Convolutional Neural Network

Q. Why does we need CNN and what are the drawbacks of neural network ?

→ Suppose we have an image having the size of 1000 × 1000 px with RGB channels.

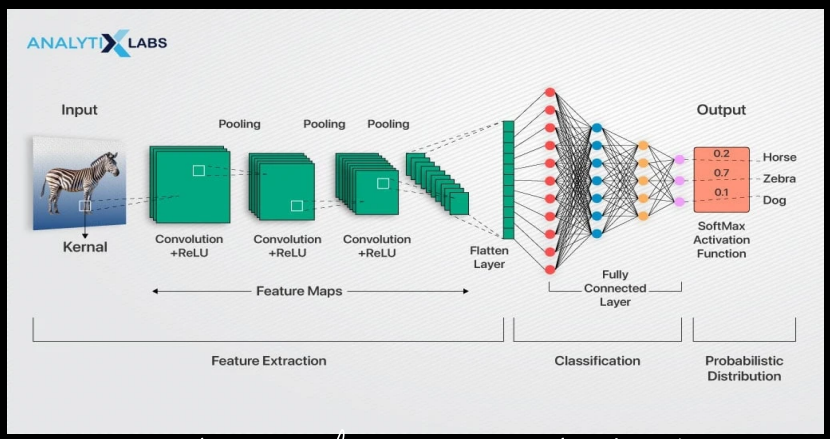
→ Hence the total number of I/p feature of NN will be $3 \times 1000 \times 1000 = 30,00,000$ (3M), even there will be lot more hidden layers exist to train entire network.

→ Our CPU/GPU my not able to handle this massive amount of calculation because it will be time consuming process.

→ More neurons and more number of weights my leads overfitting.

→ These are the limitations of traditional neural network.

→ This limitation are the mother of CNN.



Brief overview

→ CNN uses filters (Sliding windows), that detels features and pattern within images.

→ These filters helps to identify edges, colors, corners etc..

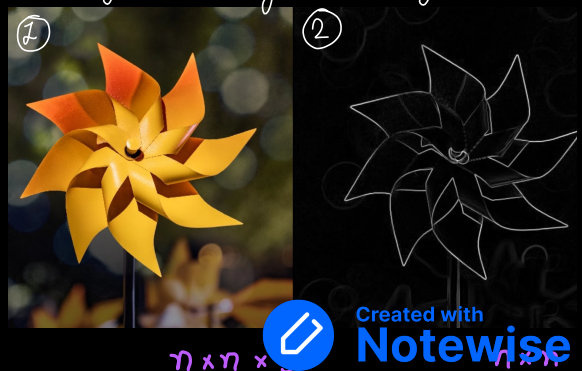
→ As below image shown.

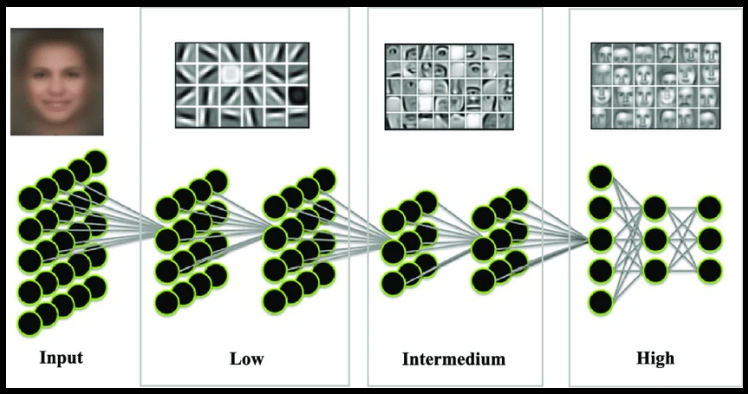
→ 1st image has n×n×3 dimentions but the 2nd image which represents edges of image has only n×n dim.

it means we have significantly reduced dimentions by preserving necessary patterns of data.

→ The filters will detect neccessary edges from image.

→ This edges will pass to the next layer, which will detect more patterns from the data. & this process will continue untill entire object is detected.





→ This is how CNN works

→ By the given input images, it will extract the necessary features like

Stage-1: Vertical / horizontal edges.

Stage-2: Face parts like nose, eyes, lips.

Stage-3: By combining all features it creates a picture of a face labeled as a human face.

Convolution Operation.

1	6	9	10	2	8
2	5	1	8	4	2
3	7	4	9	10	3
9	8	3	6	7	9
8	0	9	4	7	2
9	10	12	6	9	8

6 × 6

I/p grayscale image (0-255)

*

1	0	-1
1	0	-1
1	0	-1

filter 3×3

=

-8	-9	-2	14
6	-3	-13	9
4	-4	-8	5
2	2	1	-3

4 × 4

Visit : https://poloclub.github.io/cnn-explainer/ for better visulization.

Sum of element wise multiplication will generate a single number as answer like....

$1\times1 + 2\times1 + 3\times1 + 0\times6 + 0\times5 + 0\times7 + 9\times-1 + 1\times-1 + 4\times-1 = \boxed{-8}$

Now shift the filter to left side by one pixel.

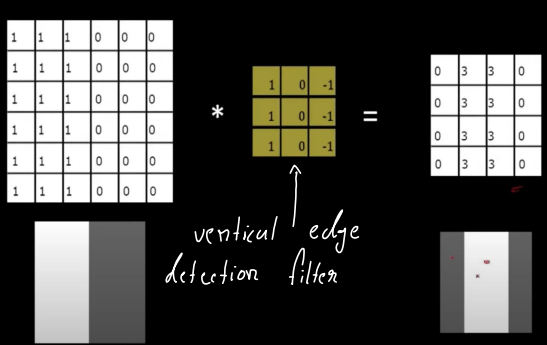
$6\times1 + 5\times1 + 7\times1 + 9\times0 + 1\times0 + 4\times0 + 10\times-1 + 8\times-1 + 9\times-1 = \boxed{-9}$

Now shift the filter to left side by one pixel

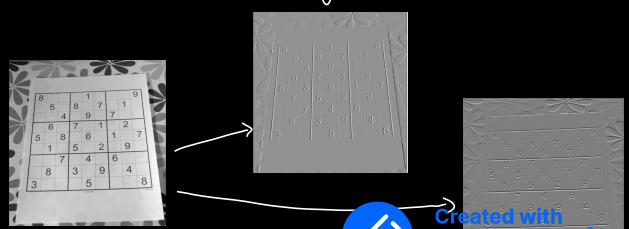
$9\times1 + 1\times1 + 4\times1 + 10\times0 + 8\times0 + 9\times0 + 2\times-1 + 4\times-1 + 10\times-1 = \boxed{-2}$

continue this process until you reach at the end (right side). once you reach at the end come one pixel down & continue the process.

The output dimention will be $(n\times n) * (f\times f) = (n-f+1)\times(n-f+1)$



→ This is how convolution operation identifies vertical edge. Practical application



1	1	1
0	0	0
-1	-1	-1

→ Horizontal edge detection filter

→ At first layer, we will use multipul filters that will detect different feature-es of an image.

→ If we apply $f \times f \times c$ filters on $n \times n \times 1$ image then the output will be $(n-f+1) \times (n-f+1) \times c$ where $n \times n \times 1$: gray scale image of size $n \times n$
$f \times f \times c$: 'c' number of filters with $f \times f$ size.

→ For RGB coloured $n \times n \times 3$ image, we will use $f \times f \times 3$ size filter. where output image will be of size $(n-f+1) \times (n-f+1) \times 1$

$f \times f \times 3$ (filter size, color channels)

single filter

if we apply $f \times f \times 3 \times c$ number of filter, the o/p image will be.
$(n-f+1) \times (n-f+1) \times c$

Padding

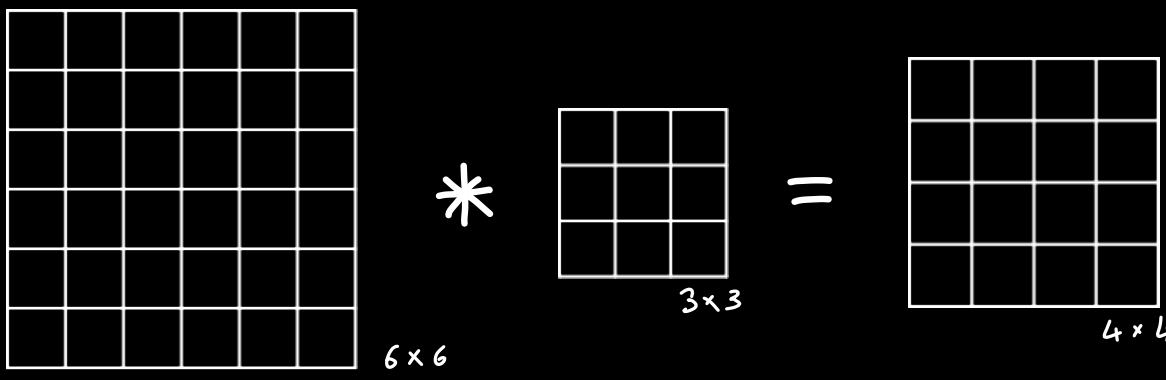
→ N number of layer around the original image is called as padding.

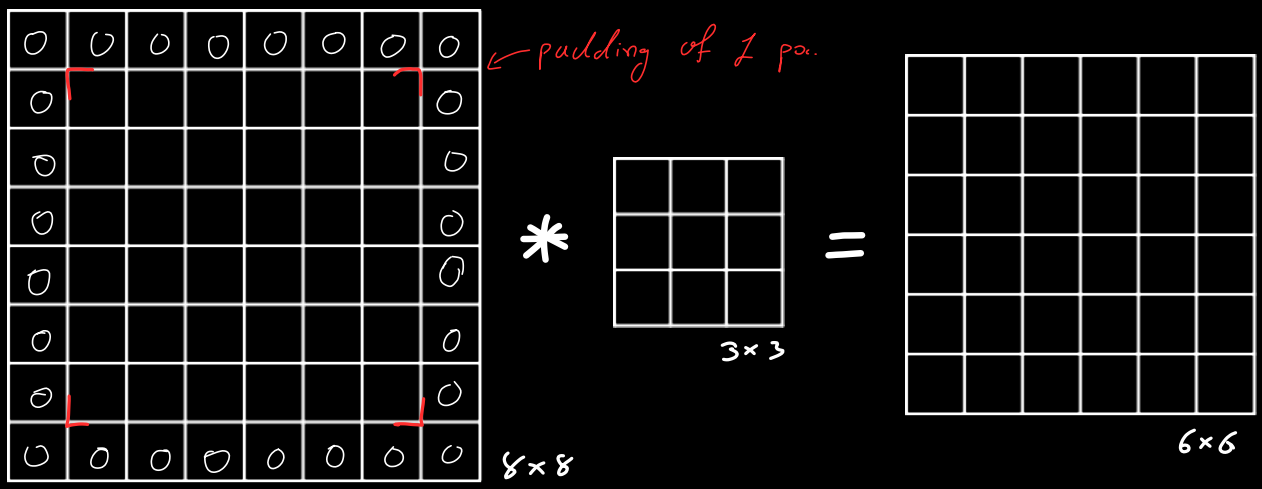
→ As we know that by performing convolution operation on an image, the output image size/shape is reduced by some amount. This problem may leads to loosing important features from image.

→ To overcome this problem, we can add 'n' number of layers around image hence the outer pixel layer of image will get focused multipul times by filter so majority features will be considered instead of ignorance.

Types of padding

1. Valid Convolution
   Performing convolution with no padding.
2. Same convolution
   after convolution operation, o/p image should have same size as i/p image.





→ As above example shows, if we apply 3×3 filter directly on original 6×6 image will give output image of 4×4 which means we lost outer layer of 1px from original image.

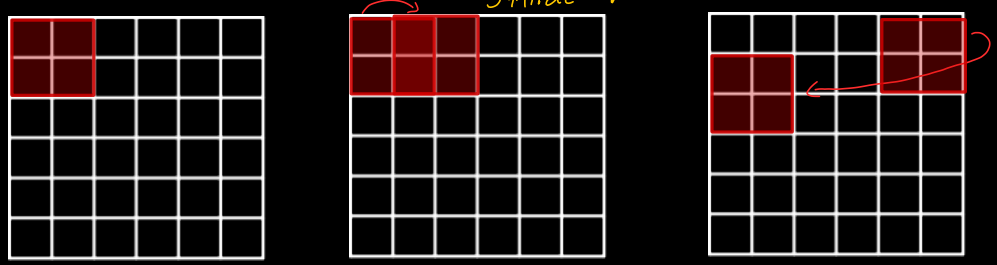
→ But in next scenario, we have added padding of 1px now our image size is 8×8 insted of 6×6.

→ By applying convolution opration with 3×3 filter on 8×8 image will return image of 6×6 similar as original one.

$$new\_image = n + 2p$$
$$= 6 + 2 \cdot 1$$
$$= 6 + 2$$
$$= \boxed{8}$$

Stride

Jump of number of pixels.



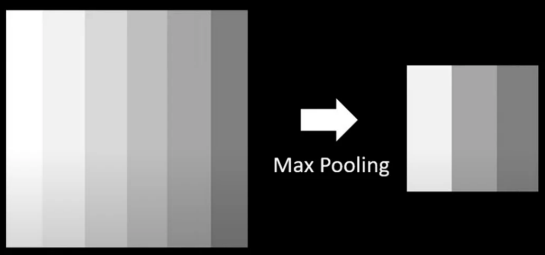
After applying strided convolution, out o/p image size will be

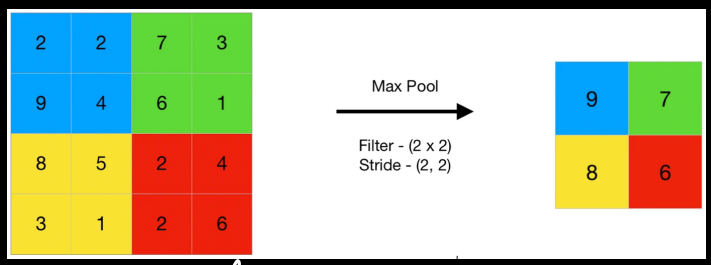
$$\text{floor}\left(\left[\frac{n-f}{s}+1\right]\right)$$ where

n : I/p dimentions.
s : Stride amount.
f : filter dimentions.

Max pooling operation / Max pooling layer

The purpose of pooling is to reduce the size on dimention of the image while preserving features in it.



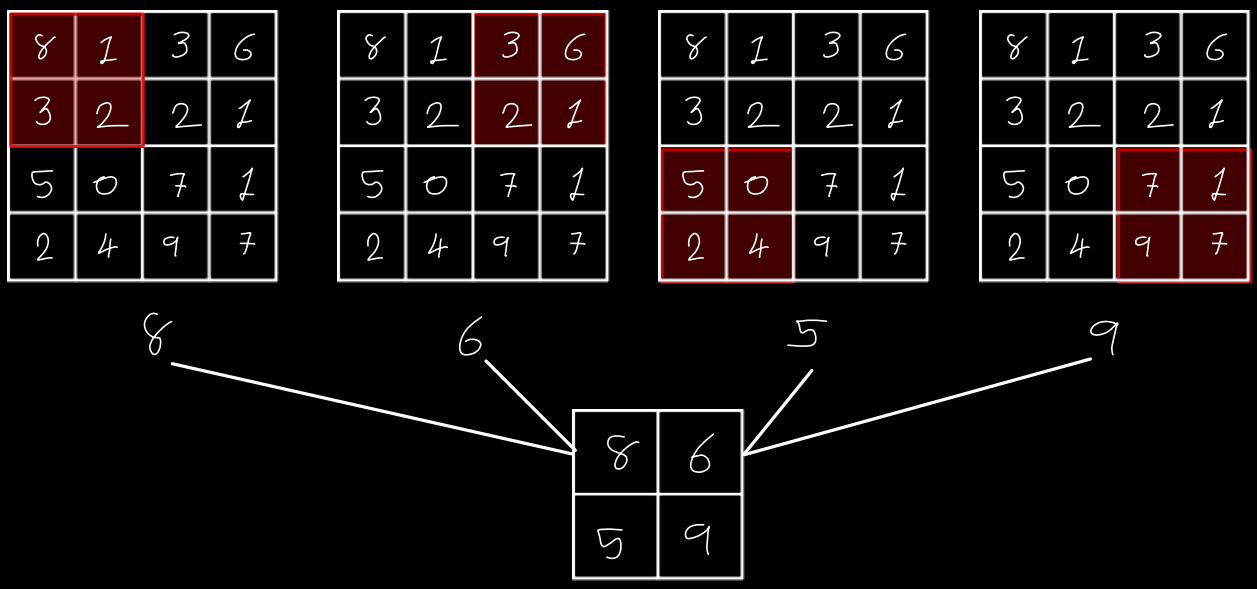


-> for this, we can take any size of filter and any amount of stride.

-> Usually stride is taken as sidelen of filter.

Eg. filter size = 2×2, Stride = 2

-> Now select the maximum value from each window.



-> The main aim for max pooling is to reduce the computing cost.

-> Max pooling always applied after the convolutional layer.

-> In earlier CNN, we uses many convolutional as well as max pooling layer.

→ It is recommended to use max pooling layer after every convolutional layer.

→ We can also skip to add max pooling layer in very big CNN, if we don't want reduce too much size of our image.

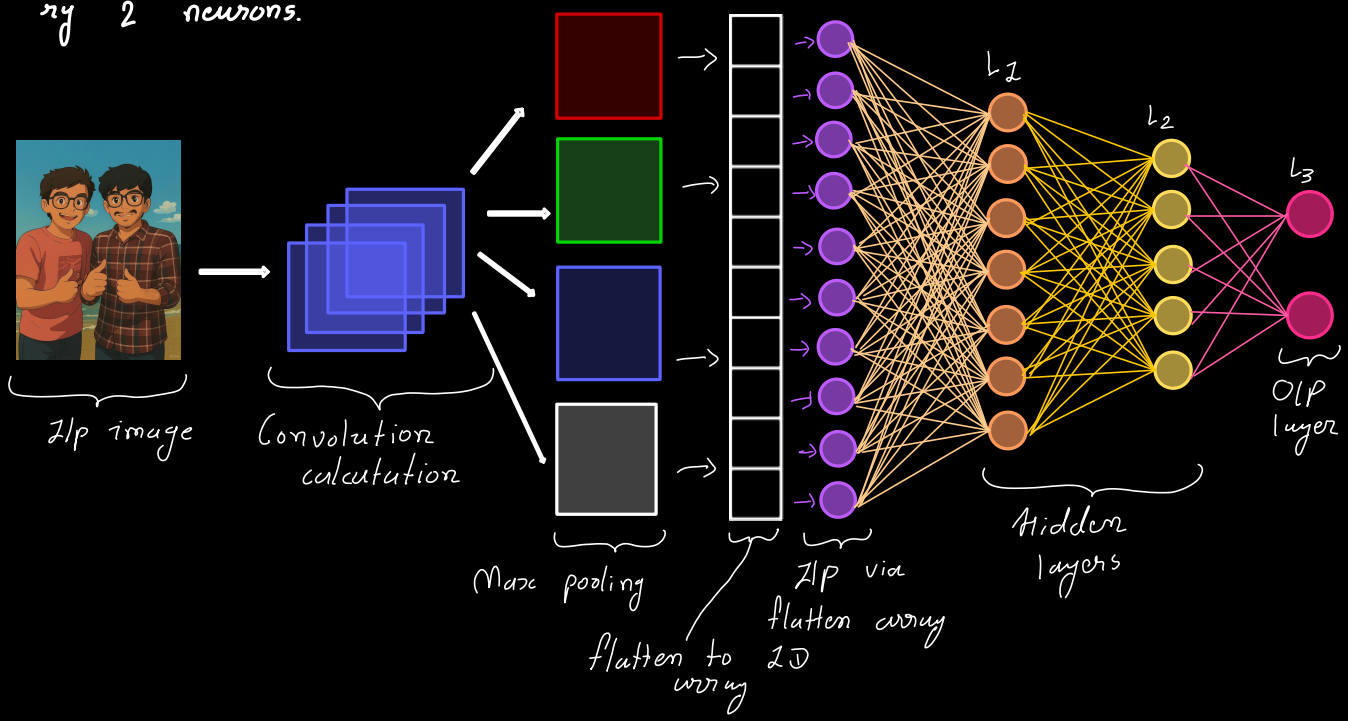
→ No parameters involved.

→ We can also use Average Pooling, where we takes average of values within filter.

→ Same number of channels in O/p as I/p.

Fully Connected Layer in CNN

→ It is the dense network of neurons & connection between every 2 neurons.



→ A dense network of neurons.

CNN Architecture

Convolutional layer

nxn

I/P image

f x f x 3

convolution operation

ReLU(+ b)

convolution operation result

A non-linear function to scale image properly. (tanh/ ReLU)

Convolutional Layer

For multipal filters.....

nxn

I/P image

f x f x 3

ReLU(+ b1)

f x f x 3

ReLU(+ b2)

f x f x 3

ReLU(+ b3)

f x f x 3

ReLU(+ b4)

In short.

nxn

I/P image

Conv

f x f x 3 x 4

(n-f+1) x (n-f+1) x 4

Example

32 × 32 × 3

Conv1
5×5×3×4
Stride (S) = 1

28×28×4

MaxPool2
2×2
S = 2

14×14×4

Conv2 3×3×4×8 S=1

12×12×8

MaxPool2
2×2
S = 2

6×6×8

For larger CNN we can skip MaxPool to prevent feature loss.

flatten array

288×1

$W_1 = 7 \times 288$

$W_2 = 5 \times 7$

$W_3 = 2 \times 5$

Forward Propogation

Prediction

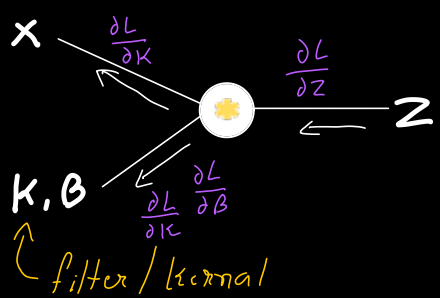
Cost - function

Back propogation

Update Parameters

Artificial Neural Network

CNN Backpropogation



$$\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} \circledast \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix} + b = \begin{bmatrix} z_{11} & z_{12} \\ z_{21} & z_{22} \end{bmatrix}$$

$$z_{11} = x_{11} \cdot k_{11} + x_{12} \cdot k_{12} + x_{21} \cdot k_{21} + x_{22} \cdot k_{22} + b$$
$$z_{12} = x_{12} \cdot k_{11} + x_{13} \cdot k_{12} + x_{22} \cdot k_{21} + x_{23} \cdot k_{22} + b$$
$$z_{21} = x_{21} \cdot k_{11} + x_{22} \cdot k_{12} + x_{31} \cdot k_{21} + x_{32} \cdot k_{22} + b$$
$$z_{22} = x_{22} \cdot k_{11} + x_{23} \cdot k_{12} + x_{32} \cdot k_{12} + x_{33} \cdot k_{22} + b$$

$$K = K - \alpha \cdot \frac{\partial L}{\partial K}$$

$$B = B - \alpha \cdot \frac{\partial L}{\partial B}$$

$$\frac{\partial L}{\partial K} = \frac{\partial L}{\partial Z} * \frac{\partial Z}{\partial K}$$

$$\frac{\partial L}{\partial K} = \begin{bmatrix} \frac{\partial L}{\partial k_{11}} & \frac{\partial L}{\partial k_{12}} \\ \frac{\partial L}{\partial k_{21}} & \frac{\partial L}{\partial k_{22}} \end{bmatrix}$$

$$\frac{\partial L}{\partial k_{mn}} = \sum \left(\frac{\partial L}{\partial z_{ij}} * \frac{\partial z_{ij}}{\partial k_{mn}} \right) \quad \begin{cases} z_{ij} \to z_{11}\ z_{12}\ z_{21}\ z_{22} \\ k_{mn} \to k_{11}\ k_{12}\ k_{21}\ k_{22} \end{cases}$$

$$\frac{\partial L}{\partial k_{11}} = \left(\frac{\partial L}{\partial z_{11}} * \frac{\partial z_{11}}{\partial k_{11}}\right) + \left(\frac{\partial L}{\partial z_{12}} * \frac{\partial z_{12}}{\partial k_{11}}\right) + \left(\frac{\partial L}{\partial z_{21}} * \frac{\partial z_{21}}{\partial k_{11}}\right) + \left(\frac{\partial L}{\partial z_{22}} * \frac{\partial z_{22}}{\partial k_{11}}\right)$$

$$\frac{\partial L}{\partial k_{12}} = \left(\frac{\partial L}{\partial z_{11}} * \frac{\partial z_{11}}{\partial k_{12}}\right) + \left(\frac{\partial L}{\partial z_{12}} * \frac{\partial z_{12}}{\partial k_{12}}\right) + \left(\frac{\partial L}{\partial z_{21}} * \frac{\partial z_{21}}{\partial k_{12}}\right) + \left(\frac{\partial L}{\partial z_{22}} * \frac{\partial z_{22}}{\partial k_{12}}\right)$$

$$\frac{\partial L}{\partial k_{21}} = \left(\frac{\partial L}{\partial z_{11}} * \frac{\partial z_{11}}{\partial k_{21}}\right) + \left(\frac{\partial L}{\partial z_{12}} * \frac{\partial z_{12}}{\partial k_{21}}\right) + \left(\frac{\partial L}{\partial z_{21}} * \frac{\partial z_{21}}{\partial k_{21}}\right) + \left(\frac{\partial L}{\partial z_{22}} * \frac{\partial z_{22}}{\partial k_{21}}\right)$$

$$\frac{\partial L}{\partial k_{22}} = \left(\frac{\partial L}{\partial z_{11}} * \frac{\partial z_{11}}{\partial k_{22}}\right) + \left(\frac{\partial L}{\partial z_{12}} * \frac{\partial z_{12}}{\partial k_{22}}\right) + \left(\frac{\partial L}{\partial z_{21}} * \frac{\partial z_{21}}{\partial k_{22}}\right) + \left(\frac{\partial L}{\partial z_{22}} * \frac{\partial z_{22}}{\partial k_{22}}\right)$$