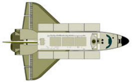
CNN

Mitesh M. Khapra

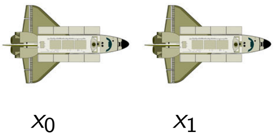
2nd March 2017



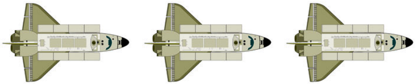
x_0

- Suppose we are tracking the position of a spaceship using a laser sensor at discrete time intervals

2/19

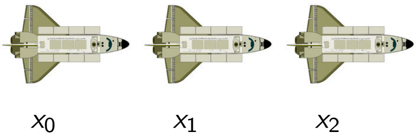


- Suppose we are tracking the position of a spaceship using a laser sensor at discrete time intervals

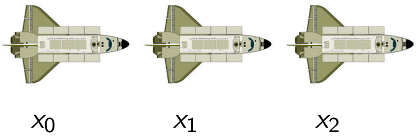


x_0 x_1 x_2

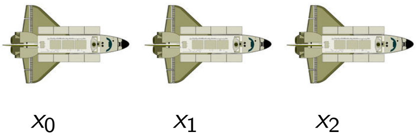
- Suppose we are tracking the position of a spaceship using a laser sensor at discrete time intervals



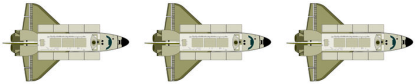
- Suppose we are tracking the position of a spaceship using a laser sensor at discrete time intervals
- Now suppose our sensor is noisy



- Suppose we are tracking the position of a spaceship using a laser sensor at discrete time intervals
- Now suppose our sensor is noisy
- To obtain a less noisy estimate we would like to average several measurements



- Suppose we are tracking the position of a spaceship using a laser sensor at discrete time intervals
- Now suppose our sensor is noisy
- To obtain a less noisy estimate we would like to average several measurements
- More recent measurements are more important so we would like to take a weighted average

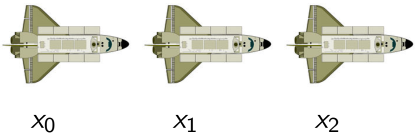


x_0 x_1 x_2

$$s_t = \sum_{a=0}^{\infty} x_{t-a} w_{-a} =$$

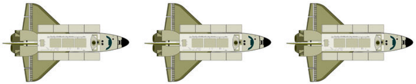
- Suppose we are tracking the position of a spaceship using a laser sensor at discrete time intervals
- Now suppose our sensor is noisy
- To obtain a less noisy estimate we would like to average several measurements
- More recent measurements are more important so we would like to take a weighted average

 2/19



$$s_t = \sum_{a=0}^{\infty} x_{t-a} w_{-a} = (x * w)_t$$

- Suppose we are tracking the position of a spaceship using a laser sensor at discrete time intervals
- Now suppose our sensor is noisy
- To obtain a less noisy estimate we would like to average several measurements
- More recent measurements are more important so we would like to take a weighted average



x_0 x_1 x_2

$$s_t = \sum_{a=0}^{\infty} x_{t-a} w_{-a} = (x * w)_t$$

input convolution filter

- Suppose we are tracking the position of a spaceship using a laser sensor at discrete time intervals
- Now suppose our sensor is noisy
- To obtain a less noisy estimate we would like to average several measurements
- More recent measurements are more important so we would like to take a weighted average

$$s_t = \sum_{a=0}^{6} x_{t-a} w_{-a}$$

- In practice, we would only sum over a small window

$$s_t = \sum_{a=0}^{6} x_{t-a} w_{-a}$$

- In practice, we would only sum over a small window
- This weight array is known as the filter

$$s_t = \sum_{a=0}^{6} x_{t-a} w_{-a}$$

- In practice, we would only sum over a small window
- This weight array is known as the filter
- Here the input (and the kernel) is one dimensional

	w_{-6}	w_{-5}	w_{-4}	w_{-3}	w_{-2}	w_{-1}	w_0					
W	0.01	0.01	0.02	0.02	0.04	0.4	0.5					
X	1.00	1.10	1.20	1.40	1.70	1.80	1.90	2.10	2.20	2.40	2.50	2.70
S							1.80					

$$s_6 = x_6 w_0 + x_5 w_{-1} + x_4 w_{-2} + x_3 w_{-3} + x_4 w_{-4} + x_5 w_{-5} + x_6 w_{-6}$$

$$s_t = \sum_{a=0}^{6} x_{t-a} w_{-a}$$

	w_{-6}	w_{-5}	w_{-4}	w_{-3}	w_{-2}	w_{-1}	w_0
W	0.01	0.01	0.02	0.02	0.04	0.4	0.5

X	1.00	1.10	1.20	1.40	1.70	1.80	1.90	2.10	2.20	2.40	2.50	2.70
S							1.80	1.96				

$$s_6 = x_6 w_0 + x_5 w_{-1} + x_4 w_{-2} + x_3 w_{-3} + x_4 w_{-4} + x_5 w_{-5} + x_6 w_{-6}$$

- In practice, we would only sum over a small window
- This weight array is known as the filter
- Here the input (and the kernel) is one dimensional
- We just slide the filter over the window and compute the value of s_t based on a window arount x_t

$$s_t = \sum_{a=0}^{6} x_{t-a} w_{-a}$$

	w_{-6}	w_{-5}	w_{-4}	w_{-3}	w_{-2}	w_{-1}	w_0					
W			0.01	0.01	0.02	0.02	0.04	0.4	0.5			
X	1.00	1.10	1.20	1.40	1.70	1.80	1.90	2.10	2.20	2.40	2.50	2.70
S							1.80	1.96	2.11			

$$s_6 = x_6 w_0 + x_5 w_{-1} + x_4 w_{-2} + x_3 w_{-3} + x_4 w_{-4} + x_5 w_{-5} + x_6 w_{-6}$$

- In practice, we would only sum over a small window
- This weight array is known as the filter
- Here the input (and the kernel) is one dimensional
- We just slide the filter over the window and compute the value of s_t based on a window arount x_t

$$s_t = \sum_{a=0}^{6} x_{t-a} w_{-a}$$

	w_{-6}	w_{-5}	w_{-4}	w_{-3}	w_{-2}	w_{-1}	w_0
W	0.01	0.01	0.02	0.02	0.04	0.4	0.5

X	1.00	1.10	1.20	1.40	1.70	1.80	1.90	2.10	2.20	2.40	2.50	2.70
S							1.80	1.96	2.11	2.16		

$$s_6 = x_6 w_0 + x_5 w_{-1} + x_4 w_{-2} + x_3 w_{-3} + x_4 w_{-4} + x_5 w_{-5} + x_6 w_{-6}$$

- In practice, we would only sum over a small window
- This weight array is known as the filter
- Here the input (and the kernel) is one dimensional
- We just slide the filter over the window and compute the value of s_t based on a window arount x_t

$$s_t = \sum_{a=0}^{6} x_{t-a} w_{-a}$$

- In practice, we would only sum over a small window
- This weight array is known as the filter
- Here the input (and the kernel) is one dimensional
- We just slide the filter over the window and compute the value of s_t based on a window arount x_t

					w_{-6}	w_{-5}	w_{-4}	w_{-3}	w_{-2}	w_{-1}	w_0	
W					0.01	0.01	0.02	0.02	0.04	0.4	0.5	
X	1.00	1.10	1.20	1.40	1.70	1.80	1.90	2.10	2.20	2.40	2.50	2.70
S							1.80	1.96	2.11	2.16	2.28	

$$s_6 = x_6 w_0 + x_5 w_{-1} + x_4 w_{-2} + x_3 w_{-3} + x_4 w_{-4} + x_5 w_{-5} + x_6 w_{-6}$$

$$s_t = \sum_{a=0}^{6} x_{t-a} w_{-a}$$

- In practice, we would only sum over a small window
- This weight array is known as the filter
- Here the input (and the kernel) is one dimensional
- We just slide the filter over the window and compute the value of s_t based on a window arount x_t

	w_{-6}	w_{-5}	w_{-4}	w_{-3}	w_{-2}	w_{-1}	w_0
W	0.01	0.01	0.02	0.02	1	0.4	0.5

X	1.00	1.10	1.20	1.40	1.70	1.80	1.90	2.10	2.20	2.40	2.50	2.70

S	1.80	1.96	2.11	2.16	2.28	2.42

$$s_6 = x_6 w_0 + x_5 w_{-1} + x_4 w_{-2} + x_3 w_{-3} + x_4 w_{-4} + x_5 w_{-5} + x_6 w_{-6}$$

$$s_t = \sum_{a=0}^{6} x_{t-a} w_{-a}$$

	w_{-6}	w_{-5}	w_{-4}	w_{-3}	w_{-2}	w_{-1}	w_0
W	0.01	0.01	0.02	0.02	1	0.4	0.5

X	1.00	1.10	1.20	1.40	1.70	1.80	1.90	2.10	2.20	2.40	2.50	2.70
S							1.80	1.96	2.11	2.16	2.28	2.42

$$s_6 = x_6 w_0 + x_5 w_{-1} + x_4 w_{-2} + x_3 w_{-3} + x_4 w_{-4} + x_5 w_{-5} + x_6 w_{-6}$$

- In practice, we would only sum over a small window
- This weight array is known as the filter
- Here the input (and the kernel) is one dimensional
- We just slide the filter over the window and compute the value of s_t based on a window arount x_t
- Can we use a Convolutional operation on a 2d input also?

- We can think of images as 2d inputs

- We can think of images as 2d inputs
- We would now like to use a 2d filter (mxn)

$$S_{ij} = (I * K)_{ij} = \sum_{a=0}^{m-1} \sum_{b=0}^{n-1} I_{i-a,j-b} K_{a,b}$$

- We can think of images as 2d inputs
- We would now like to use a 2d filter (mxn)
- First let us see what the 2d formula looks like

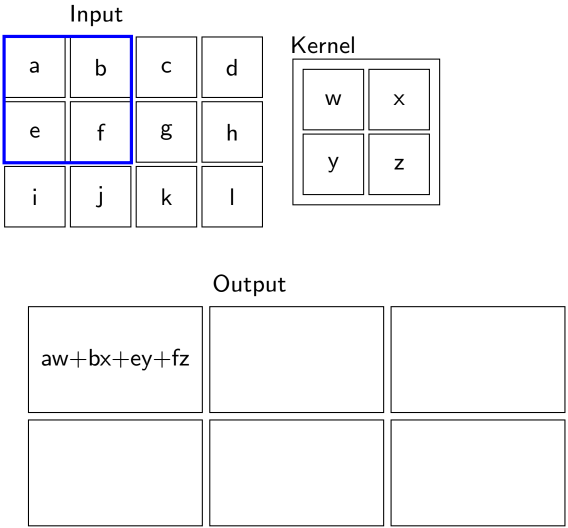
$$S_{ij} = (I * K)_{ij} = \sum_{a=0}^{m-1} \sum_{b=0}^{n-1} I_{i-a,j-b} K_{a,b}$$

- We can think of images as 2d inputs
- We would now like to use a 2d filter (mxn)
- First let us see what the 2d formula looks like
- This formula looks at all the preceding neighbours $(i - a, j - b)$

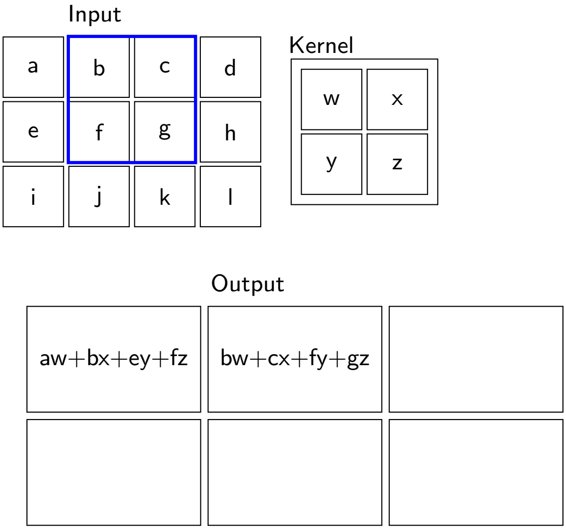
$$S_{ij} = (I * K)_{ij} = \sum_{a=0}^{m-1} \sum_{b=0}^{n-1} I_{i+a,j+b} K_{a,b}$$

- We can think of images as 2d inputs
- We would now like to use a 2d filter (mxn)
- First let us see what the 2d formula looks like
- This formula looks at all the preceding neighbours $(i - a, j - b)$
- In practice, we use the following formula which looks at the succeeding neighbours

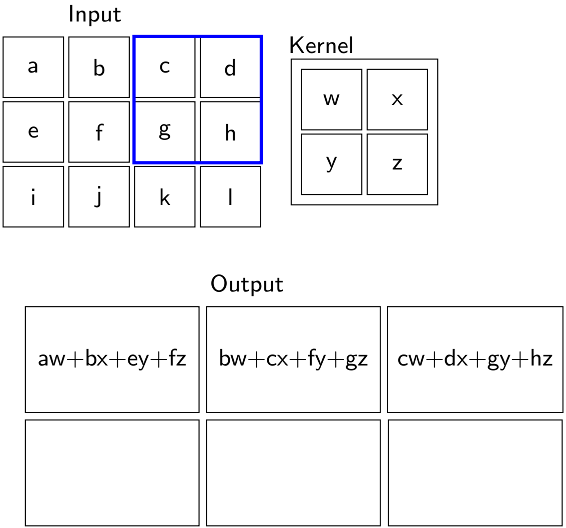
- Let us apply this idea to a toy example and see the results



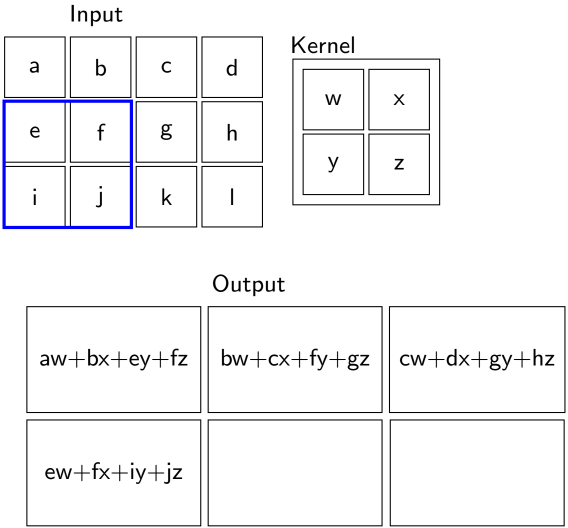
- Let us apply this idea to a toy example and see the results



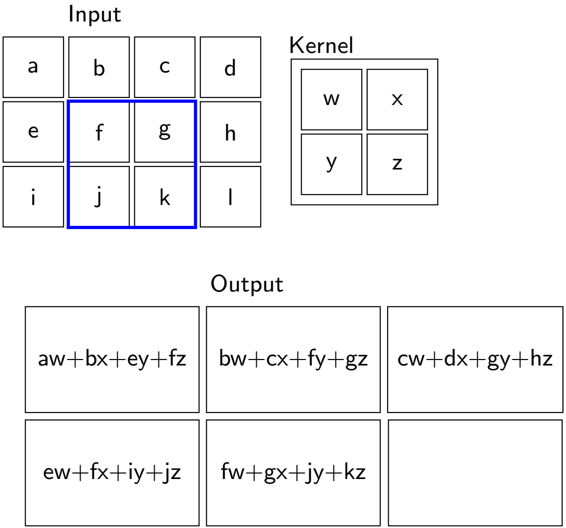
- Let us apply this idea to a toy example and see the results



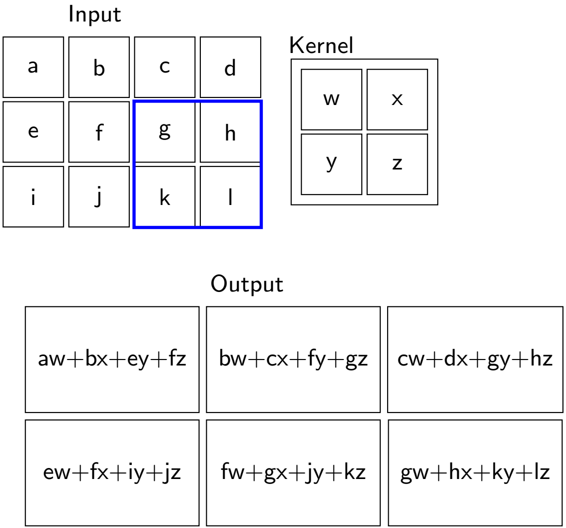
- Let us apply this idea to a toy example and see the results



- Let us apply this idea to a toy example and see the results



- Let us apply this idea to a toy example and see the results



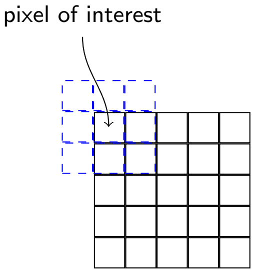
- Let us apply this idea to a toy example and see the results

- For the rest of the discussion we will use the following formula for convolution

$$S_{ij} = (I * K)_{ij} = \sum_{a=\lfloor -\frac{m}{2} \rfloor}^{\lfloor \frac{m}{2} \rfloor} \sum_{b=\lfloor -\frac{n}{2} \rfloor}^{\lfloor \frac{n}{2} \rfloor} I_{i-a,j-b} K_{\frac{m}{2}+a, \frac{n}{2}+b}$$

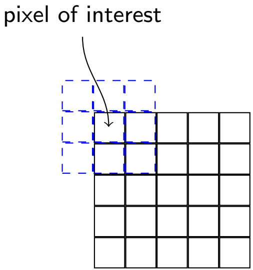
- For the rest of the discussion we will use the following formula for convolution

$$S_{ij} = (I * K)_{ij} = \sum_{a=\lfloor -\frac{m}{2} \rfloor}^{\lfloor \frac{m}{2} \rfloor} \sum_{b=\lfloor -\frac{n}{2} \rfloor}^{\lfloor \frac{n}{2} \rfloor} I_{i-a,j-b} K_{\frac{m}{2}+a, \frac{n}{2}+b}$$



- For the rest of the discussion we will use the following formula for convolution
- In other words we will assume that the kernel is centered on the pixel of interest

$$S_{ij} = (I * K)_{ij} = \sum_{a=\lfloor -\frac{m}{2} \rfloor}^{\lfloor \frac{m}{2} \rfloor} \sum_{b=\lfloor -\frac{n}{2} \rfloor}^{\lfloor \frac{n}{2} \rfloor} I_{i-a,j-b} K_{\frac{m}{2}+a, \frac{n}{2}+b}$$



- For the rest of the discussion we will use the following formula for convolution
- In other words we will assume that the kernel is centered on the pixel of interest
- So we will be looking at both preceeding and succeeding neighbors

Let us see some examples of 2d convolutions applied to images

7/19

$$* \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} =$$

$$* \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} =$$

blurs the image

$$* \begin{matrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{matrix} =$$

$$* \begin{matrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{matrix} =$$

sharpens the image

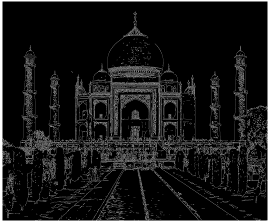
$$* \begin{matrix} 0 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{matrix} =$$

$$* \begin{matrix} 0 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{matrix} =$$

enhances the edges

$$* \begin{matrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{matrix} =$$

$$* \begin{matrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{matrix} =$$



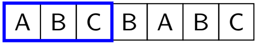
detects the edges

Question

- In 1D convolution, we slide a one dimensional filter over a one dimensional input

Question

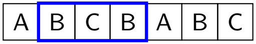
- In 1D convolution, we slide a one dimensional filter over a one dimensional input



 12/19

Question

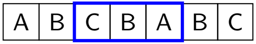
- In 1D convolution, we slide a one dimensional filter over a one dimensional input



12/19

Question

- In 1D convolution, we slide a one dimensional filter over a one dimensional input



12/19

Question

- In 1D convolution, we slide a one dimensional filter over a one dimensional input

A	B	C	B	A	B	C

12/19

Question

- In 1D convolution, we slide a one dimensional filter over a one dimensional input

A	B	C	B	A	B	C

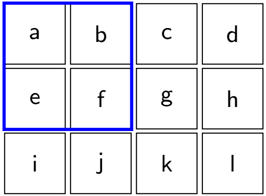
Question

- In 1D convolution, we slide a one dimensional filter over a one dimensional input
- In 2D convolution, we slide a two dimenstional filter over a two dimensional output

12/19

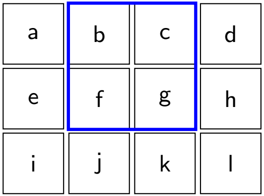
Question

- In 1D convolution, we slide a one dimensional filter over a one dimensional input
- In 2D convolution, we slide a two dimenstional filter over a two dimensional output



Question

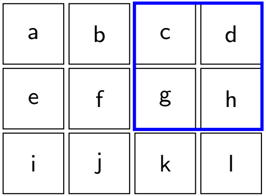
- In 1D convolution, we slide a one dimensional filter over a one dimensional input
- In 2D convolution, we slide a two dimenstional filter over a two dimensional output



12/19

Question

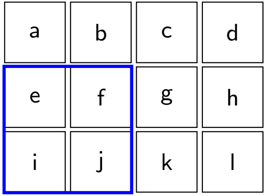
- In 1D convolution, we slide a one dimensional filter over a one dimensional input
- In 2D convolution, we slide a two dimenstional filter over a two dimensional output



12/19

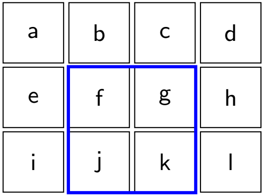
Question

- In 1D convolution, we slide a one dimensional filter over a one dimensional input
- In 2D convolution, we slide a two dimenstional filter over a two dimensional output



Question

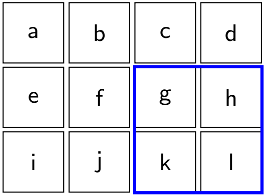
- In 1D convolution, we slide a one dimensional filter over a one dimensional input
- In 2D convolution, we slide a two dimenstional filter over a two dimensional output



12/19

Question

- In 1D convolution, we slide a one dimensional filter over a one dimensional input
- In 2D convolution, we slide a two dimenstional filter over a two dimensional output

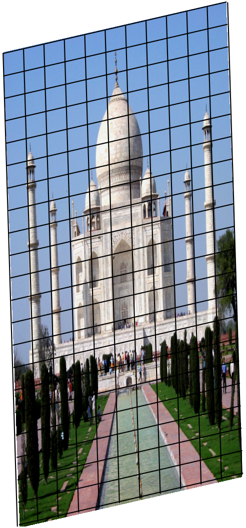


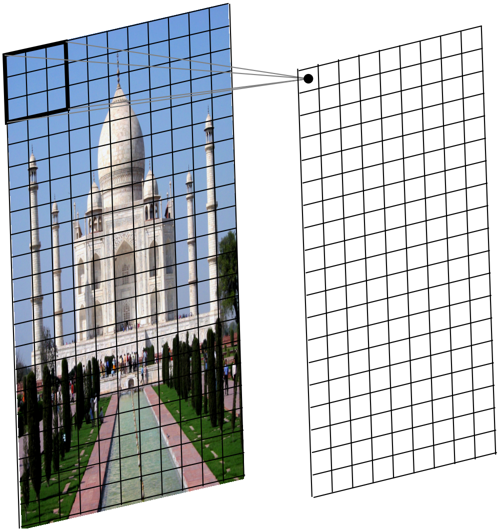
 12/19

Question

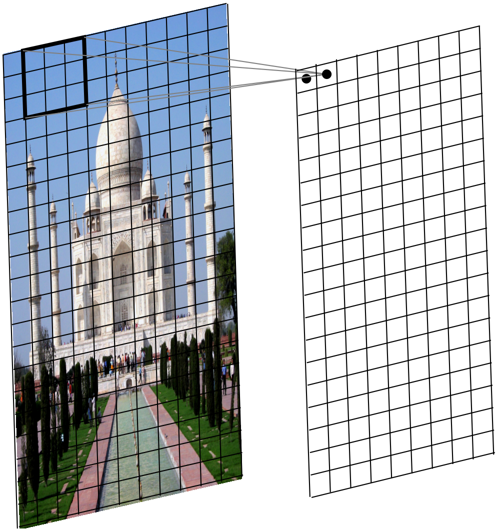
- In 1D convolution, we slide a one dimensional filter over a one dimensional input
- In 2D convolution, we slide a two dimenstional filter over a two dimensional output
- What would a 3D convolution look like?

We will now see a working example of 2D convolution.

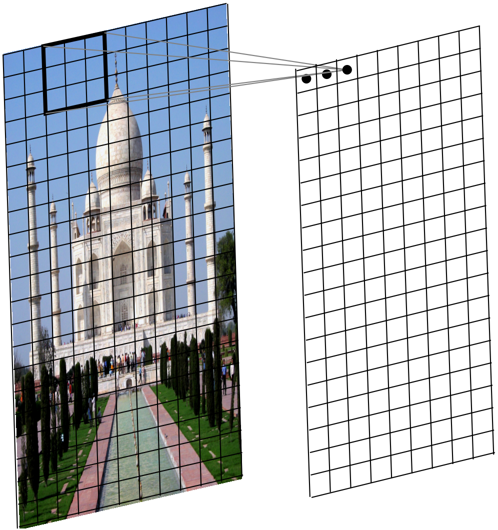




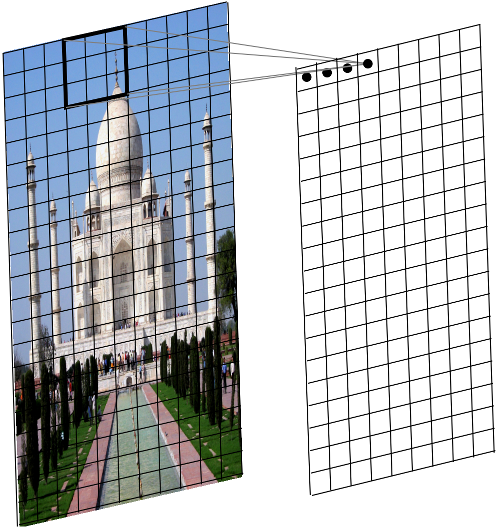
- The resulting output is called a feature map.



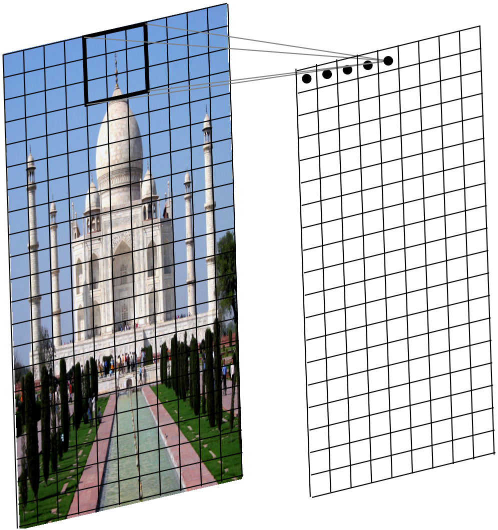
- The resulting output is called a feature map.



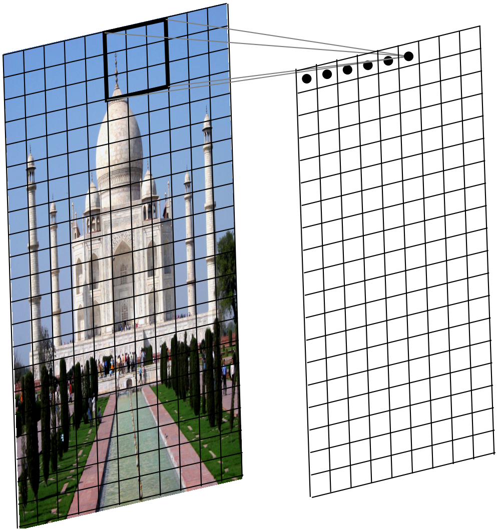
- The resulting output is called a feature map.



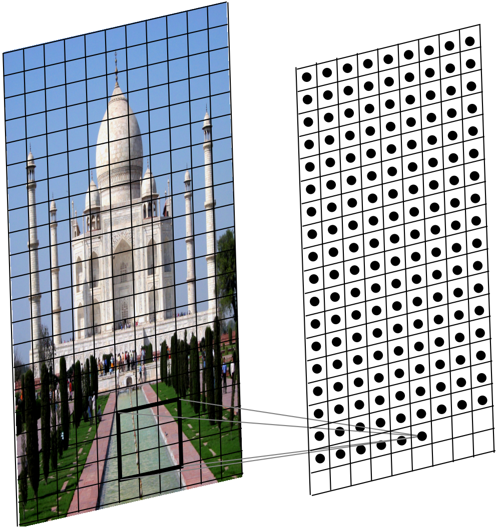
- The resulting output is called a feature map.



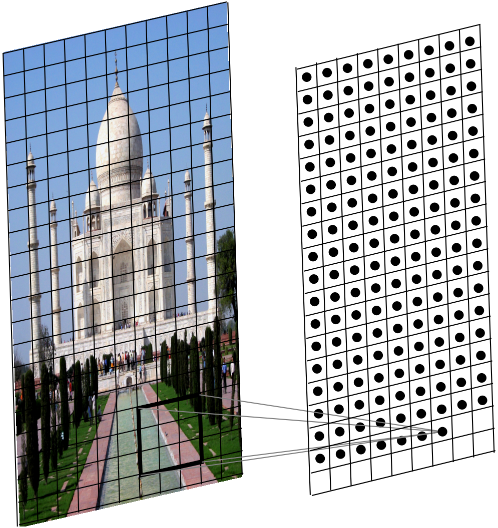
- The resulting output is called a feature map.



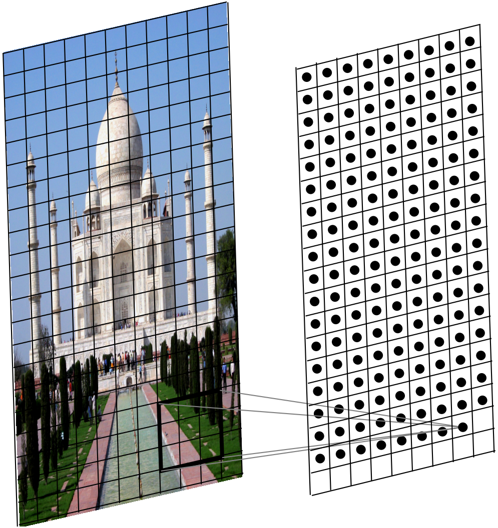
- The resulting output is called a feature map.



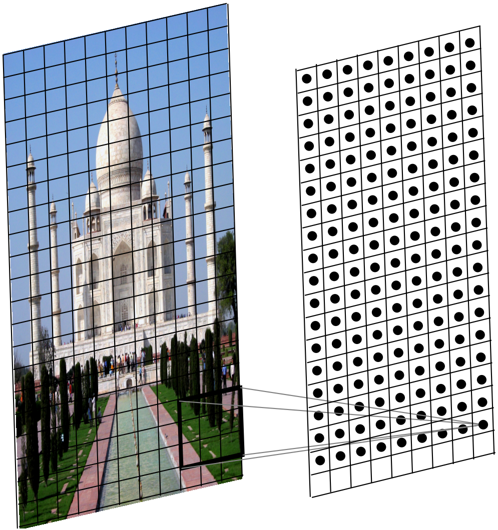
- The resulting output is called a feature map.



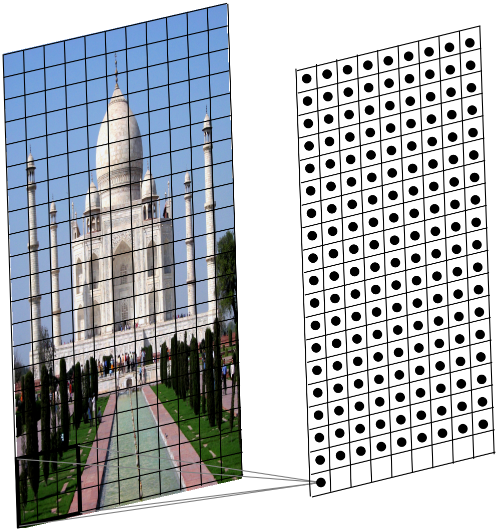
- The resulting output is called a feature map.



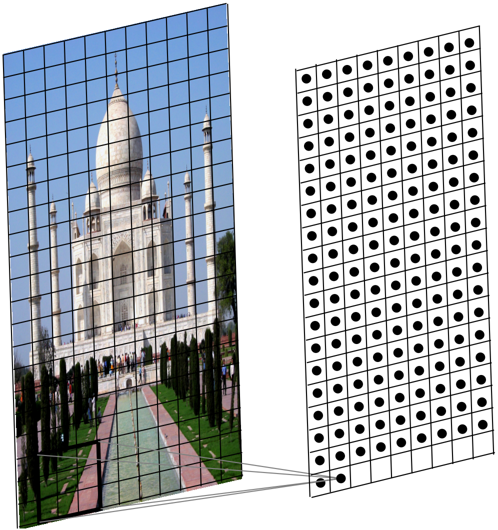
- The resulting output is called a feature map.



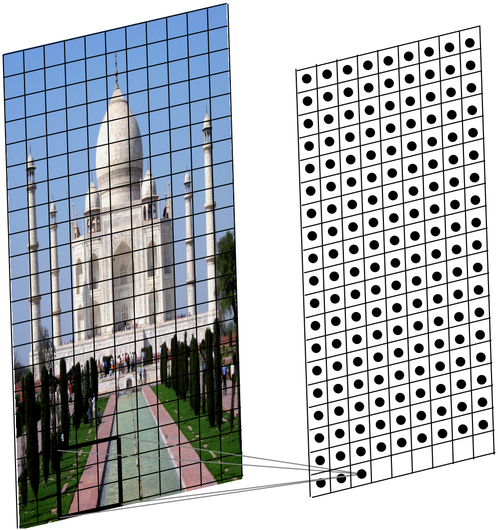
- The resulting output is called a feature map.



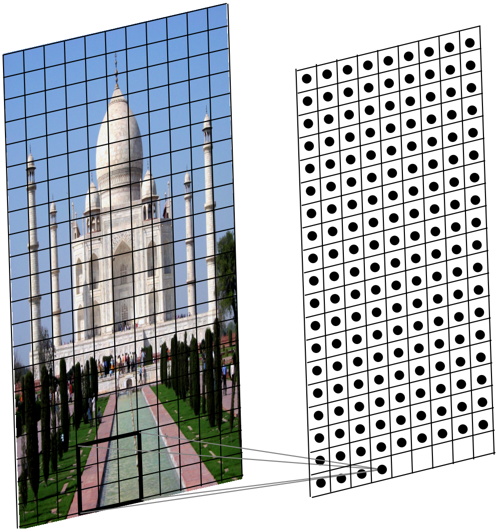
- The resulting output is called a feature map.



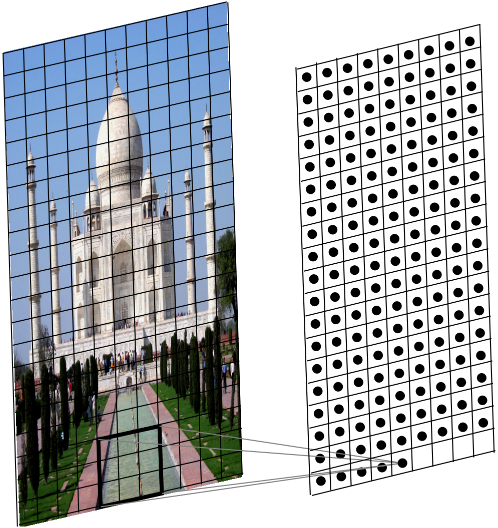
- The resulting output is called a feature map.



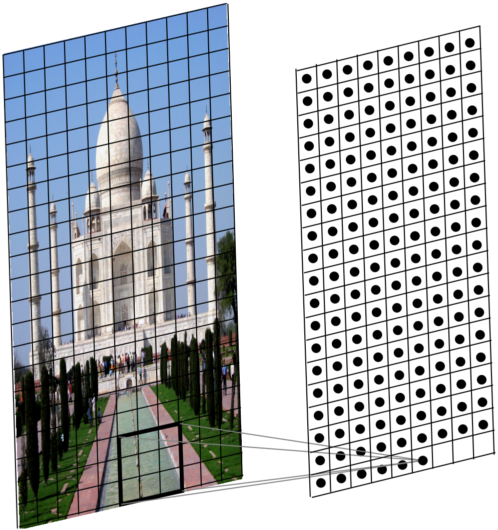
- The resulting output is called a feature map.



- The resulting output is called a feature map.

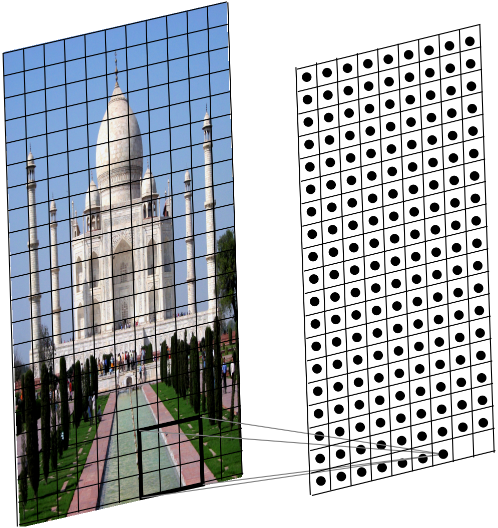


- The resulting output is called a feature map.

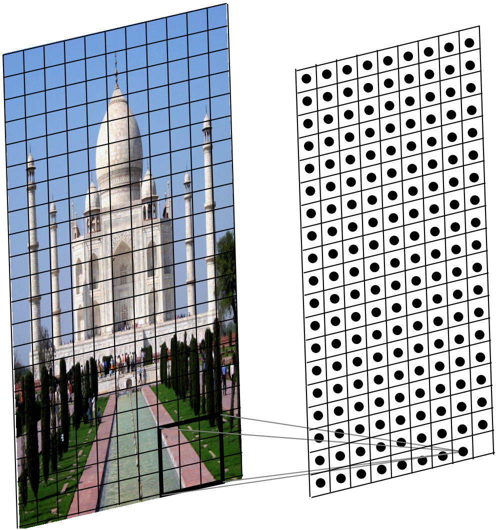


- The resulting output is called a feature map.

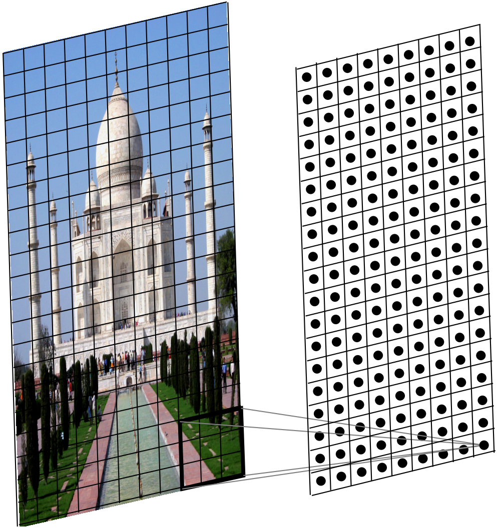
14/19



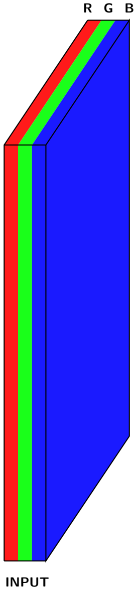
- The resulting output is called a feature map.



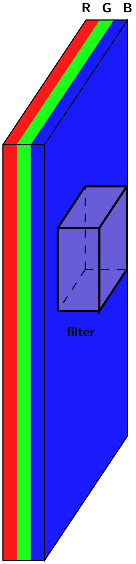
- The resulting output is called a feature map.



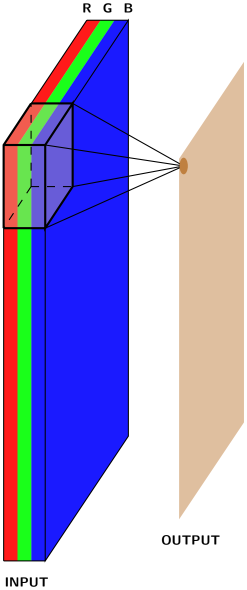
- The resulting output is called a feature map.
- We can use multiple filters to get multiple feature maps.



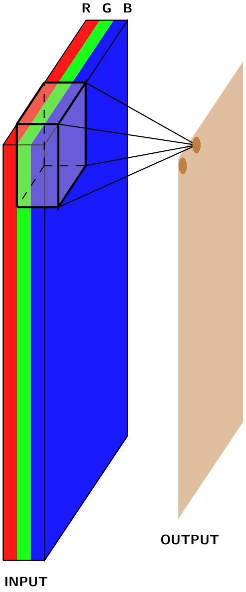
- What would a 3D filter look like?



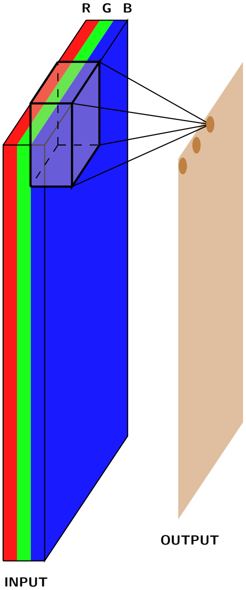
- What would a 3D filter look like?
- Once again you will slide the volume over the image and compute the convolution image.



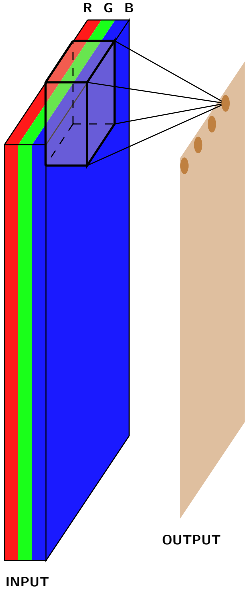
- What would a 3D filter look like?
- Once again you will slide the volume over the image and compute the convolution image.



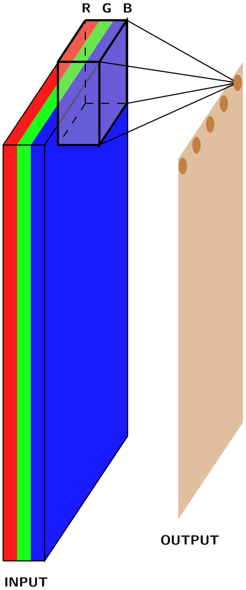
- What would a 3D filter look like?
- Once again you will slide the volume over the image and compute the convolution image.



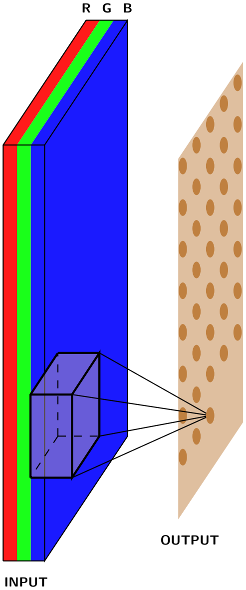
- What would a 3D filter look like?
- Once again you will slide the volume over the image and compute the convolution image.



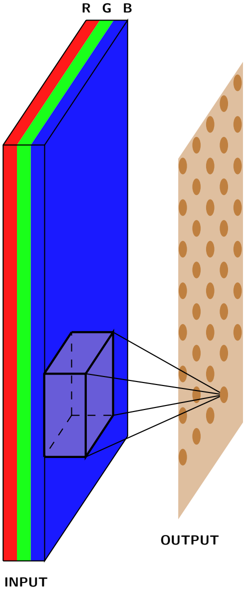
- What would a 3D filter look like?
- Once again you will slide the volume over the image and compute the convolution image.



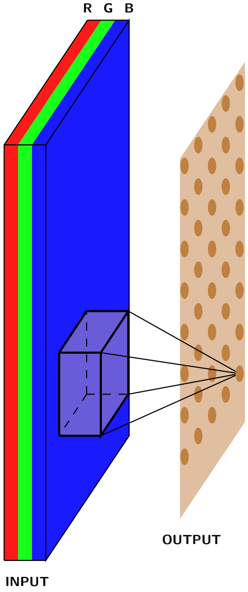
- What would a 3D filter look like?
- Once again you will slide the volume over the image and compute the convolution image.



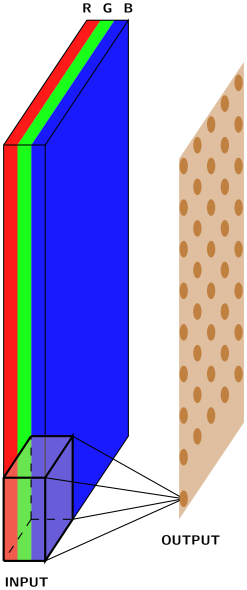
- What would a 3D filter look like?
- Once again you will slide the volume over the image and compute the convolution image.



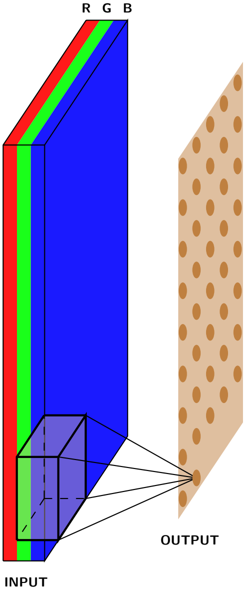
- What would a 3D filter look like?
- Once again you will slide the volume over the image and compute the convolution image.



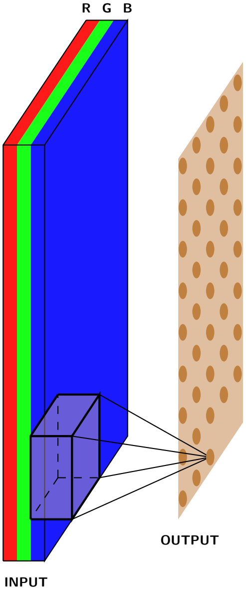
- What would a 3D filter look like?
- Once again you will slide the volume over the image and compute the convolution image.



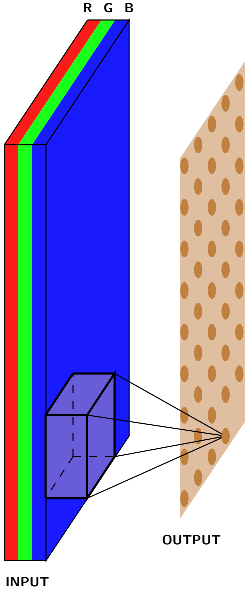
- What would a 3D filter look like?
- Once again you will slide the volume over the image and compute the convolution image.



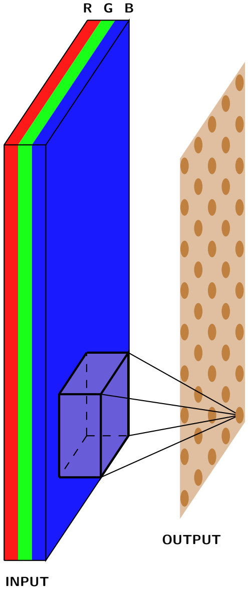
- What would a 3D filter look like?
- Once again you will slide the volume over the image and compute the convolution image.



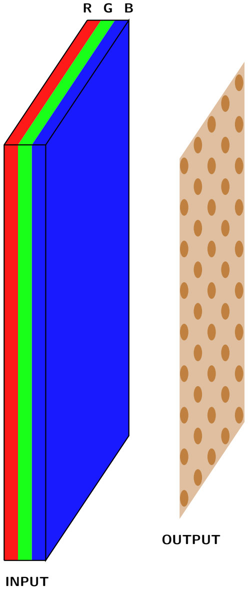
- What would a 3D filter look like?
- Once again you will slide the volume over the image and compute the convolution image.



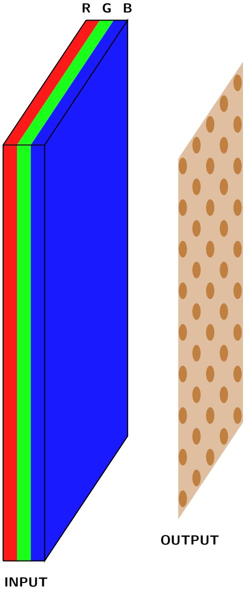
- What would a 3D filter look like?
- Once again you will slide the volume over the image and compute the convolution image.



- What would a 3D filter look like?
- Once again you will slide the volume over the image and compute the convolution image.
- Note that the filter always extends the depth of the image.



- What would a 3D filter look like?
- Once again you will slide the volume over the image and compute the convolution image.
- Note that the filter always extends the depth of the image.
- Also note that 3D filter applied to a 3D input results in a 2D output.



- What would a 3D filter look like?
- Once again you will slide the volume over the image and compute the convolution image.
- Note that the filter always extends the depth of the image.
- Also note that 3D filter applied to a 3D input results in a 2D output.
- Once again we can apply multiple filters to get multiple feature maps.

- So far we have not said anything explicit about the dimensions of the

- So far we have not said anything explicit about the dimensions of the
  1. inputs

- So far we have not said anything explicit about the dimensions of the
  1. inputs
  2. filters

16/19

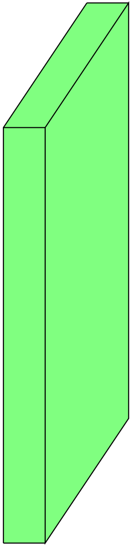
- So far we have not said anything explicit about the dimensions of the
  1. inputs
  2. filters
  3. outputs

- So far we have not said anything explicit about the dimensions of the
  1. inputs
  2. filters
  3. outputs

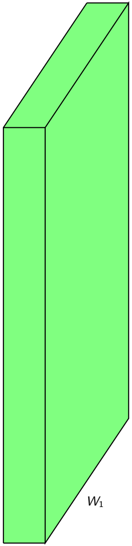
  and relations between them.

- So far we have not said anything explicit about the dimensions of the
  1. inputs
  2. filters
  3. outputs

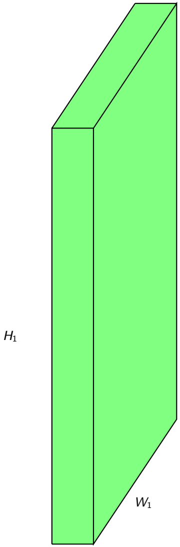
  and relations between them.
- We will see how they are related but before that we will define a few quantities.



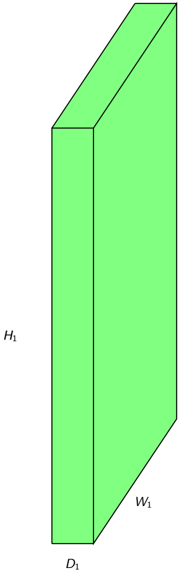
- We first define the following quantities.



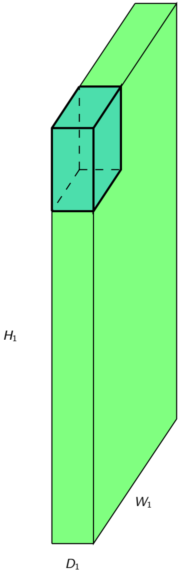
- We first define the following quantities.
- Width (W_1),



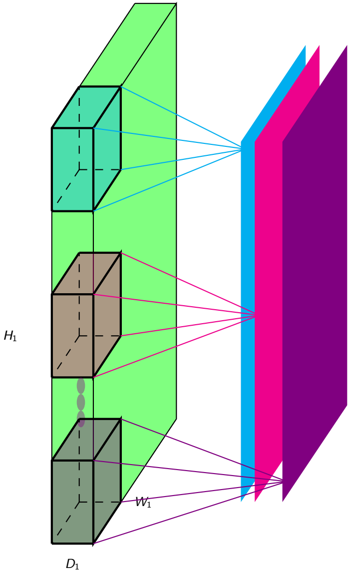
- We first define the following quantities.
- Width (W_1), Height (H_1)



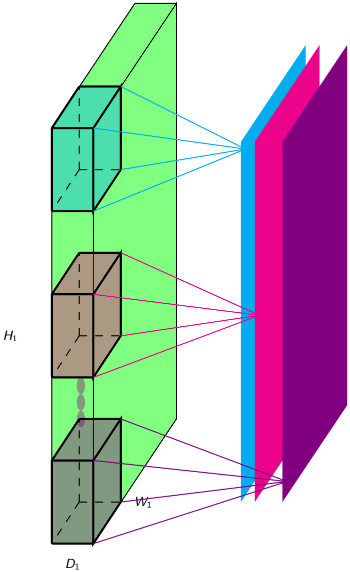
- We first define the following quantities.
- Width (W_1), Height (H_1) and Depth (D_1) of the original input.



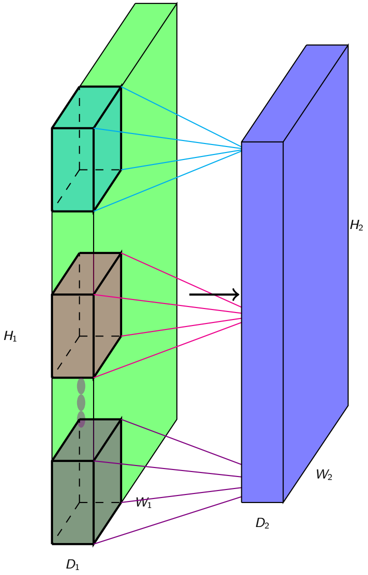
- We first define the following quantities.
- Width (W_1), Height (H_1) and Depth (D_1) of the original input.
- The Stride S.(We will come back to this later)



- We first define the following quantities.
- Width (W_1), Height (H_1) and Depth (D_1) of the original input.
- The Stride S.(We will come back to this later)
- The number of filters K.



- We first define the following quantities.
- Width (W_1), Height (H_1) and Depth (D_1) of the original input.
- The Stride S.(We will come back to this later)
- The number of filters K.
- The spatial extend (F) of each filter (the depth of each filter is same as the depth of each input.

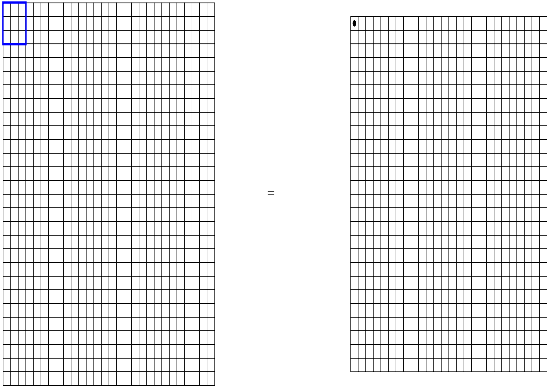


- We first define the following quantities.
- Width (W_1), Height (H_1) and Depth (D_1) of the original input.
- The Stride S.(We will come back to this later)
- The number of filters K.
- The spatial extend (F) of each filter (the depth of each filter is same as the depth of each input.
- The output is $W_2 \times H_2 \times D_2$ (we will soon see a formula for computing W_2, H_2 and D_2.

- For example $W_1 = 28$, $H_1 = 28$, $D_1 = 1$, $K = 1$, $F = 3$, $S = 1$

$$W_2 = \frac{W_1 - F}{S} + 1$$
$$= \frac{28 - 3}{1} + 1 =$$

$$H_2 = \frac{H_1 - F}{S} + 1$$
$$= \frac{28 - 3}{1} + 1 =$$

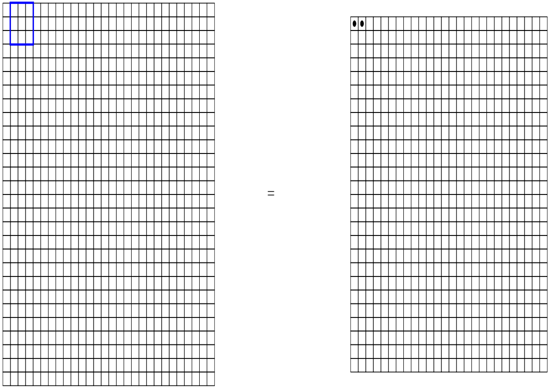


- For example $W_1 = 28$, $H_1 = 28$, $D_1 = 1$, $K = 1$, $F = 3$, $S = 1$

$$W_2 = \frac{W_1 - F}{S} + 1$$
$$= \frac{28 - 3}{1} + 1 =$$

$$H_2 = \frac{H_1 - F}{S} + 1$$
$$= \frac{28 - 3}{1} + 1 =$$

- Notice that we can't put the kernel on the corner as it goes out of the input

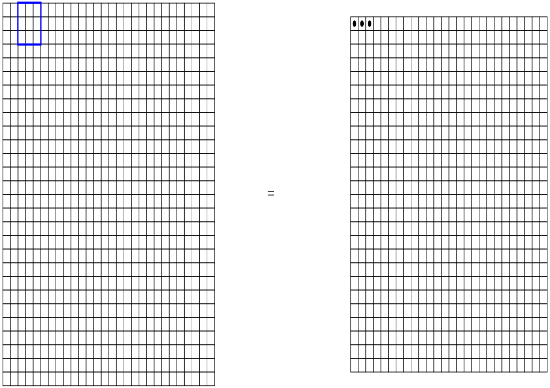


- For example $W_1 = 28$, $H_1 = 28$, $D_1 = 1$, $K = 1$, $F = 3$, $S = 1$

$$W_2 = \frac{W_1 - F}{S} + 1$$
$$= \frac{28 - 3}{1} + 1 =$$

$$H_2 = \frac{H_1 - F}{S} + 1$$
$$= \frac{28 - 3}{1} + 1 =$$

- Notice that we can't put the kernel on the corner as it goes out of the input

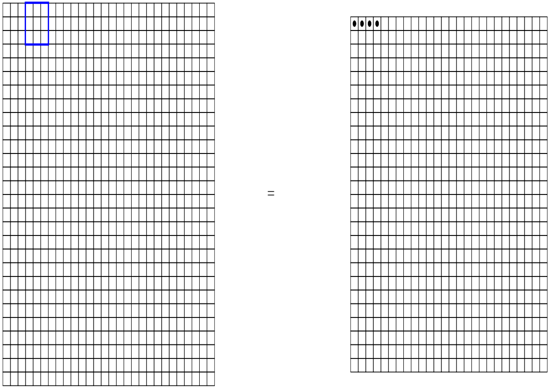


- For example $W_1 = 28$, $H_1 = 28$, $D_1 = 1$, $K = 1$, $F = 3$, $S = 1$

$$W_2 = \frac{W_1 - F}{S} + 1$$
$$= \frac{28 - 3}{1} + 1 =$$

$$H_2 = \frac{H_1 - F}{S} + 1$$
$$= \frac{28 - 3}{1} + 1 =$$

- Notice that we can't put the kernel on the corner as it goes out of the input

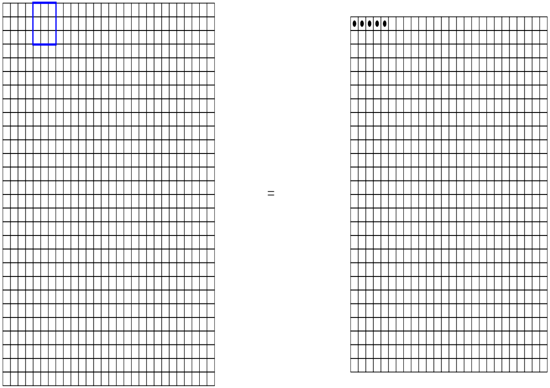


- For example $W_1 = 28,\ H_1 = 28,$ $D_1 = 1,\ K = 1,\ F = 3,\ S = 1$

$$W_2 = \frac{W_1 - F}{S} + 1$$
$$= \frac{28 - 3}{1} + 1 =$$

$$H_2 = \frac{H_1 - F}{S} + 1$$
$$= \frac{28 - 3}{1} + 1 =$$

- Notice that we can't put the kernel on the corner as it goes out of the input

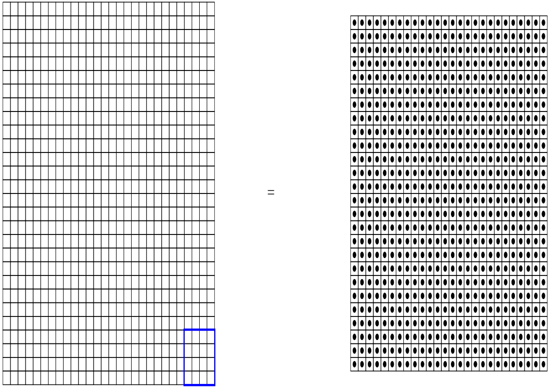


- For example $W_1 = 28,\ H_1 = 28,\ D_1 = 1,\ K = 1,\ F = 3,\ S = 1$

$$W_2 = \frac{W_1 - F}{S} + 1 = \frac{28 - 3}{1} + 1 =$$

$$H_2 = \frac{H_1 - F}{S} + 1 = \frac{28 - 3}{1} + 1 =$$

- Notice that we can't put the kernel on the corner as it goes out of the input

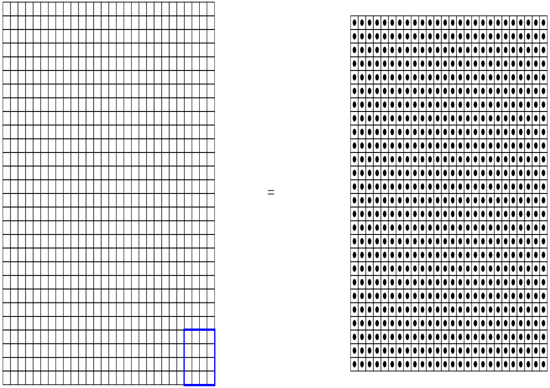


- For example $W_1 = 28,\ H_1 = 28,$ $D_1 = 1,\ K = 1,\ F = 3,\ S = 1$

$$W_2 = \frac{W_1 - F}{S} + 1$$
$$= \frac{28 - 3}{1} + 1 =$$

$$H_2 = \frac{H_1 - F}{S} + 1$$
$$= \frac{28 - 3}{1} + 1 =$$

- Notice that we can't put the kernel on the corner as it goes out of the input



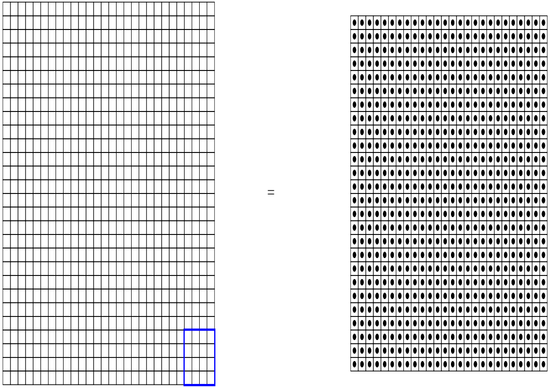
$$W_2 = \frac{W_1 - F}{S} + 1$$
$$H_2 = \frac{H_1 - F}{S} + 1$$

- For example $W_1 = 28$, $H_1 = 28$, $D_1 = 1$, $K = 1$, $F = 3$, $S = 1$

$$W_2 = \frac{W_1 - F}{S} + 1$$
$$= \frac{28 - 3}{1} + 1 =$$

$$H_2 = \frac{H_1 - F}{S} + 1$$
$$= \frac{28 - 3}{1} + 1 =$$

- Notice that we can't put the kernel on the corner as it goes out of the input
- This results in an output which is of smaller dimension than the input



$$W_2 = \frac{W_1 - F}{S} + 1$$

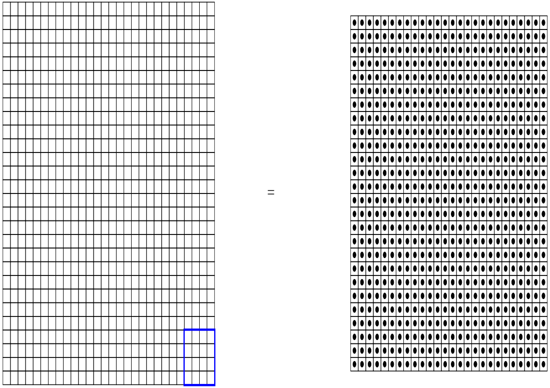
$$H_2 = \frac{H_1 - F}{S} + 1$$

- For example $W_1 = 28$, $H_1 = 28$, $D_1 = 1$, $K = 1$, $F = 3$, $S = 1$

$$W_2 = \frac{W_1 - F}{S} + 1$$
$$= \frac{28 - 3}{1} + 1 =$$

$$H_2 = \frac{H_1 - F}{S} + 1$$
$$= \frac{28 - 3}{1} + 1 =$$

- Notice that we can't put the kernel on the corner as it goes out of the input
- This results in an output which is of smaller dimension than the input
- What if we want the output to be of



$$W_2 = \frac{W_1 - F}{S} + 1$$
$$H_2 = \frac{H_1 - F}{S} + 1$$

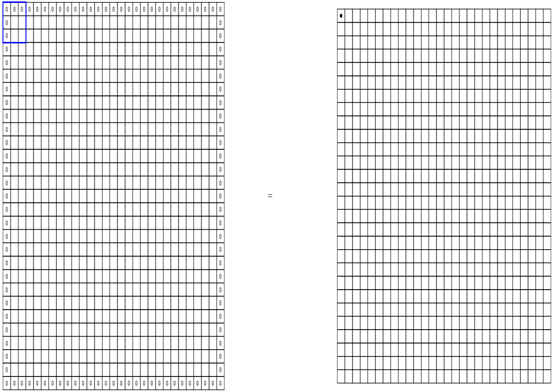
- For example $W_1 = 28$, $H_1 = 28$, $D_1 = 1$, $K = 1$, $F = 3$, $S = 1$

$$W_2 = \frac{W_1 - F}{S} + 1$$
$$= \frac{28 - 3}{1} + 1 =$$

$$H_2 = \frac{H_1 - F}{S} + 1$$
$$= \frac{28 - 3}{1} + 1 =$$

- Notice that we can't put the kernel on the corner as it goes out of the input
- This results in an output which is of smaller dimension than the input
- What if we want the output to be of

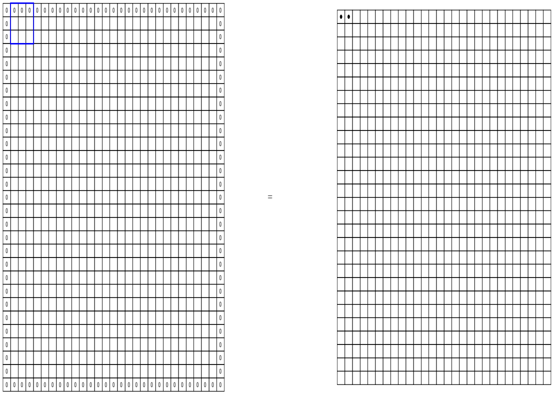
- Pad the inputs with appropriate number of 0 inputs so that you can now apply the kernel at the corners.



- Pad the inputs with appropriate number of 0 inputs so that you can now apply the kernel at the corners.
- For example, $P = 1$

$$W_2 = \frac{W_1 - F + 2P}{S} + 1$$
$$= \frac{W_1 - 3 + 2}{1} + 1 = W_1$$

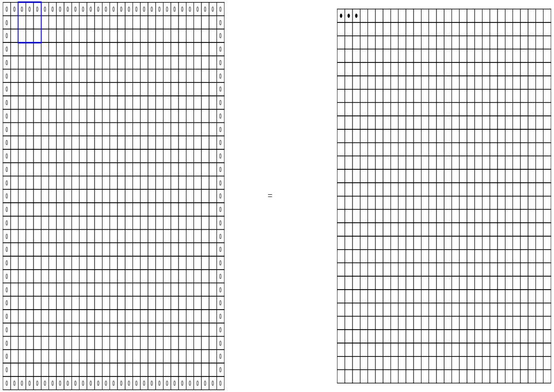
$$W_2 = \frac{H_1 - F + 2P}{S} + 1$$
$$= \frac{H_1 - 3 + 2}{1} + 1 = W_1$$



- Pad the inputs with appropriate number of 0 inputs so that you can now apply the kernel at the corners.
- For example, $P = 1$

$$W_2 = \frac{W_1 - F + 2P}{S} + 1$$
$$= \frac{W_1 - 3 + 2}{1} + 1 = W_1$$

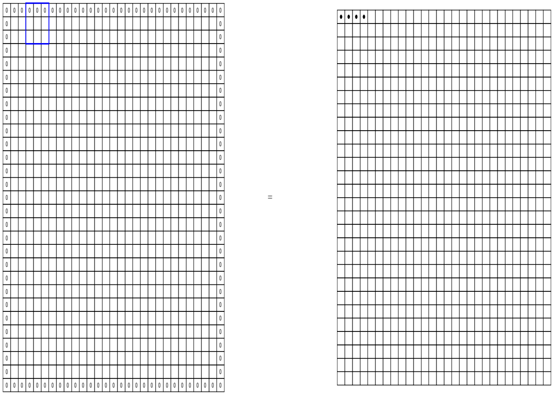
$$W_2 = \frac{H_1 - F + 2P}{S} + 1$$
$$= \frac{H_1 - 3 + 2}{1} + 1 = W_1$$



- Pad the inputs with appropriate number of 0 inputs so that you can now apply the kernel at the corners.
- For example, $P = 1$

$$W_2 = \frac{W_1 - F + 2P}{S} + 1$$
$$= \frac{W_1 - 3 + 2}{1} + 1 = W_1$$

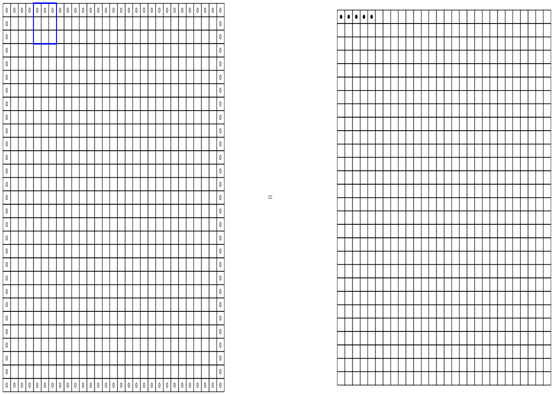
$$W_2 = \frac{H_1 - F + 2P}{S} + 1$$
$$= \frac{H_1 - 3 + 2}{1} + 1 = W_1$$



- Pad the inputs with appropriate number of 0 inputs so that you can now apply the kernel at the corners.
- For example, $P = 1$

$$W_2 = \frac{W_1 - F + 2P}{S} + 1$$
$$= \frac{W_1 - 3 + 2}{1} + 1 = W_1$$

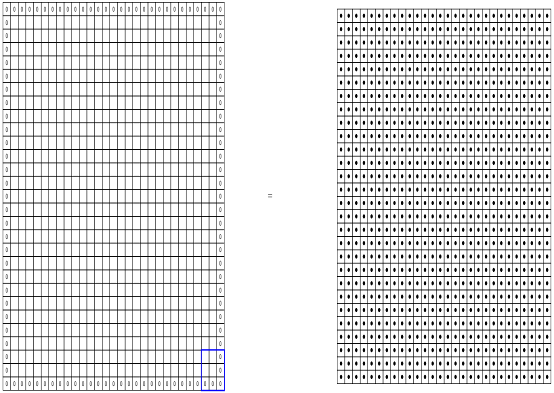
$$W_2 = \frac{H_1 - F + 2P}{S} + 1$$
$$= \frac{H_1 - 3 + 2}{1} + 1 = W_1$$



- Pad the inputs with appropriate number of 0 inputs so that you can now apply the kernel at the corners.
- For example, $P = 1$

$$W_2 = \frac{W_1 - F + 2P}{S} + 1$$
$$= \frac{W_1 - 3 + 2}{1} + 1 = W_1$$

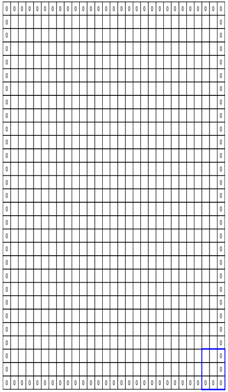
$$W_2 = \frac{H_1 - F + 2P}{S} + 1$$
$$= \frac{H_1 - 3 + 2}{1} + 1 = W_1$$

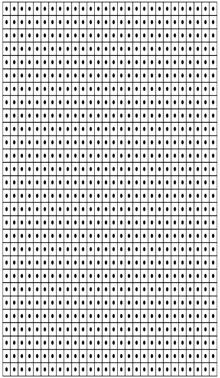


- Pad the inputs with appropriate number of 0 inputs so that you can now apply the kernel at the corners.
- For example, $P = 1$

$$W_2 = \frac{W_1 - F + 2P}{S} + 1$$
$$= \frac{W_1 - 3 + 2}{1} + 1 = W_1$$

$$W_2 = \frac{H_1 - F + 2P}{S} + 1$$
$$= \frac{H_1 - 3 + 2}{1} + 1 = W_1$$





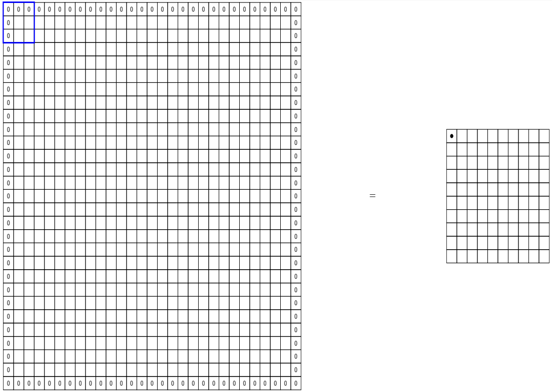
$$W_2 = \frac{W_1 - F + 2P}{S} + 1$$
$$H_2 = \frac{H_1 - F + 2P}{S} + 1$$

- Pad the inputs with appropriate number of 0 inputs so that you can now apply the kernel at the corners.
- For example, $P = 1$

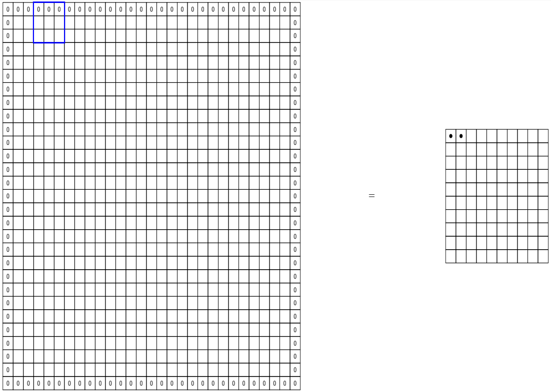
$$W_2 = \frac{W_1 - F + 2P}{S} + 1 = \frac{W_1 - 3 + 2}{1} + 1 = W_1$$

$$W_2 = \frac{H_1 - F + 2P}{S} + 1 = \frac{H_1 - 3 + 2}{1} + 1 = W_1$$

- What does the stride S do?

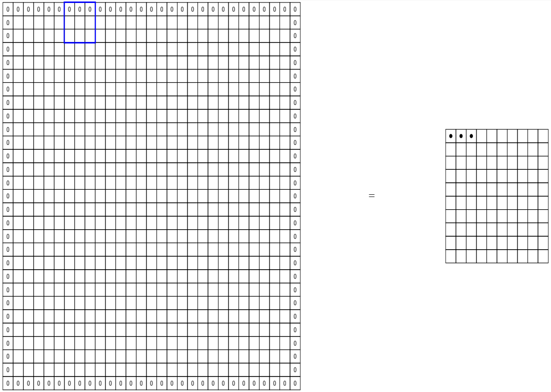


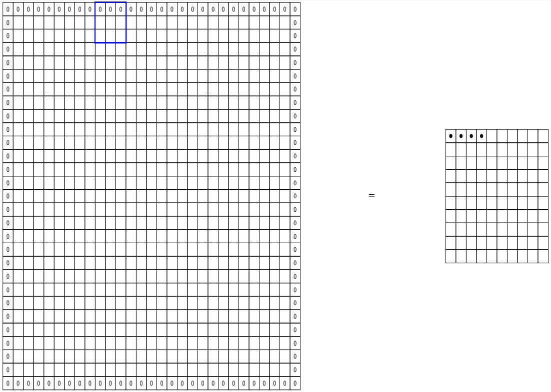
- What does the stride S do?



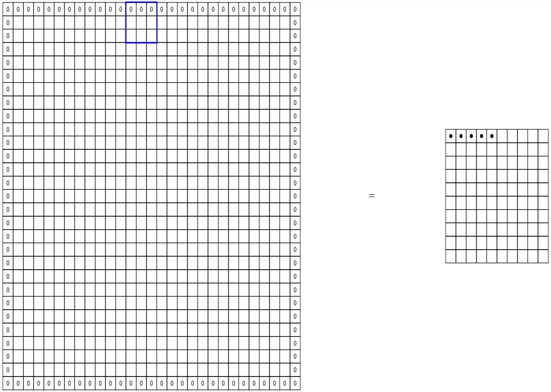
- What does the stride S do?

- What does the stride S do?

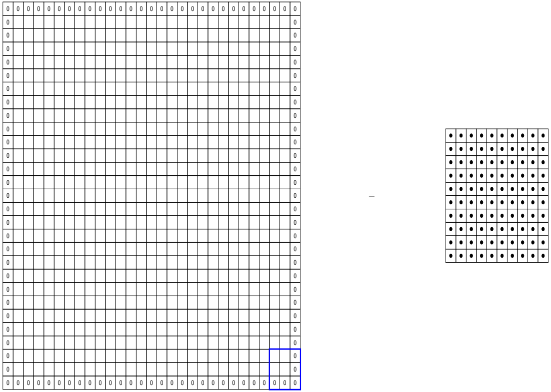




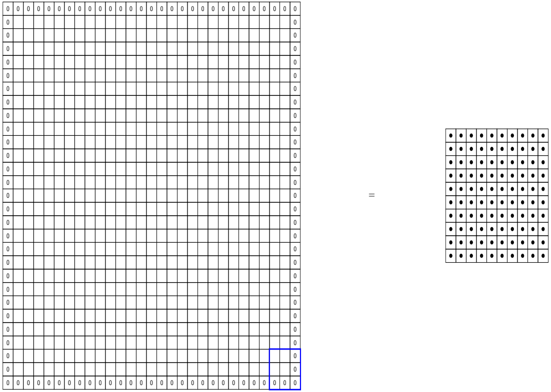
- What does the stride S do?



- What does the stride S do?



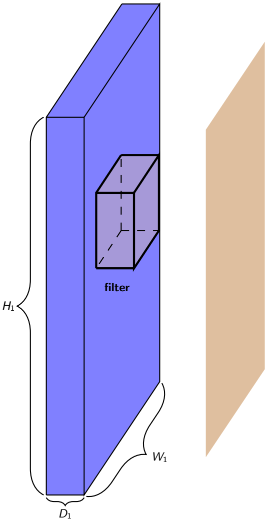
- What does the stride S do?

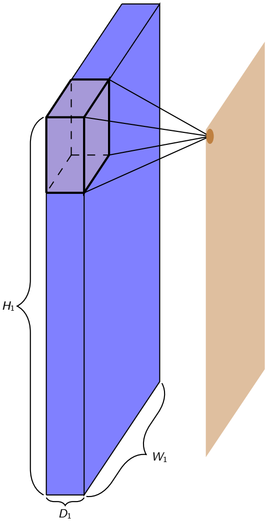


- What does the stride S do?
- It defines the intervals at which the filter is applied(here S=3)

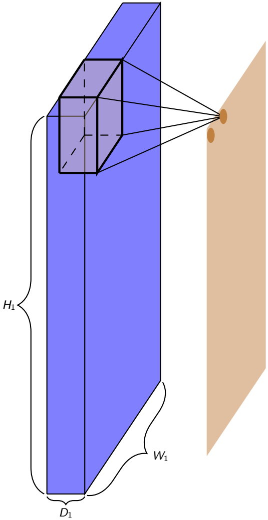
$$W_2 = \frac{W_1 - F + 2P}{S} + 1$$

$$H_2 = \frac{H_1 - F + 2P}{S} + 1$$

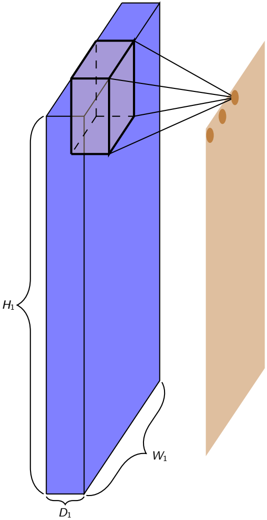

filter
H_1
W_1
D_1



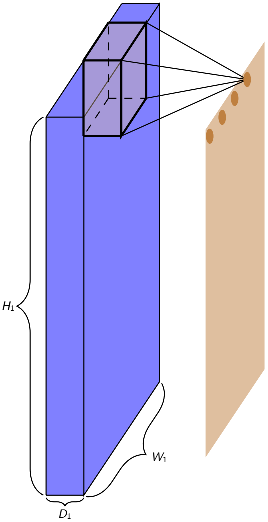
- Finally, coming to the 3d case.



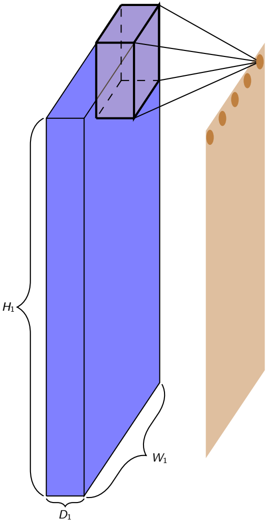
- Finally, coming to the 3d case.
- Each filter gives us one 2d output.



- Finally, coming to the 3d case.
- Each filter gives us one 2d output.
- K filters will give us K such 2D outputs

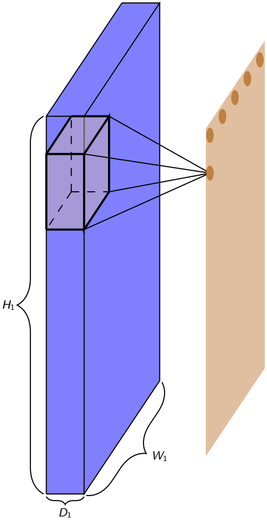


- Finally, coming to the 3d case.
- Each filter gives us one 2d output.
- K filters will give us K such 2D outputs
- We can think of the resulting output as $K \times W_2 \times H_2$ volume



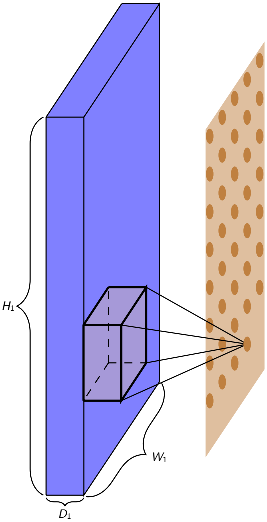
- Finally, coming to the 3d case.
- Each filter gives us one 2d output.
- K filters will give us K such 2D outputs
- We can think of the resulting output as $K \times W_2 \times H_2$ volume
- Thus equal.

- Thus equal.



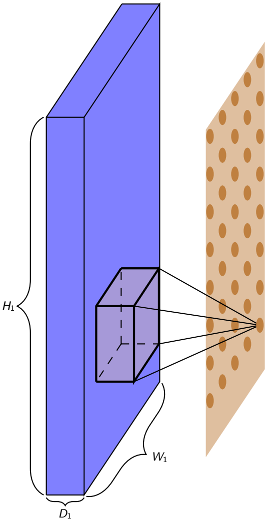
- Finally, coming to the 3d case.
- Each filter gives us one 2d output.
- K filters will give us K such 2D outputs
- We can think of the resulting output as $K \times W_2 \times H_2$ volume
- Thus equal.

- Thus equal.



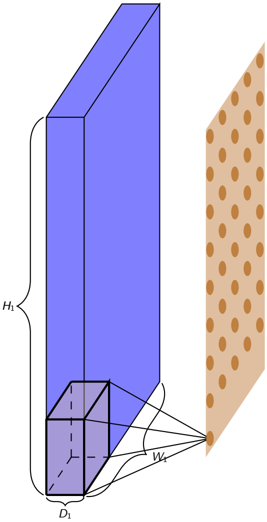
- Finally, coming to the 3d case.
- Each filter gives us one 2d output.
- K filters will give us K such 2D outputs
- We can think of the resulting output as $K \times W_2 \times H_2$ volume
- Thus equal.

- Thus equal.



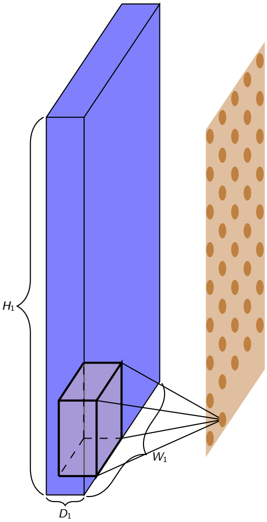
- Finally, coming to the 3d case.
- Each filter gives us one 2d output.
- K filters will give us K such 2D outputs
- We can think of the resulting output as $K \times W_2 \times H_2$ volume
- Thus equal.

- Thus equal.

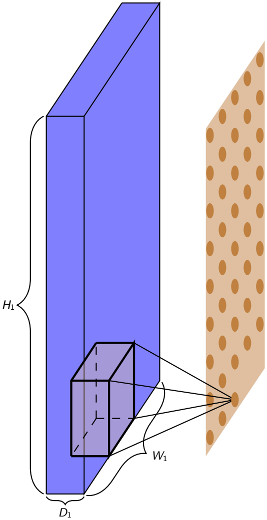


- Finally, coming to the 3d case.
- Each filter gives us one 2d output.
- K filters will give us K such 2D outputs
- We can think of the resulting output as $K \times W_2 \times H_2$ volume
- Thus equal.

- Thus equal.

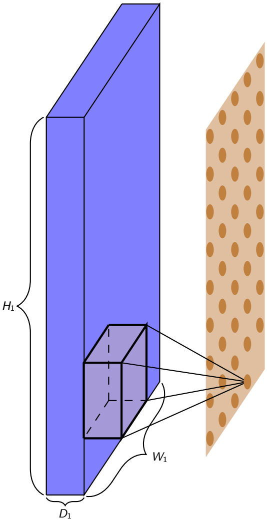


- Finally, coming to the 3d case.
- Each filter gives us one 2d output.
- K filters will give us K such 2D outputs
- We can think of the resulting output as $K \times W_2 \times H_2$ volume
- Thus equal.
- Thus equal.



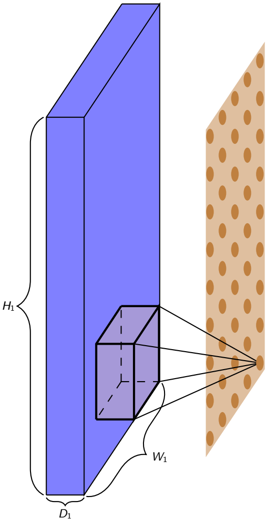
- Finally, coming to the 3d case.
- Each filter gives us one 2d output.
- K filters will give us K such 2D outputs
- We can think of the resulting output as $K \times W_2 \times H_2$ volume
- Thus equal.

- Thus equal.



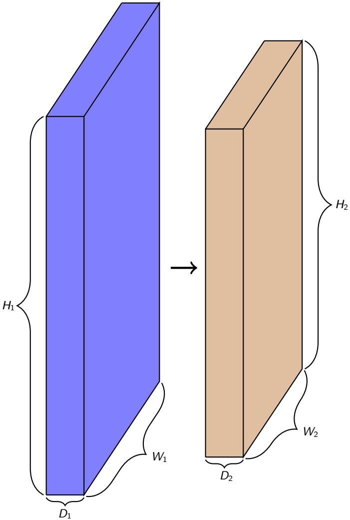
- Finally, coming to the 3d case.
- Each filter gives us one 2d output.
- K filters will give us K such 2D outputs
- We can think of the resulting output as $K \times W_2 \times H_2$ volume
- Thus equal.

- Thus equal.



- Finally, coming to the 3d case.
- Each filter gives us one 2d output.
- K filters will give us K such 2D outputs
- We can think of the resulting output as $K \times W_2 \times H_2$ volume
- Thus equal.

- Thus equal.



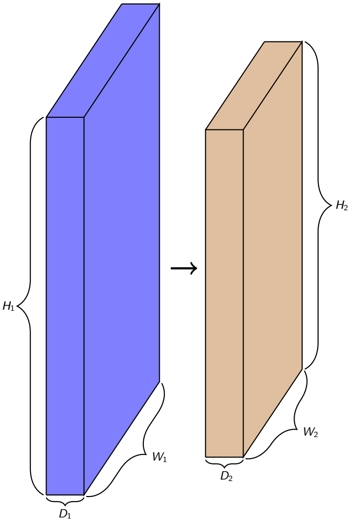
$$W_2 = \frac{W_1 - F + 2P}{S} + 1$$

$$H_2 = \frac{H_1 - F + 2P}{S} + 1$$

$$D_2 = K$$

- Finally, coming to the 3d case.
- Each filter gives us one 2d output.
- K filters will give us K such 2D outputs
- We can think of the resulting output as $K \times W_2 \times H_2$ volume
- Thus equal.

- Thus equal.



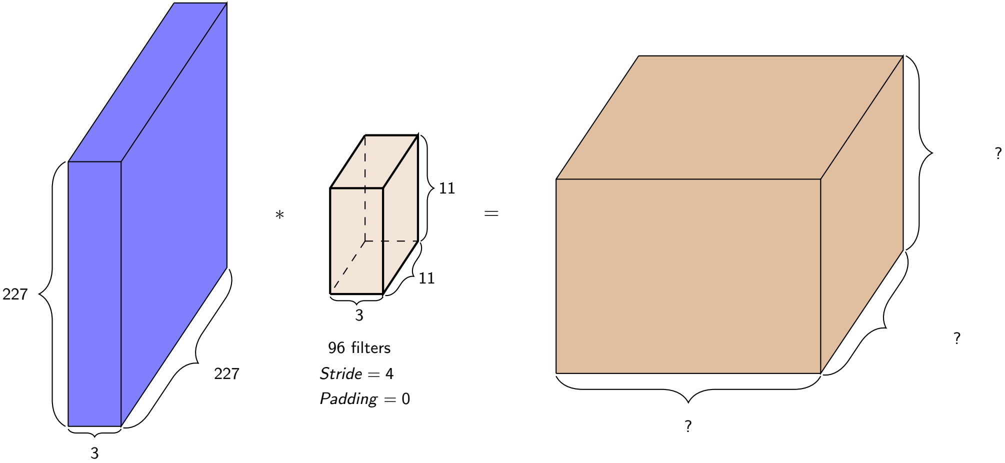
$$W_2 = \frac{W_1 - F + 2P}{S} + 1$$

$$H_2 = \frac{H_1 - F + 2P}{S} + 1$$

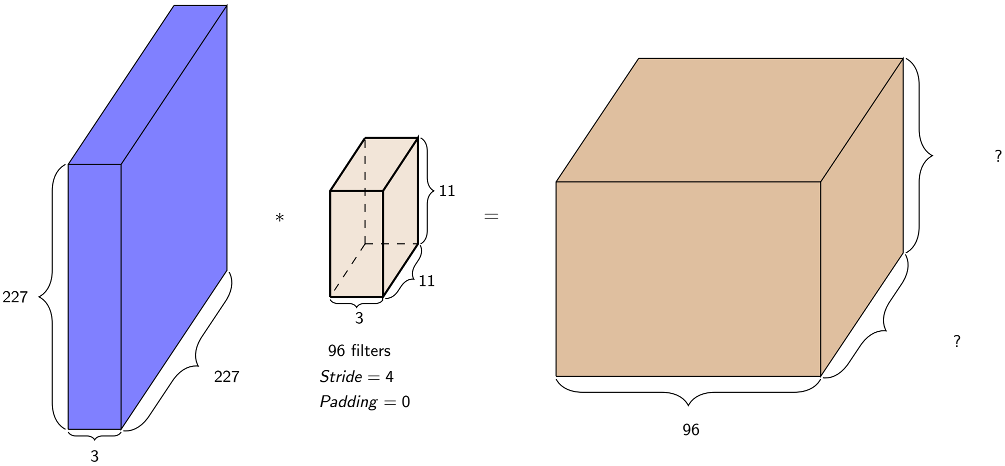
$$D_2 = K$$

- Finally, coming to the 3d case.
- Each filter gives us one 2d output.
- K filters will give us K such 2D outputs
- We can think of the resulting output as $K \times W_2 \times H_2$ volume
- Thus equal.
- The depth of the the resulting output as $K \times W_2 \times H_2$ volume
- Thus equal.
- The depth of the output is equal to number of filters.

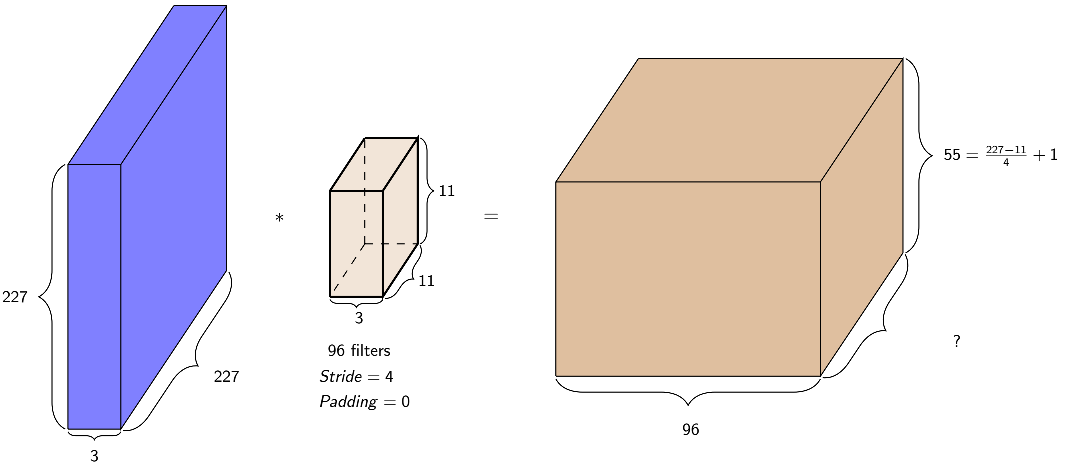
Let us do a few exercises



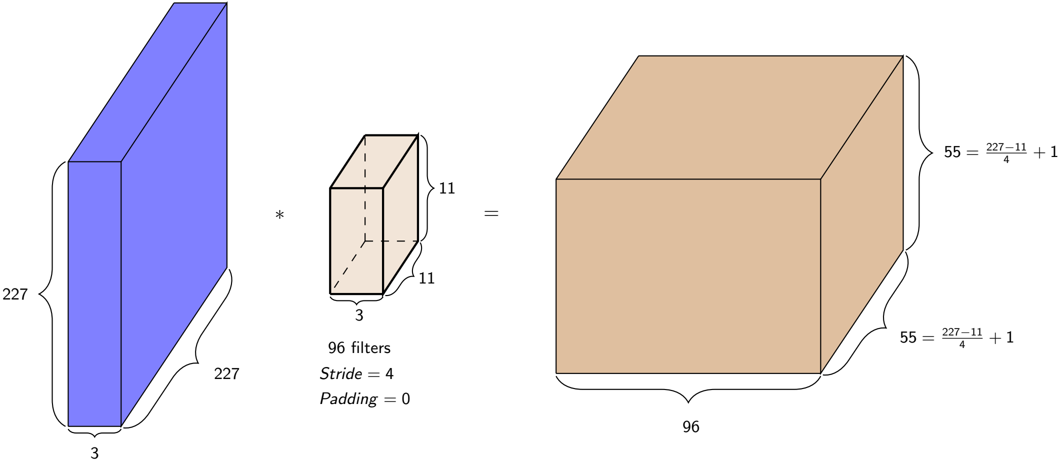
Let us do a few exercises



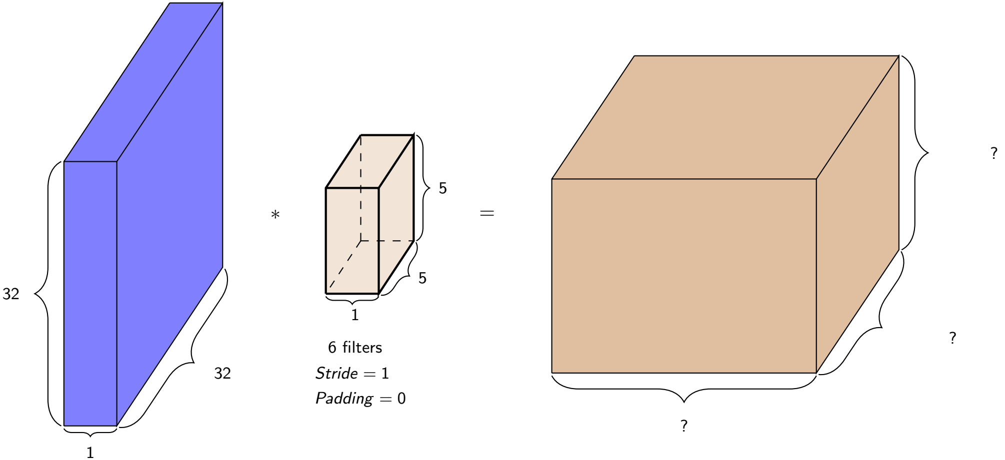
Let us do a few exercises



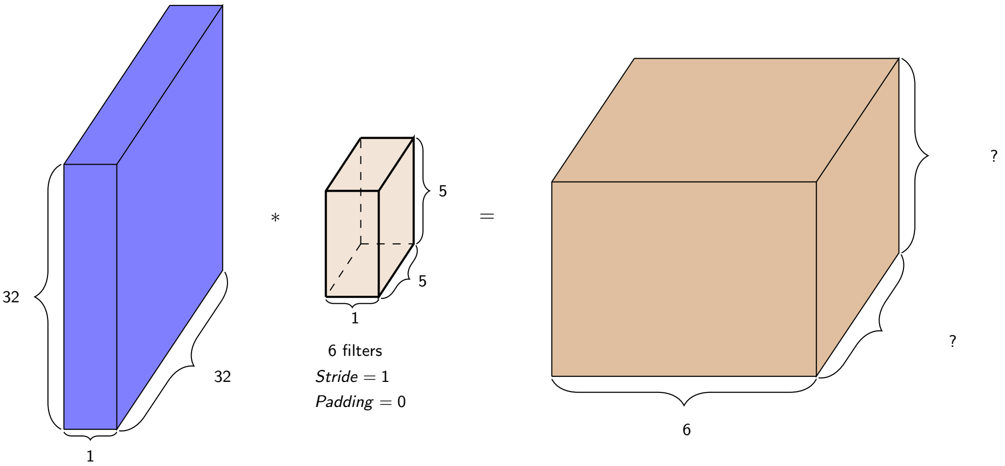
Let us do a few exercises



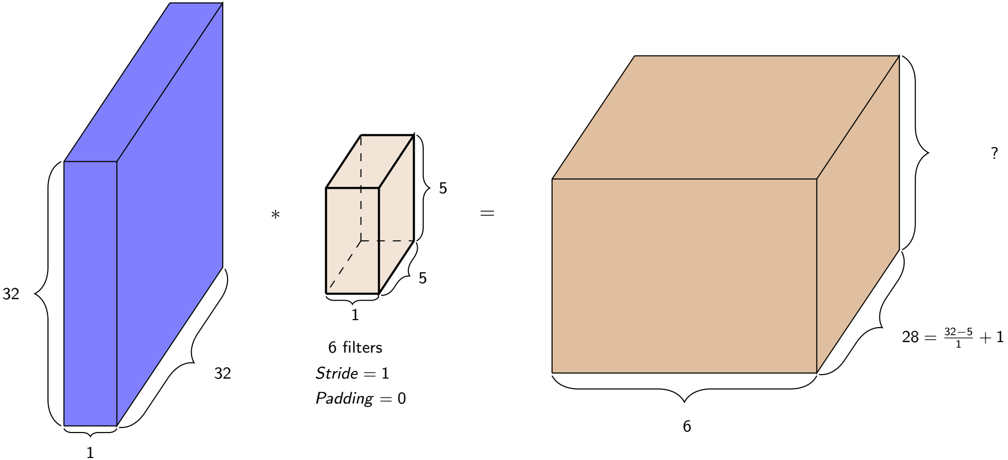
Let us do a few exercises



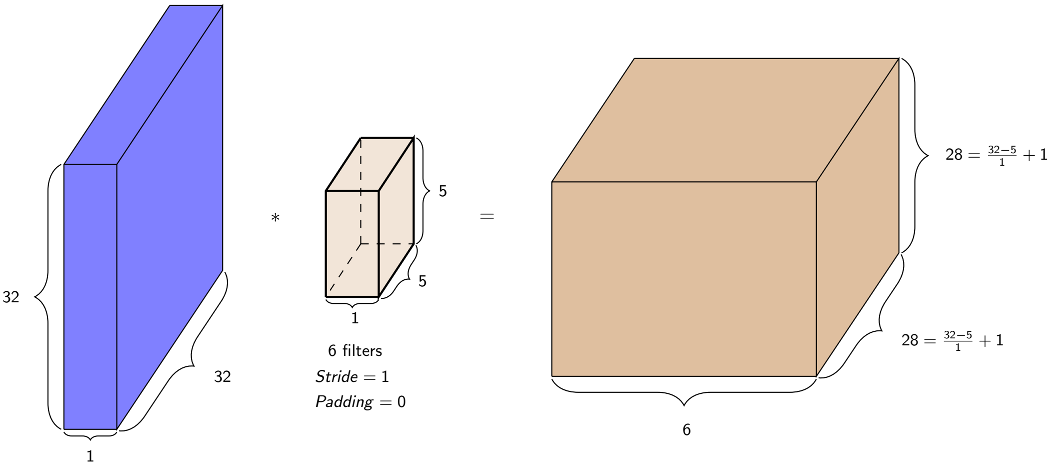
Let us do a few exercises



Let us do a few exercises



Let us do a few exercises



Putting things into perspective

Putting things into perspective

- What is the connection between this operation (convolution) and neural networks?

24/19

Putting things into perspective

- What is the connection between this operation (convolution) and neural networks?
- We will try to understand this by considering the task of "image classification".

Features

Raw pixels

Features

Raw pixels →

→ car, bus, monument, flower

Features

Raw pixels ⟶

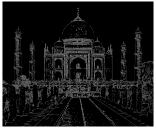
⟶ car, bus, monument, flower

Features

Raw pixels →

→ car, bus, monument, flower

Edge Detector →

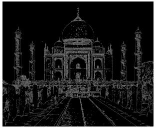


Features

Raw pixels ⟶

⟶ car, bus, monument, flower

Edge Detector ⟶



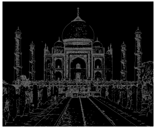
⟶ car, bus, monument, flower

Features

Raw pixels ⟶

⟶ car, bus, monument, flower

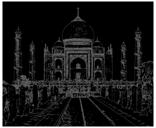
Edge Detector ⟶

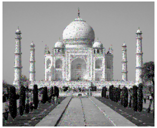
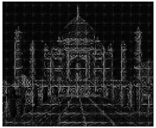


⟶ car, bus, monument, flower

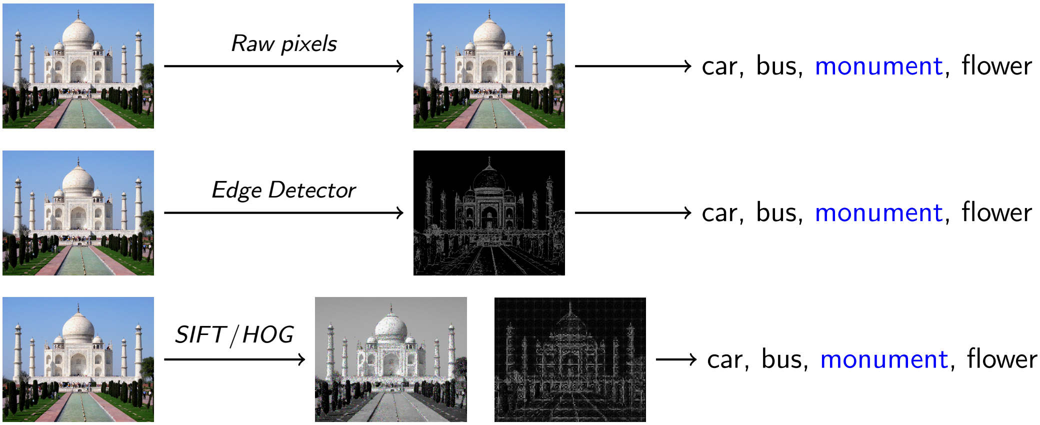
Features

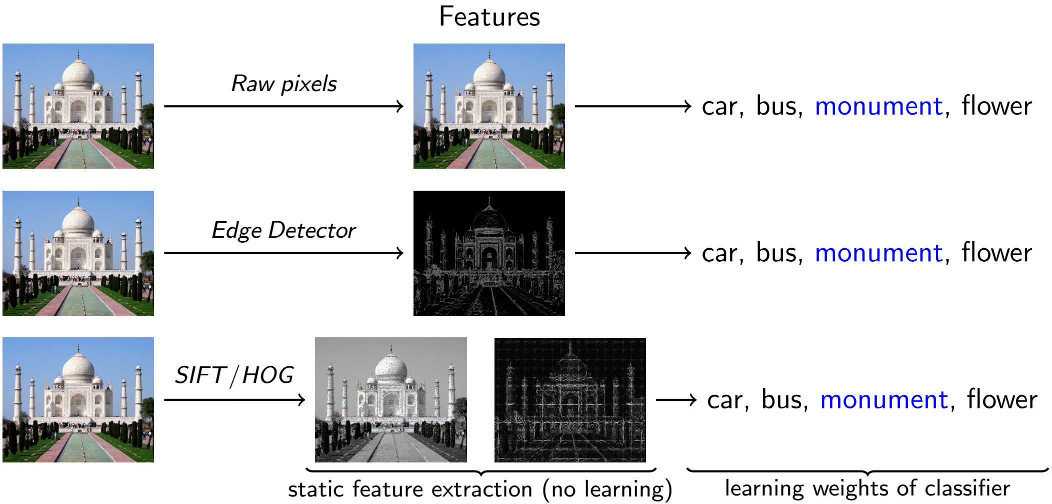
 *Raw pixels* →  → car, bus, monument, flower

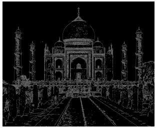
 *Edge Detector* →  → car, bus, monument, flower

 *SIFT / HOG* →  

Features

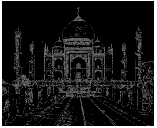

Raw pixels
car, bus, monument, flower
Edge Detector
car, bus, monument, flower
SIFT / HOG
car, bus, monument, flower


Features
Raw pixels
car, bus, monument, flower
Edge Detector
car, bus, monument, flower
SIFT / HOG
car, bus, monument, flower
static feature extraction (no learning)
learning weights of classifier



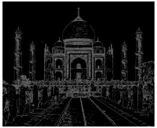
0	0	0	0	0
0	1	1	1	0
0	1	-8	1	0
0	1	1	1	0
0	0	0	0	0

→

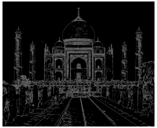


→ car, bus, monument, flower

0	0	0	0	0
0	1	1	1	0
0	1	-8	1	0
0	1	1	1	0
0	0	0	0	0

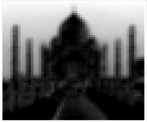
 ⟶  ⟶ car, bus, monument, flower

0	0	0	0	0
0	1	1	1	0
0	1	-8	1	0
0	1	1	1	0
0	0	0	0	0



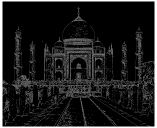
car, bus, monument, flower

0	0	0	0	0
0	1	1	1	0
0	1	-8	1	0
0	1	1	1	0
0	0	0	0	0



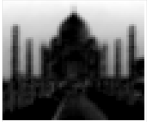
-1.21358689e-03	3.23652686e-03	···	···	-2.06615720e-02
-1.52757822e-03	2.36130832e-03	···	···	-1.19824838e-02
⋮	⋮			⋮
⋮	⋮			⋮
-8.25322699e-04	-5.14897937e-03	···	···	-9.90395527e-03

26/19



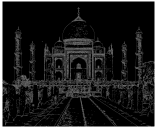
car, bus, monument, flower

0	0	0	0	0
0	1	1	1	0
0	1	-8	1	0
0	1	1	1	0
0	0	0	0	0



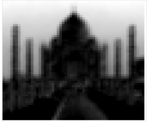
car, bus, monument, flower

-1.21358689e-03	3.23652686e-03	⋯	⋯	-2.06615720e-02
-1.52757822e-03	2.36130832e-03	⋯	⋯	-1.19824838e-02
⋮	⋮			⋮
⋮	⋮			⋮
-8.25322699e-04	-5.14897937e-03	⋯	⋯	-9.90395527e-03



car, bus, monument, flower

0	0	0	0	0
0	1	1	1	0
0	1	-8	1	0
0	1	1	1	0
0	0	0	0	0

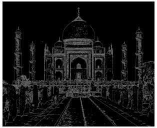


car, bus, monument, flower

-1.21358689e-03	3.23652686e-03	⋯	⋯	-2.06615720e-02
-1.52757822e-03	2.36130832e-03	⋯	⋯	-1.19824838e-02
⋮	⋮			⋮
⋮	⋮			⋮
-8.25322699e-04	-5.14897937e-03	⋯	⋯	-9.90395527e-03

Instead of using handcrafted kernels such as edge detectors Can we learn meaningful kernels/filters in addition to learning the weights of the classifier?

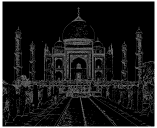
Instead of using handcrafted kernels (such as edge detectors)can we learn meaningful kernels/filters in addition to learning the weights of the classifier?



0	0	0	0	0
0	1	1	1	0
0	1	-8	1	0
0	1	1	1	0
0	0	0	0	0

Instead of using handcrafted kernels (such as edge detectors)can we learn meaningful kernels/filters in addition to learning the weights of the classifier?

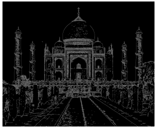
→



→ car, bus, monument, flower

0	0	0	0	0
0	1	1	1	0
0	1	-8	1	0
0	1	1	1	0
0	0	0	0	0

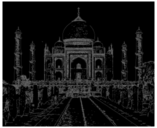
Instead of using handcrafted kernels (such as edge detectors)can we learn meaningful kernels/filters in addition to learning the weights of the classifier?



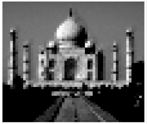
car, bus, monument, flower

0	0	0	0	0
0	1	1	1	0
0	1	-8	1	0
0	1	1	1	0
0	0	0	0	0

Instead of using handcrafted kernels (such as edge detectors)can we learn meaningful kernels/filters in addition to learning the weights of the classifier?

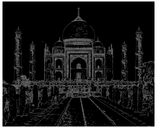
 →  → car, bus, monument, flower

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & -8 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

 → 

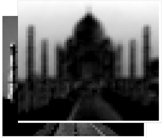
$$\begin{matrix} -1.21358689e\text{-}03 & 3.23652686e\text{-}03 & \cdots & \cdots & -2.06615720e\text{-}02 \\ -1.52757822e\text{-}03 & 2.36130832e\text{-}03 & \cdots & \cdots & -1.19824838e\text{-}02 \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ -8.25322699e\text{-}04 & -5.14897937e\text{-}03 & \cdots & \cdots & -9.90395527e\text{-}03 \end{matrix}$$

Instead of using handcrafted kernels (such as edge detectors)can we learn meaningful kernels/filters in addition to learning the weights of the classifier?



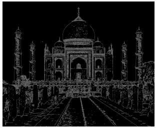
car, bus, monument, flower

0	0	0	0	0
0	1	1	1	0
0	1	-8	1	0
0	1	1	1	0
0	0	0	0	0



-0.02337041	-0.03243878	···	···	-0.04728875
-0.05375158	-0.05350766 ···	···	-0.04323674	
⋮	⋮			⋮
⋮	⋮			⋮
-0.00792501	-0.00503319	···	···	0.00174674

Instead of using handcrafted kernels (such as edge detectors)can we learn meaningful kernels/filters in addition to learning the weights of the classifier?

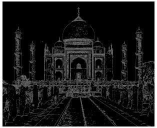


car, bus, monument, flower

0	0	0	0	0
0	1	1	1	0
0	1	-8	1	0
0	1	1	1	0
0	0	0	0	0

-0.01871333	-0.01075948	···	···	0.04684572
0.00104325	0.01935937	···	···	0.01016542
⋮	⋮			⋮
⋮	⋮			⋮
0.03008777	0.00335217	···	···	-0.02791128

Instead of using handcrafted kernels (such as edge detectors)can we learn meaningful kernels/filters in addition to learning the weights of the classifier?

 →  → car, bus, **monument**, flower

0	0	0	0	0
0	1	1	1	0
0	1	-8	1	0
0	1	1	1	0
0	0	0	0	0

 →  → car, bus, **monument**, flower

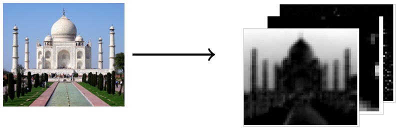
$$\begin{array}{cccc} -0.01871333 & -0.01075948 & \cdots\ \cdots & 0.04684572 \\ 0.00104325 & 0.01935937 & \cdots\ \cdots & 0.01016542 \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ 0.03008777 & 0.00335217 & \cdots\ \cdots & -0.02791128 \end{array}$$

Instead of using handcrafted kernels (such as edge detectors)can we learn meaningful kernels/filters in addition to learning the weights of the classifier?

- Can we learn multiple meaningful kernels/filters in addition to learning the weights of the classifier?

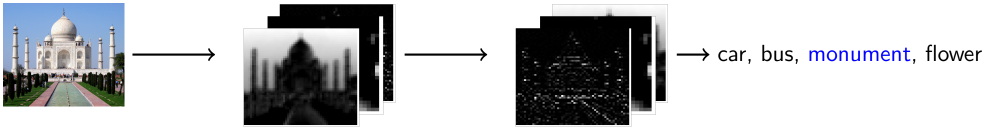
- Can we learn multiple meaningful kernels/filters in addition to learning the weights of the classifier?
- Yes, we can !

- Can we learn multiple meaningful kernels/filters in addition to learning the weights of the classifier?
- Yes, we can !

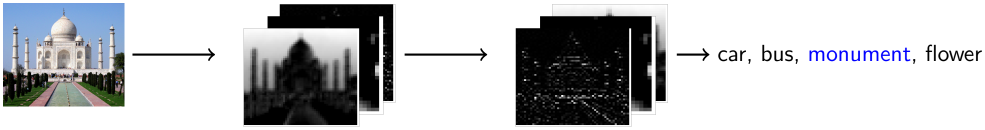


- Can we learn multiple meaningful kernels/filters in addition to learning the weights of the classifier?
- Yes, we can !

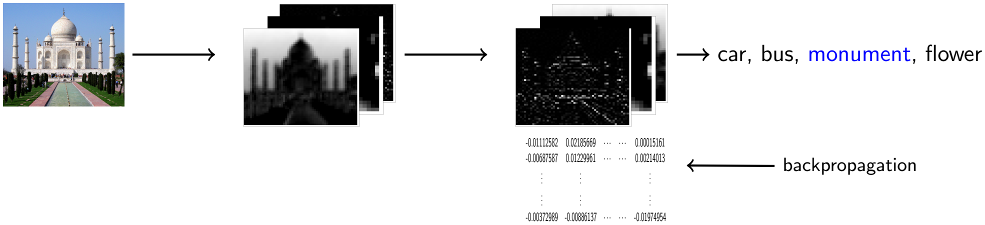
- Can we learn multiple meaningful kernels/filters in addition to learning the weights of the classifier?
- Yes, we can !



- Can we learn multiple meaningful kernels/filters in addition to learning the weights of the classifier?
- Yes, we can !

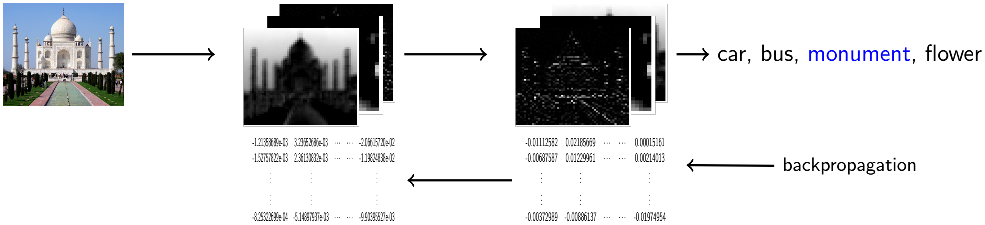


- Can we learn multiple meaningful kernels/filters in addition to learning the weights of the classifier?
- Yes, we can !

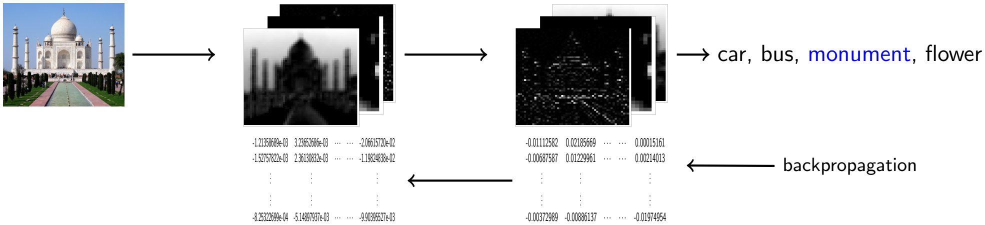


- Can we learn multiple meaningful kernels/filters in addition to learning the weights of the classifier?
- Yes, we can !

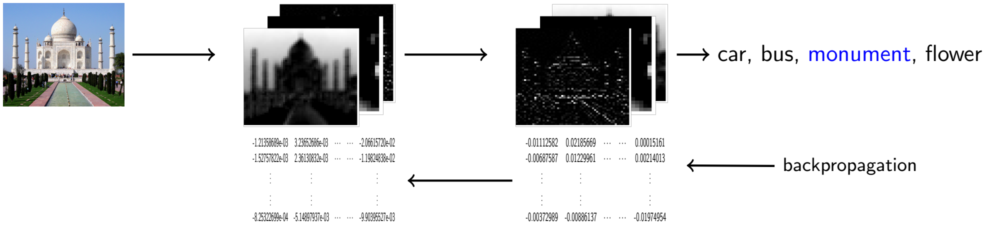
 28/19



- Can we learn multiple meaningful kernels/filters in addition to learning the weights of the classifier?
- Yes, we can !



- Can we learn multiple meaningful kernels/filters in addition to learning the weights of the classifier?
- Yes, we can !
- Simply by treating these kernels as parameters and learning them in addition to the weights of the classifier (using back propagation)



- Can we learn multiple meaningful kernels/filters in addition to learning the weights of the classifier?
- Yes, we can !
- Simply by treating these kernels as parameters and learning them in addition to the weights of the classifier (using back propagation)
- Such a network is called a Convolutional Neural Network.

- Okay, I get it that the idea is to learn the kernel/filters by just treating them as parameters of the classification model

- Okay, I get it that the idea is to learn the kernel/filters by just treating them as parameters of the classification model
- But how is this different from a regular feedforward neural network

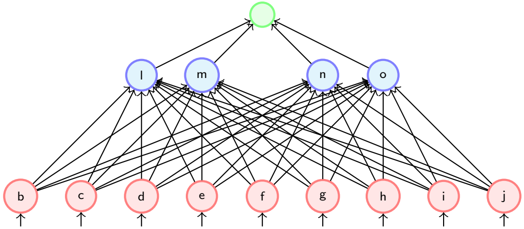
- Okay, I get it that the idea is to learn the kernel/filters by just treating them as parameters of the classification model
- But how is this different from a regular feedforward neural network
- Let us see

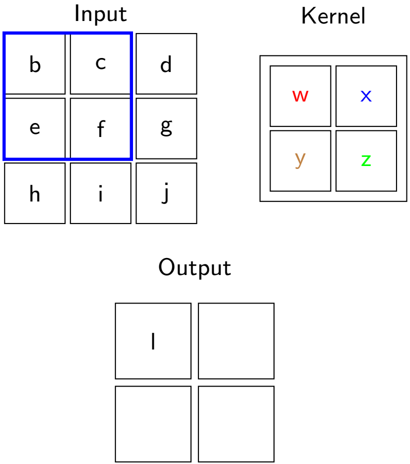
Input

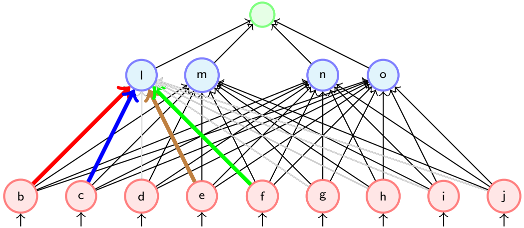
b	c	d
e	f	g
h	i	j

Kernel

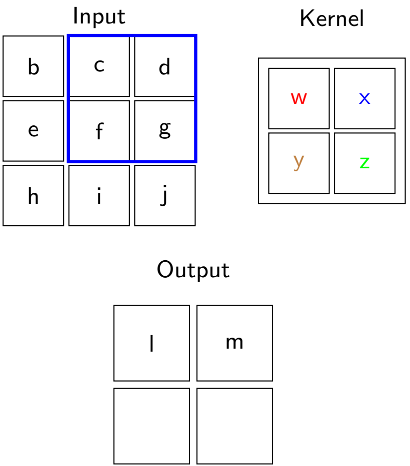
w	x
y	z

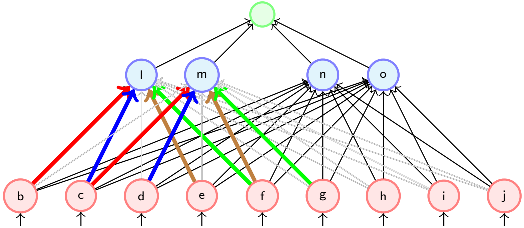




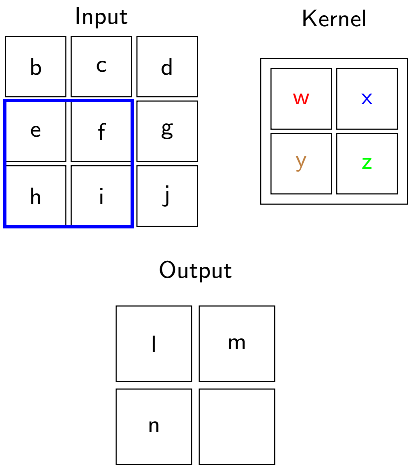


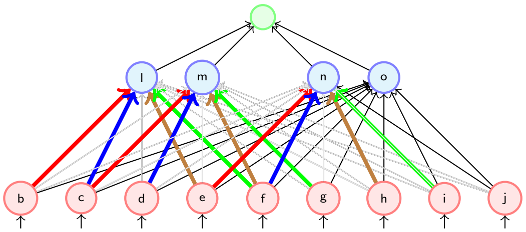
- $l = bw + cx + ey + hz$



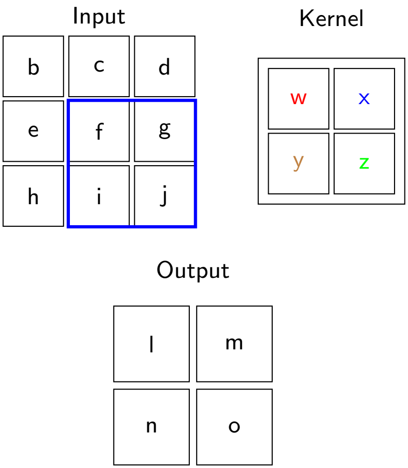


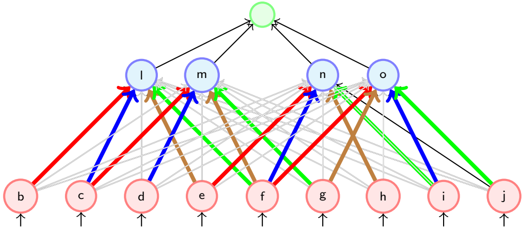
- $l = bw + cx + ey + hz$
- $m = cw + dx + fy + iz$





- $l = bw + cx + ey + hz$
- $m = cw + dx + fy + iz$
- $n = ew + fx + fy + iz$





- $l = bw + cx + ey + hz$
- $m = cw + dx + fy + iz$
- $n = ew + fx + fy + iz$
- $o = fw + gx + gy + jz$