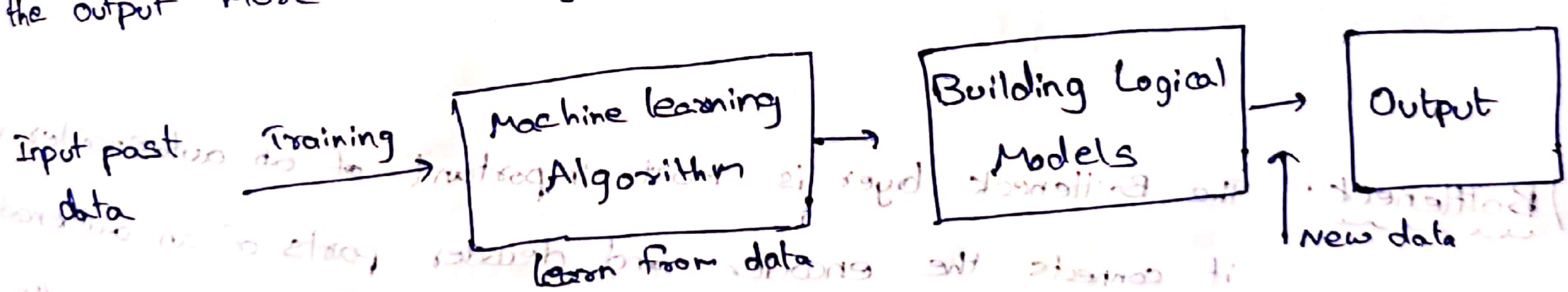
Artificial intelligence: The field of computer science aims to create intelligent machines that can think and function like humans.

Machine learning: A subset (or) subfield of artificial intelligence that focuses on developing algorithms and models that can learn from data rather than being explicitly programmed.

Deep learning: A subfield of machine learning that uses multi-layered artificial neural networks to learn complex patterns in data.

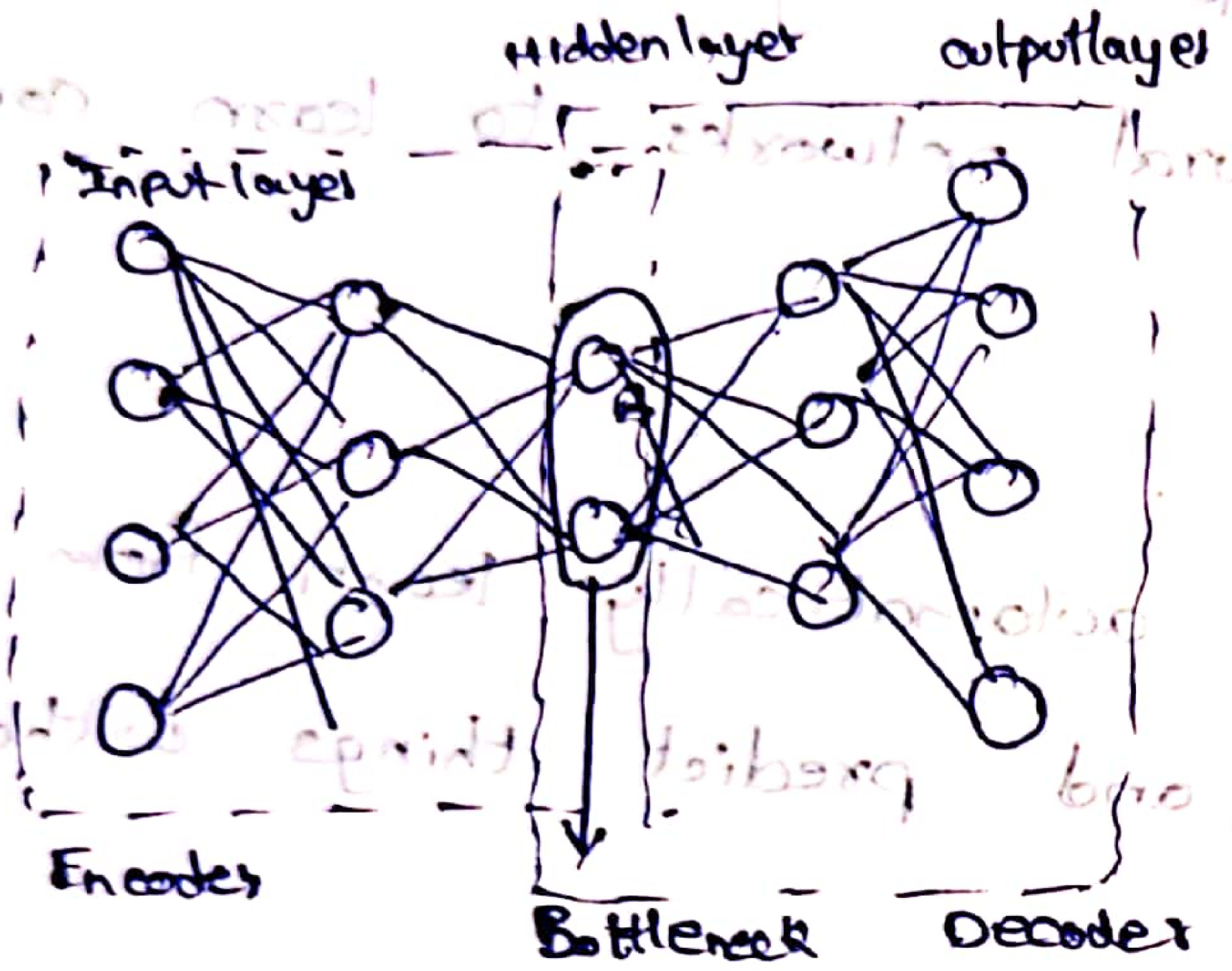
Machine learning enables a machine to automatically learn from data, improve performance from experiences, and predict things without being explicitly programmed.

A machine learning system learns from historical data, builds the prediction models, whenever it receives new data, predicts the output for it. The accuracy of the predicted output depends upon the amount of data, as the huge amount of data helps to build a better model which predicts the output more accurately.



Autoencoder :-

An Auto encoder is a type of Artificial neural network that is used for unsupervised learning. The aim of an autoencoder is to learn a lower-dimensional representation (encoding) for a higher-dimensional data typically for dimensionality reduction, by training the network to capture the most important parts of the input image.



eg of autoencoders:-
- Dimensionality Reduction
- Feature extraction
- Image Denoising, compression
- Anomaly Detection

i) Encoder :- The encoder part of the autoencoder is responsible for the compressing the input data into lower-dimensional representation. The encoder typically consists of one or more hidden layers that use non-linear activation functions to transform the input data into a lower-dimensional space.

ii) Bottleneck :- The Bottleneck layer is most important of an autoencoder, as it connects the encoder and decoder parts of an autoencoder. This layer typically has a lower dimensionality than the input and output layers and is used to create a compressed represent of the input data.

iii) **Decoder** :- The decoder part of the autoencoder is responsible for the reconstructing the original input data from the compressed representation created by the encoder. The output layer of decoder should have the same dimensionality as the input layer.

You need to set 4 hyperparameters before training an autoencoder:

i) **Code size** : most imp hyperparameter used to tune the autoencoder. The bottleneck size decides how much the data has to be compressed.

ii) **Number of layers** :- This indicates depth of encoder & decoder, while a higher depth increases model complexity, a lower depth is faster to Process.

iii) **No. of nodes per layer** :- defines the weights we use per layer. Typically the no of nodes in an encoder decrese till the bottleneck and increase in an decoder from the bottleneck.

iv) **Reconstruction loss** :- The loss function we use to train the autoencoder is highly dependent on the type of input and output we want the autoencoder adapt to.

Eg of autoencoders :-
1) Undercomplete autoencoder
2) Sparse autoencoder
3) Contractive autoencoder
4) Denoising autoencoder
5) Variational autoencoder

Variational autoencoder (VAE) :- is a type of autoencoder that is designed to learn a probability distribution over the compressed representations. Insted of encoding the input data into a fixed point in a latent space, the VAE encodes the input data into probability distrubtion over the latent space. Used for data compression, synthetic data creation, etc...

A Deep autoencoder is simply an autoencoder with multiple hidden lay in both the encoder and decoder. It is capable of learning more Complex representations of input data compared to a shallow autoencoder with only one or two hidden layers.

CNN :-

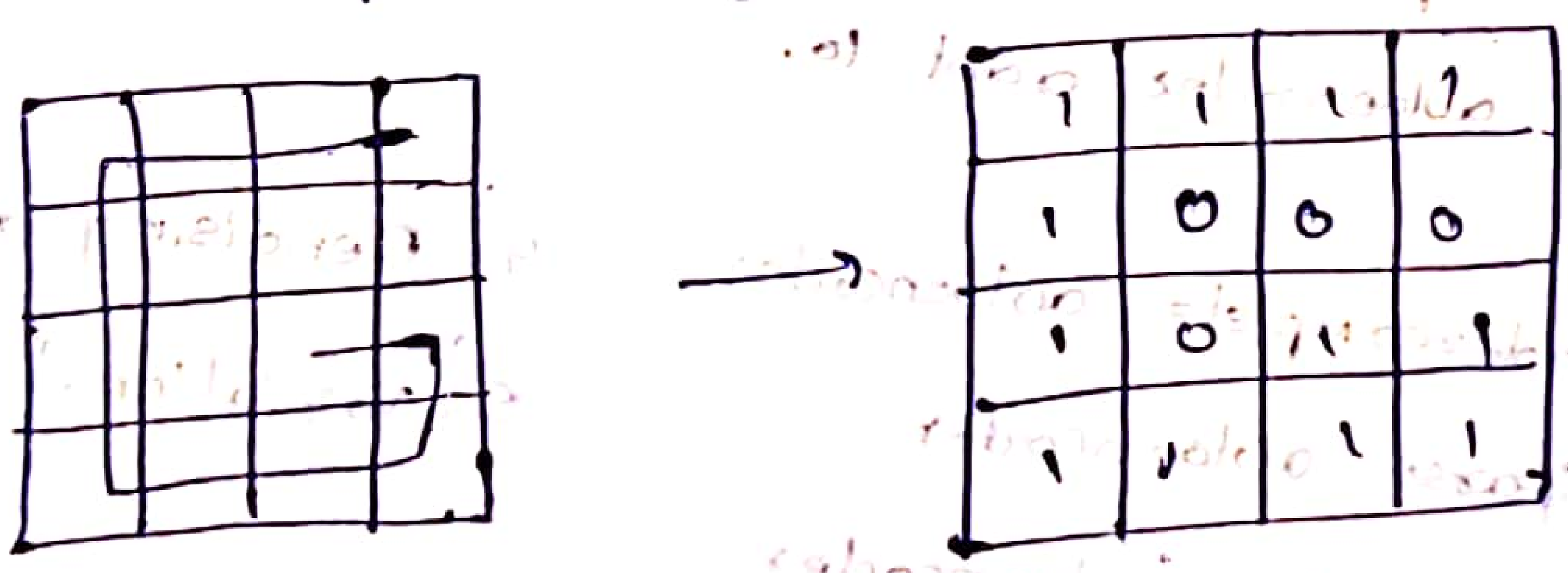
Feed For

CNN stands for convolutional neural network, which is a type of ANN Commonly used for image and video processing tasks, such as object recognition, scene reconstruction, and image classification.

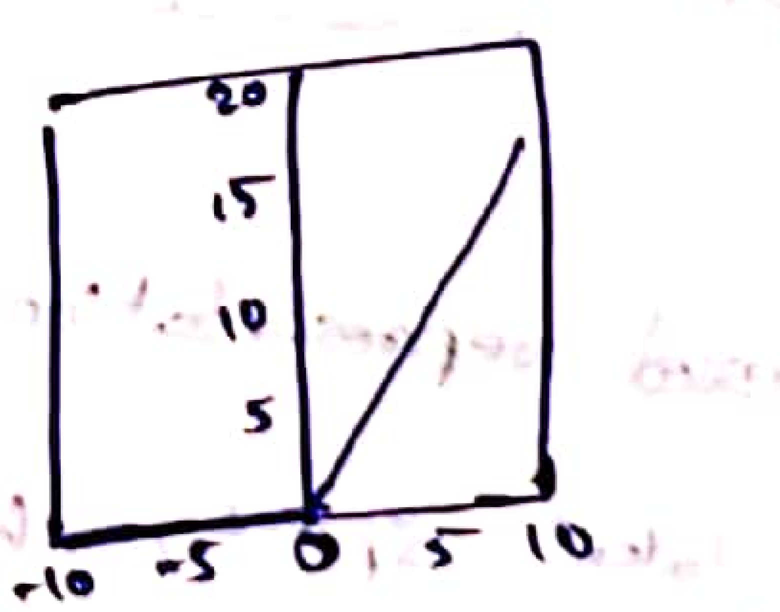
A CNN consists of multiple layers, each designed to perform a specific operation on the input data. The four import layers in CNN are,

i) Convolution layer :- This layer has multiple filters that perform the convolution operation to extract valuable features from an image. Every image is considered as matrix of pixel values, whose values is either 0 (or) (

Each filter slides over the image matrix and perform element wise multiplication & sum up the results to produce single value in feature map.

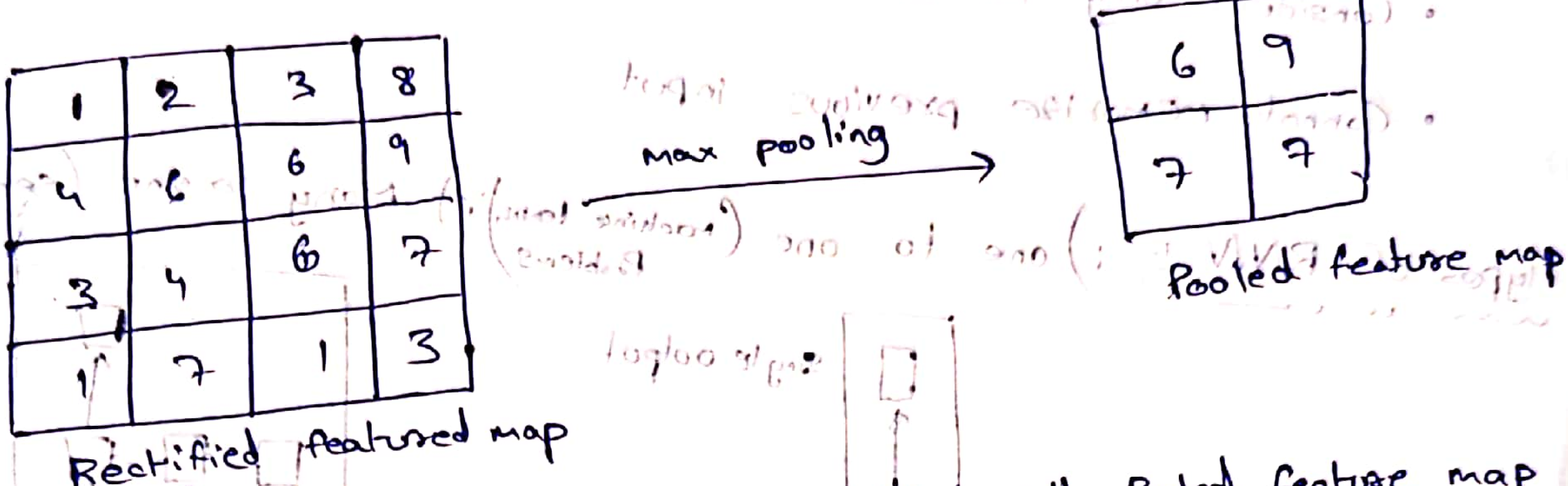


ii) RELU layer :- This layer applies rectified linear unit activation function to the output of the convolution layer. Relu sets all -ve values to 0 & +ve values remains unchanged.



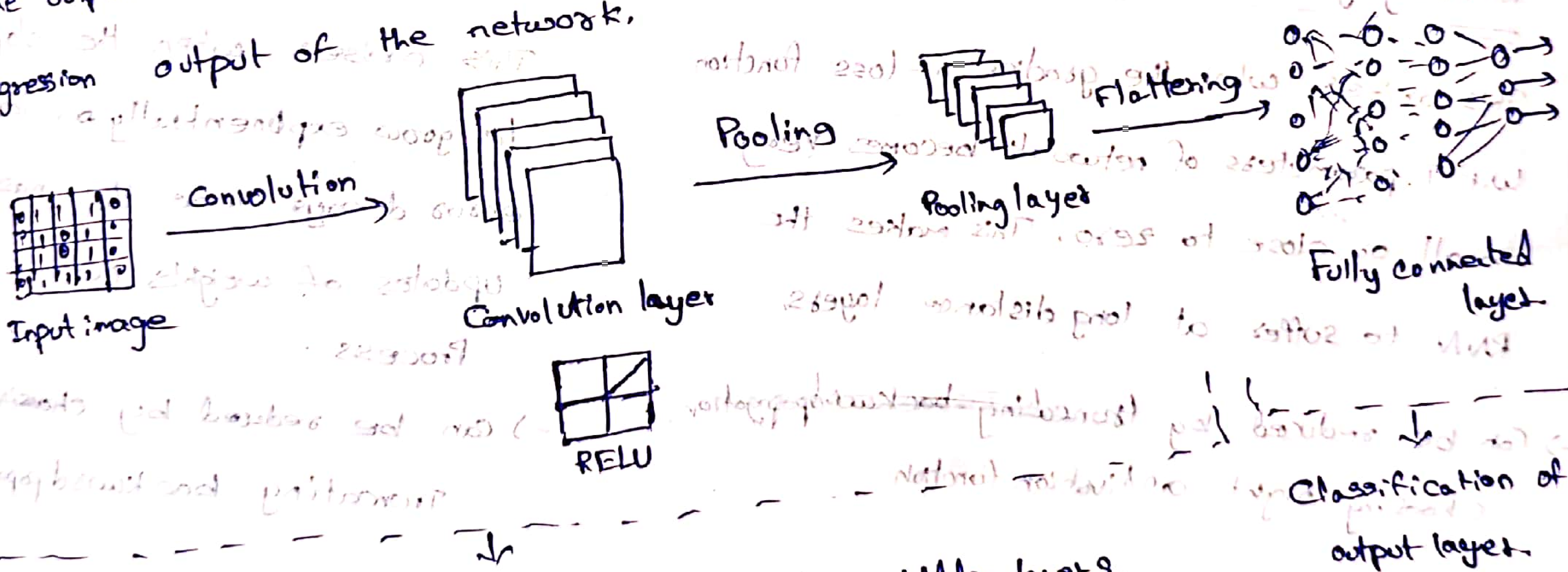
$R(z) = \max(0, z)$

iii) **Pooling layer** :- This layer reduce the spatial size of the featured map by applying a pooling function, such as max pooling, avg pooling to reduce dimensionality.

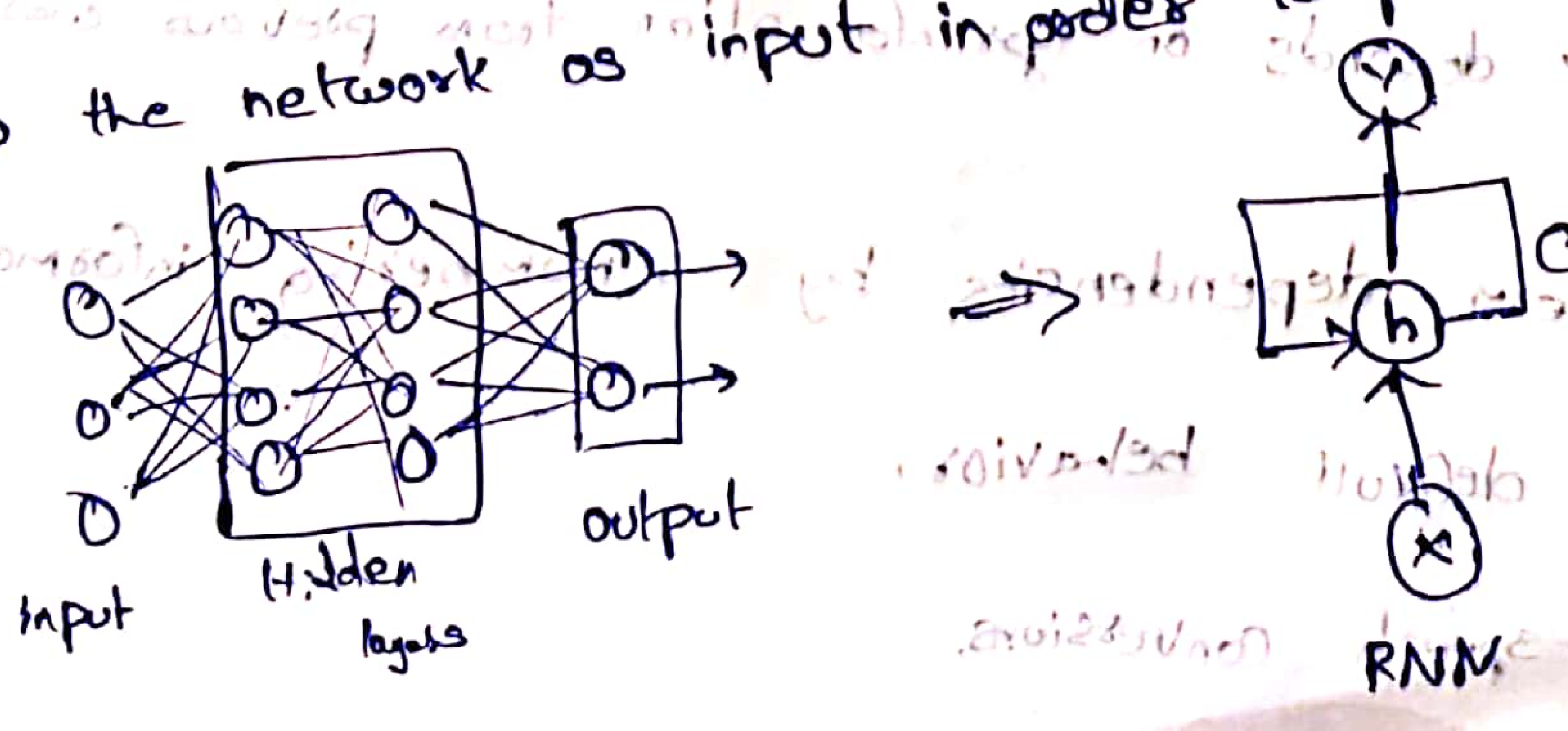


The next step is flattening. This is used to convert all pooled feature map into a single long continuous linear vector.

Fully connected layer :- This layer uses, this single long linear vector to connect with every single neuron of the next layer. The output of this generated NN is the final classification decision (or) regression output of the network.



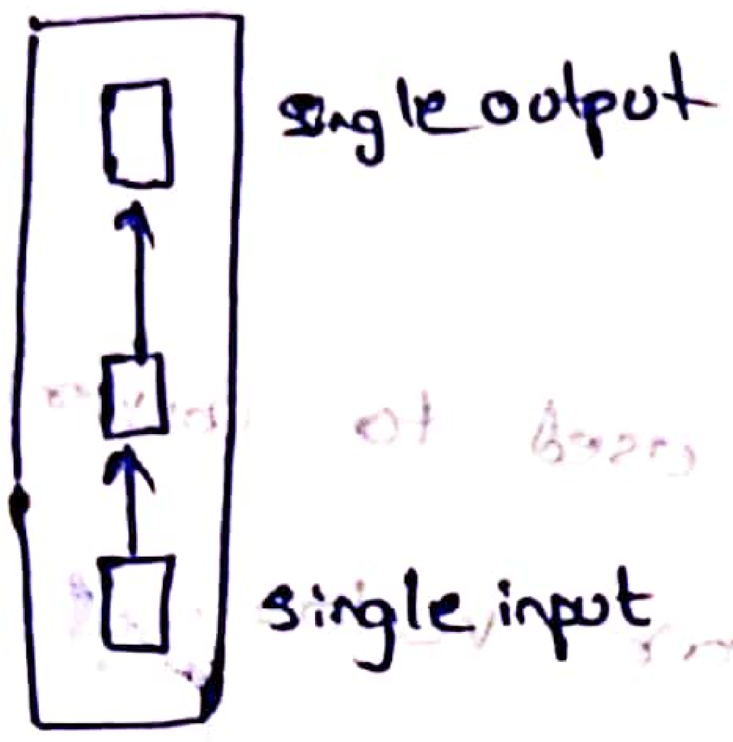
RNN :- This is a type of ANN in which the output of a network is feed back to the network as input in order to predict the output.

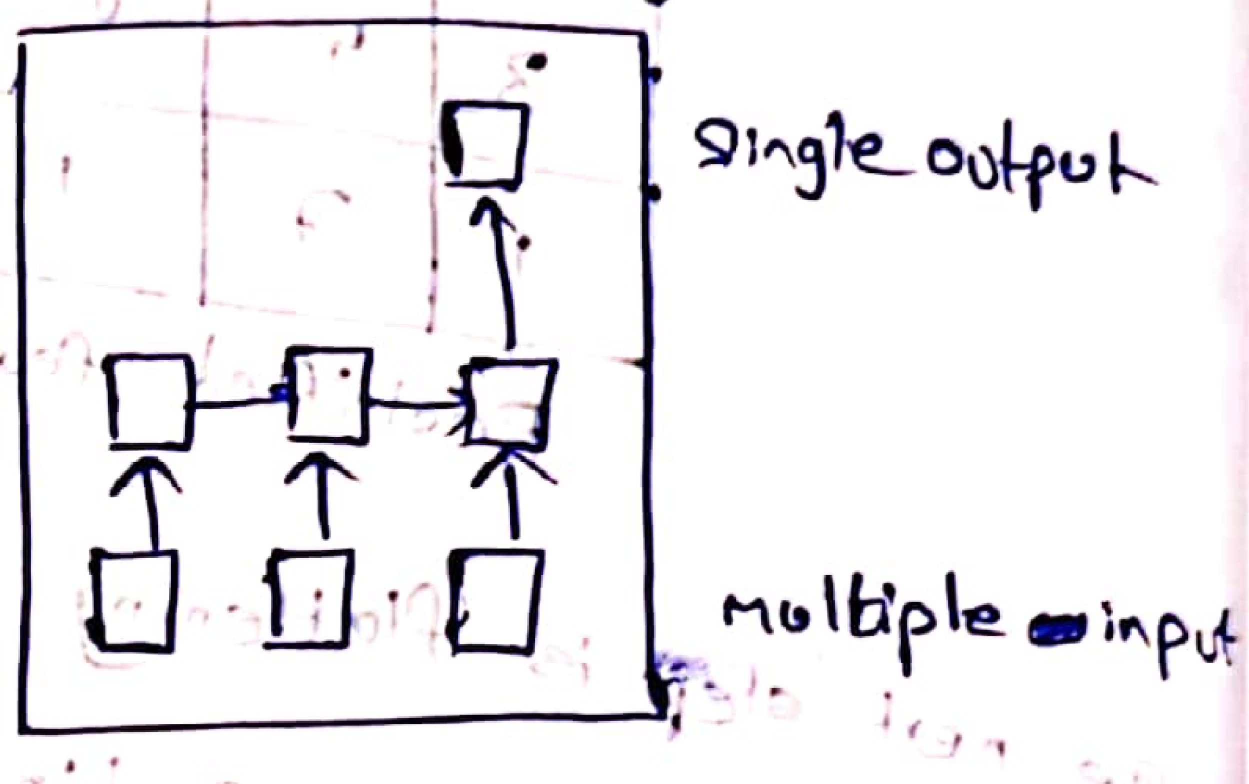


-> RNN were created because, feed-forward NN having following drawbacks

- Cannot handle sequential data
- Considers only the current input
- Cannot memorize previous input

Types of RNN :- i) one to one (machine learning problems) ii) Many to one (sentiment analysis)





similarly iii) one to many (image caption) & iv) many to many. (machine translation)

Vanishing gradient problem

This arises when the gradient of loss function W.R.T parameters of network becomes very small or close to zero. This makes the RNN to suffer at long distance layers.

-> Can be reduced by ~~truncating backward propagation~~ choosing right activation function

Exploding gradient problem

This arises when the slop tends to grow exponentially as large error decays cause the massive updates of weights during training process.

-> can be reduced by ~~choosing right~~ Truncating backward propagation

-> LSTMN -> Long short-Term memory network.

eg:- I have been in spain for last 10 Years ... I can speak fluent ______

here, the prediction work depends on ~~pr~~ information from previous words.

-> Capable of learning long-term dependencies by remembering information for long periods is the default behavior.

-> RNN is used for Text-to-speech conversions.

Bayes Rule :-

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \quad \text{(or)} \quad \frac{P(B|A)\,P(A)}{P(B|A^-)\,P(A^-) + P(B|A^+)\,P(A^+)}$$

$P(A|B) \rightarrow$ probability of event A occuring, given event B has occurred

$P(B|A) \rightarrow$ " " B " A

$P(A) \rightarrow$ Probability of event A.

$P(B) \rightarrow$ " " B.

Gradient Descent algorithm :-

is a popular optamization algorithm, used in ML & DL to minimize the cost function. (linear regrission)

- starts with initial set of parameters and updates them iteratly.
- It adjust the weights in the direction of -ve gradient of cost function.
- This is very easy to implement.

Variants of gradient descent-

i) Stochastic gradient descent (SGD) : only one training example is used to compute the gradient and update the parameter at each iteration. (may lead to noisy updates in the)

ii) Mini-batch GD : Small batch of examples are used to compute the gradient and update the parameter at each iteration. (better than batch & SGD)

iii) Momentum-based GD : A momentum term is added to gradent update to smooth out the updation process.

iv) Adagrad : Adaptive learning rate for each parameter based on historical gradients

Generative Adversarial Network (GAN):-

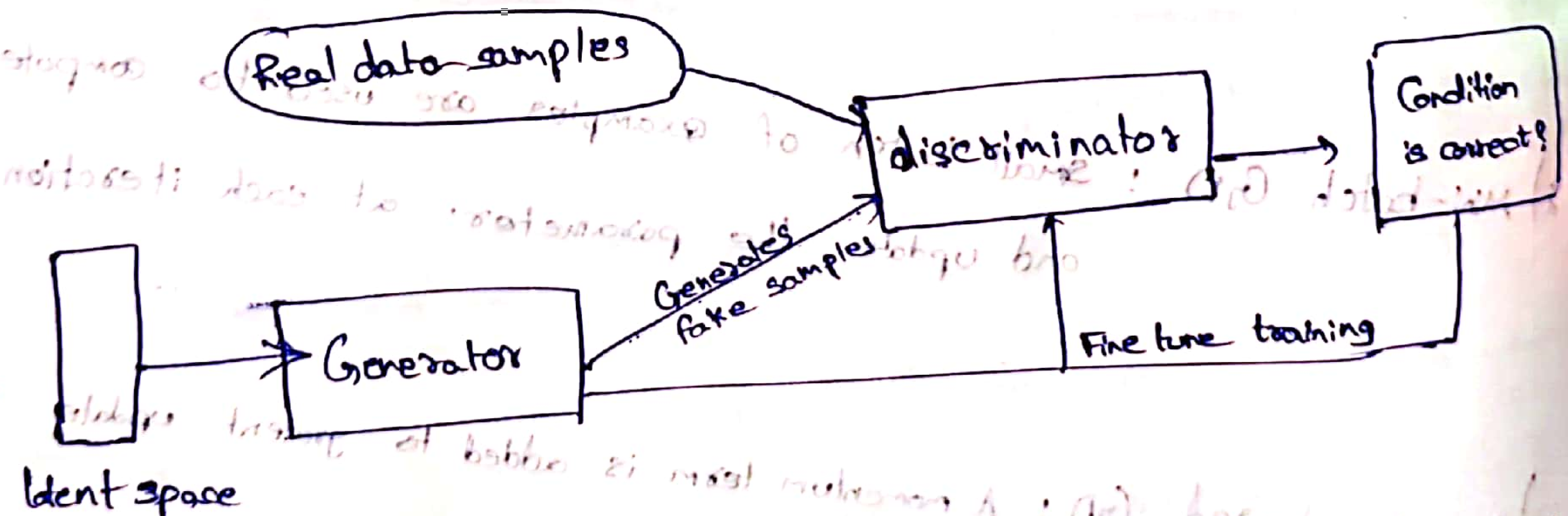
GAN is a deep learning model consisting of two neural networks: The generator and the discriminator.

Generator: The generator generates new data similar to the training data. it ~~ga~~ takes random noise vector as input and generates a new data sample as output.

Discriminator: The discriminator distinguishes between the real and generated data. It takes data sample as input and produce probability score as output, indicating wheather the input is real or generated.

~~Active~~ The generator & discriminator are trained together in an adversarial manner, where the generator tries to generate realistic samples to fool the discriminator, and discriminator tries to correctly distinguish b/w the real & generated samples.

Applications of GAN :- image and video synthesis, data augmentation and anomaly detection.



t input

Q) Perceptron networks that implements AND function:-

The output "y" is calculated by calculating net input y_{in} and applying activation function to it.

$$y_{in} = b + \sum_{i=1}^{n} x_i w_i$$

$$y = f(y_{in}) : \begin{cases} 1 & \text{if } y_{in} > \theta \\ 0 & \text{if } -\theta \le y_{in} \le \theta \\ -1 & \text{if } y_{in} < -\theta \end{cases}$$

here θ is threshold.

$w(new) = w(old) + \alpha t x_i$ (α is learning rate).

A	B	A∧B
0	0	0
0	1	0
1	0	0
1	1	1

Eg:- $w_1 = 1.2$; $w_2 = 0.6$, Threshold = 1 & learning Rate $n = 0.5$

for case i:- A=0, B=0 ; Target 0.

$w_i . x_i = 0 \times 1.2 + 0 \times 0.6 = 0 = \text{Target}$

case ii:- A=0, B=1 ; Target 0.

$w_i . x_i = 0 \times 1.2 + 1 \times 0.6 = 0.6 = 0 = \text{Target}$

case iii:- A=1 ; B=0 ; Target 0

$w_i x_i = 1 \times 1.2 + 0 \times 0.6 = 1.2 = 1 \neq 0 = \text{Target } 0$

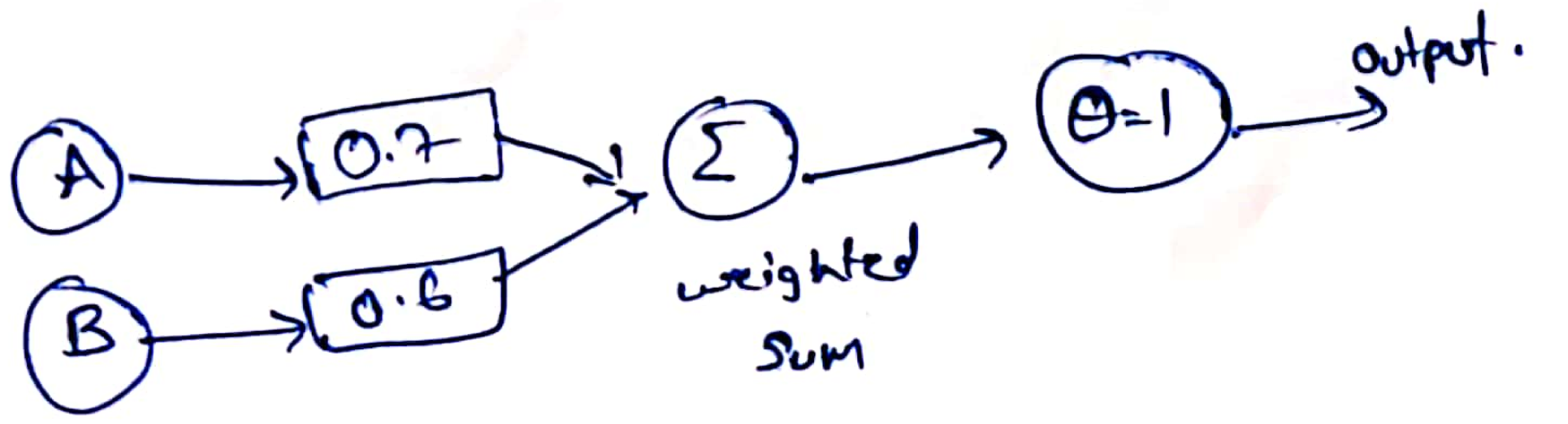
Perceptron training rule :- $w_i = w_i + n(t - \text{actual output}) x_i$ $\quad x_i = a$ in w_1, $x_i = b$ in w_2.

$w_1 = 1.2 + 0.5(0-1)1 = 0.7$

$w_2 = 0.6 + 0.5(0-1)0 = 0.6$

now $w_1 = 0.7$; $w_2 = 0.6$, Threshold = 1 & learning rate $n = 0.5$.

we will go case-by-case, in every case we will similar to above process so there is no weight updation get the calculated output = actual output, so $w_1 = 0.7$ & $w_2 = 0.6$ are final weights.



Confusion matrix :- It is a matrix used to determine the performance of the classification model for a given set of test data.

Predicted values	Actual values: Positive	Actual values: Negative
Positive	TP	FP
Negative	FN	TN

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+FN+TN}$$

(no. of correct predictions made by the classifier ratio with all other)

$$\text{Error Rate} = \frac{FP+FN}{TP+FP+FN+TN}$$

(" incorrect " " ")

$$\text{Precision} = \frac{TP}{TP+FP}$$

(ratio of actual correct output provided by correct o/p providing)

$$\text{Recall} = \frac{TP}{TP+FN}$$

(ratio of actual correct output by overall correct predic.)

$$\text{F score} = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}$$

(F score is max when Precision = Recall)