Autoencoders

→ ~~Aim~~ Just copying input to output (not exactly 100%, but 80-90%)

→ Adv of Autoencoders
1) Quality enhancement
2) We can get compressed image in intermediate stage.
3) It also learn some features during intermediate stage.

Encoder :- It is the part of network that compresses input into latent space
We can use CNN, ANN, RNN or any architecture.

Decoder :- To reconstruct the input

▷ Under complete : no. of input layers > no. of hidden layer

⇒ Types of autoencoders

1) Vanilla autoencoder

2) Auto transcoder →
3) Self encoder →

Hyper parameters
1) Number of layers
2) Nodes per layer
3) Code size

▷ Key words:
→ reconstruction should be there.
↳ not make ~~much~~ too powerful as it will overfit
→ less nodes than previous layer.
→ mirror image should be there (of layers) (in case of CNN also)

If upsample2D()
→ yolo

→ Along with classification, we can use regression too in CNN.
→ Simple classification classifies image by image.
→ Segmentation classifies image pixel by pixel.

▷ Autoencoder:
Original Input → Encoder → (hidden layers) → Latent Representation → (Hidden layers) → decoder → Reconstructed output
(can use CNN/ANN here.)

Autoencoder is a methodology not an architecture like CNN, ANN
▷ It's a PCA of non-linear data.
▷ dimentionality reduction
▷ de-noise

i) Vanilla
ii) Multilayer
iii) Convolutional
iv) Regularized

→ Upsampling process (one way of doing upsampling)

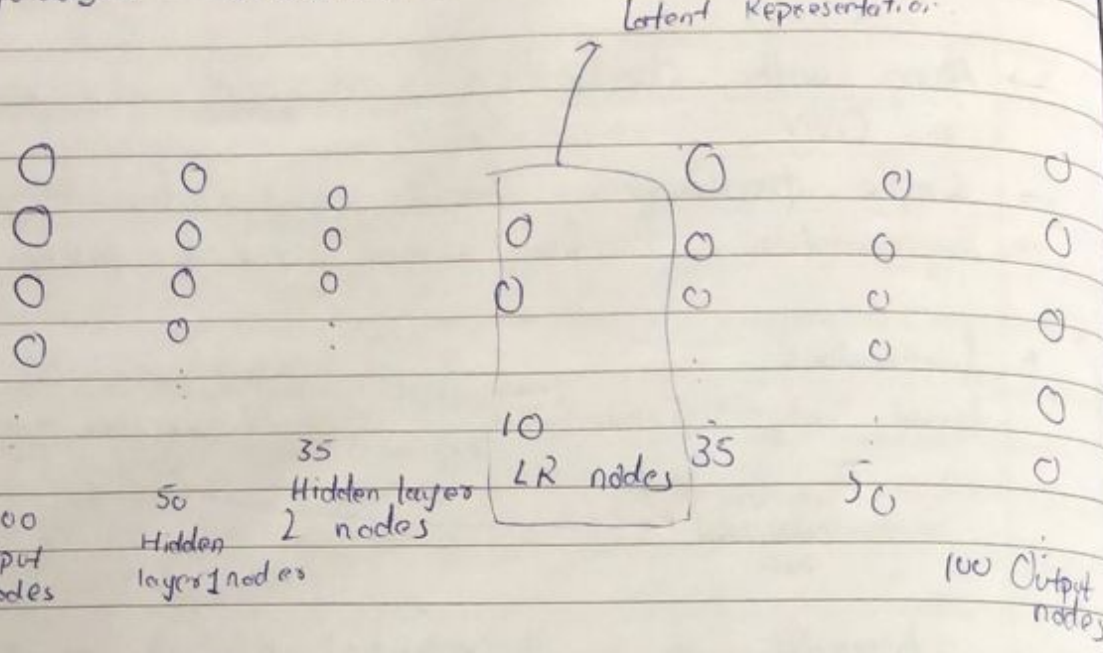
→ because acc to KNN if shares similar characteristic

Eg. [2 5 / 7 8] —Max Pooling→ [8] —Up sampling→ [8 8 / 8 8]

⇒ Convolutional & Autoencoder
⇒ We do not update weights during ~~padding~~ max pooling
→ ∴ Padding cannot be applied on max-pooling as if it will apply ~~inp~~ we will get same a input & output.

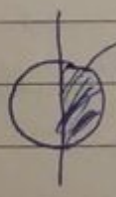
▷ in code part: (i) padding = "same" → same output 4×4 → 4×4
= "valid" → reduced output 4×4 → 2×2

→ Regularized autoencoder.



If 100 100 100 10 100 100 100 nodes are these, then we will use dropout to reduce the nodes as using all nodes is not feasible.

* Regularized autoencoder
  → dividing node in aggregation + activation.
  → applying regularization on activation function (instead of classic way of penalizing weights, we are penalizing activation function) to make some nodes inactivate. (to make ~~pred~~ output from activation fx zero).
  ↳ same as dropout in CNN



Types

i) Sparse Autoencoder

→ We will regularize output for each node during each iteration & if it is near zero, then we will disable that

node, resulting to our reduced number of nodes, i.e. dropout process.

→

▷ Some nodes - to give opportunity to model to learn imp features. (reducing the feature; picking up only imp ones)

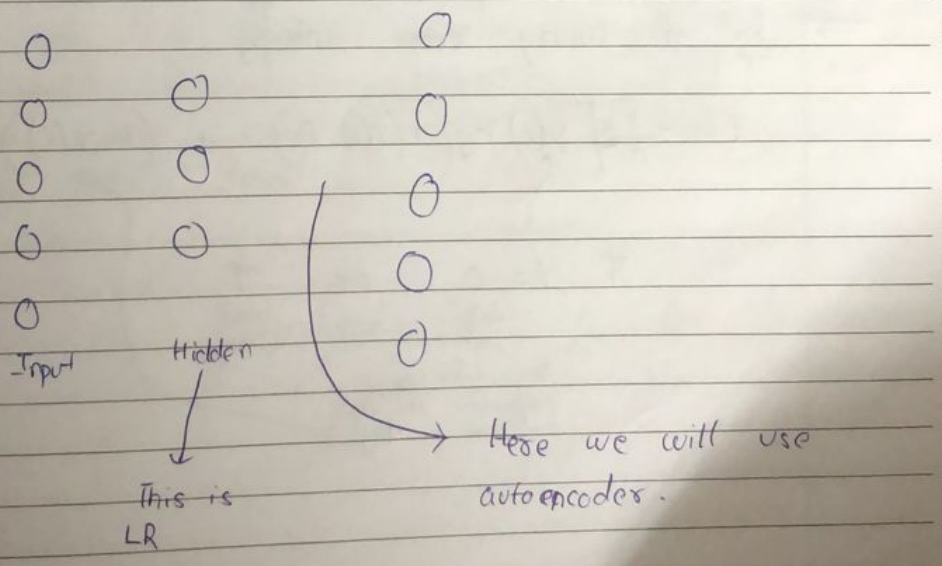
ii) Denoising autoencoders

Need :-

→ Helpful in training

→ We will use smoothening for good edge detection.

⇒ Stacked Autoencoders



▸ Why Autoencoder and not CNN

→ Can determine the problematic layer in autoencoder but not in CNN.

⇒ GAN

→ At the same time we cannot reduce loss of both generator & discriminator, one will increase & one will decrease.

→ ~~If gen~~ Unlike autoencoders, GAN generates new images

→ At first time discriminator doesn't know anything ~~but~~ so it will add noise to just generate output.

→ After then, it will backpropagate ~~the~~ & will update image.

→ Training of discriminator is done first, ~~training~~ of generator is done after, because discriminator provides feedback to the generator, thus helping it.

→ We don't use any real image to train generator.

→ Eq^n of binary cross entropy.

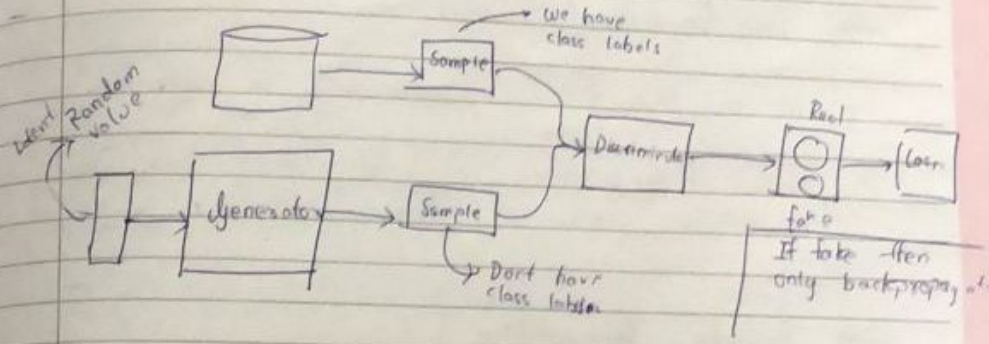
$$L = -\frac{1}{m}\sum\left[y(i) \log\left(h(x(i), \theta)\right) + (1 - y(i)) \log\left(1 - h(x(i), \theta)\right)\right]$$

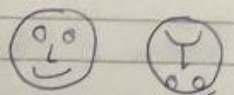
If $y = 0$ or 1

$y = 0$ → fake image

$y = 1$ → real image

So in fake image proportion of this component is higher





Here as the features are same it will identify both faces as human even though this isn't.

→ We can resolve this problem by capsule net. This problem occurs in all the studied architecture RNN, CNN, Autoencoders.

⇒ Training GANs

→ We will do Gradient ~~Descent~~ Ascent on discriminator & Gradient Descent on Generator. BUT

→ We will face vanishing gradient problem till certain level.

→ So we will do

1) Gradient Ascent on ~~Gen~~ Discriminator
2) " " " Generator.