이미지 인식의 꽃, CNN 익히기

사람이 볼 때는 쉽게 알 수 있는 글씨라 해도, 숫자 5는 어떤 특징을 가졌고, 숫자 9는 6과 어떻게 다른지를 기계가 스스로 파악하여 정확하게 읽고 판단하게 만드는 것은 머신러닝의 오랜 진입 과제였다.

MNIST 데이터셋을 이용해 딥러닝이 손글씨 이미지를 얼마나 정확히 예측할 수 있는지 알아보자

1. 데이터 전처리

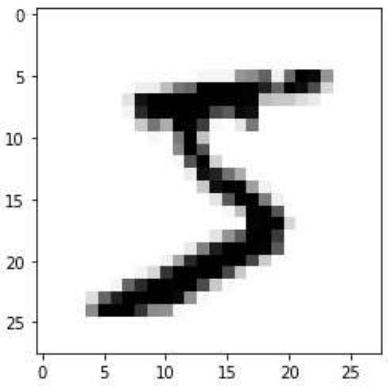
(실습코드는 github 참고)

X: 불러온 이미지 데이터, **Y_class**: 0~9까지 붙인 이름표

케라스의 MNIST 데이터는 총 70000개의 이미지 중 60000개를 학습용으로, 10000개를 테스트용으로 미리 구분해 놓고 있다.

shape() 함수 - 정수 튜플의 형태로 배열의 형태를 반환한다. 튜플의 값은 배열 차원의 길이를 보여준다.

imshow() 함수를 이용해 이미지를 출력



주어진 데이터의 이미지는 가로 28 x 세로 28 = 총 784개의 픽셀로 이루어져 있음. 각 픽셀은 밝기 정도에 따라 0부터 255까지의 등급을 매긴다.

이 이미지들은 다시 숫자의 집합으로 바뀌어 학습셋으로 사용된다. 앞서 배운 여러 예제와 마찬가지로 속성을 담은 데이터를 딥러닝에 집어넣고 클래스를 예측하는 문제로 전환시키는 것이다.

reshape() 함수 - 현재 배열의 차원을 변경하여 행렬을 반환하는 경우에 많이 이용되는 함수이다.

케라스는 데이터를 0에서 1 사이의 값으로 변환한 다음 구동할 때 최적의 성능을 보임

-> 데이터 정규화(normalization) 과정이 필요!!

데이터 정규화 - 데이터의 폭이 클 때 적절한 값으로 분산의 정도를 바꾸는 과정

데이터를 실수형으로 변환한 다음(astype() 함수 사용) 255로 나누면 된다.

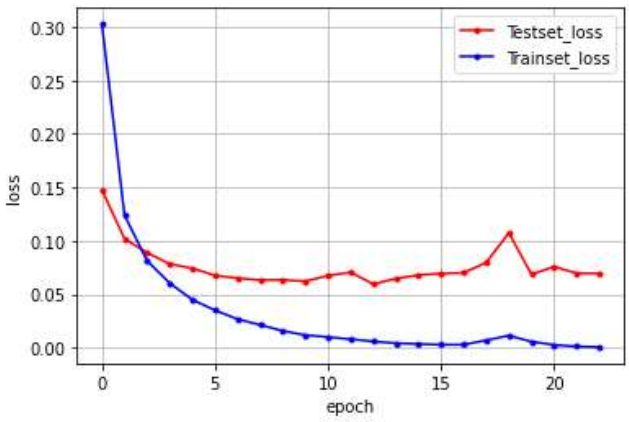
딥러닝의 분류 문제를 해결하려면 **원-핫 인코딩 방식**을 사용해야 함!
-> 0~9 까지의 정수형 값을 갖는 현재 형태에서 0 또는 1로만 이루어진 벡터로 값을 수정해야 함
-> **np_utils.to_categorical() 함수**를 사용!
to_categorical(클래스, 클래스의 개수)의 형식으로 지정

2. 딥러닝 기본 프레임 만들기
(실습코드는 github 참고)
1차원으로 변환한 데이터는 총 784개의 속성이 있고 10개의 클래스가 있다는 것을 알 수 있다.

입력 값이 784개, 은닉층이 512개 그리고 출력이 10개인 모델을 사용.
활성화 함수로 은닉층에서는 relu를, 출력층에서는 softmax를 사용.
오차 함수로 categorical_crossentropy, 최적화 함수로는 adam을 사용.

모델의 실행에 앞서 모델의 성과를 저장하고 모델의 최적화 단계에서 학습을 자동 중단하게끔 설정
-> 10회 이상 모델의 성과 향상이 없으면 자동으로 학습을 중단

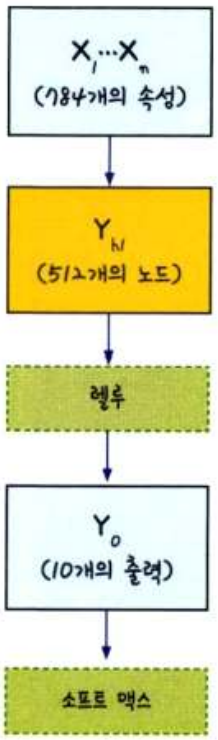
샘플 200개를 모두 30번 실하게끔 설정, 그리고 테스트셋으로 최종 모델의 성과를 측정하여 그 값을 출력



학습셋에 대한 오차는 계속해서 줄어든다. 테스트셋의 과적합이 일어나기 전 학습을 끝낸 모습

-> 학습셋의 정확도는 1.00에 가깝고 테스트셋의 오차는 0.00에 가까우므로 두 개를 비교하기가 어려움. 따라서 1에서 학습셋의 정확도를 뺀 값, 즉 학습셋의 오차를 주로 그래프에 적용하여 표현한다.

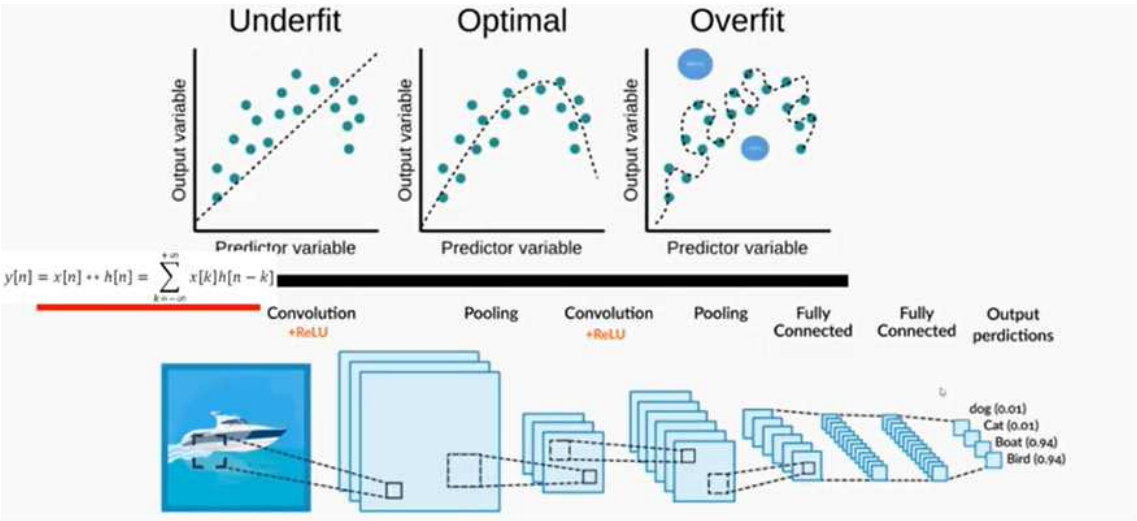
3. 더 깊은 딥러닝



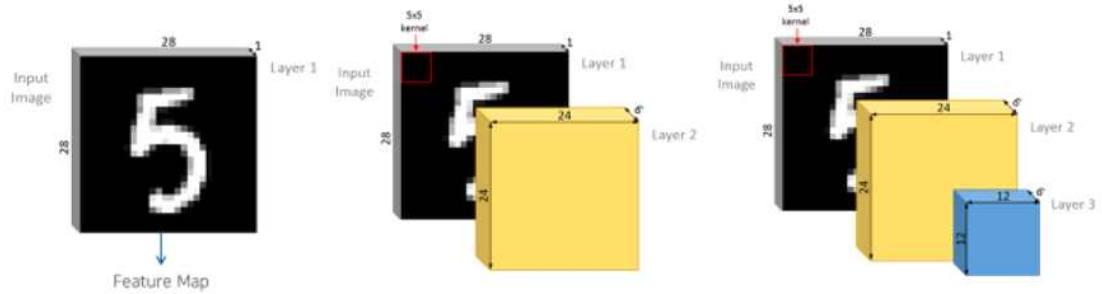
딥러닝은 이러한 기본 모델을 바탕으로, 프로젝트에 맞춰서 어떤 옵션을 더하고 어떤 층을 추가하느냐에 따라 성능이 좋아질 수 있음.

4. 컨볼루션 신경망(CNN)

컨볼루션 신경망 - 입력된 이미지에서 다시 한번 특징을 추출하기 위해 커널(슬라이딩 윈도)을 도입하는 기법



합성곱층 - CNN에서 가장 중요한 구성요소이며 아래의 그림과 같이 입력 데이터의 형상을 유지한다. 3차원의 이미지 그대로 입력층에 입력받으며, 출력 또한 3차원 데이터로 출력하여 다음 계층(layer)으로 전달하기 때문에 CNN에서는 이미지 데이터처럼 형상을 가지는 데이터를 제대로 학습할 가능성이 높다고 할 수 있다.



합성곱층의 뉴런은 입력 이미지의 모든 픽셀에 연결되는 것이 아니라 합성곱층 뉴런의 수용영역(receptive field) 안에 있는 픽셀에만 연결이 되기 때문에, 앞의 합성곱층에서는 저수준 특성에 집중하고, 그 다음 합성곱층에서는 고수준 특성으로 조합해 나가도록 해준다.

수용영역(receptive field) - 전체 영역에 대해 서로 동일한 중요도를 부여하여 처리하는 대신에 특정 범위를 한정해 처리하는 것

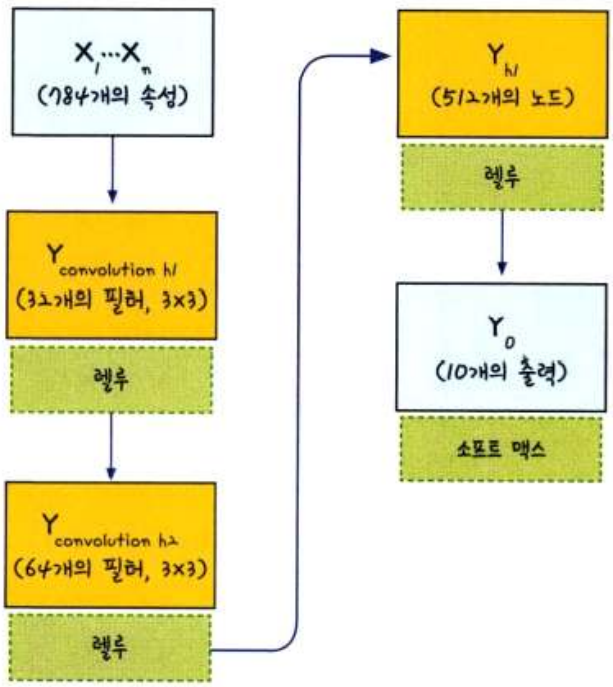
커널(kernel) - 합성곱층에서의 가중치 파라미터(W)에 해당하며 학습단계에서 적절한 필터를 찾도록 학습되며, 합성곱 층에서 입력데이터에 필터를 적용하여 필터와 유사한 이미지의 영역을 강조하는 특성맵(feature map)을 출력하여 다음 층(layer)으로 전달하는 역할을 한다.

합성곱(Convolution) - 하나의 함수와 또 다른 함수를 반전 이동한 값을 곱한 다음, 구간에 대해 적분하여 새로운 함수를 구하는 연산자

공업수학 - $(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau$

CNN - $(f * g)(i,j) = \sum_{x=0}^{h-1}\sum_{y=0}^{w-1} f(x,y)g(i-x, j-y)$

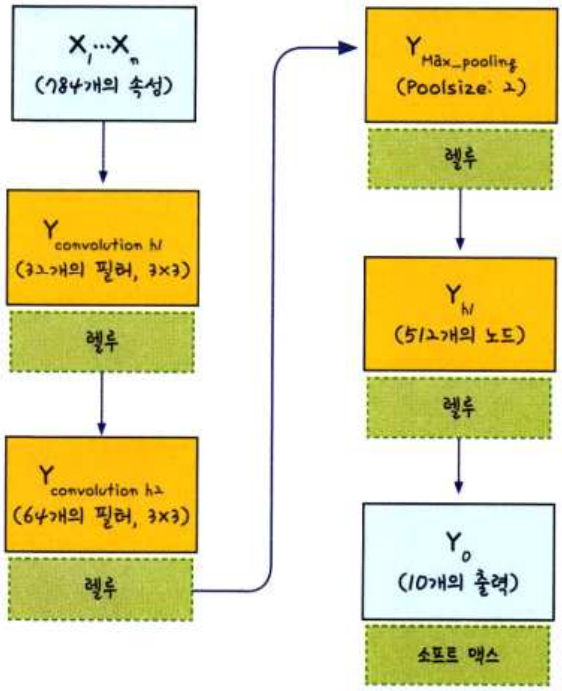
-> 해석: 이미지의 경우 2차원의 평면(높이 h, 너비 w)이며, 픽셀로 구성되어 있어 $\sum$를 이용해 나타낼 수 있으며, 한 함수가 다른 함수 위를 이동하면서 원소별 곱셈의 합을 계산하는 연산이다.
(원소별로 곱셈을 한 다음 모두 더해주면 된다.)



5. 맥스 풀링

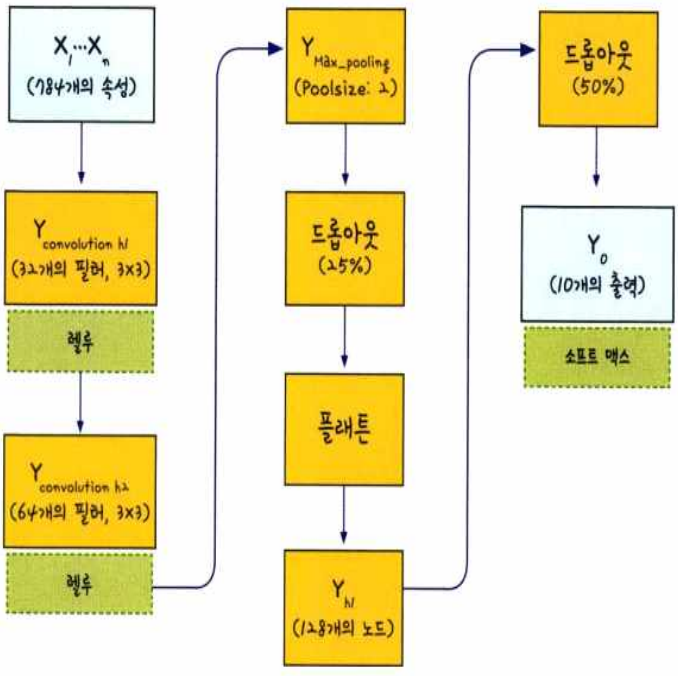
풀링(pooling) 또는 **서브 샘플링(sub sampling)** - 컨볼루션 층을 통해 이미지 특징의 결과가 여전히 크고 복잡해 다시 한번 축소하는 과정

맥스 풀링(max pooling) - 정해진 구역 안에서 최댓값을 뽑아내는 것
평균 풀링(average pooling) - 정해진 구역 안에서 평균값을 뽑아내는 것



드롭 아웃 - 과적합을 효과적으로 피해가는 방법으로 은닉층에 배치된 노드 중 일부를 임의로 꺼주는 것이다.

플래튼(flatten) - 컨볼루션 층이나 맥스 풀링은 주어진 이미지를 2차원 배열인 해로 다루기 때문에 Dense와 같이 분류를 위한 학습 레이어에서는 1차원 데이터로 바꿔서 학습이 되어야 함.



6. 컨볼루션 신경망 실행하기
(실습코드는 github 참고)

참고 자료
https://www.delftstack.com/ko/api/numpy/python-numpy-numpy.shape-function/

https://pwnbit.kr/37

https://yganalyst.github.io/data_handling/memo_5/

<CNN 합성곱 식 및 계산 참고 자료> - (https://excelsior-cjh.tistory.com/180)

https://davinci-ai.tistory.com/29