UNIT III

Topic :

Objective :

Outcomes :

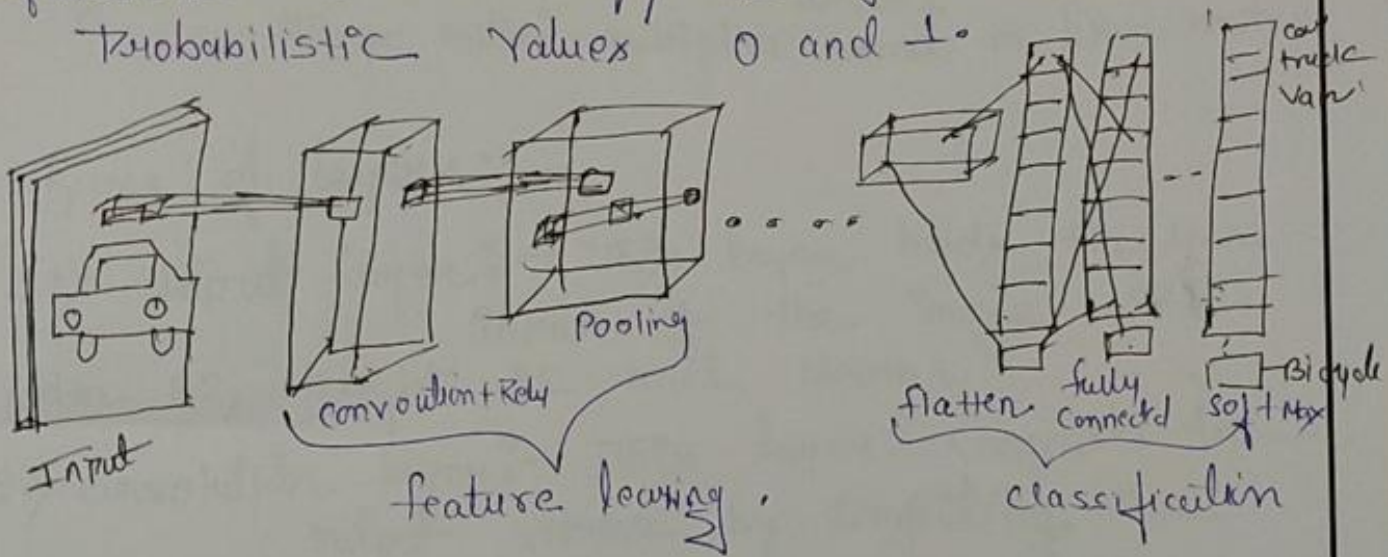
Convolutional Neural Network :—

Convolutional Neural Network is one of the main categories to do image classification and image recognition in neural networks. Scence labeling, object detections and face recognition etc are some of the area where convolutional neural network are widely used.

CNN takes an image as input, which is classified and process under a certain category such as dog, cat, lion, tiger etc. The computer sees an image as an array of pixels and depends on the resolution of the image. Based on image resolution it will see as h * w * d, where h = height w = width and d = dimension.

for example, An RGB image is 6 * 5 * 3 array of matrix, and the grayscale image is 4 * 4 * 1 array of matrix.

In CNN, each input image will pass through a sequence of Convolution layers along with pooling, fully connected layers, filters. After that, we will apply the soft-max function to classify an object with probabilistic values 0 and 1.



Convolution Layer :-

Convolution layer is the first layer to extract features from an input image. By learning image features using a small square of input data, the convolution layer preserves the relationship between pixels. It is mathematical operation which takes two input such as image matrix and a kernel or filter.

- The dimension of the image matrix is $h * w * d$.
- The dimension of the filter is $f_h \times f_w \times d$.
- The dimension of the output is $(h - f_h + 1) \times (w - f_w + 1) \times 1$

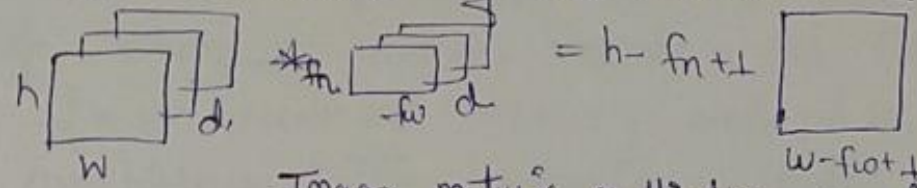


Image matrix multiplies kernel or filter matrix

Types of layers:—

1.) Input layer:— This layer holds the raw input of the image with width 32, height 32 and depth 3.

2.) Convolution layer:— This layer computes the output volume by computing the dot product between all filters and image patches. Suppose we use a total of 12 filters for this layer we will get output volume of dimension $32 \times 32 \times 12$

3.) Activation function layer:— This layer will apply an element-wise activation function to the output of the convolution layer. Some common activation function are RELU: $\max(0, x)$, Sigmoid: $1/1+e^{-x}$. Tan, leaky RELU etc.

4) Pool layer :- This layer is periodically inserted in the convnets and its main function is to reduce the size of volume which makes the computation fast reduces memory and also prevents overfitting. Two common types of pooling layers are max pooling and average pooling. If we use a max pool with 2x2 filters and stride 2, the resultant volume will be of dimension 16x16x12.

Summary :

Topic :

Objective :

Outcomes :

Flattening :-

This step is pretty simple, hence the shockingly (CNN step 8).

After finishing the previous two steps, we're supposed to have a pooled feature map by now. As the name of this step implies, we are literally going to flatten our pooled feature map into a column like the image below:-

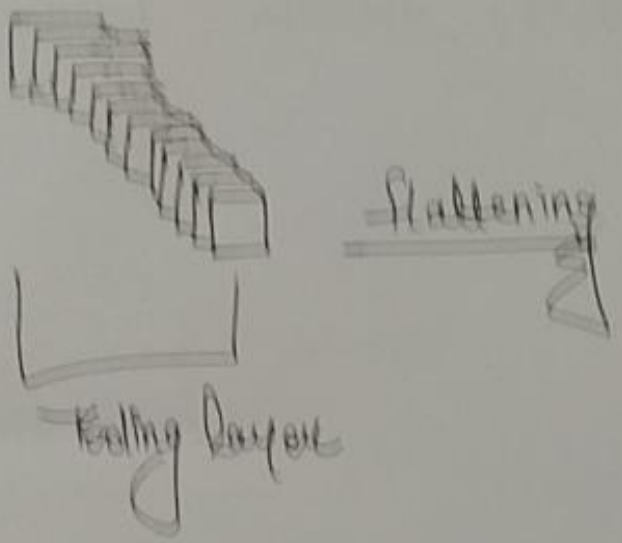
1	1	0
4	2	1
0	2	1

Pooled feature Map

f

1
1
0
4
2
1
0
2
1

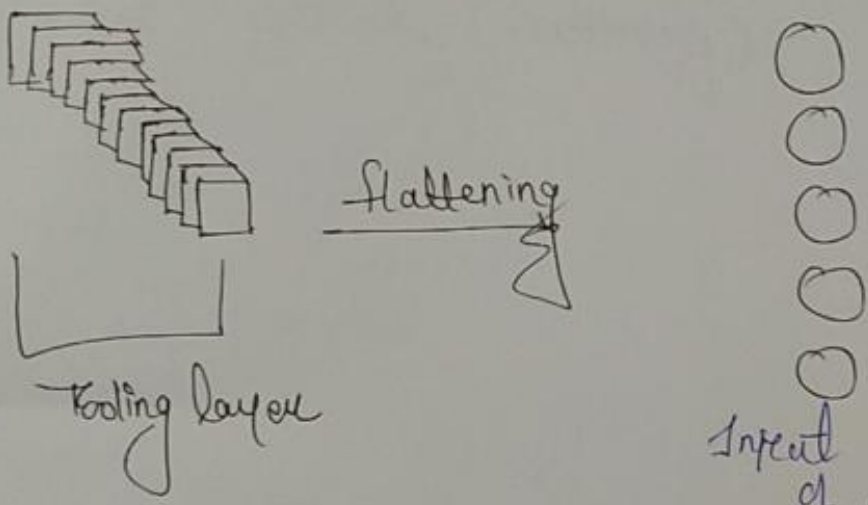
the reason we do this is that we're going to need to insert this data into an artificial neural network later on.



As you see in the image above, we have multiple pooled feature maps from the convolution step.

What happen after the flattening step is that you end up with a long vector of input data that you then pass through the artificial neural network to train it further.

The reason we do this is that we're going to need to insert this data into an artificial neural network later on.



As you see in the image above, we have multiple pooled feature maps from the previous step.

What happen after the flattening step is that you end up with a long vector of input data that you then pass through the artificial neural network to have it processed further.

- Input Image (Starting Point)
- Convolutional layer (convolution Operation)
- Pooling layer (pooling).
- Input layer for the artificial neural network (flattening).

Summary :

Sushila Devi Bansal Collage of Engineering

Umariya, A.B. Road, Indore

Page No.

Topic :

Objective :

Outcomes :

Padding :— Padding is a term relevant to convolutional neural network as it refers to the amount of pixels added to an image when it is being processed by the kernel of a CNN. for example, if the Padding in a CNN is set to zero, then every pixel value that is added will be of value zero. If, however the zero Padding is set to one, there will be one pixel border added to the image with a pixel value of zero.

filter:

1	0
0	0.5

Stride x

Padding = same

I/P

0	0	0	0	0	0
0	1	0	0.5	0.5	0
0	0	0.5	1	0	0
0	0	1	0.5	1	0
0	1	0.5	0.5	1	0
0	0	0	0	0	0

Stride Y

O/P

0.5	0	0.25	0.25
0	1.25	0.5	0.5
0	0.5	0.75	1.5
0.5	0.25	1.25	1

How does Padding work?

Padding works by extending the area of which a convolutional neural network process an image. The kernel is the neural networks filter which moves across the image, scanning each pixel and converting the data into a smaller, or sometimes larger, format. In order to assist the kernel with processing the image, padding is added to the frame of the image to allow for more space for the kernel to cover the image. Adding Padding to an image processed by a CNN allows for more accurate analysis of images.

Summary :

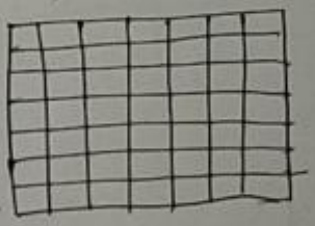
Topic :

Objective :

Outcomes :

Stride & Stride is a component of convolutional neural networks, or neural n/w tuned for the compression of images and video data. Stride is a parameter of the neural n/w filter that modifies the amount of movement over the image or video. for example, if a neural network's stride is set of 1, the filter will move one pixel, or unit - at a time. The size of the filter affects the encoded output volume, so stride is often set to a whole integer, rather than a fraction or decimal.

How does stride work?



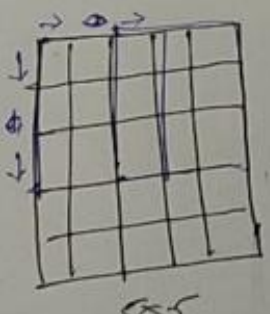
7x7 Input Volume.

5x5 o/p volume.

:) The filter is moved across the image. left to right, top to bottom with a one pixel column change on the horizontal movements, than a one-pixel row change on the vertical movements.

- The amount of movement b/w application of the filter to the I/p image is referred to as the height & width dimensions.

Stride



Stride = 2

12	7
9	14

2×2.

$$O = \frac{(i-k)}{s} + 1$$

$$= \left[\frac{5-3}{2}\right] + 1$$

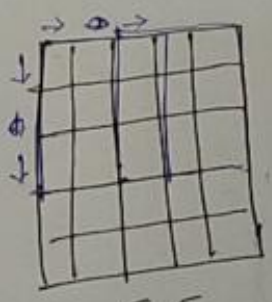
$$= \frac{2}{2} 1 + 1$$

$$\boxed{O = 2}$$

∴ The filter is moved across the image left to right, top to bottom with a one pixel column change on the horizontal movements, than a one-pixel row change on the vertical movements.

- The amount of movement b/w application of the filter to the I/p image is referred to as the height & width dimensions.

Stride — Stride = 2



12	7
9	14

2×2.

$$O = \frac{(i-k)}{s} + 1$$

$$= \left[\frac{5-3}{2}\right] + 1$$

$$= \frac{2}{2} 1 + 1$$

$$\boxed{O = 2}$$

- stride is the number of Pixels shifts over the input matrix. When the stride is 1 then we move the filters to 1 Pixel at a time. When the stride is 3 then we move the filters to 3 Pixels at a time so on.

Summary :

Topic : Pooling, Flattening, loss layer.

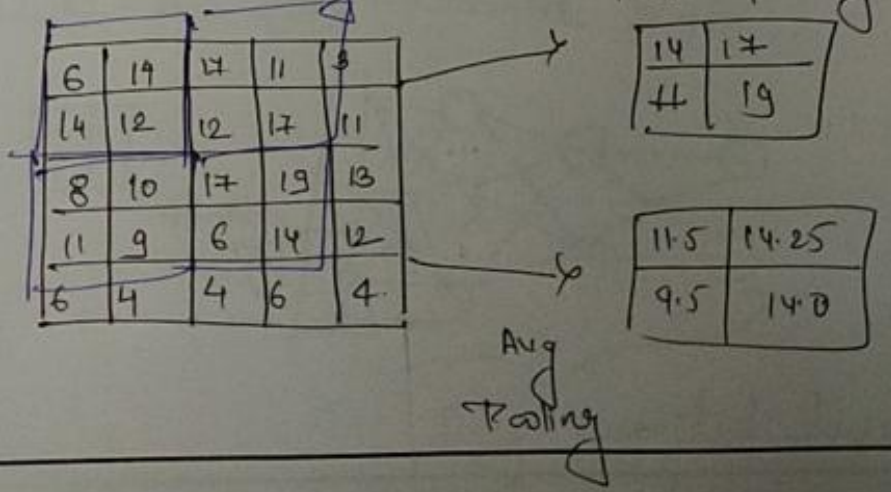
Objective :

Outcomes :

Pooling :— Pooling is the process of merging. So it's basically for the purpose of reducing the size of the data.

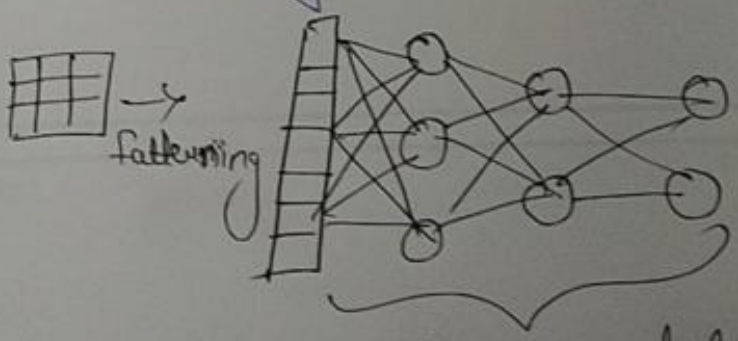
- Pooling is required to down sample the detection of features in features map.
- Pooling layers provide an approach to down sampling feature maps by summarizing the presence of feature in patches of the feature map.

- Max pooling
- Avg Pooling



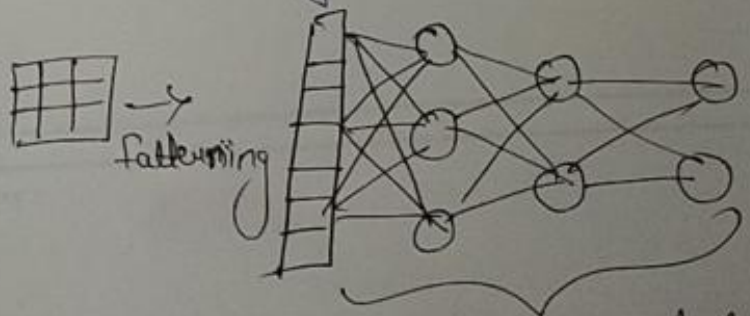
Flattening :-

- Once the Pooled featured map is obtained, the next step is to flatten it.
- It involves transforming the entire pooled feature map matrix into a single column which is then fed to the neural n/w for processing.
- Flattening is converting the data into a 1-dimensional array for inputing it to the next layer. We flatten the o/p of the convolutional layers to create a single long feature vector. And it is connected to the final classification model, which is called a fully-connected layer.



Flattening :-

- Once the Pooled featured map is obtained, the next step is to flatten it.
- It involves transforming the entire pooled feature map matrix into a single column which is then fed to the neural n/w for processing.
- Flattening is converting the data into a 1-dimensional array for inputting it to the next layer. We flatten the o/p of the convolutional layers to create a single long feature vector. And it is connected to the final classification model, which is called a fully-connected layer.



fully connected layer.

loss layer :— The "loss layer" or "loss function" specifies how training penalizes the deviation b/w the predicted o/p of the network and the true data labels.

Various loss fun can be used, depending on the specific task.

- softmax loss function is used for predicting a single class of K mutually exclusive classes.
- sigmoid cross-entropy loss is used for predicting K independent probability values in.

The function used to evaluate a candidate solution is referred to as the objective function.

Summary :

Topic : 1x1 Convolution, Transfer learning

Objective :

Outcomes :

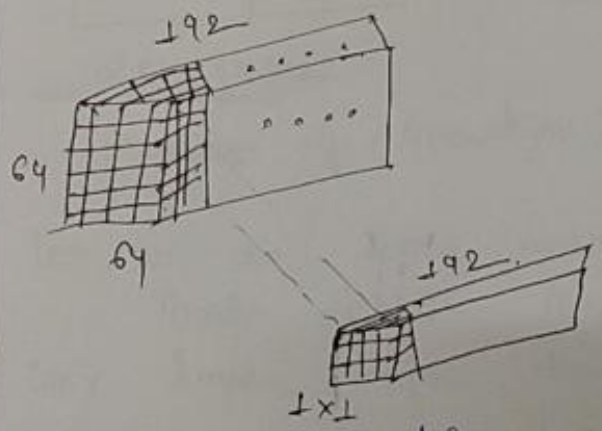
1x1 Convolution :-

Convolutions is an element wise multiplication and summation of the input and kernel/filter elements. Now the data points to remember.

- Input matrix can and in most cases, will have more than one channel. This is sometimes referred to as depth.
  - Example 64x64 pixel RGB input from an image will have 3 channels so the input is 64 x 64x3.
- The filter has the same depth as input except in some special cases.
  - Example : filter of 3x3 will have 3 channels as well, hence the filter should be represented as 3x3x3.
- Third and critical point, the output of Convolution step will have the depth equal to number of filters we choose

• Example: output of Convolution step of the 3D input (64×64×3) and the filter we chose (3×3×3) will have the depth of 1 (Because we have only one filter).

• The Convolution step on the 3D input 64×64×3 with filter size of 3×3×3 will have the filter 'sliding' along the width and height of the input.

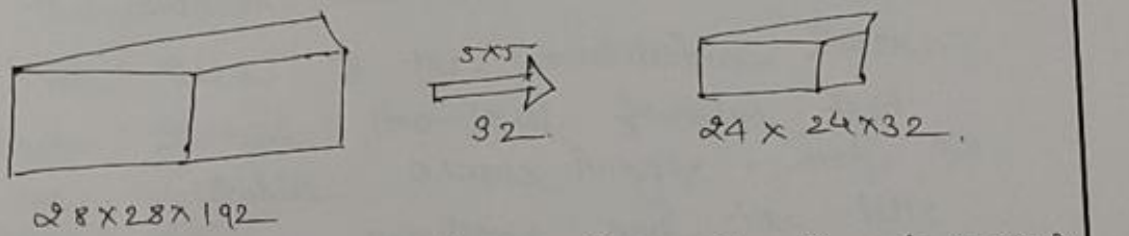


1×1 Conv (Cross Channel Pooling) was used to reduce the number of channels while introducing non-linearity.

• In 1×1 Convolution simply means that filter is of size 1×1 (Yes. that means a single number as opposed to matrix like say 3×3 filter).

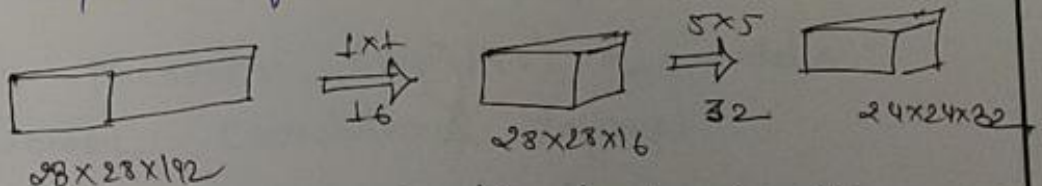
This 1×1 filter will convolve over the ENTIRE input image pixel by pixel.

let us look an example to understand how reducing dimension will reduce Computational load. Suppose we need to convolve 28×28×192 input feature maps with 5×5×32 filters. This will result in 120.422.



Number of Operations: (28×28×32) × (5×5×192)
⇒ 120.422 Million Ops.

let us do some math with the same input feature maps but with 1×1 Conv layer before the 5×5.



Number of Operation for 1×1 Conv step:
(28×28×16) × 1×1×192) = 2.4 Million

Number of Operation of 5×5 Conv. step:
(28×28×32) × (5×5×16) = 10 Million

Total Number of Operation = 12.4 Million Ops.

Transfer learning :- The task of Convolutional neural nlw (CNN) is to identify objects in images.

Transfer learning: take a model trained on a large dataset and transfer its knowledge to a smaller dataset.

The idea is the Convolutional layers extract general, low-level features that are applicable across images - such as edges, patterns, gradients and the later layers identify specific features within an image such as eyes or wheels.

- load in a pre-trained CNN model trained on a large dataset.
- freeze parameters (weights) in model's lower Convolutional layers.
- Add custom classifier layers on training data available for task.

- fine-tune hyperparameters and unfreeze more layers as needed.

Summary :

Topic :

Objective :

Outcomes :

Inception N/w :-

Let's suppose, we want to build a more complex deep neural n/w, what are the challenges we face when adding complexity?

When building a deep neural n/w we are faced two main challenges while adding a new layer:

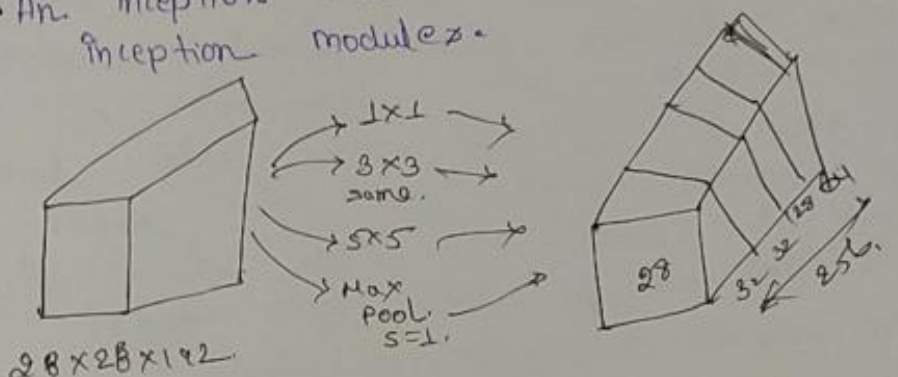
- what should be the filter size should we choose - 3×3, 5×5 or 1×3 or something else?
- should we choose a convolutional layer or a pooling layer?
- How I should proceed?
- Do I need to build multiple n/w testing every time with a different layer or may be a different kernel size?

- Is there a way to do them all and let the n/w decide what is more relevant for the problem I am solving?

* Inception n/w are the ingenious answer to all these questions

- Inception comes with a more complicated n/w architecture but it works remarkable well.
- An inception n/w says that instead of choosing what filter size we want in the Conv layer or what kind of layer we need, let's do them all.
- To have a better understanding of how an inception n/w works, we need first to understand an inception module or sometimes called inception block.

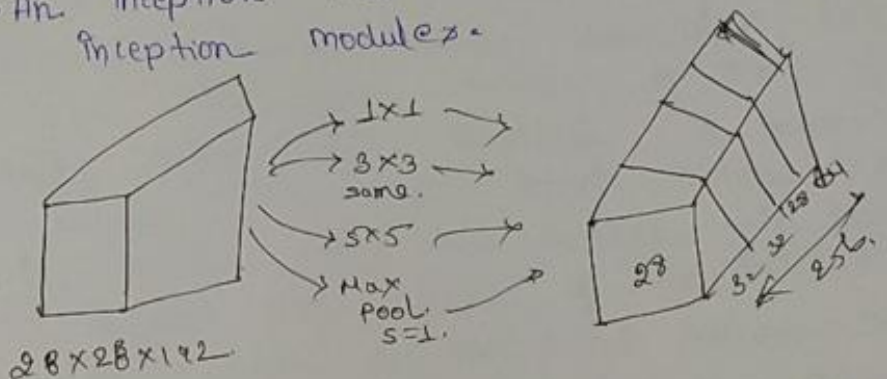
• An Inception nlw is comprosed of multiple Inception modules.



• let's imagine that we have a 28×28×192 input. Applying a 1×1×64 convolutional will result in an output volume of 28×28×64 Calculated following the rule (n+2p-f/s)+1 where, n=28, f=1, p=0 and s=1.

• So now we have an output volume, but also we may be want to try a 3×3×128 Convolution. we will have a 28×28×128 output volume and then all we need to do is to stack both volumes together.

• An Inception nlw is composed of multiple Inception modules.



• let's imagine that we have a 28×28×192 input. Applying a 1×1×64 convolutional will result in an output volume of 28×28×64 Calculated following the rule (n+2p-f/s)+1 where, n=28, f=1, p=0 and s=1.

• So now we have an output volume, but also we may be want to try a 3×3×128 Convolution. we will have a 28×28×128 output volume and then all we need to do is to stack both volumes together.

We can keep trying different filters size or even layers like seen in the picture above where we also tried to Convolve with a 5×5 filter and also we applied a max-pool layer to the input volume where in each step are needed to keep the height and width of the output volume same as the input volume. So, after stacking different o/p, the Inception module will have an output of 28×28×256 (256 = 64+128+32+32), and this is the heart of an Inception N/w.

Summary :