Lecture 5: (Deep) Neural Networks II

Soufiane Hayou

Saturday 27$^{\text{th}}$ May, 2023

Fully-Connected NNs are structure-agnostic: the input coordinates are interpreted in a similar fashion.

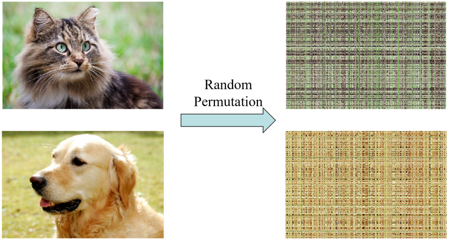


Figure: Illustration of the effect of permutation on data with spatial structure. Applying a permutation on the spatial indices destroys such structure, but fully connected neural networks treats these two cases equivalently, thus unsuitable to process such data types.

Fully-Connected NNs are structure-agnostic: the input coordinates are interpreted in a similar fashion.

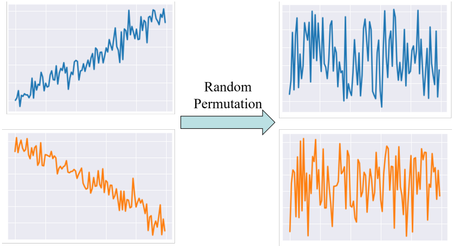


Figure: Another illustration with temporal data.

Convolutional Neural Networks

Convolutions

What is a convolution?

Convolutions

What is a convolution? Given two *infinitely-long* vectors

$w = \{w(i) : i \in \mathbb{Z}\}$ and $x = \{x(i) : i \in \mathbb{Z}\}$, we define their *discrete convolution* as

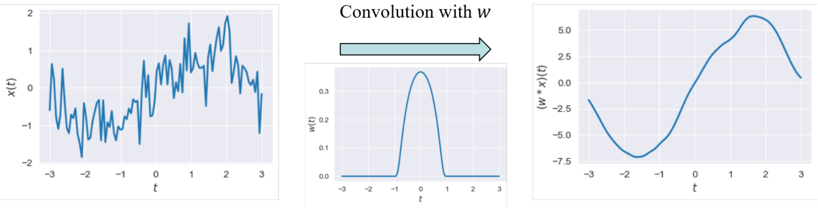
$$(w * x)(k) = \sum_{i=-\infty}^{\infty} w(i)x(k+i).$$

The convolution operator encodes how the shape of x is modified by w, the *filter*.

What is a convolution? Given two *infinitely-long* vectors $w = \{w(i) : i \in \mathbb{Z}\}$ and $x = \{x(i) : i \in \mathbb{Z}\}$, we define their *discrete convolution* as

$$(w * x)(k) = \sum_{i=-\infty}^{\infty} w(i)x(k+i).$$

The convolution operator encodes how the shape of x is modified by w, the *filter*.



Convolutional Neural Network: replace $W^\top x$ by $W * x$!

Convolutional Neural Network: replace $W^\top x$ by $W * x$!
However, $W * x$ requires that W and x are both infinite-length vectors!!

Convolutional Neural Networks

Convolutional Neural Network: replace $W^\top x$ by $W * x$!
However, $W * x$ requires that W and x are both infinite-length vectors!! We solve this by introducing Padding:

- Circular padding:

$$(w * x)(k) = \sum_{i=0}^{m-1} w(i)x(k+i), \qquad k = 0, \ldots, n-1,$$

  where x is periodically extended so that $x(j) = x(j-n)$ for $j \geq n$.

- Zero padding:

$$(w * x)(k) = \sum_{i=0}^{m-1} w(i)x(k+i-\lfloor m/2 \rfloor), \qquad k = 0, \ldots, n-1,$$

  where x is padded with zeros, so that $x(j) = 0$ for $j \notin \{0, \ldots, n-1\}$.

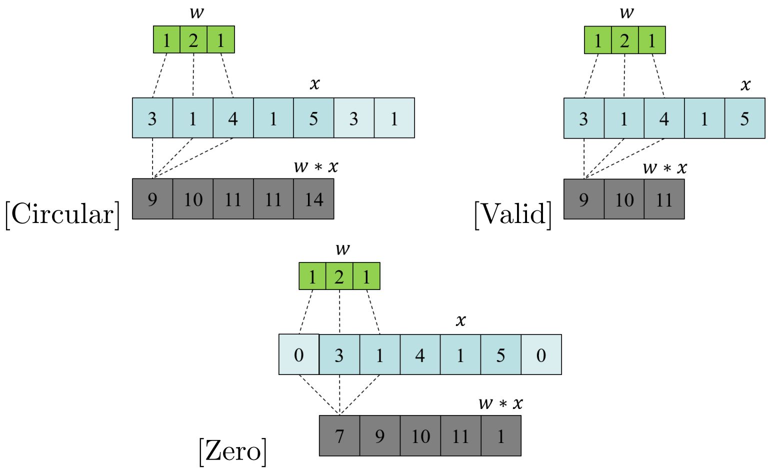


Figure: Illustration of different kind of paddings for CNNs.

2D Convolutions

- we want the discrete convolution to deal with images

we define 2D convolutions by

$$(w * x)(k, l) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} w(i, j) x(k + i, l + j)$$

Padding is also used for 2D convolutions as above.

$$w * x = \underbrace{\begin{pmatrix} w_0 & w_1 & w_2 & 0 & 0 \\ 0 & w_0 & w_1 & w_2 & 0 \\ 0 & 0 & w_0 & w_1 & w_2 \\ w_2 & 0 & 0 & w_0 & w_1 \\ w_1 & w_2 & 0 & 0 & w_0 \end{pmatrix}}_{C_w} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \tag{1}$$

- C_w (known as a Toeplitz matrix) highlights an important property of convolution networks: *weight sharing*. Unlike Fully-Connected DNNs, layer inputs share weights.
- Sparse representation when x is large and filter size is small: good for learning and storage (fewer parameters)
- Other important properties of CNNs: equivariance with translation etc. (review Lecture Notes of DSA5105 for more details)

To reduce the dimension of the layer, a Pooling layer can be used. The most commonly used version is *max pooling* with *stride* p, which corresponds to the following operation in 1D:

$$(T_{\mathrm{mp}}x)(k) = \max_{i=kp,\ldots,(k+1)p} x(i). \tag{2}$$

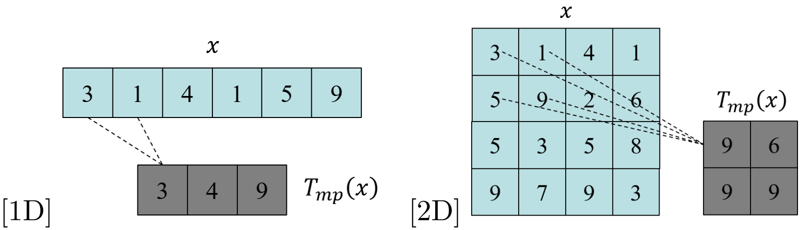


Figure: Max pooling operation in 1 and 2 dimensions.

$$x_0 = x$$
$$x_{t+1} = T_{\mathrm{mp}} T_{\mathrm{conv}} x_t, \qquad t = 0, \ldots, T-1$$
$$f(x) = T_{\mathrm{fc}} x_T,$$

where T_{mp} is a max pooling layer, T_{conv} is a convolutional layer, and T_{fc} is a fully-connected layer.

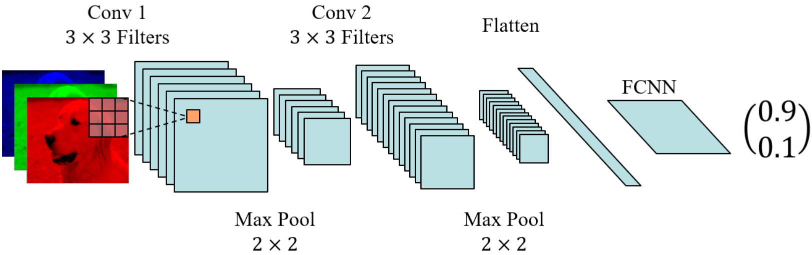


Figure: A basic deep CNN architecture.

Visualization of a trained Deep CNN

https://poloclub.github.io/cnn-explainer/

Google Colab

If you do not have access to a GPU and want to train large neural networks, you can use Google Colab which gives you access to free GPU memory for a period of time.

You can access an example of training a CNN on Google Colab from the link below:

`https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/images/cnn.ipynb`