4. Convolutional neural networks

4.1. Elements of the convolutional neural network

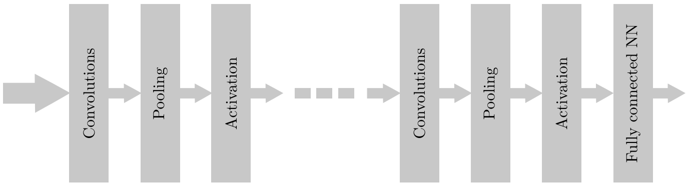
Manel Martínez-Ramón
Meenu Ajith
Aswathy Rajendra Kurup

Introduction

- The basic NN in Module 1 extracts features of a pattern through a nonlinear representation in a higher dimensional space.
- Next layers represent features with a higher level of abstraction.
- In the last layer, the features produce representations of the input pattern that can be linearly classified.
- The core of the NN is the affine transformation $\mathbf{W}^\top \mathbf{x} + \mathbf{b}$ plus a nonlinear activation.
- This is a global transformation that does not have local properties.
- Local feature representation capabilities are needed in, for example, images.

Overall structure

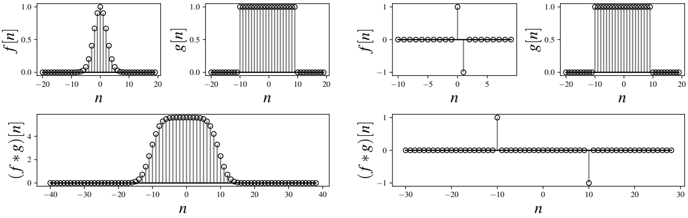
- Several convolutional blocks, including convolutions, pooling and activation layers.
- A fully connected block, consisting of a standard MLP.



- The output can contain a softmax (multiclass classification) or linear activation.

- In one dimension, the discrete convolution is defined as

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m] \tag{1}$$



- At point n, the convolution is the sum of products of one signal times the other one shifted n positions and reversed.

Convolutions in 2 dimensions

- The extension to 2 dimensions is immediate.

$$(\mathbf{I} * \mathbf{W})[m,n] = \sum_{p=0}^{M_W-1} \sum_{q=0}^{N_W-1} \mathbf{W}[p,q]\mathbf{I}[m+p,n+q] \qquad (2)$$

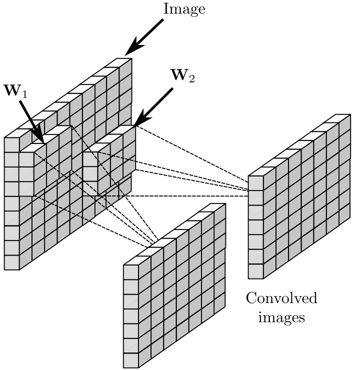


Image convolved with kernels $\mathbf{W}_1$ and $\mathbf{W}_2$.

- Image $\mathbf{I}$ of dimension $M_I \times N_I$.
- *Convolution kernel* $\mathbf{W}$: array of dimensions $M_W \times N_W$.
- Dimensions of the resulting array: $M_I - M_W + 1 \times N_I - N_W + 1$.

What is a convolutional layer for?

- Problem: detect an object or shape in an image regardless of its position.
- Its exact rotation, and scale will also be issues, but let's not mind this for now.
- The position of the object must not be relevant, so each extracted feature must receive information from all pixels.

Example of a convolution

Example of the convolution of an image with a convolution kernel designed to enhance the edges of the image.

- The image has been convolved with the kernel

$$\mathbf{W} = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} \tag{3}$$

Padding

- The convolution crops the image. The convolution dimensions are

$$M_I - M_W + 1 \times N_I - N_W + 1$$

- The pixels at the edge of the image are seen only when the convolution kernel touches the edge of the image.
- If we add p rows/columns of zeros, the output dimensions are

$$(M_I - M_W + p + 1) \times (N_I - N_W + p + 1) \tag{4}$$

- If $p/2$ rows are added to each side of the input image so the number of rows of the convolved image does not change:

$$M_I + p - M_W + 1 = M_I$$

$$\frac{p}{2} = \frac{M_W - 1}{2}$$

hence M_W must be odd.

Padding

$$\overset{\mathbf{I}}{\begin{array}{|c|c|c|c|c|c|c|}\hline 0&0&0&0&0&0&0\\ \hline 0&1&2&1&3&1&0\\ \hline 0&2&1&2&1&3&0\\ \hline 0&1&2&1&3&1&0\\ \hline 0&2&1&2&1&3&0\\ \hline 0&0&0&0&0&0&0\\ \hline \end{array}} * \overset{\mathbf{W}}{\begin{array}{|c|c|}\hline 0&1\\ \hline 1&0\\ \hline \end{array}} = \overset{\mathbf{W} * \mathbf{I}}{\begin{array}{|c|c|c|c|c|c|}\hline 0&1&2&1&3&1\\ \hline 1&4&2&5&2&3\\ \hline 2&2&4&2&6&1\\ \hline 1&4&2&5&2&3\\ \hline 2&1&2&1&3&0\\ \hline \end{array}}$$

Example of zero padding . The input, with dimensions 5×4, is padded with 2 zeros in each dimension. The output has dimensions 6×5. For a convolution kernel of dimensions 3×3, the output will have dimensions 5×4.

Stride

- Defines the amount of overlap between areas covered by the convolution kernel.
- A stride $s = 1$ means that two adjacent values of the convolution have been obtained by shifting one position (in either direction) the convolution kernel (no stride).
- A stride of s means that between two steps of the convolution in either direction, the kernel is shifted s positions.
- As a consequence, the output has lower dimensions. For a convolution with padding p and stride s, the output dimensions will be

$$\left\lfloor \frac{M_I + p - M_w + s}{s} \right\rfloor \times \left\lfloor \frac{N_I + p - N_w + s}{s} \right\rfloor \tag{5}$$

Stride

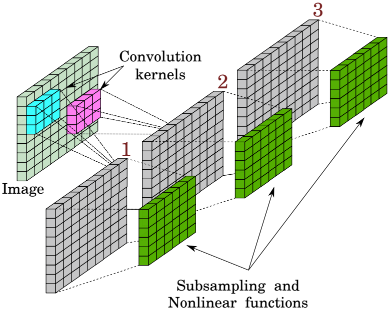
$$
\overset{\mathbf{I}}{\begin{array}{|c|c|c|c|c|c|c|}\hline 0&0&0&0&0&0&0\\\hline 0&1&2&1&3&1&0\\\hline 0&2&1&2&1&3&0\\\hline 0&1&2&1&3&1&0\\\hline 0&2&1&2&1&3&0\\\hline 0&0&0&0&0&0&0\\\hline\end{array}}
*
\overset{\mathbf{W}}{\begin{array}{|c|c|}\hline 0&1\\\hline 1&0\\\hline\end{array}}
=
\overset{\mathbf{W}*\mathbf{I}}{\begin{array}{|c|c|c|}\hline 0&2&3\\\hline 2&4&6\\\hline 2&2&3\\\hline\end{array}}
$$

Example of padding and stride, where $p = 2$ and $s = 2$. The resulting dimensions are 3×3.

Pooling

- Pooling is applied to the output to reduce the size of the convolution in a controlled way.
- This reduces the complexity of the structure.
- This is desired to limit the computational cost and the overfitting.
- The operation selects a square window with q pixels and maps them into a scalar.
- Next, the window is shifted to one or more positions and the operation is repeated.
- The most usual ones are *max pooling*, which selects the maximum value of the window, and *average pooling*.

Channels



- The paths from the original image to the convolution outputs are calles *channels*.
- We can apply the convolution again to all the channel outputs.
- An image with L channels can be mapped to an image with M channels.
- Each channel is a linear combination of convolutions.

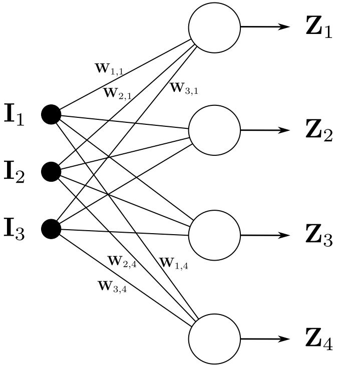
Formulation of the CNN convolution

- Assume an RGB image $\mathbf{I}$ with dimensions $[C_I, M_I, N_I]$, $C_I = 3$ corresponding to the three color channels.
- Every color of the image is convolved with several convolution kernels.
- Define $\mathbf{W}_{j,k}$, $0 \leq j \leq C_I - 1$ as the collection of convolution kernels that convolve input channel $\mathbf{I}_j$ and transform it into channel $\mathbf{Z}_k$. Then

$$\mathbf{Z}_k = \sum_{j=0}^{C_I-1} \mathbf{W}_{j,k} * \mathbf{I}_j + \mathbf{B}_k \tag{6}$$

where a bias array $\mathbf{B}_k$ is added after the convolution.

Formulation of the CNN convolution



Convolution layer with 3 input channels and 4 output channels.

- Each output channel is expressed as

$$\mathbf{Z}_k = \sum_{j=0}^{C_I-1} \mathbf{W}_{j,k} * \mathbf{I}_j + \mathbf{B}_k$$

- where convolution kernel $\mathbf{W}_{j,k}$ connects input channel j with output channel k.
- Bias matrices not represented.
- Outputs $\mathbf{Z}_k$ are applied an activation and a pooling.

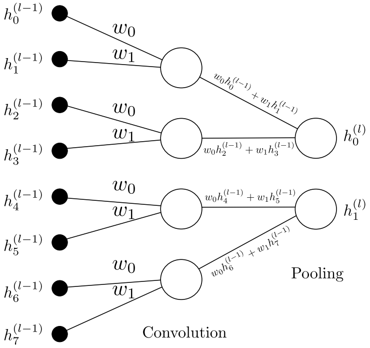
Formulation of the CNN convolution

- In a convolutional neural network, one convolutional layer computes the above operation with all the output channels of the previous layer as follows

$$
\begin{aligned}
&\mathbf{Z}_k^{(l)}[n] = \sum_{j=0}^{C^{(l-1)}-1} \mathbf{W}_{j,k}^{(l)} * \mathbf{H}_j^{(l-1)}[n] + \mathbf{B}_k^{(l)} \\
&\mathbf{H}_k^{(l)}[n] = \boldsymbol{\varphi}\left(\mathbf{Z}_k^{(l)}[n]\right) \\
&0 \leq k \leq C^{(l)} - 1
\end{aligned} \tag{7}
$$

- Here $\boldsymbol{\varphi}(\cdot)$ represents a pooling and a nonlinear activation, while padding and stride are assumed to be applied to the image and to the convoution kernels.

Interpretation as neural connectivity



Convolution in 1 dimension as sparse connectivity.

- Input $\mathbf{h}^{(l-1)}$ of dimension 8.
- Kernel $\mathbf{w}^{(l)} = \{w_0, w_1\}$.
- Stride $s = 2$, pooling $q = 2$.

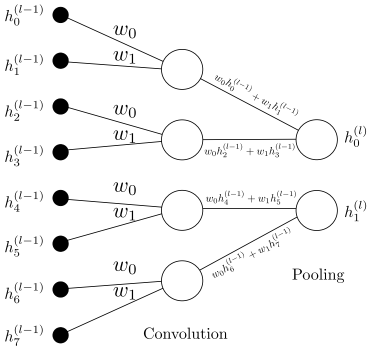
- We need a backpropagation formulation.
- A connectivity representation is convenient to make it easy.
- Equivalent connection matrix:

$$\mathcal{W}^{(l)} = \begin{pmatrix} w_0 & 0 & 0 & 0 \\ w_1 & 0 & 0 & 0 \\ 0 & w_0 & 0 & 0 \\ 0 & w_1 & 0 & 0 \\ 0 & 0 & w_0 & 0 \\ 0 & 0 & w_1 & 0 \\ 0 & 0 & 0 & w_0 \\ 0 & 0 & 0 & w_1 \end{pmatrix} \tag{8}$$

Interpretation as neural connectivity

- The first layer is the input
- The connections represent the convolution.
- $\left\lfloor \frac{M_I + p - M_w + s}{s} \right\rfloor = 4$
- The connection between layers 2 and 3 is the pooling.
- The convolution can be computed as

$$\mathcal{W}^{(l)^{\top}} \mathbf{h}^{(l-1)}$$



Convolution in 1 dimension as sparse connectivity.

Now, this new sparse connectivity notation can be used to *interpret* the backpropagation.

Backpropagation in CNN

- Assume a CNN structure that gives an output $\mathbf{o}$.
- Compute the gradient of the cost function wrt last kernel $\mathbf{W}_{j,k}^{(l)}$.
- We can represent the output as:

$$\mathbf{o} = \mathbf{o}\left(\mathbf{W}^{(L)^\top}\phi\left(\dots\mathbf{W}^{(l+1)^\top}\boldsymbol{\varphi}\left(\sum_{j=0}^{C^{(l-1)}-1}\mathbf{W}_{j,k}^{(l)} * \mathbf{H}_j^{(l-1)}[n]\right)\right)\right) \quad (9)$$

- All channels other than k and the biases have been ignored since they do not appear in the gradient wrt $\mathbf{W}_{j,k}^{(l)}$.
- Activation $\boldsymbol{\varphi}$ flattens the convolution to input dense layer $l+1$.

- Now, we assume that the convolution can be changed by some sparse connectivity matrix

$$\mathbf{o} = \mathbf{o}\left(\mathbf{W}^{(L)^\top}\boldsymbol{\phi}\left(\cdots\mathbf{W}^{(l+1)^\top}\boldsymbol{\varphi}\left(\sum_{j=0}^{C^{(l-1)}-1}\mathcal{W}_{j,k}^{(l)}\mathbf{H}_j^{(l-1)}[n]\right)\right)\right) \tag{10}$$

- And now the formulation is formally identical to the one of a dense neural network.
- Two differences
  1. Some matrices (the convolutions) are sparse.
  2. The outputs of the layers are matrices rather than vectors. Therefore, the backpropagated error is a matrix. Instead of δ, we will call it Δ.

Backpropagation in a CNN

- The backpropagation can the be derived in the same way as in Module 1 for a multilayer perceptron:

$$\mathcal{W}_{j,k}^{(l-1)} \leftarrow \mathcal{W}_{j,k}^{(l-1)} - \mu \mathbf{H}^{(l-2)} \boldsymbol{\Delta}^{(l-1)\top} - \mu\lambda\mathcal{W}_{j,k}^{(l-1)} \tag{11}$$

with the definition

$$\boldsymbol{\Delta}^{(l-1)} = \mathcal{W}_{j,k}^{(l)} \boldsymbol{\Delta}^{(l)} \odot \varphi'\left(\mathbf{Z}^{(l-1)}\right) \tag{12}$$

- But sparse matrix $\mathcal{W}_{j,k}^{(l)}$ represents a convolution.

Backpropagation in a CNN

- Therefore we can write

$$\mathbf{W}_{j,k}^{(l-1)} \leftarrow \mathbf{W}_{j,k}^{(l-1)} - \mu \mathbf{H}^{(l-2)} {\boldsymbol{\Delta}^{(l-1)}}^{\top} - \mu\lambda \mathbf{W}_{j,k}^{(l-1)} \tag{13}$$

with the definition

$$\boldsymbol{\Delta}^{(l-1)} = \mathbf{W}_{j,k}^{(l)} * \boldsymbol{\Delta}^{(l)} \odot \varphi'\left(\mathbf{Z}^{(l-1)}\right) \tag{14}$$