28/10/22

Autoencoders.

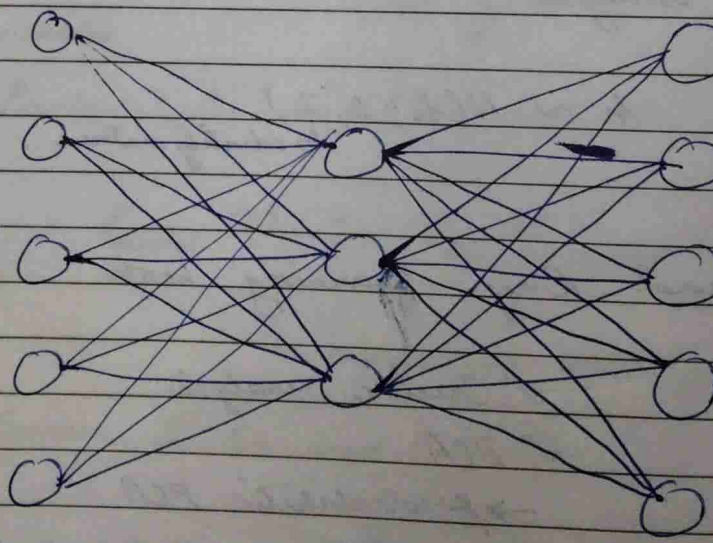
⇒ No target variable
→ unsupervised learning

⇒ Tries to reconstruct the input
↓
image, text, ...

Input is $x \in \mathbb{R}^D$

⇒ output is also ~~$x \in x$~~ $r \in \mathbb{R}^D$ → stands for reconstruction
(same dimensions)

Compression followed by decompression
(I/p layer → hidden layer) (hidden layer → output layer)



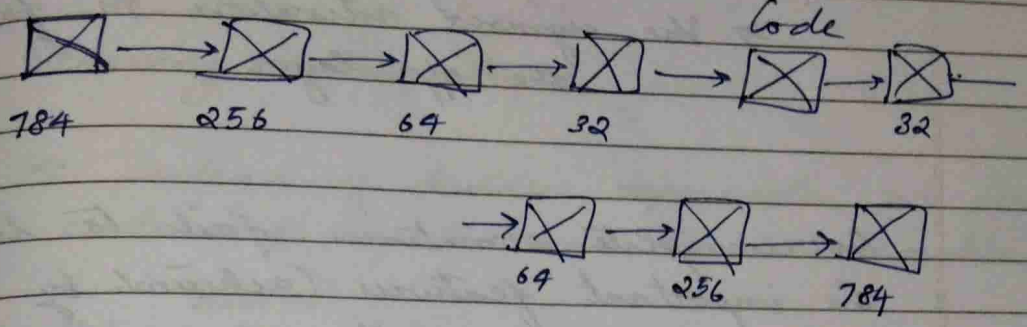
I/p
x.

hidden
(bottleneck)
$h = f(x)$
↓
encoder

O/p
$r = g(h)$
↓
decoder

compression → decompression

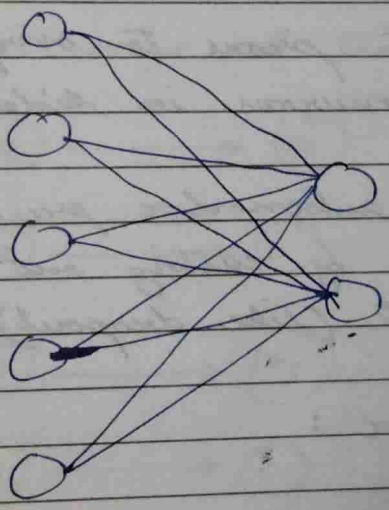
Decoder network ⇒ reverse of encoder



~~Undo~~

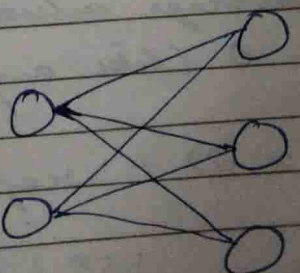
Undercomplete autoencoder :

Bottleneck has smaller dimension than i/p layer



Overcomplete autoencoder :

Bottleneck dimension > input dimension (or) previous layer's dimension?



Only Input - binary
⇒ autoencoder o/p must be binary
⇒ Use sigmoid activation fn for the o/p layer

Overcomplete sometimes fails to learn the important features (achieved by compression) because it only as good as simply copying the given image (not learning much).
⇒ can be useful if there are more latent variables

Overcomplete prone to overfitting as it uses more neurons in hidden units.

Regularized autoencoder prevents this overfitting by cutting out certain hidden units (like dropout).

Sparse autoencoder :

Here we penalize the activation of hidden layers (instead of weight as in regularisation)
⇒ only few nodes will be active

Assumption : average activation is nearly 0 (like a sparse matrix)

Objective fn = $\underbrace{L(x, \hat{x})}_{\text{loss}}$ + regularization +

hyperparameter ← $\sum_i |a_i^{(h)}|$

So we try to force $\sum_i |a_i^{(4)}|$ close to 0 to minimize the objective fn.

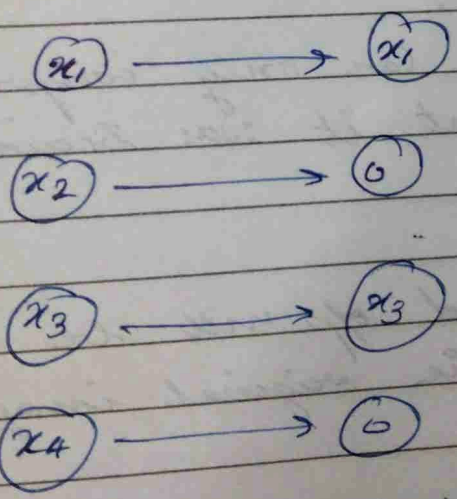
Denoising autoencoder:

→ Instead of hidden ~~inputs~~ nodes, we force certain of input nodes to be inactive

Idea: passing noisy image as input (achieved by ... certain input features) ~~such that noises are filtered out~~;

Output image - free from noise

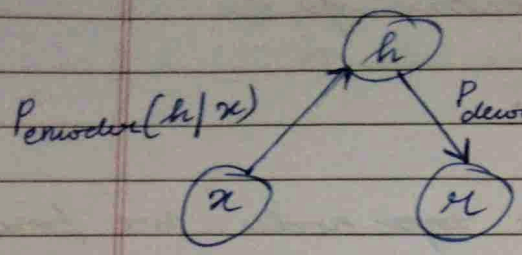
Loss computed between output and original image not the noisy image



orig ip

noisy img after dropping 50% of the ip nodes.

Stochastic Encoders and Decoders



Uses a probabilistic approach
encoding function $f(x)$ is replaced
with a probability distribution
$P_{encoder}(h|x)$.
Ulearly for $g(h)$ also $\rightarrow P_{decoder}(x|h)$.

Properties of autoencoder

1) Data-specific :
Can only compress data
similar to what it was trained on

2) Lossy :
decompressed o/p will be degraded
compared to the original image

3)

Applications :

✓ Anomaly detection
✓ ~~Object detection~~ Segmentation
(UNet, VNet, TransNet all use
autoencoders)

29/10/22

1 LSTM neuron → 4 weights

utf-8 encoding ⇒ plaintext input

Preprocessing steps (for text)

1. Convert to lowercase
2. Remove numbers
3. Map every char to int

We

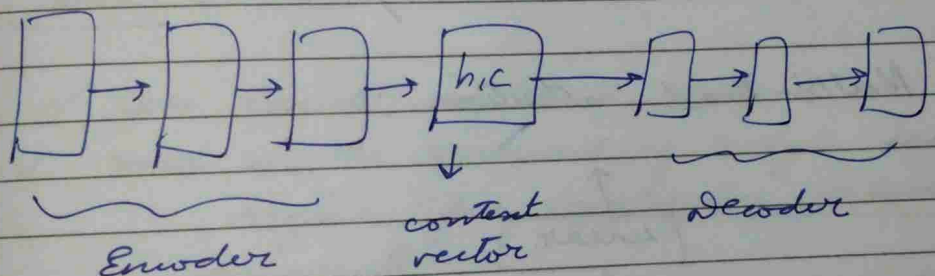
↳ candidate

(new candidate value to be stored in memory).

date 11/11/22

RNN	Transformer
Sequence to sequence processing ⇒ there is dependency	Processes all sequence at once (not word by word ⇒ there is no dependency)
No parallel computation	Parallel computation
so slower → Difficult to process longer sequences	Faster Easier to process long sequences (as there is no dependency)

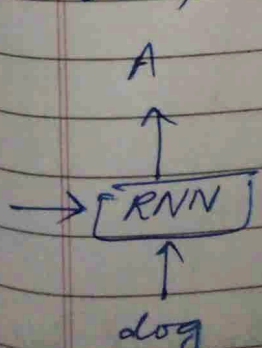
Language translation :

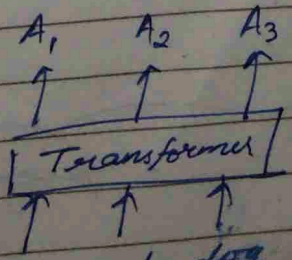


Encoder-decoder models ⇒ self-supervised

Transformer uses 2 new layers

✓ Attention
✓ Positional encoder - captures content info





1. Attention module

Used to preserve semantics

(Eg): How are you?

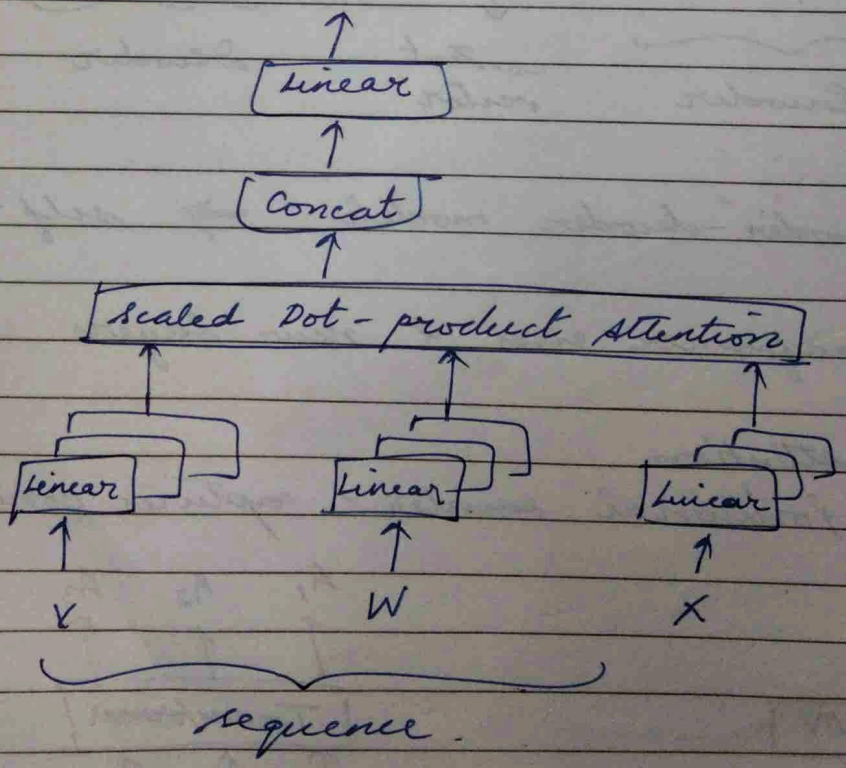
How - कैसे
are - हो
you - तुम

But after putting them together the sentence becomes

तुम कैसे हो

(These semantics are preserved by using multi-head attention).

Multi-head attention:



Positional encoding adds contextual info to a embedding word

Fine tune - calculate error & backpropagate

Transfer learning :

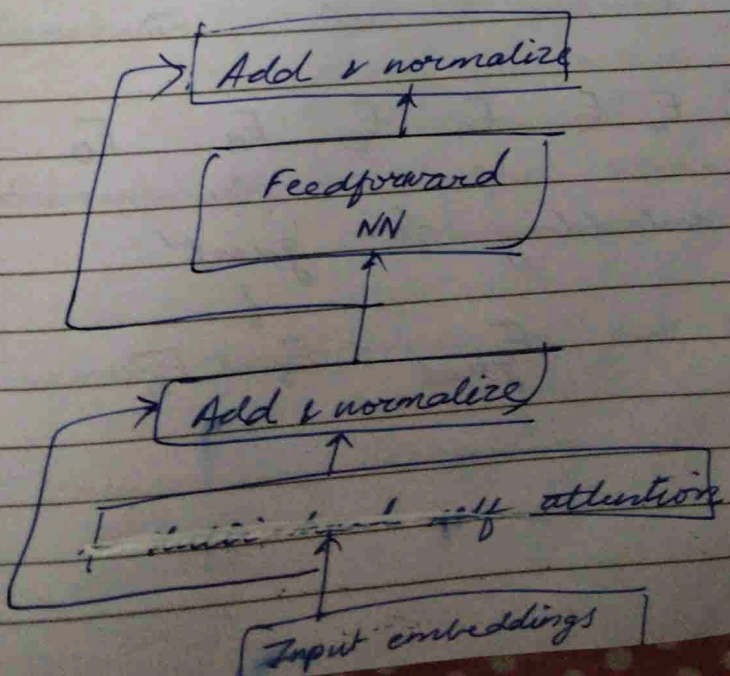
2 steps :
1. Pre-train a model
2. Fine tune

Transfer learning for NLP :

✓ BERT → Bidirectional Encoder Representations from Transformers
✓ GPT
✓

BERT :

✓ Can handle sentiment analysis
✓ used in the backend of Google search engine for identifying relevant documents
✓ Similar questions displayed on Quora



SEP → used to separate sequences

CLS → classification
used at beginning of input
to identify the task

(Eg): CLS My cat is great SEP she loves food SEP

BERT uses 3 input embeddings

- ✓ Token
- ✓ Position
- ✓ Sentence

} Input embedding

14/

CLS My cat is great SEP she loves food SEP

Token E_{cls} E_{my} E_{cat} E_{is} E_{great} E_{SEP} E_{she} E_{loves} E_{food} E_{SEP}

↓ used to id the token

position E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

↓ used to id position of token

Sentence E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B

Input embedding for 'great'

↓

$E_{great} + E_5 + E_A$

BERT Pre-training

1. Predict the mask input ↑ allows BERT to better understand words.
2. Check whether subsequent sentences follow the previous sentence or not

My cat is great. It will rain today

completely unrelated to sentence 1.

Image data – VGN transformer

14/11/22

Generative Adversarial Networks (GANs)

like autoencoder
⇒ generates i/p again at o/p

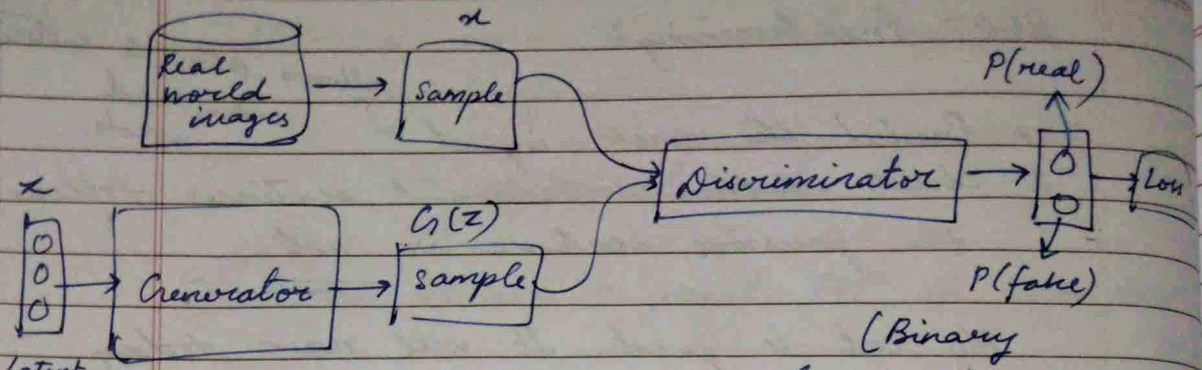
2 neural networks
→ Generator
→ Discriminator
} zero-sum game (one wins, other loses)

Generator – tries to generate fake i/p from i/p which is passed to discriminator

Discriminator – identifies which inputs are fake

Generator exists only to fool the discriminator

⇒ Unsupervised learning



z → random noise (Gaussian / uniform).

~~Steps~~

1. If generator's output is correctly classified ~~then~~ by discriminator then generator has to be improved (update by backprop)
   ∀ v
   here ~~generator's~~ discriminator's weights are frozen
2. If generator's output is incorrectly classified by discriminator then discriminator has to be improved (generator's weights - kept const ⇒ not updated).

2 types of loss fn:

Discriminator ⇒ Cross - entropy loss

$$\min_G \max_D V(D,G) = E_x[\log(D(x))] + E_z[\log(1 - D(G(z)))]$$

$$G(z) = \begin{cases} 1 & \text{if image is generated by generator} \\ 0 & \text{otherwise} \end{cases}$$

Correctly classifying wrong data

$D(G(z)) \rightarrow 0$ if $G(z) = 1$
(classify as fake if img is generated by Generator).

Correctly classifying real data

$D(x) \rightarrow 1$

Objective of generator is to minimize ~~the~~
V
Objective of discriminator is to maximize
V