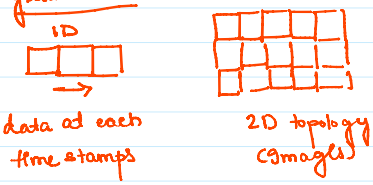
Convolutional Neural Networks (CampusX)

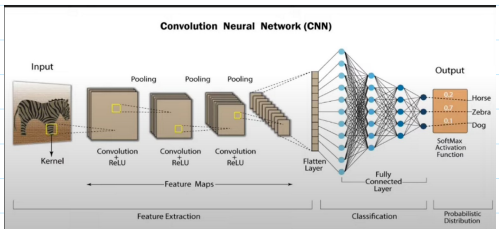
21 July 2024 11:07

What is a CNN?

Ans: Convolutional Neural Networks, also known as convnet are a special kind of neural nets for processing data that has a known grid-like topology like time-series data(1D) or images(2D).

Grid like data





ANN → uses matrix multiplication

CNN → uses convolution

↳ A CNN has 3 types of layers: Convolution layer, Pooling layer, Fully connected layer

- The CNN is inspired by how humans perceive image (working of visual cortex)

1998 → Yan LeCunn → AT&T Lab

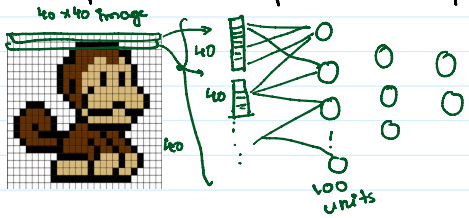
↳ Microsoft built OCR tools for handwriting recog → Currently facial recognition, self driving

Why not use ANN?

- ANN can work on image data
- But the results are not satisfactory
- There are some disadvantages as well.

① High computational cost:

- When we use ANN to process images we convert 2D → 1D
- Then pass these 1D inputs to the input layers



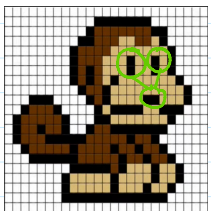
∴ For a small 40 x 40 image, we have

1600 x 100 = 1,60,000 weights at the first layer itself

② Overfitting

- As there are so many connections, it captures all minute details and hence performs well on training data.

③ Loss of important info like spatial arrangement of pixels



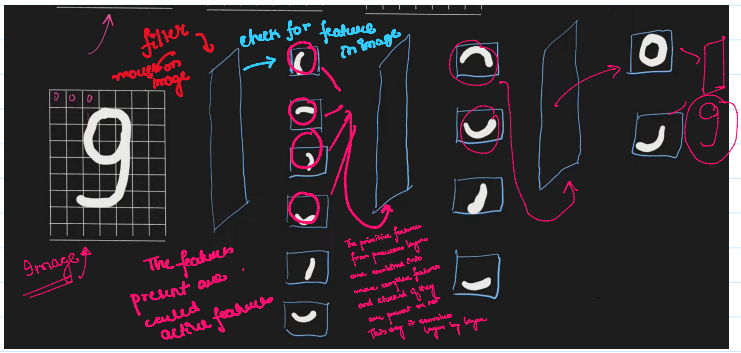
For example for the given image, the distance between eyes and nose is also an important feature and hold some meaning

But when we convert it to 1D, it becomes meaningless and the spatial info is lost.

CNN Intuition

- We convert the image into features (edges for e.g)
- Then by joining primitive features into complex features layer by layer

filters are mathematical operation that extract features



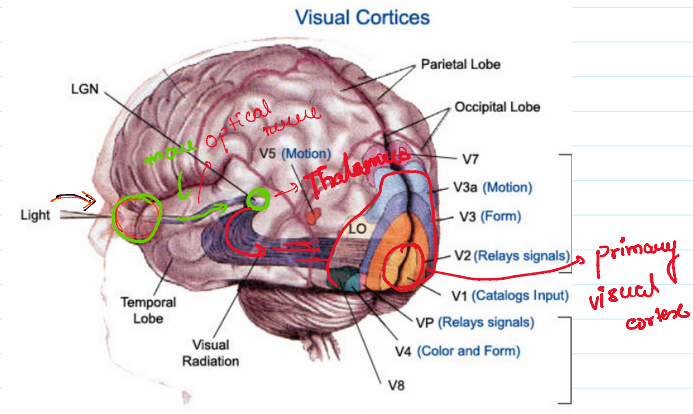
Feature extraction

Primitive features → Combine primitives → Complex features → .... → Final complete image based on all features

Applications

- Image Classification (Identify objects in image)
- Object Localization (Locate a object in image by drawing a bounding box)
- Object Detection → Facial Recognition → Image Segmentation (divide image in segments)
- Super resolution (enhance images)
- Black and white → colour
- Pose Estimation

Human Visual Cortex



The experiment by Hubble & Wiesel led to the following conclusion

– We have 2 kinds of cells

(i) Simple Cells → Orientation cell → feature detect (have a smaller receptive field)

↓

It has a preferred stimuli

(can detect edges)

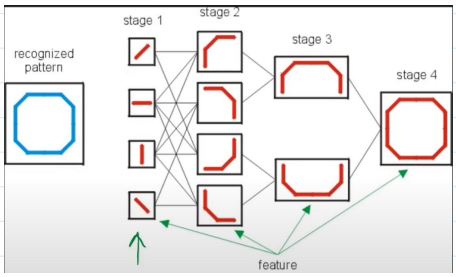
edge detection

(ii) Complex cell → have a bigger receptive field

↓

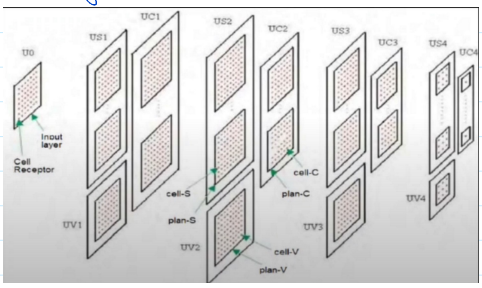
Takes edges processed by simple cell and try and build higher level / complex features.

Development

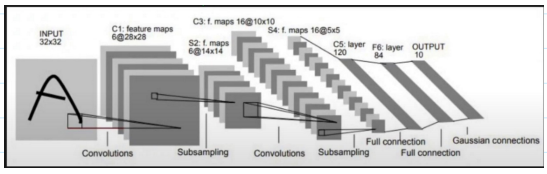


Earliest model

Neocognition → Fukushima



Yann LeCunn → Backprop → Earliest CNN



Then from 2012 a lot more CNN's were developed.

LECTURE-3: Convolution Operation

Basics of Images

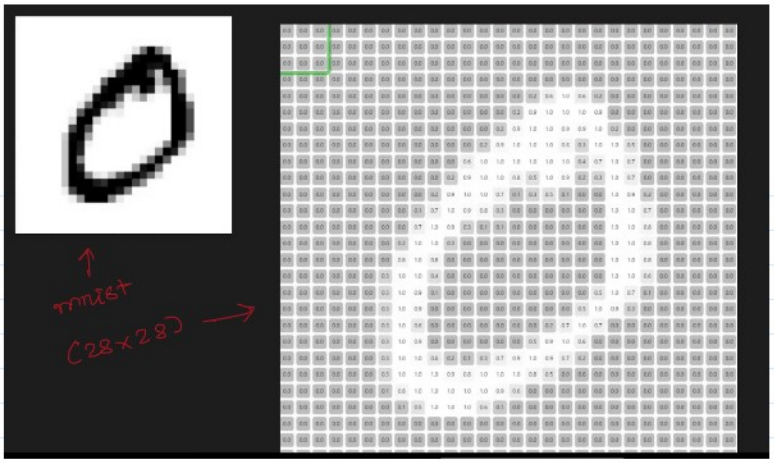
– Grey scale – Black & White → (0 – 255)

↓

– RGB – coloured Images

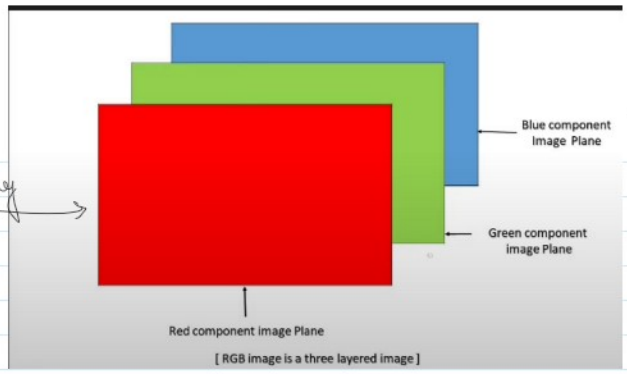
1 – black

- Greyscale - Black & White → (0 - 255)
  ↓
  0 - white 1 - black
- RGB - coloured Image



28 × 28 pixels
↓
processed as
2D numpy arrays

RGB images has 3 channels → Red, Green, Blue (primary colours)



Each channel are individually grayscale in itself

228 × 228 × 3

Q) Diff b/w RGB & Greyscale

RGB has 3 channels

Greyscale has 1 channel.

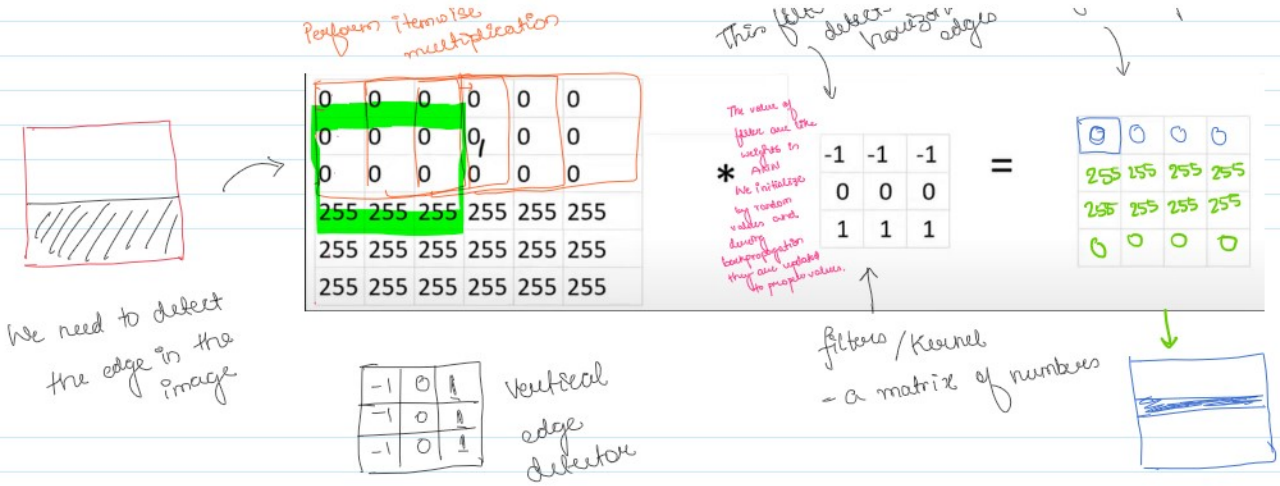
Convolution Operation

↳ Used for Edge Detection → Edges are change in intensity

Perform Element wise multiplication

This filter detects horizontal edges

feature map



Q) When we have a 6×6 image and we apply a 3×3 filter, we get a 4×4 feature map.
So what will be the size of feature map if image is 28×28 and kernel is 3×3?

Ans) $6\times6 * 3\times3 = 4\times4$

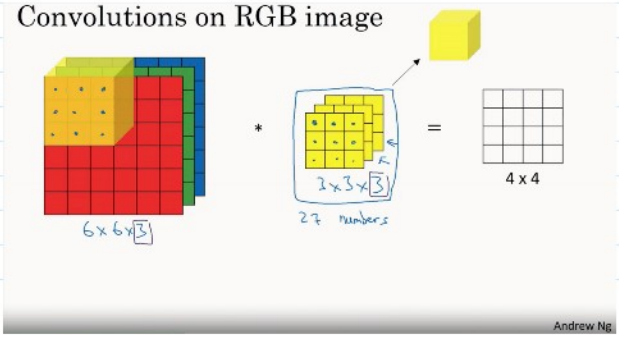
Basically → The 3×3 filter can cover the 6 rows in 4 iteration (∵ resultant row = 4)
The 3×3 filter can cover the 6 cols in 4 iteration (∵ resultant col = 4)

∴ formula: For a m×m image and n×n filter
$(m-n+1) \times (m-n+1)$

∴ for 28×28 image & 3×3 filter $m = 28, n = 3$
∴ feature map = $(28-3+1) \times (28-3+1) = 26 \times 26$

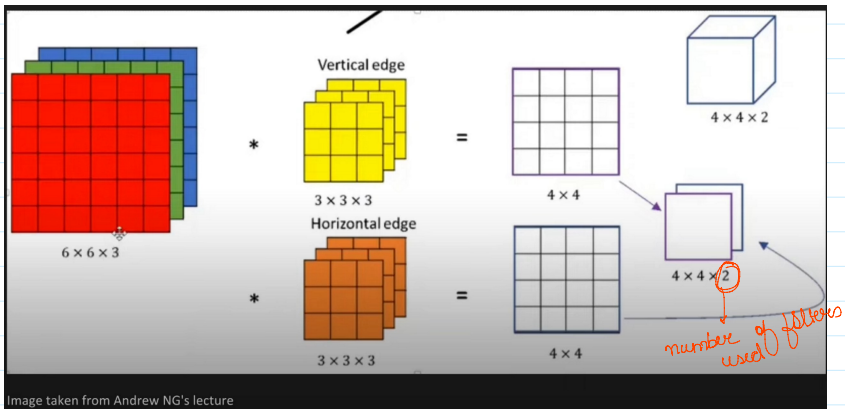
For RGB images

- The image is having 3 channels
- ∴ The filter has also one-dimension increased and have 3 channels
- The convolution operation happens over volume, cube by cube



∴ $(m \times n \times C) * (m \times n \times C) \rightarrow (m-n+1) \times (m-n+1)$
single channel feature map

Working with multiple filters



LECTURE-4: Padding and Strides

Problems with Convolution Operation

① The size of feature map is smaller than that of the image.
- So we lose information
- If we apply further convolution on feature map, it becomes very small leading to no significance

7	2	3	3	8
4	5	3	8	4
3	3	2	8	4
2	8	7	2	7
5	4	4	5	4

5×5

*

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

3×3

=

6	-9	-8
-3	-2	-3
-3	0	-2

3×3

② When the convolution operation takes place, the border pixels are involved only once whereas the centre pixels are used more times. So in the feature map the centre pixels are more dominant and have more say.

∴ We need to keep the image of the same size

m → size of image ∴ $m - n + 1 = 5$ (say)

n → size of kernel $m - 3 + 1 = 5$

$m = 7$ → We have to make the image 7×7 to obtain final 5×5

How to do that?
- Padding

Zero Padding: Add rows and columns of 0 pixels around the image.

0	0	0	0	0	0	0
0	7	2	3	3	8	0
0	4	5	3	8	4	0
0	3	3	2	8	4	0
0	2	8	7	2	7	0
0	5	4	4	5	4	0
0	0	0	0	0	0	0

The formula of size of feature map becomes

$(n + 2p - f + 1)$

$= 5 + 2(1) - 3 + 1$

$= 5 + 2 - 3 + 1$

$= 5$

$n = 5$, $f = 3$, $p = 1$

In Keras we have 2 ways of padding:
- valid → does no padding

In Keras we have 2 ways of padding
- valid → does no padding
- same → applies required padding to ensure feature map is of same shape as image

Padding demo in Keras

```
model = Sequential()

model.add(Conv2D(32,kernel_size=(3,3),padding='valid', activation='relu', input_shape=(28,28,1)))
model.add(Conv2D(32,kernel_size=(3,3),padding='valid', activation='relu'))
model.add(Conv2D(32,kernel_size=(3,3),padding='valid', activation='relu'))

model.add(Flatten())

model.add(Dense(128,activation='relu'))
model.add(Dense(10,activation='softmax'))
```

Model: "sequential_2" Input = (28, 28, 32) −2

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 26, 26, 32)	320
conv2d_5 (Conv2D)	(None, 24, 24, 32)	9248
conv2d_6 (Conv2D)	(None, 22, 22, 32)	9248
flatten_1 (Flatten)	(None, 15488)	0

Valid (no padding) size decreases

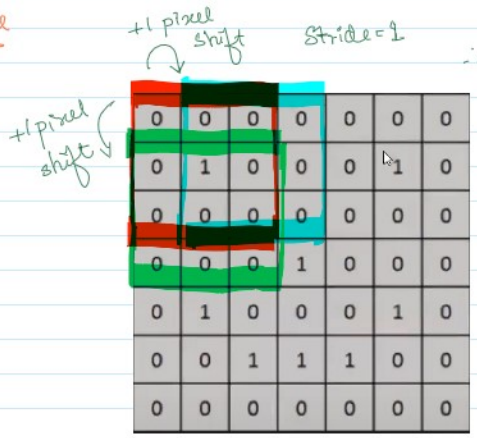
```
model = Sequential()

model.add(Conv2D(32,kernel_size=(3,3),padding='same', activation='relu', input_shape=(28,28,1)))
model.add(Conv2D(32,kernel_size=(3,3),padding='same', activation='relu'))
model.add(Conv2D(32,kernel_size=(3,3),padding='same', activation='relu'))

model.add(Flatten())

model.add(Dense(128,activation='relu'))
model.add(Dense(10,activation='softmax'))
```

Model: "sequential_3" Input = (28, 28, 32) same

Layer (type)	Output Shape	Param #
conv2d_7 (Conv2D)	(None, 28, 28, 32)	320
conv2d_8 (Conv2D)	(None, 28, 28, 32)	9248
conv2d_9 (Conv2D)	(None, 28, 28, 32)	9248
flatten_2 (Flatten)	(None, 25088)	0

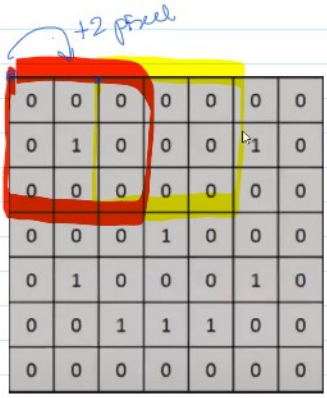
Same size remains same

Stride



∴ stride = (1,1)

We can change strides as well e.g. (2,2)



The formula for feature becomes

With padding ⇒ $\left[\frac{n+2p-f}{s}+1\right]$ without padding → $\left[\frac{n-f}{s}+1\right]$

Note
When value of Stride ≥ 1, it is called Strided Convolution

Special case
- In this case stride = 2 and kernel = 3×3

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

6x7

no pixels no convolution

- So if we continue to do this, we will see there are not enough pixels in the column
- So convolution will not be performed for that

The feature map size can be calculated as

For row $\Rightarrow \left[\frac{n-f}{s}+1\right] = \left[\frac{6-3}{2}+1\right] = (1.5)+1 = 1+1 = 2$

whenever decimal take floor

For col $\Rightarrow \left[\frac{n-f}{s}+1\right] = \left[\frac{7-3}{2}+1\right] = 2+1 = 3$

$\therefore$ Feature map 2x3

Q) Why strides are required?

Ans: 1) We only require high level features

If stride value is large, it will skip through some info and capture high level features only.

2) Computation cost: If dataset is large, it helps save time and train faster.

Using stride in Keras

```
model = Sequential()

model.add(Conv2D(32,kernel_size=(3,3),padding='same',strides=(2,2), activation='relu', input_shape=(28,28,1)))
model.add(Conv2D(32,kernel_size=(3,3),padding='same',strides=(2,2), activation='relu'))
model.add(Conv2D(32,kernel_size=(3,3),padding='same',strides=(2,2), activation='relu'))

model.add(Flatten())

model.add(Dense(128,activation='relu'))
model.add(Dense(10,activation='softmax'))
```

row ↓ col → it can be different as well

Model Summary

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 14, 14, 32)        320

 conv2d_1 (Conv2D)           (None, 7, 7, 32)          9248

 conv2d_2 (Conv2D)           (None, 4, 4, 32)          9248

 flatten (Flatten)           (None, 512)               0
```

1st conv

Input = 28x28

f = 3x3

p = 1

s = 2

$\rightarrow \frac{n+2p-f}{s}+1$

$= \frac{28+2-3}{2}+1$

$= 13.5+1 = 13+1 = 14$

2nd conv

Input = 14

f = 3

s = 2

p = 1

$\rightarrow$ Output $= \frac{14+2-3}{2}+1$

$= \frac{13}{2}+1$

$= 6.5+1 = 7$

3rd conv

Input = 7

f = 3

s = 2

p = 1

Output $\frac{7+2-3}{2}+1$

$= \frac{6}{2}+1 = 3+1 = 4$

LECTURE-5: Pooling Layers

The problem with Convolution → Memory issues
→ Translation Variance

image → filters → feature map

1) Memory issues

Suppose we have an image of 228 × 228 × 3
and we apply 100 filters (3×3)

The resultant will be
(226 × 226 × 100)

Now suppose we use a 32-bit floating memory to store the features

∴ It will take 226 × 226 × 100 × 32 = 1.9 MB in memory

this is for 1 batch → suppose 100 batches

∴ 1.6 GB approx storage in memory required

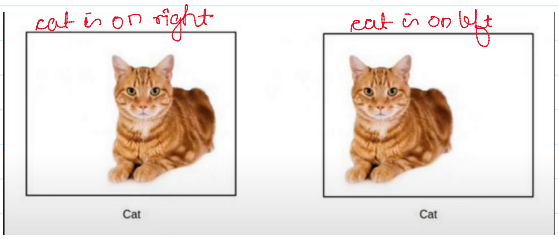
∴ Size has to be reduced

- One way to reduce size is stride.
- But pooling is preferred because it reduces size as well as solves the other problem of Translational Variance

2) Translation Variance

- When we do convolution operation, the features extracted are location dependent
- When we are doing image classification task, we are concerned only about features not their location

For example



In this case the features for both images will be detected at different locations

∴ When predicting, presence of 'ear' in left may lead to different output than 'ear' in right

We don't want this. We want irrespective of location we get 'cat'.

Pooling achieves [translation invariance]

↳ down samples over feature map to achieve this

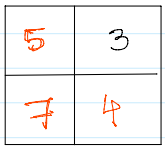
image × filters = feature map → apply relu for non-linearity → non-linear feature map → pooling → final feature map

We need 3 things

We keep the most dominant

non-linear

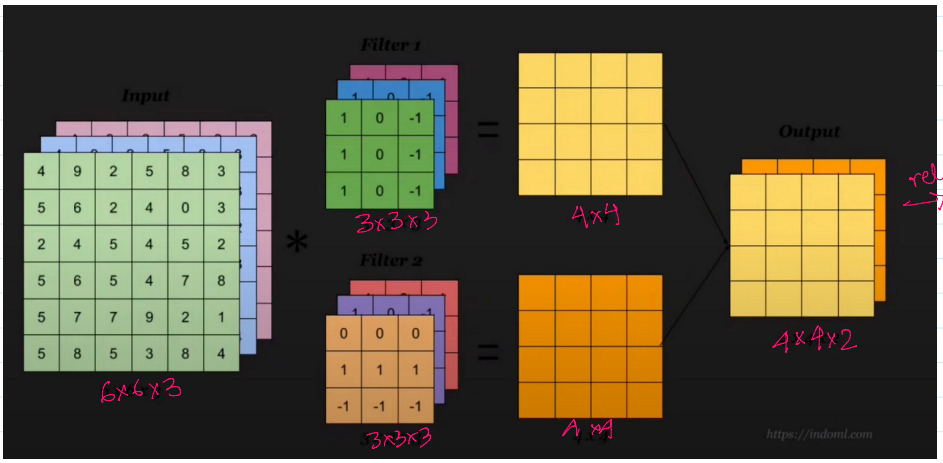
We need 3 things
size = (2,2)
stride = 2
type = max
these can be changed



pooling result

We keep the most dominant feature from a region and supress the smaller ones leading to downsampling and translational invariance

Pooling on Volumes



relu → Pooling is applied on each layer → 2x2x2

Demo in Keras

```
import tensorflow
from tensorflow import keras
from keras.layers import Dense,Conv2D,Flatten,MaxPooling2D
from keras import Sequential
from keras.datasets import mnist

(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
11501568/11490434 [==============================] - 0s 0us/step
```

```
model = Sequential()

model.add(Conv2D(32,kernel_size=(3,3),padding='valid', activation='relu', input_shape=(28,28,1)))
model.add(MaxPooling2D(pool_size=(2, 2), strides=2, padding='valid'))
model.add(Conv2D(32,kernel_size=(3,3),padding='valid', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=2, padding='valid'))

model.add(Flatten())

model.add(Dense(128,activation='relu'))
model.add(Dense(10,activation='softmax'))
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 11, 11, 32)        9248

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 32)         0
```

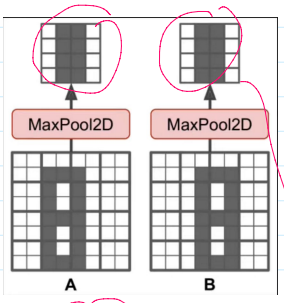
Input → (28,28,32)
↓ conv
(26,26,32)
↓ pooling
(13,13,32) → Conv → (11,11,32) → Pooling → (5,5,32)

Advantages of Pooling

1) reduced size
2) Translation Invariance

1) reduced size

2) Translation Invariance



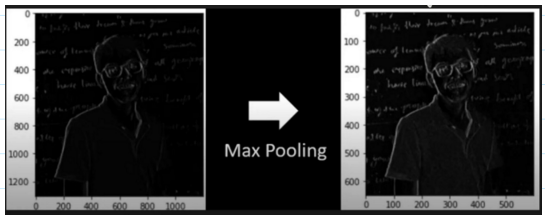
The existence of feature matters not the location

↳ same feature map

location varies

3) Enhanced features (only in case of maxpooling)

↳ because it acts on a smaller receptive field and picks up the dominant features.



4) No need of training

Pooling is just an aggregate operation

We just need → size, stride and type

during backpropagation pooling is unaffected therefore faster

Types of Pooling

1) Max pooling 2) Average Pooling 3) Global Pooling

3	1	1	3
2	5	0	2
1	4	2	1
4	7	2	4

Average

Take average of the 4 features

Global Max

Global Average

Global → Converts the entire feature map into a scalar value (1x1)

↳ Max → Max of all features

↳ Average → Average of all features

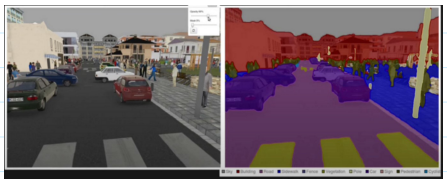
↳ Average → Average of all features

They are used in the end instead of Flatten to prevent overfitting

Disadvantages of Pooling

↳ Cannot be used for tasks where location of features matter

Ex: Image Segmentation Tasks.

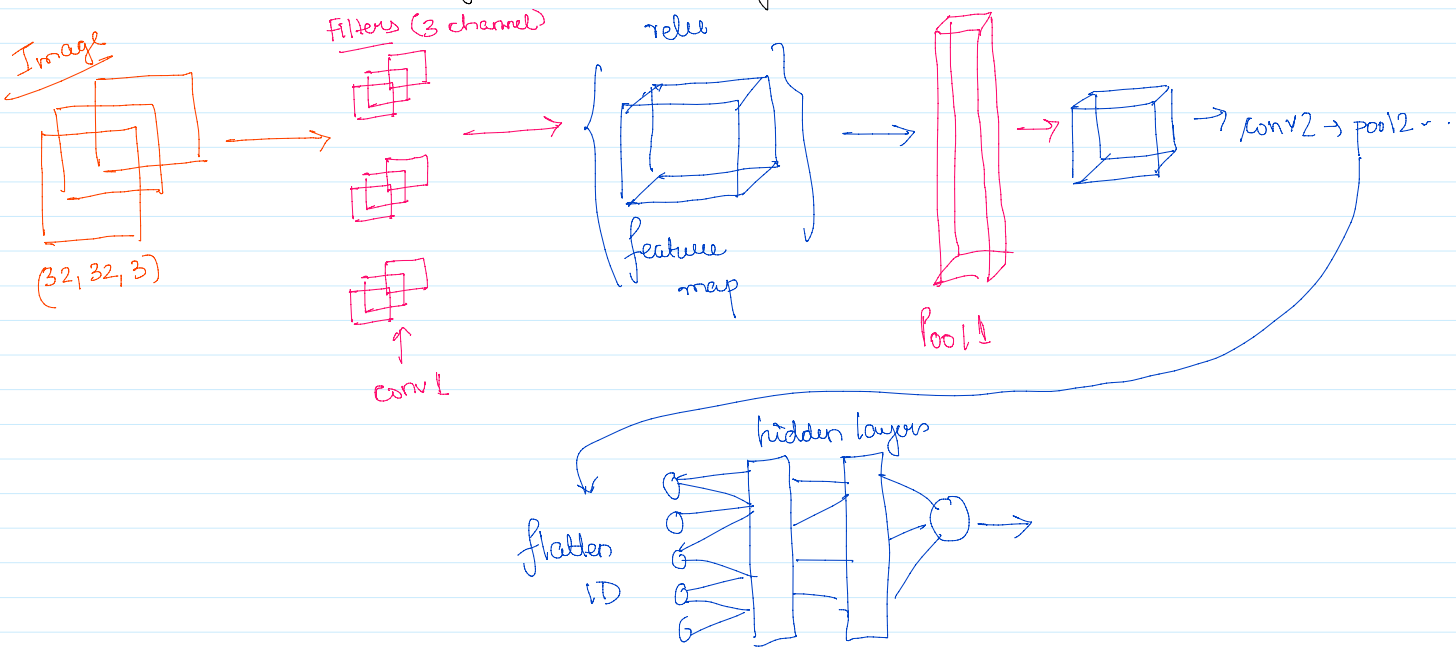


↳ Loss of lot of information

LECTURE-6: LeNET Architecture

CNN Architecture

1) Convolution 2) Padding / Stride 3) Pooling



ImageNet

↳ a competition featuring state-of-the-art CNN models

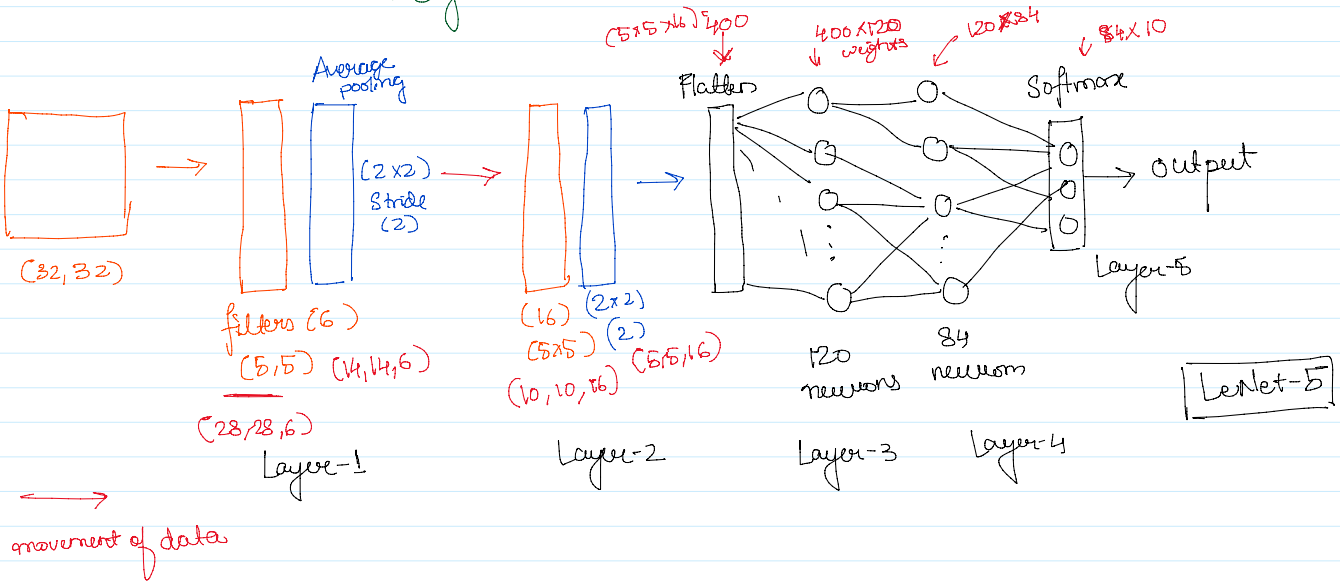
1) LeNet → Yann Le Cunn
2) AlexNet
3) GoogleNet
4) VggNet
5) ResNet
6) Inception

3) GoogleNet

6) Inception

Le-Net

- Came in 1998
- by Yann LeCunn → developed by him for US Navy Postal code identification
- also called LeNet-5 because it has 5 layers.

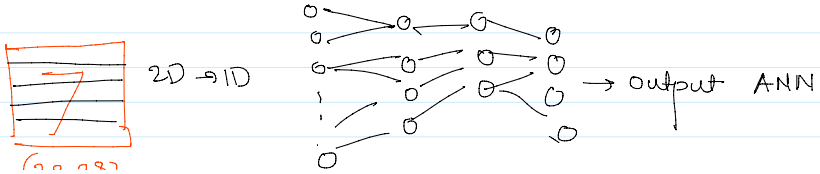


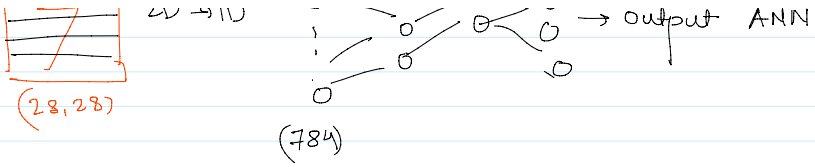
Le-Net Code

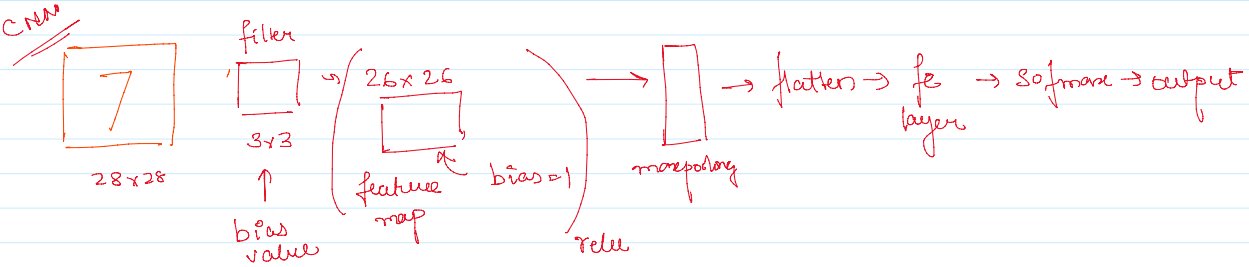
```
model = Sequential()

model.add(Conv2D(6,kernel_size=(5,5),padding='valid', activation='tanh', input_shape=(32,32,1)))
model.add(AveragePooling2D(pool_size=(2, 2), strides=2, padding='valid'))

model.add(Conv2D(16,kernel_size=(5,5),padding='valid', activation='tanh'))
model.add(AveragePooling2D(pool_size=(2, 2), strides=2, padding='valid'))

model.add(Flatten())

model.add(Dense(120, activation='tanh'))
model.add(Dense(84, activation='tanh'))
model.add(Dense(10, activation='softmax'))
```

LECTURE-7: CNN v/s ANN

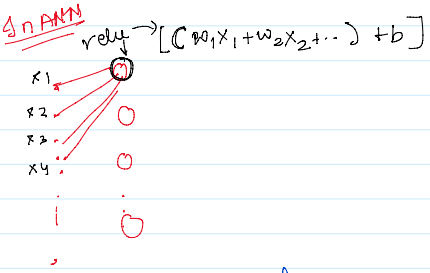
CNN V/S ANN



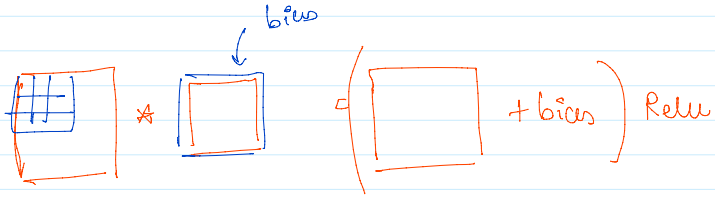




The CNN and ANN are somewhat similar



Here we train weights

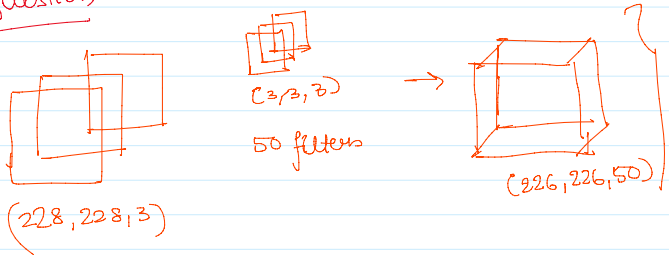


$relu\left[(x_1w_1 + x_2w_2 + \_ x_9w_9) + bias\right]$

Here we train filter values

a new filter added will have its own bias
It is like adding a new node in ANN hidden layer

Question



How many learnable parameters should we have?

The learnable params are kernel weights & bias
So we have 3x3x3 kernels and 50 of them
∴ For 1 kernel = 27 learnable params + 1 bias
∴ = 28 learnable params
∴ Total = 50 x 28 = 1400 learnable params

Followup

The image size is now changed to (1080,1080,3). Now, what will be the learnable parameters?

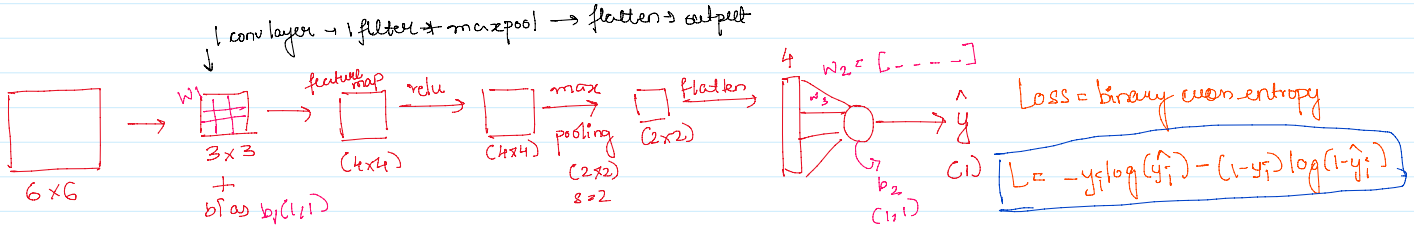
The learnable parameters will still be same because the kernel's and biases are same. They are independent of the dimension of the image.
(That is one advantage of CNN over ANN)

But in ANN, as the input size changes the number of weights changes

But in ANN, as the input size changes the number of weights changes

Increase in the number of weights leads to overfitting.

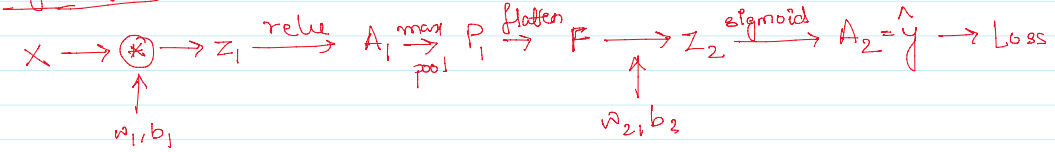
LECTURE-11: Backpropagation



Trainable Parameters: → conv filters → $w_1 = (3,3)$ 9+1=10, $b_1 = (1,1)$

→ fc layer weights → $w_2 = (1\times4) = 4$, $b_2 = (1,1) = 1$ = 15 trainable params

Logical flow



Forward Propagation

$$\begin{cases} Z_1 = \text{conv}(X, w_1) + b_1 & F = \text{flatten}(P_1) \\ A_1 = \text{relu}(Z_1) & Z_2 = w_2 F + b_2 \\ P_1 = \text{maxpool}(A_1) & A_2 = \sigma(Z_2) \end{cases}$$

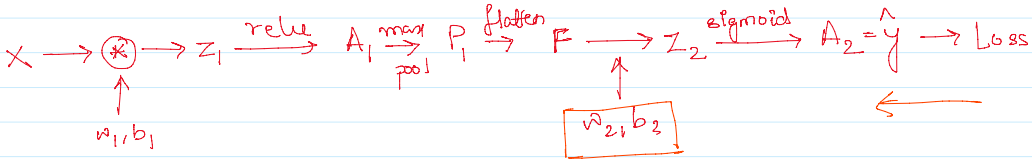
"We have to apply Gradient Descent algorithm on trainable parameters to find weights that minimize Loss.

$$w_1 = w_1 - \eta \frac{\partial L}{\partial w_1} \quad \text{(matrix)} \qquad w_2 = w_2 - \eta \frac{\partial L}{\partial w_2}$$

$$b_1 = b_1 - \eta \frac{\partial L}{\partial b_1} \qquad b_2 = b_2 - \eta \frac{\partial L}{\partial b_2}$$

We just have to find the derivatives to be able to apply gradient descent

Let's imagine the CNN and ANN part separately.



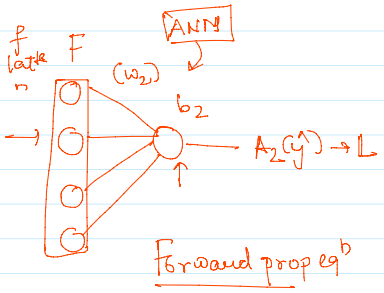
$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial A_2} \times \frac{\partial A_2}{\partial z_2} \times \frac{\partial z_2}{\partial w_2} \quad \text{(Chain rule)}$$

$$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial A_2} \times \frac{\partial A_2}{\partial z_2} \times \frac{\partial z_2}{\partial b_2}$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial A_2} \times \frac{\partial A_2}{\partial z_2} \times \frac{\partial z_2}{\partial F} \times \frac{\partial F}{\partial P_1} \times \frac{\partial P_1}{\partial A_1} \times \frac{\partial A_1}{\partial z_1} \times \frac{\partial z_1}{\partial w_1}$$

$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial A_2} \times \frac{\partial A_2}{\partial z_2} \times \frac{\partial z_2}{\partial F} \times \frac{\partial F}{\partial P_1} \times \frac{\partial P_1}{\partial A_1} \times \frac{\partial A_1}{\partial z_1} \times \frac{\partial z_1}{\partial b_1}$$

Backprop
- Convolution
- → Flatten
- → Max Pooling



Forward prop eqb

$$Z_2 = W_2 F + b_2$$

$$A_2 = \sigma(Z_2)$$

$$\frac{\partial A_2}{\partial Z_2} = \sigma(Z_2)[1-\sigma(Z_2)] = a_2[1-a_2]$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial A_2} \times \frac{\partial A_2}{\partial z_2} \times \frac{\partial z_2}{\partial w_2}$$

$$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial A_2} \times \frac{\partial A_2}{\partial z_2} \times \frac{\partial z_2}{\partial b_2}$$

assume for single image

$$\frac{\partial L}{\partial a_2} = \frac{\partial}{\partial a_2}\left[-y_i \log(a_2) - (1-y_i)\log(1-a_2)\right]$$

$$= -\frac{y_i}{a_2} + \frac{1-y_i}{(1-a_2)} = \frac{-y_i(1-a_2)+(1-y_i)a_2}{a_2(1-a_2)}$$

$$\frac{\partial L}{\partial a_2} = \frac{-y_i + y_i a_2 + a_2 - a_2 y_i}{a_2(1-a_2)} = \frac{(a_2 - y_i)}{a_2(1-a_2)}$$

$$\boxed{\frac{\partial z_2}{\partial w_2} = F} \qquad \boxed{\frac{\partial z_2}{\partial b_2} = 1}$$

$$\frac{\partial L}{\partial w_2} = \frac{a_2 - y_i}{a_2(1-a_2)} \times a_2(1-a_2) \times F = (a_2 - y_i)F = (A_2 - Y)F^T$$

matrix form

$(1,1) - (1,n)$ $\quad (n \times 1)$ $\quad (1,4) - (1,4)$ $\quad (1,1)$

$$\frac{\partial L}{\partial b_2} = \frac{a_2 - y_i}{a_2(1-a_2)} \times a_2(1-a_2) \times 1 = a_2 - y_i = (A_2 - Y)$$

$$\boxed{\frac{\partial L}{\partial w_2}_{(1,4)} = (A_2 - Y)F^T \qquad \frac{\partial L}{\partial b_2} = (A_2 - Y)}$$

If we are doing batch prediction

Suppose we have m images in a batch

∴ F will be (4,m) A_2 will be (1,m) L will be (1,m)
[1 prediction for m images]

Working on m images

$\frac{\partial L}{\partial w_2} = (A_2 - Y) F^T$
(1,m) (1,m)
(1,m) $(4,m)^T = (m,4)$
(1,4)

We will also divide by $\frac{1}{m}$ as its batch

So the derivative of loss will be of same shape as w_2 i.e constant (1,4).

Backpropagation in convolution part (till flatten)

$$\frac{\partial L}{\partial w_1} = \boxed{\frac{\partial L}{\partial A_2} \times \frac{\partial A_2}{\partial Z_2}} \times \frac{\partial Z_2}{\partial F} \times \frac{\partial F}{\partial P_1} \times \frac{\partial P_1}{\partial A_1} \times \frac{\partial A_1}{\partial Z_1} \times \frac{\partial Z_1}{\partial w_1}$$
5

$$\frac{\partial L}{\partial b_1} = \boxed{\frac{\partial L}{\partial A_2} \times \frac{\partial A_2}{\partial Z_2}} \times \frac{\partial Z_2}{\partial F} \times \frac{\partial F}{\partial P_1} \times \frac{\partial P_1}{\partial A_1} \times \frac{\partial A_1}{\partial Z_1} \times \frac{\partial Z_1}{\partial b_1}$$
1 2 3 4 6

We already have from previous ANN backprop $(A_2 - Y)$

reshape (P_1-shape)

$\frac{\partial Z_2}{\partial F} = \frac{\partial}{\partial F}(w_2 F + b)$
$= w_2$
↑ matrix

$\frac{\partial F}{\partial P_1}$ → This is on flatten layer
but there are no trainable parameters
= reshape (P_1-shape)

→ So in such cases we do not find derivatives we just reverse the operation of forward propagation

While forward pass: P_1 (2,2) → F (4,1)
backward: F → P_1 (4,1) (2,2)

$$\frac{\partial L}{\partial w_1} = \boxed{\frac{\partial L}{\partial A_2} \times \frac{\partial A_2}{\partial Z_2} \times \frac{\partial Z_2}{\partial F} \times \frac{\partial F}{\partial P_1}} \times \frac{\partial P_1}{\partial A_1} \times \frac{\partial A_1}{\partial Z_1} \times \frac{\partial Z_1}{\partial w_1}$$

$(A_2 - Y) w_2$ · reshape (P_1-shape) → basically a reshape operation of the value $\boxed{(A_2 - Y) w_2}$

$$\frac{\partial L}{\partial b_1} = \boxed{\frac{\partial L}{\partial A_2} \times \frac{\partial A_2}{\partial Z_2} \times \frac{\partial Z_2}{\partial F} \times \frac{\partial F}{\partial P_1}} \times \frac{\partial P_1}{\partial A_1} \times \frac{\partial A_1}{\partial Z_1} \times \frac{\partial Z_1}{\partial b_1}$$

- backpropagation

Now backpropagation on maxpooling → there are no trainable parameters
∴ Reverse max pooling operation

$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial A_2} \times \frac{\partial A_2}{\partial Z_2} \times \frac{\partial Z_2}{\partial P} \times \frac{\partial P}{\partial P_1} \times \frac{\partial P_1}{\partial A_1} \times \frac{\partial A_1}{\partial Z_1} \times \frac{\partial Z_1}{\partial b_1}$$

The values we calculated here is $\frac{\partial L}{\partial P_1}$ → now when we calculate $\frac{\partial P_1}{\partial A_1}$ we get the entire value as $\frac{\partial L}{\partial A_1}$

Now assume during forward pass

A_1

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} \quad \begin{bmatrix} 9 & 10 \\ 11 & 12 \end{bmatrix} \begin{bmatrix} 13 & 14 \\ 15 & 16 \end{bmatrix} \xrightarrow{\text{max pool}} \begin{bmatrix} 4 & 8 \\ 12 & 16 \end{bmatrix}$$

So basically while propagating the values that are not maximum in their window, they were not propagated forward → hence they have no role in the value of $\hat{y}$ and hence L → We are only concerned about the max values

So when we calculated $\frac{\partial L}{\partial P_1}$ it is a 2×2 matrix

A_1 $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} \quad \begin{bmatrix} 9 & 10 \\ 11 & 12 \end{bmatrix} \begin{bmatrix} 13 & 14 \\ 15 & 16 \end{bmatrix} \xrightarrow{\text{max pool}} \begin{bmatrix} 4 & 8 \\ 12 & 16 \end{bmatrix}$

Now to find $\frac{\partial L}{\partial A_1}$ it has to be converted to 4×4

Suppose $\frac{\partial L}{\partial P_1} = \begin{bmatrix} x_1 & x_2 \\ x_3 & x_4 \end{bmatrix}_{2\times 2}$ → 4×4

$\frac{\partial L}{\partial A_1} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & x_1 & 0 & x_2 \\ 0 & 0 & 0 & 0 \\ 0 & x_3 & 0 & x_4 \end{bmatrix}$ → Place the numbers/errors in $\frac{\partial L}{\partial P_1}$ in places where the max occurred before max pooling

$$\therefore \frac{\partial L}{\partial A_1} = \begin{cases} \frac{\partial L}{\partial P_{1xy}}, & \text{if } A_{mn} \text{ is the max element} \\ 0, & \text{otherwise} \end{cases}$$

Now we need to find $\frac{\partial A_1}{\partial Z_1}$

$$\frac{\partial A_1}{\partial Z_1} = \begin{cases} 1, & \text{if } Z_{1xy} > 0 \\ 0, & \text{if } Z_{1xy} < 0 \end{cases}$$

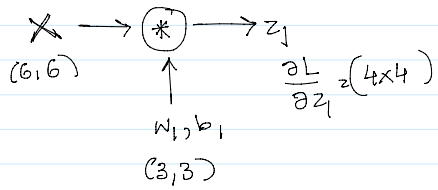
differentiation of relu

$$\therefore \frac{\partial L}{\partial z_1} = \frac{\partial L}{\partial A_1} \cdot \frac{\partial A_1}{\partial z_1}$$

Now we are left with $\frac{\partial z_1}{\partial w_1}$ and $\frac{\partial z_1}{\partial b_1}$ backpropagation on convolution (we have trainable parameters)

Backprop on Convolution

$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial z_1} \times \frac{\partial z_1}{\partial b_1}$$



For simplicity assume X is of size (3,3) & $w_1, b_1 = (2,2)$

$$\therefore z_1 = (2,2)$$

and $\frac{\partial L}{\partial z_1} = (2,2)$

$$X_1 = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} \quad W_1 = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$$

$$Z_1 = \begin{bmatrix} z_{11} & z_{12} \\ z_{21} & z_{22} \end{bmatrix} \quad \frac{\partial L}{\partial z_1} = \begin{bmatrix} \frac{\partial L}{\partial z_{11}} & \frac{\partial L}{\partial z_{12}} \\ \frac{\partial L}{\partial z_{21}} & \frac{\partial L}{\partial z_{22}} \end{bmatrix}$$

Now after convolution

$$z_{11} = x_{11}w_{11} + x_{12}w_{12} + x_{21}w_{21} + x_{22}w_{22} + b_1$$

$$z_{12} = x_{12}w_{11} + x_{13}w_{12} + x_{22}w_{21} + x_{23}w_{22} + b_1$$

$$z_{21} = x_{21}w_{11} + x_{22}w_{12} + x_{31}w_{21} + x_{32}w_{22} + b_1$$

$$z_{22} = x_{22}w_{11} + x_{23}w_{12} + x_{32}w_{21} + x_{33}w_{22} + b_1$$

$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial z_1} \times \frac{\partial z_1}{\partial b_1} = \frac{\partial L}{\partial z_{11}} \times \frac{\partial z_{11}}{\partial b_1} + \frac{\partial L}{\partial z_{12}} \times \frac{\partial z_{12}}{\partial b_1} + \frac{\partial L}{\partial z_{21}} \times \frac{\partial z_{21}}{\partial b_1} + \frac{\partial L}{\partial z_{22}} \times \frac{\partial z_{22}}{\partial b_1}$$

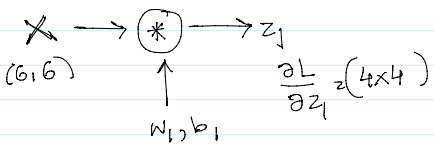
($\frac{\partial z_{11}}{\partial b_1} = 1$, $\frac{\partial z_{12}}{\partial b_1} = 1$, $\frac{\partial z_{21}}{\partial b_1} = 1$, $\frac{\partial z_{22}}{\partial b_1} = 1$)

$$= \frac{\partial L}{\partial z_{11}} + \frac{\partial L}{\partial z_{12}} + \frac{\partial L}{\partial z_{21}} + \frac{\partial L}{\partial z_{22}} = \text{sum}\left(\frac{\partial L}{\partial z_1}\right)$$

$$\therefore \boxed{\frac{\partial L}{\partial b_1} = \text{sum}\left(\frac{\partial L}{\partial z_1}\right)} \rightarrow \text{scalar}$$

$$\begin{bmatrix} x_{12} & x_{13} \end{bmatrix} \quad W_1 = \begin{bmatrix} w_{11} & w_{12} \end{bmatrix}$$

$\frac{\partial L}{\partial b_1}$ $\left(\frac{\partial L}{\partial b_1}\right)$



$\frac{\partial L}{\partial Z_1} = (4\times4)$

$$X_1 = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} \qquad W_1 = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$$

Convolution result

$$z_{11} = x_{11}w_{11} + x_{12}w_{12} + x_{21}w_{21} + x_{22}w_{22} + b_1$$
$$z_{12} = x_{12}w_{11} + x_{13}w_{12} + x_{22}w_{21} + x_{23}w_{22} + b_1$$
$$z_{21} = x_{21}w_{11} + x_{22}w_{12} + x_{31}w_{21} + x_{32}w_{22} + b_1$$
$$z_{22} = x_{22}w_{11} + x_{23}w_{12} + x_{32}w_{21} + x_{33}w_{22} + b_1$$

$$\frac{\partial L}{\partial W_1} = \begin{bmatrix} \frac{\partial L}{\partial w_{11}} & \frac{\partial L}{\partial w_{12}} \\ \frac{\partial L}{\partial w_{21}} & \frac{\partial L}{\partial w_{22}} \end{bmatrix}$$

$$\frac{\partial L}{\partial Z_1} = \begin{bmatrix} \frac{\partial L}{\partial z_{11}} & \frac{\partial L}{\partial z_{12}} \\ \frac{\partial L}{\partial z_{21}} & \frac{\partial L}{\partial z_{22}} \end{bmatrix}$$

$$\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial Z_1} \times \frac{\partial Z_1}{\partial W_1}$$

We have expanded for each z because w_{11} has contribution in all of them

$$\frac{\partial L}{\partial w_{11}} = \frac{\partial L}{\partial z_{11}} \times \frac{\partial z_{11}}{\partial w_{11}} + \frac{\partial L}{\partial z_{12}} \times \frac{\partial z_{12}}{\partial w_{11}} + \frac{\partial L}{\partial z_{21}} \times \frac{\partial z_{21}}{\partial w_{11}} + \frac{\partial L}{\partial z_{22}} \times \frac{\partial z_{22}}{\partial w_{11}}$$

$$\frac{\partial L}{\partial w_{12}} = \frac{\partial L}{\partial z_{11}} \times \frac{\partial z_{11}}{\partial w_{12}} + \frac{\partial L}{\partial z_{12}} \times \frac{\partial z_{12}}{\partial w_{12}} + \frac{\partial L}{\partial z_{21}} \times \frac{\partial z_{21}}{\partial w_{12}} + \frac{\partial L}{\partial z_{22}} \times \frac{\partial z_{22}}{\partial w_{12}}$$

$$\frac{\partial L}{\partial w_{21}} = \frac{\partial L}{\partial z_{11}} \times \frac{\partial z_{11}}{\partial w_{21}} + \frac{\partial L}{\partial z_{12}} \times \frac{\partial z_{12}}{\partial w_{21}} + \frac{\partial L}{\partial z_{21}} \times \frac{\partial z_{21}}{\partial w_{21}} + \frac{\partial L}{\partial z_{22}} \times \frac{\partial z_{22}}{\partial w_{21}}$$

$$\frac{\partial L}{\partial w_{22}} = \frac{\partial L}{\partial z_{11}} \times \frac{\partial z_{11}}{\partial w_{22}} + \frac{\partial L}{\partial z_{12}} \times \frac{\partial z_{12}}{\partial w_{22}} + \frac{\partial L}{\partial z_{21}} \times \frac{\partial z_{21}}{\partial w_{22}} + \frac{\partial L}{\partial z_{22}} \times \frac{\partial z_{22}}{\partial w_{22}}$$

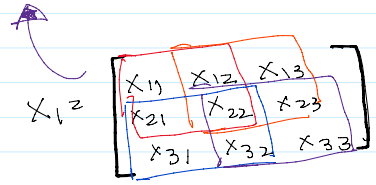
$$\frac{\partial L}{\partial w_{11}} = \frac{\partial L}{\partial z_{11}} x_{11} + \frac{\partial L}{\partial z_{12}} x_{12} + \frac{\partial L}{\partial z_{21}} x_{21} + \frac{\partial L}{\partial z_{22}} x_{22}$$

$$\frac{\partial L}{\partial w_{12}} = \frac{\partial L}{\partial z_{11}} x_{12} + \frac{\partial L}{\partial z_{12}} x_{13} + \frac{\partial L}{\partial z_{21}} x_{22} + \frac{\partial L}{\partial z_{22}} x_{23}$$

$$\frac{\partial L}{\partial w_{21}} = \frac{\partial L}{\partial z_{11}} x_{21} + \frac{\partial L}{\partial z_{12}} x_{22} + \frac{\partial L}{\partial z_{21}} x_{31} + \frac{\partial L}{\partial z_{22}} x_{32}$$

$$\frac{\partial L}{\partial w_{22}} = \frac{\partial L}{\partial z_{11}} x_{22} + \frac{\partial L}{\partial z_{12}} x_{23} + \frac{\partial L}{\partial z_{21}} x_{32} + \frac{\partial L}{\partial z_{22}} x_{33}$$

$$\therefore \frac{\partial L}{\partial W_1} = \text{conv}\left(X, \frac{\partial L}{\partial Z_1}\right)$$



$$\frac{\partial L}{\partial Z_1} = \begin{bmatrix} \frac{\partial L}{\partial z_{11}} & \frac{\partial L}{\partial z_{12}} \\ \frac{\partial L}{\partial z_{21}} & \frac{\partial L}{\partial z_{22}} \end{bmatrix}$$

Observe carefully it is exactly same as the convolution operation

$$\left.\begin{aligned} \frac{\partial L}{\partial W_1} &= \text{conv}\left(X, \frac{\partial L}{\partial Z_1}\right) \\ \frac{\partial L}{\partial b_1} &= \text{sum}\left(\frac{\partial L}{\partial Z_1}\right) \end{aligned}\right\} \Rightarrow \text{Result}$$

Backpropagation completed

LECTURE-12: Pretrained Models

Q] Why use pretrained models?

- Deep Learning models are data hungry
  - For CNN's specifically it is labelled data (images + labels)

For example you need 10K images. Now you can scrape images from internet but labelling them manually is tedious

- Time intensive → for a lot of data, model building is slow

∴ We can directly use model trained by others → pretrained models.

ImageNet Dataset (Visual database of images)

Why?

Researcher Fei Fei Li (2006) → focus on model and algorithm building → he started building the database → along with WordNet creator → 14 million images → 20,000 categories

Out of these 1 million also has bounding box images (can be used for object localization)

How

They used crowdsourcing using Amazon Mechanical Turk

This dataset changed future of deep learning

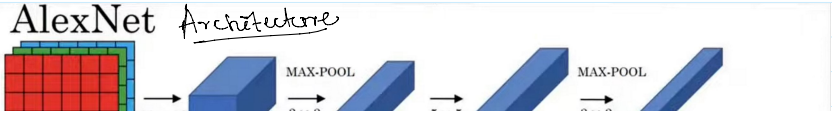
ILSVRC – ImageNet Large Scale Visual Recognition Challenge

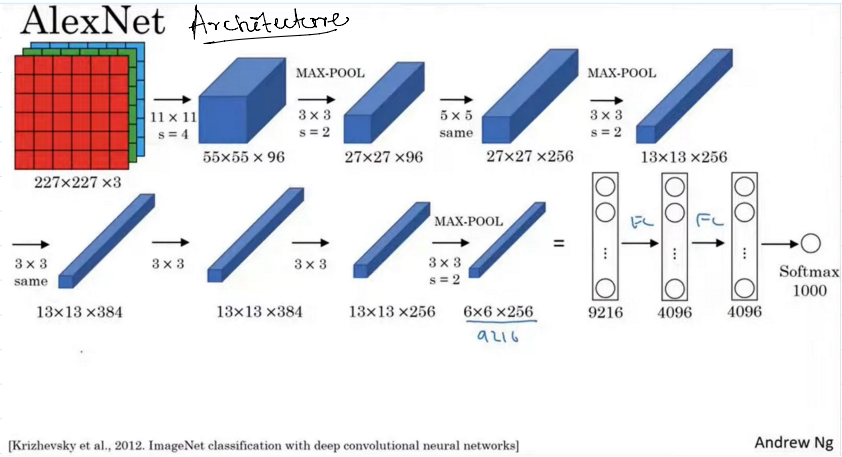
Started in 2010 → uses a subset of ImageNet (1 million, 1000 classes) → ML based models (SIFT + HOG → Ensemble) → Winning model Error rate (28%)

2011 → Error rate improved (25%)

2012 → Geoffrey Hinton ALEXNET (trademark year for deep learning) used GPU for first time, first time CNN used ReLU, Error Rate → 16%

AlexNet Architecture





2013 → 2014 → 2015 → 2016

2013 ZFNet 11.7%.

2014 VGG 7.3% famous

2015 GoogleNet 6.7%.

2016 ResNet 3.5% famous

human error rate 5%.

better than human vision

Study the architectures (better) + implement them in Keras

Idea of pretrained models

→ directly use them

→ save time

→ no need of large amount of data

→ all the trainable parameters are stored (kernel values and FC weights)

```
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import preprocess_input, decode_predictions
import numpy as np

[2] model = ResNet50(weights='imagenet')

    Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/
    102973440/102967424 [==============================] - 1s 0us/step
    102981632/102967424 [==============================] - 1s 0us/step
```

```
img_path = '/content/tomato.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

preds = model.predict(x)
print('Predicted:', decode_predictions(preds, top=3)[0])
```

→ convert images to batches

→ preprocess into format as per resnet

→ decode

```
print('Predicted:', decode_predictions(preds, top=3)[0])
```

decode predictions what are the top 3 predictions

```
preds = model.predict(x)
print('Predicted:', decode_predictions(preds, top=3)[0])
```

```
Predicted: [('n03376595', 'folding_chair', 0.9252186), ('n03201208', 'dining_table', 0.029645585), ('n03179701',
```