Page No.
Date

* CNN

→ Filters / Kernel represent the eyes.
→ Always keep odd size kernel.
→ We do not explicitely mention the importance of features.
→ In convolutional layer, point to point multiplication takes place.
→ Output size of convolutional layer is:

$$\left\lfloor \frac{n - f + 2p}{s} \right\rfloor + 1$$

→ It is usually smaller in size than input so information is lost especially the corner pixels features.
→ So, padding was introduced to give equal importance to all pixels.
→ Padding happens before the convolutional operation.
→ Padding is always given value 0 as then it will be seen as a feature.

→ 3 types of padding

① Valid [No padding]

② Same [1x1 padding]

Keep the height & width same as the input.

③ Custom padding

ZeroPadding2D(padding = (2, 2)) ⇒ 2×2

→ Stride controls the movement of kernel on the image.

Q) $n = 7 \quad f = 3 \quad s = 2 \quad p = 0$

$$\left\lfloor \frac{n - f + 2p}{s} \right\rfloor + 1 = \left\lfloor \frac{7 - 3 + 0}{2} \right\rfloor + 1 = 2 + 1 = 3 \times 3$$

Q) $n = 8 \quad f = 2 \quad s = 2 \quad p = 0$

$$\left\lfloor \frac{n - f + 2p}{s} \right\rfloor + 1 = \left\lfloor \frac{8 - 2 + 0}{2} \right\rfloor + 1 = 3 + 1 = 4$$

4×4

→ ReLU converts the features which do not contribute or which may deviate the predictions is neutralized to 0.

→ Pooling is of 3 types:

① MaxPooling
② MinPooling → not used in practical
③ Average Pooling

→ default value for sliding window of Pooling is 2x2.

→ Pooling down samples the feature map size while retaining the most important or most highlighted features.

→ If no. of channels is 1, then it might be grayscale or binary image.

→ Symbols Representation

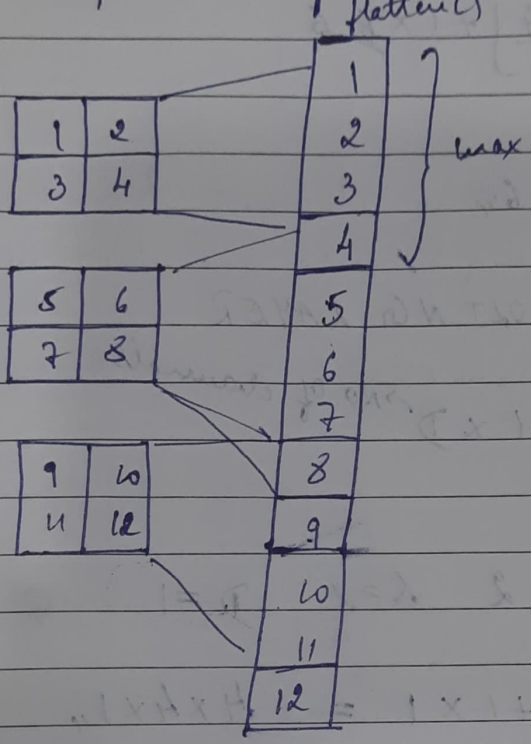
Conv ⊗
ReLU ⊘
Pooling ▶

→ Flatten() is a keras function which takes a numpy matrix to a numpy 1D array.

Page No.
Date

★ GLOBAL MAX POOLING & GLOBAL AVG POOLING

→ Before passing to MLP, we can use global max/avg pooling to pass only most important features.



★ OUTPUT OF CONV LAYER

Q) $n = 32 \times 32 \times 3$ $f = 5 \times 5$ $D = 6$ $S = 1$ $p = 0$

$$O = \left\lfloor \frac{n - f + 2p}{S} \right\rfloor + 1 \times D \rightarrow \text{no of filters}$$

$$= \left\lfloor \frac{32 - 5 + 0}{1} \right\rfloor + 1 \times 6$$

$$= 28 \times 28 \times 6$$

Q) $n = 32 \times 32 \times 3$ $f = 5 \times 5$ $D = 6$ $s = 1$ $p = 2$

$$O = \left[\frac{n - f + 2p}{s}\right] + 1 \times D$$

$$= \left[\frac{32 - 5 + 4}{1}\right] + 1 \times 6$$

$$= 32 \times 32 \times 6$$

★ OUTPUT OF POOLING LAYER

$$O = \left[\frac{n - f}{s}\right] + 1 \times D \quad \rightarrow \text{No of channels}$$

Q) $n = 8 \times 8 \times 1$ $f = 2$ $s = 2$ $D = 1$

$$O = \left[\frac{8 - 2}{2}\right] + 1 \times 1 = 4 \times 4 \times 1$$

★ TYPES OF CONVOLUTIONAL

① Standard conv

② Separable conv

2 types:

a) Spatially Separable conv:

dividing : 3×3 filter

3×1 1×3

eg: 3×3 kernel = $\begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$

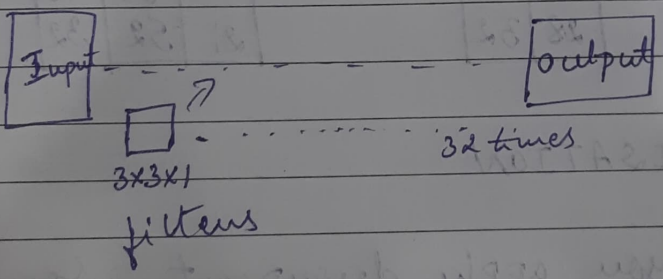
But all kernels are not separable into two, so we have depth-wise separable conv.

b) Depth-wise separable conv.

There 2 types:

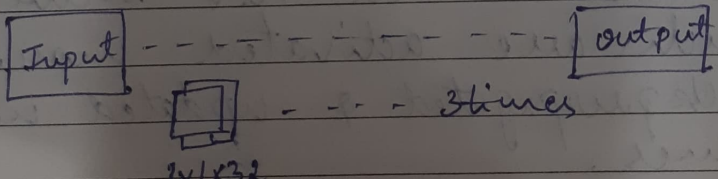
(i) Depth-wise conv

3×3×32 filters



(ii) Point-wise conv

3×3×32 filters



NOTE: These techniques are used for different scenarios and in each technique feature maps values differ.

③ ★ Transpose conv

→ It is used for upsampling in U-Net architecture.

eg:

1	2
3	4

5	6
7	8

⇓

5	6	
7	8	

+

	10	12
	14	16

+

15	18	
21	24	

+

	20	24
	28	32

=

5	16	12
22	60	40
21	52	32

★ REGULARISATION

→ When you apply dropout, some neurons are dropped randomly so some weights are activated little more than some others due to which the neurons which are activated more are backpropagated & updated more times.

→ So, avoid this scaling by variation proposed that after training, multiply all weights into the dropout percentage.

→ Another solution is, do not wait till training is over. For each neuron's output scale up by multiplying it by (1/(1-p)) (p = dropout %). But this method is not used often because ultimately the weights are reverted back to the same values without dropouts. So, the whole purpose of dropouts vanishes.

→ Dropout can be applied to input & hidden layers.

Dropout is always applied after the 1st conv + ReLU + Pooling, because only after pooling will we get the 1st output which is final with very important features.

If your dropout is applied to hidden layers, then it comes after each dense layer.

→ There 3 types of techniques of Regularization:

① L1 Regularization:

→ When model is overfitted, the model has learnt the training data perfectly due to which it has a perfect curve so in testing data the cases where it is not exactly on the curve but closer to it make the model miss predict. This happens because the weights are too perfect in the beginning.

→ Since, the weights are too perfect we have to punish it with some value called L1 / L2 / L1+L2.

$$L = \underbrace{\frac{1}{N} \sum_i L_i (f(x_i, W), y_i)}_{\text{Loss}} + \lambda R(W)$$

Regularization term

$$L1 = \lambda |W|$$

② L2 regularization:

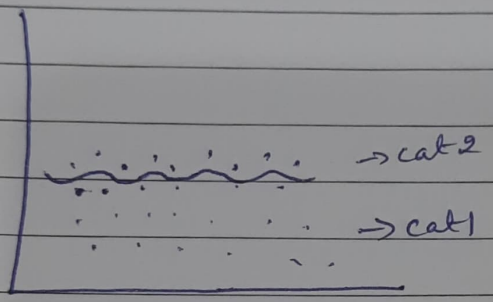
$$L2 = \lambda (W^2)$$

③ L1+L2 Regularisation (Elastic Net):

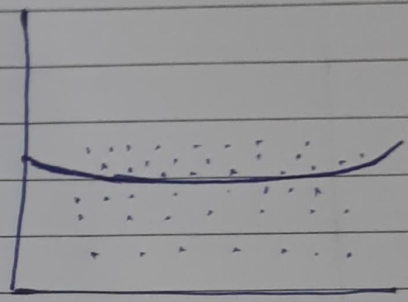
$$L_1 + L_2 = \lambda [L_1 |W| + \frac{1}{2} W^2] \rightarrow \text{Performance very good}$$

→ Kernel regularizer controls the weight initialisation in MLP.

→ Weights are updating slowly due to which the loss is getting slowly reduced due to which you get a perfectly balanced curve.



perfect curve (overfitting)



Perfectly balanced curve