

인공지능! 체험과 실습을 통한 이해

건양대학교 박 헌 규 교수

010-5084-8123 / ingenium@konyang.ac.kr

대학연계 참학력 공동교육과정 (23.7.25 ~ 8.2)

수업 일정

일차	날짜	차시	시간	수업내용	비고
1	7. 25 (화)	1~3 (3H)	2:30 ~5:20	<ul style="list-style-type: none"> 오리엔테이션 인공지능의 정의, 역사, 종류 인공지능 체험 1 	구글 계정
2	7. 26 (수)				
3	7. 27 (목)	4~7 (4H)	2:00 ~5:50	<ul style="list-style-type: none"> 인공지능, 머신러닝, 딥러닝 관계 이해 머신러닝의 종류 (지도학습, 비지도학습, 강화학습 사례) 인공지능 체험 2 인공지능 실습환경 구축 (구글 코랩 환경 설정) 	구글 계정
4	7. 28 (금)	8~10 (3H)	2:00 ~4:50	<ul style="list-style-type: none"> 인공지능으로 구현한 틱택토 게임 틱택토 게임으로 보는 인공지능 원리 학습 인공지능으로 구현한 오목 게임 인공지능 오목 게임의 원리 	구글 계정
5	7. 31 (월)	11~14 (4H)	2:00 ~5:50	<ul style="list-style-type: none"> 인공지능 바둑 "알파고" 구현 원리 이해 머신러닝 지도 학습의 종류 (분류, 회귀) 구글 코랩을 이용한 인공지능 지도학습 실습 학습한 모델을 통해 새로운 데이터의 예측 	구글 계정
6	8. 2 (수)	15~17 (3H)	2:00 ~4:50	<ul style="list-style-type: none"> 내가 쓴 손글씨 숫자 인식시키기 (이미지 인식) 구글 코랩을 이용한 MNIST 이미지 인식 실습 이미지를 인식하는 인공지능(CNN) 학습 	구글 계정

6일차

- 내가 쓴 손글씨 숫자 인식시키기 (이미지 인식)
- 구글 코랩을 이용한 MNIST 이미지 인식 실습
- 이미지를 인식하는 인공지능 (CNN) 학습

6일차 1교시

이미지를 인식하는 인공지능 (CNN) 학습

이미지 인식 (CNN 방법)

딥러닝이 어떤 방법으로 이미지를 인식하는지 알아보시다.

Preview

- 컨볼루션 신경망은 딥러닝에서 가장 성공한 모델



- 컴퓨터 비전은 인공지능의 가장 중요한 연구 분야 중 하나
 - 시각은 인간의 가장 강력한 인지 기능. 컴퓨터 비전은 인간의 시각 기능 모방
 - 딥러닝 이후 컴퓨터 비전 연구 패러다임 변화: 수작업 규칙 기반 → 대용량 영상 데이터로 기계 학습
 - 컴퓨터 비전(문제)과 딥러닝(해결 도구)은 상호작용을 통해 공진화

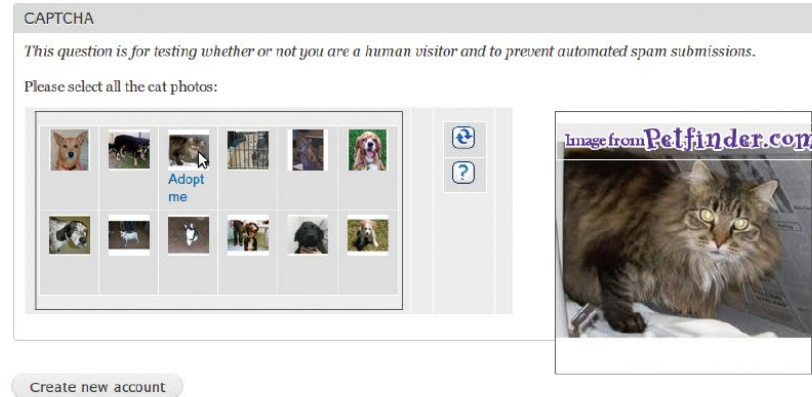
6.1 컨볼루션 신경망의 동기와 전개

■ 개와 고양이 사진을 구별하는 문제

- 예전 컴퓨터 비전에서는 매우 어려운 문제여서 캡차로 사용
 - 랜덤하게 찍는 경우 $0.5^{12}=1/4096$ 확률로 뚫림(12개 영상 사용하는 경우)
 - 고전적인 컴퓨터 비전 기술로 82.7% 정확률 달성. $0.827^{12}=1/9.77$ 확률로 뚫림



(a) 문자 인식을 이용한 캡차



(b) 개와 고양이 인식을 이용한 캡차

그림 6-3 해킹 방지를 위해 사용하는 캡차

■ 컨볼루션 신경망

- 98.914% 달성. $0.98914^{12}=0.87719$ 확률로 뚫림
- 캡차로서 기능 상실

■ 컨볼루션_{convolution} 연산 : 특징 맵 = 신호 x 커널(필터)



외곽 주변의
정보의 손실이
발생함
⇒ 패딩 필요

그림 6-4 컨볼루션 연산

특징 맵 계산 식 : 신호 x 커널

$$2 \times -1 + 9 \times 0 + 9 \times 1$$

$$= -2 + 0 + 9$$

$$= 7$$

※ 행렬 계산이 아닌 각 셀별 매칭되는 숫자끼리 곱셈

2	2	9
2	2	9
2	2	9

 \otimes

-1	0	1
-1	0	1
-1	0	1

 $=$

-2	0	9
-2	0	9
-2	0	9

 \Rightarrow

$$(-2 + 0 + 9) +$$

$$(-2 + 0 + 9) +$$

$$(-2 + 0 + 9)$$

$$= 7 + 7 + 7$$

$$= 21$$

6.2.1 컨볼루션 연산으로 특징 맵 추출

■ 2차원 컨볼루션

$$f(j, i) = z(j, i) \otimes u = \sum_{x=-(h-1)/2}^{(h-1)/2} \sum_{y=-(h-1)/2}^{(h-1)/2} z(j+y, i+x) u(y, x) \quad (6.2)$$

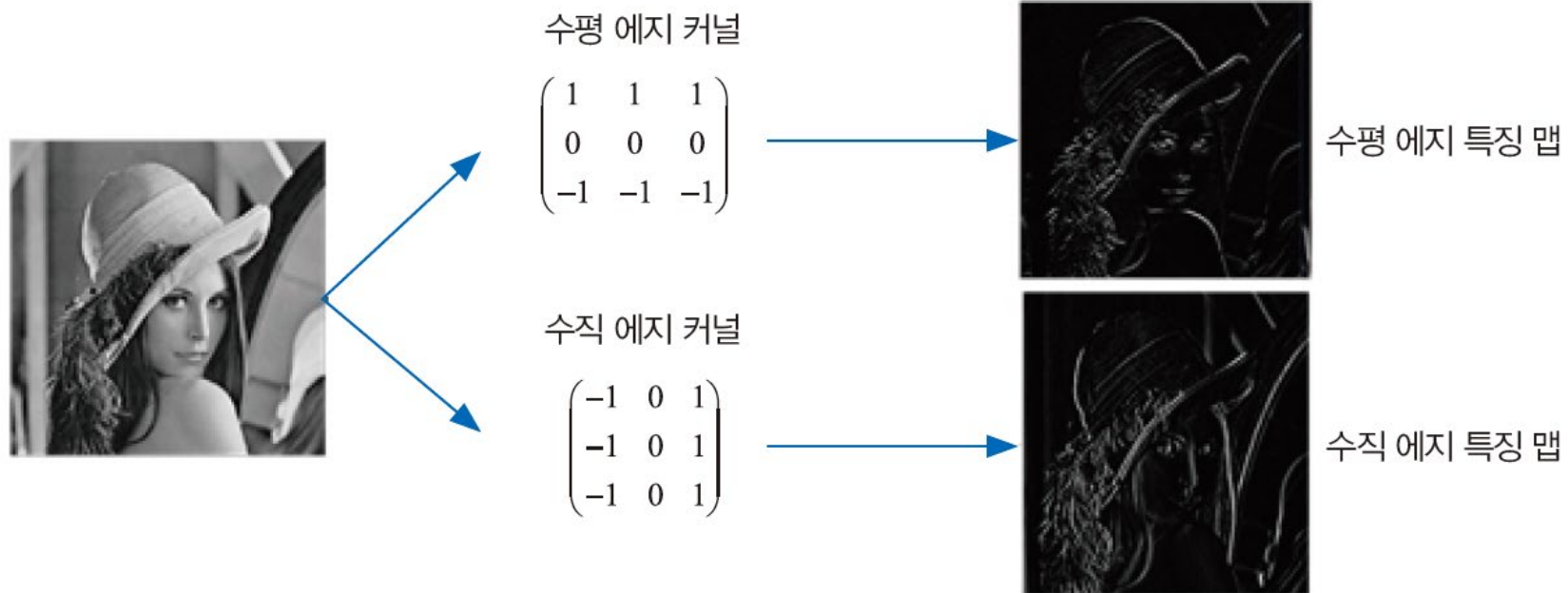


그림 6-5 에지 검출(수평 에지와 수직 에지 특징 맵)

6.2.2 컨볼루션층과 풀링층

풀링층

- 데이터의 차원을 감소시켜 신경망의 계산효율을 높임 (요약을 하는 효과)
- 최대 풀링(max pooling)은 커널 안에 있는 화소 중에 최대값을 취함 (평균 풀링(average pooling)은 평균값 취함)
- 보폭(stride)을 s 로 하면 특징 맵은 s 배만큼 줄어듦 (등성등성 보겠다는 의미)

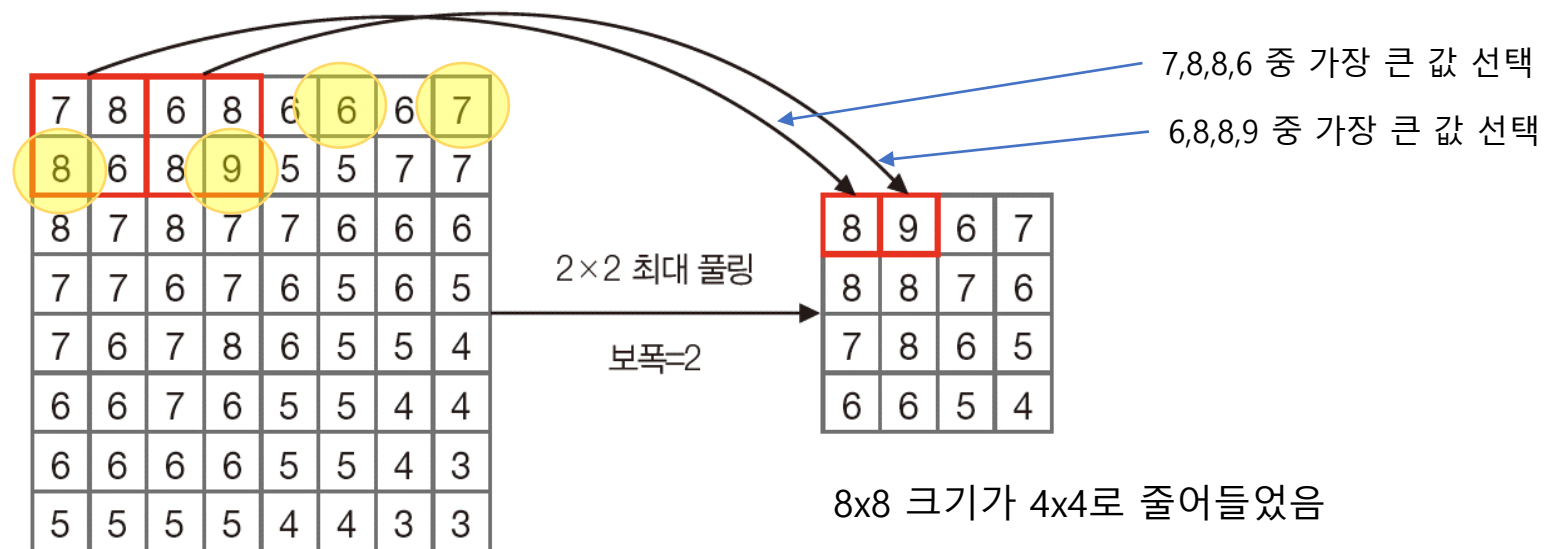


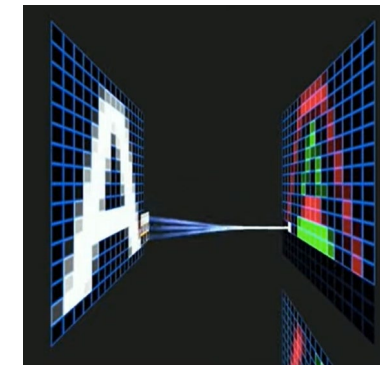
그림 6-8 최대 풀링(보폭이 2인 경우)

6.2.3 빌딩 블록을 쌓아 만드는 컨볼루션 신경망

참고 동영상

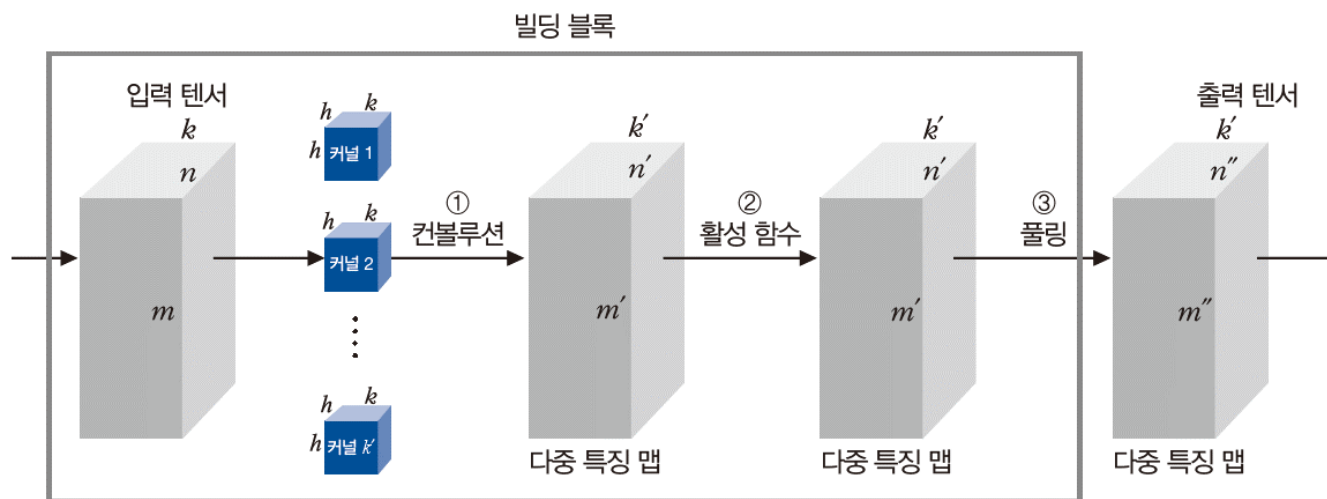
(1분43초)

Convolutional Neural Network Visualization

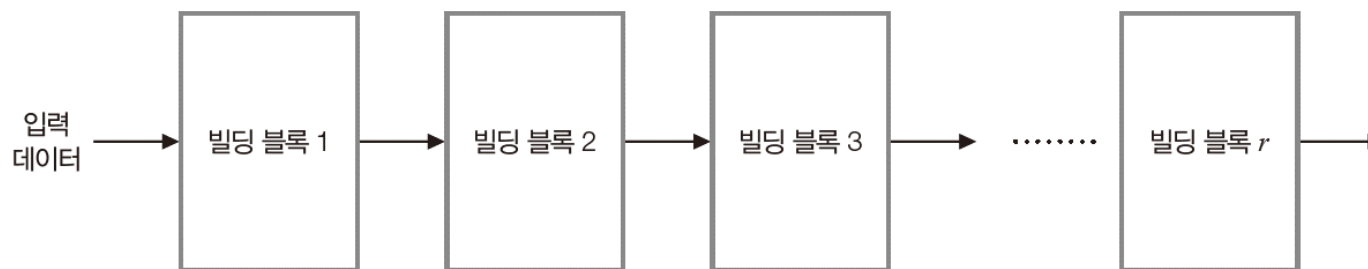


■ 빌딩 블록

- $k*m*n$ 입력 텐서에 k' 개의 커널을 적용하면 $k'*m'*n'$ 크기의 특징 맵이 됨
- 풀링을 적용하면 $k'*m''*n''$ 특징 맵이 됨. 이 텐서는 다음 빌딩 블록의 입력이 됨



(a) 빌딩 블록의 구조



(b) 빌딩 블록을 쌓아 만든 컨볼루션 신경망

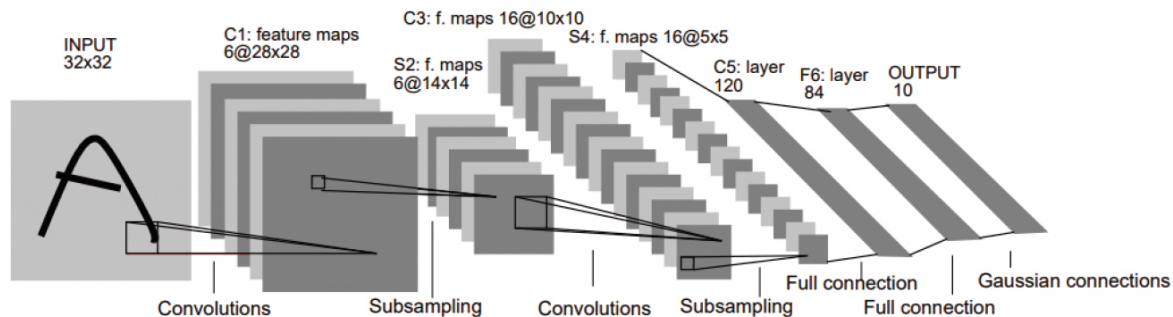
그림 6-10 컨볼루션 신경망의 구조

6.2.3 빌딩 블록을 쌓아 만드는 컨볼루션 신경망

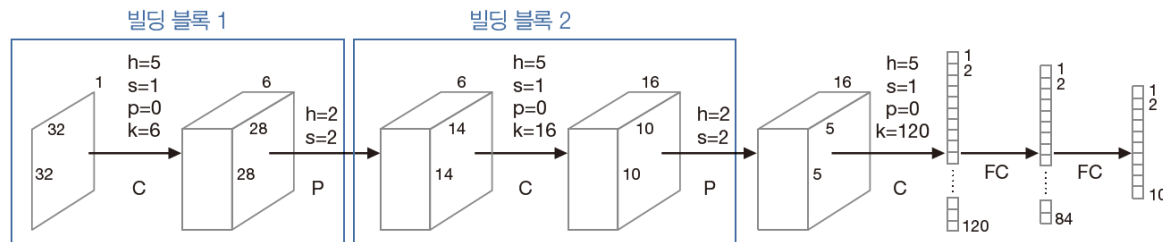
■ LeNet-5 사례 [LeCun1998]

- 현재 기준으로 보면 아주 간단한 컨볼루션 신경망(C-P-C-P-C-FC-FC)
- 입력은 $1 \times 32 \times 32$ 텐서 (32×32 크기의 필기 숫자 맵)
- 빌딩 블록1
 - 컨볼루션은 5×5 커널을 6개 사용 (보폭 1, 덧대기_{padding} 안함) : $1 \times 32 \times 32$ 텐서 \rightarrow $6 \times 28 \times 28$ 텐서
 - 풀링은 2×2 커널을 사용(보폭 2): $6 \times 28 \times 28$ 텐서 \rightarrow $6 \times 14 \times 14$ 텐서

C: Convolution
P: Pooling
FC: Fully Connected



(a) [LeCun1998]의 그림

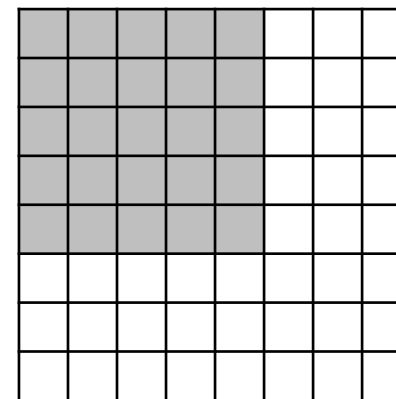


(b) 이 책의 방식에 따른 그림

그림 6-11 LeNet-5의 구조(h는 커널의 크기, s는 보폭, p는 덧대기, k는 커널 개수)

32×32 크기의 배열은
패딩을 안할 경우
 5×5 커널을 사용하면
 $32 - (5 - 1) = 28$ 로 줄어들게 됨


8×8 크기 배열은
 5×5 커널 적용시 4×4 남음

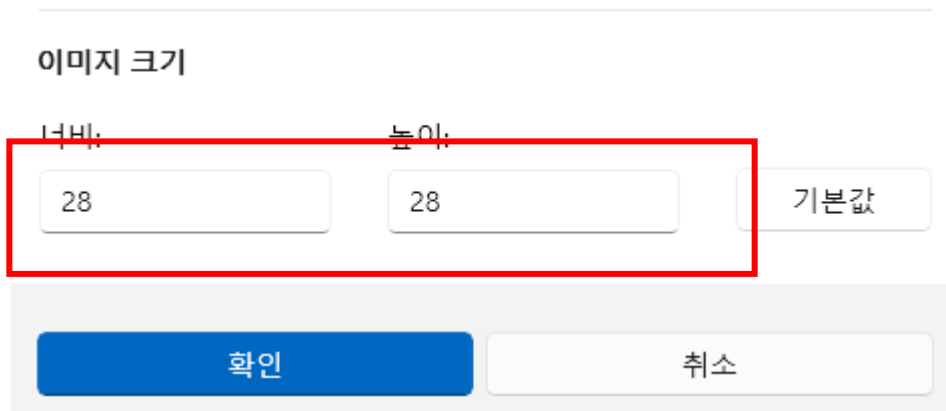
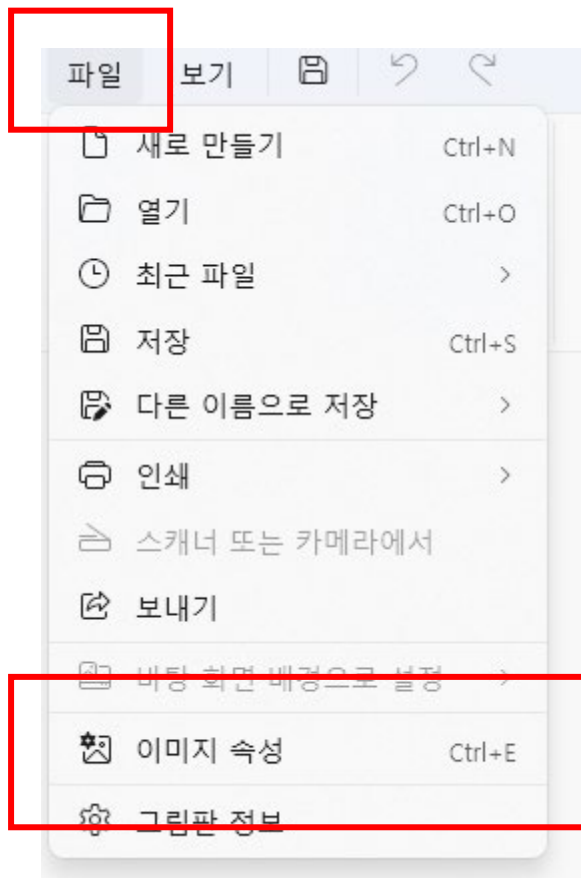


6일차 2,3교시

내가 쓴 손글씨 숫자 인식시키기 (이미지 인식)

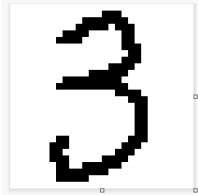
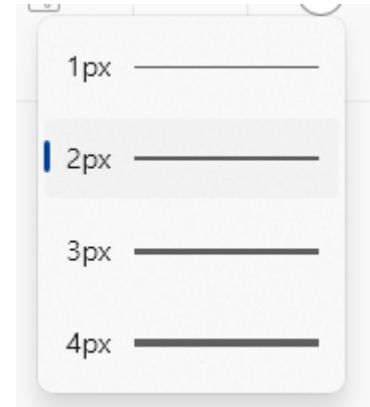
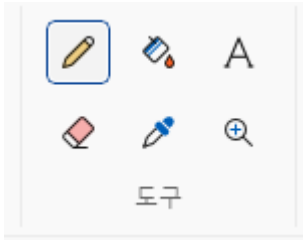
손글씨 직접 만들기 (준비작업)

- 그림판 열기  (검색: 그림판)
- 파일 – 이미지 속성 – 이미지 크기 28로 입력 - 확인



손글씨 직접 만들기 (파일 저장)

- 화면크게하기 : Ctrl 키 누른 상태에서 마우스 휠 조작
- 도구 - 연필 선택 / 크기 - 2px 선택



- 숫자 쓰기
- 파일 - 저장 - (다운로드 폴더) '본인이름손글씨3' (png 파일)

MNIST 인식 학습

MNIST (Modified National Institute of Standards and Technology database)

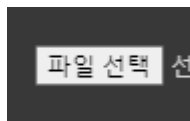
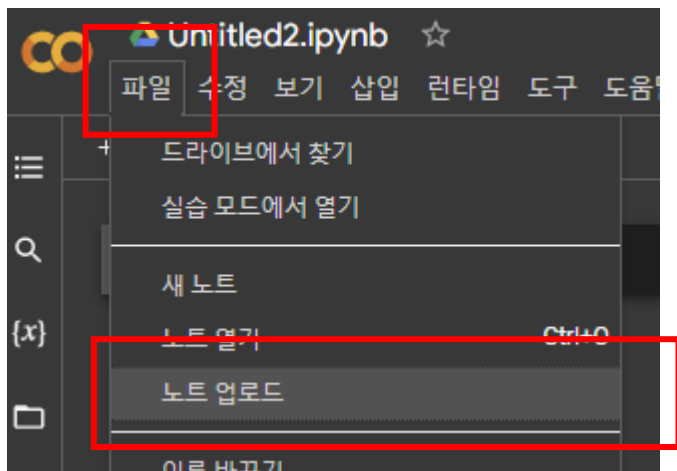
: 손으로 쓴 글자들로 이루어진 대형 데이터 베이스

LMS 소스코드 실행


- 소스코드 저장된 텍스트 파일과 ipynb 파일 다운로드
- 3가지 텍스트 문서와 소스코드가 있음
 - mnist_3dense 소스코드.txt / mnist_3dense_학생용.ipynb
 - mnist_CNN 소스코드.txt / mnist_CNN_학생용.ipynb
 - mnist_LeNet5 소스코드.txt / mnist_LeNet5_학생용.ipynb
- 텍스트 파일은 메모장에서 열기 (부분부분 copy 후 paste)
- 구글 코랩에서 ipynb 파일 순서대로 열면서 실행하기

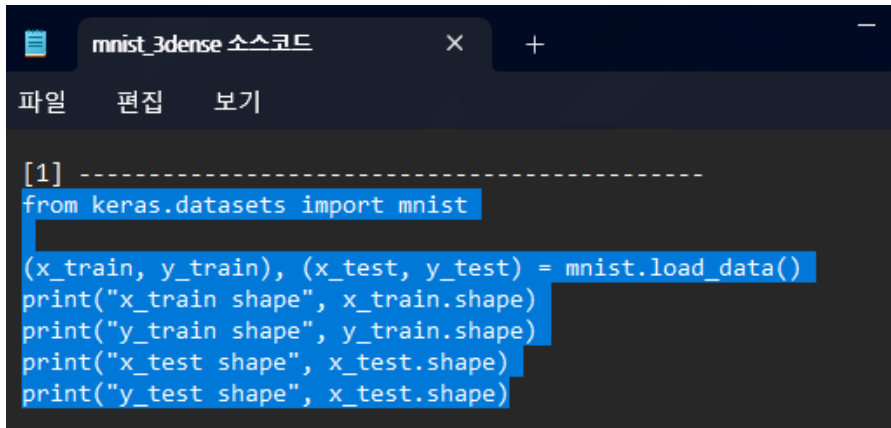
구글 코랩에서 소스코드 불러오기

- 구글 코랩 - 새노트 - 파일 - 노트업로드 - (열린 창에서) 파일선택 click - (다운 받은 폴더로 가서 해당 파일 클릭) 열기 - '업로드 중' - 아래 화면 열리면 성공

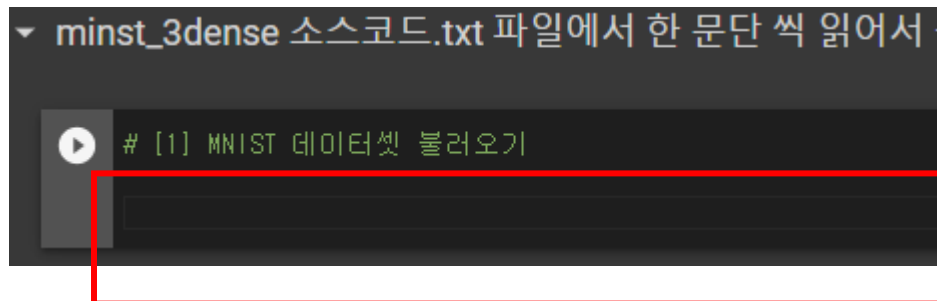


코드를 하나씩 복사하여 붙여넣기

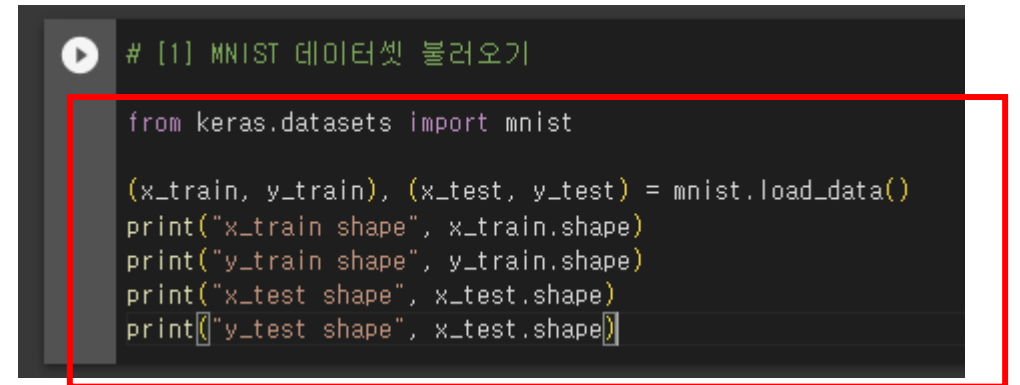
- 메모장에서 소스코드 열기 (mnist_3dense 소스코드.txt)
- [1] 등 소스코드 일부를 복사하여 구글 코랩에 붙여넣기
- 코드 하나씩 실행 ( 또는 Ctrl+Enter)



```
[1] -----  
from keras.datasets import mnist  
  
(x_train, y_train), (x_test, y_test) = mnist.load_data()  
print("x_train shape", x_train.shape)  
print("y_train shape", y_train.shape)  
print("x_test shape", x_test.shape)  
print("y_test shape", x_test.shape)
```



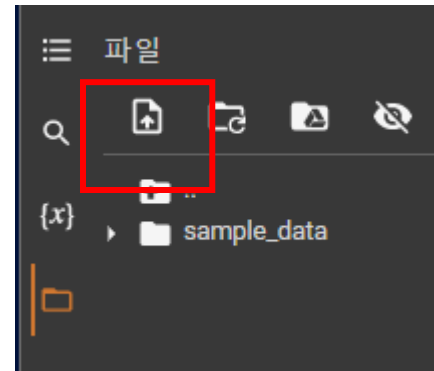
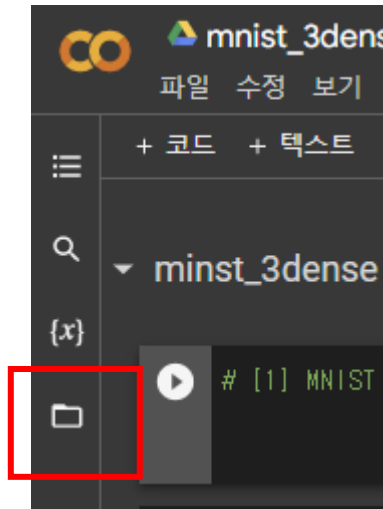
```
mnist_3dense 소스코드.txt 파일에서 한 문단 씩 읽어서  
  
# [1] MNIST 데이터셋 불러오기  
  
[1] -----  
from keras.datasets import mnist  
  
(x_train, y_train), (x_test, y_test) = mnist.load_data()  
print("x_train shape", x_train.shape)  
print("y_train shape", y_train.shape)  
print("x_test shape", x_test.shape)  
print("y_test shape", x_test.shape)
```



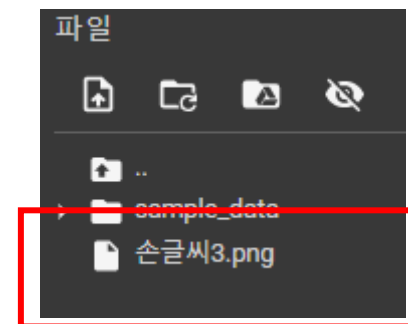
```
# [1] MNIST 데이터셋 불러오기  
  
from keras.datasets import mnist  
  
(x_train, y_train), (x_test, y_test) = mnist.load_data()  
print("x_train shape", x_train.shape)  
print("y_train shape", y_train.shape)  
print("x_test shape", x_test.shape)  
print("y_test shape", x_test.shape)
```

손글씨 직접 만들기 (파일 업로드)

- 구글 코랩 파일 선택 - '업로드' 선택 - 파일 선택 - 열기



- 경고 창 나오면 '확인'
- 파일 업로드 확인

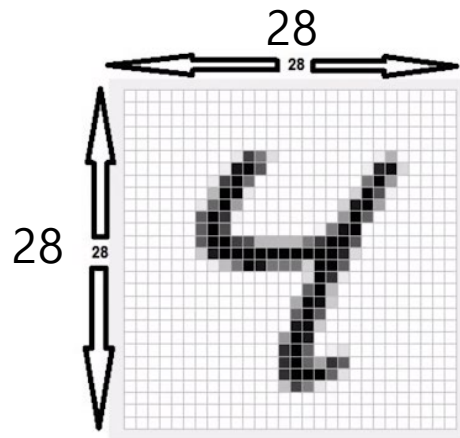


소스코드 실행

- 메모장에서 소스코드 한 단락씩 복사하여 붙여넣기 - 실행
- 본인이 만든 손글씨 이미지 인식
 - [16]번 셀의 파일명 수정
 - `img = Image.open("홍길동글씨3.png").convert("L")`
- 3개 소스코드를 각각 실행하면서 결과 비교하기
 - 3 Dense, CNN, LeNet5

MNIST

- MNIST Modified National Institute of Standards and Technology database
 - 손으로 쓴 숫자들로 이루어진 대형 데이터베이스
 - 다양한 화상처리 시스템을 학습하기 위해 사용됨
 - 60,000개의 학습용 이미지와 10,000개의 테스트 이미지
 - 이미지 크기는 28x28 (=784개의 0~255사이의 픽셀로 구성)



9	0	8	8	1	3	2	5	7	4	1	8	0	3	8	0	0	2	4	/	9	0	9	2	0	5
8	8	8	8	6	5	8	3	3	0	6	4	4	4	8	1	2	6	9	/	6	3	3	0	6	3
4	6	2	9	4	3	6	7	7	4	5	9	2	7	8	0	1	9	\	4	4	8	4	3	9	2
2	7	1	7	8	9	9	5	8	7	4	6	9	4	4	8	8	9	6	6	6	9	4	4	2	1
9	2	0	0	3	6	4	1	2	1	8	6	7	4	3	8	1	9	1	6	3	4	5	6	0	5
1	0	1	2	2	0	6	5	4	6	7	5	2	0	8	1	9	2	8	1	3	7	8	8	8	1

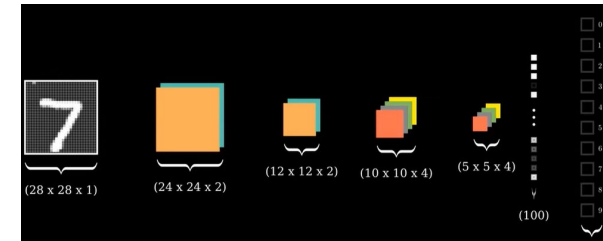
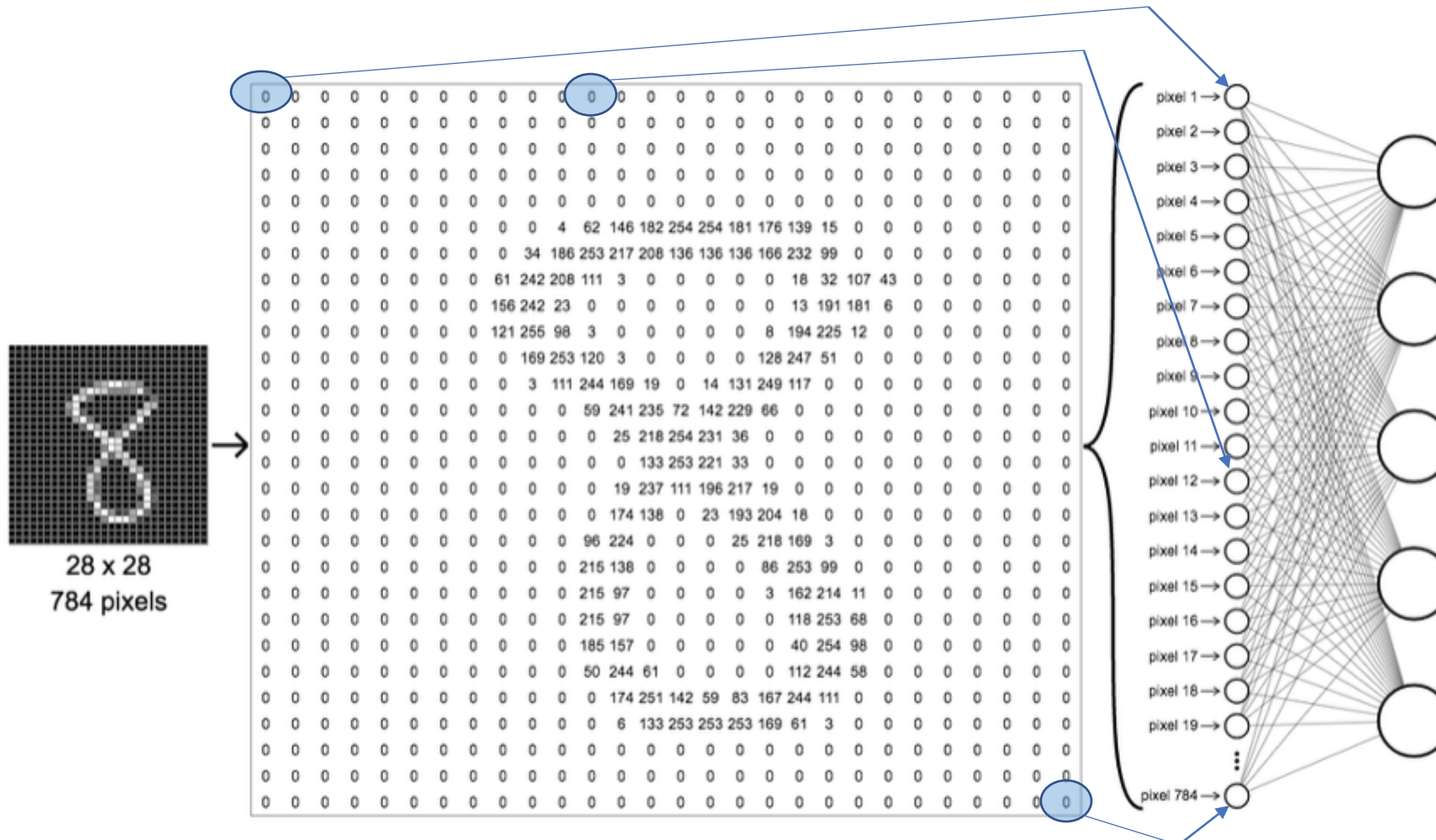
<https://puture.tistory.com/385>

MNIST 이미지 구성 예

참고 동영상

(5분34초)

Visualization convolutional neural networks



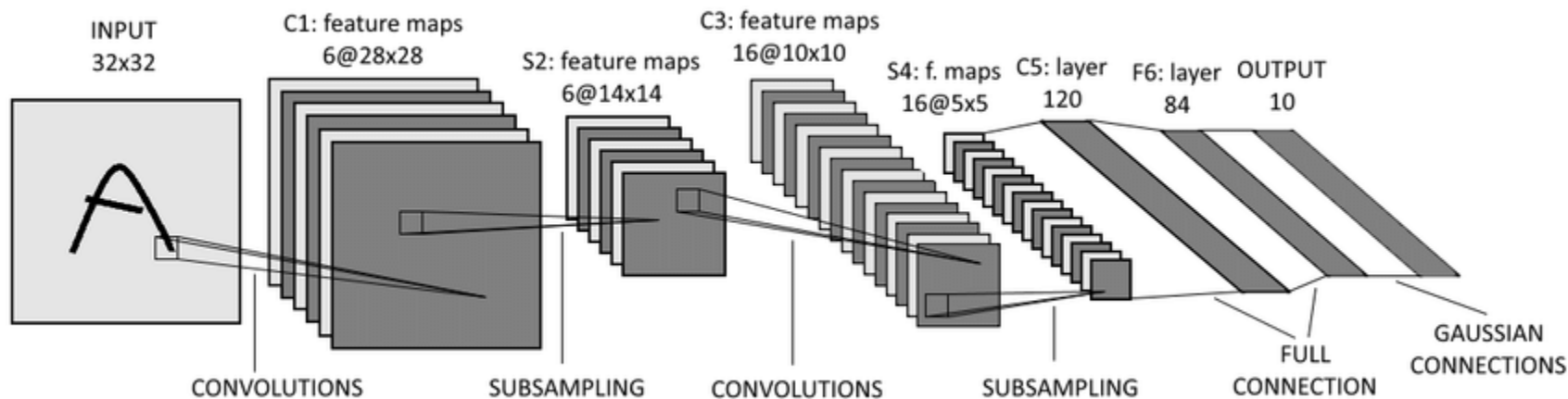
<https://puture.tistory.com/385>

이미지 분석 소스코드 해설

LeNet-5 신경망

- `model = Sequential()`
- `model.add(Conv2D(6,(5,5), padding='same', activation='relu', input_shape=(28,28,1)))`
- `model.add(MaxPooling2D(pool_size=(2,2)))`
- `model.add(Conv2D(16,(5,5), padding='same', activation='relu'))`
- `model.add(MaxPooling2D(pool_size=(2,2)))`
- `model.add(Conv2D(120,(5,5), padding='same', activation='relu'))`
- `model.add(Flatten())`
- `model.add(Dense(84, activation='relu'))`
- `model.add(Dense(10, activation='softmax'))`

C-P-C-P-C-FC-FC 구조의
컨볼루션 신경망 설계
※ C : Convolution
P : Pooling
FC : Fully Connected



6.4.1 필기 숫자 인식

■ LeNet-5 재현

- C-P-C-P-C-FC-FC 구조

```
17 # LeNet-5 신경망 모델 설계
18 cnn=Sequential()
19 cnn.add(Conv2D(6,(5,5),padding='same',activation='relu',input_shape=(28,28,1)))
20 cnn.add(MaxPooling2D(pool_size=(2,2)))
21 cnn.add(Conv2D(16,(5,5),padding='same',activation='relu'))
22 cnn.add(MaxPooling2D(pool_size=(2,2)))
23 cnn.add(Conv2D(120,(5,5),padding='same',activation='relu'))
24 cnn.add(Flatten())
25 cnn.add(Dense(84,activation='relu'))
26 cnn.add(Dense(10,activation='softmax'))
27
28 # 신경망 모델 학습
29 cnn.compile(loss='categorical_crossentropy',optimizer=Adam(),metrics=['accuracy'])
30 hist=cnn.fit(x_train,y_train,batch_size=128,epochs=30,validation_data=(x_test,
    y_test),verbose=2)
31
32 # 신경망 모델 정확률 평가
33 res=cnn.evaluate(x_test,y_test,verbose=0)
34 print("정확률은",res[1]*100)
35
```

C-P-C-P-C-FC-FC 구조의
컨볼루션 신경망 설계

성능 향상을 위해 Adam
옵티마이저 사용

6.4.1 필기 숫자 인식

Train on 60000 samples, validate on 10000 samples

Epoch 1/30

60000/60000 - 29s - loss: 0.2021 - accuracy: 0.9378 - val_loss: 0.0708 - val_accuracy: 0.9772

Epoch 2/30

60000/60000 - 31s - loss: 0.0552 - accuracy: 0.9830 - val_loss: 0.0501 - val_accuracy: 0.9828

Epoch 3/30

60000/60000 - 49s - loss: 0.0385 - accuracy: 0.9879 - val_loss: 0.0417 - val_accuracy: 0.9862

...

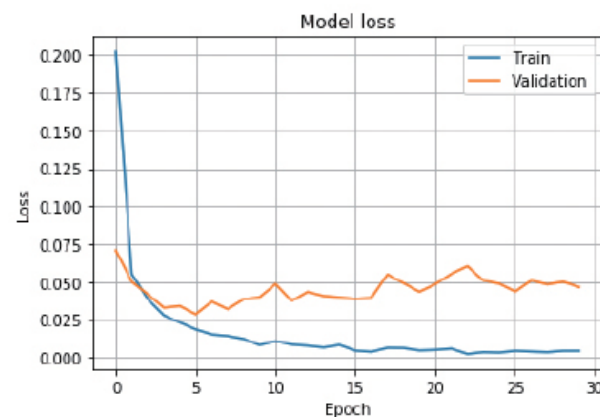
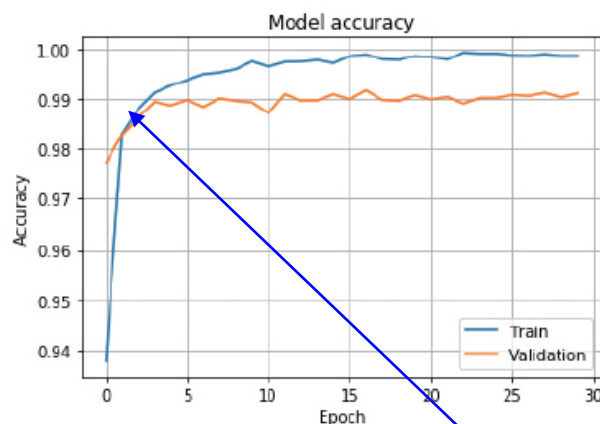
Epoch 29/30

60000/60000 - 30s - loss: 0.0043 - accuracy: 0.9987 - val_loss: 0.0500 - val_accuracy: 0.9903

Epoch 30/30

60000/60000 - 32s - loss: 0.0044 - accuracy: 0.9987 - val_loss: 0.0465 - val_accuracy: 0.9911

정확률은 99.110013256073 ← 99.11% 정확률을 얻어 97.99%의
깊은 다층 퍼셉트론에 비해 1.12% 우수



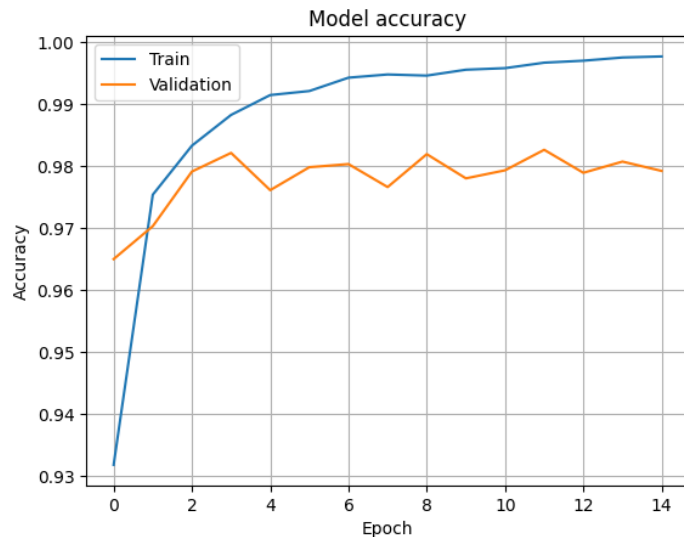
수렴 속도도 빠름

MNIST 손글씨 인식 결과

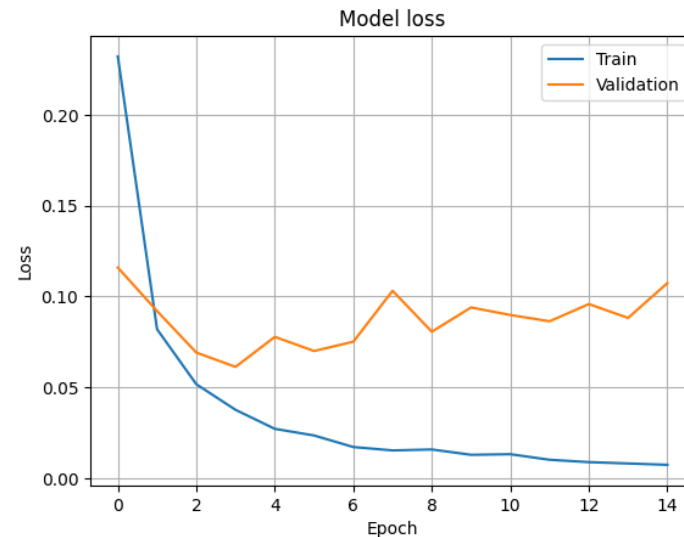
3개의 dense로 구성된 신경망 결과

- model = Sequential()
- model.add(Dense(512, input_dim=784, activation='relu'))
- model.add(Dense(256, activation='relu'))
- model.add(Dense(10, activation='softmax'))

FC-FC-FC 구조의
신경망 설계
※ FC : Fully Connected



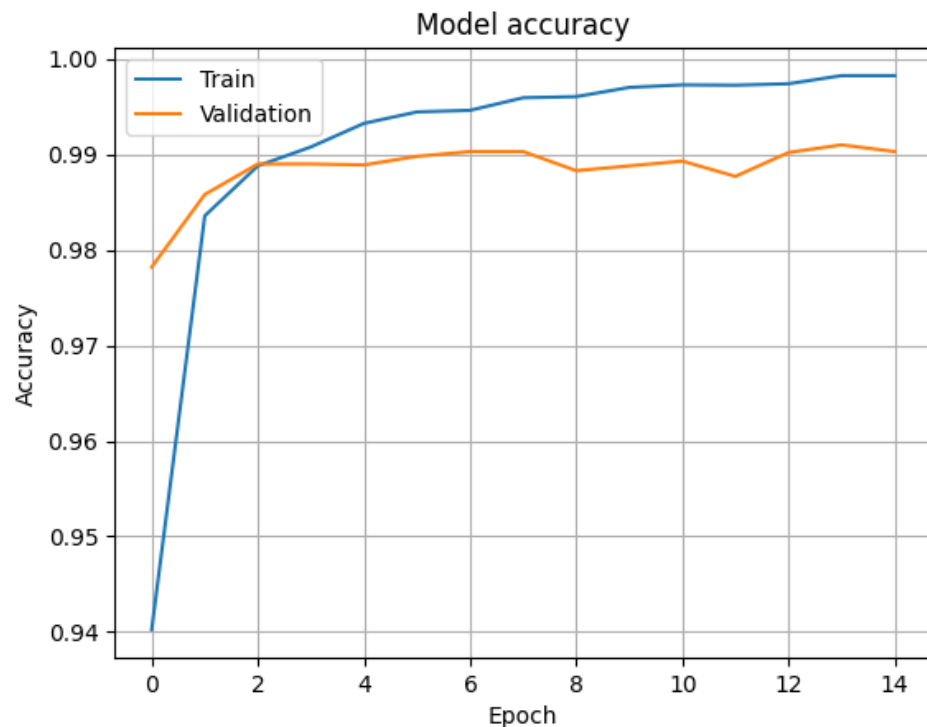
학습시 정확도는 99.8%, 검증결과 98%



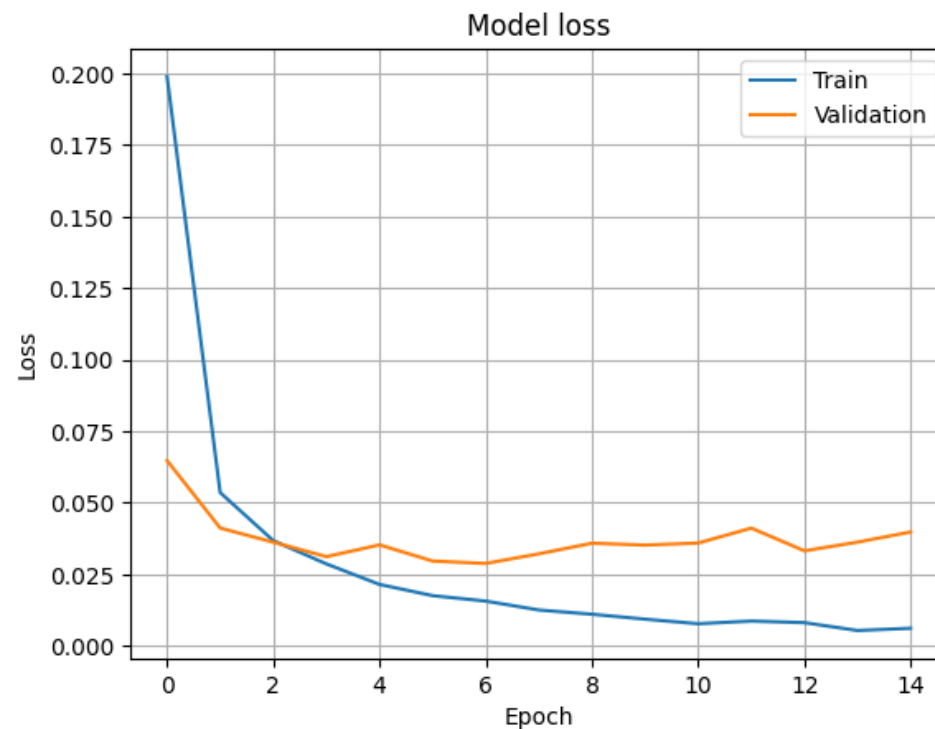
학습시 손실률은 0.01%, 검증결과 0.1%

LeNet-5 적용한 신경망 결과

학습시간 : 20분



학습시 정확도는 99.8%, 검증결과 99%



학습시 손실률은 0.006, 검증결과 0.04

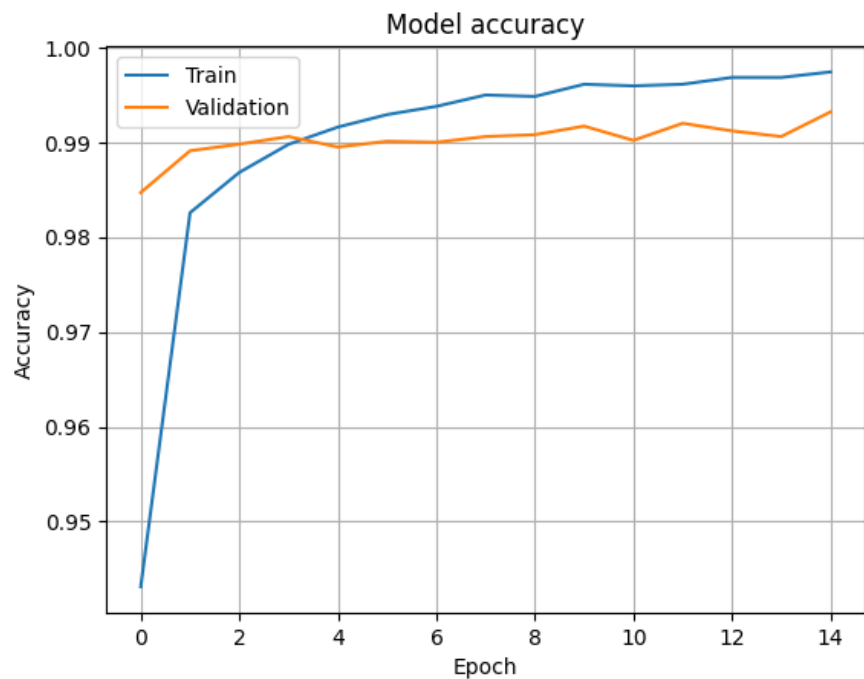
Dropout 적용한 CNN

- `model = Sequential()`
- `model.add(Conv2D(32,(3,3), activation='relu', input_shape=(28,28,1)))`
- `model.add(Conv2D(64,(3,3), activation='relu'))`
- `model.add(MaxPooling2D(pool_size=(2,2)))`
- `model.add(Dropout(0.25))` # Dropout 추가
- `model.add(Flatten())`
- `model.add(Dense(128, activation='relu'))`
- `model.add(Dropout(0.25))` # Dropout 추가
- `model.add(Dense(10, activation='softmax'))`

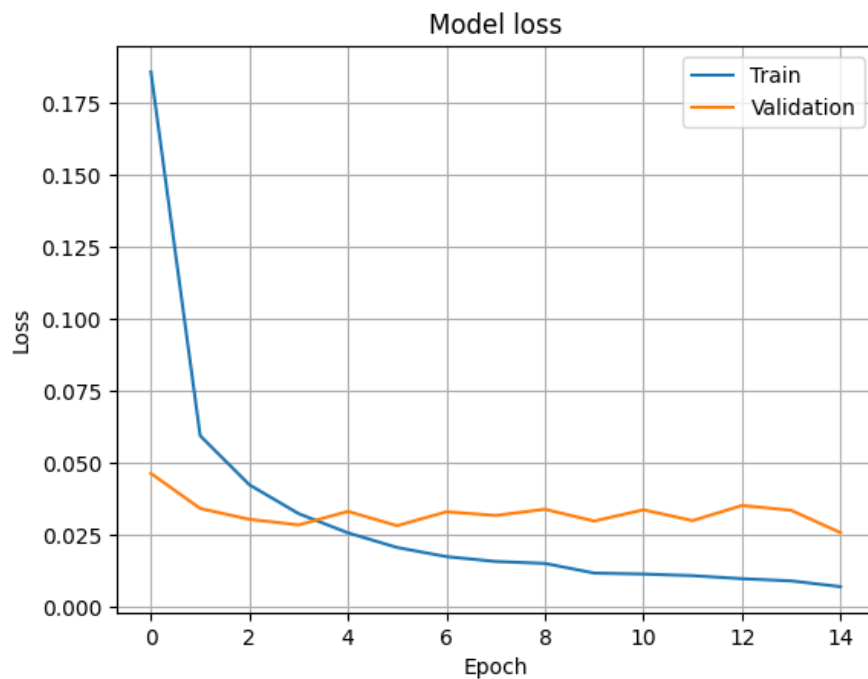
C-C-P-FC-FC 구조의
컨볼루션 신경망 설계
※ C : Convolution
P : Pooling
FC : Fully Connected

Dropout 적용한 CNN 결과

학습시간 : 45분



학습시 정확도는 99.7%, 검증결과 99.3%

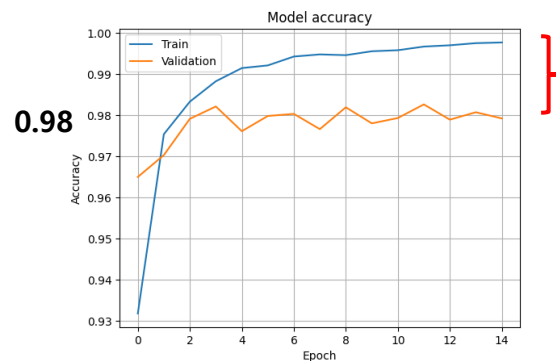


학습시 손실률은 0.007, 검증결과 0.025

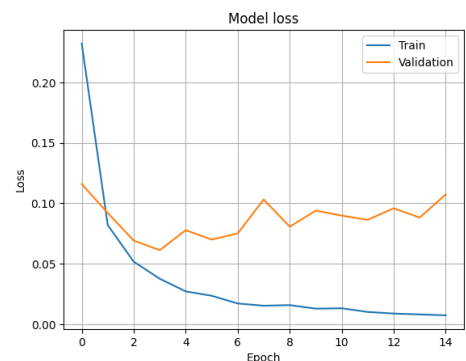
3개 모델 성능 비교

3개의 Dense
FC-FC-FC

정확도

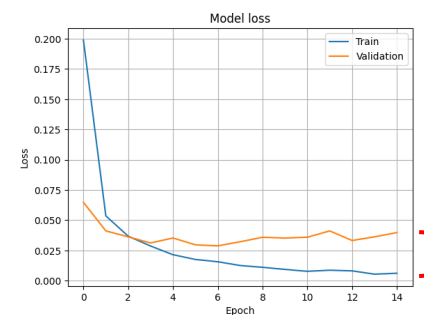
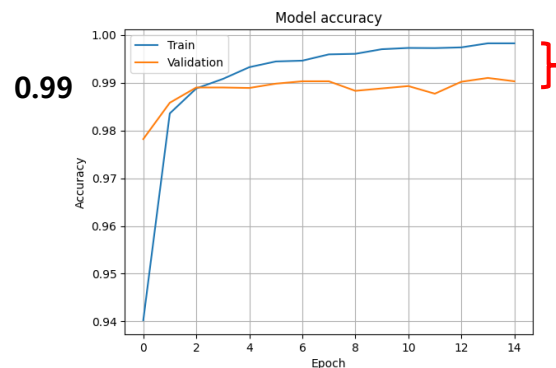


손실함수

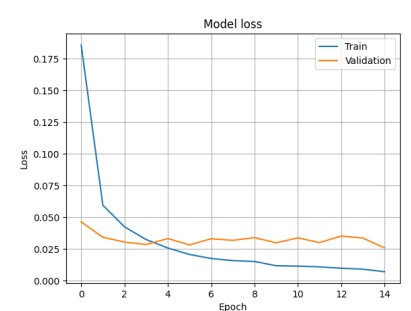
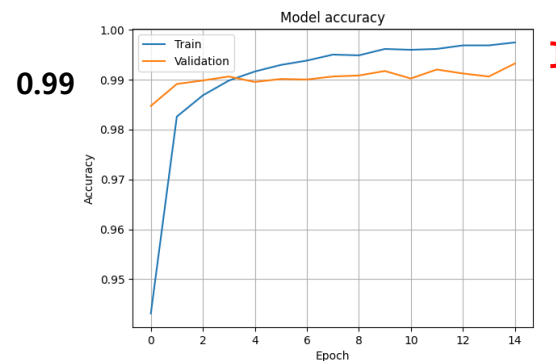


※ y축 값을
비교하기 위해
그림 크기를
축소하였음

LeNet-5
C-P-C-P-C-FC-FC



CNN with Dropout
C-C-P-Dout-FC-Dout-FC



다음 시간

- 5일차 (7.31, 월)
 - 수업 오후 2시에 시작해서 4시간 수업 (2:00~5:50)