

Fully-Connected Layers to Convolutions

Tirtharaj Dash

Dept. of CS & IS and APPCAIR
BITS Pilani, Goa Campus

September 25, 2021

Summary of the previous lecture:

- Motif detection in images and text
- Template matching
- Introducing convolution operation
- The idea of translation invariance, locality

Today, we will look closely at the internal mathematical details of the above. These are:

- Adapting fully-connected layers to convolutions
- Convolutions for images

From Fully-Connected layers to Convolutions I

MLPs to CNNs:

- We know that MLPs are able to handle data that do not really have any dependency among features.
 - However, we might anticipate that the patterns we seek could involve interactions among the features, but we do not assume any structure *a priori* concerning how the features interact.
- MLPs are the right architectures if we do not have any knowledge about how features interact.
 - A major limitation: For high-dimensional data the model could be very complex with a huge set of parameters.

From Fully-Connected layers to Convolutions II

- Example: Cat vs Dog classification
 - Let say we have collected an annotated dataset of 1 megapixel images of cats and dogs.
 - That means each input to an MLP network (just with 1 hidden layer) has 1 million dimensions (flattened in a row-major fashion).
 - If we restrict our hidden layer size to 1000 units (although 1000 units may not be able to find meaningful patterns when dealing with 10^6 inputs), the number of parameters in our first layer is: $10^6 \times 10^3 = 10^9$ (+1000 if we consider bias).
 - Learning this network is not feasible, if we do not have a lot of compute, optimisation skills and patience!

From Fully-Connected layers to Convolutions III

- Well, now you may come and object to the fact that we may not need 1 megapixel images, we could resize them to smaller sizes.
- You may change the structure of MLP to deal directly with two-dimensional grid structure (without flattening the image).
- Yes, these tricks may work: you can reduce the model size; make clever modelling; but, still your bottleneck would be the hidden dimension.
- That is: a 1000-dimensional hidden representation may not be meaningful at all to represent an image, which already lost the pixel-level spatial relationship due to flattening.

From Fully-Connected layers to Convolutions IV

- An image exhibits rich relational structure due to the spatial relationship among the raw pixels.
- Humans (and now-a-days intelligent machines) are able to exploit this rich spatial structure quite well.
- Convolutional neural networks (CNNs) are one creative way of implementing such artificially intelligent machines.

From Fully-Connected layers to Convolutions V

- In what follows next, we will be discussing how CNNs are able to exploit some of the common properties in an image, that MLPs cannot handle.

Invariance and Locality I

- Let us consider a task where our interest is in detecting an object in an image. For example, detecting 'Waldo' in 'Where's Waldo?' game:



(source: textbook)

Invariance and Locality II

- The goal is to construct Waldo detector that sweeps the whole image one patch after another and score each patch with some value between 0 and 1 indicating likelihood that the patch contains Waldo.
- CNNs systematize this idea of *spatial invariance*, exploiting it to learn useful representations with fewer parameters.
- Example of spatial invariance: The image is shifted.



(source: <https://stats.stackexchange.com/a/208949/127347>)

Invariance and Locality III

- Our discussions so far involved the following:
 - *Translation invariance*: Our network, in its earliest layers, should respond similarly to the same patch, regardless of where it appears in the image.
 - *Locality*: The earliest layers of the network should focus on local regions, without regard for the contents of the image in distant regions.
 - Eventually, these local representations can be aggregated to make predictions at the whole image level.

Next we look at the mathematics of these principles →

Constraining the MLP I

- Consider an MLP with:
 - Two-dimensional input image \mathbf{X}
 - Two-dimensional hidden representations \mathbf{H} . That is, the hidden representations also possesses spatial structure.
 - Both \mathbf{X} and \mathbf{H} have same shape.
 - $\mathbf{X}_{i,j}$ and $\mathbf{H}_{i,j}$ pixel at location (i,j) in the input image and hidden representation.

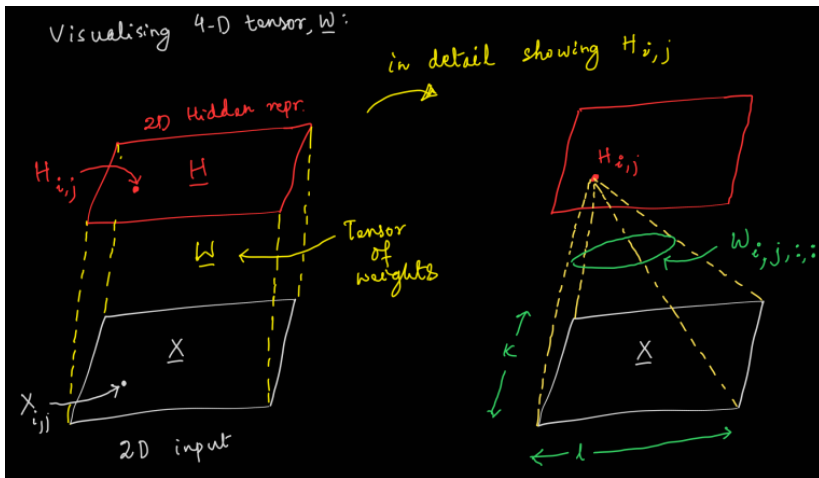
Constraining the MLP II

- Weights:
 - Recall that in an MLP we used to represent the set of parameters in layer ℓ with a weight matrix ($\in \mathbb{R}^{m^{(\ell)} \times m^{(\ell-1)}}$) where $m^{(\ell)}$ denote the size of the hidden layer ℓ . That is, input is in $\mathbb{R}^{m^{(\ell-1)}}$ and the output is in $\mathbb{R}^{m^{(\ell)}}$
 - Now, we have both the inputs and outputs both are two dimensional. So the weights will be represented by 4-dimensional tensor, W .

Note: Complicated to visualise. To get some intuition, just think that each cell in the hidden representation is a consequence of matrix multiplication of the input matrix with a weight matrix. So we need *row* \times *col* number of such weight matrices to fill the whole output representation, where *row* and *col* represent the shape of the hidden representation.

Constraining the MLP III

- Visualising W :



Constraining the MLP IV

- Fully-connected layer is then:

$$\begin{aligned}\mathbf{H}_{i,j} &= \mathbf{U}_{i,j} + \sum_k \sum_l \mathbf{W}_{i,j,k,l} \mathbf{X}_{k,l} \\ &= \mathbf{U}_{i,j} + \sum_a \sum_b \mathbf{V}_{i,j,a,b} \mathbf{X}_{i+a,j+b}\end{aligned}$$

where \mathbf{U} is the bias matrix. And,

- There is one-to-one correspondence between the elements in tensor \mathbf{W} and \mathbf{V} (just a matter of notational convenience resulting from the next point)
- Re-indexing the sub-scripts (k, l) to $k = i + a$ and $l = j + b$. That is

$$\mathbf{V}_{i,j,a,b} = \mathbf{W}_{i,j,i+a,j+b}$$

Constraining the MLP V

- The indices a and b are the indexing variables to hover around the image.
- The summary is that: The hidden representation at location (i, j) , $\mathbf{H}_{i,j}$ is computed by summing over pixels in \mathbf{X} centered around (i, j) and weighted by $V_{i,j,a,b}$.

Constraining the MLP VI

Translation Invariance:

- A shift in the input \mathbf{X} should simply lead to a shift in the hidden representation \mathbf{H} .
- This is only possible if both \mathbf{U} and \mathbf{V} do not depend on (i, j) . That is:

$$V_{i,j,a,b} = \mathbf{V}_{a,b}$$

and \mathbf{U} is a constant, say u .

- That is: instead of using different matrices for the calculation of $\mathbf{H}_{i,j}$, use a single template \mathbf{V} to scan over the whole image.
 - This is a consequence of translational invariance.
 - This is called *parameter sharing*: effectively reduces the number of parameters in CNNs, as compared to MLPs.

- So the definition of \mathbf{H} then simplifies to:

$$\mathbf{H}_{i,j} = u + \sum_a \sum_b \mathbf{V}_{a,b} \mathbf{X}_{i+a,j+b}$$

This is a *convolution* operation.

- We are effectively weighing pixels at $(i + a, j + b)$ in the vicinity of location (i, j) with coefficients $\mathbf{V}_{a,b}$ to obtain the value $\mathbf{H}_{i,j}$.

Constraining the MLP VIII

Locality:

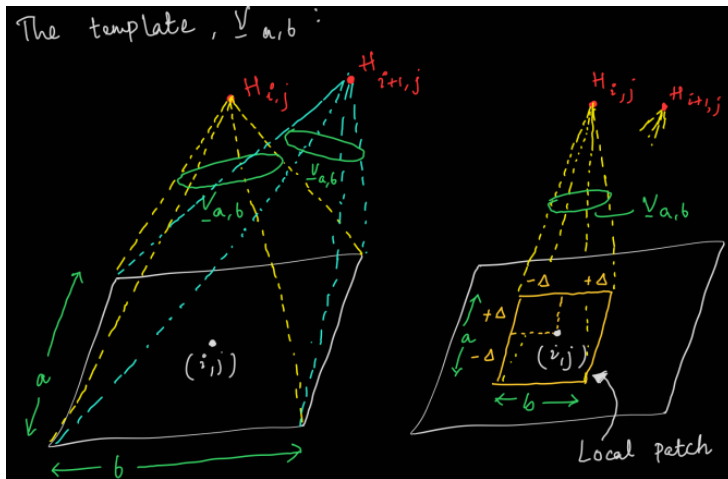
- Locality means that only a small neighborhood of pixels will be used to compute the corresponding hidden representations.
- That is: We should not have to look very far away from location (i, j) in order to obtain relevant information $\mathbf{H}_{i,j}$.
- This means that outside some range $|a| > \Delta$ or $|b| > \Delta$, we should set $\mathbf{V}_{a,b} = 0$
- Then, $\mathbf{H}_{i,j}$ becomes:

$$\mathbf{H}_{i,j} = u + \sum_{a=-\Delta}^{\Delta} \sum_{b=-\Delta}^{\Delta} \mathbf{V}_{a,b} \mathbf{x}_{i+a,j+b}$$

The above expression represents a *convolutional layer*.

Constraining the MLP IX

- A visualisation:



Constraining the MLP \mathbf{X}

- CNNs are a special family of neural models that consist of convolutional layers.
 - In deep learning research, we call \mathbf{V} as a convolution *kernel*, or *filter*, or *layer weights*. These are learnable parameters.
 - When the local region is small (that is, we impose an inductive bias), the number of parameters in CNNs is far lesser than that of MLPs (typically, a billion in MLPs vs few hundreds in CNNs), without altering the dimension of \mathbf{X} or \mathbf{H} .
 - If our inductive bias do not agree with reality, then the model might fail to perform.

- We now discuss why we call the linear algebra in the previous slides as *convolution*.
- In mathematics (and in signal processing), the convolution between two functions (or two signals), say $f, g : \mathbb{R}^d \rightarrow \mathbb{R}$ is defined as:

$$(f * g)(\mathbf{x}) = \int f(\mathbf{z})g(\mathbf{x} - \mathbf{z})d\mathbf{z}$$

- That is, we measure the overlap between f and g when one function is “flipped” and shifted by \mathbf{x} .

Convolution II

- When we are dealing with discrete objects, the integral turns into a sum:

$$(f * g)(\mathbf{x}) = \sum_a f(a)g(i - a)$$

- For two-dimensional objects, we have two indices;

$$(f * g)(\mathbf{x}) = \sum_a \sum_b f(a, b)g(i - a, j - b)$$

Note: The above expression looks similar to:

$$\mathbf{H}_{i,j} = u + \sum_{a=-\Delta}^{\Delta} \sum_{b=-\Delta}^{\Delta} \mathbf{v}_{a,b} \mathbf{x}_{i+a,j+b}$$

There is a difference though. $(i + a, j + b)$ is used instead of $i - a, j - b$. This more properly describes cross-correlation.

- Convolutions in CNNs is nothing but *cross-correlation*.
 - Convolution means sliding a flipped kernel across an image. (We don't do this.)
 - Cross-correlation means sliding a kernel (filter) across an image. (We do this.)

Convolution IV

- For our Waldo detector, the hidden representation should represent the following: Whenever the likelihood that a patch in the input image contains the object, it should light up appropriately.



(source: textbook)

Multiple Channels I

- So far we have considered that our input will be a 2D grid (that is a matrix). But, in reality, an image contains multiple color channels: Red-Green-Blue (RGB) or Cyan-Magenta-Yellow-Black (CMYK).
- In this case, our input is not a matrix, but a vector of matrices, called a tensor, denoted as X .
- Each pixel is then $X_{i,j,k}$, where k represents the channel index.
- Similarly the kernel will then also be $V_{a,b,c}$.

- This results in a computation of a *feature map* as:

$$H_{i,j} = \sum_{a=-\Delta}^{\Delta} \sum_{b=-\Delta}^{\Delta} \sum_c V_{a,b,c} X_{i+a,j+b,c}$$

Multiple Channels III

- We can use multiple V s, each representing a different template to detect some relevant information in an image.
- The whole tensor V is called a *kernel* or *filter* or *layer weights*. Or, we can say our kernel is a tensor.
- Let say we have d such kernels, each with c channels.

Multiple Channels IV

- Then the hidden representations will then comprise of number of 2D grids stacked on top of each other, called *feature maps*.
- Each feature map provides a spatialized set of learned features to the subsequent layer.
- The hidden representation computation, in general form, becomes:

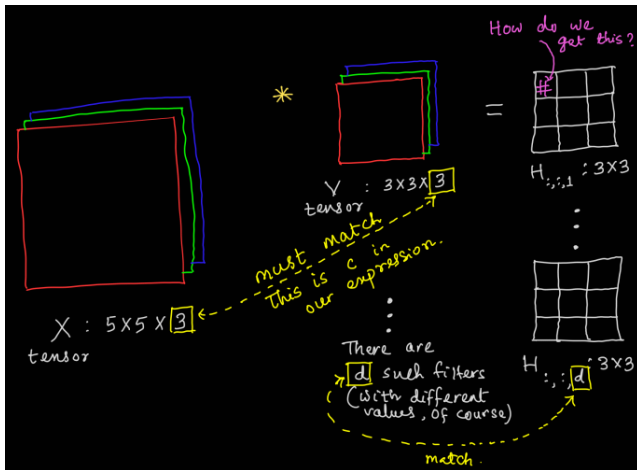
$$H_{i,j,d} = \sum_{a=-\Delta}^{\Delta} \sum_{b=-\Delta}^{\Delta} \sum_c V_{a,b,c,d} X_{i+a,j+b,c}$$

where d indexes the output channels in the hidden representations H . That is: d feature maps.

- Notice that: In the above expression, we have d kernels, each is a tensor with c channels (equal to the number of channels in the input X).

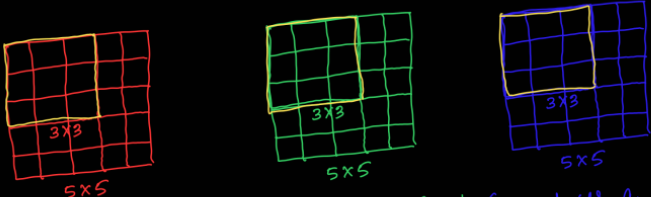
Multiple Channels V

- This is what we expressed in the previous slide:



Multiple Channels VI

How do we get '#' :



(products a single number)

$$v_{11}x_{11} + v_{12}x_{12} + v_{13}x_{13} + \dots + v_{33}x_{33} = \boxed{}$$

(products a single number)

$$v_{11}x_{11} + v_{12}x_{12} + v_{13}x_{13} + \dots + v_{33}x_{33} = \boxed{}$$

(products a single number)

$$v_{11}x_{11} + v_{12}x_{12} + v_{13}x_{13} + \dots + v_{33}x_{33} = \boxed{}$$

= $\boxed{} + \boxed{} + \boxed{}$

Note: We have considered a stride of 1 and no padding for demonstration purpose. Therefore the resulting dimension is 3×3 for each feature map.

CNNs and Invariances: I

Remember that CNNs can handle translation invariance only. That is:

- CNN cannot automatically handle rotation-invariance:



(source: <https://stats.stackexchange.com/a/208949/127347>)

CNNs and Invariances: II

- CNN cannot automatically handle size-invariance:



(source: <https://stats.stackexchange.com/a/208949/127347>)

CNNs and Invariances: III

- CNN cannot automatically handle illumination-invariance:



(source: <https://stats.stackexchange.com/a/208949/127347>)