 Convolution 레이어

→ 특징을 강화하고 모으는 Pooling 레이어 로 구성

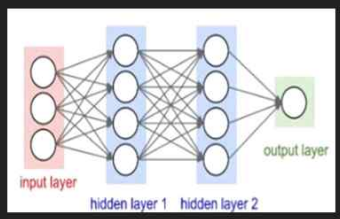
합성곱 신경망 요약 – CNN(Convolution Neural Network)

- Filter의 크기, Stride, Padding, Pooling 크기로 출력 데이터 크기 조절 & 필터의 개수로 출력 데이터의 채널 결정
- 학습 파라미터의 양 : 20% * FC Neural Network = CNN (같은 레이어 크기)

(은닉층 깊어질수록 학습 파라미터 차이는 더 벌어짐)

- CNN은 FC Neural Network와 비교하여 더 작은 학습 파라미터

→ 더 높은 인식률을 제공



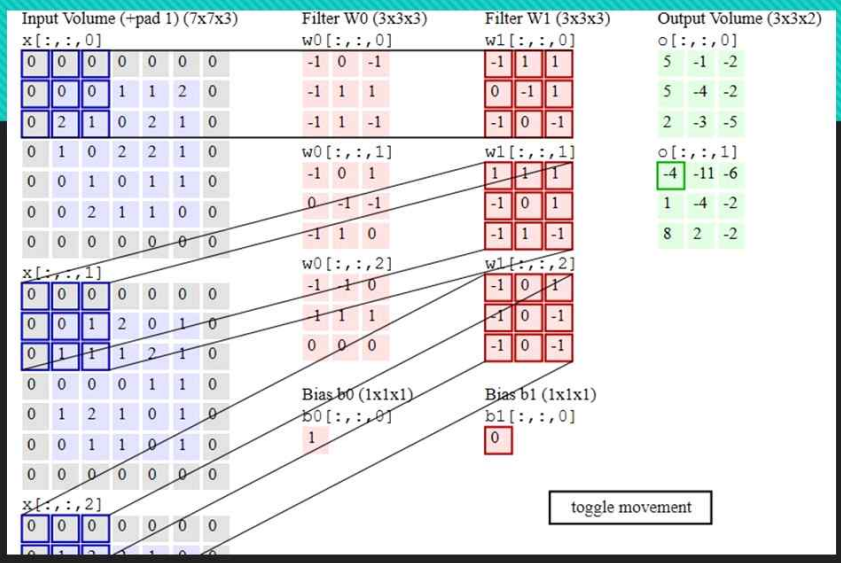
합성곱 신경망 요약 - 3차원 데이터의 합성곱 연산

- 하나의 입력 채널에는 하나의 필터 채널이 필요
  = 입력 데이터의 채널 수와 필터의 채널 수가 일치해야 함
- 각각의 필터 채널에서 연산한 값을 모두 더한 값이 output 채널

→ output 채널의 수는 필터 채널의 수와는 관련 X / 필터가 몇 개 있느냐 O

- 각 필터 채널에서 연산한 값을 모두 더해야 함

→ 모든 채널의 필터는 같은 크기여야 함

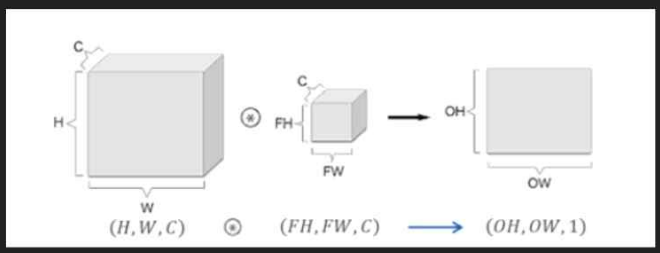


(Height, Width, Channel)

합성곱 신경망 요약 - 3차원 데이터의 합성곱 연산

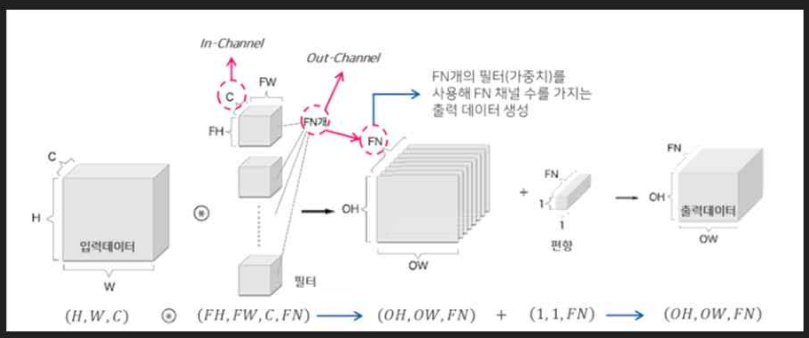
<수식으로 나타내기>

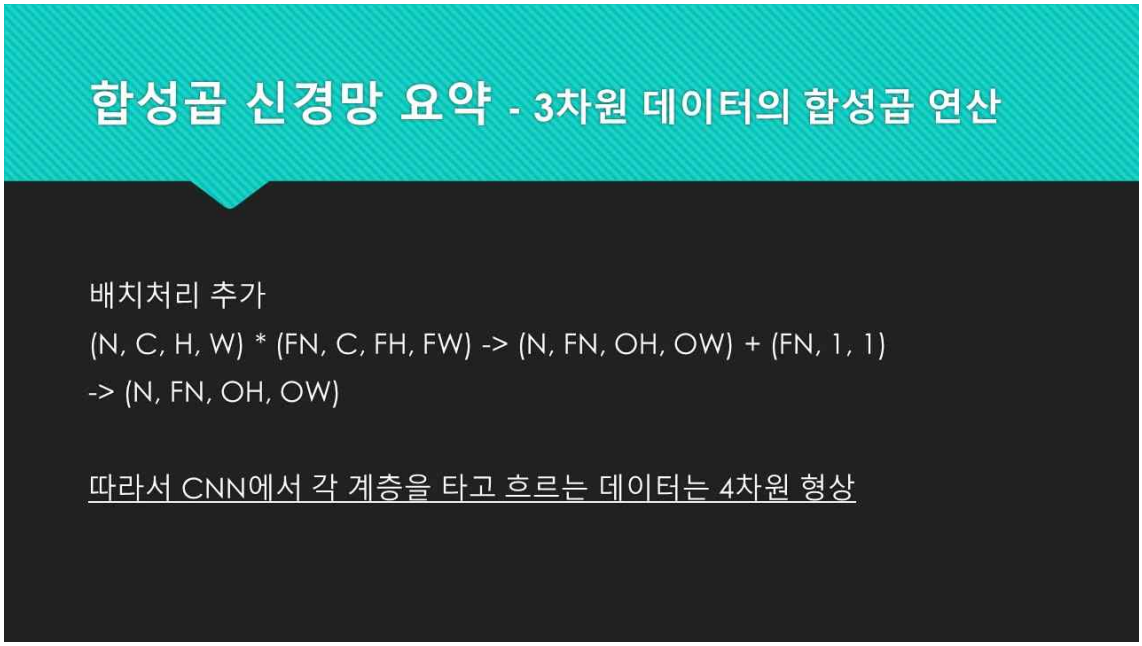
- (Number(개수), Channel, Height, Width) : 고차원 → 저차원
- (C, H, W) * (FN, C, FH, FW) -> (FN, OH, OW)



합성곱 신경망 요약 - 3차원 데이터의 합성곱 연산

편향추가 : (FN, OH, OW) + (FN, 1, 1) -> (FN, OH, OW)





< 그림 9-2. 조지수 학생의 CNN과 FC Neural Network 파라미터 비교 설명 일부 ppt >

10. 11주차 : 최종 모델 구현 및 점검

2020년 5월 26일, ZOOM을 이용하여 비대면 회의를 진행했다.

본격적인 모델을 구현하기 전에 이지수, 이정 학생이 참고할만한 코드를 수집했다. keras라는 파이썬 라이브러리를 사용하여 CNN모델을 구현하기로 했다.
모델 구현을 위한 코드 작성은 오서영, 조지수 학생이 맡았다.

```
batch_size = 128
num_classes = 10
epochs = 50
```

batch_size : 배치처리 크기
num_classes : 분류 클래스 개수
epochs : 전체 데이터를 50번 사용해서 학습을 거치는 것

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5), strides=(1, 1), padding='same',
                 activation='relu',
                 input_shape=(28,28,1)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (5, 5), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
model.summary()
```

Conv2D : 필터사이즈가 5X5이고 활성화 함수가 relu 함수인 Convolution Layer
MaxPooling2D : 사이즈가 2X2인 풀링 층이다.
Dropout(0.25) : 25% 확률로 유닛을 제거한다.
Flatten : CNN 데이터 타입을 FC Layer 데이터 타입으로 변경
첫 번째 Dense : 활성화 함수가 relu 함수인 Fully Connected Layer
두 번째 Dense : 마지막 분류를 위한 softmax 함수를 사용하는 FC Layer

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 28, 28, 32)        832
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 14, 14, 64)        51264
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
dropout_5 (Dropout)          (None, 7, 7, 64)          0
_________________________________________________________________
flatten_3 (Flatten)          (None, 3136)              0
_________________________________________________________________
dense_5 (Dense)              (None, 1024)              3212288
_________________________________________________________________
dropout_6 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_6 (Dense)              (None, 10)                10250
=================================================================
Total params: 3,274,634
Trainable params: 3,274,634
Non-trainable params: 0
_________________________________________________________________
```

< 그림 10-1. CNN 모델 구조 요약 >

```
model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])
hist = model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
```

categorical_crossentropy : softmax 함수와 같이 쓰이는 손실함수이다.
sgd : 최적화 알고리즘으로 경사하강법을 사용했다.

```
Train on 55000 samples, validate on 10000 samples
Epoch 1/50
55000/55000 [==============================] - 85s 2ms/step - loss: 1.2631 - accuracy: 0.6072 - val_loss: 0.3239 - val_accuracy: 0.9107
Epoch 2/50
55000/55000 [==============================] - 88s 2ms/step - loss: 0.3708 - accuracy: 0.8858 - val_loss: 0.2013 - val_accuracy: 0.9402
Epoch 3/50
55000/55000 [==============================] - 87s 2ms/step - loss: 0.2608 - accuracy: 0.9207 - val_loss: 0.1504 - val_accuracy: 0.9552
Epoch 4/50
55000/55000 [==============================] - 92s 2ms/step - loss: 0.2065 - accuracy: 0.9366 - val_loss: 0.1228 - val_accuracy: 0.9616

. . .

Epoch 46/50
55000/55000 [==============================] - 80s 1ms/step - loss: 0.0420 - accuracy: 0.9865 - val_loss: 0.0273 - val_accuracy: 0.9896
Epoch 47/50
55000/55000 [==============================] - 80s 1ms/step - loss: 0.0409 - accuracy: 0.9874 - val_loss: 0.0266 - val_accuracy: 0.9903
Epoch 48/50
55000/55000 [==============================] - 81s 1ms/step - loss: 0.0422 - accuracy: 0.9866 - val_loss: 0.0259 - val_accuracy: 0.9907
loss: 0.0425
Epoch 49/50
55000/55000 [==============================] - 82s 1ms/step - loss: 0.0409 - accuracy: 0.9869 - val_loss: 0.0255 - val_accuracy: 0.9907
Epoch 50/50
55000/55000 [==============================] - 80s 1ms/step - loss: 0.0400 - accuracy: 0.9876 - val_loss: 0.0258 - val_accuracy: 0.9904
```

< 그림 10-2. 50번 학습중인 모습 >

11. 12주차 : 모델 정확도 확인 및 분석

2020년 6월 3일, ZOOM을 이용하여 비대면 회의를 진행했다.

완성된 모델이 손글씨 데이터를 얼마나 잘 분류하는지 확인하기 위해 정확도를 계산하고, 잘못 예측된 데이터를 확인했다. 오서영, 조지수 학생이 코드를 작성하고 이지수, 이정학생이 결과를 분석했다.

```
score1 = model.evaluate(x_train, y_train, verbose=0)
score2 = model.evaluate(x_test, y_test, verbose=0)
print('Train accuracy:', score1[1])
print('Test accuracy:', score2[1])
```

```
Train accuracy: 0.9936909079551697
Test accuracy: 0.9904000163078308
```

훈련데이터에 대한 정확도는 약 99%, 테스트데이터에 대한 정확도 또한 약 99%이다.
정확도는 (잘 예측된 데이터의 개수) * 100 / (전체 데이터의 개수) 으로 계산된다.
- 우리가 설정한 목표 정확도는 97% 이므로, 목표를 달성했다.

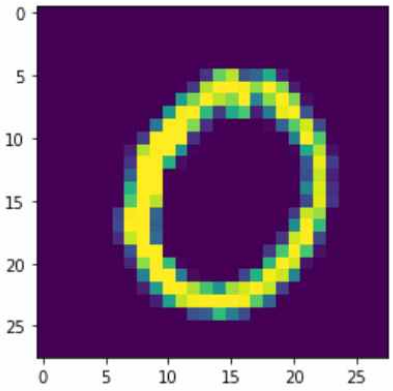
```
predictions[101]
```

```
array([9.9999607e-01, 3.3771885e-10, 2.7141516e-08, 1.3702549e-11,
       2.9556857e-11, 6.2845267e-08, 3.8045785e-06, 2.5531390e-09,
       3.3442760e-08, 1.2291128e-08], dtype=float32)
```

이것은 101번째 이미지가 예측된 결과이다. 열 개의 예측된 숫자 중 첫 번째가 가장 크다.

```
print(np.argmax(predictions[101]))
plt.imshow(x_test[101].reshape(28,28))
```

```
0
```



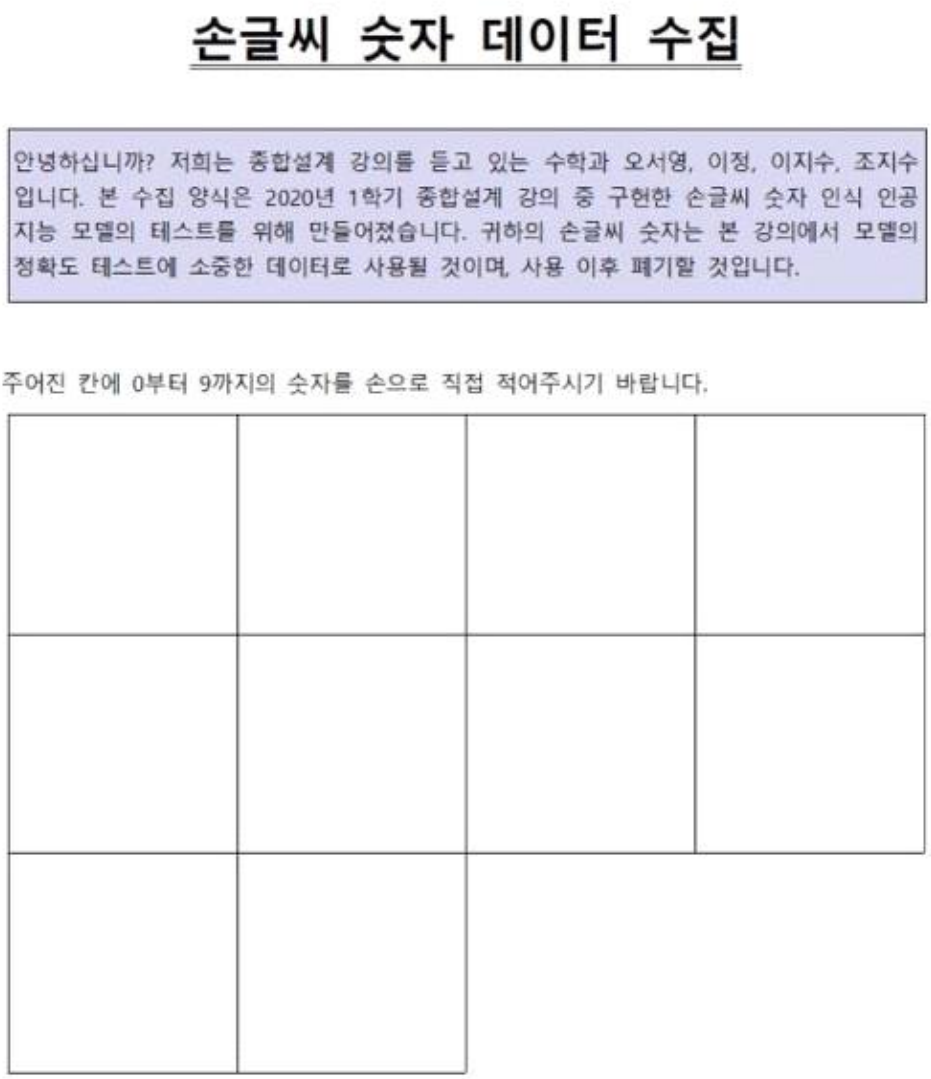
argmax 함수를 통해 가장 큰 숫자의 인덱스를 출력하면 '0'이 나오고, 이미지는 다음과 같다. 이미지와 예측된 라벨이 일치하므로 잘 예측됐다고 볼 수 있다.

12. 13주차 : 모델 테스트

2020년 6월 9일, ZOOM을 이용하여 비대면 회의를 진행했다.

최종 모델이 실제 사람 손글씨에도 잘 적용이 되는지 평가해야한다. 직접 제작한 데이터 수집 양식을 통해 실제 손글씨 데이터를 50명을 대상으로 총 500개 수집했다.

실제 손글씨 데이터를 이지수, 이정 학생이 수집했고, 모델 테스트 코드를 오서영, 조지수 학생이 작성했다.

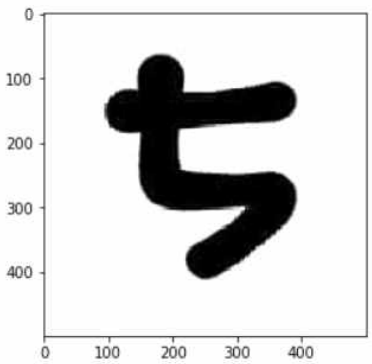


< 그림 12-1. 데이터 수집 양식>

< 그림 12-2. 실제 손글씨 데이터 일부>

```
# Fit algorithm.
image = np.array(Image.open('5/17.jpg'))
img1 = Image.fromarray(image).convert('L') # gray
img2 = img1.resize((28,28))
img3 = ((np.array(img2) / 255) - 1) * -1


plt.imshow(image)
print("Your algorithm predicts: y = " + str(loaded_model.predict_classes(img3.reshape((1, 28, 28, 1)))))
```

```
Your algorithm predicts: y = [5]
```



불러온 이미지를 흑백 이미지로 바꾸고, 모델의 입력 데이터 사이즈에 맞게 28*28 크기로 데이터를 조절한다. 그리고 데이터를 255로 나누고 1을 뺀 후 -1을 곱하여 0~1 사이의 값으로 전처리를 해준다.

전처리된 데이터를 통해 예측한 결과는 '5' 이다.

```
# accuracy
num_wrong = 0
for i in range(500):
    if y_real[:,i]!=pred_real[i]:
        num_wrong += 1

print("The number of correct prediction :", 500-num_wrong)
print("The number of wrong prediction :", num_wrong)
print("Accuracy :", (500-num_wrong)*100/500,"%")
```

```
The number of correct prediction : 452
The number of wrong prediction : 48
Accuracy : 90.4 %
```

위의 과정을 통해 500개의 데이터를 예측한 것으로 정확도를 계산한 결과는 90.4% 이다.

500개의 데이터 중 452개 데이터의 예측을 성공했다.

V. 참고자료

[1] [특별기고] 알파고, 인공지능, 그리고 수학, "인공지능 수학" , https://mathsci.kaist.ac.kr/newsletter/article/%ED%8A%B9%EB%B3%84%EA%B8%B0%EA%B3%A0-%EC%95%8C%ED%8C%8C%EA%B3%A0-%EC%9D%B8%EA%B3%B5%EC%A7%80%EB%8A%A5-%EA%B7%B8%EB%A6%AC%EA%B3%A0-%EC%88%98%ED%95%99/, (2020.03.16)

[2] [김정호의 AI시대의 전략] AI 시대, 수학 실력이 최고의 경쟁력이다, "인공지능 수학", http://news.chosun.com/site/data/html_dir/2019/11/11/2019111100009.html, (2020.03.16)

[3] [김정호의 4차혁명 오딧세이] 인공지능 반도체의 미래, "인공지능 수학", http://www.newspim.com/news/view/20190217000152, (2020.03.16)

[4] 인공 신경망이란 무엇인가?, "인공신경망", https://blog.lgcns.com/1359, (2020.03.23)

[5] 머신러닝 초보를 위한 MNIST, "손글씨 데이터 분석", https://codeonweb.com/entry/12045839-0aa9-4bad-8c7e-336b89401e10, (2020.03.23)

[6] 가볍게 읽어보는 머신 러닝 개념 및 원리, "머신러닝", https://ellun.tistory.com/103?category=276044, (2020.03.30.)

[7] MNIST database, "손글씨 데이터", http://yann.lecun.com/exdb/mnist/, (2020.04.13.)

8] 머신러닝 초보를 위한 MNIST, "MNIST 데이터 분석", https://codeonweb.com/entry/12045839-0aa9-4bad-8c7e-336b89401e10 (2020.04.23.)

[9] 합성곱 신경망, "합성곱 신경망", https://machine-geon.tistory.com/46 (2020.05.13.)

[10] 케라스: 파이썬 딥러닝 라이브러리, "케라스" https://keras.io/ko/ (2020.05.26.)