# L-1. Convolutional Neural Network (CNN, ConvNets)

→ MLP

→ Convnets → Visual Tasks.

→ MNIST

→ face?
→ object?
recoguition

Computer Vision → ML/AI, Images/Videos

1981 → Visual Cortex research.

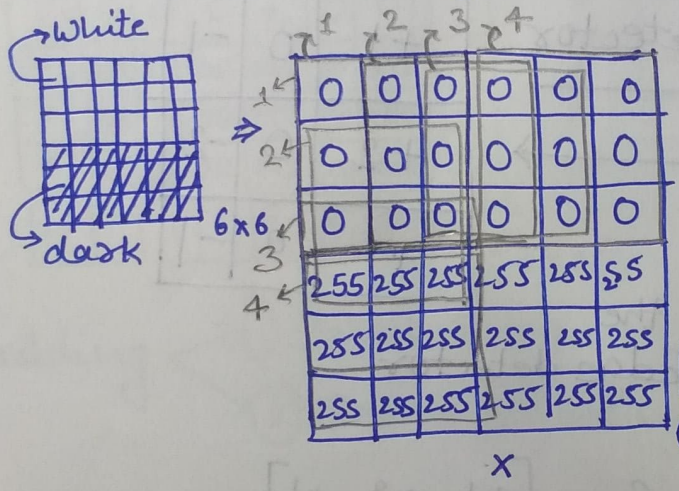
↳ finding

① some neuron fire when light fall at specific angle.

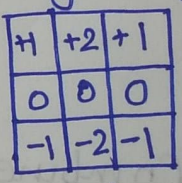
② some neuron detect motion, face, depth etc.

# L-2. Convolution: Edge detection

↳ $V_1$ - primary visual Cortex detects edge in brain




*




[Convolution operation]

y

6x6

Sobel Horizontal edge detector

⇒

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| -1020 | -1020 | -1020 | -1020 |
| -1020 | -1020 | -1020 | -1020 |
| 0 | 0 | 0 | 0 |

4x4

Componentwise multiplication followed by addition.

$$\Rightarrow \sum_{j=0}^{2} \sum_{i=0}^{2} x_{ij} \times y_{ij}$$

why this is a edge detector?

0 → min to 255 → max

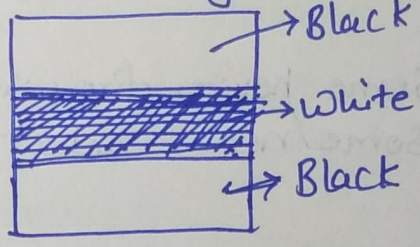
| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| -1020 | -1020 | -1020 | -1020 |
| -1020 | -1020 | -1020 | -1020 |
| 0 | 0 | 0 | 0 |

4 x 4

Norm
min & max →

| | | | |
|---|---|---|---|
| 255 | 255 | 255 | 255 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 255 | 255 | 255 | 255 |

visualizing this matrix as image




Similarly we can extend this Concept to vertical edge detection.

Sobel Vertical Edge detector →

| | | |
|---|---|---|
| +1 | 0 | -1 |
| +2 | 0 | -2 |
| +1 | 0 | -1 |

How to Combine both the Horizontal & vertical Edge detector

$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * A \qquad G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A$$

$$\boxed{G = \sqrt{G_x^2 + G_y^2}}$$

How to create Sobel Edge detector

$$\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} +1 & 0 & -1 \end{bmatrix}$$

↑ averaging kernel

↑ differentiation kernal

# L3 - Convolution Padding and strides.

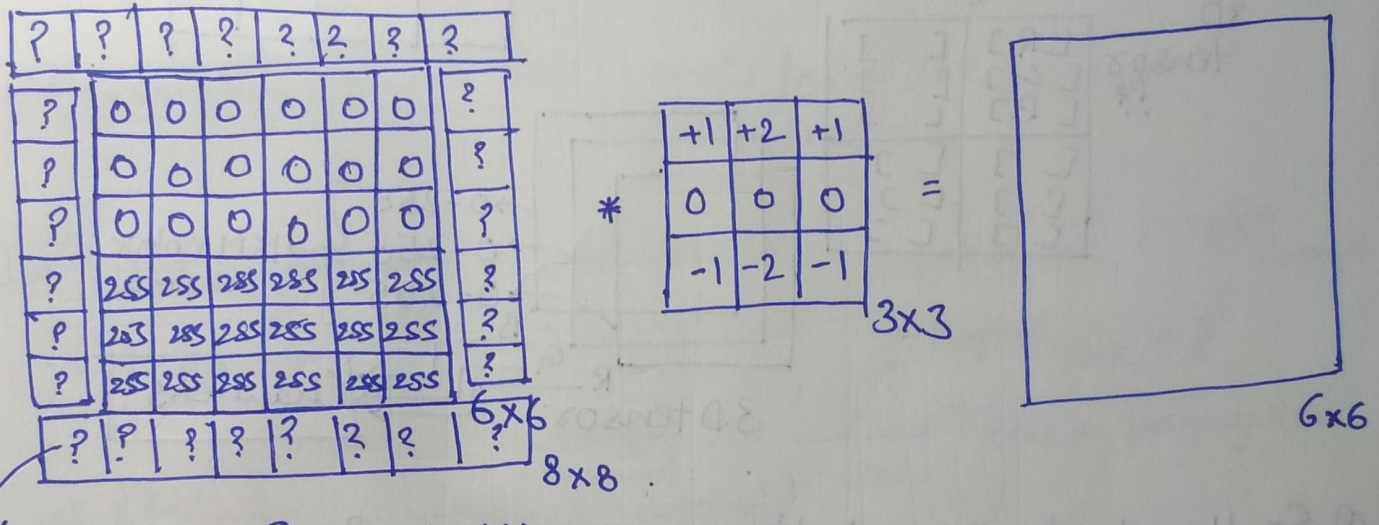
## Padding

input $(n \times n)$ $\xrightarrow[(K \times K)]{\text{Convolution}}$ output $(n-K+1) \times (n-K+1)$

⇒ $n=6$ $K=3 \longrightarrow 4 \times 4$

⇒ $n=8$ $K=5 \longrightarrow 4 \times 4$

Now, we want output I want is also 6x6.




padding → ① Zero padding
② Same adjusting value.

**Stride** → Shift like in Sobel we have shift of 1. Here this shift is called stride we can have any size of stride

$$n \times n \xrightarrow[K \times K]{S=s} \left(\left\lfloor \frac{n-K}{S} \right\rfloor + 1\right) \times \left(\left\lfloor \frac{n-K}{S} \right\rfloor + 1\right)$$

$$6 \times 6 \xrightarrow[K=3]{S=2} \left(\left\lfloor \frac{3}{2} \right\rfloor + 1\right) \times \left(\left\lfloor \frac{3}{2} \right\rfloor + 1\right) = 2 \times 2$$

Input $6 \times 6 \xrightarrow[K=3]{S=2} 2 * 2$ output

$6 \times 6 \xrightarrow[K=3]{S=1} 4 \times 4$. → as seen in Sobel.

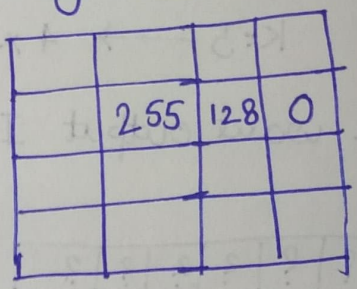
## Conv, padding, strides

$$n \times n \xrightarrow[P,S]{k \times k} \left(\left\lfloor \frac{n+k+2p}{s} \right\rfloor + 1\right) \times \left(\left\lfloor \frac{n+k+2p}{s} \right\rfloor + 1\right)$$

padding, stride

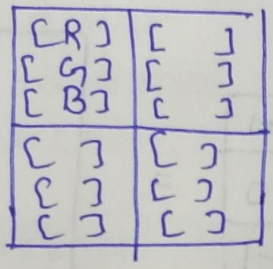
## L-4 - Convolution on Color-images.

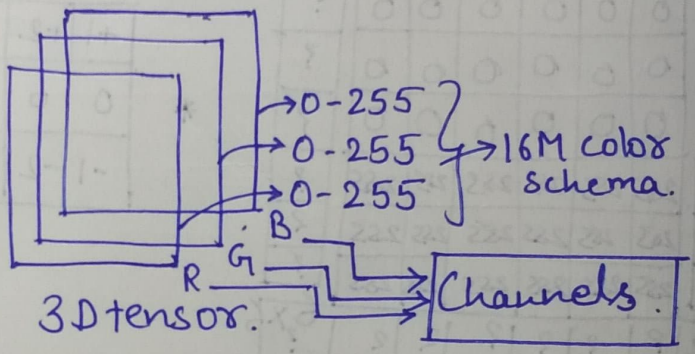
Conv → gray scale images → 2D tensor




→ Color-images

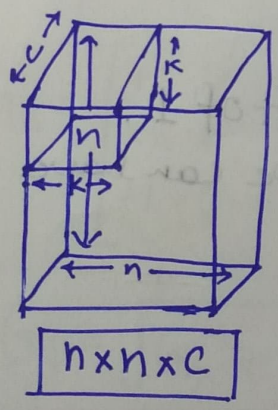
3D tensor ??

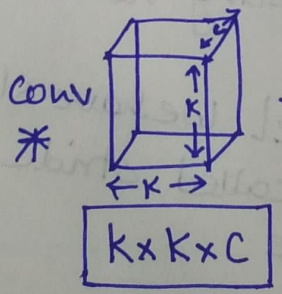


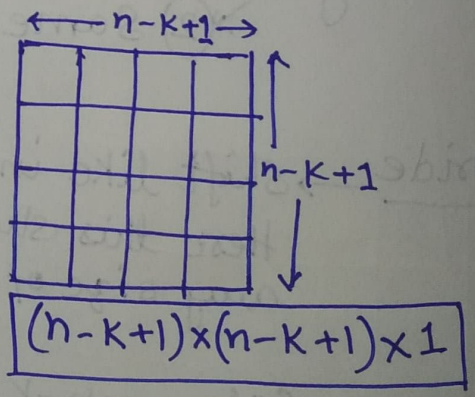




Q. So, How to do Convolution on 3-D tensors?

A.



# L-5 Convolution Layer in NN

MLP layer 

① Conv layer → biological inspiration $V_1, V_2, V_3$
↳ Kernals.

② Multiple edge detection ⟶ Multiple kernels } Seens in prev.
1 edge detection ⟶ 1 kernal

↳ Image processing classical : sobel.

Vs

Convolution Neural Nets (CNN) : learn the kernel matrix

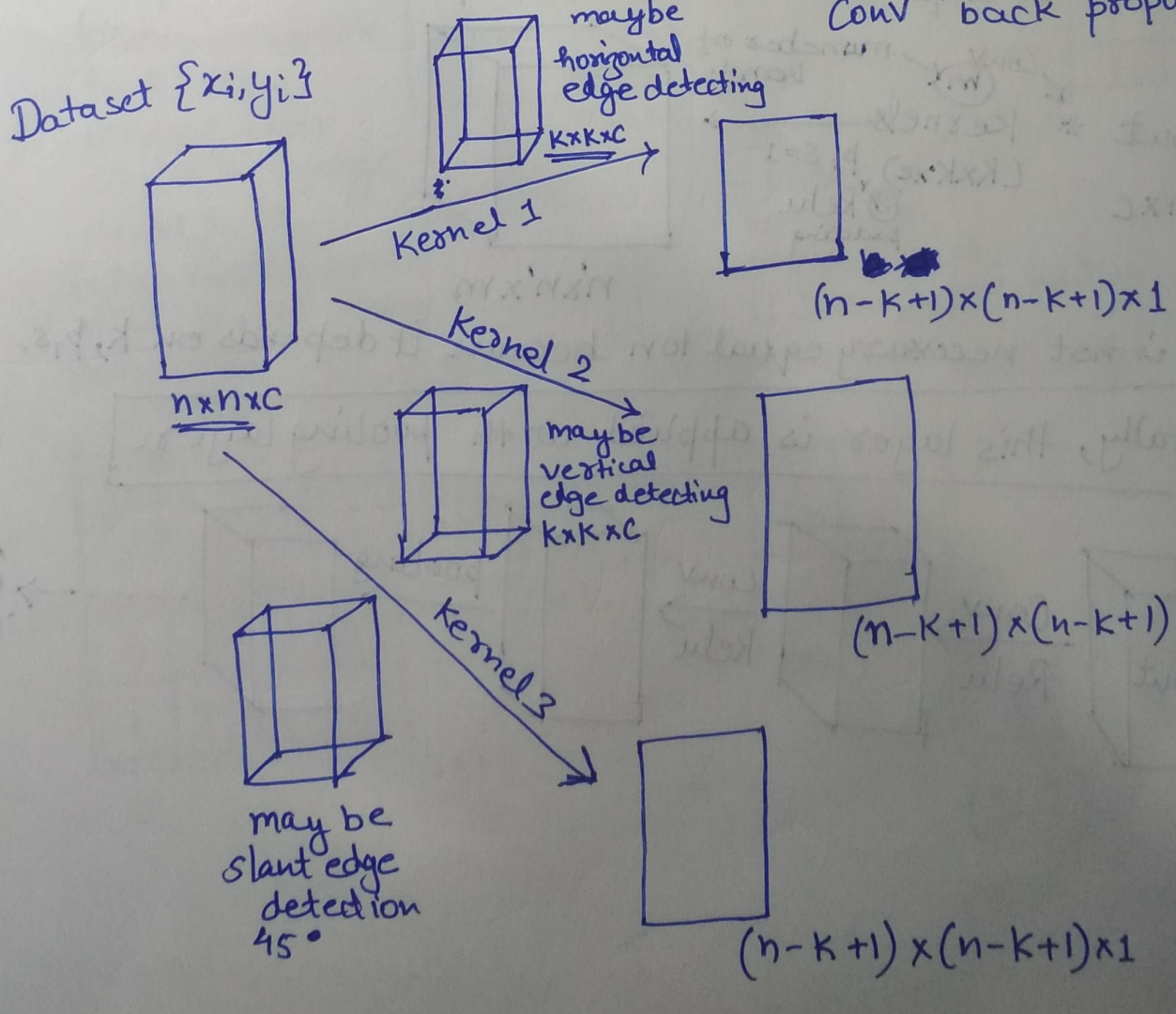
In MLP : learns weights → $W.x$

CNN : learn kernel matrix → $I * \text{kernel}_{K\times K} = E$

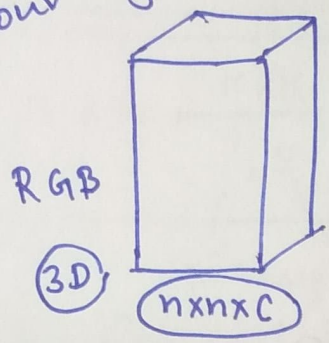
($I$ : input image, $E$ : edge image)

Conv ← back propogation.

Dataset $\{x_i, y_i\}$



Conv-layer.

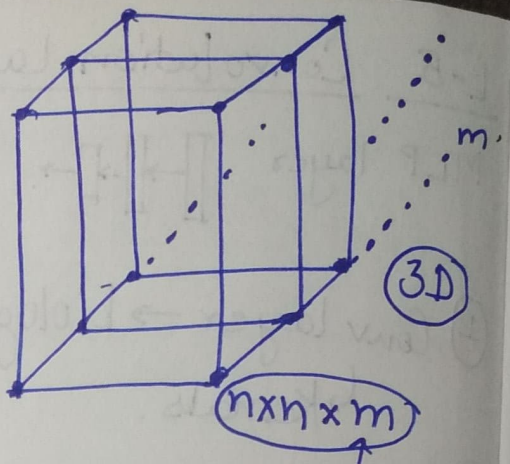



m → hyperparameter

K×K×c

(10) Multiple kernel

padding, Stride=1




Steps.

(1) Input $n \times n \times c$ * Kernel $(K \times K \times 1)$ (Conv) $\xrightarrow{p,s=1}$ $(n \times n \times 1)$

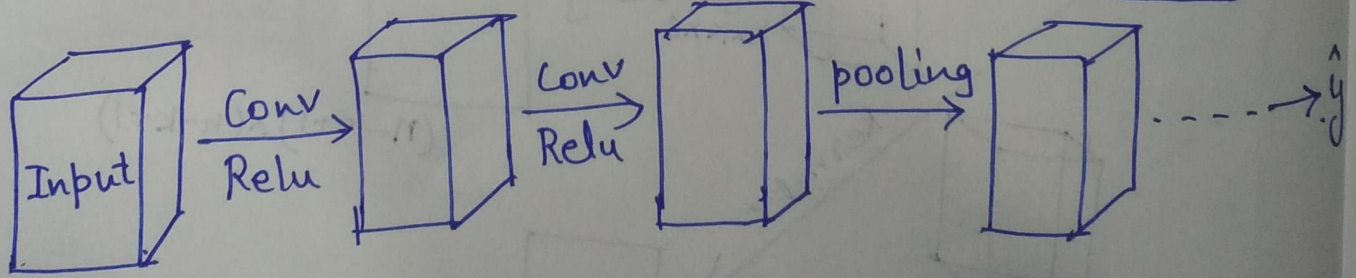
(2) $(n \times n \times 1)$ $\xrightarrow{\text{elementwise Relu}}$ $(n \times n \times 1)$

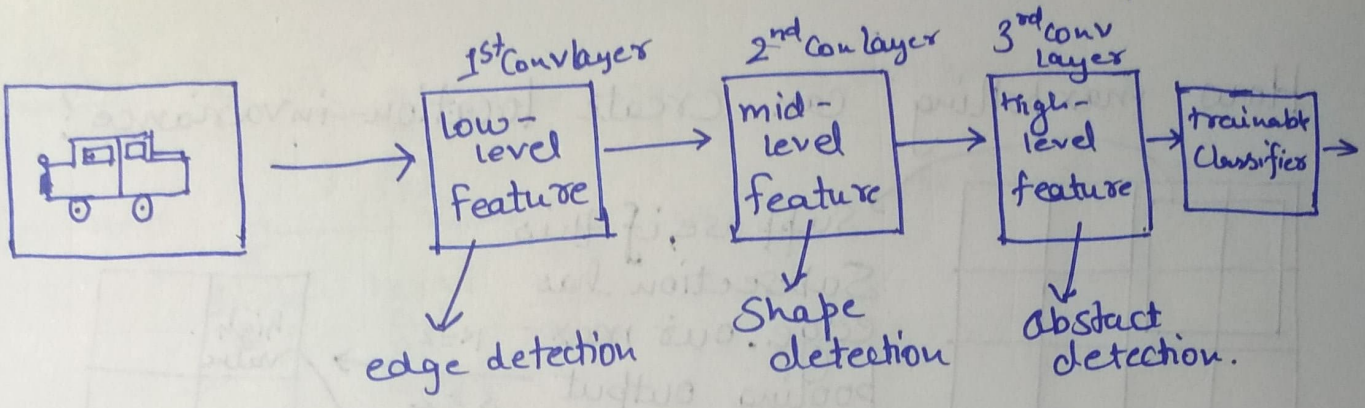
Often this 2 steps are written as 1 step.

Input $n \times n \times c$ * kernels $(K \times K \times c)$ (Conv) (m) number of kernels. $p, s=1$ (3) Relu padding $\rightarrow$ $n' \times n' \times m$

$n'$ is not necessary equal to $n$ because it depends on $K, p, s$.

**Generally, this layer is applied with pooling layer.**






Convnets are much similar to how visual cortex. works.

Key takeaway:- Kernels train in Convnets is same as weights training in MLP.

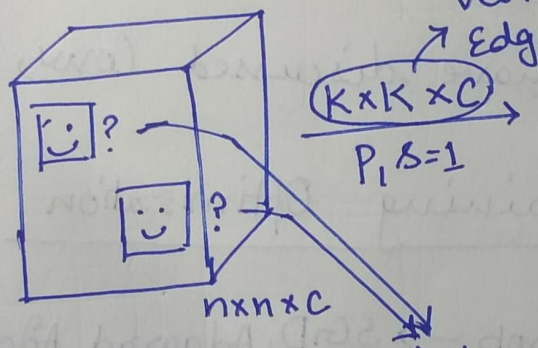
## L-6- Max pooling (pool) → CNN

→ layer

→ inspiration

location invariable in image. where ever object in image detect it.

It has nothing to do with size of image variable.




K×K×C

P, S=1

Say for Kernel is for Vertical Edges

| | | | | |
|---|---|---|---|---|
| 1 | 0 | | | |
| 0 | 0 | | | |
| 1 | 1 | | | |
| | 0 | | | |
| | | | | |
| | | | | |

it does not patter where is face it should fire neron So, we have pooling.

max value in this sub matrix.

| | | | |
|---|---|---|---|
| 1 | 1 | 2 | 4 |
| 5 | 6 | 7 | 8 |
| 3 | 2 | 1 | 0 |
| 1 | 2 | 3 | 4 |

4×4

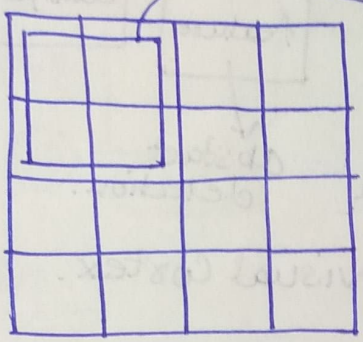
(Max-pooling) K=2, S=2 →

| | |
|---|---|
| 6 | 8 |
| 3 | 4 |

$\left(\left\lfloor\frac{4-2}{2}\right\rfloor+1, \left\lfloor\frac{4-2}{2}\right\rfloor+1\right)$

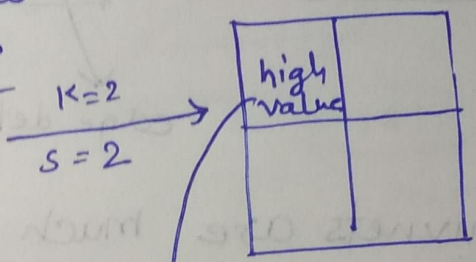
(2,2)

How max polling can create location-invariance?



this is Convoluted + Relued image

Suppose, if this Subsection has edge. our max-pooling output will have high value in region




this high value Signifies the region has edge.

Max-polling is very popular in recent time.

So till now we have discussed Conv, Relu, Max-pooling.

## L-7- CNN Training Optimization

MLP :- back prop → SGD, Adagrad, Adam

↳ differential function $x \xrightarrow{w} \hat{y} \rightarrow L(y, \hat{y})$

$\frac{\partial L}{\partial w}$ or $\nabla_w L$

Now in this also we have to make sure

→ Conv-layer → Conv ; Relu — differential already seen

→ Max-polling

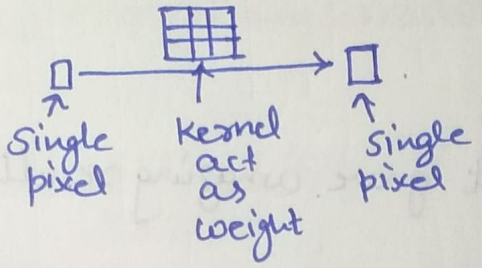
↳ element wise multiplication followed by addition on Matrix. Hence this is differentiable.

↓

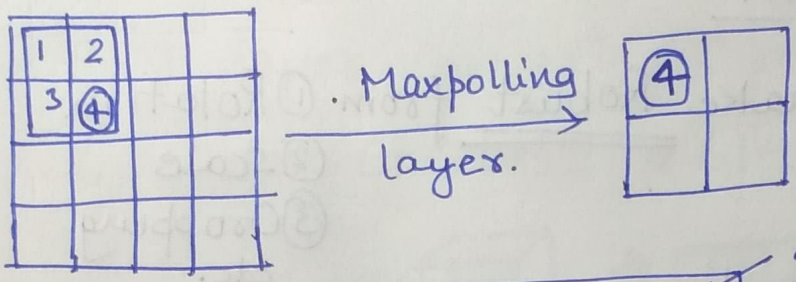
Now how to take derivative of max function & then back propopogate.

if we consider 0. So for max value we give value = 1 & 0 otherwise

MLP. x —w's→ y can be expressed as dot product

Convnets 

can be expressed as dot product.



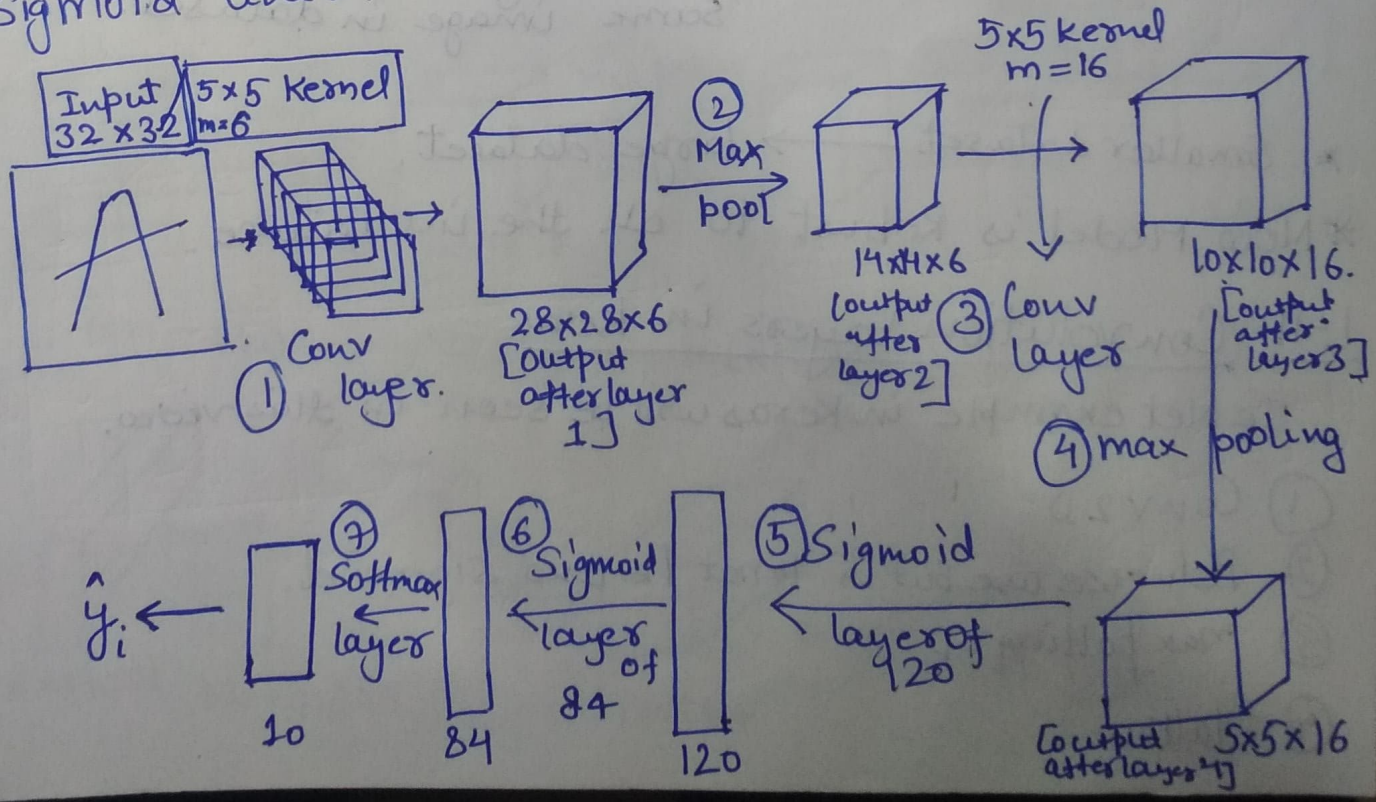

$\frac{\partial MP}{\partial x}$ ⇒ Max value derivative = 1
non-Max value derivative = 0

simple hack for derivative for max function is 0 but this is not called max function this is Max pooling layer.

We can apply dropout wherever required.

## L-8- Example CNN LeNet [1998]

In 1998, Relu's were not invented so they used sigmoid units.



## L-9 · Image Net Dataset → Benchmark dataset for Algorithm.

- Visual Image detection dataset
- 1.4 cr points
- 20 K labels.

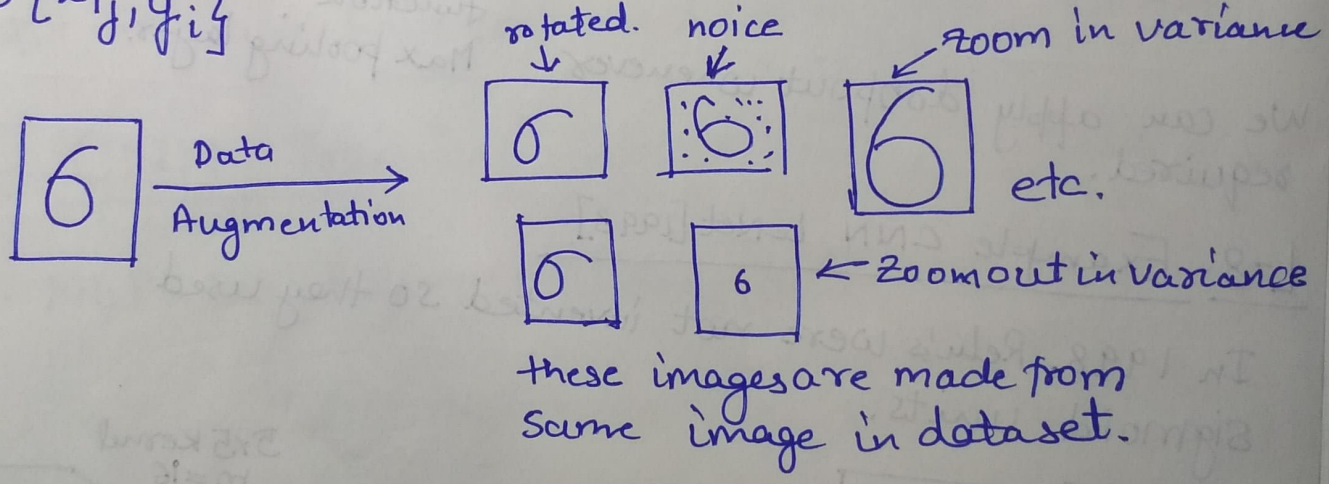
AlexNet (2012) on this dataset gave amazing results.

## L-10. Data Augmentation

CNN models to make Robust from ① Rotation
② Scale
③ Cropping
etc.

$D\{x_i, y_i\}$ → $D'\{x_{ij}, y_i\}$

We create bunch of new images in D' for data in D by little rotation, adding noice, etc.




* Smaller dataset ⟶ large dataset.
* Now Model is Robust to all the invariance.

## L-11 Convolution Layers in Keras

Le Net example in Keras will be seen in this video.

① Conv2D
② Relu → we use but in lenet he used Sigmoid.
③ Maxpolling2D
④ Flatten.

Conv 1D ← time Series Data

Conv 2D ← image

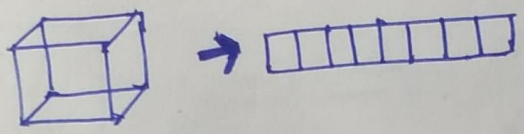
Conv 3D ← image + time series Data.

Keras.layers. & Conv2D ( so many parameter self explainatory).

⇒ Activation layer can be added extra or in this same line only.

flatten layer converts 3D matrix to 1D array

Ex→ 6x 32x32 ⇒ 6144.

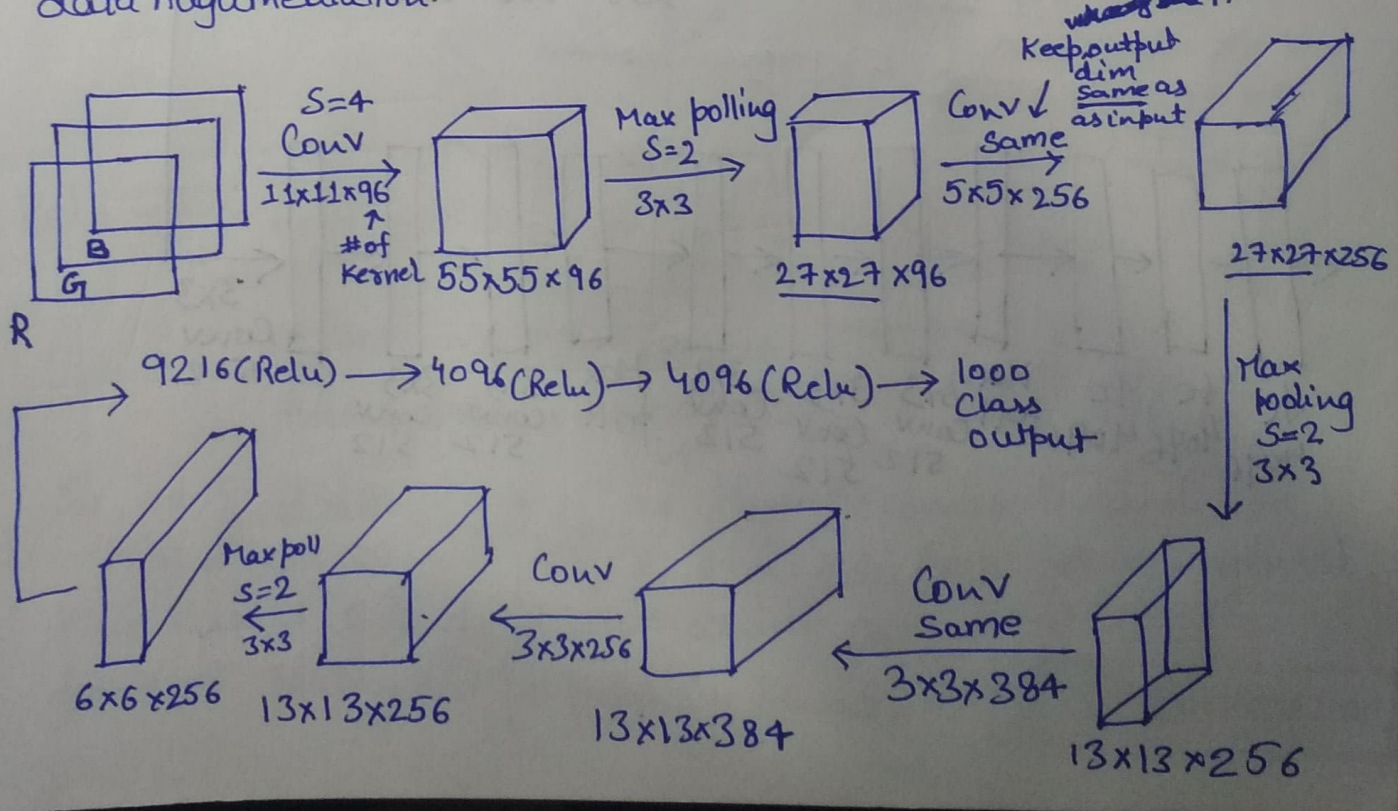


Keras
- → LeNet (1998)
- → AlexNet (2012)
- → VGG (2014)

L12 - Alex Net → Same model of 2012 that made everyone's attention on Deeplearning.

Relu, dropout, data Augumentation.

⇒ GPU's → first used by Andrew Ng.



# L-13 VGGnet

→ Andrew Zisserman

→ Visual Geometry Group

Similifing Alexnet by using only 3×3 kernel padding='same' S=1 → Conv layer.

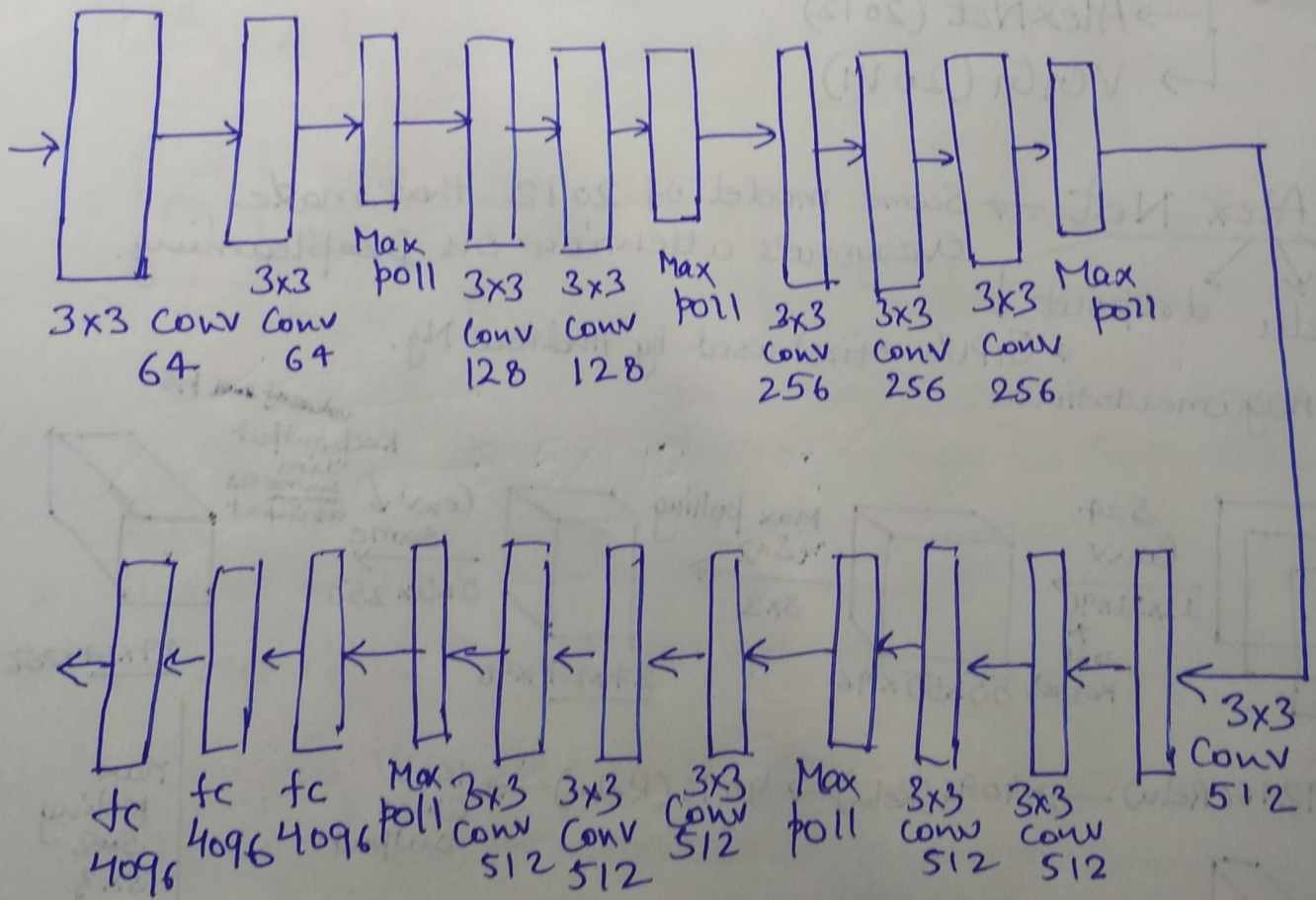
rathan than 3×3, 5×5, 11×11 so many kernels.

Max poll 2×2 S=1

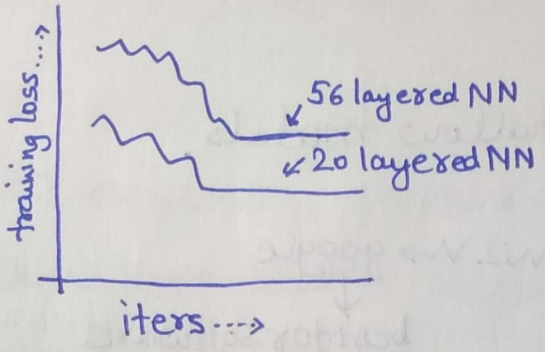
VGG16

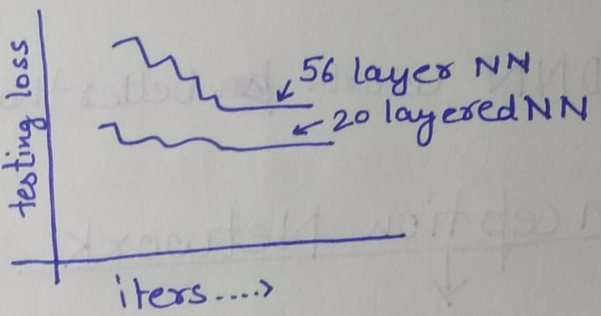
VGG19

## Architecture.

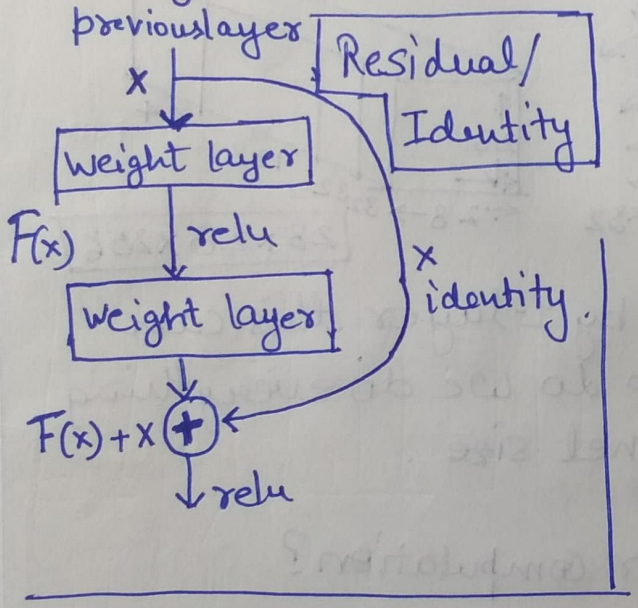


# L-14 Residual Network

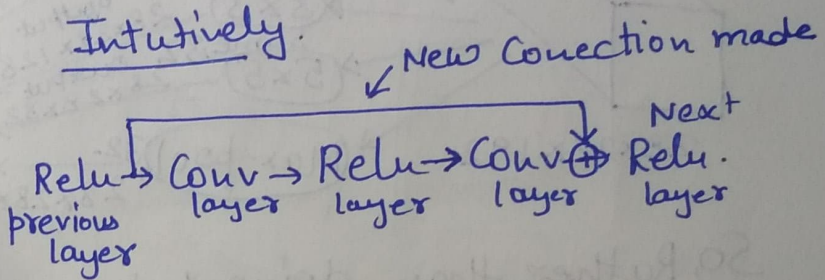



Researchers observed very deep neural network gives high loss than low layer network than very high.
So, they developed new building block.

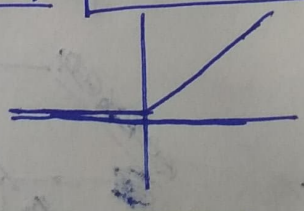



Intutively.

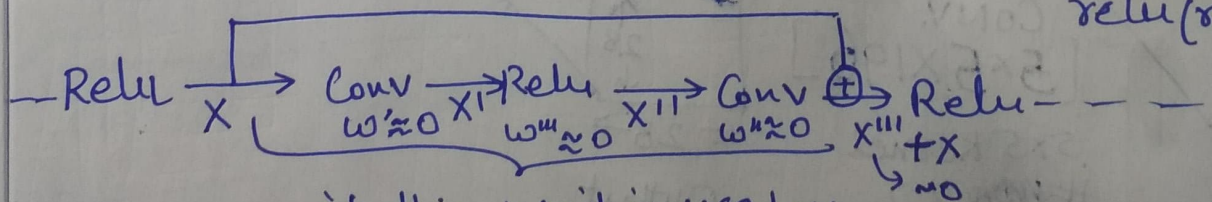



What does regularizer do $L_1, L_2$, dropout it removes effect of useless weights?
and

$$\boxed{relu(relu(x)) = relu(x)}$$



if $x>0$ $relu(x)=x$
$relu(relu(x)=x$
if $x<0$ $relu(x)=0$
$relu(relu(x)=0$




if this unit is useless
this work as a regulisor.

So not accounting $x'''$ as this becomes 0 and only effect caused by X and $relu(relu(x+x''') = relu(x) = x$.
thus this is like dead unit which cannot be further propogated. popularly known as **SKIP CONNECTION**

Make Sure X''' & X are of same size use padding as 'SAME'.

Now, DNN work better than shallow models.

## 15 - Inception Network

↓

Google

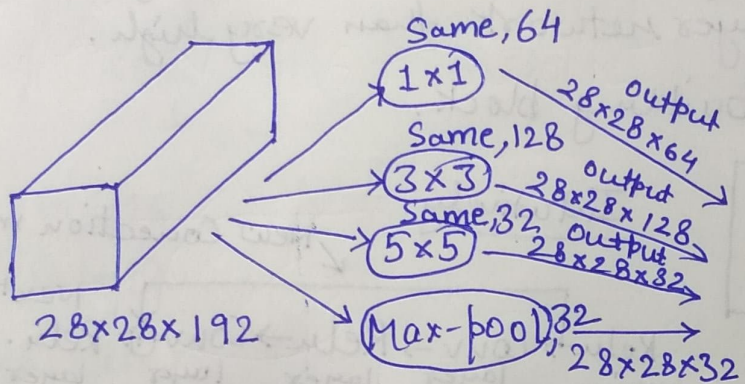
W2V → google

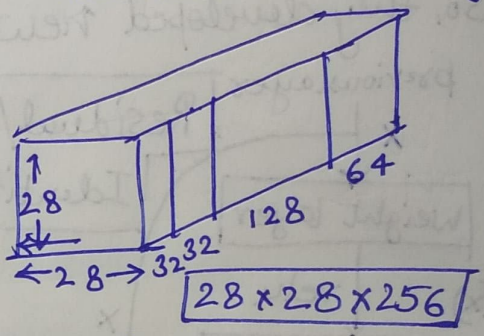
↓

best for software Optimisation

↓

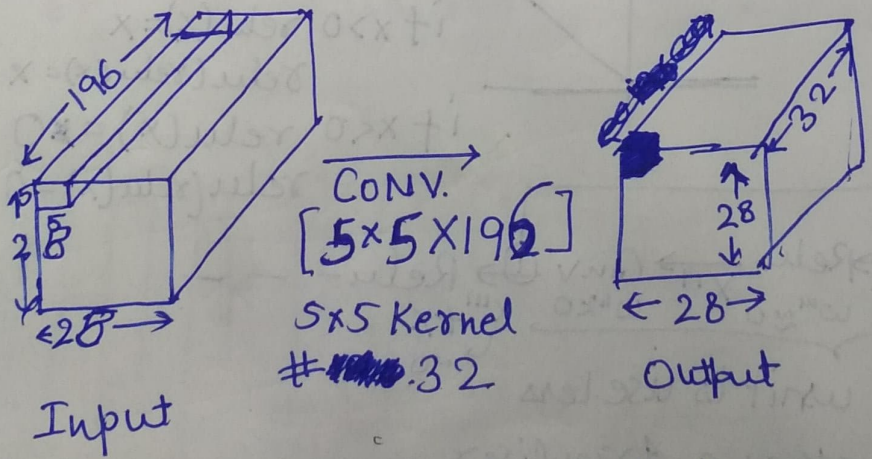
Negative Sampling, Heracical Clustering.




So, Rather than doing 1-layer by 1-layer this can Create Confusing. So what we do do we do everything together with different Kernel size.

Now How Hard this is for Computation?




To generate 1 unit of output = (5×5×196)

To generate Matrix of 28×28×196 = (5×5×196) × (28×28×32)

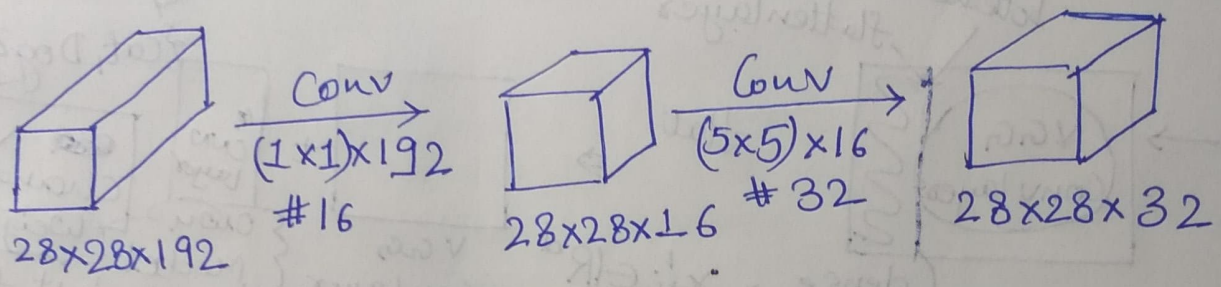
≈ 128 M

So for 1 layer we have do 128 Million multiplication. Here we have not counted for addition also.

So, How can we optimise this.

So engineer poposed use 1×1×K size kernel as a intermediate kernel also called Bottleneck layer.

→ this can be understood as Autoencoder. We first reduced the size of input then expand it.

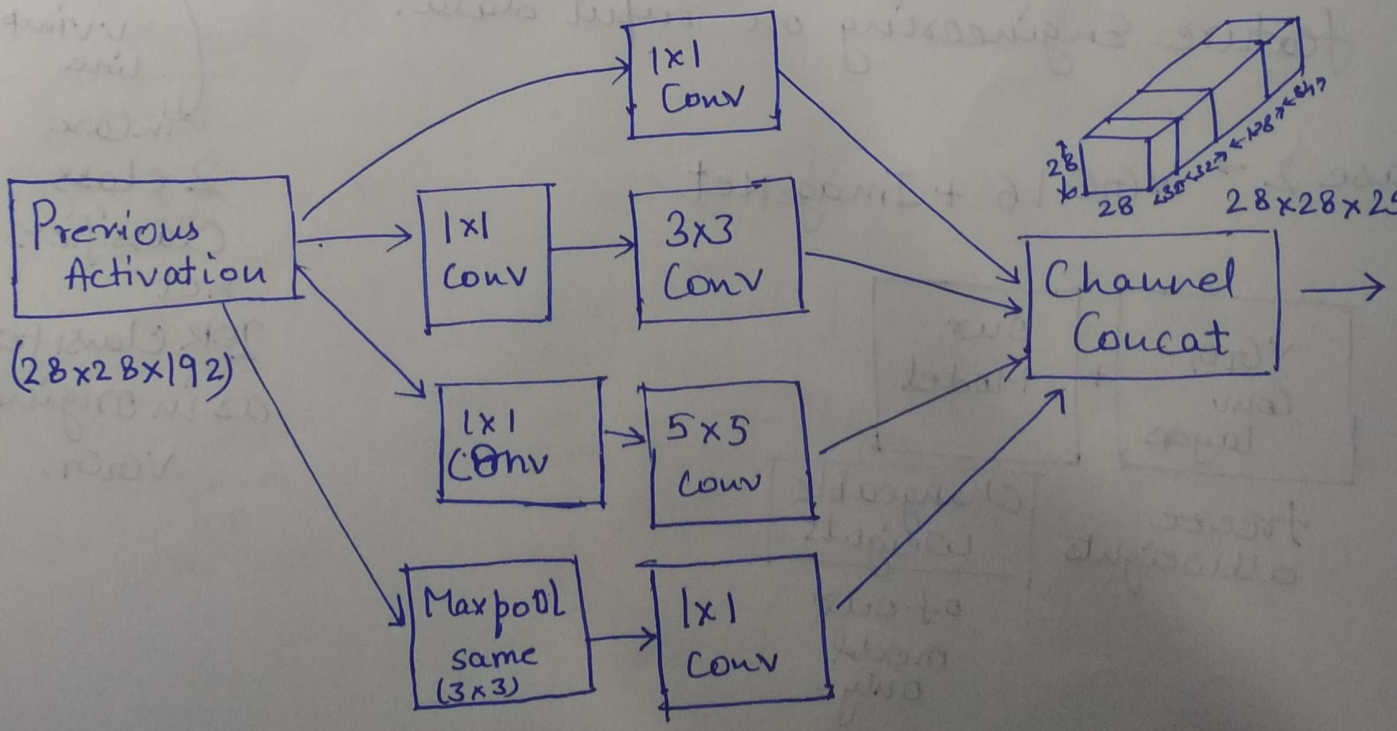



No. of multiplication:- $(28 \times 28 \times 16) \times (1 \times 1 \times 92)$

$+$

$(28 \times 28 \times 32) \times (5 \times 5 \times 16)$

$= 12.4M$ (earlier which was 120M)

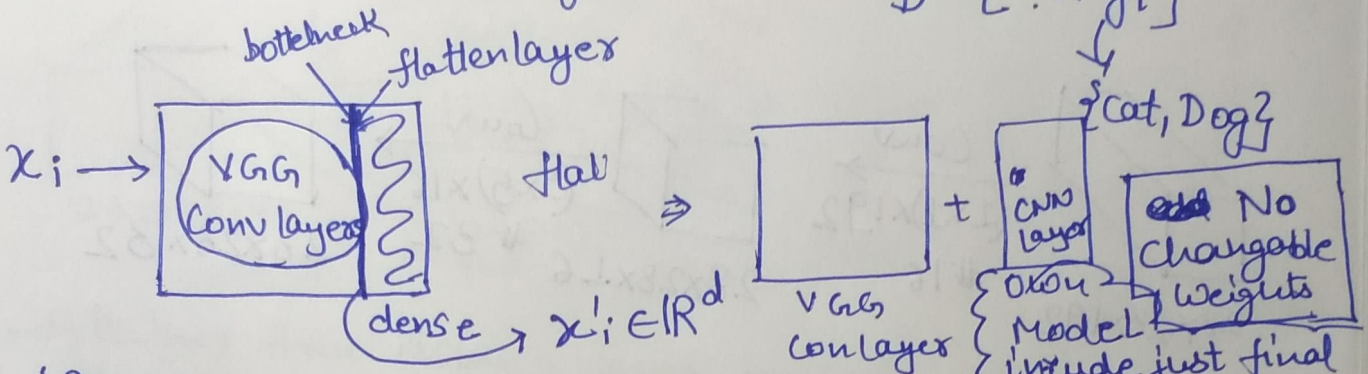


# L-16. Transfer Learning (Keras)

Idea: Instead of building a NN from scratch to solve our task, we can reuse existing models (VGG16) trained on a different dataset

Hence, Can I "transfer" learning from 1 model to other

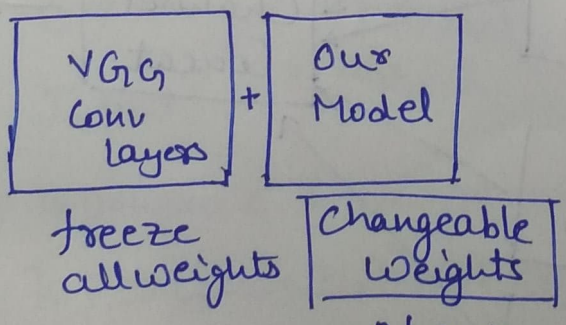
## Case I :- VGG16 + ImageNet




We use all the VGG Conv layer, don't use flatten layer & dense layer. and we can add our model after VGG Conv layer and Now this VGG Conv layers act as our feature Engineering on input data.

Case 1 vvimp line
In case 2 class Classifier. not 20k classifer as in original VGG.

## Case 2 → VGG16 + ImageNet

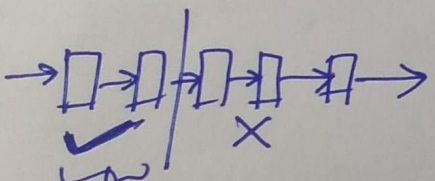


## Case 3: VGG16 + ImageNet ← init weights / same model

{take old model of init weights train model on your dataset on top of it.}
again

D ~ small
D % imagenet

take few layer of VGG which has edge detection & train remaing model.



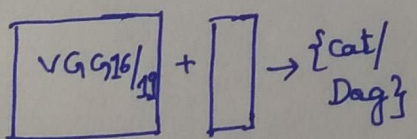
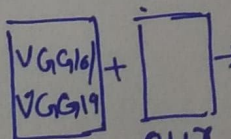
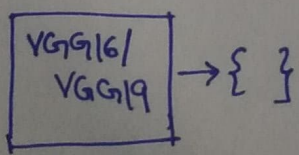
first 2 layers are generally edge detectors.

D ~ large
D % imagenet

take VGG model as initail weight initializer and train the complete model again.

---

Summary.

| Case I | Vs | Case II | Vs. | Case III |
|---|---|---|---|---|
|  | |  | |  |
| VGG16/19 + → {cat/Dog} | | VGG16/VGG19 + → { } | | VGG16/VGG19 → { } |
| No change our CNN Classifier Not 20K classifer | | No change our Complete Model added. with VGG | | Change the weights with these initial weights. |