ILSVRC

03 October 2022 12:38

Imagenet challenge → 2010 → Dataset → subset → 1 million → 10 → 20,000 classes → 1000 classes

14 crore

100 image

Machine learning → feature extraction → SIFT, HOG

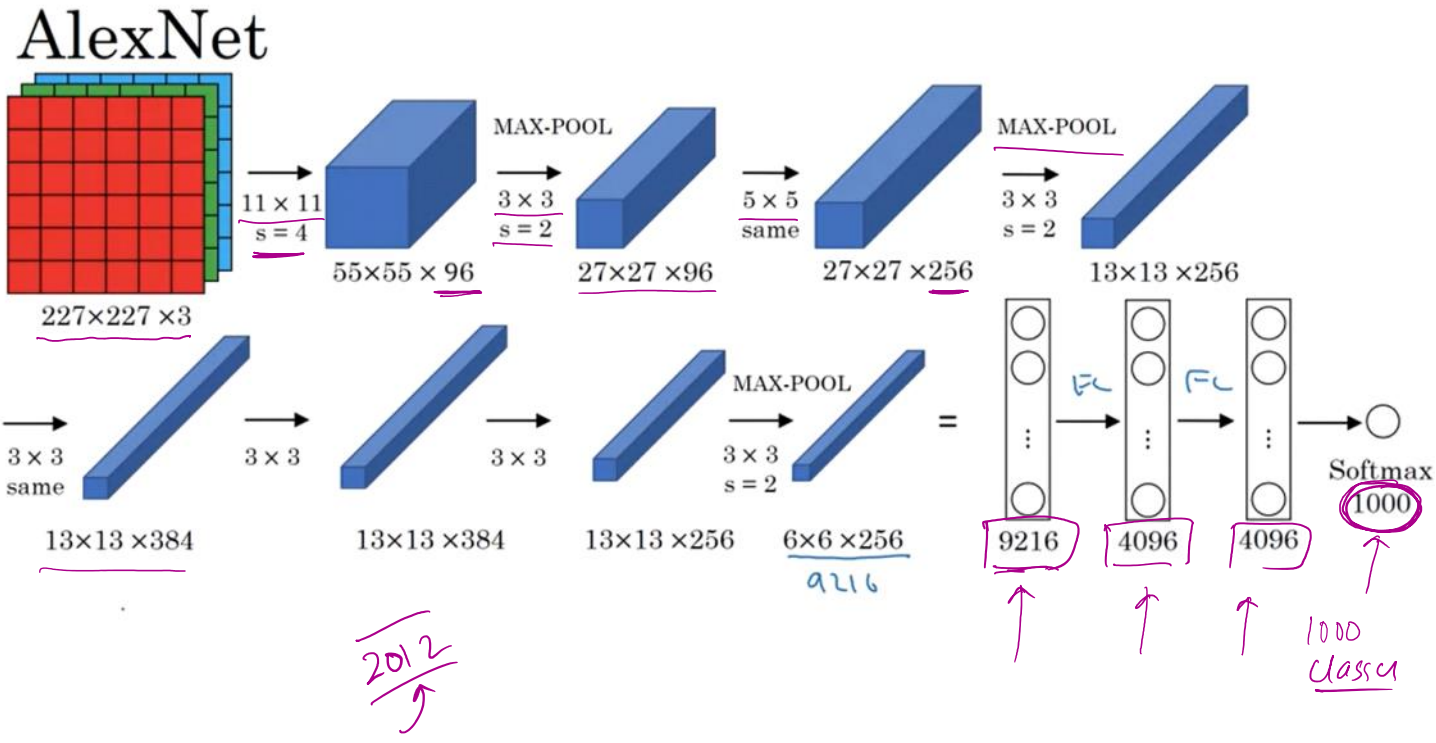
2010 → 28%.

2011 → 25%.

2012 → 16%. → CNN

2013 →

2012 → DL → AlexNET → CNN → relu, Deep, GPU

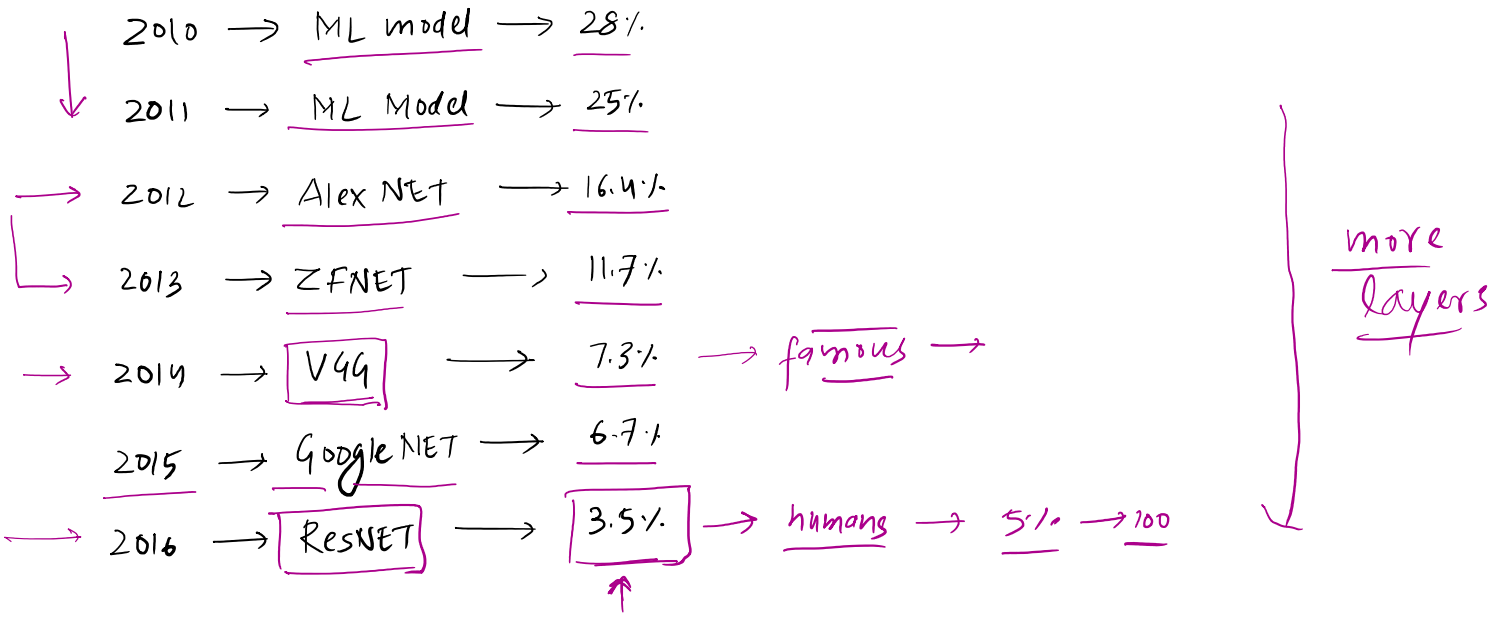


[Krizhevsky et al., 2012. ImageNet classification with deep convolutional neural networks]

Andrew Ng

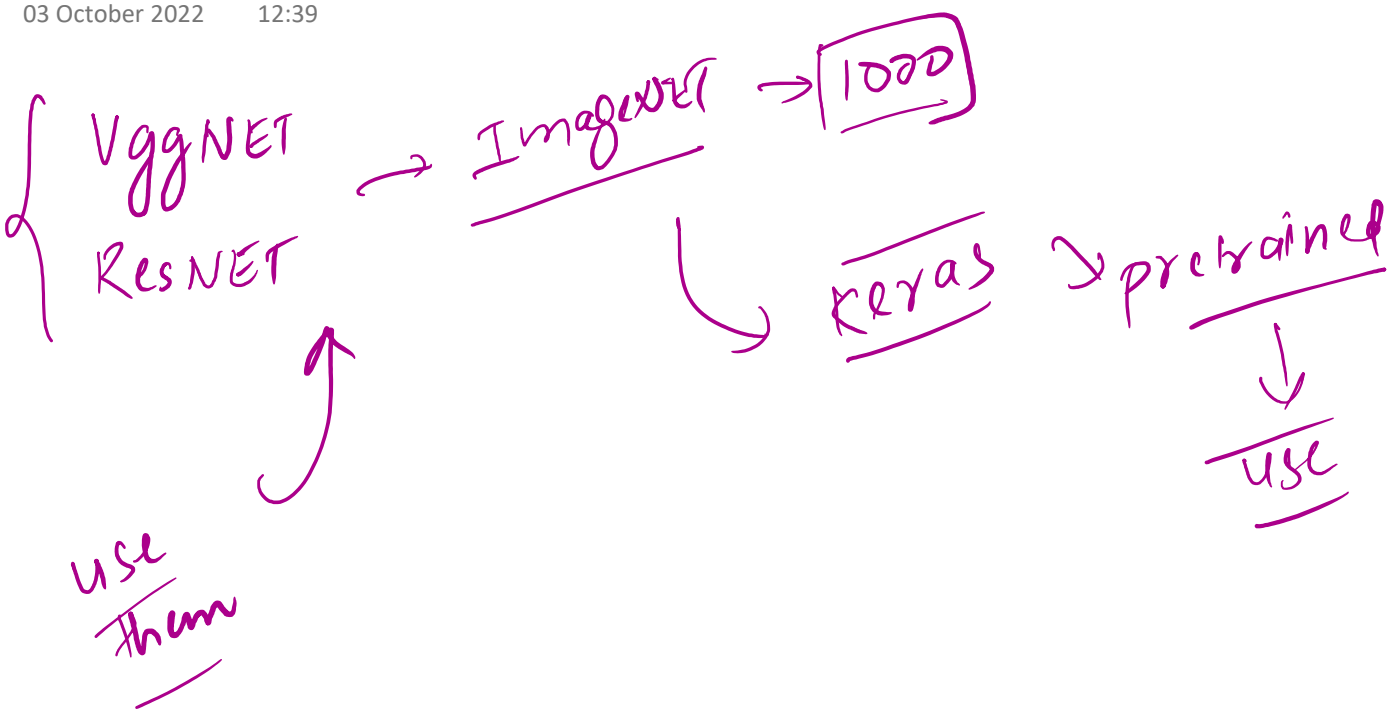
Famous Architectures

03 October 2022 12:39



Idea of Pretrained Models

03 October 2022 12:39

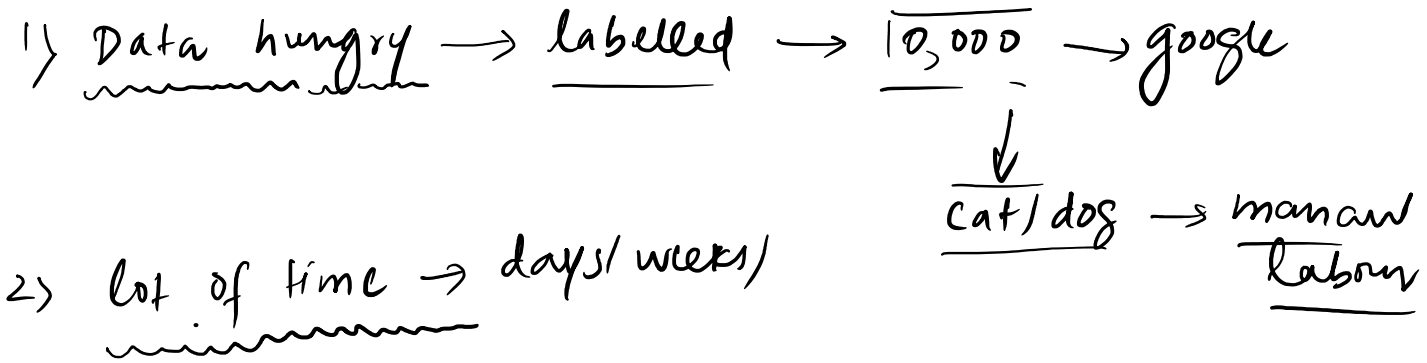


Keras Demo

03 October 2022 12:40

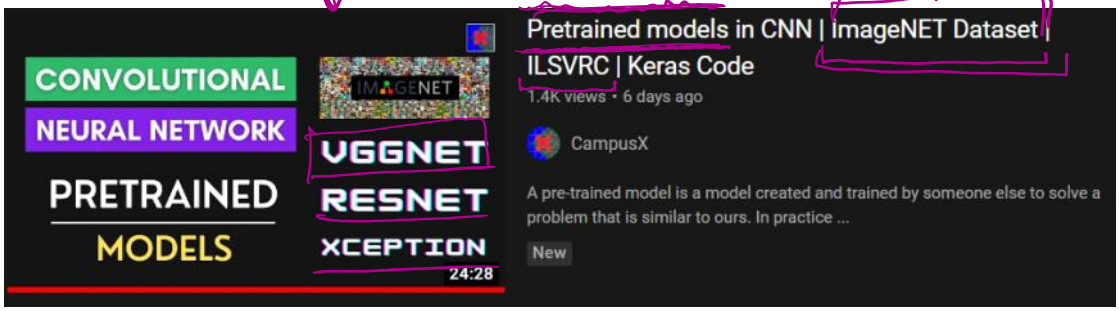
Problem with training your own model

10 October 2022 10:48



Using Pretrained Models

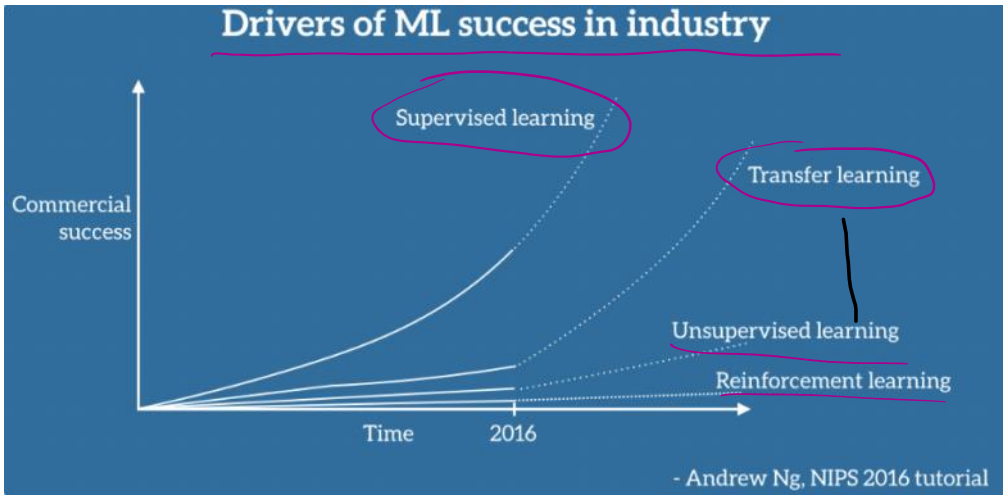
10 October 2022 10:48

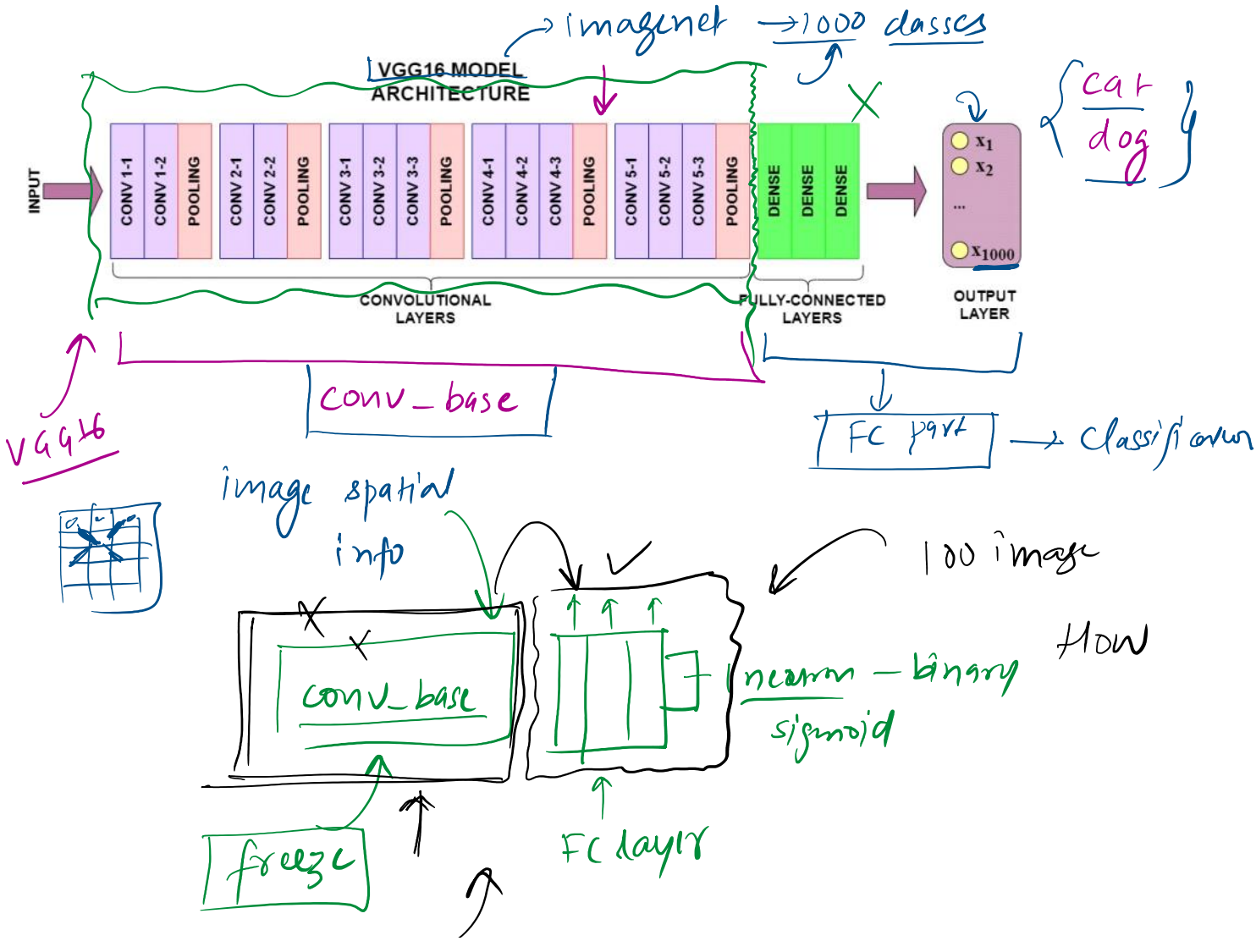


Transfer Learning

10 October 2022 10:49

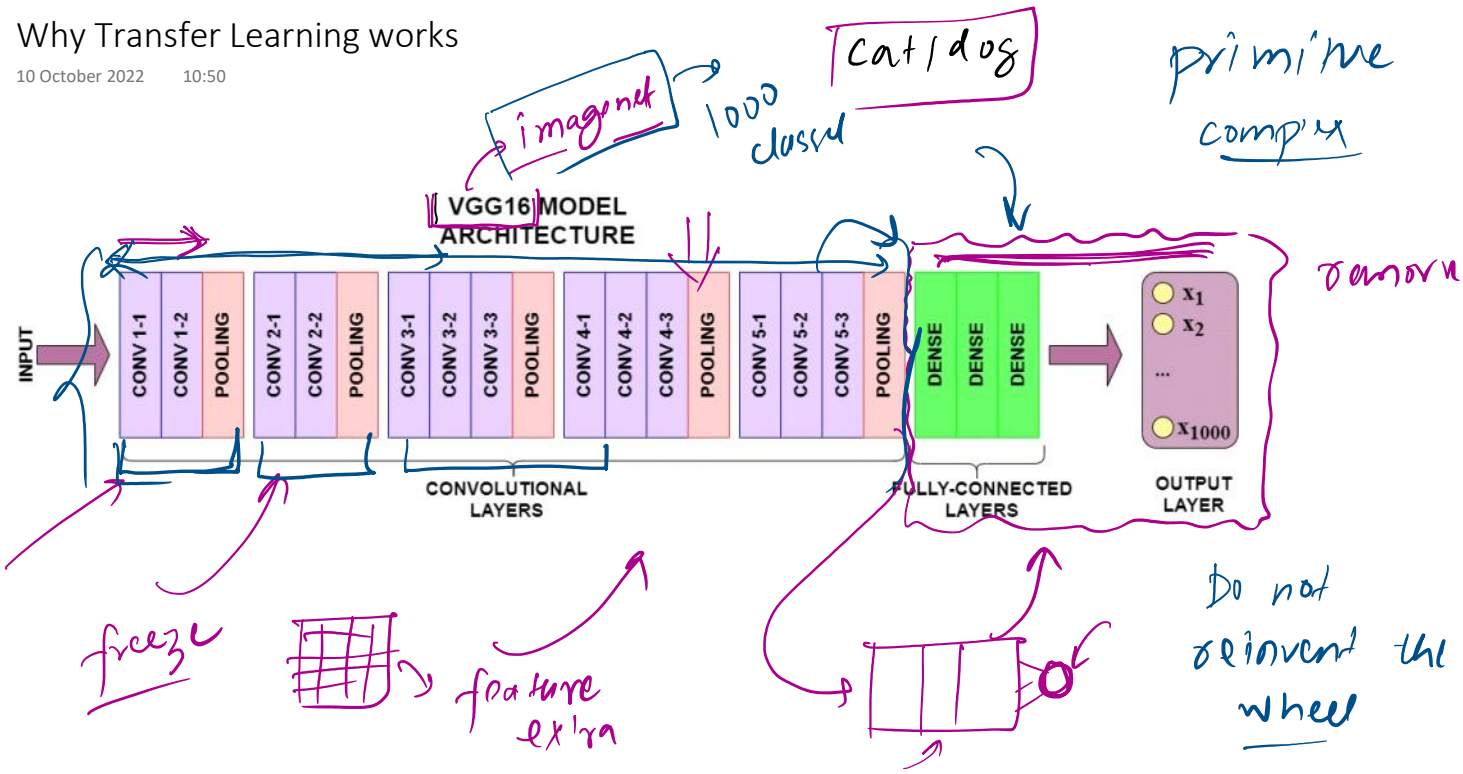
Transfer learning is a research problem in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem.





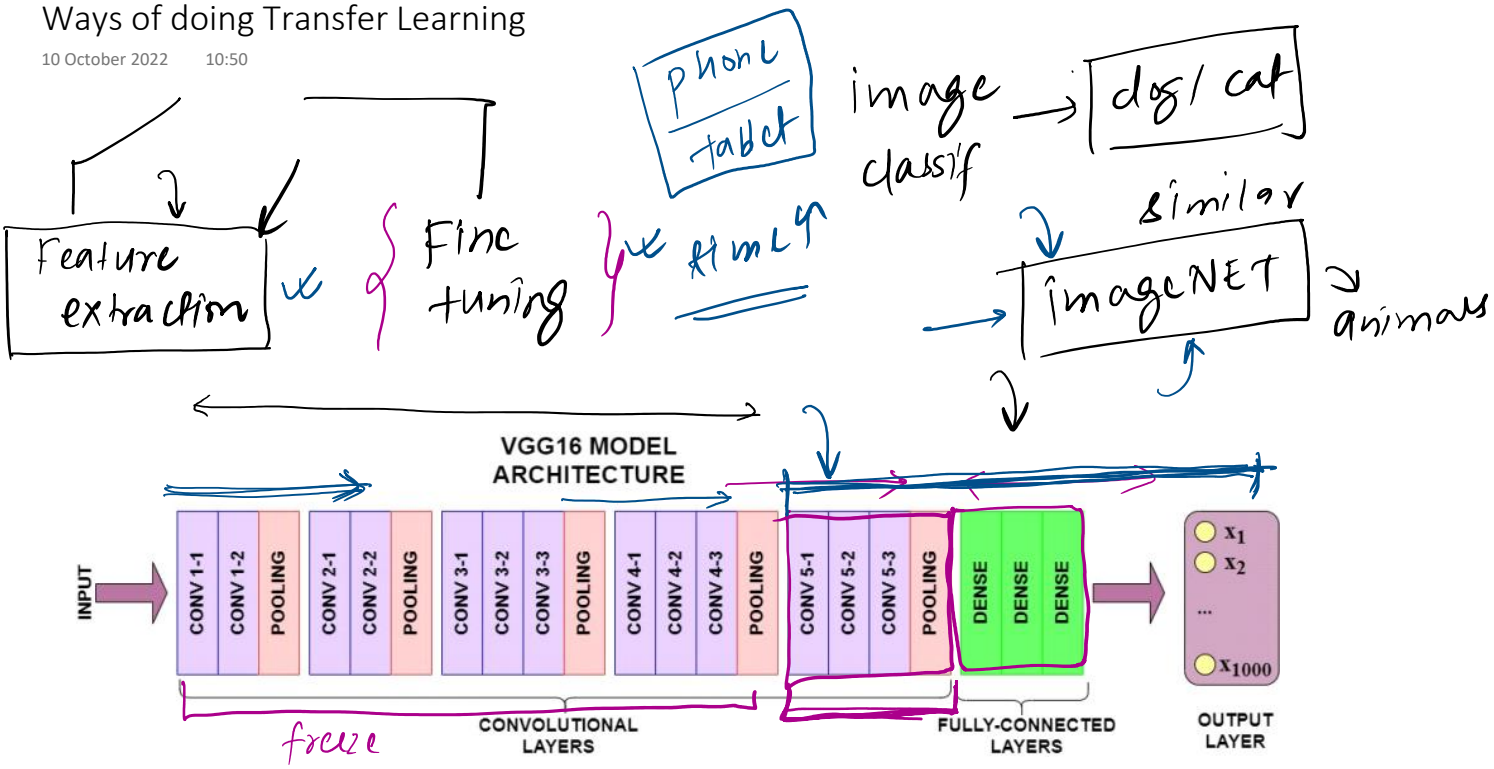
Why Transfer Learning works

10 October 2022 10:50



Ways of doing Transfer Learning

10 October 2022 10:50



Code

10 October 2022 10:50

Problem with Sequential Model

14 October 2022 16:00

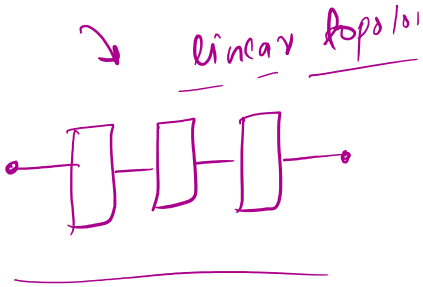
ANN ↔ CNN

↳ Sequential – Keras

→ 1 input

→ 1 output

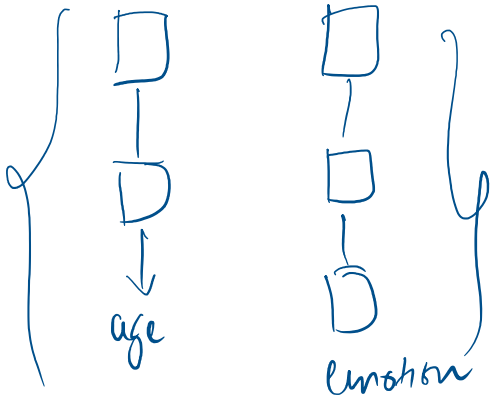
→ linear

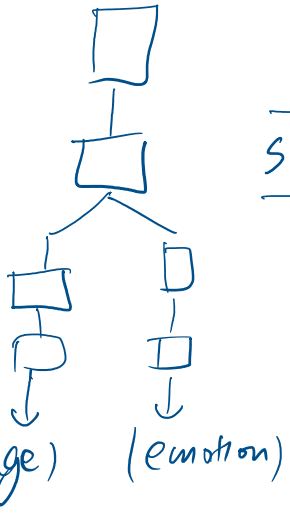


Example →  → image dataset → human faces → 25000 images

→ age?

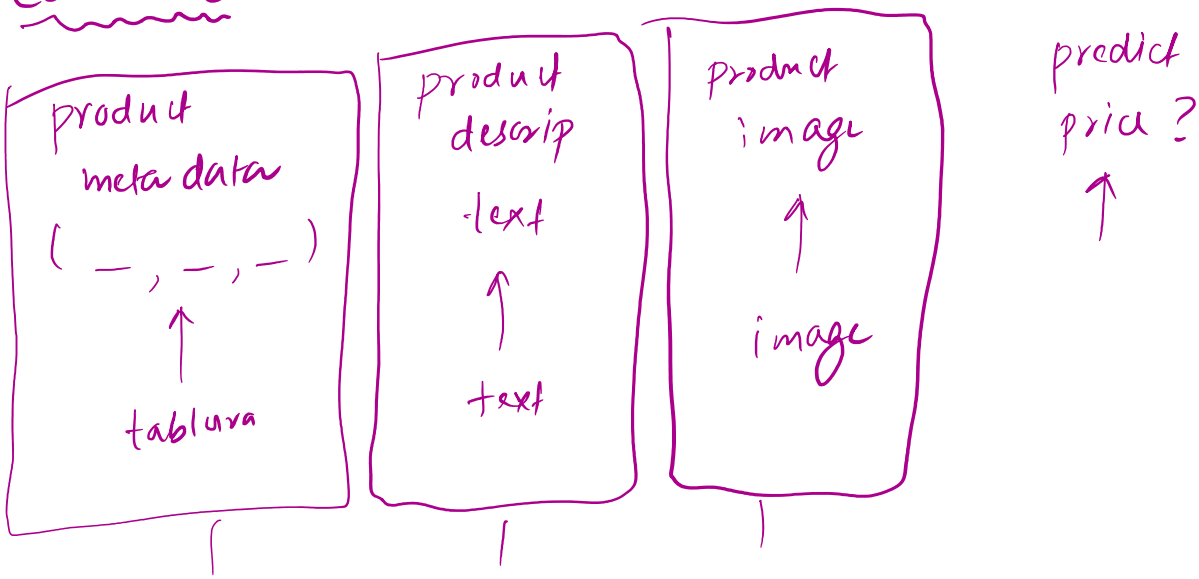
→ emotion → happy / sad / angry

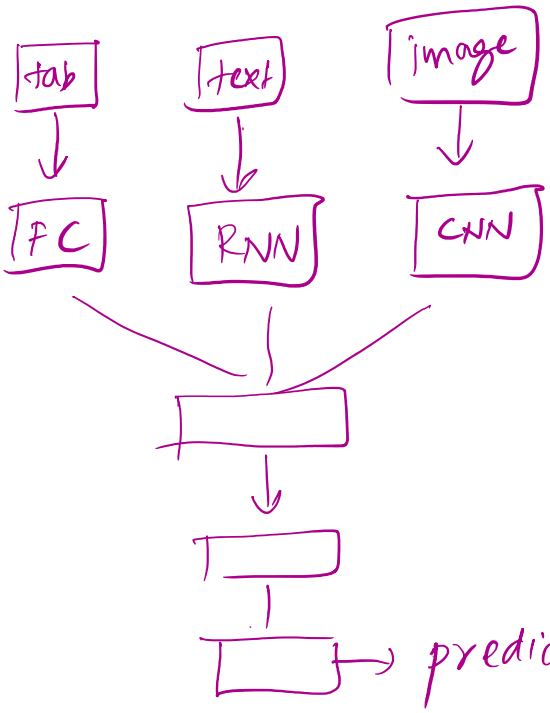




Sequential API

ecommerce





non-linear

Sequential X

Non-linear

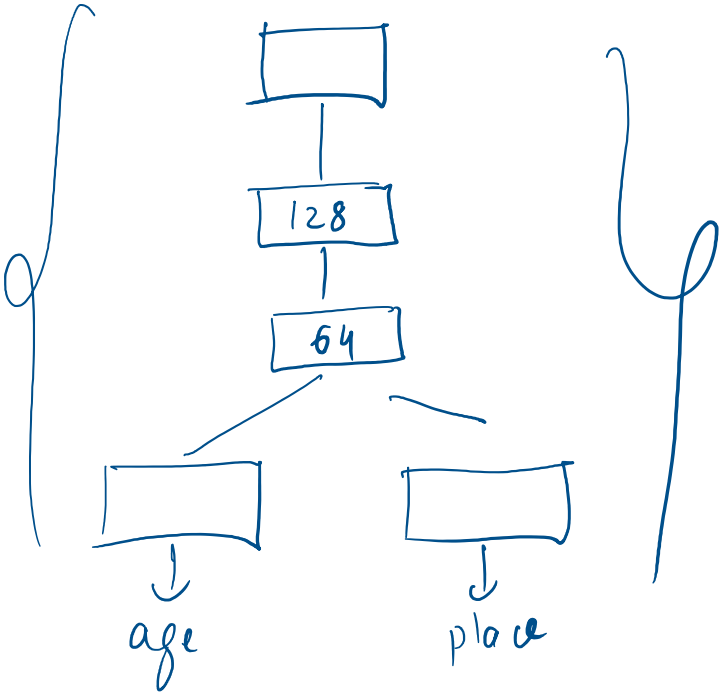
Functional API

A Simple Example

14 October 2022 16:01

| yearly salary | height | marital status |

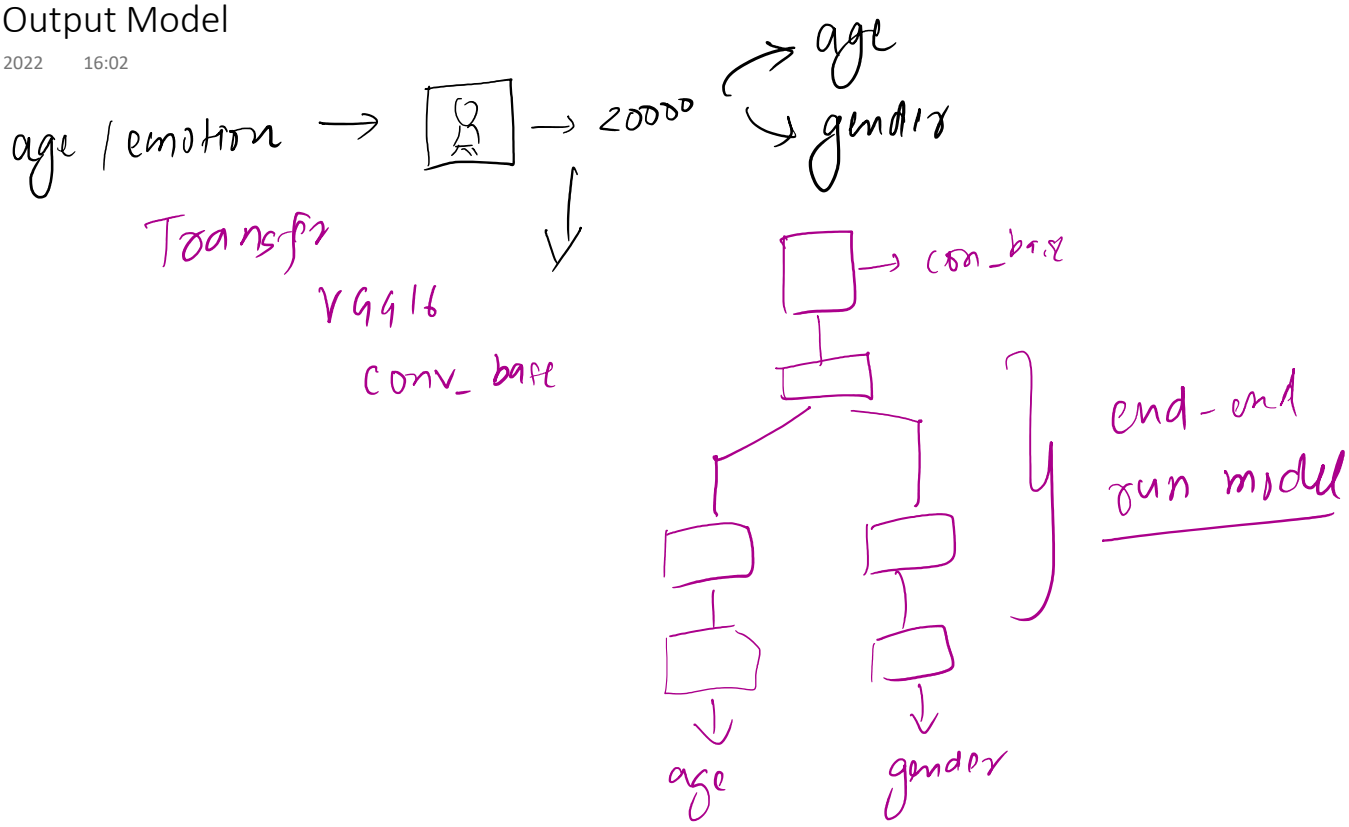
3 cols → age
→ delhi/ mumbai



Keras functional api

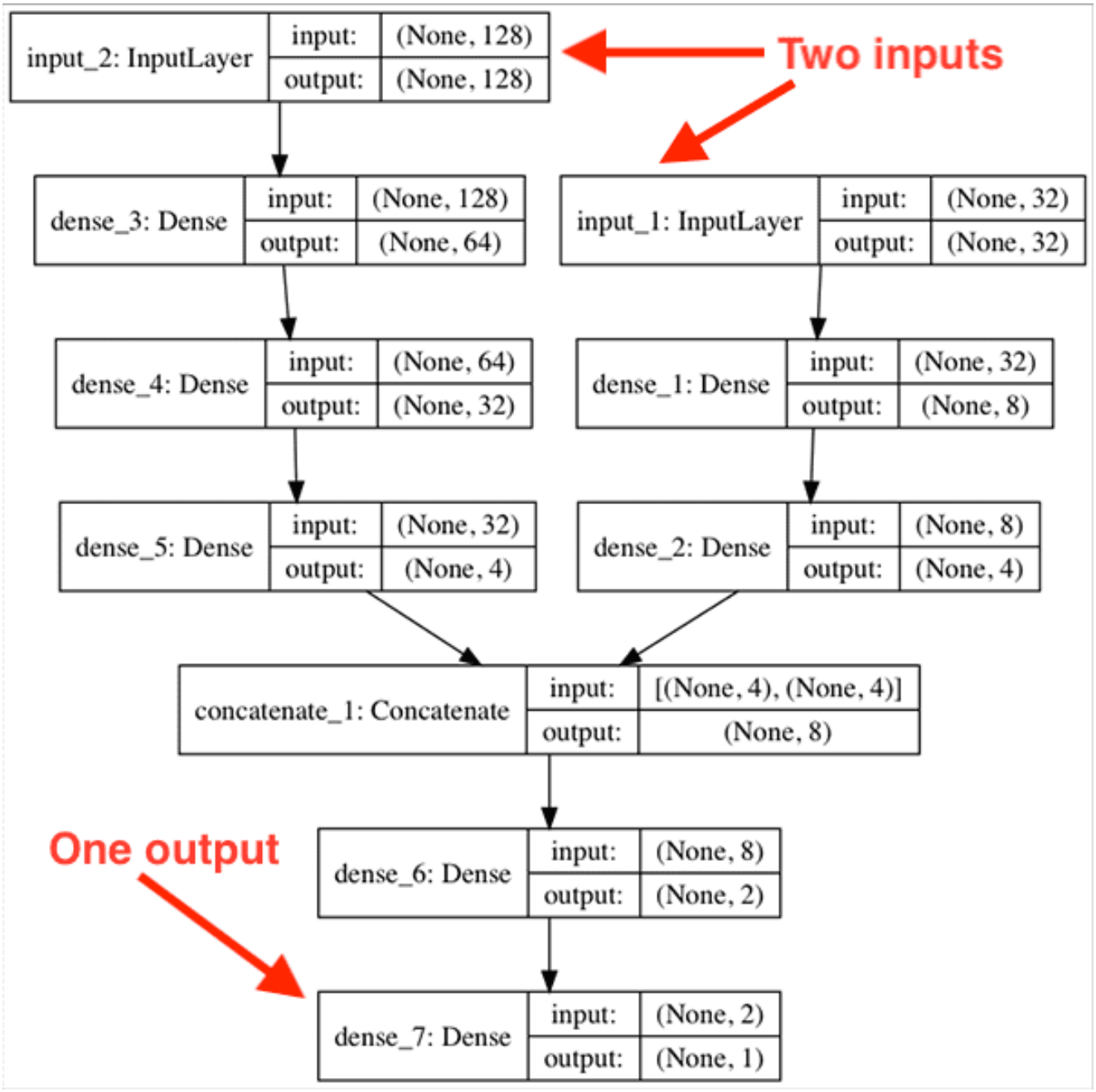
Multi Output Model

14 October 2022 16:02



Multi Input Model

14 October 2022 16:02



Shared Layers Model

14 October 2022 16:02

Sequential Data

22 October 2022 13:09

ANN → tabular data

CNN → images

RNN → Recurrent NN

↳ is type of sequential model to work on sequential data

iq | marks | gender | placement

placement → does not matter

iq, marks, gender → O → No

eg → text → sequential data

Hi my name is Nitish

RNN → NLP → ML

NLP → RNN

CNN → images → computer vision

Time series → sequential

2001 2002 2003

Speech → sequential

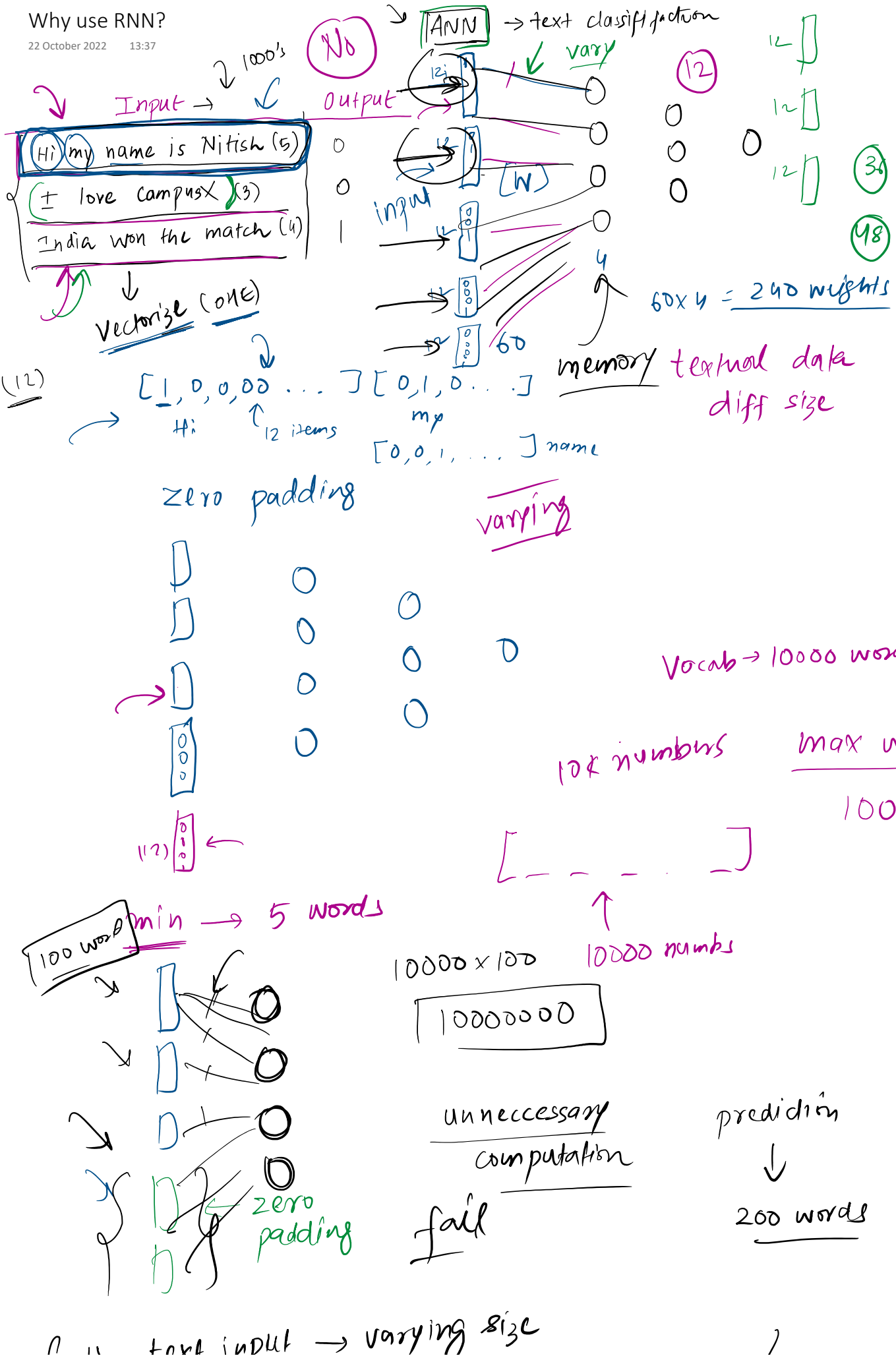
DNA sequence

RNN

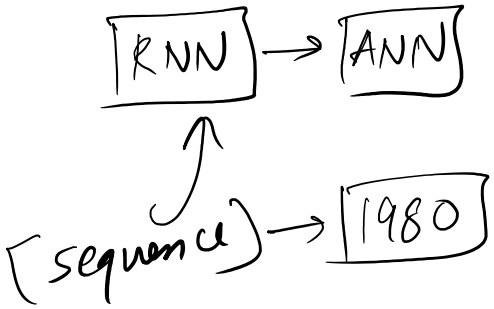
- ANN/CNN
- RNN Why?
- Application → RNN
- Roadmap

Why use RNN?

22 October 2022 13:37



1) text input → varying size
2) zero padding → unnecessary computation
3) Prediction problem
4) Totally disregarding the sequence info

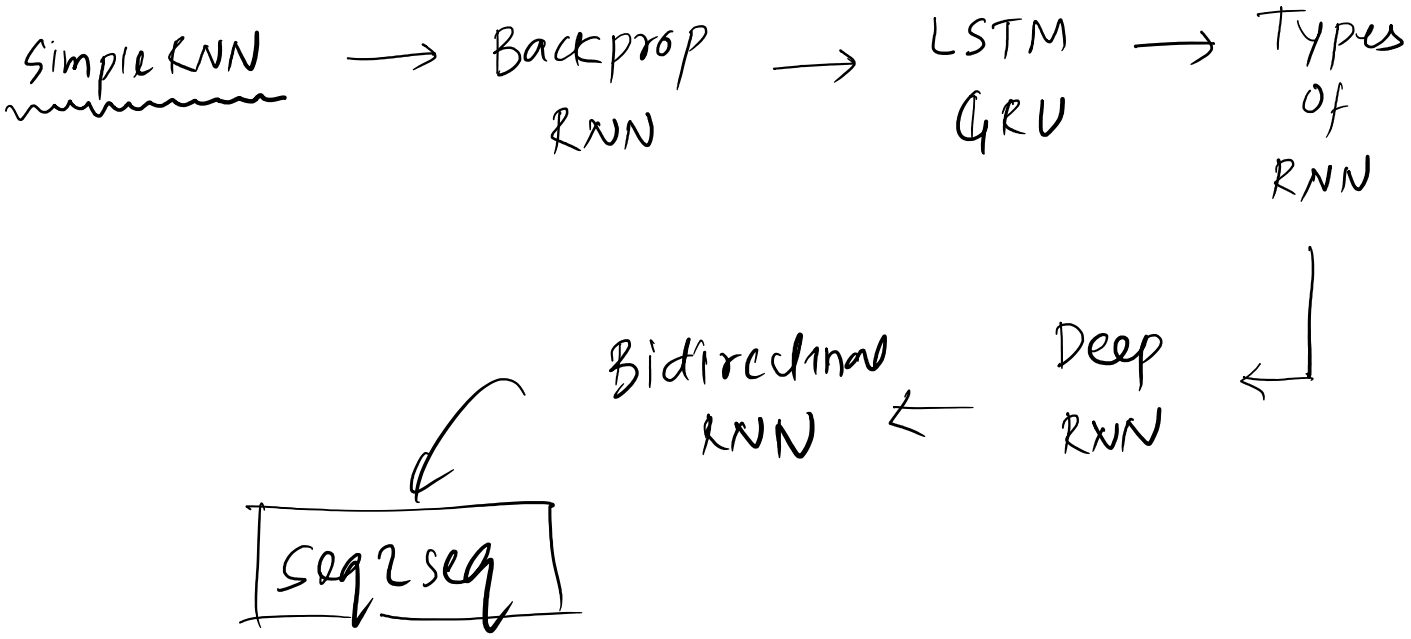


RNN Applications

22 October 2022 13:37

Roadmap

22 October 2022 13:37



Why RNNs?

29 October 2022 13:30

zero padding → cost of compu ↑

1) Sequence → any length → ANN → fixed input size

movie reviews

RNNs

NNs

same

2) sequence contains some meaning

ANN → input

memory feature

RNN architecture

RNN forward prop → prediction

↳ input → output

Codes → solidify

Data for RNN

29 October 2022 13:30

[(timesteps, input_features)]

t=1, t=2, t=3

Review	Sentiment
movie was good	1
movie was bad	0
movie was not good	0

(4)

review 1

[[10000], [01000], [00100]]

vocab → 5 word

movie was good

[10000] [01000] [00100]

[00010] [00001]

bad not

(3,5) → (# of features, no of timesteps)

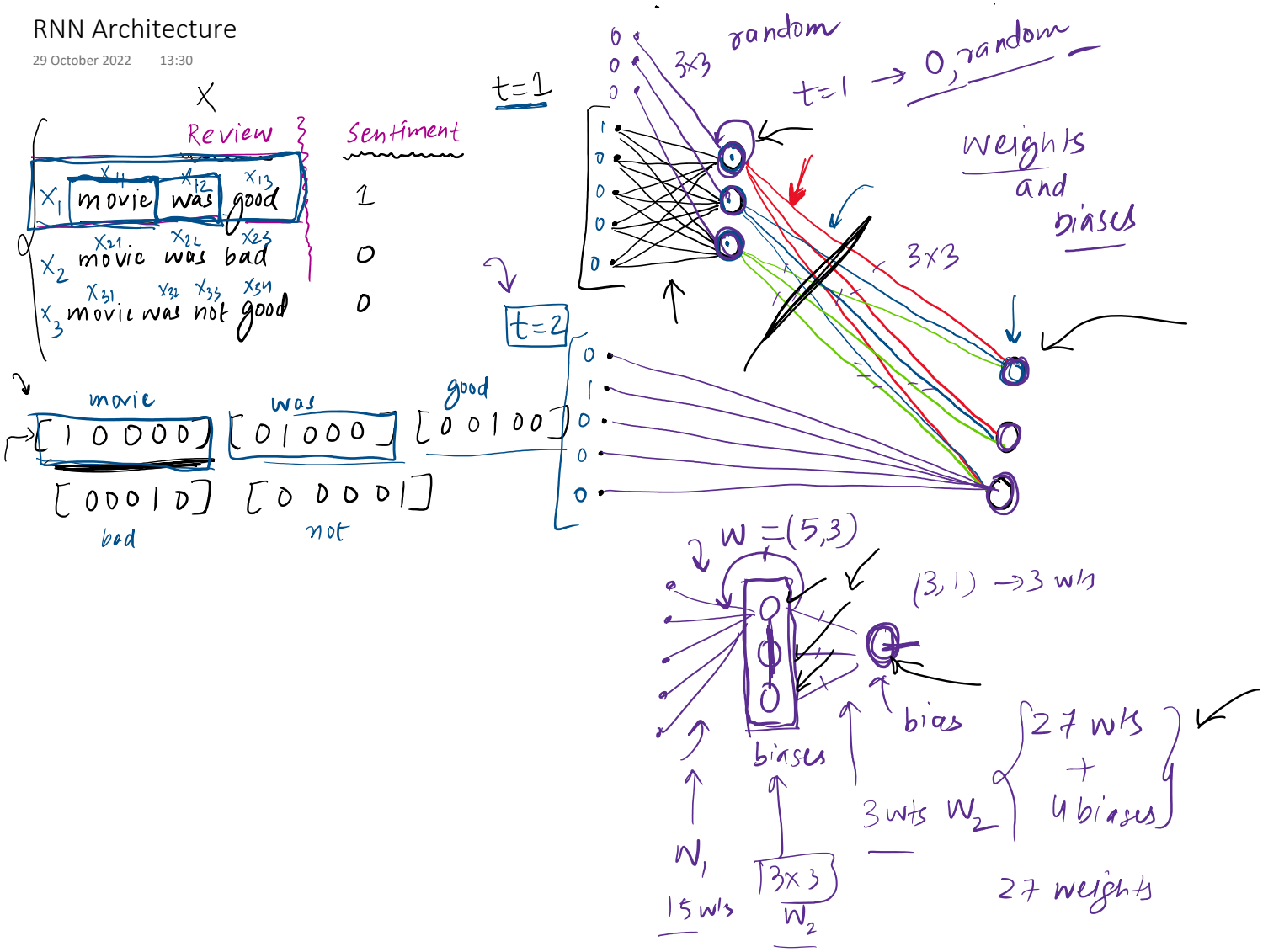
(3,5) (4,5)

Keras → SimpleRNN → batch size, timesteps, input_features

(3, 4, 5) → 3D tensor → RNN

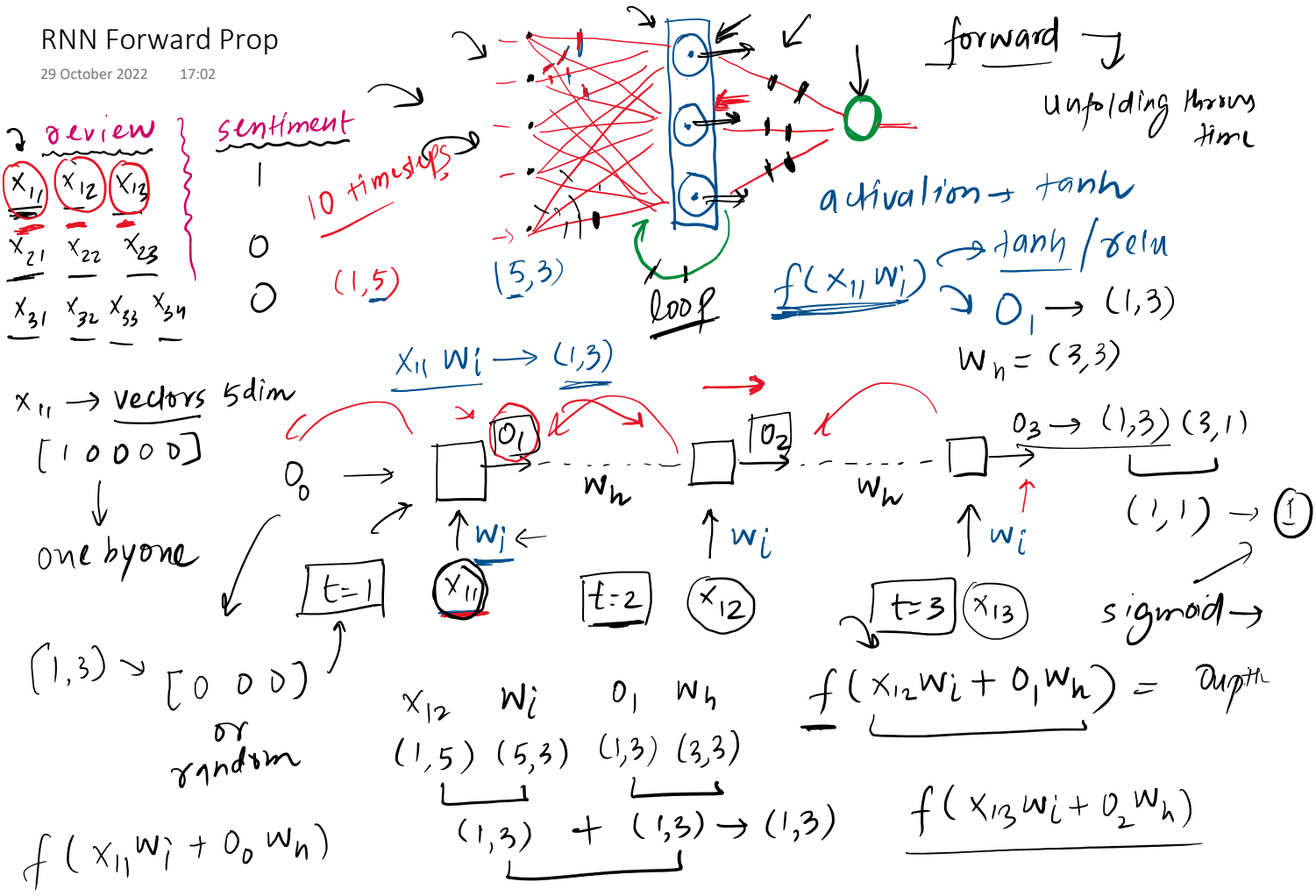
RNN Architecture

29 October 2022 13:30



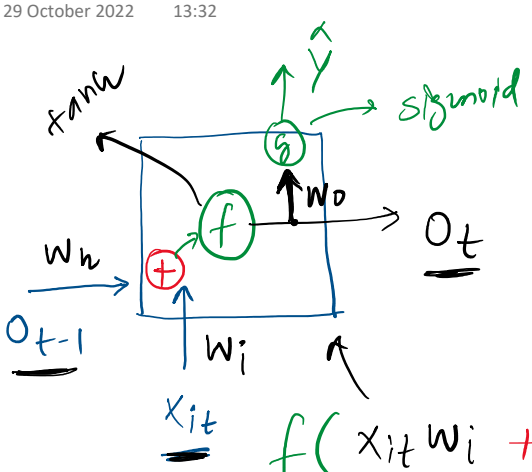
RNN Forward Prop

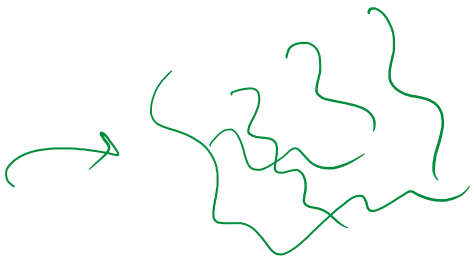
29 October 2022 17:02



Simplified Representation

29 October 2022 13:32





$\hat{y} = g(O_t W_o)$

$f(X_{it} W_i + O_{t-1} W_h) \rightarrow$ vector $\rightarrow O_t$

i = row #

t = time step

$X_{11} \rightarrow$

Hello how are you?

Code

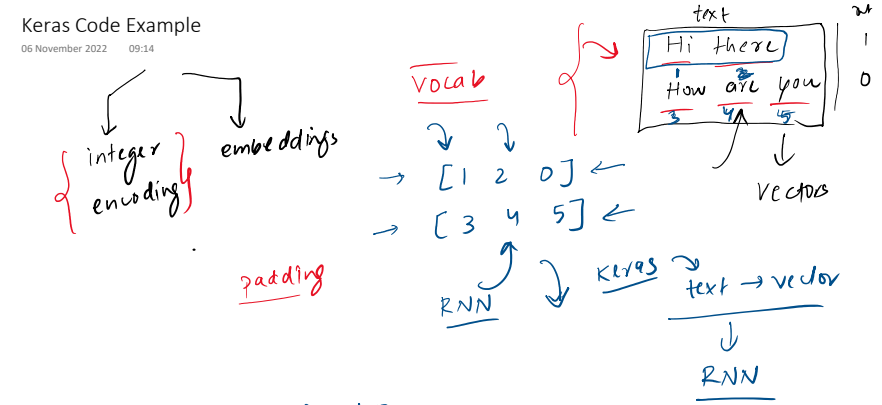
29 October 2022 13:32

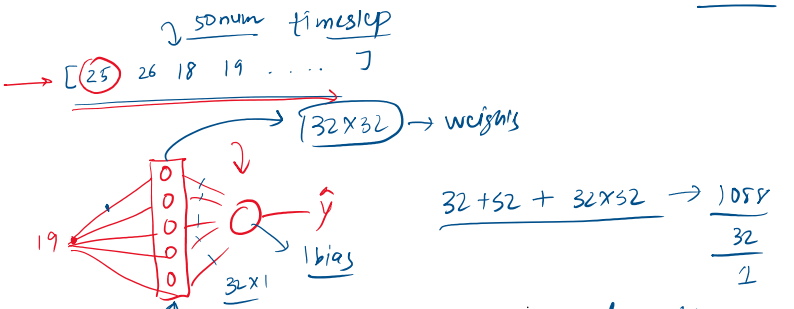
State and Memory

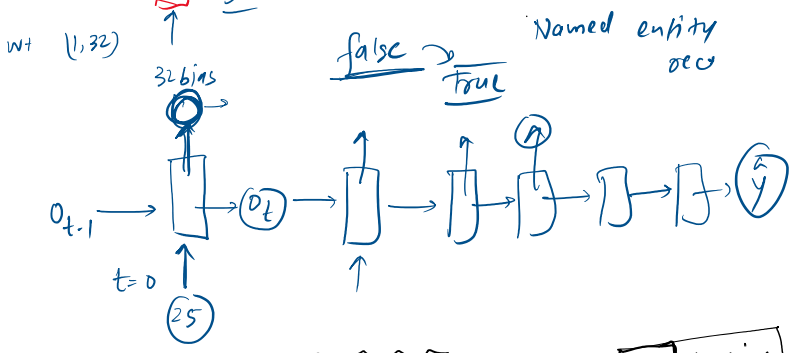
29 October 2022 13:33

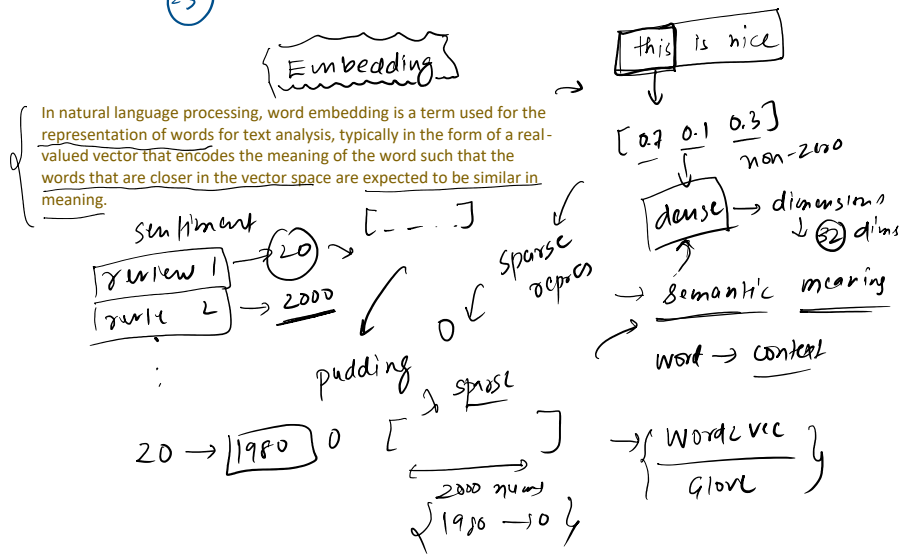
Keras Code Example

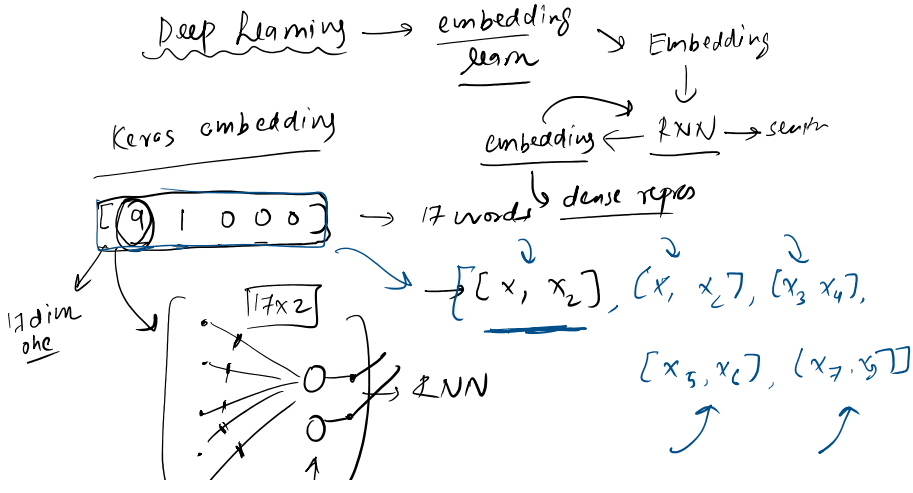
06 November 2022 09:14

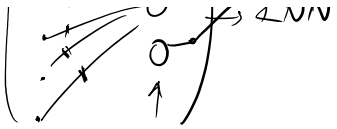




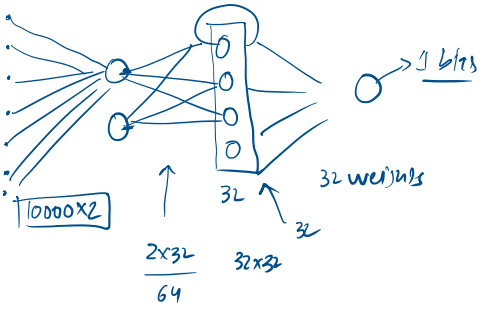








17 nodes



population in 9th year = x

$x + 10\%\text{ of } x = 10000$

$x + 0.1x = 10000$

$1.1x = 10000$

$x = \frac{10000}{1.1}$

$\frac{x-1}{x} + \frac{1}{2}\left(\frac{x-1}{x}\right)^2 + \frac{1}{2}\left(\frac{x-1}{x}\right)^3 + \frac{1}{2}\left(\frac{x-1}{x}\right)^4 + \cdots$

t(n-1)

+ 2 +

+ (n-1)

50 → 500

A

A[35] → t sec

A[35] →

35

1 x 4 x35

x

1 25 50 100

12

O(n)

$O(n^2)$ → nested loops

input → 10 (100x100)

time → $(x)^2$ 4

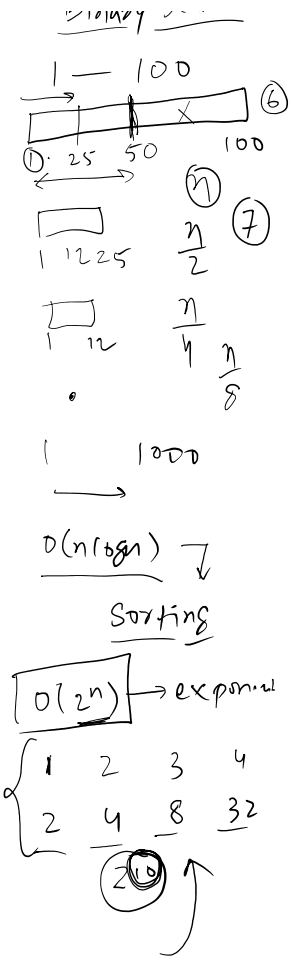
O(log(n))

Binary search

1 — 100

→ (6)

$\sqrt{(x_1}$


Binary
1 — 100
100
25 50
n/2
n/4
n/8
1000
O(nlogn)
Sorting
O(2^n) → exponential
1 2 3 4
2 4 8 32
for i in scan
O(n)
for j in (n^2)
O(n)
O(n+n) →
O(2n)
→ O(n)
O(n+n^2) O(n^2)
for i in arr:
for j in
25 → '25'
str()
n = 345%10
digits[5]
5 + ''
= '5'
345//10 → 34 != 0
34//10 → 3

34 // 10 → 4

4 + '5' →
45

34 // 10 → 3

3 != 0

3 // 10 => 3

digits[3]

3 + '45'
= '345'

Loop n → 10 times

345 = n ⇒ 345 0

3 → 4

inp → 10
time → 1 → log

O(n) → O(n+n)
O(n) O(2n) → O(n)

O(n) → O(n*n)
O(n) O(n²)

O(n)
O(n)
O(10000000)

n^2 10000000
×
$O(n^2)$

log(2)/2
1 → $\frac{n}{2}$

4 $\frac{n}{2}$ 2n

O(n) n=100

50, 100 $\frac{n}{2}$

2 → 100=n $\frac{n}{2}$

j=1 → 2
j=2 → 4 2-100
j=3 → 8
j=4 → 32

$\frac{n}{2} \times \log n$

n log n

O(1) → constant

n = 345

3+4+5 → 12

5

3 4 4

3 3

0

3450 → 4

log

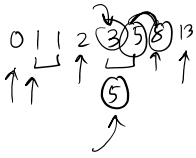
inp → 10 100

out → 1 2 3

fibonacci

↳ function

↳ recursion



n=1 n=0 → 1

fib(n)

↓

function calls

input → n → 10

#fcalls →

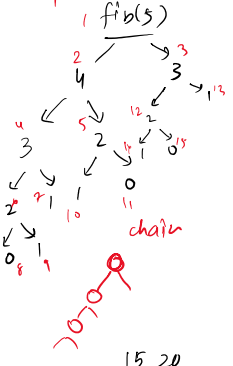
fib(3) n=3 1

fib(2) 2 fib(1) 3 → 1

fib(1) fib(0)

1 1

exponential → bad



O(2^n) 15 20 50

input 1 2 3 4

1 2 4 8 16

$O(2^n)$ (50)

input 1 2 3 4

t1 2 4 8 16

$n = 50, 100, 500$

↑ weeks

exponential

↳ days/weeks

subset → O(?)

power set

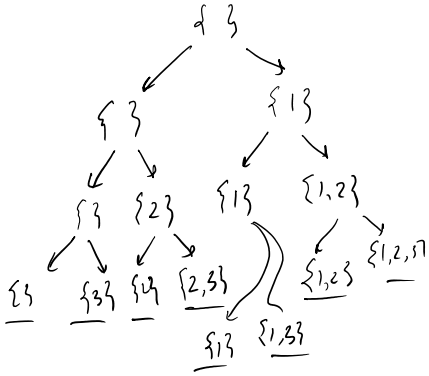
{1,2} → {{},{1},{2},{1,2}}

{1,2,3} →

{},{1},{2},{3},{1,2},{2,3}

{3,1} {1,2,3}

{1,2,3}



reduce → divide → log

Increase → multi → exp

→ exponential

{1,2} → 4 $2^2 = 4$

{1,2,3} → 8 $2^3 = 8$

$2^4 = 16$

$O(2^n)$

O(?)

$$T(n) = \begin{cases} 3T(n-1) & \text{if } n>0 \\ 1, & \text{otherwise} \end{cases}$$

$n > 0$

$T(n) = 3T(n-1)$

$= 3[3T(n-2)]$

$= 3^2 T(n-2)$

$= 3^2[3T(n-3)]$

$= 3^3 T(n-3)$

$= 3^n T(n-n)$

$= 3^n T(0)$

$T(n) = \boxed{3^n} \rightarrow O(3^n)$

$$T(n) = \begin{cases} 2T(n-1)-1 & \text{if } n>0 \\ 1, & \text{otherwise} \rightarrow \text{constant} \end{cases}$$

$=$

$T(n) = 2T(n-1)-1$

$= 2[2T(n-2)-1]-1$

$= 2^2 T(n-2) - 2 - 1$

$= 2^2[2T(n-3)-1] - 2 - 1$

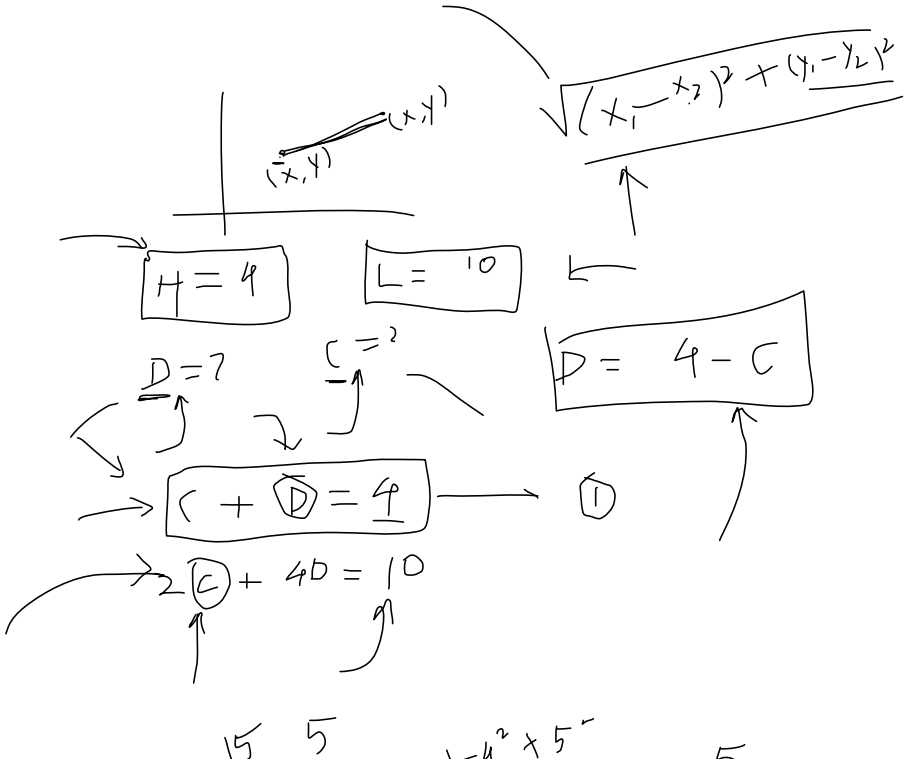
$= 2^3 T(n-3) - 2^2 - 2^1 - 2^0$

$= 2^n T(n-n) - 2^{n-1} - 2^{n-2} \ldots - 2^1 - 2^0$

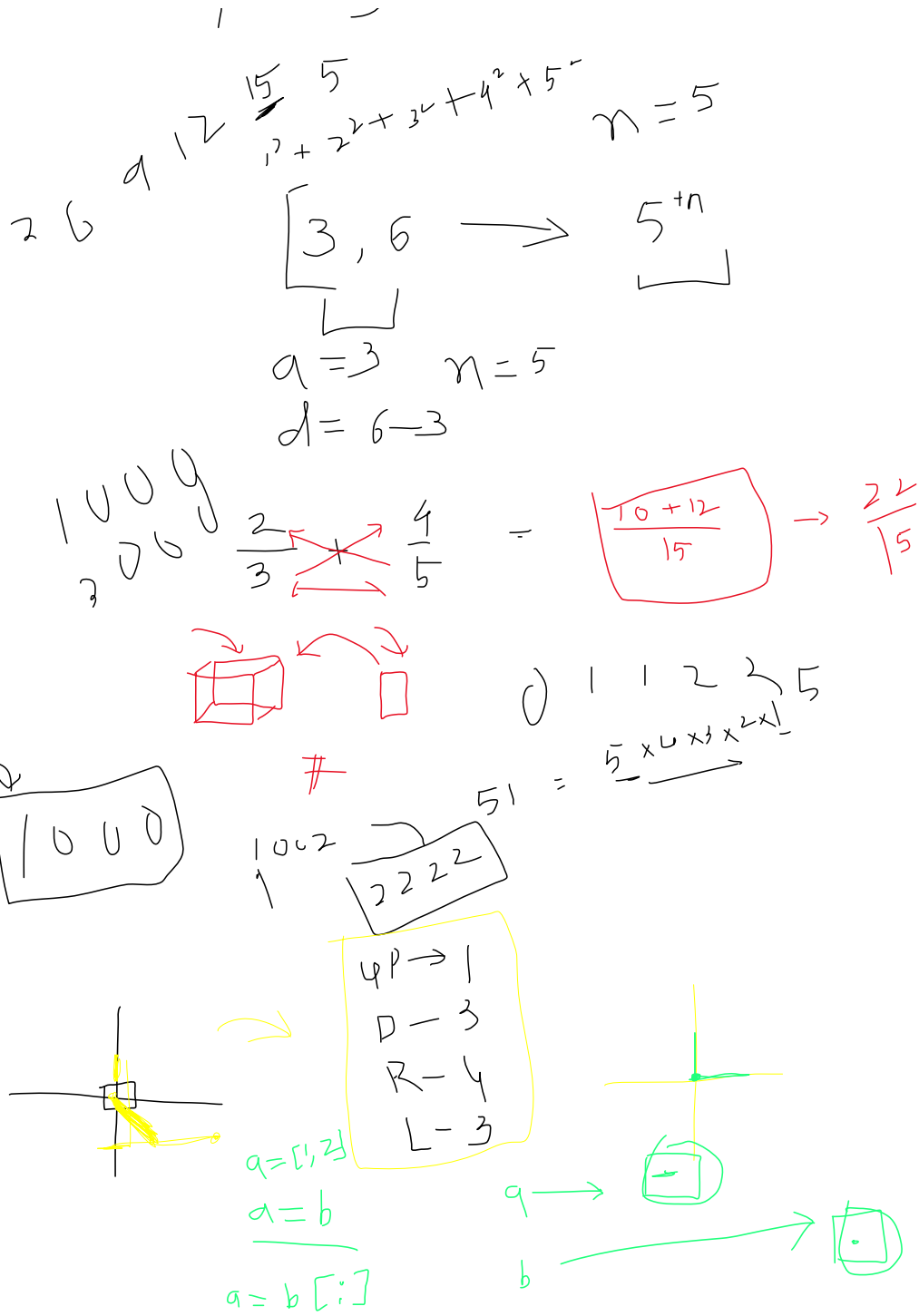
$= 2^n - [2^{n-1} + 2^{n-2} + \ldots + 2^1 + 2^0]$

$= 2^n - [2^n - 1] = 2^n - 2^n + 1$

$O(1) \rightarrow$ constant

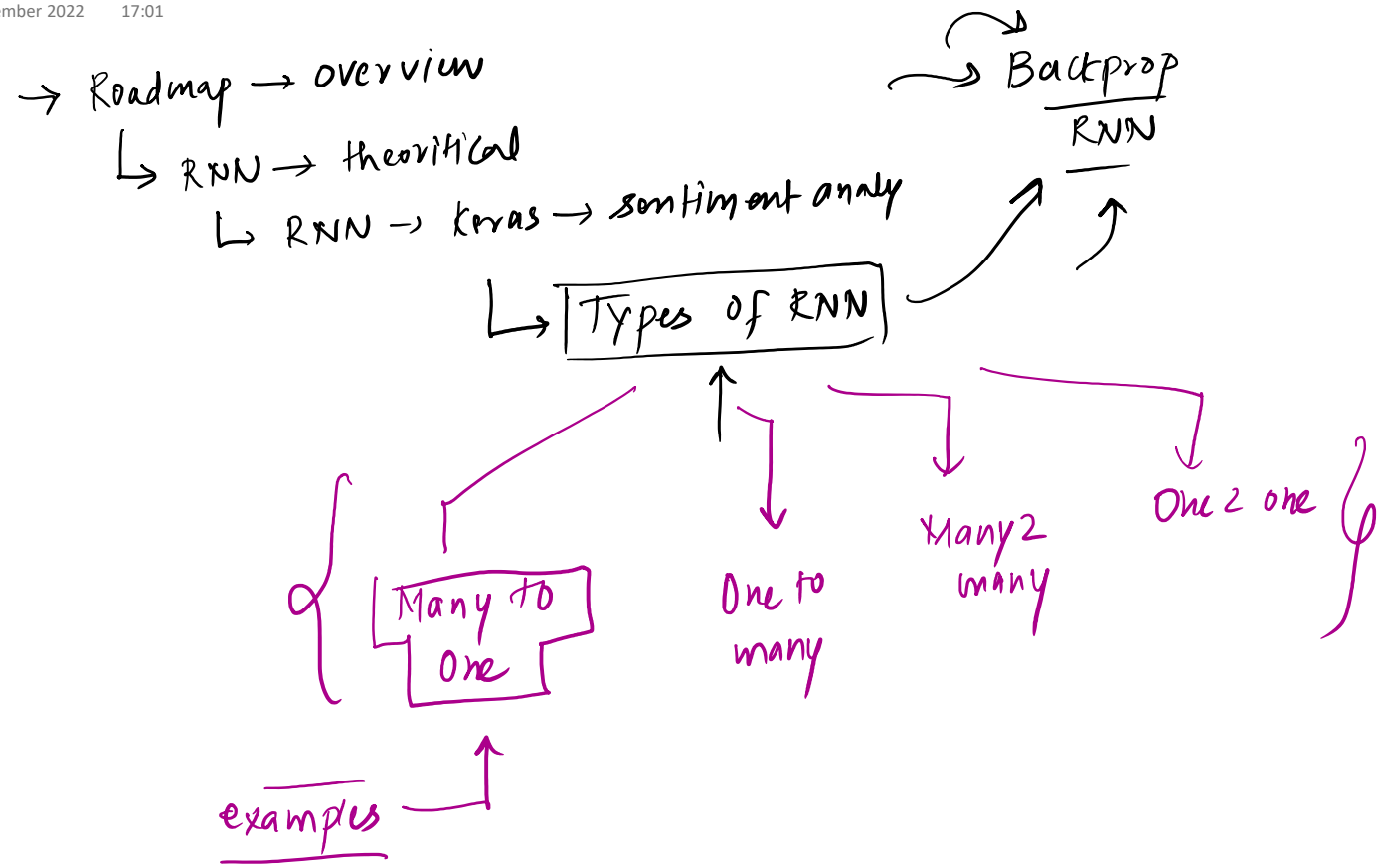


15 5 $\quad -4^2 \times 5^{-}$ $\quad -$


n = 5
[3, 6 → 5^n
q = 3 n = 5
d = 6 − 3
10 + 12 / 15 → 22/5
0 1 1 2 3 5
5 x 4 x 3 x 2 x 1
1000
1002
2222
51 =
up → 1
D − 3
R − 4
L − 3
q = [1, 2]
a = b
a = b [:]
q
b

Till Now

17 November 2022 17:01



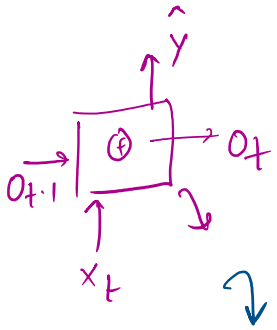
Many to One

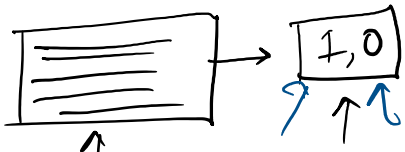
17 November 2022 17:02

input → sequence → sent, chars, time series

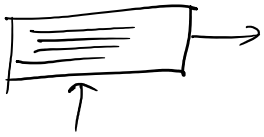
out → non-sequ → int/num → scalars → (1,0)

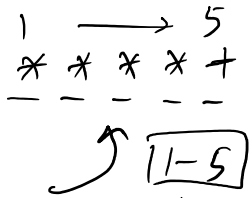
↳ sentiment → ∀any





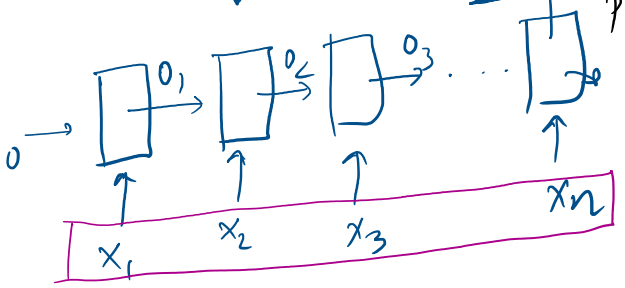
sequence





multiclass classifi

Rating prediction



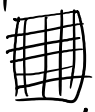
recurrent → simple RNN

many to one

One to Many

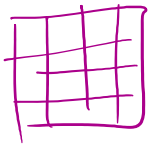
17 November 2022 17:02

→ normal non sequential

↳ 

→ Output → sequences

image captioning

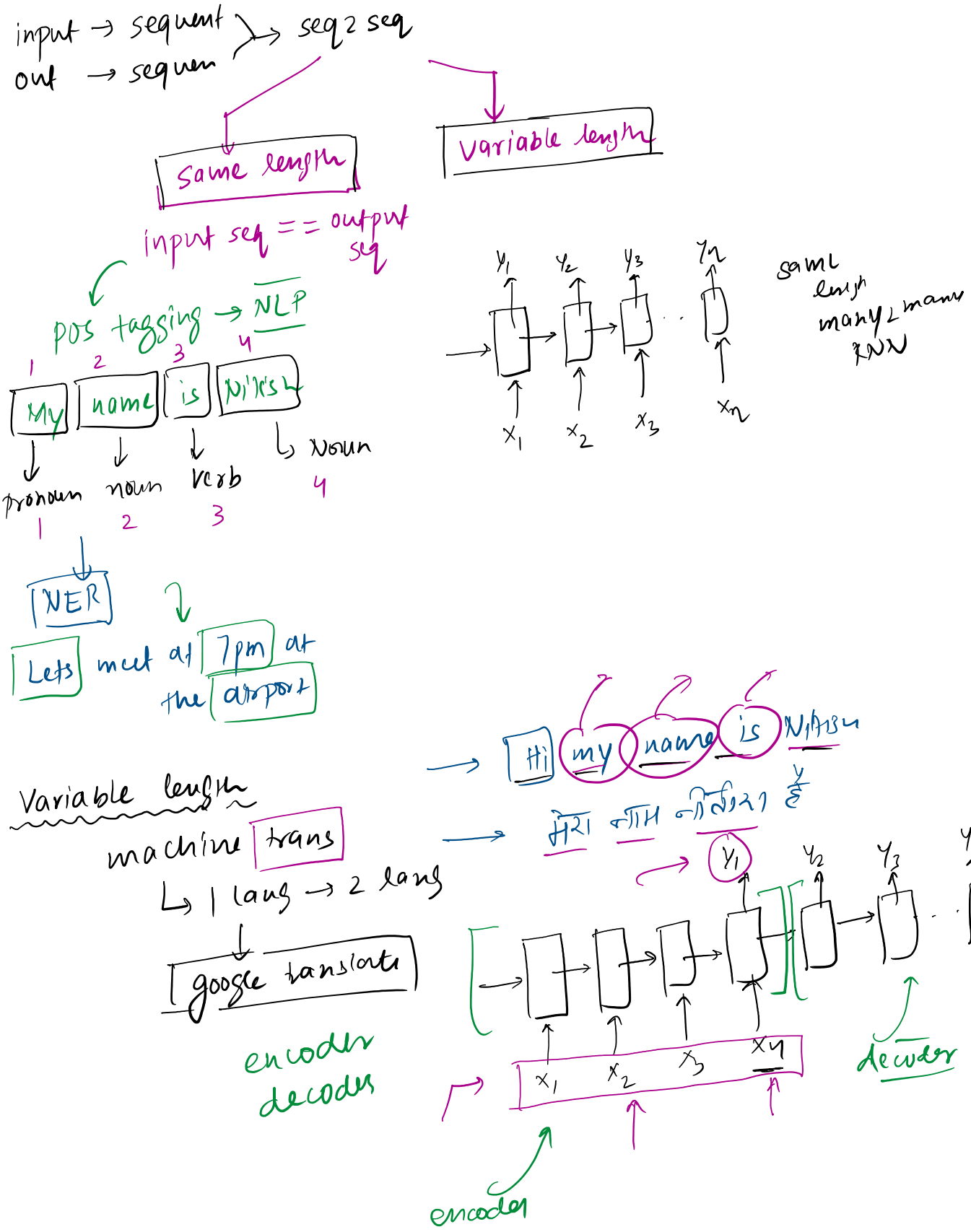
 → textual repres

 → NN → A man is playing cricket

→ music generation

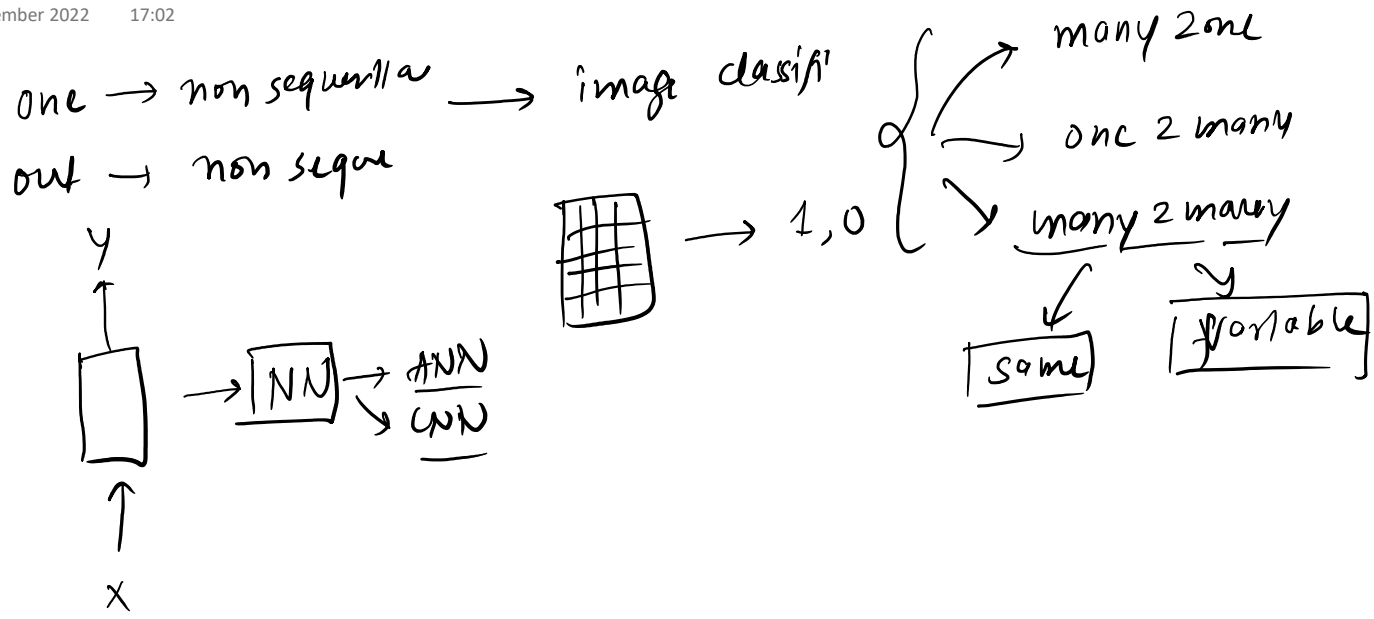
Many to Many

17 November 2022 17:02



One to One

17 November 2022 17:02



Backpropagation in RNN

01 December 2022 16:43

RNN intro → RNN → practical

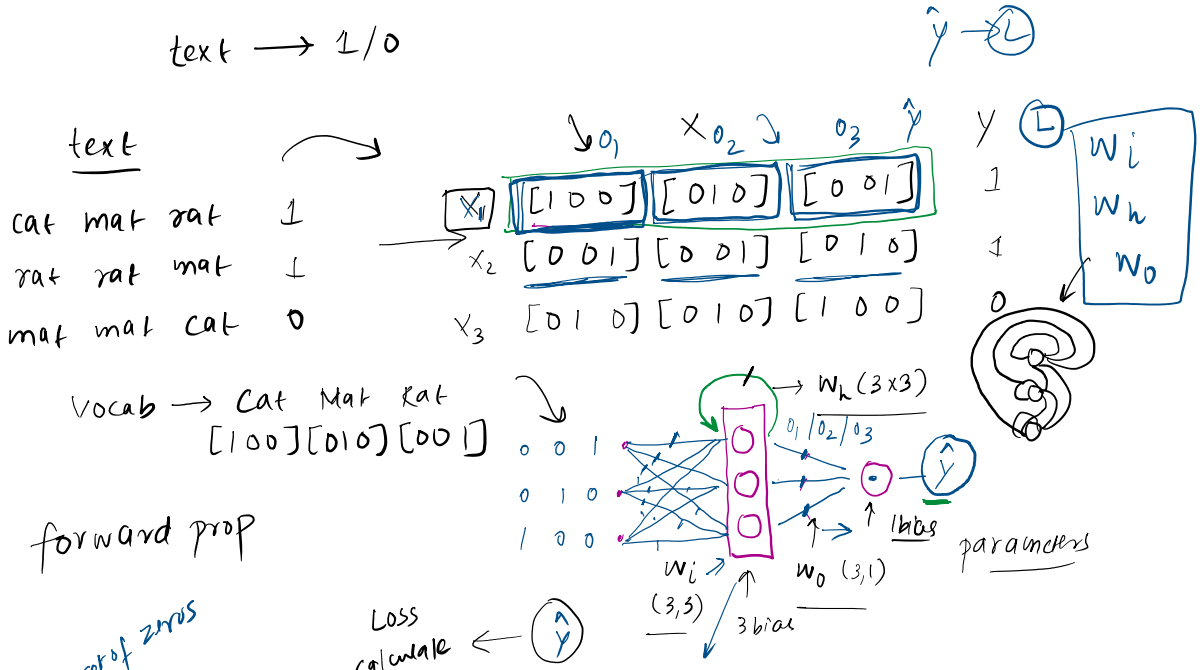
types of RNN

→ RNN → Backprop → BPTT

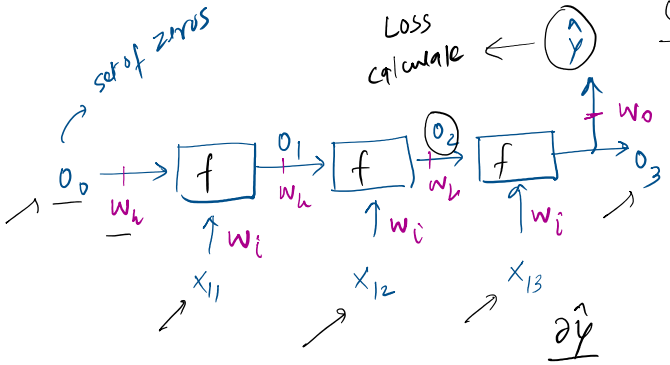
Many to One RNN

Sentiment Analysis

text → 1/0



forward prop



$O_1 = f(X_{11} W_i + O_0 W_h)$

$O_2 = f(X_{12} W_i + O_1 W_h)$

$O_3 = f(X_{13} W_i + O_2 W_h)$

$\hat{y} = \sigma(O_3 W_0)$

$L = -y_i \log \hat{y}_i - (1-y_i) \log(1-\hat{y}_i)$

loss calculate → minimize

L min gradient descent

W_i W_h W_0

$\frac{\partial \hat{y}}{\partial W_0}$ $\frac{\partial L}{\partial \hat{y}}$

$W_i = W_i - \eta \frac{\partial L}{\partial W_i}$

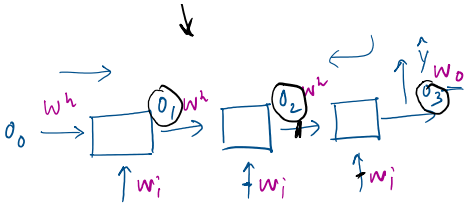
$W_h = W_h - \eta \frac{\partial L}{\partial W_h}$

$W_0 = W_0 - \eta \frac{\partial L}{\partial W_0}$

Backprop

unfold → 3 times

$\frac{\partial L}{\partial W_0}$ $L \rightarrow \hat{y} \rightarrow O_3 \rightarrow W_0$



$\frac{\partial L}{\partial w_0}, \frac{\partial L}{\partial w_i}, \frac{\partial L}{\partial w_h}$

$o_1 = f(x_{i1} w_i + o_0 w_h)$

$o_2 = f(x_{i2} w_i + o_1 w_h)$

$o_3 = f(x_{i3} w_i + o_2 w_h)$

$\hat{y} = \sigma(o_3 w_0)$

$\frac{\partial L}{\partial w_0} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w_0}$

$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial o_3} \frac{\partial o_3}{\partial w_i} + \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial o_3} \frac{\partial o_3}{\partial o_2} \frac{\partial o_2}{\partial w_i} + \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial o_3} \frac{\partial o_3}{\partial o_2} \frac{\partial o_2}{\partial o_1} \frac{\partial o_1}{\partial w_i}$

3 terms

$\frac{\partial \hat{y}}{\partial o_1} = \frac{\partial \hat{y}}{\partial o_3} \frac{\partial o_3}{\partial o_2} \frac{\partial o_2}{\partial o_1}$

10 words → 10 terms

$t = 1$

$\frac{\partial L}{\partial w_i} = \sum_{j=1}^{3} \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial o_j} \frac{\partial o_j}{\partial w_i}$

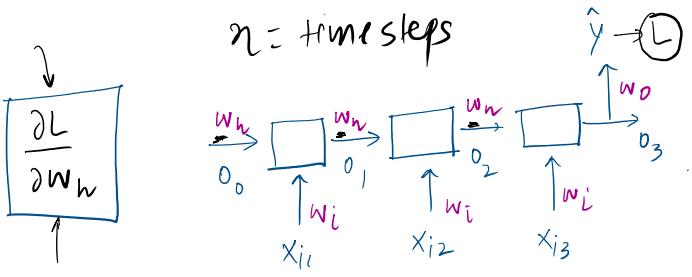
$\frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial o_3} \frac{\partial o_3}{\partial w_i}$

$= \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial o_1} \frac{\partial o_1}{\partial w_i} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial o_3} \frac{\partial o_3}{\partial o_2} \frac{\partial o_2}{\partial o_1} \frac{\partial o_1}{\partial w_i}$

$\frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial o_2} \frac{\partial o_2}{\partial w_i} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial o_3} \frac{\partial o_3}{\partial o_2} \frac{\partial o_2}{\partial w_i}$

$\frac{\partial L}{\partial w_i} = \sum_{j=1}^{n} \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial o_j} \frac{\partial o_j}{\partial w_i}$

n = timesteps



$\frac{\partial L}{\partial w_h}$

$\frac{\partial L}{\partial w_h} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial o_3} \frac{\partial o_3}{\partial w_h} +$

$\frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial o_3} \frac{\partial o_3}{\partial o_2} \frac{\partial o_2}{\partial} +$

∂w_h ∂y ∂o_3 ...

$$\frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial o_3} \frac{\partial o_3}{\partial o_2} \frac{\partial o_2}{\partial w_h} +$$

$$\frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial o_3} \frac{\partial o_3}{\partial o_2} \frac{\partial o_2}{\partial o_1} \frac{\partial o_1}{\partial w_h}$$

$$\frac{\partial L}{\partial w_h} = \sum_{j=1}^{n} \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial o_j} \frac{\partial o_j}{\partial w_h}$$

n = timesteps

for $j=3$

$$\frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial o_3} \frac{\partial o_3}{\partial w_h} \rightarrow \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial o_3} \frac{\partial o_3}{\partial o_1} \frac{\partial o_1}{\partial w_h}$$

for $j=10$

$$\frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial o_{10}} \frac{\partial o_{10}}{\partial o_1} \frac{\partial o_1}{\partial w_h}$$

$$\prod_{t=2}^{j} \frac{\partial o_t}{\partial o_{t-1}}$$

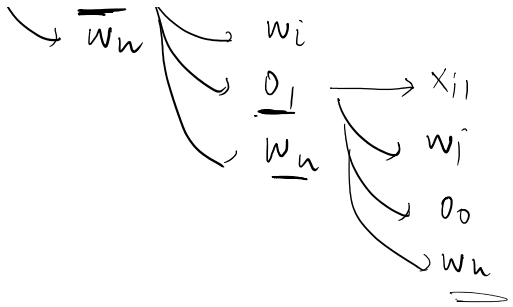
$$t=3 \quad \frac{\partial o_t}{\partial o_{t-1}} = \frac{\partial o_2}{\partial o_1} \frac{\partial o_3}{\partial o_2}$$

$t=2$

$$o_t = f(x_{it} w_{inp} + o_{t-1} w_h)$$

$$\frac{\partial o_t}{\partial o_{t-1}} = \prod_{t=2}^{j} f'(x_{it} w_{inp} + o_{t-1} w_h) w_h$$

$[0-1]$



Problem with RNN

19 December 2022 16:33

RNNs → sequential data → textual, time series

suffer (2) major problems → LTSMs (Why)

RNN

→ Problem of long term dependency
→ Unstable gradients stagnated training → Unstable gradients

Applications next word predictor

→ RNN

timesteps

Marathi is spoken in Maha

short

Maha is a beautiful place I went there last year, but I could not enjoyed properly bcz I don't understand marathi

long term

RNN

keypad

RNN

{ vanishing gradient prob }

{ Exploding gradient problem } → RNN

possible solⁿ

↳ remove

Problem # 1 → Problem of long term dependency → Vanishing

inp	out
1 0 1	1
0 0 1	0
0 0 0	0
1 1 1	1

0-1

inp hidden out 1 0 1 W_h W_{in} W_{out}

L → min

W_h, W_{in}, W_{out}

↓

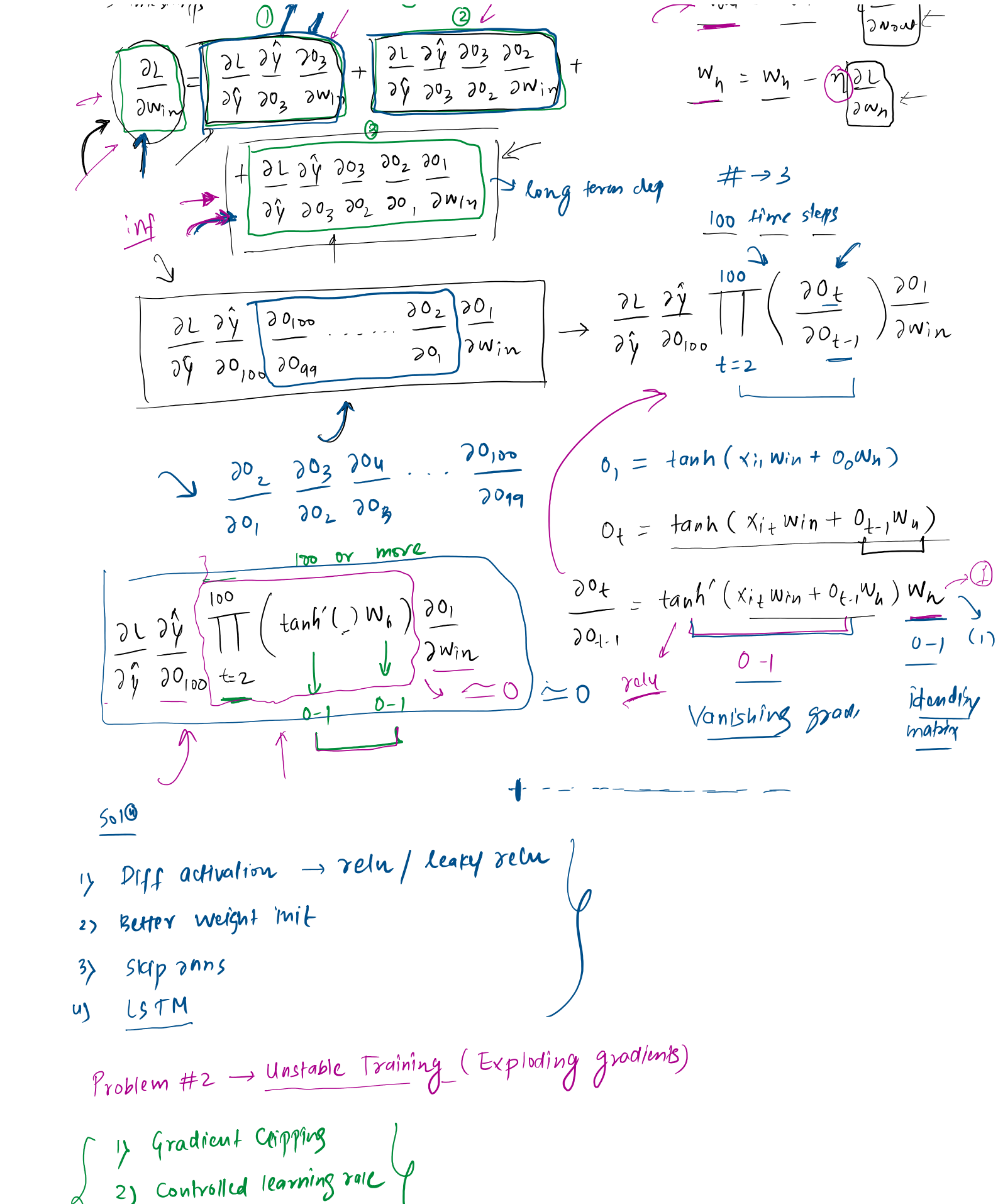
gradient des

$W_{in} = W_{in} - \eta \frac{\partial L}{\partial W_{in}}$

$W_{out} = W_{out} - \eta \frac{\partial L}{\partial W_{out}}$

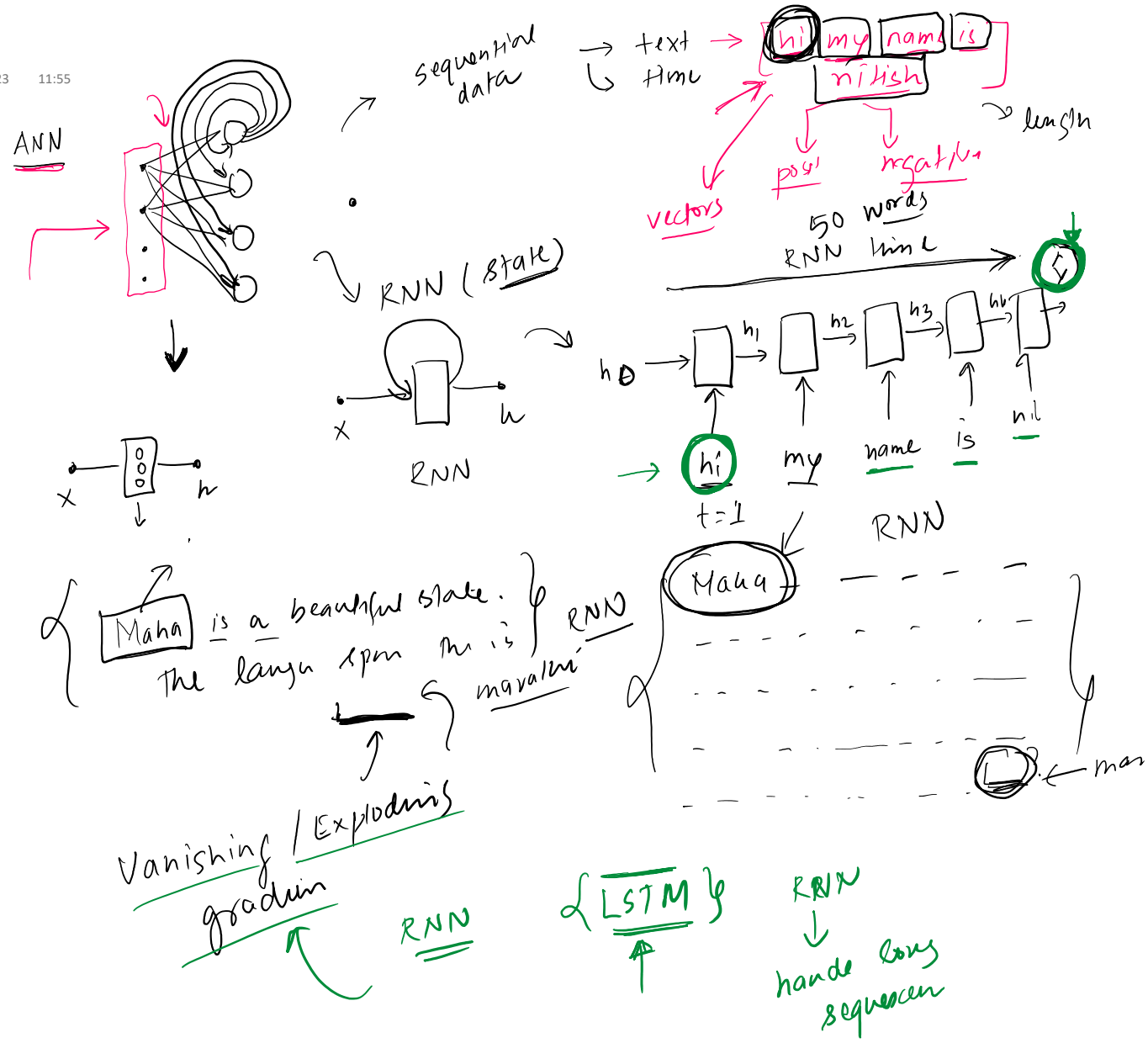
short term dep

3 time stamps


Long term dep
100 time steps
100 or more
Vanishing grad
identity matrix
Sol(n)
1) Diff activation → relu / leaky relu
2) Better weight init
3) Skip anns
4) LSTM
Problem #2 → Unstable Training (Exploding gradients)
1) Gradient Clipping
2) Controlled learning rate
3) LSTM

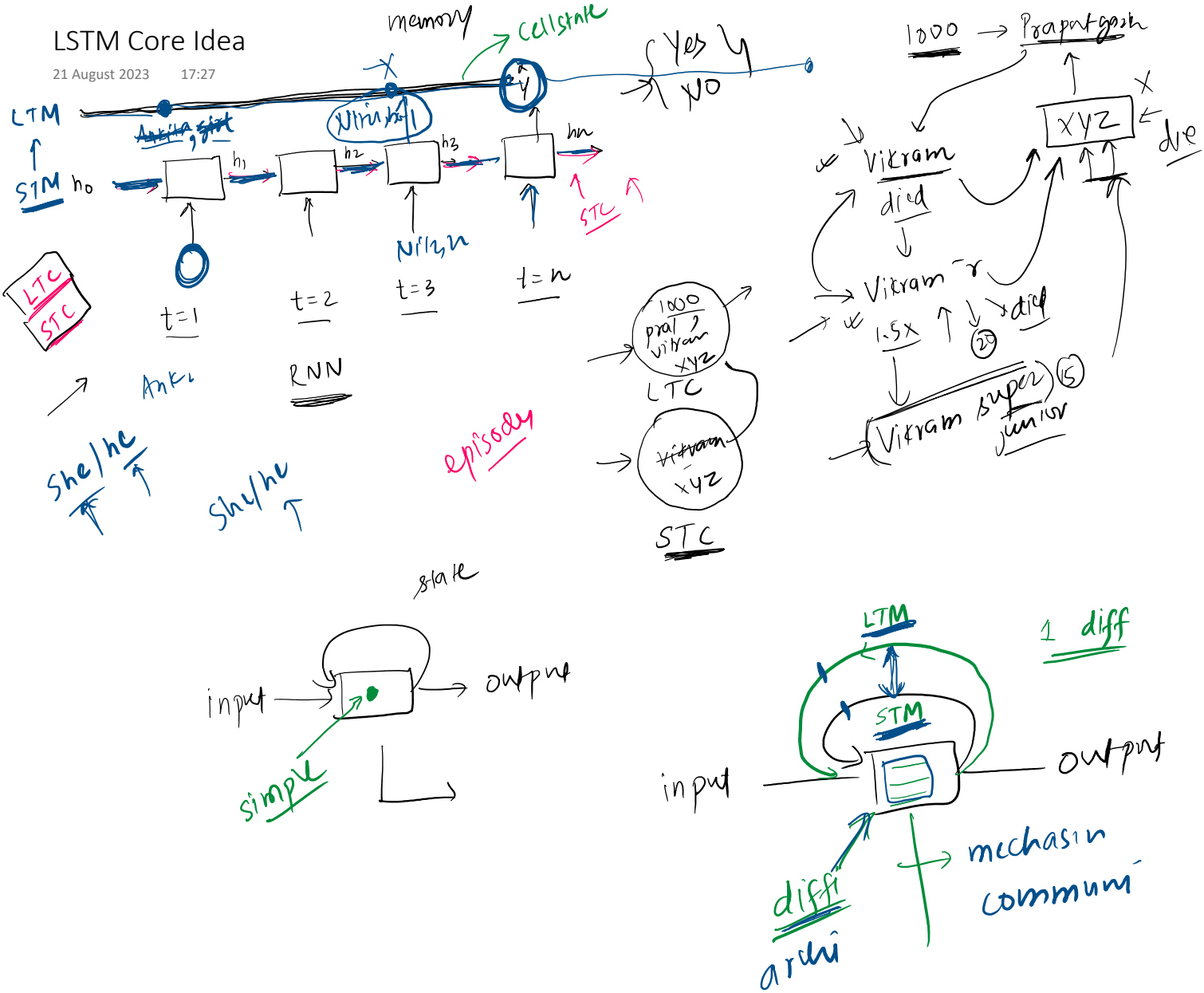
Recap

21 August 2023 11:55



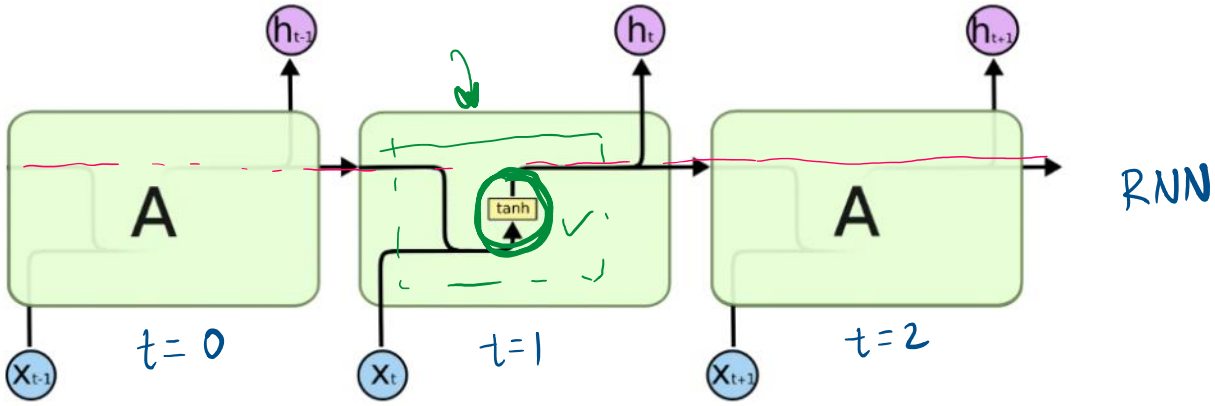
LSTM Core Idea

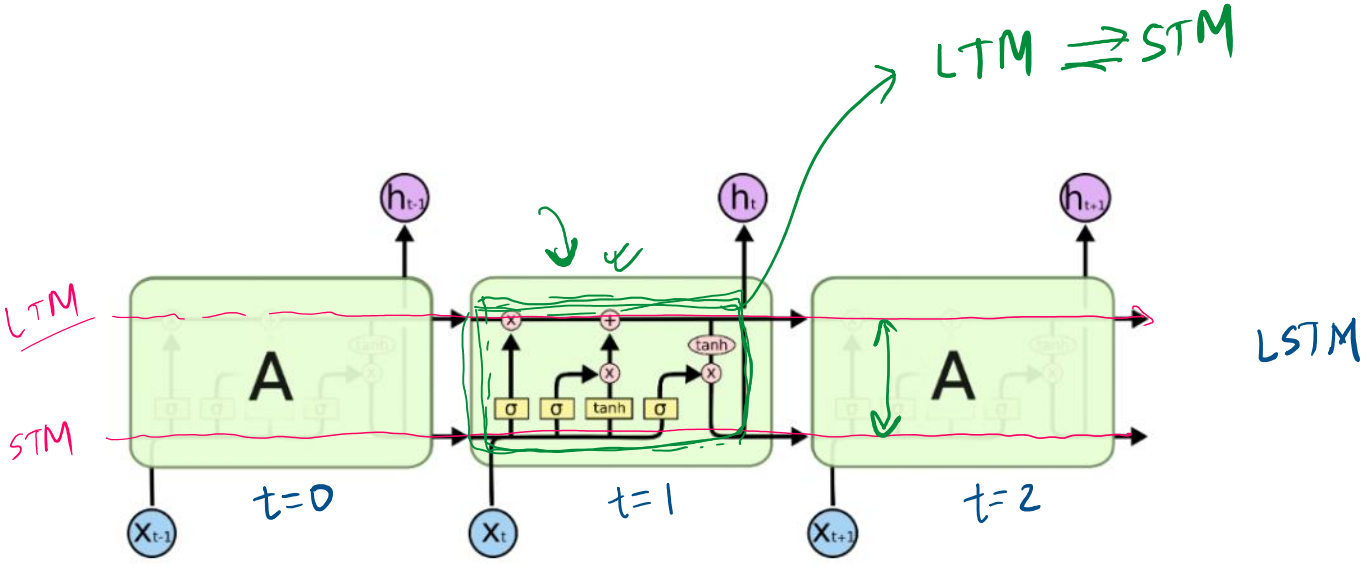
21 August 2023 17:27

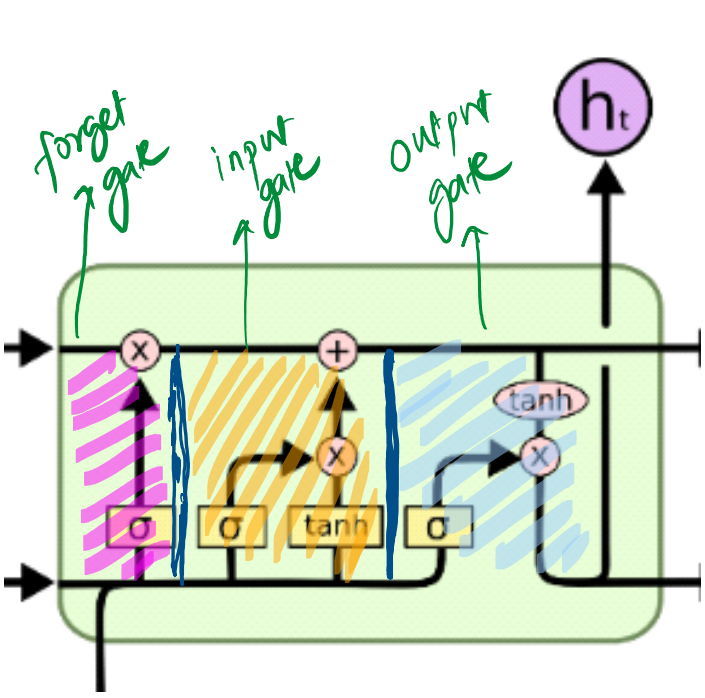


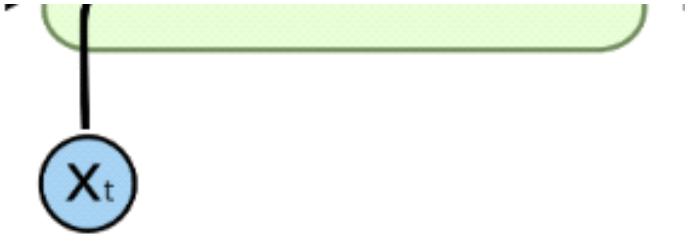
LSTM Architecture

21 August 2023 18:41



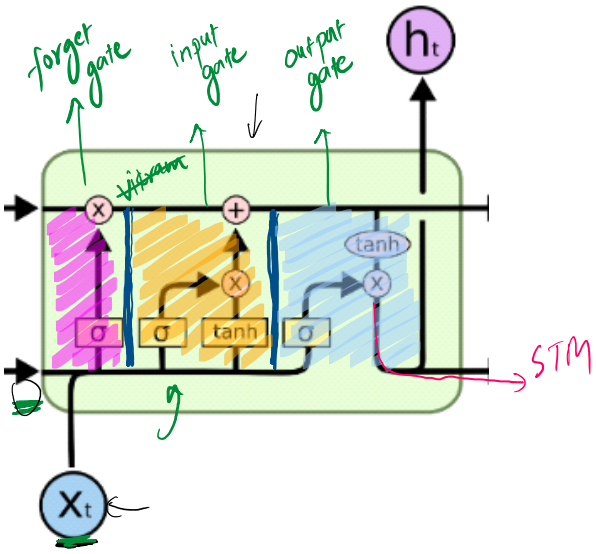





X_t

LSTM Gates

21 August 2023 19:06

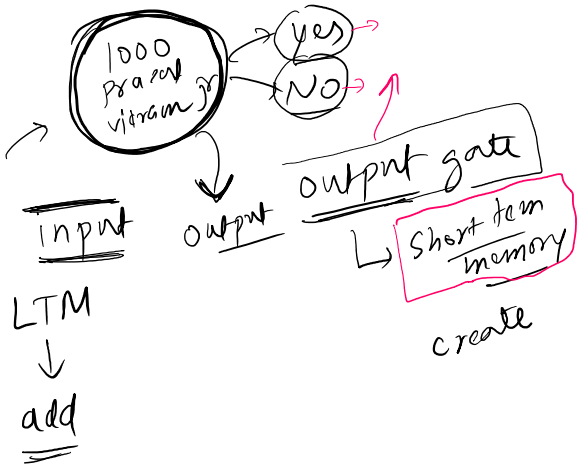


forget

input

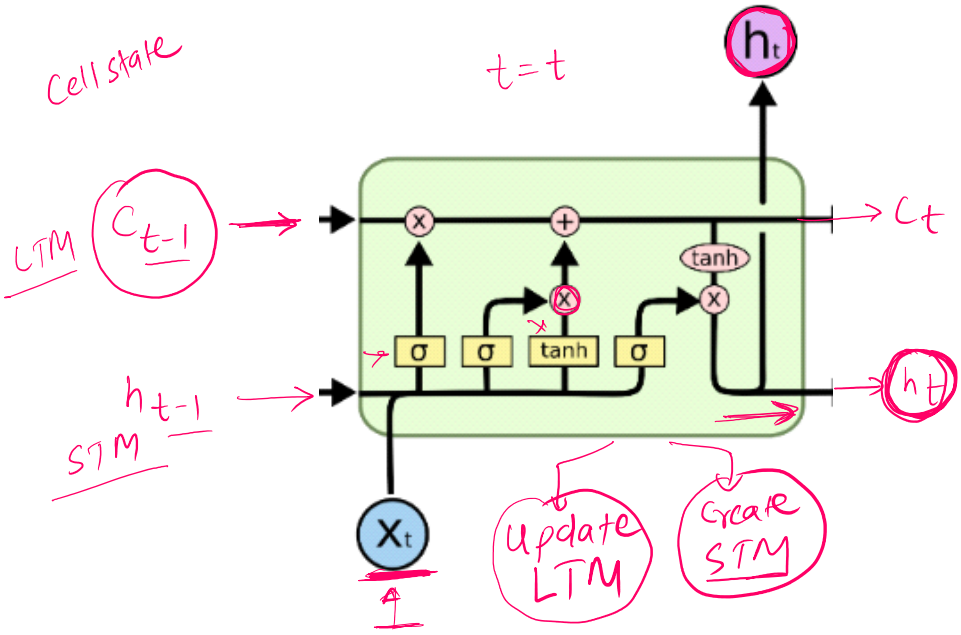
LTM

remove



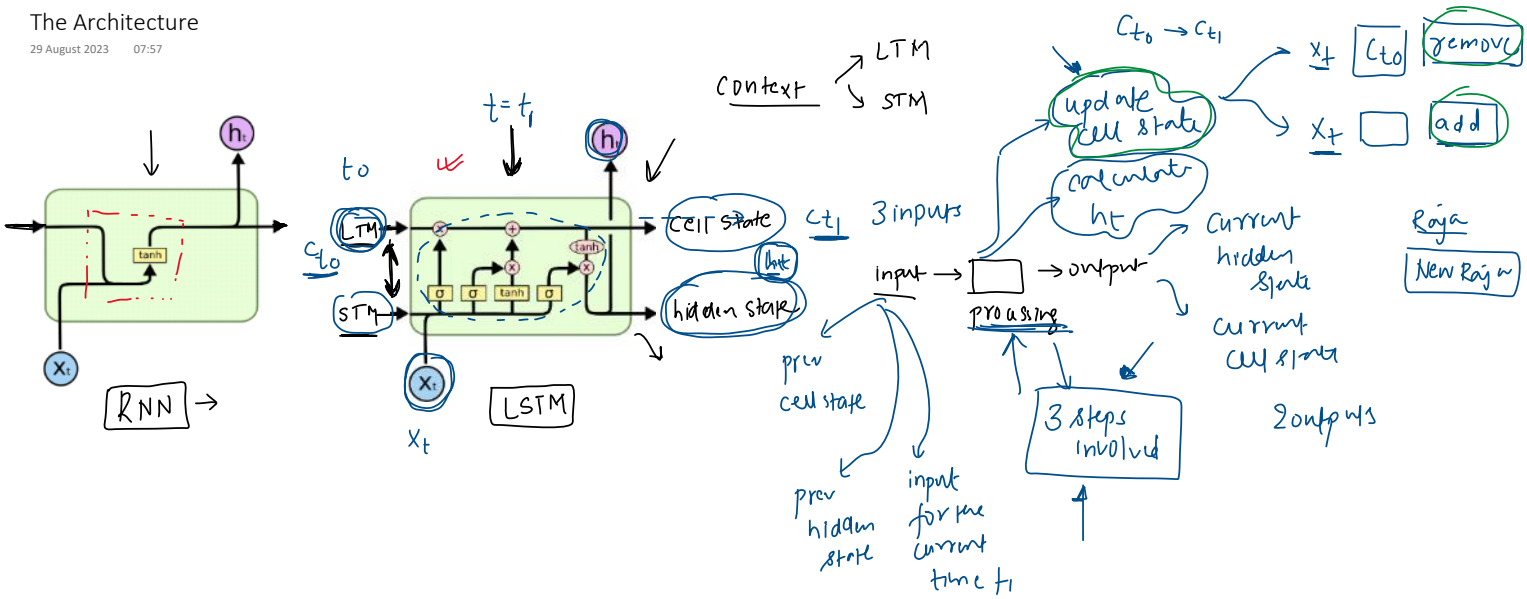
Summary

21 August 2023 19:29



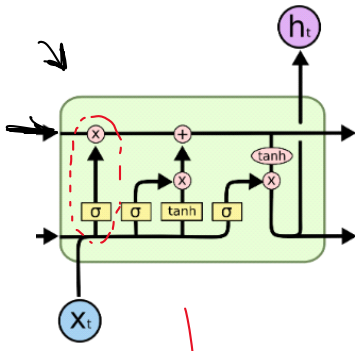
The Architecture

29 August 2023 07:57

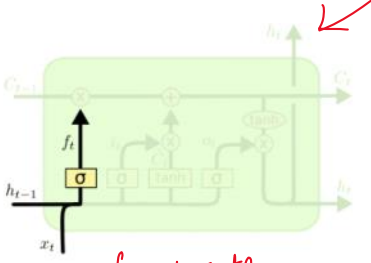


The Gates

29 August 2023 08:09



3 gates

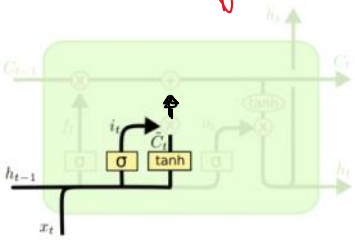


forget gate

to remove something from C_t

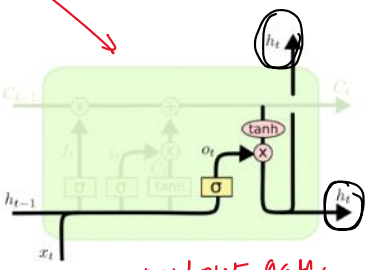
update C_t

+



input gate

to add something to the C_t

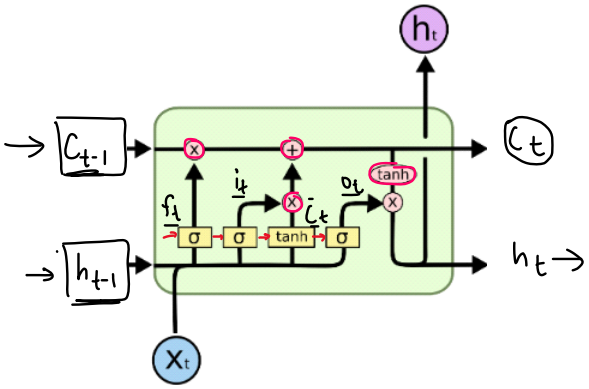


output gate

calculat hidden state h_t

What are C_t and h_t

29 August 2023 08:08



h_t C_t → [0.1 0.3 0.9]

vectors

3d vector

h_t C_t dim equal

same

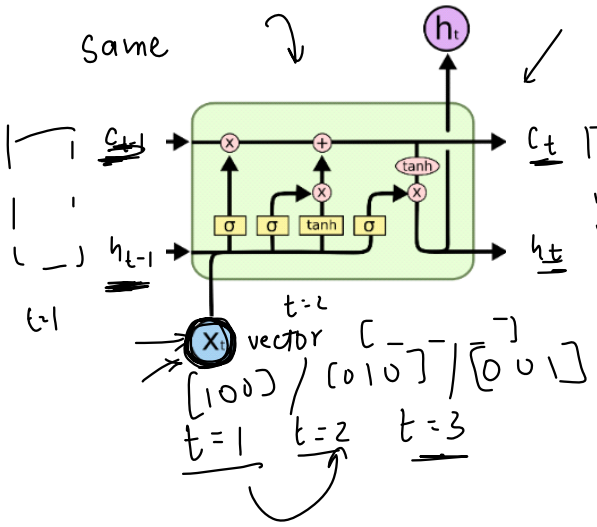
h_t [0.1 0.45 0.6]

C_t [0.55 0.6 0.0]

What is X_t

29 August 2023 17:40

RNN



sentiment

vectorization

cat mat rat	0 ←
cat rat rat	0
mat mat cat	1

[100] [010] [001]

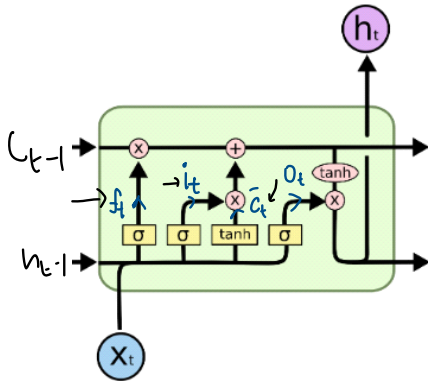
word vector

OHE

Cat	mat	rat
1	0	0
0	1	0
0	0	1

What are f_t, i_t, o_t and $\bar{C}t$

29 August 2023 08:09



f_t i_t $\bar{C}_t$ → candidate cell state o_t

forget gate — input gate — output gate

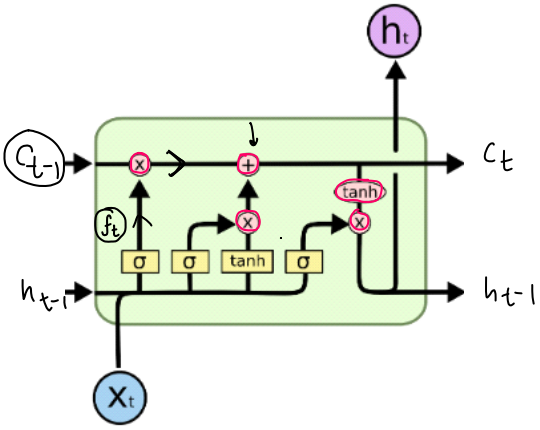
vectors → C_t h_t

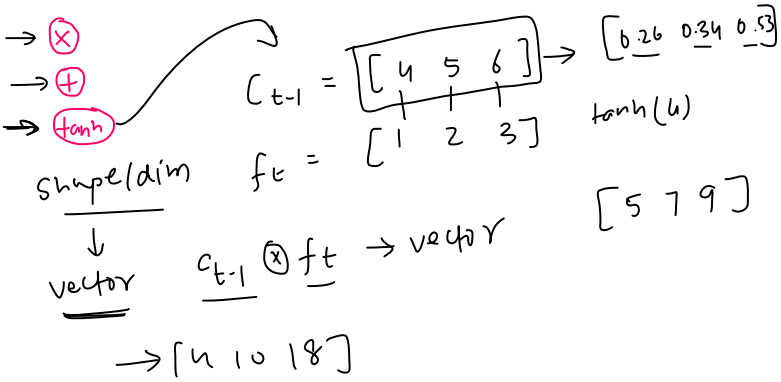
f_t i_t $\bar{C}_t$ o_t

[x y z]
[ ]

Pointwise Operations

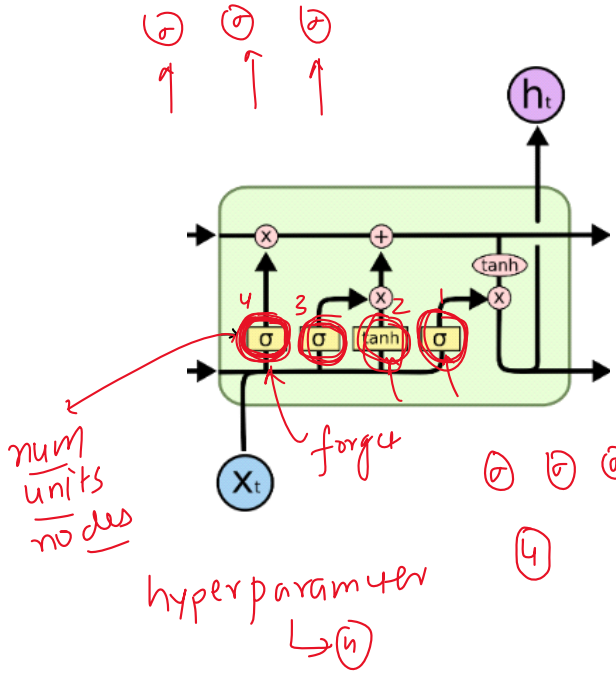
29 August 2023 18:26

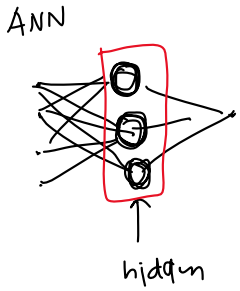


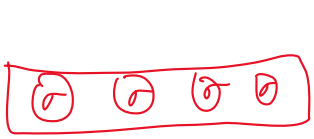


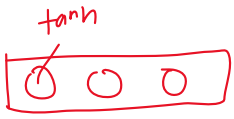
→Neural Network Layers

29 August 2023 18:34



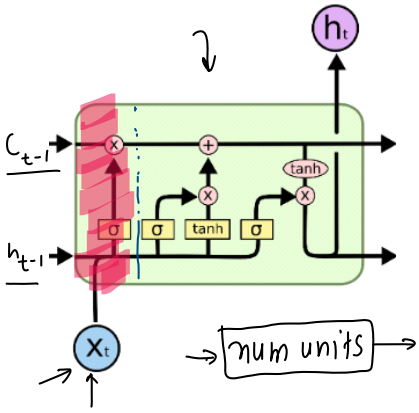






The Forget Gate

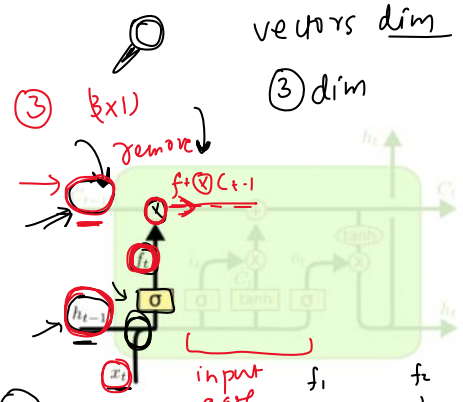
29 August 2023 19:58



→ num units → 3

4 dim vector

$[x_{i1}\ x_{i2}\ x_{i3}\ x_{in}]$



input
output → Cell state → remove

(3x1) ⊗ (3x1)

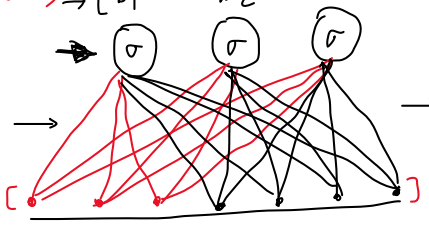
→ 1) Calculate f_t
2) $C_{t-1} \otimes f_t$

σ σ σ

f_1 f_2 f_3 → $[f_1\ f_2\ f_3]$ → $f_t(3)$

b_1 b_2 $b_3]$ (3x1) (3x7) (7x1)

→ (3x1) + (3x1) = (3x1)



→ W_{3x7} (21 weights) 21 wts
biases 4 biases

→ 7x1

forget gate ← 0%
50%
100%

remove

$[h_{t-1}, x_t]$

long term

f_t

$[4\ 5\ 6]$

C_{t-1} f_t $[0\ 0\ 0]$ $[\frac{1}{2}\ \frac{1}{2}\ \frac{1}{2}]$ ⊗

$[1\ 1\ 1]$

$[2\ 2.5\ 3]$

$[4\ 5\ 6]$

$[0\ 0\ 0]$ → 100% remove

$$f_t = \sigma\left(W_f[h_{t-1}, x_t] + b_f\right)$$

(3x1) (3x7) (7x1) (3x1) + (3x1) → (3x1) (3x1)

removal

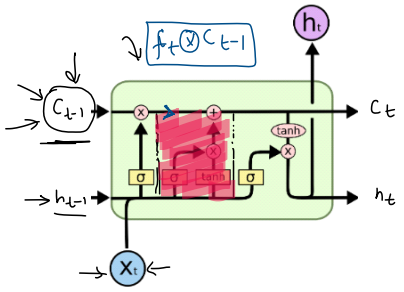
$f_t \otimes C_{t-1}$ → (3x1)

(C_{t-1}) → remove

The Input Gate

30 August 2023 04:38

add some new imp info to ml C_t



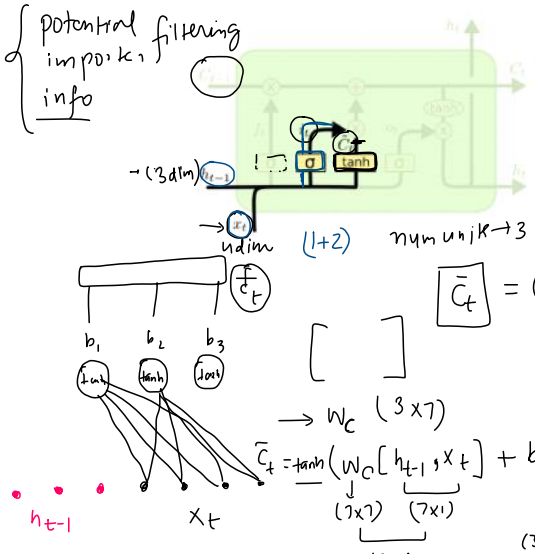
stages

1> $\bar{C}_t$ candidate cell state

2> i_t

3> C_t → current cell state

potential filtering important info



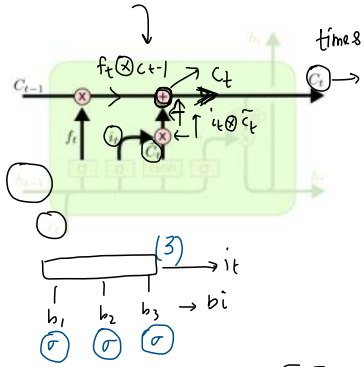
$\bar{C}_t$ = (3x1)

$\bar{C}_t = \tanh(W_C[h_{t-1}, X_t] + b_C)$

W_C (3x7)

(3x7) (7x1) (3x1)

(3x1) → (3x1)



pointwise $i_t \otimes \bar{C}_t \rightarrow \bar{C}_t^*$ (filtered candidate cell state)

(3x1) (3x1)

[0.5 0.5 0.5] [4 5 6]

1 1 1 ↓ 50% 0% 100%

[2, 2.5 3]

$C_t = f_t \otimes C_{t-1} \oplus i_t \otimes \bar{C}_t$

→ [ ] W_i

h_{t-1} X_t

$i_t = \sigma(W_i[h_{t-1}, X_t] + b_i)$

(3x7) (7x1) (3x1)

(3x1) → (3x1)

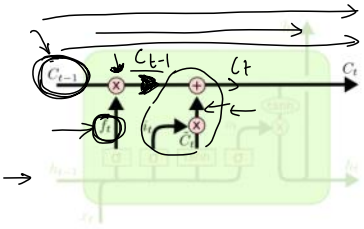
i_t - filter

$\bar{C}_t \rightarrow C_t$

10%

[4 5 6] → [4 5 6]

t=t-1 t=t



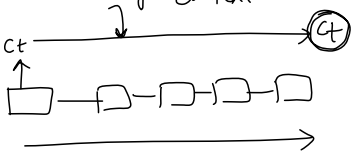
$f_t = [1\ 1\ 1]$

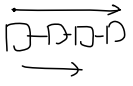
$i_t \otimes \bar{C}_t = [0\ 0\ 0]$

[4 5 6] [0 0 0]

$C_t = [4\ 5\ 6]$

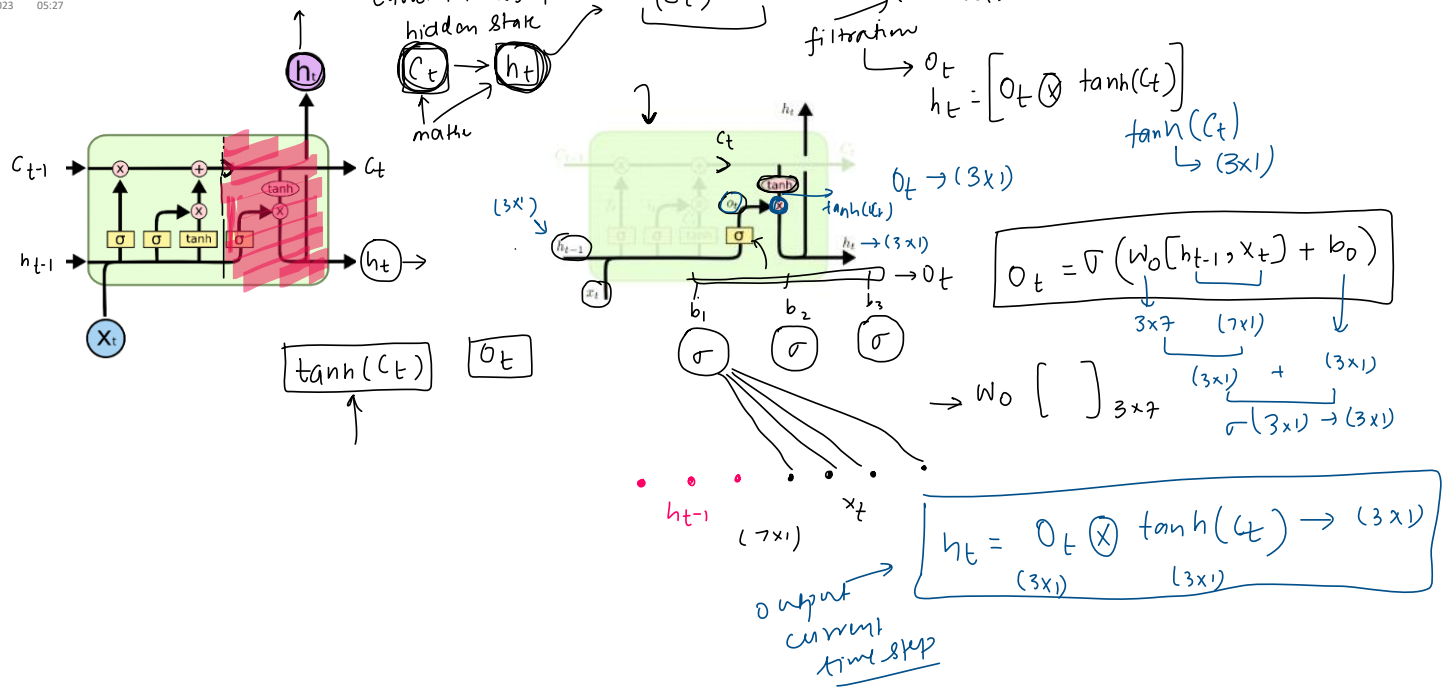
long term context





The Output Gate

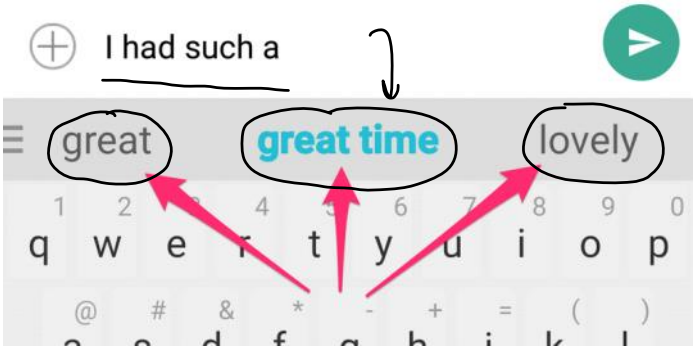
30 August 2023 05:27

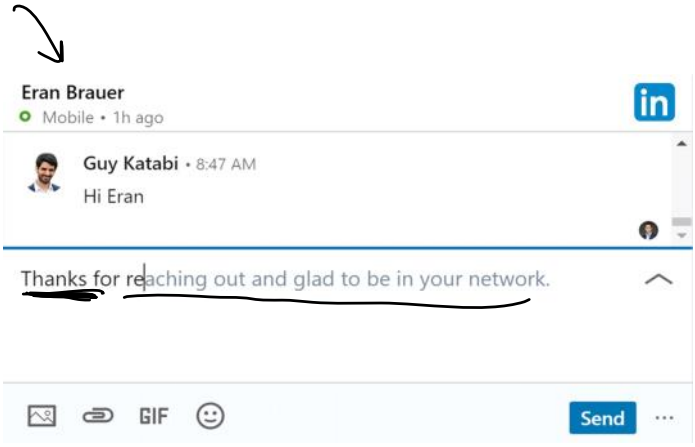


What is a Next Word Predictor

08 September 2023 08:50

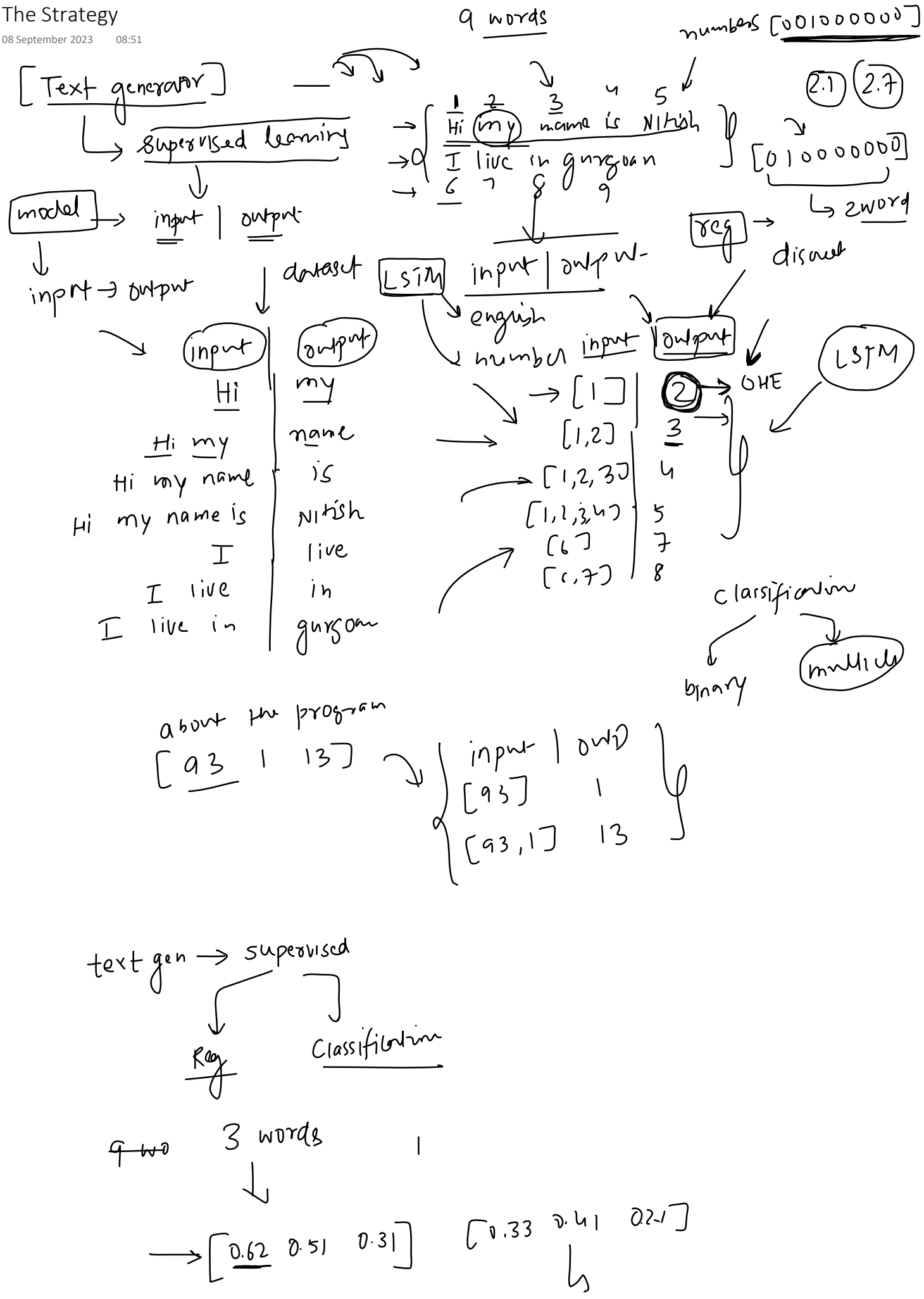
code

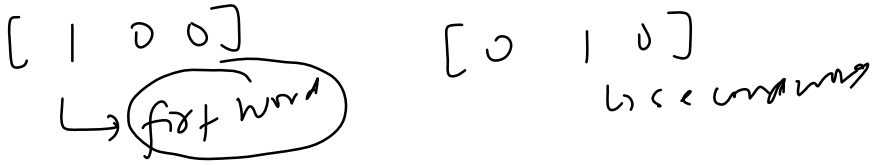




The Strategy

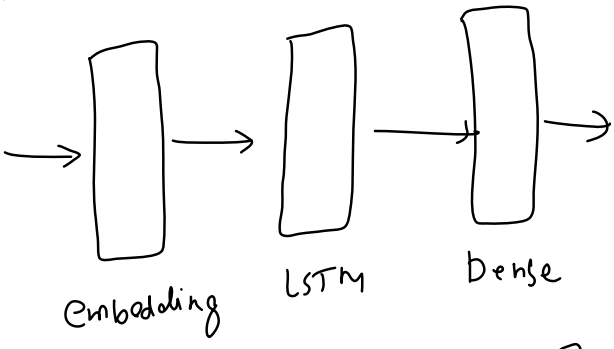
08 September 2023 08:51

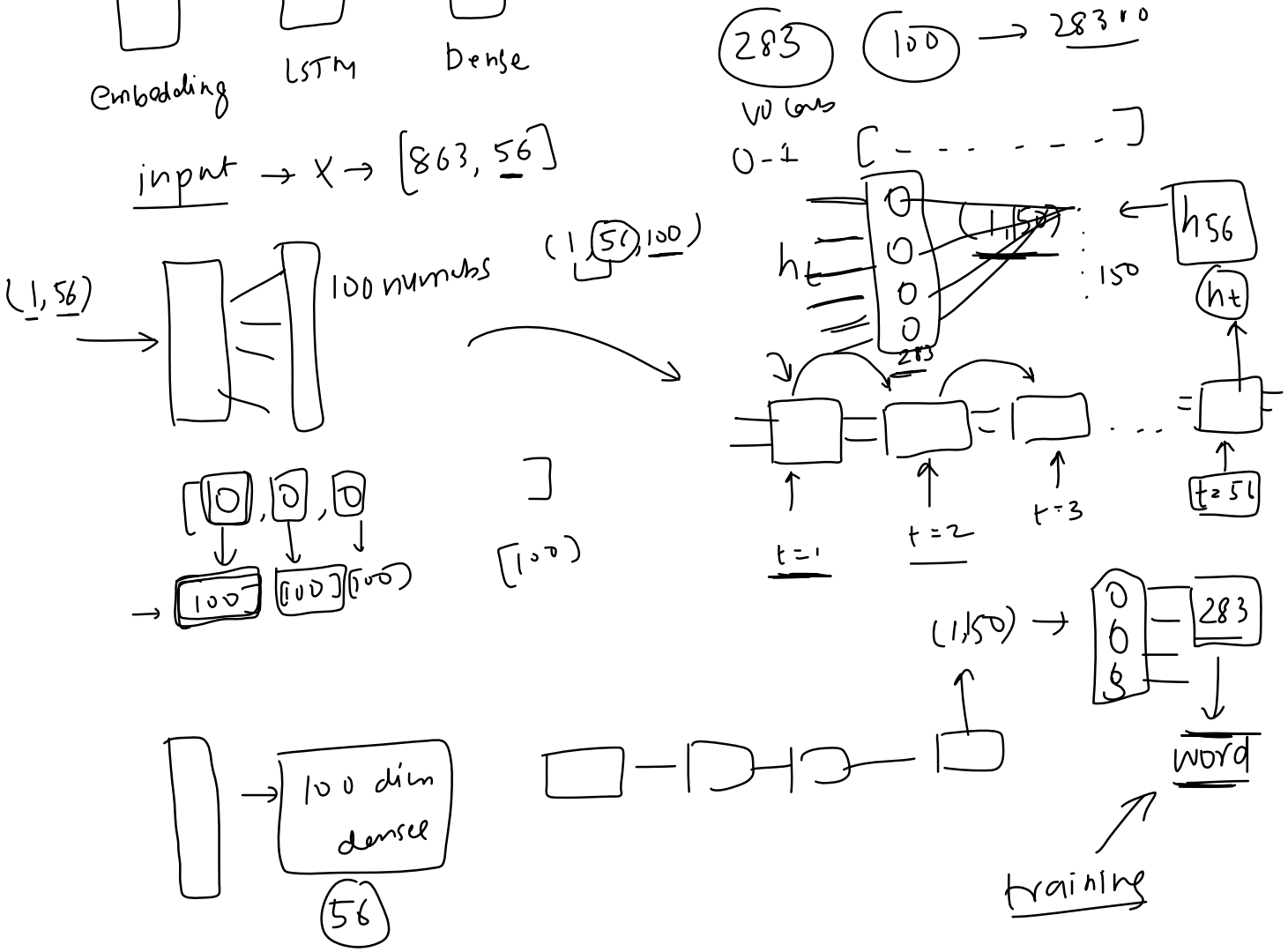



[1 0 0]
first word
[0 1 0]

The Architecture

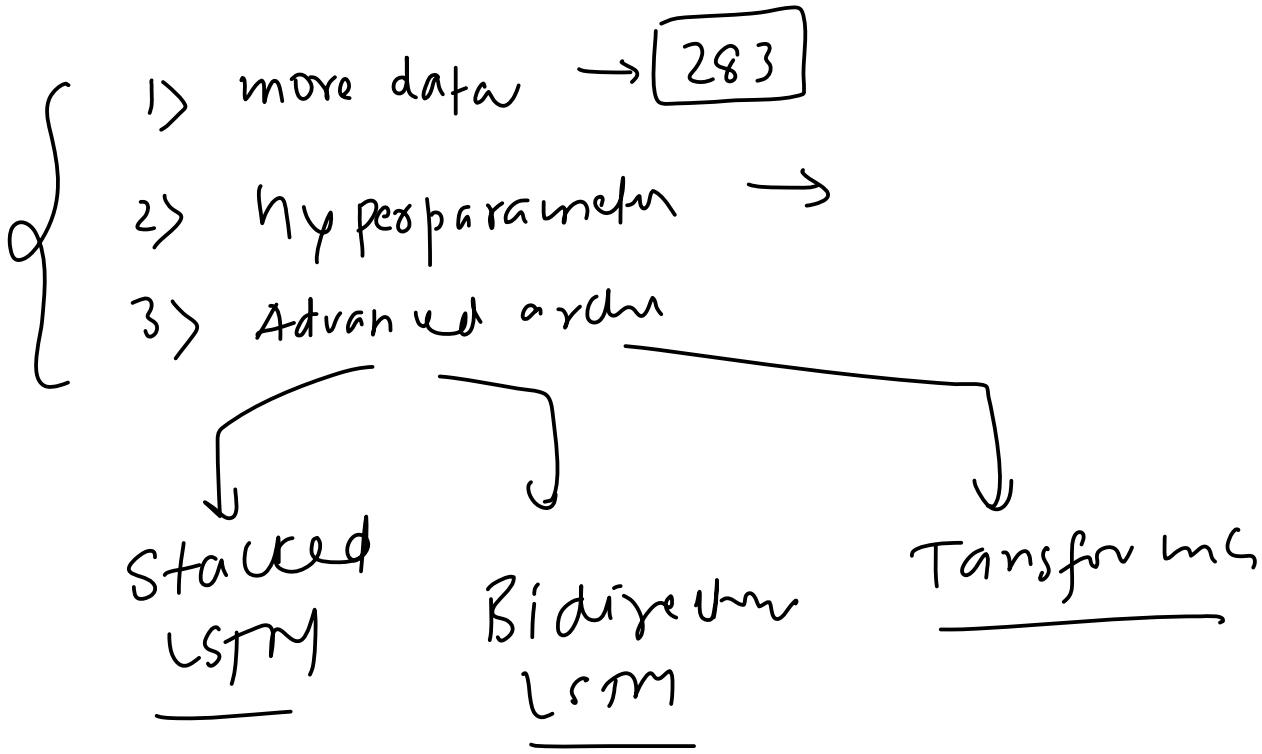
08 September 2023 08:55





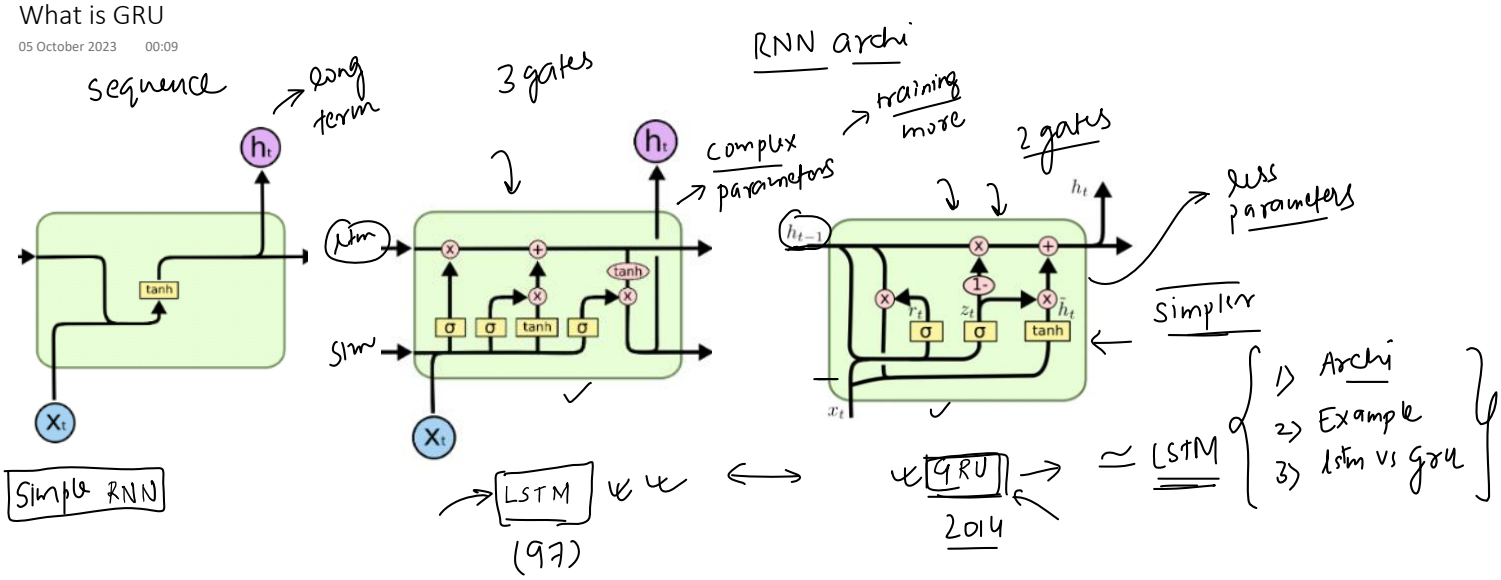
How to improve performance?

08 September 2023 08:51



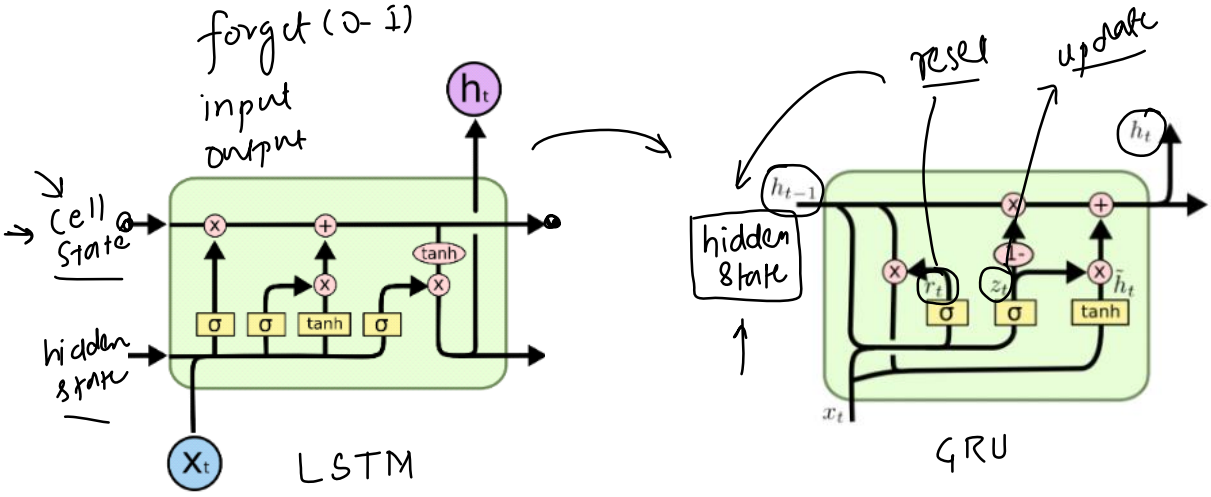
What is GRU

05 October 2023 00:09



The Big Idea Behind GRU

05 October 2023 00:47

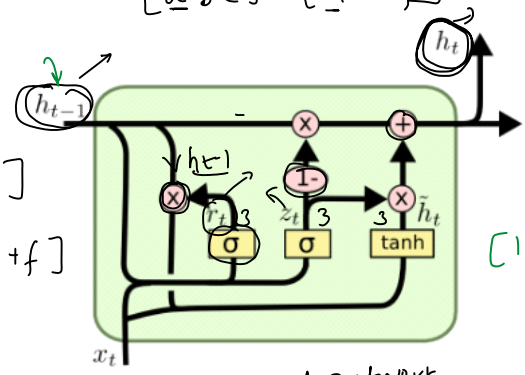


The Setup

05 October 2023 01:07

→ Advise → LSTM / GRU → confusing

goal → t



[a b c] [d e f]

[ad be cf]

[a+d b+e c+f]

neural network layers (FC layers)

prev hidden state — curr hidden state — input

h_{t-1}, x_t → h_t

vectors

h_{t-1}, h_t, x_t, r_t, z_t, $\tilde{h}_t$? → candidate hidden state

σ, tanh ?

reset gate, update gate

[1 2 3 4]

x + 1- ?

[1 2 3 4] dim same

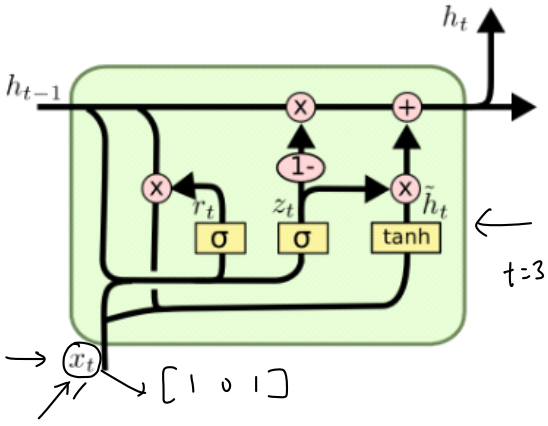
number of = 5 6

10

σ → σ σ σ σ σ

The Input Xt

05 October 2023 01:52



OHE / BOW / Word2Vec

Sentiment

text	review
cat mat rat	1
cat rat rat	0
→ mat rat mat	1

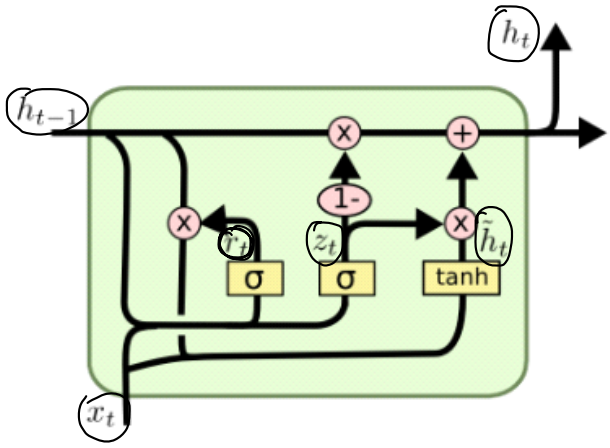
S1 [1 0 0] [0 1 0] [0 0 1]

S2 [1 0 0] [0 0 1] [0 0 1]

S3 [0 1 0] [0 0 1] [0 1 0]

Architecture

05 October 2023 02:10



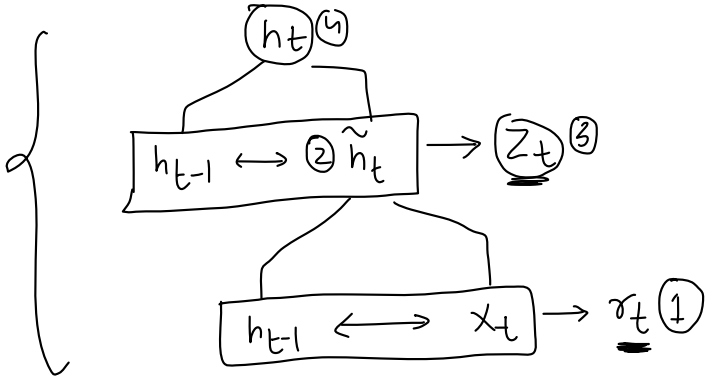
h_{t-1} , x_t ⟶ h_t

→ 1) Calculate r_t (reset gate)

→ 2) Calculate $\tilde{h}_t$ (candidate hidden state)

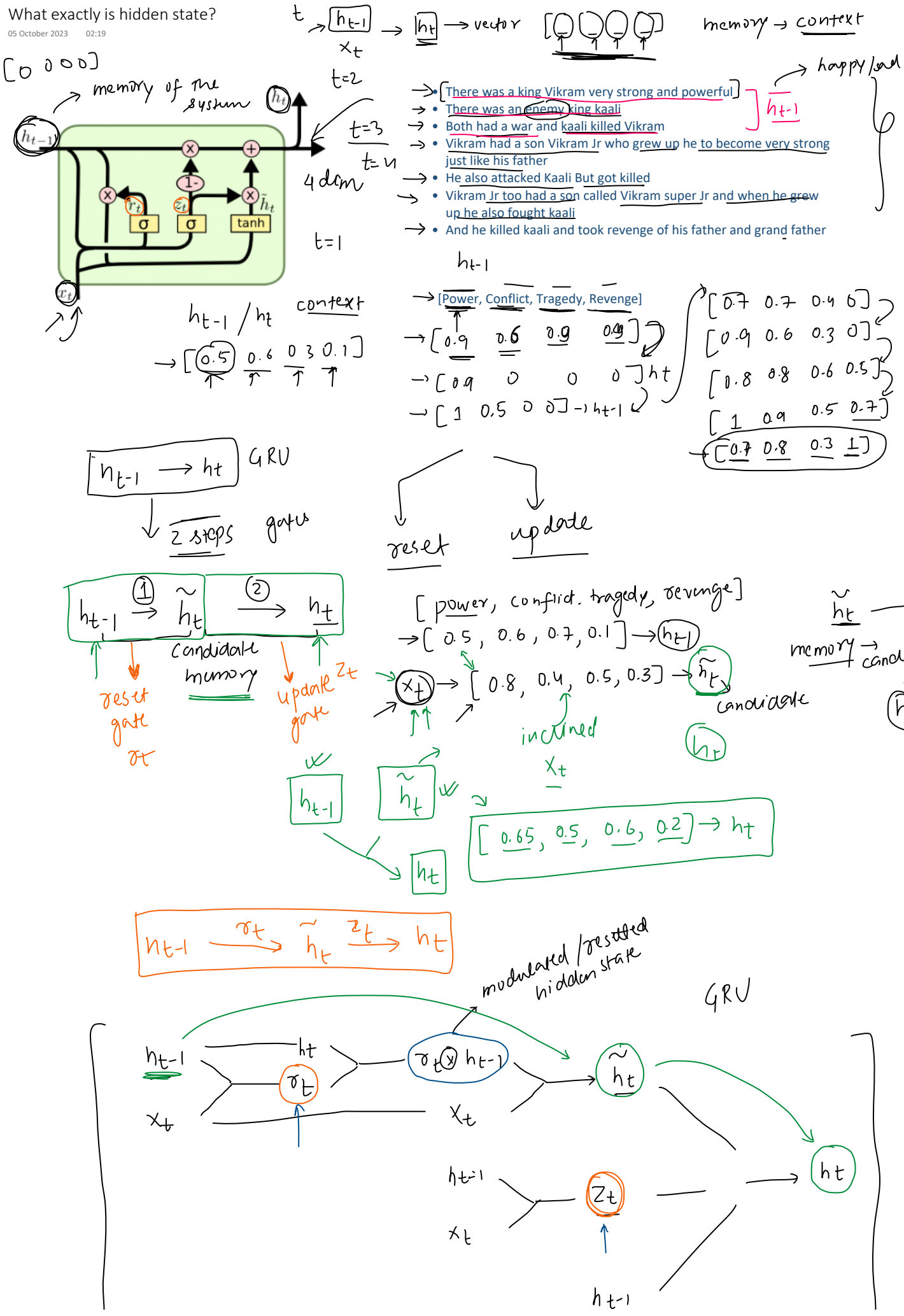
→ 3) Calculate z_t (update gate)

→ 4) Calculate h_t (current hidden state)



What exactly is hidden state?

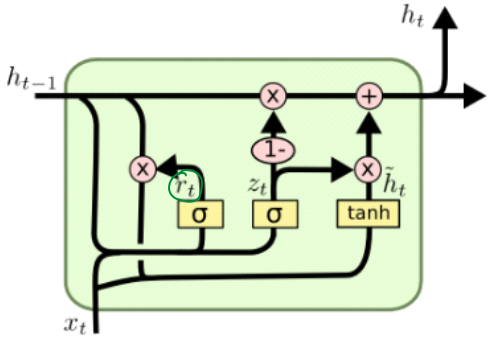
05 October 2023 02:19




h_{t-1}

Calculating the reset gate

05 October 2023 14:24



reset

r_t → vector → h_{t-1}

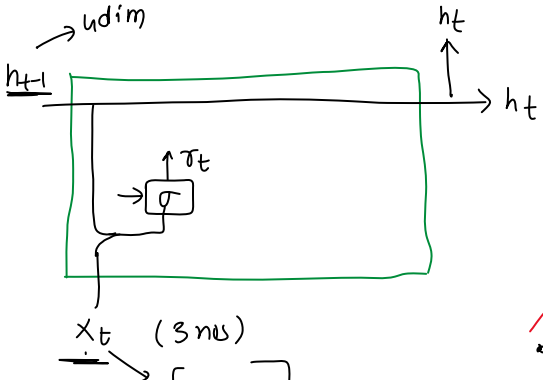
↳ gate

$h_{t-1} = [0.6, 0.6, 0.7, 0.1]$ (lower lower higher)

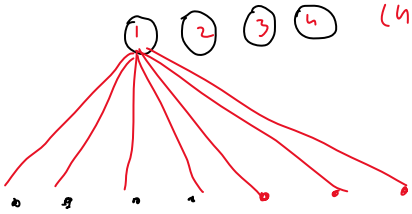
$r_t = [0.8, 0.2, 0.1, 0.9]$

gate

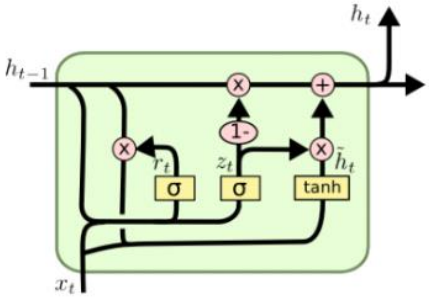
80% percent 20% 90%



$r_t = \sigma(W_r[h_{t-1}, x_t] + b_r)$



← (7x4) = 28 + 4



h_{t-1} modulated reseted

h_{t-1} → h_{t-1}, r_t, x_t → $r_t \otimes h_{t-1}$ → $\tilde{h}_t$

(1x7) (7x4)

(1x4) + (1x4) → ((1x4))

(1x4) → r_t

h_t

$h_{t-1} = [0.6, 0.6, 0.7, 0.1]$

$r_t = [0.8, 0.2, 0.1, 0.9]$

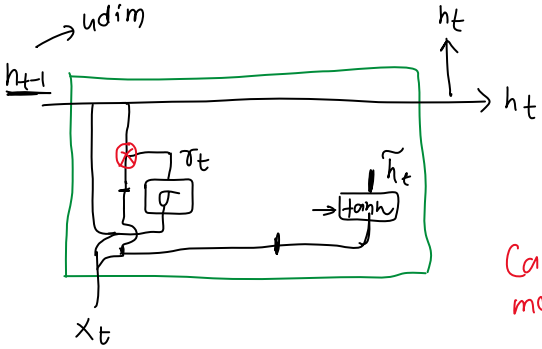
90%

power conflict tragedy revenge

$\tilde{h}_t = [0.7\ 0.2\ 0.1\ 0.2]$

x_t

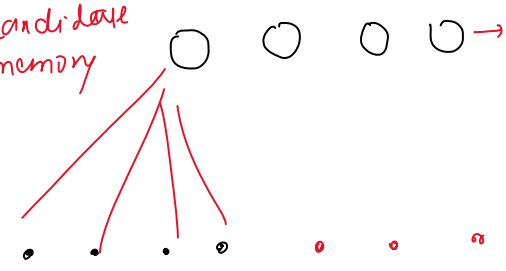
$h_{t-1} \otimes r_t = [0.48, 0.12, 0.07, 0.09]$ → x_t

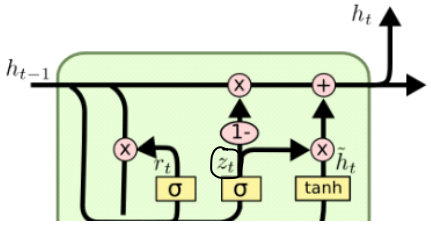


h_{t-1} / $\tilde{h}_t$ → z_t

$\tilde{h}_t = \tanh(W_c[h_{t-1} \otimes r_t, x_t] + b_c)$

Candidate memory



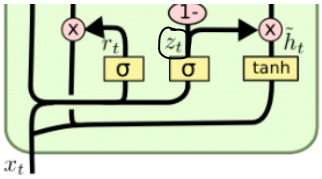


h_{t-1} → h_{t-1}, r_t, x_t → $r_t \otimes h_{t-1}$ → $\tilde{h}_t$

$\tilde{h}_t = [0.7\ 0.2\ 0.1\ 0.2]$

$h_{t-1} = [0.6\ 0.6\ 0.7\ 0.1]$

h_{t-1} → z_t



h_{t-1} x_t → z_t

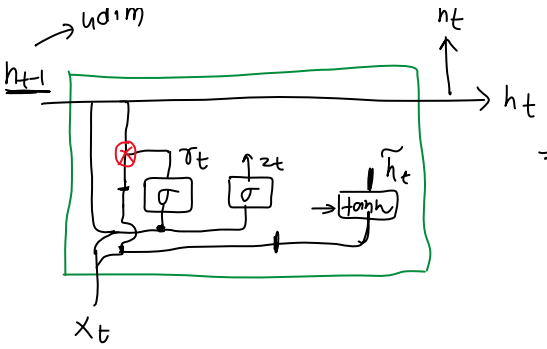
$[[- - - -]]$ (0-1)

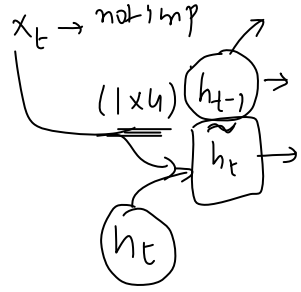
gate $z_t = \sigma(W_z[h_{t-1}, x_t] + b_z)$

h_{t-1} and $\tilde{h}_t$ (h_t)

x_t → important

x_t → not imp

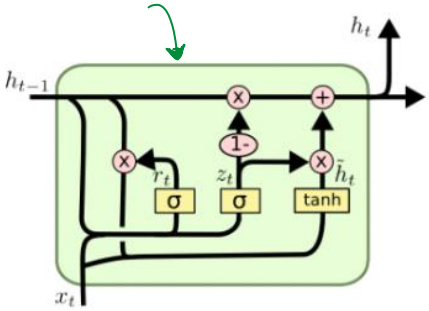




→ bias(4)

→ (7x4)

h_{t-1}



$\tilde{h}_t = [0.7\ 0.2\ 0.1\ 0.2]$

$h_{t-1} = [0.6\ 0.6\ 0.7\ 0.1]$

$z_t = [0.1\ 0.7\ 0.8\ 0.2]$

h_t [0.07 0.14 0.08 0.04]

0.1 0.1

$h_t = (1 - z_t) \otimes h_{t-1} \oplus z_t \otimes \tilde{h}_t$

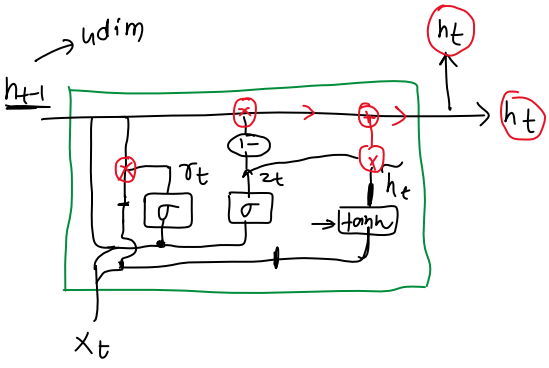
$(1-z_t)$ → 0.9

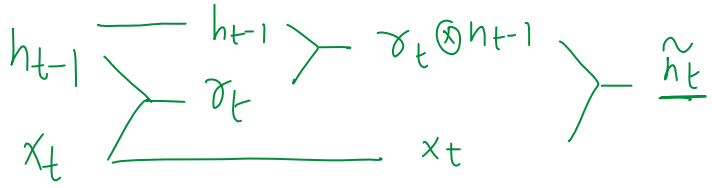
$(1-z_t)$ [0.9 0.3 0.2 0.8] [0.6 0.6 0.7 0.1]

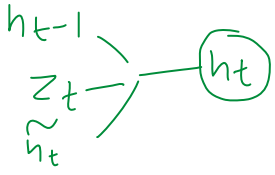
[0.54 0.18 0.14 0.08]

h_t [0.61 0.32 0.22 0.12]

power conflict tragedy revenge







LSTM vs GRU

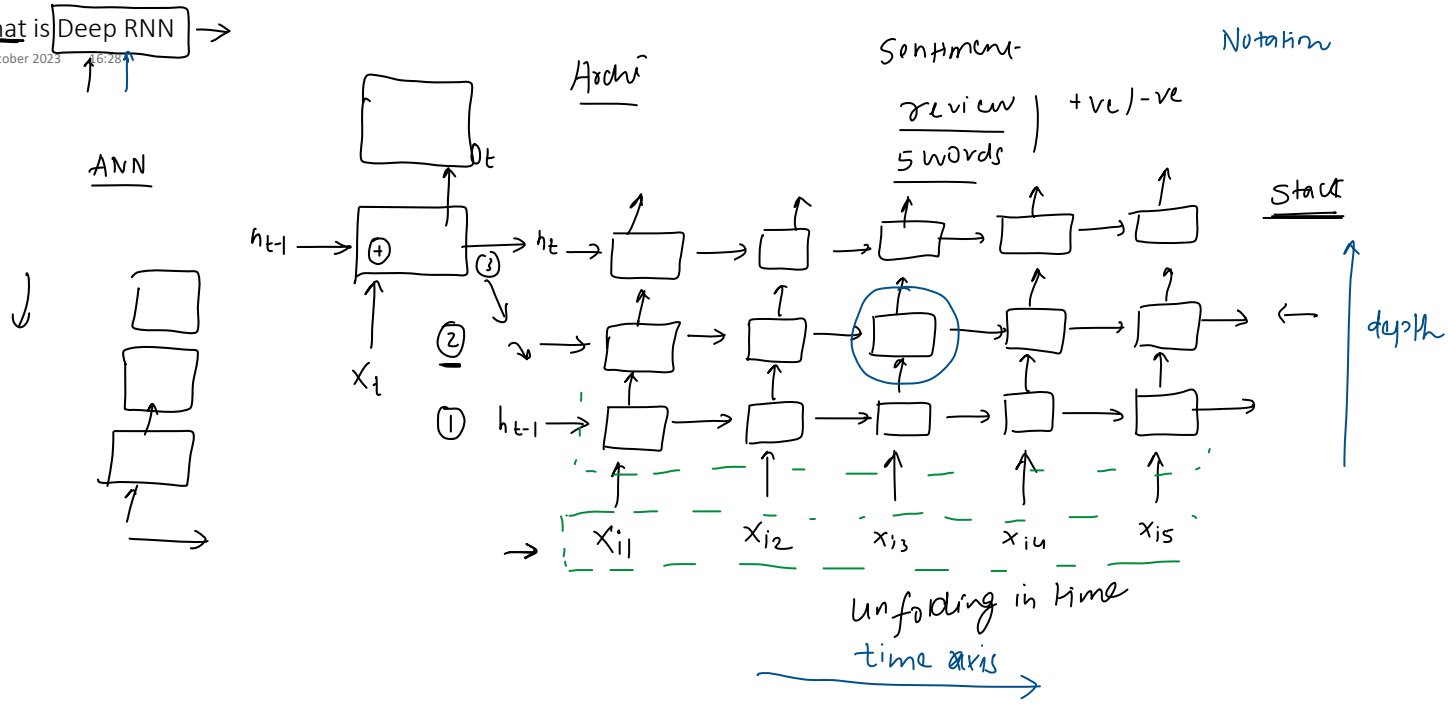
05 October 2023 16:45

Here are the main differences between LSTM and GRU:

1. Number of Gates:
   - LSTM: Has three gates — input (or update) gate, forget gate, and output gate.
   - GRU: Has two gates — reset gate and update gate.
2. Memory Units:
   - LSTM: Uses two separate states - the cell state (ct) and the hidden state (ht). The cell state acts as an "internal memory" and is crucial for carrying long-term dependencies.
   - GRU: Simplifies this by using a single hidden state (ht) to both capture and output the memory.
3. Parameter Count:
   - LSTM: Generally has more parameters than a GRU because of its additional gate and separate cell state. For an input size of d and a hidden size of h, the LSTM has 4 ×((d×h)+(h×h)+h)4×((d×h)+(h×h)+h)parameters.
   - GRU: Has fewer parameters. For the same sizes, the GRU has 3×((d×h)+(h×h)+h)3 ×((d×h)+(h×h)+h)parameters.
4. Computational Complexity:
   - LSTM: Due to the extra gate and cell state, LSTMs are typically more computationally intensive than GRUs.
   - GRU: Is simpler and can be faster to compute, especially on smaller datasets or when computational resources are limited.
5. Empirical Performance:
   - LSTM: In many tasks, especially more complex ones, LSTMs have been observed to perform slightly better than GRUs.
   - GRU: Can perform comparably to LSTMs on certain tasks, especially when data is limited or tasks are simpler. They can also train faster due to fewer parameters.
6. Choice in Practice:
   - The choice between LSTM and GRU often comes down to empirical testing. Depending on the dataset and task, one might outperform the other. However, GRUs, due to their simplicity, are often the first choice when starting out.

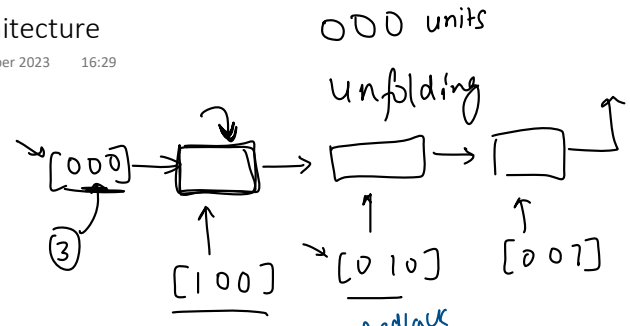
What is Deep RNN →

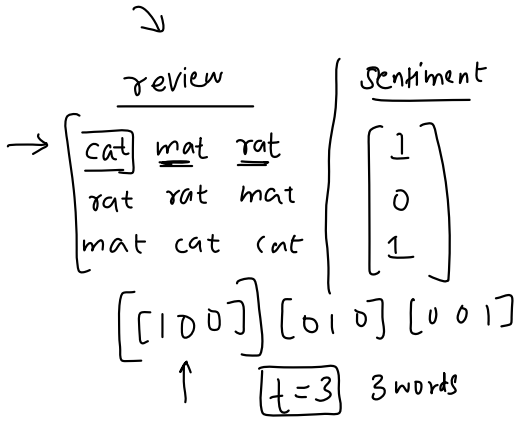
17 October 2023 16:28

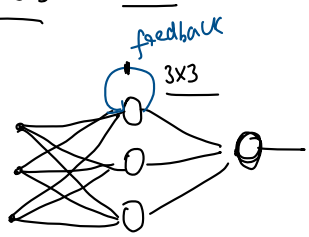


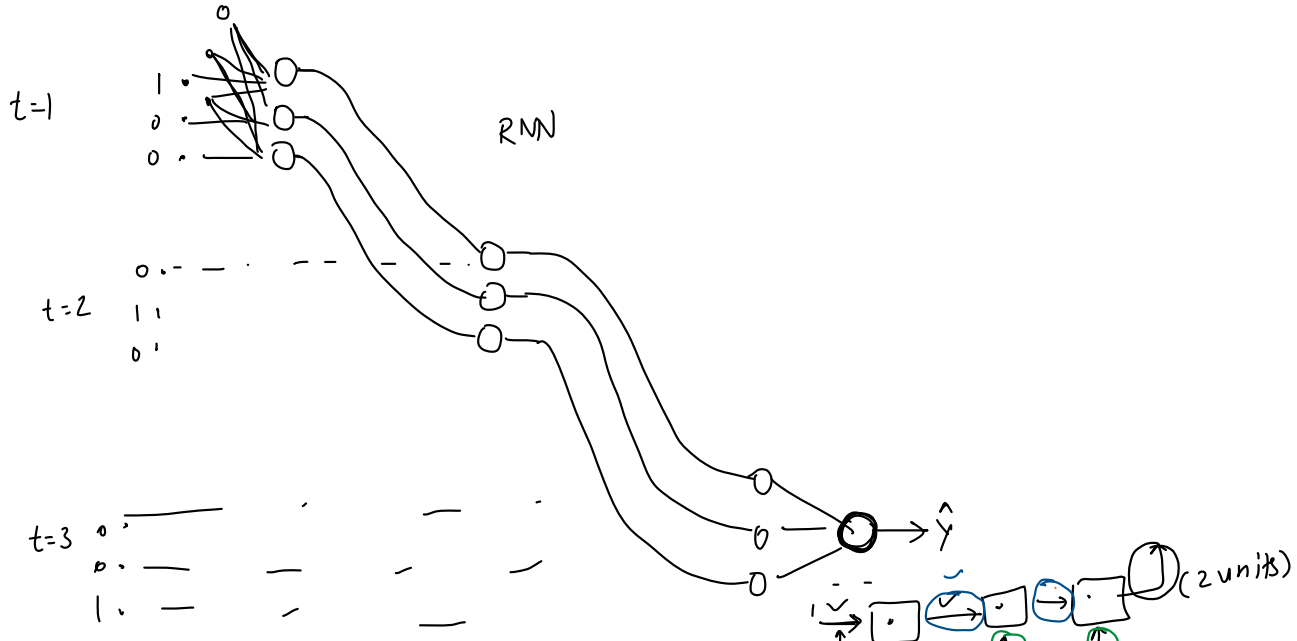
Architecture

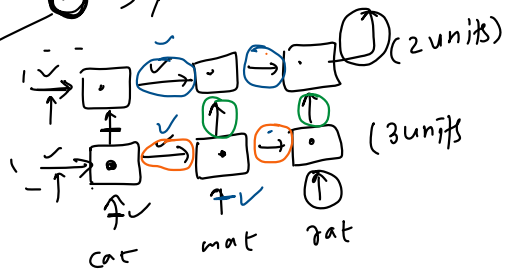
17 October 2023 16:29

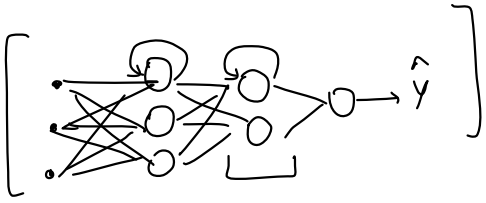


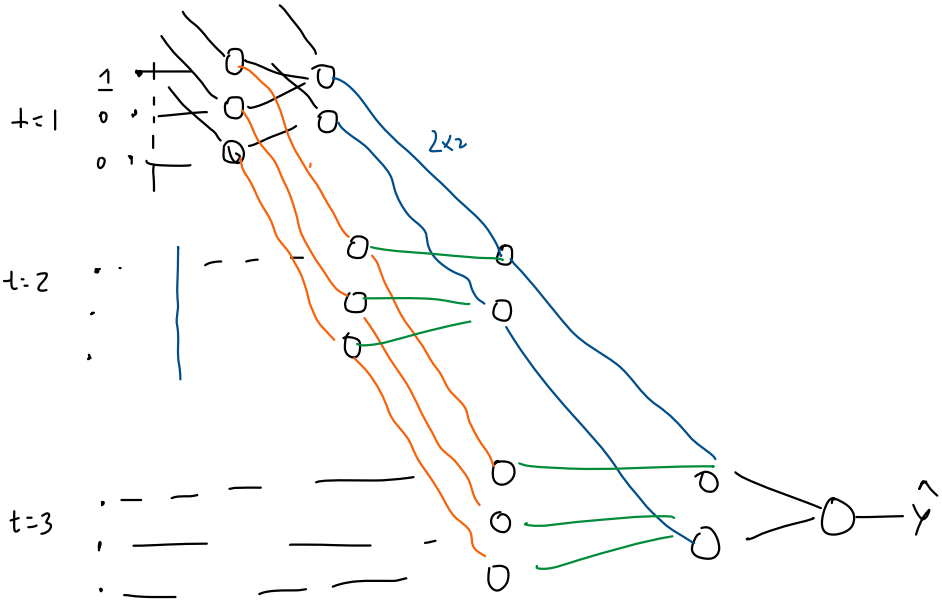








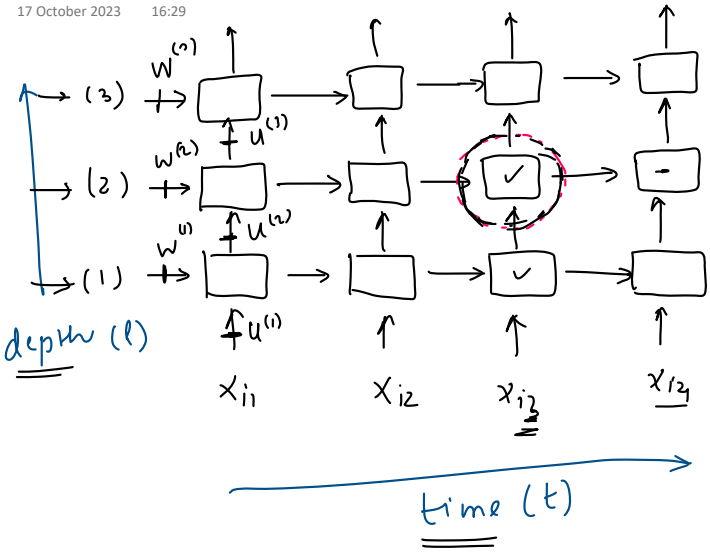


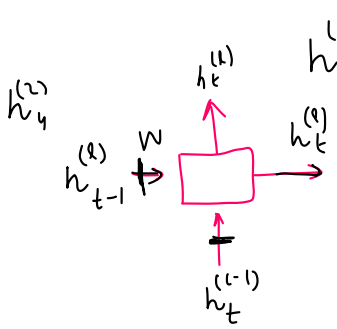



t=3

Notation

17 October 2023 16:29





$$h_t^{(l)} = \tanh\left(W^{(l)} h_{t-1}^{(l)} + U^{(l)} h_t^{(l-1)} + b^{(l)} \right)$$

Why and When to use?

17 October 2023 16:29

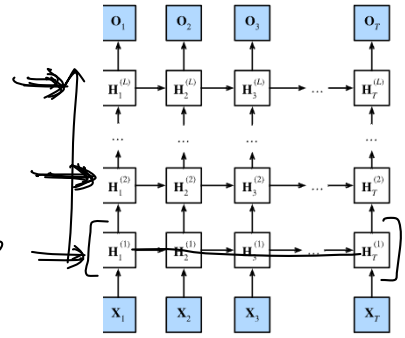
deep RNN

Stack

product

sentence

1. Hierarchical Representation ✓
2. Customization for Advanced Tasks



encoder-decoder

machine

deep RNNs

love hate amazing terrible

deep RNN

Audio is bad

Display is great

Overall I am happy

→ sentiment

When to use Deep RNNs?

Complex tasks

Speech recog

Machine translation

Large datasets

Overfitting

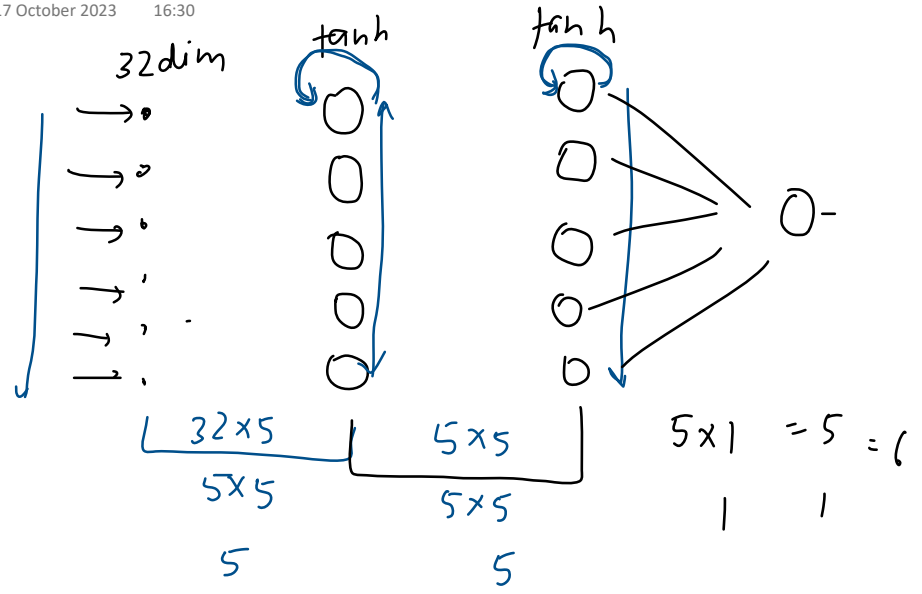
Computational

Simpler Models

Deep RNN

Code Example

17 October 2023 16:30



→ embedded

10000 x 32 = 3,20,000

simple RNN 1

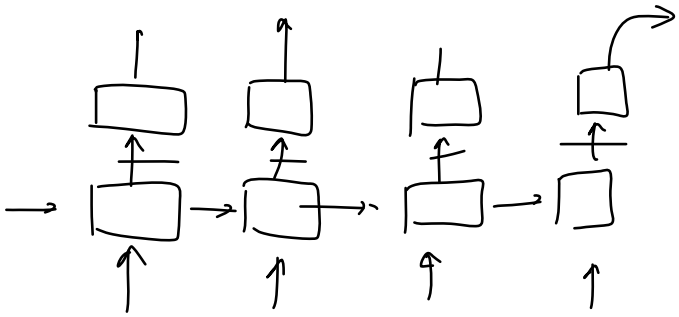
190

simple RNN 2

55

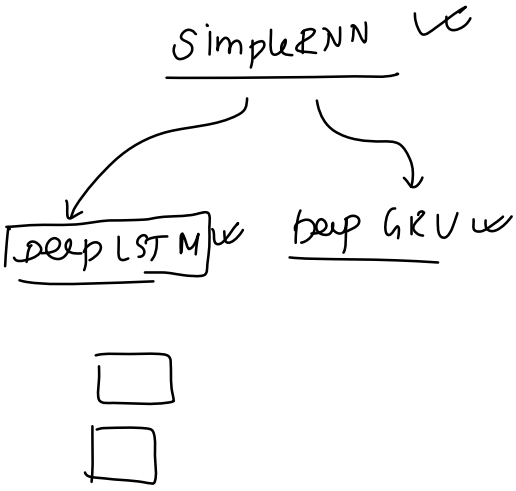
Dense

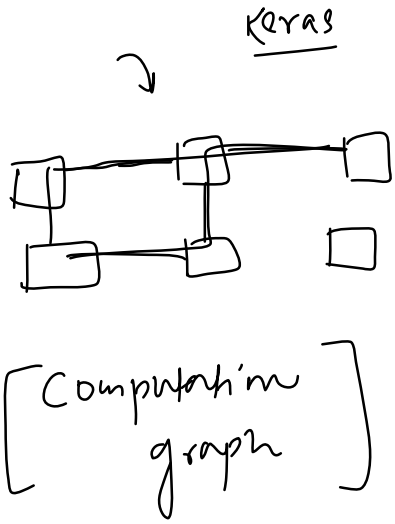
6



Variants

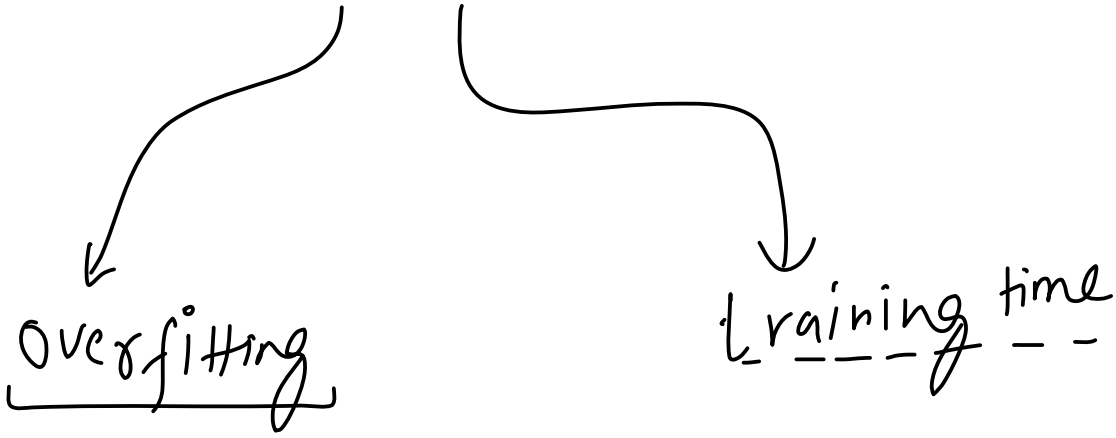
17 October 2023 16:30





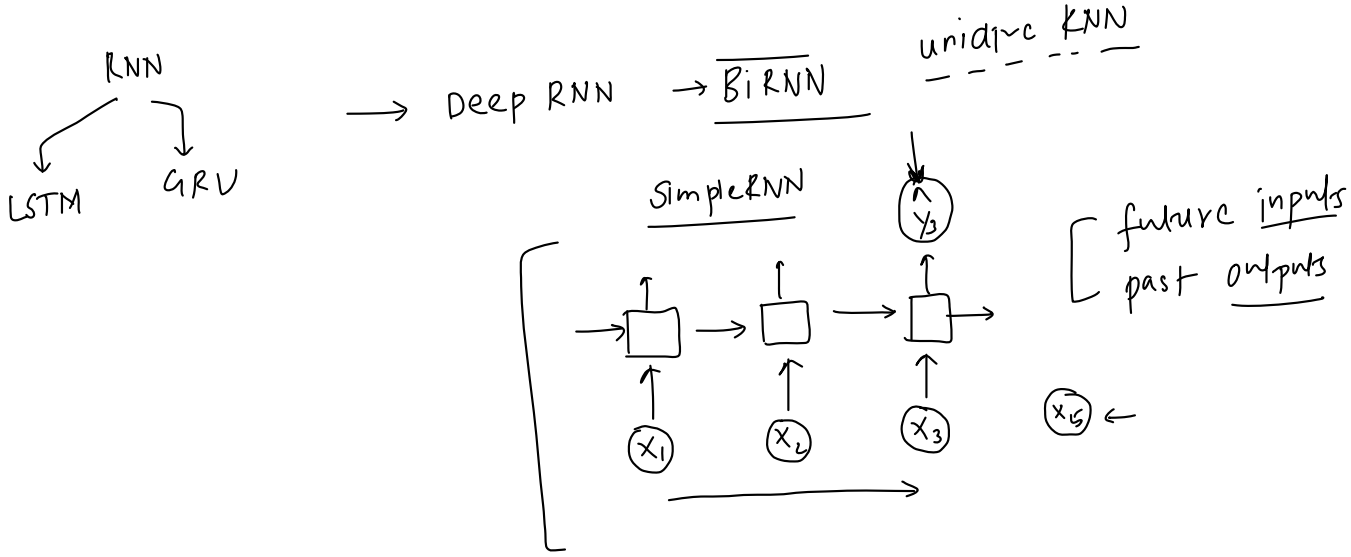
Disadvantages

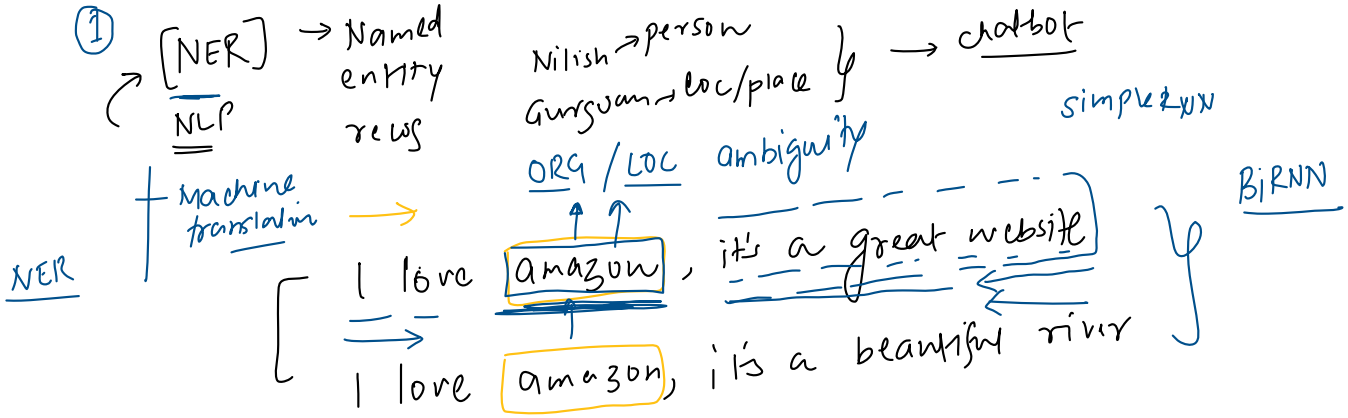
17 October 2023 16:30



The Why?
26 October 2023 15:19

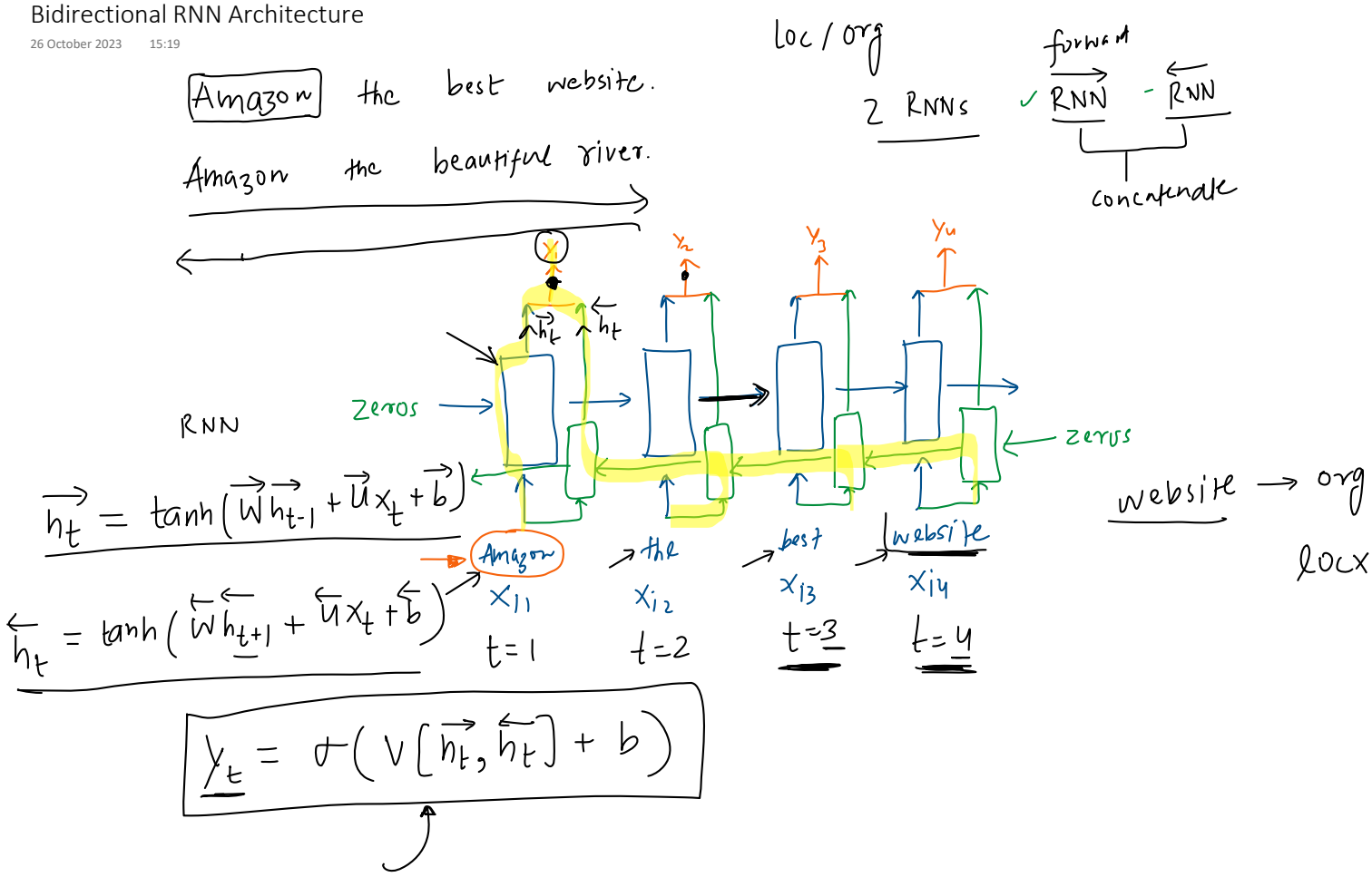
BiRNN





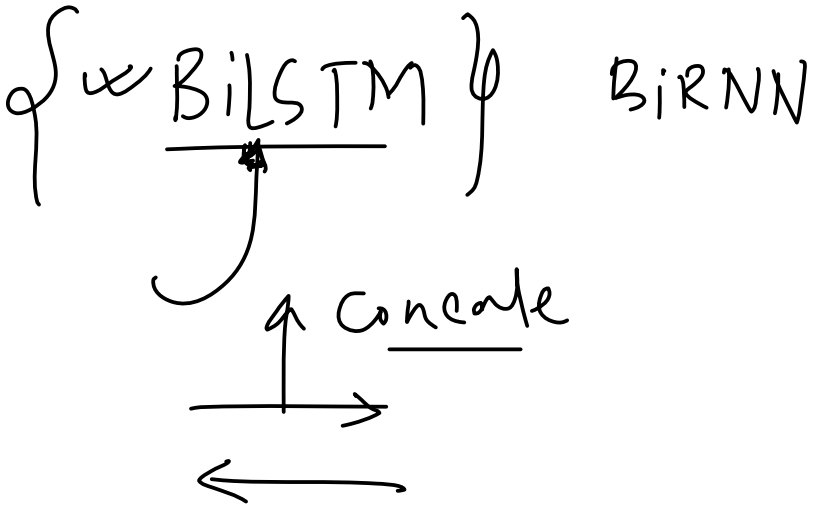
Bidirectional RNN Architecture

26 October 2023 15:19



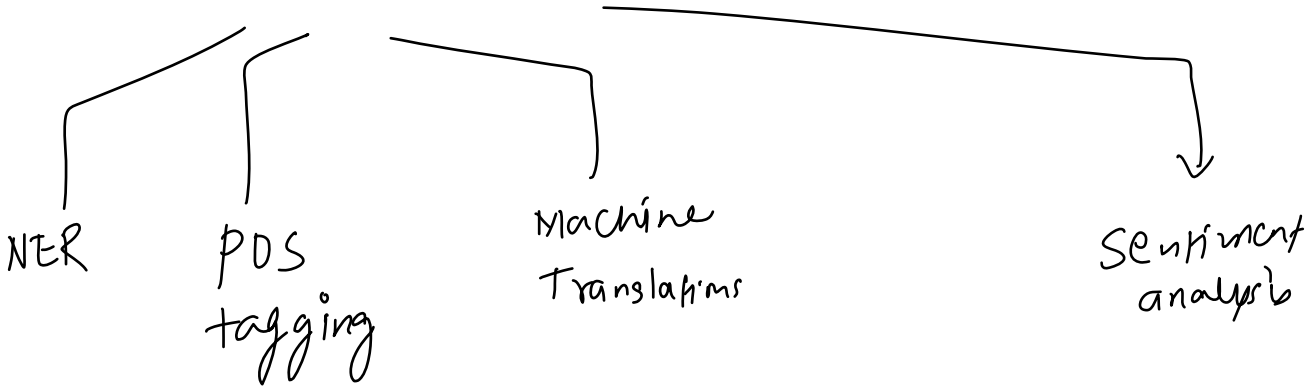
Code

26 October 2023 15:21



Applications and Drawbacks

26 October 2023 15:21



[Time series forecasting]

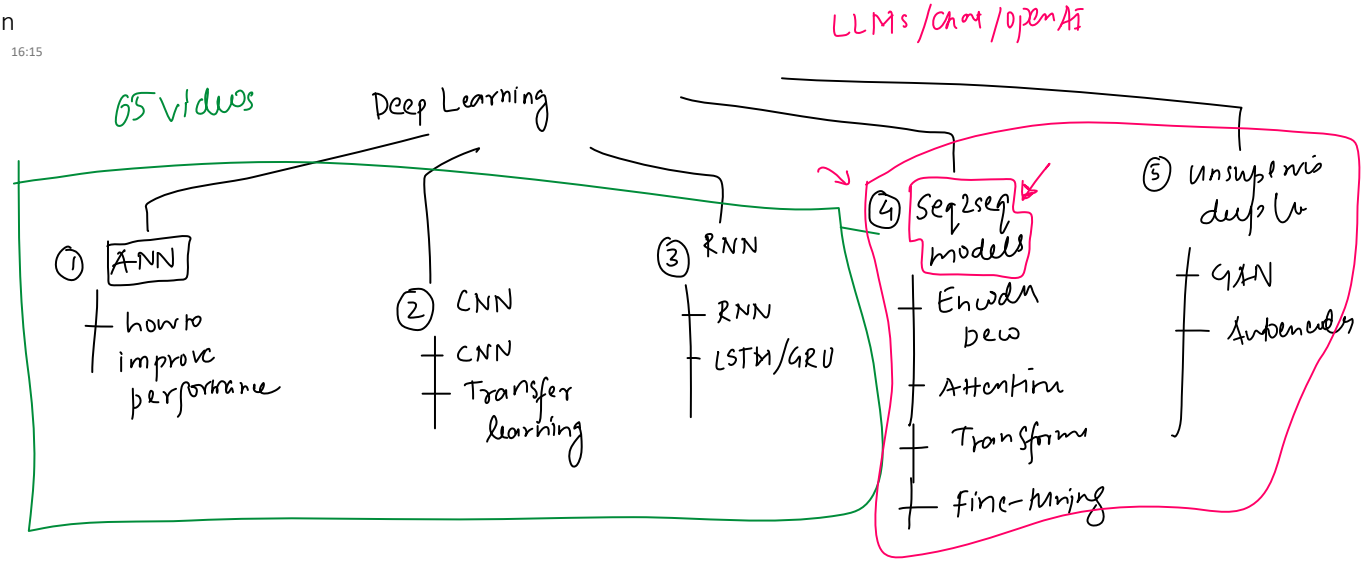
→ ←

→ Complexity → 190 → 380 → training → overfitting
double

→ → ← [Speech recog] → bi rnn
↳ latency → slow

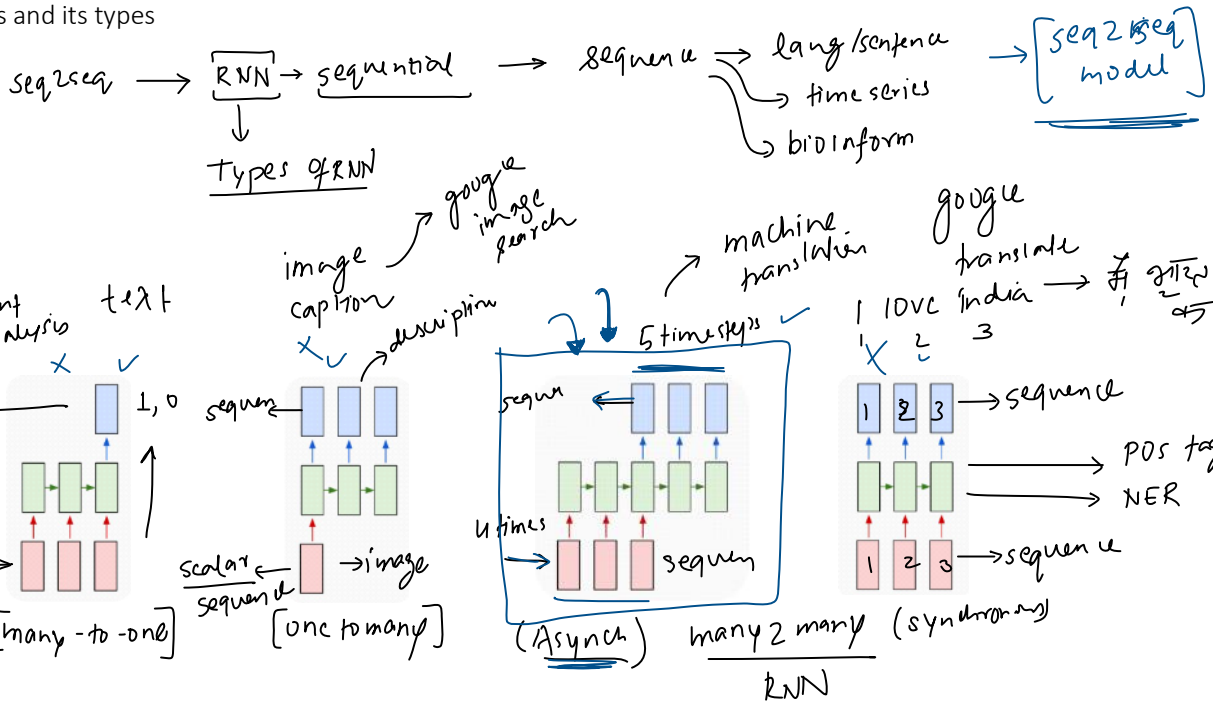
Introduction

16 November 2023 16:15



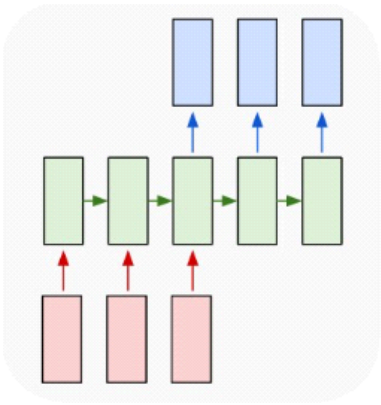
Sequence Tasks and its types

16 November 2023 16:16



Seq2Seq Tasks

16 November 2023 16:16



NLP

Seq2seq → machine trans

1) text summari → seq → seq

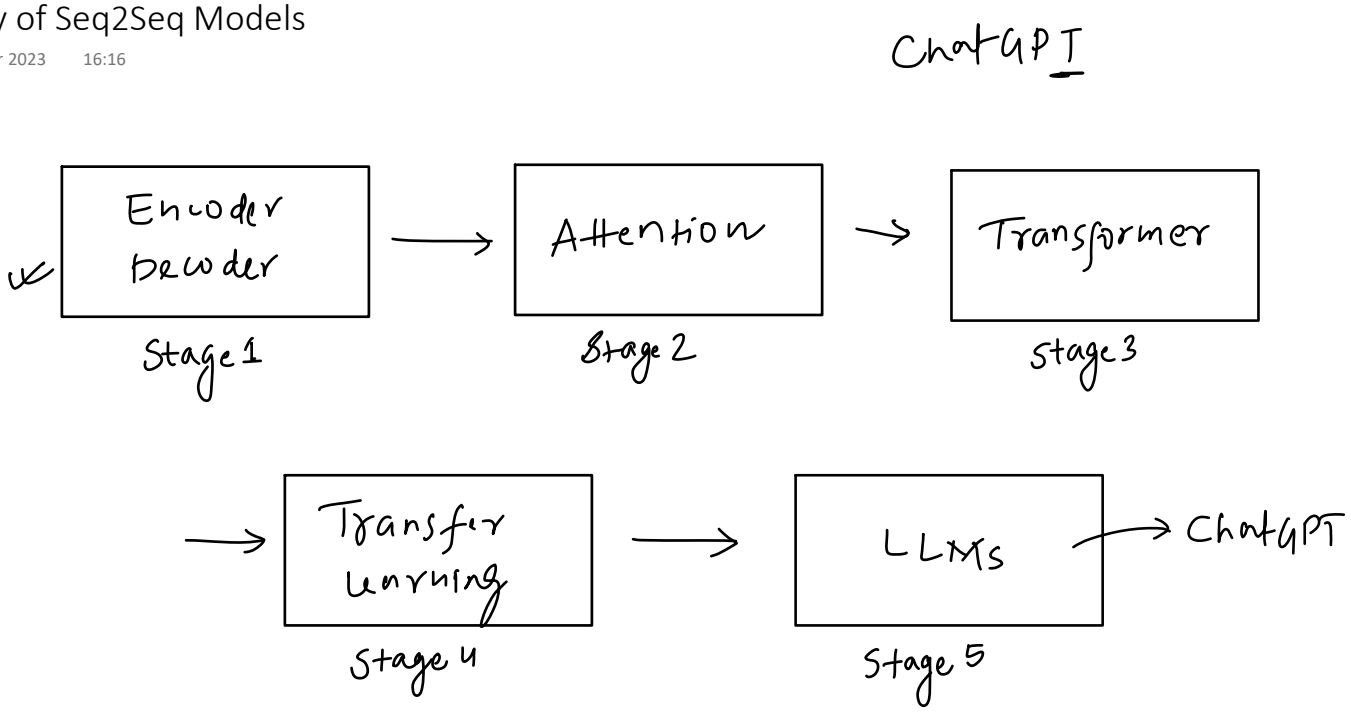
2) Question answer → seq → seq (knowledge base)

3) chatbot → input (text) → output (text)

4) speech-to-text → seq seq → seq2seq → chatgpt

History of Seq2Seq Models

16 November 2023 16:16



Stage 1 - Encoder Decoder Architecture

18 November 2023 16:16

2014 Seminal

Sequence to Sequence Learning with Neural Networks

Ilya Sutskever
Google
ilyasu@google.com

Oriol Vinyals
Google
vinyals@google.com

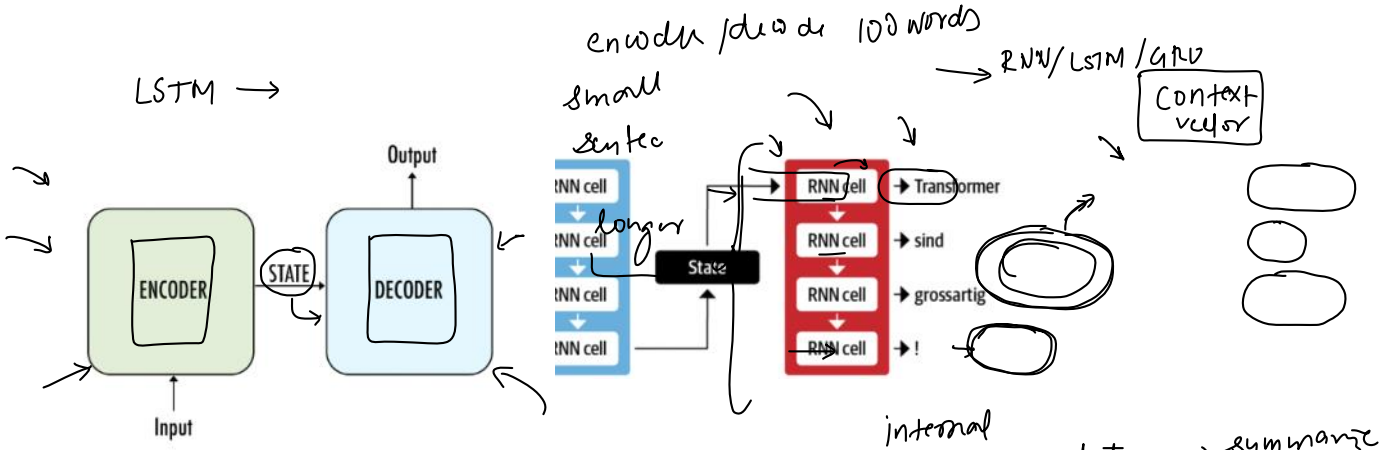
Quoc V. Le
Google
qvl@google.com

Abstract

Deep Neural Networks (DNNs) are powerful models that have achieved excellent performance on difficult learning tasks. Although DNNs work well whenever large labeled training sets are available, they cannot be used to map sequences to sequences. In this paper, we present a general end-to-end approach to sequence learning that makes minimal assumptions on the sequence structure. Our method uses a multilayered Long Short-Term Memory (LSTM) to map the input sequence to a vector of a fixed dimensionality, and then another deep LSTM to decode the target sequence from the vector. Our main result is that on an English to French translation task from the WMT'14 dataset, the translations produced by the LSTM achieve a BLEU score of 34.8 on the entire test set, where the LSTM's BLEU score was penalized on out-of-vocabulary words. Additionally, the LSTM did not have difficulty on long sentences. For comparison, a phrase-based SMT system achieves a BLEU score of 33.3 on the same dataset. When we used the LSTM to rerank the 1000 hypotheses produced by the aforementioned SMT system, its BLEU score increases to 36.5, which is close to the previous best result on this task. The LSTM also learned sensible phrase and sentence representations that are sensitive to word order and are relatively invariant to the active and the passive voice. Finally, we found that reversing the order of the words in all source sentences (but not target sentences) improved the LSTM's performance markedly, because doing so introduced many short term dependencies between the source and the target sentence which made the optimization problem easier.

Ilya Sutskever → cofounder OpenAI

seq2seq
↓
diff
↳ encoder decoder



seq2seq → input → output
machine translation
eng → hindi

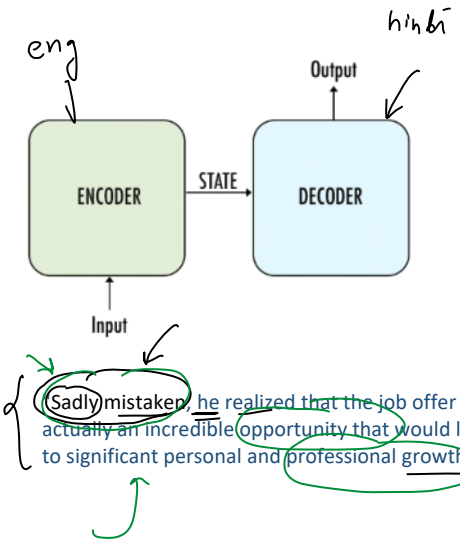
I love India → मैं भारत से प्यार करता हूँ

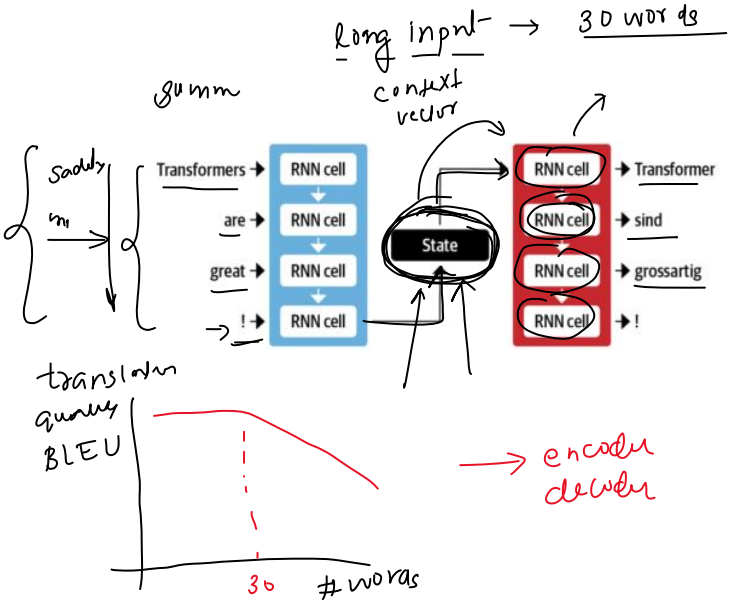
process word by word → compress

Attention mechanism

Stage 2 - Attention Mechanism

20 November 2023 10:59



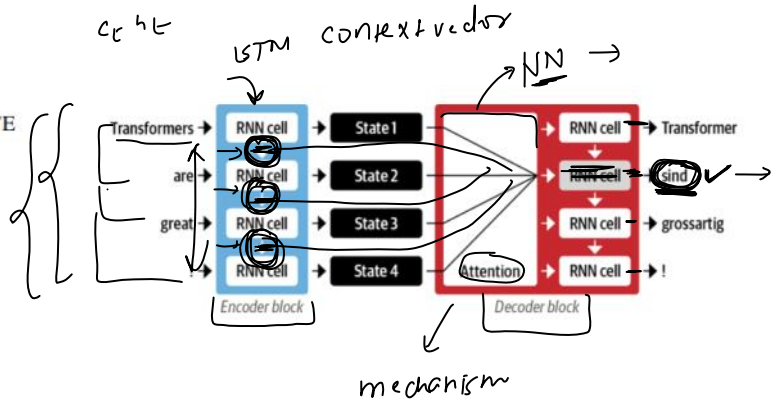


2015 Attention

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio***
Université de Montréal



ABSTRACT

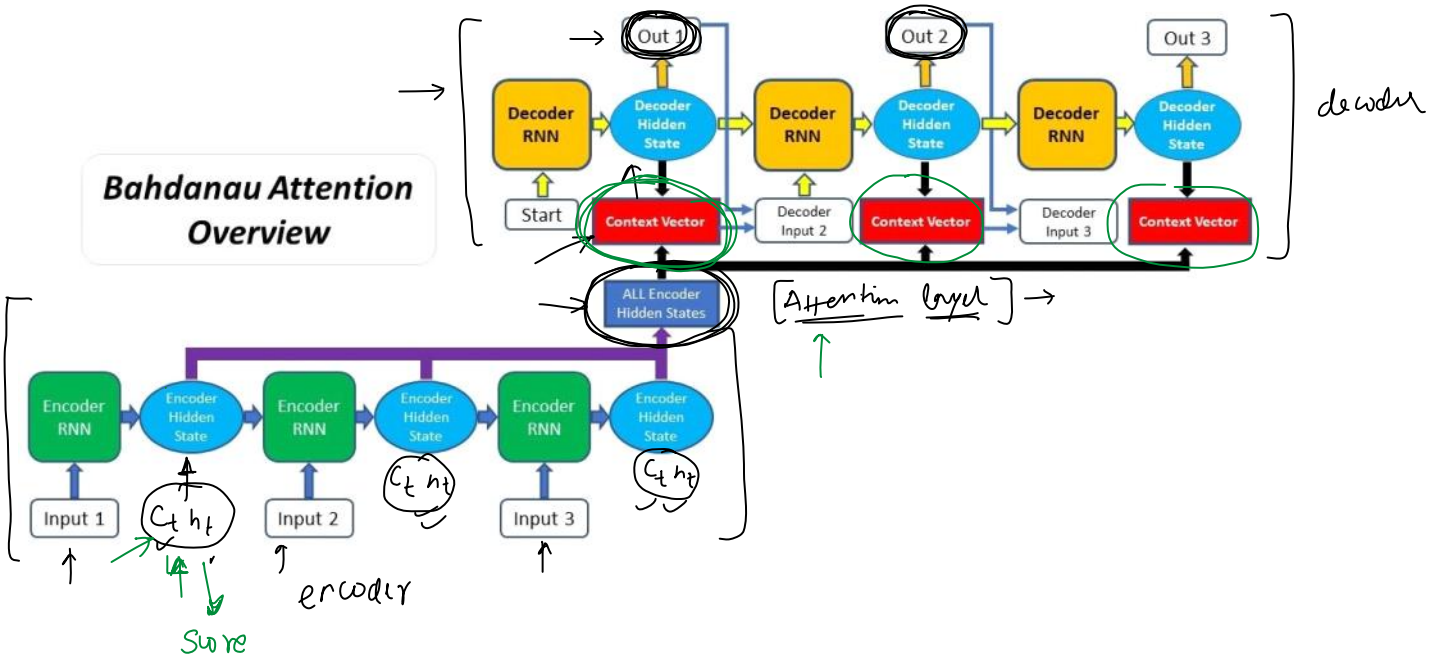
Neural machine translation is a recently proposed approach to machine translation. Unlike the traditional statistical machine translation, the neural machine translation aims at building a single neural network that can be jointly tuned to maximize the translation performance. The models proposed recently for neural machine translation often belong to a family of encoder–decoders and encode a source sentence into a fixed-length vector from which a decoder generates a translation. In this paper, we conjecture that the use of a fixed-length vector is a bottleneck in improving the performance of this basic encoder–decoder architecture, and propose to extend this by allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word, without having to form these parts as a hard segment explicitly. With this new approach, we achieve a translation performance comparable to the existing state-of-the-art phrase-based system on the task of English-to-French translation. Furthermore, qualitative analysis reveals that the (soft-)alignments found by the model agree well with our intuition.

1 INTRODUCTION

Neural machine translation is a newly emerging approach to machine translation, recently proposed by Kalchbrenner and Blunsom (2013), Sutskever *et al.* (2014) and Cho *et al.* (2014b). Unlike the traditional phrase-based translation system (see, e.g., Koehn *et al.*, 2003) which consists of many small sub-components that are tuned separately, neural machine translation attempts to build and train a single, large neural network that reads a sentence and outputs a correct translation.

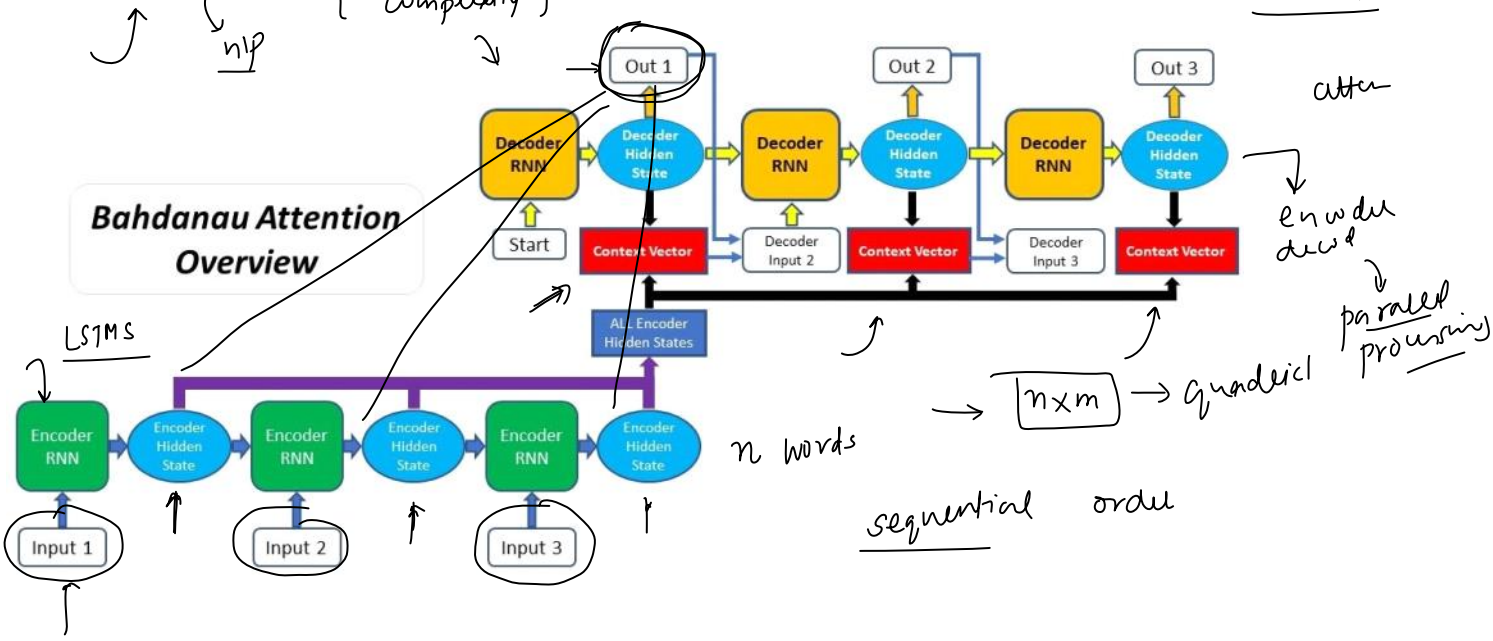
Most of the proposed neural machine translation models belong to a family of *encoder–decoders* (Sutskever *et al.*, 2014; Cho *et al.*, 2014a), with an encoder and a decoder for each language, or involve a language-specific encoder applied to each sentence whose outputs are then compared (Hermann and Blunsom, 2014). An encoder neural network reads and encodes a source sentence into a fixed-length vector. A decoder then outputs a translation from the encoded vector. The whole encoder–decoder system, which consists of the encoder and the decoder for a language pair, is jointly trained to maximize the probability of a correct translation given a source sentence.

A potential issue with this encoder–decoder approach is that a neural network needs to be able to compress all the necessary information of a source sentence into a fixed-length vector. This may make it difficult for the neural network to cope with long sentences, especially those that are longer than the sentences in the training corpus. Cho *et al.* (2014b) showed that indeed the performance of a basic encoder–decoder deteriorates rapidly as the length of an input sentence increases.


Bahdanau Attention Overview
Out 1
Out 2
Out 3
Decoder RNN
Decoder Hidden State
Start
Context Vector
Decoder Input 2
Decoder Input 3
ALL Encoder Hidden States
Encoder RNN
Encoder Hidden State
Input 1
Input 2
Input 3
decoder
Attention layer
encoder
Score

Stage 3 - Transformers

20 November 2023 12:18



Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

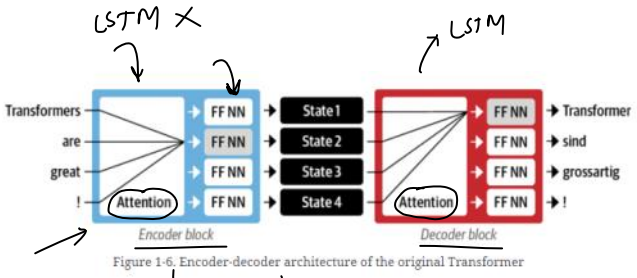


Figure 1-6. Encoder-decoder architecture of the original Transformer

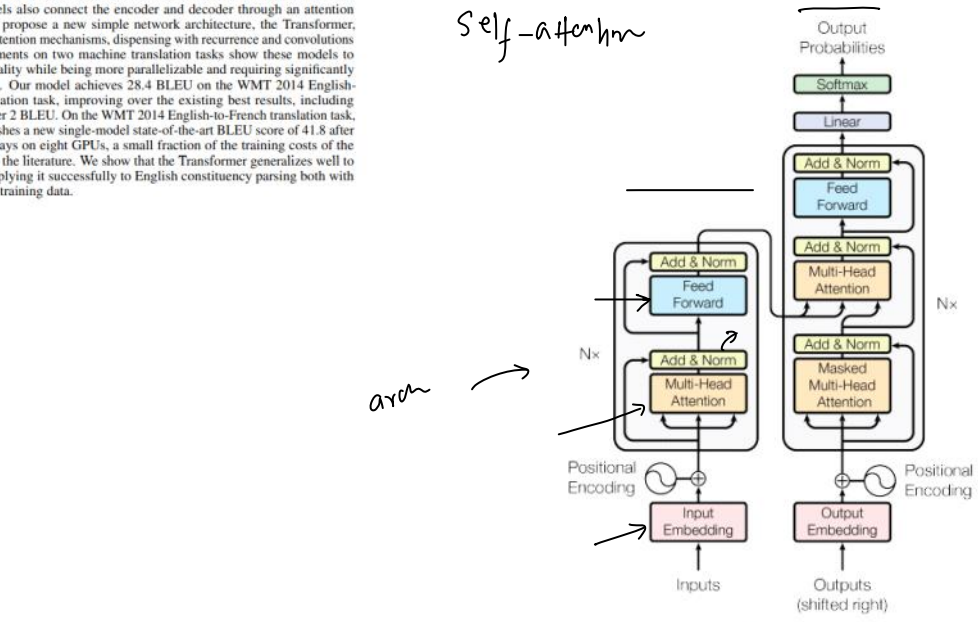
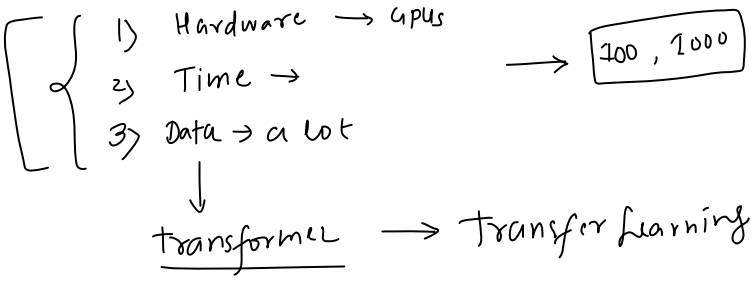


Figure 1: The Transformer - model architecture.

Stage 4 - Transfer Learning

20 November 2023 15:39



ULMFiT

Universal Language Model Fine-tuning for Text Classification

Jeremy Howard*
fast.ai
University of San Francisco
j@fast.ai

Sebastian Ruder*
Insight Centre, NUI Galway
Aylien Ltd., Dublin
sebastian@ruder.io

2018 → vision

transfer → NLP tasks

ULMFit

primary pretrain

Abstract

Inductive transfer learning has greatly impacted computer vision, but existing approaches in NLP still require task-specific modifications and training from scratch. We propose Universal Language Model Fine-tuning (ULMFiT), an effective transfer learning method that can be applied to any task in NLP, and introduce techniques that are key for fine-tuning a language model. Our method significantly outperforms the state-of-the-art on six text classification tasks, reducing the error by 18-24% on the majority of datasets. Furthermore, with only 100 labeled examples, it matches the performance of training from scratch on 100× more data. We open-source our pretrained models and code[1].

1 Introduction

Inductive transfer learning has had a large impact on computer vision (CV). Applied CV models (including object detection, classification, and segmentation) are rarely trained from scratch, but instead are fine-tuned from models that have been pretrained on ImageNet, MS-COCO, and other datasets (Sharif Razavian et al., 2014; Long et al., 2015a; He et al., 2016; Huang et al., 2017).

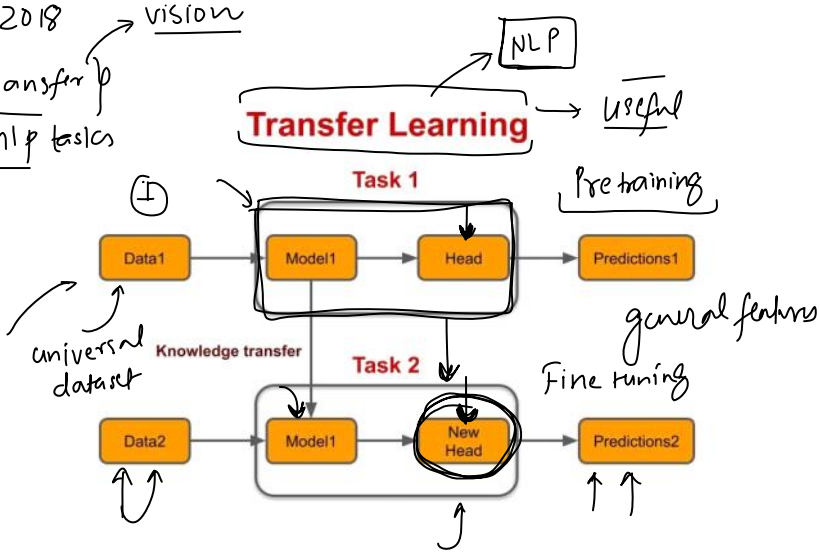
Text classification is a category of Natural Language Processing (NLP) tasks with real-world applications such as spam, fraud, and bot detection (Jindal and Liu, 2007; Ngai et al., 2011; Chu et al., 2012), emergency response (Caragea et al., 2011), and commercial document classification, such as for legal discovery (Roitblat et al., 2010).

While Deep Learning models have achieved state-of-the-art on many NLP tasks, these models are trained from scratch, requiring large datasets, and days to converge. Research in NLP focused mostly on *transductive* transfer (Blitzer et al., 2007). For *inductive* transfer, fine-tuning pretrained word embeddings (Mikolov et al., 2013), a simple transfer technique that only targets a model's first layer, has had a large impact in practice and is used in most state-of-the-art models. Recent approaches that concatenate embeddings derived from other tasks with the input at different layers (Peters et al., 2017; McCann et al., 2017; Peters et al., 2018) still train the main task model from scratch and treat pretrained embeddings as fixed parameters, limiting their usefulness.

In light of the benefits of pretraining (Erhan et al., 2010), we should be able to do better than *randomly initializing* the remaining parameters of our models. However, inductive transfer via fine-tuning has been unsuccessful for NLP (Mou et al., 2016). Dai and Le (2015) first proposed fine-tuning a language model (LM) but require millions of in-domain documents to achieve good performance, which severely limits its applicability.

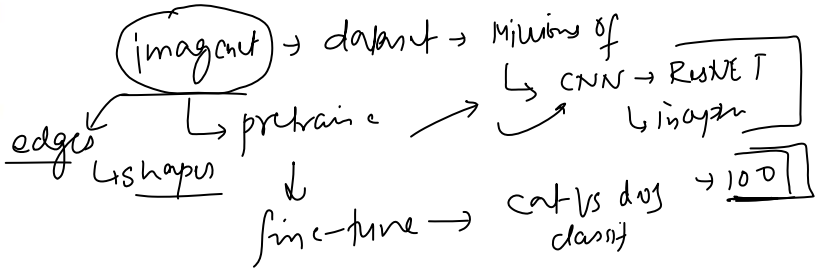
We show that not the idea of LM fine-tuning but our lack of knowledge of how to train them effectively has been hindering wider adoption. LMs overfit to small datasets and suffered catastrophic forgetting when fine-tuned with a classifier. Compared to CV, NLP models are typically more shallow and thus require different fine-tuning methods.

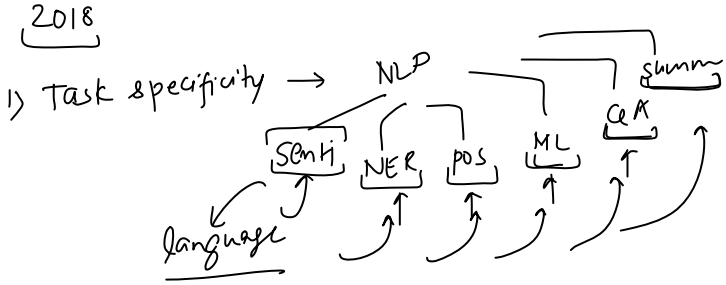
We propose a new method, Universal Language Model Fine-tuning (ULMFiT) that addresses these issues and enables robust inductive transfer learning for any NLP task, akin to fine-tuning ImageNet



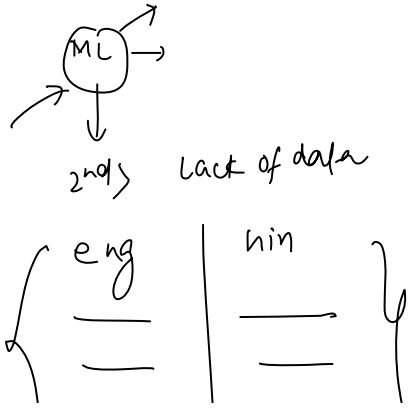
Transfer learning (TL) is a technique in which knowledge learned from a task is re-used in order to boost performance on a related task.

For example, for image classification, knowledge gained while learning to recognize cars could be applied when trying to recognize trucks.

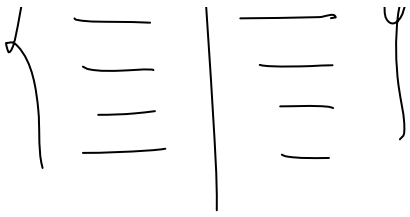




NLP → transfer learning



NLP → transfer learning



Pretraining → Machine transl X

→ language modelling → pretraining → sucessful → (2)

NLP task → NLP/DL model next word pred

I live in india. and the capital is New Delhi

Language modelling as a Pretraining task

→ unsupervised pretrain task

1) Rich feature learning

The hotel was exceptionally clean, yet the service was bad / pathetic

→ know trans

↓

text classi / quest. | textsumm | NER | pos

mt (labw → supervised labeled

eng | nin → unsupervised task

2) Huge avail of data

pdf → dataset labelling

[ULMFiT]

fine tuning

X transformer

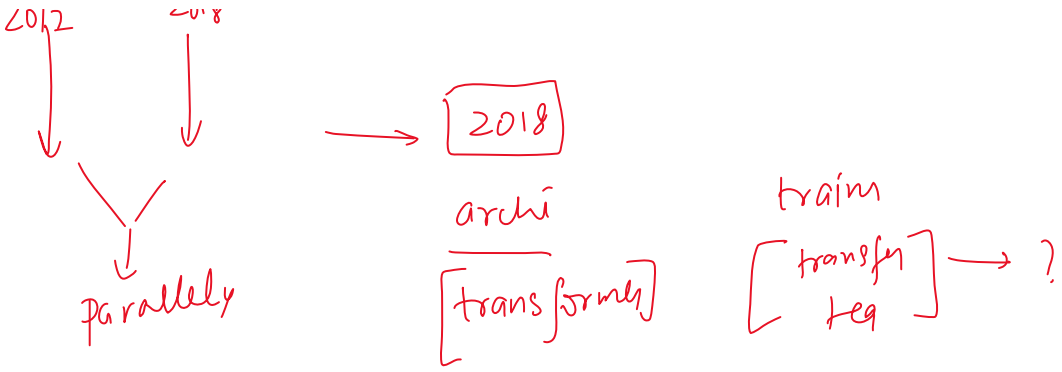
AWD LSTM → wikipedia → classifier → imdb / yelp / new data → model → test → State of the art

Unsupervised pretrain Language modelling

Scratch → 10000 rows

100 row → better →

2012 2018


2012
2018
parallely
2018
archi
transformer
train
transfer
eq
?

Stage 5 - LLMs

20 November 2023 18:51