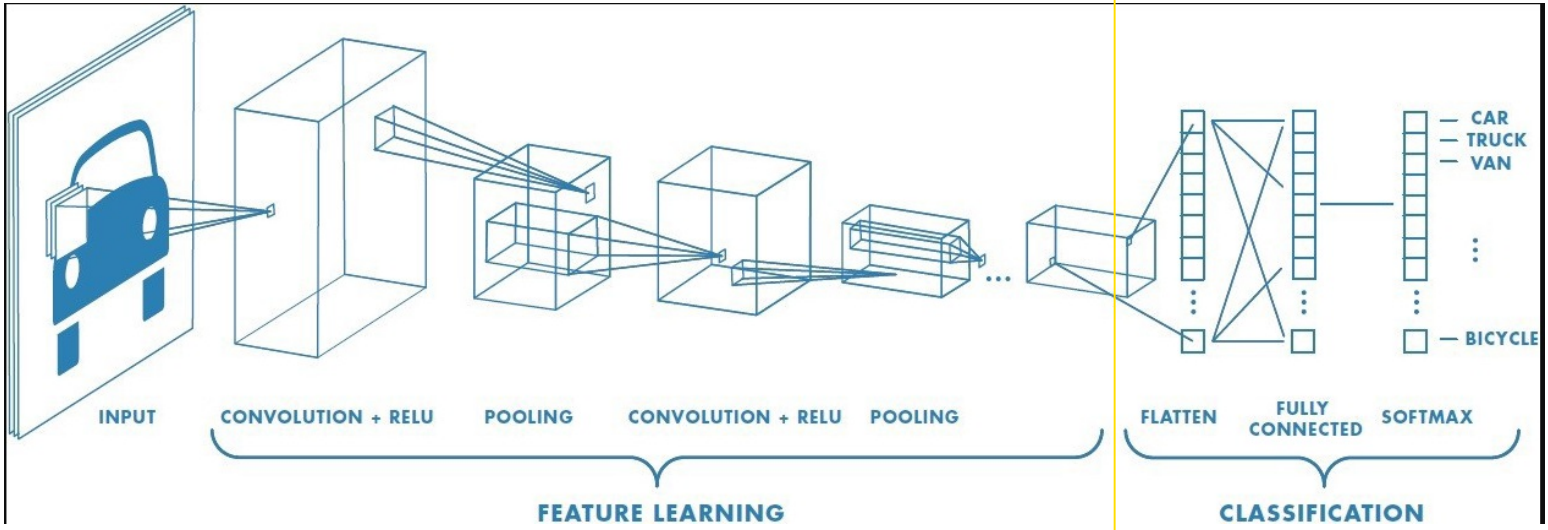
Convolutional neural network (CNN)

Convolutional + neural network



Convolutional

neural network

Convolutional neural network **คือ** NN **ที่ได้รับ** input **เป็น** data **ที่ผ่าน** convolutional **มาแล้ว เพื่อตรวจจับ** Information **ที่สำคัญ** (**จุดที่สำคัญ หรือ** feture)

ตัวอย่าง Data ที่ผ่าน Convolutional มาแล้ว

Pixel 1 Pixel 1 Pixel D

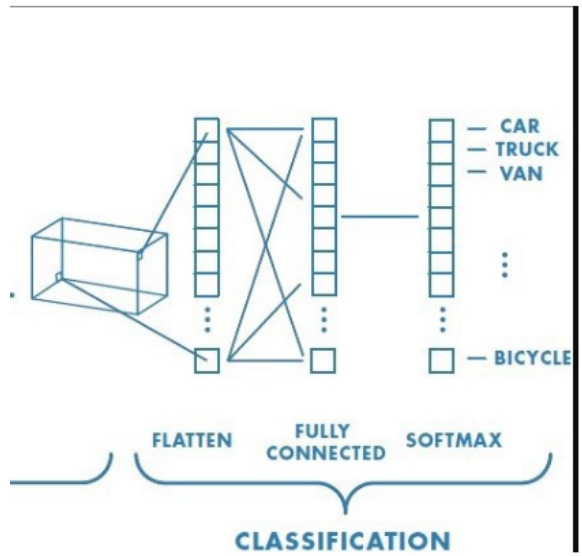
รูปที่ 1
รูปที่ 2
รูปที่ 3

$$X = \begin{bmatrix} x_1^1 & x_1^2 & \cdots & x_1^D \\ x_2^1 & x_2^2 & \cdots & x_2^D \\ x_3^1 & x_3^2 & \cdots & x_3^D \\ \vdots & \vdots & \ddots & \vdots \\ x_N^1 & x_N^2 & \cdots & x_N^D \end{bmatrix}, Y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_N \end{bmatrix}$$

รูปที่ N

car
Truck
Bicycle
N

x_3^D คือ Feature ที่ D ของ Example ที่ 3



neural network

https://medium.com/@natthawatphongchit/มาลองดูวิธีการคิดของ-cnn-กัน-e3f5d73eebaa

EX input >> conv >> RELU >> pooling >> FC

Input (6x6x1)

0	2	2	2	0	0
0	1	0	0	1	0
0	0	0	2	0	0
0	0	2	0	0	0
0	1	0	0	0	1
1	2	1	2	2	2

Filter W0 (3x3x1)

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

Filter W1 (3x3x1)

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

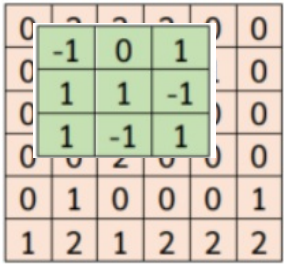
Bias b0 (1x1x1)

0

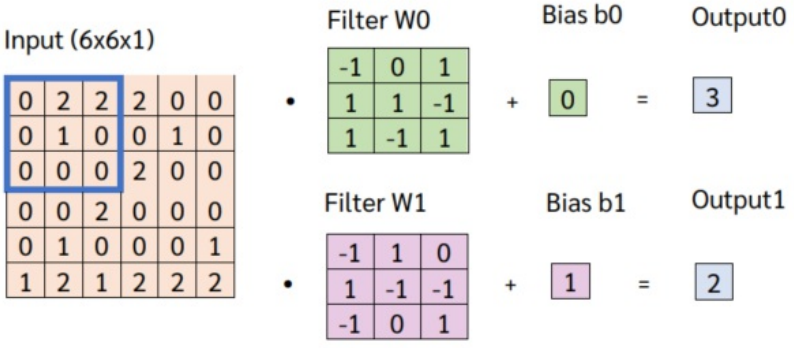
Bias b1 (1x1x1)

1

กำหนดให้ strid = 1 (Fiter W0)

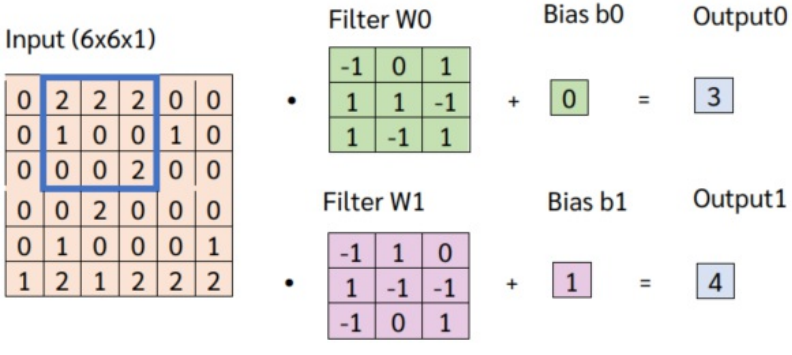


กำหนดให้ strid = 1 (Fiter W0)



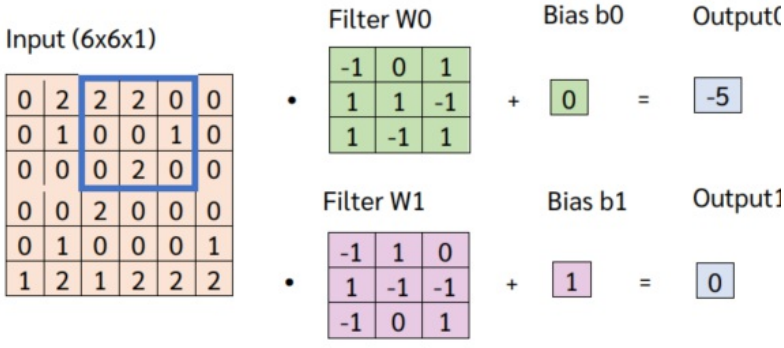
(0)(-1) + (2)(0) + (2)(1) +
(0)(1) + (1)(1) + (0)(-1) +
(0)(1) + (0)(-1) + (0)(1) = 3
>> 3 + bias(0) = 3

(0)(-1) + (2)(1) + (2)(0) +
(0)(1) + (1)(-1) + (0)(-1) +
(0)(-1) + (0)(0) + (0)(1) = 1
>> 1 + bias(1) = 2



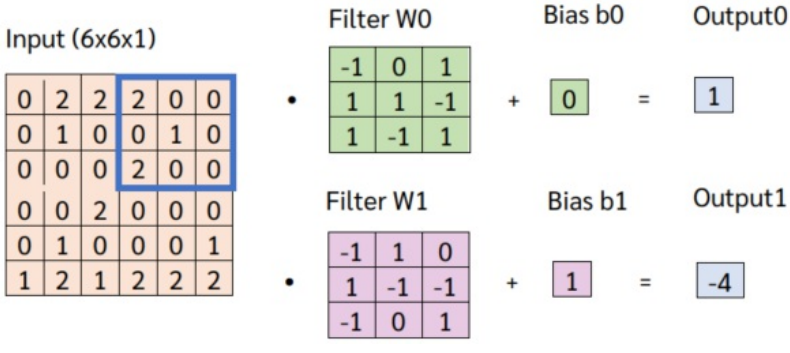
(2)(-1) + (2)(0) + (2)(1) +
(1)(1) + (0)(1) + (0)(-1) +
(0)(1) + (0)(-1) + (2)(1) = 3
>> 3 + bias(0) = 3

(2)(-1) + (2)(1) + (2)(0) +
(1)(1) + (0)(-1) + (0)(-1) +
(0)(-1) + (0)(0) + (2)(1) = 3
>> 3 + bias(1) = 4



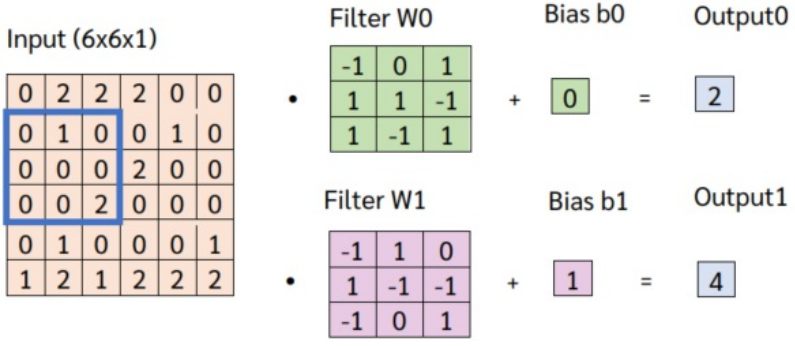
(2)(-1) + (2)(0) + (0)(1) +
(0)(1) + (0)(1) + (1)(-1) +
(0)(1) + (2)(-1) + (0)(1) = -5
>> -5 + bias(0) = -5

(2)(-1) + (2)(1) + (0)(0) +
(0)(1) + (0)(-1) + (1)(-1) +
(0)(-1) + (2)(0) + (0)(1) = -1
>> -1 + bias(1) = 3



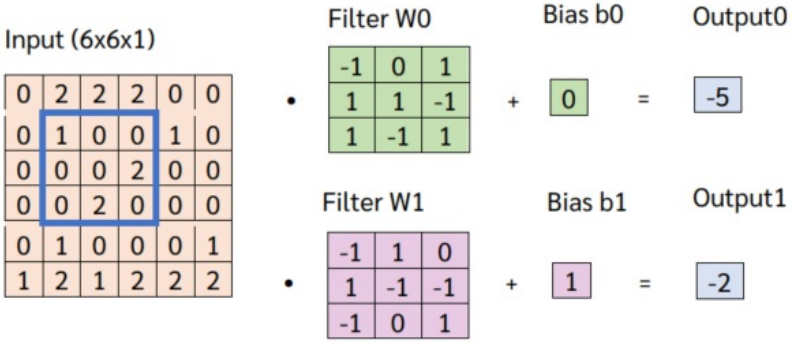
(2)(-1) + (0)(0) + (0)(1) +
(0)(1) + (1)(1) + (0)(-1) +
(2)(1) + (0)(-1) + (0)(1) = 1
>> 1 + bias(0) = 1

(2)(-1) + (0)(1) + (0)(0) +
(0)(1) + (1)(-1) + (0)(-1) +
(2)(-1) + (0)(0) + (0)(1) = -5
>> -5 + bias(1) = -4



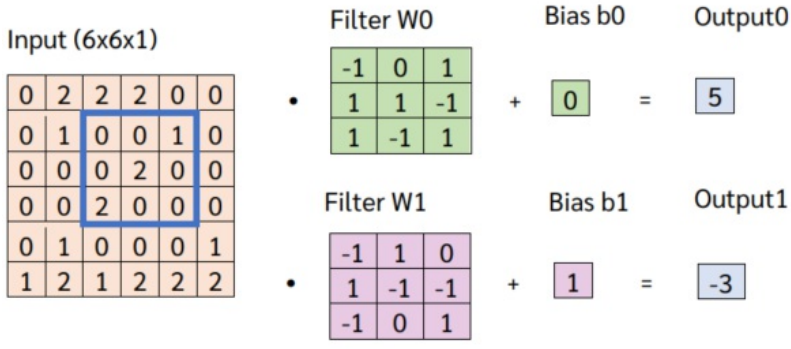
(0)(-1) + (1)(0) + (0)(1) +
(0)(1) + (0)(1) + (0)(-1) +
(0)(1) + (0)(-1) + (2)(1) = 2
>> 2 + bias(0) = 2

(0)(-1) + (1)(1) + (0)(0) +
(0)(1) + (0)(-1) + (0)(-1) +
(0)(-1) + (0)(0) + (2)(1) = 3
>> 3 + bias(1) = 4



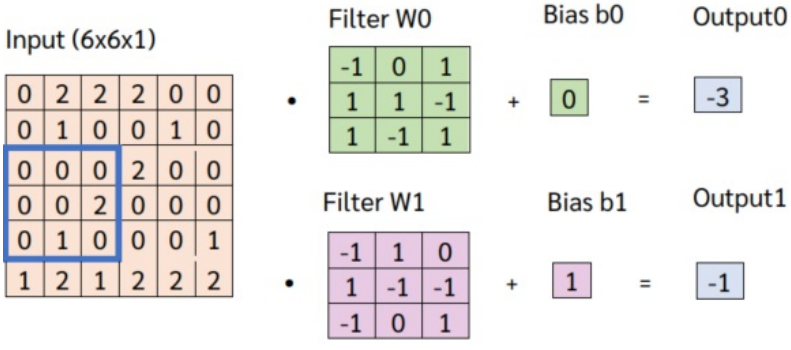
(1)(-1) + (0)(0) + (0)(1) +
(0)(1) + (0)(1) + (2)(-1) +
(0)(1) + (2)(-1) + (0)(1) = -5
>> -5 + bias(0) = -5

(1)(-1) + (0)(1) + (0)(0) +
(0)(1) + (0)(-1) + (2)(-1) +
(0)(-1) + (2)(0) + (0)(1) = -3
>> -3 + bias(1) = -2



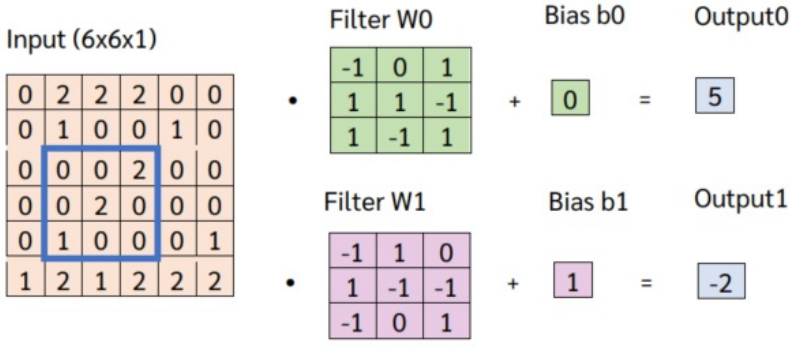
(0)(-1) + (0)(0) + (1)(1) +
(0)(1) + (2)(1) + (0)(-1) +
(2)(1) + (0)(-1) + (0)(1) = -5
>> -5 + bias(0) = -5

(0)(-1) + (0)(1) + (1)(0) +
(0)(1) + (2)(-1) + (0)(-1) +
(2)(-1) + (0)(0) + (0)(1) = -4
>> -4 + bias(1) = -3



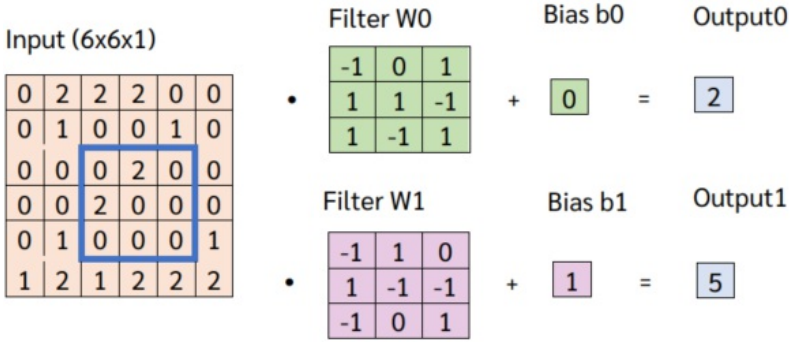
(0)(-1) + (0)(0) + (0)(1) +
(0)(1) + (0)(1) + (2)(-1) +
(0)(1) + (1)(-1) + (0)(1) = -3
>> -3 + bias(0) = -3

(0)(-1) + (0)(1) + (0)(0) +
(0)(1) + (0)(-1) + (2)(-1) +
(0)(-1) + (1)(0) + (0)(1) = -2
>> -2 + bias(1) = -1



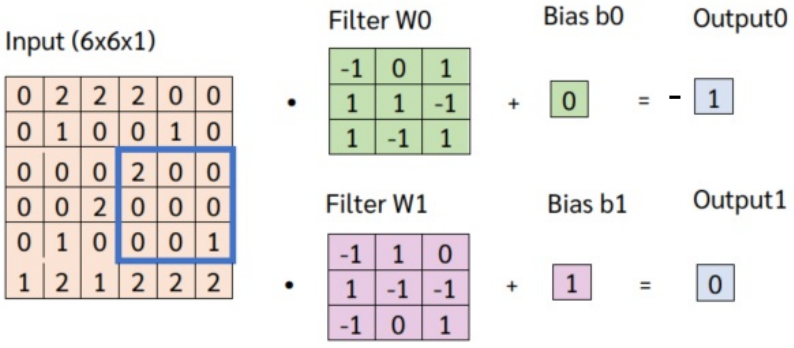
(0)(-1) + (0)(0) + (2)(1) +
(0)(1) + (2)(1) + (0)(-1) +
(1)(1) + (0)(-1) + (0)(1) = 5
>> 5 + bias(0) = 5

(0)(-1) + (0)(1) + (2)(0) +
(0)(1) + (2)(-1) + (0)(-1) +
(1)(-1) + (0)(0) + (0)(1) = -3
>> -3 + bias(1) = -2



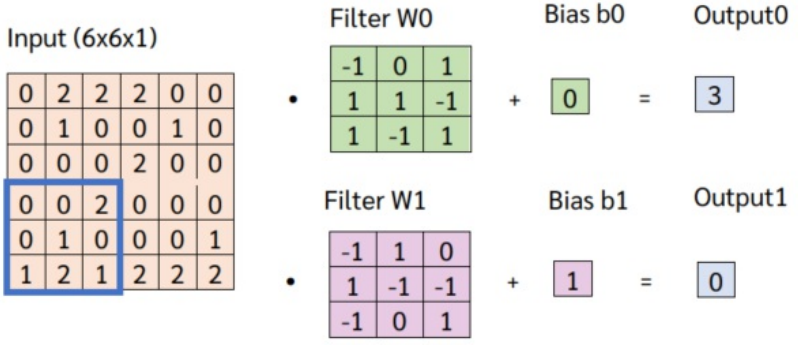
(0)(-1) + (2)(0) + (0)(1) +
(2)(1) + (0)(1) + (0)(-1) +
(0)(1) + (0)(-1) + (0)(1) = 2
>> 2 + bias(0) = 2

(0)(-1) + (2)(1) + (0)(0) +
(2)(1) + (0)(-1) + (0)(-1) +
(0)(-1) + (0)(0) + (0)(1) = 4
>> 4 + bias(1) = 5



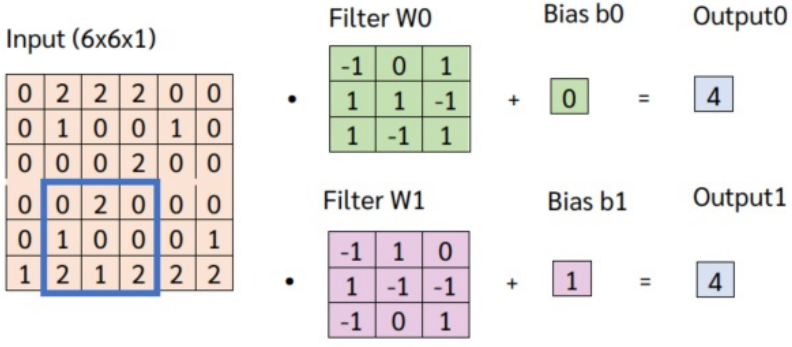
(2)(-1) + (0)(0) + (0)(1) +
(0)(1) + (0)(1) + (0)(-1) +
(0)(1) + (0)(-1) + (1)(1) = -1
>> -1 + bias(0) = -1

(2)(-1) + (0)(1) + (0)(0) +
(0)(1) + (0)(-1) + (0)(-1) +
(0)(-1) + (0)(0) + (1)(1) = -1
>> -1 + bias(1) = 0



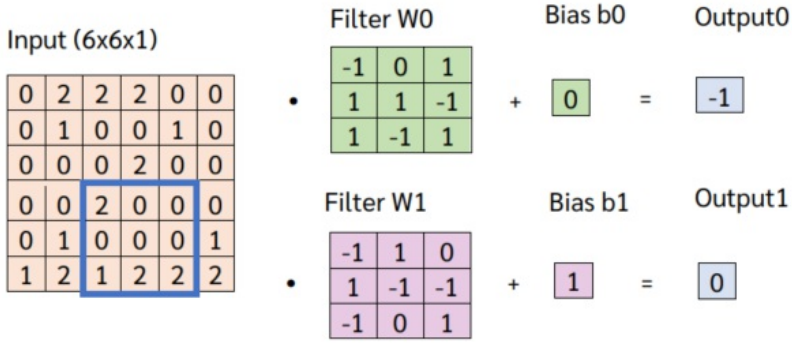
(0)(-1) + (0)(0) + (2)(1) +
(0)(1) + (1)(1) + (0)(-1) +
(1)(1) + (2)(-1) + (1)(1) = 3
>> 3 + bias(0) = 3

(0)(-1) + (0)(1) + (2)(0) +
(0)(1) + (1)(-1) + (0)(-1) +
(1)(-1) + (2)(0) + (1)(1) = -1
>> -1 + bias(1) = 0



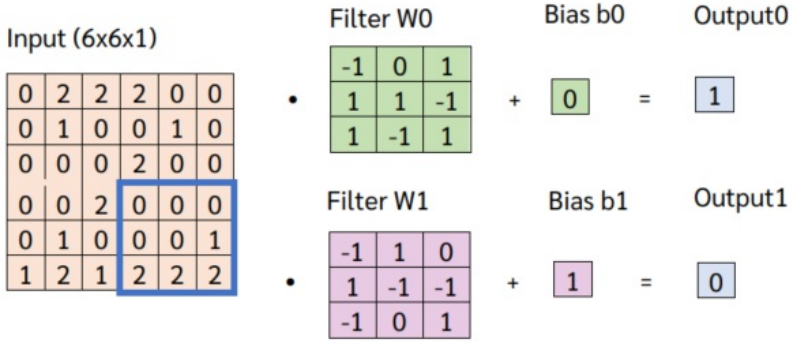
(0)(-1) + (2)(0) + (0)(1) +
(1)(1) + (0)(1) + (0)(-1) +
(2)(1) + (1)(-1) + (2)(1) = 4
>> 4 + bias(0) = 4

(0)(-1) + (2)(1) + (0)(0) +
(1)(1) + (0)(-1) + (0)(-1) +
(2)(-1) + (1)(0) + (2)(1) = 3
>> 3 + bias(1) = 4



(2)(-1) + (0)(0) + (0)(1) +
(0)(1) + (0)(1) + (0)(-1) +
(1)(1) + (2)(-1) + (2)(1) = -1
>> -1 + bias(0) = -1

(2)(-1) + (0)(1) + (0)(0) +
(0)(1) + (0)(-1) + (0)(-1) +
(1)(-1) + (2)(0) + (2)(1) = -1
>> -1 + bias(1) = 0



(0)(-1) + (0)(0) + (0)(1) +
(0)(1) + (0)(1) + (1)(-1) +
(2)(1) + (2)(-1) + (2)(1) = 1
>> 1 + bias(0) = 1

(0)(-1) + (0)(1) + (0)(0) +
(0)(1) + (0)(-1) + (1)(-1) +
(2)(-1) + (2)(0) + (2)(1) = -1
>> -1 + bias(1) = 0

Output (4x4x2)

output0

3	3	-5	1
2	-5	5	2
-3	5	2	-1
3	4	-1	1

output1

2	4	0	-4
4	-2	-3	4
-1	-2	5	0
0	4	0	0

สูตรหา shape output

[(N-F)/S] +1

[(6-3)/1] +1 = 4

- N = input size
- F = filter
- S = strid

Output (4x4x2)

output0

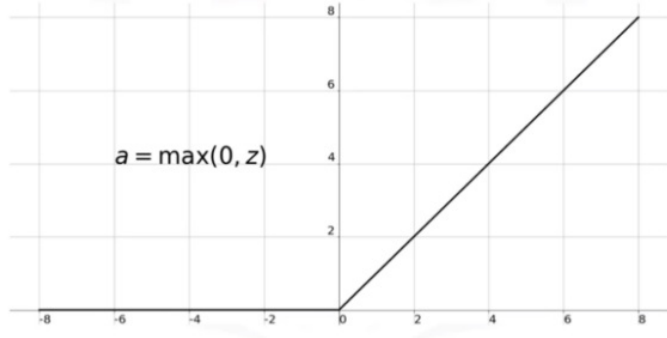
3	3	-5	1
2	-5	5	2
-3	5	2	-1
3	4	-1	1

output1

2	4	0	-4
4	-2	-3	4
-1	-2	5	0
0	4	0	0

ReLU Function

น้อยกว่า 0 ให้เท่ากับ 0



การคำนวณ ReLU

output0 = max(0, input0)
output1 = max(0, input1)

Output (4x4x2)

output0

3	3	0	1
2	0	5	2
0	5	2	0
3	4	0	1

output1

2	4	0	0
4	0	0	4
0	0	5	0
0	4	0	0

- max pooling
- Average pooling
- min pooling

ขั้นตอนที่ 3: Pooling Layer

กำหนดให้เป็น Max Pooling และค่า hyperparameters เป็นดังนี้

- Filter size = 2x2
- Stride = 2
- Zero padding = 1

Input (4x4x2)

input0

3	3	0	1
2	0	5	2
0	5	2	0
3	4	0	1

input1

2	4	0	0
4	0	0	4
0	0	5	0
0	4	0	0

การคำนวณ Pooling Layer

input0

3	3	0	1
2	0	5	2
0	5	2	0
3	4	0	1

=>

3	5
5	2

input1

2	4	0	0
4	0	0	4
0	0	5	0
0	4	0	0

=>

4	4
4	5

Output (2x2x2)

output0

3	5
5	2

output1

4	4
4	5

Zero(0/1) Padding (เพิ่มขนาด Input)

เช่น Input เดิมเป็น 5x5

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

Image

X

1	2	3
-4	7	4
2	-5	1

Filter / Kernel

=

21	59	37	-19	2
30	51	66	20	43
-14	31	49	101	-19
59	15	53	-2	21
49	57	64	76	10

Feature

Padding Input กลายเป็น 7x7

สูตรหา shape output

[(N-F)/S] +1

[(7-3)/1] +1 = 5

- N = input size
- F = filter
- S = strid

สูตรหา shape output

[(N + 2P - F)/S] +1

[(5 + (2*1) - 3)/1] +1 = 5

- N = input size (เดิม)
- P =padding
- F = filter
- S = strid

```python
import torch.nn as nn
import torch.nn.functional as F

class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)
    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 16 * 5 * 5)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x


net = Net()

device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
net.to(device)
```

channel = 3

Filters

Kernel size

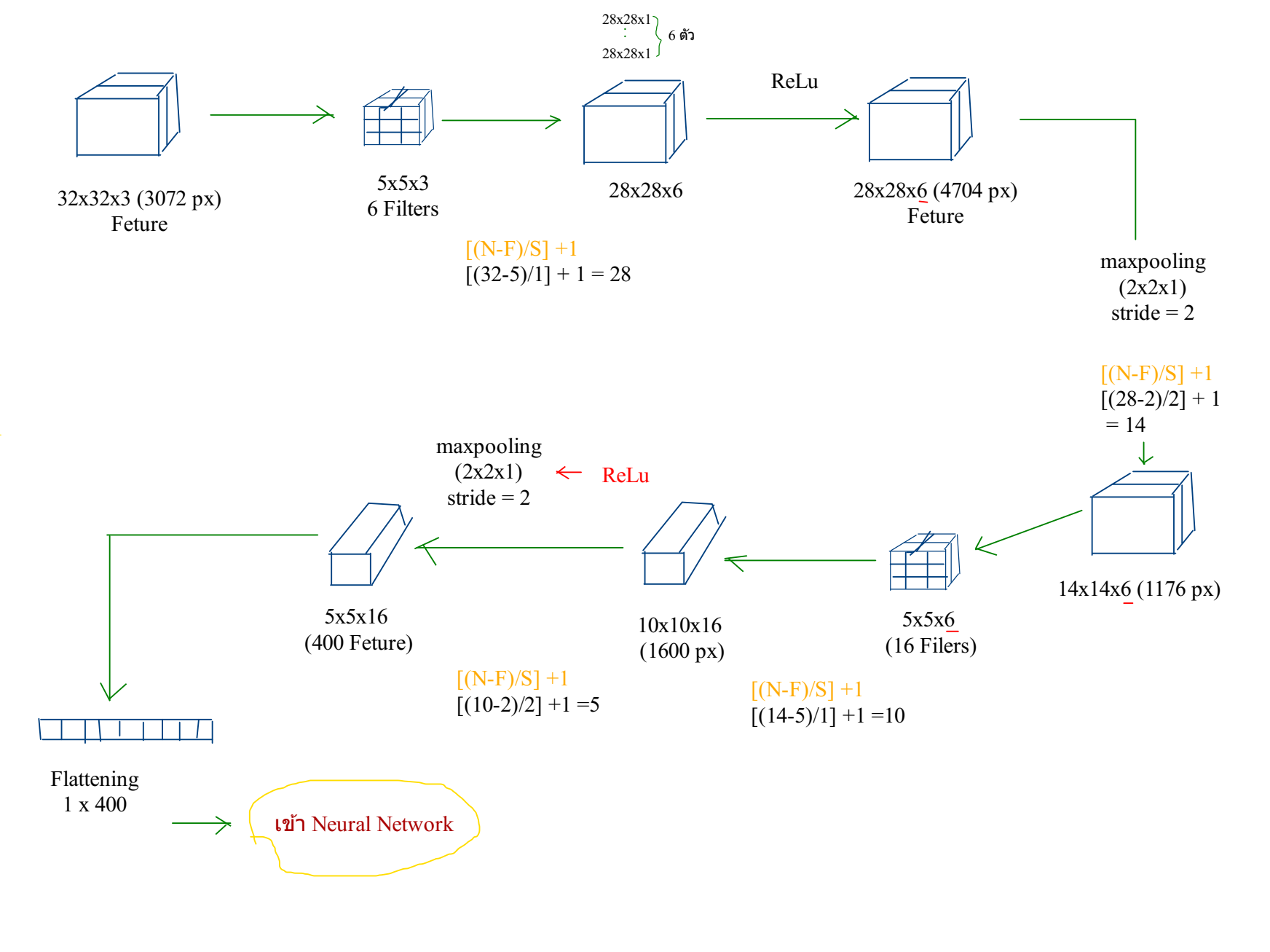
2 Kernel size, 2 stride

16*5*5 = 400 = input

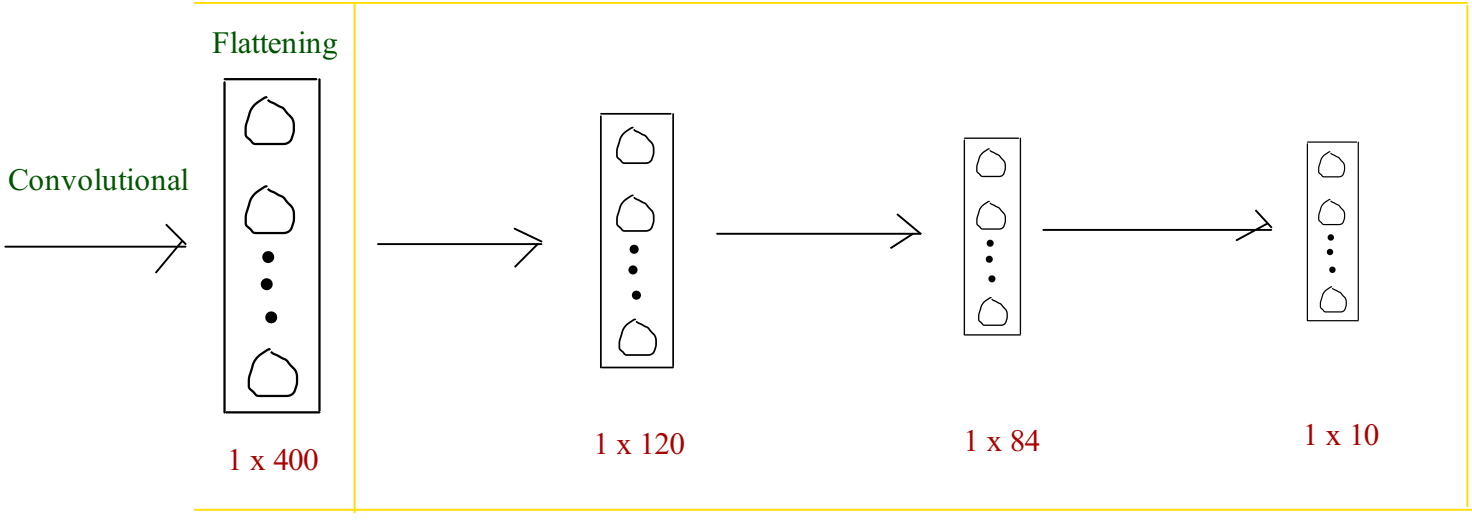
120 = output

84 = input

10 = output


28x28x1
28x28x1
6 ตัว
ReLu
32x32x3 (3072 px)
Feture
5x5x3
6 Filters
28x28x6
28x28x6 (4704 px)
Feture
[(N-F)/S] +1
[(32-5)/1] + 1 = 28
maxpooling
(2x2x1)
stride = 2
[(N-F)/S] +1
[(28-2)/2] + 1
= 14
14x14x6 (1176 px)
5x5x6
(16 Filers)
[(N-F)/S] +1
[(14-5)/1] +1 =10
10x10x16
(1600 px)
[(N-F)/S] +1
[(10-2)/2] +1 =5
maxpooling
(2x2x1)
stride = 2
ReLu
5x5x16
(400 Feture)
Flattening
1 x 400
เข้า Neural Network

เข้า Neural Network




```
self.fc1 = nn.Linear(16 * 5 * 5, 120)
```

```
self.fc2 = nn.Linear(120, 84)
```

```
self.fc3 = nn.Linear(84, 10)
```