* Backpropagation in CNN

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} * \begin{bmatrix} f_{11} & f_{12} \\ f_{21} & f_{22} \end{bmatrix}$$

↓ input x

↓ Filter F

we learn about Backpropogation in multilayer perceptron. Now, it is very important to understand this in CNN as well.

ø → we have to understand in details. what is happening. So, you can actually understand the core of it

→ Before we start backpropogation let us understand foreward pass. CNN. (This is basic convolution)

* foreward pass → So, you have input and you convolve it with the filter than you move foreward. once you get the output then you will convolve it again and then you will get another output and so on, you will go on untill you get a final required whatever you are searching for. it might be segmentation, classification or so on.

So, let us suppose this an input values

f_{11}	f_{12}
f_{21}	f_{22}

filter values. then

filter F

we are getting an output. So this is a output feature map

o_{11}	o_{12}
o_{21}	o_{22}

So, what we be doing is we will be understanding

Same as convolution.

* what is Back propagation

Local Gradients $\longrightarrow$ (A)

$$o_{11} = x_{11} f_{11} + x_{12} f_{12} + x_{21} f_{21} + x_{22} f_{22}$$

finding derivatives with respect to f_{11}, f_{12}, f_{21} and f_{22}

$$\frac{do_{11}}{df_{11}} = x_{11} \quad \frac{do_{11}}{df_{12}} = x_{12} \quad \frac{do_{11}}{df_{21}} = x_{21} \quad \frac{\partial o_{11}}{\partial f_{22}} = x_{22}$$

Similarly we can find the local gradients for o_{12}, o_{21} and o_{22}

Page No.:
Date:

→ Here we using chain rule.

$$= \frac{dL}{dF_{11}} = \frac{\partial L}{\partial O_{11}} * \frac{\partial O_{11}}{\partial f_{11}} + \frac{\partial L}{\partial O_{12}} * \frac{\partial O_{12}}{\partial F_{11}} + \frac{\partial L}{\partial O_{21}} \times \frac{\partial O_{21}}{\partial F_{11}}$$

$$+ \frac{\partial L}{\partial O_{22}} * \frac{\partial O_{22}}{\partial f_{11}}$$

$$= \frac{\partial L}{\partial f_{12}} = \frac{\partial L}{\partial O_{11}} * \frac{\partial O_{11}}{\partial f_{12}} + \frac{\partial f}{\partial O_{12}} * \frac{\partial O_{12}}{\partial F_{12}} + \frac{\partial L}{\partial O_{21}} * \frac{\partial O_{21}}{\partial F_{12}} +$$

$$\frac{\partial L}{\partial O_{22}} * \frac{\partial O_{22}}{\partial F_{12}}$$

$$= \frac{\partial L}{\partial f_{21}} = \frac{\partial L}{\partial O_{11}} \times \frac{\partial O_{11}}{\partial f_{21}} + \frac{\partial L}{\partial O_{12}} * \frac{\partial O_{12}}{\partial f_{21}} + \frac{\partial L}{\partial O_{21}} \times \frac{\partial O}{\partial f}$$

$$+ \frac{\partial L}{\partial O_{22}} * \frac{\partial O_{22}}{\partial f_{21}}$$

$$= \frac{dL}{df_{22}} = \frac{\partial L}{\partial O_{11}} * \frac{\partial O_{11}}{\partial f_{22}} + \frac{\partial L}{\partial O_{12}} * \frac{\partial O_{12}}{\partial f_{22}} + \frac{\partial L}{\partial O_{21}} * \frac{\partial}{\partial f}$$

$$+ \frac{\partial L}{\partial O_{22}} * \frac{\partial O_{22}}{\partial F_{22}}$$

M T W T F S S

Page No.:

Date:

$$= \frac{\partial L}{\partial F_{11}} = \frac{\partial L}{\partial O_{11}} * x_{11} + \frac{\partial L}{\partial O_{12}} * x_{12} + \frac{\partial L}{\partial O_{21}} * 21$$

$$+ \frac{\partial L}{\partial O_{22}} \times x_{22}$$

$$= \frac{\partial L}{\partial F_{12}} = \frac{\partial L}{\partial O_{11}} * x_{12} + \frac{\partial L}{\partial O_{12}} * x_{13} + \frac{\partial L}{O_{21}} * x_{22} +$$

$$\frac{\partial L}{\partial O_{22}} * x_{23}$$

$$= \frac{\partial L}{\partial f_{21}} = \frac{\partial L}{\partial O_{11}} * x_{21} + \frac{\partial L}{\partial O_{12}} * x_{22} + \frac{\partial L}{\partial O_{21}} * x_{31}$$

$$+ \frac{\partial L}{\partial O_{22}} * x_{32}$$

$$= \frac{\partial L}{\partial f_{22}} = \frac{\partial L}{\partial O_{11}} * x_{22} + \frac{\partial L}{\partial O_{12}} * x_{23} + \frac{\partial L}{\partial O_{21}} *$$

$$x_{32} + \frac{\partial L}{\partial O_{22}} * x_{33}$$

M	T	W	T	F	S	S

Page No.:

Date:

$$\begin{bmatrix} \frac{\partial L}{\partial f_{11}} & \frac{\partial L}{\partial f_{12}} \\ \frac{\partial L}{\partial f_{21}} & \frac{\partial L}{\partial f_{22}} \end{bmatrix} = \text{convolution} \left(\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} , \begin{bmatrix} \frac{\partial L}{\partial O_{11}} & \frac{\partial L}{\partial O_{12}} \\ \frac{\partial L}{\partial O_{21}} & \frac{\partial L}{\partial O_{22}} \end{bmatrix} \right)$$

where,

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} = \text{input } x \qquad \begin{bmatrix} \frac{\partial L}{\partial O_{11}} & \frac{\partial L}{\partial O_{12}} \\ \frac{\partial L}{O_{21}} & \frac{\partial L}{O_{22}} \end{bmatrix} = \frac{\partial L}{\partial O}$$

loss gradi

from previous layer.

M T W T F S S
Page No.:
Date:

* Methods to avoid overfitting

-> Disadvantage of CNN
(1) Overfitting

Q -> why overfitting ?
Ans -> As you are increasing the neural network (the number of layer keep increasing the number of neuron keep increasing the number of neuron keep increasing. even in CNN this time instead of using one weight per neuron we are using per particular filter. lots of number of everything what i mean to say basically is the number of trainable parameter keep increasing this might be some reason cause of overfitting. This is a big problem

Q -> There are few thing a tricks we can do to actually avoid overfitting.
Ans -> we have (Things are use for Regularization)
(1) Batch Normalization
(2) Drop out
(3) weight initializer (Kernel initializer)
(4) Early stopping.

1 → Batch Normalization → when you are inputing data. so you can input one data might be very different from an anothe data.

so let say you have white dog there. so you have a picture of white dog there. you are doing some classification. so this in the training. so your model is totally trained on white dog. but you get a black dog there. so in that case in the testing thing you need to understand is the distribution for white dog is going to be ~~very different~~ a bit different from black dogs. you in that case the distributed are different we will require something. something so that despite the color or despite different in distribution a model is able to

Q - what Batch Normalization Basicaly does?

it Keeps different ~~body~~ Values in or Batches in particular distribution so that you do not get ~~of~~ diffected from that.

Input → Values of x over a minibatch : $\mathcal{B} = \{x_{1 \ldots m}\}$;
parameters to be learned

Output : $\{y_i = BN_{\gamma,\beta}(x_i)\}$

$$\mu_\beta \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$$ // mini batch mean

$$\sigma^2_\beta \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_\beta)^2$$ // mini-batch variance

$$\hat{x}_i \leftarrow \frac{x_i - \mu_\beta}{\sqrt{\sigma^2_\beta + \epsilon}}$$ // normalize

$$y_i \leftarrow \gamma \hat{x}_i + \beta = BN_{\gamma,\beta}(x_i)$$ // Scale and shift

Batch normalization means when you are inputting data you always input in Batches you might have learned about mini batch and gradient descent and batch gradient descent everything so what so what batch normalization basically means is when you input the data or feed the data into the model you feed it in batches so you can normalize

that every time.

and also batch normalization is basically a ~~bab~~ ~~label~~ layer Keras. So you can use this even after some different layers (like → Dense layers) usually we use Batch nomalization in the middle after convulation layer ~~or~~ or after a maxpooling layer. So that basically helps is even ~~you~~ when you output the value the ~~ou~~ output can be very different from distribution.

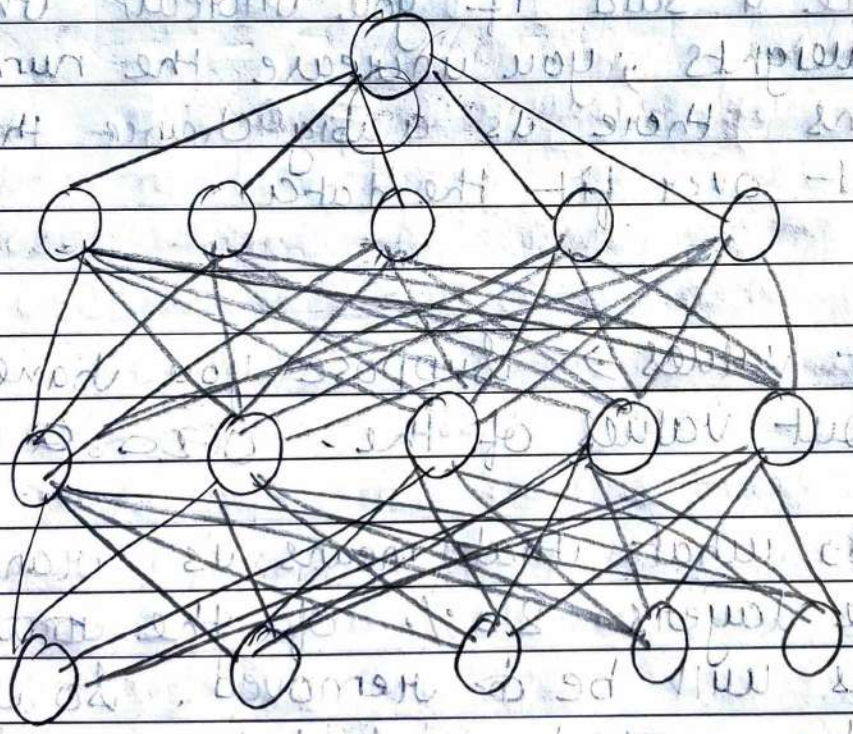
So batch distribution ~~normally~~ basically helps to normalize and keep them with in the same distribution so that you get accurate result. also it has been seen that batch normalization helps ~~o~~ in converging data, so you get to the result faster.

Pros	Cons
- it helps in converging.	

2 -> Drop out -> So drop out is one of the early stage discoveries and this was basically use to actualy understand and regularize the input more.

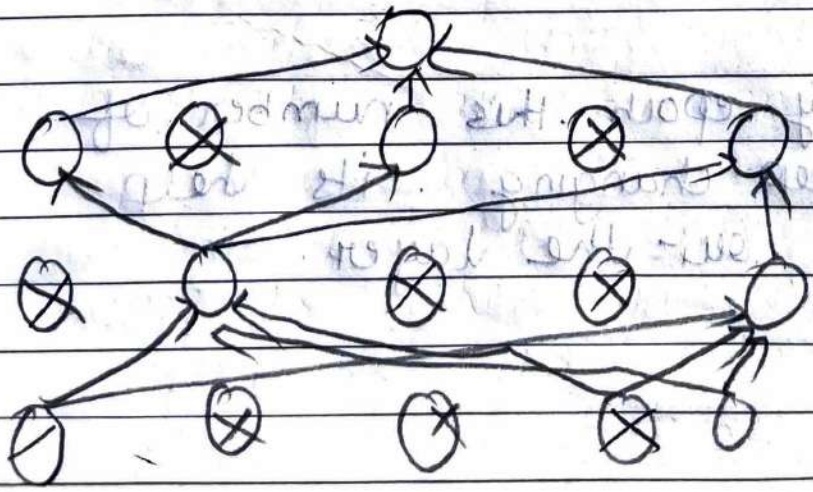
Even, when you were. i suppose starting you might have heard about L1 and L2 regularization.

So Drop out also one of regularization.



MLP

a -> Standard Neural Net.



After applying dropout

Date:

for every dropout you have dropout values. So what happens is you have the diagram of standard neural net this a multilayer perceptron. you can also use this in CNN. So what happen is

(ii) in standard neural net -> all of this are very connected it takes first lot of time. and you can have problem of overfitting.

like i said if you increase the number of weights, you increase the number of neurons there is a big chance that you might over fit the data.

in drop out values -> suppose you have a drop out values of the.

0.2. So what that means is randomly from all the layers 20% of the neurons will be removed. So rest will be used to calculate the whole thing.

Infact for every epoch this number of neurons keep changing. its help in thining out the layer.

Q -> why ~~this~~ is this useful, so what happen with thinning out is at any point of time very few neurons are active less neurons means chances of overfitting are less.

Sometime the data is less but your trainable parameter are alot. so in that case it is good to use dropout it has shown through result

Q3- Early Stopping -> I am running model and i have run that for 300 epochs but mid of this things when i am ~~learning~~ running ~~when~~ ~~i am~~ and when i am seeing the whole data that shows written how much improvement is ~~does~~ done, what is the test loss, training loss every thing. So i am sees that my loss not decreasing any more and my accuracy is not increasing and let say this is about 150 epochs ~~and~~ ~~so~~ so there is no use in running it exhausting my resource ~~of~~.

~~Ear~~ -> Early Stopping -> means is there are lot of matrix ~~is~~ in early stopping. but one of the thing is ~~or you when you want to stop you can stop. I can say~~ when ~~my error not decreasing~~ an!

you can decide how much you want to anywhere you want to stop so you can say let say when my loss stagnent

Q - How can you find your loss is stagnent.

when the [previous epoch - recent epoch] the difference is close to 0.

So you actually fit in factor. you if let say my loss = 0.005 if my loss isn't greater than this. I will keep going if the loss get less then 0.005 then stop.

if the loss 'loss' 0.005 for more than 2 or 3 epochs so that is known as patience. So what patience is basically like 2 or 3 epochs ~~for 2 or 3~~ difference in the losses is this little. If the different 0.005 then why we waste my time even epochs when I know they can have a chance of overfitting so thats why we can use early stoping and they can stop the whole thing.

Page No.:
Date:

4 -> weight initializer -> they also basically keep a check on overfitting this is also a regularizer.

He, normal, glorot ->

Q -> Today we discussed about different case study of CNN

So, you have a convolution neural network. and now there are lot of ways in which you can define and make different convolution network.

-> you can different activation function
-> you can change the number of layer
-> you can change the number of kernel in each layer

The whole thing is it can keep changing. So, what people have done is they competition which is known as Image net. Say imagenet have a competition there people have to build more and more model and they have to get better accuracy. So image net is basically classification proble. it has thousand classes.

So, we will now be studying about this different case studies of CNN, that means this different models, architectures, that has experimentaly be proven. Scientifically also. that they perform better than other architecture. I can make architecture but that is not a thing.

1 - AlexNet → 1st Architecture

Page No.:
Date:

-> It was in 2012

-> This is a very ~~deep~~ big deep architecture and it was very when it came out 2012 it took like ~~5~~ 5-6 days to complete the training it had 60 Million parameters and that time is huge

-> Alex net has 5 convolution layer and 3 fully connected layer.

-> Input is 227 x 227 x 3

-> after convolution what happens is there is a something known as Relu that takes you might have ~~[illegible]~~ remember this are activation function

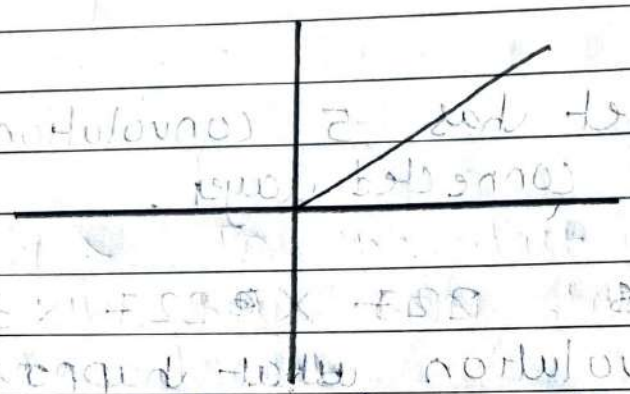
-> Alexnet the kernal/filter width is equal to the height, So there are basically Square kind of Kernal except for channels or the depth which is equal to channels of the image.

-> They have found is you have one image with a depth of 3 and whole basic thing about this is making ~~[illegible]~~ height and width smaller and its depth you keep increasing. It is basically what they have found ~~[illegible]~~ out work the best.

Date:

Q -> what made alexnet a good network?

Ans - (A) AlexNet had
1 - Relu - Activation -> This is a very good activation layer because if you ~~remember~~ might have remember



-> Relu has help in better convergence, easy convergence
-> Relu has faster convergence

2 -> After convolution passes through a Relu Activation.
but after that it also passes through local normalization.

(local normalization was later removed from alex net) ~~and~~

Local normalization is in the basically first 3 convolution layers.

Date:

3 -> Drop out ->

* Problem with alexnet

like i had mention alexnet have 60 million parameters. this brings us to our problem of overfitting like i mentioned.

So, we need some regulizers. to prevent overfitting hence we have this concept of Dropout.

4 -> so we have 256 X 256 as an image. but then we are using only 227 X 227 although the paper mention 224 X 224 but you remember its a 227 X 227 because only through using this data i will getting the actual values. & 224 X 224 is missprint.

-> we have 256 X 256 we use to do random crop so, one of this we avoid overfitting is increasing a data. this is done through Data Augmentation

-> Data Augmentation is very much use when it comes to problems with computer vision.

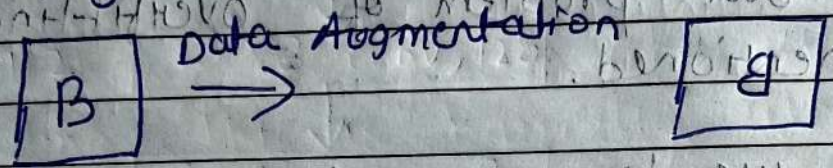
-> CNN is a part of computer vision

M T W T F S S

Page No.:

Date:

* Augmentation -> Basically means increasing the size of the dataset by random flip may be, random crop, random upside down, tilt the image like -> you have an image.



* -> In the last class we talked about the basic CNN structure that was the alexnet.

Alexnet was release in 2012. Alexnet is the greatest model at that time.

Image net competition keeps going on every time the competitor have come with better.

What happen in 2014 was the university of oxford they came up with a architecture or a model which was known as the VGG

VGG -> Visual Geometry Group. (This was released by the of oxford)

Page No.:
Date:

→ So, what is main difference that you can see here, in AlexNet what they had was ~~they~~ there was one convolution which was followed by one Max pooling. then again convolution and this went on until the very end where what they did was they add 3 convolution then one Max pooling.

In VGG → Totally different thing. Here you have 1 convolution which is followed immidealy by another convolution then we have a max pooling which decreases the dimension now again you have 2 convolution layer.

in VGG the convolution layer are the one after the other.

in alex net padding was valid

VGG → Padding is same for 2 ~~convolu~~ convolution continuously.

* Structure of VGG -> Visual Geometry Group.

Size : 224 | 3x3 conv, 64

↓

3x3 conv, 64

↓

Pool /2

↓

Size : 112 | 3x3 conv, 128

↓

3x3 conv, 128

↓

Pool /2

↓

Size : 56 | 3x3 conv, 256

↓

3x3 conv, 256

↓

3x3 conv, 256

↓

Pool /2

↓

Size : 28 | 3x3 conv, 512

↓

3x3 conv, 512

↓

3x3 conv, 512

↓

Pool /2

↓

3x3 conv, 512

↓

Size = 14 | 3x3 conv, 512

↓

3x3 conv, 512

Pool/2

↓

4069 -> FC -> fully connected

↓

Size=7 fc 4096

↓

fc 4096

So this is basically the structure of VGG if you write in this obviously. So first

→ So first what you have is the kernels keep increasing

Now, earlier is 64 becomes 128 then 256 then 512, and so on and so one thing you need to know is earlier when kernel where increasing it was 96, 358 it was very randomly increasing in AlexNet.

Now in VGG. the 2^n

So, 2^n it is increasing in that format so, this $2^6, 2^7, 2^8, 2^8$ and so on.

and VGG → itself has a different layer.

So, you have → vgg 11, vgg 13, vgg 16, and the vgg 19.

M T W T F S S

Page No.:

Date:

-> VGG 13 and 11 are not used a lot.

-> VGG 16 and 19 are used a lot.

-> So, we have pooling, so obviously its max pooling this is basically used to decrease the dimension of height and width and increasing the kernel size at the same time.

Q -> Lets see more depth what other difference there is between VGG and alexnet.

and why VGG covers in places where Alexnet fail. and what were the failure

-> So, you have another structure.

	Layer	feature Map	Size	Kernel	Stride	Activation
Input	image	1	227 X 227 X 3	-	-	-
1	2 X convolution	64	227 X 227 X 64	3X3	1	relu
	Max pooling	64	112 X 112 X 64	3X3	2	relu
3	2 X convolution	128	112 X 112 X 128	3X3	1	relu
	max pooling	128	56 X 56 X 128	3X3	2	relu
5	2 X convolution	256	56 X 56 X 128	3X3	1	relu
	Max pooling	256	28 X 28 X 256	3X3	2	relu
7	3 X convolution	512	28 X 28 X 512	3X3	1	relu
	max pooling	512	14 X 14 X 512	3X3	2	relu
10	3 X convolution	512	14 X 14 X 512	3X3	1	relu
	max pooling	512	7 X 7 X 512	3X3	2	relu
13	FC	-	25088	-	-	relu
14	FC	-	4096	-	-	relu
15	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	softma

~~This is the sh~~

→ let see the structure which was given in a paper.

So. A, A-LRN, B, C, D, E are the different model of VGG.

So, 2 different VGG 11 and you have VGG 13 ; 2 VGG 16 and 1 19.

Now let us look into the difference.

M T W T F S S
Page No.:
Date:

-> first of all you might have notice (Ox) it does not have very large kernels. if you remember what we did in alexNet we had 11 x 11, 5 x 5 all this big size kernal. But here all the kernal are very small size. why is that? one thing is that as the model size increases as the number of layer increase. as the number of parameter keeper increasing.

-> Alexnet has 60 million parameter but the problem is we need to find an efficient way where we can increase the number of layers or increase the depth but at the same time we do not increasing the parameter a lot. So that is why the have found very nice way to find the kernel size

-> They found out instead of using 5 x 5, we can use two 3 x 3 layers. so the output is same. 5 x 5 is very equivalent to two 3 x 3 layers.

-> but here two 3 x 3 layers they actually reduce the number of parameter

How? Say input we have

input and output channels.
and you have two $((3\times3)\times(C\times C))$
means if you have C channels and 3x3
is your kernal size and you are using
2 kernals obviously

$$= 2((3\times3)\times(C\times C))$$

$$= 18\times C\times C$$

→ It was trained for 2 weeks on Nvidia Tidan GPU

⇒ two layer decrease reduce 4 million in number of parameter.

*> Lets talk about
conv1 -256

1x1 kernels were added to decrease the number of parameters without change the shape or width of the kernal, so, this is basically the whole thing the whole input size is equal to the output size

but the number of parameter decrease.
So that is why 1x1 use a lot and
this is also use in non linearity.

→ if you are using ReLU there is a very
big chance and also you are using convolution

So convolution is basically what it adds
more linearity to thing. you just have
multiplication followed by addition, addition
is basically linearity.

→ So, To add more non linearity we use
1x1 kernals.

Q → How its training
→ In training what they did is
for optimisers they use SGD with
momentum. and the momentum is 0.9.

They used a Batch size for training was
256.

the regularization they use was → L2 regularizer

at the same time its use Drop out
and the Drop out value is → 0.5

So, 50% of the neuron where removed at
any training epoch randomly.

finally VGG is Better then Alexnet

M T W T F S S

Page No.:

Date:

-> VGG got more Accuracy then AlexNet

-> VGG help us to show how we could optimise the model, that means VGG increase and showed us how we could increase the layers without increasing parameters by a lot

-> And they found out How there small Kernel were very close and give very precety accurate result but at the same we reduce the parameter that were required to same

-> finally the weights were initialize normally and the weights lies between (0 and 0.01)

Bias is set to 0 -> but obviously with training it change.

But this was the initial way How they have set.

M T W T F S S
Page No.:
Date:

* In last classes discussed about AlexNet, and VGG (19, 18) → because VGG (16 and 19) are most we VGG

Q => what are main problem we are having with VGG and AlexNet

→ well as we increase the number of layers we still had the problem of more parameters so, that means we had chance of overfitting
we did not want that. we wanted something a model that could good deep but could not causes overfitting.

even with VGG (16 - 19) they had 1x1 → convolution but even that could not help that much.
we still needed something more
then we come up with the next model
inception net

Inception Net → So, over the years inception net has grown more and more and we have inception V2, V3 and different version of inception net. but for now we will be studying V1 and then later on we will be going back and learning about V2 and V3. for now lets talk about V1

Page No.:
Date:

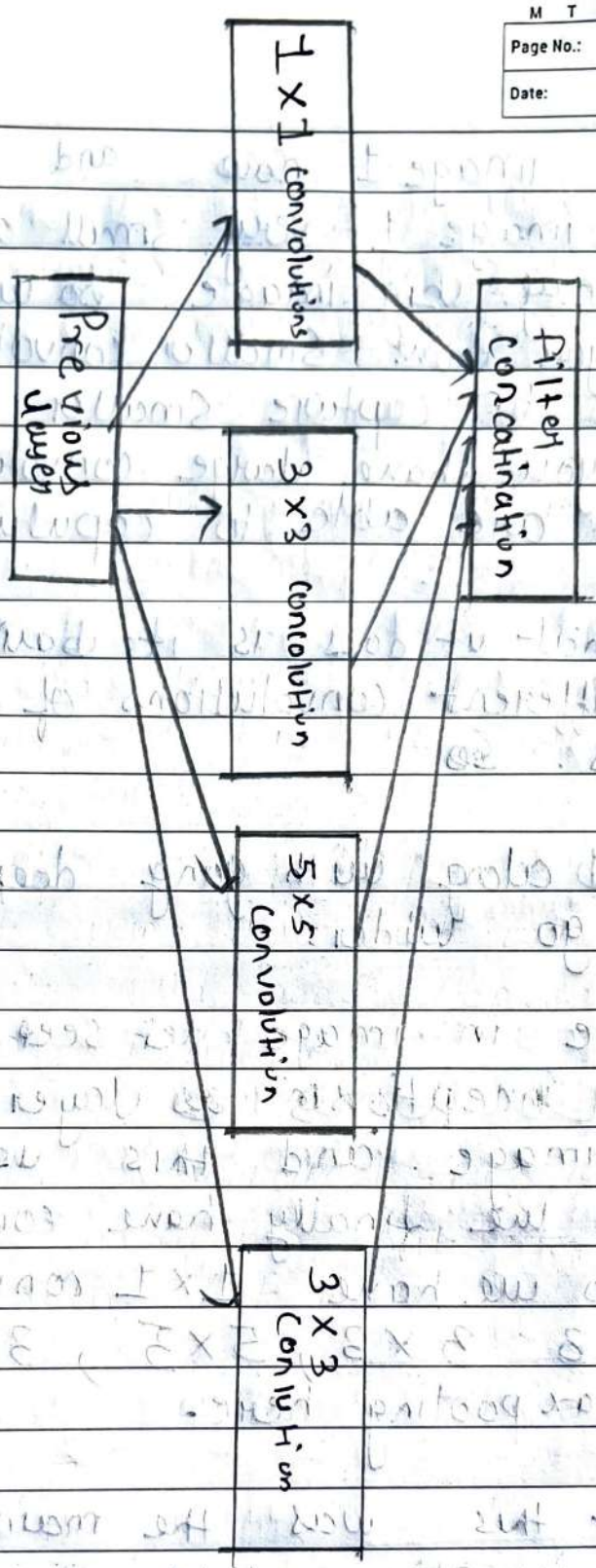
* first problem we knew about why was even after batch normalization and Relu and every thing we still had the problem of overfitting.

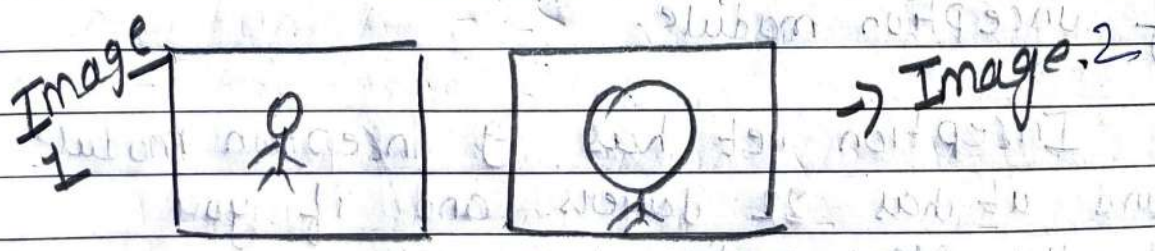
But we know if we have to get more deeper models, more accuracy we need more layers and hence our parameters are going to increase they keep on increasing hence inception net was actually a very well engineer design ~~program~~ Architecture so it was very ~~easier it~~ optimise

→ for that time it was very well engineered architecture which give not only very good accuracy at the same time it decreases the trainable parameter a lot.

Inception net → The main idea behind inception net is basically for going deeper we should try going wider as well. so

Q → Inception module, naive version





Suppose here you have two images. the main idea behind inception net was a dispite the size of image you might still be able to find an object. So let say you have

two image 1 and image 2 but in image 1 very small object and image 2 very big image so what it does is if you have smaller convolution you are able to capture smaller feature but if you have large convolution then you are able to capture larger feature

So, what it does is its basically divides into a different convolutions of different kernel sizes.

along with going deeper we should also go wider.

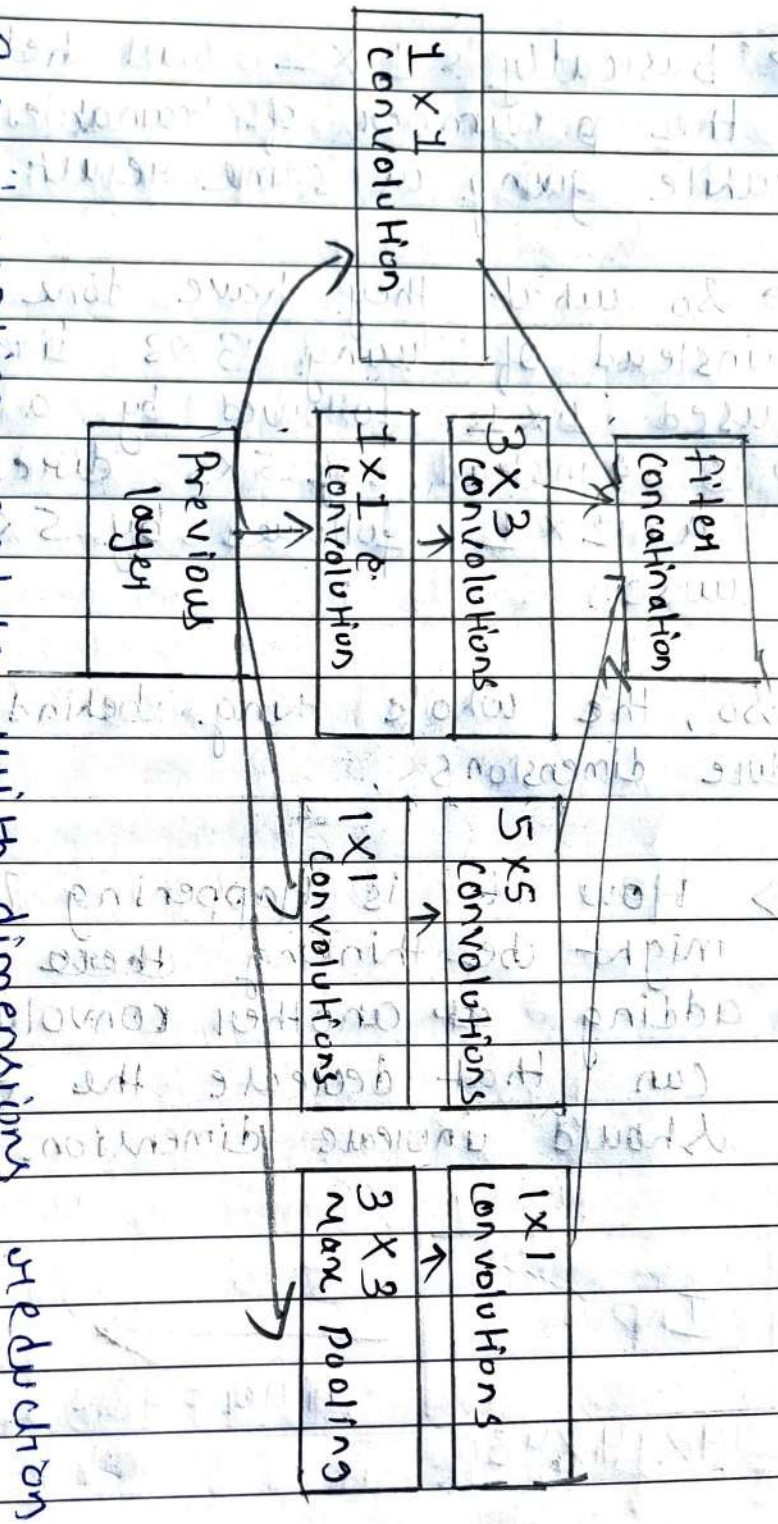
Here in image net see → The previous layer we have in image and this is going wider then we finally have concatination.

So we have 1×1 convolutions, 3×3, 5×5, 3×3 but this max pooling here.

this was the main naive version of inception module.

→ Inception net has 9 inception module and it has 22 layers. and if you add the Max pooling layers it has 27 layers.

⇒ Also this is designed by google the first inception net is known as google net.



B-> inception module with dimensions reduction

-> lets understand dimension reduction. So if you remember when i told earlier while we are learning about VGG -> i told that 1 x 1 convolution help in increasing non linearity. but they also help in reducing dimensions. Now we will talk about that more.

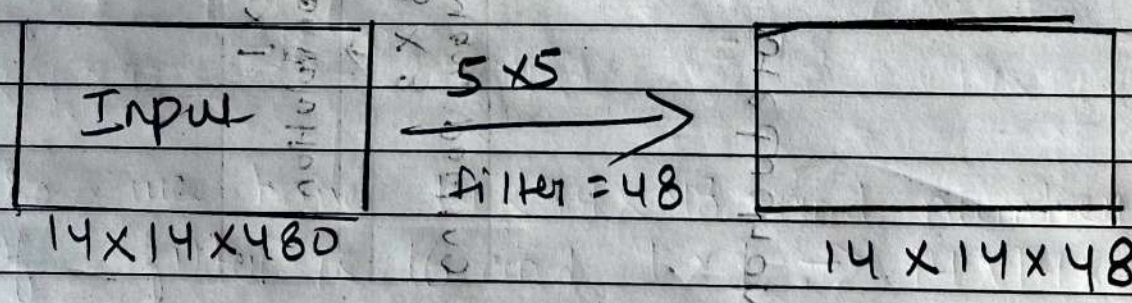
Page No.:
Date:

→ basically 1 x 1 will help in reducing the number of trainable parameter while giving us same result.

So what they have done is ~~instead~~ instead of using 3 x 3 directly they have used 1 x 1 followed by a 3 x 3, then there also instead of 5 x 5 directly it has a 1 x 1 followed by 5 x 5 and like wise.

So, the whole thing behind this is to reduce dimensions.

Q → How it is Happening? because you might be thinking ~~there~~ here we adding another convolution layer How can that decrease the dimensions. it should increase dimension.



$(14 \times 14 \times 48) \times (5 \times 5 \times 480) = 112.9M$

← So we have a input layer which is 14 x 14 x 480 and you have 48 filters and were directly doing a 5 x 5 convolution and we getting 14 x 14 x 48 → 48 being the number

that means the padding is same.

first of all for each box we will doing 5x5 filtering for every box in output.

~~if you go~~ if you got one output you have to put this whole ~~part~~ ~~input layer~~ things in one part in input layer and then you had to calculate the whole convolution. So you have to do as many calculation So for each output you had to do calculation. So, that means we have to do the calculation

$$14 \times 14 \times 48$$

because we have convolving each and every block with the filters to get one block for output.

Hence we have to $(14 \times 14 \times 48)$ this many times. and since we are using 5x5 convolution to find one value

$$(14 \times 14 \times 48) \times (5 \times 5 \times 48) = 122.9m.$$

~~So if you if you were boss, so for every one block we are doing a convolution.~~

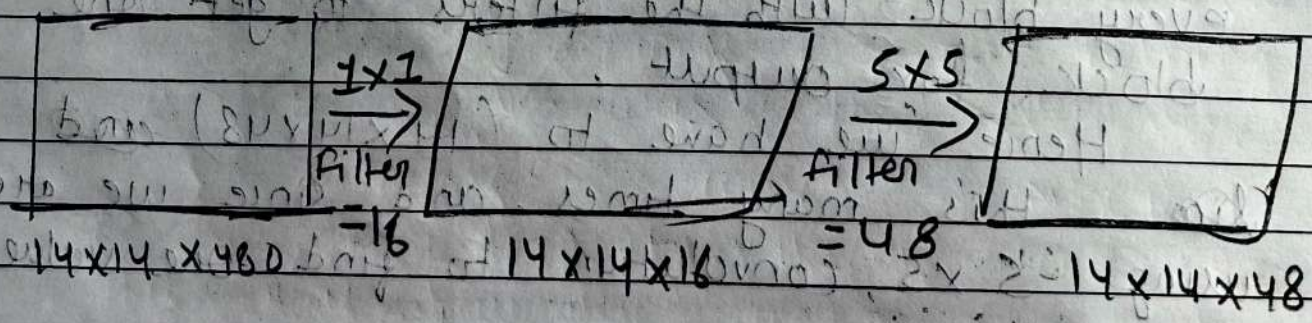
So for every one block we are calculating a convolution here.

So that if i one block, i will be doing 5X5 convolution in this case, but we have 14X14X48 blocks. So i will be doing

14X14X48 and each block of 5X5 so that is multiply. and since we have not one filter, we have 48 filters so 5X5 is multiply with 5X5X48

$$(14 \times 14 \times 48) \times (5 \times 5 \times 480) = 172.9\,M$$

Q -> How can be reduce if we use 1X1 convolution and then 5X5



So we have 14X14X48 and number of filter is 16 and 1X1 -> 14X14X16

-> 14X14X16 this will be multiply with 1X1 and then it will multiply with 480 so we get 1.5M.

M T W T F S S

Page No.:

Date:

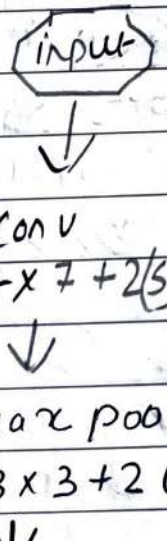
→ Number of operations for 1 x 1 convolution =
$(14 \times 14 \times 16) \times (1 \times 1 \times 480) = 1.5M$

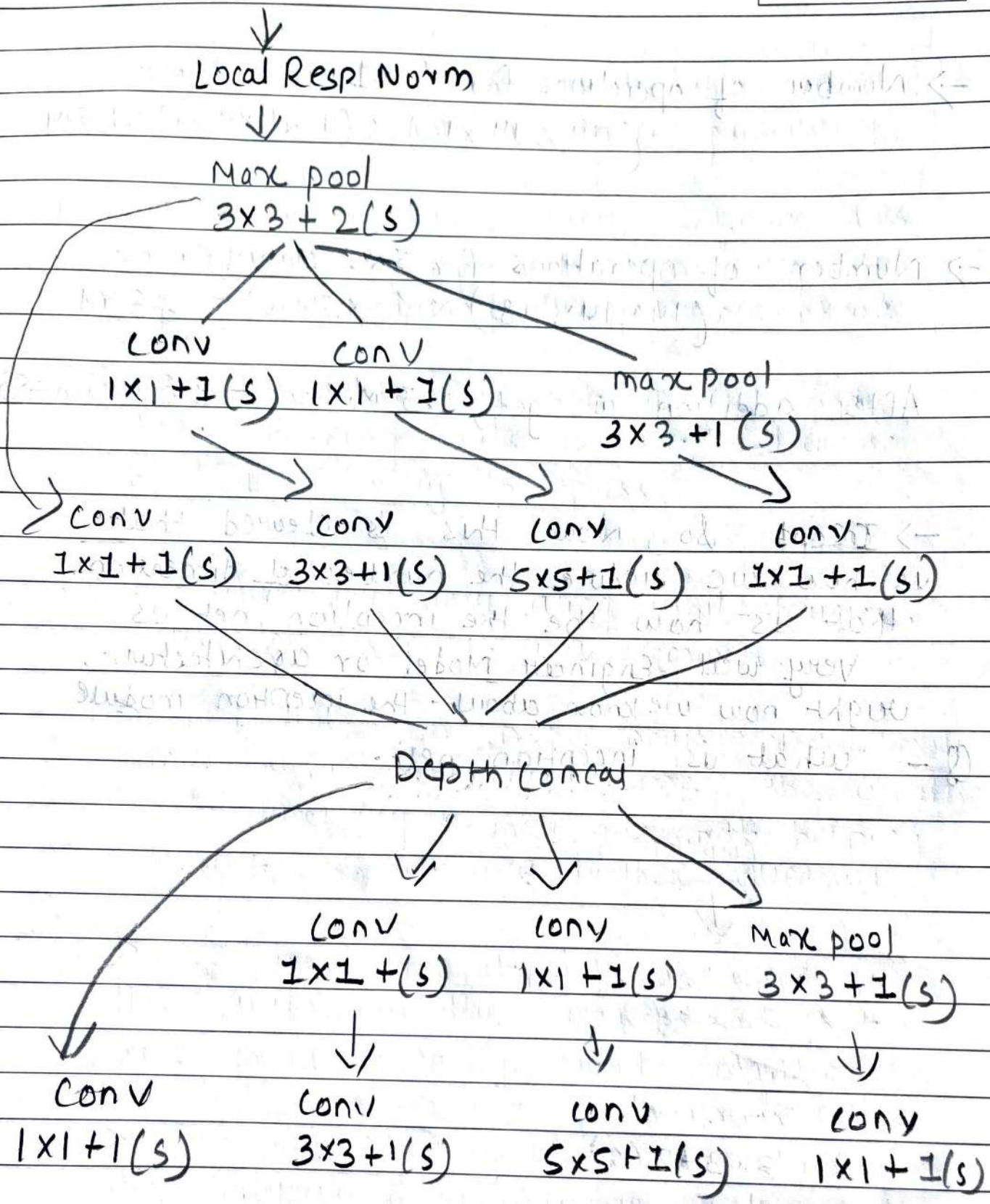
→ Number of operations for 5x5 convolution =
$(14 \times 14 \times 48) \times (5 \times 5 \times 16) = 3.8 M$

After addition we get, 1.5 Million + 3.8 Million = 5.3

→ ~~Incept~~ So, Now this is cleared that is how we reduce the number of dimension. that is how the the inception net is very well Engineer Model or architecture, right now we know about the inception module

Q - what is inception net.




Local Resp Norm
Max pool
3x3 + 2(s)
conv
1x1 + 1(s)
conv
1x1 + 1(s)
max pool
3x3 + 1(s)
conv
1x1 + 1(s)
conv
3x3 + 1(s)
conv
5x5 + 1(s)
conv
1x1 + 1(s)
Depth Concat
conv
1x1 + (s)
conv
1x1 + 1(s)
Max pool
3x3 + 1(s)
Conv
1x1 + 1(s)
conv
3x3 + 1(s)
conv
5x5 + 1(s)
conv
1x1 + 1(s)

M T W T F S S
Page No.:
Date:

→ you have nine inception module and you have 2 Auxilary Branches

Auxilary Branches → are Branches that have been seperated from the actual model in the the side to find the loss.
So, this was basically done in the training so, that it could understand how these modules were working together as a whole and which module was giving perfect result.

and it is adding a new module worth prediction or not

So, in the final output, the final loss was calculated from softmax2, softmax1, and softmax0, together.

* Summary :-

1 → 28 × 10 × 10 convolution kernels are used to reduce dimensions and modify linear activation units (like relu)

2 → A fully connected layer of 1024 units and a modified linear activation unit (like relu, tanh)

→ By the way it was trained on the CIFAR

Date:

3 - A drop out layer that drops neuron connections with a 70% probablity.

4 -> A linear layer with softmax loss (a classification predict 1000 categories) but removed during the inference phase

5 -> The momentum is set to 0.9 and the learning rate is set to decrease by 4% every 8 epochs.

-> Auxilary only using while training. in the testing it was removed.

-> and later on the new inception architecture came the realise that auxilary branches are not helping a lot. So it was further removed.

-> So, the dropout rate is 0.7 so this was basically used. So, that we could actually avoid overfitting.

-> Because Now you can understand the reason increasing the number of layer.

in VGG -> maximum is 19 layers

in inception -> we have 27 layers. if we also include the pooling layer

M T W T F S S
Page No.:
Date:

-> The dropout was used to prevent overfitting to act as a regulizer.

-> it kept on decreasing the learning rate every 8 epochs.

Q - So what was difference in inception net was.

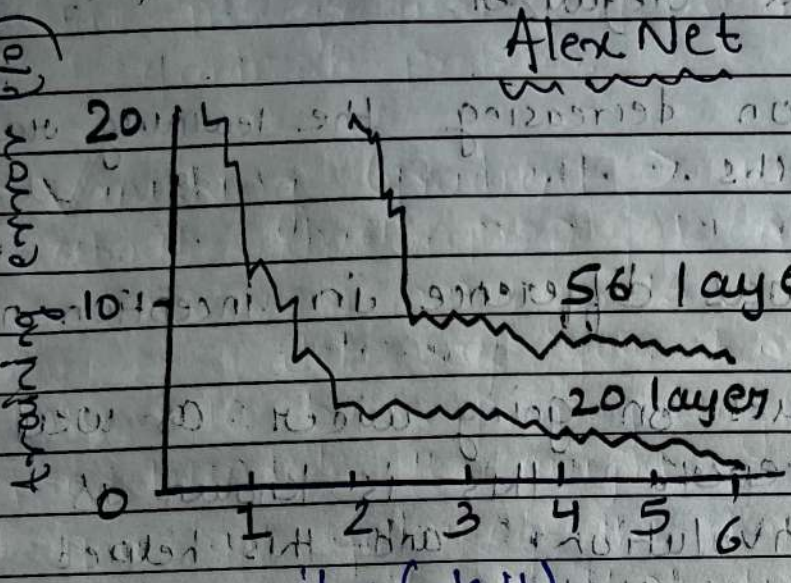
Ans -> it focuses on going wider as well as going deeper. this is known as sparse convolution. and this helped to actually understanding in depth mode how this can be used. How we not only need deeper models we can actually try going for sparser and wider model at same time.

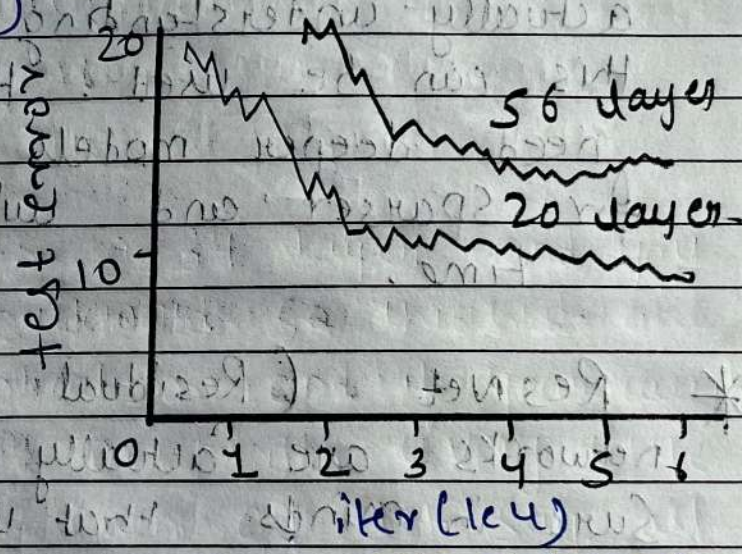
* ResNet (Residual Network) -> so this networks are actually designed keeping making sure in minds that we can go very deep in layers without actually having a lot of problems

-> problems -> so, one problem that we know that happens when going very deep or increasing a number of layer by a lot is overfitting. but there are other problems as well and we will be discussing about those problem then we will understand so, why we required residual network. and How the Actual Architecture is,

Page No.:
Date:

→ first we will discuss about module





So, here we have a simple AlexNet network, ideally what happen is if you think about it, if you increase the number of layers the accuracy should increase. that is the most ideal solution that we always know. increasing the number of layers going deeper will increase the accuracy, that is not the case.

So residual network the paper show the training error for 20 layer

of alexnet. then again they have trained the same thing with 56 layers of Alexnet and you can see very easily that the training error is very high for 56 layers. although this is the more deeper kind of network.

So what is happening exactly? Here you can also see in Test error also. the 56 layer does not perform that good as compare to 20 layers. However the 56 layer is a more deeper kind of network. So what is the problem

→ we have overfitting. but then we have ideas to prevent overfitting. So overfitting is not the only problem that we are having.

→ Network degradation → if you increase the number of layer. you network is going to degrade. that means its going to perform verse worm. but in theory we all know we have all read that when you increase the number of layer it should ideally increase the accuracy. or decrease the loss.

But there is something that is lacking in the network problem or in the trainging and practicle problem that is causing us there problem. So we will be looking into what is happening.

M T W T F S S
Page No.:
Date:

→ Network degradation → facing this problem due to gradient explosion or vanishing gradient problem.

→ Vanishing Gradient → when I was taking about back propogation the problem we were having is

Let Say you have 34 layer and you are back propogating the whole loss and to update the weights. So, for updating you are following the chain rule. that means this is all multiplication.

So, $\frac{dL}{dw_1} \times \frac{dw_1}{dw_2} \times \frac{dw_2}{dw_3}$ goes on.

now if this is a 34 layer and you have to go very backward so this is a very big multiplication and if your weights or values are updated very little let say ~~it is even~~ let say all of this are 0.1 and now is being updated 34 times $(0.1)^{34}$. this basically comes out to be nothing that mean $(0.1)^{34}$ is very negligble value

so that means its very close to zero. you are not updating the value any more. this is known as a vanishing gradient problem.

that means ~~the gradients~~ because of the gradients or chain rule the whole update thing is getting

vanishet. Hence Vanishing Gradient.

* Gradient explosion ⇒ The Gradient becomes very large. So
let say

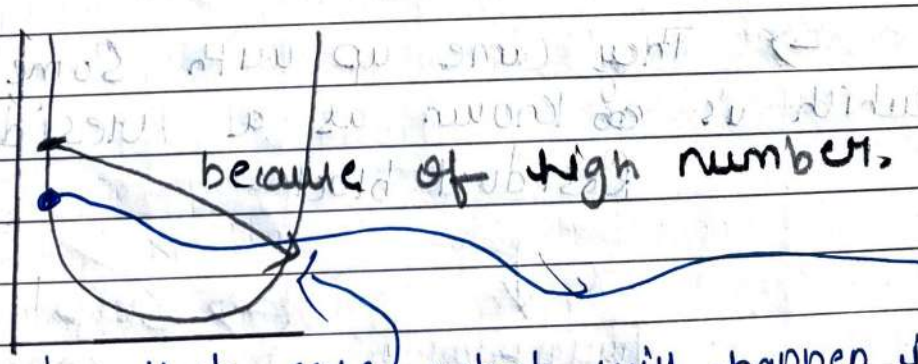
$$\frac{\partial L}{\partial O_1} \times \frac{\partial O_1}{\partial O_2} \times \frac{\partial O_2}{\partial O_3} \text{ and so on.}$$

Suppose, instead of 0.1 let say we have 2 and updated value.

So now $(2)^{34}$ is way much. its very big its very large value.

So what will happen in that case is the gradient will go very high. Since the gradient goes very high. your update is massive.

let say you have plot and you to reach minimum value.



So in that case what will happen is it will skip the whole part. it come So come where here. and then again you update but since this is a very multiplicative problem here again it will skip. and may be let say come has

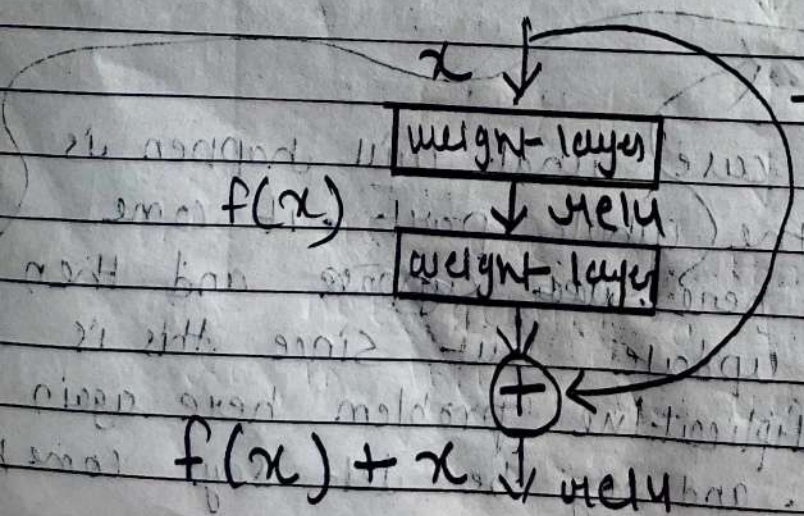
So what will happen is it will just keep bouncing back. and it will never be able to reach the minimum value. this problem is knows as Gradient explosion problem

So we understand the vanishing Gradient and Gradient explosion problem. this happens because you increase the number of layer. Now I showed for 34 layer. imagine its is 56 layer. ~~now your~~ and this leads to the degradation of the problem, hence we called as Network degradation, so this is ~~so~~ basically cause Explosion and vanishing Gradient

→ To Solve this problem what the ~~the~~ researcher did where they come up with a very brilliant solution

→ They came up with something which is ~~too~~ known as a residual block.

Residual block



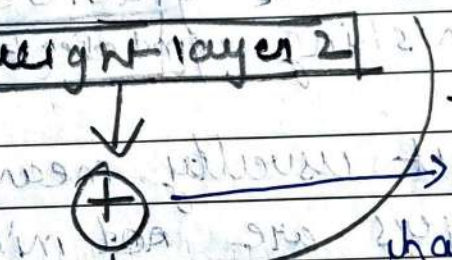
→ SKIP connection.

x

identity mapping.

what happens is basically suppose you have the X so you will do the whole con-volution weight layer in like MLP or what ever. so you will pass the whole thing to a layer. then you will do a activation function and pass it again to another layer. This thing basically is the basic neural network structure.

So after passing it to the second layer.



→ skip connection

→ Here what will happen is you will have an addition

So this addition is basically output that you get from ~~weight layer~~ ~~skip connections~~

weight layer 2 f(x) + skip connection. So you will add f(x) + x.

So in residual block they are adding a skip connection. ~~so you~~ f(x) + x. ~~so here also what they are doing is they are adding~~. and now, this will continue on with the same layer one after the other. So again you have a weight layer (x) this again

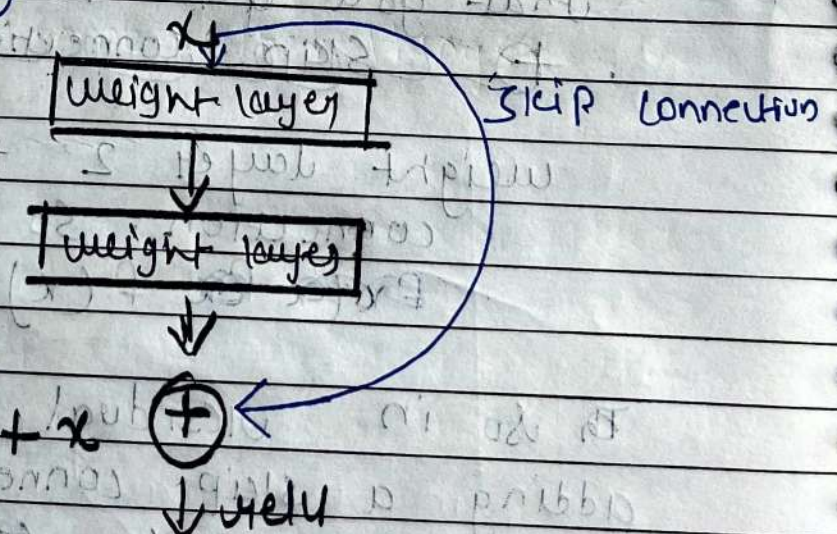
will pass through another weight layer. and might you have another skip connection there.

→ Just ~~remem~~ remember that first you'd add ~~this~~ ~~here~~ and then this is pass through an activation function.

* Let us break down what is happening. So what happens is you have this gradients. explosion and vanishing gradient problems.

So what usually means is. Sometimes some layers are ~~and~~ might not be important or those layer might not add ~~lot~~ lot of value to it.

So let say.



this is one of those layer you are not getting a features from there so in that case what happen is

this will contribute to either gradient descent or vanishing gradient problem since this make no sense why dont we keep that layer.

So in the Back propogation problem you can just say this layer here is not adding a lot of use so, we can totally skip this layer and we can go to the next layer.

$$\frac{\partial L}{\partial O_1} \times \frac{\partial O_1}{\partial O_2} \times \frac{\partial O_2}{\partial O_3} \cdots \text{ so on.}$$

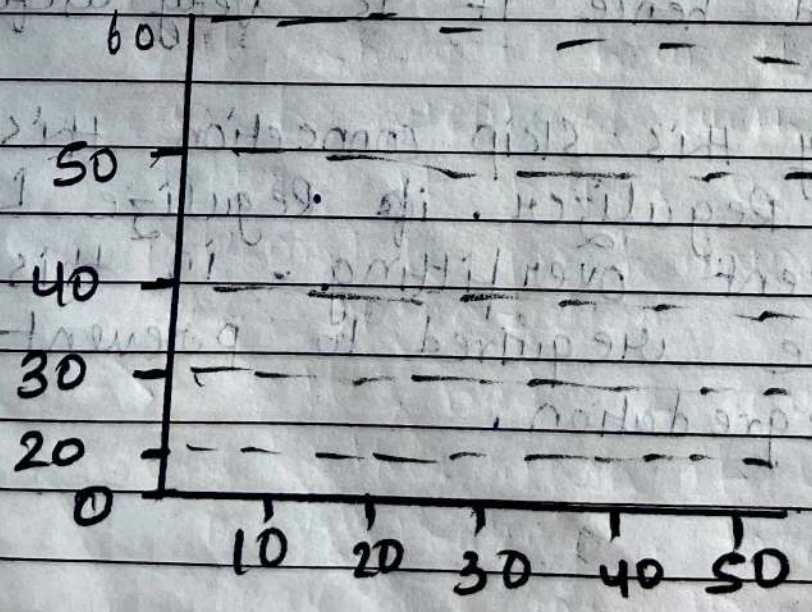
So in that way if this is not adding skip this layer, so this means is you dont have to deal with 34 multiplication. this narrows it down and it gives important to whatever network or whatever layer is important. So, the number of multiplication reduce and hence it is very useful.

-> So basically this skip connection this act as a Regulizer. Regulize is use to prevent overfitting. in this case they are required to prevent network degredation.

M T W T F S S
Page No.:
Date:

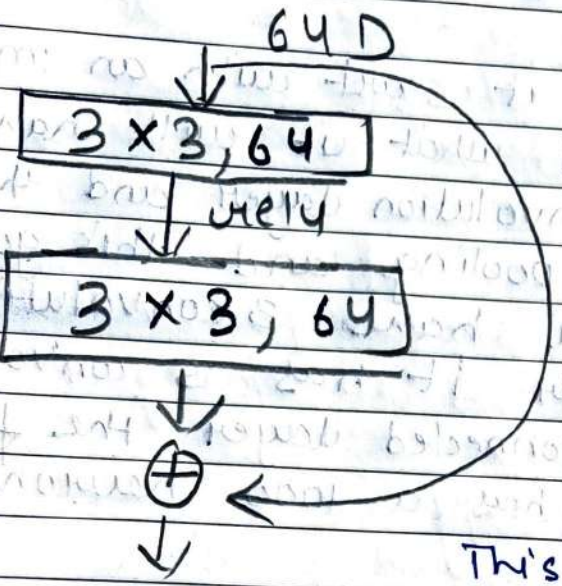
→ So Now what will happen is in last page diagram, that diagram show a layer and this layer is not such important. so you have x this will go on you will add and add the whole layer output with x. if this is not important this will technically very little. So you more focus on x part rather than the fx part. hence even if we increase the layer. This layers will been off not a lot of use so, that the model can itself decide this is required or not required. and then it can keep changing. so you need first add and this will followed by Relu.

* Lets look into some other things

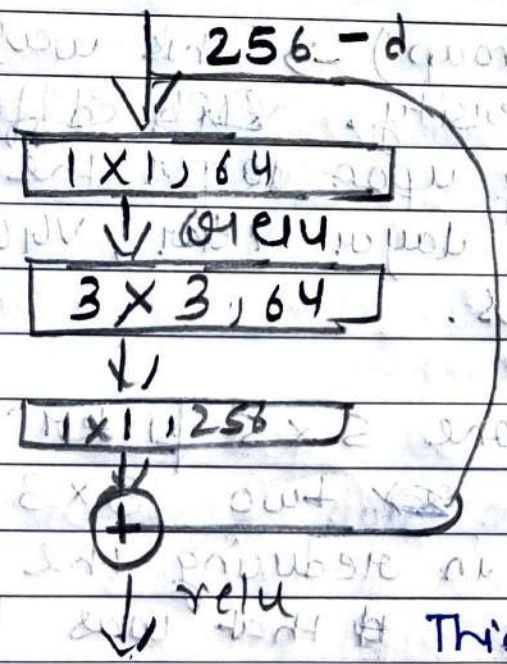


Page No.:
Date:

→ Now here



This is skip connection. that make sure that you have optimise network. or Optimise block. So what that means is basically if you remember. we add 1 X 1 to reduce dimensions. So here instead of two 3x3 it has done 1X1 followed one 3x3 and then again 1X1. So basically this 1x1 help in reducing the number of trainable parameters. reducing



This is an optimised version of network. and like inception net what will happen is residual networks itself has many different variation from time on it has being improved. So you have resnet → V_2, V_3 in the same way that you have inception net.