Module 11.1 : The convolution operation
Suppose we are tracking the position of an aeroplane using a laser sensor at discrete time intervals.
Suppose we are tracking the position of an aeroplane using a laser sensor at discrete time intervals.
Suppose we are tracking the position of an aeroplane using a laser sensor at discrete time intervals.

\[ x_0, \ x_1, \ x_2 \]
Suppose we are tracking the position of an aeroplane using a laser sensor at discrete time intervals.

Now suppose our sensor is noisy.
Suppose we are tracking the position of an aeroplane 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 an aeroplane 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.
Suppose we are tracking the position of an aeroplane 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.
Suppose we are tracking the position of an aeroplane 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.
Suppose we are tracking the position of an aeroplane 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.
- The weight array \((w)\) 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.
- The weight array \((w)\) is known as the filter.
- We just slide the filter over the input and compute the value of \(s_t\) based on a window around \(x_t\).

<table>
<thead>
<tr>
<th>(w-6)</th>
<th>(w-5)</th>
<th>(w-4)</th>
<th>(w-3)</th>
<th>(w-2)</th>
<th>(w-1)</th>
<th>(w_0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>0.01</td>
<td>0.02</td>
<td>0.02</td>
<td>0.04</td>
<td>0.4</td>
<td>0.5</td>
</tr>
</tbody>
</table>

| \(X\)  | 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   |        |        |        |        |        |        |

\[ s_6 = x_6w_0 + x_5w_{-1} + x_4w_{-2} + x_3w_{-3} + x_2w_{-4} + x_1w_{-5} + x_0w_{-6} \]
\[
S_t = \sum_{a=0}^{6} x_{t-a}w_a
\]

- In practice, we would only sum over a small window
- The weight array \( (w) \) is known as the filter
- We just slide the filter over the input and compute the value of \( s_t \) based on a window around \( x_t \)

\[
\begin{array}{cccccccc}
  w_{-6} & w_{-5} & w_{-4} & w_{-3} & w_{-2} & w_{-1} & w_0 \\
  0.01 & 0.01 & 0.02 & 0.02 & 0.04 & 0.4 & 0.5
\end{array}
\]

\[
\begin{array}{cccccccccccc}
  X & 1.00 & 1.10 & 1.20 & 1.40 & 1.70 & 1.80 & 1.90 & 2.10 & 2.20 & 2.40 & 2.50 & 2.70
\end{array}
\]

\[
S = \begin{array}{cccc}
  1.80 & 1.96 & & \\
\end{array}
\]

\[
s_6 = x_6w_0 + x_5w_{-1} + x_4w_{-2} + x_3w_{-3} + x_2w_{-4} + x_1w_{-5} + x_0w_{-6}
\]
In practice, we would only sum over a small window.

The weight array ($w$) is known as the filter.

We just slide the filter over the input and compute the value of $s_t$ based on a window around $x_t$.

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

\[
\begin{array}{ccccccc}
  w_{-6} & w_{-5} & w_{-4} & w_{-3} & w_{-2} & w_{-1} & w_0 \\
  0.01 & 0.01 & 0.02 & 0.02 & 0.04 & 0.4 & 0.5 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
  X & 1.00 & 1.10 & 1.20 & 1.40 & 1.70 & 1.80 & 1.90 & 2.10 & 2.20 & 2.40 & 2.50 & 2.70 \\
\end{array}
\]

\[
\begin{array}{cccc}
  S & 1.80 & 1.96 & 2.11 & \text{empty} \\
\end{array}
\]

\[ s_6 = x_6w_0 + x_5w_{-1} + x_4w_{-2} + x_3w_{-3} + x_2w_{-4} + x_1w_{-5} + x_0w_{-6} \]
In practice, we would only sum over a small window.

The weight array \((w)\) is known as the filter.

We just slide the filter over the input and compute the value of \(s_t\) based on a window around \(x_t\).
\[ s_t = \sum_{a=0}^{6} x_{t-a} w_a \]

- In practice, we would only sum over a small window
- The weight array \((w)\) is known as the filter
- We just slide the filter over the input and compute the value of \(s_t\) based on a window around \(x_t\)

\[
\begin{array}{cccccc}
  w_{-6} & w_{-5} & w_{-4} & w_{-3} & w_{-2} & w_{-1} & w_0 \\
  0.01 & 0.01 & 0.02 & 0.02 & 0.04 & 0.4 & 0.5 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
  X & 1.00 & 1.10 & 1.20 & 1.40 & 1.70 & 1.80 & 1.90 & 2.10 & 2.20 & 2.40 & 2.50 & 2.70 \\
\end{array}
\]

\[
\begin{array}{cccc}
  S & 1.80 & 1.96 & 2.11 & 2.16 & 2.28 \\
\end{array}
\]

\[ s_6 = x_6 w_0 + x_5 w_{-1} + x_4 w_{-2} + x_3 w_{-3} + x_2 w_{-4} + x_1 w_{-5} + x_0 w_{-6} \]
\[ s_t = \sum_{a=0}^{6} x_{t-a} w_a \]

- In practice, we would only sum over a small window
- The weight array \( w \) is known as the filter
- We just slide the filter over the input and compute the value of \( s_t \) based on a window around \( x_t \)

<table>
<thead>
<tr>
<th>W</th>
<th>0.01</th>
<th>0.01</th>
<th>0.02</th>
<th>0.02</th>
<th>0.4</th>
<th>0.5</th>
</tr>
</thead>
</table>

| 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 |

\[ s_6 = x_6 w_0 + x_5 w_{-1} + x_4 w_{-2} + x_3 w_{-3} + x_2 w_{-4} + x_1 w_{-5} + x_0 w_{-6} \]
\[ s_t = \sum_{a=0}^{6} x_{t-a} w_a \]

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

The weight array \((w)\) is known as the filter.

We just slide the filter over the input and compute the value of \(s_t\) based on a window around \(x_t\).

Here the input (and the kernel) is one dimensional.

<table>
<thead>
<tr>
<th>X</th>
<th>1.00</th>
<th>1.10</th>
<th>1.20</th>
<th>1.40</th>
<th>1.70</th>
<th>1.80</th>
<th>1.90</th>
<th>2.10</th>
<th>2.20</th>
<th>2.40</th>
<th>2.50</th>
<th>2.70</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
<td>0.01</td>
<td>0.01</td>
<td>0.02</td>
<td>0.02</td>
<td>0.04</td>
<td>0.4</td>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>1.80</td>
<td>1.96</td>
<td>2.11</td>
<td>2.16</td>
<td>2.28</td>
<td>2.42</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ s_6 = x_6 w_0 + x_5 w_{-1} + x_4 w_{-2} + x_3 w_{-3} + x_2 w_{-4} + x_1 w_{-5} + x_0 w_{-6} \]
\[ s_t = \sum_{a=0}^{6} x_{t-a}w_a \]

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

<table>
<thead>
<tr>
<th>(w)</th>
<th>0.01</th>
<th>0.01</th>
<th>0.02</th>
<th>0.02</th>
<th>0.04</th>
<th>0.4</th>
<th>0.5</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>(X)</th>
<th>1.00</th>
<th>1.10</th>
<th>1.20</th>
<th>1.40</th>
<th>1.70</th>
<th>1.80</th>
<th>1.90</th>
<th>2.10</th>
<th>2.20</th>
<th>2.40</th>
<th>2.50</th>
<th>2.70</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>(S)</th>
<th>1.80</th>
<th>1.96</th>
<th>2.11</th>
<th>2.16</th>
<th>2.28</th>
<th>2.42</th>
</tr>
</thead>
</table>

\[ s_6 = x_6w_0 + x_5w_{-1} + x_4w_{-2} + x_3w_{-3} + x_2w_{-4} + x_1w_{-5} + x_0w_{-6} \]
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 ($m \times n$)
We can think of images as 2D inputs
We would now like to use a 2D filter $(m \times n)$
First let us see what the 2D formula looks like

$$S_{ij} = (I * K)_{ij} = \sum_{a=0}^{m-1} \sum_{b=0}^{n-1} I_{i-a, j-b} K_{a,b}$$
\begin{itemize}
  \item We can think of images as 2D inputs.
  \item We would now like to use a 2D filter \((m \times n)\).
  \item First let us see what the 2D formula looks like.
  \item This formula looks at all the preceding neighbours \((i - a, j - b)\).
\end{itemize}

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

- We can think of images as 2D inputs
- We would now like to use a 2D filter \((m \times n)\)
- 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.

Input

\[
\begin{array}{cccc}
  a & b & c & d \\
  e & f & g & h \\
  i & j & k & \ell \\
\end{array}
\]

Kernel

\[
\begin{array}{cc}
  w & x \\
  y & z \\
\end{array}
\]

Output

\[
\begin{array}{ccc}
  aw+bx+ey+fz & bw+cx+fy+gz & cw+dx+gy+hz \\
  ew+fx+iy+jz & fw+gx+jy+kz & gw+hx+ky+\ell z \\
\end{array}
\]
For the rest of the discussion we will use the following formula for convolution:

$$S_{ij} = \left(I \ast K\right)_{ij} = \lfloor m_2 \rfloor \sum_{a=\lfloor -m_2 \rfloor}^{\lfloor n_2 \rfloor} I_i - a, j - b K_{m_2} + a, n_2 + b \text{ pixel of interest}$$
\[ S_{ij} = (I \ast K)_{ij} = \sum_{a=\lfloor -\frac{m}{2} \rfloor}^{\lfloor \frac{m}{2} \rfloor} \sum_{b=\lfloor -\frac{n}{2} \rfloor}^{\lfloor \frac{n}{2} \rfloor} I_{i-a,j-b} K_{\frac{m}{2} + a, \frac{n}{2} + b} \]

For the rest of the discussion we will use the following formula for convolution.
\[ S_{ij} = (I * K)_{ij} = \sum_{a=-\left\lfloor \frac{m}{2} \right\rfloor}^{\left\lfloor \frac{m}{2} \right\rfloor} \sum_{b=-\left\lfloor \frac{n}{2} \right\rfloor}^{\left\lfloor \frac{n}{2} \right\rfloor} I_{i-a,j-b}K_{m/2+a,n/2+b} \]

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 preceding and succeeding neighbors.
\[
S_{ij} = (I \ast K)_{ij} = \sum_{a = \left\lfloor -\frac{m}{2} \right\rfloor}^{\left\lfloor \frac{m}{2} \right\rfloor} \sum_{b = \left\lfloor -\frac{n}{2} \right\rfloor}^{\left\lfloor \frac{n}{2} \right\rfloor} I_{i-a,j-b}K_{\frac{m}{2}+a, \frac{n}{2}+b}
\]

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 preceding and succeeding neighbors.
Let us see some examples of 2D convolutions applied to images
\[
\begin{array}{ccc}
1 & 1 & 1 \\
\times & 1 & 1 & 1 \\
\hline
1 & 1 & 1
\end{array}
\]
blurs the image
\[
\begin{bmatrix}
0 & -1 & 0 \\
-1 & 5 & -1 \\
0 & -1 & 0
\end{bmatrix}
\]

\[
\begin{bmatrix}
0 & -1 & 0 \\
-1 & 5 & -1 \\
0 & -1 & 0
\end{bmatrix}
\]

\[
= 
\]
\[
\begin{bmatrix}
0 & -1 & 0 \\
-1 & 5 & -1 \\
0 & -1 & 0
\end{bmatrix}
\]

sharpens the image
\[
\begin{bmatrix}
1 & 1 & 1 \\
1 & -8 & 1 \\
1 & 1 & 1
\end{bmatrix}
\times
\begin{bmatrix}
1 \\
1 \\
1
\end{bmatrix}
= 
\begin{bmatrix}
1 \\
1 \\
1
\end{bmatrix}
\]
\[
\begin{array}{ccc}
1 & 1 & 1 \\
\ast & 1 & -8 & 1 \\
1 & 1 & 1 \\
\end{array}
= \\
detects the edges
\]
We will now see a working example of 2D convolution.
• We just slide the kernel over the input image
- We just slide the kernel over the input image
- Each time we slide the kernel we get one value in the output

The resulting output is called a feature map.

We can use multiple filters to get multiple feature maps.
We just slide the kernel over the input image.

Each time we slide the kernel we get one value in the output.
- We just slide the kernel over the input image
- Each time we slide the kernel we get one value in the output

The resulting output is called a feature map. We can use multiple filters to get multiple feature maps.
- We just slide the kernel over the input image
- Each time we slide the kernel we get one value in the output
- We just slide the kernel over the input image
- Each time we slide the kernel we get one value in the output

We can use multiple filters to get multiple feature maps.
- We just slide the kernel over the input image
- Each time we slide the kernel we get one value in the output

The resulting output is called a feature map.

We can use multiple filters to get multiple feature maps.
- We just slide the kernel over the input image.
- Each time we slide the kernel we get one value in the output.

The resulting output is called a feature map. We can use multiple filters to get multiple feature maps.
- We just slide the kernel over the input image
- Each time we slide the kernel we get one value in the output
- We just slide the kernel over the input image.
- Each time we slide the kernel we get one value in the output.
- We just slide the kernel over the input image
- Each time we slide the kernel we get one value in the output

The resulting output is called a feature map.

We can use multiple filters to get multiple feature maps.
- We just slide the kernel over the input image
- Each time we slide the kernel we get one value in the output

The resulting output is called a feature map. We can use multiple filters to get multiple feature maps.
- We just slide the kernel over the input image
- Each time we slide the kernel we get one value in the output

The resulting output is called a feature map.
- We just slide the kernel over the input image
- Each time we slide the kernel we get one value in the output
- We just slide the kernel over the input image
- Each time we slide the kernel we get one value in the output

The resulting output is called a feature map.

We can use multiple filters to get multiple feature maps.
- We just slide the kernel over the input image
- Each time we slide the kernel we get one value in the output
- We just slide the kernel over the input image
- Each time we slide the kernel we get one value in the output

Mitesh M. Khapra
CS7015 (Deep Learning) : Lecture 11
- We just slide the kernel over the input image
- Each time we slide the kernel we get one value in the output
- We just slide the kernel over the input image
- Each time we slide the kernel we get one value in the output
- We just slide the kernel over the input image
- Each time we slide the kernel we get one value in the output
- The resulting output is called a feature map.
- We just slide the kernel over the input image
- Each time we slide the kernel we get one value in the output
- The resulting output is called a feature map.
- We can use multiple filters to get multiple feature maps.
Question

- In the 1D case, we slide a one dimensional filter over a one dimensional input
Question

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

What would happen in the 3D case?
In the 1D case, we slide a one dimensional filter over a one dimensional input.

What would happen in the 3D case?
In the 1D case, we slide a one dimensional filter over a one dimensional input.
Question

- In the 1D case, we slide a one dimensional filter over a one dimensional input.
In the 1D case, we slide a one dimensional filter over a one dimensional input.

What would happen in the 3D case?
Question

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

Mitesh M. Khapra
CS7015 (Deep Learning) : Lecture 11
Question

- In the 1D case, we slide a one dimensional filter over a one dimensional input.
- In the 2D case, we slide a two dimensional filter over a two dimensional output.

What would happen in the 3D case?
Question

- In the 1D case, we slide a one dimensional filter over a one dimensional input
- In the 2D case, we slide a two dimensional filter over a two dimensional output
Question

- In the 1D case, we slide a one dimensional filter over a one dimensional input.
- In the 2D case, we slide a two dimensional filter over a two dimensional output.

What would happen in the 3D case?
Question

- In the 1D case, we slide a one dimensional filter over a one dimensional input
- In the 2D case, we slide a two dimensional filter over a two dimensional output
Question

- In the 1D case, we slide a one dimensional filter over a one dimensional input.
- In the 2D case, we slide a two dimensional filter over a two dimensional output.

What would happen in the 3D case?
Question

- In the 1D case, we slide a one dimensional filter over a one dimensional input.
- In the 2D case, we slide a two dimensional filter over a two dimensional output.
Question

- In the 1D case, we slide a one dimensional filter over a one dimensional input
- In the 2D case, we slide a two dimensional filter over a two dimensional output
- What would happen in the 3D case?
What would a 3D filter look like?

Once again we will slide the volume over the 3D input and compute the convolution operation. Note that in this lecture we will assume that the filter always extends to the depth of the image. In effect, we are doing a 2D convolution operation on a 3D input (because the filter moves along the height and the width but not along the depth). As a result, the output will be 2D (only width and height, no depth). Once again, we can apply multiple filters to get multiple feature maps.
What would a 3D filter look like?

It will be 3D and we will refer to it as a volume.
What would a 3D filter look like?
- It will be 3D and we will refer to it as a volume.
- Once again we will slide the volume over the 3D input and compute the convolution operation.

Note that in this lecture we will assume that the filter always extends to the depth of the image. In effect, we are doing a 2D convolution operation on a 3D input (because the filter moves along the height and the width but not along the depth). As a result, the output will be 2D (only width and height, no depth). Once again we can apply multiple filters to get multiple feature maps.
What would a 3D filter look like?

- It will be 3D and we will refer to it as a volume.
- Once again we will slide the volume over the 3D input and compute the convolution operation.

In effect, we are doing a 2D convolution operation on a 3D input (because the filter moves along the height and the width but not along the depth).

As a result, the output will be 2D (only width and height, no depth).

Once again, we can apply multiple filters to get multiple feature maps.
What would a 3D filter look like?
- It will be 3D and we will refer to it as a volume.
- Once again we will slide the volume over the 3D input and compute the convolution operation.

Note that in this lecture we will assume that the filter always extends to the depth of the image. In effect, we are doing a 2D convolution operation on a 3D input (because the filter moves along the height and the width but not along the depth). As a result, the output will be 2D (only width and height, no depth).

Once again we can apply multiple filters to get multiple feature maps.
What would a 3D filter look like?
- It will be 3D and we will refer to it as a volume
- Once again we will slide the volume over the 3D input and compute the convolution operation

Note that in this lecture we will assume that the filter always extends to the depth of the image. In effect, we are doing a 2D convolution operation on a 3D input (because the filter moves along the height and the width but not along the depth). As a result, the output will be 2D (only width and height, no depth). Once again, we can apply multiple filters to get multiple feature maps.
What would a 3D filter look like?
- It will be 3D and we will refer to it as a volume
- Once again we will slide the volume over the 3D input and compute the convolution operation
What would a 3D filter look like?
It will be 3D and we will refer to it as a volume
Once again we will slide the volume over the 3D input and compute the convolution operation
• What would a 3D filter look like?
• It will be 3D and we will refer to it as a volume
• Once again we will slide the volume over the 3D input and compute the convolution operation

Note that in this lecture we will assume that the filter always extends to the depth of the image. In effect, we are doing a 2D convolution operation on a 3D input (because the filter moves along the height and the width but not along the depth). As a result, the output will be 2D (only width and height, no depth). Once again we can apply multiple filters to get multiple feature maps.
What would a 3D filter look like?
It will be 3D and we will refer to it as a volume
Once again we will slide the volume over the 3D input and compute the convolution operation

In effect, we are doing a 2D convolution operation on a 3D input (because the filter moves along the height and the width but not along the depth)
As a result the output will be 2D (only width and height, no depth)
Once again we can apply multiple filters to get multiple feature maps
What would a 3D filter look like?
- It will be 3D and we will refer to it as a volume
- Once again we will slide the volume over the 3D input and compute the convolution operation

Note that in this lecture we will assume that the filter always extends to the depth of the image. In effect, we are doing a 2D convolution operation on a 3D input (because the filter moves along the height and the width but not along the depth). As a result, the output will be 2D (only width and height, no depth). Once again, we can apply multiple filters to get multiple feature maps.
What would a 3D filter look like?
- It will be 3D and we will refer to it as a volume
- Once again we will slide the volume over the 3D input and compute the convolution operation
- Note that in this lecture we will assume that the filter always extends to the depth of the image
- In effect, we are doing a 2D convolution operation on a 3D input (because the filter moves along the height and the width but not along the depth)
- As a result the output will be 2D (only width and height, no depth)
- Once again we can apply multiple filters to get multiple feature maps
What would a 3D filter look like?
- It will be 3D and we will refer to it as a volume
- Once again we will slide the volume over the 3D input and compute the convolution operation

Note that in this lecture we will assume that the filter always extends to the depth of the image. In effect, we are doing a 2D convolution operation on a 3D input (because the filter moves along the height and the width but not along the depth). As a result, the output will be 2D (only width and height, no depth). Once again, we can apply multiple filters to get multiple feature maps.
What would a 3D filter look like?

- It will be 3D and we will refer to it as a volume
- Once again we will slide the volume over the 3D input and compute the convolution operation

Note that in this lecture we will assume that the filter always extends to the depth of the image. In effect, we are doing a 2D convolution operation on a 3D input (because the filter moves along the height and the width but not along the depth). As a result, the output will be 2D (only width and height, no depth). Once again, we can apply multiple filters to get multiple feature maps.
What would a 3D filter look like?
- It will be 3D and we will refer to it as a volume
- Once again we will slide the volume over the 3D input and compute the convolution operation
- Note that in this lecture we will assume that the filter always extends to the depth of the image

In effect, we are doing a 2D convolution operation on a 3D input (because the filter moves along the height and the width but not along the depth)

As a result the output will be 2D (only width and height, no depth)

Once again we can apply multiple filters to get multiple feature maps
What would a 3D filter look like?

It will be 3D and we will refer to it as a volume.

Once again we will slide the volume over the 3D input and compute the convolution operation.

Note that in this lecture we will assume that the filter always extends to the depth of the image.

In effect, we are doing a 2D convolution operation on a 3D input (because the filter moves along the height and the width but not along the depth).
What would a 3D filter look like?
It will be 3D and we will refer to it as a volume
Once again we will slide the volume over the 3D input and compute the convolution operation
Note that in this lecture we will assume that the filter always extends to the depth of the image
In effect, we are doing a 2D convolution operation on a 3D input (because the filter moves along the height and the width but not along the depth)
As a result the output will be 2D (only width and height, no depth)
What would a 3D filter look like?
- It will be 3D and we will refer to it as a volume
- Once again we will slide the volume over the 3D input and compute the convolution operation
- Note that in this lecture we will assume that the filter always extends to the depth of the image
- In effect, we are doing a 2D convolution operation on a 3D input (because the filter moves along the height and the width but not along the depth)
- As a result the output will be 2D (only width and height, no depth)
- Once again we can apply multiple filters to get multiple feature maps
Module 11.2 : Relation between input size, output size and filter size
So far we have not said anything explicit about the dimensions of the inputs, filters, and outputs and the relations between them. We will see how they are related, but before that, we will define a few quantities.
So far we have not said anything explicit about the dimensions of the inputs.
So far we have not said anything explicit about the dimensions of the inputs and filters. We will see how they are related but before that we will define a few quantities.
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 inputs, filters, and outputs and the relations between them.
So far we have not said anything explicit about the dimensions of the
1 inputs
2 filters
3 outputs

and the 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:

- **Width** ($W_1$)
- **Height** ($H_1$)
- **Depth** ($D_1$)
- **The Stride** ($S$) (We will come back to this later)
- The number of filters ($K$)
- The spatial extent ($F$) 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 $D_2$).
We first define the following quantities:

- Width ($W_1$),

The Stride $S$ (We will come back to this later)

The number of filters $K$

The spatial extent ($F$) 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 $D_2$)
- 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)
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 extent ($F$) 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 extent \( F \) 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 \( D_2 \))
Let us compute the dimension \((W_2, H_2)\) of the output.

Notice that we can’t place the kernel at the corners as it will cross the input boundary. This is true for all the shaded points (the kernel crosses the input boundary) which results in an output which is of smaller dimensions than the input.
Let us compute the dimension \((W_2, H_2)\) of the output.

Notice that we can't place the kernel at the corners as it will cross the input boundary. This is true for all the shaded points (the kernel crosses the input boundary). This results in an output which is of smaller dimensions than the input.
Let us compute the dimension \((W_2, H_2)\) of the output.
Let us compute the dimension \((W_2, H_2)\) of the output.

Notice that we can’t place the kernel at the corners as it will cross the input boundary. This is true for all the shaded points (the kernel crosses the input boundary). This results in an output which is of smaller dimensions than the input.
Let us compute the dimension \((W_2, H_2)\) of the output.
Let us compute the dimension \((W_2, H_2)\) of the output.

Notice that we can't place the kernel at the corners as it will cross the input boundary. This is true for all the shaded points (the kernel crosses the input boundary). This results in an output which is of smaller dimensions than the input.
Let us compute the dimension $(W_2, H_2)$ of the output.

Notice that we can't place the kernel at the corners as it will cross the input boundary. This is true for all the shaded points (the kernel crosses the input boundary). This results in an output which is of smaller dimensions than the input.
Let us compute the dimension \((W_2, H_2)\) of the output.

- Notice that we can’t place the kernel at the corners as it will cross the input boundary.
Let us compute the dimension \((W_2, H_2)\) of the output. Notice that we can’t place the kernel at the corners as it will cross the input boundary.
Let us compute the dimension \((W_2, H_2)\) of the output.

Notice that we can’t place the kernel at the corners as it will cross the input boundary.

This is true for all the shaded points (the kernel crosses the input boundary).
Let us compute the dimension \((W_2, H_2)\) of the output.

Notice that we can’t place the kernel at the corners as it will cross the input boundary.

This is true for all the shaded points (the kernel crosses the input boundary).

This results in an output which is of smaller dimensions than the input.
In general, 
\[ W_2 = W_1 - F + 1 \]
\[ H_2 = H_1 - F + 1 \]

We will refine this formula further

Let us compute the dimension \((W_2, H_2)\) of the output

Notice that we can’t place the kernel at the corners as it will cross the input boundary

This is true for all the shaded points (the kernel crosses the input boundary)

This results in an output which is of smaller dimensions than the input

As the size of the kernel increases, this becomes true for even more pixels
In general, 
\[ W_2 = W_1 - F + 1 \]
\[ H_2 = H_1 - F + 1 \]

We will refine this formula further. Let us compute the dimension \((W_2, H_2)\) of the output.

- Notice that we can’t place the kernel at the corners as it will cross the input boundary.
- This is true for all the shaded points (the kernel crosses the input boundary).
- This results in an output which is of smaller dimensions than the input.
- As the size of the kernel increases, this becomes true for even more pixels.
- For example, let’s consider a \(5 \times 5\) kernel.

Notice that the shaded points result in the output being of smaller dimensions than the input.
In general, \[ W_2 = W_1 - F + 1 \]
\[ H_2 = H_1 - F + 1 \]

We will refine this formula further.

\[ \text{Let us compute the dimension } (W_2, H_2) \text{ of the output} \]
\[ \text{Notice that we can’t place the kernel at the corners as it will cross the input boundary} \]
\[ \text{This is true for all the shaded points (the kernel crosses the input boundary)} \]
\[ \text{This results in an output which is of smaller dimensions than the input} \]
\[ \text{As the size of the kernel increases, this becomes true for even more pixels} \]
\[ \text{For example, let’s consider a } 5 \times 5 \text{ kernel} \]
\[ \text{We have an even smaller output now} \]
In general,
\[ W_2 = W_1 - F + 1 \]
\[ H_2 = H_1 - F + 1 \]
We will refine this formula further.

Let us compute the dimension \((W_2, H_2)\) of the output.

Notice that we can’t place the kernel at the corners as it will cross the input boundary.

This is true for all the shaded points (the kernel crosses the input boundary).

This results in an output which is of smaller dimensions than the input.

As the size of the kernel increases, this becomes true for even more pixels.

For example, let’s consider a 5 \times 5 kernel.

We have an even smaller output now.
In general, $W_2 = W_1 - F + 1$

$H_2 = H_1 - F + 1$

We will refine this formula further.

Let us compute the dimension $(W_2, H_2)$ of the output.

Notice that we can’t place the kernel at the corners as it will cross the input boundary.

This is true for all the shaded points (the kernel crosses the input boundary).

This results in an output which is of smaller dimensions than the input.

As the size of the kernel increases, this becomes true for even more pixels.

For example, let’s consider a $5 \times 5$ kernel.

We have an even smaller output now.
In general, $W_2 = W_1 - F + 1$ and $H_2 = H_1 - F + 1$.

We will refine this formula further.

Let us compute the dimension $(W_2, H_2)$ of the output.

Notice that we can’t place the kernel at the corners as it will cross the input boundary.

This is true for all the shaded points (the kernel crosses the input boundary).

This results in an output which is of smaller dimensions than the input.

As the size of the kernel increases, this becomes true for even more pixels.

For example, let’s consider a $5 \times 5$ kernel.

We have an even smaller output now.
In general, \( W_2 = W_1 - F + 1 \)
\( H_2 = H_1 - F + 1 \)

We will refine this formula further.

Let us compute the dimension \((W_2, H_2)\) of the output.

- Notice that we can’t place the kernel at the corners as it will cross the input boundary.
- This is true for all the shaded points (the kernel crosses the input boundary).
- This results in an output which is of smaller dimensions than the input.
- As the size of the kernel increases, this becomes true for even more pixels.
- For example, let’s consider a \(5 \times 5\) kernel.
- We have an even smaller output now.
In general, \( W_2 = W_1 - F + 1 \)
\[ H_2 = H_1 - F + 1 \]

We will refine this formula further.

- Let us compute the dimension \((W_2, H_2)\) of the output.
- Notice that we can’t place the kernel at the corners as it will cross the input boundary.
- This is true for all the shaded points (the kernel crosses the input boundary).
- This results in an output which is of smaller dimensions than the input.
- As the size of the kernel increases, this becomes true for even more pixels.
- For example, let’s consider a 5 \( \times \) 5 kernel.
- We have an even smaller output now.
What if we want the output to be of same size as the input?
What if we want the output to be of same size as the input?

- We can use something known as padding

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

Let us use pad $P = 1$ with a $3 \times 3$ kernel.

This means we will add one row and one column of 0 inputs at the top, bottom, left and right.
What if we want the output to be of same size as the input?

- We can use something known as padding
- Pad the inputs with appropriate number of 0 inputs so that you can now apply the kernel at the corners
We now have,

\[ W_2 = W_1 - F + 2P + 1 \]

\[ H_2 = H_1 - F + 2P + 1 \]

We will refine this formula further.

What if we want the output to be of same size as the input?

We can use something known as padding.

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

Let us use pad \( P = 1 \) with a \( 3 \times 3 \) kernel.
We now have,

\[ W_2 = W_1 - F + 2P + 1 \]

\[ H_2 = H_1 - F + 2P + 1 \]

We will refine this formula further.

- What if we want the output to be of same size as the input?
- We can use something known as padding
- Pad the inputs with appropriate number of 0 inputs so that you can now apply the kernel at the corners
- Let us use pad \( P = 1 \) with a \( 3 \times 3 \) kernel
- This means we will add one row and one column of 0 inputs at the top, bottom, left and right
We now have,

\[ W_2 = W_1 - F + 2P_1 + H_2 = H_1 - F + 2P_1 \]

We will refine this formula further.

What if we want the output to be of same size as the input?

- We can use something known as padding
- Pad the inputs with appropriate number of 0 inputs so that you can now apply the kernel at the corners
- Let us use pad P = 1 with a 3 × 3 kernel
- This means we will add one row and one column of 0 inputs at the top, bottom, left and right
What if we want the output to be of same size as the input?
We can use something known as padding
Pad the inputs with appropriate number of 0 inputs so that you can now apply the kernel at the corners
Let us use pad $P = 1$ with a $3 \times 3$ kernel
This means we will add one row and one column of 0 inputs at the top, bottom, left and right

\[ W_2 = W_1 - F + 2P + 1 \]
\[ H_2 = H_1 - F + 2P + 1 \]
We now have,

\[ W_2 = W_1 - F + 2P + 1 \]

\[ H_2 = H_1 - F + 2P + 1 \]

We will refine this formula further.

- What if we want the output to be of same size as the input?
- We can use something known as padding.
- Pad the inputs with appropriate number of 0 inputs so that you can now apply the kernel at the corners.
- Let us use pad \( P = 1 \) with a \( 3 \times 3 \) kernel.
- This means we will add one row and one column of 0 inputs at the top, bottom, left and right.

\[
\begin{array}{ccccccc} 
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
\]

\[
\begin{array}{ccccccc} 
\bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet & \bullet
\end{array}
\]

\[
\begin{array}{ccccccc} 
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
\]
What if we want the output to be of same size as the input?

- We can use something known as padding
- Pad the inputs with appropriate number of 0 inputs so that you can now apply the kernel at the corners
- Let us use pad $P = 1$ with a $3 \times 3$ kernel
- This means we will add one row and one column of 0 inputs at the top, bottom, left and right

We now have,

$$W_2 = W_1 - F + 2P + 1$$

$$H_2 = H_1 - F + 2P + 1$$

We will refine this formula further.
What does the stride $S$ do?

It defines the intervals at which the filter is applied (here $S = 2$). Here, we are essentially skipping every 2nd pixel which will again result in an output which is of smaller dimensions.
What does the stride $S$ do?

- It defines the intervals at which the filter is applied (here $S = 2$)
What does the stride $S$ do?
- It defines the intervals at which the filter is applied (here $S = 2$)
- Here, we are essentially skipping every 2nd pixel which will again result in an output which is of smaller dimensions
So what should our final formula look like,

\[ W_2 = W_1 - F + 2PS + 1 \]

What does the stride \( S \) do?
- It defines the intervals at which the filter is applied (here \( S = 2 \))
- Here, we are essentially skipping every 2nd pixel which will again result in an output which is of smaller dimensions

- What does the stride \( S \) do?
- It defines the intervals at which the filter is applied (here \( S = 2 \))
- Here, we are essentially skipping every 2nd pixel which will again result in an output which is of smaller dimensions
What does the stride $S$ do?
- It defines the intervals at which the filter is applied (here $S = 2$)
- Here, we are essentially skipping every 2nd pixel which will again result in an output which is of smaller dimensions
What does the stride $S$ do?

- It defines the intervals at which the filter is applied (here $S = 2$)
- Here, we are essentially skipping every 2nd pixel which will again result in an output which is of smaller dimensions
What does the stride $S$ do?
It defines the intervals at which the filter is applied (here $S = 2$).
Here, we are essentially skipping every 2nd pixel which will again result in an output which is of smaller dimensions.

What does the stride $S$ do?
It defines the intervals at which the filter is applied (here $S = 2$).
Here, we are essentially skipping every 2nd pixel which will again result in an output which is of smaller dimensions.
What does the stride $S$ do?

- It defines the intervals at which the filter is applied (here $S = 2$)
- Here, we are essentially skipping every 2nd pixel which will again result in an output which is of smaller dimensions

![Diagram showing the effect of stride on a filter application]

What does the stride $S$ do?

- It defines the intervals at which the filter is applied (here $S = 2$)
- Here, we are essentially skipping every 2nd pixel which will again result in an output which is of smaller dimensions

![Diagram showing the effect of stride on a filter application]
So what should our final formula look like,

\[ W_2 = W_1 - F + 2P \]

\[ H_2 = H_1 - F + 2P \]

- What does the stride \( S \) do?
- It defines the intervals at which the filter is applied (here \( S = 2 \))
- Here, we are essentially skipping every 2nd pixel which will again result in an output which is of smaller dimensions

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

= 

\[
\begin{array}{cccc}
\cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \\
\end{array}
\]
What does the stride $S$ do?

- It defines the intervals at which the filter is applied (here $S = 2$)
- Here, we are essentially skipping every 2nd pixel which will again result in an output which is of smaller dimensions
So what should our final formula look like,

\[ W_2 = W_1 - F + 2P + 1 \]

\[ H_2 = H_1 - F + 2P + 1 \]

- What does the stride \( S \) do?
- It defines the intervals at which the filter is applied (here \( S = 2 \))
- Here, we are essentially skipping every 2nd pixel which will again result in an output which is of smaller dimensions
What does the stride $S$ do?

- It defines the intervals at which the filter is applied (here $S = 2$)
- Here, we are essentially skipping every 2nd pixel which will again result in an output which is of smaller dimensions
What does the stride $S$ do?
- It defines the intervals at which the filter is applied (here $S = 2$)
- Here, we are essentially skipping every 2nd pixel which will again result in an output which is of smaller dimensions
What does the stride $S$ do?

- It defines the intervals at which the filter is applied (here $S = 2$)
- Here, we are essentially skipping every 2nd pixel which will again result in an output which is of smaller dimensions
What does the stride $S$ do?
- It defines the intervals at which the filter is applied (here $S = 2$)
- Here, we are essentially skipping every 2nd pixel which will again result in an output which is of smaller dimensions
So what should our final formula look like,

\[ W_2 = W_1 - F + 2P + 1 \]

\[ H_2 = H_1 - F + 2P + 1 \]

*What does the stride S do?*

*It defines the intervals at which the filter is applied (here \( S = 2 \)).*

*Here, we are essentially skipping every 2nd pixel which will again result in an output which is of smaller dimensions.*
So what should our final formula look like,

\[ W_2 = W_1 - F + 2P + 1 \]

\[ H_2 = H_1 - F + 2P + 1 \]

- What does the stride \( S \) do?
- It defines the intervals at which the filter is applied (here \( S = 2 \))
- Here, we are essentially skipping every 2nd pixel which will again result in an output which is of smaller dimensions
What does the stride $S$ do?
- It defines the intervals at which the filter is applied (here $S = 2$)
- Here, we are essentially skipping every 2nd pixel which will again result in an output which is of smaller dimensions

So what should our final formula look like,

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

\[
H_2 = \frac{H_1 - F + 2P}{S} + 1
\]
Finally, coming to the depth of the output.
Finally, coming to the depth of the output.

Each filter gives us one 2D output.
Finally, coming to the depth of the output.

- Each filter gives us one 2D output.
- \( K \) filters will give us \( K \) such 2D outputs.

\[
\begin{align*}
W_2 &= \frac{W_1 - F + 2P}{S} + 1 \\
H_2 &= \frac{H_1 - F + 2P}{S} + 1 \\
D_2 &= K
\end{align*}
\]
Finally, coming to the depth of the output.
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 depth of the output.

- 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 $D_2 = K$

\[
W_2 = \frac{W_1 - F + 2P}{S} + 1
\]
\[
H_2 = \frac{H_1 - F + 2P}{S} + 1
\]
\[
D_2 = K
\]
Let us do a few exercises

96 filters

Stride = 4

Padding = 0

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

\[ H_2 = \frac{H_1 - F + 2P}{S} + 1 \]
Let us do a few exercises

96 filters

Stride = 4
Padding = 0

\[ W_2 = \frac{W_1 - F + 2P}{S} + 1 \]
\[ H_2 = \frac{H_1 - F + 2P}{S} + 1 \]
Let us do a few exercises

96 filters

* Stride = 4
  Padding = 0

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

\[ 55 = \frac{227 - 11}{4} + 1 \]

\[ W_2 = ? \]
Let us do a few exercises

96 filters

Stride = 4
Padding = 0

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

\[ 55 = \frac{227 - 11}{4} + 1 \]
Let us do a few exercises

- 6 filters
- Stride = 1
- Padding = 0
- \( W_2 = \frac{W_1 - F + 2P}{S} + 1 \)
- \( H_2 = \frac{H_1 - F + 2P}{S} + 1 \)

- \( H_2 = {?} \)
- \( W_2 = {?} \)
- \( D_2 = {?} \)
- \( W_1 = 32 \)
- \( H_1 = 32 \)
- \( F = 5 \)
- \( P = 1 \)
- \( S = 1 \)

Mitesh M. Khapra
CS7015 (Deep Learning) : Lecture 11
Let us do a few exercises

6 filters

- **Stride** = 1
- **Padding** = 0

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

\[ H_2 = \frac{H_1 - F + 2P}{S} + 1 \]
Let us do a few exercises

6 filters

Stride = 1
Padding = 0

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

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

\[
28 = \frac{32 - 5}{1} + 1
\]
Let us do a few exercises

6 filters

\[
\begin{align*}
\text{Stride} &= 1 \\
\text{Padding} &= 0 \\
W_2 &= \frac{W_1 - F + 2P}{S} + 1 \\
H_2 &= \frac{H_1 - F + 2P}{S} + 1
\end{align*}
\]
Module 11.3 : Convolutional Neural Networks
Putting things into perspective

- What is the connection between this operation (convolution) and neural networks?
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 \[\rightarrow\] car, bus, monument, flower
Features

Raw pixels

→

car, bus, monument, flower

Mitesh M. Khapra
CS7015 (Deep Learning) : Lecture 11
Features

Raw pixels → car, bus, monument, flower

Edge Detector

Features

- **Raw pixels**
- **Edge Detector**

- car, bus, monument, flower

Mitesh M. Khapra  
CS7015 (Deep Learning) : Lecture 11
Features

Raw pixels → car, bus, monument, flower

Edge Detector → car, bus, monument, flower
Features

Raw pixels

\[\text{car, bus, monument, flower}\]

Edge Detector

\[\text{car, bus, monument, flower}\]

SIFT/HOG

\[\text{car, bus, monument, flower}\]
Features

Raw pixels

Edge Detector

SIFT/HOG

→ car, bus, monument, flower

→ car, bus, monument, flower

→ car, bus, monument, flower
Features

Raw pixels

Edge Detector

SIFT/HOG

→ car, bus, monument, flower

→ car, bus, monument, flower

→ car, bus, monument, flower

static feature extraction (no learning)

learning weights of 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?**
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?
Even better: Instead of using handcrafted kernels (such as edge detectors) \textbf{can we learn multiple meaningful kernels/filters in addition to learning the weights of the classifier?}
Even better: Instead of using handcrafted kernels (such as edge detectors) can we learn multiple meaningful kernels/filters in addition to learning the weights of the classifier?
- **Even better:** Instead of using handcrafted kernels (such as edge detectors) can we learn multiple meaningful kernels/filters in addition to learning the weights of the classifier?
Can we learn multiple **layers** of 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 backpropagation). Such a network is called a Convolutional Neural Network.
Can we learn multiple layers of meaningful kernels/filters in addition to learning the weights of the classifier?

Yes, we can!
Can we learn multiple layers of 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 layers of 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.
This is what a regular feed-forward neural network will look like. There are many dense connections here. For example, all the 16 input neurons are contributing to the computation of $h^{11}$. Contrast this to what happens in the case of convolution.
This is what a regular feed-forward neural network will look like. There are many dense connections here. For example all the 16 input neurons are contributing to the computation of $h_1$. Contrast this to what happens in the case of convolution.
This is what a regular feed-forward neural network will look like. There are many dense connections here. For example, all the 16 input neurons are contributing to the computation of $h_{11}$. Contrast this to what happens in the case of convolution.
This is what a regular feed-forward neural network will look like. There are many dense connections here. For example, all the 16 input neurons are contributing to the computation of \( h_1 \).

Contrast this to what happens in the case of convolution.
This is what a regular feed-forward neural network will look like. There are many dense connections here. For example, all the 16 input neurons are contributing to the computation of $h_1$. Contrast this to what happens in the case of convolution.
This is what a regular feed-forward neural network will look like. There are many dense connections here. For example, all the 16 input neurons are contributing to the computation of $h_{11}$. Contrast this to what happens in the case of convolution.
This is what a regular feed-forward neural network will look like. There are many dense connections here. For example all the 16 input neurons are contributing to the computation of $h_{11}$. Contrast this to what happens in the case of convolution.
This is what a regular feed-forward neural network will look like. There are many dense connections here. For example, all the 16 input neurons are contributing to the computation of $h_{11}$. Contrast this to what happens in the case of convolution.
This is what a regular feed-forward neural network will look like. There are many dense connections here. For example, all the 16 input neurons are contributing to the computation of $h_{11}$. Contrast this to what happens in the case of convolution.
This is what a regular feed-forward neural network will look like
This is what a regular feed-forward neural network will look like.

There are many dense connections here.
This is what a regular feed-forward neural network will look like.

There are many dense connections here.

For example all the 16 input neurons are contributing to the computation of $h_{11}$
This is what a regular feed-forward neural network will look like.

There are many dense connections here.

For example all the 16 input neurons are contributing to the computation of $h_{11}$.

Contrast this to what happens in the case of convolution.
Only a few local neurons participate in the computation of $h_{11}$. For example, only pixels 1, 2, 5, 6 contribute to $h_{11}$. The connections are much sparser. We are taking advantage of the structure of the image (interactions between neighboring pixels are more interesting). This sparse connectivity reduces the number of parameters in the model.
Only a few local neurons participate in the computation of $h_{11}$.
- Only a few local neurons participate in the computation of $h_{11}$
- For example, only pixels 1, 2, 5, 6 contribute to $h_{11}$
Only a few local neurons participate in the computation of $h_{11}$

- For example, only pixels 1, 2, 5, 6 contribute to $h_{11}$
Only a few local neurons participate in the computation of $h_{11}$.

For example, only pixels 1, 2, 5, 6 contribute to $h_{11}$. 

The connections are much sparser. We are taking advantage of the structure of the image (interactions between neighboring pixels are more interesting).

This sparse connectivity reduces the number of parameters in the model.
• Only a few local neurons participate in the computation of $h_{11}$

• For example, only pixels 1, 2, 5, 6 contribute to $h_{11}$
Only a few local neurons participate in the computation of $h_{11}$

For example, only pixels 1, 2, 5, 6 contribute to $h_{11}$

The connections are much sparser
Only a few local neurons participate in the computation of $h_{11}$

For example, only pixels 1, 2, 5, 6 contribute to $h_{11}$

The connections are much sparser

We are taking advantage of the structure of the image (interactions between neighboring pixels are more interesting)
Only a few local neurons participate in the computation of $h_{11}$

- For example, only pixels 1, 2, 5, 6 contribute to $h_{11}$

- The connections are much sparser

- We are taking advantage of the structure of the image (interactions between neighboring pixels are more interesting)

- This **sparse connectivity** reduces the number of parameters in the model
But is sparse connectivity really good thing?

* Goodfellow-et-al-2016
• But is sparse connectivity really good thing?

• Aren’t we losing information (by losing interactions between some input pixels)

* Goodfellow-et-al-2016
But is sparse connectivity really good thing?

Aren’t we losing information (by losing interactions between some input pixels)

Well, not really

* Goodfellow-et-al-2016
But is sparse connectivity really good thing?

Aren’t we losing information (by losing interactions between some input pixels)

Well, not really

The two highlighted neurons ($x_1$ & $x_5$) do not interact in layer 1

* Goodfellow-et-al-2016
But is sparse connectivity really good thing?

Aren’t we losing information (by losing interactions between some input pixels)

Well, not really

The two highlighted neurons ($x_1$ & $x_5$) do not interact in layer 1

But they indirectly contribute to the computation of $g_3$ and hence interact indirectly

* Goodfellow-et-al-2016
Another characteristic of CNNs is **weight sharing**.

Consider the following network:

Do we want the kernel weights to be different for different portions of the image?

Imagine that we are trying to learn a kernel that detects edges. Shouldn’t we be applying the same kernel at all the portions of the image?
Another characteristic of CNNs is **weight sharing**.

Consider the following network:

- Do we want the kernel weights to be different for different portions of the image?
- Imagine that we are trying to learn a kernel that detects edges. Shouldn't we be applying the same kernel at all the portions of the image?
Another characteristic of CNNs is **weight sharing**.

Consider the following network:

Do we want the kernel weights to be different for different portions of the image? Imagine that we are trying to learn a kernel that detects edges. Shouldn’t we be applying the same kernel at all the portions of the image?

- Kernel 1
- Kernel 2

4x4 Image
Another characteristic of CNNs is **weight sharing**

Consider the following network

Do we want the kernel weights to be different for different portions of the image?

- Kernel 1
- Kernel 2

4x4 Image
Another characteristic of CNNs is **weight sharing**

Consider the following network

Do we want the kernel weights to be different for different portions of the image?

Imagine that we are trying to learn a kernel that detects edges
Another characteristic of CNNs is **weight sharing**.

Consider the following network.

Do we want the kernel weights to be different for different portions of the image?

Imagine that we are trying to learn a kernel that detects edges.

Shouldn’t we be applying the same kernel at all the portions of the image?
In other words shouldn’t the \textit{orange} and \textit{pink} kernels be the same

This would make the job of learning easier (instead of trying to learn the same weights/kernels at different locations again and again).

But does that mean we can have only one kernel?

No, we can have many such kernels but the kernels will be shared by all locations in the image.

This is called “weight sharing”.

\begin{center}
\begin{tikzpicture}
\node[draw] (input) at (0,0) [rectangle, rounded corners] {\textcolor{orange}{\rlap{\hspace{2.5cm}}16}};
\node[draw] (conv) at (1,1) [rectangle, rounded corners, fill=blue!20] {\textcolor{blue}{\rlap{\hspace{2.5cm}}16}};
\node[draw] (pool) at (2,2) [rectangle, rounded corners, fill=red!20] {\textcolor{red}{\rlap{\hspace{2.5cm}}16}};
\end{tikzpicture}
\end{center}
In other words shouldn’t the *orange* and *pink* kernels be the same

Yes, indeed
• In other words shouldn’t the *orange* and *pink* kernels be the same

• Yes, indeed
In other words shouldn’t the *orange* and *pink* kernels be the same

- Yes, indeed

- This would make the job of learning easier (instead of trying to learn the same weights/kernels at different locations again and again)

But does that mean we can have only one kernel?

- No, we can have many such kernels but the kernels will be shared by all locations in the image

This is called “weight sharing”
In other words shouldn’t the *orange* and *pink* kernels be the same

Yes, indeed

This would make the job of learning easier (instead of trying to learn the same weights/kernels at different locations again and again)

But does that mean we can have only one kernel?
In other words, shouldn’t the *orange* and *pink* kernels be the same?

Yes, indeed.

This would make the job of learning easier (instead of trying to learn the same weights/kernels at different locations again and again).

But does that mean we can have only one kernel?

No, we can have many such kernels but the kernels will be shared by all locations in the image.
In other words shouldn’t the *orange* and *pink* kernels be the same

- Yes, indeed

- This would make the job of learning easier (instead of trying to learn the same weights/kernels at different locations again and again)

- But does that mean we can have only one kernel?

- No, we can have many such kernels but the kernels will be shared by all locations in the image
In other words shouldn’t the *orange* and *pink* kernels be the same

Yes, indeed

This would make the job of learning easier (instead of trying to learn the same weights/kernels at different locations again and again)

But does that mean we can have only one kernel?

No, we can have many such kernels but the kernels will be shared by all locations in the image
In other words shouldn’t the *orange* and *pink* kernels be the same

- Yes, indeed

- This would make the job of learning easier (instead of trying to learn the same weights/kernels at different locations again and again)

- But does that mean we can have only one kernel?

- No, we can have many such kernels but the kernels will be shared by all locations in the image

- This is called “weight sharing”
So far, we have focused only on the convolution operation.
So far, we have focused only on the convolution operation
Let us see what a full convolutional neural network looks like
It has alternate convolution and pooling layers.

What does a pooling layer do?

Let us see...
It has alternate convolution and pooling layers
- It has alternate convolution and pooling layers
- What does a pooling layer do?
- It has alternate convolution and pooling layers
- What does a pooling layer do?
- Let us see
Input

Instead of max pooling we can also do average pooling.
Instead of max pooling we can also do average pooling
Instead of max pooling we can also do average pooling.

Mitesh M. Khapra
CS7015 (Deep Learning) : Lecture 11
Instead of max pooling we can also do average pooling.
Instead of max pooling we can also do average pooling.
Instead of max pooling we can also do average pooling.
Instead of max pooling we can also do average pooling.
Instead of max pooling we can also do average pooling.
Instead of max pooling we can also do average pooling
Instead of max pooling we can also do average pooling.
Instead of max pooling we can also do average pooling.

Input

1 filter

maxpool 2x2 filters (stride 2)

\[
\begin{array}{cccc}
1 & 4 & 2 & 1 \\
5 & 8 & 3 & 4 \\
7 & 6 & 4 & 5 \\
1 & 3 & 1 & 2 \\
\end{array}
\]

\[
\begin{array}{cc}
8 & 4 \\
7 & 5 \\
\end{array}
\]
Instead of max pooling we can also do average pooling.
Instead of max pooling we can also do average pooling.
Instead of max pooling we can also do average pooling.
Instead of max pooling we can also do average pooling.
Instead of max pooling we can also do average pooling.
Instead of max pooling we can also do average pooling
Instead of max pooling we can also do average pooling.
Instead of max pooling we can also do average pooling.
Instead of max pooling we can also do average pooling.
Instead of max pooling we can also do average pooling.
Instead of max pooling we can also do average pooling.

Mitesh M. Khapra
CS7015 (Deep Learning) : Lecture 11
Instead of max pooling we can also do average pooling
We will now see some case studies where convolution neural networks have been successful.
LeNet-5 for handwritten character recognition
LeNet-5 for handwritten character recognition

Input

Convolution Layer 1

Pooling Layer 1

Convolution Layer 2

Pooling Layer 2

FC 1 (120)

FC 2 (84)

Output (10)

$S = 1, F = 5,$
$K = 6, P = 0,$
$Param =$?
LeNet-5 for handwritten character recognition

$S = 1, F = 5,$
$K = 6, P = 0,$
$Param = 150$
LeNet-5 for handwritten character recognition

Input

Convolution Layer 1

Pooling Layer 1

Convolution Layer 2

Pooling Layer 2

FC 1(120)

FC 2(84)

Output(10)

$S = 1, F = 5,$
$K = 6, P = 0,$
$Param = 150$

$S = 1, F = 2,$
$K = 6, P = 0,$
$Param = ?$
LeNet-5 for handwritten character recognition

Input

Convolution Layer 1

Pooling Layer 1

Convolution Layer 2

Pooling Layer 2

Fully Connected 1 (120)

Fully Connected 2 (84)

Output (10)

$S = 1, F = 5,$
$K = 6, P = 0,$
$Param = 150$

$S = 1, F = 2,$
$K = 6, P = 0,$
$Param = 0$
LeNet-5 for handwritten character recognition

Input

Convolution Layer 1

Pooling Layer

Convolution Layer 2

FC 1(120)

FC 2(84)

Output(10)

$S = 1, F = 5,$

$K = 6, P = 0,$

$Param = 150$

$S = 1, F = 2,$

$K = 6, P = 0,$

$Param = 0$

$S = 1, F = 5,$

$K = 16, P = 0,$

$Param = ?$
LeNet-5 for handwritten character recognition

Input

Convolution Layer 1

Pooling Layer

Convolution Layer 2

FC 1(120)

FC 2(84)

Output(10)

\[ S = 1, F = 5, \]
\[ K = 6, P = 0, \]
\[ Param = 150 \]

\[ S = 1, F = 2, \]
\[ K = 6, P = 0, \]
\[ Param = 0 \]

\[ S = 1, F = 5, \]
\[ K = 16, P = 0, \]
\[ Param = 2400 \]
LeNet-5 for handwritten character recognition

Input

Convolution Layer 1

Pooling Layer

Convolution Layer 2

Pooling Layer 2

FC 1(120)

FC 2(84)

Output(10)

\[ S = 1, F = 5, \]
\[ K = 6, P = 0, \]
\[ Param = 150 \]

\[ S = 1, F = 2, \]
\[ K = 6, P = 0, \]
\[ Param = 0 \]

\[ S = 1, F = 5, \]
\[ K = 6, P = 0, \]
\[ Param = 0 \]

\[ S = 1, F = 2, \]
\[ K = 16, P = 0, \]
\[ K = 16, P = 0, \]
\[ Param = 2400 \]
\[ Param = ? \]

Mitesh M. Khapra
CS7015 (Deep Learning) : Lecture 11
LeNet-5 for handwritten character recognition

- **Input**: 28x28

  - **Convolution Layer 1**
    - Spatial: $S = 1, F = 5$
    - Kernel: $K = 6, P = 0$
    - Parameters: $Param = 150$

  - **Pooling Layer 1**
    - Spatial: $S = 1, F = 2$
    - Kernel: $K = 6, P = 0$
    - Parameters: $Param = 0$

  - **Convolution Layer 2**
    - Spatial: $S = 1, F = 5$
    - Kernel: $K = 16, P = 0$
    - Parameters: $Param = 2400$

  - **Pooling Layer 2**
    - Spatial: $S = 1, F = 2$
    - Kernel: $K = 16, P = 0$
    - Parameters: $Param = 0$

- **Fully Connected Layers**
  - FC 1(120)
  - FC 2(84)
  - Output(10)
LeNet-5 for handwritten character recognition

Input:

Convolution Layer 1:
- \( S = 1, F = 5 \)
- \( K = 6, P = 0 \)
- \( Param = 150 \)

Pooling Layer 1:
- \( S = 1, F = 2 \)
- \( K = 6, P = 0 \)
- \( Param = 0 \)

Convolution Layer 2:
- \( S = 1, F = 5 \)
- \( K = 16, P = 0 \)
- \( Param = 2400 \)

Pooling Layer 2:
- \( S = 1, F = 2 \)
- \( K = 16, P = 0 \)
- \( Param = 0 \)

FC 1(120):
- \( Param = ? \)

FC 2(84):

Output(10):
LeNet-5 for handwritten character recognition

Input

Convolution Layer 1

S = 1, \( F = 5 \), \( K = 6, P = 0 \),
Param = 150

28

Poolig Layer 1

S = 1, \( F = 2 \), \( K = 6, P = 0 \),
Param = 0

14

Convolution Layer 2

S = 1, \( F = 5 \), \( K = 16, P = 0 \),
Param = 2400

10

Pooling Layer 2

S = 1, \( F = 2 \), \( K = 16, P = 0 \),
Param = 0

5

FC 1(120)

Param = 48120

FC 2(84)

Output(10)
LeNet-5 for handwritten character recognition

Input

Convolution Layer 1

Pooling Layer

Convolution Layer 2

Pooling Layer 2

FC 1 (120)

FC 2 (84)

Output (10)

$S = 1, F = 5, \quad K = 6, P = 0, \quad Param = 150$

$S = 1, F = 2, \quad K = 6, P = 0, \quad Param = 0$

$S = 1, F = 5, \quad S = 1, F = 2, \quad S = 1, F = 2,$

$K = 16, P = 0, K = 16, P = 0,$

$Param = 2400 \quad Param = 0$
LeNet-5 for handwritten character recognition

Input

Convolution Layer 1

Pooling Layer 1

Convolution Layer 2

Pooling Layer 2

FC 1 (120)

FC 2 (84)

Output (10)

$S = 1, F = 5, K = 6, P = 0, \text{Param} = 150$

$S = 1, F = 2, K = 6, P = 0, \text{Param} = 0$

$S = 1, F = 5, S = 1, F = 2, K = 16, P = 0, K = 16, P = 0, \text{Param} = 2400 \text{ Param} = 0$

$\text{Param} = 48120 = 10164$

Mitesh M. Khapra

CS7015 (Deep Learning) : Lecture 11
LeNet-5 for handwritten character recognition

Input

Convolution Layer 1

Pooling Layer

Convolution Layer 2

Pooling Layer 2

FC 1(120)

FC 2(84)

Output(10)

$S = 1, F = 5, K = 6, P = 0, Param = 150$

$S = 1, F = 2, K = 6, P = 0, Param = 0$

$S = 1, F = 5, S = 1, F = 2, K = 6, P = 0, K = 16, P = 0, Param = 2400\quad Param = 0$

$Param = 48120 = 10164$
LeNet-5 for handwritten character recognition

Input

Convolution Layer 1

Pooling Layer

Convolution Layer 2

Pooling Layer 2

FC 1(120)

FC 2(84)

Output(10)

\[ S = 1, F = 5, \]
\[ K = 6, P = 0, \]
\[ Param = 150 \]

\[ S = 1, F = 2, \]
\[ K = 6, P = 0, \]
\[ Param = 0 \]

\[ S = 1, F = 5, \]
\[ K = 6, P = 0, \]
\[ Param = 0 \]

\[ S = 1, F = 2, \]
\[ K = 16, P = 0, K = 16, P = 0, \]
\[ Param = 2400 \]

\[ Param = 0 \]
How do we train a convolutional neural network?
A CNN can be implemented as a feedforward neural network.
We can thus train a convolution neural network using backpropagation by thinking of it as a feedforward neural network with sparse connections.

A CNN can be implemented as a feedforward neural network wherein only a few weights (in color) are active.

wherein only a few weights (in color) are active.
A CNN can be implemented as a feedforward neural network wherein only a few weights (in color) are active. The rest of the weights (in gray) are zero.
A CNN can be implemented as a feedforward neural network wherein only a few weights (in color) are active and the rest of the weights (in gray) are zero.
A CNN can be implemented as a feedforward neural network wherein only a few weights (in color) are active, and the rest of the weights (in gray) are zero.
A CNN can be implemented as a feedforward neural network
wherein only a few weights(in color) are active
the rest of the weights (in gray) are zero
We can thus train a convolution neural network using backpropagation by thinking of it as a feedforward neural network with sparse connections.

- A CNN can be implemented as a feedforward neural network
- wherein only a few weights (in color) are active
- the rest of the weights (in gray) are zero
Module 11.4 : CNNs (success stories on ImageNet)
ImageNet Success Stories (roadmap for rest of the talk)

- AlexNet
ImageNet Success Stories (roadmap for rest of the talk)

- AlexNet
- ZFNet
ImageNet Success Stories (roadmap for rest of the talk)

- AlexNet
- ZFNet
- VGGNet
ILSVRC'10: 28.2
ILSVRC'10: 28.2
ILSVRC'11: 25.8
ILSVRC'10: 28.2
ILSVRC'11: 25.8
ILSVRC'12: 16.4

ILSVRC'13: ZFNet 11.7
ILSVRC'14: VGG 7.3
ILSVRC'14: GoogleNet 6.7
ILSVRC'15: ResNet 3.57

AlexNet: 16.4 layers
VGG: 19 layers
GoogleNet: 22 layers
ResNet: 152 layers
ILSVRC’10: 28.2
ILSVRC’11: 25.8
ILSVRC’12: 16.4
ILSVRC’13: 11.7

AlexNet: 16.4 layers
ZFNet: 8 layers
VGG: 19 layers
GoogleNet: 22 layers
ResNet: 152 layers
ILSVRC’10: 28.2
ILSVRC’11: 25.8
ILSVRC’12: 16.4
ILSVRC’13: 11.7
ILSVRC’14: 7.3

Mitesh M. Khapra  CS7015 (Deep Learning) : Lecture 11
ILSVRC’10 28.2
ILSVRC’11 25.8
ILSVRC’12 16.4  AlexNet
ILSVRC’13 11.7  ZFNet
ILSVRC’14  7.3  VGG
ILSVRC’14  6.7  GoogleNet
ILSVRC’15  3.57  ResNet

shallow 8 layers
19 layers
22 layers
152 layers
ImageNet Success Stories (roadmap for rest of the talk)

- AlexNet
- ZFNet
- VGGNet
Input 227

Convolution 3
MaxPooling 55

Parameters: (2

Conv1: Input: 227
F \times S \times P \times M = 1, K^2 = 256,
F \times 3 \times 3 \times 3 = 0

Parameters: 4096
Output: 256

Max Pool Input: 23
Total Parameters: 27

Conv2: Input: 27
F \times 3 \times 3 \times 3 = 0

Parameters: 3
Output: 384

Max Pool Input: 55

Conv3: Input: 7
F \times 3 \times 3 \times 3 = 0

Parameters: 11
Output: 256

Max Pool Input: 5

Parameters: 0
Output: 256

Parameters: ?
Output: 32

Parameters: ?
Output: 384

Parameters: ?
Output: 96

Parameters: ?
Output: 34

Parameters: ?
Output: 327
Input: $227 \times 227 \times 3$

Conv1: $K = 96, F = 11$
$S = 4, P = 0$

Output: $W_2 = ?, H_2 = ?$

Parameters: ?

Input: $227 \times 227 \times 3$

Conv1: $K = 96, F = 11$
$S = 4, P = 0$

Output: $W_2 = 55, H_2 = 55$

Parameters: $(11 \times 11 \times 3) \times 96 = 34,556$
Input: $227 \times 227 \times 3$
Conv1: $K = 96, F = 11$
$S = 4, P = 0$
Output: $W_2 = 55, H_2 = 55$
Parameters: ?

Convolution

Max Pooling
Input: $55 \times 55 \times 96$
Conv1: $K = 3, F = 2$
Output: $W_2 = 27, H_2 = 27$
Parameters: ?

Max Pooling
Input: $27 \times 27 \times 96$
Conv1: $K = 256, F = 5$
$S = 1, P = 0$
Output: $W_2 = 23, H_2 = 23$
Parameters: ?

Max Pooling
Input: $11 \times 11 \times 256$
Conv1: $K = 384, F = 3$
$S = 1, P = 0$
Output: $W_2 = 9, H_2 = 9$
Parameters: ?

Max Pooling
Input: $3 \times 3 \times 256$
Conv1: $K = 384, F = 3$
$S = 1, P = 0$
Output: $W_2 = 7, H_2 = 7$
Parameters: ?

FC1
Parameters: $(3 \times 3 \times 384) \times 384 = 0.327$M

FC1
Parameters: $(3 \times 3 \times 384) \times 256 = 0.8$M

Total Parameters: 27.55M
Input: 227 × 227 × 3
Conv1: \( K = 96, F = 11 \)
\( S = 4, P = 0 \)
Output: \( W_2 = 55, H_2 = 55 \)
Parameters: \( (11 \times 11 \times 3) \times 96 = 34K \)
Input: $227 \times 227 \times 3$

Conv1:
- $K = 96$, $F = 11$, $S = 4$, $P = 0$

Output:
- $W_2 = ?$, $H_2 = ?$

Parameters:
- $(11 \times 11 \times 3) \times 96 = 34,992$

Max Pool
- Input: $55 \times 55 \times 96$
- $F = 3, S = 2$
- Output: $W_2 = ?, H_2 = ?$
- Parameters: $?$

Convolution
- Input: $27 \times 27 \times 96$
- $K = 256$, $F = 5$, $S = 1$, $P = 0$
- Output: $W_2 = ?, H_2 = ?$
- Parameters: $?$

Max Pool
- Input: $23 \times 23 \times 256$
- $F = 3, S = 2$
- Output: $W_2 = ?, H_2 = ?$
- Parameters: $?$

Convolution
- Input: $11 \times 11 \times 256$
- $K = 384$, $F = 3$, $S = 1$, $P = 0$
- Output: $W_2 = ?, H_2 = ?$
- Parameters: $?$

Max Pool
- Input: $9 \times 9 \times 384$
- $F = 3, S = 1$
- Output: $W_2 = ?, H_2 = ?$
- Parameters: $?$

Convolution
- Input: $7 \times 7 \times 384$
- $K = 384$, $F = 3$, $S = 1$, $P = 0$
- Output: $W_2 = ?, H_2 = ?$
- Parameters: $?$

Max Pool
- Input: $5 \times 5 \times 256$
- $F = 3, S = 2$
- Output: $W_2 = ?, H_2 = ?$
- Parameters: $?$

FC1
- Parameters: $(2 \times 2 \times 256) \times 4096 = 4096$
- Dense
- Parameters: $4096 \times 4096 = 16,777,216$
- Total Parameters: 27,550
Input: $227 \times 227 \times 3$

**Conv1:**
- $K = 96$, $F = 11$
- $S = 4$, $P = 0$

**Output:**
- $W_2 = ?$, $H_2 = ?$
- $W_2 = 55$, $H_2 = 55$

**Parameters:**
- $(11 \times 11 \times 3) \times 96 = 34,992$

MaxPool Input: $55 \times 55 \times 96$
- $F = 3, S = 2$
- Output: $W_2 = 27$, $H_2 = 27$
- Parameters: ?

**MaxPooling**
- $27 \times 27 \times 96$

Conv1:
- $K = 256$, $F = 5$
- $S = 1$, $P = 0$

**Output:**
- $W_2 = ?$, $H_2 = ?$
- $W_2 = 23$, $H_2 = 23$

**Parameters:**
- $(5 \times 5 \times 96) \times 256 = 0$

MaxPool Input: $23 \times 23 \times 256$
- $F = 3, S = 2$
- Output: $W_2 = 11$, $H_2 = 11$
- Parameters: ?

**MaxPooling**
- $11 \times 11 \times 256$

Conv1:
- $K = 384$, $F = 3$
- $S = 1$, $P = 0$

**Output:**
- $W_2 = ?$, $H_2 = ?$
- $W_2 = 9$, $H_2 = 9$

**Parameters:**
- $(3 \times 3 \times 256) \times 384 = 0$

MaxPool Input: $9 \times 9 \times 384$
- $F = 3, S = 2$
- Output: $W_2 = 7$, $H_2 = 7$
- Parameters: ?

**MaxPooling**
- $7 \times 7 \times 384$

Conv1:
- $K = 256$, $F = 3$
- $S = 1$, $P = 0$

**Output:**
- $W_2 = ?$, $H_2 = ?$
- $W_2 = 5$, $H_2 = 5$

**Parameters:**
- $(3 \times 3 \times 384) \times 256 = 0$

MaxPool Input: $5 \times 5 \times 256$
- $F = 3, S = 2$
- Output: $W_2 = 2$, $H_2 = 2$
- Parameters: ?

**MaxPooling**
- $2 \times 2 \times 256$

**FC1**
- $2 \times 2 \times 256 \times 4096 = 4,096$

**Parameters:**
- $4096 \times 4096 = 16,384$

**Total Parameters:** 27,552
Input: \(227 \times 227 \times 3\)

**Conv1:**
- \(K = 96, F = 11\), \(S = 4, P = 0\)

Output: \(W_2 = ?, H_2 = ?\)

Parameters: \((11 \times 11 \times 3) \times 96 = 34,592\)

Max Pooling
- Input: \(55 \times 55 \times 96\)
- \(F = 3, S = 2\)
- Output: \(W_2 = ?, H_2 = ?\)

Parameters: \(0\)

Input: \(27 \times 27 \times 96\)

**Conv1:**
- \(K = 256, F = 5\), \(S = 1, P = 0\)

Output: \(W_2 = ?, H_2 = ?\)

Parameters: \((5 \times 5 \times 96) \times 256 = 0,649,600\)

Max Pooling
- Input: \(23 \times 23 \times 256\)
- \(F = 3, S = 2\)
- Output: \(W_2 = ?, H_2 = ?\)

Parameters: \(0\)

Input: \(11 \times 11 \times 256\)

**Conv1:**
- \(K = 384, F = 3\), \(S = 1, P = 0\)

Output: \(W_2 = ?, H_2 = ?\)

Parameters: \((3 \times 3 \times 256) \times 384 = 0,884,176\)

Max Pooling
- Input: \(7 \times 7 \times 384\)
- \(F = 3, S = 1, P = 0\)
- Output: \(W_2 = ?, H_2 = ?\)

Parameters: \(0\)

Max Pooling
- Input: \(6 \times 6 \times 384\)
- \(F = 3, S = 2\)
- Output: \(W_2 = ?, H_2 = ?\)

Parameters: \(0\)

**FC1**
- Parameters: \((2 \times 2 \times 256) \times 4096 = 4,096,000\)
- Parameters: \(4096 \times 4096 = 16,777,216\)
- Parameters: \(4096 \times 1000 = 4,096,000\)

Total Parameters: \(27,556,000\)
Input: 227 × 227 × 3
Conv1: \( K = 96, F = 11 \)
\( S = 4, P = 0 \)

Output: \( W_2 =?, H_2 =? \)

Parameters: \( (11 \times 11 \times 3) \times 96 = 34 \)

Max Pooling Input: 55 × 55 × 96
Conv1: \( F = 3, S = 2 \)

Output: \( W_2 =?, H_2 =? \)

Parameters: \( 0 \)

Convolution Input: 27 × 27 × 96
Conv1: \( K = 256, F = 5 \)
\( S = 1, P = 0 \)

Output: \( W_2 = 23 \), \( H_2 = 23 \)

Parameters: \( (5 \times 5 \times 96) \times 256 = 0.6 \)

Max Pooling Input: 23 × 23 × 256
Conv1: \( K = 384, F = 3 \)
\( S = 1, P = 0 \)

Output: \( W_2 =?, H_2 =? \)

Parameters: \( 0 \)

Convolution Input: 11 × 11 × 256
Conv1: \( K = 384, F = 3 \)
\( S = 1, P = 0 \)

Output: \( W_2 = 9 \), \( H_2 = 9 \)

Parameters: \( (3 \times 3 \times 256) \times 384 = 0.8 \)

Max Pooling Input: 9 × 9 × 384
Conv1: \( K = 256, F = 3 \)
\( S = 1, P = 0 \)

Output: \( W_2 =?, H_2 =? \)

Parameters: \( 0 \)

Max Pooling Input: 7 × 7 × 256
Conv1: \( K = 256, F = 3 \)
\( S = 1, P = 0 \)

Output: \( W_2 = 5 \), \( H_2 = 5 \)

Parameters: \( (3 \times 3 \times 256) \times 256 = 0.8 \)

Max Pooling Input: 5 × 5 × 256

Parameters: \( 0 \)

FC1 Parameters: \( (2 \times 2 \times 256) \times 4096 = 4 \)

Total Parameters: \( 27.55 \)

Mitesh M. Khapra
Input: $227 \times 227 \times 3$

Conv1: $K = 96, F = 11$
$S = 4, P = 0$

Output: $W_2 = ?, H_2 = ?$

Parameters: (?, ?)

Max Pooling

Input: $55 \times 55 \times 96$

Conv1: $K = 3, F = 2$
$S = 2, P = 0$

Output: $W_2 = ?, H_2 = ?$

Parameters: (0, 0)

Convolution

Input: $27 \times 27 \times 96$

Conv1: $K = 256, F = 5$
$S = 1, P = 0$

Output: $W_2 = 23, H_2 = 23$

Parameters: (0, 0)

Max Pooling

Input: $23 \times 23 \times 256$

Conv1: $K = 256, F = 3$
$S = 1, P = 0$

Output: $W_2 = 11, H_2 = 11$

Parameters: (0, 0)

Max Pooling

Input: $11 \times 11 \times 256$

Conv1: $K = 384, F = 3$
$S = 1, P = 0$

Output: $W_2 = 9, H_2 = 9$

Parameters: (0, 0)

Max Pooling

Input: $9 \times 9 \times 384$

Conv1: $K = 384, F = 3$
$S = 1, P = 0$

Output: $W_2 = 7, H_2 = 7$

Parameters: (0, 0)

Max Pooling

Input: $7 \times 7 \times 384$

Conv1: $K = 256, F = 3$
$S = 1, P = 0$

Output: $W_2 = 5, H_2 = 5$

Parameters: (0, 0)

Max Pooling

Input: $5 \times 5 \times 256$

Conv1: $K = 256, F = 3$
$S = 1, P = 0$

Output: $W_2 = 2, H_2 = 2$

Parameters: (0, 0)

FC1

Parameters: (2, 2, 256) \times 4096 = 4M

FC1

Parameters: 4096 \times 4096 = 16M

Total Parameters: 27.55M

Mitesh M. Khapra
CS7015 (Deep Learning) : Lecture 11
**Convolution**

Parameters: \( (5 \times 5 \times 96) \times 256 = 0.6M \)

Input: \( 27 \times 27 \times 96 \)

Conv1: \( K = 256, F = 5 \)

\( S = 1, P = 0 \)

Output: \( W_2 = 23, H_2 = 23 \)
Input: \(227 \times 227 \times 3\)

**Conv1:**
- \(K = 96, F = 11, S = 4, P = 0\)

**Output:**
- \(W_2 = ?, H_2 = ?\)

**Parameters:**
- \((11 \times 11 \times 3) \times 96 = 34,560\)

**Max Pool Input:** \(227 \times 227 \times 96\)
- \(F = 3, S = 2\)
- **Output:**
  - \(W_2 = ?, H_2 = ?\)
  - **Parameters:** ?

**Conv1:**
- \(K = 256, F = 5, S = 1, P = 0\)

**Output:**
- \(W_2 = 23, H_2 = 23\)

**Parameters:**
- \((5 \times 5 \times 96) \times 256 = 0\)

**Max Pool Input:** \(23 \times 23 \times 256\)
- \(F = 3, S = 2\)
- **Output:**
  - \(W_2 = 11, H_2 = 11\)
  - **Parameters:** ?

**Conv1:**
- \(K = 384, F = 3, S = 1, P = 0\)

**Output:**
- \(W_2 = 9, H_2 = 9\)

**Parameters:**
- \((3 \times 3 \times 256) \times 384 = 1,290,240\)

**Max Pool Input:** \(9 \times 9 \times 384\)
- \(F = 3, S = 2\)
- **Output:**
  - \(W_2 = 5, H_2 = 5\)
  - **Parameters:** ?

**Conv1:**
- \(K = 256, F = 3, S = 1, P = 0\)

**Output:**
- \(W_2 = 5, H_2 = 5\)

**Parameters:**
- \((3 \times 3 \times 256) \times 256 = 0\)

**Max Pool Input:** \(5 \times 5 \times 256\)
- \(F = 3, S = 2\)
- **Output:**
  - \(W_2 = 2, H_2 = 2\)
  - **Parameters:** ?

**FC1**
- \((2 \times 2 \times 256) \times 4096 = 4,096,000\)

**Parameters:**
- \(4,096 \times 4,096 = 16,777,216\)

**FC1**
- \(4,096 \times 1,000 = 4,096,000\)

**Total Parameters:** \(27,550\)
Input: 227 × 227 × 3
Conv1: $K = 96$, $F = 11$, $S = 4$, $P = 0$
Output: \[ W_2 = ?, \ H_2 = ? \]
Parameters: ?

Max Pool: Input: 55 × 55 × 96
$F = 3$, $S = 2$
Output: \[ W_2 = ?, \ H_2 = ? \]
Parameters: ?

Conv1: $K = 256$, $F = 5$, $S = 1$, $P = 0$
Output: \[ W_2 = 23, \ H_2 = 23 \]
Parameters: (5 × 5 × 96) × 256 = 0.6 M

Max Pool: Input: 23 × 23 × 256
$F = 3$, $S = 2$
Output: \[ W_2 = ?, \ H_2 = ? \]
Parameters: ?

Conv1: $K = 384$, $F = 3$, $S = 1$, $P = 0$
Output: \[ W_2 = 9, \ H_2 = 9 \]
Parameters: (3 × 3 × 256) × 384 = 0.8 M

Max Pool: Input: 9 × 9 × 384
$F = 3$, $S = 2$
Output: \[ W_2 = ?, \ H_2 = ? \]
Parameters: ?

Conv1: $K = 256$, $F = 3$, $S = 1$, $P = 0$
Output: \[ W_2 = 5, \ H_2 = 5 \]
Parameters: (3 × 3 × 384) × 256 = 0.8 M

Max Pool: Input: 5 × 5 × 256
$F = 3$, $S = 2$
Output: \[ W_2 = 2, \ H_2 = 2 \]
Parameters: ?

FC1: Parameters: (2 × 2 × 256) × 4096 = 4 M
FC1: Parameters: 4096 × 4096 = 16 M
FC1: Parameters: 4096 × 1000 = 4 M
Total Parameters: 27,552 M
Input: $227 \times 227 \times 3$

**Conv1:**
- $K = 96, F = 11, S = 4, P = 0$

Output:
- $W_2 = ?$, $H_2 = ?$

Parameters: $(11 \times 11 \times 3) \times 96 = 34,996$

**Max Pool** Input: $55 \times 55 \times 96$

F = 3, S = 2

Output:
- $W_2 = ?$, $H_2 = ?$

Parameters: 0

**Conv1:**
- $K = 256, F = 5, S = 1, P = 0$

Output:
- $W_2 = 23$, $H_2 = 23$

Parameters: $(5 \times 5 \times 96) \times 256 = 0,163,840$

**Max Pool** Input: $23 \times 23 \times 256$

F = 3, S = 2

Output:
- $W_2 = 11$, $H_2 = 11$

Parameters: 0

**FC1**
- Parameters: $(2 \times 2 \times 256) \times 4096 = 4,096,000$

**Total Parameters:** 27,552

---

Max Pool Input: $23 \times 23 \times 256$

$F = 3, S = 2$

Output: $W_2 = 11$, $H_2 = 11$

Parameters: 0
Input: $11 \times 11 \times 256$

Conv1: $K = 384, F = 3$

$S = 1, P = 0$

Output: $W_2 = ?, H_2 = ?$

Parameters: ?

Max Pooling

Input: $5 \times 5 \times 256$

Conv1: $K = 256, F = 3$

$S = 1, P = 0$

Output: $W_2 = ?, H_2 = ?$

Parameters: ?

Max Pooling

Input: $7 \times 7 \times 384$

Conv1: $K = 256, F = 3$

$S = 1, P = 0$

Parameters: ?

Max Pooling

Input: $5 \times 5 \times 256$

Conv1: $K = 256, F = 3$

$S = 1, P = 0$

Parameters: ?

Max Pooling

Input: $7 \times 7 \times 384$

Conv1: $K = 256, F = 3$

$S = 1, P = 0$

Parameters: ?

Max Pooling

Parameters: 0

FC1

Parameters: (2 \times 2 \times 256) \times 4096 = 4 \ M

Parameters: 4096 \times 4096 = 16 \ M

Parameters: 4096 \times 1000 = 4 \ M

Total Parameters: 27.55 \ M
Input: 11 × 11 × 256
Conv1: K = 384, F = 3
S = 1, P = 0
Output: W₂ = 9, H₂ = 9
Parameters: ?

Input: 11 × 11 × 256
Conv1: K = 96, F = 11
S = 4, P = 0
Output: W₂ = 55, H₂ = 55
Parameters: (11 × 11 × 3) × 96 = 34

Max Pooling: Input: 55 × 55 × 96
Conv1: K = 3, F = 3
S = 2
Output: W₂ = 27, H₂ = 27
Parameters: 0

Max Pooling: Input: 27 × 27 × 96
Conv1: K = 256, F = 5
S = 1, P = 0
Output: W₂ = 23, H₂ = 23
Parameters: (5 × 5 × 96) × 256 = 0

Max Pooling: Input: 23 × 23 × 256
Conv1: K = 384, F = 3
S = 1, P = 0
Output: W₂ = 9, H₂ = 9
Parameters: (3 × 3 × 256) × 384 = 0

Max Pooling: Input: 9 × 9 × 384
Conv1: K = 256, F = 3
S = 1, P = 0
Output: W₂ = 5, H₂ = 5
Parameters: (3 × 3 × 384) × 256 = 0

Max Pooling: Input: 5 × 5 × 256
Conv1: K = 384, F = 3
S = 1, P = 0
Output: W₂ = 9, H₂ = 9
Parameters: (3 × 3 × 256) × 384 = 0

Max Pooling: Input: 2 × 2 × 256
Conv1: K = 384, F = 3
S = 1, P = 0
Output: W₂ = 9, H₂ = 9
Parameters: (3 × 3 × 256) × 384 = 0

Total Parameters: 27,559

Mitesh M. Khapra  CS7015 (Deep Learning) : Lecture 11
Input: 11 × 11 × 256
Conv1: K = 384, F = 3
S = 1, P = 0
Output: W₂ = 9, H₂ = 9
Parameters: (3 × 3 × 256) × 384 = 0.8 M
Input: $227 \times 227 \times 3$
Conv1: $K = 96, F = 11$
S = 4, P = 0
Output: $W_2 = ?, H_2 = ?$

Parameters: $(11 \times 11 \times 3) \times 96 = 34\,K$

Max Pool Input: $55 \times 55 \times 96$
Conv1: $K = 3, F = 3$
S = 2
Output: $W_2 = ?, H_2 = ?$

Parameters: 0

MaxPooling

Input: $27 \times 27 \times 96$
Conv1: $K = 256, F = 5$
S = 1, P = 0
Output: $W_2 = ?, H_2 = ?$

Parameters: $(5 \times 5 \times 96) \times 256 = 0\,K$

MaxPooling

Input: $11 \times 11 \times 256$
Conv1: $K = 384, F = 3$
S = 1, P = 0
Output: $W_2 = ?, H_2 = ?$

Parameters: $(3 \times 3 \times 256) \times 384 = 0\,K$

MaxPooling

Input: $9 \times 9 \times 384$
Conv1: $K = 384, F = 3$
S = 1, P = 0
Output: $W_2 = ?, H_2 = ?$

Parameters: $(3 \times 3 \times 384) \times 256 = 0\,K$

MaxPooling

Input: $7 \times 7 \times 384$
Conv1: $K = 256, F = 3$
S = 1, P = 0
Output: $W_2 = ?, H_2 = ?$

Parameters: $(3 \times 3 \times 384) \times 256 = 0\,K$

MaxPooling

Input: $5 \times 5 \times 256$
Conv1: $K = 384, F = 3$
S = 1, P = 0
Output: $W_2 = ?, H_2 = ?$

Parameters: $(3 \times 3 \times 256) \times 384 = 1\,K$

MaxPooling

FC1
Parameters: $(2 \times 2 \times 256) \times 4096 = 4\,K$

Total Parameters: 27\,M

Mitesh M. Khapra
CS7015 (Deep Learning) : Lecture 11
Input: 227 × 227 × 3
Conv1: $K = 96, F = 11, S = 4, P = 0$
Output: $W_2 = ?, H_2 = ?$

Output: $W_2 = 55, H_2 = 55$
Parameters: ?

Convolution

Max Pool Input: 55 × 55 × 96
Conv1: $K = 3, F = 3, S = 2$
Output: $W_2 = ?, H_2 = ?$
Output: $W_2 = 27, H_2 = 27$
Parameters: ?

MaxPooling

Convolution

Input: 9 × 9 × 384
Conv1: $K = 384, F = 3$
Output: $W_2 = ?, H_2 = ?$
Parameters: ?

Convolution

Max Pool Input: 27 × 27 × 384
Conv1: $K = 256, F = 5, S = 1, P = 0$
Output: $W_2 = ?, H_2 = ?$
Output: $W_2 = 23, H_2 = 23$
Parameters: ?

MaxPooling

Convolution

Input: 11 × 11 × 256
Conv1: $K = 384, F = 3$
Output: $W_2 = ?, H_2 = ?$
Parameters: ?

Convolution

Max Pool Input: 9 × 9 × 384
Conv1: $K = 256, F = 3$
Output: $W_2 = ?, H_2 = ?$
Parameters: ?

MaxPooling

FC1
Parameters: (2 × 2 × 256) × 4096 = 4 M
dense
4096
dense
1000
Parameters: 4096 × 1000 = 4 M
Total Parameters: 27.55 M
Input: 227 × 227 × 3
Conv1: K = 96, F = 11
S = 4, P = 0
Output: W₂ = ?, H₂ = ?
Output: W₂ = 55, H₂ = 55
Parameters: ?
Parameters: (11 × 11 × 3) × 96 = 34.96M

Max Pooling Input: 55 × 55 × 96
Conv1: K = 3, F = 2
S = 2
Output: W₂ = ?, H₂ = ?
Output: W₂ = 27, H₂ = 27
Parameters: ?
Parameters: 0

Convolution 3
Max Pooling Input: 27 × 27 × 96
Conv1: K = 256, F = 5
S = 1, P = 0
Output: W₂ = ?, H₂ = ?
Output: W₂ = 23, H₂ = 23
Parameters: ?
Parameters: (5 × 5 × 96) × 256 = 0.627M

Max Pooling 3
Input: 23 × 23 × 256
Conv1: K = 384, F = 3
S = 1, P = 0
Output: W₂ = ?, H₂ = ?
Output: W₂ = 9, H₂ = 9
Parameters: ?
Parameters: (3 × 3 × 256) × 384 = 0.827M

Max Pooling 3
Input: 9 × 9 × 384
Conv1: K = 384, F = 3
S = 1, P = 0
Output: W₂ = ?, H₂ = ?
Output: W₂ = 5, H₂ = 5
Parameters: ?
Parameters: (3 × 3 × 384) × 256 = 0.827M

Max Pooling 3
Input: 5 × 5 × 256
Conv1: K = 256, F = 3
S = 2
Output: W₂ = ?, H₂ = ?
Output: W₂ = 2, H₂ = 2
Parameters: ?
Parameters: 0

FC1 Parameters: (2 × 2 × 256) × 4096 = 4M
dense Parameters: 4096 × 4096 = 16M
dense Parameters: 4096 × 1000 = 4M
Total Parameters: 27.55M

Input: 9 × 9 × 384
Conv1: K = 384, F = 3
S = 1, P = 0
Output: W₂ = 7, H₂ = 7
Parameters: (3 × 3 × 384) × 384 = 1.327M
Input: 227 \times 227 \times 3

Conv1: K = 96, F = 11
S = 4, P = 0

Output: W_2 = ?, H_2 = ?

Parameters: (11 \times 11 \times 3) \times 96 = 34,556

Max Pooling Input: 55 \times 55 \times 96
F = 3, S = 2

Output: W_2 = ?, H_2 = ?

Parameters: 0

Conv1: K = 256, F = 5
S = 1, P = 0

Output: W_2 = 23, H_2 = 23

Parameters: (5 \times 5 \times 96) \times 256 = 0,803,856

Max Pooling Input: 23 \times 23 \times 256
F = 3, S = 2

Output: W_2 = 11, H_2 = 11

Parameters: 0

Conv1: K = 384, F = 3
S = 1, P = 0

Output: W_2 = 9, H_2 = 9

Parameters: (3 \times 3 \times 256) \times 384 = 1,322,576

Max Pooling Input: 9 \times 9 \times 384
F = 3, S = 2

Output: W_2 = 7, H_2 = 7

Parameters: (3 \times 3 \times 384) \times 384 = 1,322,576

Conv1: K = 384, F = 3
S = 1, P = 0

Output: W_2 = 7, H_2 = 7

Parameters: (3 \times 3 \times 384) \times 384 = 1,322,576

Max Pooling Input: 7 \times 7 \times 384
F = 3, S = 2

Output: W_2 = 2, H_2 = 2

Parameters: 0

FC1 Parameters: (2 \times 2 \times 256) \times 4096 = 4,096

Parameters: 4096 \times 4096 = 16,384

Total Parameters: 27,188,640

Mitesh M. Khapra
Input: $7 \times 7 \times 384$

Conv1: $K = 256, F = 3$
$S = 1, P = 0$

Output: $W_2 = 5, H_2 = 5$

Parameters: 4096

Parameters: $4096 \times 4096 = 16M$

Parameters: $4096 \times 1000 = 4M$

Total Parameters: 27.55M
Input: 227 × 227 × 3
Conv1: \(K = 96, F = 11\)
\(S = 4, P = 0\)
Output: \(W_2 = ?, H_2 = ?\)
Parameters: \((11 \times 11 \times 3) \times 96 = 34K\)

Max Pooling Input: 227 × 227 × 96
Conv1: \(F = 3, S = 2\)
Output: \(W_2 = ?, H_2 = ?\)
Parameters: \(0\)

Convolution Input: 227 × 227 × 96
Conv1: \(K = 256, F = 5\)
\(S = 1, P = 0\)
Output: \(W_2 = 23, H_2 = 23\)
Parameters: \((5 \times 5 \times 96) \times 256 = 0.6M\)

Max Pooling Input: 227 × 227 × 96
Conv1: \(K = 256, F = 3\)
\(S = 1, P = 0\)
Output: \(W_2 = 7, H_2 = 7\)
Parameters: \((3 \times 3 \times 256) \times 384 = 0.327M\)

Max Pooling Input: 227 × 227 × 96
Conv1: \(K = 384, F = 3\)
\(S = 1, P = 0\)
Output: \(W_2 = 9, H_2 = 9\)
Parameters: \((3 \times 3 \times 256) \times 384 = 0.8M\)

Num. Parameters: 27.55M

Input: 7 × 7 × 384
Conv1: \(K = 256, F = 3\)
\(S = 1, P = 0\)
Output: \(W_2 = 5, H_2 = 5\)
Parameters: \((3 \times 3 \times 384) \times 256 = 0.8M\)
Input: 227 × 227 × 3
Conv1: $K = 96, F = 11, S = 4, P = 0$
Output: $W_2 = ?, H_2 = ?$
Output: $W_2 = 55, H_2 = 55$
Parameters: ?

Max Pool Input: 55 × 55 × 96
$F = 3, S = 2$
Output: $W_2 = ?, H_2 = ?$
Parameters: ?

Input: 27 × 27 × 96
Conv1: $K = 256, F = 5, S = 1, P = 0$
Output: $W_2 = ?, H_2 = ?$
Output: $W_2 = 23, H_2 = 23$
Parameters: ?

Max Pool Input: 23 × 23 × 256
$F = 3, S = 2$
Output: $W_2 = ?, H_2 = ?$
Output: $W_2 = 11, H_2 = 11$
Parameters: ?

Input: 11 × 11 × 256
Conv1: $K = 384, F = 3, S = 1, P = 0$
Output: $W_2 = ?, H_2 = ?$
Output: $W_2 = 9, H_2 = 9$
Parameters: ?

Max Pool Input: 9 × 9 × 384
$F = 3, S = 2$
Output: $W_2 = ?, H_2 = ?$
Output: $W_2 = 7, H_2 = 7$
Parameters: ?

Input: 7 × 7 × 384
Conv1: $K = 256, F = 3, S = 1, P = 0$
Output: $W_2 = ?, H_2 = ?$
Output: $W_2 = 5, H_2 = 5$
Parameters: ?

Max Pool Input: 5 × 5 × 256
$F = 3, S = 2$
Output: $W_2 = ?, H_2 = ?$
Parameters: ?

FC1
Parameters: (2 × 2 × 256) × 4096 = 4 M
Parameters: 4096 × 4096 = 16 M
Parameters: 4096 × 1000 = 4 M
Total Parameters: 27 M.

Mitesh M. Khapra  CS7015 (Deep Learning) : Lecture 11
Input: 227 \times 227 \times 3

Conv1: K = 96, F = 11, S = 4, P = 0

Output: W^2 = ?, H^2 = ?

Output: W^2 = 55, H^2 = 55

Parameters: ?

Max Pool Input: 5 \times 5 \times 256
F = 3, S = 2
Output: W_2 = 2, H_2 = 2
Parameters: ?

Parameters: (11 \times 11 \times 3) \times 96 = 34,998

MaxPooling

Convolution

Input: 27 \times 27 \times 96
Conv1: K = 256, F = 5, S = 1, P = 0

Output: W^2 = ?, H^2 = ?

Output: W^2 = 23, H^2 = 23

Parameters: ?

Parameters: (5 \times 5 \times 96) \times 256 = 0,819,200

Convolution

Max Pool Input: 23 \times 23 \times 256
F = 3, S = 2
Output: W^2 = ?, H^2 = ?

Output: W^2 = 11, H^2 = 11

Parameters: ?

Parameters: 0

MaxPooling

Convolution

Input: 11 \times 11 \times 256
Conv1: K = 384, F = 3, S = 1, P = 0

Output: W^2 = ?, H^2 = ?

Output: W^2 = 9, H^2 = 9

Parameters: ?

Parameters: (3 \times 3 \times 256) \times 384 = 0,87,043,200

Convolution

Max Pool Input: 9 \times 9 \times 384
F = 3, S = 1, P = 0
Output: W^2 = ?, H^2 = ?

Output: W^2 = 7, H^2 = 7

Parameters: ?

Parameters: (3 \times 3 \times 384) \times 256 = 0,87,043,200

Convolution

Max Pool Input: 7 \times 7 \times 384
F = 3, S = 1, P = 0
Output: W^2 = ?, H^2 = ?

Output: W^2 = 5, H^2 = 5

Parameters: ?

Parameters: (3 \times 3 \times 384) \times 256 = 0,87,043,200

Convolution

Max Pool Input: 5 \times 5 \times 256
F = 3, S = 2
Output: W^2 = ?, H^2 = ?

Parameters: ?

Parameters: 0

Convolution

Max Pooling

FC1 Parameters: (2 \times 2 \times 256) \times 4096 = 4,096,384

FC1 Parameters: 4096 \times 4096 = 16,777,216

Total Parameters: 27,655,664

Mitesh M. Khapra

CS7015 (Deep Learning) : Lecture 11
Input: 227 × 227 × 3
Conv1: K = 96, F = 11, S = 4, P = 0
Output: W₂ = ?, H₂ = ?
Output: W₂ = 55, H₂ = 55
Parameters: ?

Max Pool Input: 55 × 55 × 96
F = 3, S = 2
Output: W₂ = ?, H₂ = ?
Output: W₂ = 27, H₂ = 27
Parameters: ?

Convolution

MaxPooling

Input: 27 × 27 × 96
Conv1: K = 256, F = 5, S = 1, P = 0
Output: W₂ = ?, H₂ = ?
Output: W₂ = 23, H₂ = 23
Parameters: ?

MaxPooling

Input: 23 × 23 × 256
Conv1: K = 384, F = 3, S = 1, P = 0
Output: W₂ = ?, H₂ = ?
Output: W₂ = 9, H₂ = 9
Parameters: ?

Convolution

MaxPooling

Input: 9 × 9 × 384
Conv1: K = 384, F = 3, S = 1, P = 0
Output: W₂ = ?, H₂ = ?
Output: W₂ = 7, H₂ = 7
Parameters: ?

Convolution

MaxPooling

Input: 7 × 7 × 384
Conv1: K = 256, F = 3, S = 1, P = 0
Output: W₂ = ?, H₂ = ?
Output: W₂ = 5, H₂ = 5
Parameters: ?

MaxPooling

Input: 5 × 5 × 256
Max Pool: F = 3, S = 2
Output: W₂ = ?, H₂ = ?
Output: W₂ = 2, H₂ = 2
Parameters: 0

Parameters: (11 × 11 × 3) × 96 = 3456
Parameters: (5 × 5 × 96) × 256 = 0.6
Parameters: (3 × 3 × 256) × 384 = 0.8
Parameters: (3 × 3 × 384) × 256 = 1.0

Parameters: 4096 × 4096 = 16777216
Total Parameters: 27455
Input: 227 \times 227 \times 3

Conv1: 
\text{K} = 96, \ F = 11, \ S = 4, \ P = 0

Output: 
W^2 =?, \ H^2 =?

Parameters: 
(11 \times 11 \times 3) \times 96 = 34K

Max Pool: 
Input: 55 \times 55 \times 96 
\text{F} = 3, \ S = 2

Output: 
W^2 =?, \ H^2 =?

Parameters: 
0

Conv1: 
\text{K} = 256, \ F = 5, \ S = 1, \ P = 0

Output: 
W^2 =?, \ H^2 =?

Parameters: 
(5 \times 5 \times 96) \times 256 = 0K

Max Pool: 
Input: 23 \times 23 \times 256
\text{F} = 3, \ S = 2

Output: 
W^2 =?, \ H^2 =?

Parameters: 
0

Conv1: 
\text{K} = 384, \ F = 3, \ S = 1, \ P = 0

Output: 
W^2 =?, \ H^2 =?

Parameters: 
(3 \times 3 \times 256) \times 384 = 0K

Conv1: 
\text{K} = 384, \ F = 3, \ S = 1, \ P = 0

Output: 
W^2 =?, \ H^2 =?

Parameters: 
(3 \times 3 \times 384) \times 384 = 1K

Max Pool: 
Input: 9 \times 9 \times 384
\text{F} = 3, \ S = 2

Output: 
W^2 =?, \ H^2 =?

Parameters: 
0

Conv1: 
\text{K} = 256, \ F = 3, \ S = 1, \ P = 0

Output: 
W^2 =?, \ H^2 =?

Parameters: 
(3 \times 3 \times 384) \times 256 = 0K

Max Pool: 
Input: 7 \times 7 \times 256
\text{F} = 3, \ S = 2

Output: 
W^2 =?, \ H^2 =?

Parameters: 
0

FC1: 
Parameters: 
(2 \times 2 \times 256) \times 4096 = 4M

Total Parameters: 
27.55M

Mitesh M. Khapra
CS7015 (Deep Learning) : Lecture 11
Input: 227 × 227 × 3

Conv1:
- \( K = 96 \), \( F = 11 \), \( S = 4 \), \( P = 0 \)

Output:
- \( W_2 = ? \), \( H_2 = ? \)

Parameters:
- \( (11 \times 11 \times 3) \times 96 = 34,992 \)

Max Pool
- Input: 55 × 55 × 96
- \( F = 3 \), \( S = 2 \)
- Output:
  - \( W_2 = ? \), \( H_2 = ? \)
  - Output:
    - \( W_2 = 27 \), \( H_2 = 27 \)
    - Parameters: 0

Convolution
- Input: 27 × 27 × 96
- \( K = 256 \), \( F = 5 \), \( S = 1 \), \( P = 0 \)
- Output:
  - \( W_2 = ? \), \( H_2 = ? \)
  - Output:
    - \( W_2 = 23 \), \( H_2 = 23 \)
    - Parameters: 0

Max Pool
- Input: 23 × 23 × 256
- \( F = 3 \), \( S = 2 \)
- Output:
  - \( W_2 = ? \), \( H_2 = ? \)
  - Output:
    - \( W_2 = 11 \), \( H_2 = 11 \)
    - Parameters: 0

Convolution
- Input: 11 × 11 × 256
- \( K = 384 \), \( F = 3 \), \( S = 1 \), \( P = 0 \)
- Output:
  - \( W_2 = ? \), \( H_2 = ? \)
  - Output:
    - \( W_2 = 9 \), \( H_2 = 9 \)
    - Parameters: 0

Max Pool
- Input: 9 × 9 × 384
- \( F = 3 \), \( S = 2 \)
- Output:
  - \( W_2 = ? \), \( H_2 = ? \)
  - Output:
    - \( W_2 = 5 \), \( H_2 = 5 \)
    - Parameters: 0

Convolution
- Input: 5 × 5 × 384
- \( K = 256 \), \( F = 3 \), \( S = 1 \), \( P = 0 \)
- Output:
  - \( W_2 = ? \), \( H_2 = ? \)
  - Output:
    - \( W_2 = 3 \), \( H_2 = 3 \)
    - Parameters: 0

Max Pool
- Input: 3 × 3 × 256
- \( F = 3 \), \( S = 2 \)
- Output:
  - \( W_2 = ? \), \( H_2 = ? \)
  - Output:
    - \( W_2 = 1 \), \( H_2 = 1 \)
    - Parameters: 0

FC1
- Parameters: \( 4096 \times 4096 = 16M \)

Total Parameters: 27,552M
Input: 227 × 227 × 3
Conv1: K = 96, F = 11, S = 4, P = 0
Output: W_2 = ?, H_2 = ?
Output: W_2 = 55, H_2 = 55
Parameters: (11 × 11 × 3) × 96 = 34,944

MaxPooling Input: 55 × 55 × 96
F = 3, S = 2
Output: W_2 = ?, H_2 = ?
Output: W_2 = 27, H_2 = 27
Parameters: 0

Convolution Input: 27 × 27 × 96
Conv1: K = 256, F = 5, S = 1, P = 0
Output: W_2 = ?, H_2 = ?
Output: W_2 = 23, H_2 = 23
Parameters: (5 × 5 × 96) × 256 = 0

MaxPooling Input: 23 × 23 × 256
F = 3, S = 2
Output: W_2 = ?, H_2 = ?
Output: W_2 = 11, H_2 = 11
Parameters: 0

Convolution Input: 11 × 11 × 256
Conv1: K = 384, F = 3, S = 1, P = 0
Output: W_2 = ?, H_2 = ?
Output: W_2 = 9, H_2 = 9
Parameters: (3 × 3 × 256) × 384 = 0

MaxPooling Input: 9 × 9 × 384
F = 3, S = 2
Output: W_2 = ?, H_2 = ?
Output: W_2 = 5, H_2 = 5
Parameters: 0

Convolution Input: 7 × 7 × 384
Conv1: K = 256, F = 3, S = 1, P = 0
Output: W_2 = ?, H_2 = ?
Output: W_2 = 5, H_2 = 5
Parameters: (3 × 3 × 384) × 256 = 0

MaxPooling Input: 5 × 5 × 256
F = 3, S = 2
Output: W_2 = ?, H_2 = ?
Output: W_2 = 2, H_2 = 2
Parameters: 0

FC1
Parameters: 4096 × 1000 = 4M
Input: $227 \times 227 \times 3$

Conv1: $K = 96$, $F = 11$, $S = 4$, $P = 0$

Output: $W_2 = \_\_\_\_$, $H_2 = \_\_\_\_$

Parameters: $(11 \times 11 \times 3) \times 96 = 34K$

Max Pooling

Input: $55 \times 55 \times 96$

Conv1: $K = 3$, $F = 3$, $S = 2$

Output: $W_2 = \_\_\_\_$, $H_2 = \_\_\_\_$

Parameters: $0K$

Max Pooling

Input: $27 \times 27 \times 96$

Conv1: $K = 256$, $F = 5$, $S = 1$, $P = 0$

Output: $W_2 = \_\_\_\_$, $H_2 = \_\_\_\_$

Parameters: $(5 \times 5 \times 96) \times 256 = 0M$

Max Pooling

Input: $11 \times 11 \times 256$

Conv1: $K = 384$, $F = 3$, $S = 1$, $P = 0$

Output: $W_2 = \_\_\_\_$, $H_2 = \_\_\_\_$

Parameters: $(3 \times 3 \times 256) \times 384 = 0M$

Max Pooling

Input: $9 \times 9 \times 384$

Conv1: $K = 384$, $F = 3$, $S = 1$, $P = 0$

Output: $W_2 = \_\_\_\_$, $H_2 = \_\_\_\_$

Parameters: $(3 \times 3 \times 384) \times 384 = 1M$

Max Pooling

Input: $7 \times 7 \times 384$

Conv1: $K = 256$, $F = 3$, $S = 1$, $P = 0$

Output: $W_2 = \_\_\_\_$, $H_2 = \_\_\_\_$

Parameters: $(3 \times 3 \times 384) \times 256 = 0M$

Max Pooling

Input: $5 \times 5 \times 256$

Conv1: $K = 384$, $F = 3$, $S = 1$, $P = 0$

Output: $W_2 = \_\_\_\_$, $H_2 = \_\_\_\_$

Parameters: $(3 \times 3 \times 384) \times 384 = 1M$

Max Pooling

Input: $2 \times 2 \times 256$

FC1 Parameters: $(2 \times 2 \times 256) \times 4096 = 4M$

Total Parameters: $27.55M$
Let us look at the connections in the fully connected layers in more detail.

MaxPooling
Let us look at the connections in the fully connected layers in more detail.

We will first stretch out the last conv or maxpool layer to make it a 1d vector.

\[ 2 \times 2 \times 256 = 1024 \]
Let us look at the connections in the fully connected layers in more detail.

We will first stretch out the last conv or maxpool layer to make it a 1d vector.

This 1d vector is then densely connected to other layers just as in a regular feedforward neural network.

\[
\begin{align*}
&\text{MaxPooling} \\
&2 \times 2 \times 256 = 1024 \\
&\text{dense} \\
&4096
\end{align*}
\]
Let us look at the connections in the fully connected layers in more detail.

We will first stretch out the last conv or maxpool layer to make it a 1d vector.

This 1d vector is then densely connected to other layers just as in a regular feedforward neural network.
Let us look at the connections in the fully connected layers in more detail.

We will first stretch out the last conv or maxpool layer to make it a 1d vector.

This 1d vector is then densely connected to other layers just as in a regular feedforward neural network.
ImageNet Success Stories (roadmap for rest of the talk)

- AlexNet
- ZFNet
- VGGNet
Layer 1:

\[ F = 11 \rightarrow 7 \]

Difference in Parameters:

\[(11^2 - 7^2) \times 3 \times 96 = 2072\]

Layer 2: No difference

Layer 3: No difference

Layer 4: No difference

Layer 5:

\[ K = 384 \rightarrow 512 \]

Difference in Parameters:

\[(3 \times 3 \times 256) \times (512 - 384) = 0\]

Layer 6:

\[ K = 384 \rightarrow 1024 \]

Difference in Parameters:

\[(3 \times 3 \times (384 \times 384) - (512 \times 1024)) = 0\]

Layer 7:

\[ K = 256 \rightarrow 512 \]

Difference in Parameters:

\[(3 \times 3 \times (384 \times 256) - (1024 \times 512)) = 0\]

Layer 8: No difference

Layer 9: No difference

Layer 10: No difference

Difference in Total No. of Parameters: 145,024
Layer 1: $F = 11 \rightarrow 7$
Difference in Parameters

$((11^2 - 7^2) \times 3) \times 96 = 20.7K$
Layer 1: $F = 11 \rightarrow 7$

Difference in Parameters

\[ ((11^2 - 7^2) \times 3) \times 96 = 20.7K \]
Layer 1: $F = 11 \rightarrow 7$

Difference in Parameters: $\left(11^2 - 7^2\right) \times 3 \times 96 = 20$. 7

Layer 2: No difference

Layer 3: No difference

Layer 4: No difference

Layer 5: $K = 384 \rightarrow 512$

Difference in Parameters: $3 \times 3 \times 256 \times (512 - 384) = 0$. 29

Layer 6: $K = 384 \rightarrow 1024$

Difference in Parameters: $3 \times 3 \times (384 \times 384) - (512 \times 1024) = 0$. 8

Layer 7: $K = 256 \rightarrow 512$

Difference in Parameters: $3 \times 3 \times (384 \times 256) - (1024 \times 512) = 0$. 36

Layer 8: No difference

Layer 9: No difference

Layer 10: No difference

Difference in Total No. of Parameters: 1,452,968

Layer 2: No difference
Layer1: $F = 11 \rightarrow 7$

Difference in Parameters:

$((11^2 - 7^2) \times 3) \times 96 = 20.79$

Layer2: No difference

Layer3: No difference

Layer4: No difference

Layer5: $K = 384 \rightarrow 512$

Difference in Parameters:

$(3 \times 3 \times 256) \times (512 - 384) = 0.29$

Layer6: $K = 384 \rightarrow 1024$

Difference in Parameters:

$(3 \times 3 \times 384 \times 384) - (512 \times 1024) = 0.8$

Layer7: $K = 256 \rightarrow 512$

Difference in Parameters:

$(3 \times 3 \times 384 \times 256) - (1024 \times 512) = 0.36$

Layer8: No difference

Layer9: No difference

Layer10: No difference

Difference in Total No. of Parameters: $1.45$
Layer1: Difference in Parameters \((11^2 - 7^2) \times 3 \times 96 = 20\).

Layer2: No difference

Layer3: No difference

Layer4: No difference

Layer5: Difference in Parameters \((3 \times 3 \times 256) \times (512 - 384) = 0\).

Layer6: Difference in Parameters \((3 \times 3 \times (384 \times 384) - (512 \times 1024)) = 0\).

Layer7: Difference in Parameters \((3 \times 3 \times (384 \times 256) - (1024 \times 512)) = 0\).

Layer8: No difference

Layer9: No difference

Layer10: No difference

Difference in Total No. of Parameters \(1.45\).
Layer1:

\[ F = 11 \rightarrow 7 \]

Difference in Parameters

\[ ((11^2 - 7^2) \times 3) \times 96 = 20 \cdot 7 \]

Layer2: No difference

Layer3: No difference

Layer4: No difference

Layer5:

\[ K = 384 \rightarrow 512 \]

Difference in Parameters

\[ (3 \times 3 \times 256) \times (512 - 384) = 0 \cdot 29 \]

Layer6:

\[ K = 384 \rightarrow 1024 \]

Difference in Parameters

\[ (3 \times 3 \times ((384 \times 384) - (512 \times 1024))) = 0 \cdot 8 \]

Layer7:

\[ K = 256 \rightarrow 512 \]

Difference in Parameters

\[ (3 \times 3 \times ((384 \times 256) - (1024 \times 512))) = 0 \cdot 36 \]

Layer8: No difference

Layer9: No difference

Layer10: No difference

Layer10: No difference

Difference in Total No. of Parameters

1.45M
Layer 1:
\[ F = 11 \rightarrow 7 \]
Difference in Parameters:
\[ \left(11 - 7\right)^2 \times 3 \times 96 = 20 \]

Layer 2: No difference

Layer 3: No difference

Layer 4: No difference

Layer 5:
\[ K = 384 \rightarrow 512 \]
Difference in Parameters:
\[ \left(3 \times 3 \times 256\right) \times \left(512 - 384\right) = 0 \]

Layer 6:
\[ K = 384 \rightarrow 1024 \]
Difference in Parameters:
\[ \left(3 \times 3 \times 384\right) \times \left(1024 - 512\right) = 0 \]

Layer 7:
\[ K = 256 \rightarrow 512 \]
Difference in Parameters:
\[ \left(3 \times 3 \times 256\right) \times \left(512 - 1024\right) = 0 \]

Layer 8: No difference

Layer 9: No difference

Layer 10: No difference

Difference in Total No. of Parameters:
\[ 1.45 \times 10^6 \]
Layer 1:

- Input 11 $\rightarrow$ 7

Difference in Parameters: 
\[(11 - 7) \times 3 \times 96 = 20,760\]

Layer 2: No difference

Layer 3: No difference

Layer 4: No difference

Layer 5:

- Input 384 $\rightarrow$ 512

Difference in Parameters: 
\[3 \times 3 \times 256 \times (512 - 384) = 0,29,520\]

Layer 6:

- Input 384 $\rightarrow$ 1024

Difference in Parameters: 
\[3 \times 3 \times 384 \times (512 - 1024) = 0,81,984\]

Layer 7:

- Input 256 $\rightarrow$ 512

Difference in Parameters: 
\[3 \times 3 \times 384 \times (512 - 1024) = 0,36,928\]

Layer 8: No difference

Layer 9: No difference

Layer 10: No difference

Difference in Total No. of Parameters: 
\[14,529,764\]
Layer 1:

\[ K = 11 \rightarrow 7 \]

Difference in Parameters

\[ (11^2 - 7^2) \times 3 \times 96 = 20.7 \]

Layer 2: No difference

Layer 3: No difference

Layer 4: No difference

Layer 5:

\[ K = 384 \rightarrow 512 \]

Difference in Parameters

\[ (3 \times 3 \times 256) \times (512 - 384) = 0.29M \]

Layer 6:

\[ K = 384 \rightarrow 1024 \]

Difference in Parameters

\[ (3 \times 3 \times (384 \times 384)) \times (1024 - 512) = 0 \]

Layer 7:

\[ K = 256 \rightarrow 512 \]

Difference in Parameters

\[ (3 \times 3 \times (384 \times 256)) \times (512 - 1024) = 0 \]

Layer 8: No difference

Layer 9: No difference

Layer 10: No difference

Difference in Total No. of Parameters

\[ 1.45M \]

Mitesh M. Khapra

CS7015 (Deep Learning): Lecture 11
Layer 1:
- \( F = 11 \rightarrow 7 \)
- Difference in Parameters: \((11 - 7)^2 \times 3 \times 96 = 20\).  

Layer 2: No difference.

Layer 3: No difference.

Layer 4: No difference.

Layer 5:
- \( K = 384 \rightarrow 512 \)
- Difference in Parameters: 
  \[(3 \times 3 \times 256) \times (512 - 384) = 0.29M\]

Layer 6:
- \( K = 384 \rightarrow 1024 \)
- Difference in Parameters: 
  \[(3 \times 3 \times 384) \times (1024 - 512) = 0 \].

Layer 7:
- \( K = 256 \rightarrow 512 \)
- Difference in Parameters: 
  \[(3 \times 3 \times 256) \times (512 - 1024) = 0 \].

Layer 8: No difference.

Layer 9: No difference.

Layer 10: No difference.

Difference in Total No. of Parameters: 1.45M
Layer 1: $F = 11 \rightarrow 7$

Difference in Parameters \((11^2 - 7^2) \times 3 \times 96 = 2070\).

Layer 2: No difference

Layer 3: No difference

Layer 4: No difference

Layer 5: $K = 384 \rightarrow 512$

Difference in Parameters \((3 \times 3 \times (384 \times 256) - (512 \times 1024)) = 0.8M\).

Layer 6: $K = 384 \rightarrow 1024$

Difference in Parameters \((3 \times 3 \times (384 \times 256) - (1024 \times 512)) = 0.8M\).

Layer 7: $K = 256 \rightarrow 512$

Difference in Parameters \((3 \times 3 \times (384 \times 256) - (1024 \times 512)) = 0.8M\).

Layer 8: No difference

Layer 9: No difference

Layer 10: No difference

Layer 10: No difference
Layer 1:
- Convolution: \( F = 11 \rightarrow 7 \)
- Difference in Parameters: \((11^2 - 7^2) \times 3 \times 96 = 20,736\)

Layer 2:
- No difference

Layer 3:
- No difference

Layer 4:
- No difference

Layer 5:
- \( K = 384 \rightarrow 512 \)
- Difference in Parameters: \((3 \times 3 \times (384 \times 384) - (512 \times 1024)) = 0.8M\)

Layer 6:
- \( K = 384 \rightarrow 1024 \)
- Difference in Parameters: \((3 \times 3 \times (384 \times 256) - (1024 \times 512)) = 0\)

Layer 7:
- \( K = 256 \rightarrow 512 \)
- Difference in Parameters: \((3 \times 3 \times (256 \times 256) - (1024 \times 512)) = 0\)

Layer 8:
- No difference

Layer 9:
- No difference

Layer 10:
- No difference
Layer 1:
\[ F = 11 \rightarrow 7 \]

Difference in Parameters
\[ (11^2 - 7^2) \times 3 \times 96 = 20 \times 96 = 1920 \]

Layer 2: No difference

Layer 3: No difference

Layer 4: No difference

Layer 5:
\[ K = 384 \rightarrow 512 \]

Difference in Parameters
\[ (3 \times 3 \times 256) \times (512 - 384) = 3 \times 3 \times 256 \times 128 = 13824 \times 128 = 1792304 \]

Layer 6:
\[ K = 384 \rightarrow 1024 \]

Difference in Parameters
\[ (3 \times 3 \times (384 \times 384) - (1024 \times 512)) = 3 \times 3 \times (384 \times 384) - 3 \times 3 \times (1024 \times 512) = 3 \times 3 \times 384 \times 384 - 3 \times 3 \times 1024 \times 512 = 13824 \times 384 - 13824 \times 1024 = 5242880 - 14457920 = -9215040 \]

Layer 7:
\[ K = 256 \rightarrow 512 \]

Difference in Parameters
\[ (3 \times 3 \times (384 \times 256) - (1024 \times 512)) = 3 \times 3 \times (384 \times 256) - 3 \times 3 \times (1024 \times 512) = 13824 \times 256 - 13824 \times 1024 = 3574464 - 14275872 = -10701408 \]

Layer 8: No difference

Layer 9: No difference

Layer 10: No difference

Difference in Total No. of Parameters
\[ 1.45M \]
Input

Layer 1:
- \( F = 11 \rightarrow 7 \)
- Difference in Parameters: \((11^2 - 7^2) \times 3 \times 96 = 20\)

Layer 2: No difference

Layer 3: No difference

Layer 4: No difference

Layer 5:
- \( K = 384 \rightarrow 512 \)
- Difference in Parameters: \((3 \times 3 \times 256) \times (512 - 384) = 0.36M\)

Layer 6:
- \( K = 384 \rightarrow 1024 \)
- Difference in Parameters: \((3 \times 3 \times (384 \times 384) - (1024 \times 512)) = 0\)

Layer 7:
- \( K = 256 \rightarrow 512 \)
- Difference in Parameters: \((3 \times 3 \times (384 \times 256) - (1024 \times 512)) = 0\)

Layer 8: No difference

Layer 9: No difference

Layer 10: No difference

Difference in Total No. of Parameters: 1.45M
Layer 1:
$F = 11 \rightarrow 7$

Difference in Parameters:
$((11^2 - 7^2) \times 3) \times 96 = 2072$.

Layer 2:
No difference

Layer 3:
No difference

Layer 4:
No difference

Layer 5:
$K = 384 \rightarrow 512$

Difference in Parameters:
$(3 \times 3 \times 256) \times (512 - 384) = 0$.

Layer 6:
$K = 384 \rightarrow 1024$

Difference in Parameters:
$(3 \times 3 \times (384 \times 384) - (512 \times 1024)) = 0$.

Layer 7:
$K = 256 \rightarrow 512$

Difference in Parameters:
$(3 \times 3 \times (384 \times 256) - (1024 \times 512)) = 0$.

Layer 8:
No difference

Layer 9:
No difference

Layer 10:
No difference

Difference in Total No. of Parameters:
$1.45 \times 10^5$.
Layer 1: 
\[ F = 11 \rightarrow 7 \]
Difference in Parameters:
\[ ((11^2 - 7^2) \times 3) \times 96 = 20 \]

Layer 2: No difference

Layer 3: No difference

Layer 4: No difference

Layer 5: 
\[ K = 384 \rightarrow 512 \]
Difference in Parameters:
\[ (3 \times 3 \times 256) \times (512 - 384) = 0 \]

Layer 6: 
\[ K = 384 \rightarrow 1024 \]
Difference in Parameters:
\[ (3 \times 3 \times (384 \times 384) - (512 \times 1024)) = 0 \]

Layer 7: 
\[ K = 256 \rightarrow 512 \]
Difference in Parameters:
\[ (3 \times 3 \times (384 \times 256) - (1024 \times 512)) = 0 \]

Layer 8: No difference

Layer 9: No difference

Layer 10: No difference

Difference in Total No. of Parameters: 1.45 M

Mitesh M. Khapra | CS7015 (Deep Learning) : Lecture 11
Layer 1:

Difference in Parameters

\[(11 \times 2) \times 3 \times 96 = 20.7\]

Layer 2:

No difference

Layer 3:

No difference

Layer 4:

No difference

Layer 5:

\[K = 384 \rightarrow 512\]

Difference in Parameters

\[(3 \times 3 \times 256) \times (512 - 384) = 0.29\]

Layer 6:

\[K = 384 \rightarrow 1024\]

Difference in Parameters

\[(3 \times 3 \times (384 \times 384) - (512 \times 1024)) = 0.8\]

Layer 7:

\[K = 256 \rightarrow 512\]

Difference in Parameters

\[(3 \times 3 \times (384 \times 256) - (1024 \times 512)) = 0.36\]

Layer 8:

No difference

Layer 9:

No difference

Layer 10:

No difference
Input

Layer 1:

\[ F = 11 \rightarrow 7 \]

\[
\text{Difference in Parameters} \quad ((11^2 - 7^2) \times 3) \times 96 = 20.
\]

Convolution

Layer 2: No difference

MaxPooling

Layer 3: No difference

Convolution

Layer 4: No difference

MaxPooling

Layer 5:

\[ K = 384 \rightarrow 512 \]

\[
\text{Difference in Parameters} \quad (3 \times 3 \times 256) \times (512 - 384) = 0.
\]

Convolution

Layer 6:

\[ K = 384 \rightarrow 1024 \]

\[
\text{Difference in Parameters} \quad (3 \times 3 \times (384 \times 384) - (512 \times 1024)) = 0.
\]

Convolution

Layer 7:

\[ K = 256 \rightarrow 512 \]

\[
\text{Difference in Parameters} \quad (3 \times 3 \times (384 \times 256) - (1024 \times 512)) = 0.
\]

MaxPooling

Layer 8: No difference

MaxPooling

Layer 9: No difference

Dense

Layer 10: No difference

Dense

Difference in Total No. of Parameters \[ 1.45 \text{M} \]
Layer 1:

\[ F = 11 \rightarrow 7 \]

Difference in Parameters:

\[
(11^2 - 7^2) \times 3 \times 96 = 20
\]

Layer 2: No difference

Layer 3: No difference

Layer 4: No difference

Layer 5:

\[ K = 384 \rightarrow 512 \]

Difference in Parameters:

\[
(3 \times 3 \times 256) \times (512 - 384) = 0
\]

Layer 6:

\[ K = 384 \rightarrow 1024 \]

Difference in Parameters:

\[
(3 \times 3 \times (384 \times 384) - (512 \times 1024)) = 0
\]

Layer 7:

\[ K = 256 \rightarrow 512 \]

Difference in Parameters:

\[
(3 \times 3 \times (384 \times 256) - (1024 \times 512)) = 0
\]

Layer 8: No difference

Layer 9: No difference

Layer 10: No difference

Difference in Total No. of Parameters:

\[ 1.45 \text{M} \]

Layer 10: No difference
Layer 1:
- F = 11 → 7

Difference in Parameters
\[(11^2 - 7^2) \times 3 \times 96 = 20736\]

Layer 2: No difference

Layer 3: No difference

Layer 4: No difference

Layer 5:
- K = 384 → 512
- Difference in Parameters
  \[3 \times 3 \times 256 \times (512 - 384) = 0\]

Layer 6:
- K = 384 → 1024
- Difference in Parameters
  \[3 \times 3 \times (384 \times 384) - (512 \times 1024) = 0\]

Layer 7:
- K = 256 → 512
- Difference in Parameters
  \[3 \times 3 \times (384 \times 256) - (1024 \times 512) = 0\]

Layer 8: No difference

Layer 9: No difference

Layer 10: No difference

Difference in Total No. of Parameters
\[1.45M\]
ImageNet Success Stories (roadmap for rest of the talk)

- AlexNet
- ZFNet
- VGGNet
Input

Kernel size is 3 × 3 throughout

Total parameters in non FC layers = ∼16 M

Total Parameters in FC layers = (512 × 7 × 7 × 4096) + (4096 × 4096) + (4096 × 1024) = ∼122 M

Most parameters are in the first FC layer (∼102 M)
Kernel size is 3 × 3 throughout.

Total parameters in non FC layers = ∼16M

Total Parameters in FC layers = (512 × 7 × 7 × 4096) + (4096 × 4096) + (4096 × 1024) = ∼122M

Most parameters are in the first FC layer (∼102M)
Kernel size is 3 × 3 throughout.

Total parameters in non-FC layers = \( \sim 16 \text{M} \)

Total parameters in FC layers = \( (512 \times 7 \times 7 \times 4096) + (4096 \times 4096) + (4096 \times 1024) = \sim 122 \text{M} \)

Most parameters are in the first FC layer (\( \sim 102 \text{M} \)).
Kernel size is 3 × 3 throughout.

Total parameters in non FC layers = ∼16M

Total parameters in FC layers = (512 × 7 × 7 × 4096) + (4096 × 4096) + (4096 × 1024) = ∼122M

Most parameters are in the first FC layer (∼102M).
Kernel size is 3 × 3 throughout.

Total parameters in non FC layers = \( \approx 16 \times 10^6 \)

Total parameters in FC layers =
\[
(512 \times 7 \times 7 \times 4096) + (4096 \times 4096) + (4096 \times 1024) = \approx 122 \times 10^6
\]

Most parameters are in the first FC layer (\( \approx 102 \times 10^6 \)).
Kernel size is 3 $\times$ 3 throughout.

Total parameters in non-FC layers = $\sim 16$ M

Total parameters in FC layers = $(512 \times 7 \times 7 \times 4096) + (4096 \times 4096) + (4096 \times 1024) = \sim 122$ M

Most parameters are in the first FC layer ($\sim 102$ M)
Kernel size is 3 × 3 throughout.

Total parameters in non-FCLayers = \( \sim 16 \text{ M} \)

Total Parameters in FCLayers = \((512 \times 7 \times 7 \times 4096) + (4096 \times 4096) + (4096 \times 1024) = \sim 122 \text{ M}\)

Most parameters are in the first FCLayer (\( \sim 102 \text{ M} \)).
Input 224 224
Conv 64 64
maxpool 112 112
Conv 128 128
maxpool 56 56
Conv 256 256
maxpool 28 28
Conv 512 512
maxpool 14 14
Conv 512 512
maxpool 7 7
fc 4096 4096
softmax 1000

Kernel size is 3 × 3 throughout.

Total parameters in non FC layers = \( \sim 16 \times 10^6 \)

Total Parameters in FC layers = \( (512 \times 7 \times 7 \times 4096) + (4096 \times 4096) + (4096 \times 1024) = \sim 122 \times 10^6 \)

Most parameters are in the first FC layer (\( \sim 102 \times 10^6 \)).
Kernel size is 3 × 3 throughout.

Total parameters in non-FC layers = \( \sim 16\text{M} \)

Total Parameters in FC layers = 
\[ (512 \times 7 \times 7 \times 4096) + (4096 \times 4096) + (4096 \times 1024) = \sim 122\text{M} \]

Most parameters are in the first FC layer (\( \sim 102\text{M} \)).
Kernel size is 3 × 3 throughout.

Total parameters in non-FC layers = \( \sim 16 \) M

Total parameters in FC layers = 
\[ (512 \times 7 \times 7 \times 4096) + (4096 \times 4096) + (4096 \times 1024) = \sim 122 \] M

Most parameters are in the first FC layer (\( \sim 102 \) M)
Total parameters in non FC layers = \( \sim 16 \text{M} \)

Total parameters in FC layers = 
\[ (512 \times 7 \times 7 \times 4096) + (4096 \times 4096) + (4096 \times 1024) = \sim 122 \text{M} \]

Most parameters are in the first FC layer (\( \sim 102 \text{M} \))
Input 224 224
Conv 224 224 64
maxpool 112 112 64
Conv 112 112 128
maxpool 56 56 128
Conv 56 56 256
maxpool 28 28 256
Conv 28 28 512
maxpool 14 14 512
Conv 14 14 512
maxpool 7 7 512
fc 4096 4096
softmax 1000

Kernel size is 3×3 throughout.
Total parameters in non-FC layers = ∼16M
Total parameters in FC layers = (512 × 7 × 7 × 4096) + (4096 × 4096) + (4096 × 1024) = ∼122M
Most parameters are in the first FC layer (∼102M)
Kernel size is 3×3 throughout.

Total parameters in non-FC layers = \( \sim 16 \text{ M} \)

Total parameters in FC layers = 
\[
(512 \times 7 \times 7 \times 4096) + (4096 \times 4096) + (4096 \times 1024) = \sim 122 \text{ M}
\]

Most parameters are in the first FC layer (\( \sim 102 \text{ M} \)).
Kernel size is 3 × 3 throughout.

Total parameters in non-FC layers = \( \sim 16 \) M

Total parameters in FC layers = 
\[ \begin{align*}
(512 \times 7 \times 7 \times 4096) + (4096 \times 4096) + (4096 \times 1024) = \sim 122 \text{ M}
\end{align*} \]

Most parameters are in the first FC layer (\( \sim 102 \) M)
Input 224 224
Conv 224 224
maxpool 112 112
Conv 112 112
64
Conv 112 112
maxpool 56 56
256
maxpool 28 28
512
maxpool 14 14
512
fc 4096 4096
softmax 1000

Kernel size is 3 \times 3 throughout.

Total parameters in non FC layers = \sim 16M

Total Parameters in FC layers = (512 \times 7 \times 7 \times 4096) + (4096 \times 4096) + (4096 \times 1024) = \sim 122M

Most parameters are in the first FC layer (\sim 102M)
Kernel size is $3 \times 3$ throughout.

Total parameters in non-FC layers = $\sim 16 \text{M}$

Total Parameters in FC layers = $(512 \times 7 \times 7 \times 4096) + (4096 \times 4096) + (4096 \times 1024) = \sim 122 \text{M}$

Most parameters are in the first FC layer ($\sim 102\text{M}$).
Kernel size is 3 × 3 throughout.

Total parameters in non-FC layers = \( \sim 16 \) M

Total parameters in FC layers = \( (512 \times 7 \times 7 \times 4096) + (4096 \times 4096) + (4096 \times 1024) = \sim 122 \) M

Most parameters are in the first FC layer (\( \sim 102 \) M)
Kernel size is 3 × 3 throughout.

Total parameters in non-FC layers = \( \sim 16 \text{M} \)

Total parameters in FC layers = \( (512 \times 7 \times 7 \times 4096) + (4096 \times 4096) + (4096 \times 1024) = \sim 122 \text{M} \)

Most parameters are in the first FC layer (\( \sim 102 \text{M} \)).
Kernel size is 3 × 3 throughout.

Total parameters in non-FC layers = \( \sim 16 \) M

Total Parameters in FC layers = \( (512 \times 7 \times 7 \times 4096) + (4096 \times 4096) + (4096 \times 1024) \) = \( \sim 122 \) M

Most parameters are in the first FC layer (\( \sim 102M \))
Input 224 Conv 224 maxpool 112 Conv 112 maxpool 56 Conv 56 maxpool 28 Conv 28 maxpool 14 Conv 14 maxpool 7 fc 4096

Kernel size is 3 × 3 throughout.

Total parameters in non-FC layers = \( \sim 16 \) M

Total parameters in FC layers = 
\[ (512 \times 7 \times 7 \times 4096) + (4096 \times 4096) + (4096 \times 1024) = \sim 122 \] M

Most parameters are in the first FC layer (\( \sim 102 \) M).
Input 224 224 Conv 224 224 maxpool 112 112 Conv 112 112 maxpool 56 56 Conv 56 56 maxpool 28 28 Conv 28 28 maxpool 14 14 Conv 14 14 maxpool 7 7 fc 4096 4096

Kernel size is 3 × 3 throughout.

Total parameters in non FC layers = ∼16 M

Total Parameters in FC layers = (512 × 7 × 7 × 4096) + (4096 × 4096) + (4096 × 1024) = ∼122 M

Most parameters are in the first FC layer (∼102 M).
Input 224
Conv 224
maxpool 112
Conv 112
maxpool 56
Conv 56
maxpool 28
Conv 28
maxpool 14
Conv 14
maxpool 7
fc
fc

Kernel size is 3 × 3 throughout.

Total parameters in non FC layers = \(\sim 16\) M

Total Parameters in FC layers = \((512 \times 7 \times 7 \times 4096) + (4096 \times 4096) + (4096 \times 1024)\) = \(\sim 122\) M

Most parameters are in the first FC layer (\(\sim 102\) M).

Mitesh M. Khapra
CS7015 (Deep Learning) : Lecture 11
Kernel size is 3×3 throughout.

Total parameters in non-FC layers = \( \sim 16 \) M.

Total parameters in FC layers = 
\[
(512 \times 7 \times 7 \times 4096) + (4096 \times 4096) + (4096 \times 1024) = \sim 122 \) M.

Most parameters are in the first FC layer (\( \sim 102 \) M).
- Kernel size is $3 \times 3$ throughout
• Kernel size is $3 \times 3$ throughout
• Total parameters in non FC layers = $\sim 16M$
Kernel size is $3 \times 3$ throughout

Total parameters in non FC layers $= \sim 16M$

Total Parameters in FC layers $=$
• Kernel size is $3 \times 3$ throughout

• Total parameters in non FC layers $= \sim 16M$

• Total Parameters in FC layers $= (512 \times 7 \times 7 \times 4096)$
• Kernel size is $3 \times 3$ throughout
• Total parameters in non FC layers $= \sim 16M$
• Total Parameters in FC layers $= (512 \times 7 \times 7 \times 4096) + (4096 \times 4096)$
- Kernel size is $3 \times 3$ throughout
- Total parameters in non FC layers $= \sim 16M$
- Total Parameters in FC layers $= (512 \times 7 \times 7 \times 4096) + (4096 \times 4096) + (4096 \times 1024)$
- Kernel size is $3 \times 3$ throughout
- Total parameters in non FC layers = $\sim 16M$
- Total Parameters in FC layers = $(512 \times 7 \times 7 \times 4096) + (4096 \times 4096) + (4096 \times 1024) = \sim 122M$
- Kernel size is $3 \times 3$ throughout
- Total parameters in non FC layers = $\sim 16M$
- Total Parameters in FC layers = $(512 \times 7 \times 7 \times 4096) + (4096 \times 4096) + (4096 \times 1024) = \sim 122M$
- Most parameters are in the first FC layer ($\sim 102M$)
Module 11.5 : Image Classification continued (GoogLeNet and ResNet)
Consider the output at a certain layer of a convolutional neural network.
Consider the output at a certain layer of a convolutional neural network. After this layer, we could apply a max-pooling layer.

Question: Why choose between these options (convolution, max-pooling, filter sizes)?

Idea: Why not apply all of them at the same time and then concatenate the feature maps?
Consider the output at a certain layer of a convolutional neural network.

After this layer we could apply a max-pooling layer.

Or a $1 \times 1$ convolution.

Question: Why choose between these options (convolution, max-pooling, filter sizes)?

Idea: Why not apply all of them at the same time and then concatenate the feature maps?
Consider the output at a certain layer of a convolutional neural network.

After this layer we could apply a max-pooling layer.

Or a $1 \times 1$ convolution.

Or a $3 \times 3$ convolution.
Consider the output at a certain layer of a convolutional neural network. After this layer we could apply a max-pooling layer. Or a $1 \times 1$ convolution. Or a $3 \times 3$ convolution. Or a $5 \times 5$ convolution.
Consider the output at a certain layer of a convolutional neural network.

After this layer we could apply a max-pooling layer.

Or a $1 \times 1$ convolution.

Or a $3 \times 3$ convolution.

Or a $5 \times 5$ convolution.

**Question:** Why choose between these options (convolution, maxpooling, filter sizes)?
Consider the output at a certain layer of a convolutional neural network.

After this layer we could apply a max-pooling layer.

Or a $1 \times 1$ convolution.

Or a $3 \times 3$ convolution.

Or a $5 \times 5$ convolution.

**Question:** Why choose between these options (convolution, maxpooling, filter sizes)?

**Idea:** Why not apply all of them at the same time and then concatenate the feature maps?
Well this naive idea could result in a large number of computations.
- Well this naive idea could result in a large number of computations
- If $P = 0 \& S = 1$ then convolving a $W \times H \times D$ input with a $F \times F \times D$ filter results in a $(W - F + 1)(H - F + 1)$ sized output
Well this naive idea could result in a large number of computations

If $P = 0$ & $S = 1$ then convolving a $W \times H \times D$ input with a $F \times F \times D$ filter results in a $(W - F + 1)(H - F + 1)$ sized output

Each element of the output requires $O(F \times F \times D)$ computations
Well this naive idea could result in a large number of computations.

If $P = 0 \& S = 1$ then convolving a $W \times H \times D$ input with a $F \times F \times D$ filter results in a $(W - F + 1)(H - F + 1)$ sized output.

Each element of the output requires $O(F \times F \times D)$ computations.

Can we reduce the number of computations?
Yes, by using $1 \times 1$ convolutions
Yes, by using $1 \times 1$ convolutions

Huh?? What does a $1 \times 1$ convolution do?
- Yes, by using $1 \times 1$ convolutions
- Huh?? What does a $1 \times 1$ convolution do?
- It aggregates along the depth

Yes, by using $1 \times 1$ convolutions
Huh?? What does a $1 \times 1$ convolution do?
It aggregates along the depth
Yes, by using $1 \times 1$ convolutions

Huh?? What does a $1 \times 1$ convolution do?

It aggregates along the depth

So convolving a $D \times W \times H$ input with $D_1 \ 1 \times 1$ ($D_1 < D$) filters will result in a $D_1 \times W \times H$ output ($S = 1, P = 0$)
Yes, by using $1 \times 1$ convolutions

Huh?? What does a $1 \times 1$ convolution do?

It aggregates along the depth

So convolving a $D \times W \times H$ input with $D_1 1 \times 1$ ($D_1 < D$) filters will result in a $D_1 \times W \times H$ output ($S = 1, P = 0$)

If $D_1 < D$ then this effectively reduces the dimension of the input and hence the computations
Yes, by using $1 \times 1$ convolutions

Huh?? What does a $1 \times 1$ convolution do?

- It aggregates along the depth
- So convolving a $D \times W \times H$ input with $D_1 \ 1 \times 1$ ($D_1 < D$) filters will result in a $D_1 \times W \times H$ output ($S = 1, P = 0$)
- If $D_1 < D$ then this effectively reduces the dimension of the input and hence the computations
- Specifically instead of $O(F \times F \times D)$ we will need $O(F \times F \times D_1)$ computations
Yes, by using $1 \times 1$ convolutions

Huh?? What does a $1 \times 1$ convolution do?

It aggregates along the depth

So convolving a $D \times W \times H$ input with $D_1 \times 1 \times 1$ ($D_1 < D$) filters will result in a $D_1 \times W \times H$ output ($S = 1, P = 0$)

If $D_1 < D$ then this effectively reduces the dimension of the input and hence the computations

Specifically instead of $O(F \times F \times D)$ we will need $O(F \times F \times D_1)$ computations

We could then apply subsequent $3 \times 3$, $5 \times 5$ filter on this reduced output
But we might want to use different dimensionality reductions before the $3 \times 3$ and $5 \times 5$ filters
- But we might want to use different dimensionality reductions before the $3 \times 3$ and $5 \times 5$ filters
- So we can use $D_1$ and $D_2$ $1 \times 1$ filters before the $3 \times 3$ and $5 \times 5$ filters respectively
But we might want to use different dimensionality reductions before the 3 × 3 and 5 × 5 filters.

So we can use \( D_1 \) and \( D_2 \) 1 × 1 filters before the 3 × 3 and 5 × 5 filters respectively.

We can then add the maxpooling layer followed by dimensionality reduction.
But we might want to use different dimensionality reductions before the $3 \times 3$ and $5 \times 5$ filters.

So we can use $D_1$ and $D_2$ $1 \times 1$ filters before the $3 \times 3$ and $5 \times 5$ filters respectively.

We can then add the maxpooling layer followed by dimensionality reduction.

And a new set of $1 \times 1$ convolutions.
But we might want to use different dimensionality reductions before the $3 \times 3$ and $5 \times 5$ filters.

So we can use $D_1$ and $D_2$ $1 \times 1$ filters before the $3 \times 3$ and $5 \times 5$ filters respectively.

We can then add the maxpooling layer followed by dimensionality reduction.

And a new set of $1 \times 1$ convolutions.

And finally we concatenate all these layers.
- But we might want to use different dimensionality reductions before the $3 \times 3$ and $5 \times 5$ filters.
- So we can use $D_1$ and $D_2$ $1 \times 1$ filters before the $3 \times 3$ and $5 \times 5$ filters respectively.
- We can then add the maxpooling layer followed by dimensionality reduction.
- And a new set of $1 \times 1$ convolutions.
- And finally we concatenate all these layers.
- This is called the Inception module.
But we might want to use different dimensionality reductions before the $3 \times 3$ and $5 \times 5$ filters.

So we can use $D_1$ and $D_2$ $1 \times 1$ filters before the $3 \times 3$ and $5 \times 5$ filters respectively.

We can then add the maxpooling layer followed by dimensionality reduction.

And a new set of $1 \times 1$ convolutions.

And finally we concatenate all these layers.

This is called the **Inception module**.

We will now see *GoogLeNet* which contains many such inception modules.
Input

Conv 112 112 64
maxpool 56 56 64
Conv 56 56 192
maxpool 28 28 192
Inception 28 28 256
3a
Inception 28 28 480
3b
maxpool 14 14 480
Inception 14 14 512
4b
4c
Inception 14 14 528
4d
Inception 14 14 832
4e
maxpool 7 7 832
Inception 7 7 832
5a
Inception 7 7 1024
5b
avgpool 1 1 1024
dropout (40%)
1 1 1024
1000
fc
softmax
Input

Conv
maxpool
Conv

229
112
64
maxpool
56
56
64
Conv

56
56
192
maxpool
28
28
192
Inception

28
28
256
3a
Inception

28
28
480
3b
maxpool
14
14
480
Inception

14
14
512
4b
4c
Inception

14
14
528
4d
Inception

14
14
832
4e
maxpool
7
7
832
Inception

7
7
832
5a
Inception

7
7
1024
5b
avgpool
1
1
1024
dropout (40%)
1
1
1024
fc
1000
softmax
Mitesh M. Khapra  
CS7015 (Deep Learning) : Lecture 11
229
Conv
64
maxpool
56
Conv
3a
112
maxpool
28
Inception
3b
56
Inception
4a
192
maxpool
14
Inception
4b
Inception
4c
256
Inception
4d
832
maxpool
7
Inception
4e
832
Inception
5a
832
Inception
5b
avgpool
1
1000
fc
1000
softmax
1
96
1 × 1 convolutions (dimensionality reduction)
128
3 × 3 convolutions (on reduced input)
32
5 × 5 convolutions (on reduced input)
32
1 × 1 convolutions
Filter concatenation
16
1 × 1 convolutions (dimensionality reduction)
3 × 3 Maxpooling (dimensionality reduction)
64
1 × 1 convolutions
128
3 × 3 convolutions (on reduced input)
32
5 × 5 convolutions (on reduced input)
32
1 × 1 convolutions
16
1 × 1 convolutions (dimensionality reduction)
3 × 3 Maxpooling (dimensionality reduction)
64
1 × 1 convolutions
128
3 × 3 convolutions (on reduced input)
32
5 × 5 convolutions (on reduced input)
32
1 × 1 convolutions
Filter concatenation

192 1 × 1 convolutions
208 3 × 3 convolutions (on reduced input)
18 5 × 5 convolutions (on reduced input)
64 1 × 1 convolutions

96 1 × 1 convolutions (dimensionality reduction)
16 1 × 1 convolutions (dimensionality reduction)
3 × 3 Maxpooling (dimensionality reduction)
112 $1 \times 1$ convolutions (dimensionality reduction)

224 $3 \times 3$ convolutions (on reduced input)

64 $5 \times 5$ convolutions (on reduced input)

3 $3 \times 3$ Maxpooling (dimensionality reduction)

64 $1 \times 1$ convolutions

Filter concatenation

160 $1 \times 1$ convolutions
Input
Conv
64
maxpool
Conv
112
3a
8
maxpool
Inception
192
3b
8
maxpool
64
Conv
56
maxpool
192
Inception
56
192
maxpool
28
Inception
28
256
Inception
28
Inception
28
480
Inception
28
Inception
28
512
maxpool
14
Inception
14
528
Inception
14
Inception
14
832
maxpool
7
Inception
7
832
Inception
7
Inception
7
1024
avgpool
1
Inception
1
dropout
(avgpool)
1
Inception
1
1024
fc
1000
softmax
1000
128
1 × 1 convolutions (dimensionality reduction)
256
3 × 3 convolutions (on reduced input)
64
5 × 5 convolutions (on reduced input)
64
1 × 1 convolutions
Filter concatenation
128
3 × 3 convolutions (dimensionality reduction)
24
1 × 1 convolutions (dimensionality reduction)
3 × 3 Maxpooling (dimensionality reduction)
Input

229

Conv

112

maxpool

56

Conv

56

maxpool

28

Inception

28

Inception

28

Inception

28

maxpool

14

Inception

14

Inception

14

avgpool

1

dropout (40%)

1000

fc

softmax

1

144 1x1 convolutions (dimensionality reduction)

112 1x1 convolutions

288 3x3 convolutions (on reduced input)

64 5x5 convolutions (on reduced input)

3 x 3 Maxpooling (dimensionality reduction)

Filter concatenation

32 1x1 convolutions (dimensionality reduction)

64 1x1 convolutions
Mitesh M. Khapra

CS7015 (Deep Learning) : Lecture 11
Input

229

Conv

112

maxpool

56

64

maxpoolConv

56

192

maxpoolInception

56

384

480

192

Conv

112

maxpool

56

64

maxpoolConv

56

192

maxpoolInception

56

384

480

192

Inception

28

256

3a

Inception

28

256

480

3b

maxpool

14

128

512

4a

maxpool

14

128

832

4b

Inception

14

128

832

4c

maxpool

14

128

832

4d

Inception

14

128

832

4e

Inception

14

128

832

5a

maxpool

14

128

832

5b

Inception

14

128

832

5c

avgpool

1

1024

dropout(40%)

1

1024

fc

1000

softmax

192

1 × 1 convolutions (dimensionality reduction)

384

3 × 3 convolutions (on reduced input)

128

5 × 5 convolutions (on reduced input)

48

1 × 1 convolutions (dimensionality reduction)

3 × 3 Maxpooling (dimensionality reduction)

128

1 × 1 convolutions

Filter concatenation

Mitesh M. Khapra
CS7015 (Deep Learning) : Lecture 11
Input

Conv

maxpool

Inception

maxpool

Inception

maxpool

Inception

maxpool

Inception

maxpool

Inception

avgpool

Mitesh M. Khapra

CS7015 (Deep Learning) : Lecture 11
Input

maxpool

Conv

maxpool

Conv

maxpool

Inception

maxpool

Inception

maxpool

Inception

maxpool

avgpool

dropout (40%)

fc

softmax

Mitesh M. Khapra

CS7015 (Deep Learning) : Lecture 11
Important Trick: Got rid of the fully connected layer

Notice that output of the last layer is $7 \times 7 \times 1024$ dimensional. What if we were to add a fully connected layer with 1000 nodes (for 1000 classes) on top of this. We would have $7 \times 7 \times 1024 \times 1000 = 49$ M parameters. Instead they use an average pooling of size $7 \times 7$ on each of the 1024 feature maps. This results in a 1024 dimensional output. Significantly reduces the number of parameters.
• **Important Trick:** Got rid of the fully connected layer

• Notice that output of the last layer is $7 \times 7 \times 1024$ dimensional

\[ W \in \mathbb{R}^{50176 \times 1000} \]

\[ W \in \mathbb{R}^{1024 \times 1000} \]

flatten

```
61/68
```
- Important Trick: Got rid of the fully connected layer
- Notice that output of the last layer is $7 \times 7 \times 1024$ dimensional
- What if we were to add a fully connected layer with 1000 nodes (for 1000 classes) on top of this
- **Important Trick:** Got rid of the fully connected layer
- Notice that output of the last layer is $7 \times 7 \times 1024$ dimensional
- What if we were to add a fully connected layer with 1000 nodes (for 1000 classes) on top of this
- We would have $7 \times 7 \times 1024 \times 1000 = 49M$ parameters
**Important Trick:** Got rid of the fully connected layer

- Notice that output of the last layer is $7 \times 7 \times 1024$ dimensional.

- What if we were to add a fully connected layer with 1000 nodes (for 1000 classes) on top of this?

- We would have $7 \times 7 \times 1024 \times 1000 = 49M$ parameters.

- Instead they use an average pooling of size $7 \times 7$ on each of the 1024 feature maps.
Important Trick: Got rid of the fully connected layer

- Notice that output of the last layer is $7 \times 7 \times 1024$ dimensional
- What if we were to add a fully connected layer with 1000 nodes (for 1000 classes) on top of this
- We would have $7 \times 7 \times 1024 \times 1000 = 49M$ parameters
- Instead they use an average pooling of size $7 \times 7$ on each of the 1024 feature maps
- This results in a 1024 dimensional output
Important Trick: Got rid of the fully connected layer

Notice that output of the last layer is $7 \times 7 \times 1024$ dimensional

What if we were to add a fully connected layer with 1000 nodes (for 1000 classes) on top of this

We would have $7 \times 7 \times 1024 \times 1000 = 49M$ parameters

Instead they use an average pooling of size $7 \times 7$ on each of the 1024 feature maps

This results in a 1024 dimensional output

Significantly reduces the number of parameters
12× less parameters than AlexNet
- $12 \times$ less parameters than AlexNet
- $2 \times$ more computations
- GoogLeNet
- ResNet
Suppose we have been able to train a shallow neural network well.
Suppose we have been able to train a shallow neural network well.

Now suppose we construct a deeper network which has few more layers (in orange).
Suppose we have been able to train a shallow neural network well.

Now suppose we construct a deeper network which has few more layers (in orange).

Intuitively, if the shallow network works well then the deep network should also work well by simply learning to compute identity functions in the new layers.

Essentially, the solution space of a shallow neural network is a subset of the solution space of a deep neural network.
Suppose we have been able to train a shallow neural network well.

Now suppose we construct a deeper network which has few more layers (in orange).

Intuitively, if the shallow network works well then the deep network should also work well by simply learning to compute identity functions in the new layers.

Essentially, the solution space of a shallow neural network is a subset of the solution space of a deep neural network.
But in practice it is observed that this doesn’t happen
- But in practice it is observed that this doesn’t happen
- Notice that the deep layers have a higher error rate on the test set
Consider any two stacked layers in a CNN.
Consider any two stacked layers in a CNN

The two layers are essentially learning some function of the input
Consider any two stacked layers in a CNN.

The two layers are essentially learning some function of the input.

What if we enable it to learn only a residual function of the input?

\[
H(x) = F(x) + x
\]
Why would this help?

Remember our argument that a deeper version of a shallow network would do just fine by learning identity transformations in the new layers. This identity connection from the input allows a ResNet to retain a copy of the input. Using this idea, they were able to train really deep networks.
Why would this help?
- Remember our argument that a deeper version of a shallow network would do just fine by learning identity transformations in the new layers.
Why would this help?

- Remember our argument that a deeper version of a shallow network would do just fine by learning identity transformations in the new layers.
- This identity connection from the input allows a ResNet to retain a copy of the input.
Why would this help?
- Remember our argument that a deeper version of a shallow network would do just fine by learning identity transformations in the new layers.
- This identity connection from the input allows a ResNet to retain a copy of the input.
- Using this idea they were able to train really deep networks.
ResNet, 152 layers

1st place in all five main tracks

- **ImageNet Classification**: “Ultra-deep” 152-layer nets
ResNet, 152 layers

1st place in all five main tracks

- **ImageNet Classification**: “Ultra-deep” 152-layer nets
- **ImageNet Detection**: 16% better than the 2nd best system
ResNet, 152 layers

1<sup>st</sup> place in all five main tracks

- **ImageNet Classification**: “Ultra-deep” 152-layer nets
- **ImageNet Detection**: 16% better than the 2nd best system
- **ImageNet Localization**: 27% better than the 2nd best system
ResNet, 152 layers

1st place in all five main tracks

- **ImageNet Classification**: “Ultra-deep” 152-layer nets
- **ImageNet Detection**: 16% better than the 2nd best system
- **ImageNet Localization**: 27% better than the 2nd best system
- **COCO Detection**: 11% better than the 2nd best system
ResNet, 152 layers

1\textsuperscript{st} place in all five main tracks

- **ImageNet Classification:** “Ultra-deep” 152-layer nets
- **ImageNet Detection:** 16% better than the 2nd best system
- **ImageNet Localization:** 27% better than the 2nd best system
- **COCO Detection:** 11% better than the 2nd best system
- **COCO Segmentation:** 12% better than the 2nd best system
Bag of tricks

- Batch Normalization after every CONV layer
ResNet, 152 layers

### Bag of tricks

- Batch Normalization after every CONV layer
- Xavier/2 initialization from [He et al]
ResNet, 152 layers

Bag of tricks

- Batch Normalization after every CONV layer
- Xavier/2 initialization from [He et al]
- SGD + Momentum(0.9)
ResNet, 152 layers

---

**Bag of tricks**

- Batch Normalization after every CONV layer
- Xavier/2 initialization from [He et al]
- SGD + Momentum(0.9)
- Learning rate: 0.1, divided by 10 when validation error plateaus
ResNet, 152 layers

Bag of tricks
- Batch Normalization after every CONV layer
- Xavier/2 initialization from [He et al]
- SGD + Momentum(0.9)
- Learning rate: 0.1, divided by 10 when validation error plateaus
- Mini-batch size 256
ResNet, 152 layers

**Bag of tricks**

- Batch Normalization after every CONV layer
- Xavier/2 initialization from [He et al]
- SGD + Momentum(0.9)
- Learning rate: 0.1, divided by 10 when validation error plateaus
- Mini-batch size 256
- Weight decay of 1e-5
Bag of tricks

- Batch Normalization after every CONV layer
- Xavier/2 initialization from [He et al]
- SGD + Momentum(0.9)
- Learning rate: 0.1, divided by 10 when validation error plateaus
- Mini-batch size 256
- Weight decay of 1e-5
- No dropout used

ResNet, 152 layers