Convolutional Neural Networks

23 September 2024 15:18

Convolution layers V. Layers in MLP.

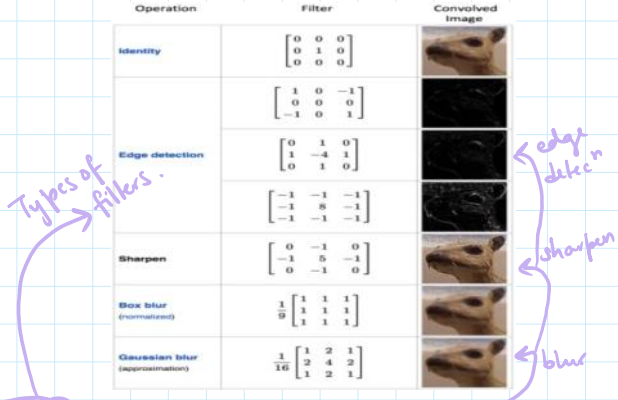
Only a subset of I/P is connected to O/P.
→ Each O/P captures only a SMALL SET of adjacent input. → local view → local patterns captured.

Every O/P is connected to every I/P
global view → O/P captures full I/P pattern.

No. of trainable params is HUGE.

one layer uses one filter
↓
only one type of local pattern is captured.

Many convolution layers capture different local patterns of I/P based on kernel.
→ Since "same kernel" slides over entire I/P, weight to be learnt <<< MLP.



1D Convolution

Input: 1D array
Kernel: Small 1D array
Output: 1D array

also: filter — small matrix to apply effects to image

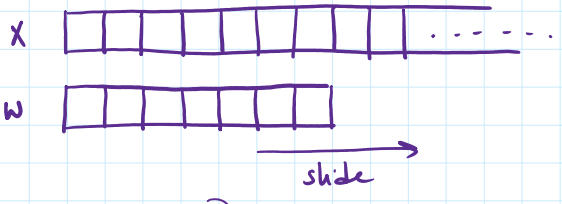
① Kernel slides over I/P data
② Elementwise multiplication (kernel × I/P values)
③ Summation of multiplicn — get single value
④ Repeat across entire I/P.

Stride: number of units the filter shifts @ each step

Padding → Valid: no padding — filter slides within bounds of I/P
Smaller output — filter can't go beyond edges

→ Zero: extra zeroes added around edge of input
output size = input size.

Assume kernel has dimension k_w.



Output when you slide:

$$Y_n = \sum_{j=0}^{k_w} X_{n+j} \cdot w_j$$

Express this as matrix multiplication:

(assume) $[w_0\ w_1\ w_2]$ ⊗ (ELEMENT WISE MULT) $[x_0\ x_1\ x_2 \dots x_{n-1}\ x_n]$

$\bar{w} \otimes \bar{x}$

for stride = 1, convolution weight matrix:

$$\begin{bmatrix} w_0 & w_1 & w_2 & 0 & \cdots & 0 & 0 & 0 \\ 0 & w_0 & w_1 & w_2 & \cdots & 0 & 0 & 0 \\ \vdots & & & & & & & \\ 0 & 0 & 0 & 0 & \cdots & w_0 & w_1 & w_2 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{bmatrix}$$

Convolution weight matrix — input

→ if stride = 2:

$$\begin{bmatrix} w_0 & w_1 & w_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & w_0 & w_1 & w_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & w_0 & w_1 & w_2 & 0 \end{bmatrix}$$

Extras:

for a CWM,

no. of rows = $\left(\frac{n - \text{size of kernel}}{\text{stride}}\right) + 1$

no. of columns = length of I/P vector (i.e. n)
↓
Independent of stride, kernel size.

matrix

↓

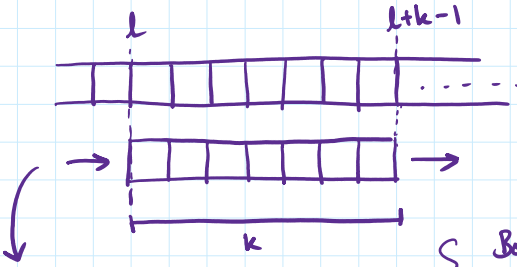
Independent of stride, kernel size.

Output size in 1D convolution:

|Input| = n

|Kernel| = k

|Stride| = s



After m strides,

$(ms) + (k-1) \leq (n-1)$

↓

Boundary condⁿ (max val.)

$m = \frac{n-k}{s}$

Initial pos. + m strides (each produces one o/p)

↓ SIZE OF OUTPUT

$m+1 = \frac{n-k}{s} + 1$ → Cuz of initial position

VALID PADDING

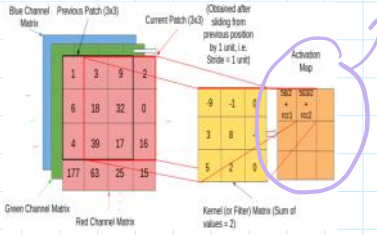
If ZERO PADDING

|Input| = n+2p (cuz p zeroes on each side)

SIZE OF OUTPUT

$|output| = \frac{n+2p-k}{s} + 1$

0	0	0	0	0	0
0	35	19	25	6	0
0	13	22	16	53	0
0	4	3	7	10	0
0	9	8	1	3	0
0	0	0	0	0	0



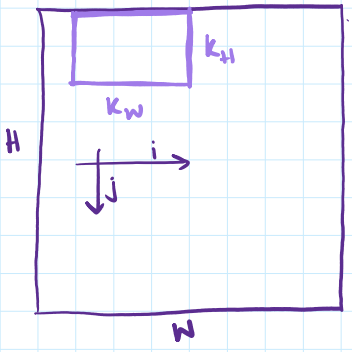
indicate "activated" regions → features specific to kernel detected in o/p.

2 Dimensions Now.

↳ Will just be sliding our tile on a wall.

↓

Stride is also 2D — $S = (S_W, S_H)$



Corner cases? $x \to x + S_W$, $y \to y + S_H$

RHS top corner: $x = x$, $y = y+1$

RHS bottom corner: STOP.

all ghost I/P replaced by 0.

→ now, $Y_{y,x} = \sum_{j}^{K_H} \sum_{i=0}^{K_W} X_{x+i, y+j} \cdot W_{ij}$

no params that affect bp.

doesn't change no. of trainable params

not trained during backprop.

Pooling ≃ Local translation invariance.

Minor changes in I/P features → diff. o/p feature map.

Now what?

Types of Pooling

↑ If object in image moves:

Max pooling: capture most prominent features (OR)

Average —"— : smooth out variations.

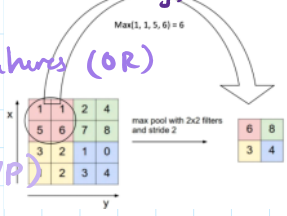
Downsampling — reduce size of data → reduce spatial dimensions. (O/P image size ↓)

→ return same info → lower precision.

less computation, less overfitting

POOLING → Max (max. value of each patch from feature map)

→ Average (same, but take avg.)

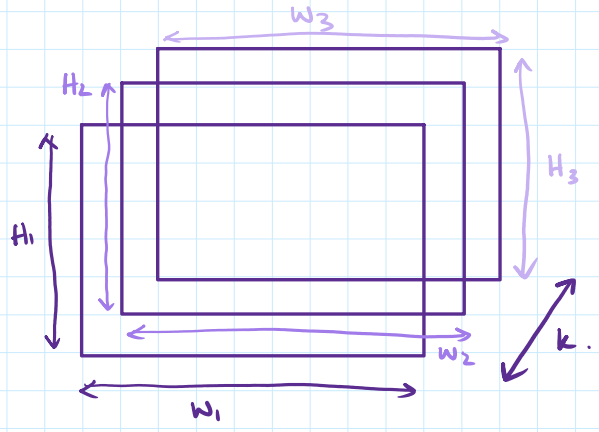


Depth = no. of filters in a layer (filter is small matrix that slides over I/P)

↳ How it works:

① Input image w/ 3 channels (RGB)

② In convolutional layer, apply multiple features, each filter = own feature map



① Input image w/ 3 channels (RGB)
② In convolutional layer, apply multiple features, each filter = own feature map
③ If you use 32 filters, O/P feature map = 32 features.

* Size of filter = F × F → Spatial extent.
no. of filters = k.
↳ Depth = k

Output size is just like before: → where you start.

$$W_2 = \frac{W_1 + 2p - F}{s} + 1$$

$$H_2 = \frac{H_1 + 2p - F}{s} + 1$$

Now solve koshan.

Ex1: Dimension of I/P = $H_1 \times W_1 \times B_1$ = $5 \times 5 \times 3$
No. of filter = 2 — k
Spatial extent = 3 — F
Zero padding = 1 — P
Stride = 2 — S

a) Output size = ?
b) Can we change stride to 3?

a) $W_2 = \frac{W_1 + 2p - F}{s} + 1 = \frac{(5) + (2)(1) - (3)}{(2)} + 1 = 3$

$H_2 = \frac{H_1 + 2p - F}{s} + 1 = \frac{(5) + (2)(1) - (3)}{(2)} = 3$

$D_2 = k = 2$
⇒ O/P size = $(3 \times 3 \times 2)$

b) if $s = 3$, $W_2 = H_2 = 4/3$ — not possible.

Ex2. Input size = $W_1 \times H_1 \times B_1$ = $288 \times 288 \times 3$
No zero padding — $p = 0$
Stride = 2 — $s = 2$
Output vol = $W_2 \times H_2 \times k$ = $143 \times 143 \times k$

a) filter size? — F = ?
b) max pooling layer connected to o/p volume.
filter = 3 × 3
stride = 2
o/p volume of max pooling? ↳ (W_3, H_3, D)

a) $W_2 = \frac{W_1 + 2p - F}{s} + 1 \Big] \; 143 = \frac{288 - F}{2} + 1$

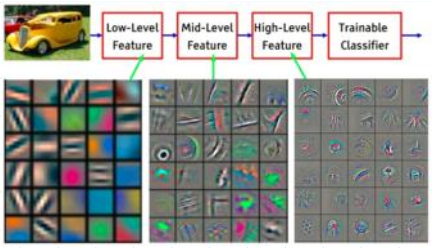
$F = 288 - 2(142)$
$\boxed{F = 4}$

b) Input: $(143, 143, k)$
$s = 2$
$f = 3$

$H_3 = \frac{H_2 + 2p - F}{s} + 1 = \frac{143 - 3}{2} + 1 = 71$

$W_3 = 71$ too.
⇒ o/p volume = $(71 \times 71 \times k)$

Stacking layers. Why?
→ CNN learns more complex features
↓
initially — simple features → build onto more high level.



initially — simple features → build onto more high level.

Also: images shrink after each layer: size(output) < size(input)

filters @ each layer remain the same (they find larger informed patterns)

I/P is weighted sum from prev. layers.

How many parameters in max pool layer?

none.

It doesn't need to learn anything.

size of O/P image $O = \frac{I - P_s}{S} + 1$ (I = input, P_s = pool size, S = stride)

just replace filter size (F) w/ pool size | same formula.

Calculating Number of Parameters.

W_c = no. of weights in convol. layer
B_c = no. of biases
P_c = no. of parameters.

k = size of kernels (filters)
N = no. of kernels (filters)
C = no. of channels in I/P.

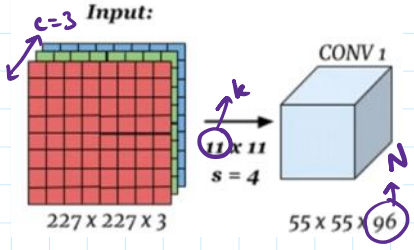
$P_c = W_c + B_c$

$W_c = k^2 \times C \times N$ (each filter has (k×k) elements; N → no. of filters doing this; each of these filters operates on each channel.)

W_c = no. of filters and no. of channels and size of filter

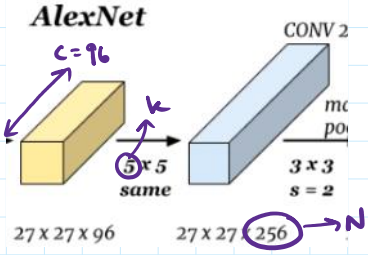
$B_c = N$ → no. of biases = no. of kernels (cuz every filter has its own bias)

Example:



$W_c = (11 \times 11) \times (3) \times (96) = 34\,848$
$B_c = N = 96$
$P_c = 96 + 34848 = 34944.$

Example:



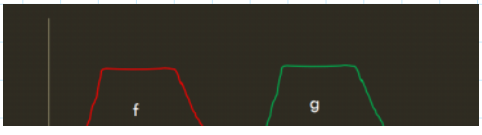
$W_c = k^2 \times C \times N = (5 \times 5) \times (96) \times (256) = 614400$
$B_c = N = 256$
$P_c = 256 + 614400 = 614656.$

— x —

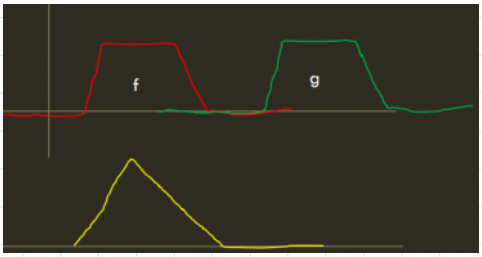
Ok bro.
But what is a "convolution"?
*sigh.

Conv = mechanism to blend two functions of time

$$(f*g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau$$



"Summation of point-wise product of two functions' values subject to traversal."



summation of pointwise product of two functions values subject to traversal."

Now,

cross correlation = comp. of two diff time series to detect correl^n b/w metrics w/ same max et min.

auto correlation = same but w/ itself.

CNN is mostly just cross correlation.

↓

"measure of similarity b/w a signal & a time delayed version of another."

— x —

CNN for text:

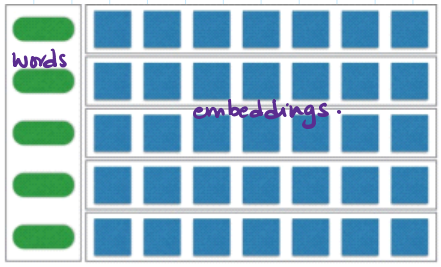
word embedding + CNN = text classification.

word embedding → use to represent a document.

CNN → find features to discriminate documents → non linearity = heck yeah

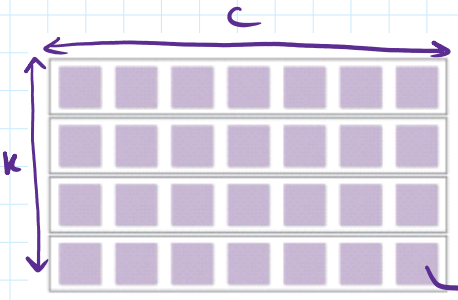
3 key aspects:

- word embedding layer
- convolution model.
- Fully connected layer (for interpretation of extracted features) — softmax.



Stage 1: The representation of your Input

① Embed.



Stage 2: This is a filter of length 4

Multiple filters? Can.
Same length? Can.
↓
produce multiple filter maps that's all.

② Can make 2 also.

k = no. of rows
c = no. of cols.

→ This is embedding btw

CNN - 1D vs 2D Convolution

	1D Convolution	2D Convolution
Input Data	Sequential data (e.g., time series, audio signals)	Image data (e.g., RGB images, grayscale images)
Input Shape	(Length, Channels)	(Height, Width, Channels)
Filter Shape	(Filter Length, Channels)	(Filter Height, Filter Width, Channels)
Operation	Slides filters over the sequence	Slides filters over the 2D spatial dimensions of the image
Stride	Step size for moving the filter along the sequence	Step size for moving the filter along height and width
Padding	Adds values (usually zeros) at the start and end of sequence	Adds values (usually zeros) around the image border
Output Shape	(Output Length, Output Channels)	(Output Height, Output Width, Output Channels)
Computational Complexity	Generally less complex due to fewer dimensions involved	Generally more complex due to higher dimensional data
Applications	Time series analysis, speech recognition, text processing	Image classification, object detection, image segmentation