손글씨 숫자인식 인공지능 구현하기

2017010698 오서영
2018080048 이 정
2017010709 조지수
2017010702 이지수

Ⅰ. 수행목적과 목표

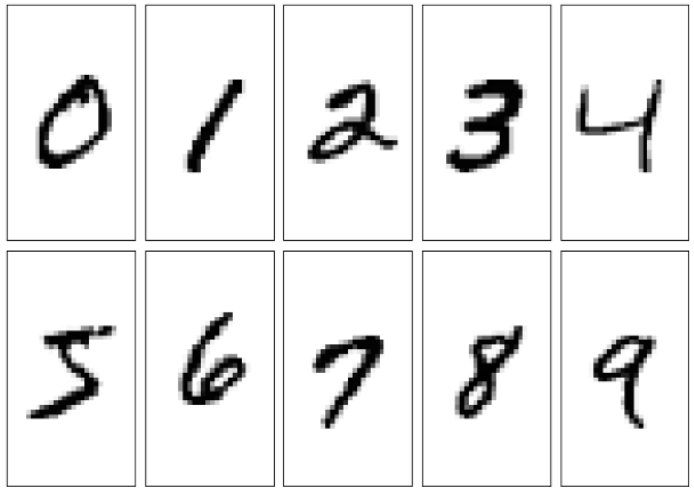
► 수행 목적

4차 산업혁명 시대를 맞아 인공지능의 역할이 커지고 있지만 아직 수학은 학문으로만 취급을 받고 있다. 수행 목적은 수학의 또 다른 측면을 발견하기 위해 수학이 많이 사용되는 인공지능 모델을 구현하는 것이다.

► 수행 목표

수행 목표는 최종 구현한 모델이 손으로 쓴 0부터 9까지의 숫자 이미지를 잘 분류하는 것이다.

즉, 테스트 정확도가 97%이상 나오는 것이다.



1~2주차

Ⅱ. 수행계획

1. 선행학습

Neural Network 모델 구현에 필요한 컴퓨터 언어와 수학적 개념에 대해 학습.

1) 컴퓨터 언어 : 파이썬

2) 인공지능 기초학습

3) 수학적 개념 : 행렬, 손실함수, 경사하강법

2. 모델구현하기

모델 구현 과정은 다음 순서와 같다.

모델 구상 및 데이터 수집 ➔ Neural Network 코드 초안 작성 ➔ 모델 구현 및 수정 ➔ 정확도 확인 및 분석 ➔ 최종 모델 구현 및 점검 ➔ 제작완료

3. 모델 평가

1) 모델 분석

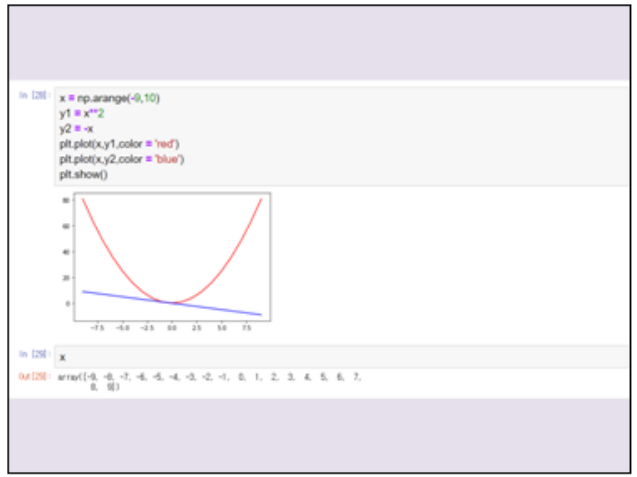
2) 실제 데이터 테스트

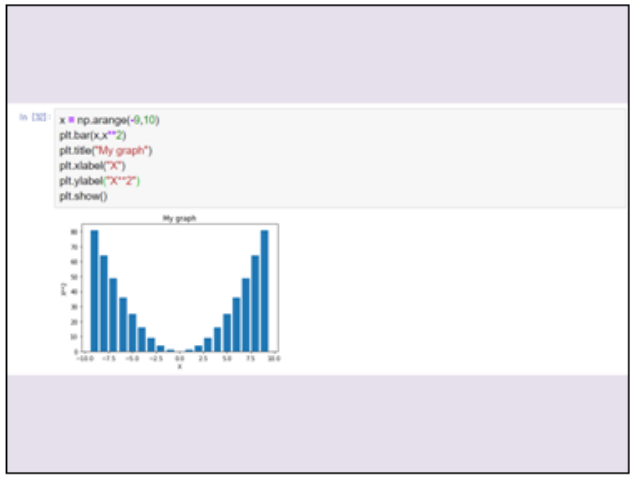
(1) 실제 손글씨 데이터 수집

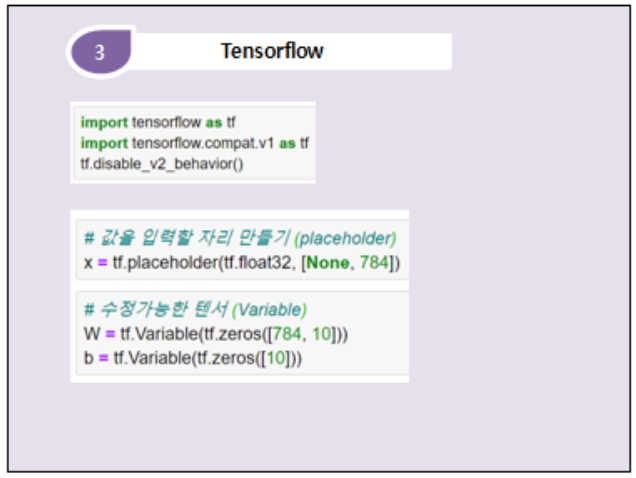
(2) 최종 모델 테스트

Ⅲ. 수행내용

► 파이썬(Python)







3주차

Ⅲ. 수행내용

► 인공지능(Aritificial Intelligence)

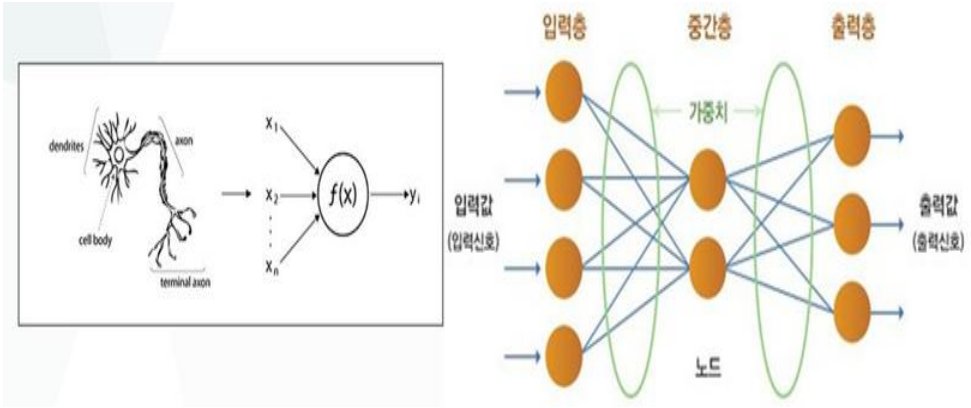
인간의 지능적인 행동들을 컴퓨터로 프로그램으로 실현하는 기술

► 머신 러닝 (Machine Learning)

컴퓨터에게 데이터들을 제공하여 학습하게 함으로써 새로운 지식을 얻어내게 하는 분야

►인공신경망(Aritificial Neural Network)

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



Ⅲ. 수행내용

► 활성화 함수(Activation function)

출력 값을 활성화를 일으키게 할 것이냐를 결정하고, 그 값을 부여하는 함수

즉, 선형함수를 비선형함수로 바꾸기 위해서 사용한다.

► 활성화 함수의 종류

시그모이드(sigmoid) 함수, 계단(step) 함수 , 렐루(Relu) 함수

► 소프트맥스 함수(Softmax function)

분류에 사용되는 함수로 모든 입력신호로 부터 영향을 받는 함수

Ⅲ. 수행내용

► 손실함수 (Loss function)

예상한 값과 실제 타깃 값의 차이를 함수로 정의한 것

► 미분 (Differential)

수치미분 : 실제 미분 값이 아니라 실제 값에 대한 근사값

해석적 미분 : 실제로 수식을 미분해 도함수를 구하는 것

►기울기(Gradient)

미분을 통해 계산되는 기울어짐의 정도

►경사(하강) 법 (Gradient descent)

기울기를 이용해 손실함수의 최솟값을 찾는 법

► 오차역전파법 (Backpropagation)

역방향으로 해당 함수의 국소적 미분을 곱해 나가는 것

5주차

Ⅳ. 데이터 수집 및 모델구상

► 데이터 수집

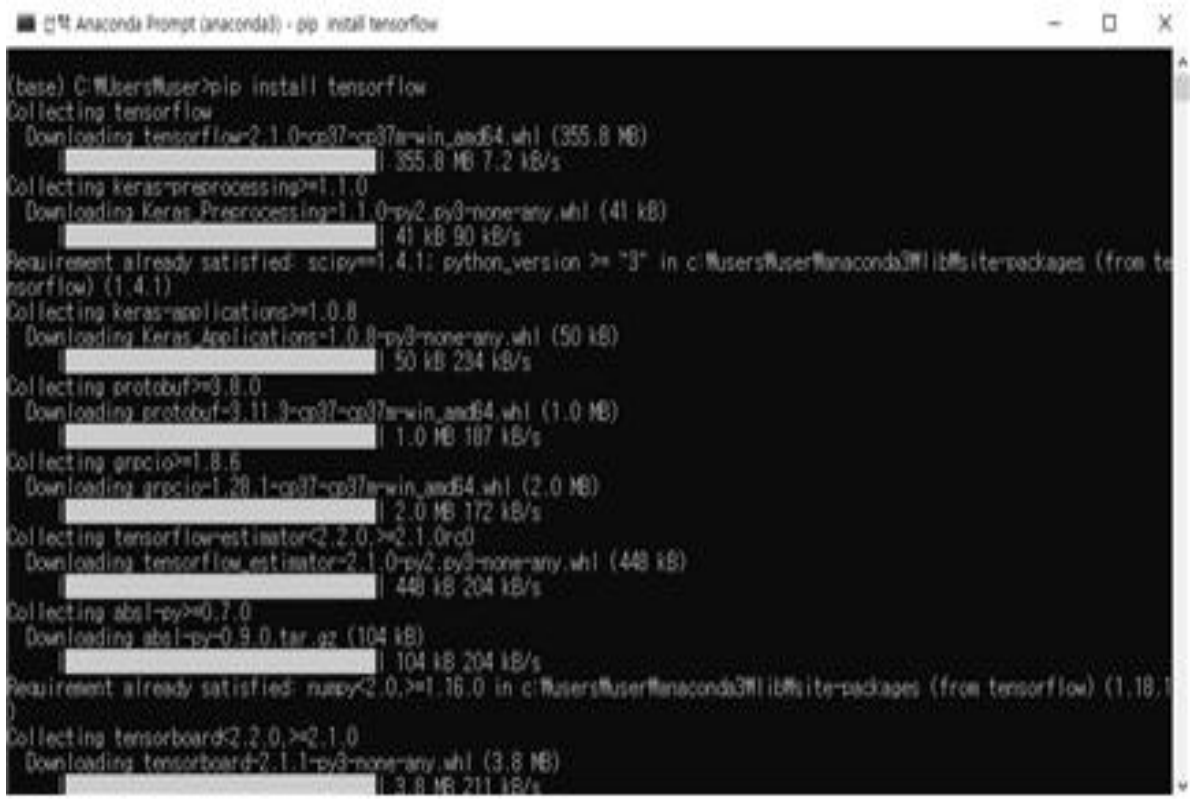
Yann LeCun의 웹사이트에 호스팅되어 있는 'MNIST 데이터셋'을 사용

다운로드한 데이터는 55,000개의 학습 데이터, 10,000개의 테스트 데이터, 그리고 5,000개 의 검증 데이터이다.

Ⅳ. 데이터 수집 및 모델구상

► 프로그램 설치

파이썬과 라이브러리를 설치하기 위해 'Anaconda' 프로그램에 내장되어있는 'Anaconda Prompt' 와 'Pip'을 사용하여 텐서플로우(tensorflow) 를 설치

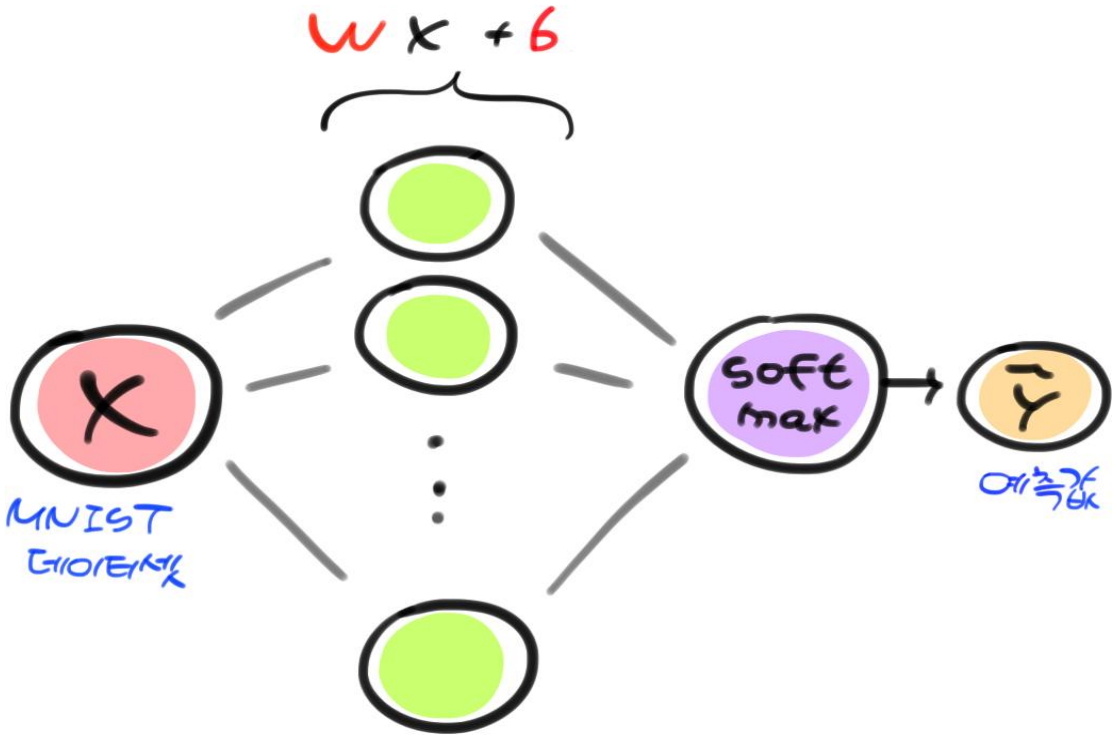


5주차

Ⅳ. 데이터 수집 및 모델구상

► 모델 구상

기본적인 Single-Layer Neural Network 모델을 만들었다. 활성화 함수로는 다중분류 회귀인 Softmax 함수를 사용

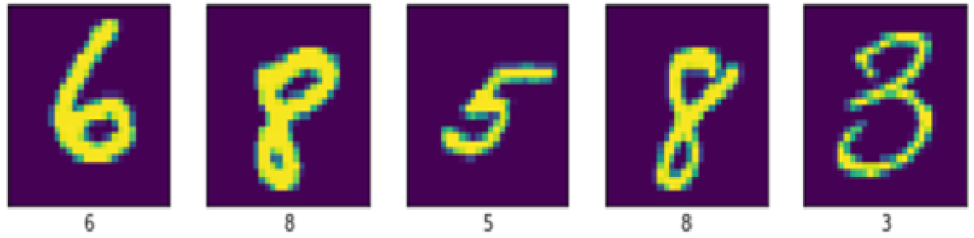


V. Neural Network 모델 구현

1) 패키지 불러오기

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

2) 데이터 확인하기



matplotlib 패키지를 활용하여 이미지와 실제 정답 라벨을 확인

3) 데이터셋 만들기

```
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, 10000개의 테스트데이터를 불러와 데이터 셋을 만듦

V. Neural Network 모델 구현

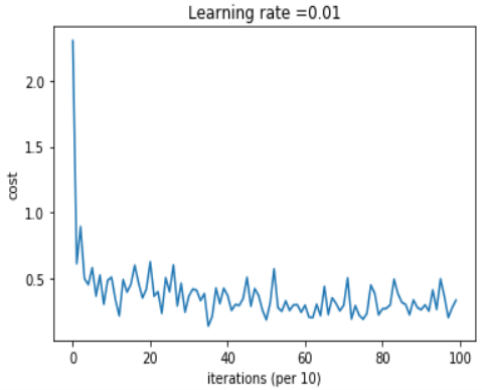
4) 모델 구현하기

```
# 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.

Ⅵ.모델 정확도 확인 및 분석

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.

9주차

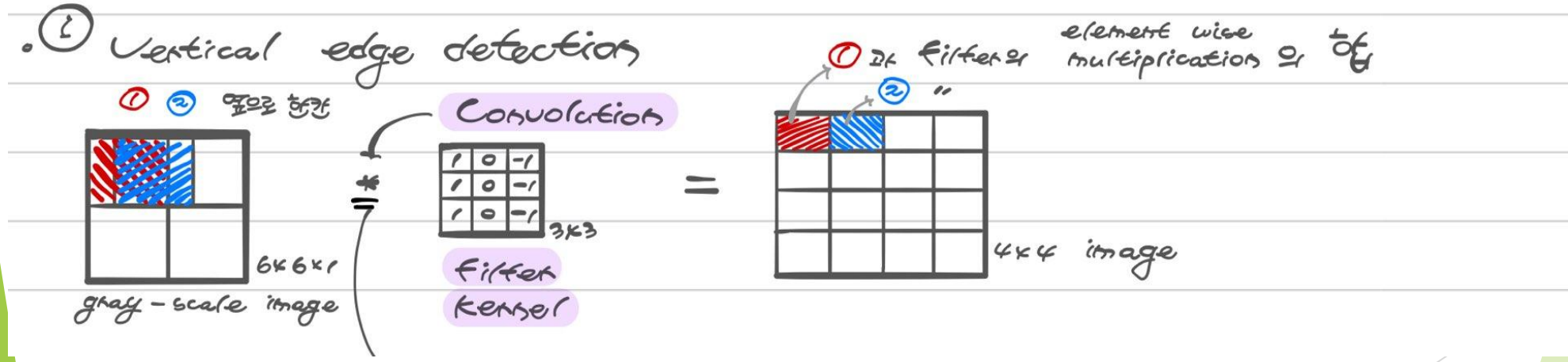
Ⅶ. 새 모델 및 기법 학습 1

► CNN

모델이 직접 이미지, 비디오, 텍스트 또는 사운드를 분류하는 딥러닝에 가장 많이 사용되는 알고리즘

► 합성곱 (Convolution)

두 함수 f, g 가운데 하나의 함수를 반전(reverse), 전이(shift) 시킨 다음, 다른 하나의 함수와 곱한 결과를 적분하는 것을 의미



9주차

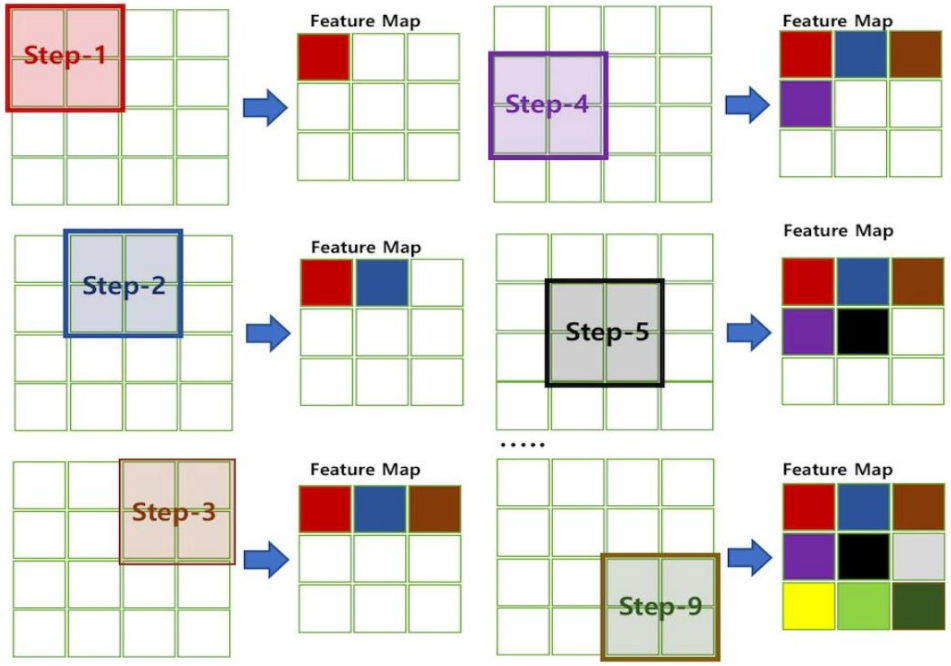
Ⅶ. 새 모델 및 기법 학습 1

► 필터(Filter)

이미지의 특징을 찾아내기 위한 공용 피라미터

►스트라이드(Stride)

지정된 간격으로 필터를 순회하는 간격



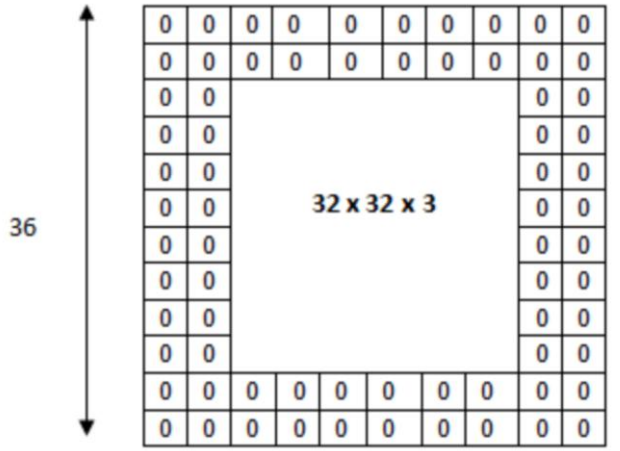
9주차

Ⅶ. 새 모델 및 기법 학습 1

► 패딩(Padding)

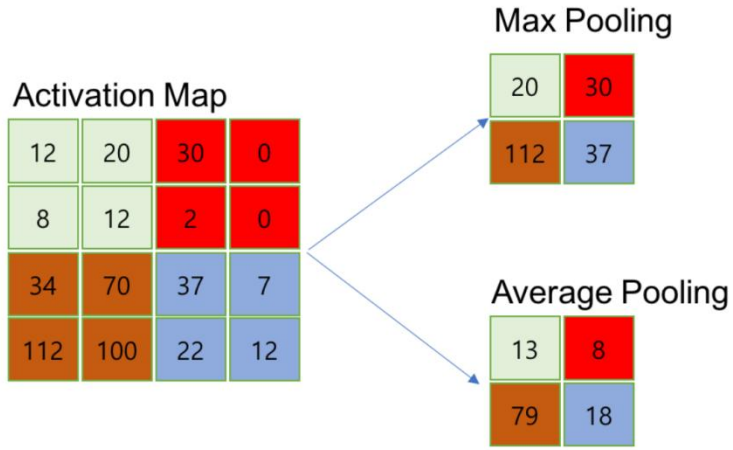
입력 데이터의 외각에 지정된 픽셀만큼 특정 값을 채워 넣는 것

보통 패딩 값은 '0'으로 채워 넣는다.



► 풀링(Pooling)

데이터의 크기를 줄이거나 특정 데이터를 강조하는 용도



Ⅶ. 새 모델 및 기법 학습 1

► Convolution layer 출력데이터 산정

입력 데이터에 대한 필터의 크기와 스트라이드 크기에 따라서 feature map 크기가 산정

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

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

►Pooling layer 출력데이터 산정

Pooling 레이어의 출력 데이터의 크기는 행과 열의 크기를 Pooling 사이즈로 나눈 몫

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

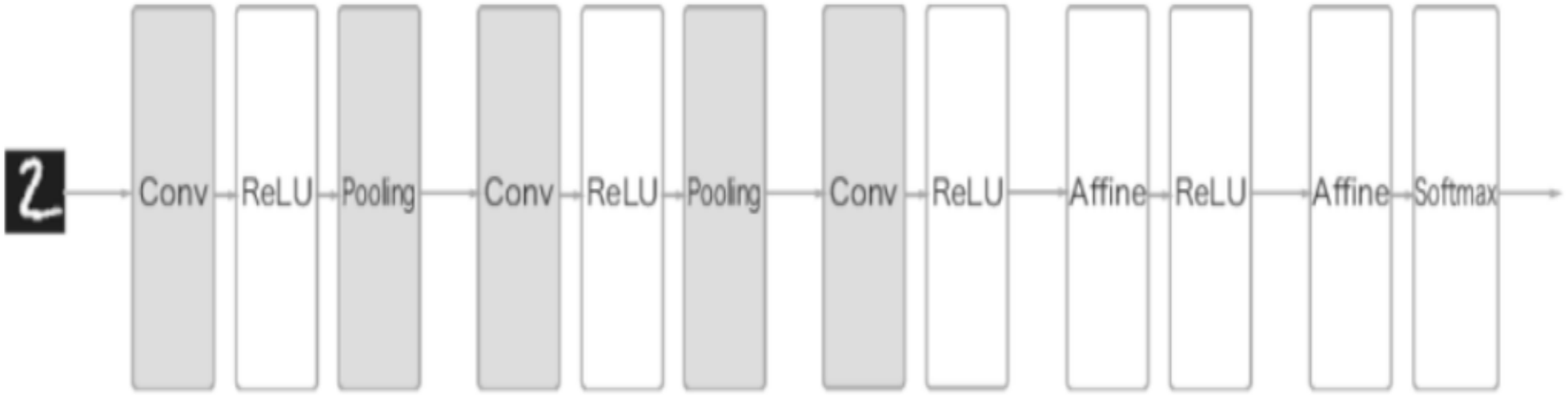
Output ColumnSize= $\frac{Input\ Column\ Size}{Pooling\ Size}$

9주차

Ⅶ. 새 모델 및 기법 학습 1

►CNN구성

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.



10주차

Ⅶ. 새 모델 및 기법 학습 2

►Flatten Layer

CNN 데이터 타입 ➔ Fully Connected Neural Network로 변경

► Softmax Layer

Flatten Layer 에서의 출력데이터가 입력데이터로 바뀌면서 Weight로 학습.

► CNN과 FC Neural Network 피라미터 비교

CNN은 학습 피라미터 수가 매우 작음

학습 파라미터가 작고, 학습이 쉽고 네트워크 처리 속도가 빠름

Ⅷ. 최종 모델 구현 및 점검

keras라는 파이썬 라이브러리를 사용하여 CNN모델을 구현했다.

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

```
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()
```

IX. 모델 정확도 확인 및 분석

► 모델 정확도 확인

완성된 모델이 손글씨 데이터를 얼마나 잘 분류하는지 확인하기 위해 정확도를 계산하고, 잘못 예측된 데이터를 확인했다.

```
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
```

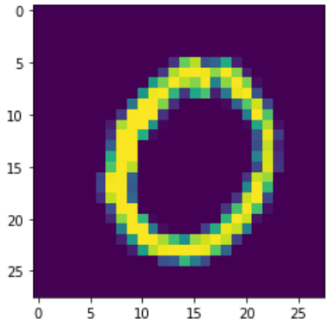
► 모델 정확도 분석

```
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)
```

```
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.

X.모델 테스트

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

<데이터 수집>

<정확도 테스트>

```
# 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 %
```

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.)

감사합니다