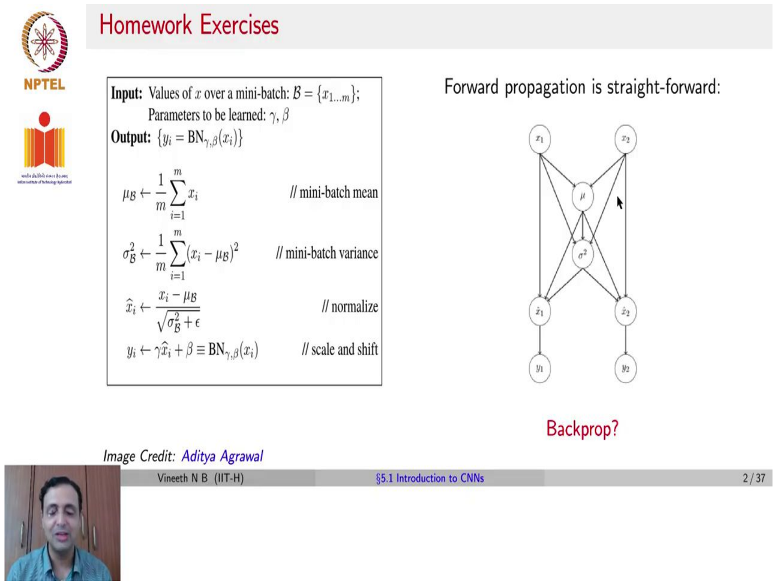
Deep Learning for Computer Vision
Professor. Vineeth N Balasubramanian
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad
Lecture 31
Convolutional Neural Networks: An Introduction Part 01

Continuing our discussion, we will now move on from simple feed forward neural networks into the building blocks of deep learning for computer vision, which are Convolution Neural Networks.

(Refer Slide Time: 00:35)



Homework Exercises

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \text{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Forward propagation is straight-forward:

Backprop?

Image Credit: *Aditya Agrawal*

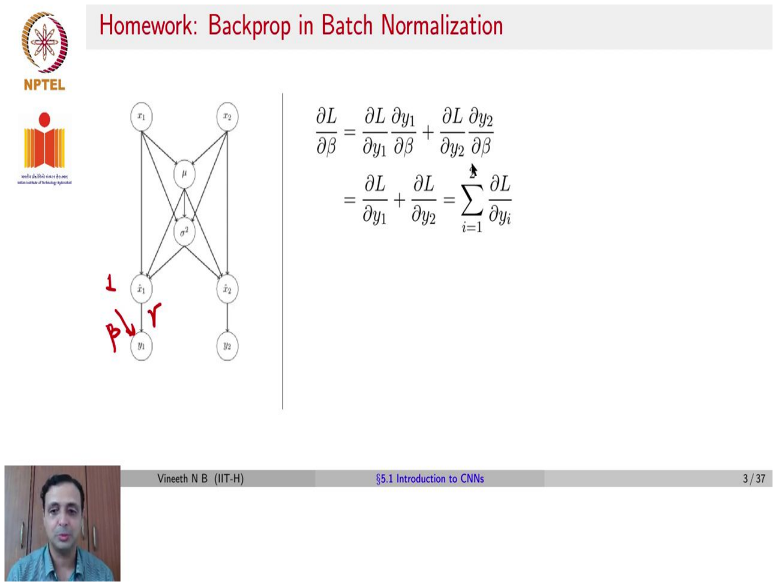
Vineeth N B (IIT-H) §5.1 Introduction to CNNs 2 / 37

Before we start that discussion, we will review the homework or the exercise that we left behind in the last lecture of the previous week. One of the questions there was, how do you backprop across the batch normalization layer. So, what you see here is the algorithm of batch normalization that we saw last time. Let us quickly review that in terms of how it looks on this graph, what is known as the computational graph. So, you have forward propagation which is given by, you are given x1 and x2.

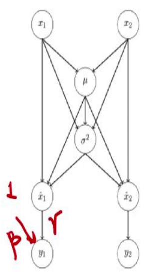
And given a mini batch of data, you find the mean and variance of x the inputs out of a particular layer, out of a particular layer the activations out of a particular layer. And then you normalize that data using the mean and variance. Once you normalize, you renormalize them by scaling and shifting, rather you adopt a variance γ, and a mean β, which is suitable to solve that problem.

This is the main idea of batch normalization and that is represented by this graph here. The forward prop as we can see, is quite straightforward and we just follow this algorithm. The question here is, how do we do the backprop. Let us try to work through the different terms here and understand this.

(Refer Slide Time: 02:04)

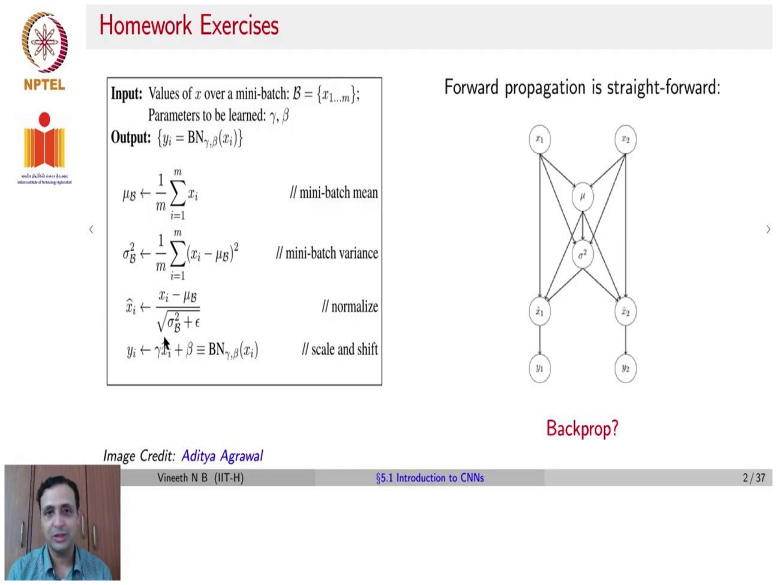


Homework: Backprop in Batch Normalization



$$\frac{\partial L}{\partial \beta} = \frac{\partial L}{\partial y_1}\frac{\partial y_1}{\partial \beta} + \frac{\partial L}{\partial y_2}\frac{\partial y_2}{\partial \beta}$$
$$= \frac{\partial L}{\partial y_1} + \frac{\partial L}{\partial y_2} = \sum_{i=1} \frac{\partial L}{\partial y_i}$$

Vineeth N B (IIT-H) §5.1 Introduction to CNNs 3 / 37

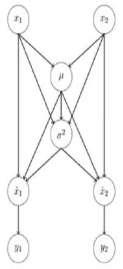


Homework Exercises

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \text{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$
$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$
$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$
$$y_i \leftarrow \gamma\hat{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

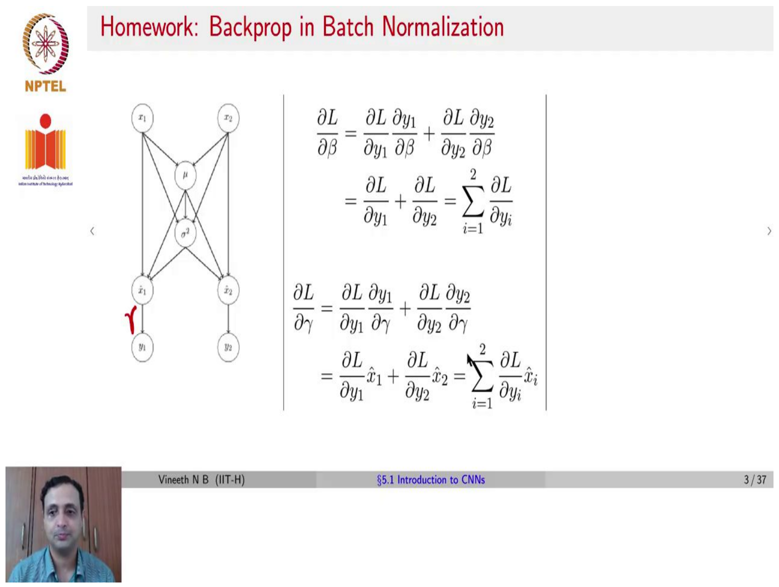
Forward propagation is straight-forward:



Backprop?

Image Credit: *Aditya Agrawal*

Vineeth N B (IIT-H) §5.1 Introduction to CNNs 2 / 37

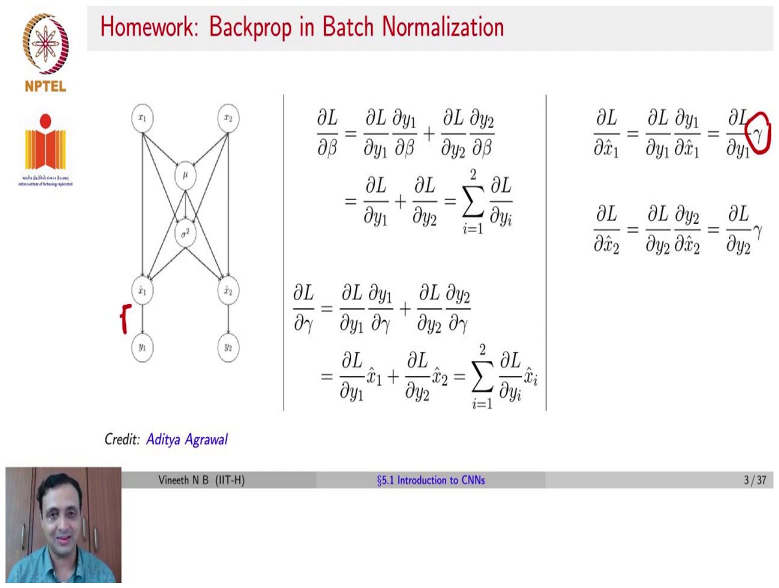


Homework: Backprop in Batch Normalization

$$\frac{\partial L}{\partial \beta} = \frac{\partial L}{\partial y_1}\frac{\partial y_1}{\partial \beta} + \frac{\partial L}{\partial y_2}\frac{\partial y_2}{\partial \beta}$$
$$= \frac{\partial L}{\partial y_1} + \frac{\partial L}{\partial y_2} = \sum_{i=1}^{2}\frac{\partial L}{\partial y_i}$$

$$\frac{\partial L}{\partial \gamma} = \frac{\partial L}{\partial y_1}\frac{\partial y_1}{\partial \gamma} + \frac{\partial L}{\partial y_2}\frac{\partial y_2}{\partial \gamma}$$
$$= \frac{\partial L}{\partial y_1}\hat{x}_1 + \frac{\partial L}{\partial y_2}\hat{x}_2 = \sum_{i=1}^{2}\frac{\partial L}{\partial y_i}\hat{x}_i$$

Vineeth N B (IIT-H) §5.1 Introduction to CNNs 3 / 37



Homework: Backprop in Batch Normalization

$$\frac{\partial L}{\partial \beta} = \frac{\partial L}{\partial y_1}\frac{\partial y_1}{\partial \beta} + \frac{\partial L}{\partial y_2}\frac{\partial y_2}{\partial \beta}$$
$$= \frac{\partial L}{\partial y_1} + \frac{\partial L}{\partial y_2} = \sum_{i=1}^{2}\frac{\partial L}{\partial y_i}$$

$$\frac{\partial L}{\partial \gamma} = \frac{\partial L}{\partial y_1}\frac{\partial y_1}{\partial \gamma} + \frac{\partial L}{\partial y_2}\frac{\partial y_2}{\partial \gamma}$$
$$= \frac{\partial L}{\partial y_1}\hat{x}_1 + \frac{\partial L}{\partial y_2}\hat{x}_2 = \sum_{i=1}^{2}\frac{\partial L}{\partial y_i}\hat{x}_i$$

$$\frac{\partial L}{\partial \hat{x}_1} = \frac{\partial L}{\partial y_1}\frac{\partial y_1}{\partial \hat{x}_1} = \frac{\partial L}{\partial y_1}\gamma$$

$$\frac{\partial L}{\partial \hat{x}_2} = \frac{\partial L}{\partial y_2}\frac{\partial y_2}{\partial \hat{x}_2} = \frac{\partial L}{\partial y_2}\gamma$$

Credit: Aditya Agrawal

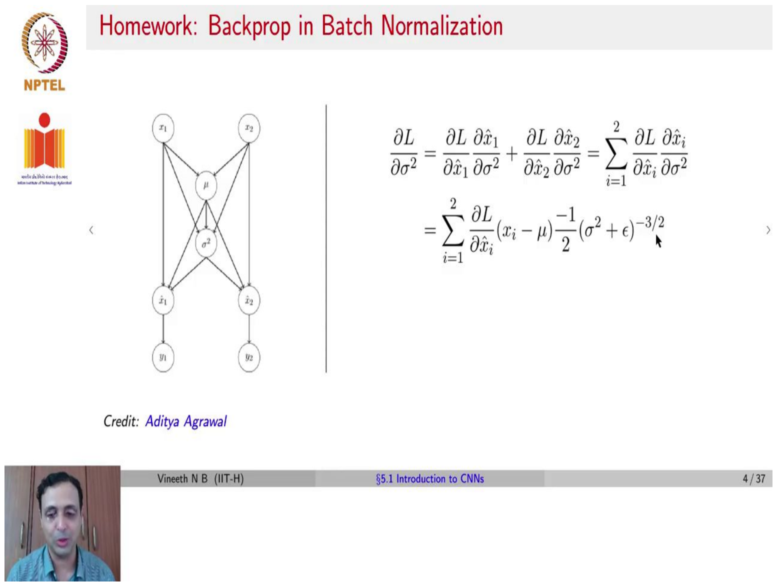
Vineeth N B (IIT-H) §5.1 Introduction to CNNs 3 / 37

The first term that we have to find out is β That is one of the simplest terms to evaluate so we will start with that. To recall, let us go back and see what β was. Note here, that in the last line of the algorithm, $y_i = \beta + \gamma\ \hat{x}_i$. Keeping that in mind, so this is the main equation that you may want to keep in mind for the next few slides. So, you can assume that β is almost an input into y with, so you have this here, $y_1 = \gamma\hat{x}_i + \beta.1$. So, So, $\frac{\partial L}{\partial \beta} = \frac{\partial L}{\partial y_1}\frac{\partial y_1}{\partial \beta} + \frac{\partial L}{\partial y_2}\frac{\partial y_2}{\partial \beta}$. $\frac{\partial y_1}{\partial \beta}$ as you can see here is going to be 1, so this is going to simplify to $\frac{\partial L}{\partial y_1} + \frac{\partial L}{\partial y_2}$, which is the final $\frac{\partial L}{\partial \beta}$.

Let us move on to the next one. Similarly, $\frac{\partial L}{\partial \gamma}$, remember γ is the constant that multiplies $\hat{x_i}$. So, $\frac{\partial L}{\partial \gamma} = \frac{\partial L}{\partial y_1}\frac{\partial y_1}{\partial \gamma} + \frac{\partial L}{\partial y_2}\frac{\partial y_2}{\partial \gamma}$ and so on and so forth for all the y terms. And $\frac{\partial y_1}{\partial \gamma}$ as you can see is going to be $\hat{x_1}$ by the very definition of these terms. So, that term gets replaced by $\hat{x_1}$. So, you have $\frac{\partial L}{\partial y_1}\hat{x_1} + \frac{\partial L}{\partial y_2}\hat{x_2}$ and that gets summarized as a summation in this way.

Let us move on to the next term. The next term is $\frac{\partial L}{\partial \hat{x_1}}$, x1 is not a parameter, but we need to be able to compute this so as to propagate the gradient to previous parameters in the network. $\frac{\partial L}{\partial \hat{x_1}} = \frac{\partial L}{\partial y_1}\frac{\partial y_1}{\partial \hat{x_1}}$, $\frac{\partial y_1}{\partial \hat{x_1}} = \gamma$ as we just saw, that is what we put in here. And similarly, you would be able to calculate $\frac{\partial y_1}{\partial \hat{x_2}}$.

(Refer Slide Time: 04:45)



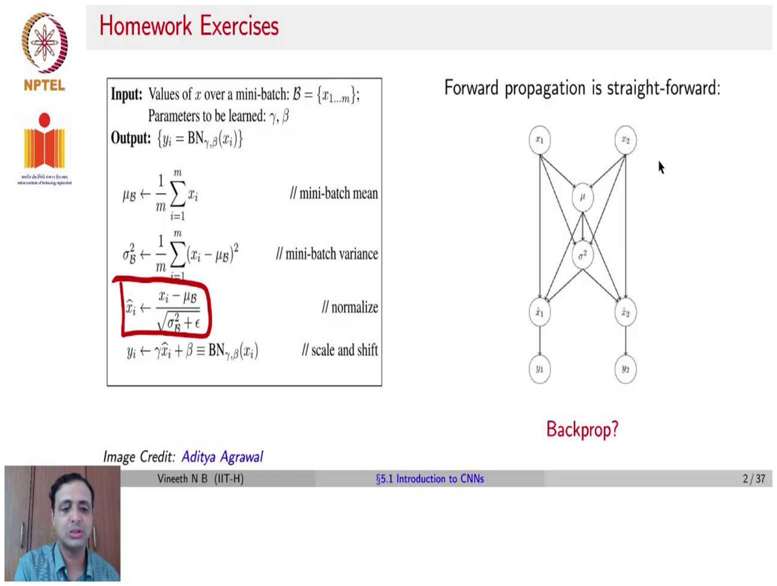
Homework: Backprop in Batch Normalization

$$\frac{\partial L}{\partial \sigma^2} = \frac{\partial L}{\partial \hat{x}_1}\frac{\partial \hat{x}_1}{\partial \sigma^2} + \frac{\partial L}{\partial \hat{x}_2}\frac{\partial \hat{x}_2}{\partial \sigma^2} = \sum_{i=1}^{2}\frac{\partial L}{\partial \hat{x}_i}\frac{\partial \hat{x}_i}{\partial \sigma^2}$$

$$= \sum_{i=1}^{2}\frac{\partial L}{\partial \hat{x}_i}(x_i - \mu)\frac{-1}{2}(\sigma^2 + \epsilon)^{-3/2}$$

Credit: Aditya Agrawal

Vineeth N B (IIT-H) §5.1 Introduction to CNNs 4 / 37



Homework Exercises

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \text{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_\mathcal{B} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_\mathcal{B}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_\mathcal{B})^2 \qquad \text{// mini-batch variance}$$

$$\boxed{\widehat{x}_i \leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Forward propagation is straight-forward:

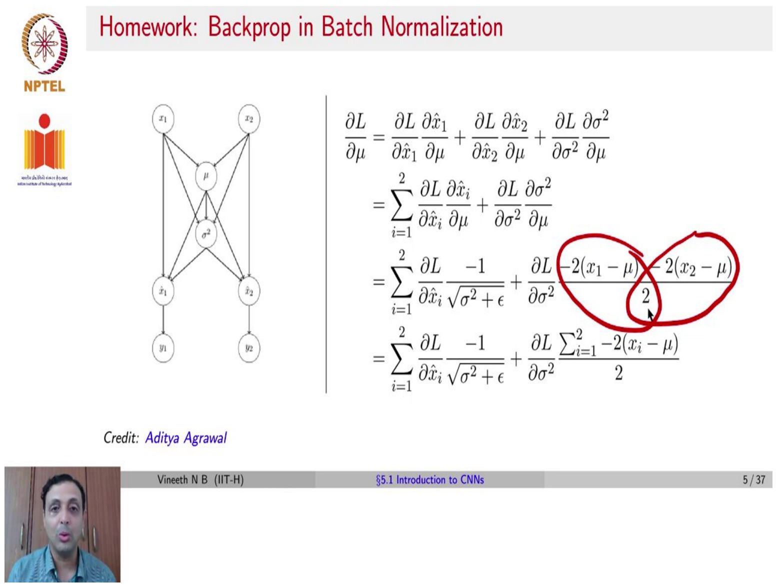
Backprop?

Image Credit: Aditya Agrawal

Let us move on to the next term, which is going to be $\frac{\partial L}{\partial \sigma^2}$. We again need this because the gradient needs to pass through this to be able to go to $\frac{\partial L}{\partial x}$ which is going to be an earlier term and all the other gradients in the previous layers. $\frac{\partial L}{\partial \sigma^2}$ is given by $\frac{\partial L}{\partial \hat{x}_1}$, which we already computed on the previous slide into $\frac{\partial \hat{x}_1}{\partial \sigma^2}$, which is the new term that we need to compute.

Similarly, we need to do this for all the x terms. So, you have $\frac{\partial L}{\partial \hat{x}_2}\frac{\partial \hat{x}_2}{\partial \sigma^2}$ and it is written as a summation. The main term that we need to evaluate here is $\frac{\partial \hat{x}_i}{\partial \sigma^2}$. So, which is formed by the gradient of the, form of x y hat itself. Let us go back a couple of slides and see what that is. So, $\hat{x}_i$ is given by this term here. So, the gradient of $\frac{\partial \hat{x}_i}{\partial \sigma^2}$ just follows the gradient of this particular term which can be written by $(x_i - \mu)\frac{-1}{2}(\sigma^2 + \varepsilon)^{-3/2}$.

(Refer Slide Time: 06:05)

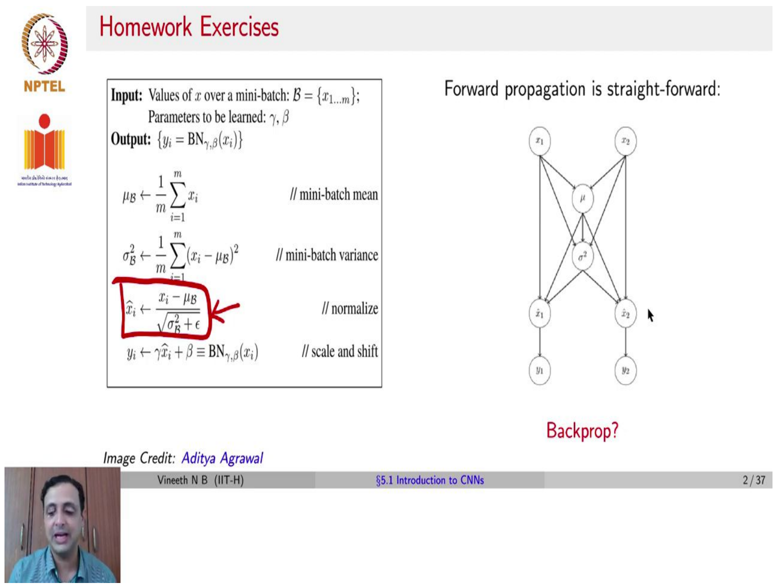


Homework: Backprop in Batch Normalization

$$\begin{aligned}\frac{\partial L}{\partial \mu} &= \frac{\partial L}{\partial \hat{x}_1}\frac{\partial \hat{x}_1}{\partial \mu} + \frac{\partial L}{\partial \hat{x}_2}\frac{\partial \hat{x}_2}{\partial \mu} + \frac{\partial L}{\partial \sigma^2}\frac{\partial \sigma^2}{\partial \mu} \\ &= \sum_{i=1}^{2}\frac{\partial L}{\partial \hat{x}_i}\frac{\partial \hat{x}_i}{\partial \mu} + \frac{\partial L}{\partial \sigma^2}\frac{\partial \sigma^2}{\partial \mu} \\ &= \sum_{i=1}^{2}\frac{\partial L}{\partial \hat{x}_i}\frac{-1}{\sqrt{\sigma^2+\epsilon}} + \frac{\partial L}{\partial \sigma^2}\frac{-2(x_1-\mu) - 2(x_2-\mu)}{2} \\ &= \sum_{i=1}^{2}\frac{\partial L}{\partial \hat{x}_i}\frac{-1}{\sqrt{\sigma^2+\epsilon}} + \frac{\partial L}{\partial \sigma^2}\frac{\sum_{i=1}^{2} -2(x_i-\mu)}{2}\end{aligned}$$

Credit: Aditya Agrawal

Vineeth N B (IIT-H) §5.1 Introduction to CNNs 5 / 37



Homework Exercises

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \text{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\boxed{\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\hat{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Forward propagation is straight-forward:

Backprop?

Image Credit: Aditya Agrawal

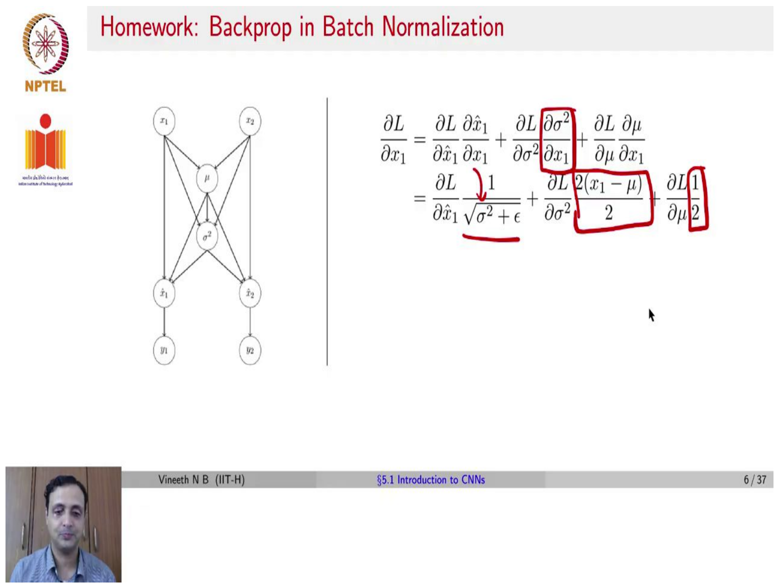
Vineeth N B (IIT-H) §5.1 Introduction to CNNs 2 / 37

Similarly, we have $\frac{\partial L}{\partial \mu} = \frac{\partial L}{\partial \hat{x_1}}\frac{\partial \hat{x_1}}{\partial \mu} + \frac{\partial L}{\partial \hat{x_2}}\frac{\partial \hat{x_2}}{\partial \mu} + \frac{\partial L}{\partial \sigma^2}\frac{\partial \sigma^2}{\partial \mu}$ So, the first two terms get absorbed into a common summation and the last term stays as it is. $\frac{\partial \hat{x_i}}{\partial \mu}$ is given by this gradient here; this once again follows directly from the definition of $\hat{x_i}$ to be $(x_i - \mu)/\sqrt{\sigma^2 + \varepsilon}$. And the term here $\frac{\partial \sigma^2}{\partial \mu}$ gets written in terms of x's and μ's.

Because σ^2, once again by the same expression can be rewritten in terms of all the x's and the corresponding μ, which is to repeat that. Remember that if we go back to the algorithm from the same boxed equation here, you could take sigma b squared + epsilon the other side and get xi hat as the denominator and you would do this for $\hat{x_1}$, $\hat{x_2}$ and all of that.

So, doh sigma doh μ by sigma square becomes a summation of all of these gradients and that is what we get here. Each of this is one gradient with respect to one of those inputs x1. Similarly, this would be the gradient with respect to x2, and so on and so forth. And that is how you get your doh L by doh μ.

(Refer Slide Time: 07:47)

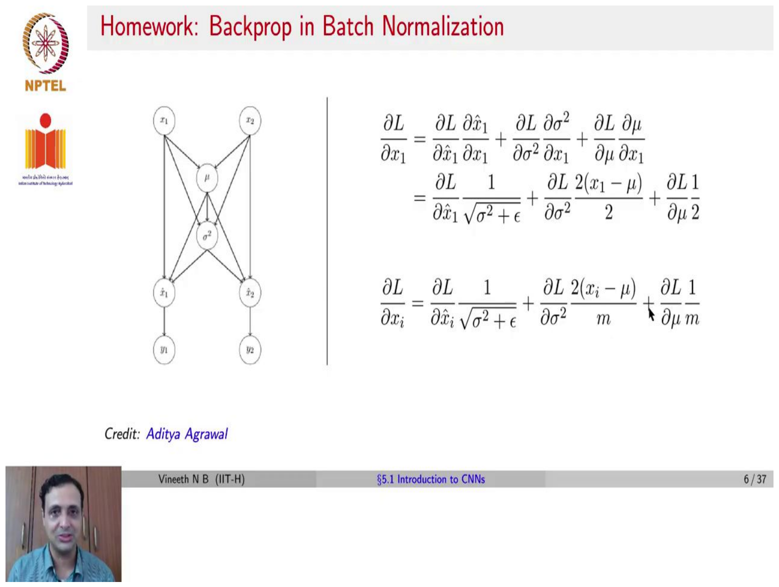


Homework: Backprop in Batch Normalization

x_1
x_2
μ
σ^2
$\hat{x}_1$
$\hat{x}_2$
y_1
y_2

$$\frac{\partial L}{\partial x_1} = \frac{\partial L}{\partial \hat{x}_1}\frac{\partial \hat{x}_1}{\partial x_1} + \frac{\partial L}{\partial \sigma^2}\boxed{\frac{\partial \sigma^2}{\partial x_1}} + \frac{\partial L}{\partial \mu}\frac{\partial \mu}{\partial x_1}$$
$$= \frac{\partial L}{\partial \hat{x}_1}\underline{\frac{1}{\sqrt{\sigma^2+\epsilon}}} + \frac{\partial L}{\partial \sigma^2}\boxed{\frac{2(x_1-\mu)}{2}} + \frac{\partial L}{\partial \mu}\boxed{\frac{1}{2}}$$

Vineeth N B (IIT-H) §5.1 Introduction to CNNs 6 / 37



Homework: Backprop in Batch Normalization

x_1
x_2
μ
σ^2
$\hat{x}_1$
$\hat{x}_2$
y_1
y_2

$$\frac{\partial L}{\partial x_1} = \frac{\partial L}{\partial \hat{x}_1}\frac{\partial \hat{x}_1}{\partial x_1} + \frac{\partial L}{\partial \sigma^2}\frac{\partial \sigma^2}{\partial x_1} + \frac{\partial L}{\partial \mu}\frac{\partial \mu}{\partial x_1}$$
$$= \frac{\partial L}{\partial \hat{x}_1}\frac{1}{\sqrt{\sigma^2+\epsilon}} + \frac{\partial L}{\partial \sigma^2}\frac{2(x_1-\mu)}{2} + \frac{\partial L}{\partial \mu}\frac{1}{2}$$

$$\frac{\partial L}{\partial x_i} = \frac{\partial L}{\partial \hat{x}_i}\frac{1}{\sqrt{\sigma^2+\epsilon}} + \frac{\partial L}{\partial \sigma^2}\frac{2(x_i-\mu)}{m} + \frac{\partial L}{\partial \mu}\frac{1}{m}$$

Credit: Aditya Agrawal

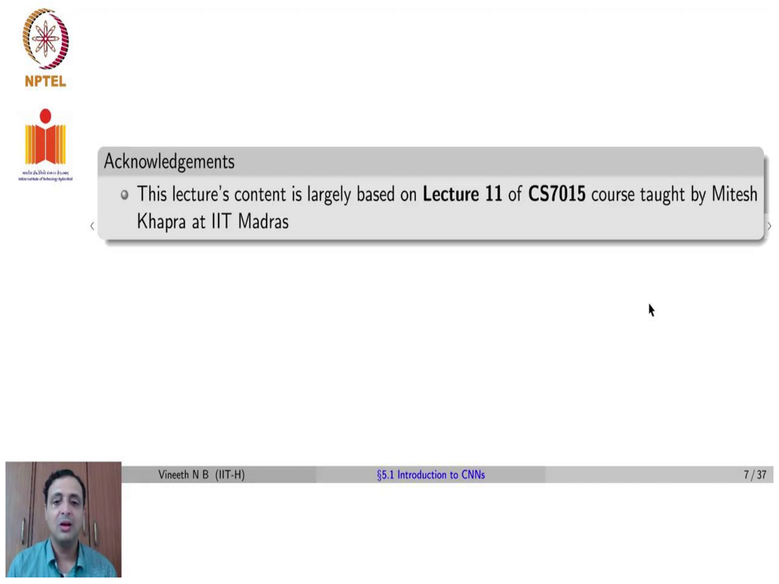
Vineeth N B (IIT-H) §5.1 Introduction to CNNs 6 / 37

Finally $\partial L/\partial x_1$, that is the final gradient that you need to be able to compute to back propagate to earlier layers, that is going to be $\partial L/\partial \hat{x_1} . \partial \hat{x_1}/\partial x_1 + \partial L/\partial \sigma^2 . \partial \sigma^2/\partial x_1 + \partial L/\partial \mu . \partial \mu/\partial x_1$. So, this one follows as it is $\partial \hat{x_1}/\partial x_1$ by once again the definition of $\hat{x_1}$, you will get it to be this term.

$\partial \sigma^2/\partial x_1$, once again, by the expansion of sigma with respect to x1. Now, you do not need to worry about the x_1 term. So, you will have only one gradient term here, you get this. And finally, $\partial \mu/\partial x_1$ would simply be 1/2 here, by definition of x1 by definition was μ and how it is connected to x1.

Similarly, you would get this for $\partial L/\partial x_1$, and so on and so forth for all the doh L by doh xi's. That is how the backprop is done for batch normalization. A takeaway for you at this stage is, for any of these new parameters that you introduce in a neural network, a careful working out of chain rule for every term involved would take care of, would take care of how you should be able to compute the gradients and back propagate in that scenario. All that you need to ensure that you have only differentiable components in all your new additions to a neural network.

(Refer Slide Time: 09:26)



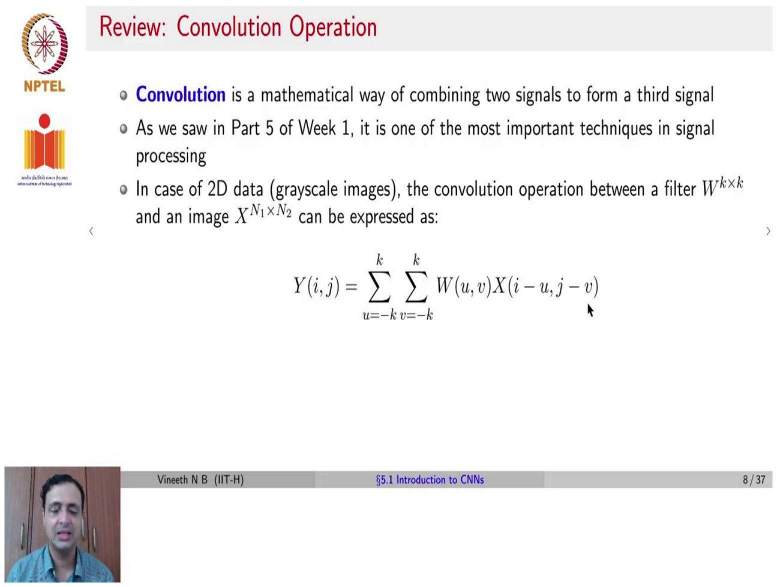
Acknowledgements

- This lecture's content is largely based on **Lecture 11** of **CS7015** course taught by Mitesh Khapra at IIT Madras

Vineeth N B (IIT-H) §5.1 Introduction to CNNs 7 / 37

With that, let us move on to convolutional neural networks. And this lecture is largely based on the excellent lectures of Mitesh Khapra, from IIT Madras.

(Refer Slide Time: 09:37)



Review: Convolution Operation

- **Convolution** is a mathematical way of combining two signals to form a third signal
- As we saw in Part 5 of Week 1, it is one of the most important techniques in signal processing
- In case of 2D data (grayscale images), the convolution operation between a filter $W^{k \times k}$ and an image $X^{N_1 \times N_2}$ can be expressed as:

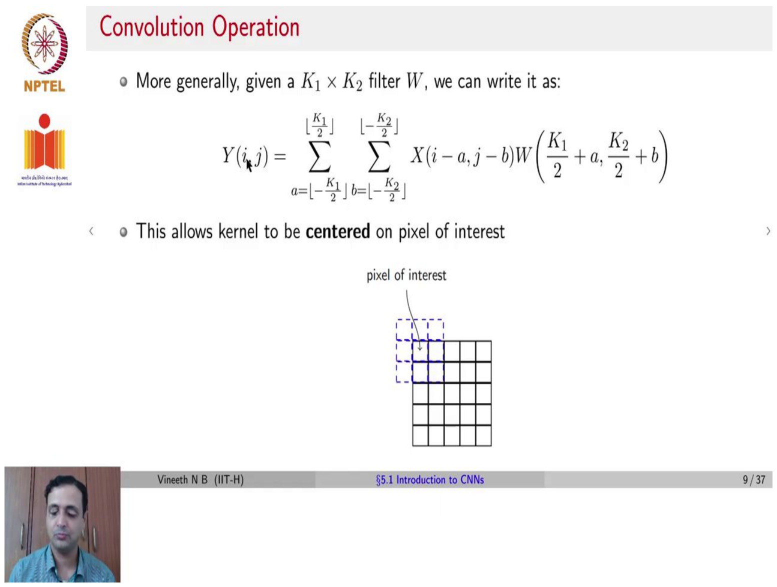
$$Y(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} W(u,v)X(i-u, j-v)$$

Vineeth N B (IIT-H) §5.1 Introduction to CNNs 8 / 37

Let us start with reviewing convolution. So, we did do it in the first week of this course. We know now that convolution is a mathematical way of combining two signals to form a third signal. We also know that it is an extremely important technique in signal processing in general, images are 2D signals. But convolution is also used for other kinds of signals such as speech or accelerometer data, time series data, so on and so forth.

In case of 2D data, let us assume the simple case of grayscale images, convolution operation between a filter W, whose dimension is k x k, and an image X, whose dimension is n1 x n2 is expressed as y of yj index u going from -k to k, index v going from -k to k, W u, v, which is the filter an X i-u, j-k, which is the input image. We have already seen this as the definition of convolution.

(Refer Slide Time: 10:46)



Convolution Operation

- More generally, given a $K_1 \times K_2$ filter W, we can write it as:

$$Y(i,j) = \sum_{a=\lfloor -\frac{K_1}{2} \rfloor}^{\lfloor \frac{K_1}{2} \rfloor} \sum_{b=\lfloor -\frac{K_2}{2} \rfloor}^{\lfloor -\frac{K_2}{2} \rfloor} X(i-a, j-b) W\left(\frac{K_1}{2}+a, \frac{K_2}{2}+b\right)$$

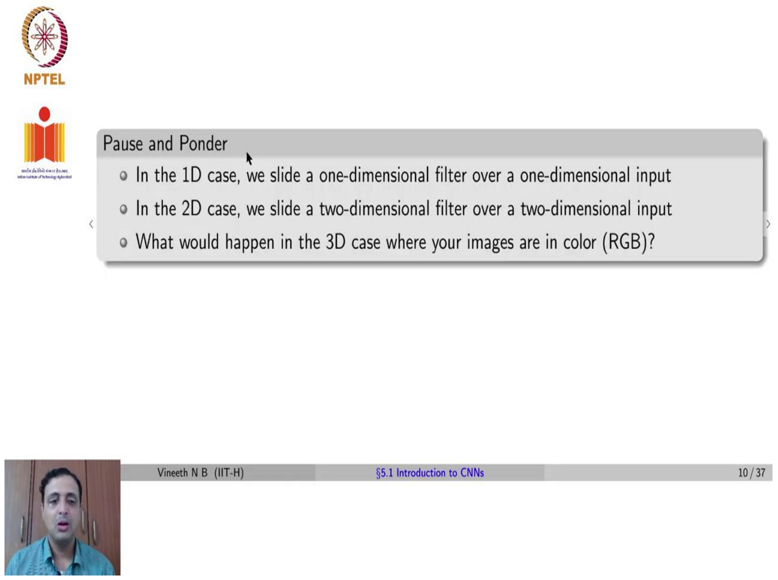
- This allows kernel to be **centered** on pixel of interest

Vineeth N B (IIT-H) §5.1 Introduction to CNNs 9 / 37

More generally, we can also write convolution in the form that you see here. Where we are writing the indices going from the floor of -k1 by 2 to k1 by 2 and -k2 by 2 to k2 by 2. There is a particular reason we are talking about it. Because if we write the equation of convolution this way, that is the only difference, otherwise the rest of it is more or less the same. Obviously, the indices of W change because of how we wrote the summation here, the indices in the summation here.

All that we are saying in this definition is that, we are going to center the filter on the pixel around which we are computing the convolution. We already did that earlier, too. But we are just writing that formally here. So, for different parts in the course, based on what we want to establish, we may write the definition of convolution slightly differently. But you can see that, that the heart of it, it is the same equation, you do have W indices, and X indices defined this way, and you get the output at a particular pixel.

(Refer Slide Time: 11:58)

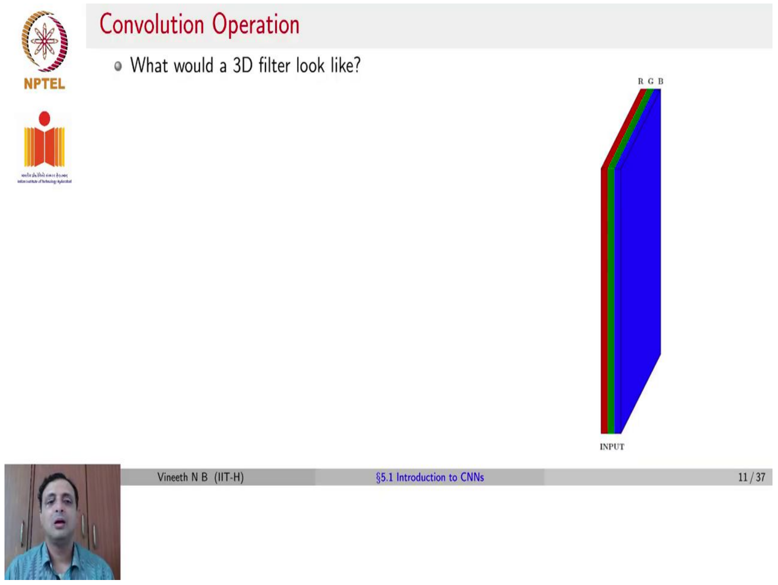

Pause and Ponder

- 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 input
- What would happen in the 3D case where your images are in color (RGB)?

Vineeth N B (IIT-H) §5.1 Introduction to CNNs 10 / 37

We know from what we have seen so far, that for the 1D case, when you do convolution, you slide a 1 dimensional filter over a 1 dimensional input. Similarly, in the 2D case, you slide a 2 dimensional filter, as an images such as a sobel edge filter, and so on over a 2 dimensional input. Let us try to take this further. What would happen in a 3D case, why do you need a 3D case is not an image 2D, an image is 2D. But when you get to color images, you have 3 channels of images which becomes a third dimension, if you want to convolve with them together at one go. So, let us try to understand how this convolution would look in the 3D case.

(Refer Slide Time: 12:51)

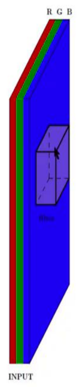

Convolution Operation

- What would a 3D filter look like?

Vineeth N B (IIT-H) §5.1 Introduction to CNNs 11 / 37

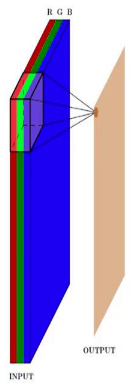
Convolution Operation

- What would a 3D filter look like?
- It will be in 3D too and we will refer to it as a volume



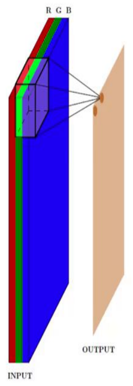
Convolution Operation

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



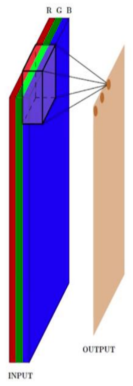
Convolution Operation

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



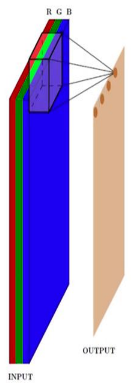
Convolution Operation

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



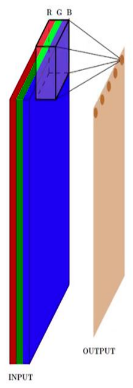
Convolution Operation

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



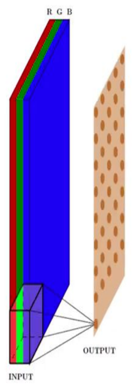
Convolution Operation

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



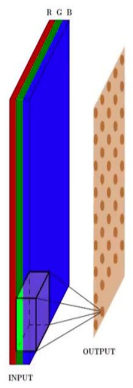
Convolution Operation

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



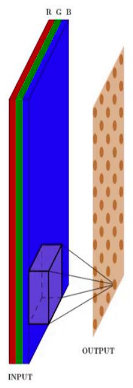
Convolution Operation

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



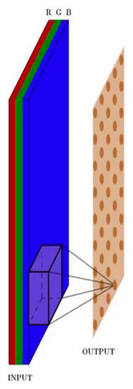
Convolution Operation

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

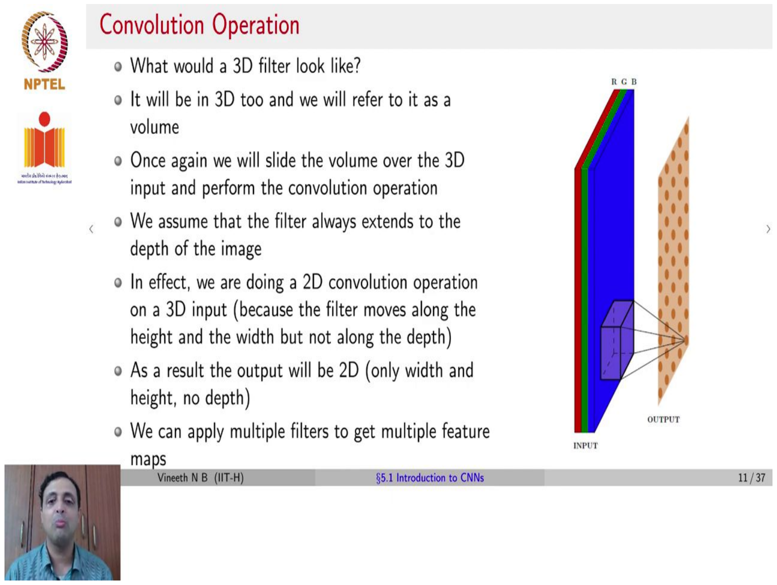


Convolution Operation

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



Vineeth N B (IIT-H) §5.1 Introduction to CNNs 11 / 37



Convolution Operation

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

Vineeth N B (IIT-H) §5.1 Introduction to CNNs 11 / 37

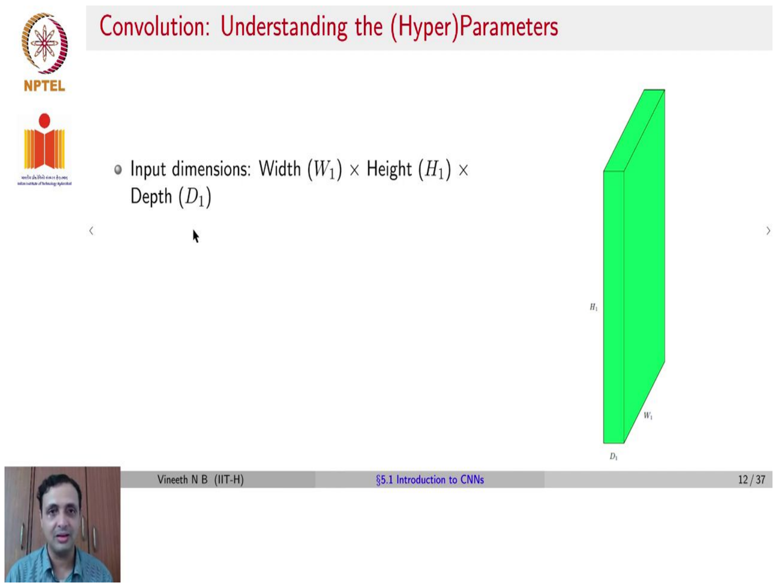
Let us try to understand what a 3D filter would look like. Let us assume now that our input will be a color image, which has 3 channels R, G, and B which makes the input now a volume. And what would a 3D filter look like, it would also be a volume. And we are going to refer to that as a volume going forward. So, when you now slide the volume over your input volume and perform the convolution operation, you will be moving it across the input volume at different locations, and getting 1 scalar output as the output of the convolution in the output image.

See you can see, that as you move the volume on different locations in the input volume, you get the output as pixels in the output image. We also call this output image as a feature map in convolution neural networks. One assumption we make here is that the depth of this filter is the same as the depth of your input volume and this is important. Why is it important, because the depth of the filter is the same as the depth of the input volume.

In effect, we are doing a 2D convolution on a 3D input. Why do we say that, we slide along the height and the width of the input, but we do not slide along the depth, because the depth of the filter is the same as the depth of your input volume. So, your output of this operation will also be a 2D image. When you did 1D convolution with a 1D filter on a 1D signal, you got a 1D output. When you took a 2D filter on a 2D input and image you got a 2D output.

Now we are taking a 3D filter on a 3D input and our output now is not 3D, but 2D. And why is that so, because the depth of the filter is the same as the depth of the input. Then can not you get a volume, can not you get a 3D as output here, you can, even with this operation you can by defining multiple filters. If you define multiple such filter volumes, for each filter volume, you would get 1 such output or 1 such feature map, 1 such feature map. And by using multiple filters, you would get multiple such feature maps, which together form a volume.

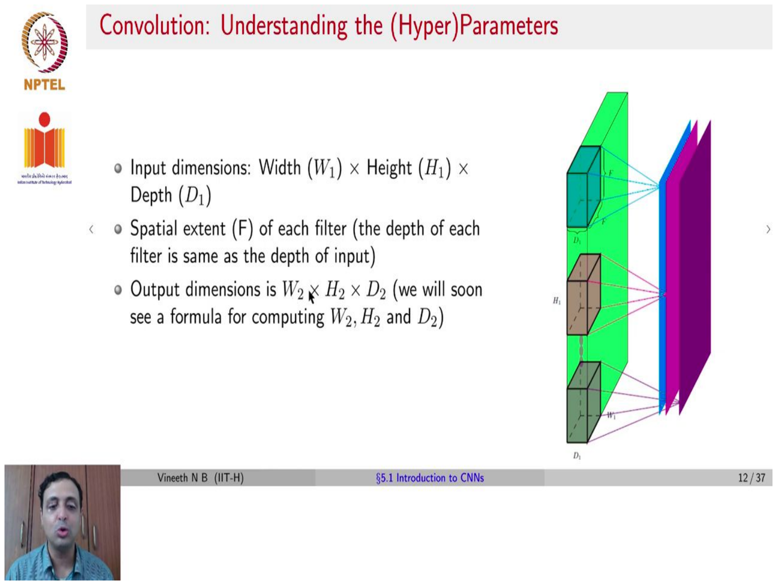
(Refer Slide Time: 15:48)



Convolution: Understanding the (Hyper)Parameters

- Input dimensions: Width (W_1) $\times$ Height (H_1) $\times$ Depth (D_1)

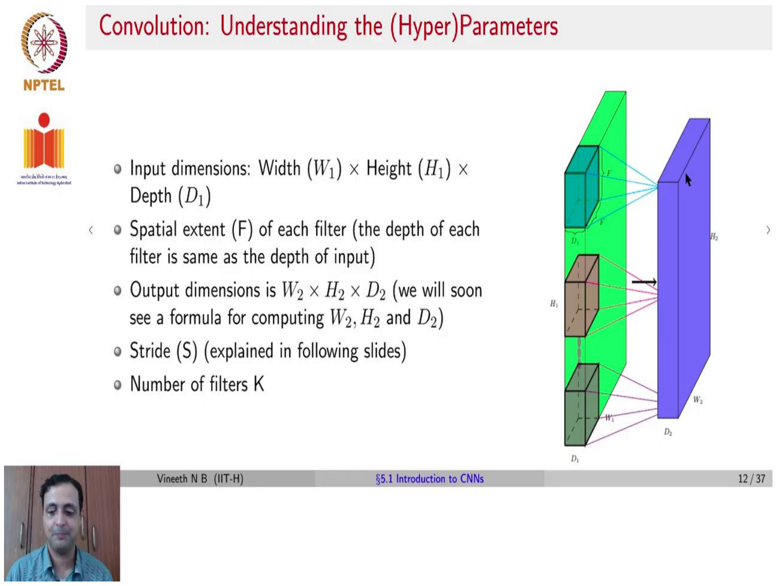
Vineeth N B (IIT-H) §5.1 Introduction to CNNs 12 / 37



Convolution: Understanding the (Hyper)Parameters

- Input dimensions: Width (W_1) $\times$ Height (H_1) $\times$ Depth (D_1)
- Spatial extent (F) of each filter (the depth of each filter is same as the depth of input)
- Output dimensions is $W_2 \times H_2 \times D_2$ (we will soon see a formula for computing W_2, H_2 and D_2)

Vineeth N B (IIT-H) §5.1 Introduction to CNNs 12 / 37



Convolution: Understanding the (Hyper)Parameters

- Input dimensions: Width (W_1) $\times$ Height (H_1) $\times$ Depth (D_1)
- Spatial extent (F) of each filter (the depth of each filter is same as the depth of input)
- Output dimensions is $W_2 \times H_2 \times D_2$ (we will soon see a formula for computing W_2, H_2 and D_2)
- Stride (S) (explained in following slides)
- Number of filters K

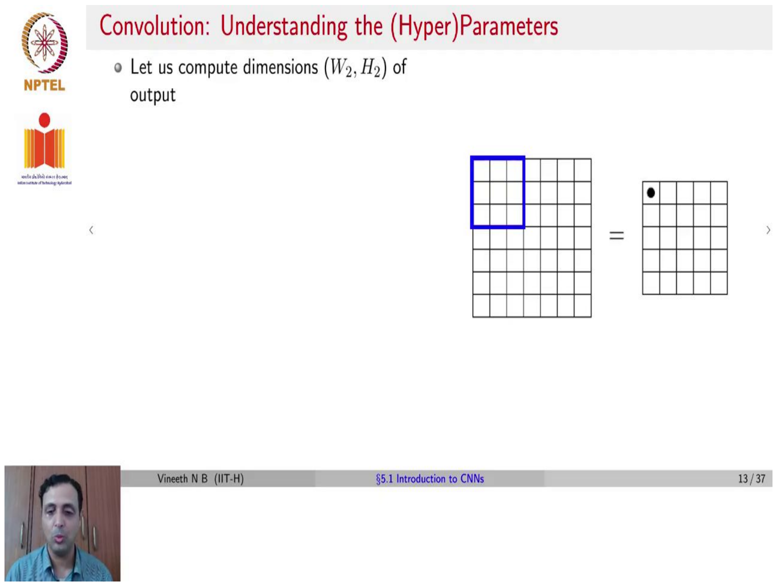
Vineeth N B (IIT-H) §5.1 Introduction to CNNs 12 / 37

Let us now try to understand the dimensions. We know that even when we did convolution, we had some issues around the edges of the image where convolution cannot be performed and we use techniques such as padding, let us revisit some of those in the context of convolution the way we see it for color images.

Let us assume now that our input dimensions are W1 x H1 x D1 with height, and depth. And the spatial extent F of each filter, the depth of each filter is the same as the depth of the input, so you have a say spatial extent to be F x F, and the depth is same as D1, because that is the depth of the input volume. Let us assume now, that the output dimensions are W2 x H2 x D2. So, we are now allowing the output to have some depth D2, if you are used only 1 filter this D2 will be 1, you can increase D2, but increasing the number of filters that you have. Let us try to work out a formula for W2 and H2 in this context.

We will also talk about a quantity called stride, which decides how much you overlap successive convolutions on different parts of the image. And we will also talk about the number of filters in deciding the size of this output volume or output feature maps.

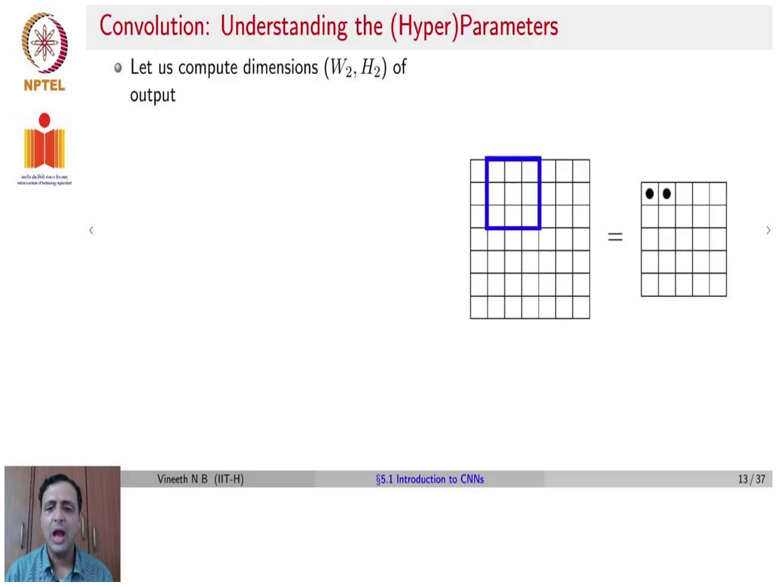
(Refer Slide Time: 17:26)



Convolution: Understanding the (Hyper)Parameters

- Let us compute dimensions (W_2, H_2) of output

Vineeth N B (IIT-H) §5.1 Introduction to CNNs 13 / 37



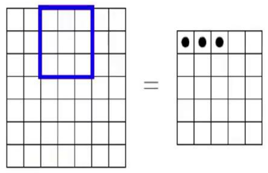
Convolution: Understanding the (Hyper)Parameters

- Let us compute dimensions (W_2, H_2) of output

Vineeth N B (IIT-H) §5.1 Introduction to CNNs 13 / 37

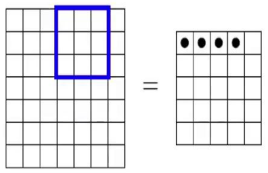
Convolution: Understanding the (Hyper)Parameters

- Let us compute dimensions (W_2, H_2) of output



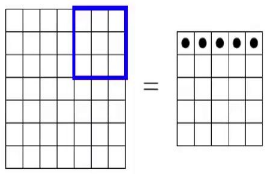
Convolution: Understanding the (Hyper)Parameters

- Let us compute dimensions (W_2, H_2) of output



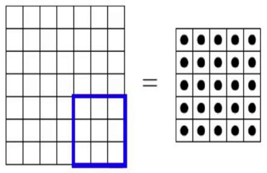
Convolution: Understanding the (Hyper)Parameters

- Let us compute dimensions (W_2, H_2) of output



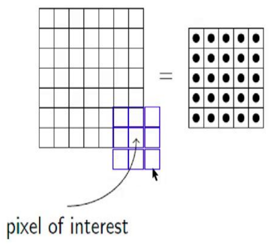
Convolution: Understanding the (Hyper)Parameters

- Let us compute dimensions (W_2, H_2) of output



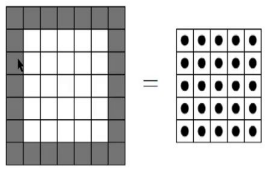
Convolution: Understanding the (Hyper)Parameters

- Let us compute dimensions (W_2, H_2) of output
- Recall that we can't place the kernel at corners as it will cross the input boundary



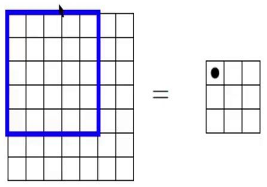
Convolution: Understanding the (Hyper)Parameters

- Let us compute dimensions (W_2, H_2) of output
- Recall that we can't place the kernel at corners as it will cross the input boundary
- This is true for all shaded points (the kernel crosses the input boundary)
- This results in an output which is of smaller dimensions than input
- As size of kernel increases, this becomes true for even more pixels



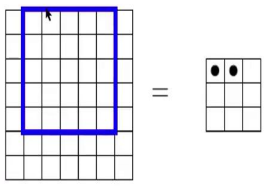
Convolution: Understanding the (Hyper)Parameters

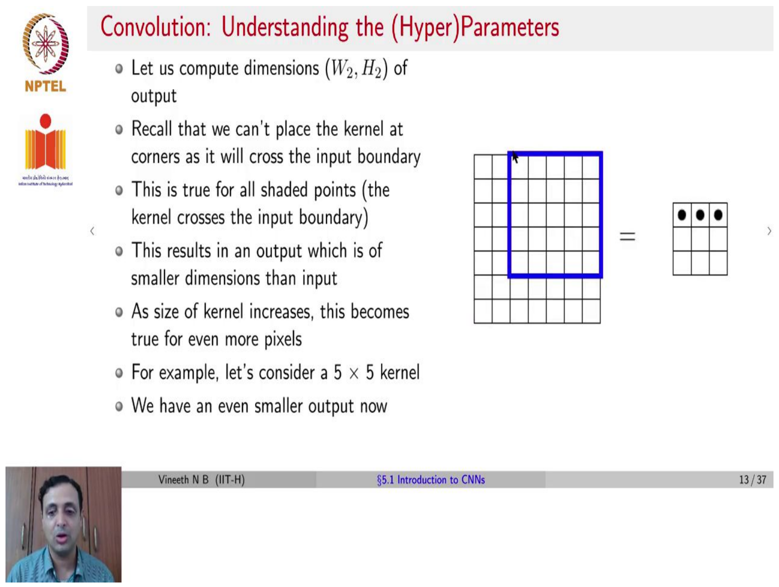
- Let us compute dimensions (W_2, H_2) of output
- Recall that we can't place the kernel at corners as it will cross the input boundary
- This is true for all shaded points (the kernel crosses the input boundary)
- This results in an output which is of smaller dimensions than input
- As size of kernel increases, this becomes true for even more pixels
- For example, let's consider a 5 × 5 kernel



Convolution: Understanding the (Hyper)Parameters

- Let us compute dimensions (W_2, H_2) of output
- Recall that we can't place the kernel at corners as it will cross the input boundary
- This is true for all shaded points (the kernel crosses the input boundary)
- This results in an output which is of smaller dimensions than input
- As size of kernel increases, this becomes true for even more pixels
- For example, let's consider a 5 × 5 kernel
- We have an even smaller output now

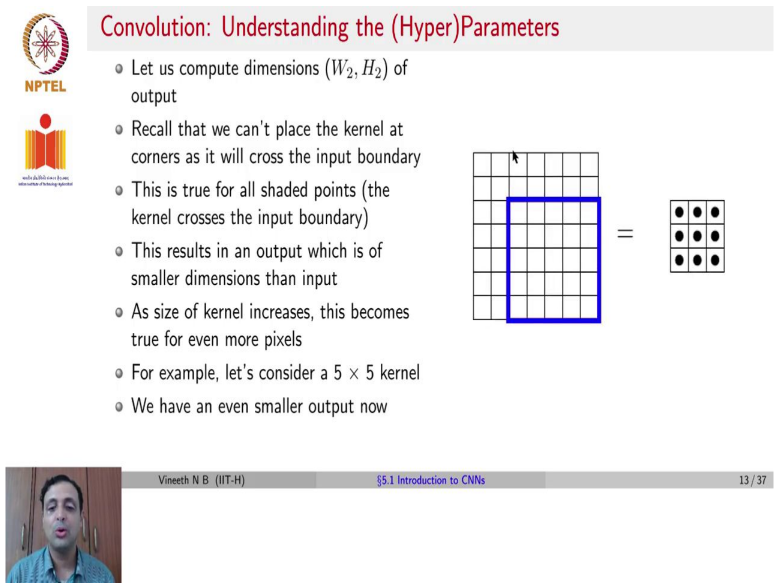




Convolution: Understanding the (Hyper)Parameters

- Let us compute dimensions (W_2, H_2) of output
- Recall that we can't place the kernel at corners as it will cross the input boundary
- This is true for all shaded points (the kernel crosses the input boundary)
- This results in an output which is of smaller dimensions than input
- As size of kernel increases, this becomes true for even more pixels
- For example, let's consider a 5 × 5 kernel
- We have an even smaller output now

Vineeth N B (IIT-H) §5.1 Introduction to CNNs 13 / 37



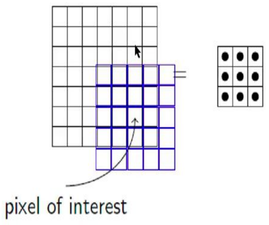
Convolution: Understanding the (Hyper)Parameters

- Let us compute dimensions (W_2, H_2) of output
- Recall that we can't place the kernel at corners as it will cross the input boundary
- This is true for all shaded points (the kernel crosses the input boundary)
- This results in an output which is of smaller dimensions than input
- As size of kernel increases, this becomes true for even more pixels
- For example, let's consider a 5 × 5 kernel
- We have an even smaller output now

Vineeth N B (IIT-H) §5.1 Introduction to CNNs 13 / 37

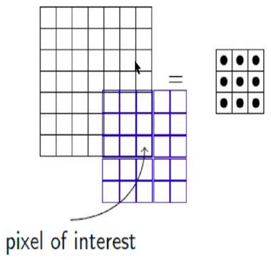
Convolution: Understanding the (Hyper)Parameters

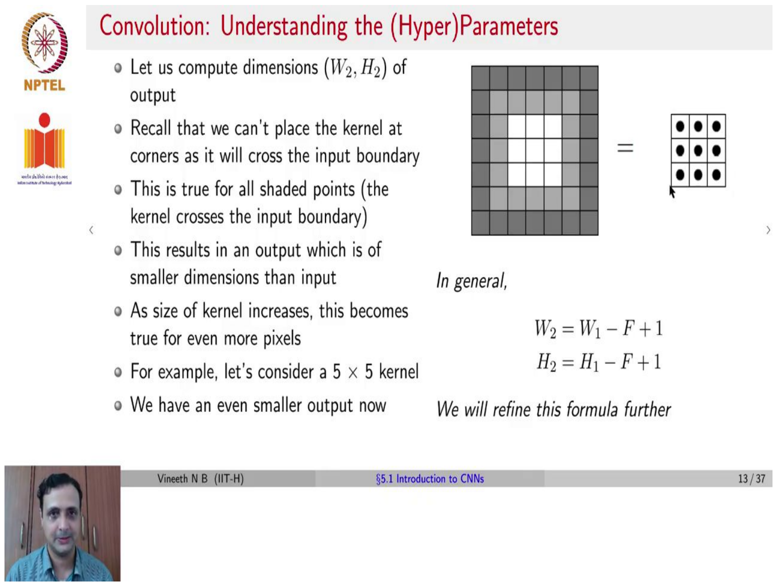
- Let us compute dimensions (W_2, H_2) of output
- Recall that we can't place the kernel at corners as it will cross the input boundary
- This is true for all shaded points (the kernel crosses the input boundary)
- This results in an output which is of smaller dimensions than input
- As size of kernel increases, this becomes true for even more pixels
- For example, let's consider a 5 × 5 kernel
- We have an even smaller output now



Convolution: Understanding the (Hyper)Parameters

- Let us compute dimensions (W_2, H_2) of output
- Recall that we can't place the kernel at corners as it will cross the input boundary
- This is true for all shaded points (the kernel crosses the input boundary)
- This results in an output which is of smaller dimensions than input
- As size of kernel increases, this becomes true for even more pixels
- For example, let's consider a 5 × 5 kernel
- We have an even smaller output now





Convolution: Understanding the (Hyper)Parameters

- Let us compute dimensions (W_2, H_2) of output
- Recall that we can't place the kernel at corners as it will cross the input boundary
- This is true for all shaded points (the kernel crosses the input boundary)
- This results in an output which is of smaller dimensions than input
- As size of kernel increases, this becomes true for even more pixels
- For example, let's consider a 5 × 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

Vineeth N B (IIT-H) §5.1 Introduction to CNNs 13 / 37

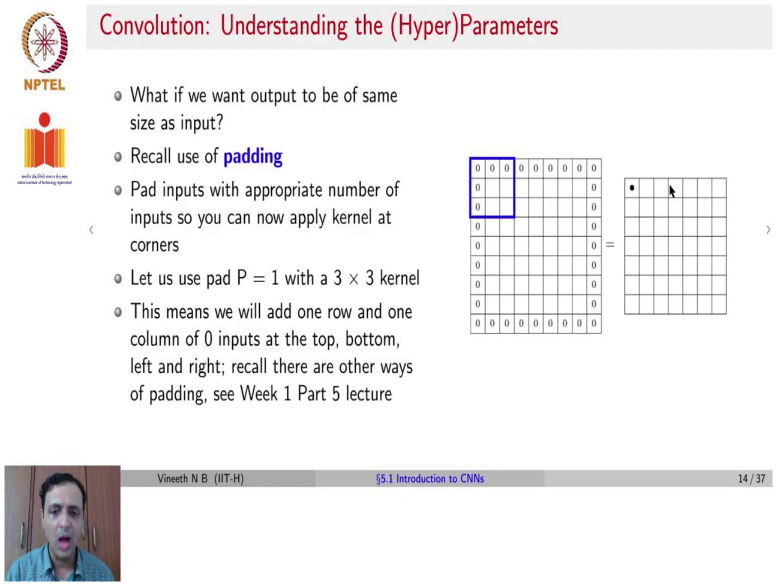
Let us take it one by one. Let us first start with trying to compute the dimensions W2 and H2 of the output. So, you have your standard convolution, where you place your filter on a particular location in your input image, you convolve the filter with the input at that location. And you get one single scalar output at the centered pixel in your output, at the centered pixel location in your output. And you move this all along and keep getting different outputs in the, different pixels as output in the output feature map.

Let us try to understand, what happens if you place the kernel at the corner. So, remember that if you place the kernel at the corners, you do not have inputs to compute the convolution of. So, this is a problem with all these shaded regions here, depending on the size of the filter. So, this results in an output, which is of smaller dimensions than the input. We have seen this earlier with convolution. But we are revisiting this, to find out what would be the size of W2 and H2. As the size of the filter, or the kernel increases, you can make out that even more pixels along the boundary are going to get affected.

And for example, if you take a 5 x 5 kernel now, and as you slide this 5 x 5 kernel across, you are going to get an even smaller output, because more border pixels are going to get affected, because of you will not be able to convolve with those pixels that go outside the boundary. So, you would have an entire region which is outside the boundary for which you will not be able to compute convolution, so your output in this case would be a 3 x 3 matrix.

We have seen this before. In general, we can say with this restricted set up, the W2 will be W1-F + 1, and H2 will be H1- F + 1. So, if your input was W1, in this case, it is 7 x 7, W1 and H1, F was 5 x 5. So, you have 7 - 5 + 1, which is 3, and the same for H2 and that makes your output a 3 x 3 matrix.

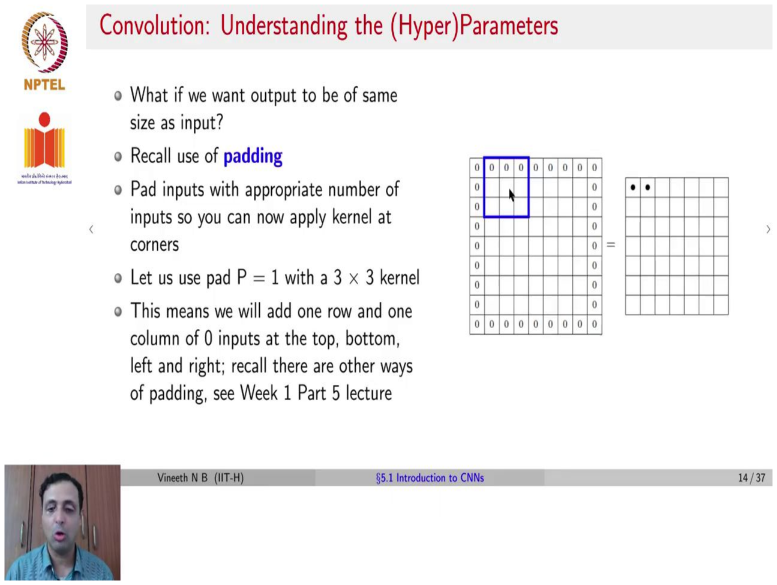
(Refer Slide Time: 20:02)



Convolution: Understanding the (Hyper)Parameters

- What if we want output to be of same size as input?
- Recall use of **padding**
- Pad inputs with appropriate number of inputs so you can now apply kernel at 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; recall there are other ways of padding, see Week 1 Part 5 lecture

Vineeth N B (IIT-H) §5.1 Introduction to CNNs 14 / 37



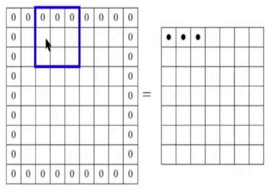
Convolution: Understanding the (Hyper)Parameters

- What if we want output to be of same size as input?
- Recall use of **padding**
- Pad inputs with appropriate number of inputs so you can now apply kernel at 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; recall there are other ways of padding, see Week 1 Part 5 lecture

Vineeth N B (IIT-H) §5.1 Introduction to CNNs 14 / 37

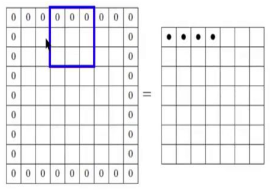
Convolution: Understanding the (Hyper)Parameters

- What if we want output to be of same size as input?
- Recall use of **padding**
- Pad inputs with appropriate number of inputs so you can now apply kernel at 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; recall there are other ways of padding, see Week 1 Part 5 lecture



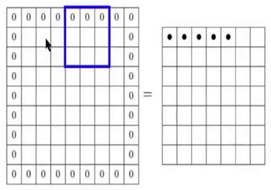
Convolution: Understanding the (Hyper)Parameters

- What if we want output to be of same size as input?
- Recall use of **padding**
- Pad inputs with appropriate number of inputs so you can now apply kernel at 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; recall there are other ways of padding, see Week 1 Part 5 lecture



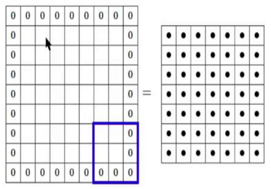
Convolution: Understanding the (Hyper)Parameters

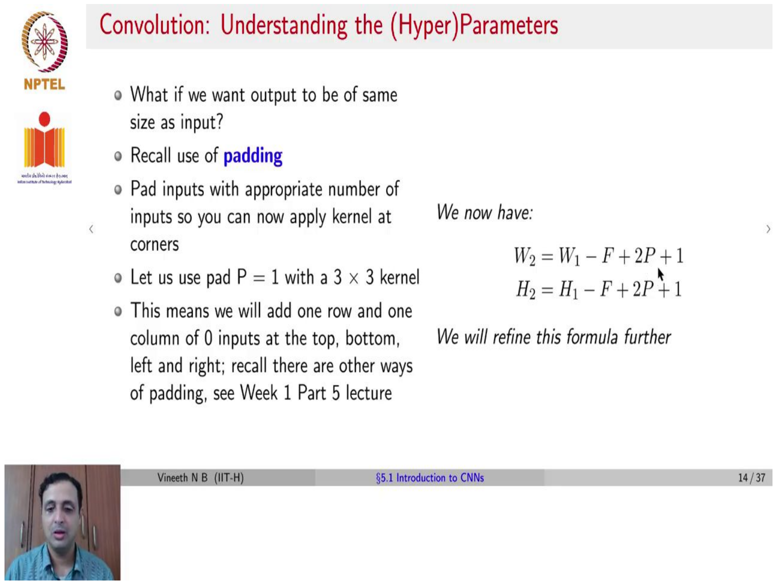
- What if we want output to be of same size as input?
- Recall use of **padding**
- Pad inputs with appropriate number of inputs so you can now apply kernel at corners
- Let us use pad $P = 1$ with a 3×3 kernel
- This means we will add one row and one column of 0 inputs at the top, bottom, left and right; recall there are other ways of padding, see Week 1 Part 5 lecture



Convolution: Understanding the (Hyper)Parameters

- What if we want output to be of same size as input?
- Recall use of **padding**
- Pad inputs with appropriate number of inputs so you can now apply kernel at 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; recall there are other ways of padding, see Week 1 Part 5 lecture





Convolution: Understanding the (Hyper)Parameters

- What if we want output to be of same size as input?
- Recall use of **padding**
- Pad inputs with appropriate number of inputs so you can now apply kernel at 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; recall there are other ways of padding, see Week 1 Part 5 lecture

We now have:

$$W_2 = W_1 - F + 2P + 1$$
$$H_2 = H_1 - F + 2P + 1$$

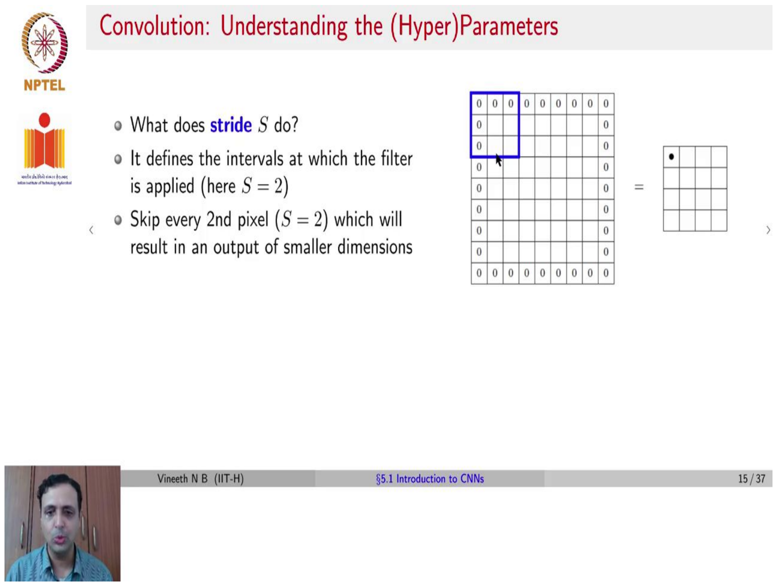
We will refine this formula further

Vineeth N B (IIT-H) §5.1 Introduction to CNNs 14 / 37

We saw earlier, that if we want our output to be the same size as the input, we do what is known as padding. In the first week, we also talked about a few different ways of doing padding. The simplest way to do padding is to add zeros along multiple rows and columns outside the image, so as to extend the original dimensions of the image.

So that way you can convolve, even the boundary pixels. Now, you would have something like this. So, your 7 by 7 becomes a 9 by 9. And now you can place your kernel, even at the corners, as you convolve your 3 x 3 filter on this image, and your output remains 7 x 7 as your input image. We now have, W2 to be W1-F + 2P + 1, where to P is the padding, that is you have done, the padding that you have done. And it is 2P because you have to pass along both ends along the columns, and both ends along the rows too, that is the reason you need 2P. You are going to further refine this formula.

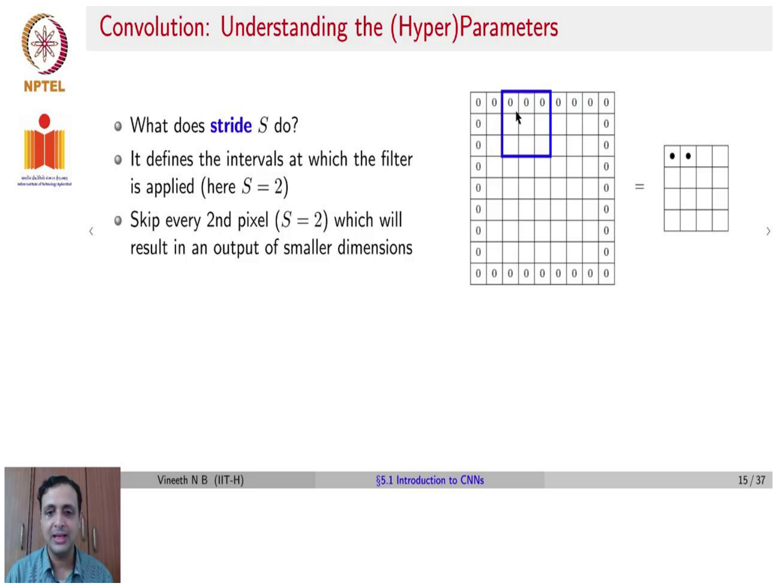
(Refer Slide Time: 21:26)



Convolution: Understanding the (Hyper)Parameters

- What does **stride** S do?
- It defines the intervals at which the filter is applied (here $S = 2$)
- Skip every 2nd pixel ($S = 2$) which will result in an output of smaller dimensions

Vineeth N B (IIT-H) §5.1 Introduction to CNNs 15 / 37



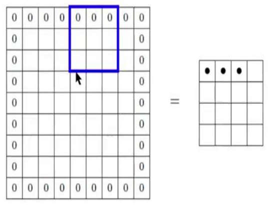
Convolution: Understanding the (Hyper)Parameters

- What does **stride** S do?
- It defines the intervals at which the filter is applied (here $S = 2$)
- Skip every 2nd pixel ($S = 2$) which will result in an output of smaller dimensions

Vineeth N B (IIT-H) §5.1 Introduction to CNNs 15 / 37

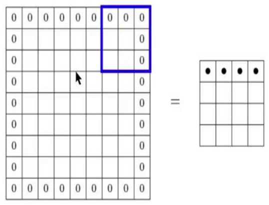
Convolution: Understanding the (Hyper)Parameters

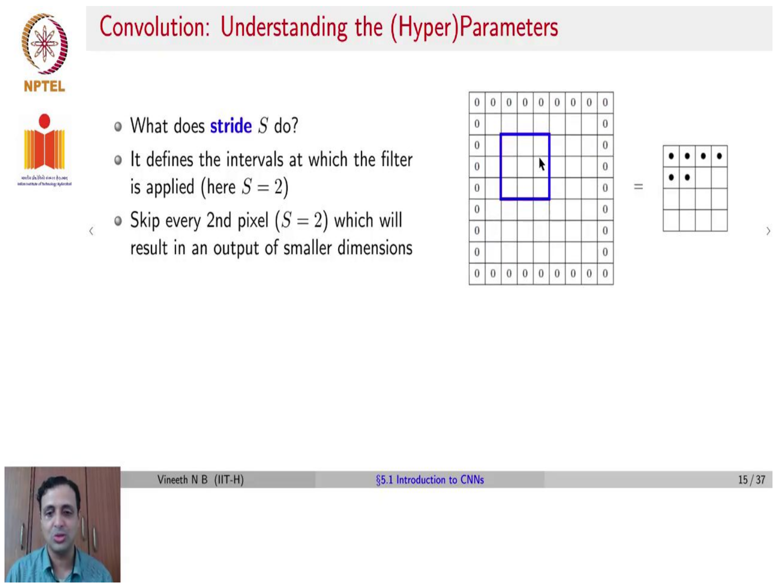
- What does **stride** S do?
- It defines the intervals at which the filter is applied (here $S = 2$)
- Skip every 2nd pixel ($S = 2$) which will result in an output of smaller dimensions



Convolution: Understanding the (Hyper)Parameters

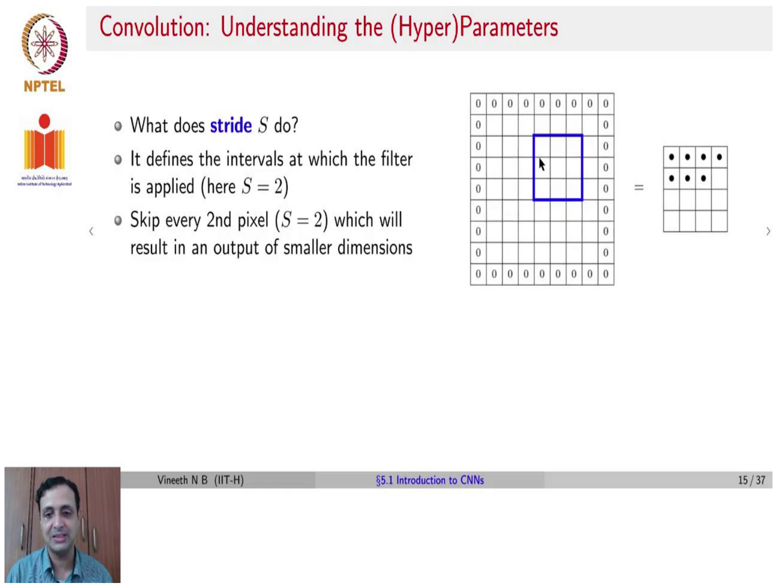
- What does **stride** S do?
- It defines the intervals at which the filter is applied (here $S = 2$)
- Skip every 2nd pixel ($S = 2$) which will result in an output of smaller dimensions





Convolution: Understanding the (Hyper)Parameters

- What does **stride** S do?
- It defines the intervals at which the filter is applied (here $S = 2$)
- Skip every 2nd pixel ($S = 2$) which will result in an output of smaller dimensions

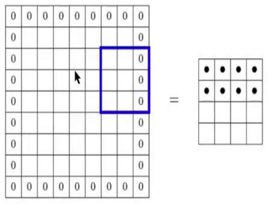


Convolution: Understanding the (Hyper)Parameters

- What does **stride** S do?
- It defines the intervals at which the filter is applied (here $S = 2$)
- Skip every 2nd pixel ($S = 2$) which will result in an output of smaller dimensions

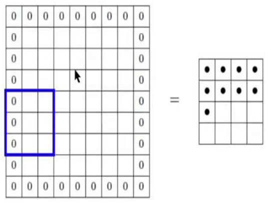
Convolution: Understanding the (Hyper)Parameters

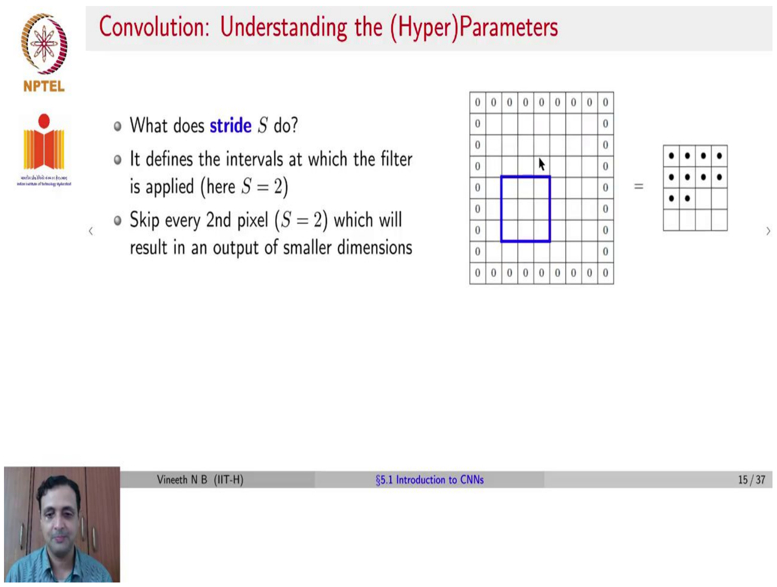
- What does **stride** S do?
- It defines the intervals at which the filter is applied (here $S = 2$)
- Skip every 2nd pixel ($S = 2$) which will result in an output of smaller dimensions



Convolution: Understanding the (Hyper)Parameters

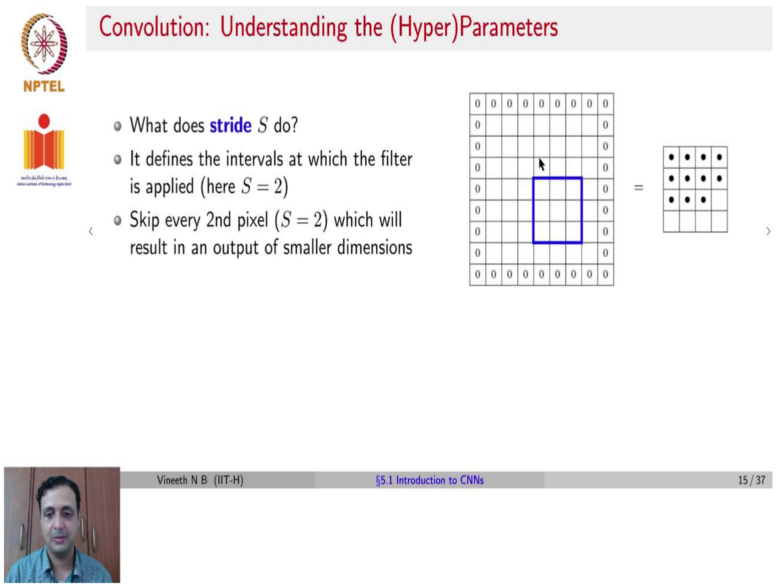
- What does **stride** S do?
- It defines the intervals at which the filter is applied (here $S = 2$)
- Skip every 2nd pixel ($S = 2$) which will result in an output of smaller dimensions





Convolution: Understanding the (Hyper)Parameters

- What does **stride** S do?
- It defines the intervals at which the filter is applied (here $S = 2$)
- Skip every 2nd pixel ($S = 2$) which will result in an output of smaller dimensions

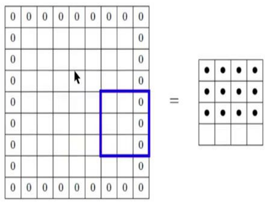


Convolution: Understanding the (Hyper)Parameters

- What does **stride** S do?
- It defines the intervals at which the filter is applied (here $S = 2$)
- Skip every 2nd pixel ($S = 2$) which will result in an output of smaller dimensions

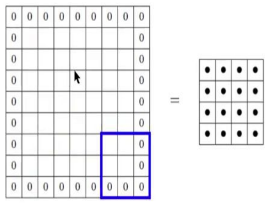
Convolution: Understanding the (Hyper)Parameters

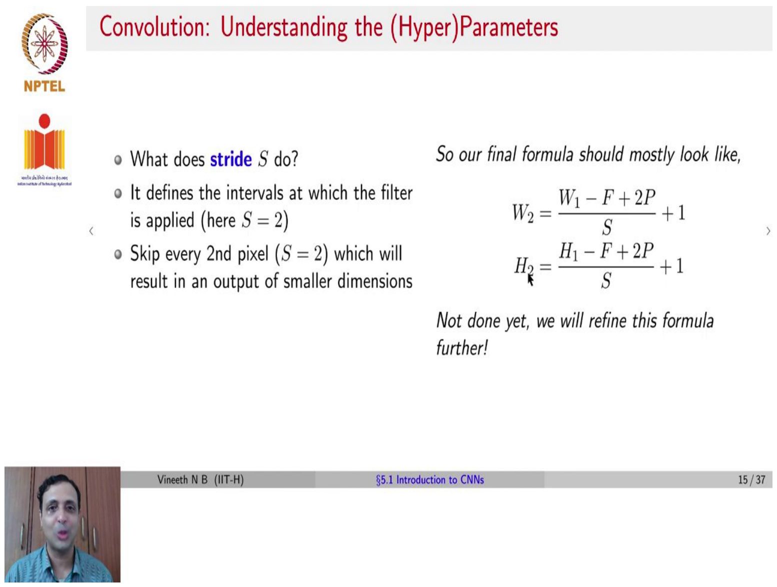
- What does **stride** S do?
- It defines the intervals at which the filter is applied (here $S = 2$)
- Skip every 2nd pixel ($S = 2$) which will result in an output of smaller dimensions



Convolution: Understanding the (Hyper)Parameters

- What does **stride** S do?
- It defines the intervals at which the filter is applied (here $S = 2$)
- Skip every 2nd pixel ($S = 2$) which will result in an output of smaller dimensions





Convolution: Understanding the (Hyper)Parameters

- What does **stride** S do?
- It defines the intervals at which the filter is applied (here $S = 2$)
- Skip every 2nd pixel ($S = 2$) which will result in an output of smaller dimensions

So our final formula should mostly look like,

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

Not done yet, we will refine this formula further!

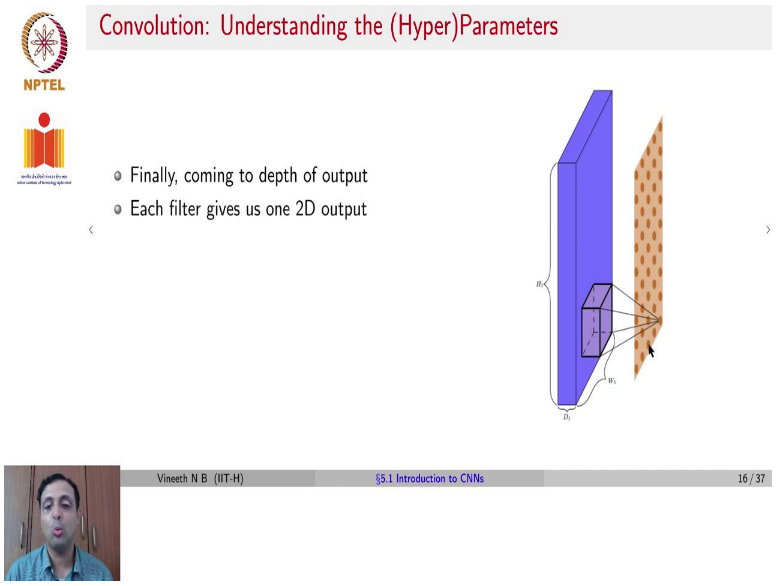
Vineeth N B (IIT-H) §5.1 Introduction to CNNs 15 / 37

We are not done yet. And that is where we bring in the concept of a stride. When we convolve, a filter over an input, we can also define the interval at which the filter is applied. When you move the filter from 1 location on the input to the next location, you can decide whether you would take it to the immediate next pixel, or whether you want to jump 2 steps and go to 2 pixels later and that decides what your stride S.

So, if you went to the immediate pixel, your stride is 1, that is the standard convolution. If you went 2 pixels further, your stride is 2 and that is where stride comes into the picture. So, let us see an example. In this example, S is equal to 2 or stride is equal to 2. So, you can see here that when you took your first 3 x 3 window, and the next 3 x 3 window, you skipped 1 step in between, and then went forward. And let us see these skips a bit more carefully, you can see those skips again. It is not only along the columns, it is also along the rows, you also skip pixels along the rows, the stride happens in both directions.

And this is another way in which the size of your image gets output image gets reduced. And your new formula now with the stride included becomes W2 is equal to W1-F + 2P which is what we had earlier, which is now divided by S the stride. Since the stride along the columns and the rows are the same, you use the same S. Technically speaking, the strides could be different between the rows and the columns. But in practice, it is generally retained as the same value. Are we done with finding the values of W and H2, not yet done, there is one more detail left.

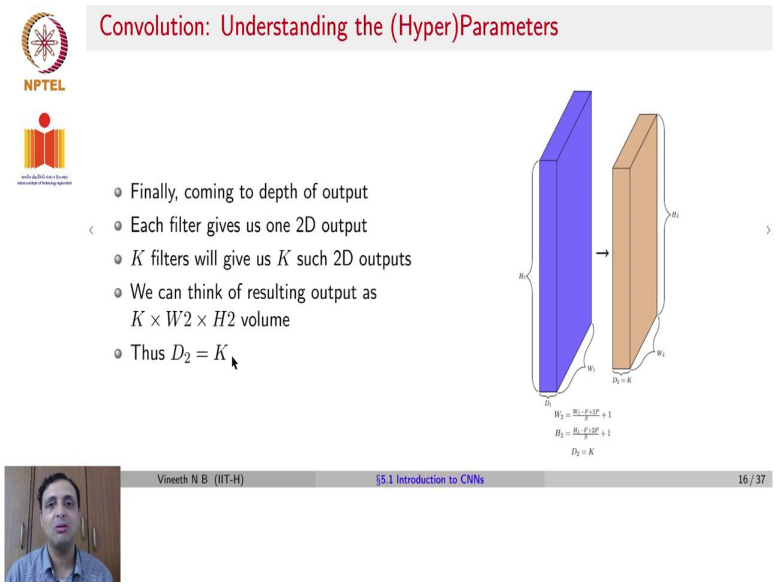
(Refer Slide Time: 23:29)



Convolution: Understanding the (Hyper)Parameters

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

Vineeth N B (IIT-H) §5.1 Introduction to CNNs 16 / 37



Convolution: Understanding the (Hyper)Parameters

- Finally, coming to depth of output
- Each filter gives us one 2D output
- K filters will give us K such 2D outputs
- We can think of resulting output as $K \times W2 \times H2$ volume
- Thus $D_2 = K$

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

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

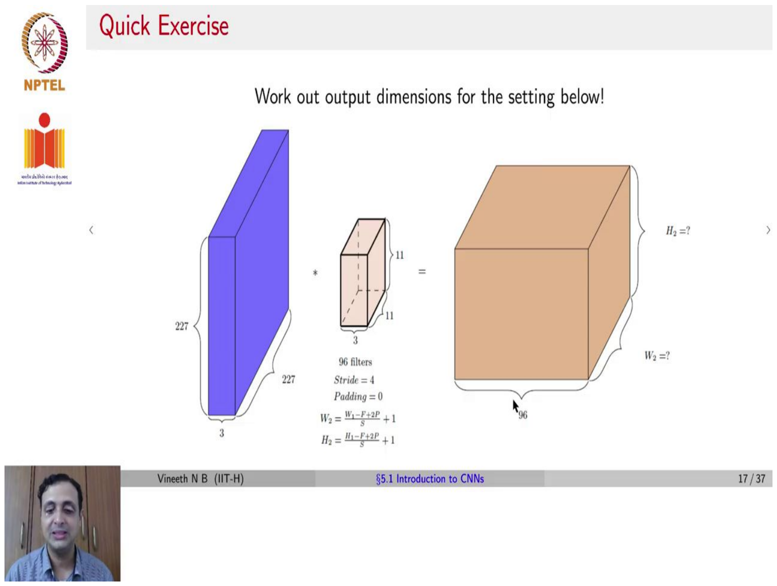
$D_2 = K$

Vineeth N B (IIT-H) §5.1 Introduction to CNNs 16 / 37

And that detail is the depth of the output. As we already mentioned, the depth of the output comes from using multiple feature maps or multiple filters. Each filter gives us one 2D output and K such filters will give us K such 2D outputs, as you can see now in the output region. We are saying now that the depth D2 is K, that is the number of filters you used. For each filter, you would completely convolve your filter volume with the input volume get 1 2D image.

You take another filter, you could imagine taking a sobel filter along the vertical sobel filter and a horizontal sobel detect those would be 2 different filters. So, you run both of them, and you will get 2 images as output. You just place them one after the other to form a volume. So, your D2 on the depth of the output would be your number of filters, or K.

(Refer Slide Time: 24:33)



Quick Exercise

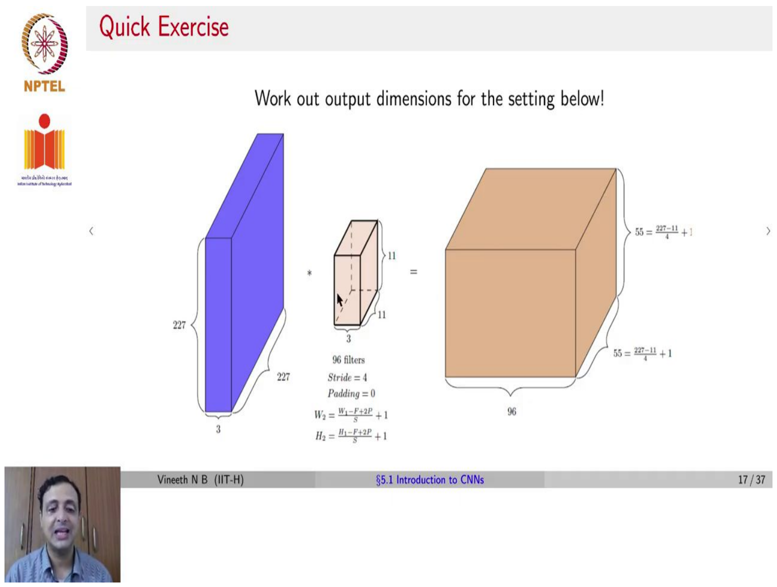
Work out output dimensions for the setting below!

227
227
3
*
11
11
3
96 filters
Stride = 4
Padding = 0
$W_2 = \frac{W_1 - F + 2P}{S} + 1$
$H_2 = \frac{H_1 - F + 2P}{S} + 1$
=
$H_2 = ?$
$W_2 = ?$
96
Vineeth N B (IIT-H)
§5.1 Introduction to CNNs
17 / 37

Let us try to work out an exercise so you are clear about this. Let us consider this input to be 227 x 227 x 3. Let us assume that your filter size is 11 x 11 x 3. Let us assume that we are going to use 96 such filters. These numbers have a purpose which will probably be clear to you a few, a couple of lectures down but take these numbers for now as they are, you have 96 such filters. Let us assume a stride of 4 and a padding of 0, no padding.

We already know that the formula W2=((W1-F + 2P)/S)+1. And similarly, for H2, can we now find out what D2 W2 and H2 are. Let us start with the simplest one. What is D2, D2 as we said, is the number of filters, which means D2 is 96.

(Refer Slide Time: 25:40)



Quick Exercise

Work out output dimensions for the setting below!

227
227
3
*
11
11
3
96 filters
Stride = 4
Padding = 0
$W_2 = \frac{W_1 - F + 2P}{S} + 1$
$H_2 = \frac{H_1 - F + 2P}{S} + 1$
=
$55 = \frac{227-11}{4} + 1$
$55 = \frac{227-11}{4} + 1$
96

Vineeth N B (IIT-H) §5.1 Introduction to CNNs 17 / 37

Let us then go to H2, H2 was given by H2 = ((227-11)/4)+1 = 55. And we get a similar value for W2. That is how you can get the output, the dimensions of the feature map. One important thing to keep in mind here is when you embed these kinds of operations in a neural network. Unlike a feed forward neural network, where you would specify how many neurons you want in a first hidden layer, here, the size of that next layer is automatically decided by the size of the filter and these choices that you make.

In a feed forward neural network, the number of weights gets decided by the number of hidden layers and the neurons in each hidden layer. And here, it is the reverse. You decide the number of weights and the output map gets decide. We will explain this in more detail as we go forward now.