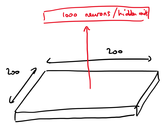
Video 28

Tuesday, November 10, 2020 2:45 PM

Why fully connected layers are not a good fit for vision?

input image 200x200 pixels, encoded in RGB



fully connected 1st hidden layer

input size: $200 \times 200 \times 3 = 120,000$

number of parameters: $200 \times 200 \times 3 \times 1000 = 120$ million

⇒ fully connected layers would require too many parameters.

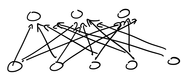
As we analyze one part of the image, we may use features / patterns that are useful to recognize other parts of the image.

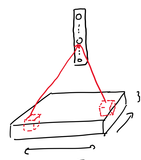
edges, corners, contours / any part of an object

⇒ We need to be able to detect these features / patterns in all image locations

Convolution layer

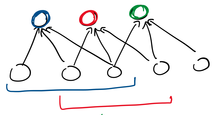
Fully connected

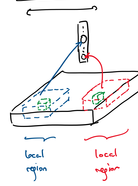




Locally-connected

↳ each neuron looks at a small region of the image

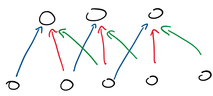




Convolution

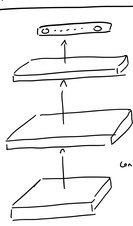
↳ each neuron looks at a small region of the image

↳ weights (connections between neurons) are shared between all image regions



Same visualization but with shared weights for the ≠ local regions.

Stack convolutions to obtain a deep neural network



Convolution operation

Assuming the input is 1D

Takes two input vectors:

$$(a * b)_t = \sum_z a_z b_{t-z}$$

vector, vector

scaling translation (controlled by z)

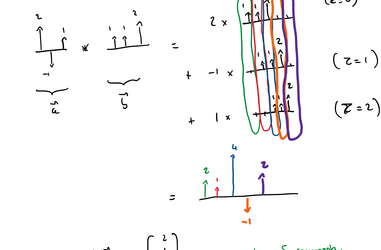
a and b are vectors with n components

$z \in 0..n-1$

$t-z \in 0..n-1$

Example: $\vec{a} = \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}$ $\vec{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

Translate & scale to visualize the convolution



$$(\vec{a} * \vec{b}) = \begin{bmatrix} 2 \\ 1 \\ 4 \\ -1 \\ 2 \end{bmatrix}$$

output has 5 components.

$z=0 \Rightarrow 0 \le t \le 2 \Rightarrow t \ge 0$

$t-z \in 0..2$, $z \in 0..2$ $\Rightarrow z=2 \Rightarrow 2 \le t \le 4 \Rightarrow t \in 4$

$t \in 0..4$ (where input had 3 components)

Matrix multiplication:

$$\begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} * \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix}$$

($z=0$, $z=1$, $z=2$)

More efficient, can be accelerated with GPUs.

2D convolution

output is 2D

$$(A * B)_{ij} = \sum_s \sum_t A_{st} B_{i-s, j-t}$$

2D inputs

Translate & scale: