Nabeel Mohamed
7547

# Sheet [3]

[1] Padded :- [0, 3, 4, 6, 3, 2, 3, 4, 0]

1st element :- [0, 3, 4] * [1, 2, -1] = 2

2nd :- [3, 4, 6] * [1, 2, -1] = 5

3rd :- [4, 6, 3] * [1, 2, -1] = 13

4th :- [6, 3, 2] * [1, 2, -1] = 10

5th :- [3, 2, 3] * [1, 2, -1] = 4

6th :- [2, 3, 4] * [1, 2, -1] = 4

7th :- [3, 4, 0] * [1, 2, -1] = 11

→ output :- [2, 5, 13, 10, 4, 4, 11]

2) Length output = $L - f + 1$

amount of padding (Size) = $\frac{f-1}{2}$

3) $\begin{bmatrix} 3 & 5 & 2 & 2 \\ 1 & -1 & 3 & -1 \\ 0 & 6 & 3 & 7 \\ 10 & -3 & 5 & 6 \end{bmatrix} \circledast \begin{bmatrix} 2 & 1 \\ -1 & 2 \end{bmatrix} = \begin{bmatrix} X_1 & X_2 & X_3 \\ X_4 & X_5 & X_6 \\ X_7 & X_8 & X_9 \end{bmatrix}$

$X_1 = \begin{bmatrix} 3 & 5 \\ 1 & -1 \end{bmatrix} * \begin{bmatrix} 2 & 1 \\ -1 & 2 \end{bmatrix} = 6+5-1-2 = 8$

$X_2 = \begin{bmatrix} 5 & 2 \\ -1 & 3 \end{bmatrix} * \begin{bmatrix} 2 & 1 \\ -1 & 2 \end{bmatrix} = 10+2+1+6 = 19$

$X_3 = \begin{bmatrix} 2 & 2 \\ 3 & -1 \end{bmatrix} * \begin{bmatrix} 2 & 1 \\ -1 & 2 \end{bmatrix} = 4+2-3-2 = 1$

$X_4 = \begin{bmatrix} 1 & -1 \\ 0 & 6 \end{bmatrix} * \begin{bmatrix} 2 & 1 \\ -1 & 2 \end{bmatrix} = 2-1+12 = 13$

$X_5 = \begin{bmatrix} -1 & 3 \\ 6 & 3 \end{bmatrix} * \begin{bmatrix} 2 & 1 \\ -1 & 2 \end{bmatrix} = -2+3-6+6 = 1$

$X_6 = \begin{bmatrix} 3 & -1 \\ 3 & 7 \end{bmatrix} * \begin{bmatrix} 2 & 1 \\ -1 & 2 \end{bmatrix} = 6-1-3+14 = 16$

$X_7 = \begin{bmatrix} 0 & 6 \\ 10 & -3 \end{bmatrix} * \begin{bmatrix} 2 & 1 \\ -1 & 2 \end{bmatrix} = 6-10-6 = -10$

$X_8 = \begin{bmatrix} 6 & 3 \\ -3 & 5 \end{bmatrix} * \begin{bmatrix} 2 & 1 \\ -1 & 2 \end{bmatrix} = 12+3+3+10 = 28$

$X_9 = \begin{bmatrix} 3 & 7 \\ 5 & 6 \end{bmatrix} * \begin{bmatrix} 2 & 1 \\ -1 & 2 \end{bmatrix} = 6+7+5+12 = 20$

$$\therefore \text{output} = \begin{bmatrix} 8 & 19 & 1 \\ 13 & 1 & 16 \\ -10 & 28 & 20 \end{bmatrix}$$

**4** assume 4 bytes per parameter (bias)

① input layer: $512 \times 512 \times 3$

② Convolutional layer 1: size(filter) $= 11 \times 11 \times 3$
# of filters $= 96$
Parameters $= (11 \times 11 \times 3)(96) + 96 = \boxed{34944}$
memory $= (34944)(4) = \boxed{139776}$ bytes

③ Max pooling 1: size(pooling) $= 3 \times 3$
stride $= 2$

④ Convolutional 2: filter size $= 5 \times 5 \times 96$
# of filters $= 256$
parameters $= (5 \times 5 \times 96)(256) + 256 = 614656$
memory $= (614656)(4) = \boxed{2458624}$ bytes

⑤ Max pooling 2: size of pooling $= 3 \times 3$
stride $= 2$

⑥ Convolutional 3:- Size(filter) = 3x3x256
# of filters = 384
parameters = (3x3x256)(384) + 384 = 885120
memory = (885120)(4) = 3540480 bytes

⑦ Convolutional 4:- Size (filter) = 3x3x384
# filter = 384
Parameters = (3x3x384)(384) + 384 = 1327488
memory = (1327488)(4)

⑧ Convolutional 5:- Size (filter) = 3x3x384
# filter = 256
parameters = (3x3x384)(256) + 256 = 884992
Memory = (884992)(4)

⑨ Max pooling:- Size pooling = 3x3
stride = 2

⑩ Fully connected 1:- # nurons = 4096
Parameters = (6x6x25)(4096) + 4096 = 37752832
memory = (37752832)(4)

⑪ Fully Connected 2:- # neurons = 4096
parameters = (4096)(4096) + 4096 = 16781312
memory = (16781312)(4)

(12) output:-

# neurons = # of classes

Parameters = (4096)(# classes) + # of classes

memory = (parameters)(4)

---

5 (a) $[[1, 0, -1], [1, 0, -1], [1, 0, -1]]$

* this filter indicate the intensity difference bet. adjacent pixels in the vertical direction.

(b) Commonly used filter for edge detection as (prewitt).

---

6 $\frac{256}{6} = \sqrt{42}$

---

7 * fully connected layers in CNN are commonly used in classification and other tasks, but have some limitations as fixed input size and high parameter count.

* Alternatives like GAP (global average pooling) and fully convolutional networks reduce parameters

* GoogleNet (inception) was the first CNN to replace fully connected layers with "GAP".

8] $RF = (W-1)(\text{stride}) + 1$

1st layer:- $RF = (3-1)(1)+1 = \boxed{3}$

2nd layers:- $RF = (3-1)(1)+3 = \boxed{5}$

3rd layers:- $RF = (3-1)(1)+5 = \boxed{7}$

∴ the RF of 3rd layer = 7x7 covering same ~~convolutional~~ region as single Convolutional layer

∴ Stack of ~~three~~ of ~~3x3~~ convolutional layers with stride 1 has the same effective RF as one 7x7 convolutional layer.

---

9] VGGNet despite its simplicity & straightforward architecture, tends to use more memory compared to more recent architectures like RESNet and Google Net. and better performance due to their innovative architecture design. ResNet introduce residual connections that mitigate the vanishing gradient problem allowing for deeper networks without degradation in performance.

[10] Early CNN network were limited from comparisons 100 layer or more due to the vanishing gradient problem, where gradients diminish as they propagate through many layers. Residual connections introduced in ResNet allowed for deeper network by enabling easier gradients flow, ResNet the first to successfully train more than 100 layer

---

[11] (1) Number of layers :- For MNIST, a relatively simple dataset, a moderately sized network with a few layers should suffice

(2) Types of layers :- (a) input layer :- Single input layer 28×28 neuron
(b) Convolutional :- 1-2 layers with small filter (3×3)
(c) Pooling :- after each conv. layer apply max pooling (2×2) to down sample the feature maps.
(d) Fully connected :- 1-2 fully connected layers to learn higher level from the features
(e) output :- Softmax output layer with 10 neurons, representing probabilities (0-9)

(3) activation functions :- (a) Convolutional layer :- RELU commonly used due to its effectiveness in training
(b) Fully connected :- RELU is used, followed by Softmax

(4) optimization :- Adam optimizer is popular choice for neural networks, it adapt learning rates for each parameter → faster convergence.

(5) Loss function: Since MNIST is multi-class classification problem, Categorical cross-entropy is commonly used. it measures that dissimilarity between the predicted class probabilities & the true labels

(6) Evaluation Metrices: accuracy:- the % of correctly classified images is a common evaluation metric for classification task