Chapter-8
Autoencoder

⇒ **Why encoder is needed:**

An encoder is used to transform high-dimensional input data into a compact & meaningful representation.

The purpose of encoder is:

i) Dimensionality reduction
ii) Feature extraction
iii) Data Compression
iv) Noise removal
v) Representation learning

⇒ **Autoencoder:**

An autoencoder is an unsupervised neural network.

Input (X) → Encoder → Embedding (Latent space) → Decoder → Re-construct Input (X')

⇒ **Applications of autoencoder:**

i) **Dimensionality reduction:**
- Alternate to PCA.
- Learns non-linear representation.

ii) **Image Denoising:** Noise image → Encoder → Decoder → Clean image.

iii) **Data Compression:** To compress data.

iv) **Feature extraction:** To extract the feature.

v) **Pretraining Deep networks:** Autoencoder can initiale the weights before supervised learning.

10-12-25

1. Q) An img of size $28 \times 28 \times \underbrace{1}_{\text{i/p channel}}$ is passed through a convolution layer having filter size = 5×5, no. of filters = 10, stride = 1, padding = 2. Find the o/p dimension & no. of learnable parameters.

2. Q) A feature map of size $20 \times 20 \times 32$ is passed through a max-pooling layer with pool size 2×2 & stride = 2. Find the size of the output feature map.

3. Q) An image of size $28 \times 28 \times 3$ is convolved with 5×5 filter using stride = 1. Find how much padding is required to ensure the output size remains 28×28.

1. Ans: Size of feature map = $\frac{28-5+4}{1} + 1 = 28$

No. of learnable parameters = $(5 \times 5 \times 1 + 1)10$

2. Ans: Size of output feature map = $10 \times 10 \times 32$

3. Ans: $\frac{28-5+2p}{1} + 1 = 28$

$28 - 5 + 2p = 27$

$2p = 27 - 23$

$2p = 4$

$p = 2$

Q) Input size = $64 \times 64 \times 3$

Conv1 ⇒ 32 filter of size 3×3, Stride = 1, padding = 1

Max pool1 ⇒ 2×2, Stride = 2

Conv2 ⇒ 64 filter of size 3×3, Stride = 1, padding = 1

Max pool2 ⇒ 2×2, Stride = 2

Fully connected layer ⇒ 10 output classes.

Find output size after each layer & total no. of learnable parameters.

- After Conv1, size of feature map = $64 \times 64 \times 32$

  No. of parameters = $(3 \times 3 \times 3 + 1) \times 32 = 896$

  In Maxpool1, size = $32 \times 32 \times 32$

  No. of parameters = 0

- After Conv2 is applied, where input = $32 \times 32 \times 32$, size = $32 \times 32 \times 64$

  No. of parameters = $(3 \times 3 \times 32 + 1) \times 64 = 18496$

  In Maxpool2, size = $16 \times 16 \times 64 = 16384$

  No. of parameters = 0

- Fully connected layer = 10 output classes, i/p size = $16 \times 16 \times 64 = 16384$

  No. of parameters = $16384 \times 10 + 10 = 163850$

Total no. of learnable parameters = $896 + 18496 + 163850 = 183242$.

11-12-25

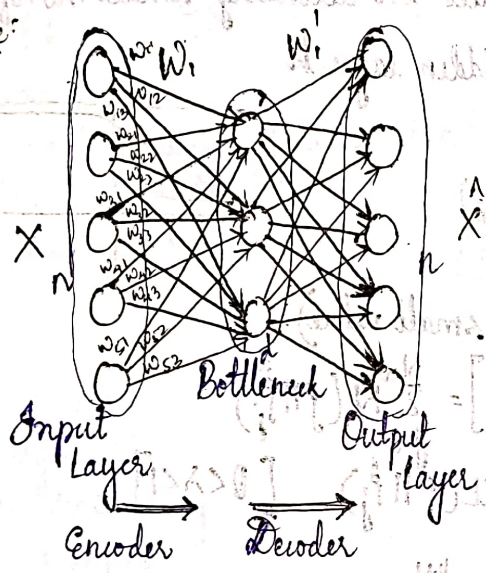
→ Autoencoder:

- Unsupervised Learning where neutral networks are subject to the task of representation learning.
- Impose a bottleneck in the network.
- The bottleneck forces a compressed knowledge representation of the input.
- The goal of autoencoder is to use encoding part of i/p data to reduce the dimensionality & decoding part will ensure the correctness of encoding layer by determining that x (i/p) will give $\hat{x}$ (o/p).

→ Assumption:

- High degree of correlation/structure exists in the data.

→ Structure:



$d << n$

$f'(f(x)) = \hat{x}$

→ Expectation:

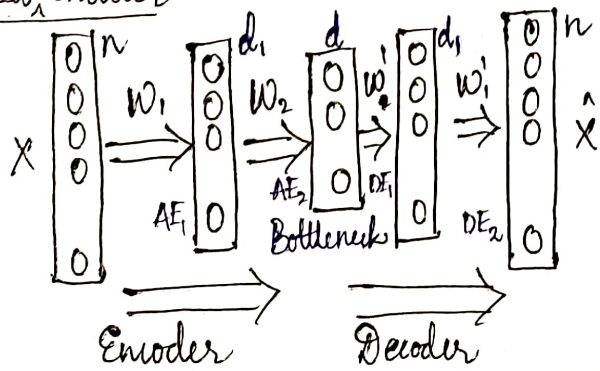
- Sensitive enough to i/p for accurate reconstruction.
- Insensitive enough that it doesn't memorize or overfit the training data.

⇓

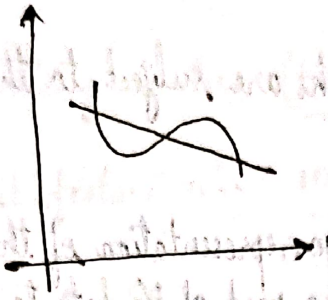
Loss Function ⇒ $L(x, \hat{x}) + \text{Regularizer}$

$$L(x, \hat{x}) = \frac{1}{2} \sum_{x} \| x - \hat{x} \|^2 \qquad \left[|x - \hat{x}| = \sqrt{(x_1 - \hat{x}_1)^2 + (x_2 - \hat{x}_2) + \dots} \text{ where } x = \{x_1, x_2 \dots x_n\} \right]$$

• Stacked Auto-encoder:



⇒ Autoencoder V/S PCA:



15-12-25

⇒ Sparse Autoencoder:

- Interesting features can be learnt even when no. of nodes in hidden layer is large.
- Introduce sparsity constraint on the hidden layer nodes that penalize activations within a layer.
- Network learns encoding-decoding that relies on activating small no. of neurons.

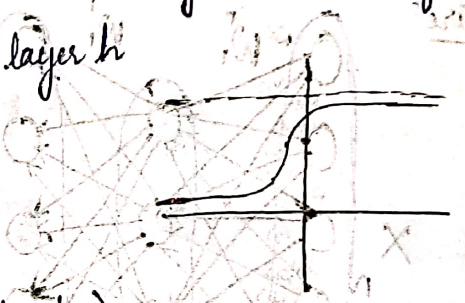
$a_j^h \rightarrow$ Activation of j^{th} Neuron in hidden layer h

$a_j^h \rightarrow 1 \Rightarrow$ Neuron is active

Average Activation $\rightarrow \hat{\rho}_j = \frac{1}{m}\sum_{i=1}^{m} a_j^h(x_i)$

Constraint $\rightarrow \hat{\rho}_j = \rho$

$\rho \rightarrow$ sparsity parameter (typically a small value).



Regularization: $\sum_{j=1}^{N_h}\left[\rho \log\frac{\rho}{\hat{\rho}_j} + (1-\rho)\log\frac{1-\rho}{1-\hat{\rho}_j}\right] = \sum_{j=1}^{N_h} KL(\rho \| \hat{\rho}_j)$

$$J_{sparse}(W) = L(x,\hat{x}) + \lambda \sum_j KL(\rho \| \hat{\rho}_j) \quad [0<\lambda<1]$$

Sparsity Constraint: $\delta_i^k = O_i^k(1-O_i^k)\sum_{j=1}^{M^{k+1}} \delta_j^{k+1} W_{ij}^{k+1}$

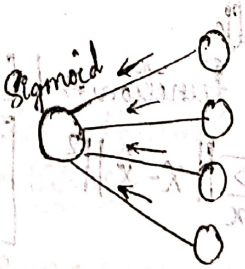
$$\delta_i^k = O_i^k(1-O_i^k)\left[\sum_{j=1}^{M^{k+1}} \delta_j^{k+1} W_{ji}^{k+1} + \lambda\left(-\frac{\rho}{\hat{\rho}_i} + \frac{1-\rho}{1-\hat{\rho}_i}\right)\right]$$

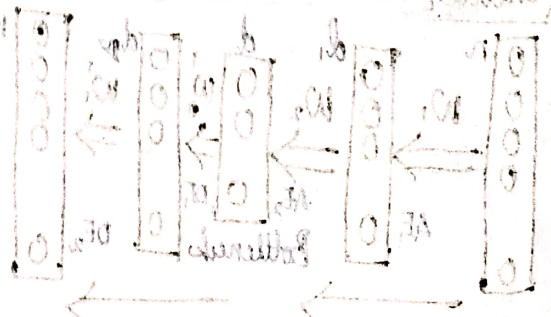
Sigmoid: $y = \frac{1}{1+e^{-z}}$

$$\frac{\partial y}{\partial z} = \frac{e^z}{(1+e^{-z})^2} = \frac{1+e^z}{(1+e^{-z})^2} - \frac{1}{(1+e^{-z})^2}$$

$$= y - y^2$$

$$= y(1-y)$$





17-12-25

Chapter-4
Recurrent Neural Network (RNN)

⇒ What is sequence learning?

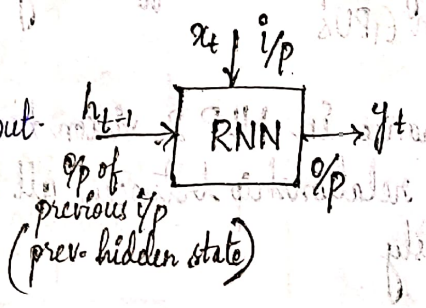
Sequence learning is the study of machine learning algorithms designed for sequential data.
Eg- Language translation.

⇒ Limitation of ANN & CNN:

i) No memory of past inputs.
ii) Fixed input size.
iii) Poor in sequential data.
iv) Parameter Explotion.
v) High data requirement.

⇒ Why RNN?:

i) It has memory.
ii) It handles variable-length sequential input.
iii) It remembers past information.
iv) It model time dependencies.



⇒ Problems:

i) Language modeling
ii) Speech recognization
iii) Time series forecasting
iv) Sentiment analysis
v) Video analysis

⇒ Types of RNN:

i) One-to-One

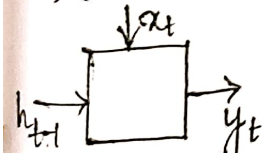


Image classification

ii) One-to-Many

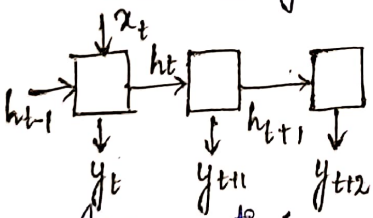
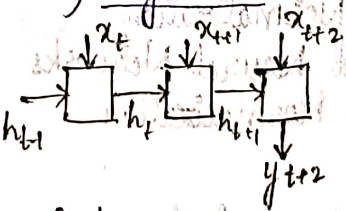


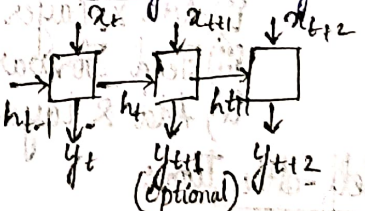
Image captioning

iii) Many-to-One



Sentiment analysis

iv) Many-to-Many



Speech Recognition

22-12-25

→ Limitations of RNN/LSTM:

i) It perform sequential computation, it cannot perform parallel computation.
ii) Vanishing / Exploding Gradient.
iii) Memory Bottleneck

I ate an apple.
↓ ↓ ↓ ↓
x_1 x_2 x_3 x_4

To handle this type of limitation, transformer concept is introduced.

→ Transformer:

It cannot process data one by one, it will process whole data.

Why?

i) Parallel processing of sequence
ii) Handle long-range dependence efficiently.
iii) Faster training in GPUs.
iv) Better scalability.
v) Superior performance in NLP & vision task.

Transformer learns relationship between all words in a sentence & look at all tokens simultaneously.

→ Architecture:

Input → Embedding → Positional Encoding → EncoderStack → DecoderStack → Output

- A position of data is important so positional encoding is used.
- It has two main components:
  i) Encoder
  ii) Decoder

1) Encoder: Each encoder layer contains:
   i) Multi-Head Self Attention
   ii) Add & Layer Normalization
   iii) Feed Forward Neural Networks
   iv) Add & Layer Normalization

→ Self-Attention:

Self Attention allows each word to -

- attend to all other words in the sentence.
- Decide which word is important.
- Using self attention it will decide the relationship between the words i.e, how the words are related to each other.

→ Embedding:

Input : $x_{(Q,K,V)}$

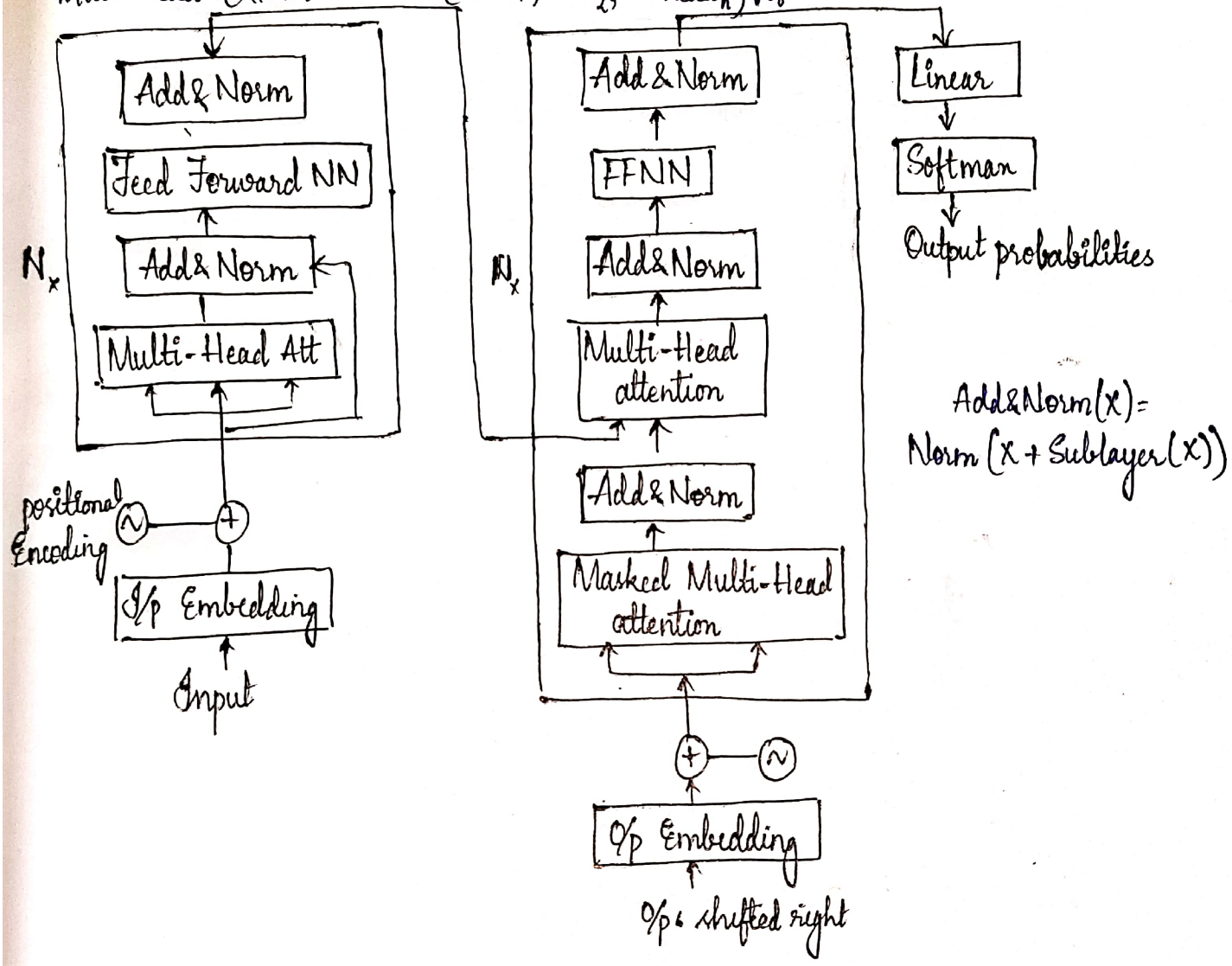
For each input word embedding:-

$Q = xW_Q$, $K = xW_K$, $V = xW_V$

$head_i = Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$ where d_k = dimension of key vector k

Multi-head Attention-

Multi Head $(Q,K,V) = concat(head_1, head_2, \ldots head_n)W_0$



Add&Norm(x) = Norm (x + Sublayer(x))