딥러닝 이해하기

leejeyeol92@gmail.com

Image data

Data

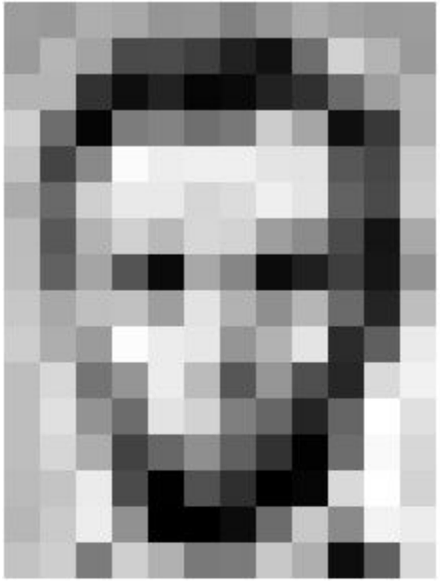
Microsoft Excel - input_resp_data.xls

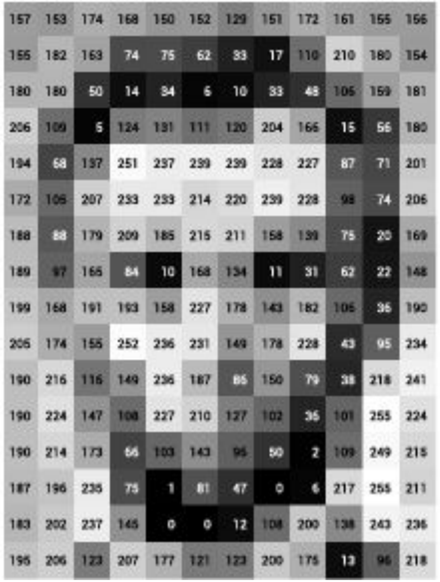
	A	B	C	D	E	F	G
1	Time	InputAil	InputEle	InputRud	RespAil	RespEle	RespRud
2	0	0.00E+00	2.8827	-0.0004868	0	0	0
3	0	0.00E+00	2.8827	-0.0004868	0	0	0
4	0	0.00E+00	2.8827	-0.0004868	0	0	0
5	0.00E+00	0.00E+00	2.8827	-0.0004868	0	0.00E+00	0.00E+00
6	0.00E+00	0.00E+00	2.8827	-0.0004868	0.00E+00	0.00E+00	0.00E+00
7	0.00E+00	0.00E+00	2.8828	-0.0004868	0.00E+00	0.00E+00	0.00E+00
8	0.00E+00	0.00E+00	2.8832	-0.0004868	0.00E+00	0.00E+00	0.00E+00
9	0.00E+00	0.00E+00	2.8853	-0.0004873	0.00E+00	0.00E+00	0.00E+00
10	0.000141	0.00E+00	2.8955	-0.0004995	0.00E+00	0.00E+00	0.00E+00
11	0.000358	0	2.9154	-0.0005692	0.00E+00	0.00035612	0.00E+00
12	0.000358	0	2.9154	-0.0005692	0.00E+00	0.00035612	0.00E+00
13	0.000358	0	2.9154	-0.0005692	0.00E+00	0.00035612	0.00E+00
14	0.000876	0.00E+00	2.9628	-0.0009769	0	0.0021199	0.00E+00
15	0.000876	0.00E+00	2.9628	-0.0009769	0	0.0021199	0.00E+00
16	0.000876	0.00E+00	2.9628	-0.0009769	0	0.0021199	0.00E+00
17	0.000876	0.00E+00	2.9628	-0.0009769	0	0.0021199	0.00E+00
18	0.000876	0.00E+00	2.9628	-0.0009769	0	0.0021199	0.00E+00

Sheet1 / Sheet2 / Sheet3

Ready

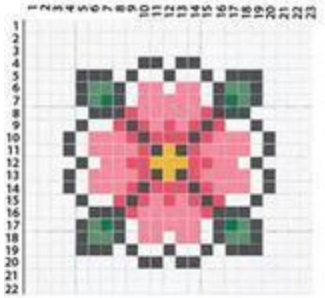
image data

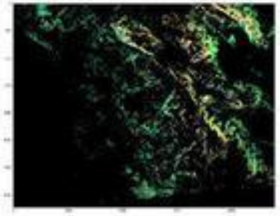




157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

color image





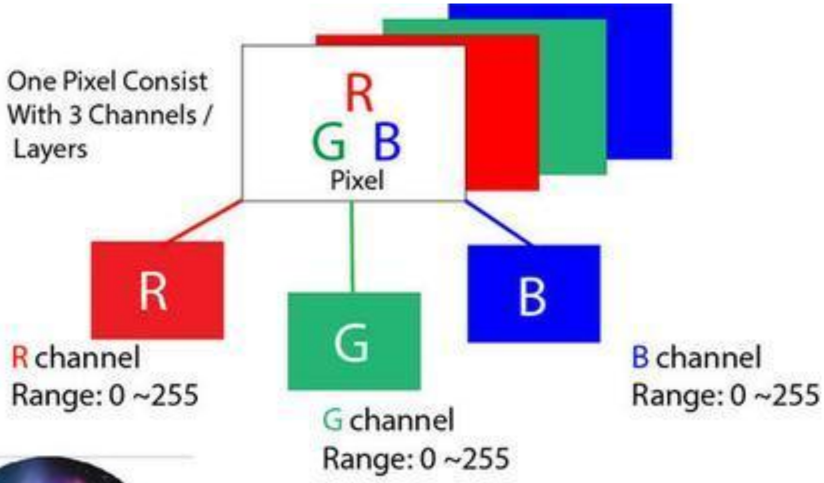

One Pixel Consist
With 3 Channels /
Layers
R
G B
Pixel
R
G
B
R channel
Range: 0 ~255
G channel
Range: 0 ~255
B channel
Range: 0 ~255
@ITPUB博客

image and neural net

4	5	9	10	47	0
0	74	5	78	5	6
12	34	0	8	1	8
2	5	8	1	8	7
84	87	48	87	85	3
1	22	45	43	21	44

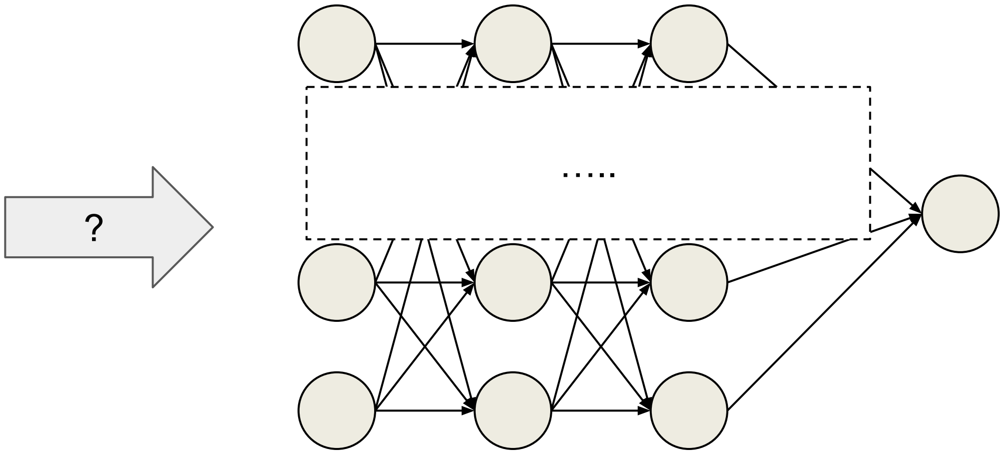


image and neural net

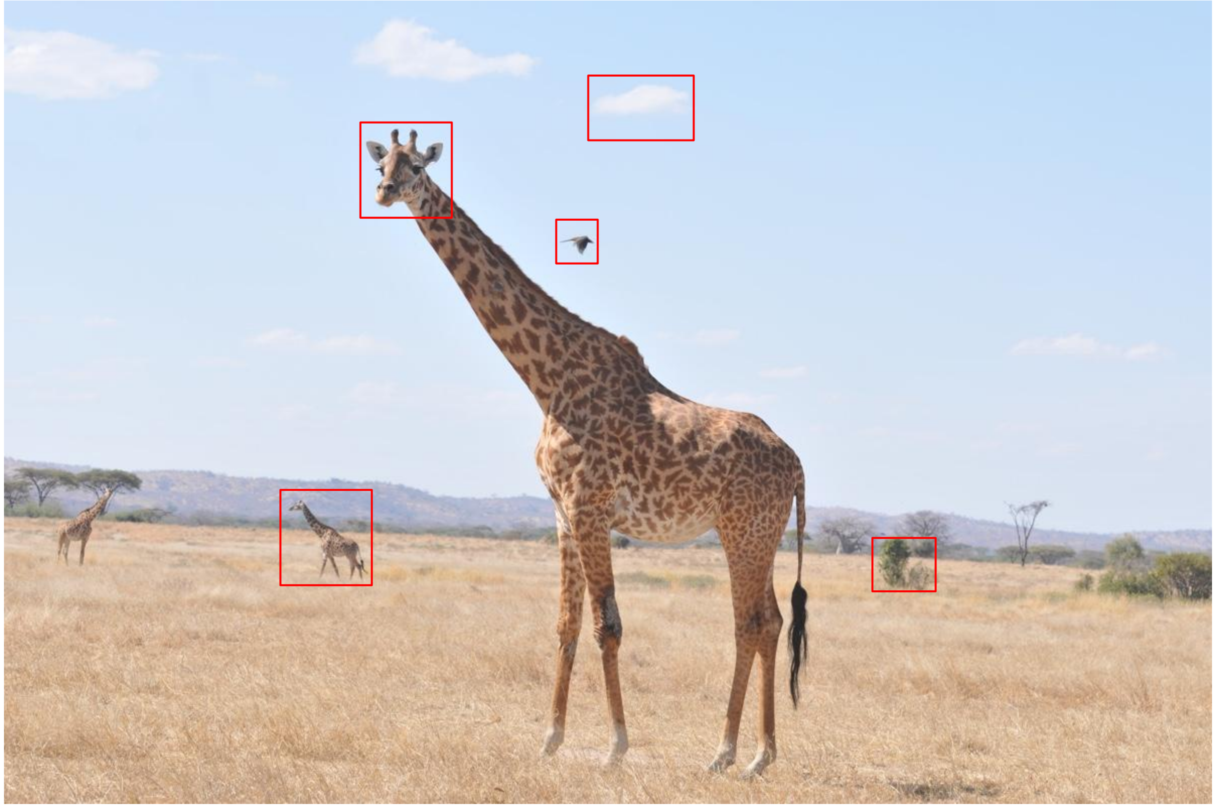
4	5	9	10	47	0
0	74	5	78	5	6
12	34	0	8	1	8
2	5	8	1	8	7
84	87	48	87	85	3
1	22	45	43	21	44

4	5	9	10	47	0	...	1	22	45	43	21	44

```
X = X.view(-1, 784) # 1 x 28 x 28 형태임으로, 784 형태의 벡터로 바꿔준다.
```

신경망과 이미지

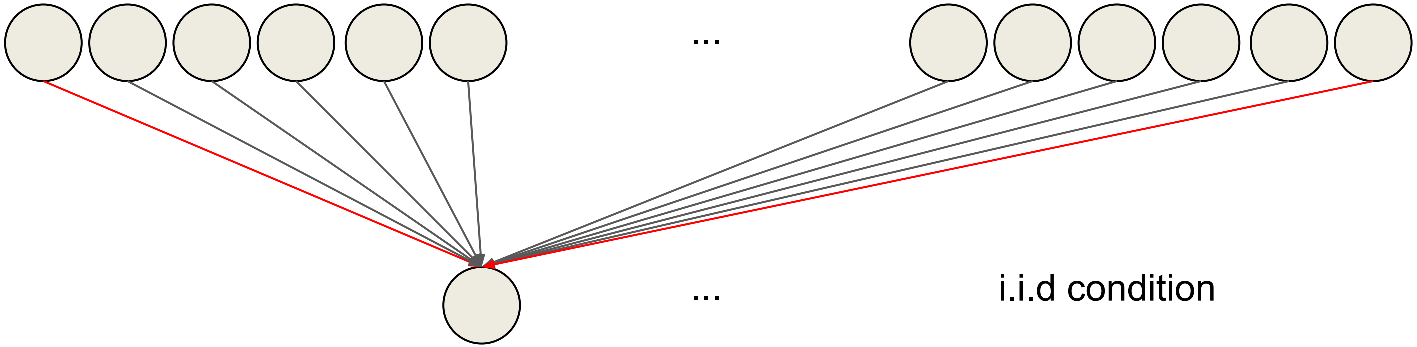
신경망과 이미지



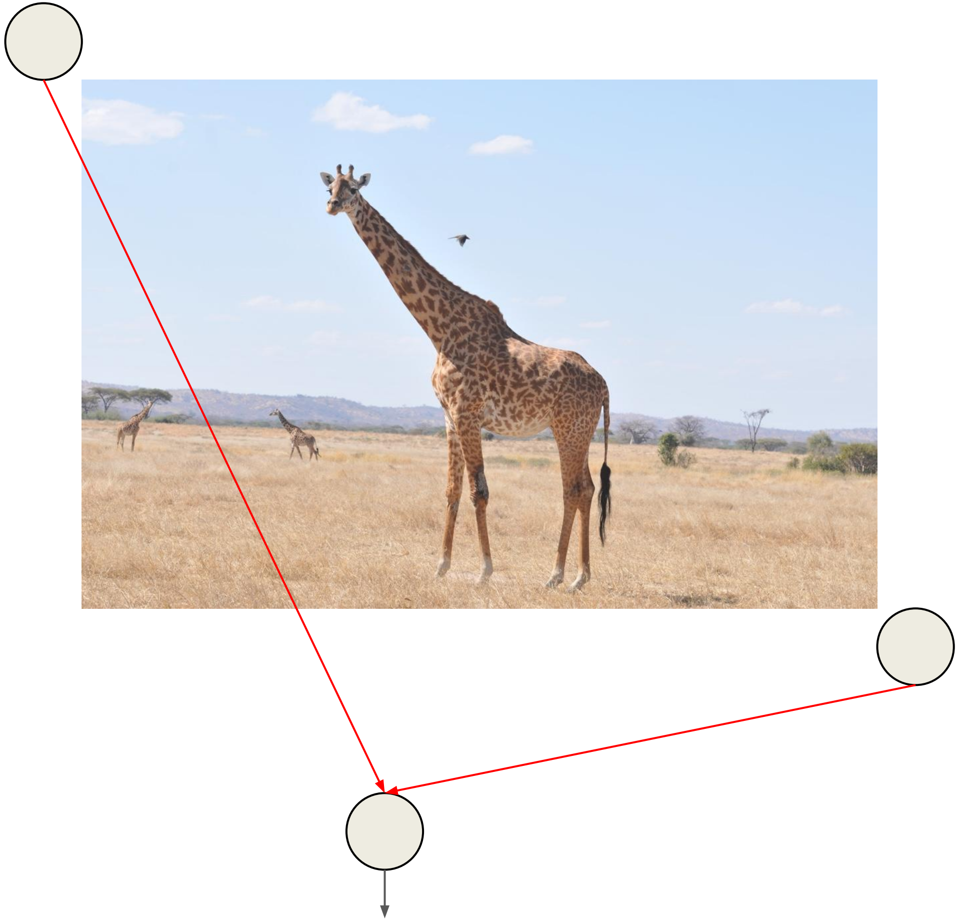
vectorize(flatten)

12	34	0	8	1	8
2	5	8	1	8	7
84	87	48	87	85	3
1	22	45	43	21	44

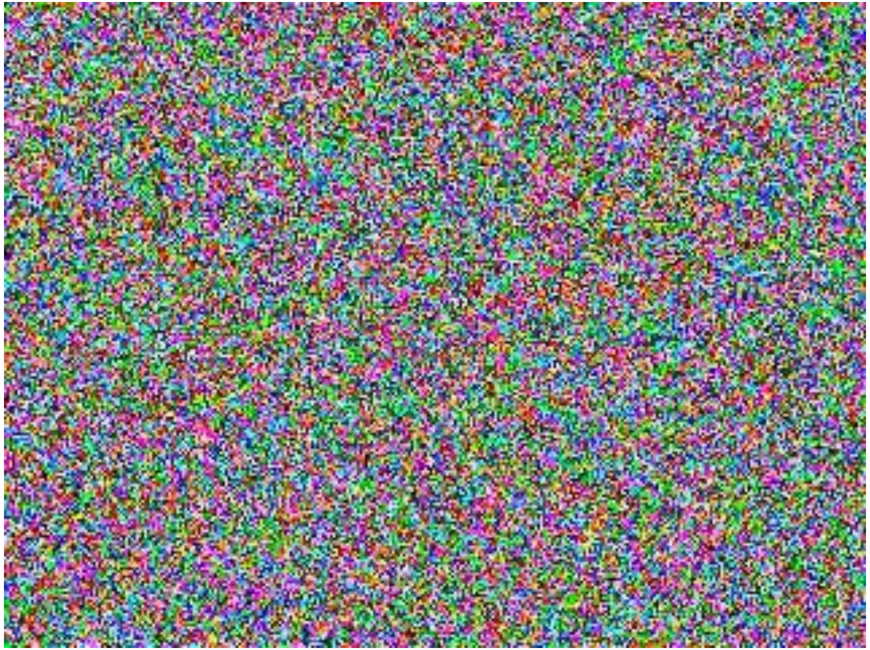
4	5	9	10	47	0	...	1	22	45	43	21	44



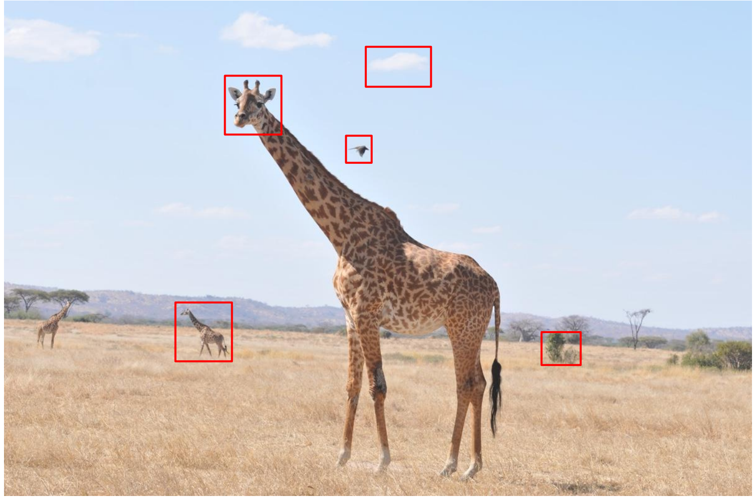
신경망과 이미지



신경망과 이미지

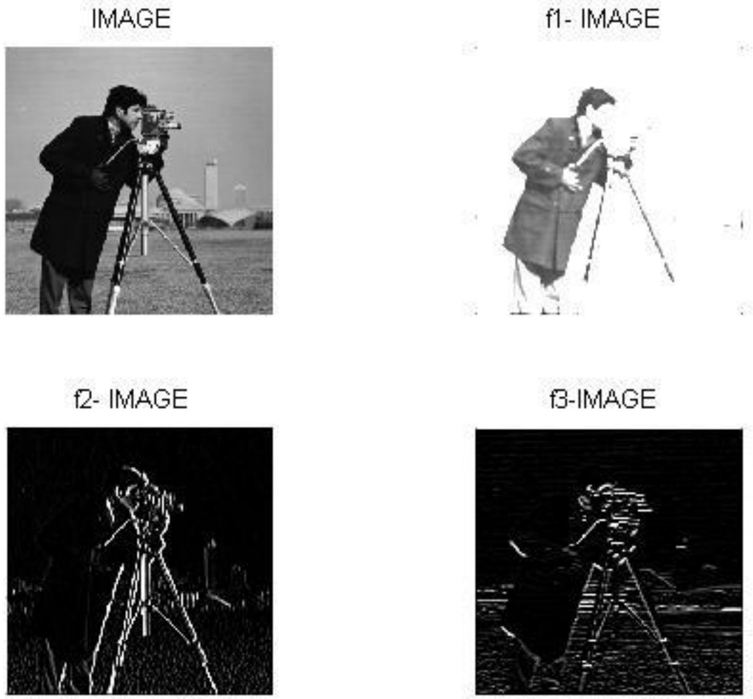


신경망과 이미지

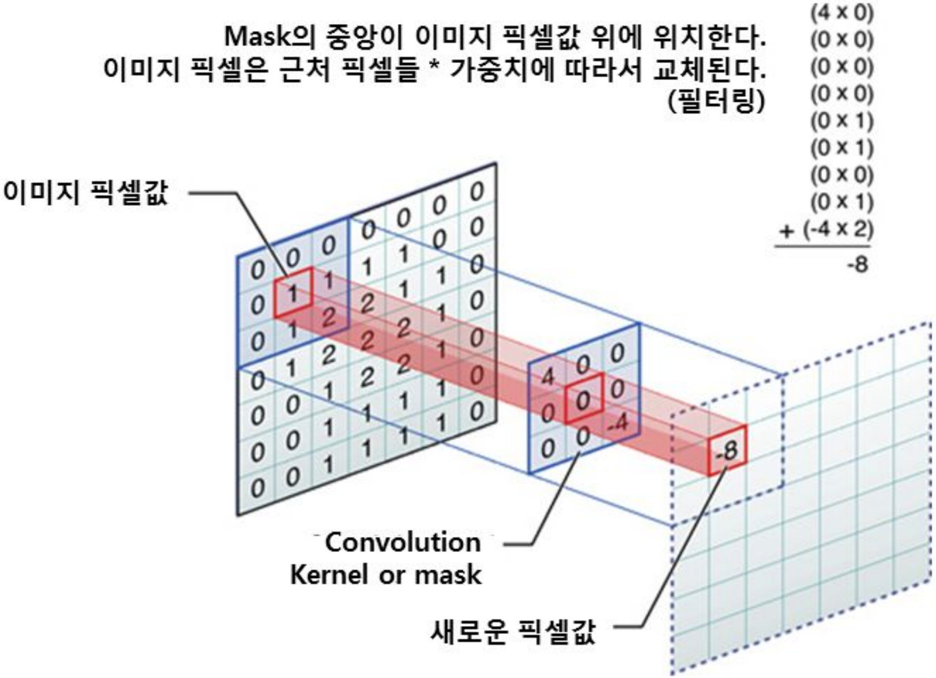


Convolutional Neural Network

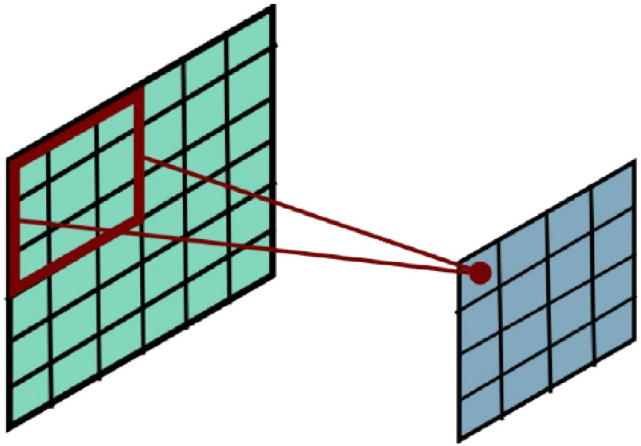
이미지 필터링



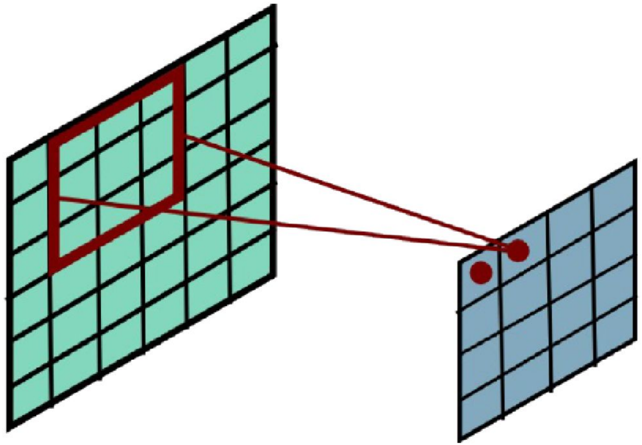
convolution



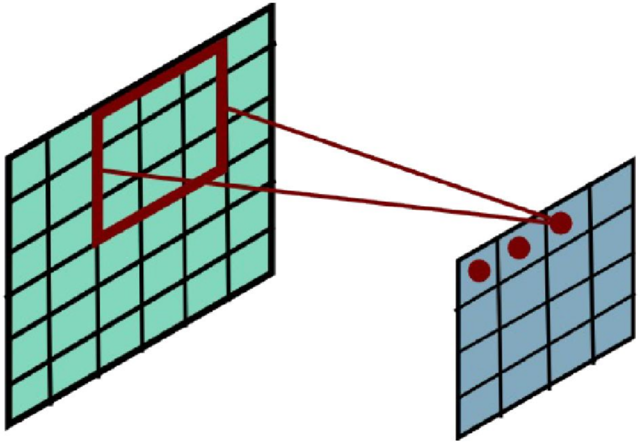
convolution



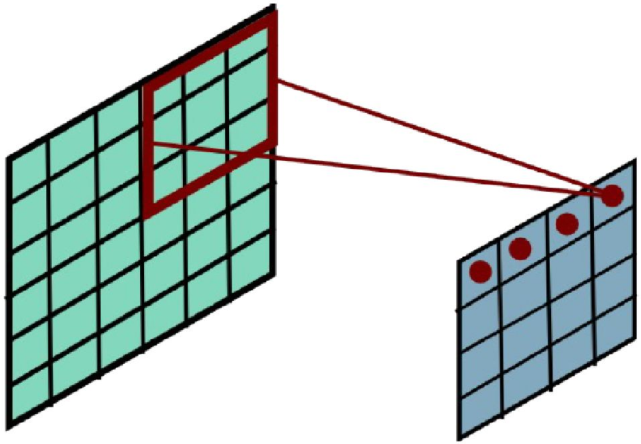
convolution



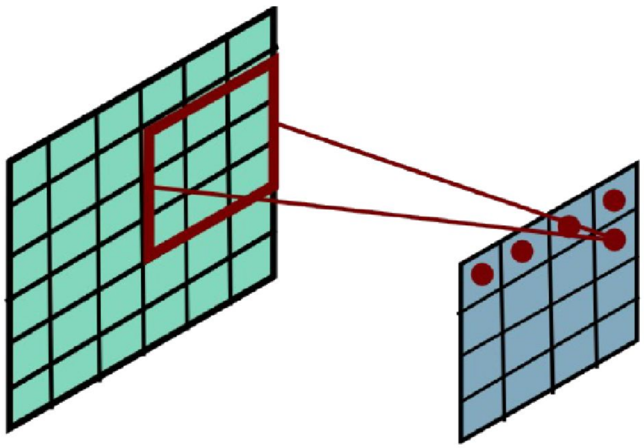
convolution



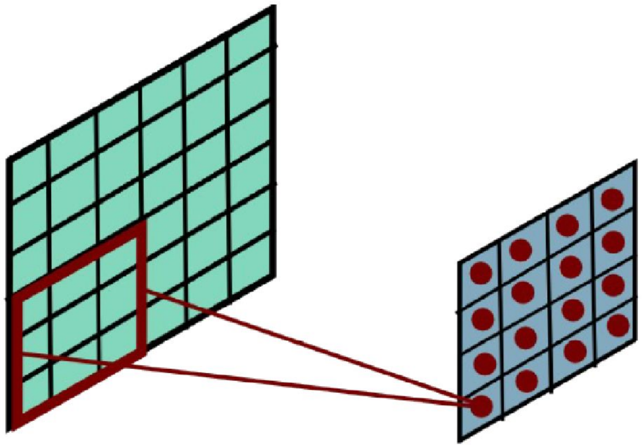
convolution



convolution



convolution



zero padding

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} * \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 15 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 \\ 0 & 4 & 5 & 6 & 0 \\ 0 & 7 & 8 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 6 & 8 \\ 6 & 15 & 14 \\ 12 & 14 & 9 \end{bmatrix}$$

stride

stride 1

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

*

0	0	1
0	1	0
1	0	0

=

1	6	8
6	15	14
12	14	9

stride 2

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

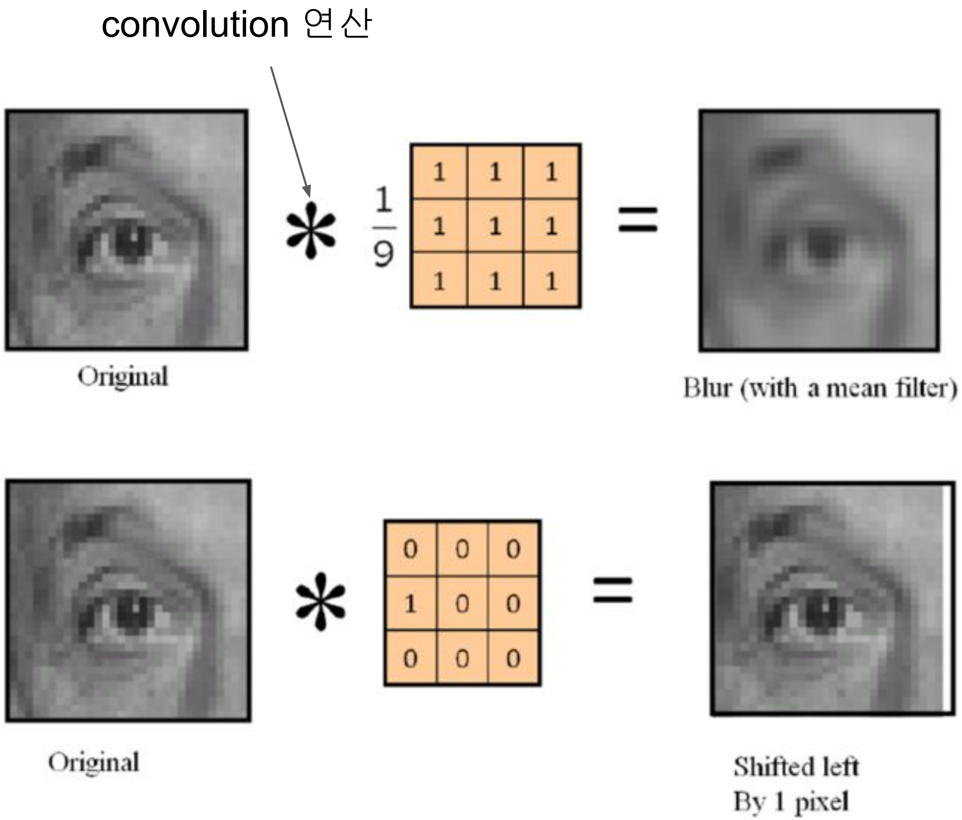
*

0	0	1
0	1	0
1	0	0

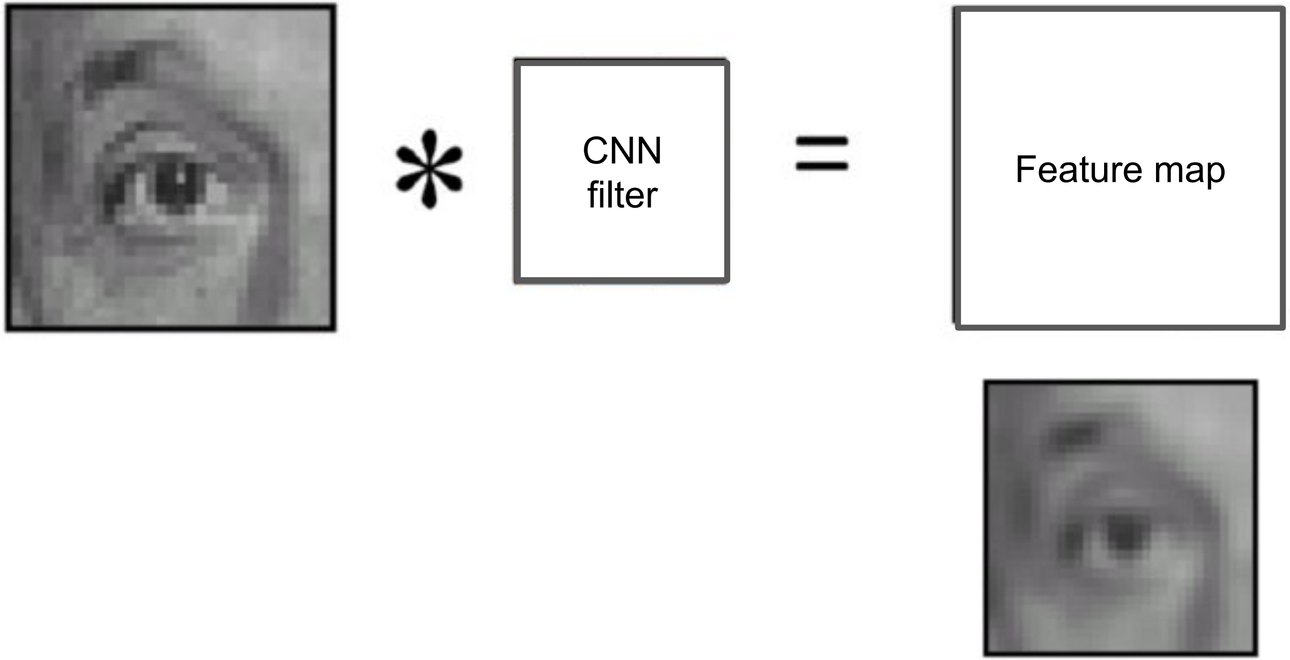
=

1	8
12	9

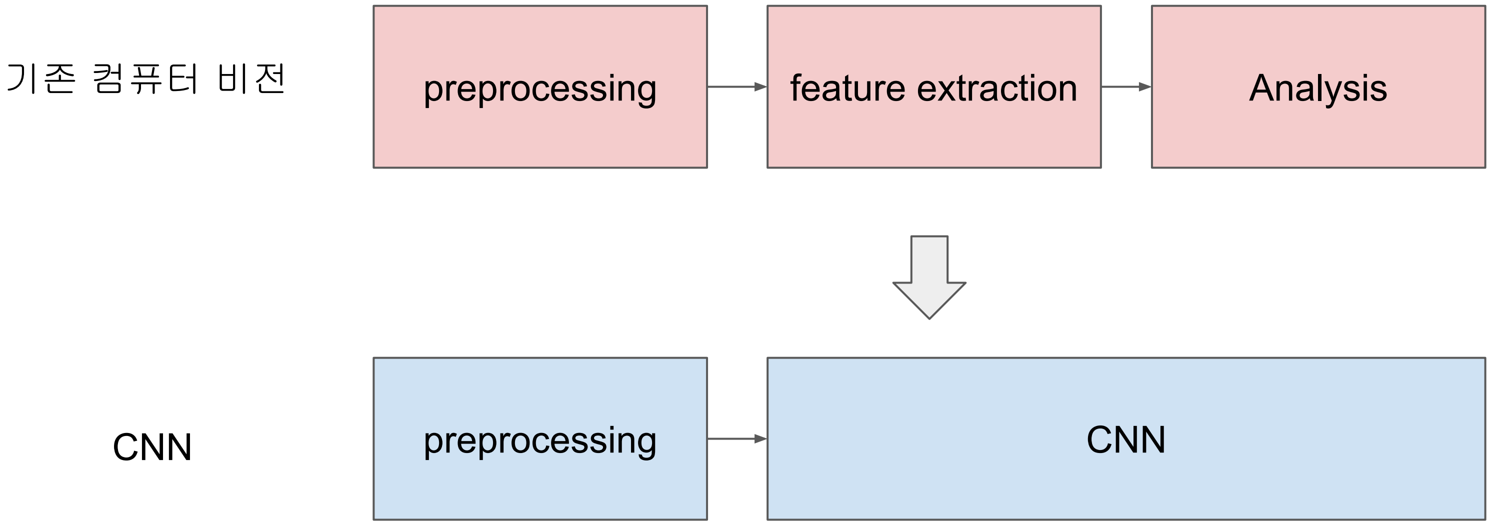
convolution



CNN

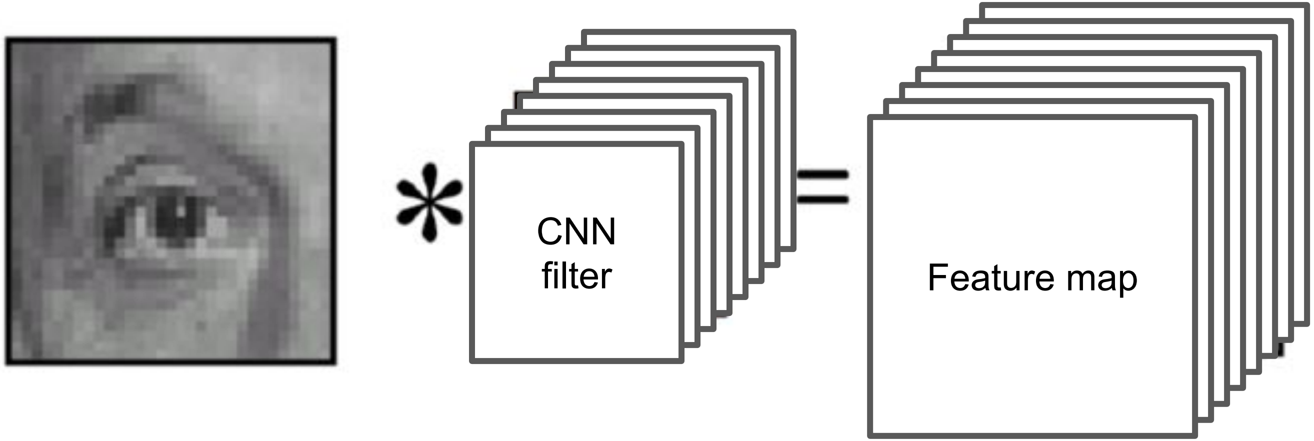


CNN

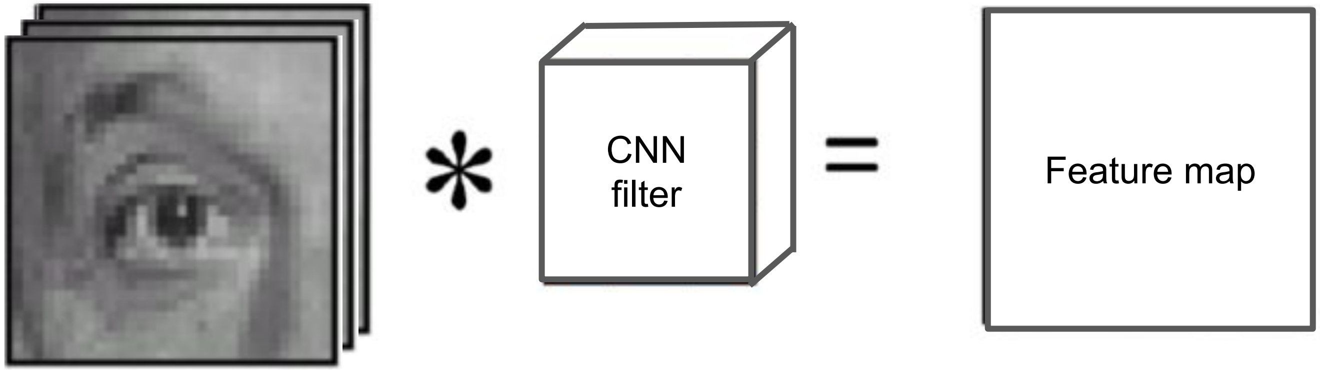


기존 컴퓨터 비전
preprocessing
feature extraction
Analysis
CNN
preprocessing
CNN

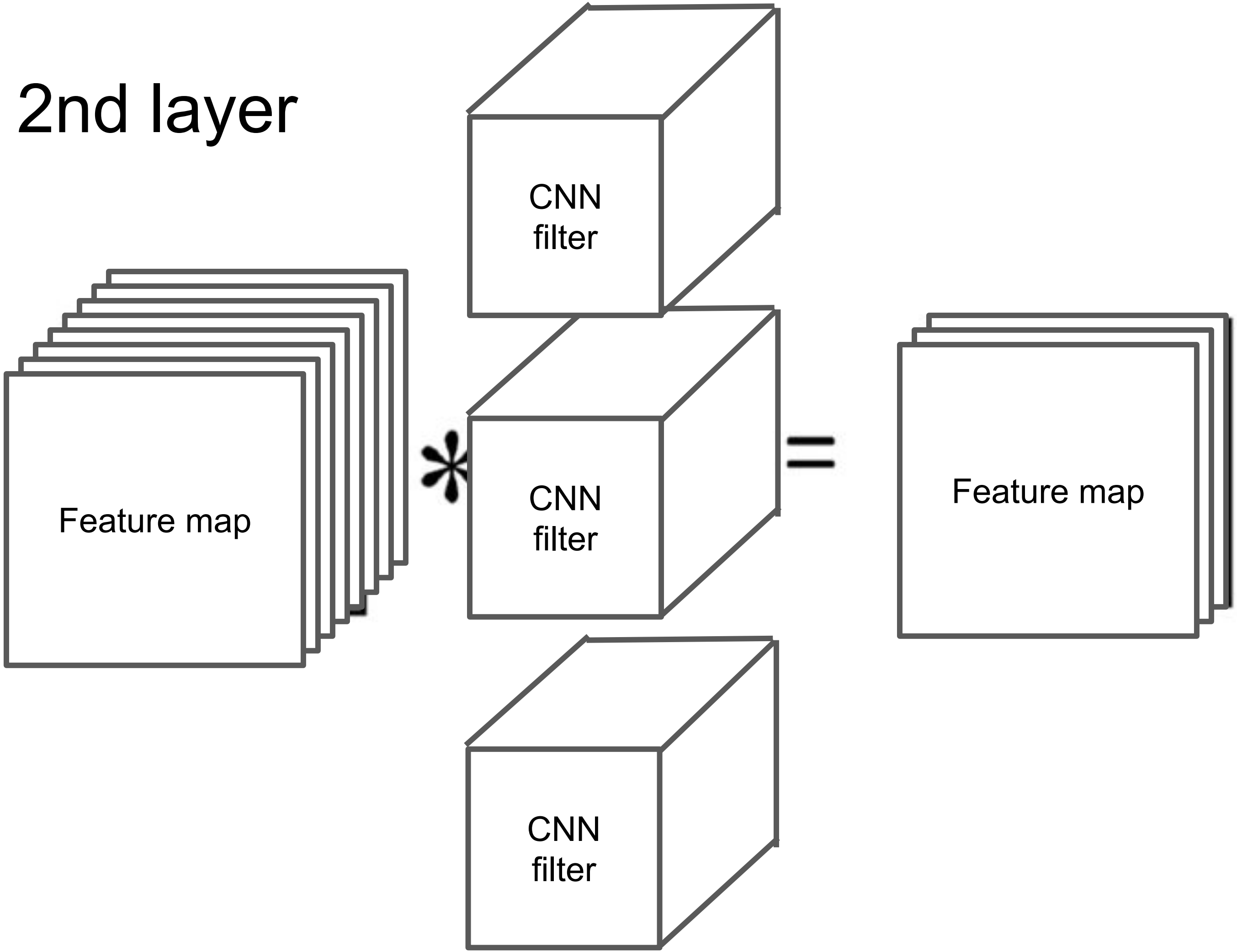
CNN



CNN - channel

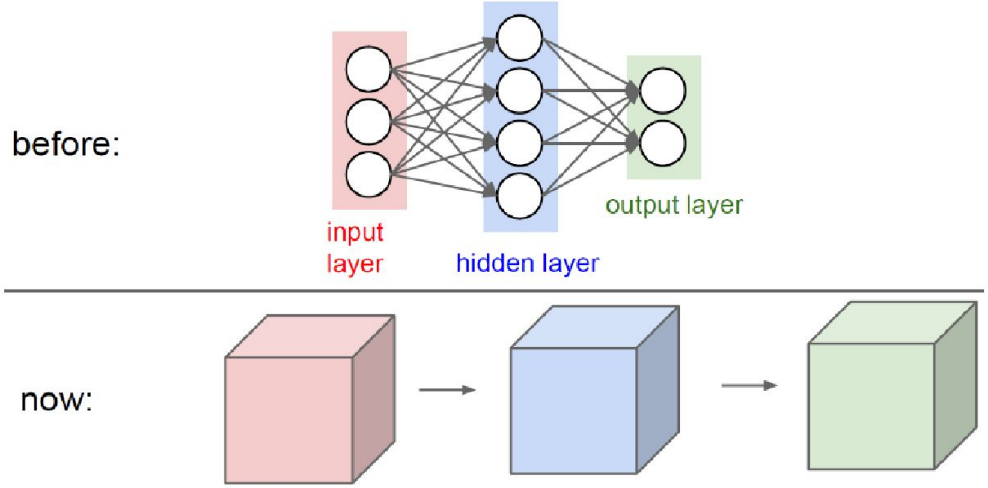


CNN 2nd layer



3 dim weight

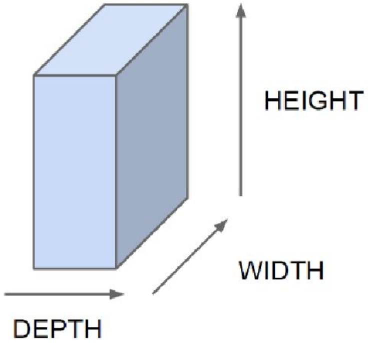
- Number of filters (neurons) is considered as a new dimension (depth)
  $\Rightarrow$ Volumetric representation



depth

- Number of filters (neurons) is considered as a new dimension (depth)
  ⇒ Volumetric representation

All Neural Net activations arranged in **3 dimensions:**



For example, a CIFAR-10 image is a 32x32x3 volume
32 width, 32 height, 3 depth (RGB channels)

실습 9

CNN 연산 구현하기

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

*

0	0	1
0	1	0
1	0	0

=

1	6	8
6	15	14
12	14	9

실습 9

CNN 연산 구현하기 convolution 연산의 행렬 연산을 위한 평탄화 코드

```python
import numpy as np
def im2col(input_data, filter_h, filter_w, stride=1, pad=0):
    """다수의 이미지를 입력받아 2차원 배열로 변환한다(평탄화).

    Parameters
    ----------
    input_data : 4차원 배열 형태의 입력 데이터(이미지 수, 채널 수, 높이, 너비)
    filter_h : 필터의 높이
    filter_w : 필터의 너비
    stride : 스트라이드
    pad : 패딩

    Returns
    -------
    col : 2차원 배열
    """
    N, C, H, W = input_data.shape
    out_h = (H + 2*pad - filter_h)//stride + 1
    out_w = (W + 2*pad - filter_w)//stride + 1

    img = np.pad(input_data, [(0,0), (0,0), (pad, pad), (pad, pad)], 'constant')
    col = np.zeros((N, C, filter_h, filter_w, out_h, out_w))

    for y in range(filter_h):
        y_max = y + stride*out_h
        for x in range(filter_w):
            x_max = x + stride*out_w
            col[:, :, y, x, :, :] = img[:, :, y:y_max:stride, x:x_max:stride]

    col = col.transpose(0, 4, 5, 1, 2, 3).reshape(N*out_h*out_w, -1)
    return col
```

실습 9

CNN 연산 구현하기
Convolution layer

```python
class Convolution:
    def __init__(self, W, b, stride=1, pad=0):
        self.W = W
        self.b = b
        self.stride = stride
        self.pad = pad

        # 중간 데이터 (backward 시 사용)
        self.x = None
        self.col = None
        self.col_W = None

        # 가중치와 편향 매개변수의 기울기
        self.dW = None
        self.db = None

    def forward(self, x):
        FN, C, FH, FW = self.W.shape
        N, C, H, W = x.shape
        out_h = 1 + int((H + 2*self.pad - FH) / self.stride)
        out_w = 1 + int((W + 2*self.pad - FW) / self.stride)

        col = im2col(x, FH, FW, self.stride, self.pad)
        print("input data -> im2col is")
        print(col)
        col_W = self.W.reshape(FN, -1).T
        print("Weight = filter ... -> im2col is")
        print(col_W)

        out = np.dot(col, col_W) + self.b
        print("affine 연산 수행 결과")
        print(out)

        out = out.reshape(N, out_h, out_w, -1).transpose(0, 3, 1, 2)

        self.x = x
        self.col = col
        self.col_W = col_W

        return out
```

print들은 실습을 위한 것으로, 실제 계산그래프에서는 모두 지웁니다.

실습 9

CNN 연산 구현하기
Convolution layer의 backward
(오늘 실습에선 안씁니다)

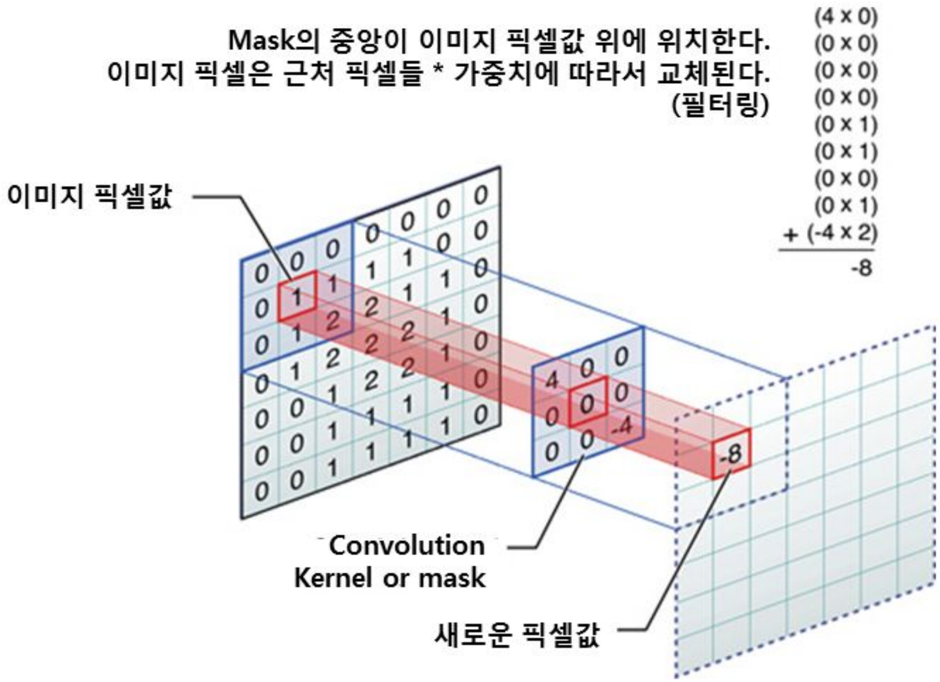
```
'''
def backward(self, dout):
    FN, C, FH, FW = self.W.shape
    dout = dout.transpose(0,2,3,1).reshape(-1, FN)

    self.db = np.sum(dout, axis=0)
    self.dW = np.dot(self.col.T, dout)
    self.dW = self.dW.transpose(1, 0).reshape(FN, C, FH, FW)

    dcol = np.dot(dout, self.col_W.T)
    dx = col2im(dcol, self.x.shape, FH, FW, self.stride, self.pad)

    return dx
'''
```

실습 9

CNN 연산 구현하기

```
filter_num = 1
input_channels = 1

# 입력데이터 만들기
x1 = np.array([[0,0,0,0,0],[0,1,2,3,0],[0,4,5,6,0],[0,7,8,9,0],[0,0,0,0,0]]).reshape(1, input_channels, 5, 5)
print("input data is")
print(x1)

# weight = convolution filter 만들기
W1 = np.array([[0,0,1],[0,1,0],[1,0,0]]).reshape([filter_num, input_channels, 3, 3])
b1 = np.zeros(filter_num) # bias는 0으로...
'''
W1 = np.array([[[0,0,1],[0,1,0],[1,0,0]],[[1,0,1],[0,1,0],[1,0,1]]]).reshape([2, input_channels, 3, 3])
b1 = np.zeros(2)
'''

print("weight = filter = kernel = mask is")
print(W1)

conv1 = Convolution(W1, b1) # convolution layer 정의
y=conv1.forward(x1) # convlution 연산 수행

print("convolution 수행 결과")
print(y)
```

실습 9



Mask의 중앙이 이미지 픽셀값 위에 위치한다.
이미지 픽셀은 근처 픽셀들 * 가중치에 따라서 교체된다.
(필터링)
(4 x 0)
(0 x 0)
(0 x 0)
(0 x 0)
(0 x 1)
(0 x 1)
(0 x 0)
(0 x 1)
+ (-4 x 2)
-8
이미지 픽셀값
Convolution
Kernel or mask
새로운 픽셀값

실습 9

```
input data is
[[[[0 0 0 0 0]
   [0 1 2 3 0]
   [0 4 5 6 0]
   [0 7 8 9 0]
   [0 0 0 0 0]]]]
```

```
weight = filter = kernel = mask is
[[[[0 0 1]
   [0 1 0]
   [1 0 0]]]]
```

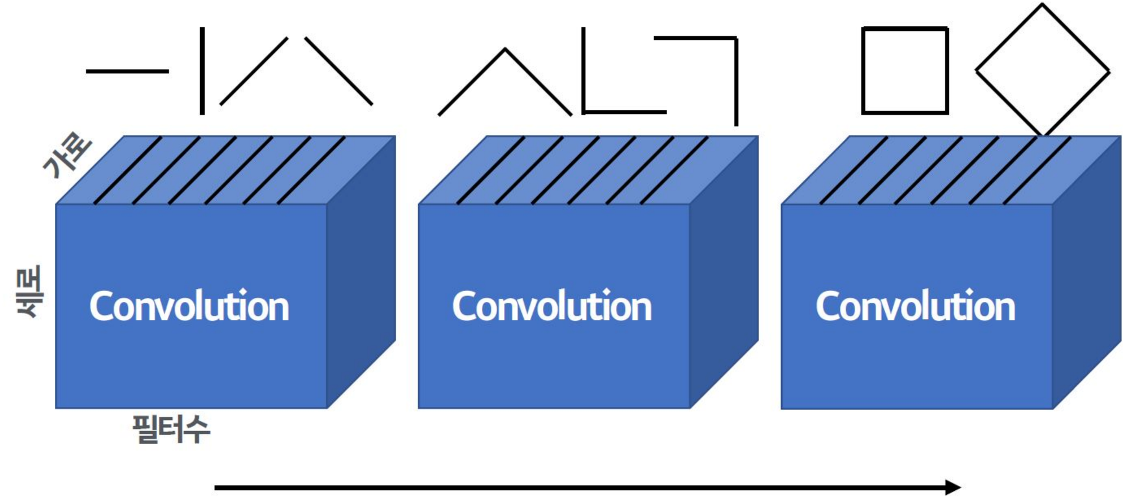
```
input data -> im2col is
[[0. 0. 0. 0. 1. 2. 0. 4. 5.]
 [0. 0. 0. 1. 2. 3. 4. 5. 6.]
 [0. 0. 0. 2. 3. 0. 5. 6. 0.]
 [0. 1. 2. 0. 4. 5. 0. 7. 8.]
 [1. 2. 3. 4. 5. 6. 7. 8. 9.]
 [2. 3. 0. 5. 6. 0. 8. 9. 0.]
 [0. 4. 5. 0. 7. 8. 0. 0. 0.]
 [4. 5. 6. 7. 8. 9. 0. 0. 0.]
 [5. 6. 0. 8. 9. 0. 0. 0. 0.]]
Weight = filter ... -> im2col is
```

```
leight = filter ... -> im2col is
[[0]
 [0]
 [1]
 [0]
 [1]
 [0]
 [1]
 [0]
 [0]]
```

```
affine 연산 수행 결과
[[ 1.]
 [ 6.]
 [ 8.]
 [ 6.]
 [15.]
 [14.]
 [12.]
 [14.]
 [ 9.]]
```

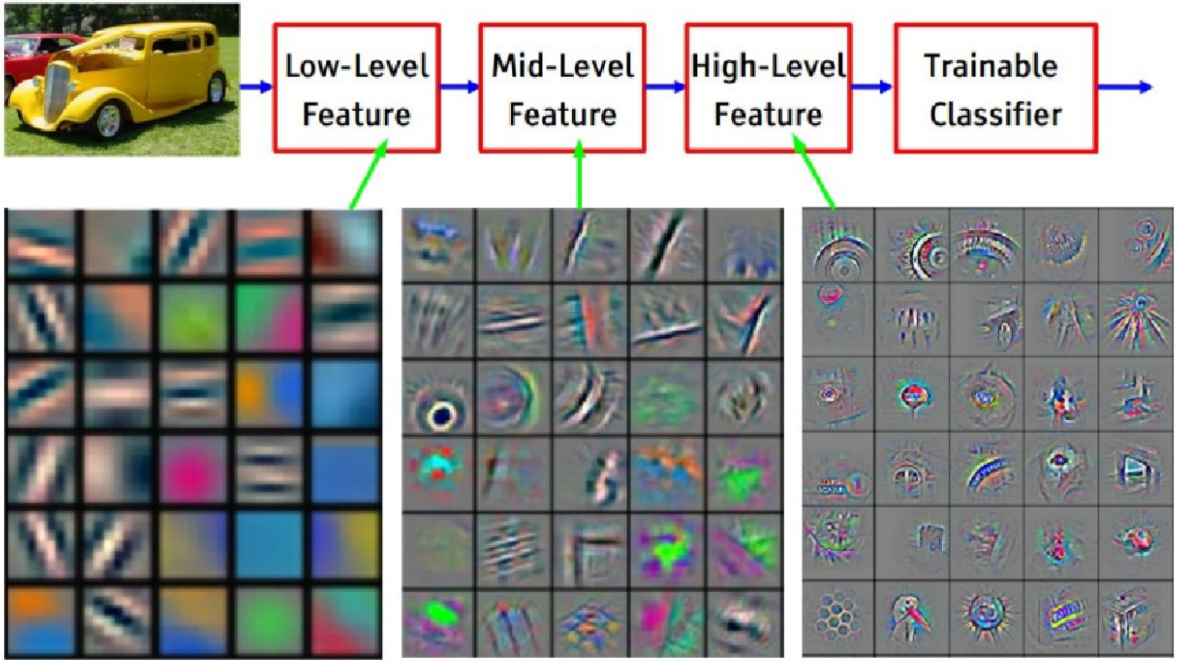
```
convolution 수행 결과
[[[[ 1.  6.  8.]
   [ 6. 15. 14.]
   [12. 14.  9.]]]]
```

feature



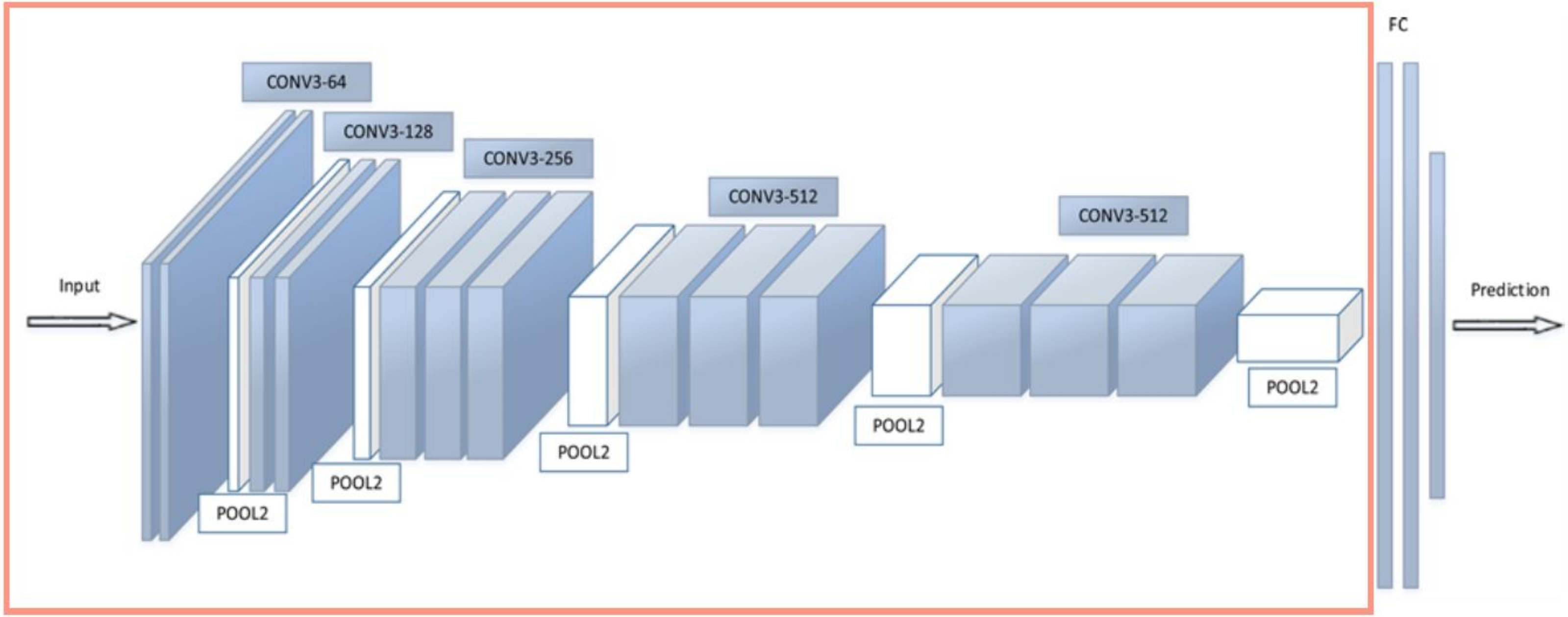
Convolution의 좋은점 부품을 조립해 더 복잡한 부품을 만든다

feature map



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

CNN classifier



Pooling

Activation Map

12	20	30	0
8	12	2	0
34	70	37	7
112	100	22	12

Max Pooling

20	30
112	37

Average Pooling

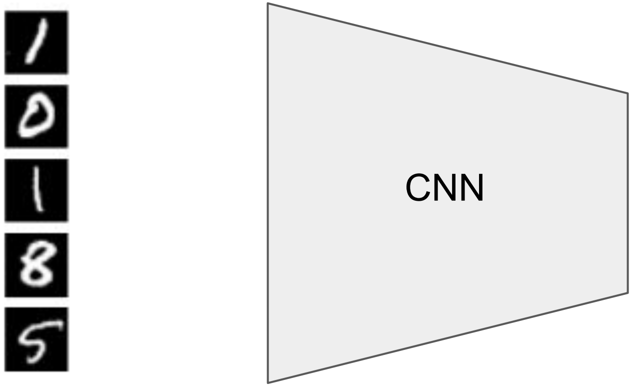
13	8
79	18

실습 10

CNN classifier 만들기

입력

출력



[0,1,0,0,0,0,0,0,0,0]
[1,0,0,0,0,0,0,0,0,0]
[0,1,0,0,0,0,0,0,0,0]
[0,0,0,0,0,0,0,0,1,0]
[0,0,0,0,0,1,0,0,0,0]

실습 10

```python
class MNIST_classifier_CNN(nn.Module):
  def __init__(self, class_num):
    super().__init__()
    self.class_num = class_num

    self.conv_net = nn.Sequential(
        nn.Conv2d(in_channels=1, out_channels=10, kernel_size=5),
        nn.BatchNorm2d(10),
        nn.ReLU(),
        nn.MaxPool2d(2),

        nn.Conv2d(in_channels=10, out_channels=20, kernel_size=5),
        nn.BatchNorm2d(20),
        nn.ReLU(),
        nn.MaxPool2d(2)
    )
    self.fc_net = nn.Sequential(
        nn.Linear(320,50),
        nn.BatchNorm1d(50),
        nn.ReLU(),
        nn.Linear(50,self.class_num),
        nn.Softmax()
    )
  def forward(self, x):
    feature = self.conv_net(x)
    feature = feature.view(-1,320)
    y = self.fc_net(feature)
    return y
```

```python
net = MNIST_classifier_CNN(class_num=10).cuda() # gpu 사용.(뒤에 .cuda())
net.apply(weight_init)
```

실습 10

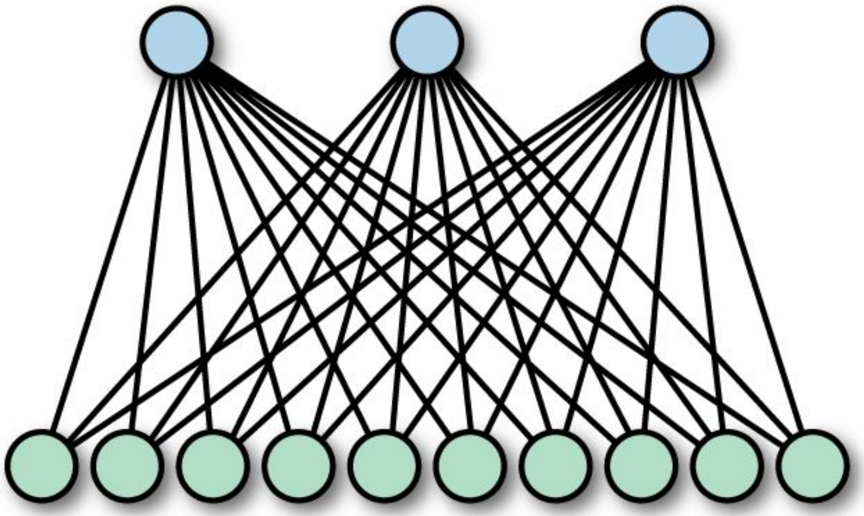
```python
def weight_init(m):
    # Conv layer와 batchnorm layer를 위한 가중치 초기화를 추가함.
    classname = m.__class__.__name__
    if classname.find('Conv') != -1:
        m.weight.data.normal_(0.0, 0.02)
    elif classname.find('BatchNorm') != -1:
        m.weight.data.normal_(1.0, 0.02)
        m.bias.data.fill_(0)
    elif classname.find('Linear')!=-1:
        m.weight.data.normal_(0.0, 0.02)
        m.bias.data.fill_(0)
```

실습 10

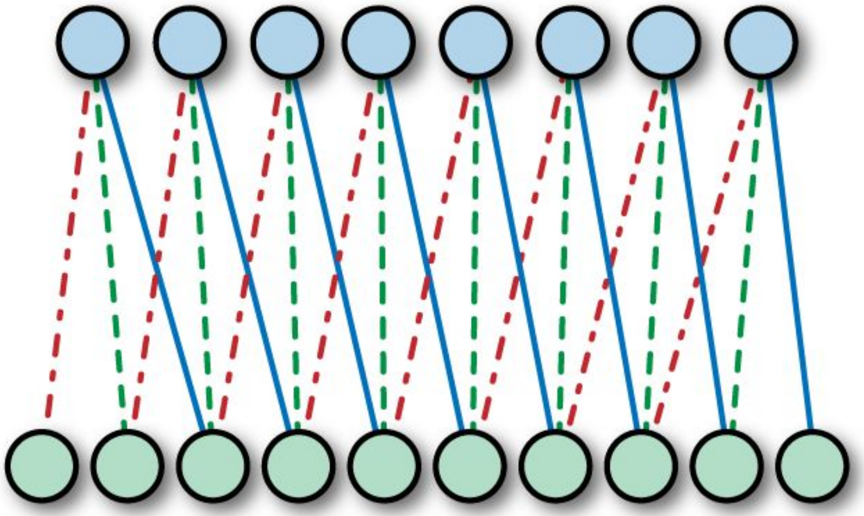
```
train_loss_list = []
val_loss_list = []
net.train()
for epoch in range(epochs):
    for i, (X, t) in enumerate(train_loader):
        X = X.cuda() # gpu 사용.(뒤에 .cuda()) => view를 이용해 vectorize하는 부분 사라짐
        t = one_hot_embedding(t, 10).cuda() # gpu 사용.(뒤에 .cuda())

        Y = net(X)
```

wasted weights

Fully Connected



Convolutional Layer



실습 11

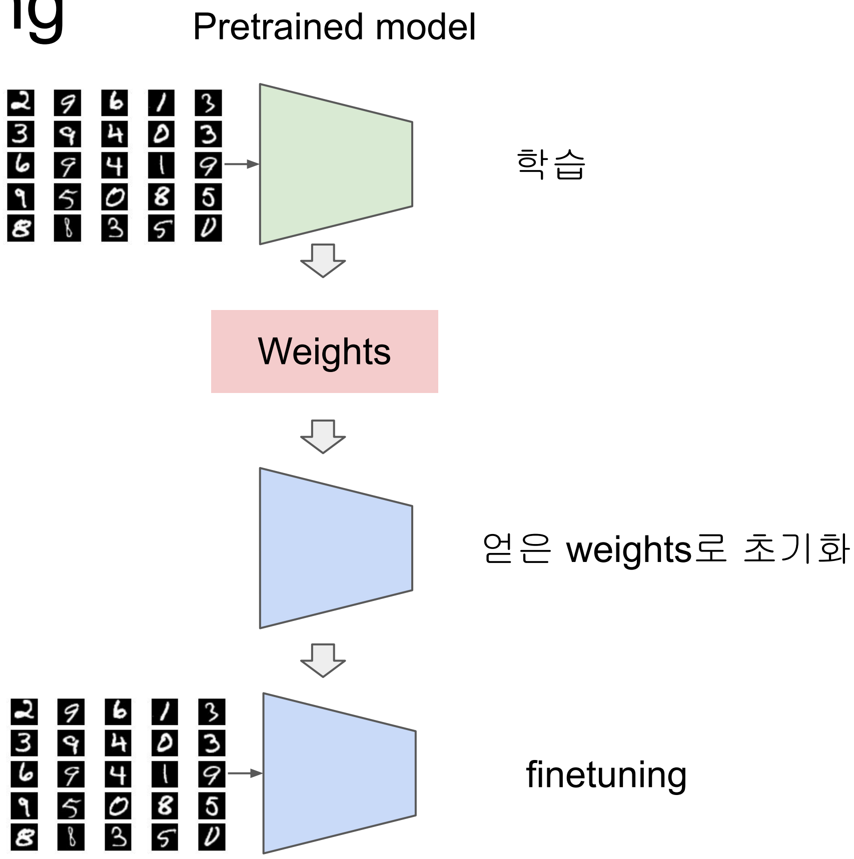
```python
class MNIST_classifier_FCN(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super().__init__()
        self.input_size = input_size
        self.hidden_size = hidden_size
        self.output_size = output_size

        self.network1 = nn.Sequential(
            nn.Linear(self.input_size, self.hidden_size),
            nn.Sigmoid(),
            nn.Linear(self.hidden_size, self.output_size),
            nn.Softmax()
        )

    def forward(self, x):
        y = self.network1(x)
        return y
```

```python
class MNIST_classifier_CNN(nn.Module):
  def __init__(self, class_num):
    super().__init__()
    self.class_num = class_num

    self.conv_net = nn.Sequential(
        nn.Conv2d(in_channels=1, out_channels=10, kernel_size=5),
        nn.BatchNorm2d(10),
        nn.ReLU(),
        nn.MaxPool2d(2),

        nn.Conv2d(in_channels=10, out_channels=20, kernel_size=5),
        nn.BatchNorm2d(20),
        nn.ReLU(),
        nn.MaxPool2d(2)
    )
    self.fc_net = nn.Sequential(
        nn.Linear(320,50),
        nn.BatchNorm1d(50),
        nn.ReLU(),
        nn.Linear(50,self.class_num),
        nn.Softmax()
    )
  def forward(self, x):
    feature = self.conv_net(x)
    feature = feature.view(-1,320)
    y = self.fc_net(feature)
    return y
```

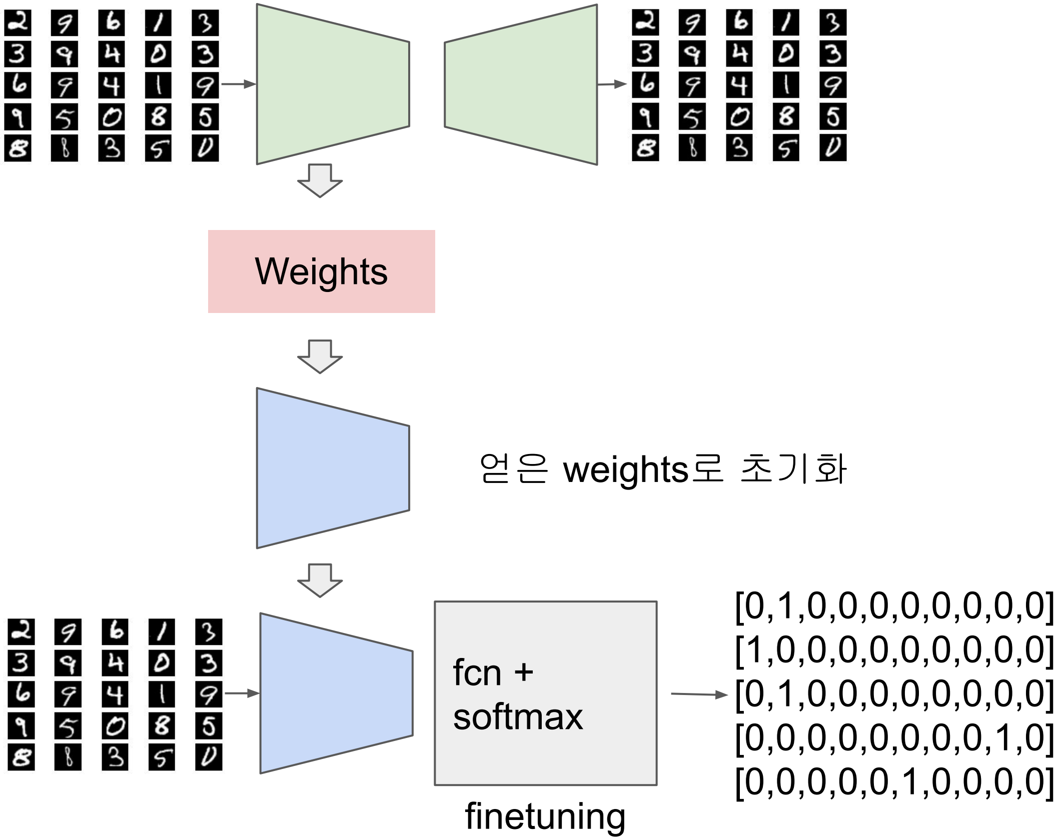
실습 11

```python
fc_net = MNIST_classifier_FCN(input_size=784, hidden_size=50, output_size=10)
conv_net = MNIST_classifier_CNN(class_num=10)

print("fc_net's parameters")
fc_num_weight = 0
for parameter in fc_net.parameters():
    print(parameter.shape)
    fc_num_weight+=np.asarray(parameter.shape).prod()


print("conv_net's parameters")
conv_num_weight = 0
for parameter in conv_net.parameters():
    print(parameter.shape)
    conv_num_weight+=np.asarray(parameter.shape).prod()

print("The number of fc_net's parameters")
print(fc_num_weight)

print("The number of conv_net's parameters")
print(conv_num_weight)
```

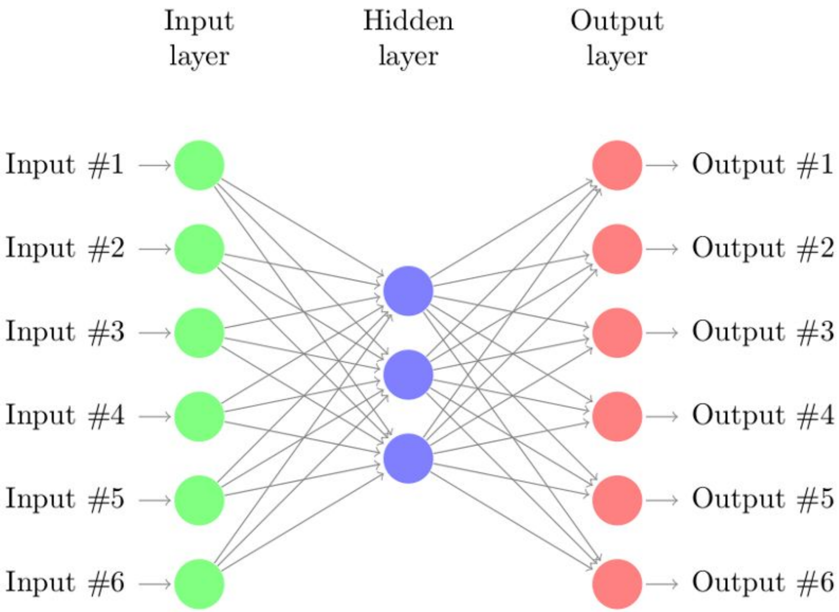
Transfer learning



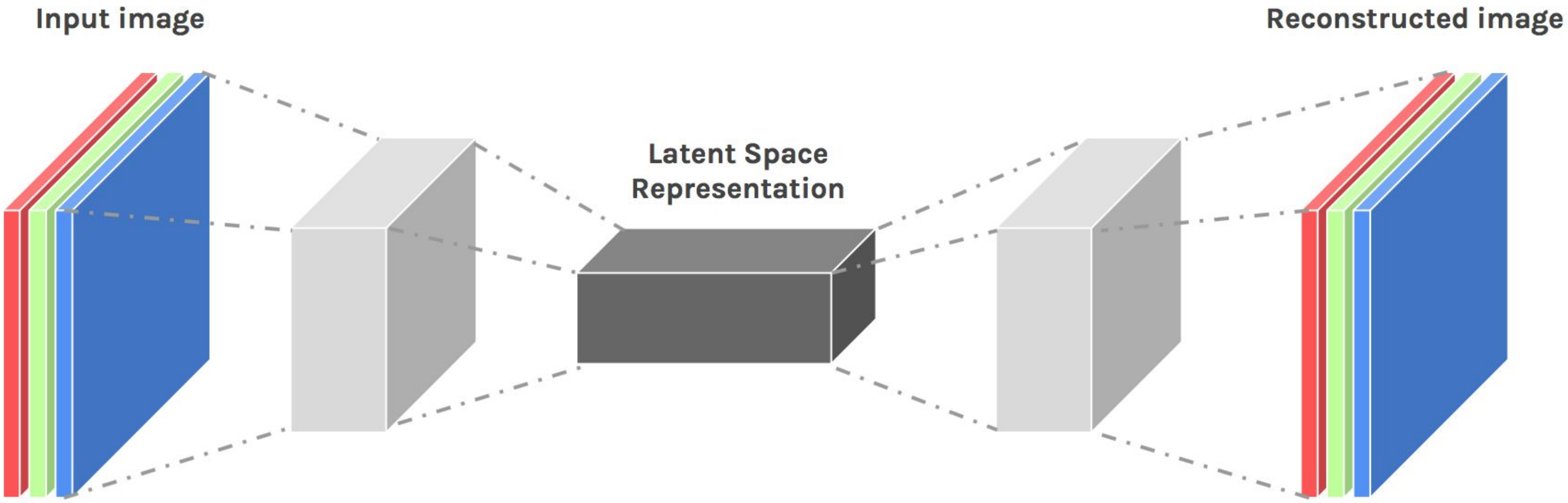
Transfer learning



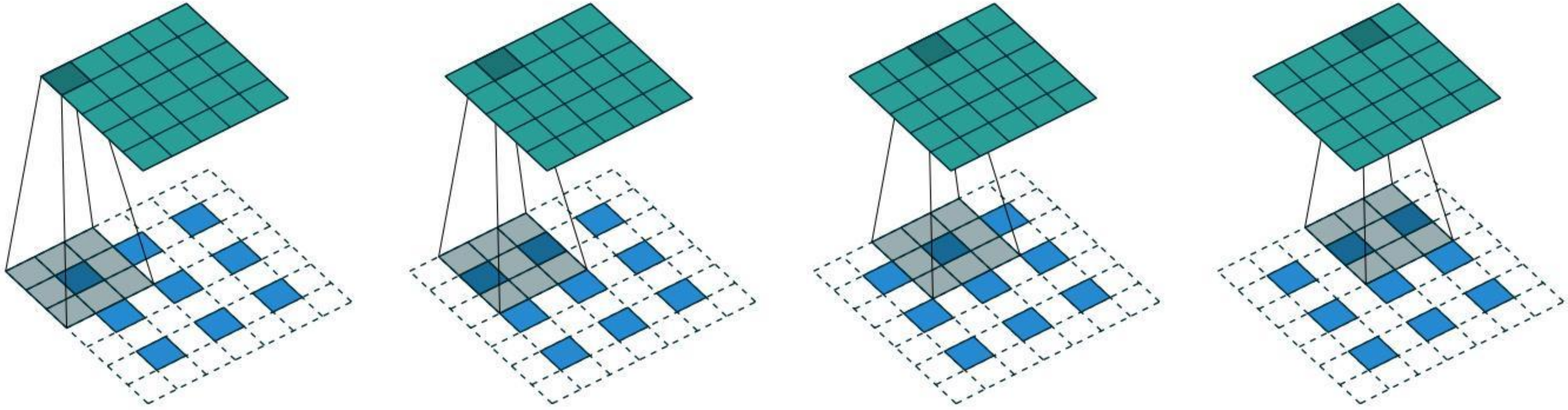
Autoencoder



Autoencoder

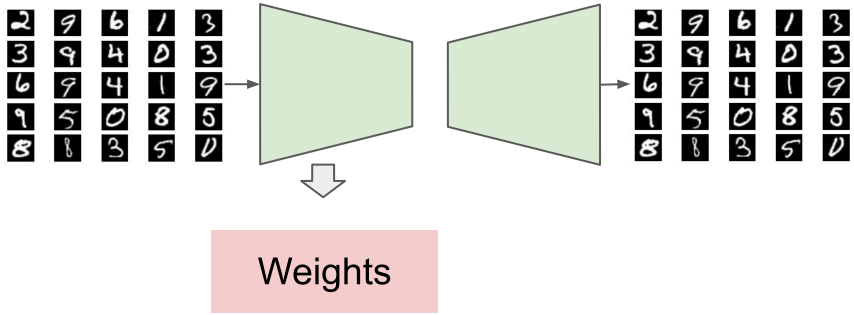


Transposed convolution



실습 12

CNN Autoencoder 만들고 weight 저장하기



실습 12

colab + 구글드라이브

```
from google.colab import drive
drive.mount('/content/gdrive/')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3

Enter your authorization code:
```

링크 클릭 후 로그인.
로그인 이후 나오는 코드를 아래쪽 빈칸에 입력.
`Mounted at /content/gdrive/` 라고 나오면 성공

실습 12

colab + 구글드라이브

```
import os
#os.mkdir("/content/gdrive/My Drive/AI") #폴더를 만드는 코드이니 한번만 실행하세요. 구글드라이브에서 직접 폴더 만들어도 됩니다.
with open('/content/gdrive/My Drive/AI/hello.txt', 'w') as f:
  f.write('Hello Google Drive colab !') # 테스트용 텍스트파일 생성
!cat /content/gdrive/My\ Drive/AI/hello.txt #텍스트 파일 내용 출력하기
```

실습 12

```
class MNIST_CNN_Encoder(nn.Module):
    def __init__(self):
        super().__init__()

        self.encoder = nn.Sequential(
            nn.Conv2d(1, 16, 3, stride=3, padding=1),
            nn.ReLU(True),
            nn.MaxPool2d(2, stride=2),
            nn.Conv2d(16, 8, 3, stride=2, padding=1),
            nn.ReLU(True),
            nn.MaxPool2d(2, stride=1)
        )

    def forward(self, x):
        z = self.encoder(x)
        return z


class MNIST_CNN_Decoder(nn.Module):
    def __init__(self):
        super().__init__()

        self.decoder = nn.Sequential(
            nn.ConvTranspose2d(8, 16, 3, stride=2),
            nn.ReLU(True),
            nn.ConvTranspose2d(16, 8, 5, stride=3, padding=1),
            nn.ReLU(True),
            nn.ConvTranspose2d(8, 1, 2, stride=2, padding=1),
            nn.Tanh()
        )

    def forward(self, z):
        x_ = self.decoder(z)
        return x_
```

실습 12

```
epochs = 10
learning_rate = 0.01
batch_size = 100
loss_function = nn.MSELoss()

# load the dataset
dataset = datasets.MNIST('../data', train=True,
                         download=True, transform=transforms.Compose([
        transforms.ToTensor()
        , transforms.Normalize((0.5,), (0.5,))
    ]))
```

-1~1 scaling으로 normalize하였으며 test loader도 이렇게 만들어줍시다.

실습 12

```
encoder = MNIST_CNN_Encoder().cuda()
encoder.apply(weight_init)

decoder = MNIST_CNN_Decoder().cuda()
decoder.apply(weight_init)

net_params = list(encoder.parameters())+list(decoder.parameters())
optimizer = optim.Adam(net_params, betas=(0.5, 0.999),lr=learning_rate)
```

실습 12

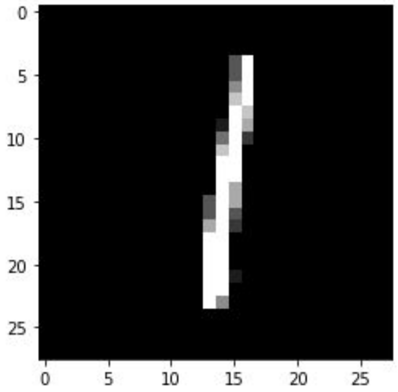
```
train_loss_list = []
val_loss_list = []
encoder.train()
decoder.train()
for epoch in range(epochs):
    for i, (X, _) in enumerate(train_loader):
        X = X.cuda()
        z = encoder(X)
        recon_X = decoder(z)

        loss = loss_function(recon_X, X)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        # validation loss 계산.
        if i % 100 == 0:
            with torch.no_grad():
                val_100_loss = []
                for (X, _) in valid_loader:
                    X = X.cuda()
                    z = encoder(X)
                    recon_X = decoder(z)
                    loss = loss_function(recon_X, X)

                    val_100_loss.append(loss)
                train_loss_list.append(loss)
                val_loss_list.append(np.asarray(val_100_loss).sum() / len(valid_loader))
        print("[%d/%d][%d/%d] loss : %f" % (i, len(train_loader), epoch, epochs, loss))
```

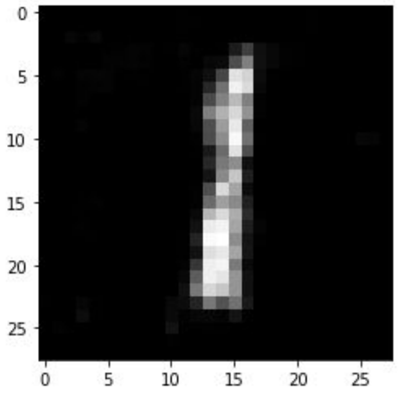
실습 12

```
# 학습된 모델의 weight를 저장하는 코드
project_root_path = '/content/gdrive/My Drive/AI'
encoder_save_path = '%s/pretrained_encoder.pth' % (project_root_path)
torch.save(encoder.state_dict(), encoder_save_path)
```

실습 12

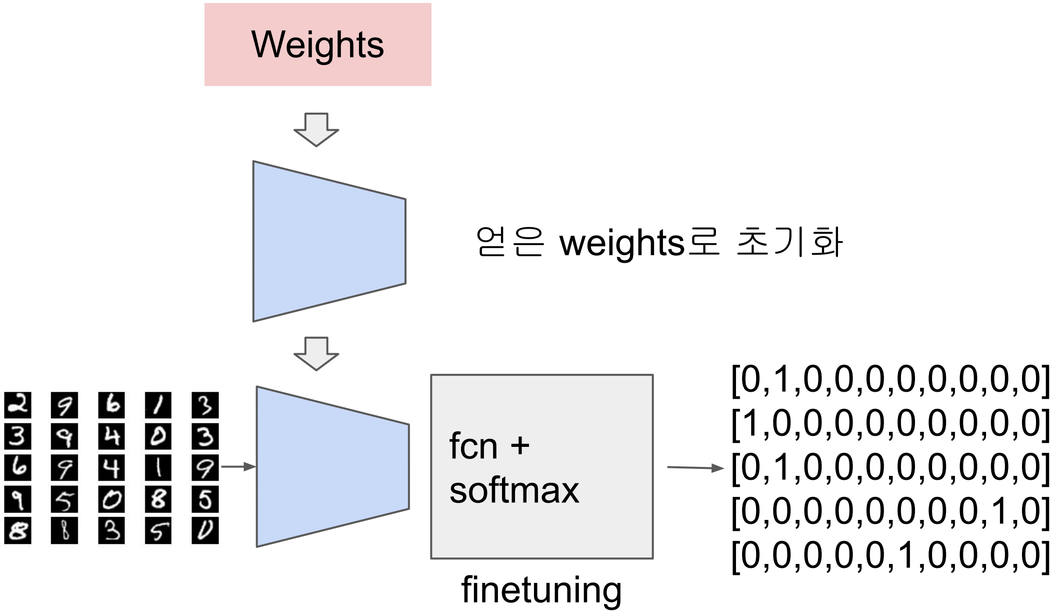
```python
print("testing")
encoder.eval()
decoder.eval()
correct = 0
with torch.no_grad():
    for i, (X, _) in enumerate(test_loader):
        X = X.cuda()
        z = encoder(X)
        recon_X = decoder(z)

        print("오토인코더 테스트 결과")
        for i in range(5):
            plt.imshow(X[i].cpu().reshape(28, 28))
            plt.gray()
            plt.show()

            plt.imshow(recon_X[i].cpu().reshape(28, 28))
            plt.gray()
            plt.show()
        break

plt.plot(np.column_stack((train_loss_list, val_loss_list)))
```





실습 12

CNN Autoencoder 만들고 weight 저장하기



실습 12

```
fcn = MNIST_FCN(class_num=10).cuda()
fcn.apply(weight_init)

# 저장해둔 weight를 불러와 해당 weight로 초기화 시킨다.
pretrained_encoder = MNIST_CNN_Encoder().cuda()
project_root_path = '/content/gdrive/My Drive/AI'
encoder_save_path = '%s/pretrained_encoder.pth' % (project_root_path)
saved_weights = torch.load(encoder_save_path)
pretrained_encoder.load_state_dict(saved_weights)
#pretrained_encoder.apply(weight_init) # 처음부터 학습하는 것을 테스트하고 싶을 경우
```

```
epochs = 5
learning_rate = 0.01
batch_size = 100
loss_function = nn.BCELoss()

optimizer = optim.Adam(list(fcn.parameters())+list(pretrained_encoder.parameters()), betas=(0.5, 0.999), lr=learning_rate)
#optimizer = optim.Adam(fcn.parameters(), betas=(0.5, 0.999), lr=learning_rate)  # Adam optimizer로 변경. betas =(0.5, 0.999) # encoder는 고정하고 fcn만 학습하는 코드
```

실습 12

```
train_loss_list = []
fcn.train()
for epoch in range(epochs):
    for i, (X, t) in enumerate(train_loader):
        X = X.cuda()
        t = one_hot_embedding(t, 10).cuda()
        z = pretrained_encoder(X)
        Y = fcn(z)

        loss = loss_function(Y, t)
        train_loss_list.append(loss)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        print("[%d/%d][%d/%d] loss : %f"%(i,len(train_loader),epoch,epochs, loss))
```

test 부분도 이런 구조로 짜줘야합니다!