11 March 2021 09:15

(0)

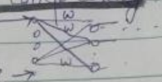
CNN - CONVOLUTIONAL NEURAL NETWORK

Introduction

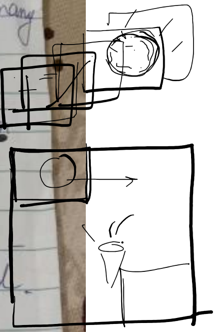
→ We studied fully connected MultiLayer Perceptron MLP Neural Network, using sigmoid function to activate a neuron, and back propagation to adjust weights. They are not suitable for images because :-

(1) One pixel going to one perception at input stage means for 1024 x 1024 x 3 (RGB) of them at the input and then fully connected at hidden layer.
→ Excessive Computation complexity. Too many weights to be trained!

pixel 1



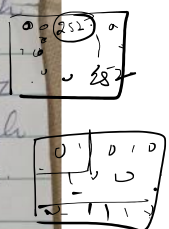
pixel 3145728 →



(2) MLP architecture does not exploit spatial correlation which is typical in images.

(3) It cannot make sense of shifts in positions - translations of an object cannot be handled.

→ CNNs use filters - 2D slices of weights which are much fewer than number of pixels. These filters are applied all over the image. Their weights are "learnt" by the CNN through supervised learning with back propagation. Thus, CNNs are most suitable for image analysis.



(i) CNNs are the only ML technique that automatically learn features in an image. 1024 x 1024

(ii) The filter weights are shared by different portions of the image. Thus, there are fewer weights. Ofcourse there are several filters that are learnt for any given image. (25 x 25)

(iii) CNNs are translation - invariant in learning features

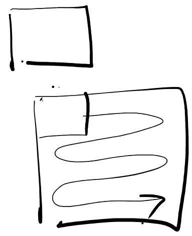
(iv) CNNs are deep rather than wide as in MLP

(v) CNNs typically use ReLU activation function instead of sigmoid to avoid vanishing gradient problem.

Topic: Convolutional Neural Network CNN

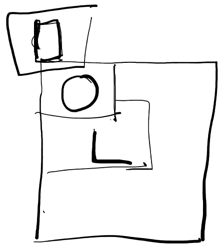
Date : Page No. :

1



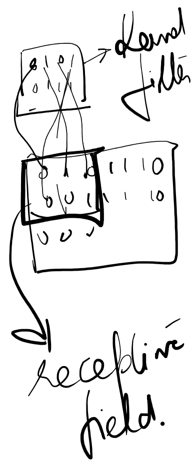
- Convolutional Neural Networks (CNN) are deep neural networks with a series of NN layers:
I/P → convolutional, nonlinear, pooling... fully connected → o/p
Thus, the input is fed into the first convolutional layer and after a series of transformations finally pass through a fully connected layer for classification at the output.

- CNNs are most used for image classification like the "visual cortex" of human brain where different neuronal cells fire in the presence of different features of the visual field (e.g. some fire in the presence of edges), so also, in a CNN different parts of an input image trigger different sets of neurons in the conv. layer. Thus, each convolutional neuron has its own "receptive field" in the image. Note that ~~this is in~~ contrast with a Multi Layer Perceptron Feed Forward NN, which is fully connected from input to hidden layers.

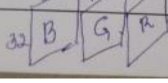


→ Discrete convolution is $\sum_{m=-\infty}^{+\infty} f[m]\, g[n-m]$ → Shift, multiply, add ←

- Despite the name "Convolution", the mathematical function that is performed is a "Dot Product" between an (array of numbers) and the receptive field of an image.
  - → The array is called Filter or Kernel
  - → The numbers in this array are the weights



So, assume the image is of size 32 pixels × 32 pixels × 3 (for each component of RGB) with each pixel an 8-bit integer.



NEW DAY

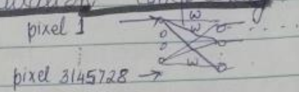
0

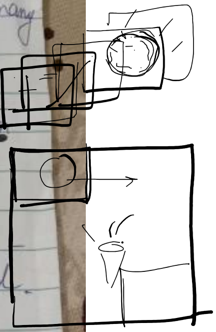
CNN - CONVOLUTIONAL NEURAL NETWORK

Introduction

→ We studied fully connected Multi-Layer Perceptron MLP Neural Network, using sigmoid function to activate a neuron, and back propagation to adjust weights. They are not suitable for images because:—

(1) One pixel going to one perception at input stage means for 1024 × 1024 × 3 (RGB) of item at the input and then fully connected at hidden layer.
⇒ **Excessive Computation complexity**. Too many weights to be trained!

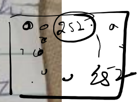


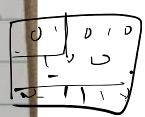


(2) MLP architecture does not exploit spatial correlation which is typical in images.

(3) It cannot make sense of shifts in positions - translations of an object cannot be handled.

→ CNNs use filters — 2D slices of weights which are much fewer than number of pixels. These filters are applied all over the image. Their weights are "learnt" by the CNN through supervised learning with back propagation. Thus, CNNs are most suitable for image analysis.





(i) CNNs are the only ML technique that **automatically learn features** in an image. 1024 × 1024

(ii) The filter **weights are shared** by different portions of the image. Thus, there are fewer weights. Of course there are several filters that are learnt for any given image. (25 × 25)

(iii) CNNs are **translation-invariant** in learning features

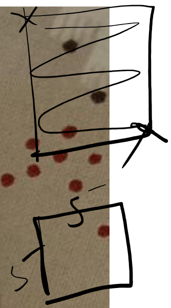
(iv) CNNs are **deep** rather than wide as in MLP

(v) CNNs typically use **ReLU activation function** instead of sigmoid to avoid vanishing gradient problem.

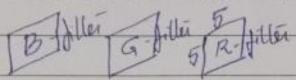
Topic: CNN

Date : Page No. :

2

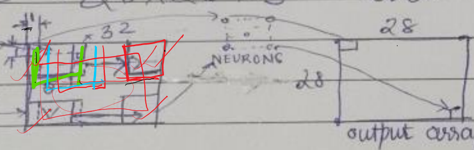


The Kernel is an array of (5 x 5 x 3) weights.



The filter slides through the image from Top Left to Bottom Right, allowing each convolutional neuron to perform a dot product in its own receptive field of 5x5x3 pixels with the filter. Thus each convolutional neuron performs 75 multiplications & adds them up to produce a single no. A bias is also added.
Assume the filter slides by one pixel, right & down.

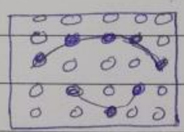
- The kernel, like a flash light moves over the image to produce (32 − 5 + 1) x (32 − 5 + 1) x 3 = 28 x 28 x 3 such numbers.



28 x 28 x 1 array for each depth, R, G, B.

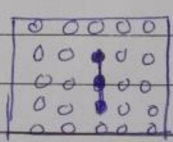
For a single filter, we get an output of 28 x 28 x 1. With n filters, we can get 28 x 28 x n such matrices, from as many neurons as 784.

- The filters represent different kinds of possible features in the image such as
→ curves



[black dots have high value say 45
white dots are zero.

→ straight edges.



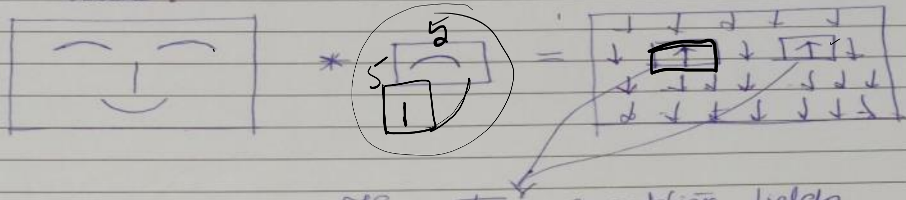
→ colour :- Say red region for cheeks, in R-FILTER

NEW DAY

Topic: CNN
Date:
Page No.:

3

The imp. point is → if the input image contains that feature in the receptive field for a conv. neuron, then it will output a large value.



These two receptive fields will give high value ↑ for above filter, others will give low values ↓

Thus, the dot product discovers the "correlation" between filter and image to learn certain aspects of the image.

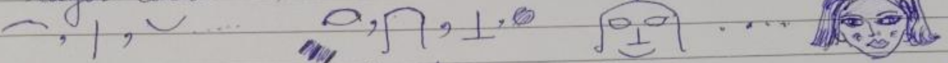
When there are several filters, then more features can be learnt.

— The significance of a CNN is that we do not have to pre-design or handcraft these filters, rather the CNN learns the weights of the filters by itself, through supervised training with backpropagation.

— The CNN is a deep architecture with multiple convolution layers. As we go further in the series from input to output, each conv layers gathers as input the features extracted from previous layer and increases its "receptive field" size. The aim is to

detect higher forms of features. For example
~~input conv layer~~

input conv layer detects →	next conv layer detects →	next conv. layers → . . . →	final conv layer
curves & edges	combination of basic features - semi-circle, curve + edge, hor + vert. edge, red areas, patterns	higher combination of features, facial parts	a proper face with say brown skin, black eyes, dark hair

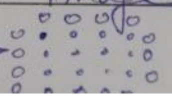


Thus, a deep NN begins with low level features like curves and edges and gradually increases the receptive fields to extract high level features such as face, paws, wings, beak, blue eyes, red portions, patterns of colour & shades etc.

PADDING.
— It may be noticed that the image pixels at the edges are not touched by the kernel as frequently as the middle. This means they are not searched thoroughly for features. To enhance their chances, "padding" is followed. This overcomes two problems.

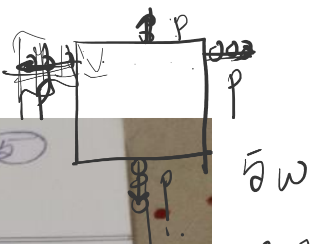
1) Losing corner information
2) Shrinking output of convolution from $n \times n$ image to $(n-w+1) \times (n-w+1)$, where n :- image dimension, w :- filter dimension.

— When there is padding, additional zeros are added on the border of the image.

padding →


Topic.: CNN. (5)

Date : ...... Page No. : ......



- For a padding width of size p, the output image is $(n+2p-w+1)\times(n+2p-w+1)$

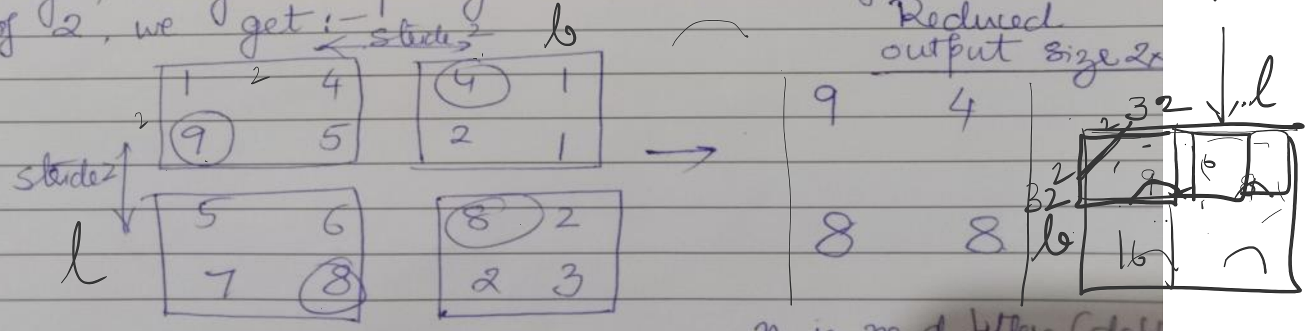
- Valid padding means no padding while 'Same' padding is when the pad restores size of original image : $n+2p-w+1=n$ Thus $p=\frac{w-1}{2}$.

STRIDE :- So far filters were moving over image with a stride = 1. For a stride of s :— Output is $\left[\frac{n+2p-w}{s}+1\right]\times\left[\frac{n+2p-w}{s}+1\right]$, where s is both horizontal & vertical stride. Stride reduce the size of an image, taking advantage of spatial relatedness.

conv

POOLING Layer.

Once a feature is captured by convolution layer, it is not necessary to use the entire picture but certain highlights only. Pooling layer uses special filters to capture the average or maximum of the relevant numbers. Thus for a 4x4 block within an image, using max pooling with a 2x2 filter, stride of 2, we get :—



n_c is no. of filters (depth)

In general if $l \times b$ is image block, w is filter width & stride then output $=\left\{\frac{l-w}{s}+1\right\}\times\left\{\frac{b-w}{s}+1\right\}\times n_c$

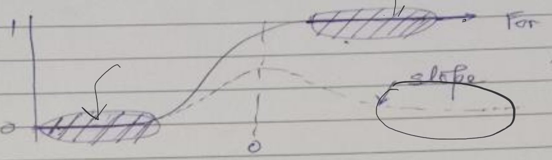
31379131

Topic.: CNN 6

Date : Page No. :

ReLU { Rectified Linear Unit Activation Function }

The sigmoid activation function has a narrow transition between 0 and 1. Hence it is called a "soft switch"



For very large or very small values the slope is 0.

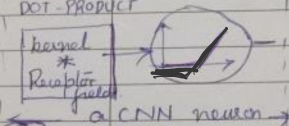
$s'(y) = (1+e^{-y})(1-\sigma y)$

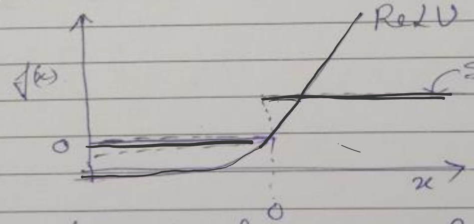
Since output error is propagated gradually from output to input by differentiating the cost function (squared error) w.r.t. weights and their activation function $f'(z^l)$ at each layer l, gradually the terms become zero for highly activated neurons. Thus the weights are hardly adjusted in consecutive iterations.

$\frac{dC}{dW_l}$,

To avoid this situation, one must use a non linear function with +ve and constant slope.

DOT-PRODUCT





$f(x) = max(0,x)$

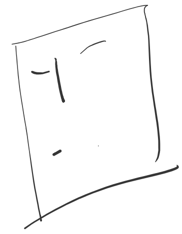
① ReLU
$g(z) = max(0,z)$

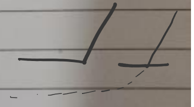
② Leaky ReLU
$g(z)$
$max(\epsilon z, z)$
$\epsilon << 1$

③ $max(\alpha(e^z-1), z)$
$\alpha << 1$

Now the slope is large enough to remain significant for learning, so that weights are adjusted according to Eq. $W_{l(new)} = W_{l(old)} - \eta\left(\frac{dC}{dW_l}\right)$, at each layer l,

η being the user defined learning rate





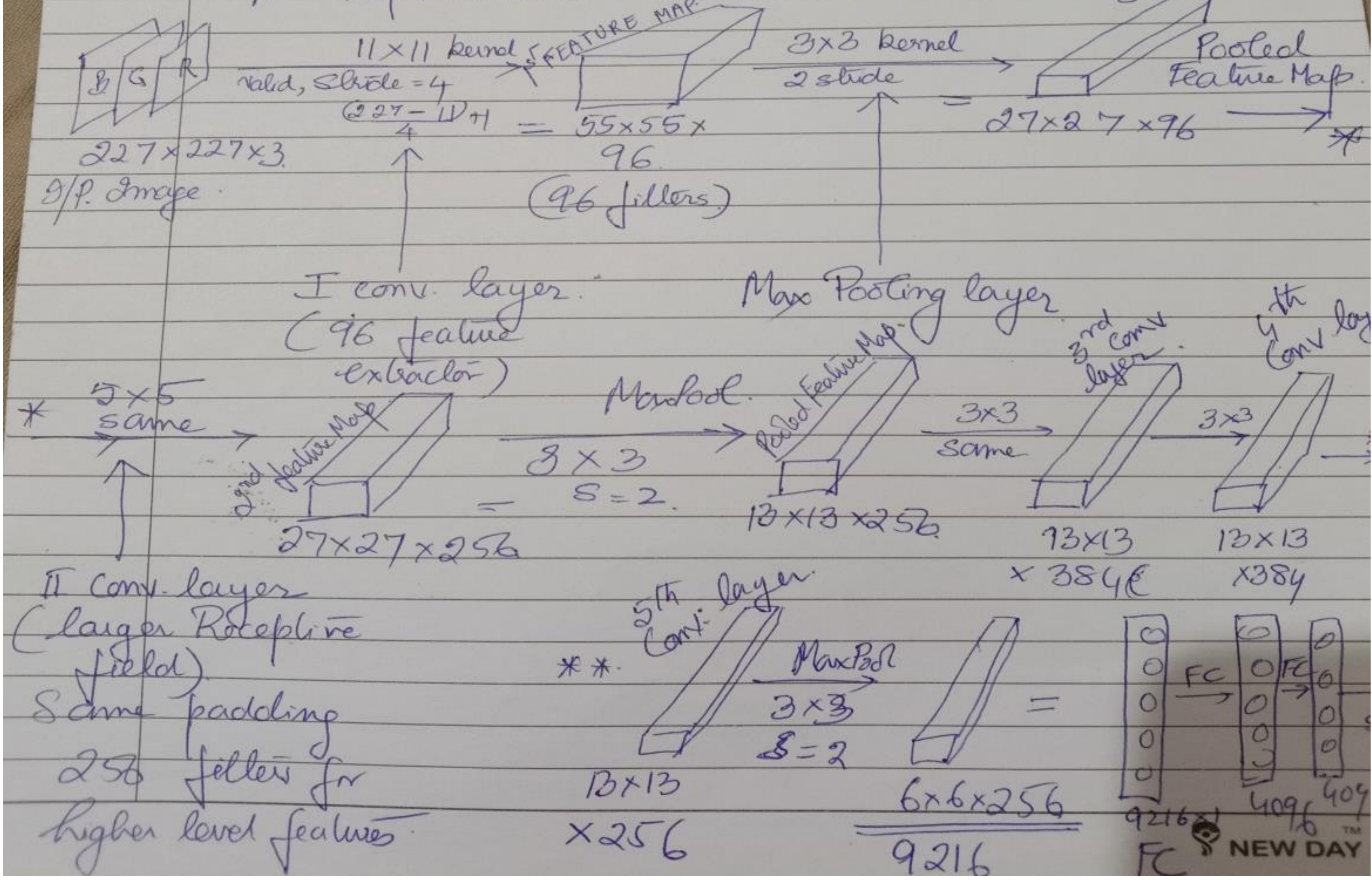
FULLY CONNECTED LAYER:

The final layer of a CNN is fully connected MLP. It works on a greatly filtered and

NEW DAY

reduced form of the original image due to a series of convolutions & pooling. Therefore is this is the "flat" part of the CNN, where results are classified just like any MLP. There can be one or more FC layers. The activation function used in the FC layer may be ReLU if aggregating image information or Softmax if performing classification.

A representative architecture (Alex Net)



Topic.: CNN

Date : Page No. : 8

Such elaborate CNNs have millions of weight parameters to be trained besides a host of other hyper parameters like kernel width, height, no. of kernels, stride, padding, regularization to contain overfitting, learning rate.

→ Some Regularization methods :-

Too many weights, all trained thoroughly to some training data, becomes "rigid" → meaning they cannot generalize to even slightly different test data thus giving low accuracy. To prevent this; typically following are done :-

1) Early stopping :- Training and validation are done simultaneously using different parts of the available data set. Initially both improve accuracy of prediction/classification. After a while, training accuracy improves but validation accuracy drops. That when further training is stopped. Next, testing follows, with much reduced overfitting.

2) Dropout :- Train the network with reduced nodes. Each node is either kept with prob. p or kept out with prob. (1-p). Input nodes are typically kept in so that input info is not lost. The FC output layer has most of the weight, so they are dropped out with higher prob. Intermediate nodes are kept with ~0.5 prob. Thus the network is trained on fewer nodes during each training iteration, with "out" nodes re-inserted before next iteration. This relaxes overfitting.

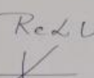
During testing, each node's output is simply weighted with prob p_{node} (full network is tested). The network gives more robust results, correctly predicting / classifying varied test inputs.

3) Data perturbations :- To make the training data more varied, it is perturbed by artificial & deliberate changes such as cropping, erasing, rotating objects etc, while retaining same label as the original.

NEW DAY™

9.

Different variants for ReLU.

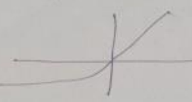
1) Linear $\alpha(z, m) = z \times m$  $\alpha'(z, m) = m$

~~2) ELU~~

+ve :- large ~~range~~ range of activations, not just binary.

-ve :- slope is constant and not dep. on change in X.

2) ELU :- Exponential linear Unit



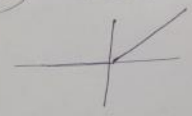
$$ELU(z) = \begin{cases} z & z > 0 \\ \alpha(e^{z} - 1) & z \leq 0 \end{cases}$$

$$ELU'(z) = \begin{cases} 1 & z > 0 \\ \alpha e^{z} & z < 0 \end{cases}$$

+ve :- Unlike ReLU, can produce -ve outputs.

-ve :- ~~Exploding~~ Exploding activation for +ve z

3) ReLU $R(x) = x \quad x > 0$; $= 0 \quad x \leq 0$. $R'(x) = 1 \quad x > 0$; $0 \quad x \leq 0$.

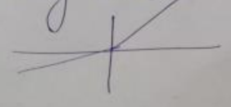


+ve :- avoids vanishing gradient prob

-ve :- can be used only in hidden layers

dead neurons for -ve region.

4) Leaky ReLU.



$$Rz = \begin{cases} z & z > 0 \\ \alpha z & z < 0 \end{cases}$$

+ve :- fixed dying ReLU

-ve :- only for hidden layers. Not for multi class classification.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$ K: no. of classes. (10)

Softmax activation

→ A vector of outputs is converted into a vector of probabilities in a well-calibrated manner.

→ Outputs are normalized so that their sum is zero.

→ Each prob. is membership degree in that output class.

$$\text{n output classes} \begin{Bmatrix} \text{Raw values} \\ O_1 \\ O_2 \\ O_3 \\ \vdots \\ O_n \end{Bmatrix} \rightarrow \begin{Bmatrix} \text{prob values} \\ e^{O_1} / \sum_i e^{O_i} \\ \\ \\ \\ \vdots \end{Bmatrix}$$

→ Softmax is a soft version of argmax

→ It is used because ~~ReLU~~ Linear & Sigmoid are inappropriate for multi class classification.

(i) Linear (output = input) → no activation generates any value.

(ii) Sigmoid is OK for binary classification {binomial distribution}, as well as for non-mutually exclusive multiclass classification. But not for mutually exclusive multi-class classification (multinomial prob. distribution).