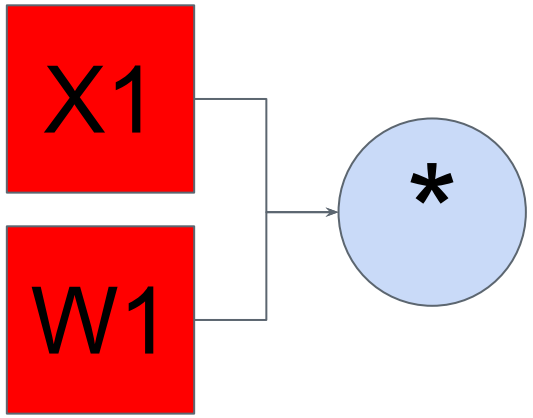
DATA 442: Neural Networks & Deep Learning

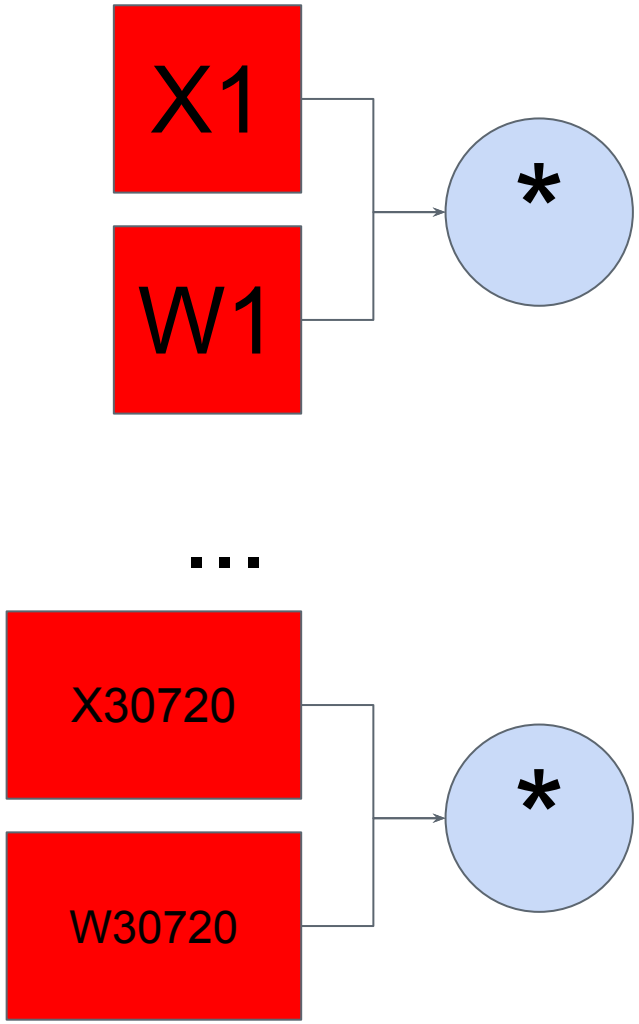
Dan Runfola – danr@wm.edu

icss.wm.edu/data442/

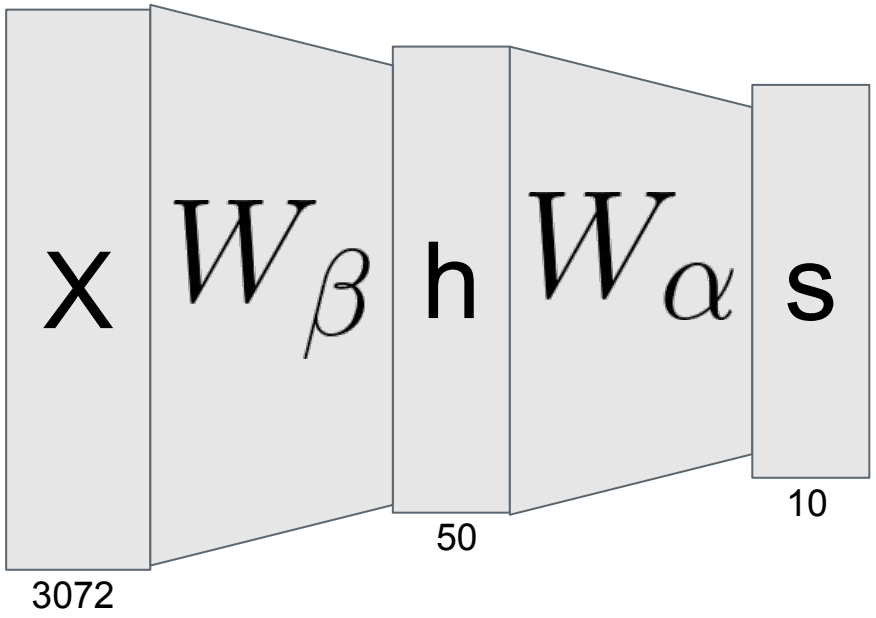


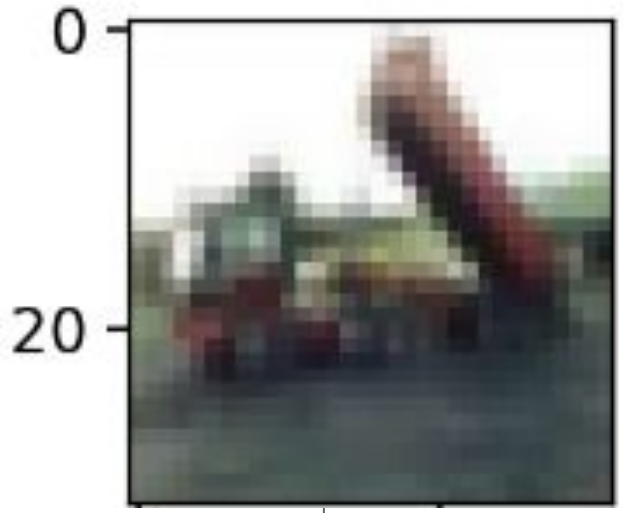

```
class MultiplicationNode():
    def forwardPass(input1,input2):
        output = input1 * input2
        self.input1 = input1
        self.input2 = input2
        return output

    def backwardPass(dOutput):
        dInput1 = self.input2 * dOutput
        dInput2 = self.input1 * dOutput
        return [dInput1, dInput2]
```


X1
W1
*
...
X30720
W30720
*

$$f = W_\alpha * max(0, W_\beta * X)$$





Input Image Array

1

3072

●

Weights for each Pixel, for each of 10 CIFAR classes

3072

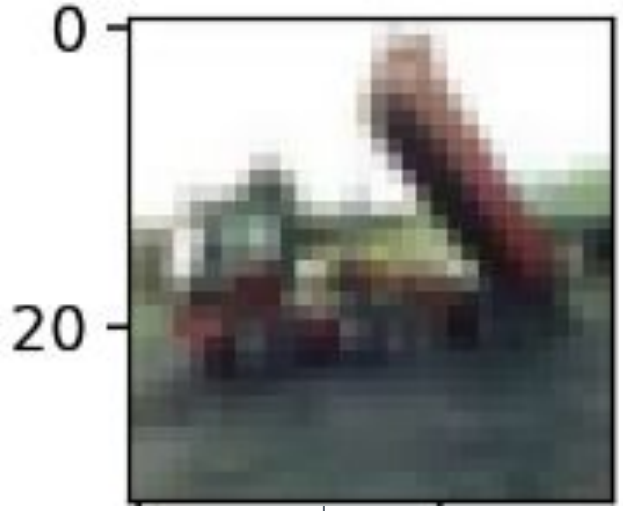
10

=

Output Scores Array

1

10



Input Image Array

1

3072

●

Weights for each Pixel, for each of 10 CIFAR classes

10

3072

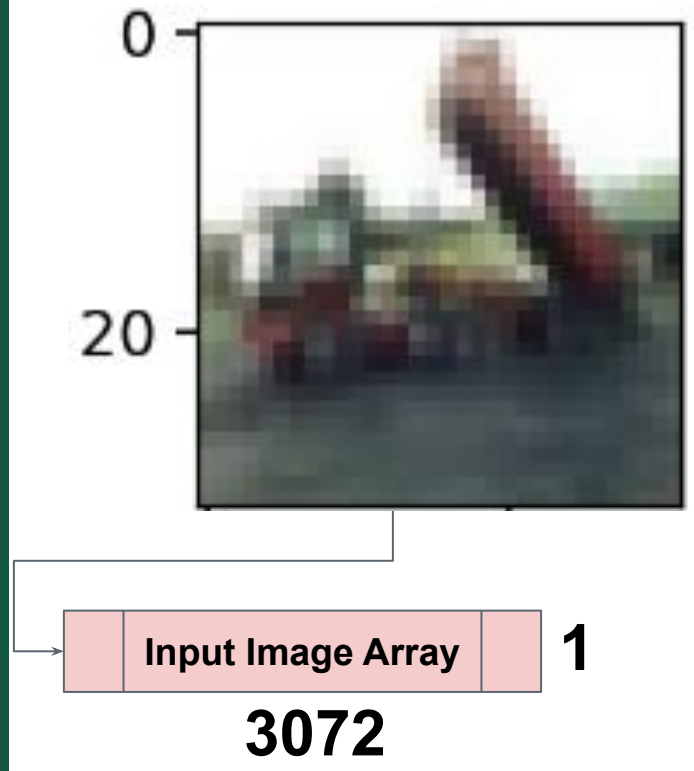
=

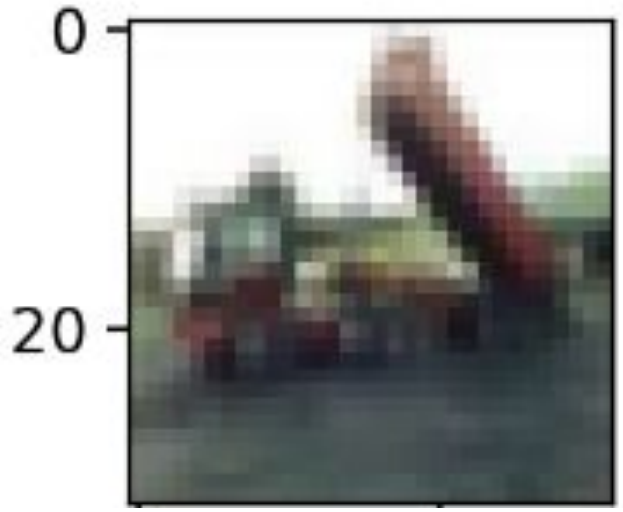
Output Scores Array

1

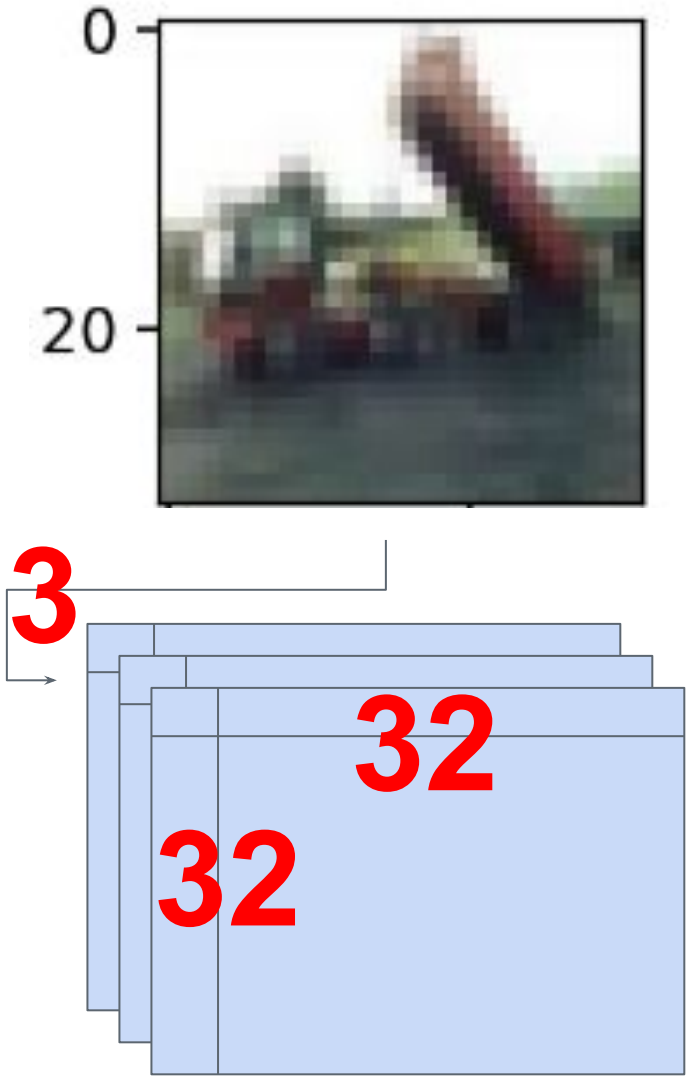
10

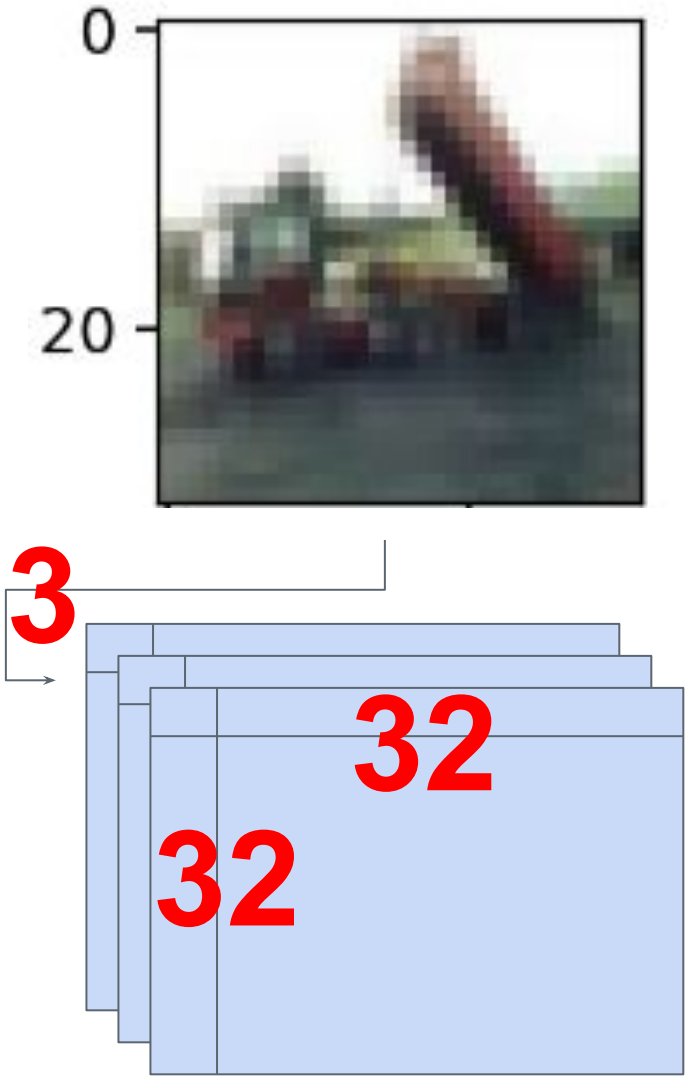
Each of the individual scores is the result of the dot product between a row of the weights matrix and the input array.


0
20
Input Image Array
1
3072


0
20

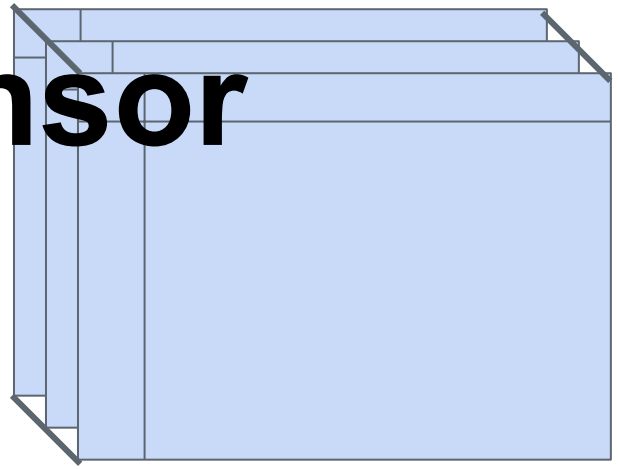
?

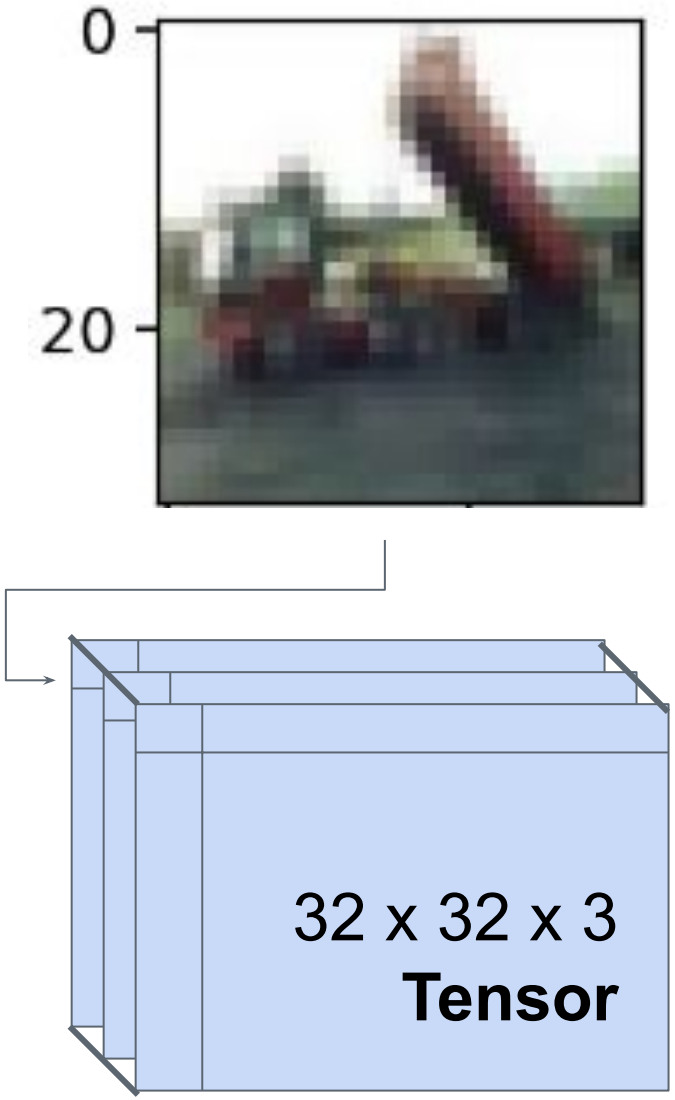

0
20
3
32
32

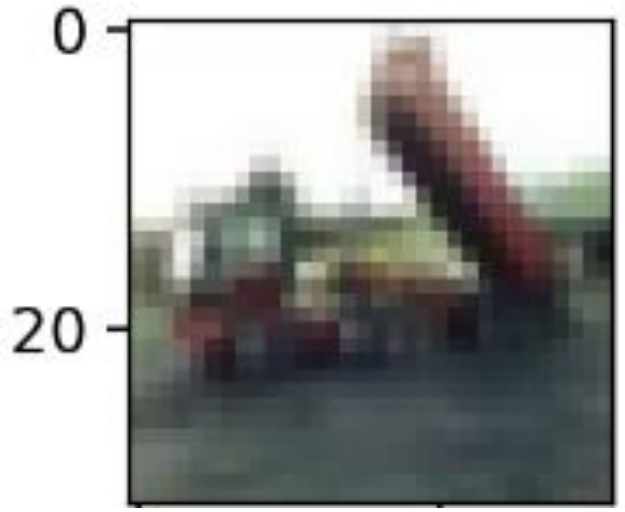


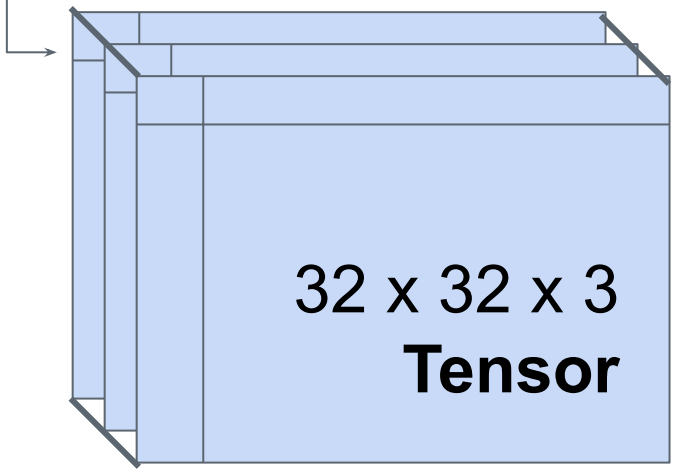
32 x 32 x 3

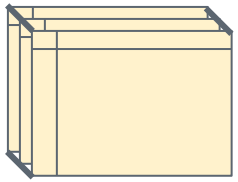
Tensor



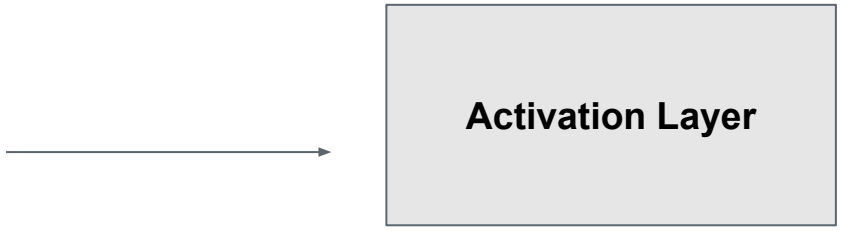

0
20
32 x 32 x 3
Tensor
5x5x3
Filter



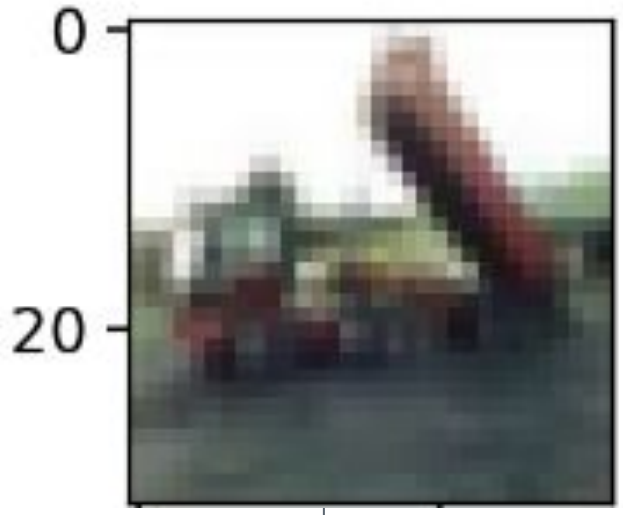




5x5x3
Filter



28 x 28 x 1
Matrix



Convolution (5x5x3 Filter)

Activation Layer (28 x 28 x 1 Matrix)

1

Input Activation Array

784

●

Weights for each Pixel, for each of 10 CIFAR classes

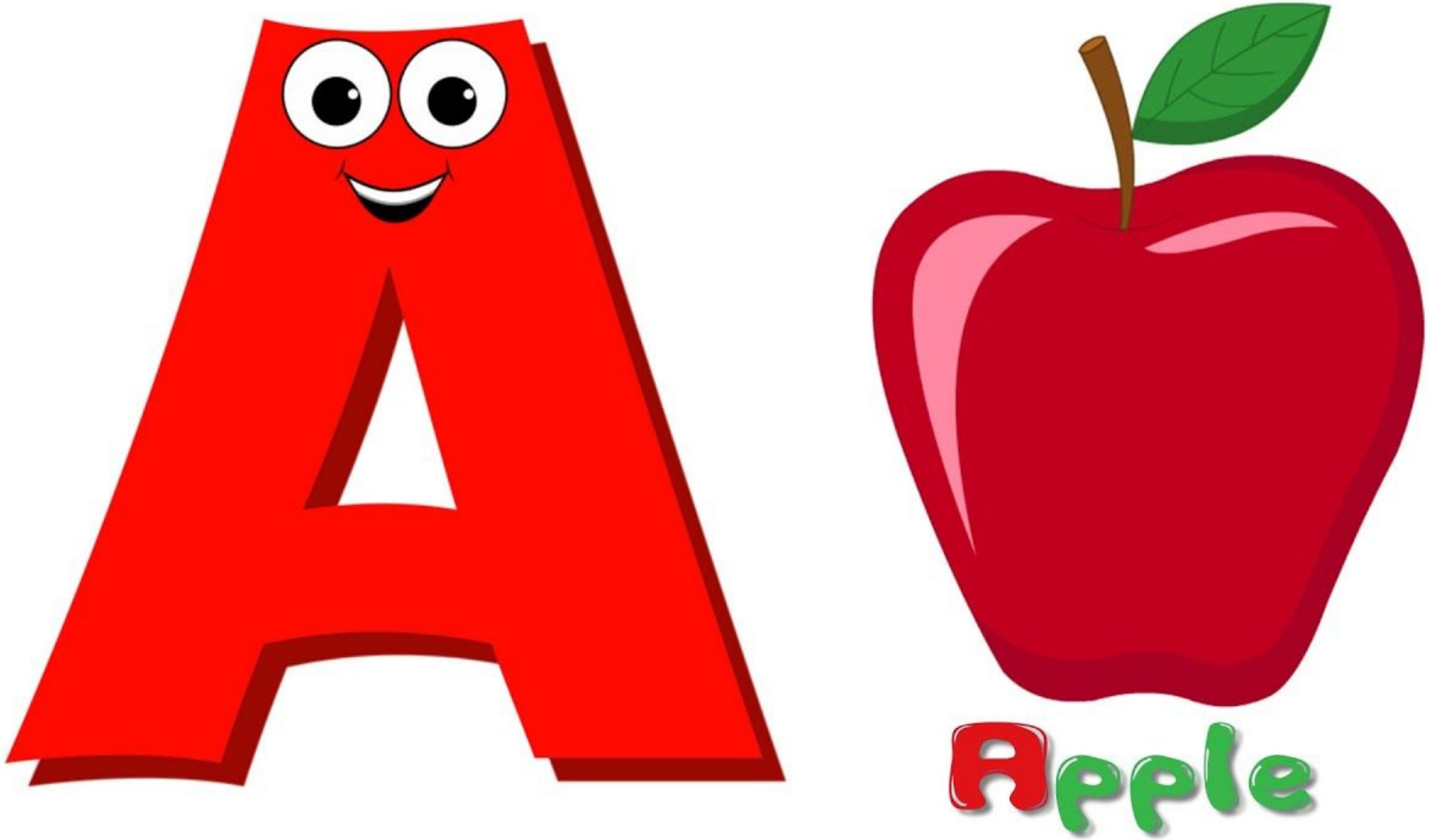
784

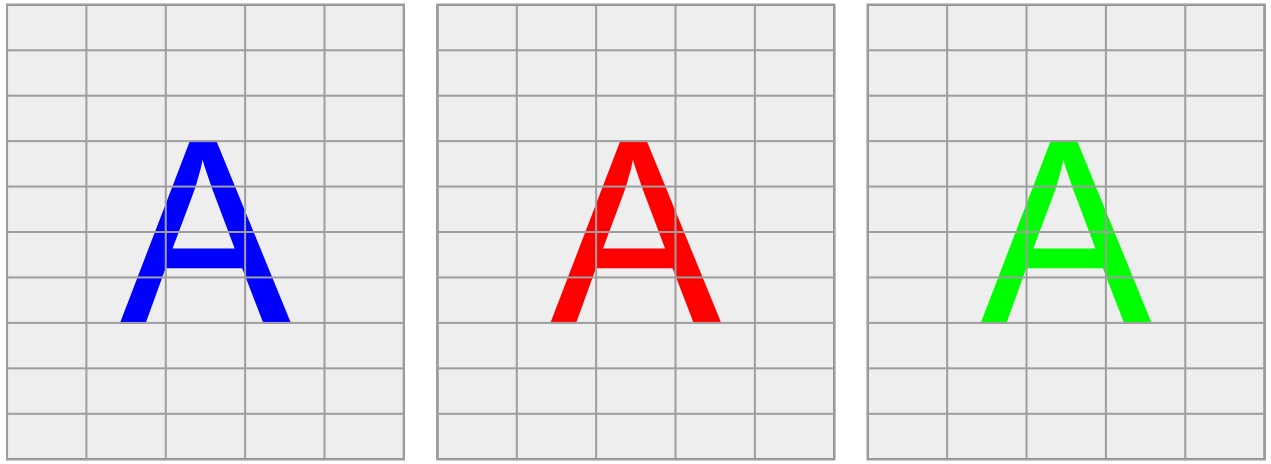
10

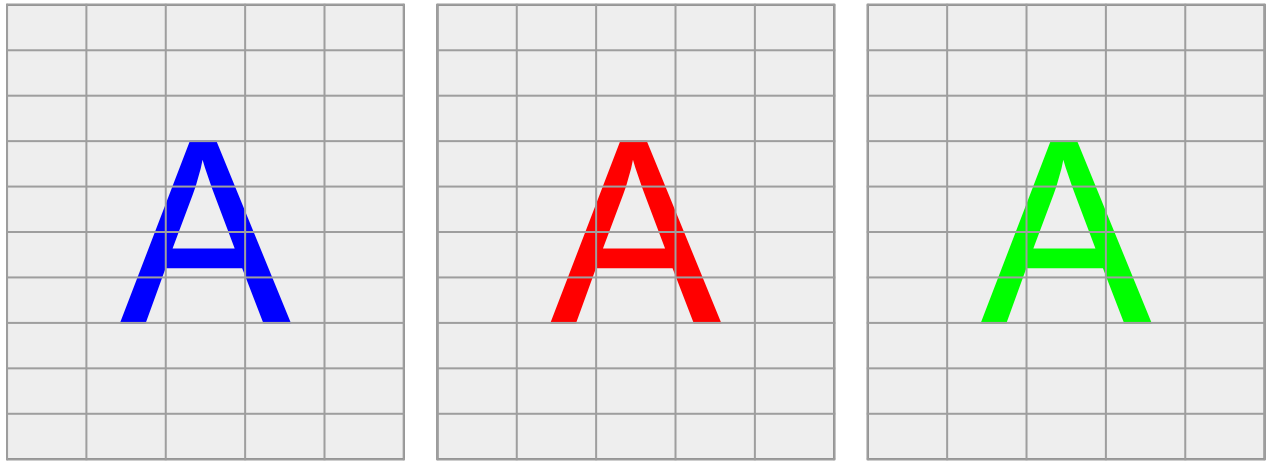
=

Output Scores Array

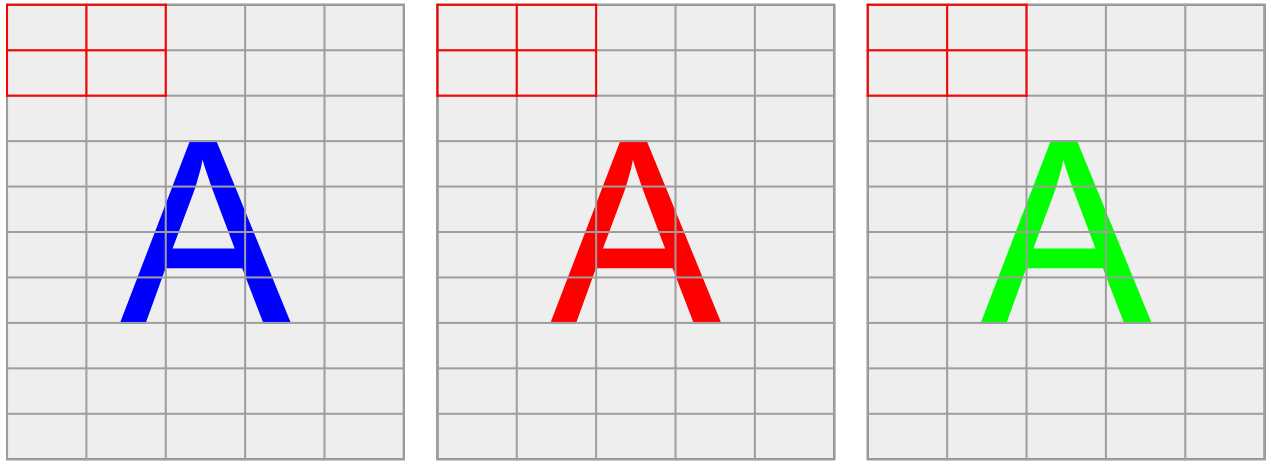
10


A
Apple

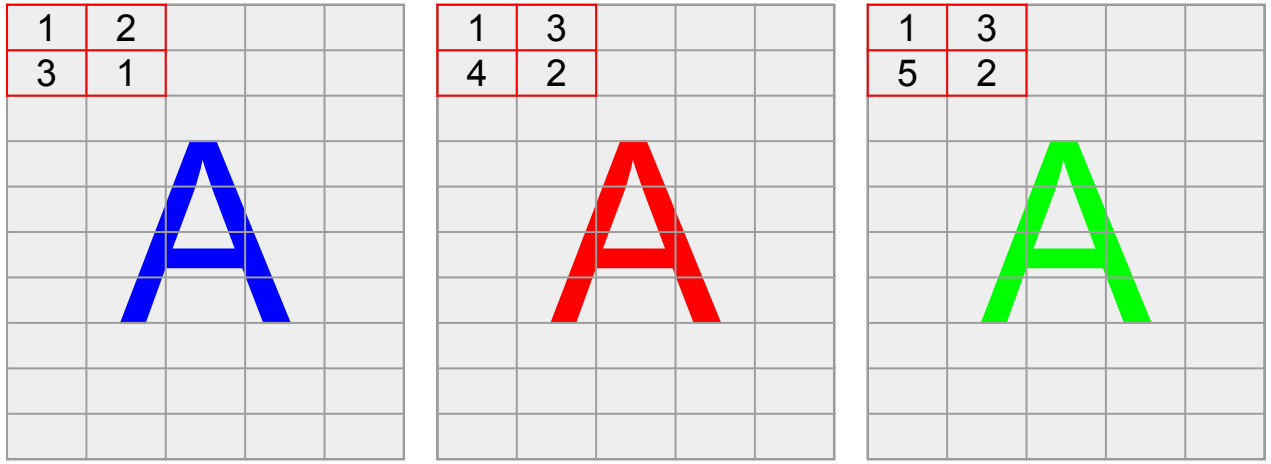

A
A
A



First, a *filter* is defined. This example is a 2x2x3 filter.

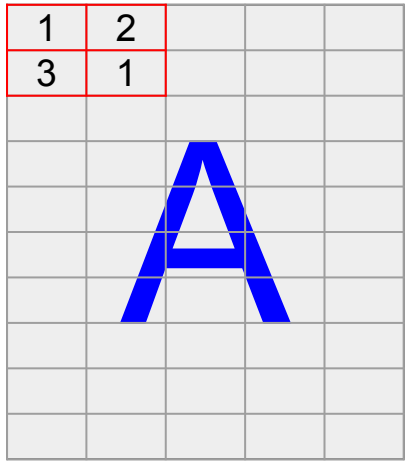


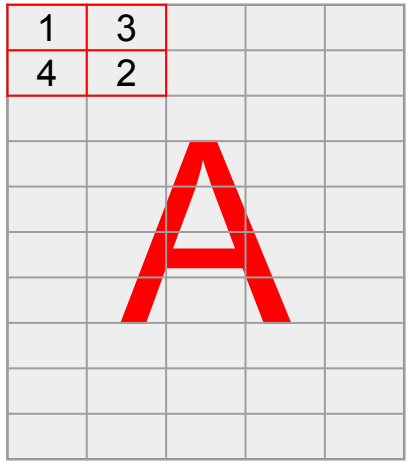
When we “convolve”, we are sliding our filter over each subset of the image.

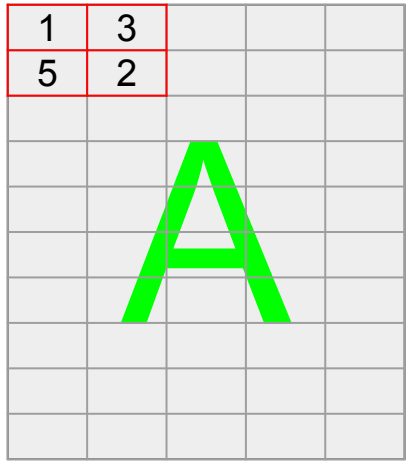


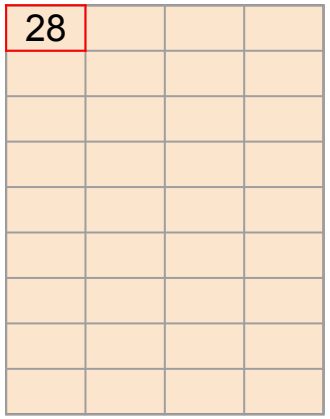
When we “convolve”, we are sliding our filter over each subset of the image.

The filter itself is defined based on weights - i.e., the simplest filter would have weights of “1” for every cell, so this convolution would be the sum of the 12 red cells **(28)**.

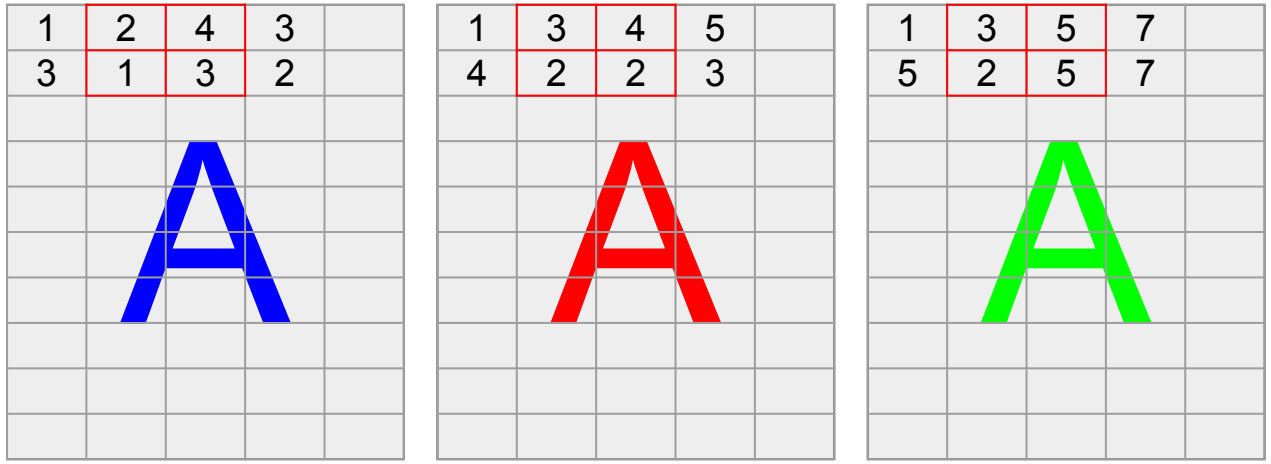




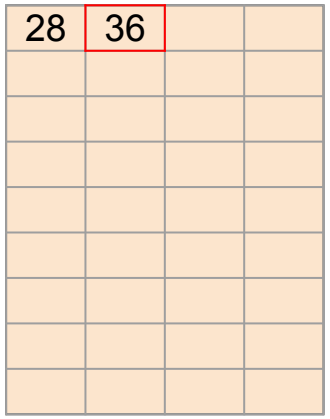


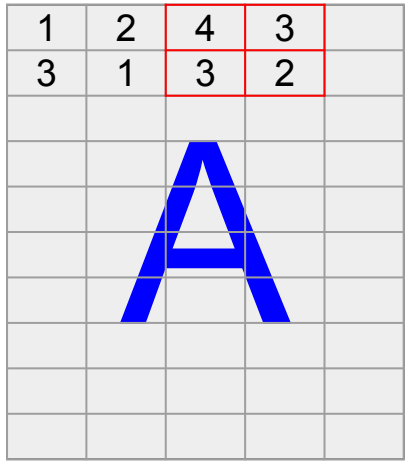


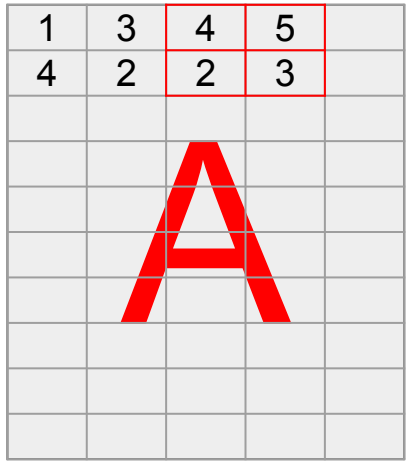
This becomes the first cell in the activation layer.

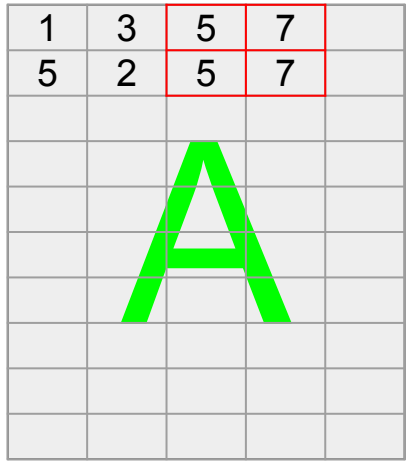


This becomes the **second** cell in the activation layer.

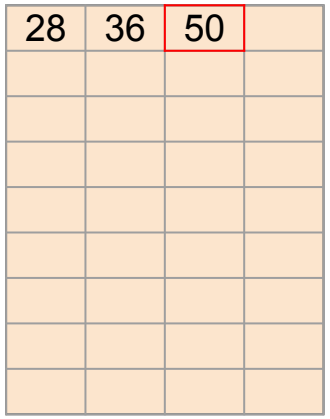


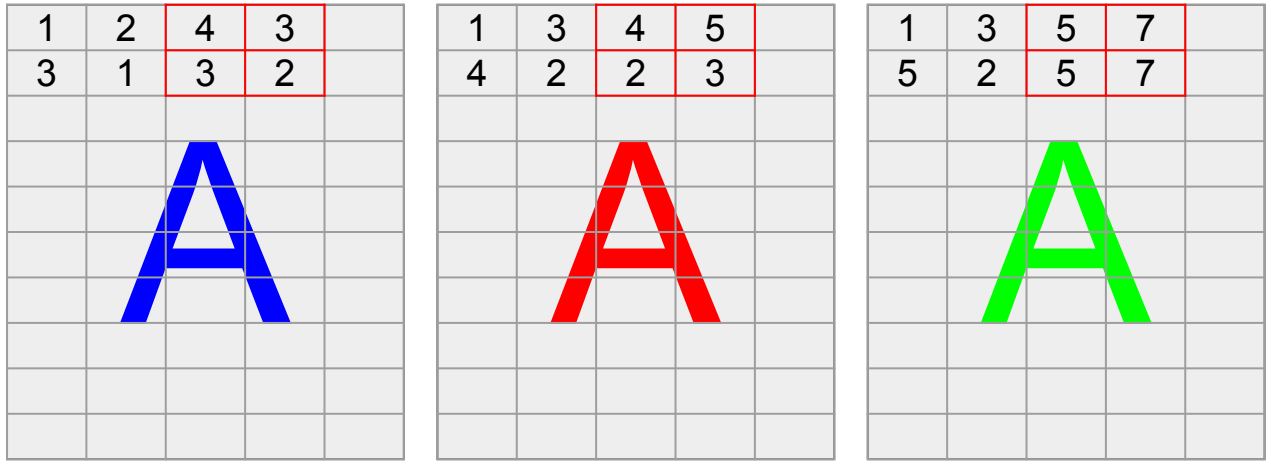






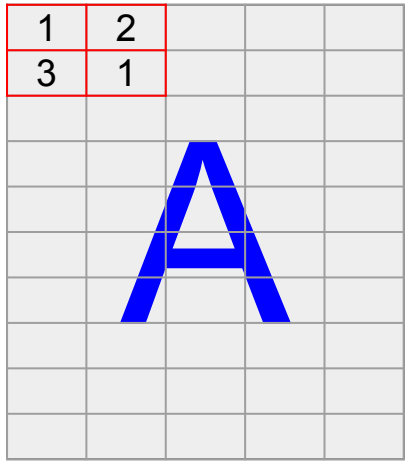
This becomes the **third** cell in the activation layer.

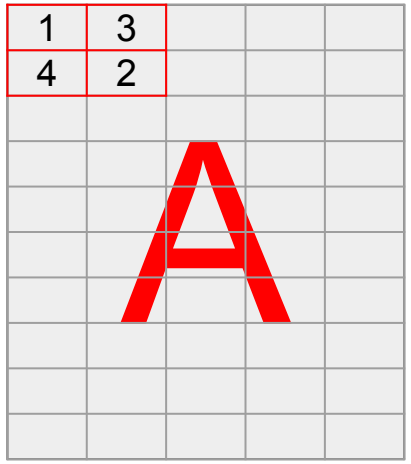


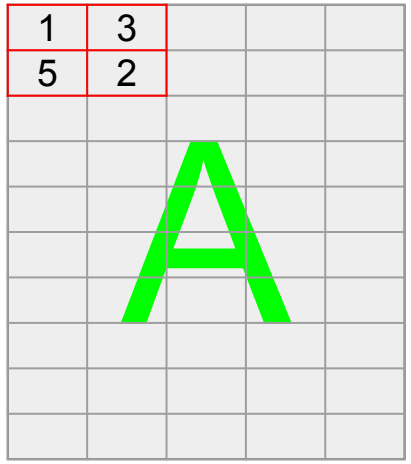


...and so on to define an activation layer.

28	36	50	45
35	45	15	43
15	1	65	25
15	5	38	41
25	6	78	45
35	15	15	15
65	25	35	5
15	5	68	2
78	8	97	15



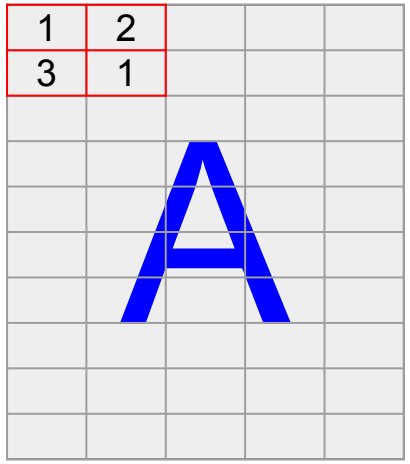


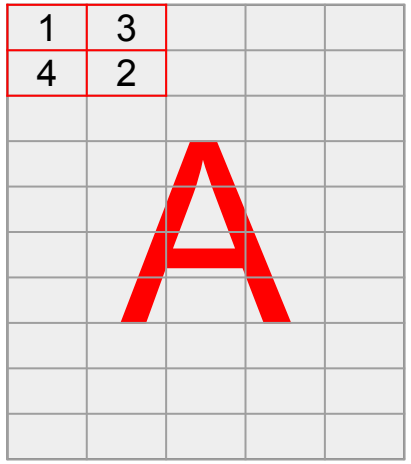


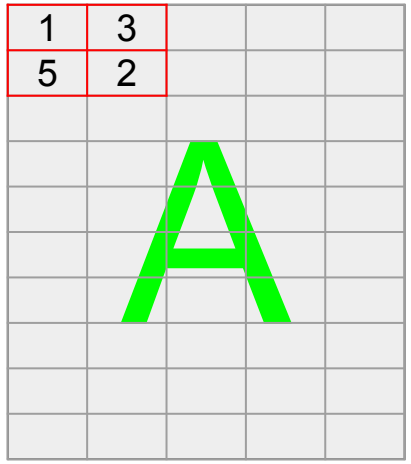
1	1
1	1

1	1
1	1

1	1
1	1



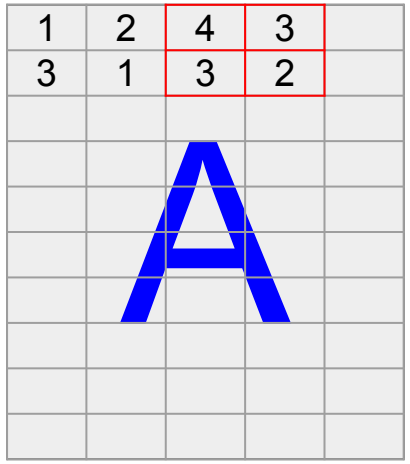


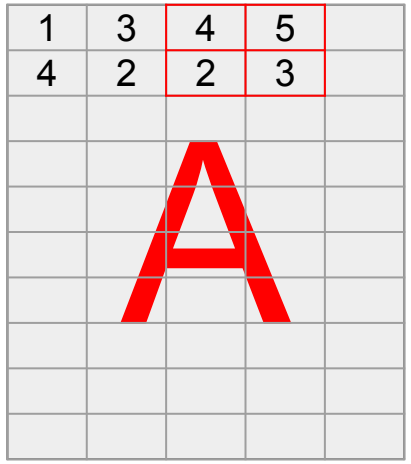


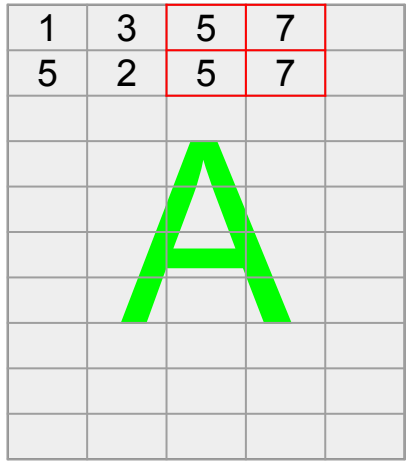
1	1
1	1

0	0
0	0

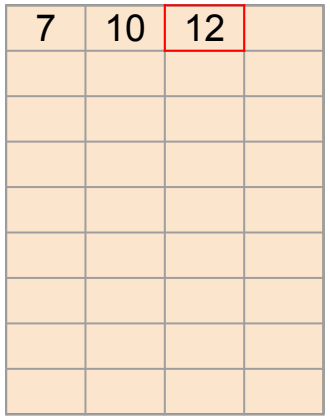
0	0
0	0

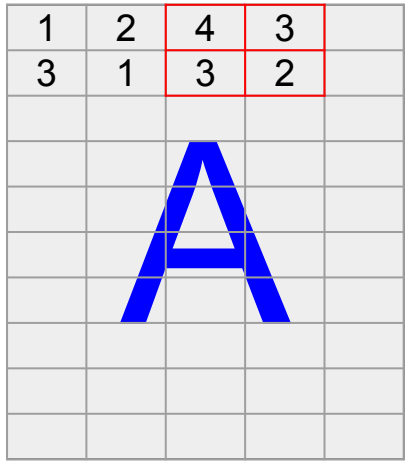


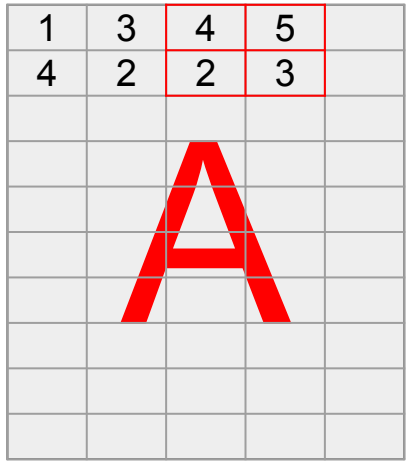


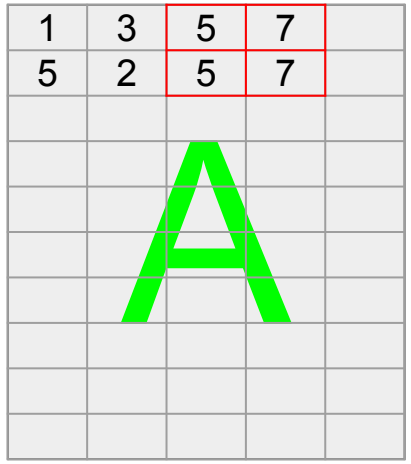


This would make a new activation layer, representing blue colors.

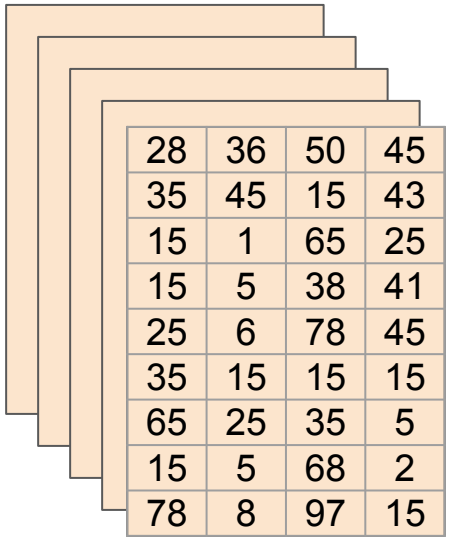


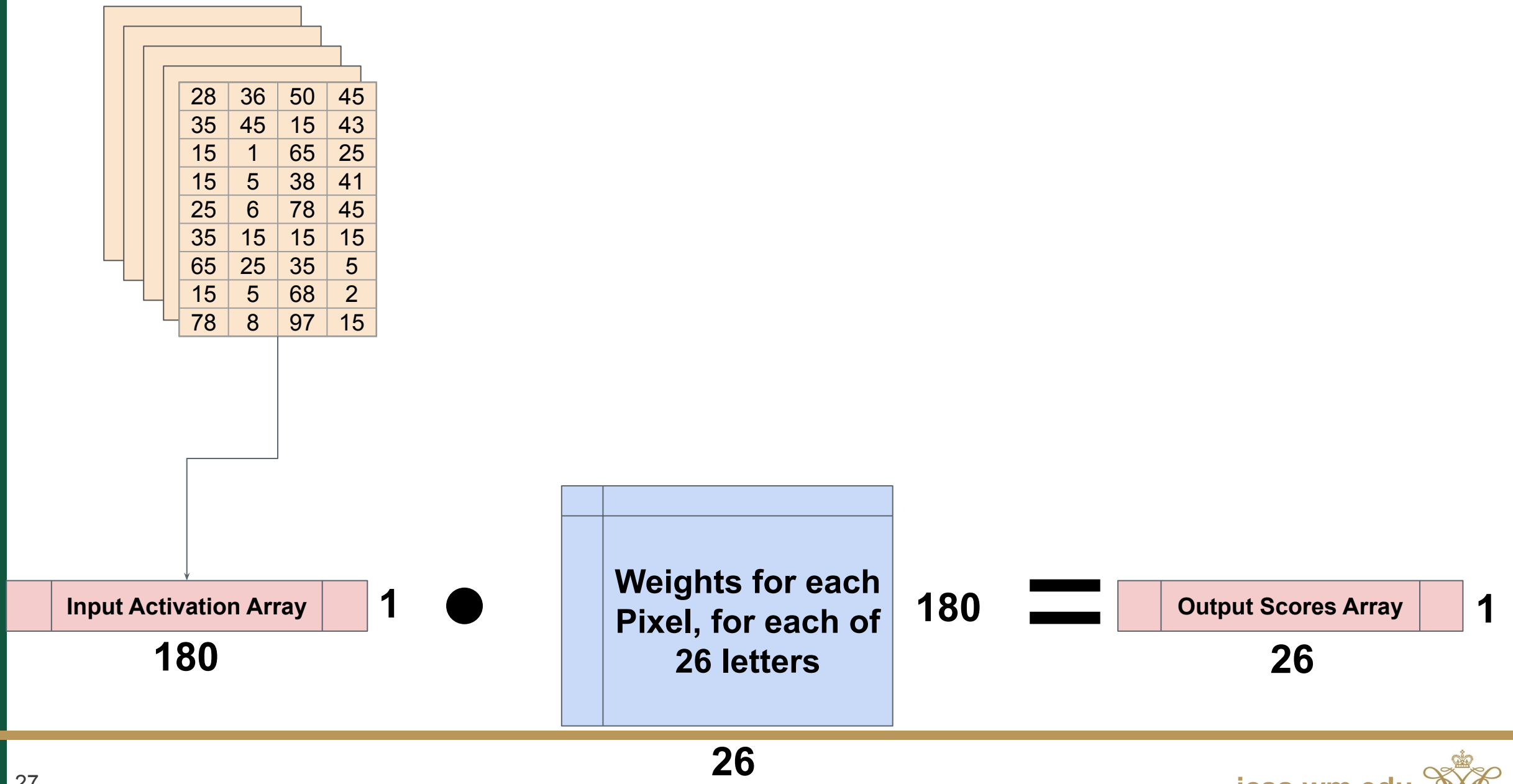


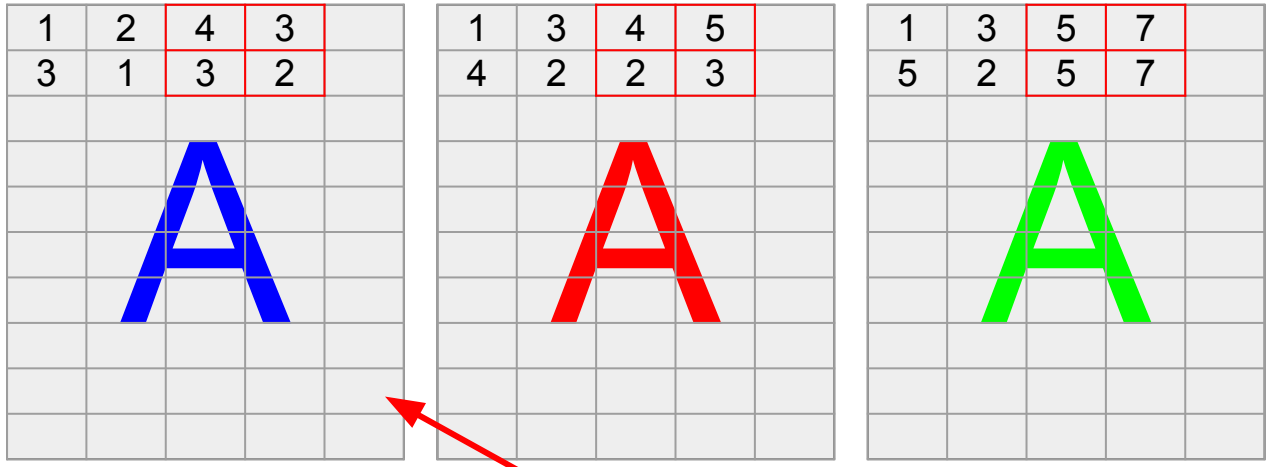




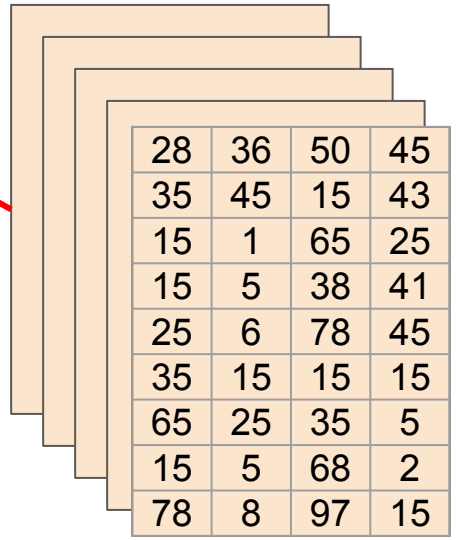
If we apply 5 different filters, we get 5 new activation layers.




28 36 50 45
35 45 15 43
15 1 65 25
15 5 38 41
25 6 78 45
35 15 15 15
65 25 35 5
15 5 68 2
78 8 97 15
Input Activation Array
1
180
Weights for each Pixel, for each of 26 letters
180
26
Output Scores Array
1
26



These activation layers then become the source images for the next round of the network.



All Color Filter

28	36	50	45
35	45	15	43
15	1	65	25
15	5	38	41
25	6	78	45
35	15	15	15
65	25	35	5
15	5	68	2
78	8	97	15

Blue Color Filter

2	6	50	45
35	4	15	3
5	1	5	5
15	5	38	41
25	6	78	45
8	15	4	15
65	25	35	5
15	5	3	2
78	8	97	15

Red Color Filter

1	36	50	45
25	4	15	43
44	1	65	25
9	5	0	41
25	6	6	4
75	10	15	15
65	25	35	5
15	5	68	20
78	8	97	15

Green Color Filter

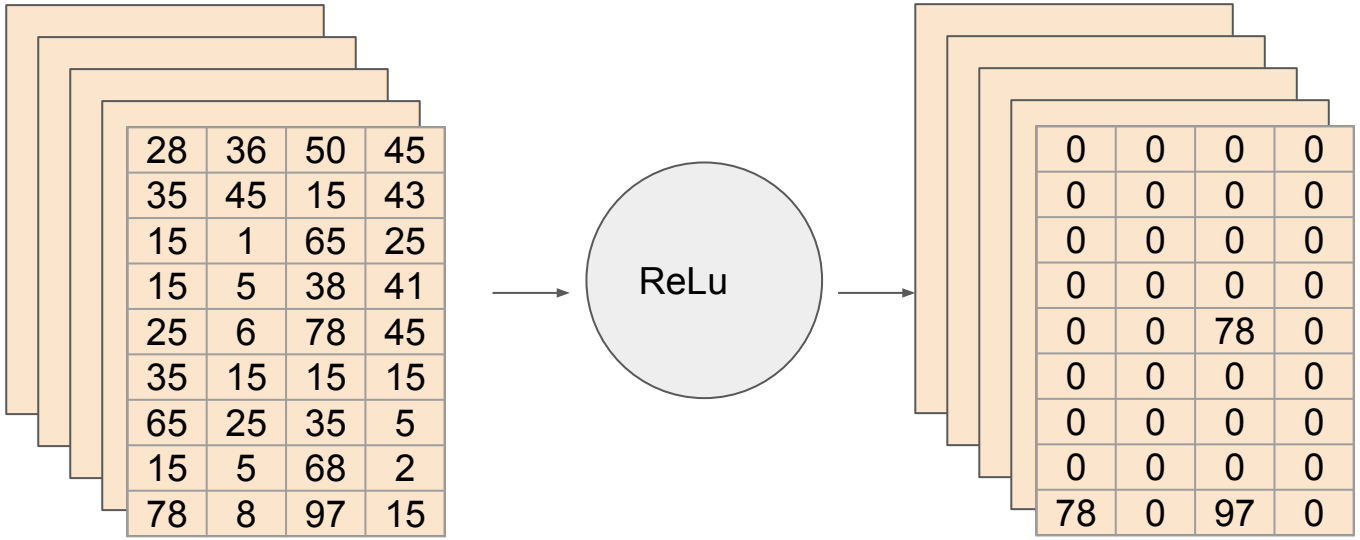
3	16	0	45
7	6	1	4
85	8	5	25
95	5	8	41
12	6	8	45
45	15	15	15
35	25	35	5
85	5	8	2
78	8	97	15

Green + Blue

28	36	50	45
35	45	15	43
15	1	65	25
15	5	38	41
25	6	78	45
35	15	15	15
65	25	35	5
15	5	68	2
78	8	97	15

These activation layers - after passing through the neurons - then become the source images for the next round of the network.

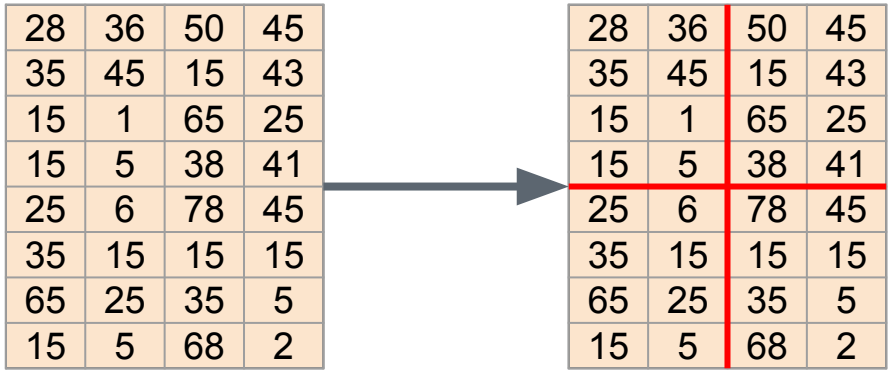
433		


28 36 50 45
35 45 15 43
15 1 65 25
15 5 38 41
25 6 78 45
35 15 15 15
65 25 35 5
15 5 68 2
78 8 97 15
ReLu
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
0 0 78 0
0 0 0 0
0 0 0 0
0 0 0 0
78 0 97 0

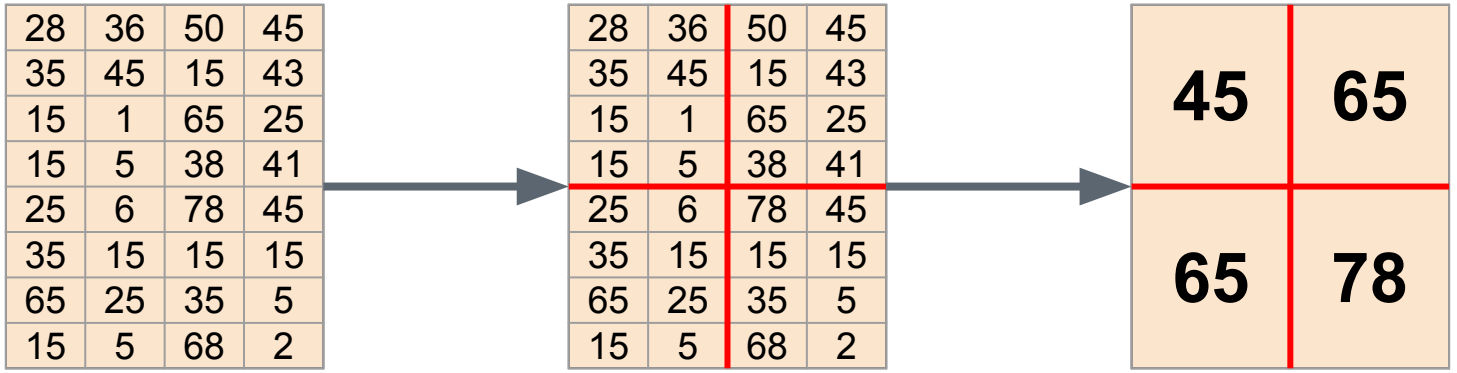
Pooling

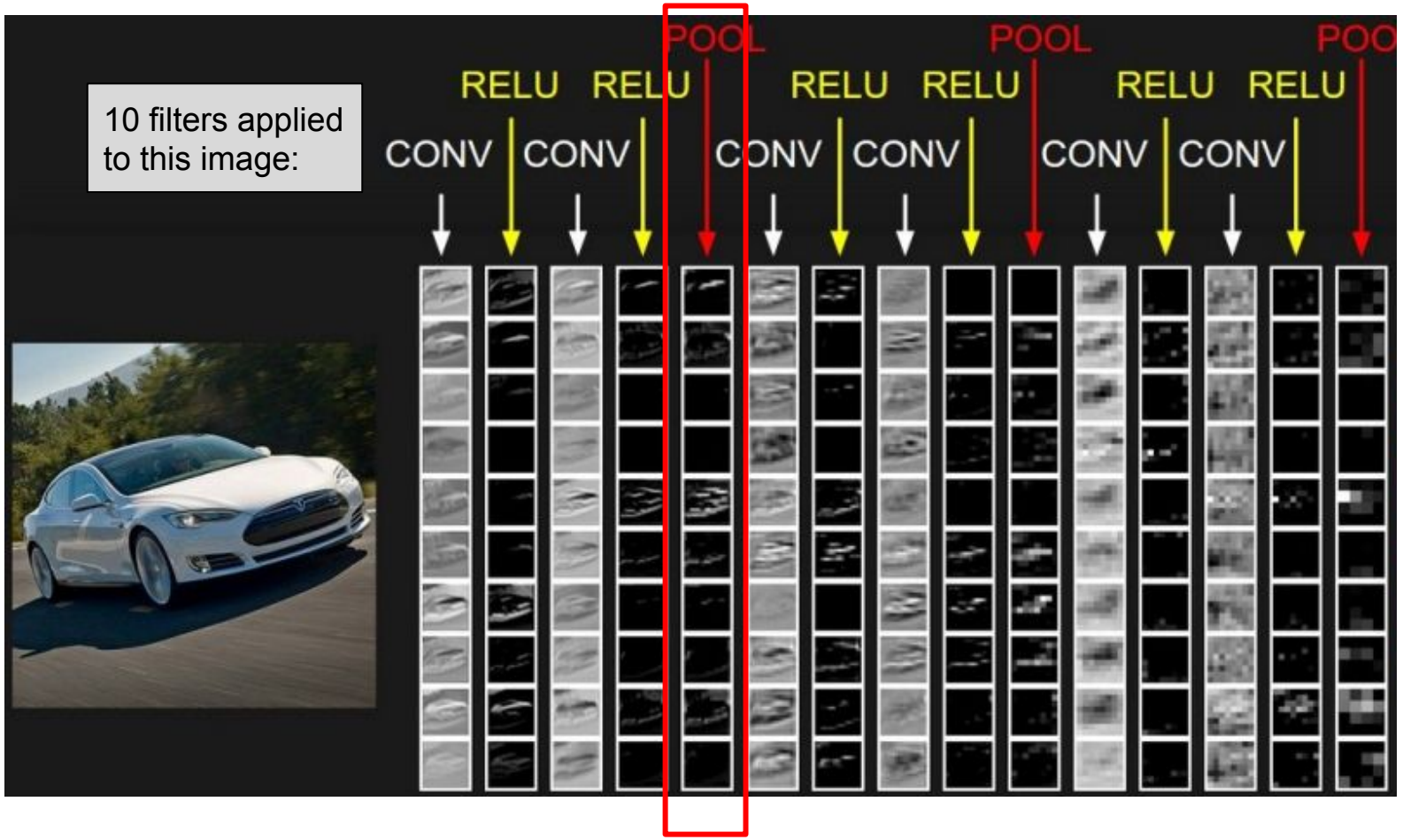
28	36	50	45
35	45	15	43
15	1	65	25
15	5	38	41
25	6	78	45
35	15	15	15
65	25	35	5
15	5	68	2

Pooling

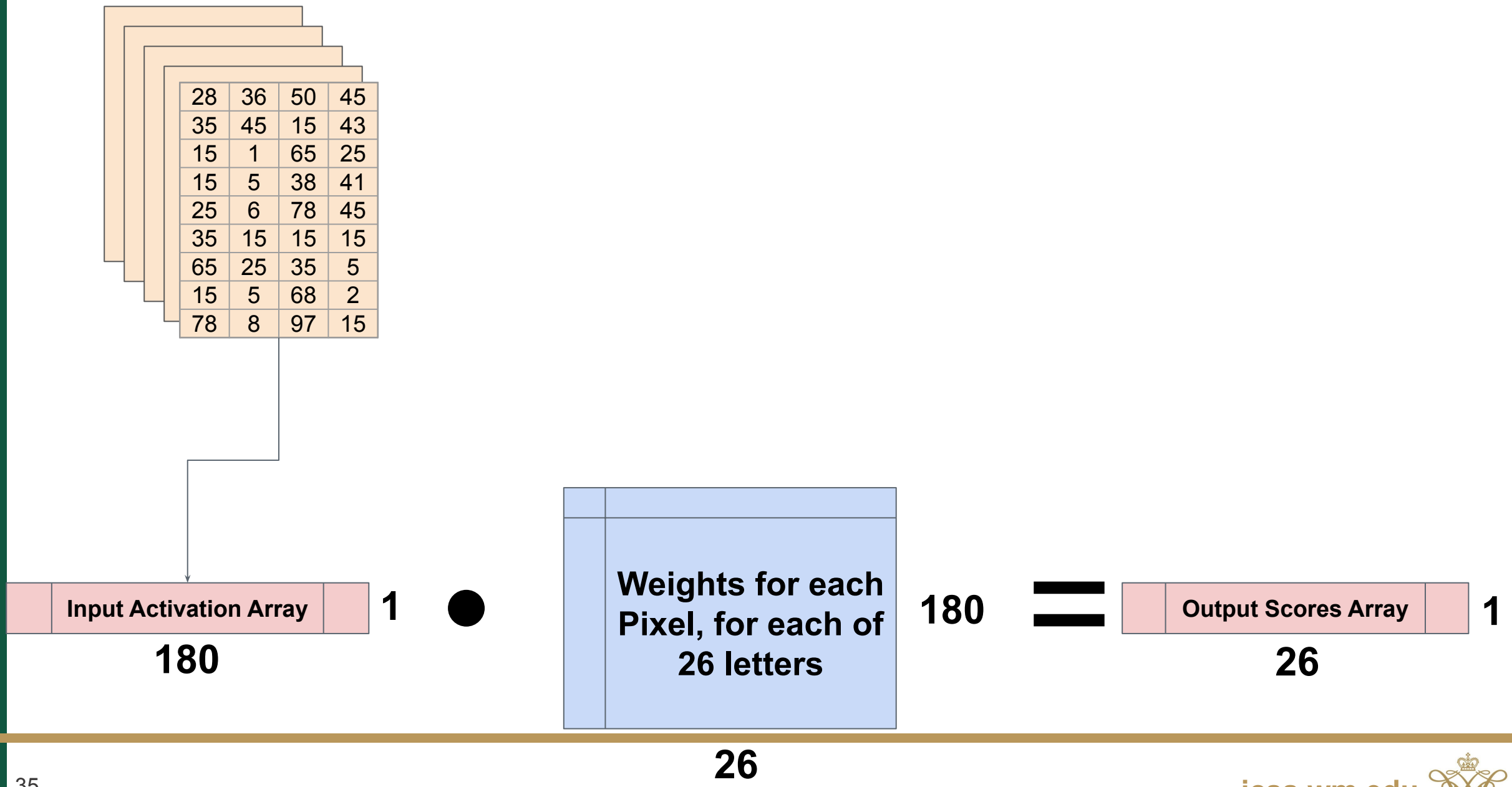

28 36 50 45
35 45 15 43
15 1 65 25
15 5 38 41
25 6 78 45
35 15 15 15
65 25 35 5
15 5 68 2
28 36 50 45
35 45 15 43
15 1 65 25
15 5 38 41
25 6 78 45
35 15 15 15
65 25 35 5
15 5 68 2

Pooling

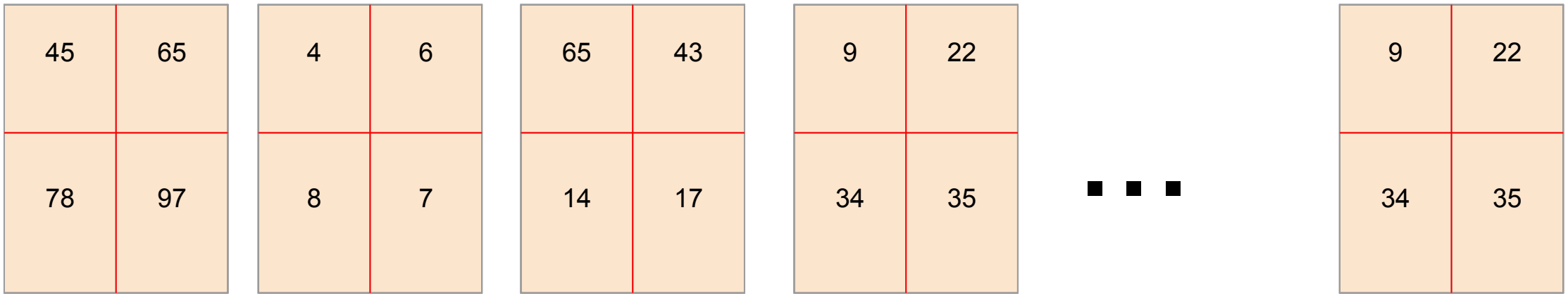

28 36 50 45
35 45 15 43
15 1 65 25
15 5 38 41
25 6 78 45
35 15 15 15
65 25 35 5
15 5 68 2
28 36 50 45
35 45 15 43
15 1 65 25
15 5 38 41
25 6 78 45
35 15 15 15
65 25 35 5
15 5 68 2
45 65
65 78

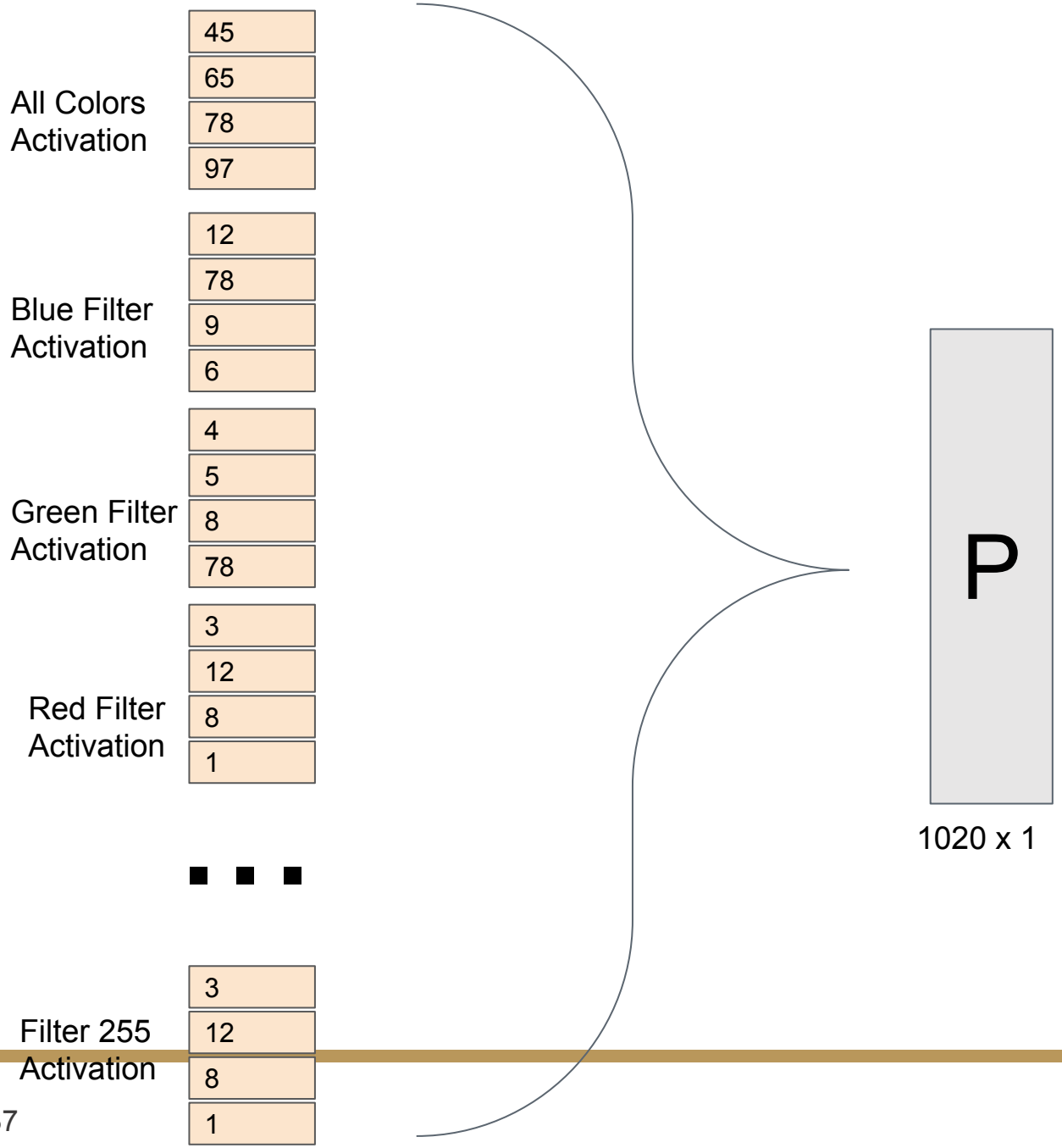


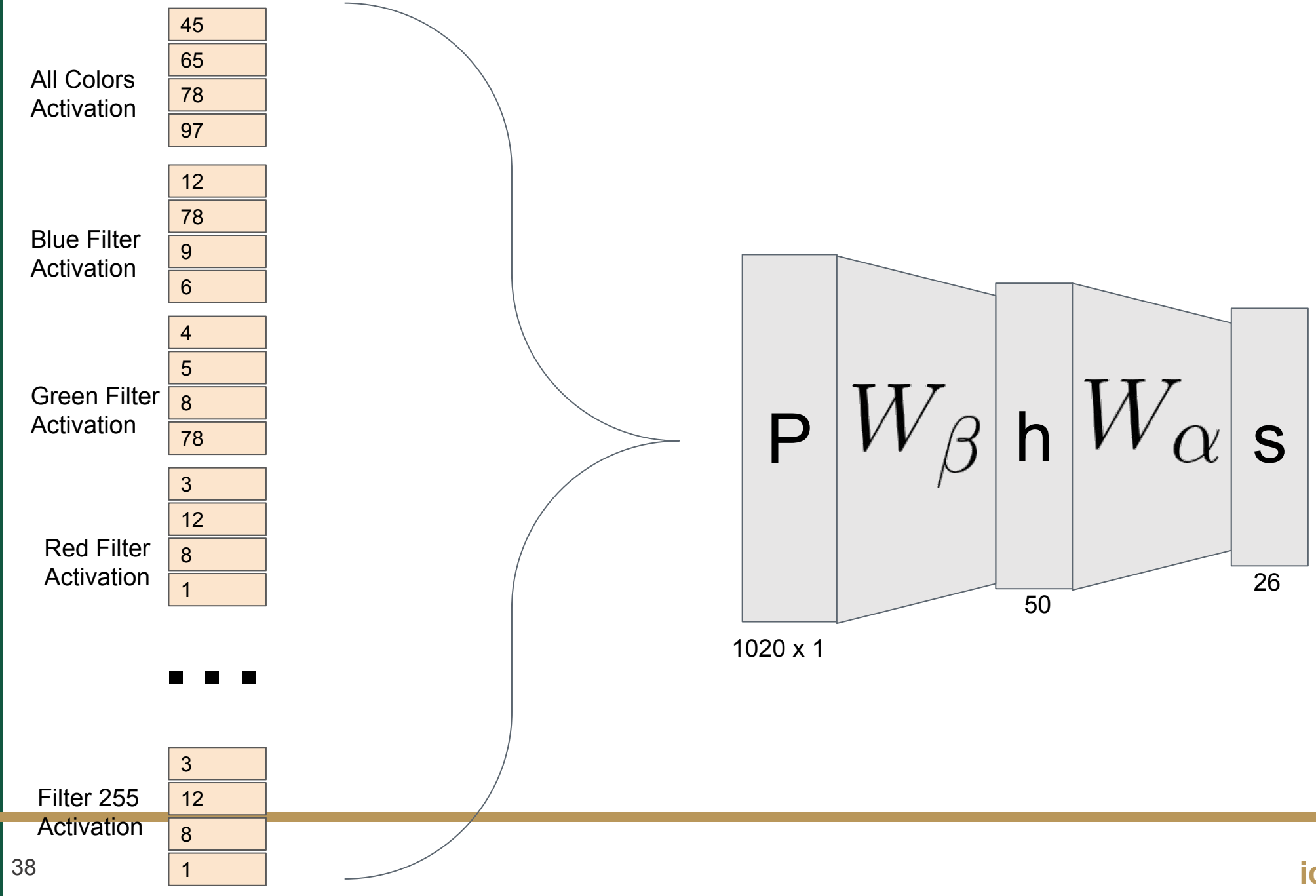
Fe-Fei Li, Andrej Karpathy, Justin Johnson

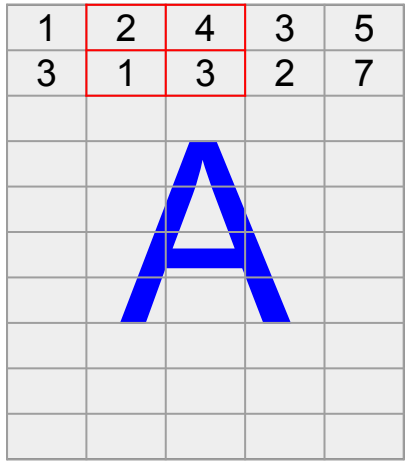

28 36 50 45
35 45 15 43
15 1 65 25
15 5 38 41
25 6 78 45
35 15 15 15
65 25 35 5
15 5 68 2
78 8 97 15
Input Activation Array
1
180
Weights for each Pixel, for each of 26 letters
180
26
Output Scores Array
1
26

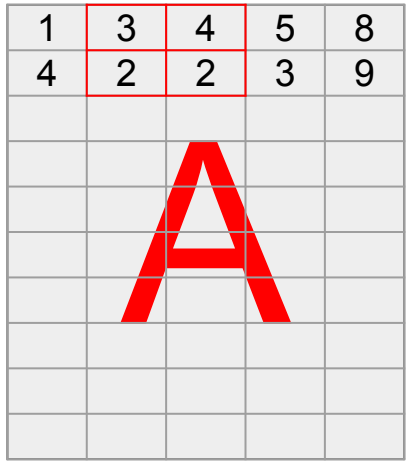
Fully Connected Layer

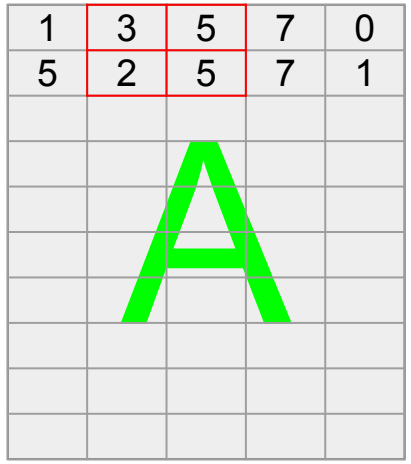



All Colors Activation
45
65
78
97
Blue Filter Activation
12
78
9
6
Green Filter Activation
4
5
8
78
Red Filter Activation
3
12
8
1
Filter 255 Activation
3
12
8
1
P
1020 x 1

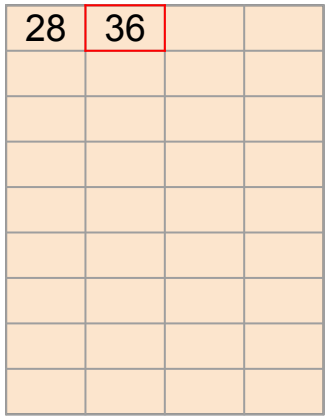

All Colors Activation
45
65
78
97
Blue Filter Activation
12
78
9
6
Green Filter Activation
4
5
8
78
Red Filter Activation
3
12
8
1
Filter 255 Activation
3
12
8
1
P
W_β
h
W_α
s
1020 x 1
50
26

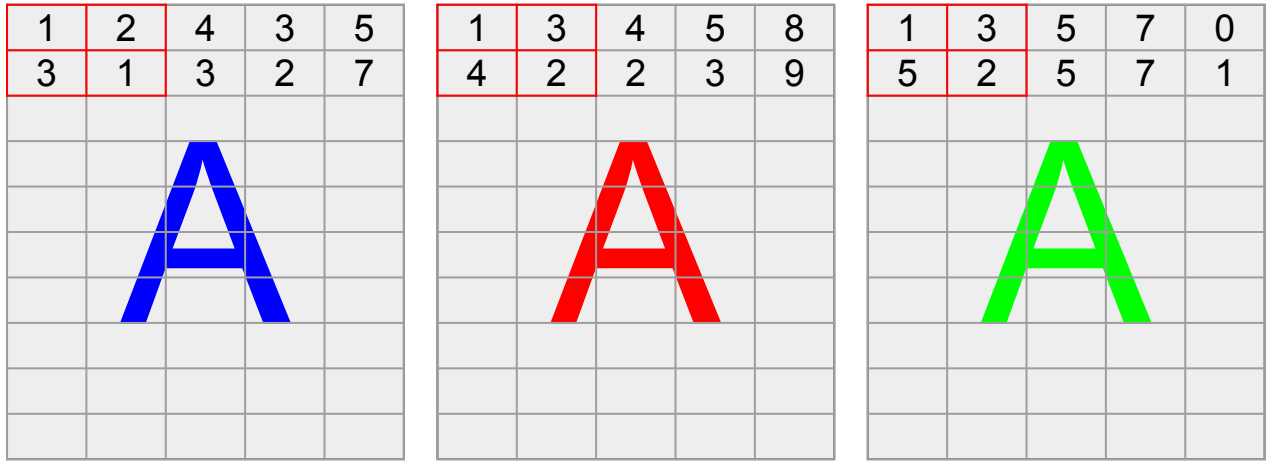




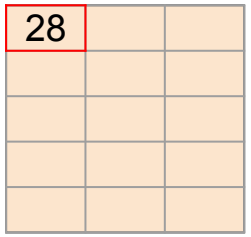


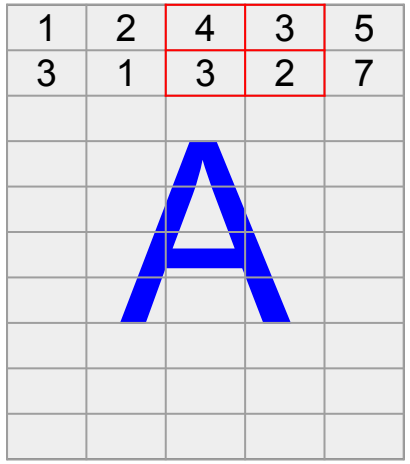
This example is an example where stride = 1 (i.e., we always shift one cell during our convolutions).

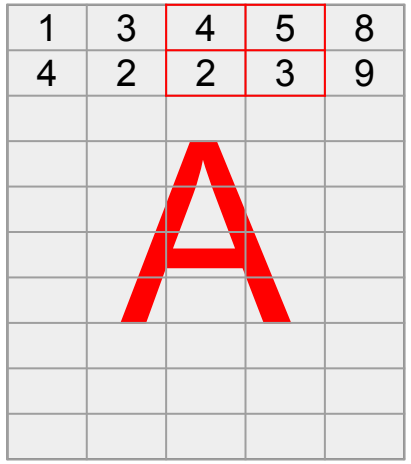


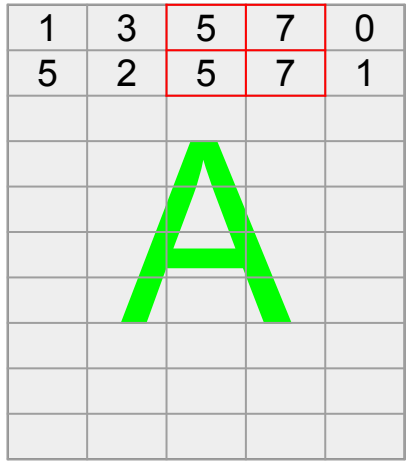


Stride = 2 behaves like this.

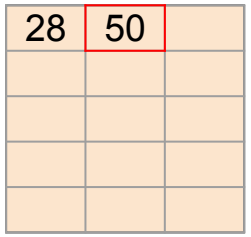


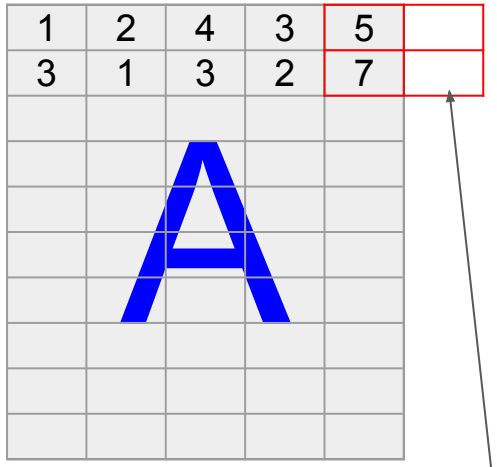


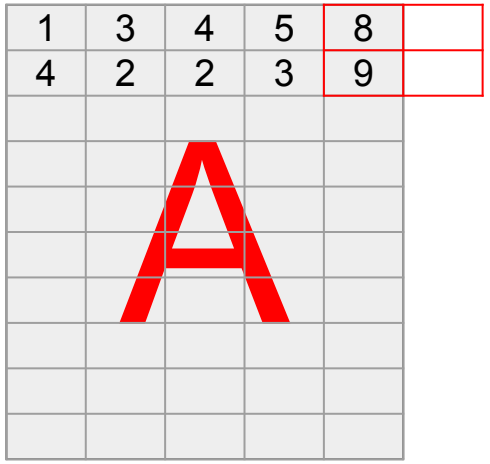


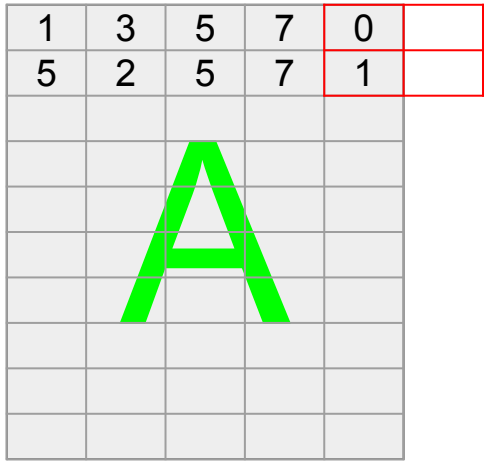


Stride = 2 behaves like this.

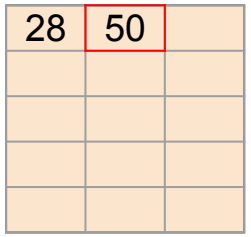








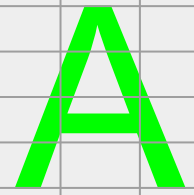
In stride=2, the box 3 runs out of space - there are no values.



1	2	4	3	5	0
3	1	3	2	7	0
					0
					0
					0
					0
					0
					0
					0
					0

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

1	3	5	7	0	0
5	2	5	7	1	0
					0
					0
					0
					0
					0
					0
					0
					0



Zero padding is frequently used to ensure strides fit within the images, given the filter size.

28	50	30

Summary

Convolutional Layers

- Input Image with a Width, Height and Depth (Colors)
- Four Choices (Hyperparameters)
  - Number of Filters
  - Filter Dimensions
  - Stride
  - Zero Padding
- Generally strung together interspersed with computational (i.e., reLu) and pooling layers.