2nd dec

Deep Learning Assignment

2018004036
Hrithik Sagar Rachakonda
180101240
CSE C G2 [VII]

---

① GAN :- Generative Adversarial Network

→ Its a ML model in which 2 NN compete with each other to become more accurate in their predictions.

→ They are typically unsupervised

→ They use cooperative zero sum game framework to learn

→ There 2 NN's are known as :-

(i) Generator :- CNN

(ii) Discriminator :- de-CNN

Generator goal :- artificially manufacture outputs that could easily be mistaken for real data

Discriminator goal :- Identify which output it receives have been artificially created

→ GANs create their own training data
→ While feedback btw GAN continues, Generator will begin to produce higher quality output and discriminator will become better at flagging data thats artificially created.

Working :-
→ Step 1 is to identify desired end output then gather initial training dataset until it aquires basic accuracy in producing outputs.
→ Step 2 :- Generated images are fed to discriminator along with actual data points from original concept.
discriminator filters through info and returns a prob btw "0 and 1"
1 = real
0 = fake / artificial.
these values are manually checked for success and looped until desired outcome is reached

uses
① fully image from outline
② generating realistic image from text
③ b & w image → colour
④ Predicting subsequent video frames
⑤ Create deep fakes

beyond art :-

→ Image to image translation
  Using Sentence/phrase mapping

→ Text Generation
→ Generate Network Graph
→ Audio Synthesis

---

(2) Generator :-

Consist of Deconvolutional layer / Transposed - CNN layer that perform reverse of convolution operation.

Creates fake data by incorporating feedback from discriminator.

Learns to make discriminator classify its output as real.

Generator training include :-

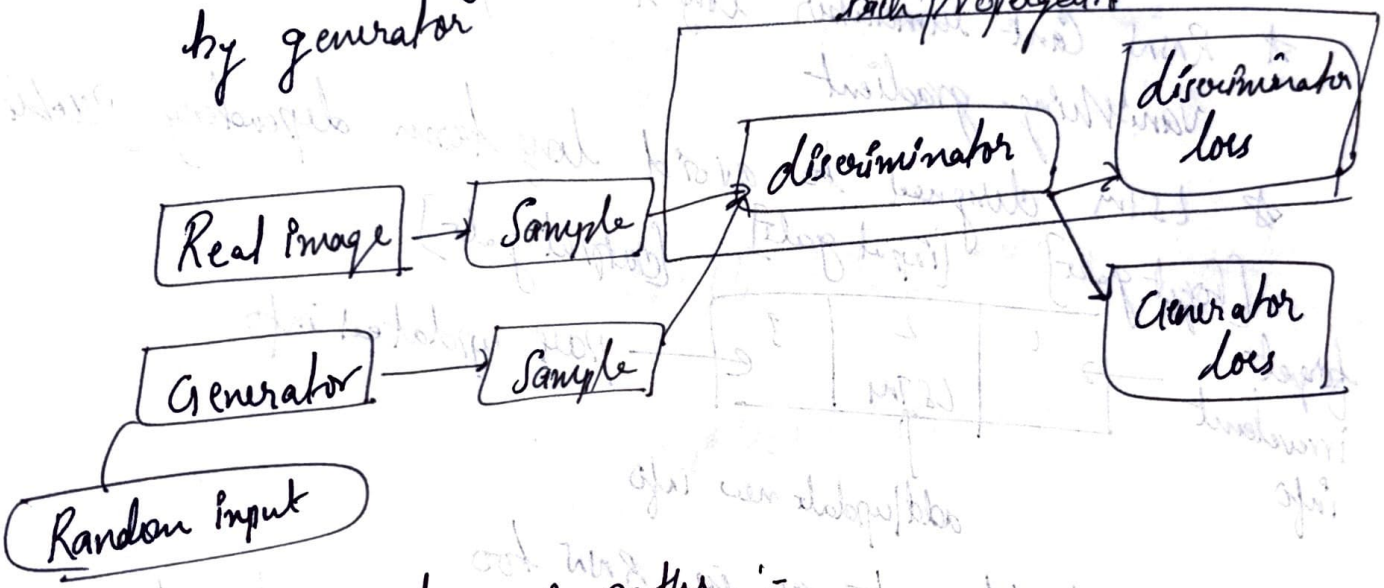
1 Random input
2 Generator network [transforms random input into data instance]
3 discriminator network [classifies generated data]
4 discriminated output.
5 Generator loss [penalizes generator for failing to fool the discriminator]

Discriminator

→ Binary classifier, consists of several CNN layer finally, flattened activation maps mapped to a Probability output to predict if the image is real / fake

→ This want to predict generator output as fake at same time, must predict any real img as real.

→ Tries to distinguish over real and fake data created by generator



data comes from 2 paths :-

Real data :- +ve images

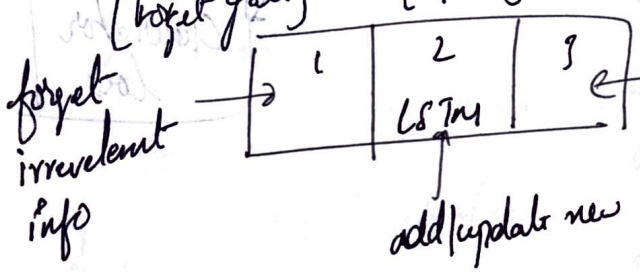
Fake data :-
created by generator
-ve examples

During discriminator training :-

① classifies both real and fake
② loss penalizes the discriminator for mis classifying.
③ discriminator updates its weights through backpropergation from discriminator loss through discriminator netw

③ Long Short-Term Memory [LSTM]

* type of RNN capable of handling long term dependencies.
* Advanced RNN : Allows info to persist
* Capable of handling vanishing problem of RNN
* Remember previous info use it for processing current input.
* RNN Cant remember long term dependencies due to vanishing gradient
* LSTM designed to avoid long term dependency problem



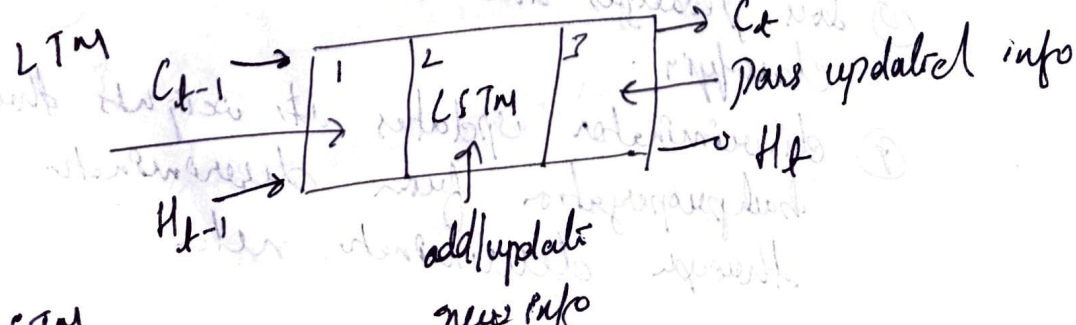
have a hidden gate as in RNN too

H(t-1) : Hidden state of previous timestamp

Ht : " " current "

① Hidden state :- Short term memory

② Cell state :- long term memory



Cell State : Carries info along with all timestamp

Forget Gate :- $f_t = \sigma[(x_t * U_f) + (H_{t-1} * W_f)]$

x_t : current timestamp

H_{t-1} : previous "

U_f : weight of input

W_f : weight matrix associated with hidden state

$f = 0$ forget everything : $C_{t-1} f_t = 0$ $(f_t = 0)$

$f = 1$ forget nothing : $C_{t-1} * f_t = C_{t-1}$ $(f_t = 1)$

Input Gate :- $i_t = \sigma[(x_t * U_i) + (H_{t-1} * W_i)]$

new info :- $N_t = \tanh[(x_t * U_c) + (H_{t-1} * W_c)]$

$-1 < N_t < 1$

info is subtracted from cell state

info is added to cell state at current time stamp

$C_t = (f_t \times C_{t-1}) + (i_t \times N_t)$

Output :- $O_t = \sigma((x_t \times U_o) + (H_{t-1} \times W_o))$

$0 < O_t < 1$

current Hidden State = $H_t = O_t * \tanh(C_t)$

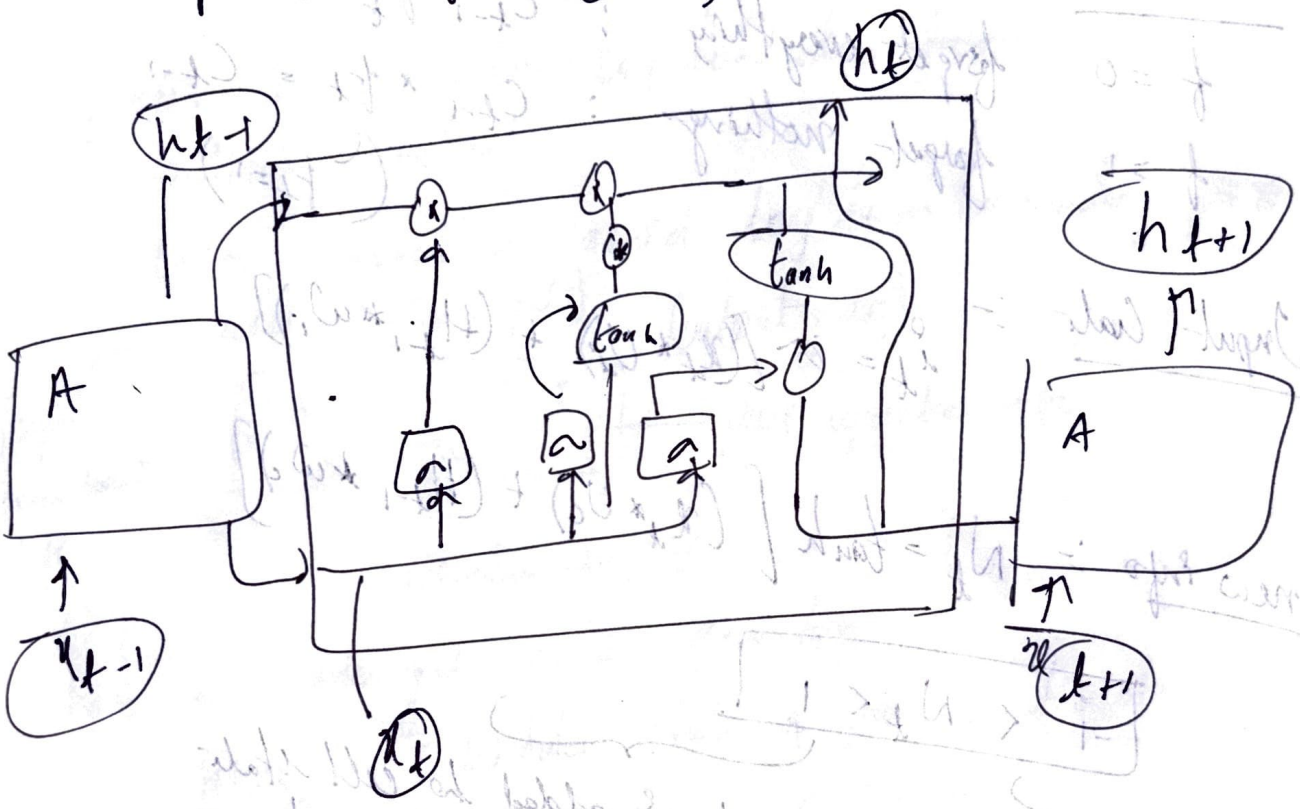
H_t = fn of LTM (C_t)

(Softmax fn)

(Q) need output of current timestamp?

Ans) Apply softmax activ. fn on H_t (Hidden C_t)

Output = Softmax (H_t)



④ RNN Architecture

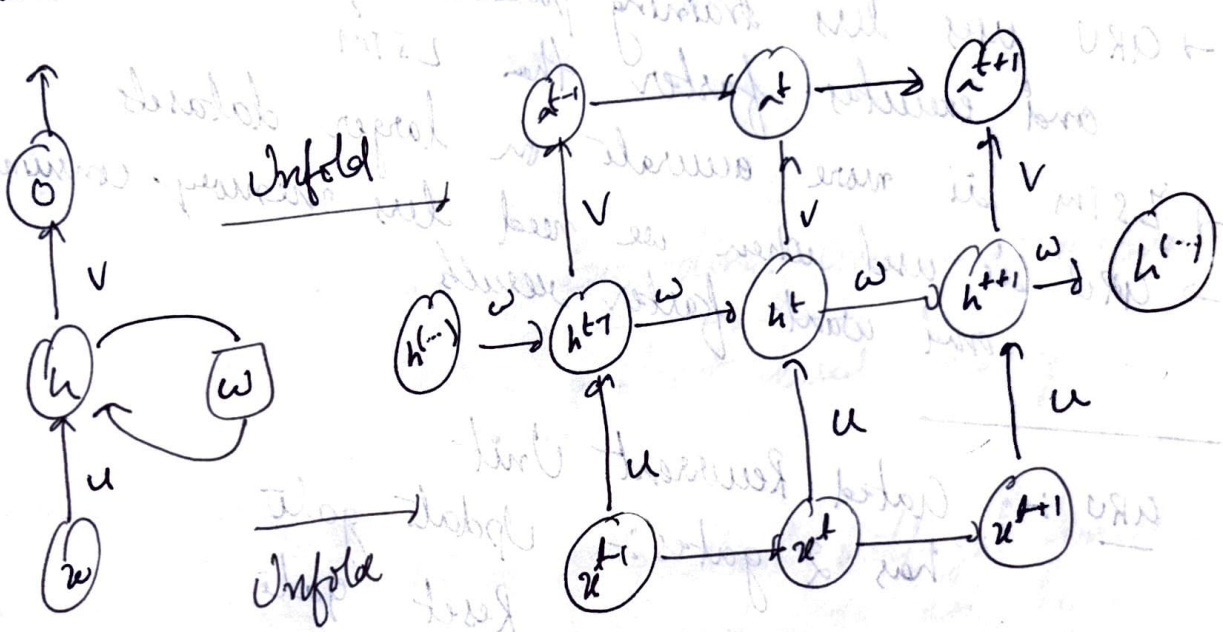
→ Specialised for processing a sequence of data

Used :- Speech and language

→ NLP problem, to predict next word in sequence.

→ Called as recurrent because, perform same task for every element of sequence, output being dependant on on previous computation.

→ They have memory, that capture info that has been calculated



notation RNN

Unfolded RNN

hidden state = "memory"

$h_t = f(U x_t + W h_{(t-1)})$

["tanh", "Relu"]

V : hidden to Output

U : input to hidden

W = weighted matrix

⑤ GRU Vs LSTM

① has 2 gates	① has 3 gates
② no processing of internal memory, doesn't have a output gate (cell state)	② have a output gate (no cell state)
③ Reset gate is applied directly to previous hidden gate state (more complex)	③ Input and output gate is coupled by update gate (less complex)

→ GRU uses less training parameter, hence less memory and executes faster than LSTM

→ LSTMs is more accurate on larger datasets

→ GRU is used when we need less memory consumption and want faster results

---

GRU :- Gated Recurrent Unit
has 2 gates :- Update gate
Reset gate

Update Gate

determines amount of previous info that needs to pass along next state

→ this is very powerful as model can decide to copy all the info from the past and eliminate the risk of vanishing gradient

Reset gate :- needed to decide how much part info is needed to neglect
→ Decided weather previous cell state is imp/not-

→ move the past info to next step
→ multiplied the input vector and hidden state with their weights
→ Calculates element wise multiplication between the reset gate and previous hidden gate state multiple.
→ Summation of above steps.
→ and non linear steps applied and next sequence is generated

---

⑥ Vanishing Problem & Exploding Gradient Problem

When training DNN using gradient based learning and back propagation.

→ Occurs when derivative / slope of gradient will get smaller and smaller as we go backward with every layer during backpropag..

* then training take too much time else completely stops
Occour with Sigmoid / tanh activation fn : 0 – 0.25
0 – 1

* Can avoid using RELU activation fn as its gradient is 0 : zero & -ve input
1 : +ve input-

Exploding Gradient :-

→ Occurs when gradients slope/derivative will get larger and larger as we go backward with every layer during back propagation

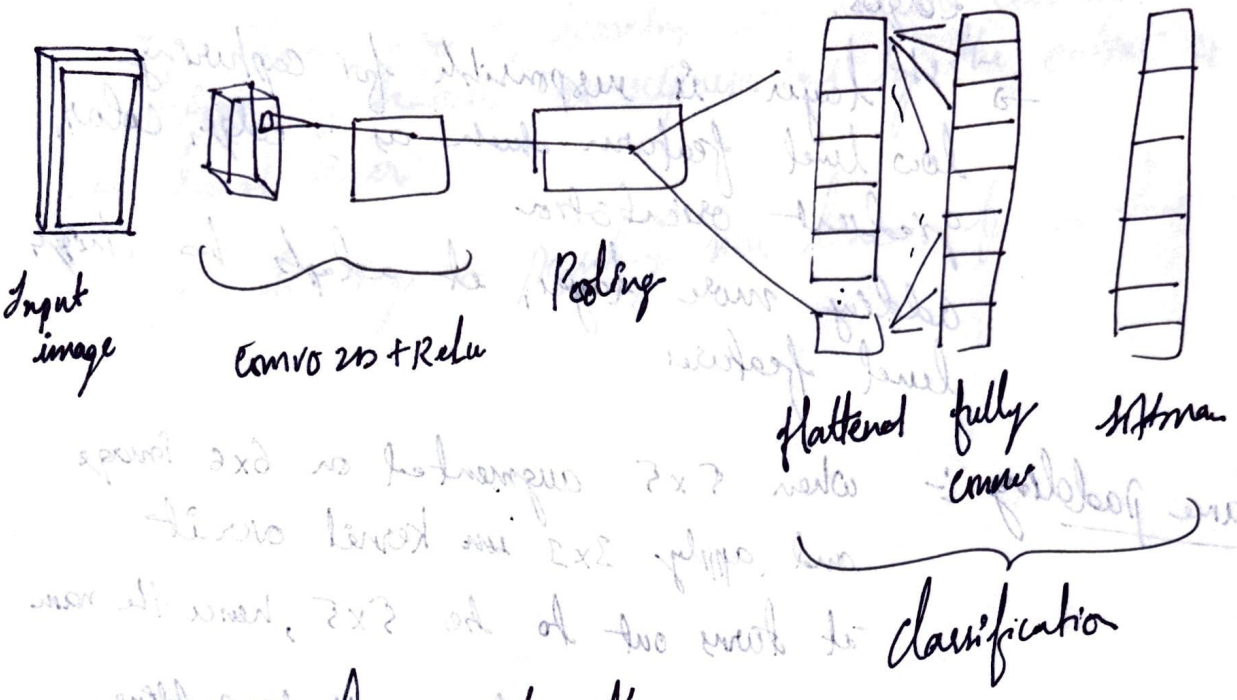
→ exactly opp to vanishing gradients

→ happens because of weights not because of Activation function

→ due to high weight values derivative will also higher so that the new weights a lot to the older weight the gradient will never converg

So, it may result in oscillating around minima and never come to global minima point

⑦ Architecture of CNN :-



CNN :- deep learning algorithm

→ takes input an image, assign importance to various aspects / objects and able to differentiate one from other

→ Preprocessing Required is lower than other classification algo.

Architecture → Analogous connectivity pattern of neuron in human brain

→ Performs better fitting to image dataset due to reduction in the number of parameters involved and reusability of weight

Convolution layer / Kernal :-

element involved in carrying out convolution operation is 1st part of convolutional layer.

Obj :- extract high level features such as edges.

→ 1st layer is responsible for capturing low level features such as :- edge, color, gradient orientation
adding more layer, it adapts to high level features

Same padding :- when 5x5 augmented on 6x6 image and apply 3x3 on kernal over it
- it turns out to be 5x5, hence the name

Valid padding :- Performing same operation without padding

Pooling layer :- Reduces spatial size of convoluted feature
decreases computational power required to process the data through dimensionality reduction
Used while extracting dominant features
"Rotational & positional invariant"

Max pooling	Average pooling
Perform noise suppressant denoising dimensionality reduction	dimensionality red.

└ a lot better than Average Pooling