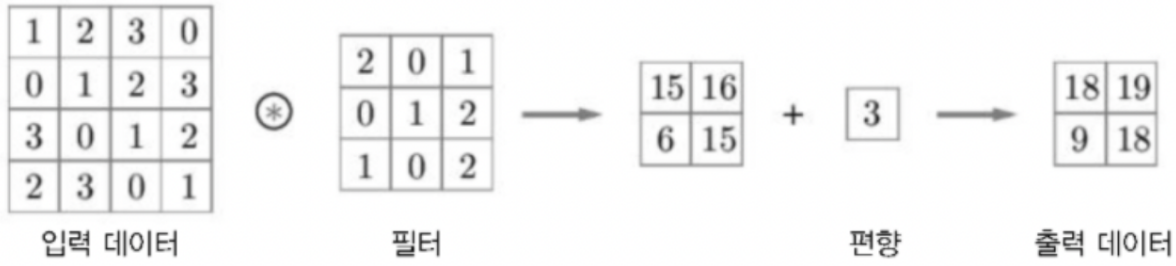
11주차 CNN

참고 사이트

파라미터 분석 : https://sodayeong.tistory.com/142



합성곱 연산의 편향 : 필터를 적용한 원소에 고정값(편향)을 더합니다.

그림과 같이 편향은 필터를 적용한 후의 데이터에 더해집니다.
그리고 편향은 항상 하나(1X1)만 존재합니다.
그 하나의 값을 필터를 적용한 모든 원소에 더하는 것입니다.

https://gaussian37.github.io/dl-keras-number-of-cnn-param/

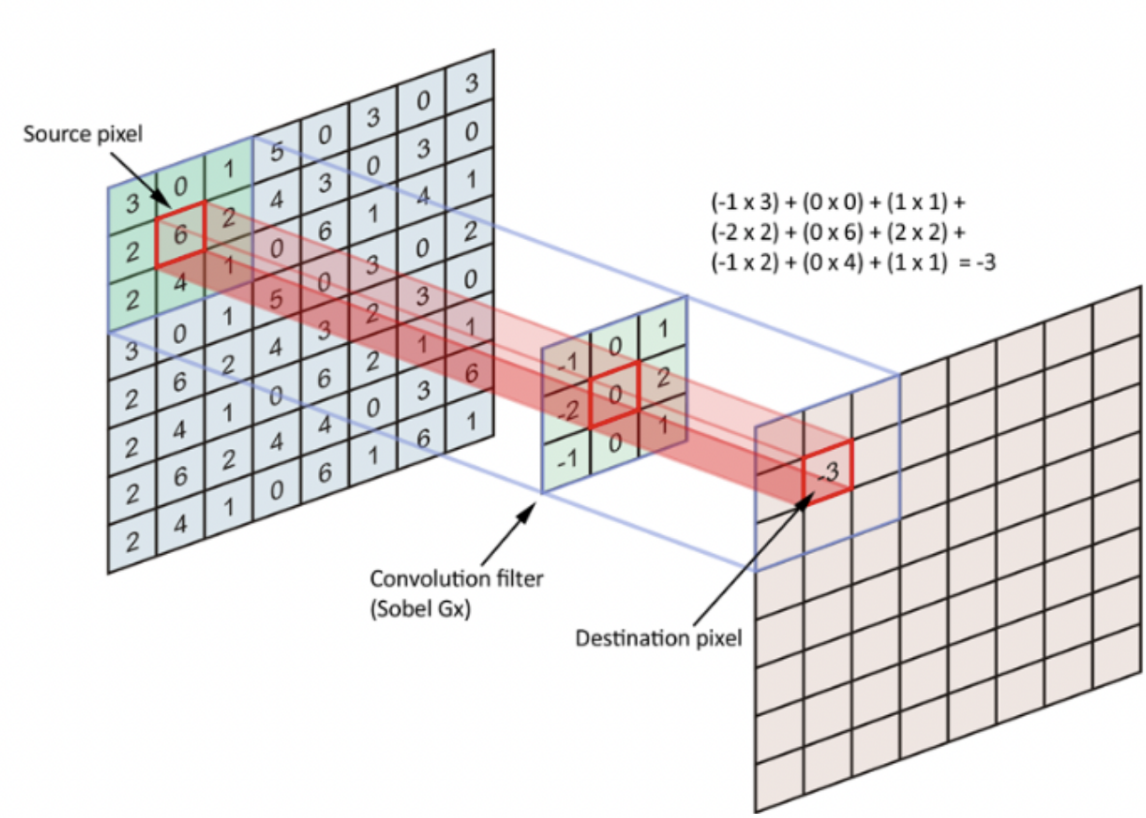
컨볼루션 연산의 이해를 위해

https://jjeongil.tistory.com/544

복잡성 감소 - maxpooling - 차원이 축소되기때문에 입력데이터의 형태가 줄어듦

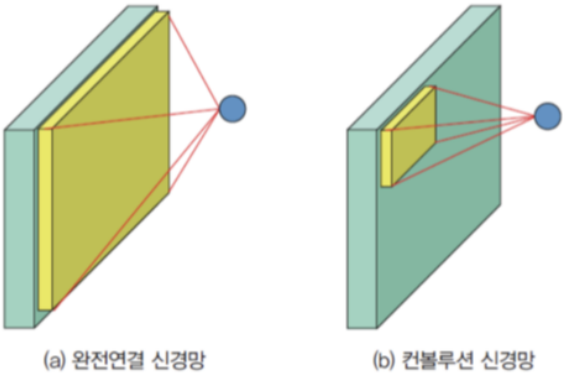
최대값을 뽑아서

CNN은 시세포를 모방해서 만든 딥러닝 기술 -> 2차원 데이터를 그래픽데이터를 바로 쓴다 (flatten으로 펼쳐서 쓰ㅡ지 않음) -그렇기때문에 컨볼루션 + 채널을 받아들일 수 있게 됨 (DNN는 채널이 없어…..) -> 신호처리 디미원..



CNN : convolution neural network : 컨볼루션 신경망 : 신경망에서는 하위 레이어의 노드들과 상위 레이어 노드들이 부분적으로 연결

- Not Full Connected → but, Dropout사용 가능
- 각 layer 입출력 데이터의 형상 유지
- 이미지 공간 정보를 유지하면서 인접 이미지와의 특징을 효과적으로 인식
- 복수의 필터 사용 → feature map : 이미지의 특징 추출 및 학습
- 추출한 이미지 특징을 모으고 강화 → Pooling layer (MaxPooling)
- 필터를 공유 파라미터로 사용 → filter (strides) - 일반 인공 신경망과 비교해서 학습 파라미터가 매우 작음



CNN의 역사

- 네오코그니크론 : 1980 후쿠시마에 의해 소개 (비지도학습)
- CNN : 1989 얀 르쿤 + 지도학습
- Relu,Drop out 제프리 힌튼 → 과적합 해결
- 시각 피질을 참고하여 만듦 - 낮은 계층 → 상위 계층으로 갈 수록 패턴들을 조합하여 복잡한 이미지로 추상화

CNN의 중요성

- 2차원 형태의 입력을 처리 - 이미지 처리에 적합
- 각 Layer에서 일련의 필터가 이미지에 적용됨
    1. Convolution : 특징맵 추출

       정사각형 필터 (2X2) 등 : 상단에서 오른쪽으로 이동 : 포개지는 숫자마다 곱해서 다 더함 == 특성맵

       특성 맵의 숫자가 높으면 그 위치가 필터와 유사

1. Subsampling : 입력의 차원을 줄임 (= Pooling) : 데이터의 크기를 줄임
    a. MaxPooling : 윈도우 안에 있는 숫자에서 가장 큰 값만 추출

풀링 연산

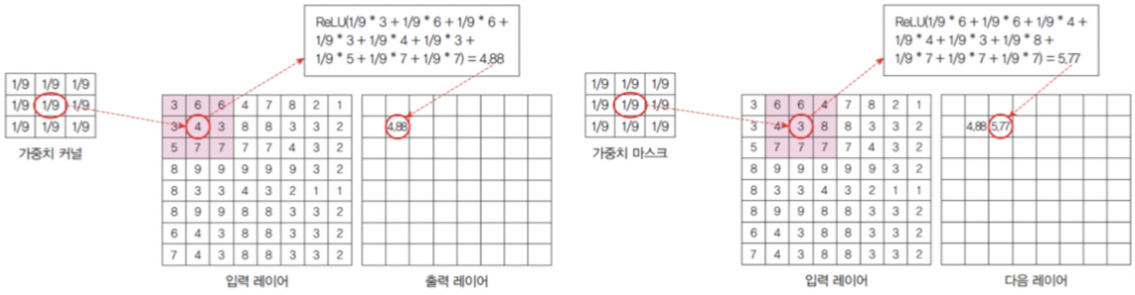
장점 : 레이어 크기 감소

1. 계산이 빨라짐
2. 신경망의 매개변수가 작아짐 - 과적합 가능성 감소
3. 물체의 공간이동에 대해 둔감

b. MeanPooling : 평균 풀링 → 평균값으로 대체

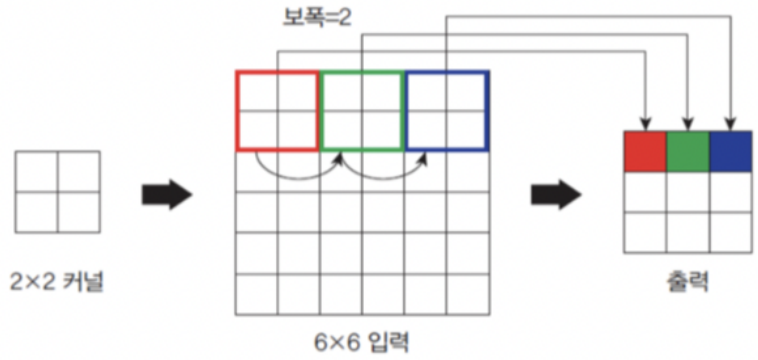
2. Convolution
3. Subsampling
4. Full Connected - > Output - CNN : 필터의 가중치를 학습함 → 미리 고정된 연결이 없으므로 마지막에 연결하여 Output 출력

컨벌루션의 구체적인 예

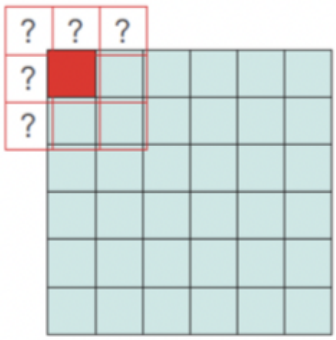


- 컨볼루션 신경망에서의 컨볼루션 연산
  - 커널이 입력층의 각 픽셀을 중심으로 덮여 씌워짐?

        - 앞 레이어 X → W*X + b → activation RELU → Relu(WX+b)
    - 여러개의 필터를 이용 가능 → 미리 정해진 것이 아니고 학습된다 (특징 맵의 갯수 : 필터 갯수 → 필터하나에 특징점 하나!)
- 보폭 : 커널을 적용하는 거리 (default 1 : 1픽셀씩 이동하면서 2라면 2칸씩 이동!)

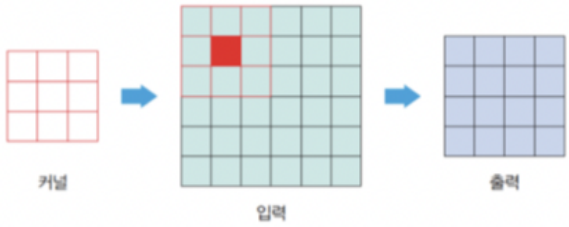


- 패딩 : 이미지의 가장처리를 처리하는 기법

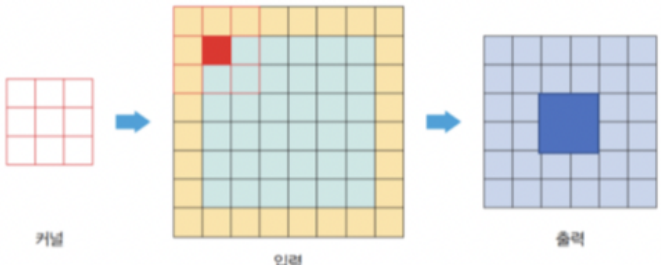


패딩이 필요한 이유

- Valid: 커널을 입력 이미지 안에서만 움직인다.

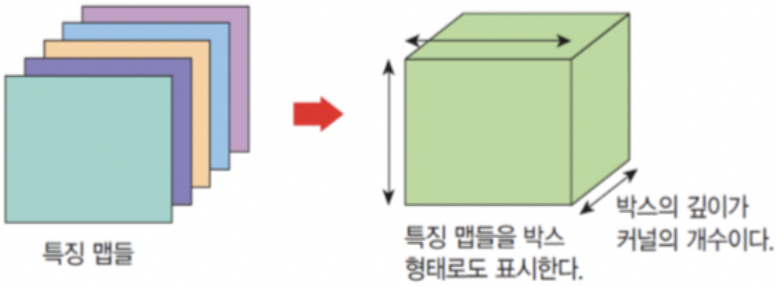


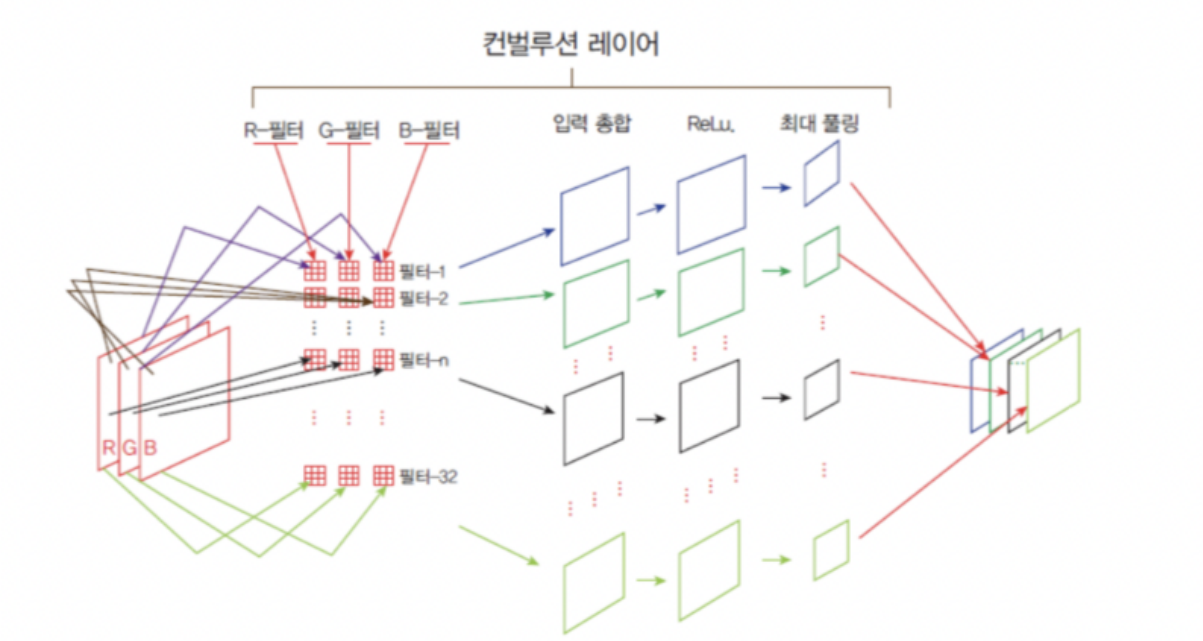
- Same :입력 이미지의 주변을 특정값(예를 들면 0, 또는 이웃 픽셀값)으로 채우는 것



Same padding → 입력과 출력이 동일하게 나옴

필터가 여러 개일 때의 컨벌루션 레이어





컨벌루션 레이어의 분석

```
#Convolution Layer
tf.keras.layers.Conv2D(filters,kernel_size,stridee=(1,1),activation=None,input_shape,padding='same')

filters: 필터 갯수 = 출력층의 갯수
kernel_size: 필터 크기
strieds: 보폭
activaton : 활성화 함수
input_shape: 입력 배열의 형상
padding: default = valid #이미지 가장자리를 처리하기 위해  -> LSTM 컨볼루션 계산하는 과정 참고

tf.keras.MaxPooling2D(pool_size=(2,2),strides=(2,2),padding="valid")

pool_size: 풀링 윈도우의 크기 : 정수 or 정수 2개의 튜플 (2,2)라면 2x2 풀링 윈도우에서 최댓값 추출
strides : 보폭 각 풀링 단계에서 풀링 윈도우가 이동하는 거리
padding: "valid":패딩이 없는 경우, "same":출력과 입력이 동일한 높이 /너비, 치수를 갖도록 균일하게 패딩한다.
```

```
(X_train,y_train),(x_test,y_test)=mnist.load_data()

x_train.shape
(60000, 28, 28)
y_train.shape


X_train=X_train.reshape((-1,28,28,1)) /255
X_test=X_test.reshape((-1,28,28,1)) /255
```

```
inputs=keras.Input(shape=(28,28,1))
Cov2d_01=layers.Conv2D(32,(3,3),activation='relu')(inputs)
MaxPooling2D_01=layers.MaxPooling2D((2,2))(Conv2d_01)

Conv2d_02=layers.Conv2D(64,(3,3),activation='relu')(MaxPooling2D_01)
MaxPooling2D_02=layers.Maxpooling2D((2,2))(Conv2d_02)
Conv2d_03=layers.Con2D(64,(3,3),activation='relu')(MaxPooling2D_02)

flat=layers.Flatten()(Conv2d_03)
Dense_01=layers.Dense(64,activation='relu')(flat)
Dense_02=layers.Dense(10,activation='softmax')(Dense_01)
model=models.Model(iputs=inputs,outputs=Dense_02)
```

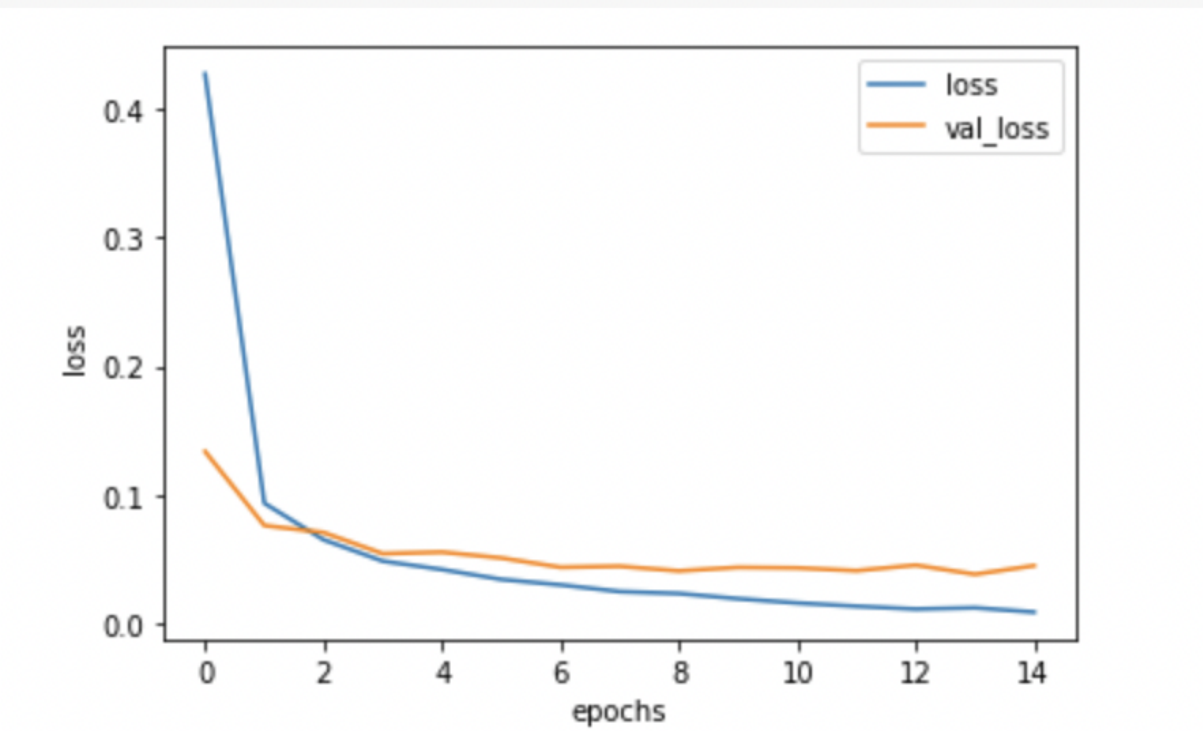
```
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 28, 28, 1)]       0

 conv2d_3 (Conv2D)           (None, 26, 26, 32)        320

 max_pooling2d_2 (MaxPooling  (None, 13, 13, 32)       0
 2D)

 conv2d_4 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_3 (MaxPooling  (None, 5, 5, 64)         0
 2D)

 conv2d_5 (Conv2D)           (None, 3, 3, 64)          36928

 flatten_1 (Flatten)         (None, 576)               0

 dense_2 (Dense)             (None, 64)                36928

 dense_3 (Dense)             (None, 10)                650

=================================================================
Total params: 93,322
Trainable params: 93,322
Non-trainable params: 0
_________________________________________________________________
```

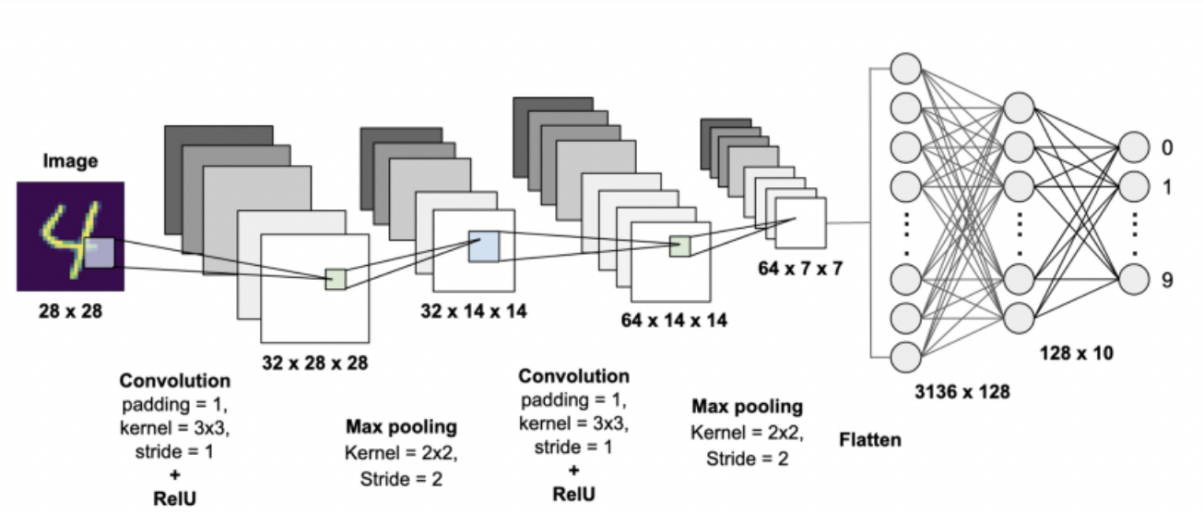
CNN Parameter : 필터(3x3) x 채널 x 출력층 (filter 갯수, 특징 갯수) + 출력층

```
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['accurac
y'])

#autoencoder.compile(optimizer=keras.optimizers.Adam(), loss='mse')

modelpath ="/content/drive/MyDrive/Datasets/{epoch:02d}-{val_accuracy:4f}.hdf5"
checkpointer=keras.callbacks.ModelCheckpoint(filepath=modelpath,verbose=1)

history=model.fit(x_train,y_train,epochs=15,validation_split=0.2,batch_size=256,verbose
=1,callbacks=[checkpointer])
```

```
from google.colab import drive

drive.mount('/content/drive')

import matplotlib.pyplot as plt
import numpy as np

y=history.history['loss']
y1=history.history['val_loss']

x=np.arange(len(history.history['loss']))
#x=np.arange(len(y))

plt.figure(figsize=(6,4))
plt.plot(x,y,label='loss')
plt.plot(x,y1,label='val_loss')

plt.legend()
plt.xlabel('epochs')
plt.ylabel('loss')
plt.show()

model.evaluate(x_test,y_test)
```



CNN 그림으로 구현하기




```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import models
from tensorflow.keras import layers
from tensorflow.keras.datasets import mnist

input=keras.Input(shape=(28,28,1))
Conv2D_1=layers.Conv2D(32,(3,3),stride=1,padding='same')(input)
MaxPooling2D_01=layers.MaxPooling2D((2,2),strides=2)(Conv2D_1)
```

```
Conv2D_2=layers.Conv2D(32,(3,3),stride=1,padding='same')(MaxPooling2D_01)
MaxPooling2D_02=layers.MaxPooling2D((2,2),strides=2)(Conv2D_2)

flatten=layers.Flatten()(MaxPooling2D_02)
Dense_1=layers.Dense(3136,activation='relu')(flat)
Dense_2=layers.Dense(128,activation='relu')(Dense_1)
Output=layers.Dense(10,activation='softmax')(Dense_2)

model=models.Model(inputs=input,outputs=Output)

model.summary()
```

```
[ ] model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 28, 28, 1)]       0

 conv2d (Conv2D)             (None, 28, 28, 32)        320

 max_pooling2d (MaxPooling2D  (None, 14, 14, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 14, 14, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 7, 7, 64)         0
 2D)

 flatten (Flatten)           (None, 3136)              0

 dense (Dense)               (None, 3136)              9837632

 dense_1 (Dense)             (None, 128)               401536

 dense_2 (Dense)             (None, 10)                1290

=================================================================
Total params: 10,259,274
Trainable params: 10,259,274
Non-trainable params: 0
_________________________________________________________________
```

Model: "model"

Layer (type) Output Shape Param

input_1 (InputLayer) [(None, 28, 28, 1)] 0

conv2d (Conv2D) (None, 28, 28, 32) **320**

max_pooling2d (MaxPooling2D (None, 14, 14, 32) 0
)

conv2d_1 (Conv2D) (None, 14, 14, 64) 18496

max_pooling2d_1 (MaxPooling (None, 7, 7, 64) 0
2D)

flatten (Flatten) (None, 3136) 0

dense (Dense) (None, 3136) 9837632

dense_1 (Dense) (None, 128) 401536

dense_2 (Dense) (None, 10) 1290

===

Total params: 10,259,274
Trainable params: 10,259,274
Non-trainable params: 0

```
(X_tran,y_train),(x_test,y_test)=mnist.load_data()

x_train=x_train.reshape((-1,28,28,1)) /255
x_test=x_test.reshape((-1,28,28,1)) /255

model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrices=['accura
cy'])
model.fit(x_train,y_train,epochs=19,validation_split=0.2)
model.evaluate()
```

https://sunway-light.tistory.com/m/entry/CNN-파라미터-수-계산-방법

https://gaussian37.github.io/dl-keras-number-of-cnn-param/