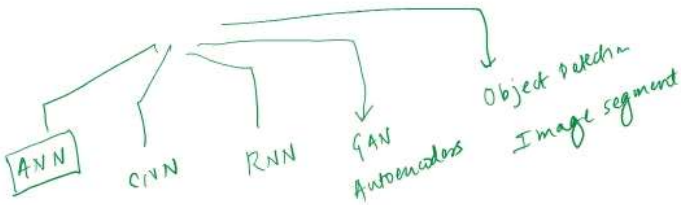
Curriculum

Tuesday, February 15, 2022 3:30 PM



Features

Tuesday, February 15, 2022 3:31 PM

1) Well Researched
2) Easy to consume
3) Well structured → 100 days
4) Tensorflow + Keras
5) Projects

Prerequisites

Tuesday, February 15, 2022 3:31 PM

1) Python →
2) Basics of ML
3) Linear Algebra → 3 Blue 1 Brown / Linear Alg → 5

Extra Content

Tuesday, February 15, 2022 3:32 PM

1) Deep learning Roadmap
2) Deep learning Project ideas
3) Projects
4) Interview questions

→ Next video → Thu

→ 10th Apr
→ ANN → CNN → 3-4 months

What is Deep Learning?

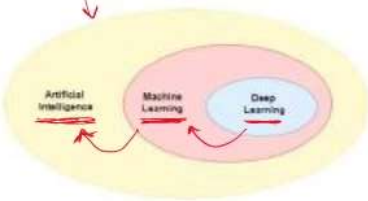
16 February 2022 06:32

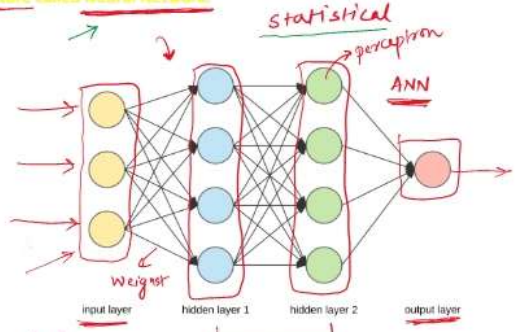
Deep Learning is a subfield of Artificial Intelligence and Machine Learning that is inspired by the structure of a human brain. Deep learning algorithms attempt to draw similar conclusions as humans would by continually analyzing data with a given logical structure called Neural Network.

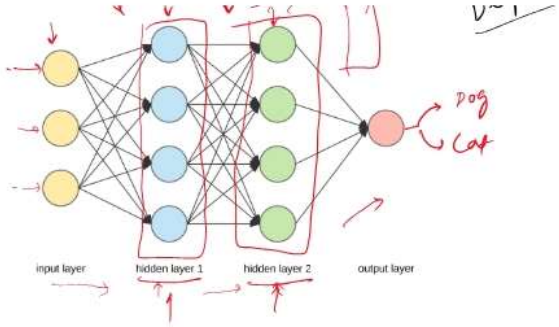
learn from data

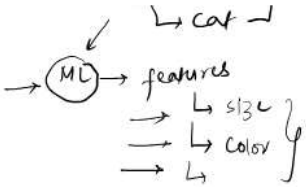
input | output

statistical









- ANN →
- CNN → convolutional NN ✓ image
- RNN → recurrent NN → speech/text
- GAN → generate (text) images

why (dL) getting so famous?

1) Applicability →

2) Performance → state-of-the-art

Applications

Go → AlphaGo → AI

5 → 4

human experts

selfdriving

drugs

→ chemistry

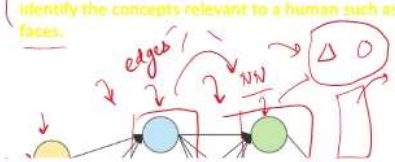
Deep learning is part of a broader family of machine learning methods based on artificial neural networks with representation learning.

Deep Learning Algorithms uses multiple layers to progressively extract higher-level features from the raw input. For example, in image processing, lower layers may identify edges, while higher layers may identify the concepts relevant to a human such as digits or letters or faces.

1) Representation

feature extraction engineering

features



Deep learning Dog/cat classifier

→ Dog

→ cat

Deep Learning VS Machine Learning

18 February 2022 08:33

1. Data Dependency
2. Hardware Dependency
3. Training Time
4. Feature Selection
5. Interpretability

costly DL

Data ↑ hungry

Data ↑ performance ↑

Representation learning → features extract

CPU → run slow

matrix multiplication → GPU → powerful

complex

↳ training time - high → weeks

ML → low minutes - hrs

less data

ML wins

Prediction

DL → fast

ML

KNN

PL → Algorithm → feature

Resume based

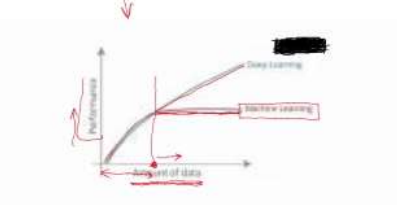
↳ predict → Y/N

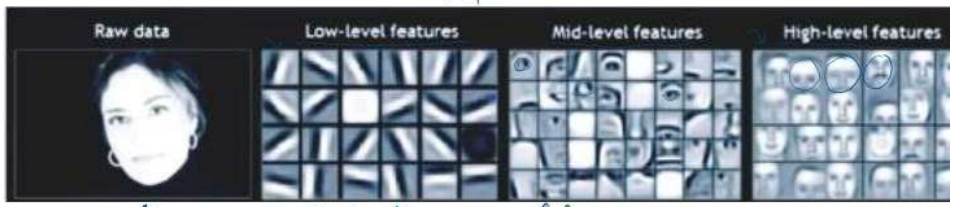
↳ features

domain expert → 12th marks, 10th marks

teachers → No. of ECA, No. of Course, quality of College

Deep learning





feature → interpre

Deep learning

comment → ban → user

Black box

Dog

Cat

logistic

cgpa | iq | placement

w_1 w_2 N

DT

flow chart

explainability

High ↑

Machine learning →

Why now?

Wednesday, February 16, 2022 6:59 AM

1960's → Alan Turing

→ 2012

Why now

- Hardware
- Datasets
- Frameworks
- Community
- Architecture

Data hungry

Deep learning

ML → 100 rows → 1000 rows

2010

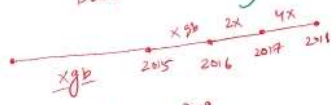
Twitter, Snapchat

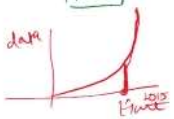
Data

Smart phone

Internet pricing → 2015

Free





unlabelled

puppy

create labels

Microsoft, google

image classi → dog, cat → training

Unlabelled

↳ labelled data

public dataset → DL

cat

dog

1000's public dataset

↓

Kaggle, google

Image → Microsoft COCO

Video → YouTube 8M

6.1 million videos

Text → SQuAD

Wikipedia 1,50,000

Audio → google AudioSet

↓

20,00,000 sound clips

↙ Youtube

600 categories

Reason #2 → Hardware

(10) → custom hardware

i7 → transistor → 2 years (2)

→ cost → half

→ electronic

↳ improve

deep learning → 128 MB → 12 GB

NVIDIA → CUDA → GPU

Deep learning → lot of data → matrix operations → CPU → low speed → 2010

gainers → DL researchers

parallel processing → render

CPU → 10 - 20 times

parallel processing → GPU

GPU → 10-20 times reduce CPU

FPGA → ECE → fast, low power, reprog → custom

↳ expensive

Xilinx ~ Bing → AI

ASIC

→ TPU → Tensor processing unit (Google) → google colab

→ Edge TPU → Drone, watch, smart glass

→ NPU → Neural Processing units → accelerate

start → CPU

large → GPU, TPU

smartphone → mobile CPU, GPU, DSP, NPU

smart watch, glass → Edge TPU, NPUs

Factor #3 → Frameworks / Libraries ← Deep Learning

train → difficult → scratch → code

sklearn → ML

2011 → Deep Learning

Dist Belief → google

↳ 2015 → Tensorflow → powerful → Difficult

Tensorflow by Google → Industry

PyTorch Facebook → Research

2016 → PyTorch

AI Research

Reasearch servers

2018 → Caffe2

PyTorch

format wrd → Keras → user

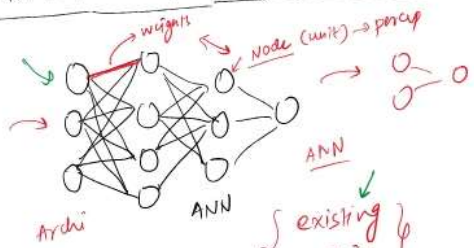
2019 TF 2.0 → Keras tf.f

PyTorch → Tensorflow

Drop-down GUI → export → PyTorch, TF

AutoML

Factor #4 → Deep Learning Architectures



CNN RNN

D NN archi

experiment → time, effort, money

Transfer learning

State-of-the-art best accuracy

ANN → existing archite

NN → Archi → Dataset → train → ready to use

files down

image classification → ResNET

Text classifi → BERT

Image segment → UNet

image tran → Pix2Pix

Object dect → YOLO

speech gen → WaveNET

#5 → People / community

↳ Deep learning → 1960 → 2011 → PL →

Research

↳ DL engin

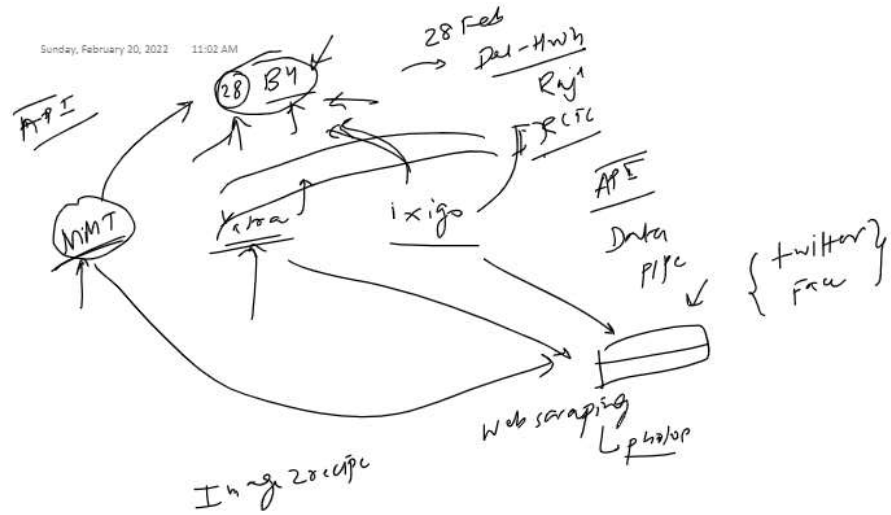
↳ Teacher → student → kaggle

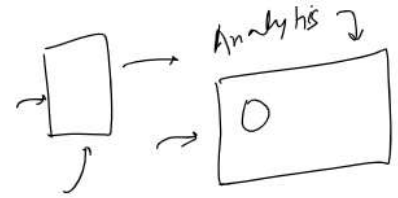
Factor #4 → Deep Learning Architectures

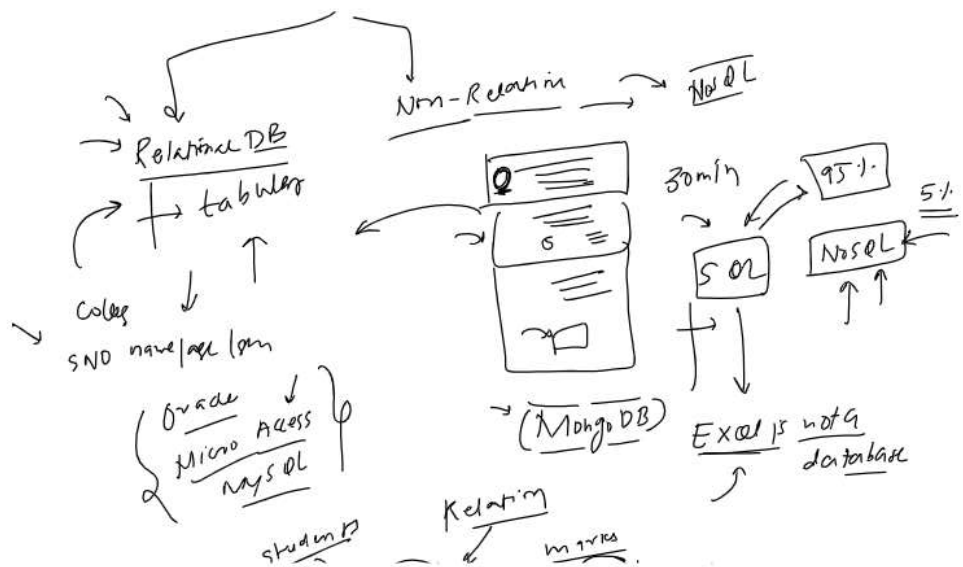
Code Example

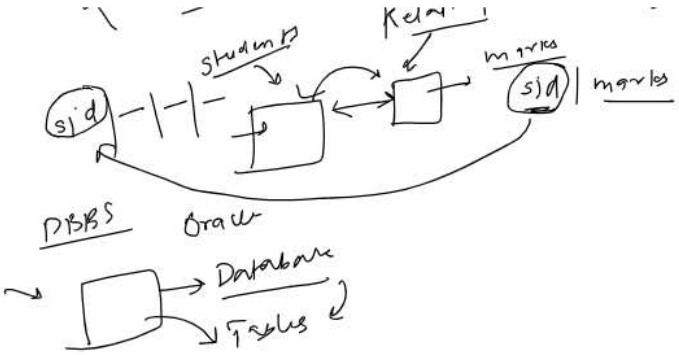
Saturday, February 19, 2022 7:10 AM

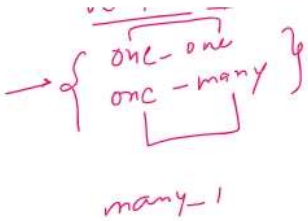
Sunday, February 20, 2022 11:02 AM

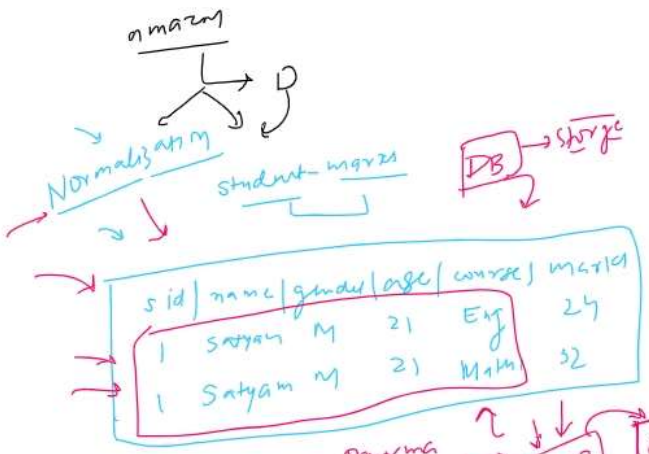


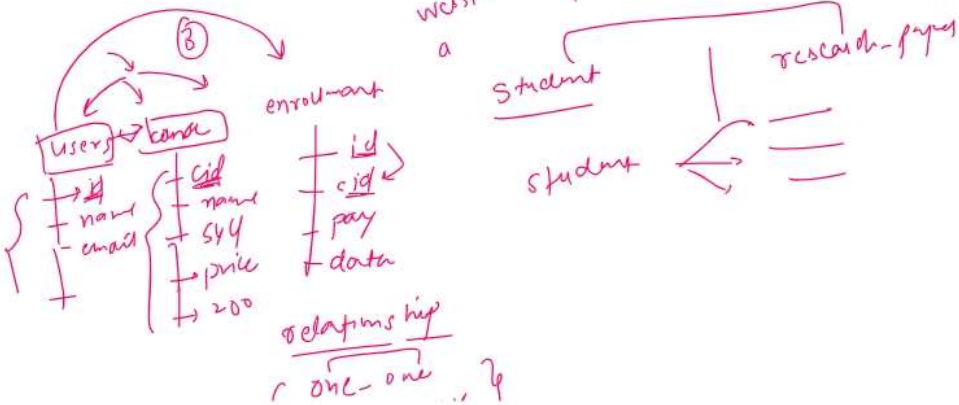





Relat
student
marks
sid
sid | marks
DBMS
Oracle
Database
Tables


one-one
one-many
many-1

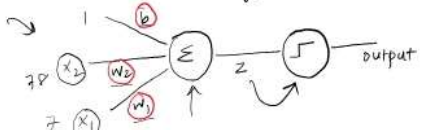

amazon
Normalization
student-marks
DB
storage
sid | name | gender | age | course | marks
1 satyam M 21 Eng 24
1 Satyam M 21 Maths 32
Denorma
OLTP
OLAP
website


8
users
courses
enrollment
id
name
email
cid
name
sly
price
200
id
cid
pay
date
relationship
one-one
student
research-papers
student

Recap

22 February 2022 14:12

$z = w_1x_1 + w_2x_2 + b$



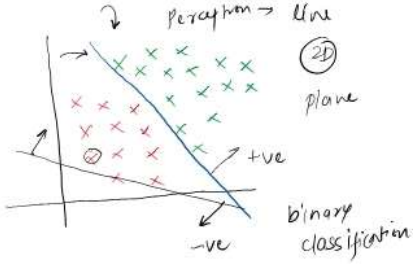
cgpa (x_1)	iq (x_2)	placed
7	78	1
6	61	0
9	92	1

1) mathematical model → neuron

$z \geq 0 \rightarrow 1$

$z < 0 \rightarrow 0$

1020 random



Prediction

→ Training

$\boxed{\text{Perceptron Rule}}$

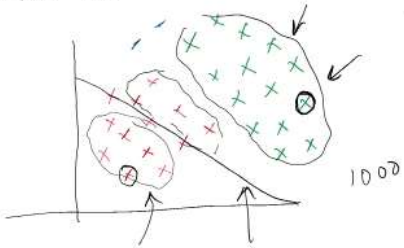
$Ax + By + c = 0$

$A \rightarrow w_1$

$B \rightarrow w_2$ $\quad C \rightarrow b$

Problem with Perceptron Trick

22 February 2022 16:10



1000

line → $\boxed{w_1\ w_2\ b}$

randomly per

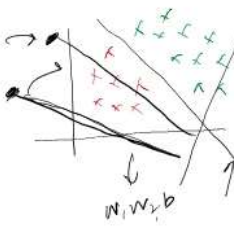
converge

quantify

$\boxed{w_1\ w_2\ b}$ → loss functions

Loss Function

22 February 2022 16:46



$w_1 w_2 b$ → loss → 3var → $\boxed{f(w_1 w_2 b)}$ → minimum → $w_1 w_2 b$ → final line

loss function

number

$w_1 w_2 b$

$w_1 w_2 b$

25 → error

23 → error

23 < 25

linear reg

↳ mse $(y_i - \hat{y}_i)^2$

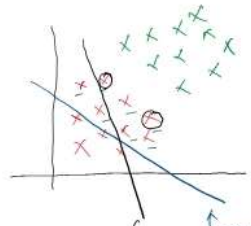
Logic → Logloss { SVM → hinge loss, 1-0 loss }

Perceptron Loss Function

22 February 2022 17:47

$f(w_1 w_2 b)$ → number → error

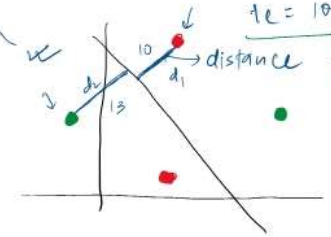
misclassified points



$w_1 w_2 b$ → 7

5

$|1+1+1+1+1| = 5$



w

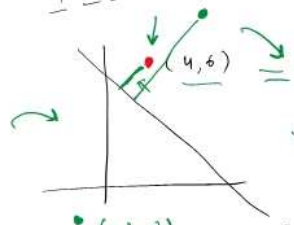
d_1, d_2, 10, 13

distance

$te = 10 + 13$

complex

↑ distance



$(4,6)$

$(-2,-2)$

$2x + 3y + 4 = 0$

$= 2(4) + 3(6) + 4 = 30$

↑ dot product

$\boxed{30+6}$

$2(-2) + 3(-2) + 4 = |-6| = 6$

Sklearn

$$L(w_1 w_2 b) = \boxed{\frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i))} + \boxed{\alpha R(w_1 w_2)}$$

Regularization

cgpalia placement

y_i

$$L(y_i, f(x_i)) = \boxed{\max(0, -y_i f(x_i))}$$

$$f(x_i) = \boxed{w_1 x_1 + w_2 x_2 + b}$$

$z =$

hinge

$$\boxed{L = \frac{1}{n}\sum_{i=1}^{n} \boxed{\max(0, -y_i f(x_i))}}$$

$L(w_1 w_2 b)$

n: # rows in data

$\boxed{w_1 w_2 b}$

$$L = \underset{w_1 w_2 b}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} \max(0, -y_i f(x_i))$$

minimum

gradient descent

$L = \underset{w_1, w_2, b}{\operatorname{argmin}} \; n \; i=1$

gradient descent

Explanation of Loss Function

23 February 2022 08:07

$$L = \boxed{\frac{1}{n} \sum_{i=1}^{n} \max(0, -y_i f(x_i))}$$

$w_1 w_2 b$

where $f(x_i) = w_1 x_{i1} + w_2 x_{i2} + b$

	x_1	x_2	y
1	x_{11}	x_{12}	y_1
2	x_{21}	x_{22}	y_2
i			
n			

n rows

x_{ij} row col

2 points

$\boxed{\max(0, -y_i f(x_i))} = \max(0, x)$ $= x$ $= 0$

$x \geq 0$

$x < 0$

$f(x_2) = w_1 x_{21} + w_2 x_{22} + b$

$\boxed{-y_i f(x_i)} \geq 0$

< 0 $= 0$

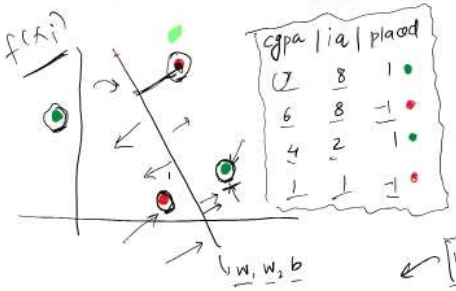
$y_i f(x_i)$

2 points n points

distance

$$L = \frac{1}{2}\left[\max(0, -y_1 f(x_1)) + \boxed{\max(0, -y_2 f(x_2))}\right] = 0$$

$\max(0, -y_i f(x_i))$



cgpa	ia	placed
7	8	1
6	8	-1
4	2	1
1	1	-1

y_i	$\hat{y}_i$
1	1
-1	-1
1	-1
-1	1

$\boxed{w_1 x_1 + w_2 x_2 + b} \geq 0$

$\max(0, -ve) = 0$

y_i $f(x_i)$ = +ve

-1 -ve -ve -ve

1 $f(x_i)$ = +ve

+ve +ve = -ve

$\max(0, -ve) = 0$

y_i $f(x_i)$

+ve -ve = -ve = +ve

+1

$\max(0, +ve) = +ve$

y_i $f(x_i)$

-ve +ve → -ve -1 = +ve

Gradient Descent

23 February 2022 08:50

$$L = \underset{w_1, w_2, b}{\operatorname{argmin}} \boxed{\frac{1}{n}\sum_{i=1}^{n} \max(0, -y_i f(x_i))}$$

$L(w_1, w_2, b)$

$L(w_1)$

where $f(x_i) = w_1 x_{i1} + w_2 x_{i2} + b$

$w_1, w_2, b = 1$

100/1000

$\boxed{\eta = 0.01}$

for i in epochs:

$$w_1 = w_1 + \eta \frac{\partial L}{\partial w_1}$$

$$w_2 = w_2 + \eta \frac{\partial L}{\partial w_2}$$

$$b = b + \eta \frac{\partial L}{\partial b}$$

$$\left[\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \frac{\partial L}{\partial b}\right]$$

Loss Function Differentiation

23 February 2022 12:57

$$L = \frac{1}{n}\sum_{i=1}^{n} \max(0, -y_i f(x_i))$$

where $f(x_i) = w_1 x_{i1} + w_2 x_{i2} + b$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial f(x_i)} \times \frac{\partial f(x_i)}{\partial w_1}$$

$$\frac{\partial L}{\partial f(x_i)} = \begin{cases} 0 & \text{if } \boxed{y_i f(x_i) \geq 0} \\ -y_i & \text{if } y_i f(x_i) < 0 \end{cases} \qquad \frac{\partial f(x_i)}{\partial w_1} = x_{i1}$$

cgpa i_2 | placed

$\rightarrow x_{i1}$

$$\boxed{\frac{\partial L}{\partial w_1} = \begin{cases} 0 & \text{if } y_i f(x_i) \geq 0 \\ -y_i x_{i1} & \text{if } y_i f(x_i) < 0 \end{cases}}$$

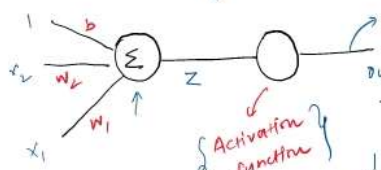
$$\boxed{\frac{\partial L}{\partial w_2} = \begin{cases} 0 & \text{if } y_i f(x_i) \geq 0 \\ -y_i x_{i2} & \text{if } y_i f(x_i) < 0 \end{cases}}$$

$\rightarrow$ code using gradient descent

$$\boxed{\frac{\partial L}{\partial b} = \begin{cases} 0 & \text{if } y_i f(x_i) \geq 0 \\ -y_i & \text{if } y_i f(x_i) < 0 \end{cases}}$$

More Loss Functions

23 February 2022 13:35



Activation function

Loss function

mathematical model

flexible by design

output $\hat{y}$

SGD

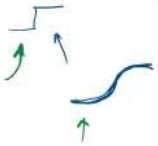
1, -1

prob

loss function → binary cross entropy

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

$z = w_1x_1 + w_2x_2 + b$



$$\boxed{\sigma(z) = \frac{1}{1+e^{-z}}}$$

SGD → output prob.

Perceptron == Logistic regression

Activation: sigmoid

Loss func

multiclass (3)

Activation → softmax

$$f = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

loss function → categorical cross entropy

$$L = \sum_{j=1}^{M} y_j \log(\hat{y}_j)$$

Softmax regression

$$f = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

$z \geq 0$ 1

$z < 0$ -1

(z) → (z)

Regression

Activation → linear

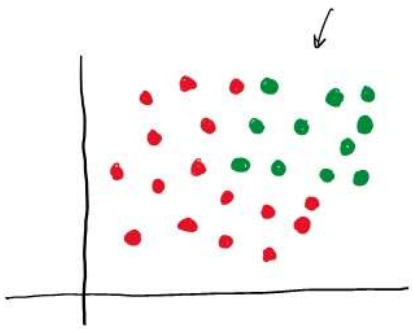
Loss function → linear regression

↓ mse $(y_i - \hat{y}_i)^2$

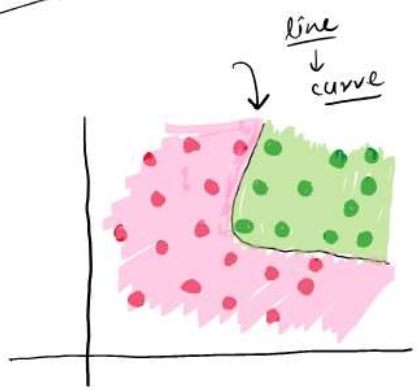
Loss function	Activation	output
Hinge loss	Step	Perceptron → binary classifi -1 1
log-loss	sigmoid	Logistic regression → binary
log-loss (binary cross entro	sigmoid	Logistic regression 0 - 1 → binary classifi
categorical cross entropy	softmax	softmax regression → multiclass classifi probabi
mse	linear	linear regression → number

The Problem

24 February 2022 13:41

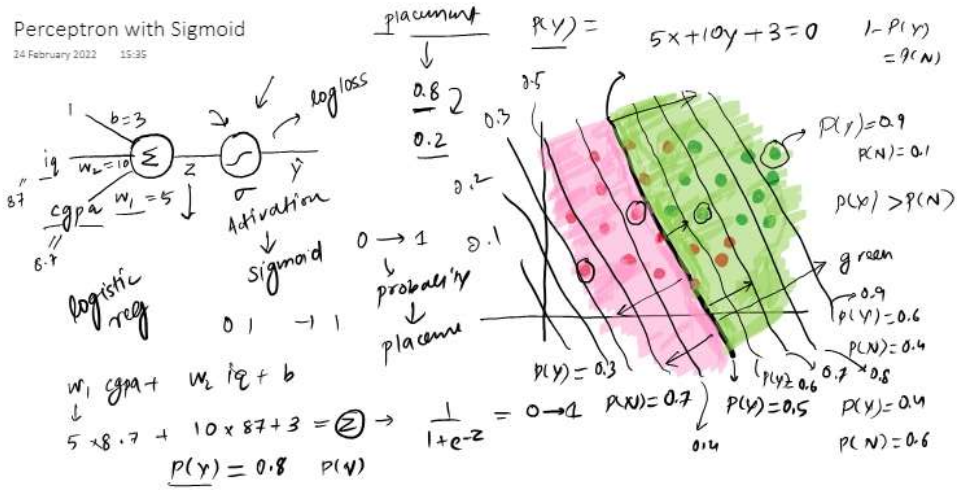


perceptron



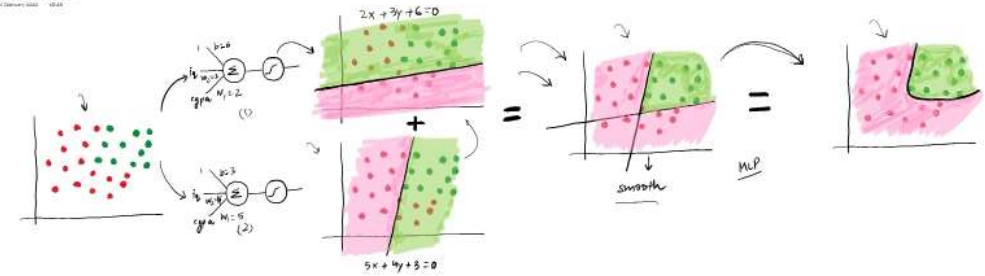
Perceptron with Sigmoid

24 February 2022 15:35



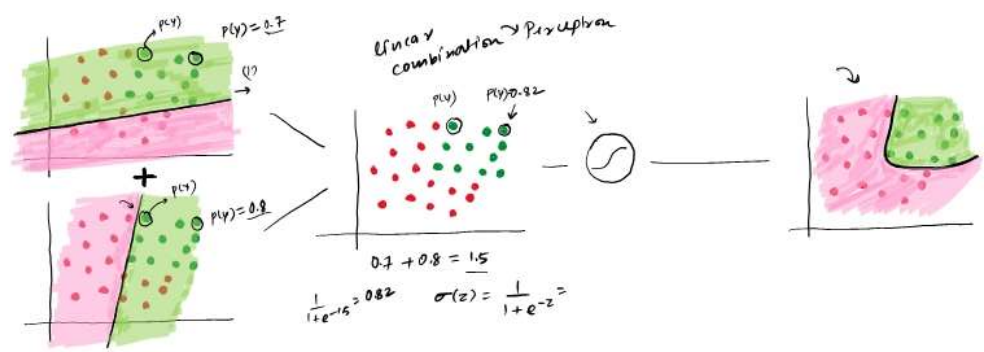
Abstract Solution

24 February 2022 18:49



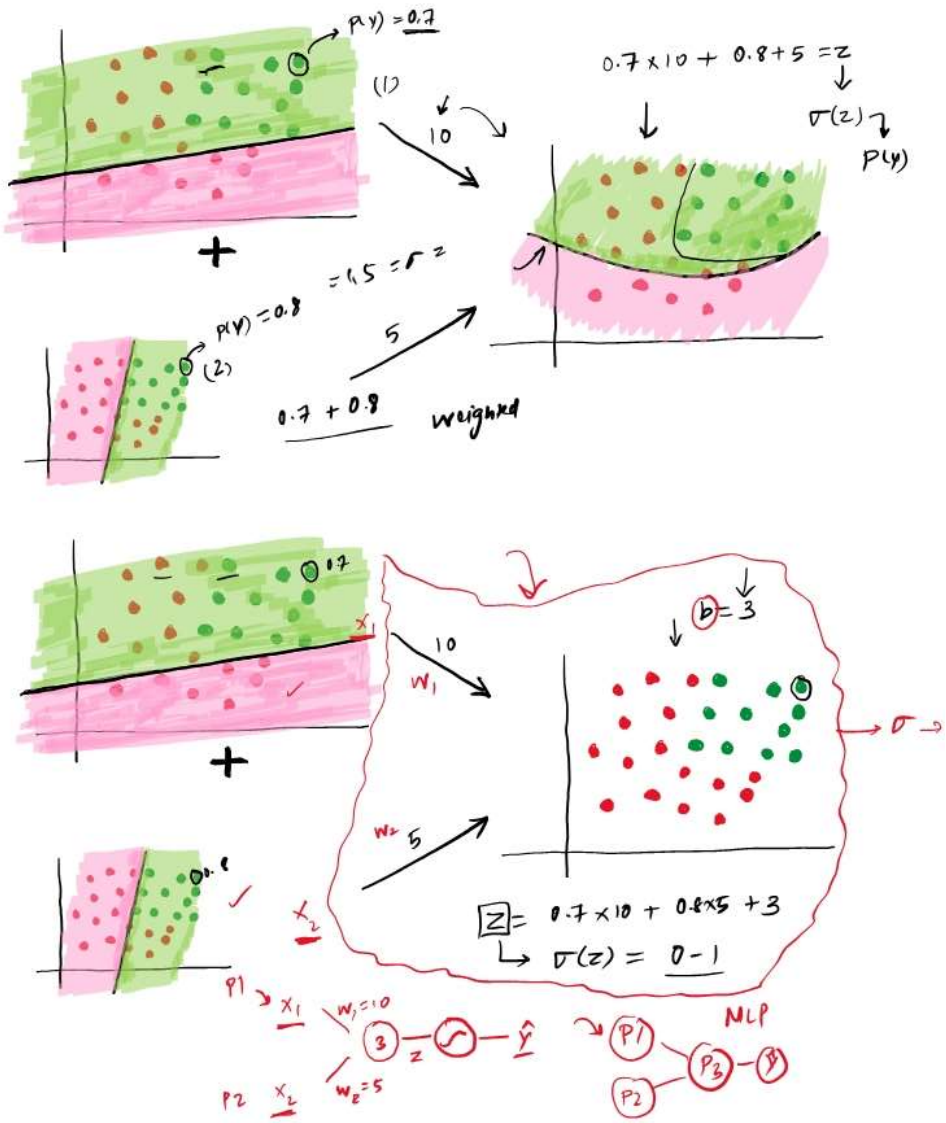
Mathematical Reasoning

24 February 2022 18:06



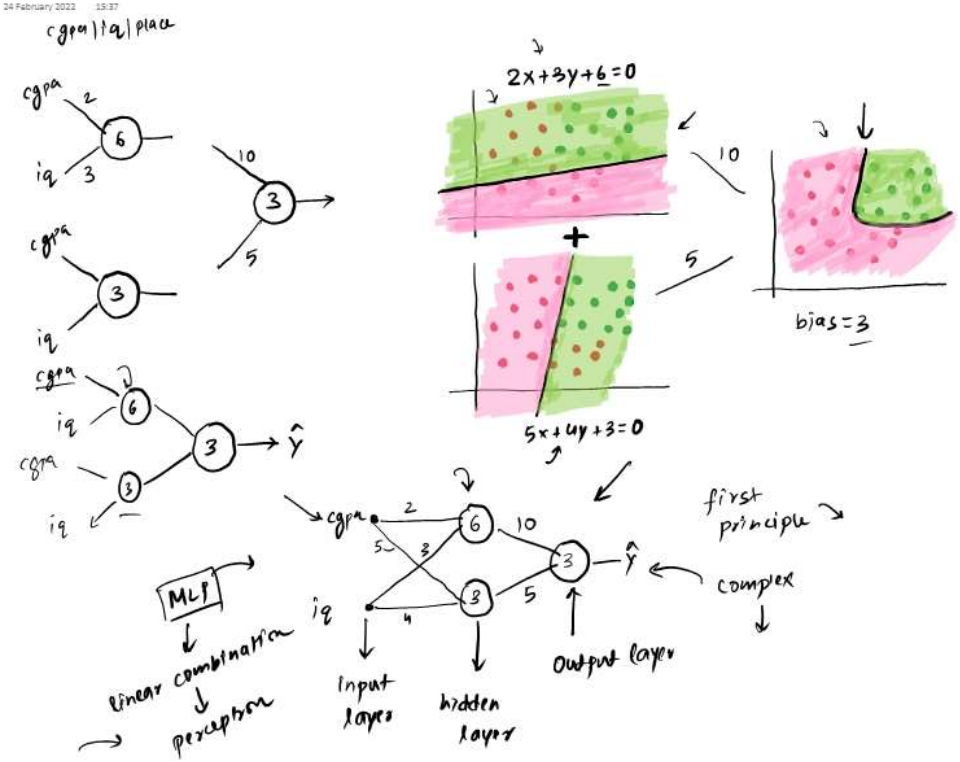
Adding Weights

24 February 2022 15:37



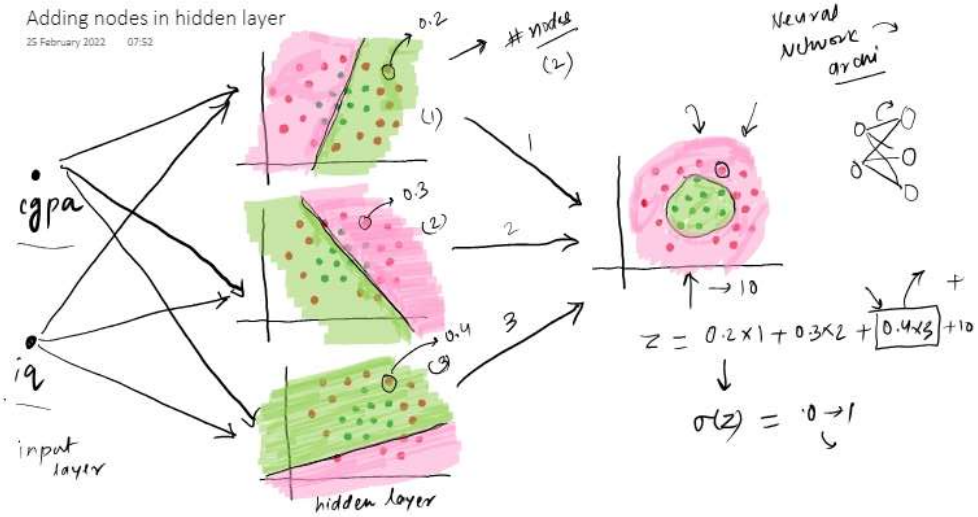
The Idea of MLP

24 February 2022 15:37



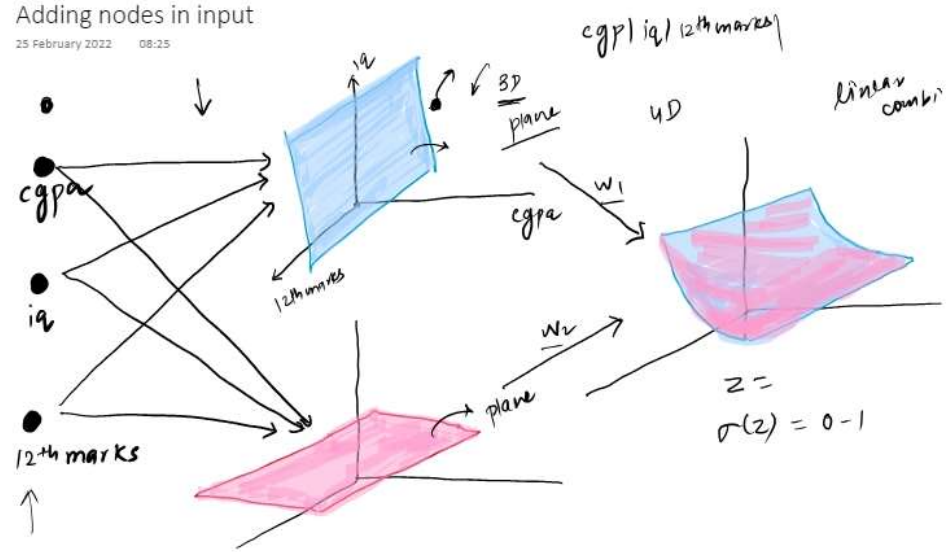
Adding nodes in hidden layer

25 February 2022 07:52



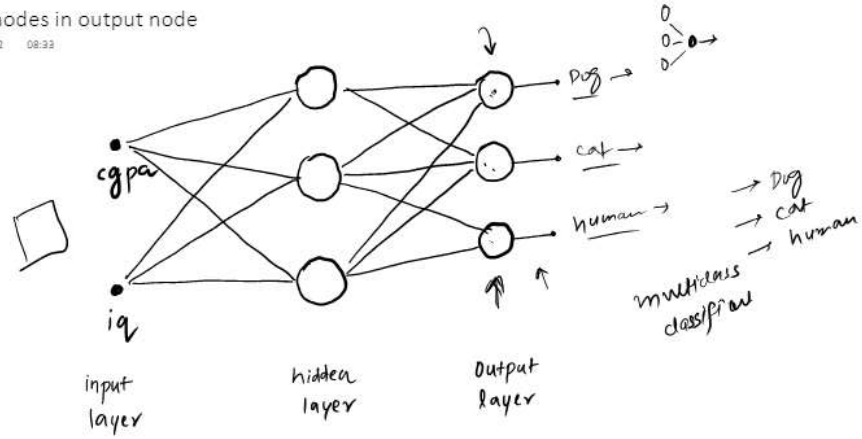
Adding nodes in input

25 February 2022 08:25



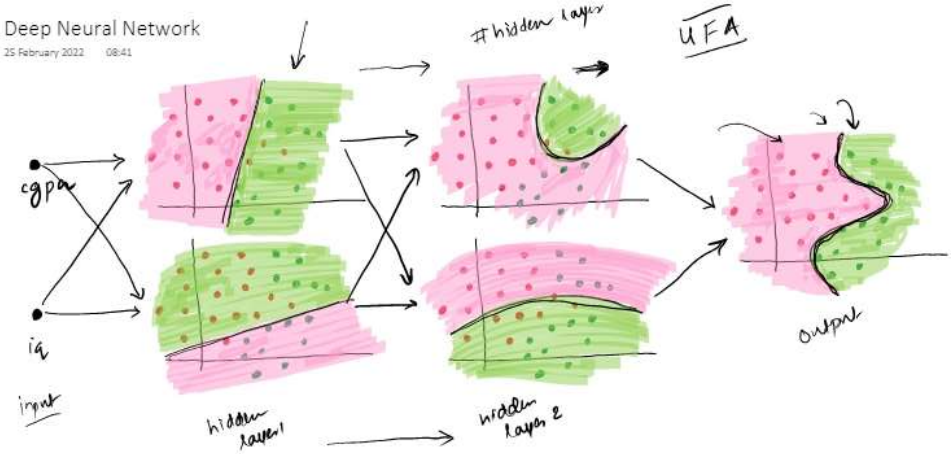
Adding nodes in output node

25 February 2022 08:33

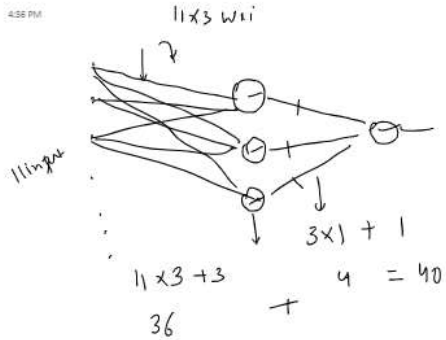


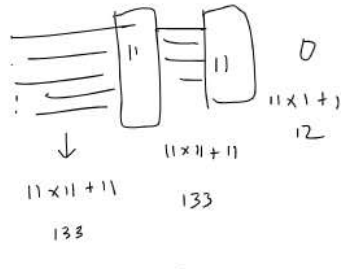
Deep Neural Network

25 February 2022 08:41



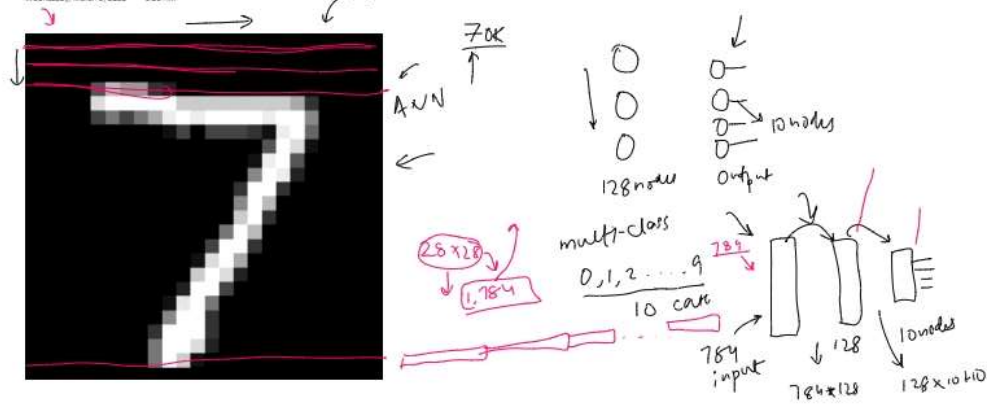
Sunday, March 6, 2022 4:58 PM

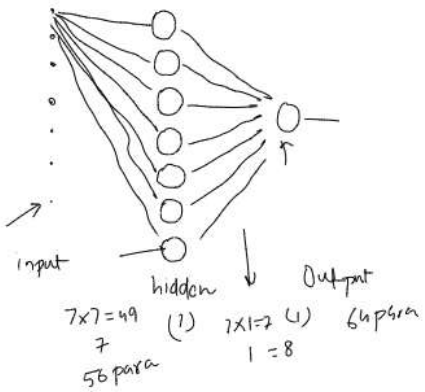




MNIST Dataset

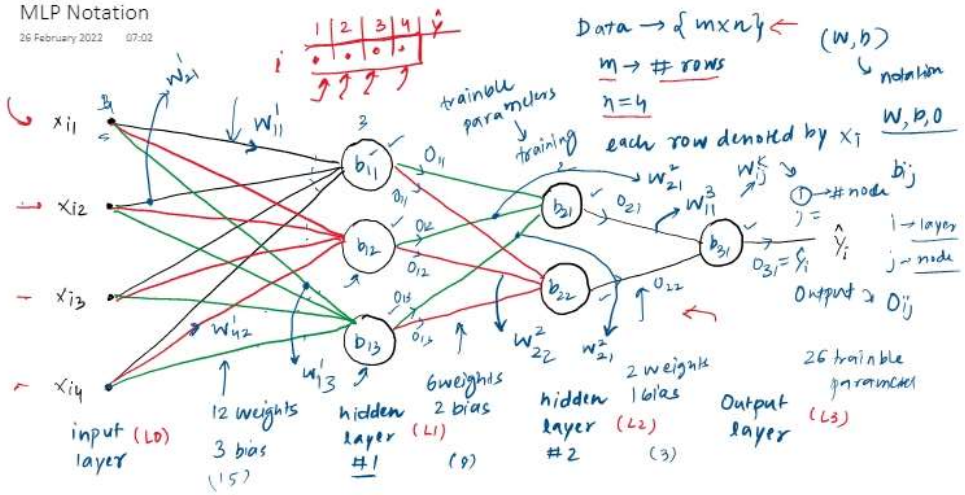
Wednesday, March 9, 2022 6:59 AM

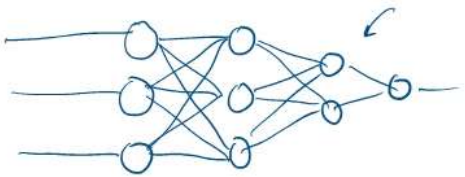




MLP Notation

26 February 2022 07:02





train

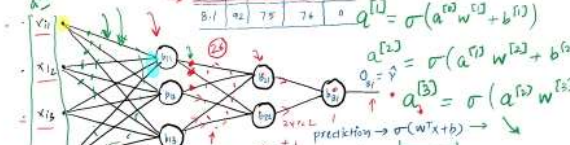
↓

backprop

Forward Propogation

Cg	Pa	Ig	Ch	m	Ch	m	placed
7.2	72	69		81			1
8.1	82	75		74			0

# of trainable parameters



$a^{[1]} = \sigma(a^{[0]} w^{[1]} + b^{[1]})$

$a^{[2]} = \sigma(a^{[1]} w^{[2]} + b^{[2]})$

$a^{[3]} = \sigma(a^{[2]} w^{[3]} + b^{[3]})$

prediction → $\sigma(w^T x + b)$ →

$$\sigma\Big(\sigma\big(\underbrace{(\sigma(a^{[0]} w^{[1]} + b^{[1]}))}_{a^{[1]}} w^{[2]} + b^{[2]}\big) w^{[3]} + b^{[3]}\Big)$$

$a^{[2]}$

$a^{[3]} = \hat{y}$

Layer #1

$$\begin{bmatrix} w^1_{11} & w^1_{12} & w^1_{13} \\ w^1_{21} & w^1_{22} & w^1_{23} \\ w^1_{31} & w^1_{32} & w^1_{33} \\ w^1_{41} & w^1_{42} & w^1_{43} \end{bmatrix}^T \begin{bmatrix} x_{i1} \\ x_{i2} \\ x_{i3} \\ x_{i4} \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \end{bmatrix}$$

4×3 → 3×4 3×1 4×1 3×1 → 3×1

$$= \begin{bmatrix} w^1_{11} x_{i1} + w^1_{21} x_{i2} + w^1_{31} x_{i3} + w^1_{41} x_{i4} \\ w^1_{12} x_{i1} + w^1_{22} x_{i2} + w^1_{32} x_{i3} + w^1_{42} x_{i4} \\ w^1_{13} x_{i1} + w^1_{23} x_{i2} + w^1_{33} x_{i3} + w^1_{43} x_{i4} \end{bmatrix} \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \end{bmatrix}$$

3×1

$$= \sigma\left(\begin{bmatrix} w^1_{11} x_{i1} + w^1_{21} x_{i2} + w^1_{31} x_{i3} + w^1_{41} x_{i4} + b_{11} \\ w^1_{12} x_{i1} + w^1_{22} x_{i2} + w^1_{32} x_{i3} + w^1_{42} x_{i4} + b_{12} \\ w^1_{13} x_{i1} + w^1_{23} x_{i2} + w^1_{33} x_{i3} + w^1_{43} x_{i4} + b_{13} \end{bmatrix}\right)$$

$$= \begin{bmatrix} o_{11} \\ o_{12} \\ o_{13} \end{bmatrix} \rightarrow a^{[1]}$$

Layer #2

$$\begin{bmatrix} w^2_{11} & w^2_{12} \\ w^2_{21} & w^2_{22} \\ w^2_{31} & w^2_{32} \end{bmatrix}^T \begin{bmatrix} o_{11} \\ o_{12} \\ o_{13} \end{bmatrix} + \begin{bmatrix} b_{21} \\ b_{22} \end{bmatrix} \rightarrow$$

2×1

3×2 → 2×3 3×1

$$= \sigma\left(\begin{bmatrix} w^2_{11} o_{11} + w^2_{21} o_{12} + w^2_{31} o_{13} + b_{21} \\ w^2_{12} o_{11} + w^2_{22} o_{12} + w^2_{32} o_{13} + b_{22} \end{bmatrix}\right) = \begin{bmatrix} o_{21} \\ o_{22} \end{bmatrix} \leftarrow a^{[2]}$$

Layer #3

$$\rightarrow \begin{bmatrix} w^3_{11} \\ w^3_{21} \end{bmatrix}^T \begin{bmatrix} o_{21} \\ o_{22} \end{bmatrix} + \begin{bmatrix} b_{31} \end{bmatrix}$$

1×1

2×1 → 1×2 2×1 1×1

$$= \sigma\left(\begin{bmatrix} w^3_{11} o_{21} + w^3_{21} o_{22} + b_{31} \end{bmatrix}\right) = \hat{y}_i = [a_{31}] \quad a^{[3]}$$

What is Loss Function?

23 March 2022 07:15

Loss function is a method of evaluating how well your algorithm is modelling your dataset.

high → poor

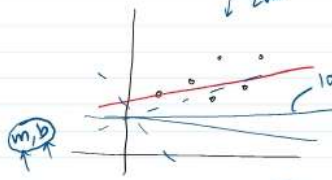
small → great

$f(x) = x^2 + 2$

↓ 2var

Loss | error minimise

L(parameters)



(m, b)

$L = (y_i - \hat{y}_i)^2$

$m x_i + b$

loss function eye of ML algorithm

GD

$L(m, b) = \min$

Why is Loss function important?

[You can't improve what you can't measure.]

Peter Drucker

Loss Function in Deep Learning?

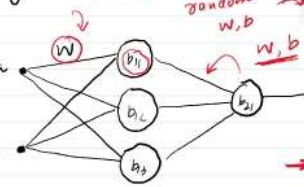
multiple epochs

Backprop

random w, b → gradient descent

w, b

cgpa	iq	package(lpa)
7.1	83	3.2
8.5	91	4.5
6.3	102	6.1
5.1	87	2.7
..	..	..
..	..	..
..	..	..



(3.7)

$L = (y_i - \hat{y}_i)^2$

$(3.2 - 3.7)^2$

4.3

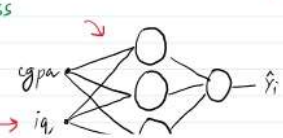
→ forward prop

Keras

Loss functions in DL

- Object detection: focal loss
- Regression
  - mse
  - mae
  - huber loss
- Classification
  - binary cross entropy
  - categorical cross entro
  - hinge loss
- Autoencoders
  - KL divergence
- GAN
  - discriminator loss
  - minimax gan loss
- Embedding: Triplet loss

Loss Function Vs Cost Function



Loss Function Vs Cost Function

Error function

Loss function → single training eg

$y_i = 6.3 \quad \hat{y}_i = 6.1$

$(y_i - \hat{y}_i)^2$

$(6.3 - 6.1)^2 =$

batch Cost function

$\frac{1}{4}\left[(6.1-6.3)^2 + (4.1-4)^2 + (3.5-3.7)^2 + (7.2-7)^2\right] = CF$

$\frac{1}{n}\sum(y_i - \hat{y}_i)^2$

(B) lpa $\sqrt{5}$ lpa $(lpa)^2$

1. Mean Squared Error (MSE)

Squared loss L2 Loss

$(y_i - \hat{y}_i)^2$

(true − predict)^2

$(6.3 - 6.1)^2 = -$

$\boxed{(y_i - \hat{y}_i)^2}$

Advan DISADV

mae

$\boxed{(y_i - \hat{y}_i)^2}$

quadratic punish

$\wedge^2$

cgpa	iq	y_i package	$\hat{y}_i$ Prediction	$(y_i - \hat{y}_i)^2$ error
6.3	100	6.3	6.1	0.2
7.1	91	4.1	4	0.1
8.5	83	3.5	3.7	−0.2
9.2	102	7.2	7	0.2
8.1	100	50	8	Overall error

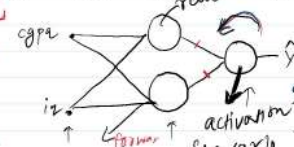
return

true − predic → 1unit → 1unit, 2unit → 4unit, 4 unit → 16 unit

magnify

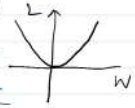
$(y_i - \hat{y}_i)^2$

$(w, b) \to L$ min



$(42)^2$

drastic big



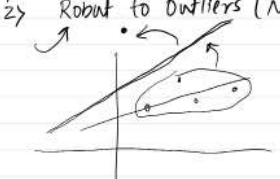
Advantages

1) Easy to intreprt
2) Diffrentiable (GD)
3) 1 local minima

Disadvantage

1) Error unit (squared) → diff
2) Robust to Outliers (Not)

$CF = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$



mse

2. Mean Absolute Error (MAE) → L1 loss

2 lpa

cgpa | iq | pack lpa $|(y_i - \hat{y}_i)|$ lpa

$L = |y_i - \hat{y}_i|$

2 | ~abs

$C = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|^2$

punish

|true − predict|

$$C = \frac{1}{n}\sum_{i=1} |y_i - y_i|$$ | true - predict |

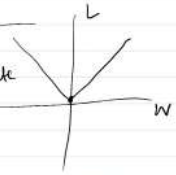
punish

Advantages

1) Intuitive and easy
2) Unit → same − y
3) Robust to outliers

Disadvange

1) Not differentiable
   → GD → differentiate
   → Subgradient

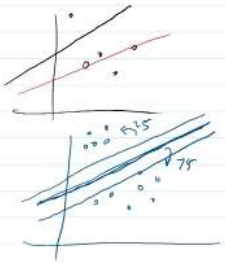


3. Huber Loss ✓ (mae, huber, mse)

$$L = \begin{cases} \frac{1}{2}(y-\hat{y})^2 & \text{for } |y-\hat{y}| \le \delta \\ \delta|y-\hat{y}| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases}$$

mse, mae, hyper

mse — outliers ✓
mae — normal points ✓

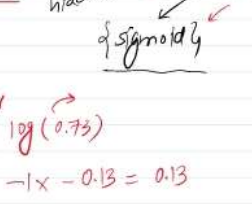


4. Binary Cross Entropy (→ Logloss)
→ classification ✓
→ Two classes ✓

100 days

cgpa	iq	placement	
8	80	1	0.73
7	70	0	0.25
6	60	0	0.13

→ 1 0 GD FP



0.12

Loss function $= \boxed{-y\log(\hat{y}) - (1-y)\log(1-\hat{y})}$

Keras

y → actual value / target
$\hat{y}$ − NN prediction

$-(1-0)\log(1-0.25)$ 0.12
$-1\log 0.75$ −0.12

$-1\log(0.73)$
$-1 \times -0.13 = 0.13$

cost function $= -\frac{1}{n}\left[\sum_{i=1}^{n} y_i \log\hat{y}_i + (1-y_i)\log(1-\hat{y}_i)\right]$

maximum likeli

Logistic Reg

Advantage
* Differentiable

Disadvan
→ multi local minima
→ intuitive

5. Categorical Cross Entropy [used in Softmax Regression] ↓↙ OHE
→ [Multi-class] {classification} ✓

cgpa	iq	placed?	Yes	No	Maybe
8	80	Yes 1	1	0	0
6	60	No 2	0	1	0
7	70	Maybe 3	0	0	1

1 point

$$L = -\sum_{j=1}^{K} y_j \log(\hat{y}_j)$$

where K is # classes in the data

1 point

$$L = -y_1\log(\hat{y}_1) - y_2\log(\hat{y}_2) - y_3\log(\hat{y}_3)$$

[0.2 0.3 0.5]
[1 0 0]

$\frac{e^{z_3}}{e^{z_1}+e^{z_2}+e^{z_3}}$ $\frac{e^{z_2}}{e^{z_1}+e^{z_2}+e^{z_3}}$

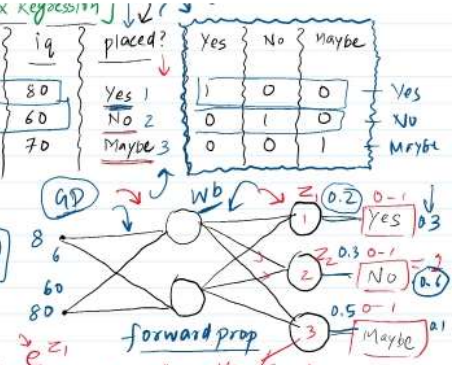
$f(z) = \frac{e^{z_1}}{e^{z_1}+e^{z_2}+e^{z_3}}$ {softmax}

$L = -1 \times \log(0.2)$

[0.3 0.6 0.1]
[0 1 0]

$-1 \times \log(0.6) = -$

GD, wb



forward prop
Activation
neurons
class ← categories

Sparse Categorical Cross Entropy

cgpa	iq	placed	OHE
7	70	Yes	1
8	80	No	2
6	60	Maybe	3

1/2/3

[0.1 0.4 0.5]

L

$-1 \times \log(0.1)$

[0.1 0.4 0.6]

{SCE, CCE} Loss function

$-1 \times \log(0.4)$

[0.6 0.2 0.2]

SCE → fast
fast

$-1 \times \log(0.2)$

OHE — fast
SCE

$$L = -\sum_{j=1}^{K} y_j \log(\hat{y}_j)$$ → Cost function

$$C = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{K} y_{ij}\log(\hat{y}_{ij})$$

23 March 2022 07:31

- Reg → mse
  - → outliers — mae → huber loss
- classification → binary → bce
  - → multi → CCE
    - → SCE

Backpropogation

Pre requisite

→ Gradient Descent (✓) →
→ Forward Propogation (✓) →

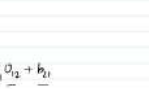
Backprop → Algo → train nn (How)

training

cgpa	iq	lpa
8	80	8
7	70	7

lpa

predict



$\sigma(z)$

$O_{21} = w^2_{11} O_{11} + w^2_{21} O_{12} + b_{21}$

6 + 3 → 9 trainable

L min → $\hat{y} \to O_{21}$

Backprop

4 students

iq	cgpa	lpa
80	8	8
60	9	5
70	5	8
120	7	11

crossp 225

$W \to 1, b \to 0$

→ 18 lpa

Steps →
0) Init w, b
1) You select a point (row) ↳ student
2) Predict (lpa) → forward prop [Dot product]
3) Choose a loss function (mse)
4) Weights and bias update ↳ Gradient Descent

$L = (y - \hat{y})^2$

$(3-18)^2 = 225$ error

Hierarchy Backprop

$$W_{new} = W_{old} - \eta \frac{\partial L}{\partial W_{old}}$$

$\eta = 0.1$

$$w^2_{11\,new} = w^2_{11\,old} - \eta \frac{\partial L}{\partial w^2_{11}}$$

derivative

$$b_{new} = b_{old} - \eta \frac{\partial L}{\partial b_{old}}$$

$$w^2_{21\,new} = w^2_{21\,old} - \eta \frac{\partial L}{\partial w^2_{21}}$$

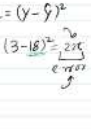
$$b_{21\,new} = b_{21\,old} - \eta \frac{\partial L}{\partial b_{21}}$$

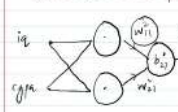
derivative of loss wrt weight

$\frac{\partial L}{\partial w^2_{11}}$ ← partial derivative

derivative $\frac{\partial L}{\partial w^2_{11}}$

$\frac{\partial L}{\partial w^2_{11}}, \frac{\partial L}{\partial w^2_{21}}, \frac{\partial L}{\partial b_{21}}$ | $\frac{\partial L}{\partial w_{11}}, \frac{\partial L}{\partial w_{21}}, \frac{\partial L}{\partial b_{11}}$ | $\frac{\partial L}{\partial w_{12}}, \frac{\partial L}{\partial w_{22}}, \frac{\partial L}{\partial b_{12}}$





$$\frac{\partial L}{\partial w^2_{11}} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial w^2_{11}}$$ → chain rule of differ

$\left\{\frac{dy}{dx}\right\}$

$$\frac{\partial L}{\partial \hat{y}} = \frac{\partial}{\partial \hat{y}} (y - \hat{y})^2 = \boxed{-2(y - \hat{y})}$$

$$\frac{\partial \hat{y}}{\partial w^2_{11}} = \frac{\partial}{\partial w^2_{11}} [O_{11} w^2_{11} + O_{12} w^2_{21} + b_{21}]$$

$= O_{11}$

$\hat{y} \to L$, $\frac{\partial L}{\partial w^2_{21}}$

$$\boxed{\frac{\partial L}{\partial w^2_{11}} = -2(y - \hat{y}) O_{11}}$$ 1

$$\frac{\partial L}{\partial w^2_{21}} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial w^2_{21}}$$

$$\frac{\partial \hat{y}}{\partial w^2_{21}} = \frac{\partial}{\partial w^2_{21}} [O_{11} w^2_{11} + O_{12} w^2_{21} + b_{21}]$$

$= O_{12}$

$$\boxed{\frac{\partial L}{\partial w^2_{21}} = -2(y - \hat{y}) O_{12}}$$ 2

$\hat{y} \to b$, L

$$\frac{\partial \hat{y}}{\partial b_{21}} = 1$$

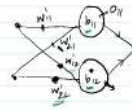
$$\boxed{\frac{\partial L}{\partial w^2_{21}} = -2(y - \hat{y}) O_{12}}$$ 2

$\hat{y} \to b$, L

$$\frac{\partial \hat{y}}{\partial b_{21}} = 1$$

$$\boxed{\frac{\partial L}{\partial b_{21}} = -2(y - \hat{y})}$$ 3

$$\frac{\partial L}{\partial b_{21}} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial b_{21}}$$



$$\frac{\partial L}{\partial w^1_{11}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial O_{11}} \frac{\partial O_{11}}{\partial w_{11}} \to \boxed{-2(y - \hat{y}) w^2_{11} x_{i1}}$$ 4

$$\frac{\partial L}{\partial w^1_{21}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial O_{11}} \frac{\partial O_{11}}{\partial w^1_{21}} \to \boxed{-2(y - \hat{y}) w^2_{11} x_{i2}}$$ 5

$$\frac{\partial L}{\partial b_{11}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial O_{11}} \frac{\partial O_{11}}{\partial b_{11}} \to \boxed{-2(y - \hat{y}) w^2_{11}}$$ 6

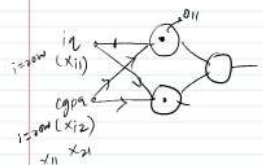
$$\frac{\partial L}{\partial w^1_{12}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial O_{12}} \frac{\partial O_{12}}{\partial w^1_{12}} \to \boxed{-2(y - \hat{y}) w^2_{21} x_{i1}}$$ 7

w^2_{11}

$$\frac{\partial L}{\partial w^1_{22}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial O_{12}} \frac{\partial O_{12}}{\partial w^1_{22}} \to \boxed{-2(y - \hat{y}) w^2_{21} x_{i2}}$$ 8

$\frac{\partial \hat{y}}{\partial O_{12}} = w^2_{21}$

$$\frac{\partial L}{\partial b_{12}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial O_{12}} \frac{\partial O_{12}}{\partial b_{12}} \to \boxed{-2(y - \hat{y}) w^2_{21}}$$ 9



$$\frac{\partial O_{11}}{\partial w^1_{11}} = \frac{\partial}{\partial w^1_{11}} [iq\, w^1_{11} + cgpa\, w^1_{21} + b_{11}] = (iq)$$

x_{i1}

$$\frac{\partial O_{11}}{\partial w^1_{21}} = x_{i2}$$

$$\frac{\partial O_{11}}{\partial b_{11}} = 1$$

$$\frac{\partial O_{12}}{\partial w^1_{12}} = \frac{\partial}{\partial w^1_{12}} [iq\, w^1_{12} + cgpa\, w^1_{22} + b_{12}] = iq\,(x_{i1})$$

$$\frac{\partial O_{12}}{\partial w^1_{22}} = x_{i2}$$

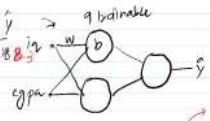
$$\frac{\partial O_{12}}{\partial b_{12}} = 1$$

epochs

multiple chain

	cgpa	iq	lpa	$\hat{y}$
w/w	8	80	8	
	6	60	6	
	7	70	7	
	9	90		

9 trainable



L ↓

Steps (once again)

0) weights/bias → init → random, w=1, b=0

Loop - 100/120 times

1) for i in range(4):
   1a) 1 student → forward prop → lpa) → 12 min
   1b) Loss calculate (mse) →
   1c) Adjust all weights and bias

$(y - \hat{y})^2$ → Backprop

$$W_{new} = W_{old} - \eta \frac{\partial L}{\partial W_{old}}$$ → 9 times

The How

Backpropagation Algorithm

epochs = 5
for i in range(epochs):
    for j in range(x.shape[0]):

$$\frac{\partial L}{\partial w^1_{11}} = -2(y - \hat{y}) O_{11}$$

$$\frac{\partial L}{\partial w^2_{21}} = -2(y - \hat{y}) O_{12}$$

Ready

for i in range(epochs):

for j in range(X.shape[0]):

→ Select 1 row (random)

→ Predict (using Forward prop)

→ Calculate loss (using Loss function → mse)

→ Update weights and bias using GD

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

→ Calculate avg loss for the epoch

Loss

$$\frac{\partial L}{\partial w^2_{21}} = -2(y-\hat{y})o_{12}$$

$$\frac{\partial L}{\partial b_{21}} = -2(y-\hat{y})$$

$$\frac{\partial L}{\partial w^1_{11}} = -2(y-\hat{y})w^2_{11}x_{11}$$

$$\frac{\partial L}{\partial w^1_{21}} = -2(y-\hat{y})w^2_{11}x_{12}$$

$$\frac{\partial L}{\partial b_{11}} = -2(y-\hat{y})w^2_{11}$$

$$\frac{\partial L}{\partial w^1_{12}} = -2(y-\hat{y})w^2_{21}x_{11}$$

$$\frac{\partial L}{\partial w^1_{22}} = -2(y-\hat{y})w^2_{21}x_{12}$$

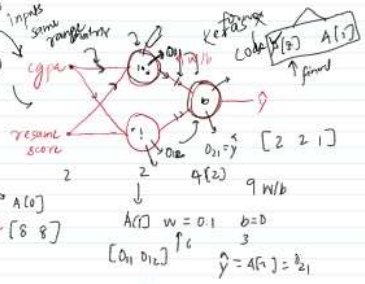
$$\frac{\partial L}{\partial b_{12}} = -2(y-\hat{y})w^2_{21}$$

Regression Problem

cgpa (x_{11})	resume score (x_{12})	package (lpa)
8	8	4
7	9	5
6	10	6
5	12	7

4 students

NN inputs same range/matrix



Keras

code: X[0] A[1]

↑ final

→ A[0]

code [8 8]

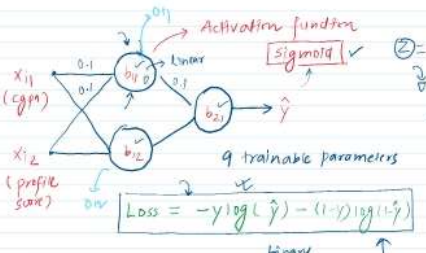
A[1] w = 0.1 b = 0

[o_{11} o_{12}] ↑6 3

$\hat{y} = A[1] = o_{21}$

Classification Example

cgpa	profile score	placement
8	8	1
7	9	1
6	10	0
5	5	0



Activation function

sigmoid

$z = 0.1 \times 8 + 0.1 \times 8 + 0$

$\sigma(z) = o_{11}$

9 trainable parameters

$$\boxed{Loss = -y\log(\hat{y}) - (1-y)\log(1-\hat{y})} \quad (y-\hat{y})^2$$

binary cross entropy

Back propagation Algorithm

epoch = 5

for i in range(epochs): (rows)

for j in range(X.shape[0]):

→ Select 1 row (random) → 1by1

→ Predict (using Forward prop) → predict

→ Calculate loss (using Loss function → bce) → {code convert}

→ Update weights and bias using GD

Back propagation Algorithm

epoch = 5

for i in range(epochs): (rows)

for j in range(X.shape[0]):

→ Select 1 row (random) → 1by1

→ Predict (using Forward prop) → predict

→ Calculate loss (using Loss function → bce) → {code convert}

→ Update weights and bias using GD

cross entropy

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

9 w, b update

→ Calculate avg loss for the epoch

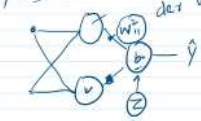
Loss

$z = w^2_{11}o_{11} + w^2_{21}o_{12} + b_{21}$

$\hat{y} = \sigma(z)$

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

der wrt L



$$\frac{\partial L}{\partial w^2_{11}} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z} \times \frac{\partial z}{\partial w^2_{11}} = -(y-\hat{y})o_{11} \quad (3)$$

$$\frac{\partial L}{\partial w^2_{21}} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z} \times \frac{\partial z}{\partial w^2_{21}} = -(y-\hat{y})o_{12}$$

$z_{final} = w^2_{11}o_{11} + w^2_{21}o_{12} + b_{21}$

$$\frac{\partial L}{\partial b_{21}} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z} \times \frac{\partial z}{\partial b_{21}} = -(y-\hat{y})$$

z_{final}

$\hat{y} \to w^2_{11} \times w^2_{21}$

L

$$\frac{\partial L}{\partial \hat{y}} = \frac{\partial}{\partial \hat{y}}\left[-y\log(\hat{y}) - (1-y)\log(1-\hat{y})\right]$$

$$= -\frac{y}{\hat{y}} + \frac{(1-y)}{(1-\hat{y})} = \frac{-y(1-\hat{y}) + \hat{y}(1-y)}{\hat{y}(1-\hat{y})} = \frac{-y + \cancel{y\hat{y}} + \hat{y} - \cancel{y\hat{y}}}{\hat{y}(1-\hat{y})}$$

$$= \boxed{\frac{-(y-\hat{y})}{\hat{y}(1-\hat{y})}}$$

$\hat{y} = \sigma(z)$

$$\frac{\partial}{\partial z}\sigma(z) = \sigma(z)[1-\sigma(z)]$$

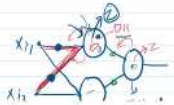
$\sigma(z) = \hat{y}$

$$\boxed{\frac{\partial \hat{y}}{\partial z}} = \frac{\partial}{\partial z}(\sigma(z)) = \sigma(z)[1-\sigma(z)] = \hat{y}(1-\hat{y})$$

$$\frac{\partial L}{\partial \hat{y}} = \frac{-(y-\hat{y})}{\hat{y}(1-\hat{y})} \qquad \frac{\partial \hat{y}}{\partial z} = \hat{y}(1-\hat{y}) \Rightarrow \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z} = \frac{-(y-\hat{y})}{\cancel{\hat{y}(1-\hat{y})}} \times \cancel{\hat{y}(1-\hat{y})}$$

$$-(y-\hat{y})$$

$$\frac{\partial}{\partial w^1_{11}} z_p = w^1_{11}x_{11} + w^1_{21}x_{12} + b_{11}$$

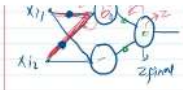


$$\frac{\partial L}{\partial w^1_{11}}, \frac{\partial L}{\partial w^1_{21}}, \frac{\partial L}{\partial b_{11}}$$

$-(y-\hat{y})$

$z_f = w^2_{11}o_{11} + w^2_{21}o_{12} + b_{21}$ $o_{11}(1-o_{11})$

$o_{11} = \sigma(z_{prev})$



$\frac{\partial L}{\partial w^1_{11}}, \frac{\partial L}{\partial w^1_{21}}, \frac{\partial L}{\partial b_{11}}$

$z_f = w^2_{11} o_{11} + w^2_{21} o_{12} + b_{21}$

$o_{11} = \sigma(z_{prev}) = \sigma(z_{prev})[1-\sigma(z_{prev})]$

$$\frac{\partial L}{\partial w^1_{11}} = \boxed{\frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z_f}} \times \frac{\partial z_f}{\partial o_{11}} \times \frac{\partial o_{11}}{\partial z_{prev}} \times \frac{\partial z_{prev}}{\partial w^1_{11}}$$

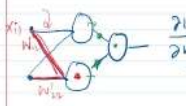
$-(y-\hat{y})$; w^2_{11} ; $o_{11}(1-o_{11})$; x_{i1}

$L \to \hat{y} \to z_f \to o_{11} \to z_{prev} \to w^1_{11}$

$$\boxed{\frac{\partial L}{\partial w^1_{11}} = -(y-\hat{y}) w^2_{11} o_{11}(1-o_{11}) x_{i1}}$$ 3

$$\boxed{\frac{\partial L}{\partial w^1_{21}} = -(y-\hat{y}) w^2_{11} o_{11}(1-o_{11}) x_{i2}}$$

$$\boxed{\frac{\partial L}{\partial b_{11}} = -(y-\hat{y}) w^2_{11} o_{11}(1-o_{11})}$$



$\frac{\partial L}{\partial w^1_{12}}, \frac{\partial L}{\partial w^1_{22}}, \frac{\partial L}{\partial b_{12}}$

$\frac{\partial z_f}{\partial o_{12}} = w^2_{11} o_{11} + w^2_{21} o_{12} + b_{21}$

$$\frac{\partial L}{\partial w^1_{12}} = \boxed{\frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z_f}} \times \frac{\partial z_f}{\partial o_{12}} \times \frac{\partial o_{12}}{\partial z_p} \times \frac{\partial z_p}{\partial w^1_{12}}$$

w^2_{21} ; $o_{12}(1-o_{12})$; x_{i1}

$L \to \hat{y} \to z_f \to o_{12} \to z_p \to w^1_{12}$

$$\boxed{\frac{\partial L}{\partial w^1_{12}} = -(y-\hat{y}) w^2_{21} o_{12}(1-o_{12}) x_{i1}}$$ 3

$$\boxed{\frac{\partial L}{\partial w^1_{22}} = -(y-\hat{y}) w^2_{21} o_{12}(1-o_{12}) x_{i2}}$$

$$\boxed{\frac{\partial L}{\partial b_{12}} = -(y-\hat{y}) w^2_{21} o_{12}(1-o_2)}$$

Backpropagation → The Why?

The Intuition behind the algorithm ✓

Backpropagation Algorithm

epoch = 5 ✓ # epochs × rows

```
for i in range(epochs):
    for j in range(X.shape[0]):
        → Select 1 row (random)
        → Predict (using Forward prop)
        → Calculate loss (using loss function → mse)
        → Update weights and bias using GD
            → W_n = W_0 - η ∂L/∂W  rule
    → Calculate avg loss for the epoch
```

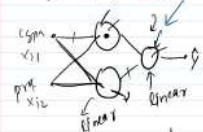
$\hat{y}$ y ; bce ; magic

→ Loss function is a function of all trainable parameters

data → target

$L = (y - \hat{y})^2$

cgpa | profile | lpa



$mse = (y-\hat{y})^2$; variable

$L = (y-\hat{y})^2$; constant → mathematical function

$L(?)$; $L(\theta) =$; $y = f(x)$; thing

$L(w, b)$ ↓ ; 9 things ; w, b

$\hat{y} = w^2_{11} o_{11} + w^2_{21} o_{12} + b_{21}$

$\hat{y} = w^2_{11}[w^1_{11} x_{i1} + w^1_{21} x_{i2} + b_{11}] + w^2_{21}[w^1_{12} x_{i1} + w^1_{22} x_{i2} + b_{12}] + b_{12}$

→ Concept of Gradient → Gradient Descent

↳ fancy word for derivative

$y = f(x) = x^2 + x$

$\frac{dy}{dx} = \frac{d}{dx}(fx) = \frac{d}{dx}(x^2+x) = 2x+1$

derivative

$y \to x \to$ derivative $\frac{d}{dx}$

$z = f(x,y) = \boxed{x^2+y^2}$

$\frac{\partial z}{\partial x} = 2x$; $\frac{\partial z}{\partial y} = 2y$

gradient $\boxed{\frac{\partial}{\partial x}}$

gradient complex $L(w^1_{11}, w^1_{12} \ldots w^2_{21}, b_{21}, b_{11}, b_{12})$

↳ 9D function → 9 diff slopes wrt each dim

$\frac{\partial L}{\partial w}, \frac{\partial L}{\partial b}$

9 parameters

3D → $z = x^2 + y^2$ $z = f(x,y)$

$y = f(x) = x^2$

→ Concept of Derivative → Derivative at a point

intuition

$\frac{dy}{dx}$ → rate of change

(y) func x → (x)

$\frac{dy}{dx} = 2$ (+ve) → mag, sign

$\frac{dy}{dx} = -2$ (-ve)

$x \to$

$y \to$

$\boxed{\frac{\partial L}{\partial w_{11}^1}}$

w_{11}^1 — 1 unit

$L \to ?$ (mag) (sign)

derivative

$y = x^2 + 2x$

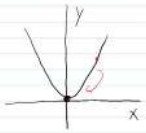
$\boxed{x=5}$ → deriv

$\frac{dy}{dx} = (2x+1)_{x=5}$

$\left(\frac{dy}{dx}\right)_{x=5} = 11$ → slope

$W_{new} = W_{old} - \eta \boxed{\frac{\partial L}{\partial w}}$

→ the concept of minima



$\boxed{y = x^2}$

$\frac{dy}{dx} = 2x = 0$

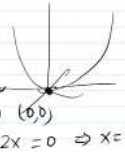
$\boxed{x=0}$

$z = x^2 + y^2$

$(x,y) \to z$ min

$\frac{\partial z}{\partial x} = 0$ $x \to$

$\frac{\partial z}{\partial y} = 0$ $y \to$



z min

(x, y) (0,0)

$2x = 0 \Rightarrow x = 0$

$2y = 0 \Rightarrow y = 0$

L (9 param)

(w, b) min

$L \downarrow$

$\frac{\partial L}{\partial w_{11}^1} \cdots \frac{\partial L}{\partial b_{12}} = 0$

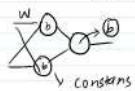
9 dim → minima

→ Backprop Intuition

$\boxed{W_{new} = W_{old} - \eta \frac{\partial L}{\partial w}}$ ← 9 step

$\eta = 1$

$\boxed{W_{new} = W_{old} - \frac{\partial L}{\partial w}}$



(each parameters)

→ constants

$L = L(w, b)$ → 9, 8 → constant

$L(b_{21})$

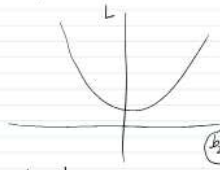
$L = b_{21}^2$

$L = (y - \hat{y})^2$

$L(b_{21})$ $\hat{y} = (b_{21})$

$b_{21} = \boxed{b_2} + \boxed{\frac{\partial L}{\partial b_{21}}}$ → +ve, -ve

$b_{21} = 5$



(b_{21}) L min

$\boxed{\frac{\partial L}{\partial b_{21}}}$

smart

L derivative wrt b_{21}

? 1 unit → $L \downarrow$

$b_{21} \downarrow$

$b_{21} \downarrow$

$L \downarrow$

$\frac{\partial L}{\partial b_{21}} = +ve$

$b_{21} \uparrow$ $L \uparrow$

$b_{21} \uparrow$

$\frac{\partial L}{\partial b_{21}} = -ve$

$b_{21} \uparrow$ $L \downarrow$

$b_{21} \uparrow$

game derivative

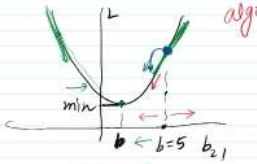
-ve of the gradient

$b_{21} = b_{21} \boxed{\frac{\partial L}{\partial b_{21}}}$

(b_{21})

$\frac{\partial L}{\partial b} = \text{slope}$

(+ve)



min

$b \leftarrow b=5$ b_{21}

algo

$b = -5$

$\frac{\partial L}{\partial b} = -ve$

-ve

slope (gradient) → -ve

$\boxed{\text{Slope}}$

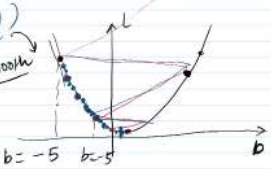
$W_n = W_0 - \boxed{\frac{\partial L}{\partial w}}$ $\boxed{\eta}$ $\boxed{W_n = W_0 - \eta \frac{\partial L}{\partial w}}$

→ Effect of Learning Rate (η)

$W_n = W_0 - \eta \boxed{\frac{\partial L}{\partial w}}$

0.1/0.01

smooth



$b = -5$ $b = -5$ b

train

$\boxed{\eta = 1}$

η = parameter → 0.01

$\eta = 0.00001$

$\frac{\partial L}{\partial b} = -50$

$b = -5 - (-50) = 45$

$b = 45$

$\frac{\partial L}{\partial b} = 50$

$b = 45 - 50$

$b = -5$ $\frac{\partial L}{\partial b} = -50$ $\eta = 0.01$

$b = -5 + (0.01 \times 50) = 0.5$

$= -4.5$

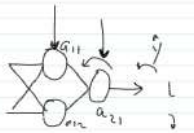
→ What is convergence?

$W_n = W_0$

$W_n = W_0 - \eta \frac{\partial L}{\partial w} = 0$

while → con → for 1000

epoch → 100



a_{11} a_{21} $\hat{y}$ L

$i = 1, 0$

$-(y - \hat{y})$

activation = 2

$\hat{y}(1 - \hat{y})\, o_{11}$

$-y \log(a_{21}) - (1-y)\log(1 - a_{21})$

$-(y - \hat{y})\, o_{11}$

$-\frac{y}{\hat{y}} + \frac{(1-y)}{(1-\hat{y})}$

$\frac{\partial L}{\partial w_{11}^2} = \boxed{\frac{\partial L}{\partial a_{21}}} \times \frac{\partial a_{21}}{\partial w_{11}^2}$

$\frac{\partial L}{}$ =

$$-\frac{y}{\hat{y}} + \frac{(1-y)}{(1-\hat{y})}$$

$$\frac{\partial L}{\partial w^2_{21}} = \left\{ \begin{array}{l} \boxed{-(y-\hat{y})\ a_{21}(1-a_{21})\ a_{11}} \\ -(y-\hat{y})\ a_{21}(1-a_{21})\ a_{12} \\ -(y-\hat{y})\ a_{21}(1-a_{21}) \end{array} \right\}$$

curr_activa $= a[1]$

$$\frac{\partial L}{\partial b_{21}} =$$

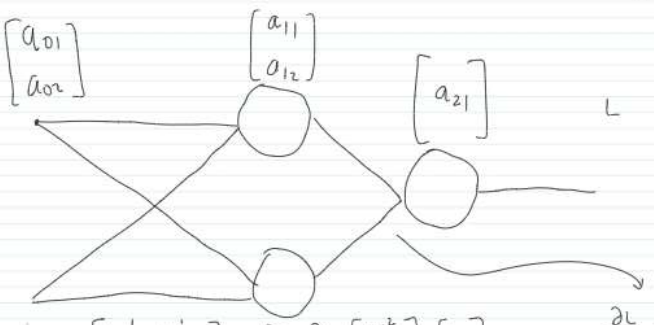
$$\text{delta} = (y-\hat{y})\ \underset{1\times1}{a^{(2)}(1-a^{(2)})}$$

$\boxed{w[1]}$ $\boxed{1\times2}$ $\boxed{1\times2}$

$$\boxed{-(y-\hat{y})\ a_{21}(1-a_{21})} \times a[1] \begin{bmatrix} a_{11} \\ a_{12} \end{bmatrix}$$

1×1 1×2

1×2 $\times$ 2×2



L $\quad L \to a_2 \to a_1 \to$

$$\frac{\partial L}{\partial a_2} \times \frac{\partial a_2}{\partial a_1} \frac{\partial a_1}{\partial w}$$

$$\begin{bmatrix} w^1_{11} & w^1_{12} \\ w^1_{21} & w^1_{22} \end{bmatrix} \quad \begin{bmatrix} b_{11} \\ b_{12} \end{bmatrix} \quad \begin{bmatrix} w^2_{11} \\ w^2_{12} \end{bmatrix} \quad \begin{bmatrix} b_{21} \end{bmatrix}$$

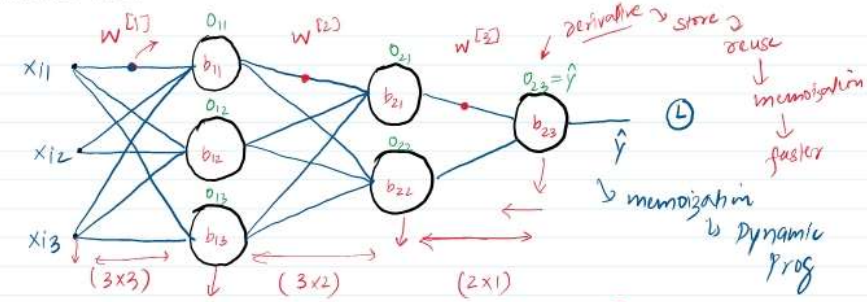
$$\frac{\partial L}{\partial w} = \boxed{\frac{\partial L}{\partial a_2}} \times \boxed{\frac{\partial a_2}{\partial w}}$$

$$\begin{bmatrix} \partial L/\partial w^1_{11} & \frac{\partial L}{\partial w^1_{12}} \\ \frac{\partial L}{\partial w^1_{21}} & \frac{\partial L}{\partial w^1_{22}} \end{bmatrix} \qquad \begin{bmatrix} \frac{\partial L}{\partial w^2_{11}} \\ \frac{\partial L}{\partial w^2_{12}} \end{bmatrix}$$

MLP Memoization

Thursday, April 7, 2022 7:45 AM

backprop update → w / b) deriv



derivative → store → reuse → memoization → faster

memoization → Dynamic Prog

Chain rule

$L \to \hat{y} \to o_{21} \to o_{11} \to w^1_{11}$

$$\frac{\partial L}{\partial w^3_{11}} = \boxed{\frac{\partial L}{\partial \hat{y}}} \times \boxed{\frac{\partial \hat{y}}{\partial w^3_{11}}} \qquad \boxed{\frac{\partial L}{\partial w^2_{11}}} = \boxed{\frac{\partial L}{\partial \hat{y}}} \times \left\{ \frac{\partial \hat{y}}{\partial o_{21}} \right\} \times \left\{ \frac{\partial o_{21}}{\partial w^2_{11}} \right\}$$

$$x \to f(x),\ g(x) \to h(f(x), g(x)) = \frac{\partial h}{\partial x} = \left[\frac{\partial h}{\partial f(x)} \times \frac{\partial f(x)}{\partial x} \right] + \left[\frac{\partial h}{\partial g(x)} \times \frac{\partial g(x)}{\partial x} \right]$$

$\frac{\partial \hat{y}}{\partial w^1_{11}}$ $\quad w^1_{11} \to o_{11} \to o_{21}, o_{22} \to \hat{y} \to L \qquad \frac{\partial L}{\partial w^1_{11}}$

$$\left[\frac{\partial L}{\partial w^1_{11}} = \boxed{\frac{\partial L}{\partial \hat{y}}} \left[\boxed{\frac{\partial \hat{y}}{\partial o_{21}}} \times \boxed{\frac{\partial o_{21}}{\partial o_{11}}} \times \frac{\partial o_{11}}{\partial w^1_{11}} + \frac{\partial \hat{y}}{\partial o_{22}} \times \frac{\partial o_{22}}{\partial o_{11}} \times \frac{\partial o_{11}}{\partial w^1_{11}} \right] \right]$$

Backprop → Chain diff (rule) + Memorization

maths

(is)

Keras

TF

Gradient Descent

Thursday, April 7, 2022 7:44 AM

Gradient descent is one of the most popular algorithms to perform optimization and by far the most common way to optimize neural networks.

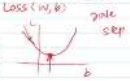
→ optimization

$J(\theta)$ = Loss function

Neural

$\eta \frac{\partial L}{\partial w}$

Loss (w, b)



b

Gradient descent is a way to minimize an objective function $J(\theta)$ parameterized by a model's parameters $\theta \in R^d$ by updating the parameters in the opposite direction of the gradient of the objective function $\nabla_\theta J(\theta)$ w.r.t. to the parameters. The learning rate η determines the size of the steps we take to reach a (local) minimum. In other words, we follow the direction of the slope of the surface created by the objective function downhill until we reach a valley.

Back propagation Algorithm

epochs = 5 ↓ SGD

for i in range(epochs): — 1 point

  for j in range(x.shape[0]): — gradient descent

    → Select 1 row (random)

    → Predict (using forward prop)

    → Calculate loss (using loss function → mse)

    → Update weights and bias using GD

$$w_n = w_0 - \eta \frac{\partial L}{\partial w}$$

  → Calculate avg loss for the epoch

  Log

50 points

1 epoch

↳ 50 times

w, b update

Batch GD

epoch = 5

↳ 50 points → predict ↓ — current weight

dot product

y_hat = np.dot(X, w) + b

↳ 50 predict

1 single (w, b)

y = 50 actual ocu.

y y_hat ↓ loss

$$\sum_{i=1}^{50}$$

Batch / Stochastic / mini batch

There are three variants of gradient descent, which differ in how much data we use to compute the gradient of the objective function. Depending on the amount of data, we make a trade-off between the accuracy of the parameter update and the time it takes to perform an update.

$\frac{\partial L}{\partial w}$ derivative ↓ 3 types

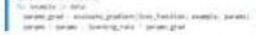
accuracy → time → bandwidth.

→ Batch GD (Vanilla GD)

entire dataset ↓ update

epochs = # of updates

Stochastic GD



epoch → 10 (50 rows) — 50 rows

for i in range(10): → shuffle

  for i in range(X.shape[0])

    ↳ 1 random point → bias eliminate

    ↳ y_hat → forward

    ↳ loss

    ↳ w, b update → $w_n = w_0 - \eta \frac{\partial L}{\partial w}$

  avg loss print → for the epoch

50×10 = 500

10 × 50 = 500 update

5 epochs

↳ 5 times

↓

w, b update

frequency of weight update higher

Batch GD → 1 loop

epoch = 10 — epoch

for in range(10): — rows — loop → total → 10 times w, b update

y_hat = np.dot(X, w) + b

50 values — dot → smart replacement → loops

↳ vectorization ↳ faster loop

y = 50 values

50 values — dot → smart replacement → loops

↳ vectorization ↳ faster loop

y = 50 values

y_hat, y → loss

w, b update $w_n = w_0 - \eta \frac{\partial L}{\partial w}$

Optimized

→ loss

Which is faster (given same no. of epochs)

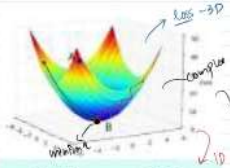
10n

320 (n) 1

Which is the faster to converge (given same # epochs)

5

Loss function → 2 Algo



loss → 3D

complex

weight

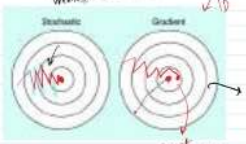
1D

Stochastic

↳ random → point ↓ updates

Batch ↓ dataset

all point ↓ 1 update



Spiky SGD useful

Yes | No

SGD

BGD

SGD ↓ help the algo to move out of local minima

local minima

Exact solution

Approximate diff

Vectorization — faster → loops — big dataset

↳ np.dot(X, w) + b — RAM

X → dataset of 10 crore

Mini Batch Gradient Descent

SGD → BGD

$\frac{n}{x}$ = # of batches

Keras batch-size = x

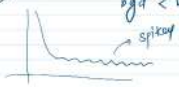
updates/epoch

bgd > mbgd > sgd

cost

bgd < mbgd < sgd

MBGD



spiky SGD

BGD ↔ SGD — Best of both worlds

320 rows

in every epoch

10 batches — batches ↓ 32

10 update

for i in epochs →

  for j in num of batch

    1 batch

    ↳ y_pred (vector — np.dot

    ↳ loss

    ↳ update

Vectorization → ↓ smaller batch

→ Why batch_size is provided in multiple of(2)?

2, 4, 8, 32, 64, 128, 256

10, 15

RAM effective → binary RAM → optimization

faster

→ What if batch_size doesn't divide # rows properly

e.g # of rows $n = 400$

batch_size = 150

# of batch = $\frac{400}{150} = \boxed{2.66}$

150, 150, left 100

1 batch, 2 batch, 3rd batch

Vanishing Gradient Problem

Thursday, April 7, 2022 7:45 AM

In machine learning, the **vanishing gradient problem** is encountered when training artificial neural networks with gradient-based learning methods and backpropagation. In such methods, during each iteration of training each of the neural network's weights receives an update proportional to the partial derivative of the error function with respect to the current weight. The problem is that in some cases, the gradient will be vanishingly small, effectively preventing the weight from changing its value. In the worst case, this may completely stop the neural network from further training. ~~A frequent example of the problem in deep feedforward networks~~

GD

1) $0.1 \times 0.1 \times 0.1 \times 0.1 = \boxed{0.0001}$ → VGP

2) Deep NN

3) Sigmoid / tanh → Af →

$\boxed{0-1}$

Backprop

$\boxed{0-0.5}$

deep

sigmoid

$\hat{y} - y = L$

$L \to \hat{y} \to o_{11} \to w'_{11}$

$$\frac{\partial L}{\partial w'_{11}} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z} \times \frac{\partial z}{\partial o_{11}} \times \frac{\partial o_{11}}{\partial w'_{11}}$$

$0 < x < 1$ $\quad 0 < x < 1$ $\quad 0.1$

sigmoid derivative

0.0001

$$w_n = w_0 - \eta \frac{\partial L}{\partial w}$$

derivative of L wrt weight

small change

$w_0 = 1$

$w_n = 1 - 0.01 \times 0.0001$

$w_n = 0.99999$ → vanishingly small

did not change

{ sigmoid, tanh }

$$\frac{\partial L}{\partial w'_{11}} = 0.000001$$

Backprop → model training → ×

80's, 90's → main reasons DL failed

$0 - 100000$

$\boxed{0-1}$

$\boxed{0-1}$

How to recognize?

1) Loss focus → epoch → no changes → VGP

2) weights → graph value → Tensorboard → callbacks

w'_{11}

Keras

{ loss after epoch }

epoch

$$w_n = w_0 - \eta \frac{\partial L}{\partial w}$$

$$\left| \frac{w_0 - w_n}{\eta} \right| =$$

$$w_n = w_0 - \eta \boxed{\frac{\partial L}{\partial w}} \qquad \boxed{\frac{w_0 - w_n}{\eta}} =$$

How to handle Vanishing Gradient Problem →

1) Reduce model complexity

$\boxed{\frac{\partial L}{\partial w}}$ = bigger

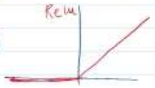
$\boxed{d_1 d_2 d_3}$

$d_1 \times d_2 \times d_3 \times d_4 \times \ldots$

Applicable → find out complex patterns

2) Using $\boxed{\text{ReLU}}$ Activation functions

max (0, z)



→ $\boxed{0}$, $\boxed{1}$

$= 1 \times 1 \times 1 = 1$ → VGP

1000 − 1000 → squish

$\boxed{0}$ − 1000

$\boxed{0-1}$

dying relu → derivative $w_n = w_0$

{leaky relu}

3) Proper weight Init → Glorot, Xavier

4) Batch norm → layer →

5) Residual Network → CNN → ResNET

↳ building block → ANN

Exploding Gradient Problem → $\boxed{\text{RNN}}$

big number

10, 10, 10, 10 → 10000

$\frac{\partial L}{\partial w_{11}} = d_1 \times d_1 \times d_2 \times d_3$

random

$w_n = w_0 - \eta \frac{\partial L}{\partial w}$ (0.1, 1000)

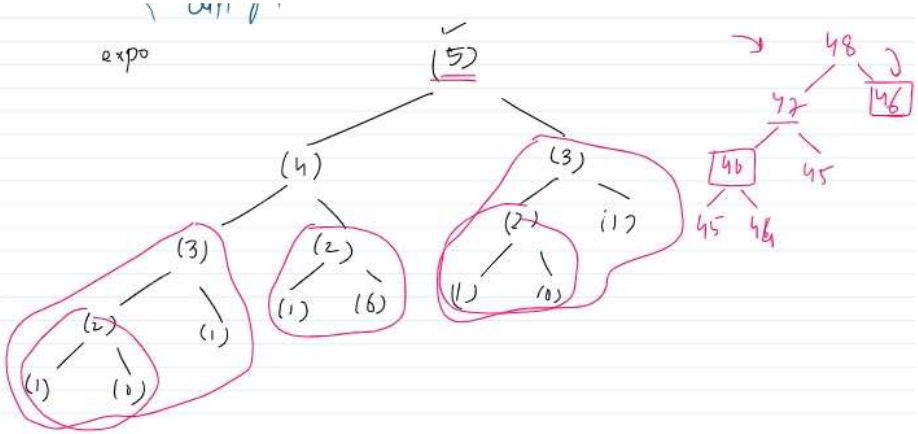
Loss↓ ×

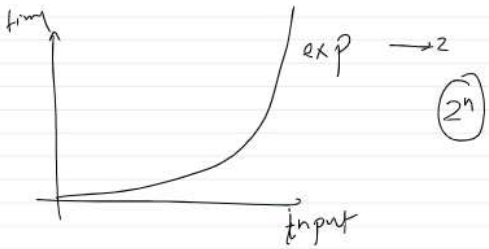
1 — 100 → (97) → 1000

{Gradient clipping}

expo (5)

48





2^n

Saturday, April 9, 2022 10:01 AM

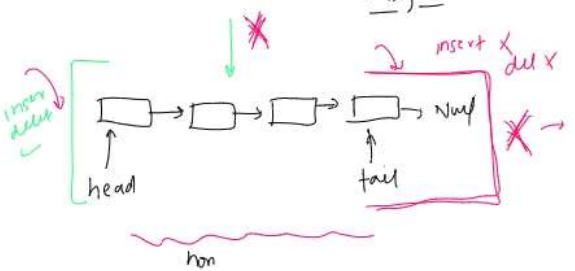
DS Algo → Time Complex
→ Arrays → Dynamic Arrays
→ Linked List

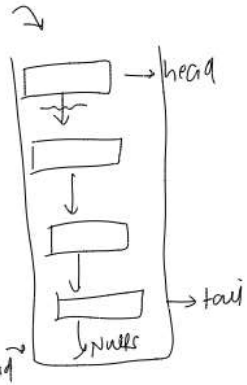
Today → Stacks → LL
→ Arrays

→ 3 → Tuesday
last 3 videos

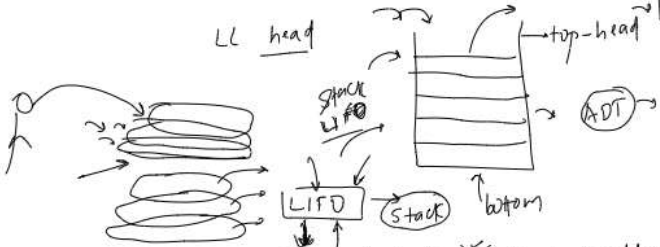
Next → Queues
→ Practice →

→ Stacks → DS
useful





LL head



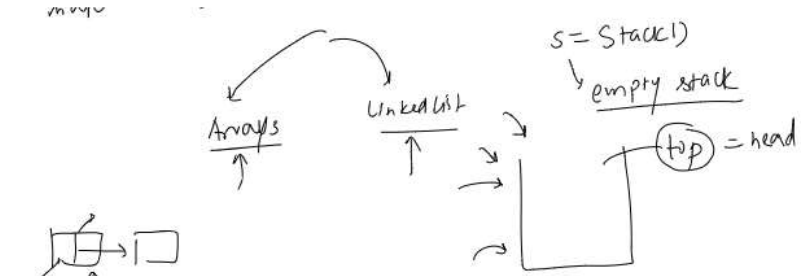
ADT → conceptual thing
↓
Normal data type

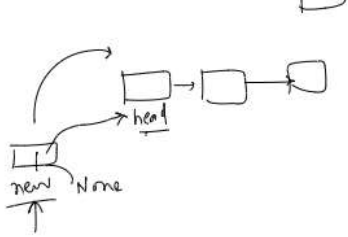
LIFO → Stack
last in first out

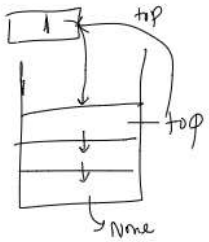
push → adding new node from top
is_empty → empty
pop → removing from top
peek → top node → value
size → kitna nodes

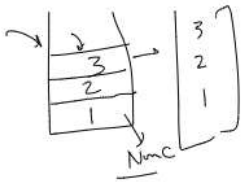
LIFO
→ Expression evaluation
→ Undo redo
→ browser back button

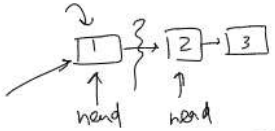
s = Stack()



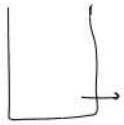




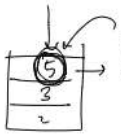




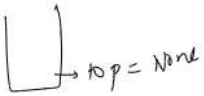
[head = head.next]



top = None
none → Next



peek (5)



top = None

LIFO
Advantage of Stack

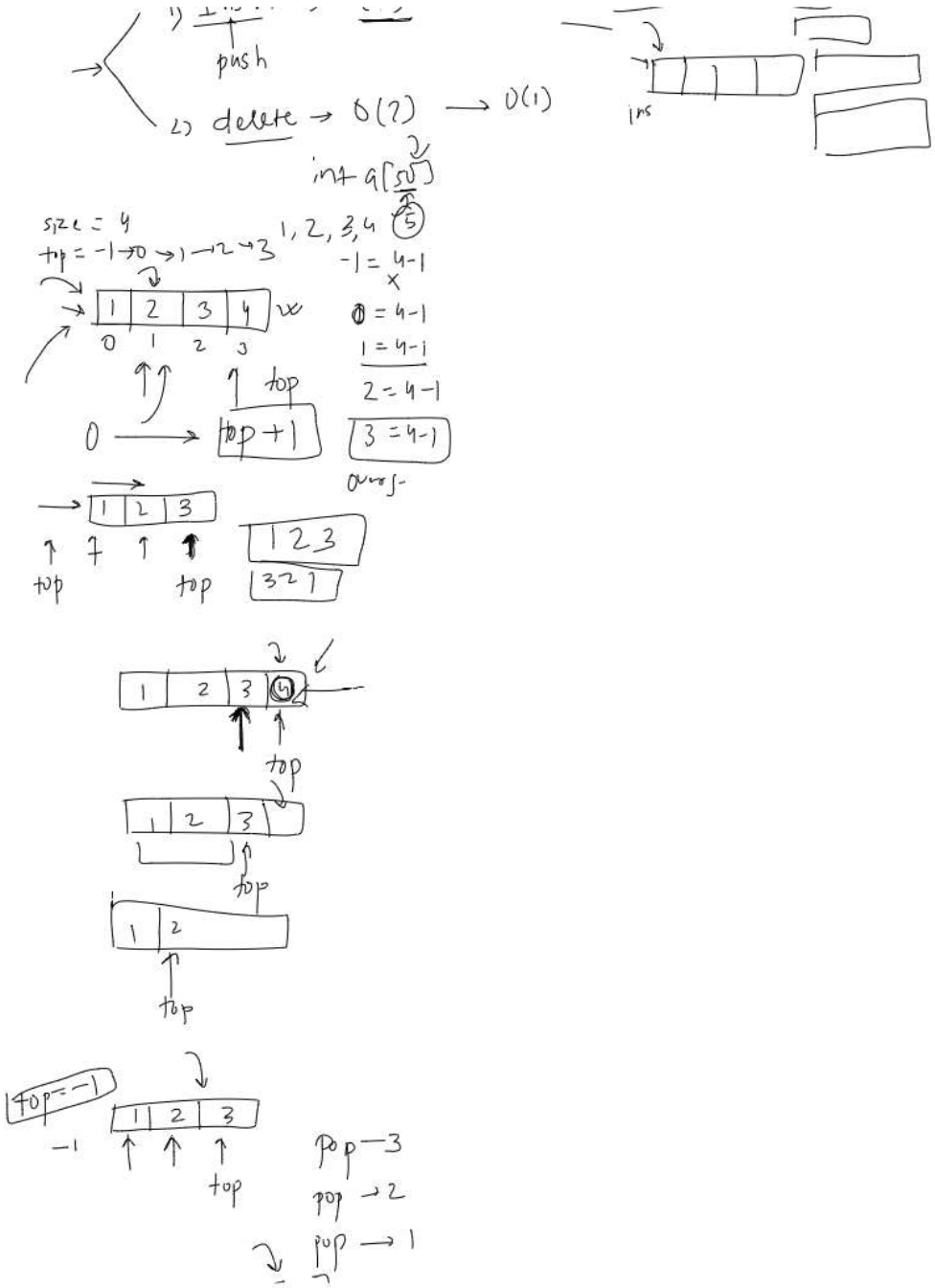
1) Insert → O(?) → O(1)
push

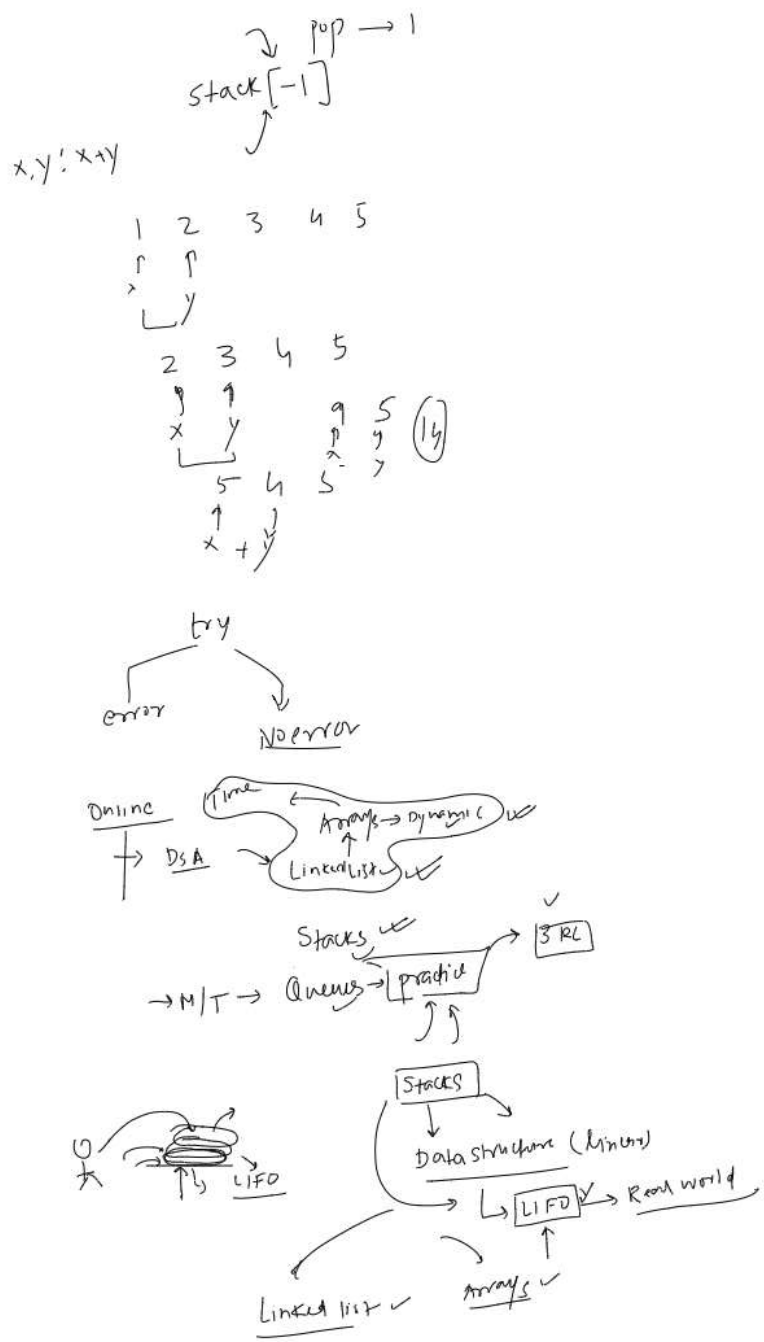
Write

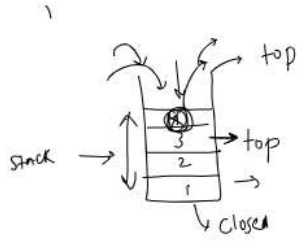
LL →
Arrays

Static → int a[50]

Dynamic

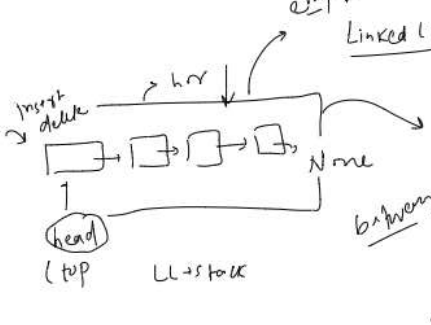

push
delete → O(2) → O(1)
ins
int a[5]
size = 4
top = -1 → 0 → 1 → 2 → 3
1, 2, 3, 4
-1 = 4-1
0 = 4-1
1 = 4-1
2 = 4-1
3 = 4-1
top
top+1
queue
123
321
top
top
top
top
top = -1
pop → 3
pop → 2
pop → 1

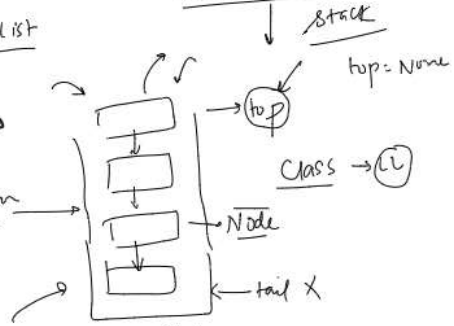

pop → 1
stack[-1]
x, y : x+y
1 2 3 4 5
2 3 4 5
5 4 5
try
error
No error
Online
DSA
Time
Arrays → Dynamic
Linked List
Stacks
Queues
practice
3 RC
M/T
Stacks
Data structure (linear)
LIFO
Real world
Linked list
Arrays



push
pop
peek
is empty
size

Simple

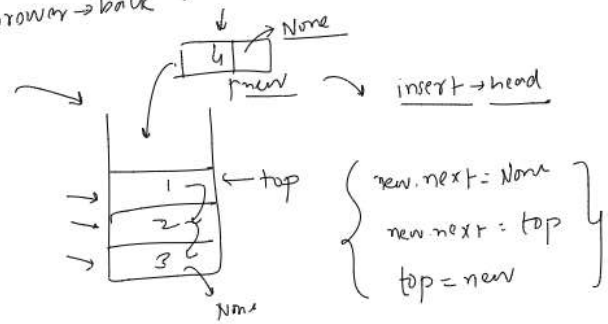




LL

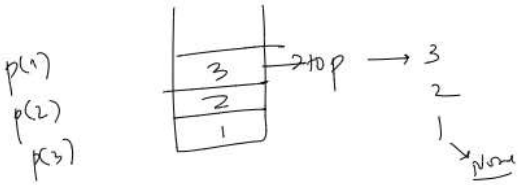
Application
- → expression
- → undo/redo
- → recursion →
- → browser → back

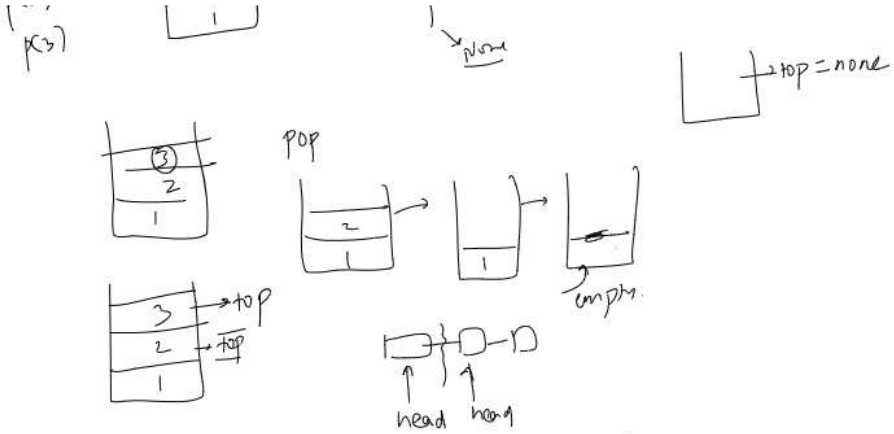
LIFO

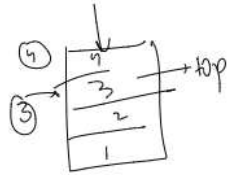


new.next = None
new.next = top
top = new

1 2 3







Array → Space

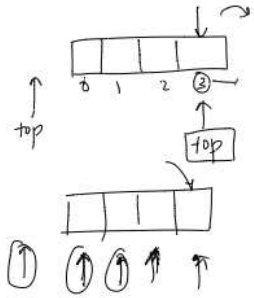
Static (different) ↔ Dynamic (HW) easy

C++ / Java

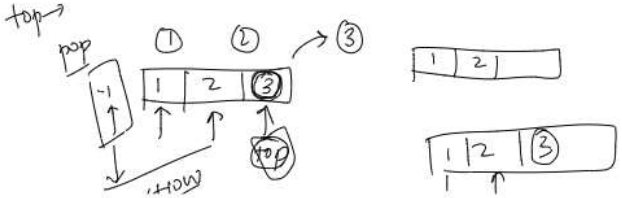
fixed

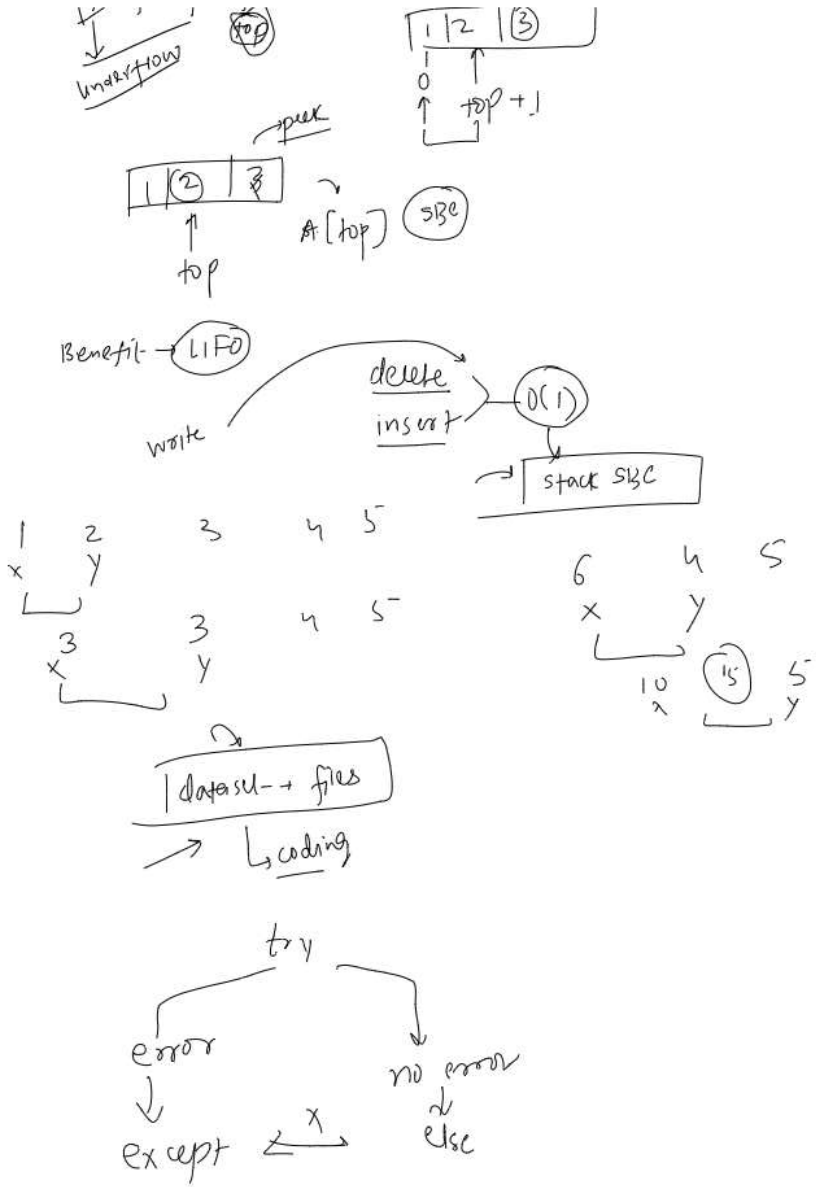
size = 4 − 1

value = 2



check
top = = size − 1

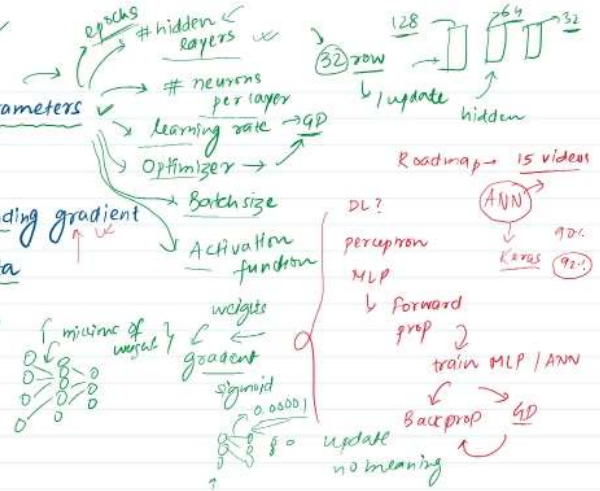


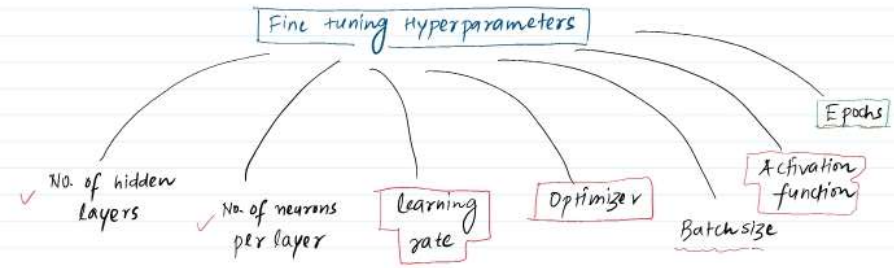


How to improve a neural network

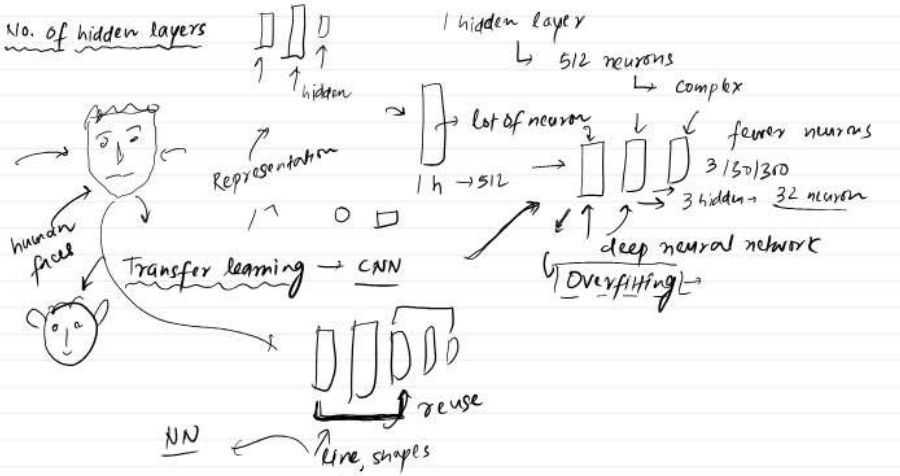
29 April 2022 13:51

1. Fine tuning NN hyperparameters
   - epochs
   - # hidden layers
   - # neurons per layer
   - learning rate → GD
   - Optimizer →
   - Batch size
   - Activation function
2. By solving problems:
   - → Vanishing / Exploding gradient
   - → Not enough data
   - → Slow training
   - → Overfitting





1 No. of hidden layers



2 Neuron / layer

equal (a) place

2 neurons → pyramid

experiment

64 → 32 → 16

decrease

primitiv

more → less

32 32 32

nodes sufficient

1 capture → recover

more than what is required

3. Batch size
- Batch → all row → weight update
- Stochastic → 1 row → weight
- Mini Batch → batch → 32 → row → update

hyper parameter

smaller (8 to 32) → slow

generalize → result better

large (8/92) → fast

good result

learning rate → small / large

GPU ram

warming up the lr

epochs →

warming up the LR

4. Epochs → 100 / 500 / 1000 → early stopping

Keras → callback

Keras → stable

Problems with Neural Networks

Vanishing and exploding gradients
- → Weight Init →
- → Activation function →
- → Batch Norm →
- → Gradient clipping →

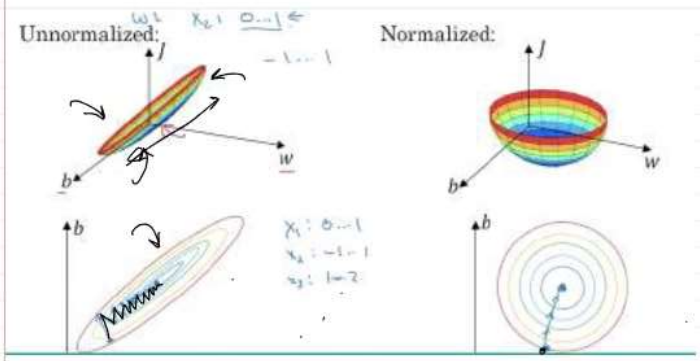
Not enough data
- → Transfer learning
- → Unsupervised pretraining

Slow training → Adam
- → Optimizers
- → learning rate schedular

Overfitting
- → l_1 and l_2 reg
- → Dropouts

Feature Scaling

05 May 2022 17:37



(age)

x_1 x_2 (salary)

w_1 $\boxed{w_2}$

$$w_2 = w_2 - \eta \frac{\partial L}{\partial w_2}$$

Standard

$$\frac{x_i - \mu}{\sigma}$$

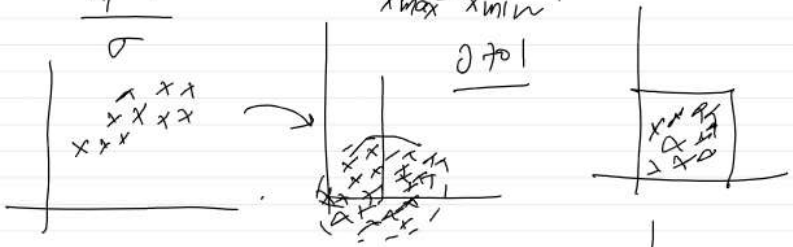
Normalize

$$\frac{x_i - x_{min}}{x_{max} - x_{min}}$$

min = 0

max = 1

0 to 1



$\boxed{-1 \text{ to } +1}$

$\boxed{2\times2 \qquad 2\times3}$

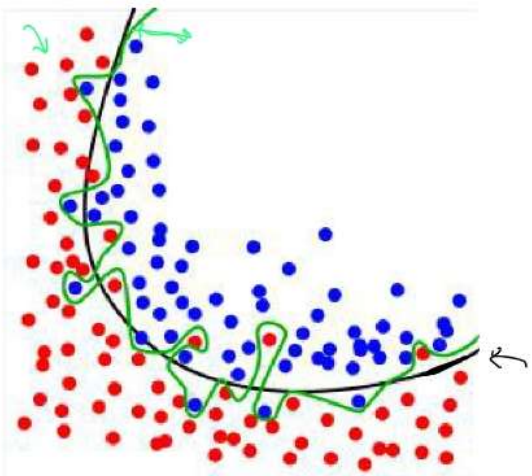
$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} \boxed{1} & \boxed{2} & \boxed{3} \\ \boxed{4} & \boxed{5} & \boxed{6} \end{bmatrix}$$

$$\begin{bmatrix} 1+8 & 2+10 & 3+12 \\ 3+16 & 6+20 & 9+24 \end{bmatrix}$$

$$\begin{bmatrix} 9 & 12 & 15 \\ 19 & 26 & 33 \end{bmatrix}$$

The Problem of Overfitting

12 May 2022 09:32

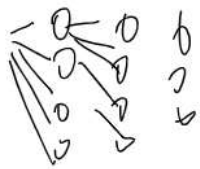


ANN

overfitting

complex

epochs



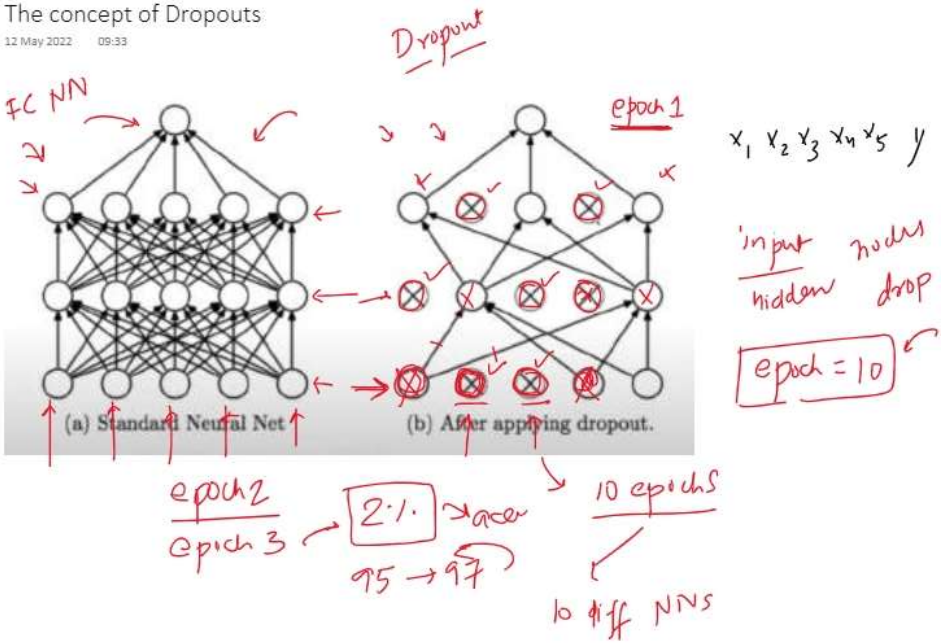
Possible Solutions

12 May 2022 09:33

1) Add more data
2) Reduce complexity
3) Early stopping
4) Regularization → L1, → L2
5) Dropout

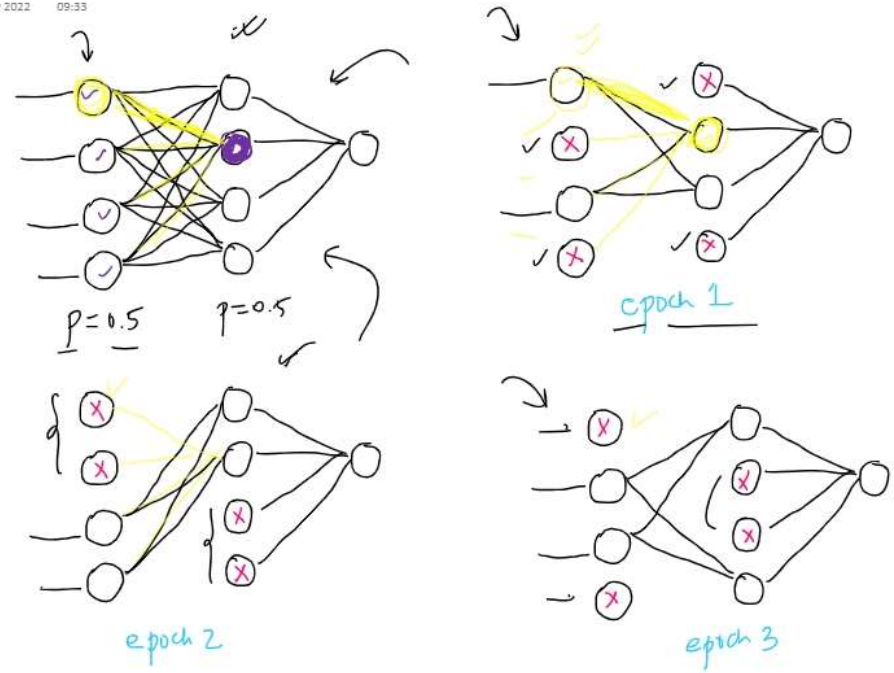
The concept of Dropouts

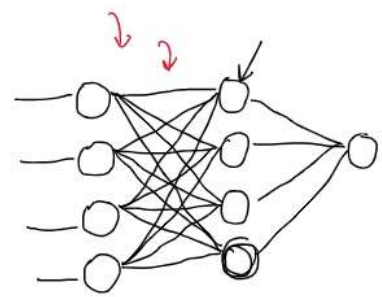
12 May 2022 09:33



Why this works?

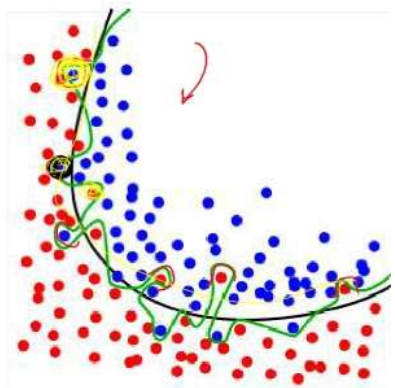
12 May 2022 09:33





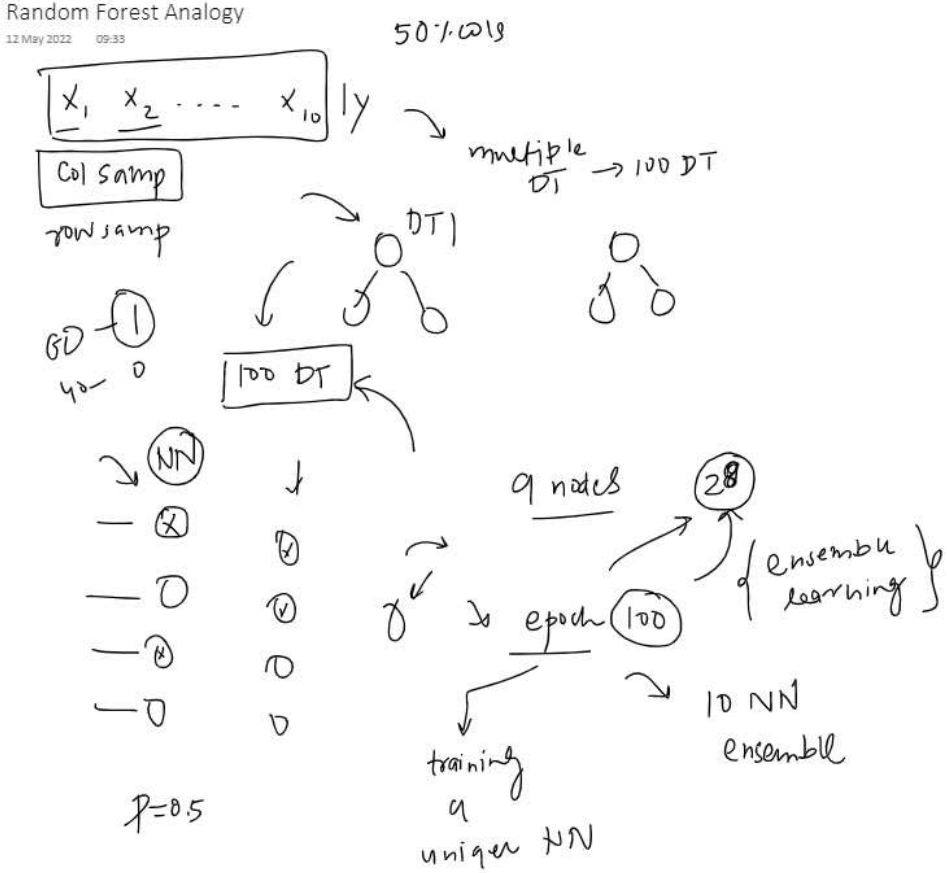
→ Reduce the # nodes

→



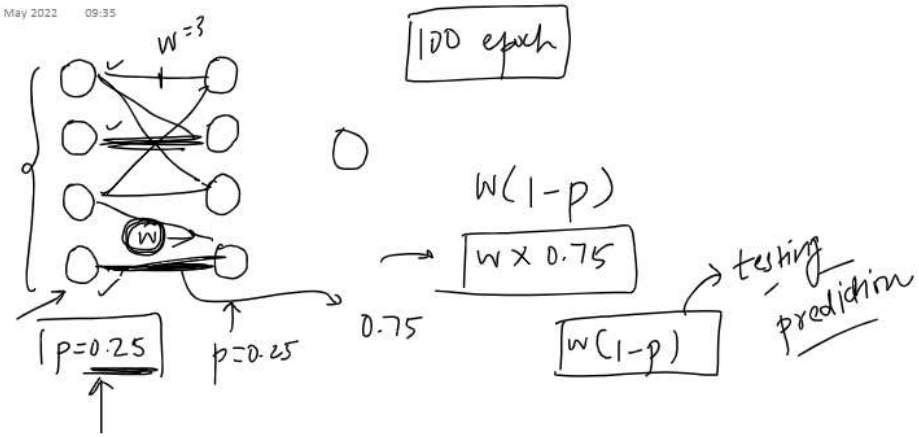
Random Forest Analogy

12 May 2022 09:33



How prediction works?

12 May 2022 09:35



Regression Code Example

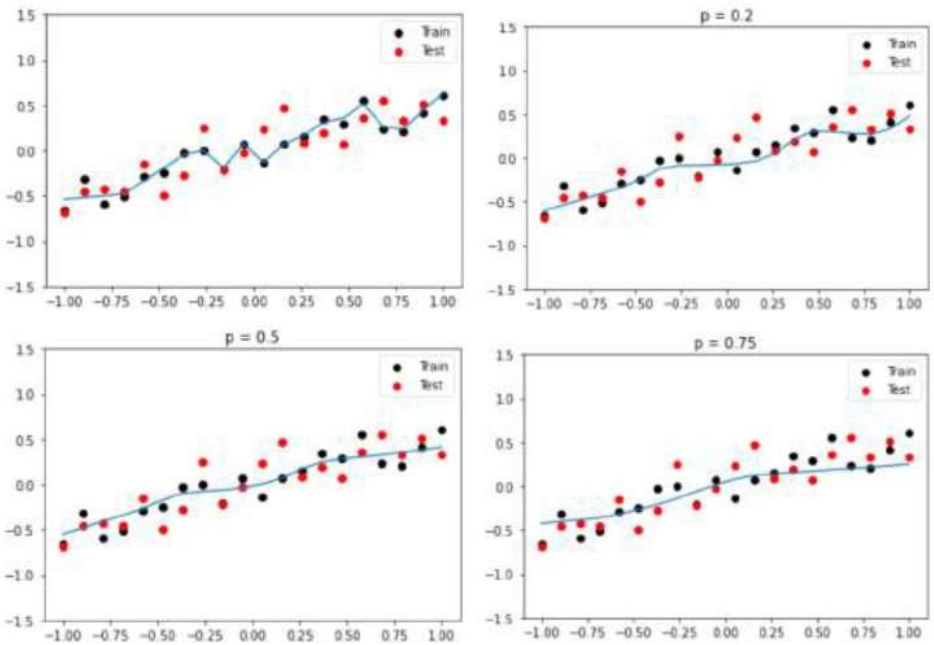
12 May 2022 09:34

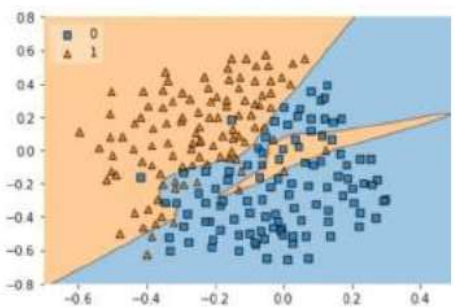
Classification Code Example

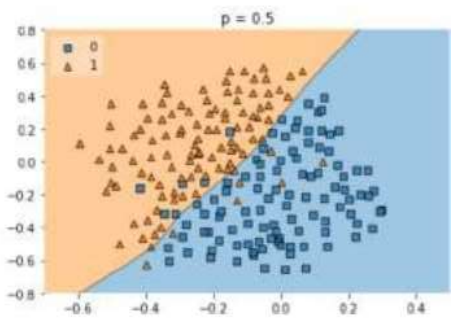
12 May 2022 09:34

Effect of p

12 May 2022 10:06





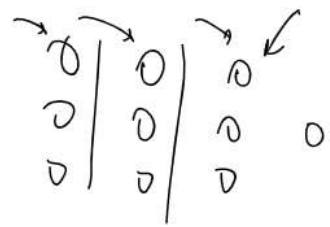


Practical Tips and Tricks

12 May 2022 09:34

1) Overfitting P↑, underfitting P↓

2) Last layer → dropout



3) CNN → 40-50 · P → ✓

20-30 RNN

50>

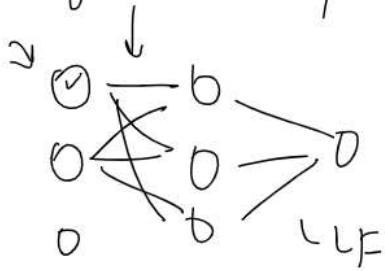
ANN → 10-50

Drawbacks

12 May 2022 09:34

1) convergence → delay

2)



{ loss function }

value changes

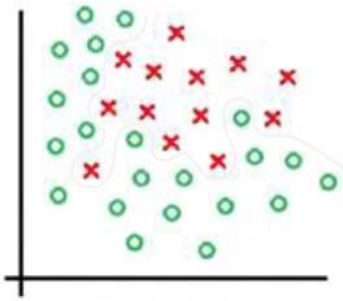
LF – has to change

Resources

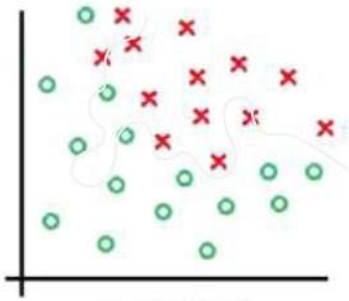
12 May 2022 12:38

Overfitting

20 May 2022 08:14



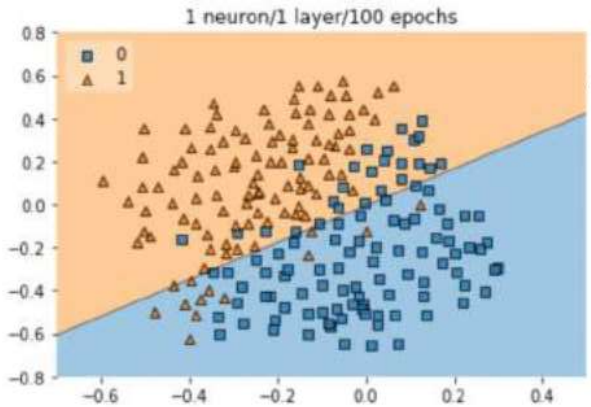
Train Dataset

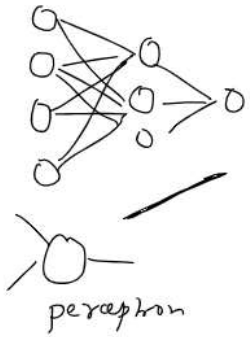


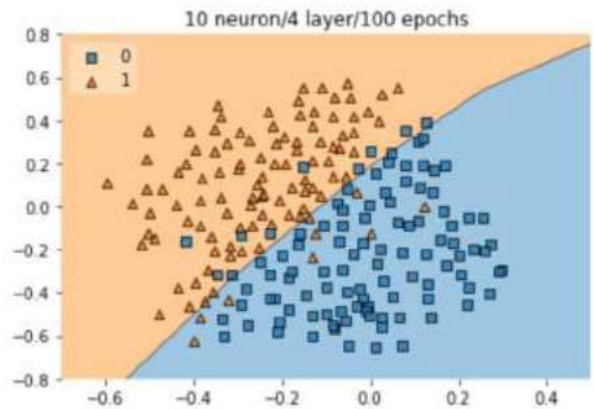
Test Dataset

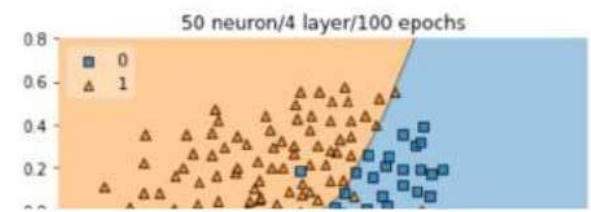
Why Neural Networks Overfit?

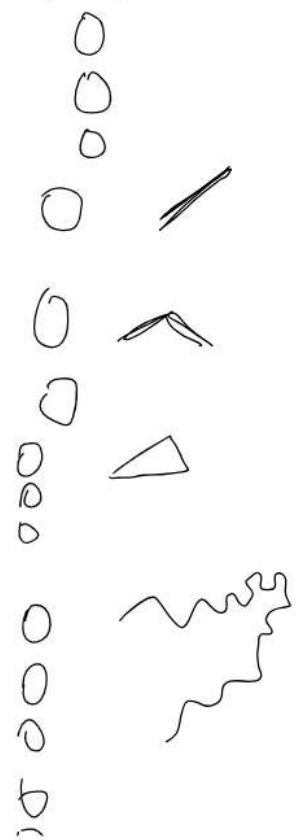
19 May 2022 07:34

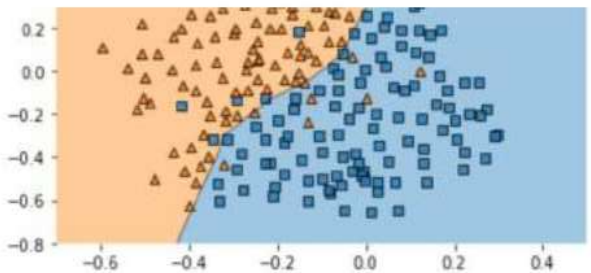


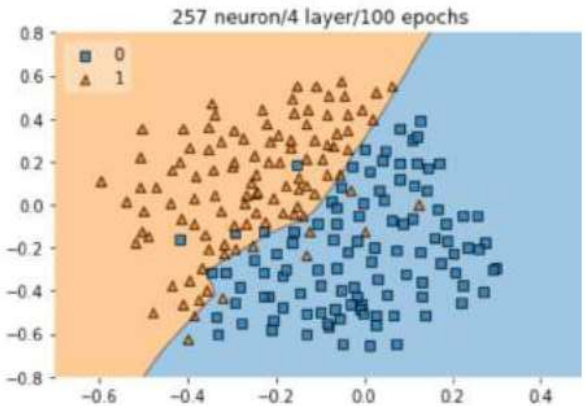


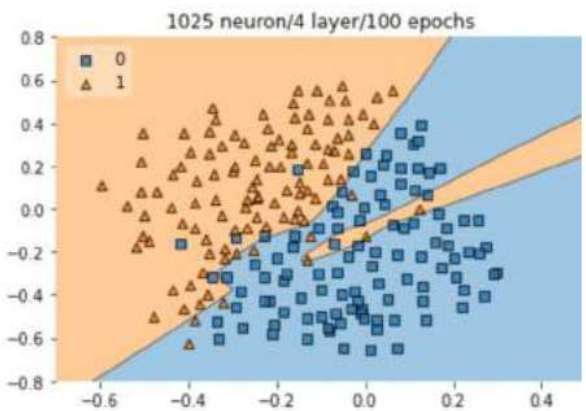


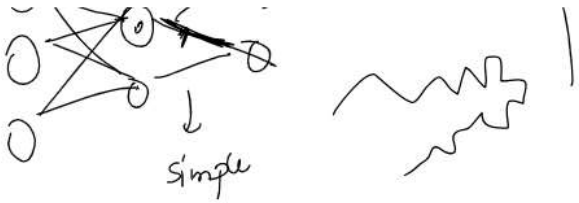


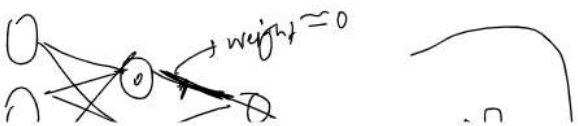





257 neuron/4 layer/100 epochs
0
1


1025 neuron/4 layer/100 epochs
0
1


simple


weight ≃ 0

Ways to solve overfitting

20 May 2022 08:13

- Adding more data
  - → Add more rows
  - → Data augmentation
- Reducing the complexity model
  - → Dropout
  - → Early stopping
  - → Regularization
    - → L1
    - → L2 → W
    - → L1 + L2

Regularization

20 May 2022 08:14

ANN → weights | bias

↳ min Loss function

$$C = \boxed{\frac{1}{n}\sum_{i=1}^{n} L(y_i, \hat{y}_i)} + \text{penalty term}$$

L = mse

↳ binary

$\boxed{L2}$

L1

$\boxed{w_1 \to w_{10}}$

$$C = L + \boxed{\frac{\lambda}{2n}\sum_{i=1}^{k} \|w_i\|^2} \to \text{weightage}$$

$$\frac{\lambda}{2n}\left[w_1^2 + w_2^2 + \dots + w_{10}^2\right]$$

↑ λ = hyperparameter ↑

$\boxed{\lambda = 0}$

$$C = \sum L(y_i, \hat{y}_i) + P$$

$$\downarrow$$

$$w \simeq 0$$

L2 → (L1)

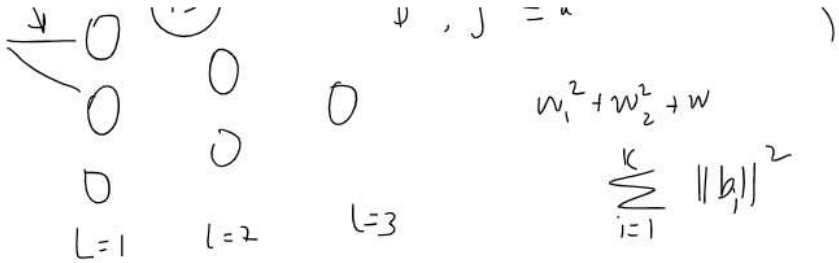
→ L1 norm

$$C = L + \frac{\lambda}{2n}\sum \|w_i\|$$

$$C = \sum_{i=1}^{n} L(y_i, \hat{y}_i) + \boxed{\sum_{l=1}^{L}\sum_{i=1}\sum_{j=1} \|w_{ij}^{l}\|^2} \quad \boxed{w \simeq 0}$$

$\to 0$ (15) $\cap$

$l, j = l$



$w_1^2 + w_2^2 + w$

$$\sum_{i=1}^{k} \|b_i\|^2$$

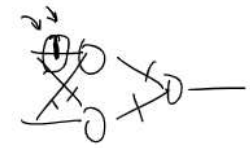
$\boxed{\text{bias } X}$

Intuition behind Regularization

20 May 2022 08:15

$$\boxed{w_n = w_0 - \eta \frac{\partial L}{\partial w_0}}$$

$\boxed{1-\eta\lambda}$ positive



$$L' = L + \frac{\lambda}{2} \sum \|w_i\|^2$$

$w_2^1 + \boxed{w_2^2 + w_3 + \dots}$

$2w$

$$\frac{\partial L'}{\partial w_0} = \frac{\partial L}{\partial w_0} + \frac{\lambda}{\cancel{2}} \cancel{2} w_0$$

$$= \frac{\partial L}{\partial w_0} + \lambda w_0$$

$$w_n = w_0 - \eta\left(\frac{\partial L}{\partial w_0} + \lambda w_0\right)$$

$$w_n = w_0 - \eta\lambda w_0 - \eta \frac{\partial L}{\partial w_0}$$

$$\boxed{w_n = (1-\eta\lambda) w_0 - \eta \frac{\partial L}{\partial w_0}}$$

$w_0 << w_0$

L2 reg → weight decay

weight decay → w_0

Code Demo

20 May 2022 08:15

Further Resources

20 May 2022 08:15

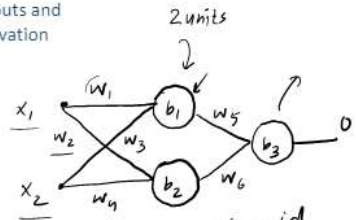
What are Activation Functions?

31 May 2022 14:49

In artificial neural networks, each neuron forms a weighted sum of its inputs and passes the resulting scalar value through a function referred to as an activation function or transfer function. If a neuron has n inputs then the output or activation of a neuron is

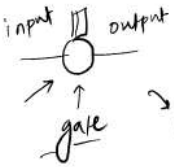
$$a = g(w_1x_1 + w_2x_2 + w_3x_3 + \ldots w_nx_n + b)$$

This function g is referred to as the activation function.



$$\underbrace{g}_{\text{activation}}(w_1x_1 + w_2x_2 + b_1)$$

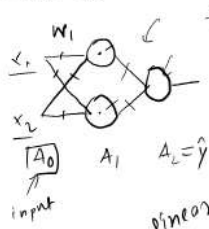
sigmoid
relu
softmax



activated (Y/N)
how much

Why Activation Functions are needed?

31 May 2022 14:50



forward prop

$$Z_1 = \boxed{W_1 A_0 + b_1}$$

$$A_1 = g(Z_1) = Z_1$$

linear → sigmoid

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

ANN

↓

Non-linear

Non-linear

output = input

$$A_2 = g(W_2 A_1 + b_1)$$

$$= W_2 A_1 + b_2 = W_2(W_1 A_0 + b_1) + b_2$$

$$= W_2 W_1 A_0 + W_2 b_1 + b_2$$

$$= W' A_0 + b'$$

output

input

$$A_2 = \hat{y} = W' A_0 + b'$$

linear

Ideal Activation Function

31 May 2022 14:50

$y = f(x)$ linear x non-linear

$\sigma(z) = \frac{1}{1+e^{-z}}$

1) Non-linear → Universal Approx Theorem

2) Differentiable → gradient des → derivatives → relu (0)

3) Computationally inexpensive → derivative → simples, easy, fast — training slow

4) Zero centered → zero cent normalized → mean = 0 → normalized data input → later layers → tanh

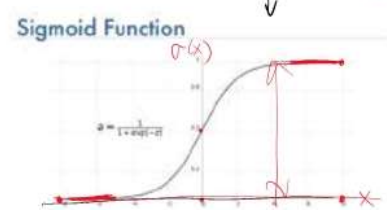
5) Non-saturating → Sigmoid → $\boxed{0,1}$ → $f(x) = \max(0,x)$

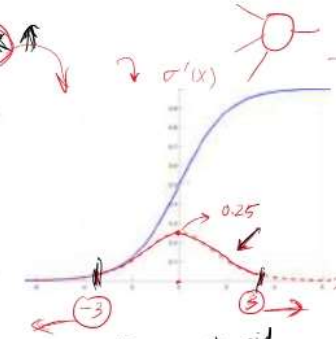
tanh → (−1, 1) — faster — Relu

update x $\boxed{0,1}$ → backprop ← 0.01 ← 0.0001 ← 0.0001 — saturating AF → Vanishing Gradient Problem

Sigmoid Activation Function

31 May 2022 14:50





$w_1x_1 + w_2x_2 + b$

input ↑ → Norm

$x \simeq 0$

$\sigma'(x) \simeq 0$

$\boxed{w_1\ w_2\ b}$ weight init

$\boxed{\sigma(x) = \frac{1}{1+e^{-x}}}$ Yes / No

sigmoid → 0-1 — yes / No

Advantages

1) [0, 1] → probability → output layer → Binary classification

2) Non-linear → Non-linear data → good option

3) Differentiable → Backprop → $\boxed{\frac{\partial L}{\partial w}}$ ✓ x

Disadvantages

1) Saturating function → $[-\infty, \infty] \to [0,1]$

↳ Vanishing gradient problem

hidden layer — sigmoid — training x

output → binary classification

Backprop → update $w_n = w_0 - \eta \frac{\partial L}{\partial w}$

$\boxed{w_n = w_0}$ training x

No update will take place

2) Non Zero centered → +ve / −ve → Training slow

Normalized → mean = 0

zero

Convergence problem

+ve −ve

$\boxed{0 \to 1}$

$\boxed{\frac{\partial L}{\partial w}} = \boxed{\frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z}}$ a_{21} → +ve

128 neurons

$$\frac{\partial L}{\partial w_{21}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_{31}} a_{21} \rightarrow +ve$$

$$\frac{\partial L}{\partial w_{22}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_{31}} a_{22} \rightarrow +ve$$

$+ve$

$-ve \quad +ve$

$$\frac{\partial z_{31}}{\partial w_{21}} = a_{21} w_{21} + a_{22} w_{22} + b_{31}$$

$L \rightarrow \hat{y} \rightarrow z$, $\sigma(z)$

b_{31}, a_{21}, w_{21}, a_{22}, w_{22}

gradient

change $\{+ve, -ve\}$

$+ve$, $-ve$

restricted

training slow

$\boxed{+ve}$ $\boxed{-ve}$

(W) mani

(L) min

(L) min

left turn

time more

slowly con

zigzag

3) $\sigma(x) = \frac{1}{1+e^{-x}}$ → computation (exp) time

hidden layer → sigmoid

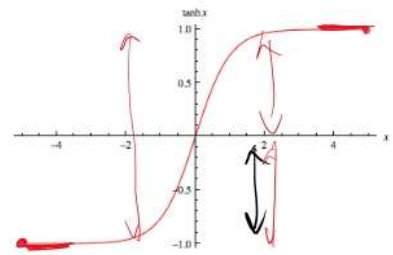
output → b cpx

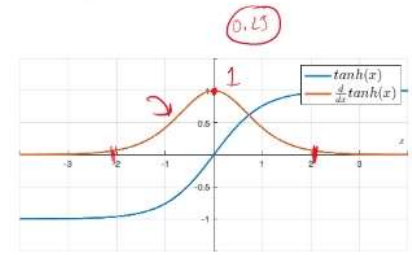
relu

Tanh Activation Function

31 May 2022 14:50

(0,1) (-1,1)





$$f(x) = \frac{(e^{x} - e^{-x})}{e^{x} + e^{-x}}$$

$$f'(x) = (1 - \tanh^2(x))$$

Advantages

1) Non-linear
2) Differentiable
3) zero centered $\{+ve, -ve\}$ training faster sigmoid

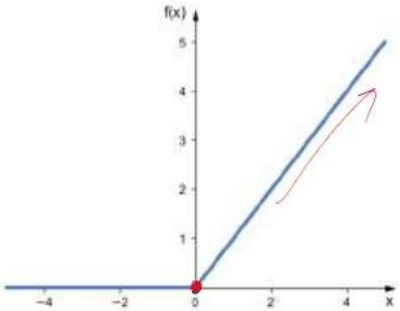
Disadvantage

→ saturating function → vanishing gradient prob

→ computationally exp $\boxed{exp}$ slow

Relu Activation Function

01 June 2022 16:43



activation

$f(x) = \max(0, x)$ $\quad f(x) = x$

Advantage

$\boxed{f(x) = \max(0, x+1) - \max(0, x-1)}$

1) Non-linear

2) Not saturated in the +ve region

3) Computationally inexpensive

4) converge → faster → sigmoid
↘ tanh

Disadvantage

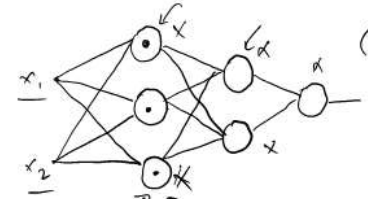
→ Differentiable $\quad x < 0 \rightarrow 0$, $\quad x \geq 0 \rightarrow 1$

→ Non zero centered → sigmoid → Batch Normalization
normalize →

↳ Dying ReLU problem

Dying Relu Problem

08 June 2022 22:53



dead neuron → forever dead (0) → $\boxed{50\% >}$ → model data

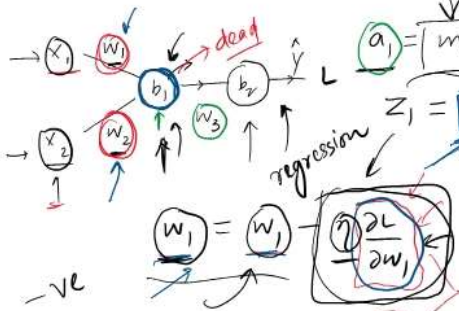
low level represn

$\boxed{\text{ReLu}}$ → vanishing grad

↳ Best choice hidden layers

→ Dying Relu

$\boxed{100\% \text{ dead}}$ worst



$a_1 = \boxed{\max(0, z_1)} = 0$ $\quad \frac{\partial a_1}{\partial z_1} = 0$

$z_1 = \boxed{w_1 x_1 + w_2 x_2 + b_1} < 0$

regression

+ve

$w_1 = w_1 - \boxed{\eta \frac{\partial L}{\partial w_1}}$ −ve

+ve

$w_2 = w_2 - \eta \frac{\partial L}{\partial w_2}$ −ve

$L \rightarrow \hat{y} \rightarrow w_3, b_2$

$a_1 \rightarrow z_1 \rightarrow w_1, w_2, x_1, b_1, x_2$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_1} \frac{\partial a_1}{\partial z_1} \frac{\partial z_1}{\partial w_1}$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_1} \frac{\partial a_1}{\partial z_1} \frac{\partial z_1}{\partial w_2}$$

dying relu

$z_1 = \boxed{w_1 x_1 + w_2 x_2 + b_1} < 0$ (−ve)

$b_1 << 0$ (−ve)

$\boxed{\text{dead}}$

1) High learning rate (η)

2) High −ve bias

(−ve) $b_1 = b_1 - \eta \frac{\partial L}{\partial b_1}$

(-ve) $\frac{\partial L}{\partial b_1}$

$z_1 = \boxed{w_1} x_1 + \boxed{w_2} x_2 + \boxed{b_1}$ $\boxed{<0}$ normalize $\boxed{0-1}$

η high

-ve

$\boxed{50\%}$ $\boxed{\text{dead}}$ → not recover

Solutions

→ Set low learning rate

→ bias → +ve value → $\boxed{0.01}$

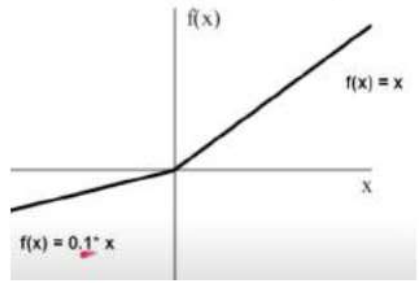
→ Don't use relu → variants (+ve keep) / remove

Variants

- Linear
  - Leaky Relu
  - Parametric Relu
- Non linear
  - Elu
  - Selu

Leaky Relu

08 June 2022 22:59



$f'(z) = \max(\boxed{0.01} z, z)$

$w_1 = w_1 - \eta \frac{\partial L}{\partial w_1}$

$z \geq 0 \rightarrow z$

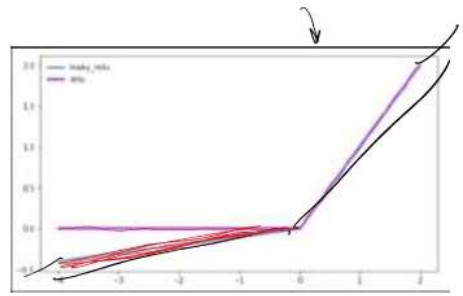
$z < 0 \rightarrow \frac{1}{100} z$ (fraction of z)

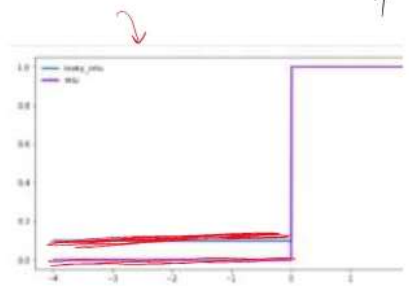
-ve (z) → $\neq 0$ $\boxed{0.01} z$

↳ derivative

$f'(z)$ $z \geq 0 \rightarrow \boxed{1}$

$z < 0 \rightarrow \boxed{0.01}$





Advantages

→ Non-saturated → unbounded

→ Easily computed

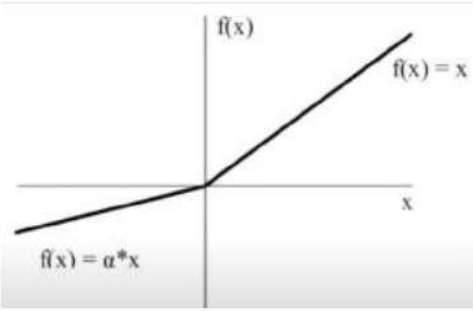
→ No dying relu problem

→ Close to 0 centered (-ve/+ve)

Disadv

Parametric Relu

08 June 2022 22:53



$$f(x) = \begin{cases} x & \text{if } x > 0 \\ ax & \text{otherwise} \end{cases}$$

0.01x

a → trainable parameter

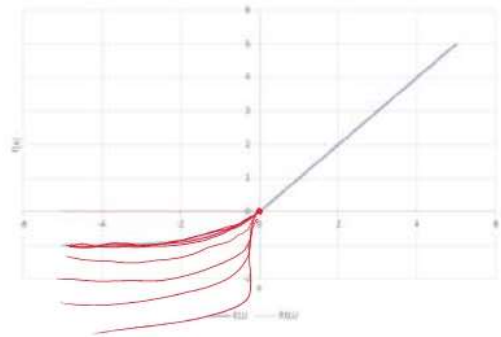
depend on data

flexibility

Elu - Exponential Linear Unit

09 June 2022 00:29

performance better Relu

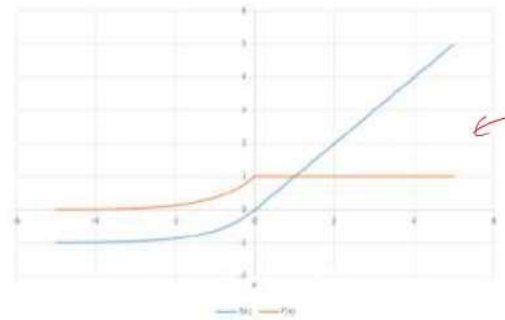


$$\mathrm{ELU}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{if } x < 0 \end{cases}$$

0.1 to 0.3

constant

$$\mathrm{ELU}'(x) = \begin{cases} 1 & \text{if } x > 0 \\ \mathrm{ELU}(x) + \alpha & \text{if } x \leq 0 \end{cases}$$



Advan

→ Close zero centered
↳ convergence faster

→ Better generalized

→ Dying relu

→ Always continous as well diffen

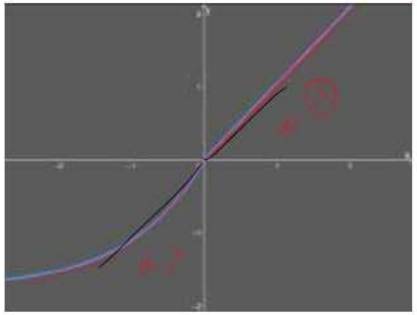
Disadv

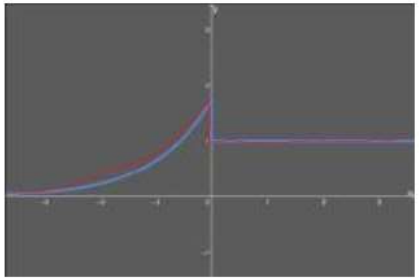
→ computation expensive (ex)

Selu - Scaled Exponential Linear Unit

09 June 2022 00:29

Recent → new → 90+ pages





$$\mathrm{SELU}(x) = \lambda \begin{cases} x & \text{if } x > 0 \\ \alpha e^x - \alpha & \text{if } x \leq 0 \end{cases}$$

$$\alpha \approx 1.6732632423543772848170429916717$$

$$\lambda \approx 1.0507009873554804934193349852946$$

fixed ✓

train but ✗

$$\mathrm{SELU}'(x) = \lambda \begin{cases} 1 & \text{if } x > 0 \\ \alpha e^x & \text{if } x \leq 0 \end{cases}$$

self-normalizing → activation → normalized

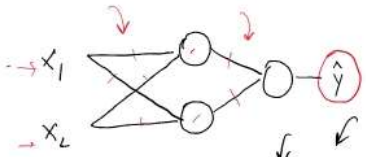
$m = 0 \quad \sigma = 1$

converge faster

Why Weight Initialization is Important?

22 June 2022 12:48

2000's Deep Learning



1. Initialize the parameters
2. Choose an optimization algorithm ✓
3. Repeat these steps:
   1. Forward propagate an input
   2. Compute the cost function
   3. Compute the gradients of the cost with respect to parameters using backpropagation
   4. Update each parameter using the gradients, according to the optimization algorithm

→ Vanishing GP
→ Exploding GP
→ Slow converge

Wrong w.r. techniques → What to do?

Sigmoid Activ

Wrong weight init

What not to do?

22 June 2022 12:48

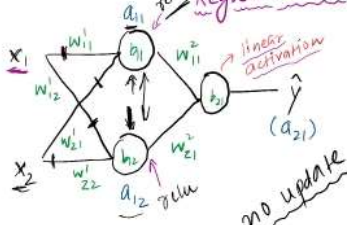
Case1 → Zero Initialization

→ relu
→ tanh
→ sigmoid

$w = 0,\ b = 0$

$a_{11} = \dfrac{e^{z_{11}} - e^{-z_{11}}}{e^{z_{11}} + e^{-z_{11}}} = 0$

Regression Example



$a_{11} = \max(0, z_{11}) = \boxed{0}$

$z_{11} = w^1_{11} x_1 + w^1_{21} x_2 + b_{11} = 0$

$a_{12} = \max(0, z_{12}) = \boxed{0}$

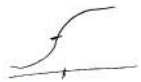
$z_{12} = w^1_{12} x_1 + w^1_{22} x_2 + b_{12} = 0$

$\boxed{a_{11} = a_{12}}$ → equal and 0

no update

$w^1_{11} = w^1_{11} - \eta \boxed{\dfrac{\partial L}{\partial w^1_{11}}}$

$\boxed{w = 0}$ No training will place

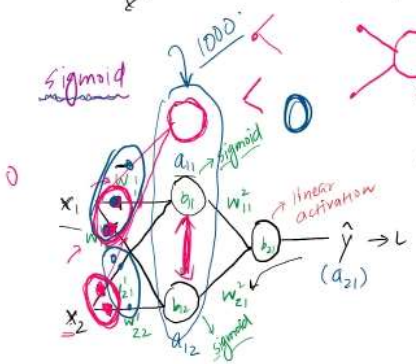


$\boxed{0}$

Non-linear part

$w = w - \eta \boxed{\dfrac{\partial L}{\partial w}}$

sigmoid



$a_{11} = \sigma(z_{11}) = 0.5$

$z_{11} =$

$a_{12} = 0.5$

$a_{11} = a_{12}$

$a_{11} = a_{12}$

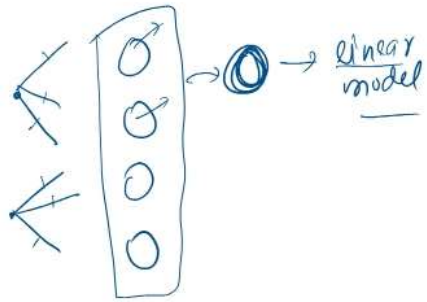
$z_{11} = z_{12}$

$\boxed{\dfrac{\partial L}{\partial w^1_{11}}} = \boxed{\dfrac{\partial L}{\partial \hat{y}} \dfrac{\partial \hat{y}}{\partial a_{11}}} \boxed{\dfrac{\partial a_{11}}{\partial z_{11}}} X_1$

$\boxed{\dfrac{\partial L}{\partial w^1_{12}}} = \boxed{\dfrac{\partial L}{\partial \hat{y}} \dfrac{\partial \hat{y}}{\partial a_{12}}} \boxed{\dfrac{\partial a_{12}}{\partial z_{12}}} X_1$

$\boxed{\dfrac{\partial L}{\partial w^1_{21}}} = \boxed{\dfrac{\partial L}{\partial \hat{y}} \dfrac{\partial \hat{y}}{\partial a_{11}}} \boxed{\dfrac{\partial a_{11}}{\partial z_{11}}} X_2$

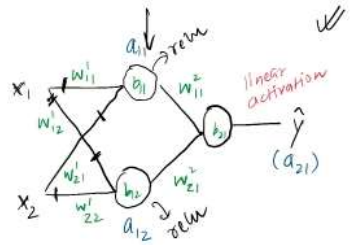
$\boxed{\dfrac{\partial L}{\partial w^1_{22}}} = \boxed{\dfrac{\partial L}{\partial \hat{y}} \dfrac{\partial \hat{y}}{\partial a_{12}}} \boxed{\dfrac{\partial a_{12}}{\partial z_{12}}} X_2$



Case2 → Non-0 constant value

$w = 0.5 \quad b = 0.5$

sigmoid → $\boxed{0}$



$a_{11} = \max(0, z_{11})$ → non-zero

$z_{11} = w^1_{11} x_1 + w^1_{21} x_2 + b_{11} \neq 0$

$a_{12} = \max(0, z_{12}) \neq 0$

$z_{12} = w^1_{12} x_1 + w^1_{22} x_2 + b_{12} \neq 0$

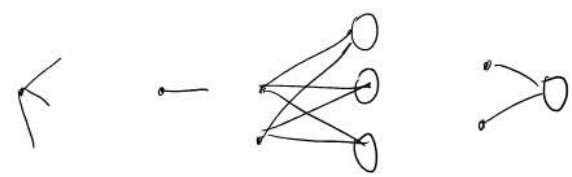
$z_{12} = z_{11}$ $\boxed{a_{11} = a_{12}}$

$\boxed{\dfrac{\partial L}{\partial w^1_{11}}} = \dfrac{\partial L}{\partial \hat{y}} \dfrac{\partial \hat{y}}{\partial a_{11}} \dfrac{\partial a_{11}}{\partial z_{11}} X_1$

$\boxed{\dfrac{\partial L}{\partial w^1_{12}}} = \dfrac{\partial L}{\partial \hat{y}} \dfrac{\partial \hat{y}}{\partial a_{12}} \dfrac{\partial a_{12}}{\partial z_{12}} X_1$

$\dfrac{\partial L}{\partial w^1_{21}} = \dfrac{\partial L}{\partial \hat{y}} \dfrac{\partial \hat{y}}{\partial a_{11}} \dfrac{\partial a_{11}}{\partial z_{11}} X_2$

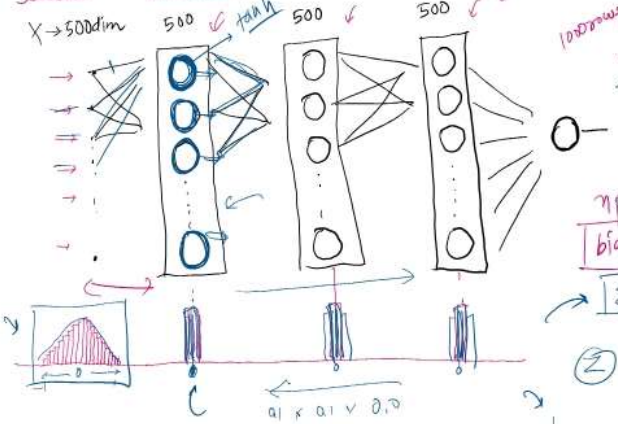
$\dfrac{\partial L}{\partial w^1_{22}} = \dfrac{\partial L}{\partial \hat{y}} \dfrac{\partial \hat{y}}{\partial a_{12}} \dfrac{\partial a_{12}}{\partial z_{12}} X_2$



{ random init }

Case3 – Random Initialization → small / large → tanh / sigmoid → VGP, sigmoid

(1000, 500)



1000 rows → $x_1 \ldots x_{500}$ | y → gaussian distribution

weight init → random → small values

np.random.randn(500, 500) × 0.01

bias = 0

$x_i \to$ 0-1 → −1 to 1

$\sum w_i x_i$ → 0.0007

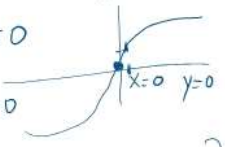
small number tanh

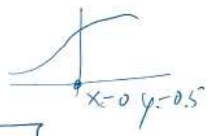
Σ is a small

very close to 0

small small = 0

$w = w - \eta \frac{\partial L}{\partial w} = 0$





- vanishing gradient problem

VGP → deep

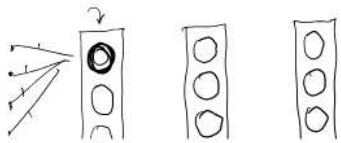
Relu → small values → slow convergence

small weights → 0.0007

tanh → VGP

sigmoid → VGP

Random Initialization (Large values)

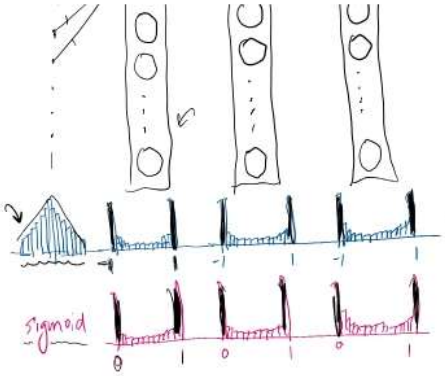


np.random.randn(500, 500) → 0-1

$\sum w_i x_i$ → 500 → 100 → 500

0-1

tanh / sigmoid



0-1

tanh / sigmoid

saturation → g(z) → saturate

$\sum w_i x_i$ → 100

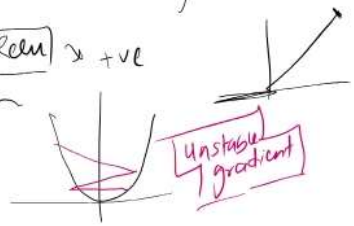
$\sum w_i x_i$ → Relu → 250 → 250 → gradients

slow training

vanishing gradient problem

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

Relu → +ve



unstable gradient

→ zero ×

→ non-zero same ×

→ small random ✓

→ large random ×

Key Insights

22 June 2022 12:49

Things not to do—

1) Zero initialization → no training

2) Non-zero constant initialization → linear

3) Random initialization with small weights → vanish

4) Random initialization with large weights → exploding

random

np.random.randn(shape)*0.01

(2,2)

$\boxed{-0.01 \text{ to } 0.01}$

Vanishing grad

Slow converges

np.random.randn(2,2)

$\boxed{-3 \text{ to } 3}$

What can be done

→ Heuristics (Jugad) → practical solⁿ

→ Xavier/Glorot Init (tanh, sigmoid) → Normal / Uniform

→ He Init (relu) → Normal / Uniform

Keras

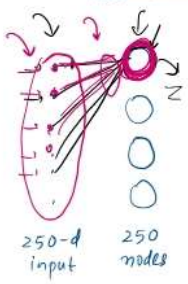
formulas

-1 to 1

0-1

-0.01 to 0.01

Intuition



250-d input

250 nodes

$z = \sum w_i x_i$

$w_i \downarrow \uparrow$

small ←

np.random.randn(250,250) * $\boxed{0.01}$

$\boxed{250}$

-250 to 250

large

np.random.randn(250,250) * $\boxed{1}$ → -3 to 3

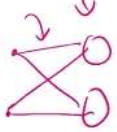
Variance $= \boxed{\frac{1}{n}} = \frac{1}{250}$ * $\sqrt{\frac{1}{250}} \rightarrow \boxed{sd}$

$n \rightarrow \uparrow \downarrow \boxed{\frac{1}{n}} \downarrow \uparrow w$

Xavier init (Normal)

$\sqrt{\frac{1}{fan_in}}$

# of inputs coming to the node



np.random.randn(2,2) * $\sqrt{\frac{2}{2}}$

$\sqrt{\frac{2}{fan_in + fan_out}}$

He Normal

$\boxed{relu}$

$\boxed{\sqrt{\frac{2}{fan_in}}}$

Uniform Distribution

Xavier Uniform

[-limit, limit]

$limit = \sqrt{\frac{6}{(fan_in + fan_out)}}$

He Uniform

[-limit, limit]

$limit = \sqrt{\frac{6}{fan_in}}$

Keras

Weight Initialization Techniques

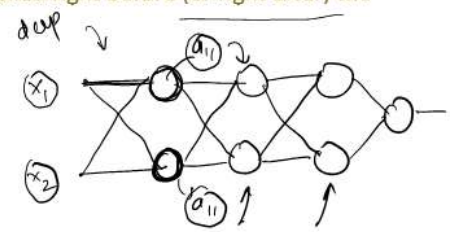
22 June 2022 12:49

What is Batch Norm?

27 June 2022 11:00

Batch-Normalization (BN) is an algorithmic method which makes the training of Deep Neural Networks (DNN) faster and more stable.

It consists of normalizing activation vectors from hidden layers using the mean and variance of the current batch. This normalization step is applied right before (or right after) the nonlinear function.



normalized

$m=0$
$sd=1$

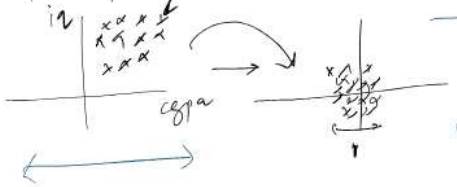
normalize

$m=0$ $sd=1$

Why use Batch Norm?

30 June 2022 16:08

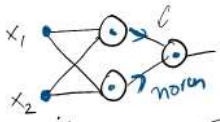
cgpa	iq	placed
7	70	1
8	80	0
9	90	1
6	60	0



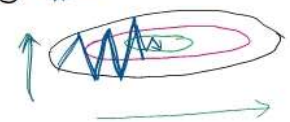
normalize

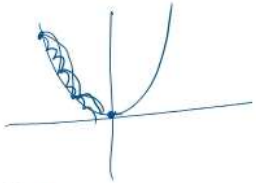
$\mu = 0$

$\sigma = 1$



lr slow





{training slow}

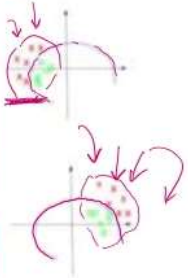
training faster stable

Internal Covariate Shift

27 June 2022 11:02

Covarite Shift

$X \to Y$



$X \to Y$

input cols distrib

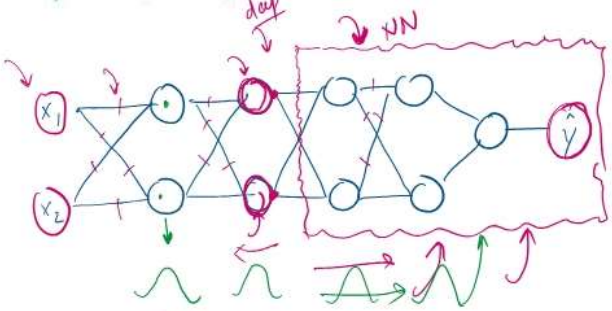
learning rate

low

parameter init

The authors' precise definition is:

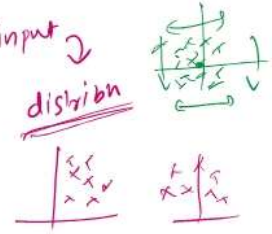
> We define Internal Covariate Shift as the change in the distribution of network activations due to the change in network parameters during training.



$\mu = 0$

$\sigma = 1$

input distribn



Batch Norm - The How

27 June 2022 11:02

→ Mini-Batch GD

→ Layer by layer

cgpa	iq	placed
8.9	100	1
6.2	89	0
9.1	91	0
7.7	76	1
⋮	⋮	⋮
6.7	91	0

batch_size : 4

$(4,2) * (2,2)$

$(4,2) + (1,2)$

$Z_{11} = w_1 \, cgpa + w_2 \, iq + b$

$g(Z_{11}) = \boxed{a_{11}}$

(1) $Z_{11} \rightarrow Z_{11}^N \rightarrow g(Z_{11}^N) = a_{11}$

(2) $Z_{11} \rightarrow g(Z_{11}) = a_{11} \rightarrow a_{11}^N$

$$\boxed{\frac{Z_{11} - \mu}{\sigma}} = Z_{11}^N \quad (4,2)$$

$$\mu_B = \frac{1}{m} \sum_{i=1}^{m} z_{11}^i \qquad \boxed{m = 4}$$

$$\boxed{z_{11}^i = \frac{z_{11}^i - \mu_B}{\sigma_B + \epsilon}}$$

$$\sigma_B = \sqrt{\frac{1}{m} \sum_{i=1}^{m} (z_{11}^i - \mu_B)}$$

$\boxed{0-1}$ $\mu = 0$ $\boxed{\sigma = 1}$

learnable parameters w, b

$\boxed{\gamma = 1 \quad \beta = 0}$

$$\boxed{Z_{11}^{BN} = \gamma Z_{11}^N + \beta}$$

$\mu = 0, \sigma = 1$

$Z_{11} \rightarrow Z_{11}^N \rightarrow Z_{11}^{BN} \rightarrow g(Z_{11}^{BN}) = a_{11}$

$Z_{11} \rightarrow$ normalize

$\mu = 0 \quad \sigma = 1$

$\gamma = \sigma + \epsilon \qquad \beta = \mu$

flexibility

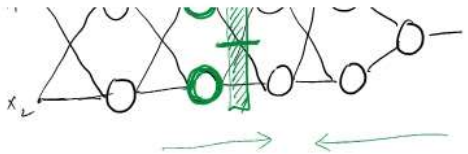
$Z_{11} \rightarrow Z_{11}^N \rightarrow Z_{11}^{BN}$

Batch Norm → Layer

4 learnable $\gamma \beta$ $\gamma \beta$

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

Keras



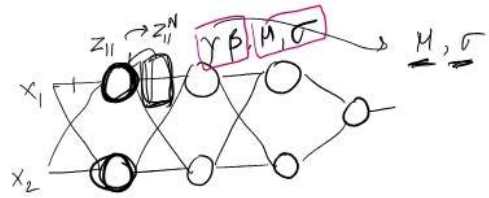
Batch Norm during test

27 June 2022 11:03

cgpa	iq	placed
8	80	1
7	70	0
6	60	1
⋮	⋮	⋮
9	90	1

100 students

$\boxed{b=4}$



testing
1 query point

Exp Weighted moving avg.

$\boxed{\text{EWMA}}$

$\boxed{25 \text{ times}}$

1 batch

2

$\mu_{B1}, \mu_{B2} \ldots \mu_{B25}$ μ

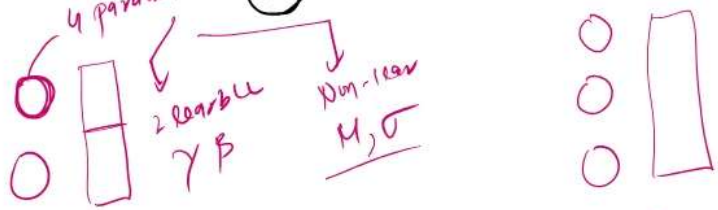
ex EWMA

σ

4 parameters

2 learable $\gamma\ \beta$

Non-learn μ, σ



$3 \times 4 = 12$

6 6

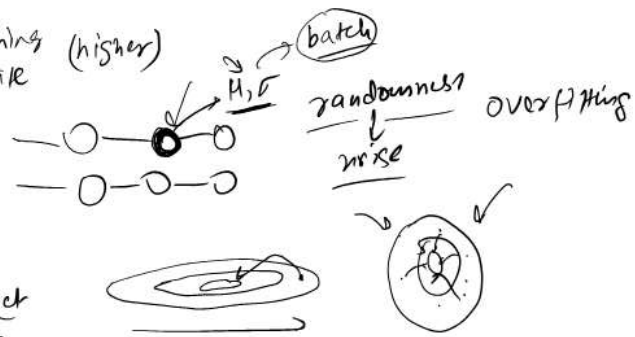
Advantages

27 June 2022 11:02

1) Stable → hyperpara → wider range of values

2) faster → learning rate (higher)

3) Regularizer → dropout



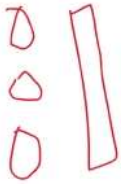
4) Weight init impact reduce

Keras Implementation

27 June 2022 11:03

```
model = Sequential()

model.add(Dense(3,activation='relu',input_dim=2))
model.add(BatchNormalization())
model.add(Dense(2,activation='relu'))
model.add(BatchNormalization())
model.add(Dense(1,activation='sigmoid'))
```

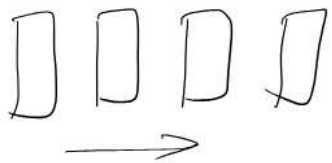


Introduction

05 July 2022 09:57

Performance of NN

how the speed the training



Optimizers

→ Weigh init

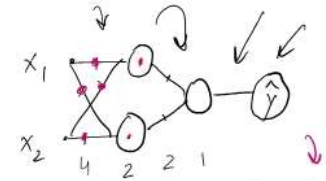
→ Batch Norm

→ Activation function

Role of Optimizer

05 July 2022 10:01

cgpa	iq	placed
8	80	1
9	90	1
7	70	0



9 parameters

2 weights

3D

$L(\ ,\ ,\ ,\ ,\ ,\ )$

C/L()

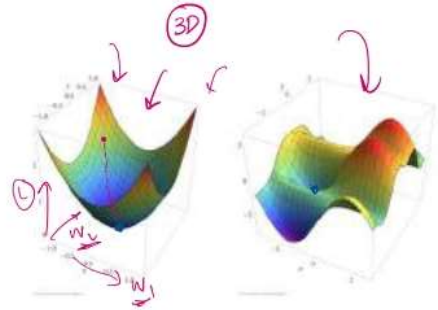
9 para

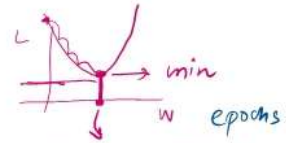
10D

optimization

2 min

$y \quad \hat{y} \quad (w, b)$





Gradient Descent

$$w_n = w_0 - \eta \frac{\partial L}{\partial w}$$

Types of Optimizers

05 July 2022 10:02

- → Batch GD
- → Stochastic GD
- → Mini batch GD

(10)

epochs = (10)

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

500 rows → pred

↓

loss → weight update

$\boxed{10 \times 500}$ → 5000

batch size = 100

5 batch → 10 × 5 times

Challenges

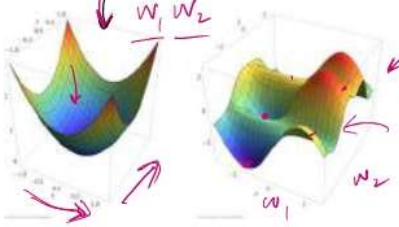
05 July 2022 10:02

1) learning rate

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

2) learning rate scheduling → pre define

3)



(η) same both the directions

complex

sub optimum

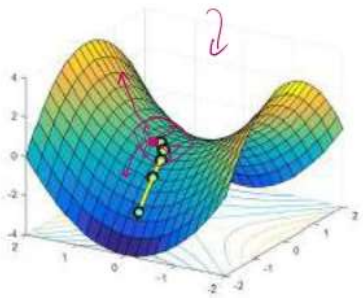
(9) weights and biases 9 sep learn

10-D → 10 directions

minima

4) local minima

5) saddle point



$$\frac{\partial L}{\partial W} = 0$$

$$W_n = W_0 - \eta \boxed{\frac{\partial L}{\partial W}}$$

What next?

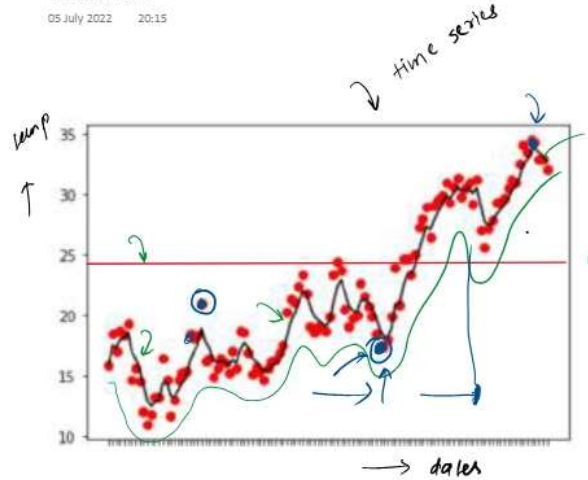
05 July 2022 10:02

1) Momentum
2) Adagrad
3) NAG
4) RMSprop
5) Adam

→ Exponentially Weighted Moving Average

The What

05 July 2022 20:15



→ time series
→ financial
→ signal process
→ Deep learning optimizers

D1
D2
D3
D4
D5

Mathematical Formulation

05 July 2022 20:16

$$\boxed{V_t = \beta V_{t-1} + (1-\beta)\theta_t}$$

$\beta \rightarrow 0 < \beta < 1$

$\boxed{\beta = 0.9}$

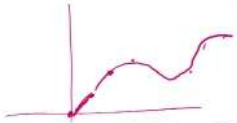
index	temp (θ)
D1	$\boxed{25}$
D2	13
D3	17
D4	31
D5	43

$V_0 = \boxed{0}$ $\quad V_0 = \theta_0$

$V_1 = 0.9 \times V_0 + 0.1 \times 13$

$= 1.3$

$V_2 = 0.9 \times 1.3 + 0.1 \times 17 =$



$\beta = 0.5$

$\frac{1}{1-0.5} = \frac{1}{0.5}$

2 days

$\beta \rightarrow \boxed{\frac{1}{(1-\beta)}} \rightarrow \beta = 0.9$

$\frac{1}{1-0.9} = \frac{1}{0.1} = 10$

$V_t = \beta V_{t-1} + (1-\beta)\theta_t$

$V_0 = 0$

$V_1 = (1-\beta)\theta_1$

$V_2 = \beta V_1 + (1-\beta)\theta_2$

$= \beta(1-\beta)\theta_1 + (1-\beta)\theta_2$

$V_3 = \beta V_2 + (1-\beta)\theta_3$

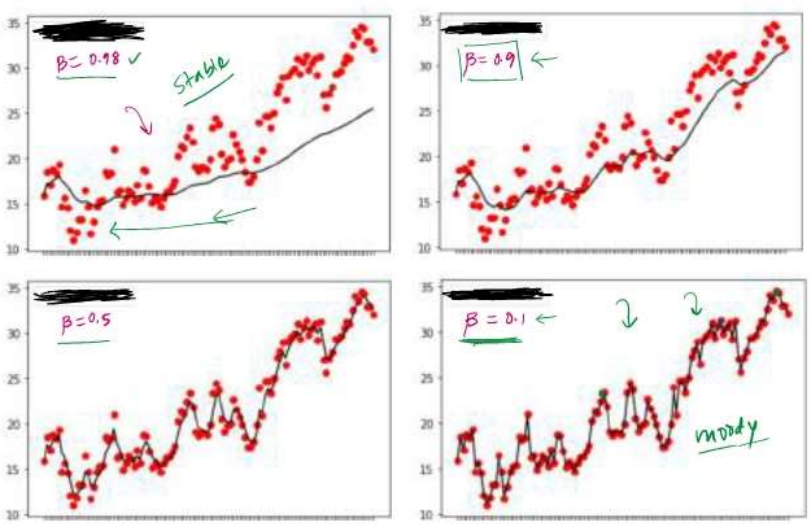
$= \beta^2(1-\beta)\theta_1 + \beta(1-\beta)\theta_2 + (1-\beta)\theta_3$

$V_4 = \beta^3(1-\beta)\theta_1 + \beta^2(1-\beta)\theta_2 + \beta(1-\beta)\theta_3 + (1-\beta)\theta_4$

$= (1-\beta)[\boxed{\beta^3\theta_1} + \beta^2\theta_2 + \beta\theta_3 + \theta_4]$

$\beta^3 < \beta^2 < \beta \longrightarrow$ $\quad 0 < \beta < 1$

Effect of β



Mathematical Intuition

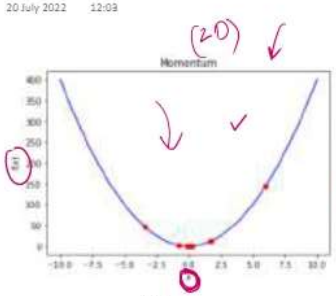
$V_t = \beta V_{t-1} + (1-\beta)\theta_t$

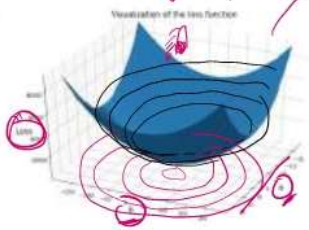
$V_4 = \beta V_3 + (1-\beta)\theta_4$

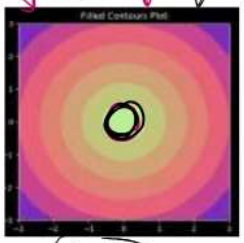
$V_4 = \beta^3(1-\beta)\theta_1 + \beta^2(1-\beta)\theta_2 + \beta(1-\beta)\theta_3 + (1-\beta)\theta_4$

Understanding Graphs

20 July 2022 12:03







(2D)

3para → 3+1 → 4D

2D proje

2D

3D

Contour

1S 1O 2S O

x_1, x_2, x_3 → $\hat{y}$

Loss function → $f(x)$ → $\hat{y} / y$ → $(y-\hat{y})^2$

(L) → ↑

$L = f(w_1, w_2 \dots b)$ → 3D

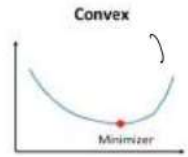
$y = f(x)$

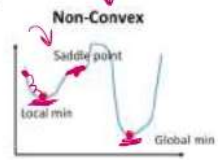
x_1 — w, b — $\hat{y}$ $L = f(w)$

$L = f(w, b)$

Convex Vs Non-Convex Optimization

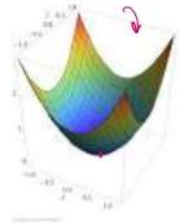
20 July 2022 13:06

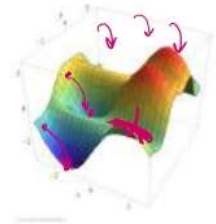




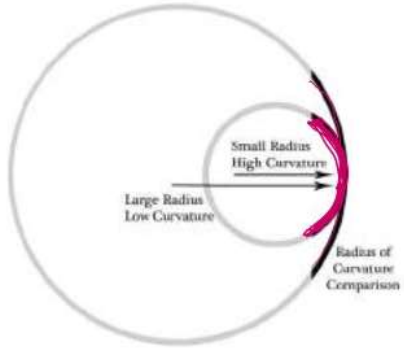
L (w)

minima



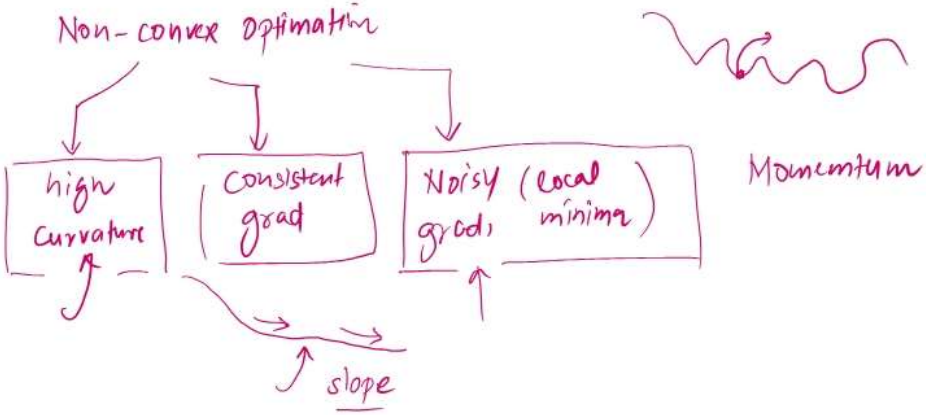


1) local minima
2) Saddle point
3) High curvature



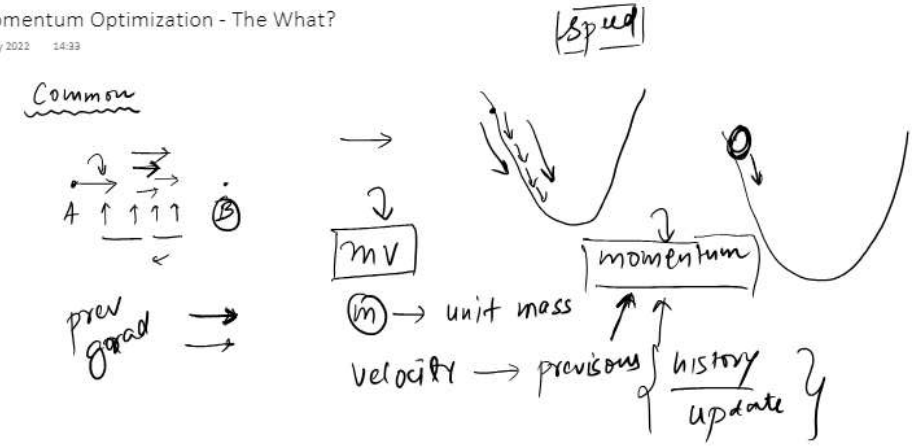
Momentum Optimization - The Why?

20 July 2022 14:28



Momentum Optimization - The What?

20 July 2022 14:33



Momentum Optimization - Mathematics(The How?)

20 July 2022 14:56

$$w_{t+1} = w_t - \boxed{\eta \nabla w_t}$$

$v \rightarrow$ velocity

$$w_{t+1} = w_t - v_t$$

$$v_t = \boxed{\beta * v_{t-1}} + \eta \nabla w_t$$

acceleration

momentum

$\boxed{0.9}$

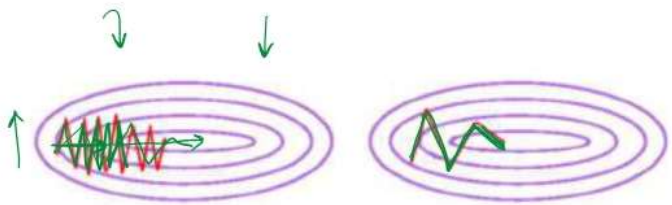
history of past velo-

$\boxed{\text{use}}$

$\boxed{0 < \beta < 1}$

prev velocity

$\boxed{v_{t-2} \quad v_{t-3}}$



SGD

SGD momentum

faster convergence

Effect of beta

20 July 2022 15:09

$$w_{t+1} = w_t - \eta \nabla w_t$$

$$w_{t+1} = w_t - v_t$$

$$v_t = \beta v_{t-1} + \boxed{\eta \nabla w_t}$$

$\boxed{\text{SGD}}$

$\beta = 0$ momentum $\rightarrow$ SGD

$\boxed{\beta = 0}$

decaying factor

$\frac{1}{1-0.9} = \frac{1}{0.1} = 10$

$\frac{1}{1-\beta}$

0.9

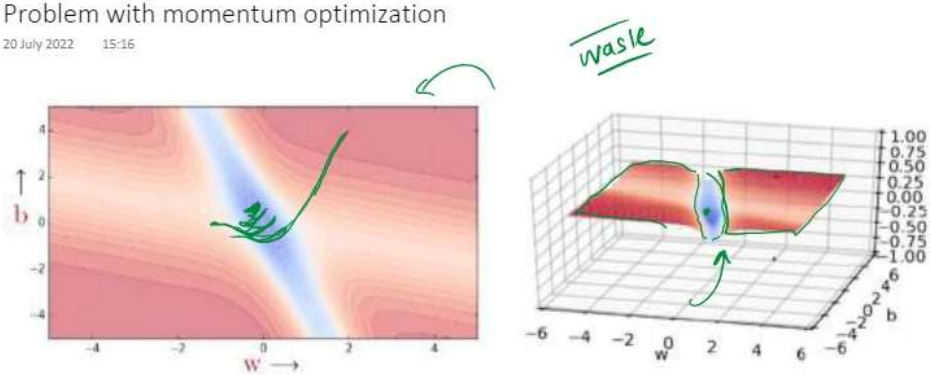
current avg past 10 veloci

up1 up2 . - $\boxed{up3}$

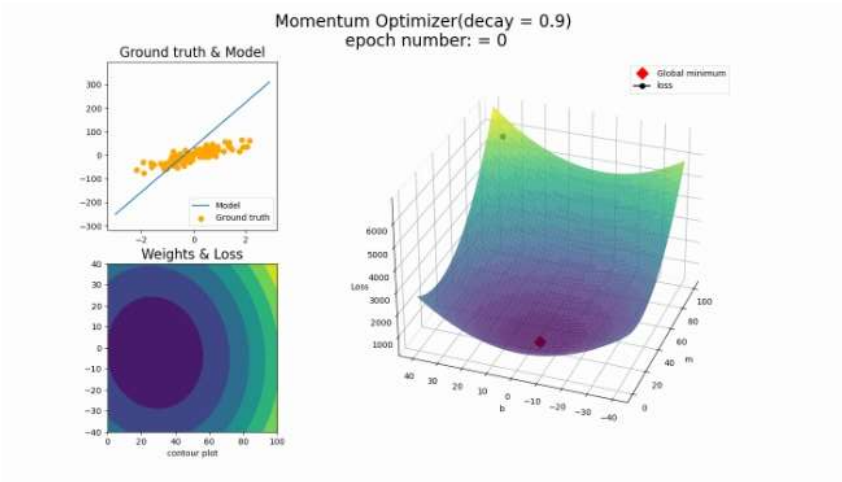
$\boxed{\beta = 1}$

Problem with momentum optimization

20 July 2022 15:16



decay 0.9



Demo

24 July 2022 13:17

Mathematical Intuition

24 July 2022 13:17

$$w_{t+1} = w_t - (\beta v_{t-1} + \eta \nabla w_t)$$

past velocity

gradient at that point

Momentum

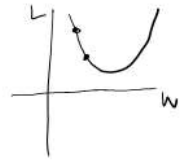
$$\boxed{w_{t+1} = w_t - v_t}$$

where

$$\boxed{v_t = \beta v_{t-1} + \eta \nabla w_t}$$

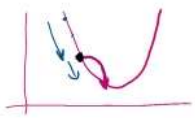
$\beta \rightarrow$ decay factor

$$w_{t+1} = w_t - \boxed{\eta \nabla w_t}$$



NAG

Nesterov Accelerated Gradient better



history of velocity + gradient at that pt

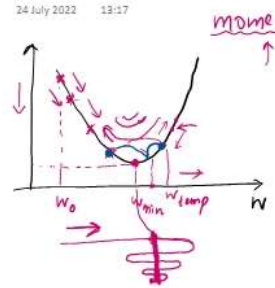
look ahead

$$\begin{cases} w_{la} = w_t - \beta v_{t-1} \\ v_t = \beta v_{t-1} + \eta \nabla w_{la} \\ w_{t+1} = w_t - v_t \end{cases}$$

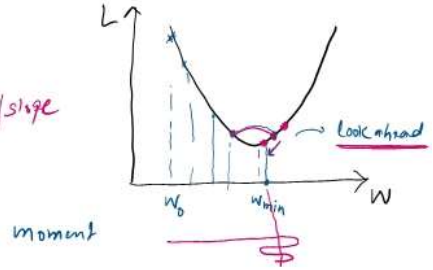
$$\rightarrow v_t = (w_t - w_{la}) + \eta \nabla w_{la}$$

Geometric Intuition

24 July 2022 13:17

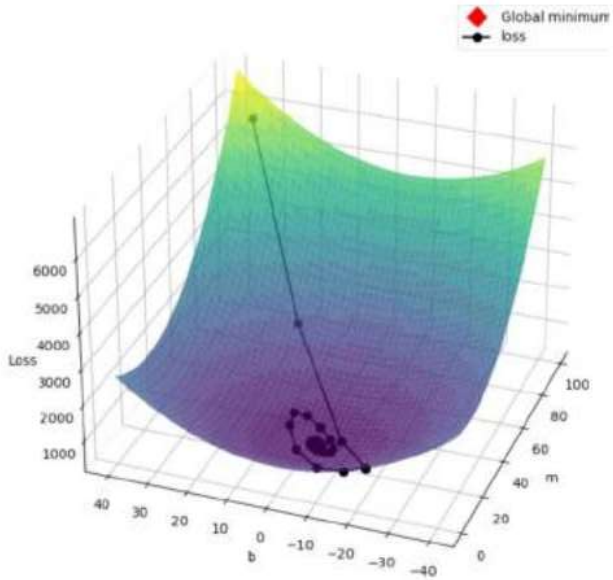


−ve of grad / slope



Disadvantage

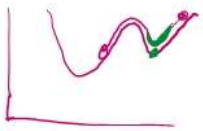
24 July 2022 13:18



Disadvantage

NAG → oscillation
↓
dampen

local minima



Keras Code

24 July 2022 13:18

```
tf.keras.optimizers.SGD(
    learning_rate=0.01, momentum=0.0, nesterov=False, name="SGD", **kwargs
)
```

SGD ✓

Momentum
momentum = 0.9
nesterov = False ✓

NAG
momentum = 0.9
nesterov = True ✓

AdaGrad Intro

02 August 2022 18:45

Adaptive Gradient → learning rate fix X

change

adapt →

cgpa | salary | bank loan

0-10 0-Lakh

Normalize

→ input features → scale

different

⇒ features → [sparse] → 0 (most)

↳ Adagrad

iq | cgpa | [from iit] | package

0 → 0,0,0,

1 →

sparse

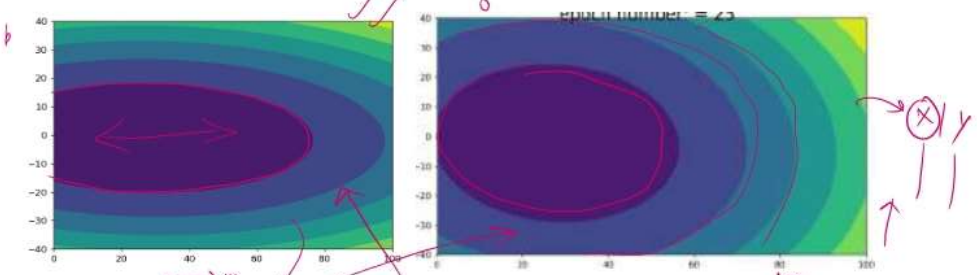
X | Y

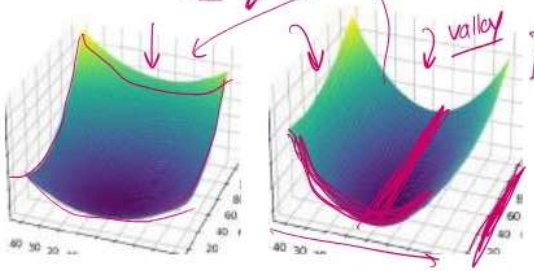
0

0

0

Elongated bowl problem





valley

Bgd → momentum → NAG

Adagrad

loss → wrt → w, b

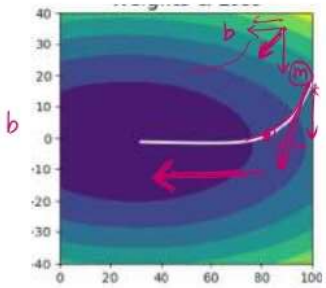
elongated

slope → 1axis – along

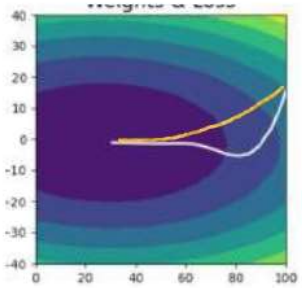
2nd axis constant

How optimizers behave(Why?)

02 August 2022 18:44



Batch GD



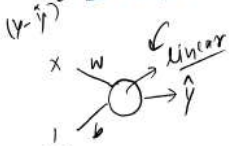
Momentum

b	m X	Y
	0.72	0.3
	0.81	—
	0	—
	0	—
	0	—
	0.2	—

less movem

normal movement

$(y-\hat{y})^2$



BGD → sparspe

for i in epochs:

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

small in every epoch

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

→ big update in every epoch

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

$$= -2(y-\hat{y})\boxed{X} \text{ add}$$

$$\frac{\partial L}{\partial b} = \boxed{-2(y-\hat{y}) * 1}$$

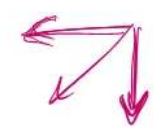
bias num

[100 rows] 0 X sparse

small number

sparse col → small

non sparse → normal/big



Adagrad mathematics + Intuition

02 August 2022 18:44

$$\boxed{w = w - \eta \frac{\partial L}{\partial w}}$$ update

$$b = b - \boxed{\eta \frac{\partial L}{\partial b}}$$

Adagrad

optimization

lr → same

big lr → small

update - small

Adagrad → diff learning rates

past gradients

sum (squared)

lr → small → gradient → big

epoch

$$v_t = v_{t-1} + (\nabla w_t)^2$$

$$\boxed{\frac{\partial L}{\partial w}}$$

v_t $\left(\frac{\partial L}{\partial w}\right)^2 \left(\frac{\partial L}{\partial b}\right)^2$

$$\boxed{w_{t+1} = w_t - \eta \nabla w_t}$$

η small $w_{t+1} = w_t - \eta \Delta w_t$

$$\boxed{\sqrt{v_t + \epsilon}}$$

η ϵ = small num

$\begin{pmatrix} -ve \\ +ve \end{pmatrix}$

$v_t = 0$

sparse data

Adagrad Demo

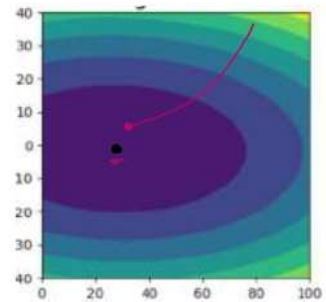
02 August 2022 18:44

Keras Implementation

02 August 2022 18:44

Disadvantage

02 August 2022 18:43



NN → use X → linear regress

$\frac{\eta}{\sqrt{v_t}}$ → small → decrease

v_t

past updates → gradient

epochs

update small

global minima

→ RMSprop

→ Adam

The Why?

03 August 2022 14:03

RMSprop → Root Mean square Prop

↳ improvement

↓

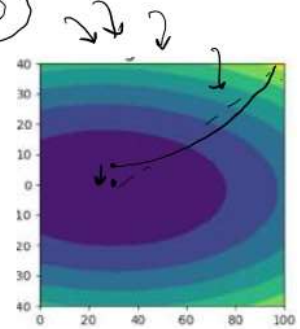
$\boxed{\text{Adagrad}}$ ← sparse data

BAD → momentum

x_1	x_2	y
0		
0		
0		
1		
0		

Disadvantage → learning rate ↓

Update → small



Mathematical Formulation

03 August 2022 14:04

epochs

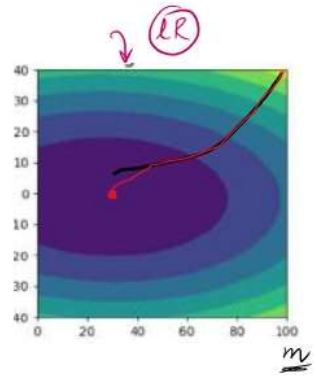
Adagrad

$$\boxed{v_t} = v_{t-1} + (\nabla w_t)^2$$

gradients

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{v_t + \epsilon}} \nabla w_t$$

(b) small



b	x	m	y
	0		
	0		
	0		
	1		
	0		

↓ b move

non-sparse

RMSprop

$$\boxed{v_t} = \beta v_{t-1} + (1-\beta)(\nabla w_t)^2$$

exp decaying avg

$\boxed{\beta = 0.95}$

Old epoch

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{v_t + \epsilon}} \nabla w_t$$

→ $v_0 = 0$

→ $v_1 = 0.95 \times 0 + 0.05(\nabla w_1)^2$

→ $v_2 = 0.95 \times 0.05 (\nabla w_1)^2 + 0.05 (\nabla w_2)^2$

→ $v_3 = \underbrace{0.95 \times 0.95 \times 0.05}_{\text{epoch 1}} (\nabla w_1)^2 + \underbrace{0.95 \times 0.05 (\nabla w_2)^2}_{\text{epoch 2}} + \underbrace{0.05 (\nabla w_3)^2}_{\text{epoch 3}}$

smaller ↑ epoch 1

largest

v_t shoot → η small

Adagrad → NN → non-convex optimization

↳ convex optimis → linear reg

convex optimiz → linear reg

global minima

RMSprop

Disadvantage

03 August 2022 14:54

No, optimization

Adam

Introduction

04 August 2022 10:45

$\boxed{\text{Adam}}$ → - Adaptive Moment Estimation

most powerful → ANN, CNN, RNN

SGD / BGD / MBGD
- → Momentum → NAG
- → Adagrad
- → RMS Prop

→ 1) momentum
→ 2) learning decay

→ merge

Mathematical Formulation

04 August 2022 10:45

$$w_{t+1} = w_t - \frac{\boxed{\eta}}{\sqrt{v_t + \epsilon}} * m_t$$

Keras → $\boxed{\text{Bias correction}}$ epoch #

$$\rightarrow \hat{m}_t = \frac{m_t}{1-\beta_1^{\boxed{t}}} \rightarrow \hat{v}_t = \frac{v_t}{1-\beta_2^{\boxed{t}}}$$

$\beta_1 = 0.9$ $\beta_2 = 0.99$ → Keras

where

$$\boxed{m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla w_t}$$ → momentum

$$\boxed{v_t = \beta_2 v_{t-1} + (1-\beta_2)(\nabla w_t)^2}$$ → Adagrad

$\eta = 0.1$, 0.01, $0 \rightarrow$

$m_0 = 0$ $m_t = 0$ $v_t = 0$

$v_0 = 0$

Demo

04 August 2022 10:45

Verdict

04 August 2022 10:45

Adam

↓

RMSProp → momentum

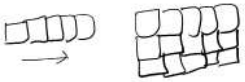
hyperparameter

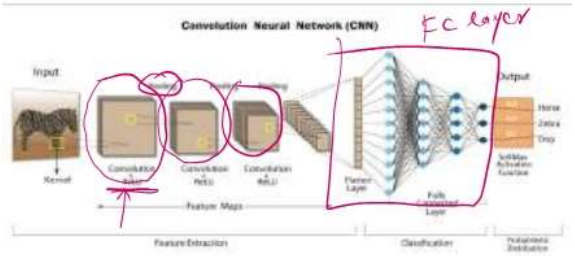
What is a CNN?

17 August 2022 06:47

Convolutional neural networks, also known as convnet, or CNNs, are a special kind of neural network for processing data that has a known grid-like topology like time series data(1D) or images(2D).

→ Convolution
→ Pooling
→ FC layer – ANN





→ ANN → Convolution
↓
CNN convolution ⊗
ANN → matrix mult.

Inspirations
↳ visual cortex
↑

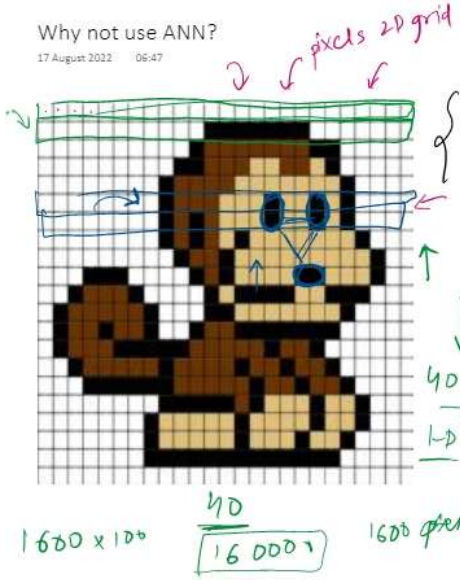
1998 Yann Lecun → AT&T →
↳ Microsoft → OCR hand writing reog } CNN

{ facial reog
self driving } CNN RNN

Why not use ANN?

17 August 2022 06:47

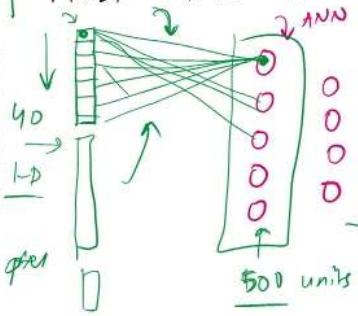
CNN >> ANN



1. High Computation Cost ✓ 1D
2. Overfitting →
3. Loss of imp info like spatial arrangement of pixels

40 X 40 image

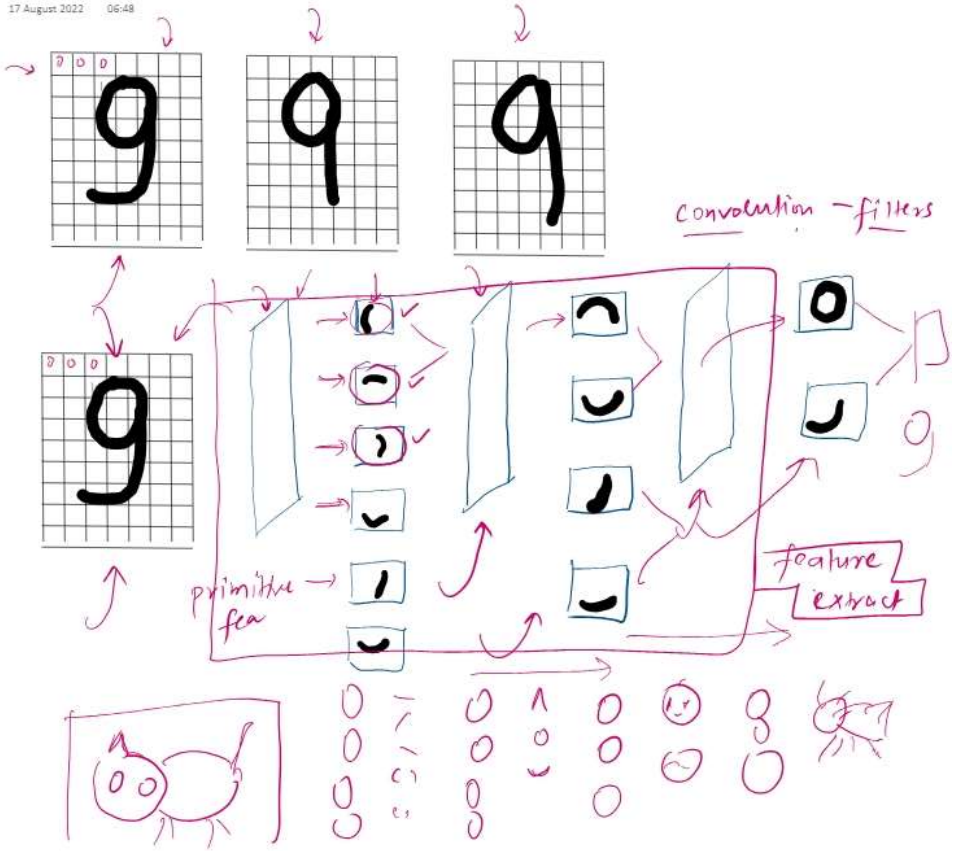
MNIST → ANN → 98%



ANN
1000 x 1000
Overfitting
1000 x 1000 x 500

CNN Intuition

17 August 2022 06:48



CNN Applications

17 August 2022 06:48

Roadmap

17 August 2022 06:48

→ Biological connects

- → Convolution
  - ↳ Padding + Stride
- → Pooling

→ CNN Architecture

↳ CNN Vs ANN

→ Dog Vs Cat Classifier

↳ Data Augment

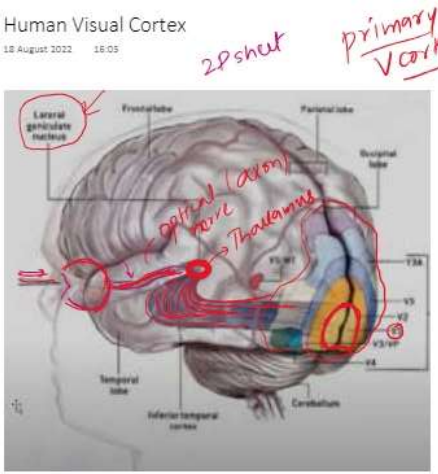
↳ Popular CNN Arch

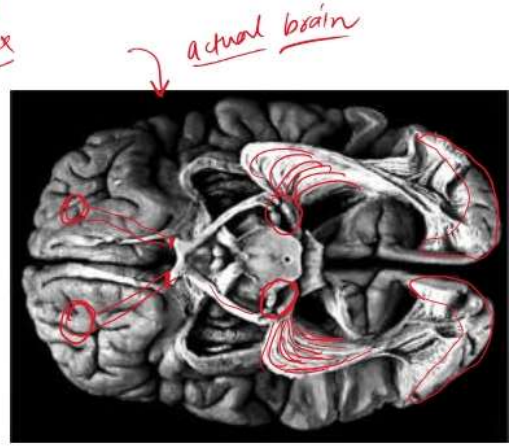
- → LeNET
- → AlexNET
- → VGGNet
- → ResNet
- → Xception

→ Transfer Learning

Human Visual Cortex

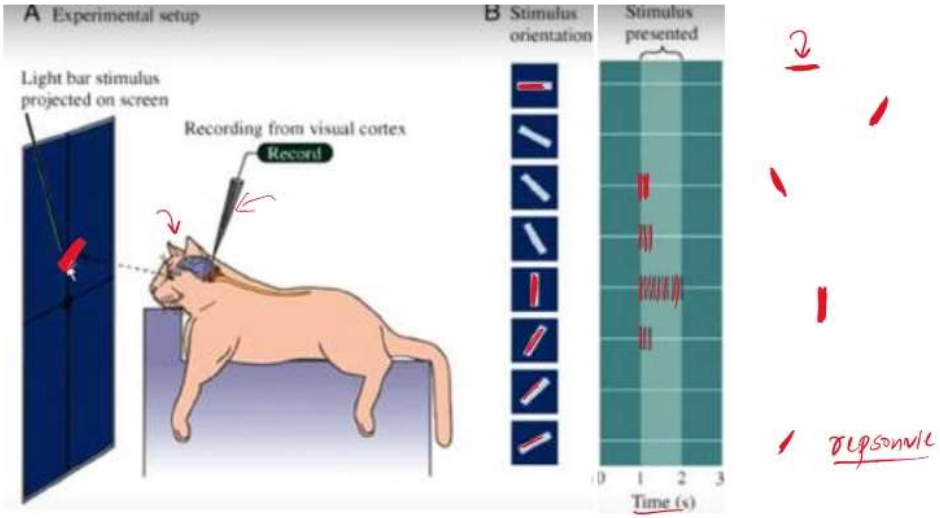
18 August 2022 16:05

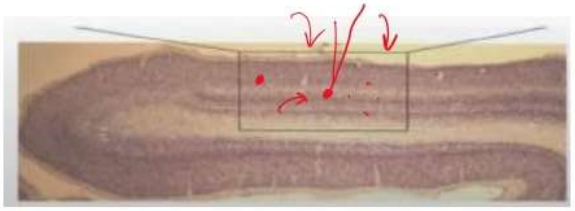




The Experiment

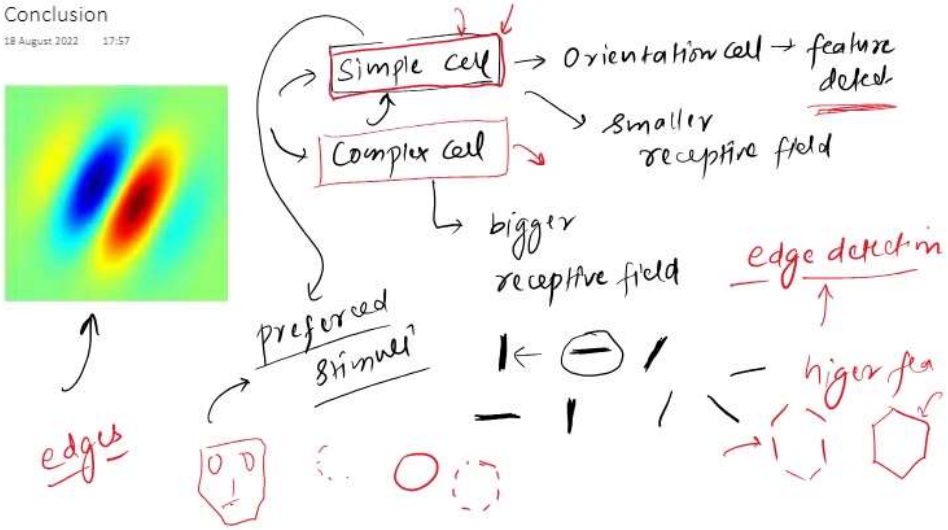
18 August 2022 16:09





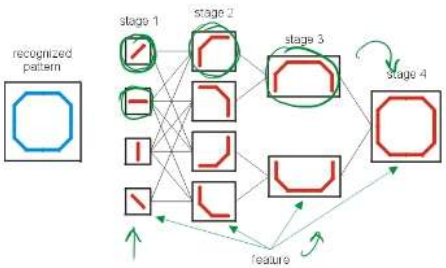
Conclusion

18 August 2022 17:57

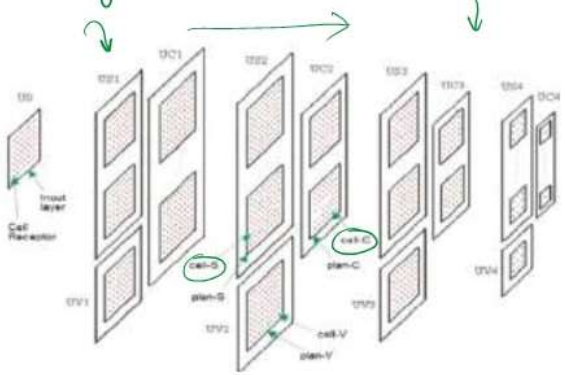


Development

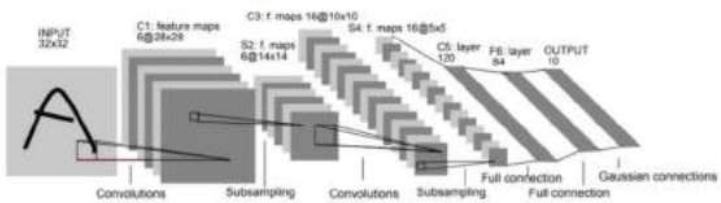
18 August 2022 16:15



Neocognitron → Fukushima



Yann Lecun → CNN → Backprop convolution



early CNN

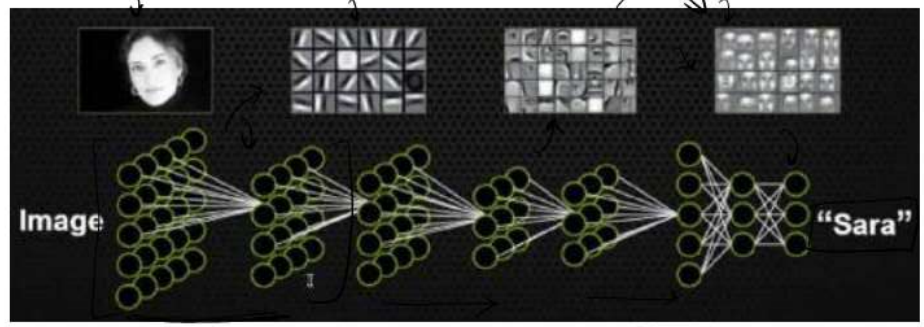
2012 → AlexNET → ImageNET → CNNs

Introduction

19 August 2022 16:45

edges

images → Visual Cortex



CNN → NN special → convolution, Pooling, FC

CNN

Basics of Images

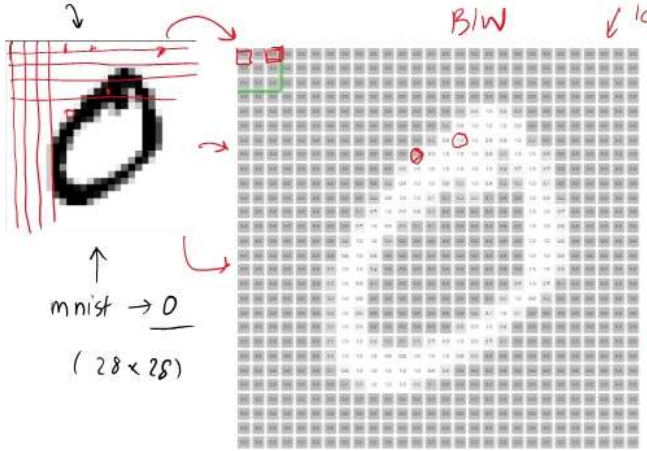
19 August 2022 16:53

0 - 255 → 0-1

B/W

1 channel - greyscale - B/W

RGB → colored imgs



28 x 28 pixels

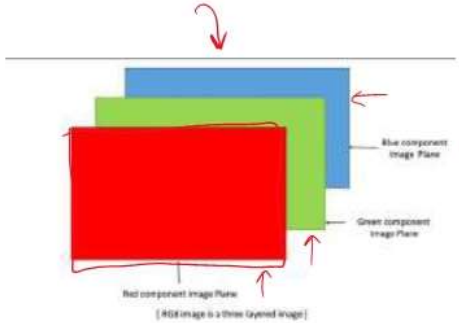
2D arrays →

2D numpy arrays

mnist → 0

(28 x 28)

3 channels → Red Green Blue (primary)

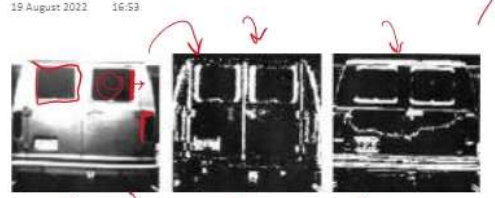


228 x 228 x 3

3channels

Edge Detection (Convolution Operation)

19 August 2022 16:53



255x1 + 255x1 + 255x1

horizontal edge decho

film

-1x0 + -1x0 + -1x0

image

0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
255	255	255	255	255	255
255	255	255	255	255	255
255	255	255	255	255	255

↑ 6x6

*

-1	-1	-1
0	0	0
1	1	1

ANN weight

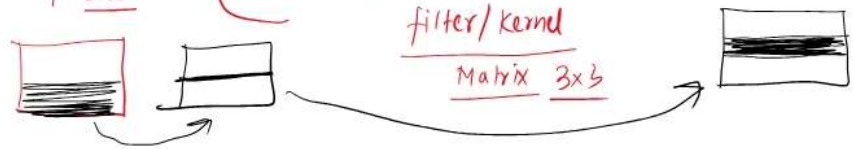
=

0	0	0	0
255	255	255	255
255	255	255	255
0	0	0	0

→ feature map

filter/kernel

Matrix 3x3



0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
255	255	255	255	255	255
255	255	255	255	255	255
255	255	255	255	255	255

(6 x 6)

*

-1	-1	-1
0	0	0
1	1	1

(3x3)

=

(4x4)

(28x28) → (26x26) (3x3) → ? (26 x 26)

n x n m x m → (n - m + 1) x (n - m + 1)

(64 x 64) (3x3) → (62 x 62)

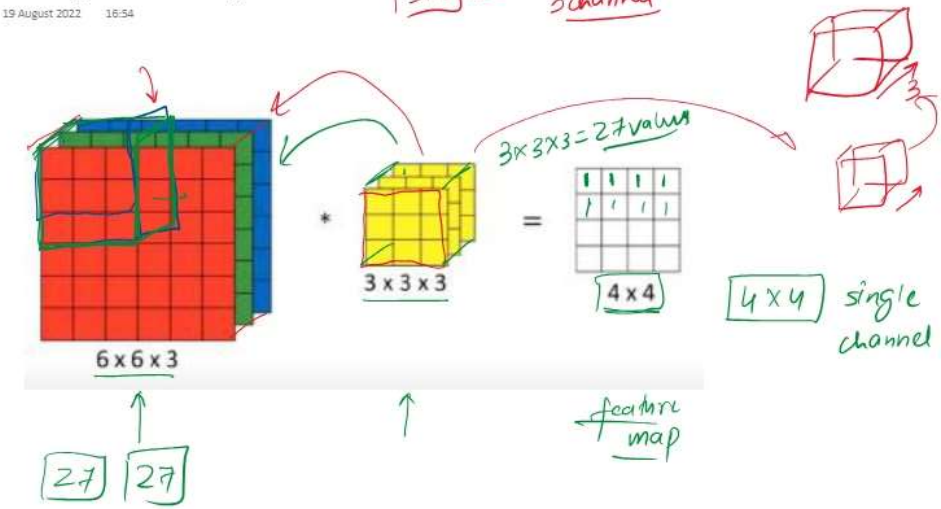
Demo

19 August 2022 16:54

Working with RGB Images

19 August 2022 16:54

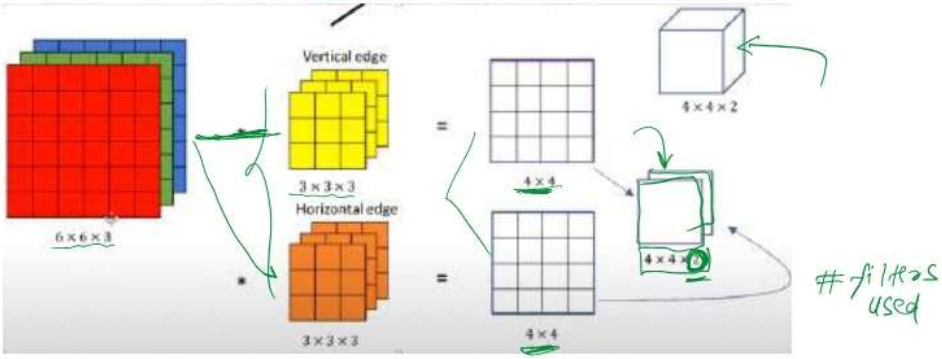
$\boxed{3\times 3}\times 3$ 3 channel



$\boxed{m\times m\times c}$ $\boxed{n\times n\times c}$ $\longrightarrow$ $\underbrace{(m-n+1)\ (m-n+1)}_{\text{single channel}}$

Multiple Filters

23 August 2022 08:24



filters used

Image taken from Andrew NG's lecture

10 → $\boxed{4 \times 4 \times 10}$

Problem with Convolution

26 August 2022 14:25

What is Padding?

26 August 2022 14:26

7	2	3	3	8
4	5	3	8	4
3	3	2	8	4
2	8	7	2	7
5	4	4	5	4

*

1	0	-1
1	0	-1
1	0	-1

7x1+4x1+3x1+
2x0+5x0+3x0+
3x-1+3x-1+2x-1
= 6

=

6		

$n-f+1=n$

$n \times n$

5×5

$\boxed{7 \times 7}$

$f \times f$

3×3

$\rightarrow (n-f+1)(n-f+1)$

$(5-3+1) = 3 \times 3$

$n-f+1 = 5$

$n-3+1=5 \Rightarrow n=8-1$

$n=7$

0	0	0	0	0	0	0
0	7	2	3	3	8	0
0	4	5	3	8	4	0
0	3	3	2	8	4	0
0	2	8	7	2	7	0
0	5	4	4	5	4	0
0	0	0	0	0	0	0

7×7

Zero padding

convolution

$5 \times 5 \rightarrow 3 \times 3$

$(n-f+1)$

$\downarrow$

$(n+2p-f+1)$

$5+2(1)-3+1$

$= 7-3+1 = 5$

Keras

Valid

Same

$5 \times 5 \rightarrow 3 \times 3$

Strides

27 August 2022 08:23

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

$\boxed{7\times 7}$

0	0	1
1	0	0
0	1	1

3x3

Stride = 1

(1,1)

3 x3

right

bottom

→ result feature

Stride = (2,2)

Stride = 2 → $\frac{7-3}{2} + 1$

$2+1=3$

s

$(n-f+1) \longrightarrow \left[\frac{n-f}{s} + 1\right]$ ↘ $p=p$

$\left[\frac{n+2p-f}{2} + 1\right] \longrightarrow$ strided convolution

$\frac{7+2-3}{2} + 1 = [4\times 4]$

$\frac{n-f}{2}$ $\frac{6-3}{2}$ $1.5 = 1 + 1 = 2$

Special case

Stride = 2

1	6	9	10	2	8	5
2	5	1	8	4	2	4
3	7	4	9	10	3	7
9	8	3	6	7	9	3
8	0	9	4	7	2	1

$\boxed{2\times 3}$

$\frac{6\times 7}{3\times 3}$

$\left[\frac{n-f}{s} + 1\right]$

floor

$1.9 \to 1$

$1.1 \to 1$

9	8	3	6	7	9	3
8	0	9	4	7	2	1
9	10	12	6	9	8	0

x6

$\left[\frac{\cdot}{s} + 1\right]$ floor

$\left[2 \quad \frac{6-3}{2} + 1 = 1.5 + 1\right.$

$1 + 1 = 2$

$\frac{7-3}{2} + 1 = 2 + 1 = 3$

Why Strides are required?

27 August 2022 08:24

1) High level features

2) Computing →

Keras → stride

$$\left[\frac{n+2p-f}{s}+1\right]$$

$$\frac{28+2-3}{2}+1$$

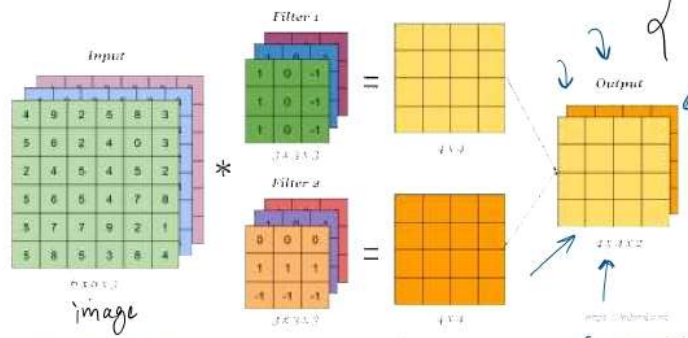
$$13.5+1$$

$$13+1=14$$

The Problem with Convolution

01 September 2022 09:55

stride=2



image

228 × 228 × 3

filters (100) → feature map → (226 × 226) × 100 × 32 → 19MB

1.5GB

1) Memory issue

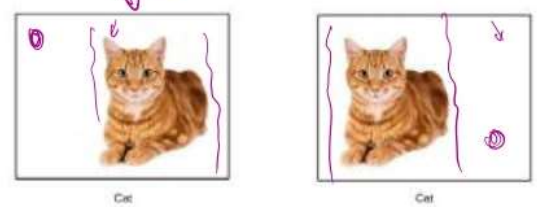
2) Translation variance

reduce size of feature map

pooling

1 training → 100

Translation Variance



features

{ location dependent }

location

pooling { down sample your feature map }

Pooling

01 September 2022 09:55

0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
255	255	255	255	255	255
255	255	255	255	255	255
255	255	255	255	255	255

*

-1	-1	-1
0	0	0
1	1	1

= relu

(4x4)

-	-	-	-
-			

feature map (non-linear)

Max pooling
Min pooling
Avg pooling
L2 pooling
Global pooling

max pooling

3	1	1	3
2	5	0	2
1	4	2	1
4	7	2	4

4x4

(size) → (2,2)
stride → (2)
type → Max

receptive field

5	3
7	4

2x2

feature map

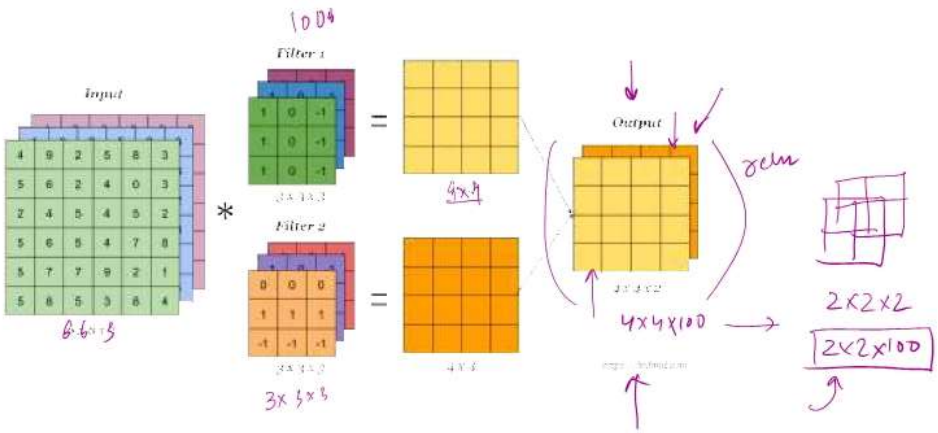
low level details eliminat

Demo

01 September 2022 09:57

Pooling on Volumes

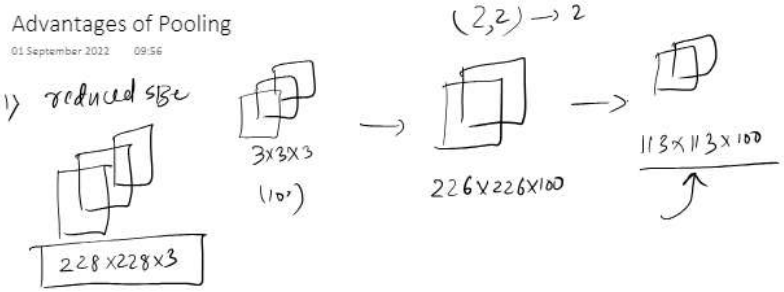
01 September 2022 09:56



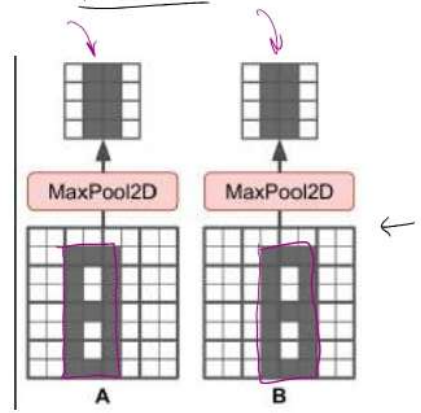
Advantages of Pooling

01 September 2022 09:56

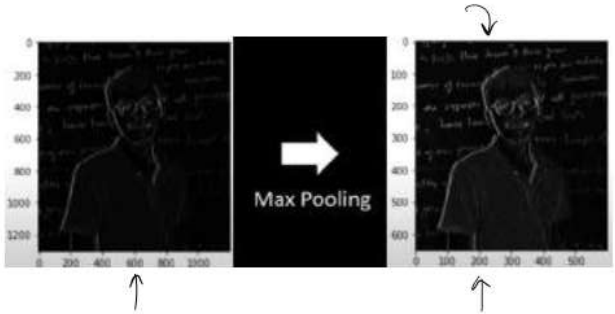
1) reduced size



2) Translation invariance



3) Enhanced features (only in case of Max pooling)



Keras Code

01 September 2022 09:56

4) No need of training

min

fastor

max pooling → aggregate

avg pooling

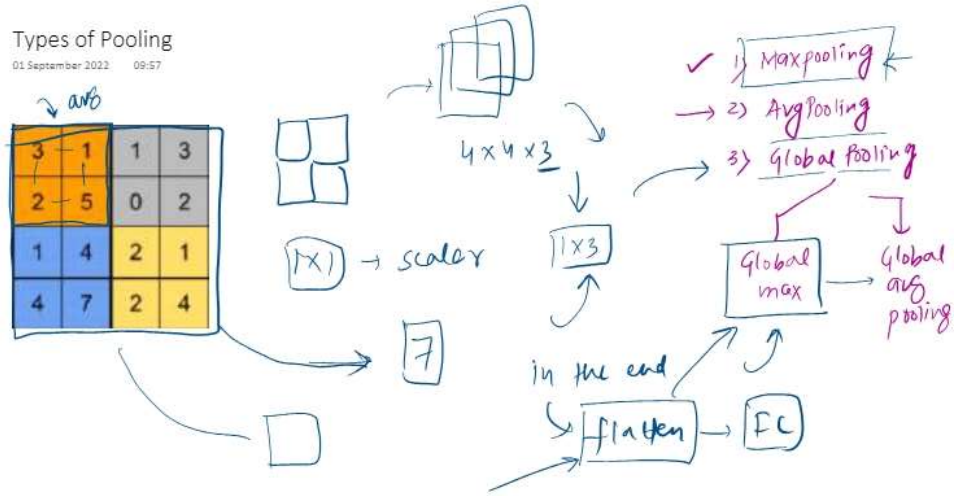
→ (2x2)

→ 2

→ type

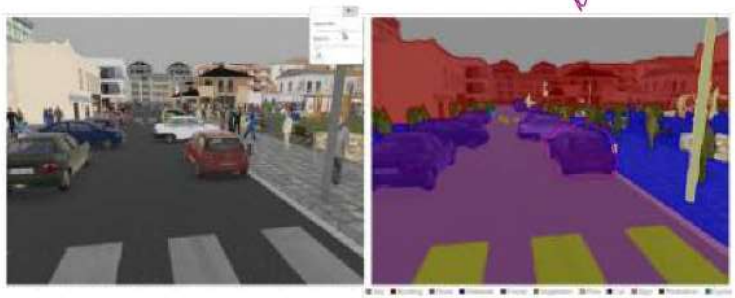
Types of Pooling

01 September 2022 09:57



Disadvantages of Pooling

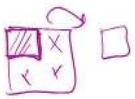
01 September 2022 09:57



lose a lot of info

4x4 → 16

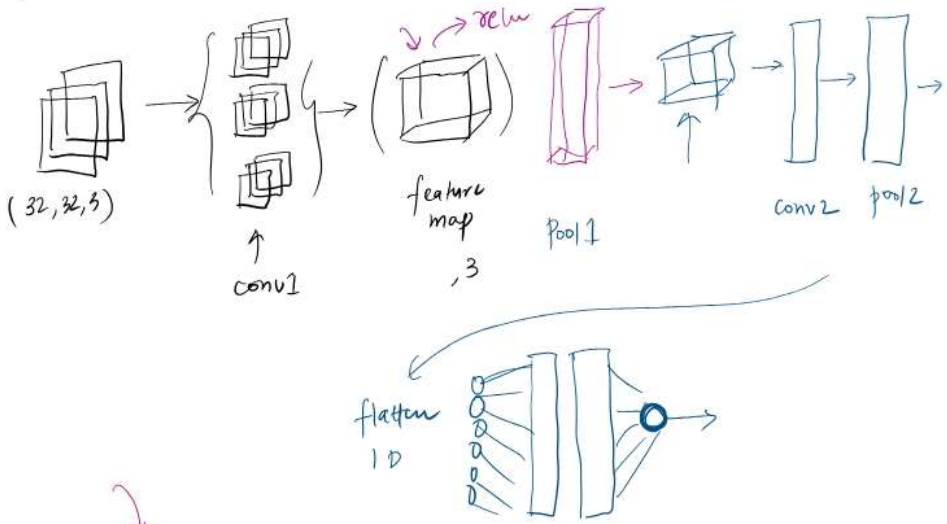
2x2 → 4



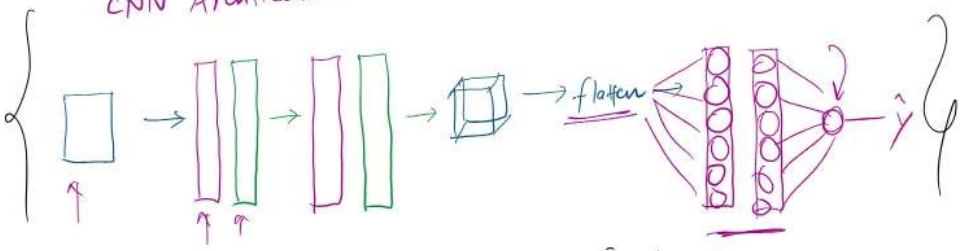
CNN Architecture

02 September 2022 10:55

1) Convolution 2) Padding /stride 3) Pooling



CNN Architecture



# conv # filters	stride padding	FC nodes # FC layer	activation dropout	Batch Norm

ImageNET

1) [LeNET] → Yann LeCunn
2) AlexNET
3) GoogLeNET
4) VggNET
5) ResNTE
6) Inception

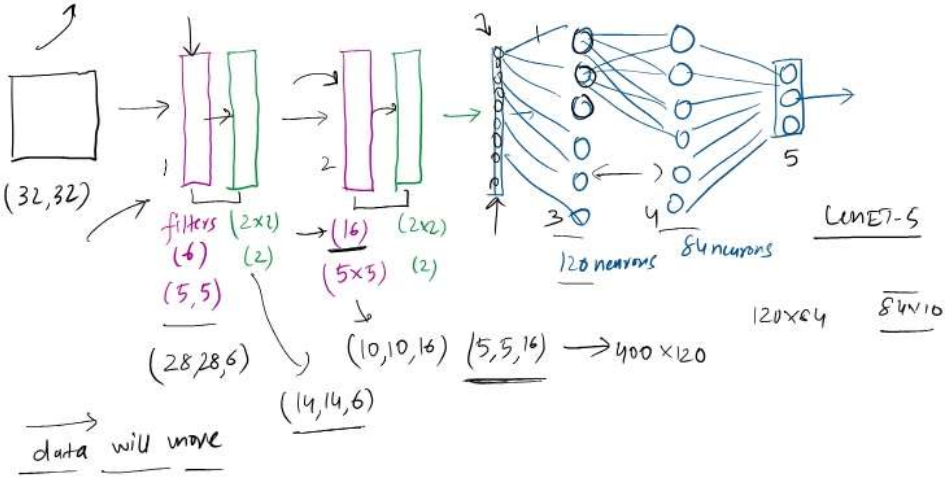
3) GoogLeNET

LeNet

02 September 2022 10:55

Yann LeCun → 1989 → 1998 → LeNET → CNN → US Navy Postal Service

LeNET-5



→ data will move

AlexNET
VggNET
ResNET
Inception

Guidelines

02 September 2022 10:58

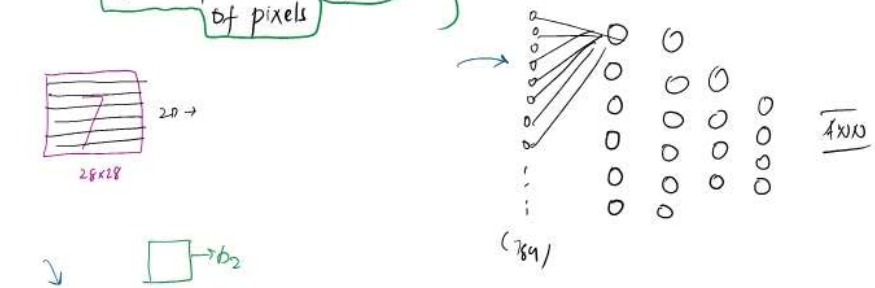
Keras Code

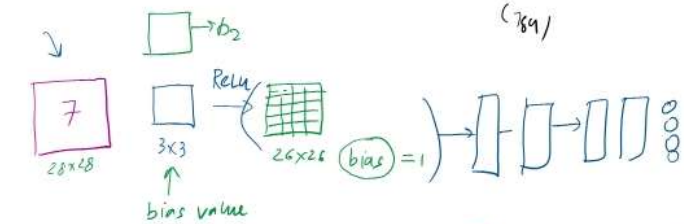
02 September 2022 14:58

CNN Vs ANN

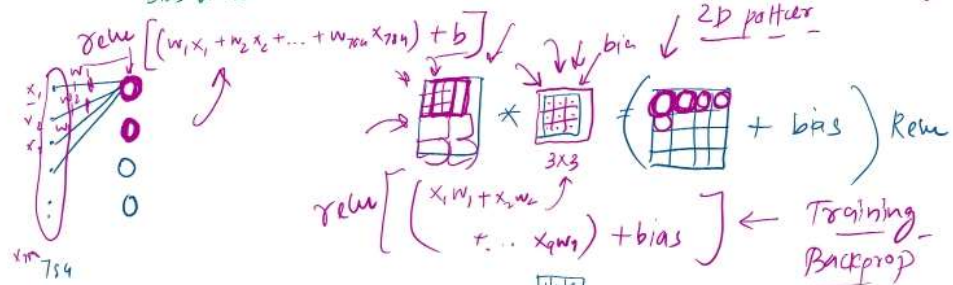
06 September 2022 10:00

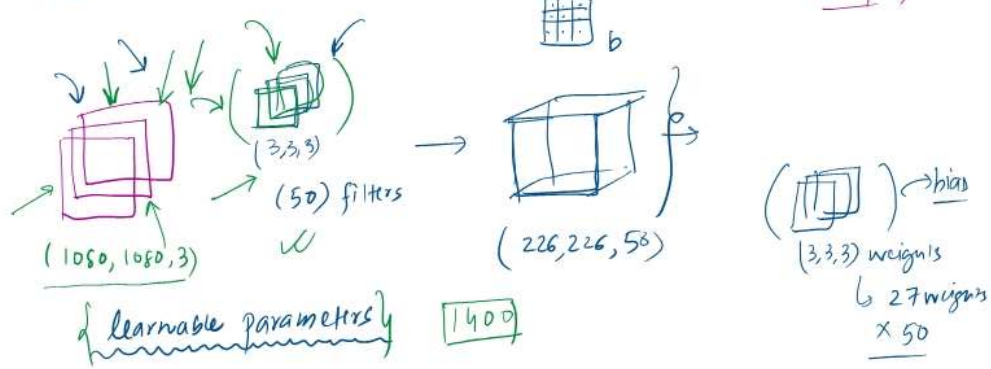
1) Computation Cost → ↓↓
2) Overfitting →
3) Loss of imp features like spatial arrangement of pixels



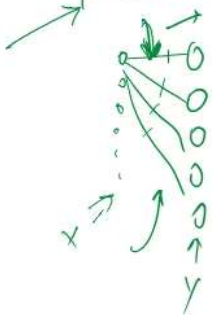


similarity ANN CNN





228, 228, 3 → 288 x 228 x 3 1050 X 1050 X 3



$\circledast * y \rightarrow$ Overfitting AND

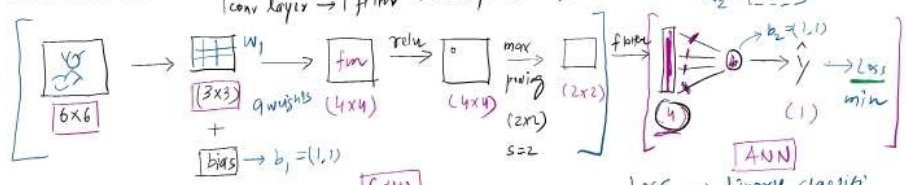
→ Backprop In CNN

1350 wts
+50
1400

Backpropagation in CNN

10 September 2022 11:12

1 conv layer → 1 filter + max pooling →



CNN

Loss → binary classif

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

$A_2 = \hat{y}$

Trainable Parameters

$W_1 = (3,3)$ $W_2 = (1,4)$ = 15 trainable parameters

$b_1 = (1,1)$ $b_2 = (1,1)$

Logical Flow

Forward Prop

$$\left\{\begin{array}{ll} Z_1 = \text{conv}(X, W_1) + b_1 & F = \text{flatten}(P_1) \\ A_1 = \text{relu}(Z_1) & Z_2 = W_2 F + b_2 \\ P_1 = \text{maxpool}(A_1) & A_2 = \sigma(Z_2) \end{array}\right\}$$

Gradient Descent

$$W_1 = W_1 - \eta \frac{\partial L}{\partial W_1} \checkmark \qquad W_2 = W_2 - \eta \frac{\partial L}{\partial W_2} \checkmark$$

$$b_1 = b_1 - \eta \frac{\partial L}{\partial b_1} \checkmark \qquad b_2 = b_2 - \eta \frac{\partial L}{\partial b_2} \checkmark$$

Loss is minimized

$\frac{\partial L}{\partial W_2}$

$X \rightarrow \circledast \rightarrow Z_1 \xrightarrow{\text{relu}} A_1 \xrightarrow[\text{pool}]{\text{max}} P_1 \xrightarrow{\text{flatten}} F \xrightarrow{W_2, b_2} Z_2 \xrightarrow{\text{sigmoid}} A_2 (\hat{y}) \rightarrow L$

W_1, b_1

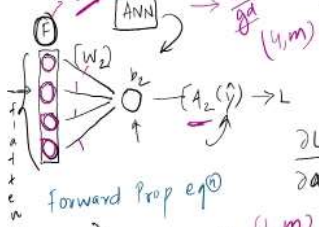
$$\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial A_2} \times \frac{\partial A_2}{\partial Z_2} \times \frac{\partial Z_2}{\partial W_2} \qquad \frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial A_2} \times \frac{\partial A_2}{\partial Z_2} \times \frac{\partial Z_2}{\partial b_2}$$

Backprop

$$\boxed{\frac{\partial L}{\partial w_1}} = \frac{\partial L}{\partial A_2} \times \frac{\partial A_2}{\partial Z_2} \times \frac{\partial Z_2}{\partial F} \times \boxed{\frac{\partial F}{\partial P_1}} \times \boxed{\frac{\partial P_1}{\partial A_1}} \times \frac{\partial A_1}{\partial Z_1} \times \boxed{\frac{\partial Z_1}{\partial w_1}}$$

$$\boxed{\frac{\partial L}{\partial b_1}} = \frac{\partial L}{\partial A_2} \times \frac{\partial A_2}{\partial Z_2} \times \frac{\partial Z_2}{\partial F} \times \frac{\partial F}{\partial P_1} \times \frac{\partial P_1}{\partial A_1} \times \frac{\partial A_1}{\partial Z_1} \times \boxed{\frac{\partial Z_1}{\partial b_1}}$$

Backprop
- → Convolution
- → Flatten
- → Max pooling



$$\boxed{\frac{\partial L}{\partial w_2} = \boxed{\frac{\partial L}{\partial A_2}} \frac{\partial A_2}{\partial Z_2} \frac{\partial Z_2}{\partial w_2}} \qquad \boxed{\frac{\partial L}{\partial b_2} = \boxed{\frac{\partial L}{\partial A_2}} \frac{\partial A_2}{\partial Z_2} \frac{\partial Z_2}{\partial b_2}}$$

$$\frac{\partial L}{\partial a_2} = \frac{\partial}{\partial a_2}\left[-y_i \log(a_2) - (1-y_i)\log(1-a_2)\right]$$

Forward Prop eq^n

$$\left\{\begin{aligned} Z_2 &= W_2 F + b_2 \\ A_2 &= \sigma(z_2) \end{aligned}\right\}$$

$$\underset{Y=(1,m)}{(1,m)} = \frac{-y_i}{a_2} + \frac{(1-y_i)}{(1-a_2)} = \frac{-y_i(1-a_2) + a_2(1-y_i)}{a_2(1-a_2)}$$

$$\frac{\partial L}{\partial a_2} = \frac{-y_i + \cancel{y_i a_2} + a_2 - \cancel{a_2 y_i}}{a_2(1-a_2)} = \frac{(a_2 - y_i)}{a_2(1-a_2)}$$

$$\frac{\partial A_2}{\partial z_2} = \sigma(z_2)[1-\sigma(z_2)] = a_2[1-a_2]$$

$$\boxed{\frac{\partial Z_2}{\partial w_2} = F} \qquad \frac{\partial Z_2}{\partial b_2} = 1$$

W_2 update

Shape =

$$\boxed{\frac{\partial L}{\partial w_2}}^{(1,4)} = \frac{(a_2 - y_i)}{\cancel{a_2(1-a_2)}} \times \cancel{a_2(1-a_2)} \times F = (a_2 - y_i)F = \underset{(1,1)}{(A_2} - \underset{(1,1)}{Y)}\underset{(4,1)}{F^T}$$

$$(1,1)(1,4)$$

$$(1,4)$$

$$\frac{\partial L}{\partial b_2} = \frac{(a_2 - y_i)}{\cancel{a_2(1-a_2)}} \cancel{a_2(1-a_2)} \times 1 = (A_2 - Y)$$

m images

$$\boxed{\boxed{\frac{\partial L}{\partial w_2}} = (A_2 - Y)F^T} \qquad \boxed{\frac{\partial L}{\partial b_2} = (A_2 - Y)}$$

$$\boxed{\boxed{\frac{\partial L}{\partial w_2}} = (A_2 - Y)F^T} \qquad \boxed{\frac{\partial L}{\partial b_2} = \cdots}$$

$$\underbrace{(1,m) - (1,m)}$$

$$\underbrace{(1,m)\ (m,4)} \rightarrow \boxed{(1,4)}$$

1 image

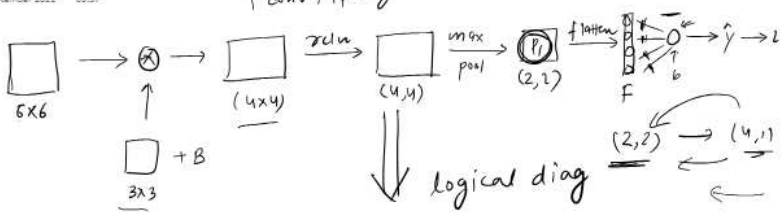
batch of image

$$W_2 \rightarrow (1,4)$$

Backpropogation in CNN

15 September 2022 09:37

1 conv + 1 pooling →



$\frac{\partial L}{\partial A_1} = (4,4)$

Flatten → no trainable parameters

$$\frac{\partial L}{\partial P_1} = \begin{bmatrix} x_1 & x_2 \\ x_3 & x_4 \end{bmatrix}_{2\times 2}$$

A_1: $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ $\begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$ $\begin{bmatrix} 9 & 10 \\ 11 & 12 \end{bmatrix}$ $\begin{bmatrix} 13 & 14 \\ 15 & 16 \end{bmatrix}$ → $\begin{bmatrix} 4 & 8 \\ 12 & 14 \end{bmatrix}$ $\hat{y}$ no grad

L

$\frac{\partial L}{\partial A_1}$ 2×2 → (4×4) error

(4×4)

$$\frac{\partial L}{\partial A_1} \rightarrow \begin{bmatrix} 0 & x_1 & x_2 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & x_3 & x_4 & 0 \end{bmatrix}$$

Forward Prop

$Z_1 = \text{conv}(X, W_1) + b_1$

$A_1 = \text{relu}(Z_1)$

$P_1 = \text{maxpool}(A_1)$

$\boxed{F = \text{flatten}(P_1)}$

flatten

$Z_2 = W_2 F + b_2$

$A_2 = \sigma(Z_2)$

$L = \frac{1}{m}\sum_{i=1}^{m}\left[-y_i \log(A_2) - (1-y_i)\log(1-A_2)\right]$

$[\ldots]$

$$\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial A_2} \times \frac{\partial A_2}{\partial Z_2} \frac{\partial Z_2}{\partial F} \frac{\partial F}{\partial P_1} \frac{\partial P_1}{\partial A_1} \frac{\partial A_1}{\partial Z_1} \frac{\partial Z_1}{\partial W_1}$$

$(A_2 - Y)$

$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial A_2} \frac{\partial A_2}{\partial Z_2} \frac{\partial Z_2}{\partial F} \frac{\partial F}{\partial P_1} \frac{\partial P_1}{\partial A_1} \frac{\partial A_1}{\partial Z_1} \frac{\partial Z_1}{\partial b_1}$$

reshape(P_1.shape)

6 derivatives

$\left[\frac{\partial Z_2}{\partial F}\right] = W_2 \rightarrow$

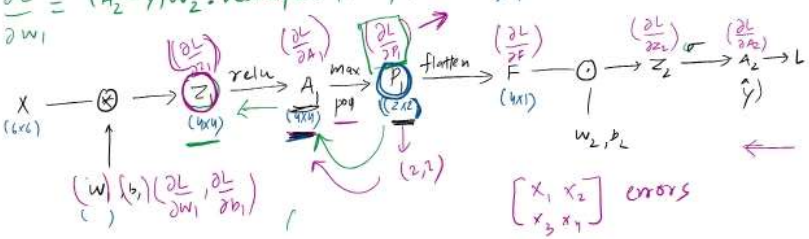
Shape ? → (F)

$\frac{\partial F}{\partial P_1}$ no trainable parameters

$\text{conv}\left(X, \frac{\partial L}{\partial Z_1}\right)$

$$\frac{\partial L}{\partial W_1} = (A_2 - Y) W_2 . \text{reshape}(P_1.\text{shape})$$

$\begin{bmatrix} x_1 & x_2 \\ x_3 & x_4 \end{bmatrix}$



$\begin{bmatrix} x_1 & x_2 \\ x_3 & x_4 \end{bmatrix}$ errors

$\frac{\partial L}{\partial A_1} = (4,4)$

$$\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial A_2} \frac{\partial A_2}{\partial Z_2} \frac{\partial Z_2}{\partial F} \frac{\partial F}{\partial P_1} \frac{\partial P_1}{\partial A_1} \frac{\partial A_1}{\partial Z_1} \frac{\partial Z_1}{\partial W_1}$$

$(A_2 - Y) W_2 . \text{reshape}(P_1.\text{shape})$ → $\frac{\partial L}{\partial A_1}$

$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial A_2} \frac{\partial A_2}{\partial Z_2} \frac{\partial Z_2}{\partial F} \frac{\partial F}{\partial P_1} \frac{\partial P_1}{\partial A_1} \frac{\partial A_1}{\partial Z_1} \frac{\partial Z_1}{\partial b_1}$$

$\frac{\partial L}{\partial P_1}$ $\boxed{\frac{\partial L}{\partial A_1}}$

$$\boxed{\frac{\partial L}{\partial A_1}} = \begin{cases} \frac{\partial L}{\partial P_{1xy}}, & \text{if } A_{mn} \text{ is the max element} \\ 0, & \text{otherwise} \end{cases}$$

A_1, P

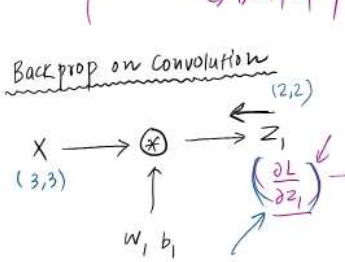
$$\frac{\partial A_1}{\partial Z_1} = \begin{cases} 1 & \text{if } Z_{1xy} > 0 \\ 0 & \text{if } Z_{1xy} < 0 \end{cases}$$

$\left(\frac{\partial L}{\partial Z_1}\right)$

Convolution
↳ Backprop → max pooling
→ Flatten

$$\frac{\partial L}{\partial Z_1} = \begin{bmatrix} \frac{\partial L}{\partial Z_{11}} & \frac{\partial L}{\partial Z_{12}} \\ \frac{\partial L}{\partial Z_{21}} & \frac{\partial L}{\partial Z_{22}} \end{bmatrix}$$

Backprop on Convolution



$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial Z_1} \times \boxed{\frac{\partial Z_1}{\partial b_1}}$$

$$Z_1 = \begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix}$$

w_1, b_1

$(2,2)$

$[z_{21} \quad z_{22}]$

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} \circledast W_1 = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$$

b_1

$$\boxed{\frac{\partial L}{\partial b_1}} = \frac{\partial L}{\partial z_1} \times \frac{\partial z_1}{\partial b_1} = \boxed{\frac{\partial L}{\partial z_{11}} \times \frac{\partial z_{11}}{\partial b_1} + \frac{\partial L}{\partial z_{12}} \times \frac{\partial z_{12}}{\partial b_1} + \frac{\partial L}{\partial z_{21}} \times \frac{\partial z_{21}}{\partial b_1} + \frac{\partial L}{\partial z_{22}} \times \frac{\partial z_{22}}{\partial b_1}}$$

$$= \left(\frac{\partial L}{\partial z_{11}} + \frac{\partial L}{\partial z_{12}} + \frac{\partial L}{\partial z_{21}} + \frac{\partial L}{\partial z_{22}} \right) = \text{sum}\left(\frac{\partial L}{\partial z_1} \right)$$

$$\boxed{\frac{\partial L}{\partial b_1}} = \text{sum}\left(\frac{\partial L}{\partial z_1} \right) \rightarrow \text{scalar}$$

bias

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} \quad W_1 = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} + b$$

$X \longrightarrow \circledast \longrightarrow Z_1$

(3×3)

W_1, b_1

(2×2)

$\left(\frac{\partial L}{\partial z_1} \right)$

$$\frac{\partial L}{\partial w_1} = \begin{bmatrix} \frac{\partial L}{\partial w_{11}} & \frac{\partial L}{\partial w_{12}} \\ \frac{\partial L}{\partial w_{21}} & \frac{\partial L}{\partial w_{22}} \end{bmatrix} \qquad \frac{\partial L}{\partial z_1} = \begin{bmatrix} \frac{\partial L}{\partial z_{11}} & \frac{\partial L}{\partial z_{12}} \\ \frac{\partial L}{\partial z_{21}} & \frac{\partial L}{\partial z_{22}} \end{bmatrix}$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial z_1} \times \frac{\partial z_1}{\partial w_1}$$

$$\begin{cases} \frac{\partial L}{\partial w_{11}} = \frac{\partial L}{\partial z_{11}} \times \left| \frac{\partial z_{11}}{\partial w_{11}} \right| + \frac{\partial L}{\partial z_{12}} \times \left| \frac{\partial z_{12}}{\partial w_{11}} \right| + \frac{\partial L}{\partial z_{21}} \times \left| \frac{\partial z_{21}}{\partial w_{11}} \right| + \frac{\partial L}{\partial z_{22}} \times \frac{\partial z_{22}}{\partial w_{11}} \\ \frac{\partial L}{\partial w_{12}} = \frac{\partial L}{\partial z_{11}} \times \frac{\partial z_{11}}{\partial w_{12}} + \frac{\partial L}{\partial z_{12}} \times \frac{\partial z_{12}}{\partial w_{12}} + \frac{\partial L}{\partial z_{21}} \times \frac{\partial z_{21}}{\partial w_{12}} + \frac{\partial L}{\partial z_{22}} \times \frac{\partial z_{22}}{\partial w_{12}} \\ \frac{\partial L}{\partial w_{21}} = \frac{\partial L}{\partial z_{11}} \times \frac{\partial z_{11}}{\partial w_{21}} + \frac{\partial L}{\partial z_{12}} \times \frac{\partial z_{12}}{\partial w_{21}} + \frac{\partial L}{\partial z_{21}} \times \frac{\partial z_{21}}{\partial w_{21}} + \frac{\partial L}{\partial z_{22}} \times \frac{\partial z_{22}}{\partial w_{21}} \\ \frac{\partial L}{\partial w_{22}} = \frac{\partial L}{\partial z_{11}} \times \frac{\partial z_{11}}{\partial w_{22}} + \frac{\partial L}{\partial z_{12}} \times \frac{\partial z_{12}}{\partial w_{22}} + \frac{\partial L}{\partial z_{21}} \times \frac{\partial z_{21}}{\partial w_{22}} + \frac{\partial L}{\partial z_{22}} \times \frac{\partial z_{22}}{\partial w_{22}} \end{cases}$$

$$\begin{cases} \frac{\partial L}{\partial w_{11}} = \frac{\partial L}{\partial z_{11}} x_{11} + \frac{\partial L}{\partial z_{12}} x_{12} + \frac{\partial L}{\partial z_{21}} x_{21} + \frac{\partial L}{\partial z_{22}} x_{22} \\ \frac{\partial L}{\partial w_{12}} = \frac{\partial L}{\partial z_{11}} x_{12} + \frac{\partial L}{\partial z_{12}} x_{13} + \frac{\partial L}{\partial z_{21}} x_{22} + \frac{\partial L}{\partial z_{22}} x_{23} \\ \frac{\partial L}{\partial w_{21}} = \frac{\partial L}{\partial z_{11}} x_{21} + \frac{\partial L}{\partial z_{12}} x_{22} + \frac{\partial L}{\partial z_{21}} x_{31} + \frac{\partial L}{\partial z_{22}} x_{32} \\ \frac{\partial L}{\partial w_{22}} = \frac{\partial L}{\partial z_{11}} x_{22} + \frac{\partial L}{\partial z_{12}} x_{23} + \frac{\partial L}{\partial z_{21}} x_{32} + \frac{\partial L}{\partial z_{22}} x_{33} \end{cases}$$

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} \qquad \frac{\partial L}{\partial z_1} = \begin{bmatrix} \frac{\partial L}{\partial z_{11}} & \frac{\partial L}{\partial z_{12}} \\ \frac{\partial L}{\partial z_{21}} & \frac{\partial L}{\partial z_{22}} \end{bmatrix}$$

$$\frac{\partial L}{\partial w_1} = \text{conv}\left(X, \frac{\partial L}{\partial z_1} \right)$$

$$\frac{\partial L}{\partial w_1} = \text{conv}\left(X, \frac{\partial L}{\partial z_1} \right)$$

$$\frac{\partial L}{\partial b_1} = \text{sum}\left(\frac{\partial L}{\partial z_1} \right)$$

Why use Pretrained models?

03 October 2022 12:52

1) Data hungry → image data → labelled data → cat dog → google

pinterest

labelling → 10000 photos → manually

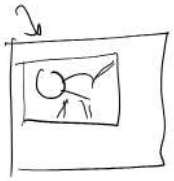
2) Time → model → training

Pretrained model

ImageNET Dataset

03 October 2022 12:36

Visual Database of images (Why What and How)



Why

2006 → Fei fei Li → model and algorithms

1.4 million images → 1.4 crore image → 20,000 categories → labelling

1 million images → to batch → bounding box → object local

How → crowd sourcing

→ Amazon mechanical Turk

↳ Dataset → Deep learning

ImageNET challenge

ILSVRC

03 October 2022 12:38

Imagenet Challenge → 2010

Dataset → subset → 1 million → 10

14 crore

20,000 classes

1000 classes

100 image

Machine learning

→ feature extraction → SIFT, HOG

2010 → 28%

2011 → 25%

2012 → 16% → CNN

2013 →

2012 → DL

AlexNET → CNN → relu

Deep

GPU

Famous Architectures

03 October 2022 12:39

2010 → ML model → 28%

2011 → ML Model → 25%

2012 → Alex NET → 16.4%

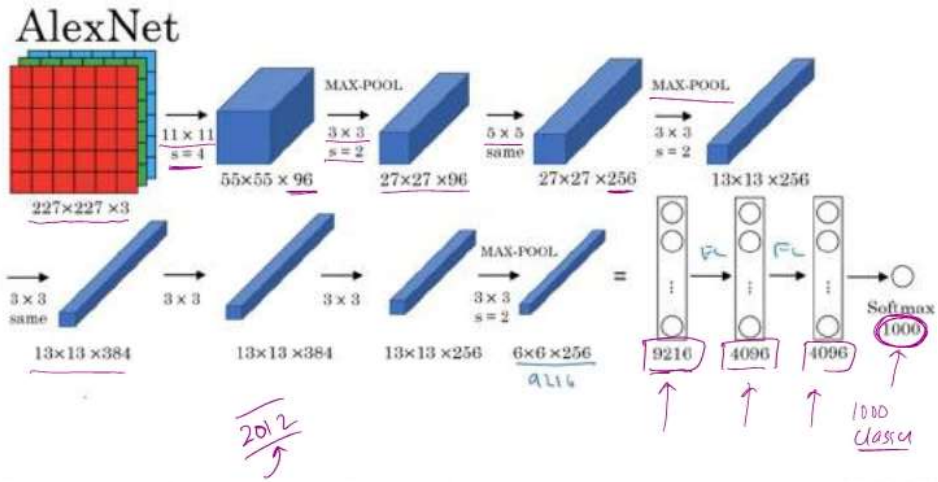
2013 → ZFNET → 11.7%

2014 → VGG → 7.3% → famous →

2015 → GoogleNET → 6.7%

2016 → ResNET → 3.5% → humans → 5% → 100

more layers



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

Andrew Ng

Idea of Pretrained Models

03 October 2022 12:39

VggNET
ResNET → ImageNet → 1000

Keras → pretrained → use

use them

Keras Demo

03 October 2022 12:40

Problem with training your own model

10 October 2022 10:48

1) Data hungry → labelled → 10,000 → google

↓

cat/dog → manual labour

2) lot of time → days/weeks/

Using Pretrained Models

10 October 2022 10:48

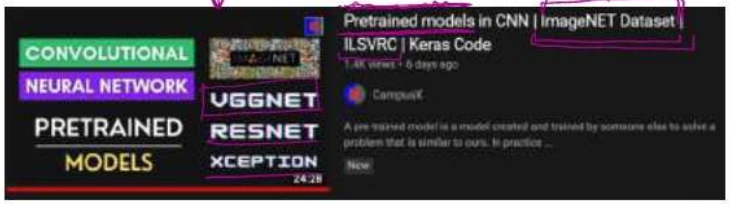
2010 2012

images → daily objects animal

1.4 million

1000 classes

chair
bread
dog breed



VGG ResNET

Deep learning

chair
bread
dog
tomato

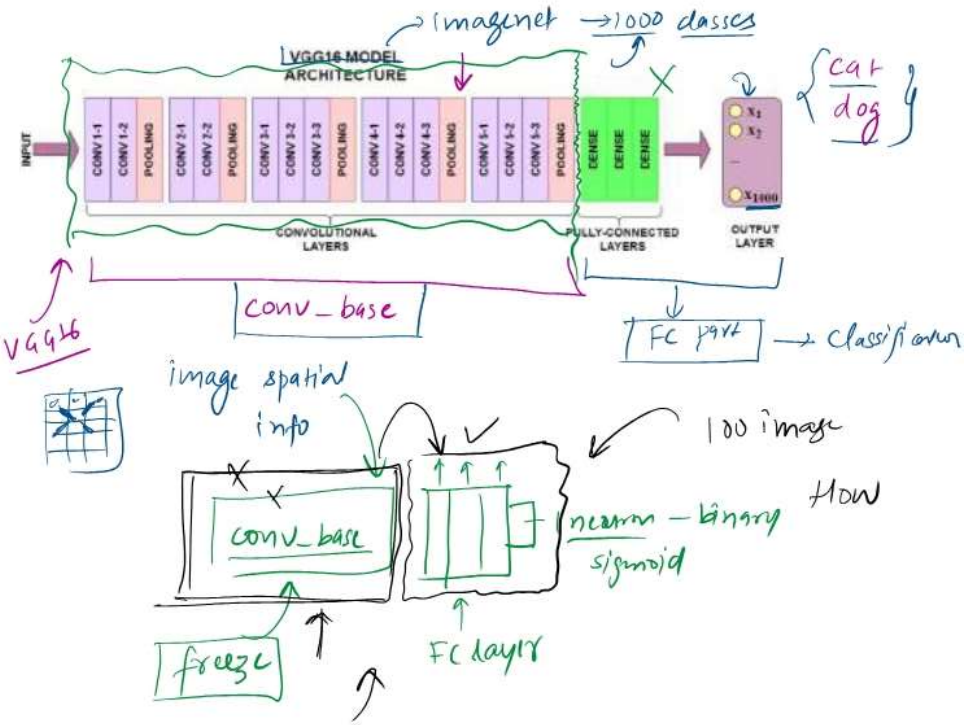
→ Transfer learning

phone/tablet

Transfer Learning

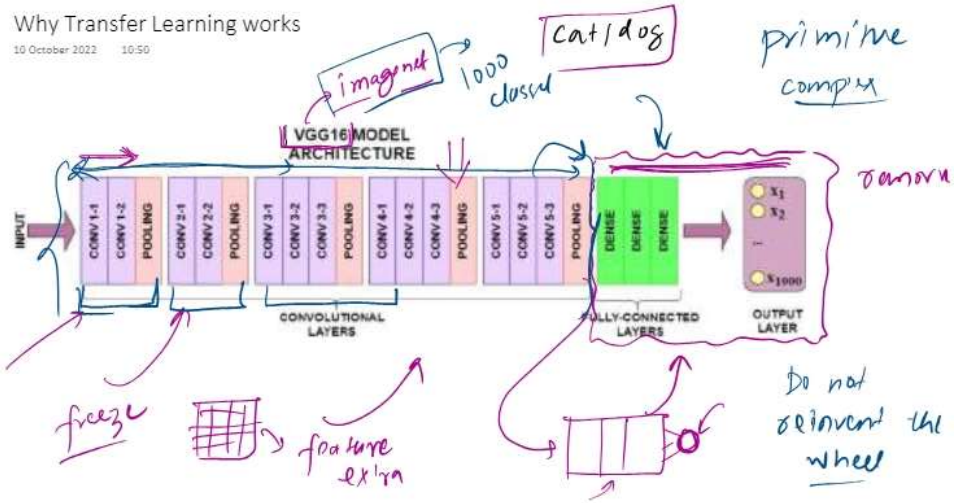
10 October 2022 10:49

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.



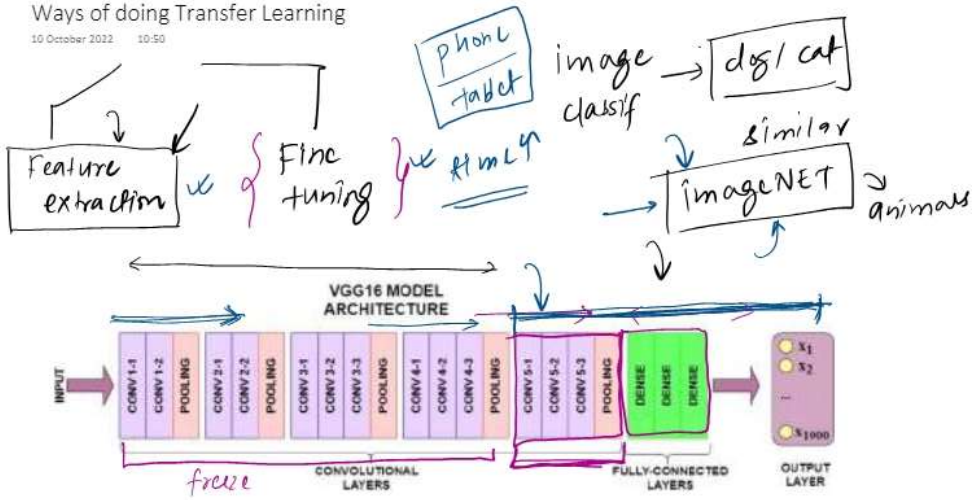
Why Transfer Learning works

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

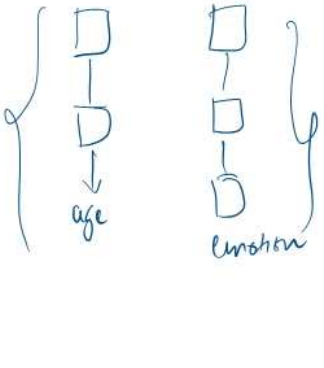
→ 1 output

→ linear

Example → → image dataset → human faces → 25000 images

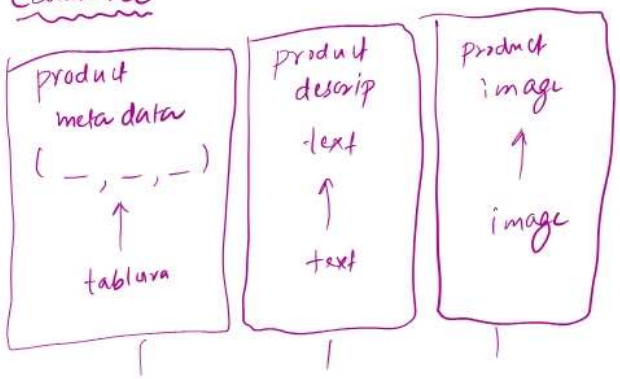
→ age ?

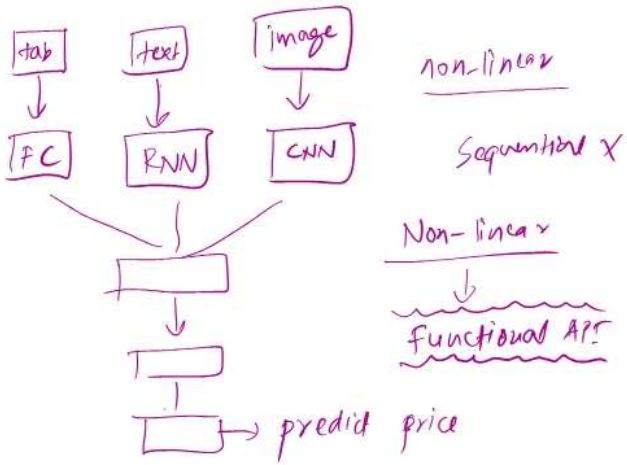
→ emotion → happy, sad, angry



Sequential API

ecommerce





non-linear

Sequential X

Non-linear

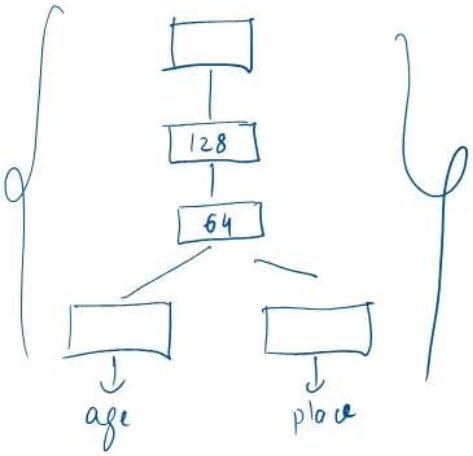
Functional API

A Simple Example

14 October 2022 16:01

yearly salary	height	marital status

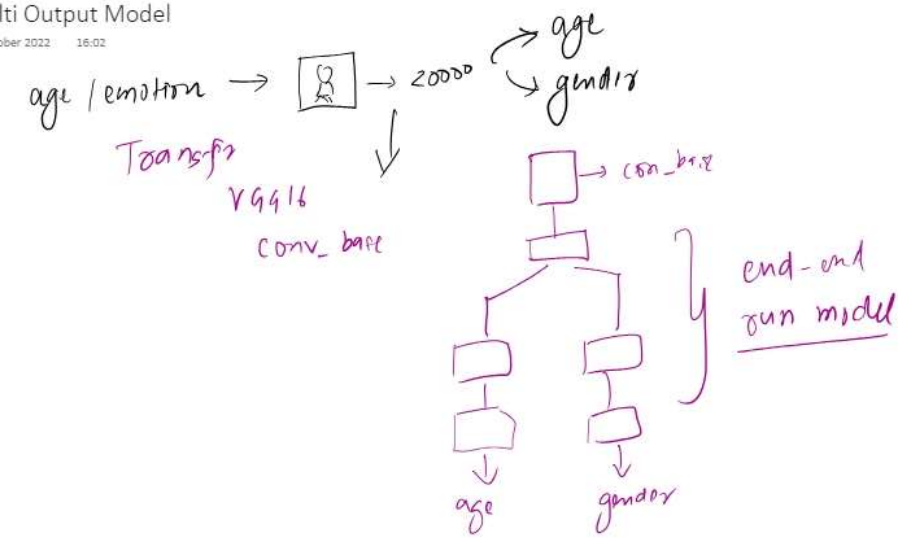
3 cols → age, delhi/mumbai



Keras functional api

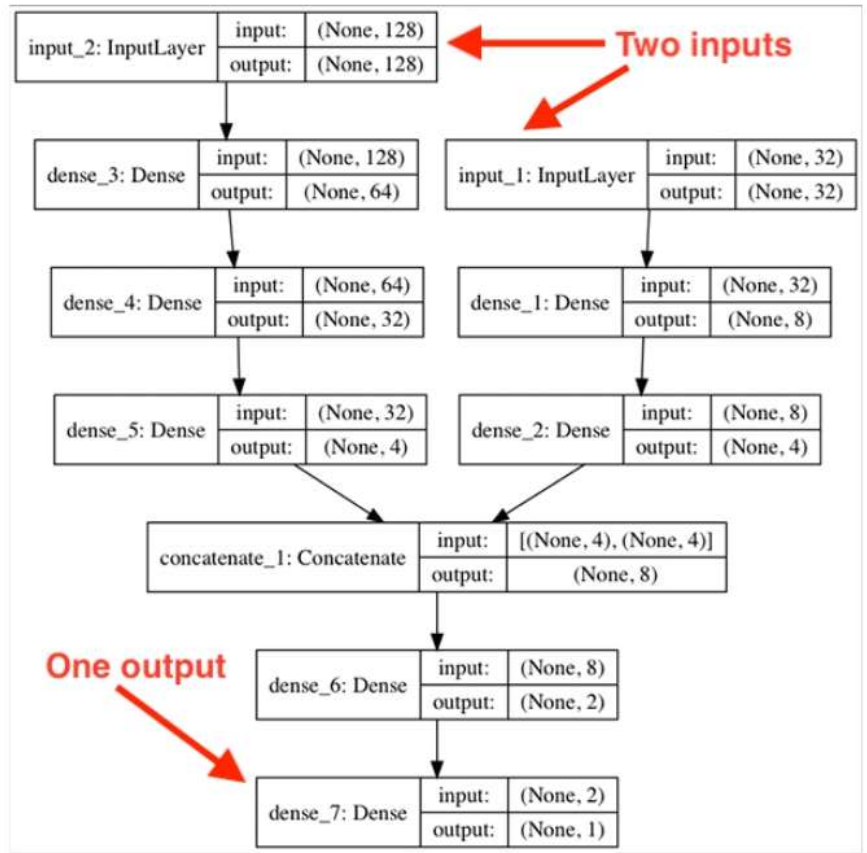
Multi Output Model

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 15:09

ANN → tabular data

CNN → images

RNN → Recurrent NN

↳ is type of sequential model to work on sequential data

| iq | marks | gender | placement

does not matter

No

iq, marks, gender.

RNN → NLP → ML

→ RNN

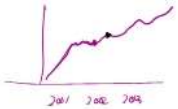
CNN → images → computer vision

eg → text → sequential data

Hi my name is Nitish

Time series

sequential



Speech

sequential

DNA sequence

RNN

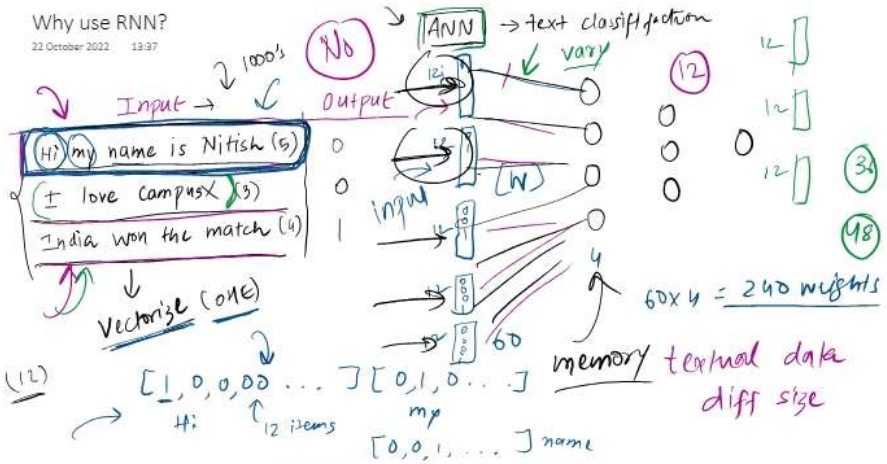
→ RNN → ANN/CNN Why?

→ Application → RNN

→ Roadmap

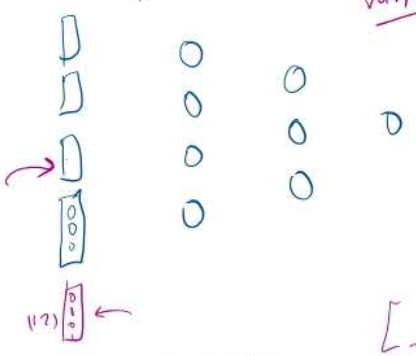
Why use RNN?

22 October 2022 13:37



zero padding

varying



Vocab → 10000 words

10K numbers

max word

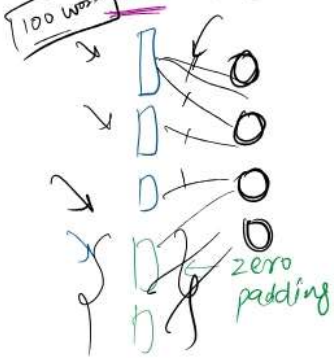
100 words

[_ _ _ . . _]

10000 numbs

100 words

min → 5 words



10000 x 100

10000000

unneccessary computation

fail

prediction

↓

200 words

1) text input → varying size

2) zero padding → unneccesary computation

3) Prediction problem

4) Totally disregarding the sequence info

RNN → ANN

[sequence] → 1980

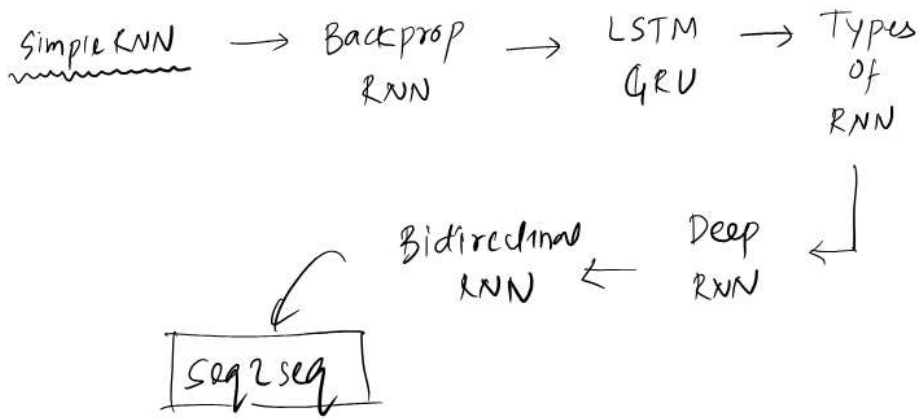
1) text input → varying size

RNN Applications

22 October 2022 13:37

Roadmap

22 October 2022 13:37



Why RNNs?

29 October 2022 13:30

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

zero padding → cost of compn ↑

movie reviews

RNNs

2) sequence contains some meaning

same

ANN → input

NNs → 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

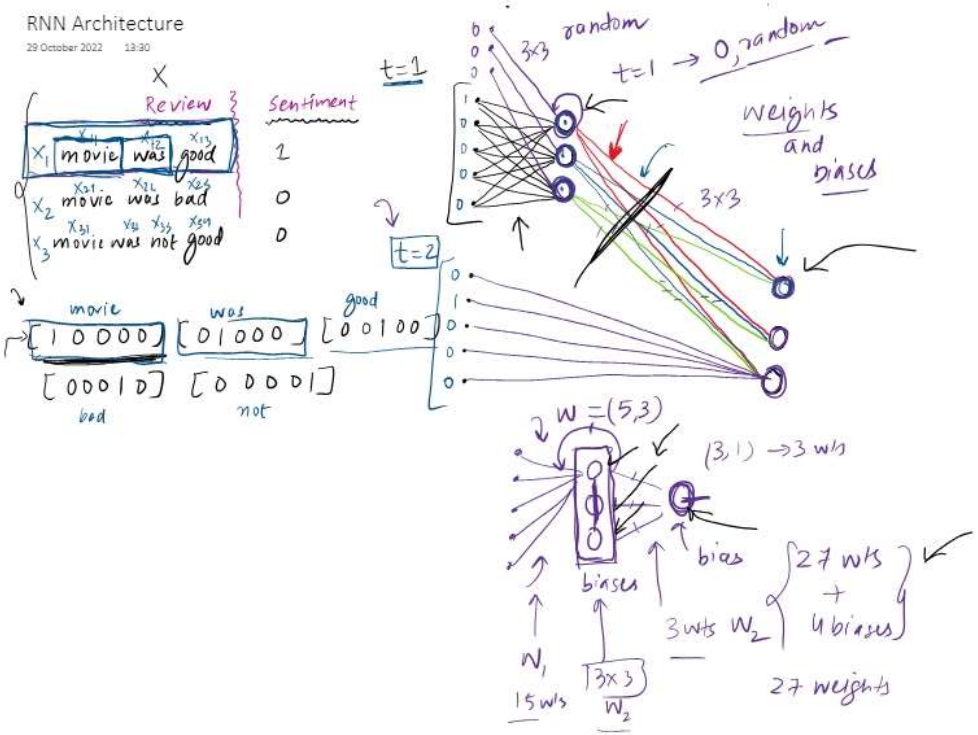
(3,5) (4,5)

Keras → SimpleRNN → batch size, timesteps, input_features

(3, 4, 5) → 3D tensor → RNN

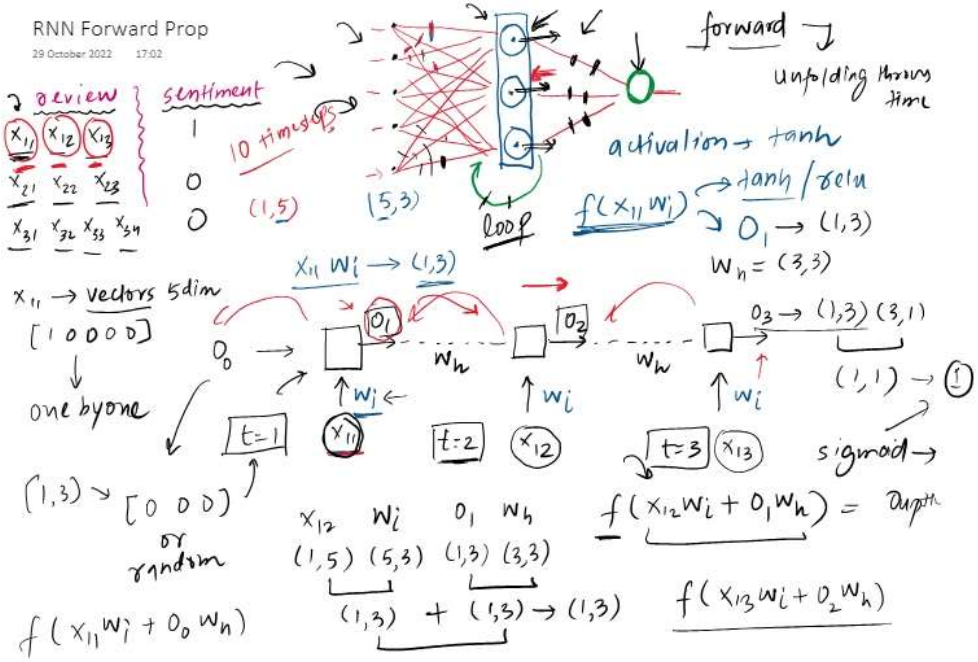
RNN Architecture

29 October 2022 13:30



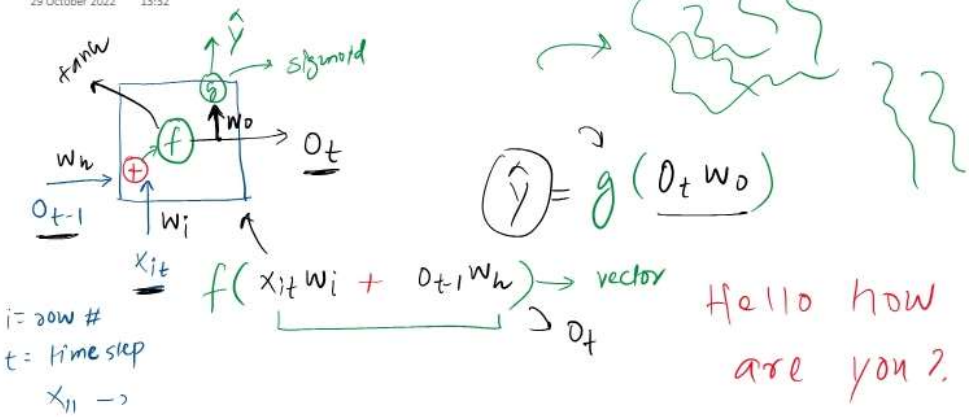
RNN Forward Prop

29 October 2022 17:02



Simplified Representation

29 October 2022 13:32



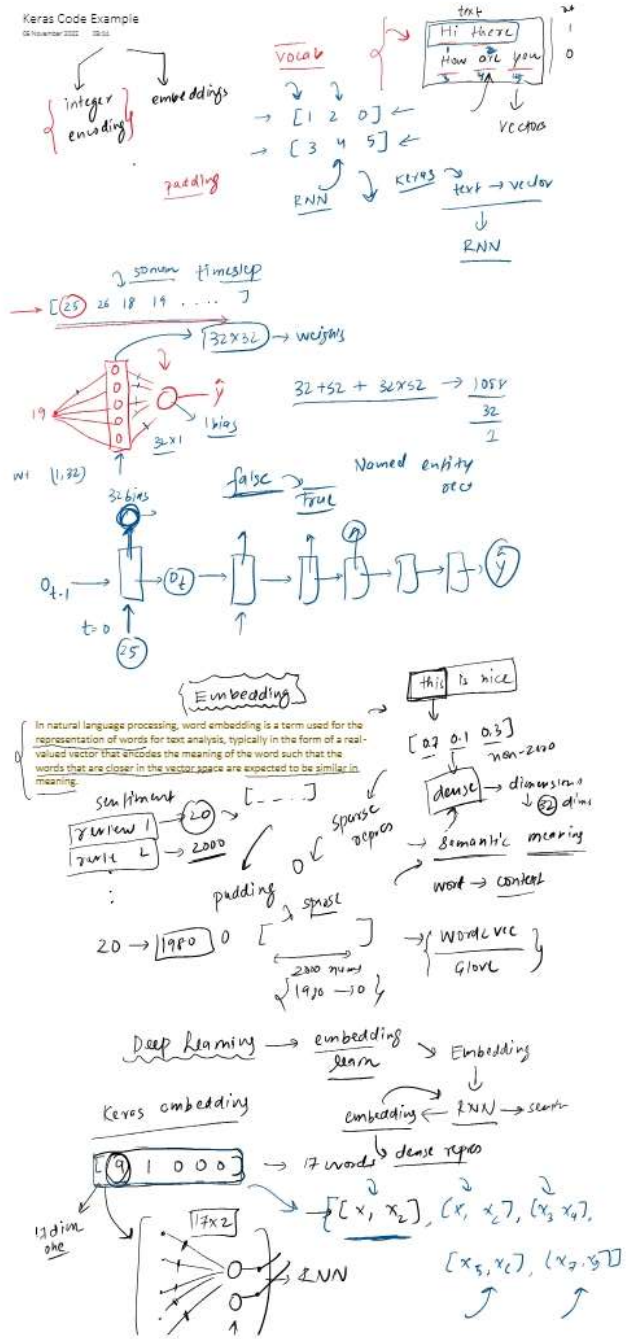
Hello how are you?

Code

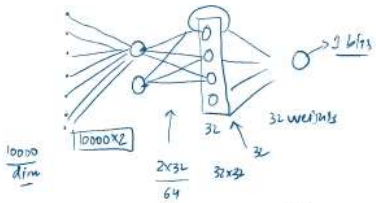
29 October 2022 13:32

State and Memory

29 October 2022 13:33



12 nodes



population in 9th year x

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

$x + 0.1x = 10000$

$1.1x = 10000$

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

$t(n-1)$

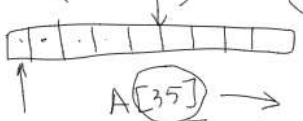
$\frac{x-1}{x} + \frac{1}{2}\left(\frac{x-1}{x}\right)^2 + \frac{1}{3}\left(\frac{x-1}{x}\right)^3 + \frac{1}{4}\left(\frac{x-1}{x}\right)^4 +$

$+ 2t$

$+ (n-1)$

50 → 500

A

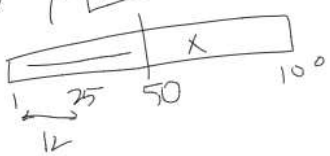


A[35] → t sec

A[35] →

35

$1 \times 4 \times 35$



O(n)

O(n²) → nested loops

input → 10 10+10

time [x]² 16 4

O(log(n))

Binary Search

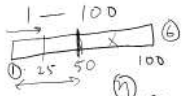
1 — 100

(6)

$\sqrt{(x_1}$

Binary

1 — 100



n

7

1 12 25 $\frac{n}{2}$

1 12 $\frac{n}{4}$ $\frac{n}{8}$

1 1000

O(n log n)

Sorting

$O(2^n)$ → exponential

1	2	3	4
2	4	8	32

for i in range
— O(n)

for j in range(n²)
O(n)

O(n+n) →
O(2n)
→ O(n)

$O(n + n^2)$ O(n²)

for i in range(n)
for j in

0123456789 n)

25 → '25'

str()

n = 345%10

digits[5]

5 + ''
= '5'

345//10 → 34 != 0

34%10 → 4

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

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(1000000)

n² 1000000
×
O(n²)

1 → n/2 log(L)/2

4 n/2 2n

O(n) n=100

50, 100 n/2

2 → 100=n n/2

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

n/2 × log n
n log n

O(1) → constant

n = 345

3+4+5 → 12

5

3 4

3 3

0

3450 → 4

log

inp → 10 100
out → 1 2 3

fibonacci

↳ function
↳ recursion

0 1 1 2 3 5 8 13

5

n=1 n=0 → 1

fib(n)

function calls

input → 4 → 10
#fcalls →

fib 3 n=3

fib(2) fib(1) → 2

fib(1) fib(0)

exponential → bad

fib(3)

chain

O(2^n) 15 20 30

input 1 2 3 4
... 2 4 8 16

$O(2^n)$ 50

input 1 2 3 4

4 2 4 8 16

$\boxed{n = 50, 100, 500}$

↑ weeks

exponential

↳ days/weeks

subset → $O(?)$

power set

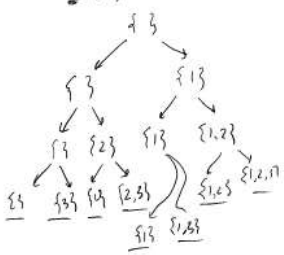
$\{1,2\} \to \{\{3\},\{1\},\{2\},\{1,2\}\}$

$\{1,2,3\} \to$

$\{3\},\{2,3\},\{2\},\{3\},\{1,3\}\{2,3\}$

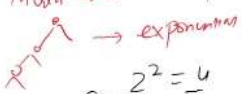
$\{3,1\}\ \{1,2,3\}$

$\{1,2,3\}$



reduce → divide → log

increase → double → exp



→ exponential

$2^2 = 4$

$2^3 = 8$

$2^4 = 16$

$\{1,2\} \to 4$

$\{1,2,3\} \to 8$

$O(2^n)$

$O(?)$

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

$n>0$

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

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

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

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

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

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

$= 3^n T(0)$

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

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

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

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

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

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

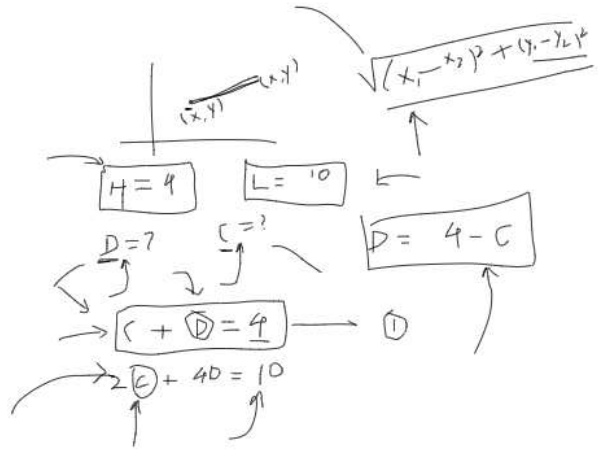
$= 2^3 T(n-3) - 2^2 - 2^1 - 2^0$

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

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

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

$O(1) \to$ constant



2 6 9 12 15 5

$1^2 + 2^2 + 3^2 + 4^2 + 5^2$ $n = 5$

[3, 6 → 5th

$a = 3$ $n = 5$

$d = 6 - 3$

1009

2060

$\frac{2}{3} \times \frac{4}{5} = \frac{10 + 12}{15} \rightarrow \frac{22}{15}$

0 1 1 2 3 5

$5! = 5 \times 4 \times 3 \times 2 \times 1$

1000

1042

2222

UP → 1

D – 3

R – 4

L – 3

a=[1,2]

a=b

a=b[:]

a → b

Till Now

17 November 2022 17:01

→ Roadmap → overview

↳ RNN → theoritical

↳ RNN → Keras → sentiment analy

↳ Types of RNN → Backprop RNN

Many to One

One to many

Many2 many

One 2 one

examples

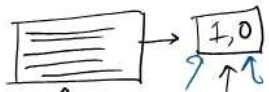
Many to One

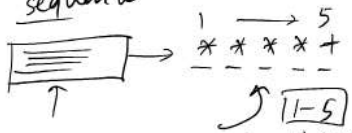
17 November 2022 17:02

input → sequence → sent, charg, time series

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

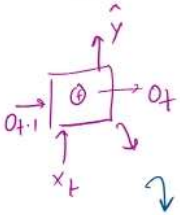
↳ sentiment analy →



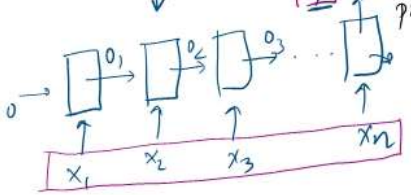


sequence

multiclass classifi



$\hat{y}$ → Rating prediction



recurrent → simple RNN

many to one

One to Many

17 November 2022 17:02

→ normal non sequential

↳

→ Output → sequentials

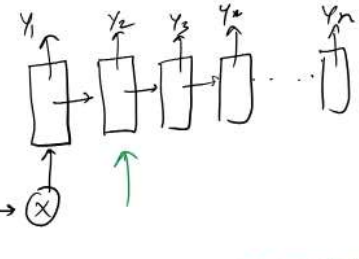
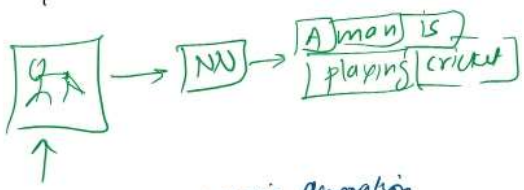
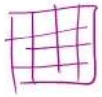


image captioning





→ textual repres

→ music generation

Many to Many

17 November 2022 17:02

input → sequent
out → sequen
→ seq2 seq

Same length

input seq == output seq

Variable length

pos tagging → NLP

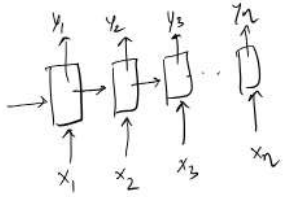
1 2 3 4

My name is Nitish

pronoun 1, noun 2, verb 3, Noun 4

NER

Lets meet at 7pm at the airport



same length
many2many
RNN

Variable length

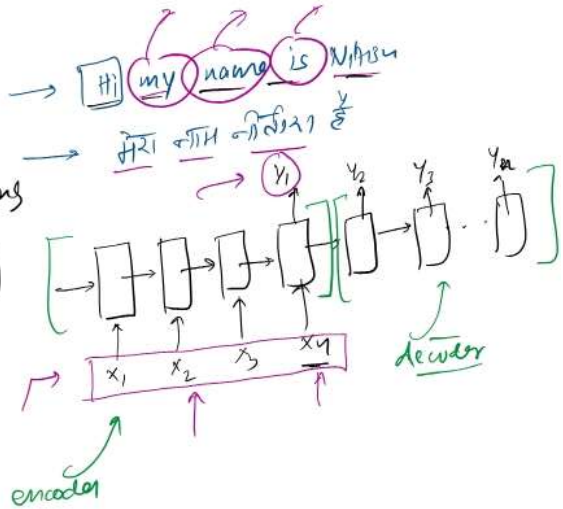
machine trans
↳ 1 lang → 2 lang

google translate

encoder
decoder

→ Hi my name is Nitish

→ मेरा नाम नीतिश है



One to One

17 November 2022 17:02

one → non sequential → image classif
out → non sequ

→ NN → ANN / CNN

→ 1,0

- many 2 one
- one 2 many
- many 2 many
  - same
  - variable

Backpropagation in RNN

01 December 2022 18:43

RNN Intro → RNN → practical → types of RNN

→ RNN → Backprop → BPTT

Many to One RNN

Sentiment Analysis

text → 1/0

text

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

x_1 [1 0 0] [0 1 0] [0 0 1]

x_2 [0 0 1] [0 0 1] [0 1 0]

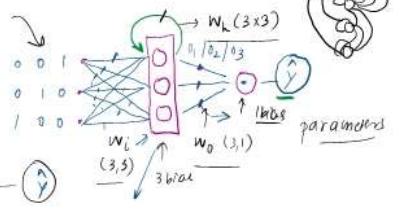
x_3 [0 1 0] [0 1 0] [1 0 0]

o_1 x_{o_2} o_3 $\hat{y}$ y L

W_i W_h W_o

Vocab → Cat Mat Rat

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

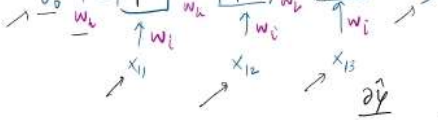


parameters

forward prop

set zeros

Loss calculate ← $\hat{y}$



$o_1 = f(x_{11} w_i + o_0 w_h)$

$o_2 = f(x_{12} w_i + o_1 w_h)$

$o_3 = f(x_{13} w_i + o_2 w_h)$

$\frac{\partial \hat{y}}{\partial w_o}$

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

loss calculate → minimize

L min gradient descent

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

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

W_i W_h W_o

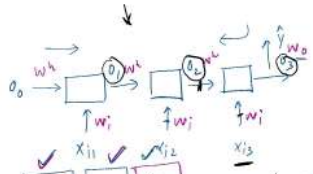
$$W_i = W_i - \eta \frac{\partial L}{\partial W_i}$$

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

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

Backprop

unfold → 3 times



$\frac{\partial L}{\partial W_o}$ $L \to \hat{y} \to o_3, W_o$

$$\boxed{\frac{\partial L}{\partial W_o} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial W_o}}$$

$\boxed{\frac{\partial L}{\partial W_o}}, \boxed{\frac{\partial L}{\partial W_i}}, \boxed{\frac{\partial L}{\partial W_h}}$

$o_1 = f(x_{11} w_i + o_0 w_h)$

$o_2 = f(x_{12} w_i + o_1 w_h)$

$o_3 = f(x_{13} w_i + o_2 w_h)$

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

$\frac{\partial L}{\partial W_i}$ $L \to \hat{y} \to o_3, W_o, x_{i3}, W_i, o_2, W_h, x_{i2}, W_i, o_1, W_h, W_i, o_0, W_h, x_{i1}$

3 terms

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

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

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

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

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

10 words → 10 terms

$t = 1$

$$\boxed{\frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial o_3} \frac{\partial o_3}{\partial W_i}}$$

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

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

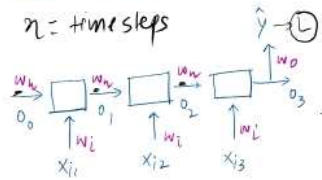
$$\frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial o_2} \frac{\partial o_2}{\partial W_i} = \boxed{\frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial o_3} \frac{\partial o_3}{\partial o_2} \frac{\partial o_2}{\partial W_i}}$$

$$\boxed{\frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial o_3} \frac{\partial o_3}{\partial W_i}}$$

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

n = time steps

$$\boxed{\frac{\partial L}{\partial W_h}}$$



$L \to \hat{y} \to W_o, o_3 \to x_{i3}, W_i, o_2, W_h, x_{i2}, W_i, o_1, W_h, x_{i1}$

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

$\partial L \quad \partial \hat{y} \quad \partial o_2 \quad \partial o_2 \quad +$

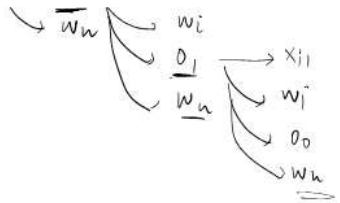
$\partial w_h \quad \partial y \quad \partial o_3 \quad \partial w_h$

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

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

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

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 $j=10$

$$\frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial o_{10}} \frac{\partial o_{10}}{\partial o_1} \frac{\partial o_1}{\partial w_h} \quad \boxed{\prod_{t=2}^{j} \frac{\partial o_t}{\partial o_{t-1}}}$$

$$\prod_{t=2}^{t=3} \frac{\partial o_t}{\partial o_{t-1}} = \frac{\partial o_2}{\partial o_1} \frac{\partial o_3}{\partial o_2}$$

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

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

$[0-1]$

Problem with RNN

19 December 2022 16:33

RNNs → sequential data → textual, time series

Suffer (2) major problems → LTSMs (Way)

RNN

(1) Problem of long term dependency
(2) ~~Unstable~~ gradients Stagnated values } → Unstable gradients

Applicant next word predicts

→ RNN

→ Marathi is spoken in Maha

short

→ Maha is a beautiful place I went there last year, but I could not enjoyed proper bcz I dont understand marathi

long term

→ RNN

→ RNN → keypad

{ Vanishing gradient prob } → RNN

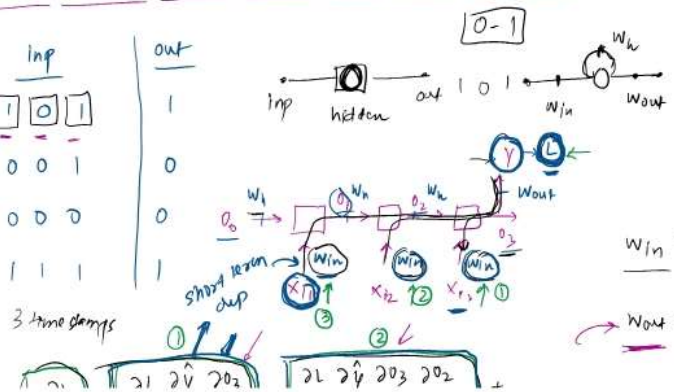
{ Exploding gradient problem } → RNN

possible sol(n)
→ 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

3 time stamps



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

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

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

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

long term dep

inf

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

→ 3

100 time steps

$$\frac{\partial L}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial o_{100}}\left[\frac{\partial o_{100}}{\partial o_{99}} \cdots \frac{\partial o_2}{\partial o_1}\right]\frac{\partial o_1}{\partial w_{in}} \rightarrow \frac{\partial L}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial o_{100}}\prod_{t=2}^{100}\left(\frac{\partial o_t}{\partial o_{t-1}}\right)\frac{\partial o_1}{\partial w_{in}}$$

$$\frac{\partial o_2}{\partial o_1}\ \frac{\partial o_3}{\partial o_2}\ \frac{\partial o_4}{\partial o_3} \cdots \frac{\partial o_{100}}{\partial o_{99}}$$

$$o_1 = \tanh(x_{i_1} w_{in} + o_0 w_h)$$

$$o_t = \tanh(x_{i_t} w_{in} + o_{t-1} w_h)$$

$$\frac{\partial o_t}{\partial o_{t-1}} = \tanh'(x_{i_t} w_{in} + o_{t-1} w_h)\, w_h$$

$$\frac{\partial L}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial o_{100}}\left[\prod_{t=2}^{100}\left(\tanh'(_)\, w_h\right)\right]\frac{\partial o_1}{\partial w_{in}}$$

100 or more

0-1 0-1 ≃ 0 ≃ 0

rely

0-1

Vanishing grad.

0-1 (1)

identity matrix

Sol⓪

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

ANN

sequential data → text → time

hi my name is nitish

length

vectors

pos

negative

50 words

RNN time

RNN (state)

RNN

RNN

h_0 h_1 h_2 h_3 h_4 h_5

hi my name is nil

t=1

RNN

Mahn is a beautiful state.
The langu spkn in is marathi

RNN

Mahn

man

Vanishing / Exploding gradients

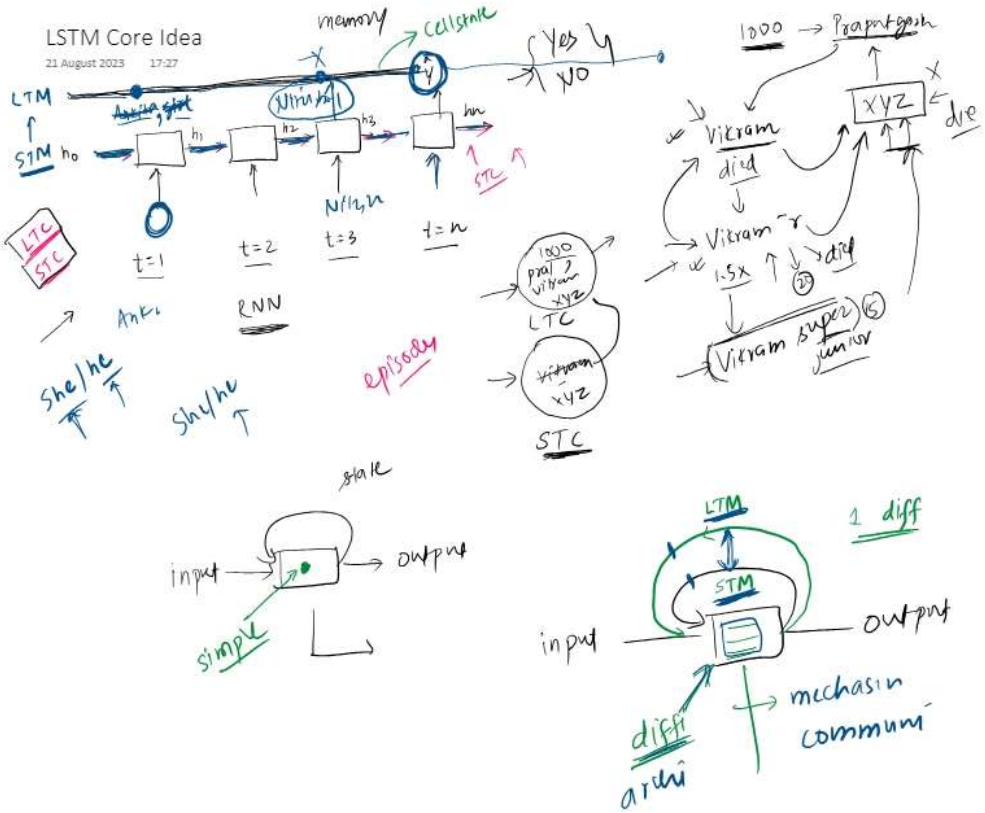
RNN

LSTM

RNN → handle long sequences

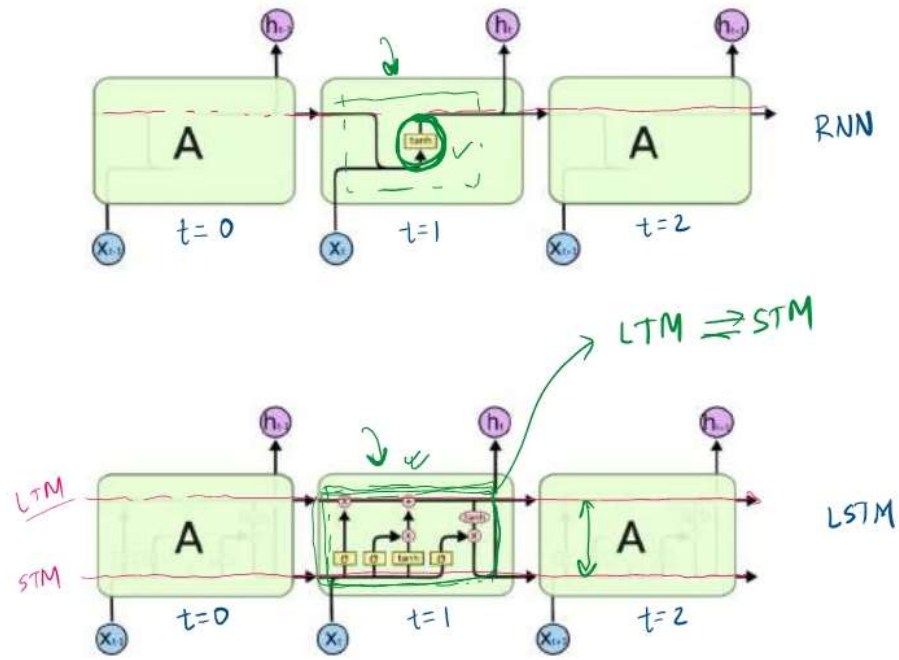
LSTM Core Idea

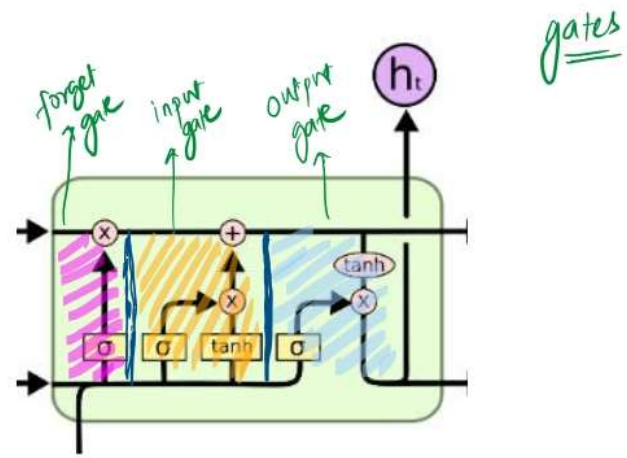
21 August 2023 17:27



LSTM Architecture

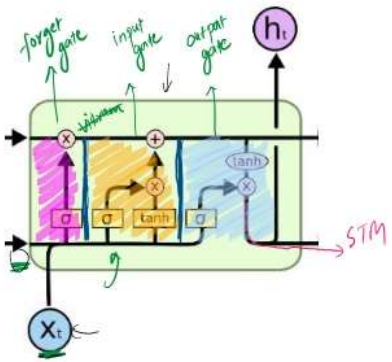
21 August 2023 18:41





LSTM Gates

21 August 2023 19:06

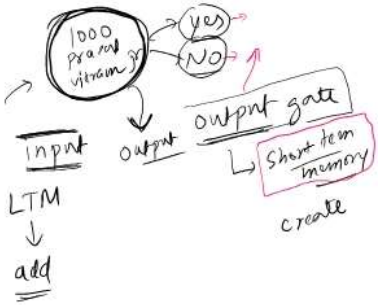


forget

input

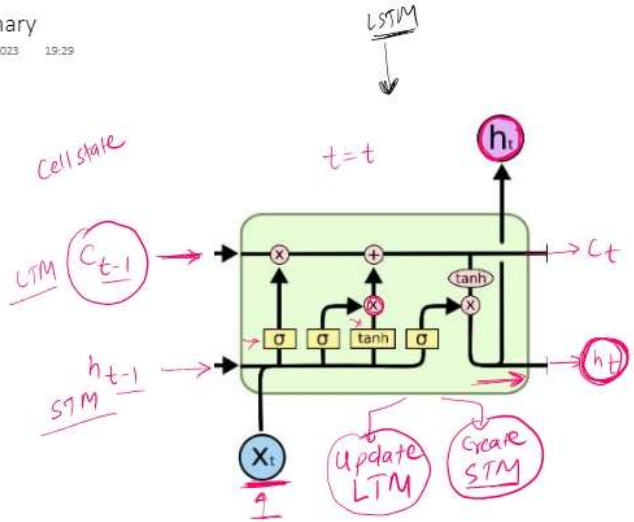
LTM

remove



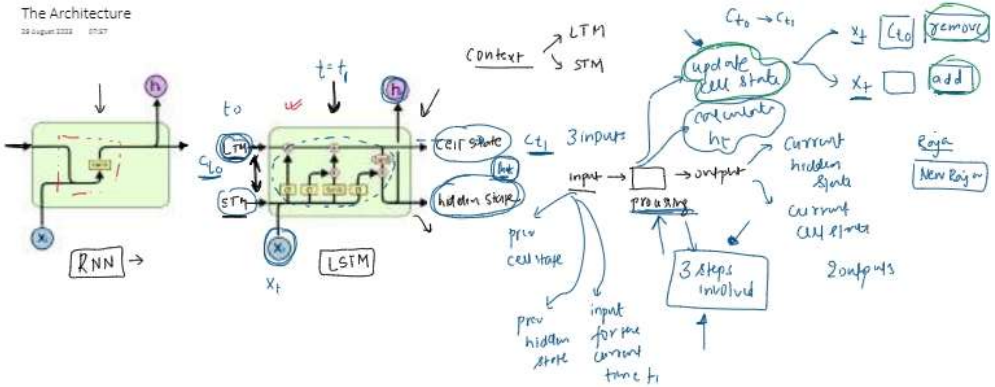
Summary

21 August 2023 19:29



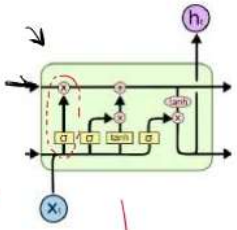
The Architecture

28 August 2023 07:57

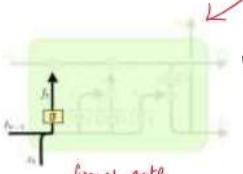


The Gates

29 August 2023 08:09



3 gates

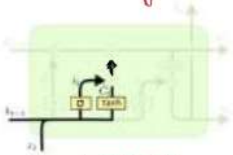


forget gate

to remove something from C_t

update C_t

+



input gate

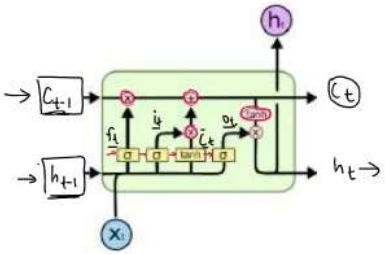
to add something to the C_t

output gate

calculate hidden state h_t

What are C_t and h_t

29 August 2023 08:08



$\boxed{h_t \quad C_t}$

vectors

[0.1 0.3 0.9]

3d vector

rule

same

$\boxed{h_t \quad C_t}$ dim equal

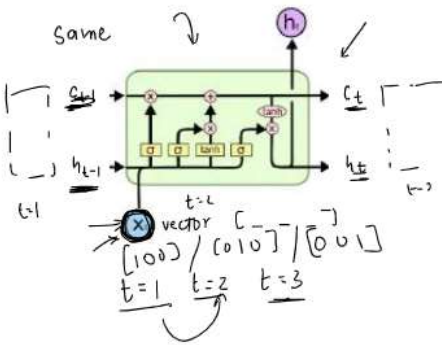
h_t [0.1 0.55 0.6]

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]

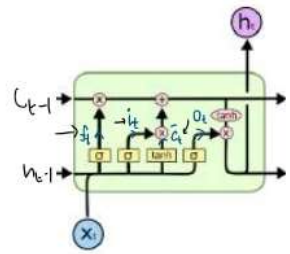
word vector

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 — forget gate

i_t — input gate

$\bar{C}_t$ → candidate cell state

o_t — output gate

vectors → $\boxed{C_t \ h_t}$

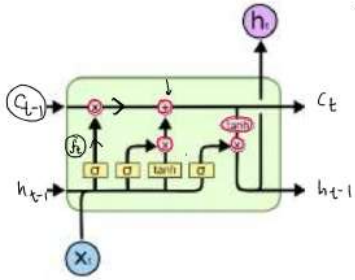
f_t i_t $\bar{C}_t$ o_t

[x y z]

[ ]

Pointwise Operations

29 August 2023 18:26



→ ⊗
→ ⊕
→ tanh

$C_{t-1} = [4\ 5\ 6]$ → [0.26 0.34 0.53]

tanh(6)

$f_t = [1\ 2\ 3]$

[5 7 9]

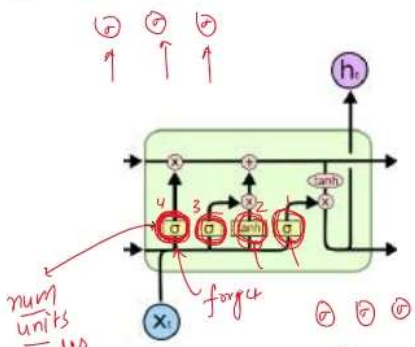
shape/dim
↓
vector

$C_{t-1} \otimes f_t$ → vector

→ [4 10 18]

→Neural Network Layers

29 August 2023 18:34

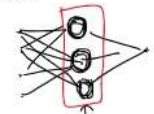


num units
nodes

forget

hyperparameter
↳ (4)

ANN



hidden

σ σ σ σ

tanh

σ σ σ

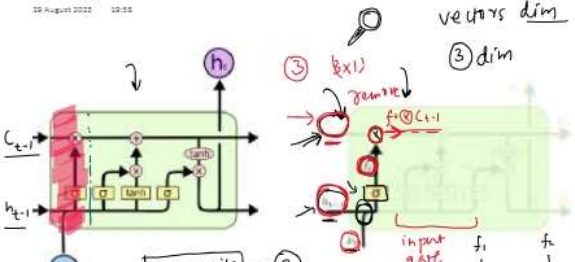
σ σ σ σ

(4)

σ σ σ σ

The Forget Gate

22 August 2022 19:55



4 dim vector $[x_{i1}\ x_{i2}\ x_{i3}\ x_{in}]$

→ num units → 3

vectors dim

3 dim

3 (3x1)

remove

$f_t \otimes C_{t-1}$

input gate (add)

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

$[\ ]$ → $[\ ]$

h_{t-1} (3) x_t (4) → 7x1

input

output → cell state → remove

(3x1) ⊗ (3x1)

σ σ σ

→ 1) calculate f_t

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

forget gate ← 0% 50% 100%

remove

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

long term

$[4\ 5\ 6]$

C_{t-1} → $[0\ 0\ 0]$ ⊗

f_t → $[\frac{1}{2}\ \frac{1}{2}\ \frac{1}{2}]$

$[1\ 1\ 1]$

$[4\ 5\ 6]$

$[2\ 2.5\ 3]$

$[0\ 0\ 0]$ → 100% remove

f_t

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

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

(3x1)

removal

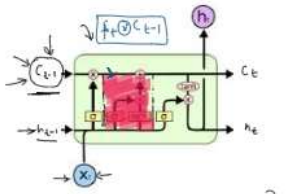
$f_t \otimes C_{t-1}$ → (3x1)

(C_{t-1}) → remove

The Input Gate

22 August 2022 20:05

add some new imp info to the C_t



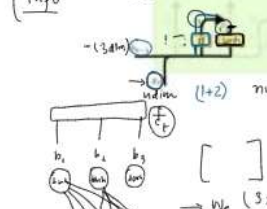
stages

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

2) i_t

3) C_t → current cell state

potential filtering important info



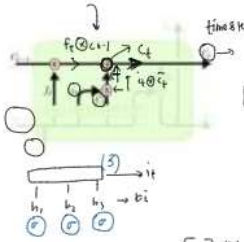
$\bar{C}_t$ = (3x1)

$[\ ]$ → W_c (3x7)

$\bar{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c)$

(3x7) (7x1) (3x1) (3x1) (3x1) → (3x1)

h_{t-1} x_t



timestep

pointwise $i_t \otimes \bar{C}_t$ → $\bar{C}_t'$ (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

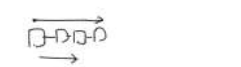
$\bar{C}_t$ → C_t

0%

$[4\ 5\ 6]$ $t = t-1$ → $[4\ 5\ 6]$ $t = t$

$f_t = [1\ 1\ 1]$

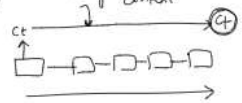
$i_t \otimes \bar{C}_t = [0\ 0\ 0]$



$[4\ 5\ 6]$ $[0\ 0\ 0]$

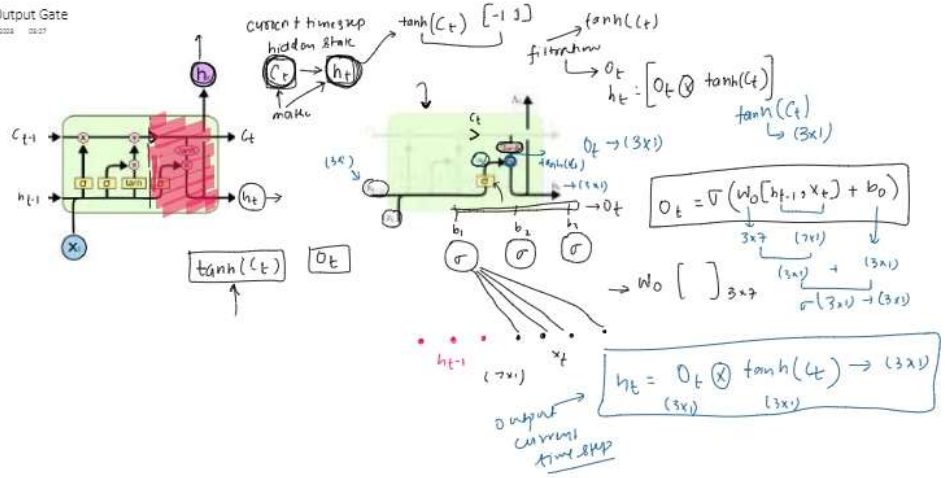
$C_t = [4\ 5\ 6]$

long term context



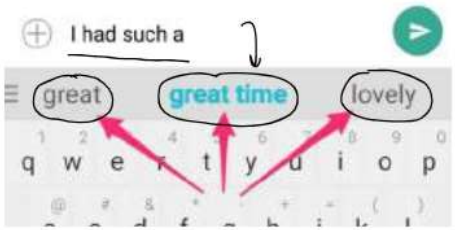
The Output Gate

30 August 2023 09:27

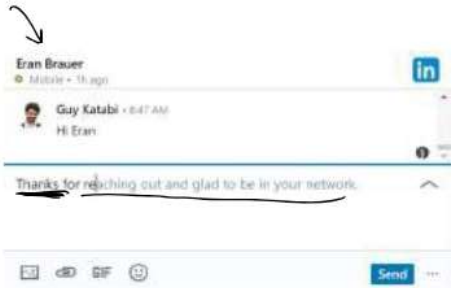


What is a Next Word Predictor

08 September 2023 08:50



code



The Strategy

08 September 2023 08:51

[Text generator]

→ supervised learning

model → input | output

input → output

dataset

input	output
Hi	my
Hi my	name
Hi my name	is
Hi my name is	Nitish
I	live
I live	in
I live in	gurgaon

9 words

1	2	3	4	5
Hi	my	name	is	Nitish
I	live	in	gurgaon	
6	7	8	9	

numbers [001000000]

2.1 2.7

[010000000]

↳ 2 word

LSTM

input	output
[1]	2
[1,2]	3
[1,2,3]	4
[1,2,3,4]	5
[6]	7
[6,7]	8

reg → discard

input	output
English	
number	

OHE

LSTM

classification

binary

multi class

about the program

[93 1 13]

input	output
[93]	1
[93,1]	13

text gen → supervised

Reg

Classification

~~9 two~~ 3 words

→ [0.62 0.51 0.31]

[0.33 0.41 0.21]

[1 0 0]

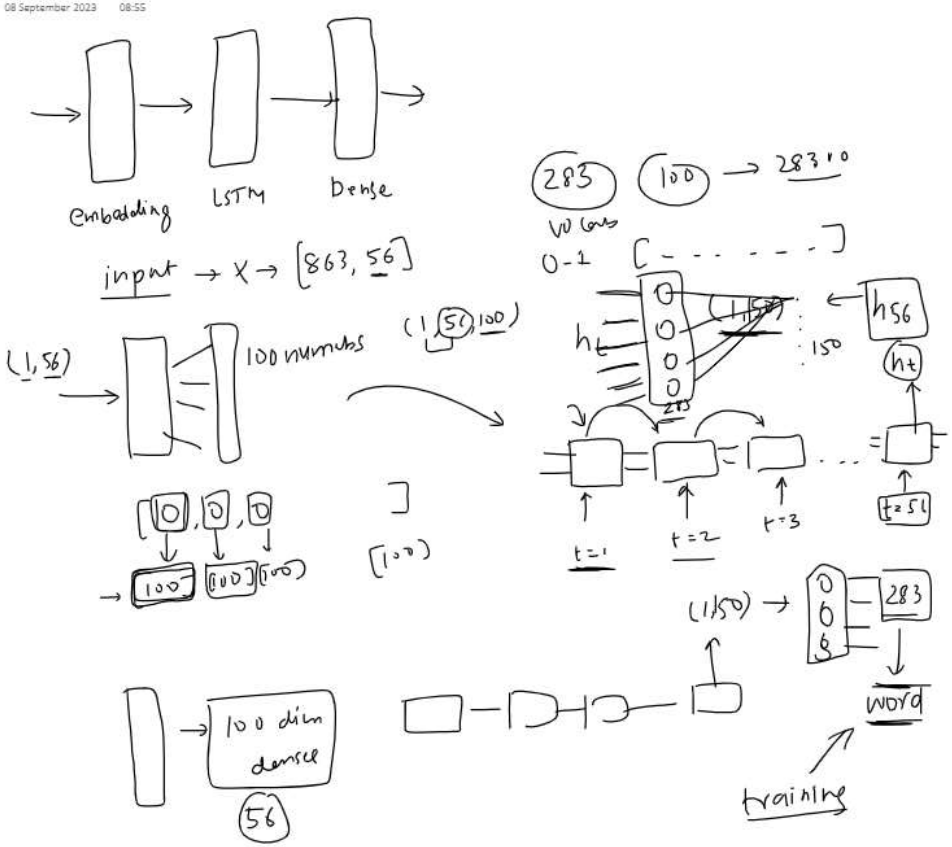
↳ first word

[0 1 0]

↳ second word

The Architecture

08 September 2023 08:55



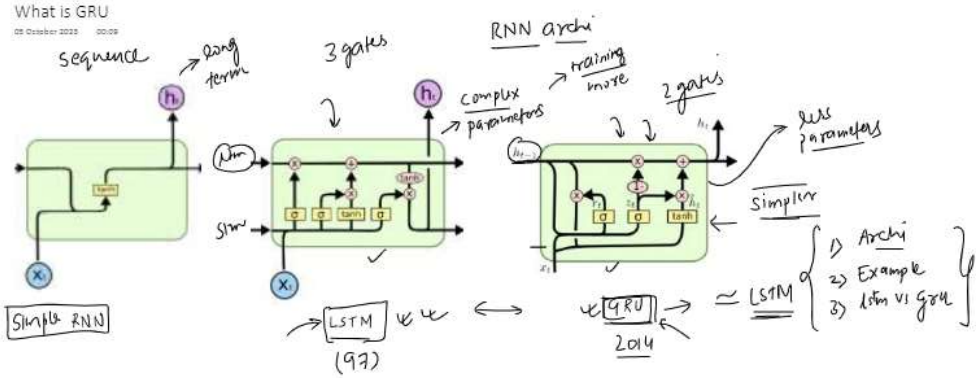
How to improve performance?

08 September 2023 08:51

1) more data → 283
2) hyperparameters →
3) Advanced archs
   - Stacked LSTM
   - Bidirectional LSTM
   - Transformers

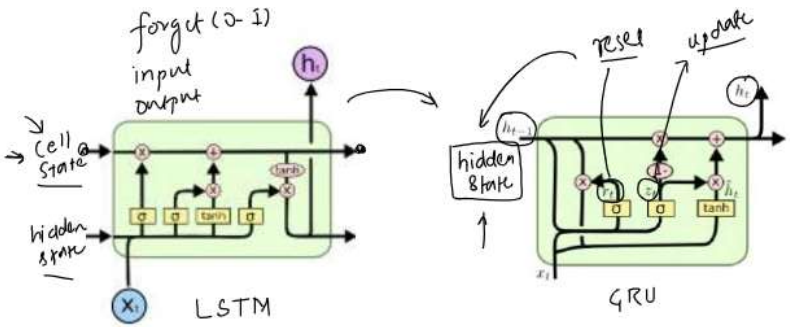
What is GRU

05 October 2023 00:09



The Big Idea Behind GRU

05 October 2023 00:47



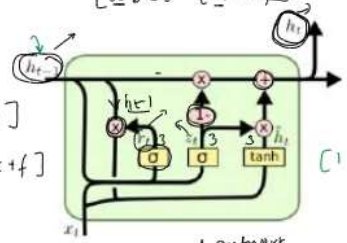
The Setup

05 October 2023 01:07

→ Advise → LSTM/GRU → confusing

goal → $\boxed{t}$

[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

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

h_t → vectors

$\tilde{h}_t$ → candidate hidden state

r_t → reset gate

z_t → update gate

σ tanh ?

[1 2 3 4]

0 0 0 ?

[1 2 3 4] udim same

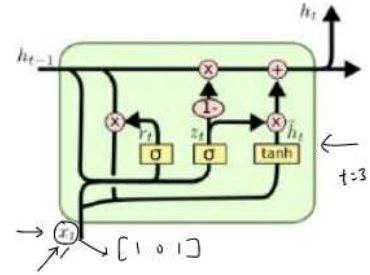
10 number of = 5 6

$\boxed{\sigma}$ → σ σ σ σ σ

The Input Xt

05 October 2023 01:52

t input → vector



t=3

[1 0 1]

sentiment

$\boxed{\text{OHE / BOW / Word2Vec}}$

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

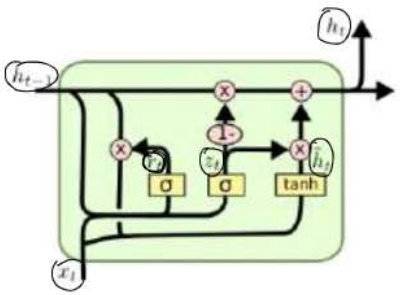
S1 [1 0 0] [0 1 0] [0 0 1]

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



$h_{t-1}, x_t \longrightarrow h_t$

→ 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)

h_t (4)

$h_{t-1} \longleftrightarrow$ (2) $\tilde{h}_t$ → z_t (3)

$h_{t-1} \longleftrightarrow x_t$ → r_t (1)

What exactly is hidden state?

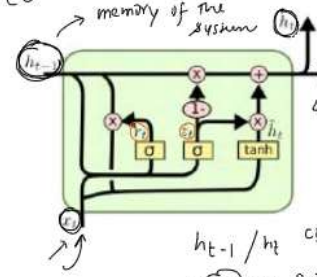
05 October 2023 03:19

$t \rightarrow \boxed{h_{t-1}} \rightarrow \boxed{h_t}$ → vector

memory → context

[0 0 0 0]

memory of the system



$t=2$

$t=3$

$t=4$

4 dim

$t=1$

→ [There was a king Vikram very strong and powerful]

→ • There was an enemy king kaali

→ • Both had a war and kaali killed Vikram

→ • Vikram had a son Vikram Jr who grew up he to become very strong just like his father

→ • He also attacked Kaali But got killed

→ • Vikram Jr too had a son called Vikram super Jr and when he grew up he also fought kaali

→ • And he killed kaali and took revenge of his father and grand father

h_{t-1} → happy/sad

h_{t-1}

→ [Power, Conflict, Tragedy, Revenge]

→ [0.9 0.6 0.9 0.9]

→ [0.9 0 0 0] h_t

→ [1 0.5 0 0] → h_{t-1}

[0.7 0.7 0.4 0]

[0.9 0.6 0.3 0]

[0.8 0.8 0.6 0.5]

[1 0.9 0.5 0.7]

→ [0.7 0.8 0.3 1]

h_{t-1} / h_t context

→ [0.5 0.6 0.3 0.1]

$\boxed{h_{t-1} \rightarrow h_t}$ GRU

↓ 2 steps gate

reset

update

$h_{t-1} \xrightarrow{1} \tilde{h}_t \xrightarrow{2} h_t$

reset gate r_t

candidate memory

update gate z_t

[power, conflict, tragedy, revenge]

→ [0.5, 0.6, 0.7, 0.1] → h_{t-1}

x_t → [0.8, 0.4, 0.5, 0.3] → $\tilde{h}_t$ candidate

inclined x_t

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

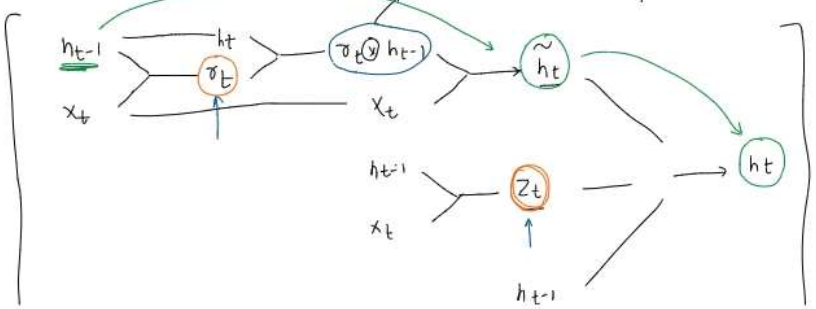
[0.65, 0.5, 0.6, 0.2] → h_t

$\tilde{h}_t$ memory → candidate → h_t

$\boxed{h_{t-1} \xrightarrow{r_t} \tilde{h}_t \xrightarrow{z_t} h_t}$

modulated / resetted hidden state

GRU



h_{t-1}

Calculating the reset gate

08 October 2023 14:24

0.1

σ(weighted sum)

reset

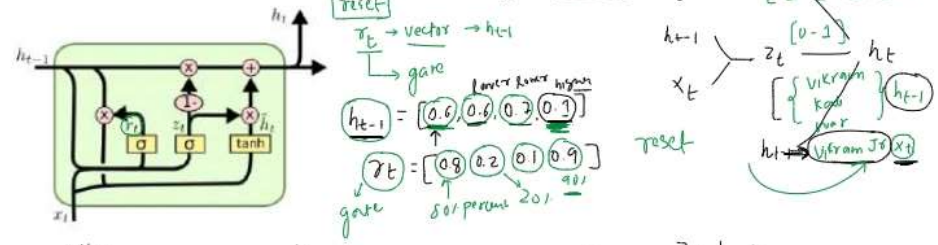
r_t → vector → h_{t-1}

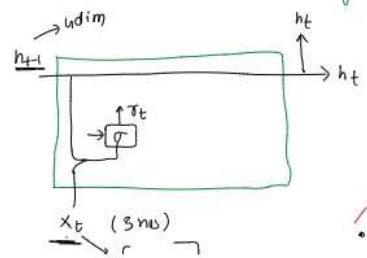
→ gate

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

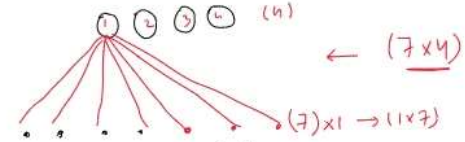
$r_t = [0.8, 0.2, 0.1, 0.9]$

gate 80% percent 20% 90%

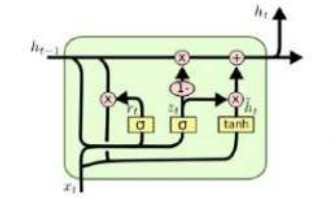




$r_t = \sigma(W_r[h_{t-1}, x_t] + b_r)$



← (7×4) = 28 + 4



h_{t-1} modulated reset

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

(1×7) (7×4)

(1×4) + (1×4) → ((1×4))

(1×4) → 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%

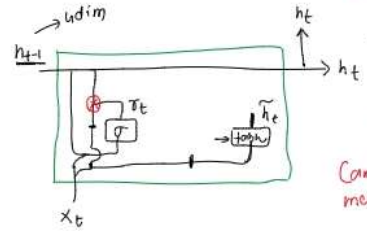
$\tilde{h}_t = [0.7\ 0.2\ 0.1\ 0.2]$

x_t

→ $h_{t-1} \otimes r_t = [0.48, 0.12, 0.07, 0.09]$ (power control troginy reverse) → x_t

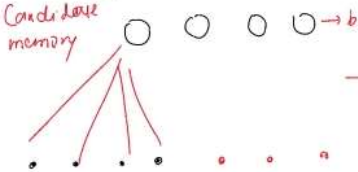
h_{t-1} / $\tilde{h}_t$

z_t



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

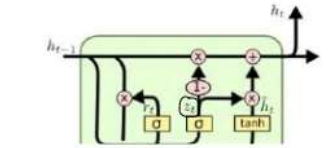
Candidate memory



→ b_c

→ W_c (7×4)

$(h_{t-1} \otimes r_t)$

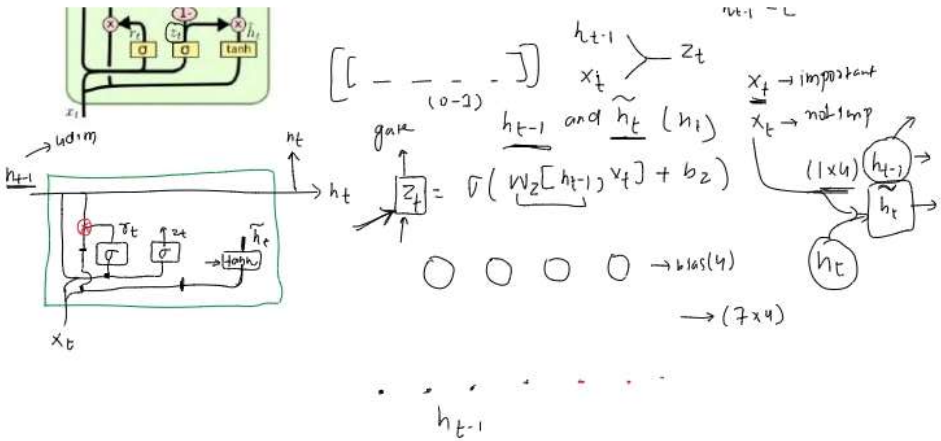


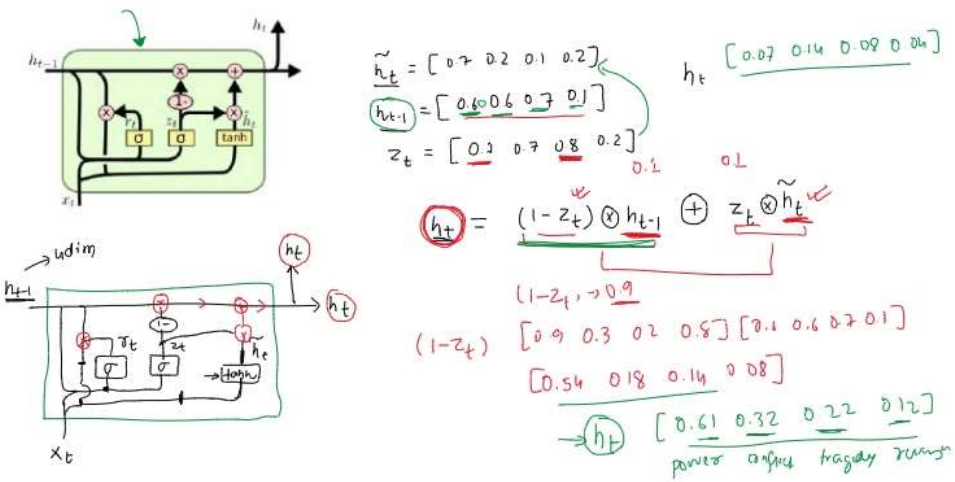
h_{t-1}, r_t → $r_t \otimes h_{t-1}$, x_t → $\tilde{h}_t$

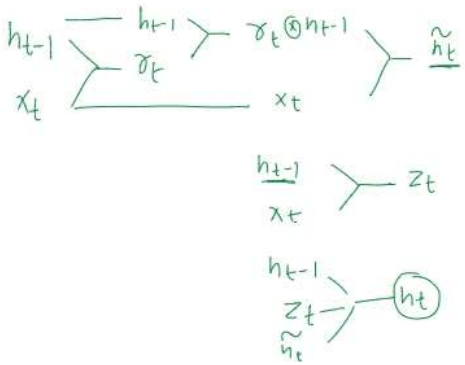
$\tilde{h}_t = [0.7\ 0.2\ 0.1\ 0.2]$

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

h_{t-1}, z_t







LSTM vs GRU

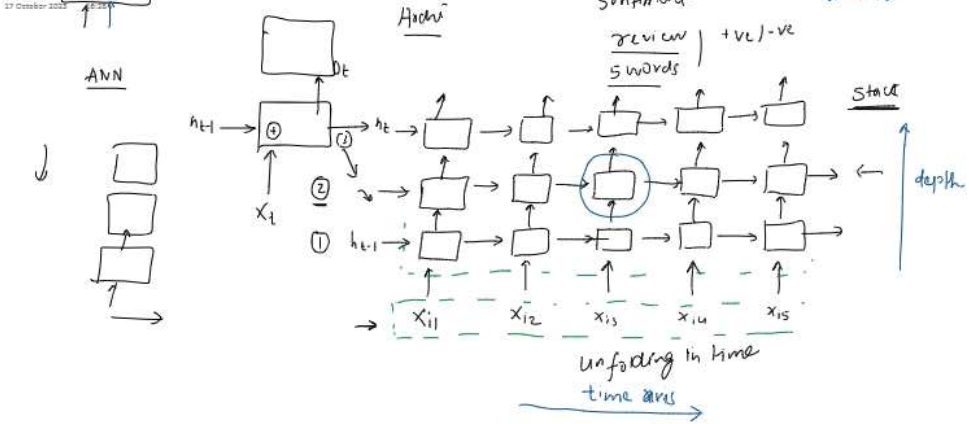
05 October 2023 16:45

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.

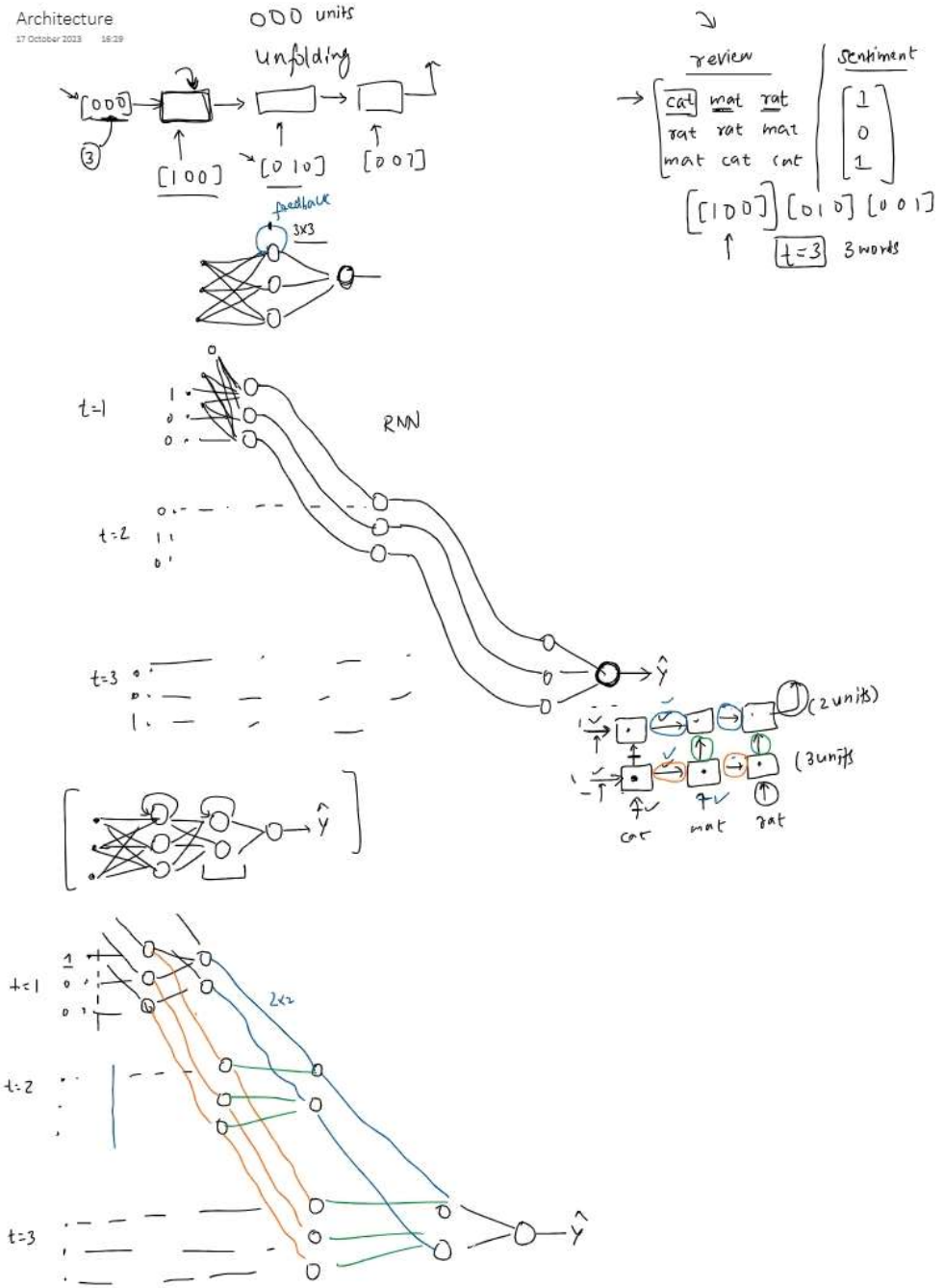
What is Deep RNN

17 October 2023 8:24



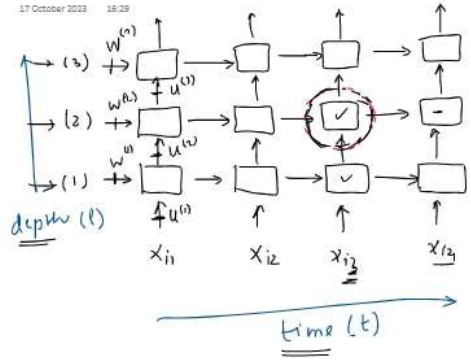
Architecture

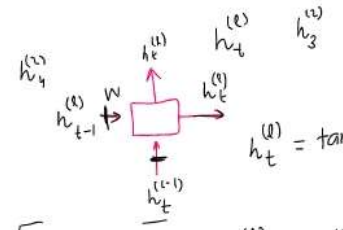
17 October 2023 16:29



Notation

17 October 2023 16:29





$h_t^{(l)} = \tanh($

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

Why and When to use?

17 October 2023 16:29

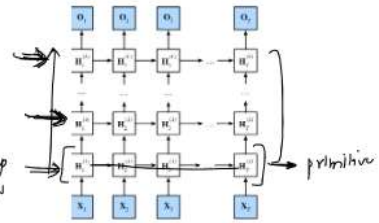
1. Hierarchical Representation ✓
2. Customization for Advanced Tasks

deep RNN

product

Stock

sentence



encoder-decoder

↓

machine

{ deep RNN }

{ love hate amazing terrible

Auto is bad

Display is great

Overall I am happy } → sentiment

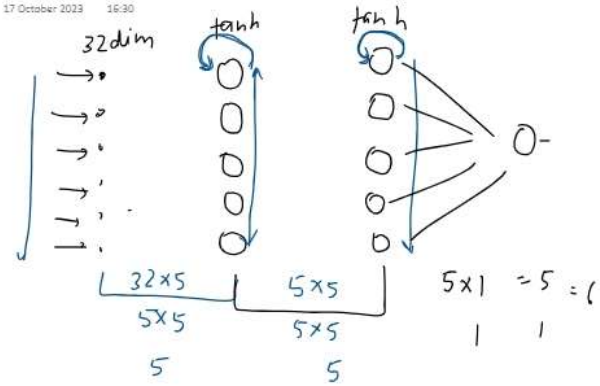
□ →

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 = 320,000

Simple RNN 1

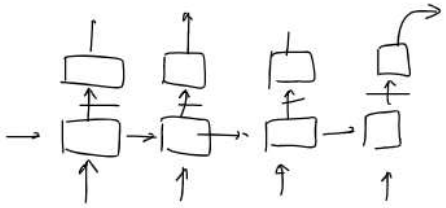
190

Simple RNN 2

55

Dense

6



Variants

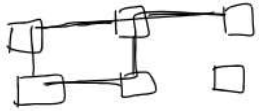
17 October 2023 16:30

SimpleRNN

Deep LSTM

Deep GRU

Keras



[Computation graph]

Propont

Disadvantages

17 October 2023 16:30

Overfitting

Training time

The Why? BiRNN

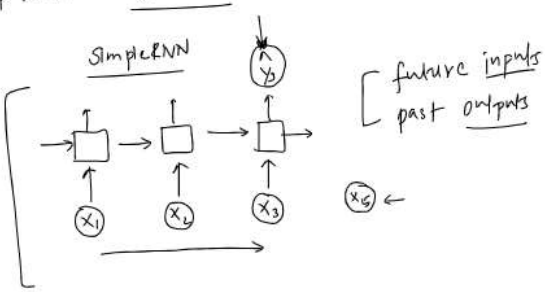
26 October 2023 15:19

RNN → LSTM, GRU

→ Deep RNN → BiRNN

unidirec RNN

SimpleRNN



future inputs

past outputs

① [NER] → Named entity recog

NLP

Nilish → person

Gurgaon → loc/place } → chatbot

ORG / LOC ambiguity

simpleRNN

+ Machine translation →

NER

I love amazon, it's a great website

I love amazon, it's a beautiful river } BiRNN

Bidirectional RNN Architecture

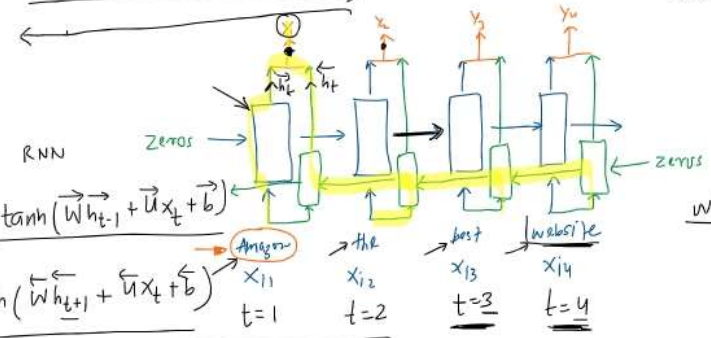
28 October 2023 15:12

Amazon the best website.

Amazon the beautiful river.

loc / org

2 RNNs ✓ forward $\overrightarrow{RNN}$ - $\overleftarrow{RNN}$ concatenate



RNN

$$\overrightarrow{h_t} = \tanh(\overrightarrow{W}\overrightarrow{h_{t-1}} + \overrightarrow{U}x_t + \overrightarrow{b})$$

$$\overleftarrow{h_t} = \tanh(\overleftarrow{W}\overleftarrow{h_{t+1}} + \overleftarrow{U}x_t + \overleftarrow{b})$$

$$\boxed{y_t = \sigma(V[\overrightarrow{h_t}, \overleftarrow{h_t}] + b)}$$

website → org

loc x

Code

26 October 2023 15:21

{ w BiLSTM } BiRNN

↑ Concate

→

←

Applications and Drawbacks

26 October 2023 15:21

- NER
- POS tagging
- Machine Translations
- Sentiment analysis

[Time series forecasting]

→ ←

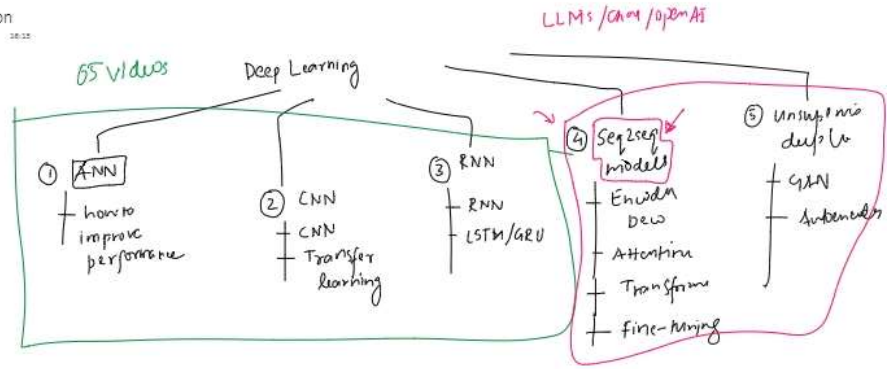
→ Complexity → 190 → 380 double → training → overfitting

→ → ← [Speech recog] → bi rnn

↳ latency → slow

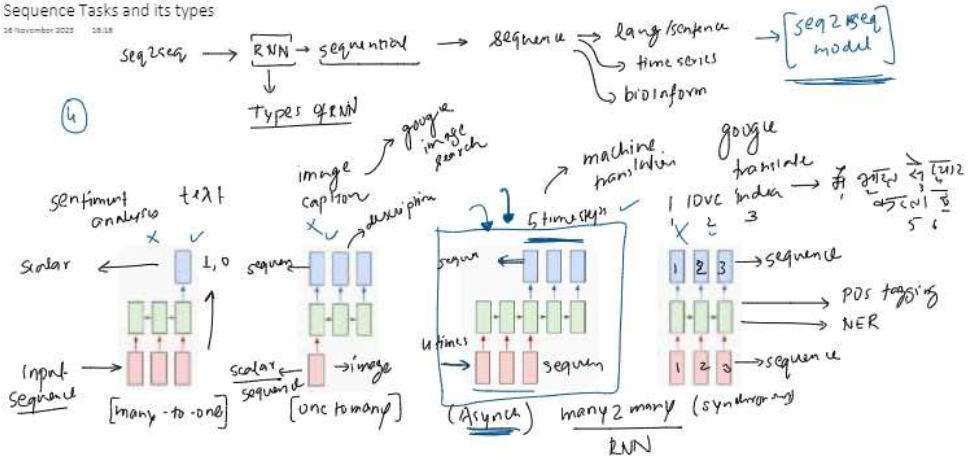
Introduction

16 November 2023 18:15



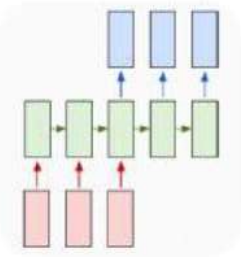
Sequence Tasks and its types

16 November 2023 18:18



Seq2Seq Tasks

16 November 2023 16:16



NLP

Seq2seq → machine trans

1) text summarization → 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

ChatGPT

Encoder decoder (Stage 1) → Attention (Stage 2) → Transformer (Stage 3) → Transfer learning (Stage 4) → LLMs (Stage 5) → ChatGPT

Stage 1 - Encoder Decoder Architecture

18 November 2023 18:18

2014 Seminal

Sequence to Sequence Learning with Neural Networks

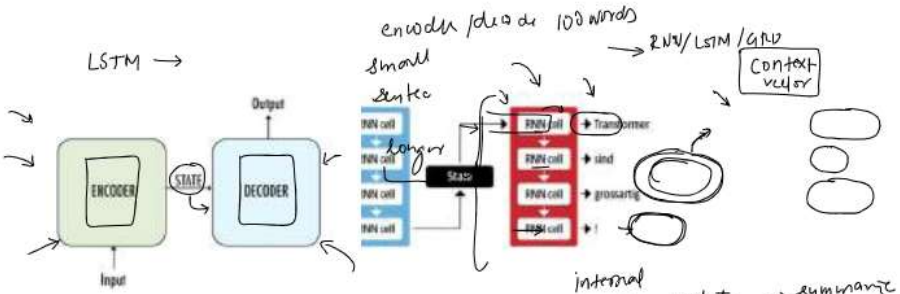
Ilya Sutskever, Google, ilyasu@google.com
Oriol Vinyals, Google, vinyals@google.com
Quoc V. Le, Google, qvl@google.com

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



internal C_t / h_t update → summarize

seq2seq → input → output

machine translation

eng → hindi

compressed repre

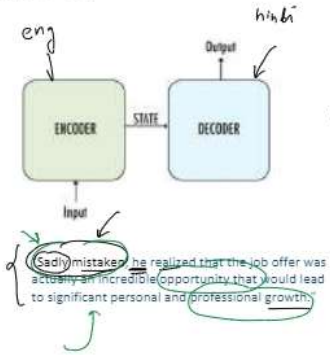
Attention mechanism

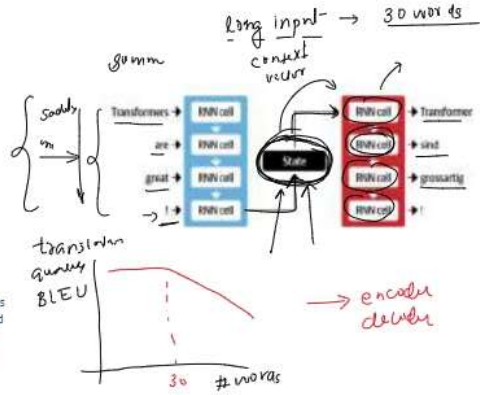
← [compress] ← process word by word

[I love India] → 나는 인도를 사랑해

Stage 2 - Attention Mechanism

20 November 2023 10:53





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

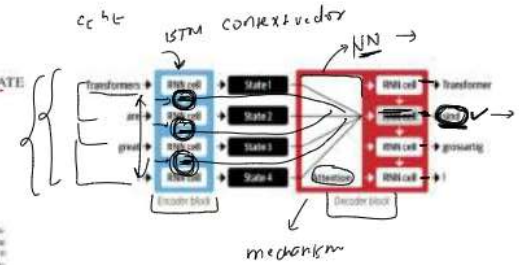
Neural machine translation is a recently proposed approach to machine translation. Unlike the traditional statistical machine translation, the neural machine translation aims at building a single neural network that can be jointly tuned to maximize the translation performance. The models proposed recently 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 analyses reveals that the (soft-)alignments found by the model agree well with our intuition.

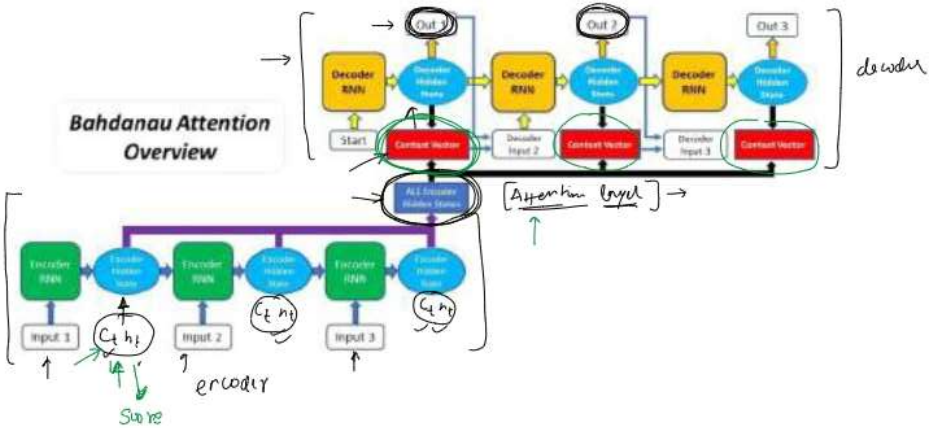
1 Introduction

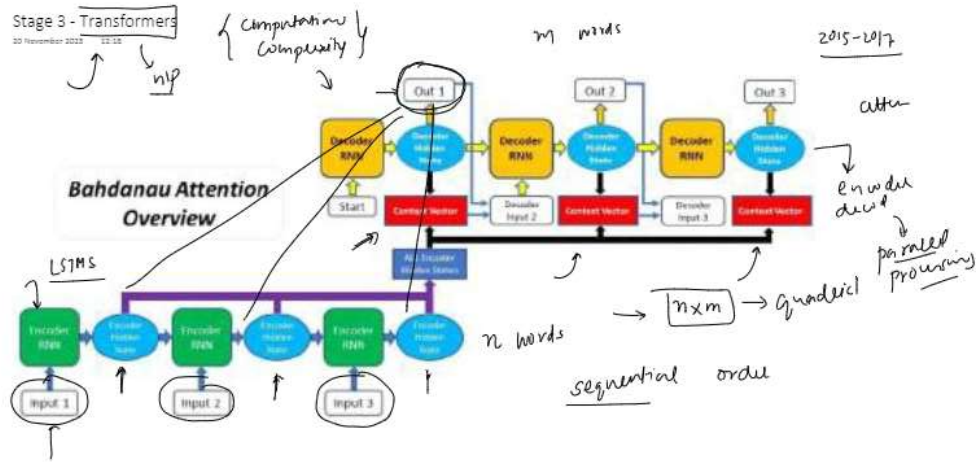
Neural machine translation is a newly emerging approach to machine translation, recently proposed by Kalchbrenner and Blunsom (2013), Sutskever et al. (2014) and Cho et al. (2014a). Unlike the traditional phrase-based translation system (see, e.g., Koehn et al., 2003) which consists of many small sub-components 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.

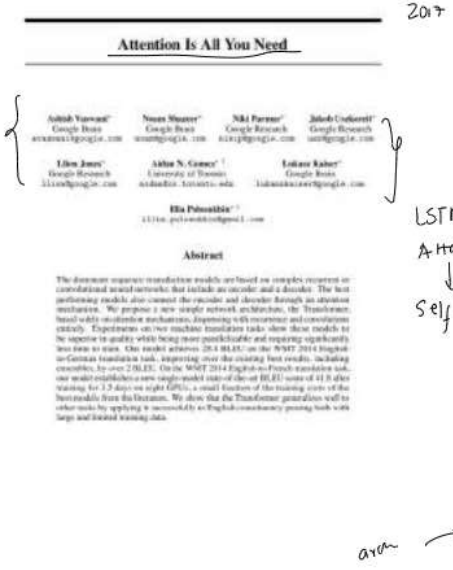
Most of the proposed neural machine translation models belong to a family of encoder-decoders (Sutskever et al., 2014; Cho et al., 2014a), with an encoder and a decoder for each language, or involve a language-specific encoder applied to each sentence whose outputs are then compared (Hermans and Blunsom, 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.









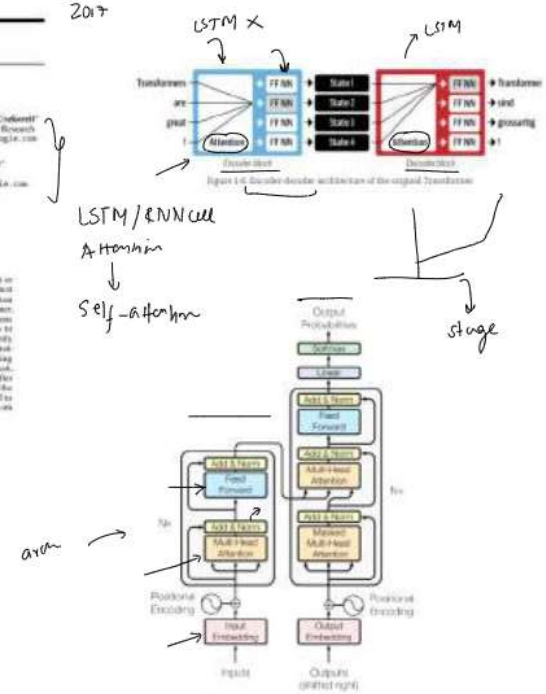


Figure 1: The Transformer - model architecture.

Stage 4 - Transfer Learning

20 November 2023 15:59

1) Hardware → GPUs
2) Time →
3) Data → a lot

→ 200, 1000

↓

transformer → Transfer learning

ULMFiT

Universal Language Model Fine-tuning for Text Classification

Jeremy Howard* — fast.ai, University of San Francisco

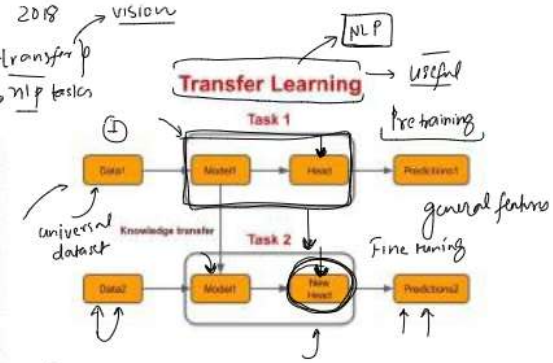
Sebastian Ruder* — Insight Centre, NUI Galway; Aylien Ltd., Dublin

ULMFit

primary pretrain

2018 → vision

transfer → NLP tasks



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.

imagenet → dataset → Millions of
↳ CNN → ResNet
↳ pretrain → ↳ image
edges ↳ shapes
↓
fine-tune → cat vs dog classif → 100

2018

1) Task specificity → NLP
SentA, NER, POS, ML, QA, summ
language

NLP → transfer learning

ML →
2nd) Lack of data

eng	hin

NLP → transfer learning

Pretraining → Machine translation X
↳ language modelling → pretraining → successful → (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 pretra. task

1) Rich feature learning
The hotel was exceptionally clean, yet the service was bad/pathetic

→ Know trans
↓
text class / quest. | text summ | NER / POS

mt (label → supervised labeled
eng | hin → unsupervised task

2) Huge avail of data
pdf → dataset
labelling

fine tuning

[ULMFit]

X transformer

AWD LSTM → wikipedia → [ ] → imdb, yelp, new data

classifier

unsupervised pretrain language modelling

Scratch → 10000 rows
100 row → better →

→ model
↓
test
↓
State of the art

2012 2018

2012 → parallelly

2018 → 2018

archi → [transformer]

train → [transfer learning] → ?

Stage 5 - LLMs

22 November 2025 18:51

2018 → 20 ULMFit Jan out [Language]

Transform

Google → BERT [Encoder only]

OpenAI [decoder only model] → GPT

transformer

dataset → huge

↳ Transfer learning → text summ, ques, sentim, ner, pos

downsd

limited dataset

finetune

SOTA

GPT → GPT2 → GPT3 → language model → LLM → LLMs

↳ parameters

↳ dataset

[LM → LLMs]

Large scale LLMs

Qualities of LLMs

1) Data → billions → GPT3 → 45 TBs

↳ book, websites, internet

↳ diversity → bias

2) Hardware → Cluster of GPU → GPT3 → Supercomp → 1000s NVIDIA GPU

3) training → days to weeks

4) Cost → hardware + elect + infra + experts → individual

↳ millions

- companies
- govt
- institutes

4) energy consump (gpt3) → . . . . . .

4) energy consump → gpt3 → small town → month

The Grand Finale - ChatGPT

22 November 2023 10:13

GPT3 → chatGPT → [diff]

GPT → model

[ChatGPT] → application → Chatbot

NOV

hp → chatgpt

intel → GPT → Bard, Jasper

→ dell

→ AMD

GPT3 → [ChatGPT]

1) RLHF → Reinf learn from human feed
- 1 Supervised Fine tuning → dataset } labelled
- 2 Reinforce → prompt produces
  - responses
  - human → respone rank

2) Incorporate safety and ethical guideline
- minize biases

3) improvement in contextual point
- context retain → maintain context } dialogue convers

4) Dialogue specific training
- conversation
- better under → dialogue lang → patterns

5) ChatGPT continuous imp → human feedback → usu

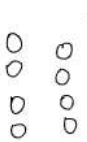
train → refining → GPT4 → GPT3

train → training × GPT4 → [illegible]

Seq2Seq Data

08 December 2023 17:12

1) Tabular → [ANN] × (DNN)



gpu | ia | placement

2) Image data → [2d grid] → CNN ×

dog
cat

3) Sequential → textual
→ timeseries

RNN → LSTMs, GRUs

Encoder Decoder Arch

4) Seq2seq data — difficult

[input seq → output seq] — machine translation

मेरा नाम मोहित है → आप से मिल कर अच्छा लगा

input → sent → eng → variable length

output → sent → hindi → variable leng → variable length → [lstm/gru] → input [output]

3 words eng → 3 words

Before Starting

08 December 2023 19:24

Prerequisite

- RNN / LSTM

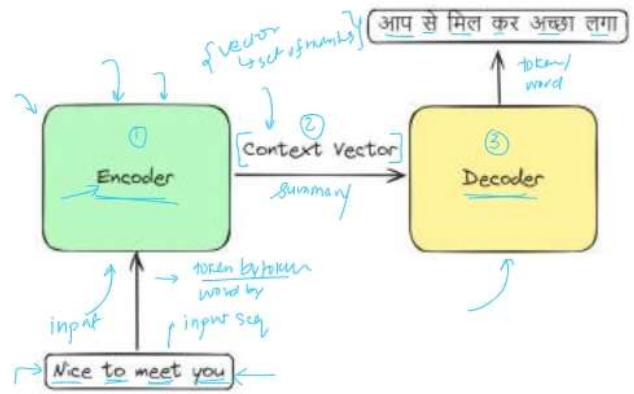
Plan of attack

→ simple version
→ deep
→ improvements

High Level Overview

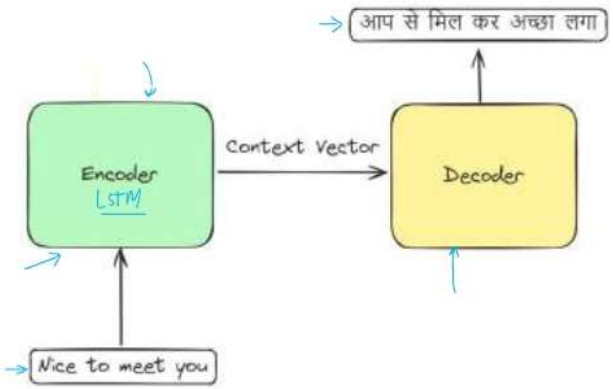
08 December 2023 17:34

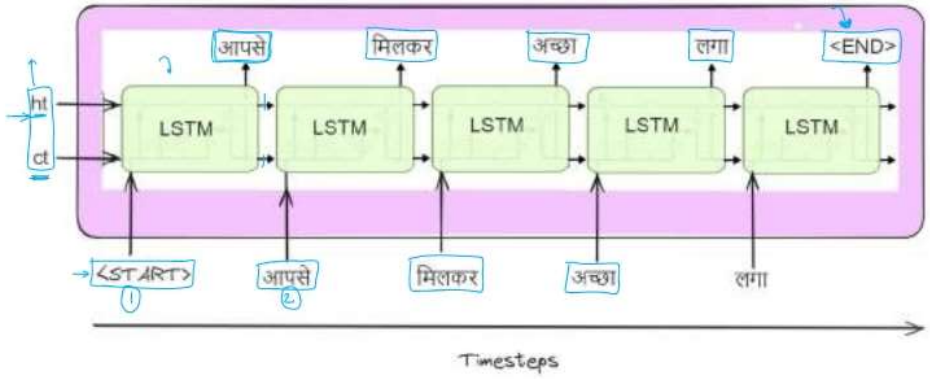
Machine tran → exam

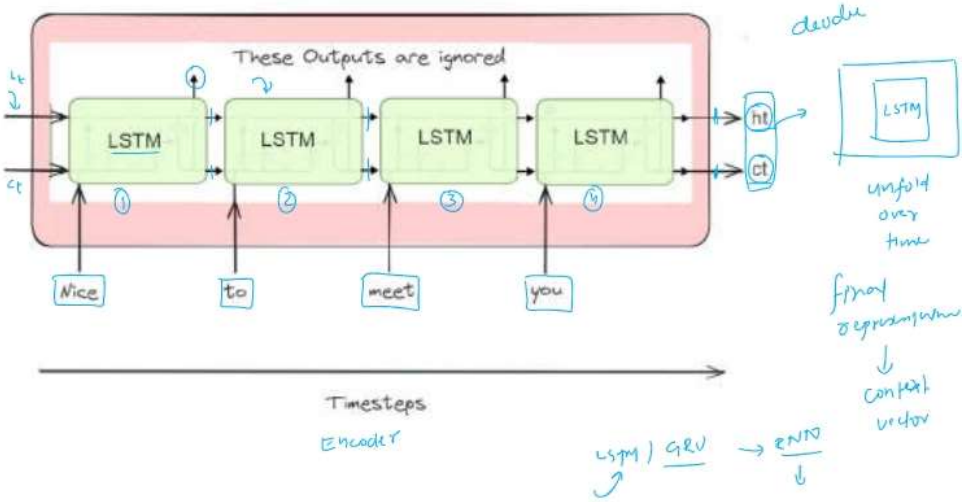


What's under the hood?

08 December 2023 17:34

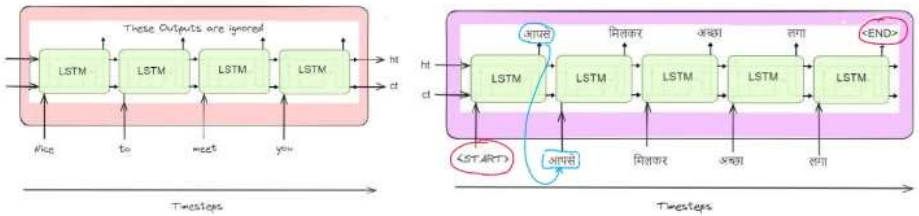


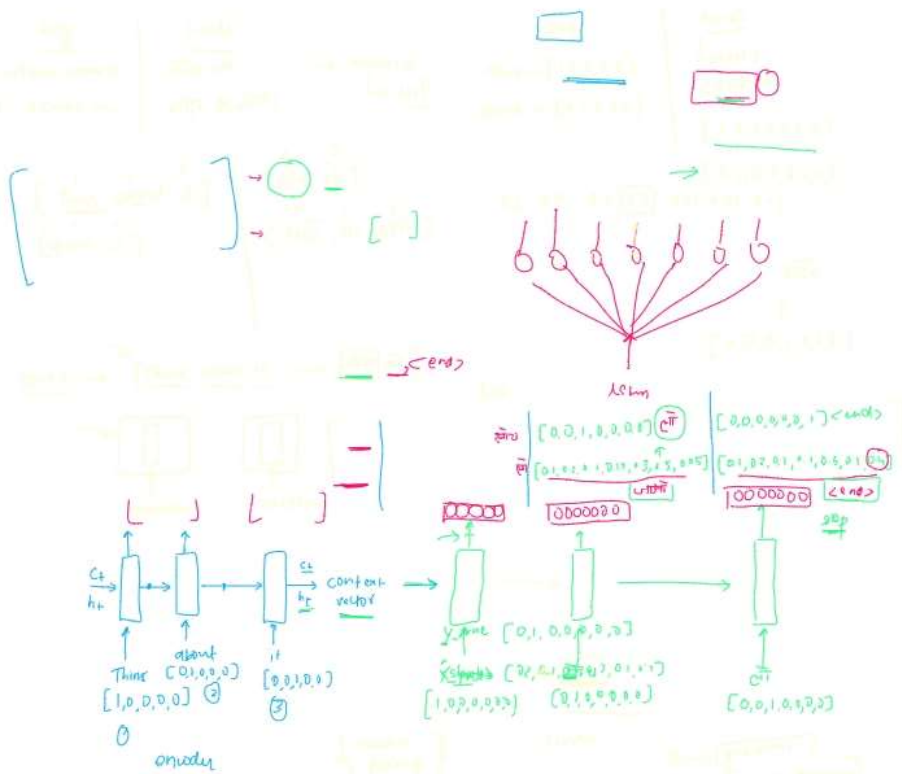




Training the Architecture using Backpropagation

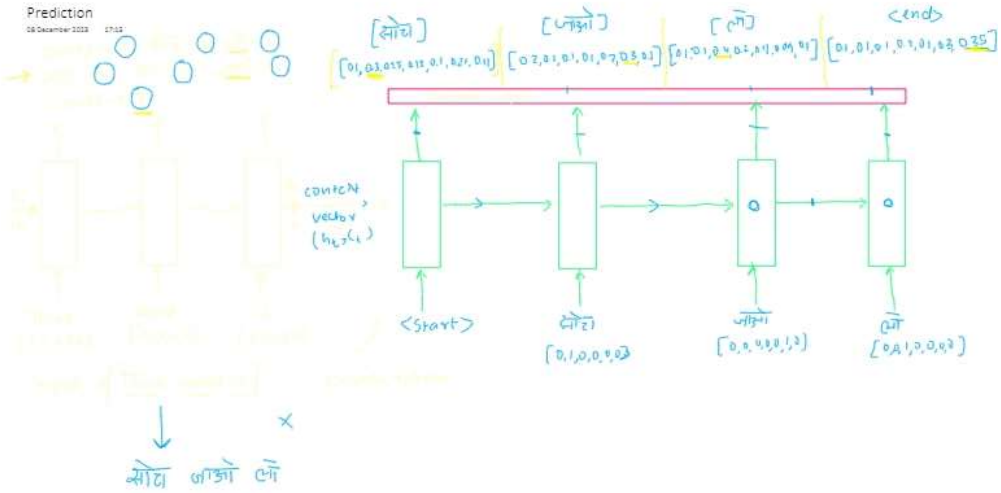
08 December 2023 17:15





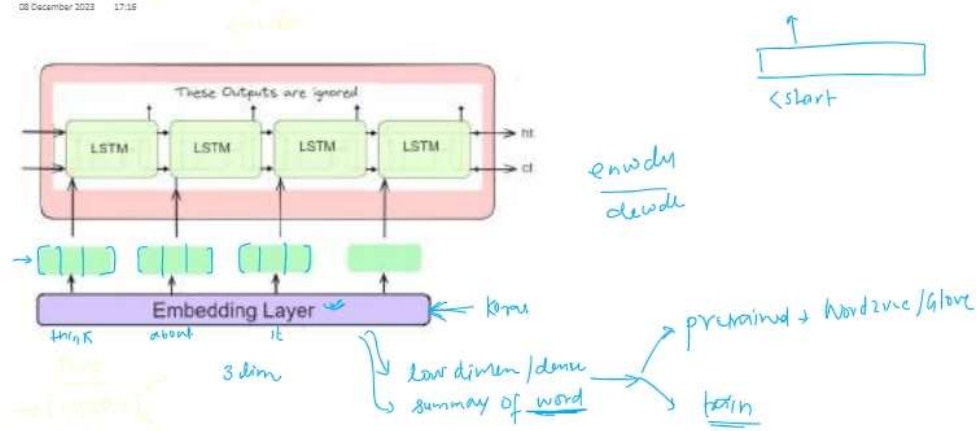
Prediction

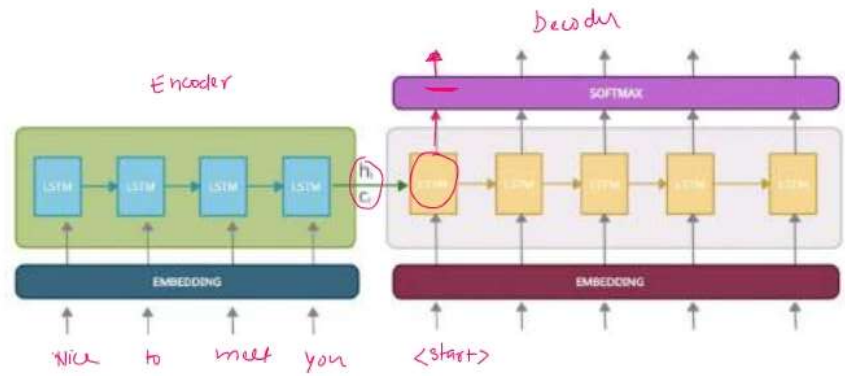
08 December 2023 17:12



Improvement 1 - Embeddings

08 December 2023 17:16

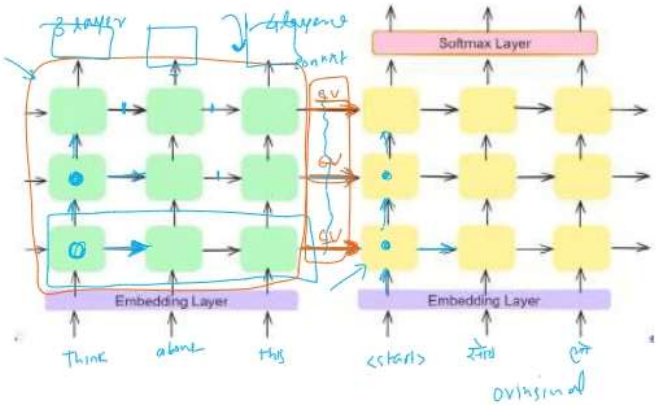




Improvement 2 - Deep LSTMs

08 December 2023 17:18

long para single layer



3) deep learning NN learning

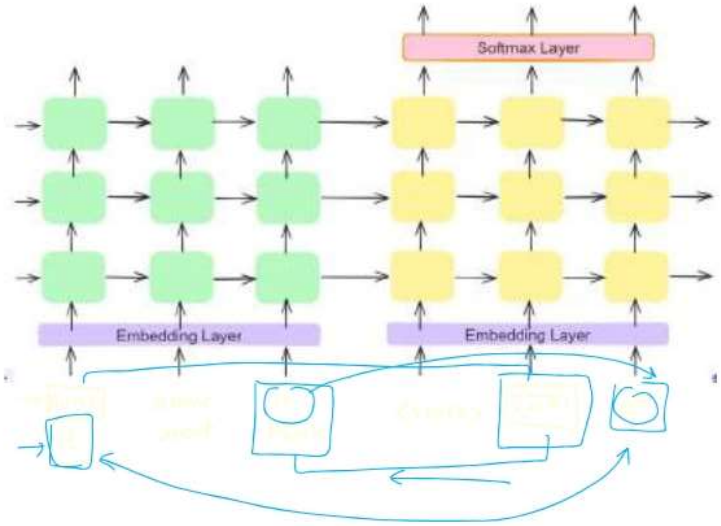
1) long term dependency

2) layered representation

the phone battery is bad but the overpriced is great

Improvement 3 - Reversing the Input

08 December 2023 17:16



language

↓

initial context

eng - french

The Sutskever Architecture

08 December 2025 17:18

Application to Translation: The model focused on translating English to French, demonstrating the effectiveness of sequence-to-sequence learning in neural machine translation.

Special End-of-Sentence Symbol: Each sentence in the dataset was terminated with a unique end-of-sentence symbol ("<EOS>"), enabling the model to recognize the end of a sequence.

Dataset: The model was trained on a subset of 12 million sentences, comprising 348 million French words and 304 million English words, taken from a publicly available dataset.

Vocabulary Limitation: To manage computational complexity, fixed vocabularies for both languages were used, with 160,000 most frequent words for English and 80,000 for French. Words not in these vocabularies were replaced with a special "UNK" token.

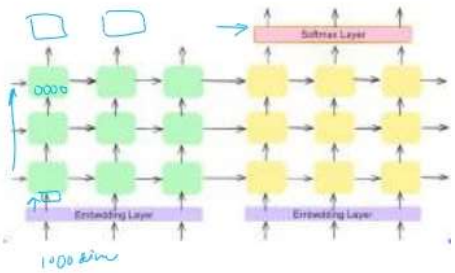
Reversing Input Sequences: The input sentences (English) were reversed before feeding them into the model, which was found to significantly improve the model's learning efficiency, especially for longer sentences.

Word Embeddings: The model used a 1000-dimensional word embedding layer to represent input words, providing dense, meaningful representations of each word.

Architecture Details: Both the input (encoder) and output (decoder) models had 4 layers, with each layer containing 1,000 units, showcasing a deep LSTM-based architecture.

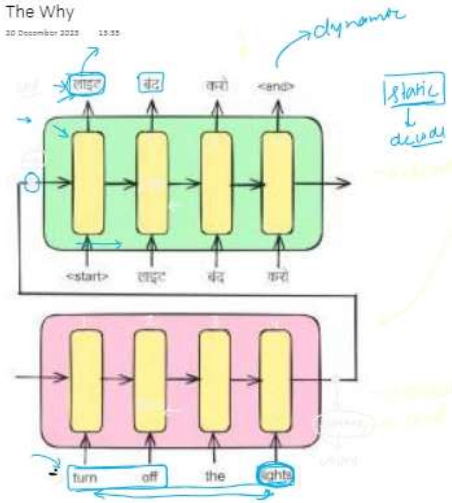
Output Layer and Training: The output layer employed a Softmax function to generate the probability distribution over the target vocabulary. The model was trained end-to-end with these settings.

Performance - BLEU Score: The model achieved a BLEU score of 34.81, surpassing the baseline Statistical Machine Translation (SMT) system's score of 33.30 on the same dataset, marking a significant advancement in neural machine translation.



The Why

30 December 2025 15:55



Encoder

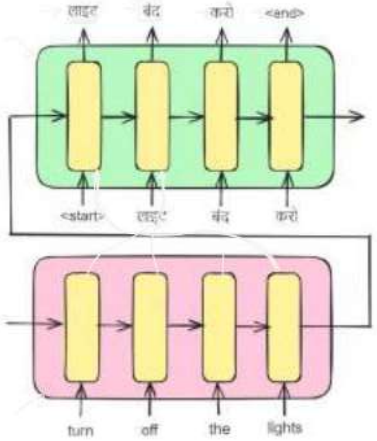
Once upon a time in a small Indian village, a mischievous monkey stole a turban from a sleeping barber, wore it to a wedding, danced with the bewildered guests, accidentally got crowned the 'Banana King' by the local kids, and ended up leading a vibrant, impromptu parade of laughing villagers, cows, and street dogs, all while balancing a stack of mangoes on its head, creating a hilariously unforgettable spectacle and an amusing legend that the village still chuckles about every monsoon season.

Decoder

The Solution

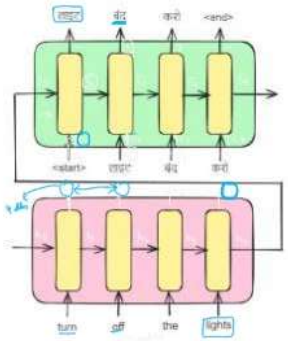
20 December 2023 17:32

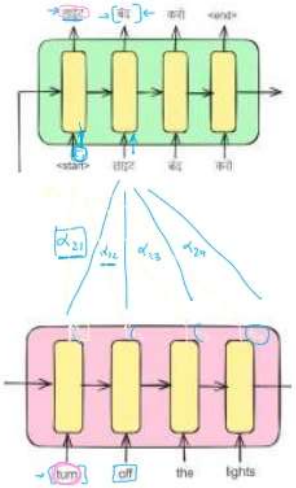
Once upon a time in a small Indian village, a mischievous monkey stole a turban from a sleeping barber, wore it to a wedding, danced with the bewildered guests, accidentally got crowned the 'Banana King' by the local kids, and ended up leading a vibrant, impromptu parade of laughing villagers, cows, and street dogs, all while balancing a stack of mangoes on its head, creating a hilariously unforgettable spectacle and an amusing legend that the village still chuckles about every monsoon season.

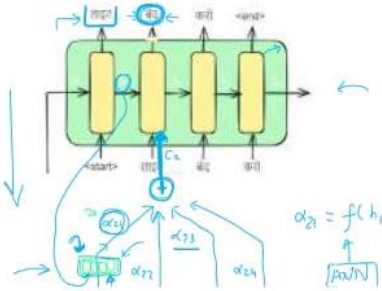


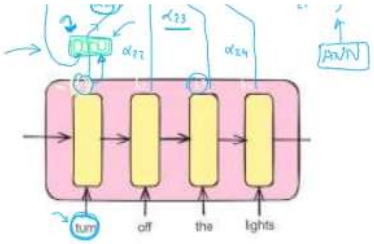
The What

11 December 2023 04:01









$\alpha_{21} \rightarrow f(h_1, s_1)$

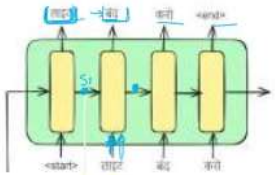
$\alpha_{ij} = f(h_j, s_{i-1})$

$\alpha_{23} \rightarrow f(h_3, s_1)$

ANN

unique function approx.

what function?

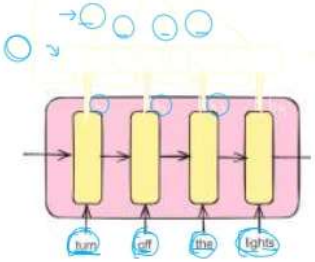


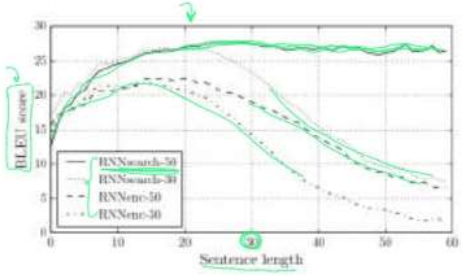
→ 16

	turn	off	the	lights
α_{11}			α_{13}	α_{14}
α_{12}	α_{13}	α_{22}		

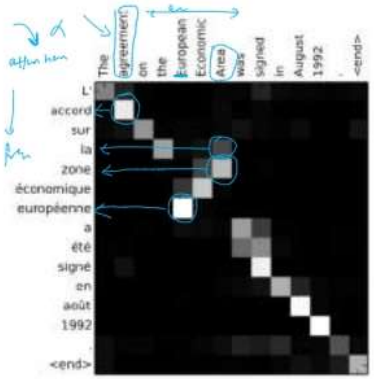
end

$\boxed{c_2} = \alpha_{21}h_1 + \alpha_{22}h_2 + \alpha_{23}h_3 + \alpha_{24}h_4$



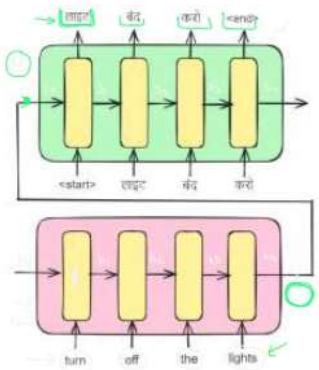


eng-fren



(a)

Recap



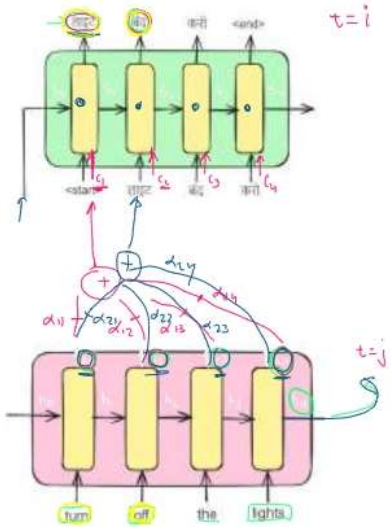
sentence > 30 words

paragraph

document

translation

bottleneck → Attention mechanism



c_1 c_2 c_3 c_4

weighted sum

$$c_i = \sum_{j=1}^{4} \alpha_{ij} h_j$$

α → alignment score

$4 \times 4 = 16$

$c_1 = \alpha_{11}h_1 + \alpha_{12}h_2 + \alpha_{13}h_3 + \alpha_{14}h_4$

$c_2 = \alpha_{21}h_1 + \alpha_{22}h_2 + \alpha_{23}h_3 + \alpha_{24}h_4$

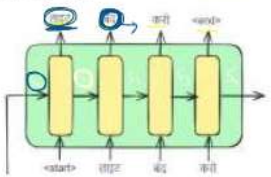
α → find out

Bahdanau additive attention

Luong attention

Bahdanau Attention

16 January 2024 16:11



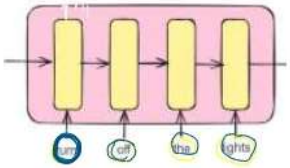
given

α

alignment

$\alpha_{11} \rightarrow$ [illegible] $\rightarrow$ turn

$\alpha_{12} \rightarrow$ [illegible] $\rightarrow$ off



$\alpha_{11} = f(h_1, s_0)$

approximate

FFN $\rightarrow$ ANN $\rightarrow$ UFA

α_{ij}

softmax

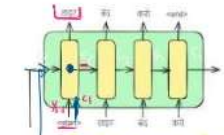
e_{ij}

s_{i-1} h_j

s_{i-1} h_j FFN e_{ij} softmax $\rightarrow$ α_{ij} $\rightarrow$ c_i $\rightarrow$ Y_t

So Y_{t-1} c_1 $\rightarrow$ dshm $\rightarrow$ Y_t (लाइट) S_1

$i=1$ $t=2$



$\hookrightarrow h_1\ h_2\ h_3\ h_4$

S_{i-1} h_j

$c_1 = \sum \alpha_{1j} h_j$

$[e_{11}\ e_{12}\ e_{13}\ e_{14}] \rightarrow$ softmax $\rightarrow [\alpha_{11}\ \alpha_{12}\ \alpha_{13}\ \alpha_{14}]$

$\leftarrow$ 1×1 $c_1 = \alpha_{11} h_1 + \alpha_{12} h_2 + \alpha_{13} h_3 + \alpha_{14} h_4$

$\leftarrow$ 1×1 $c_1 = \alpha_{11} h_1 + \alpha_{12} h_2 + \alpha_{13} h_3 + \alpha_{14} h_4$

$\rightarrow$ 8×4

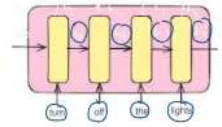
batch operation 4×8 8×3 $\rightarrow$ 4×3

ran(4×3)

α ($s_0 = [e\ f\ g\ h]$

$[s_{i-1}; h_j]$

$h_1\ h_2\ h_3\ h_4$



4 rows / 8 cols

8 cols

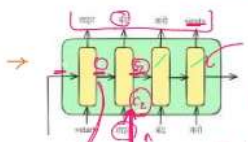
α 4 rows

$$\begin{bmatrix} s_{01} & s_{02} & s_{03} & s_{04} \\ s_{01} & s_{02} & s_{03} & s_{04} \\ s_{01} & s_{02} & s_{03} & s_{04} \\ s_{01} & s_{02} & s_{03} & s_{04} \end{bmatrix} \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \\ h_{41} & h_{42} & h_{43} & h_{44} \end{bmatrix}$$

$(4\times3)(3\times1)$

$\left[\frac{e^{e_{11}}}{e^{e_{11}}+e^{e_{12}}+e^{e_{13}}+e^{e_{14}}}\right]$ $(4\times1) \rightarrow$ 4 numbers

time lobul



$c_2 = \alpha_{21} h_1 + \alpha_{22} h_2 + \alpha_{23} h_3 + \alpha_{24} h_4$

α

$$\begin{bmatrix} S_{11} & S_{12} & S_{13} & S_{14} \\ S_{11} & S_{12} & S_{13} & S_{14} \\ S_{11} & S_{12} & S_{13} & S_{14} \\ S_{11} & S_{12} & S_{13} & S_{14} \end{bmatrix} \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \\ h_{41} & h_{42} & h_{43} & h_{44} \end{bmatrix}$$

4×8

time distributed FFN

softmax

$[\alpha_{21}\ \alpha_{22}\ \alpha_{23}\ \alpha_{24}]$

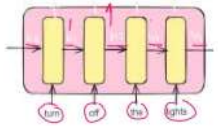
4×1 $\rightarrow [e_{21}\ e_{22}\ e_{23}\ e_{24}]$

4×3

alignment

weight

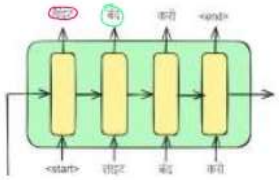
1×8

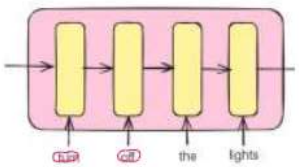


additive attention

Luong Attention

17 January 2024 00:09





parameters → slow

$c_i = \sum \alpha_{ij} h_j$ → FFN $\{[V \tan(W[s_{i-1}, h_j] + b)]\}$

$\alpha_{ij} = f(s_{i-1}, h_j)$ ×

$\alpha_{ij} = f(s_i, h_j)$ → $[s_i^T h_j]$ → dot product

current ① diff

up dated info → dynam adjust

$s_i = [a\ b\ c\ d]$

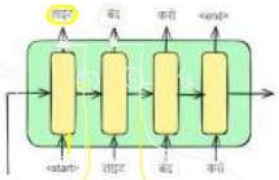
$h_j = [e\ f\ g\ h]$

$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} [e\ f\ g\ h]$

$[ae + bf + cg + dh]$

α_{ij}

softmax ← $\boxed{e_{ij}}$ ← score → attention



output

softmax

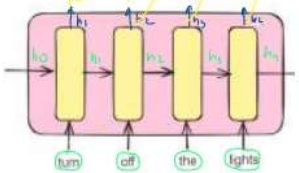
$[e_{21}\ e_{22}\ e_{23}\ e_{24}]$ softmax → $\alpha_{21}\ \alpha_{22}\ \alpha_{23}\ \alpha_{24}$ → c_2

$s_i h_1$ $s_i h_2$ $s_i h_3$ $s_i h_4$

$[e_{11}]$ e_{12} e_{13} e_{14}

α_{11} α_{12} α_{13} α_{14} → $\boxed{c_1}$

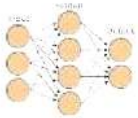
$\sum \alpha_{ij} h_j$



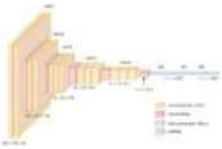
What is Transformer?

27 January 2024 18:41

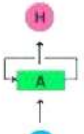
→ self attention scalable



ANN
tabular



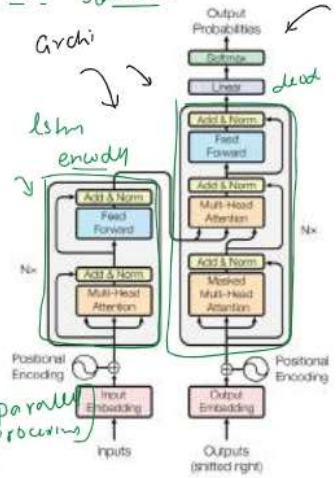
CNN
image



RNN
sequence
↳ text

Grchi

lstm
encoder



→ [transformer] → seq2seq task

NN arch

- machine translation
- question ans
- text summary

[parallel processing]

[Attention Is All You Need]

2017

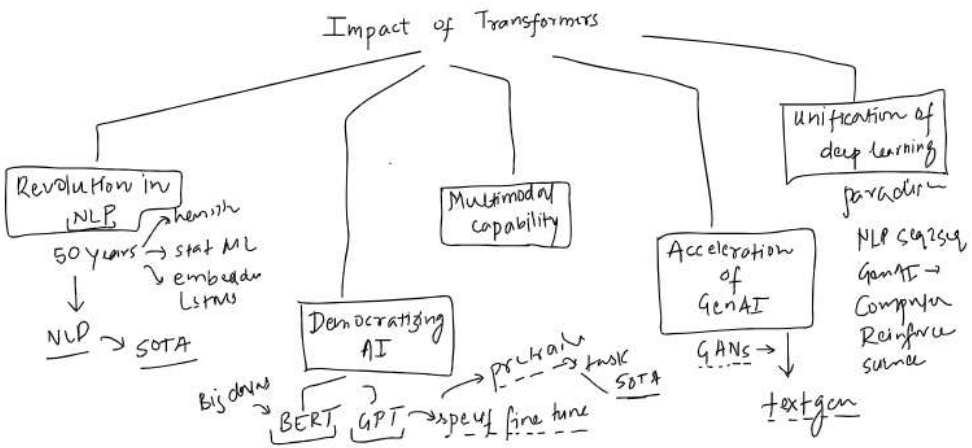
↘ Google Brain

Deep learning

Abstract

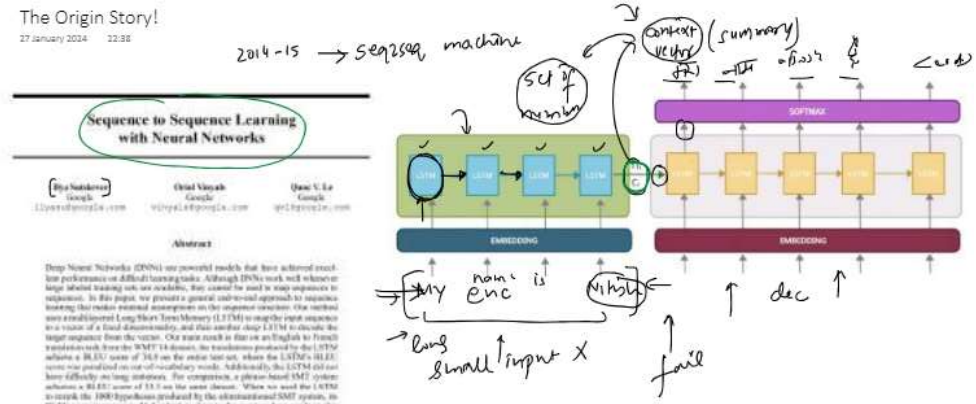
Impact of Transformers

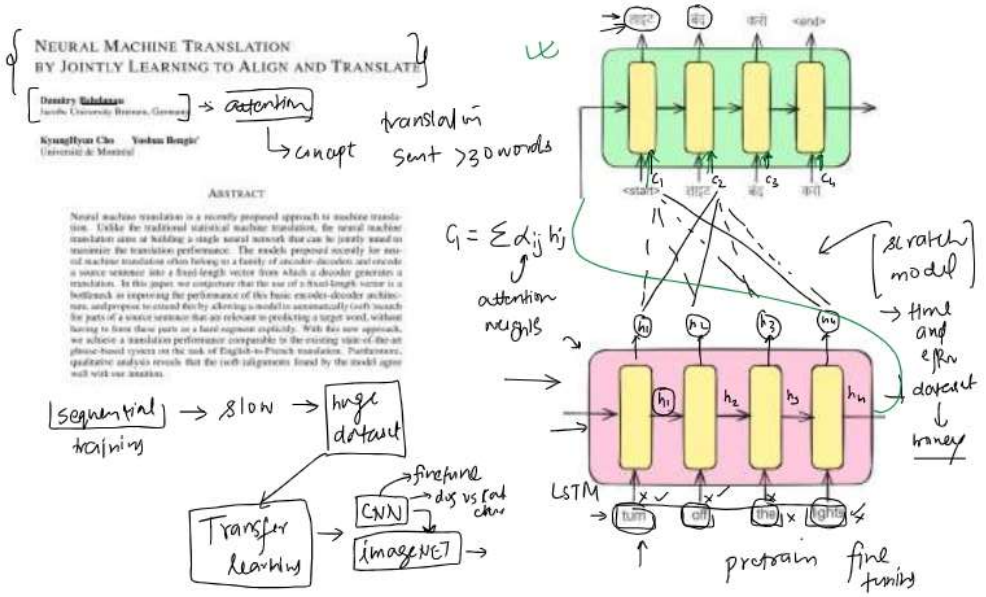
27 January 2024 20:03



The Origin Story!

27 January 2024 22:38

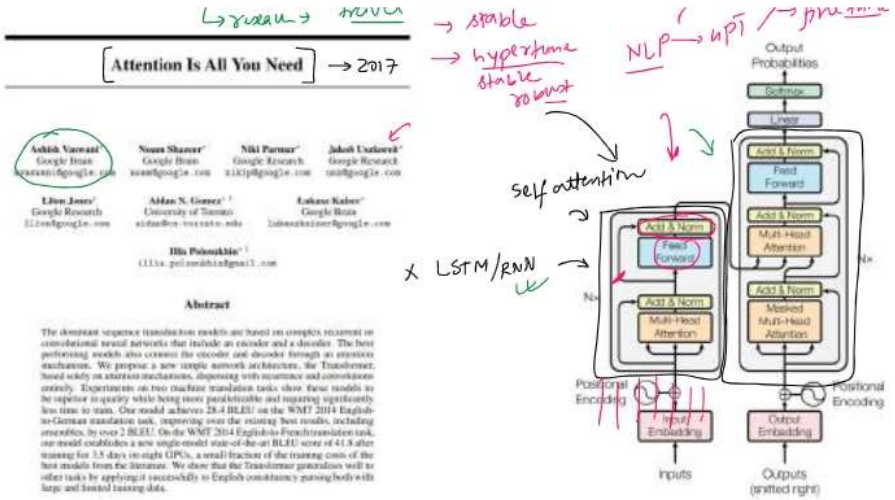




research → incremental time travel
→ RNN/self → parallely
→ stable
→ hypertune

NLP → BERT, GPT → fine tune

Attention Is All You Need → 2017



The Timeline

28 January 2024 00:55

2000 - 2014 → RNNs / LSTMs

↓

2014 → Attention

↓

2017 → Transformer

↓

2018 → BERT / GPT (Transfer learning)

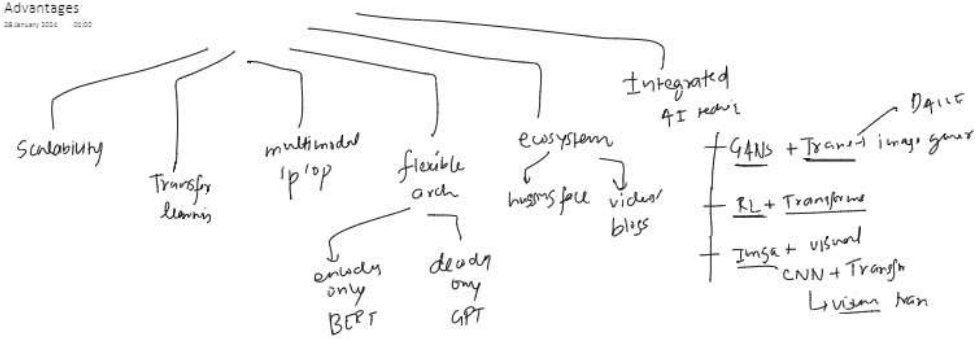
2018 - 2020 → Vision Transfor / Alphafold-2

2021 → Gen AI

2022 → ChatGPT / Stable Diffusion

Advantages

28 January 2024 01:00

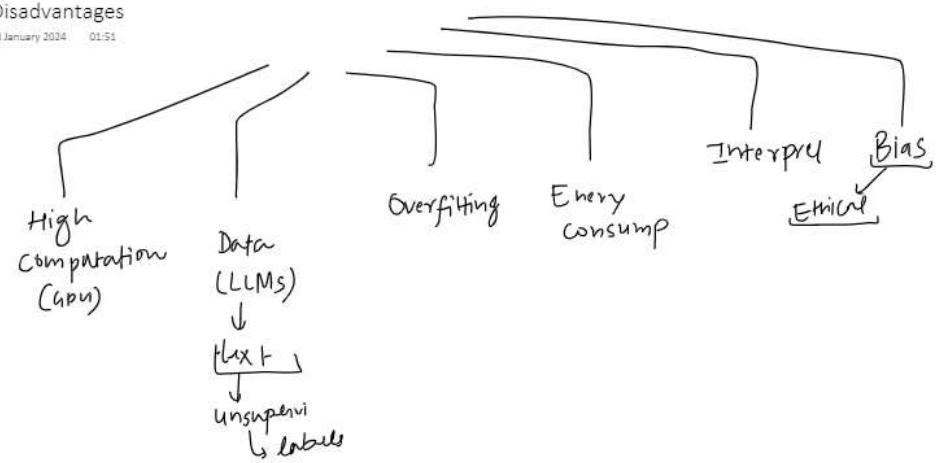


Famous Applications

28 January 2024 01:17

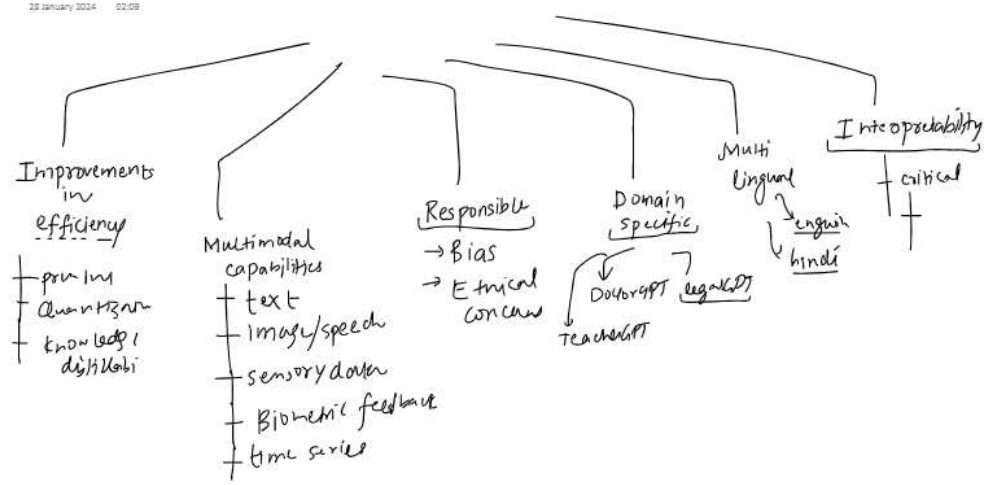
Disadvantages

28 January 2024 01:51

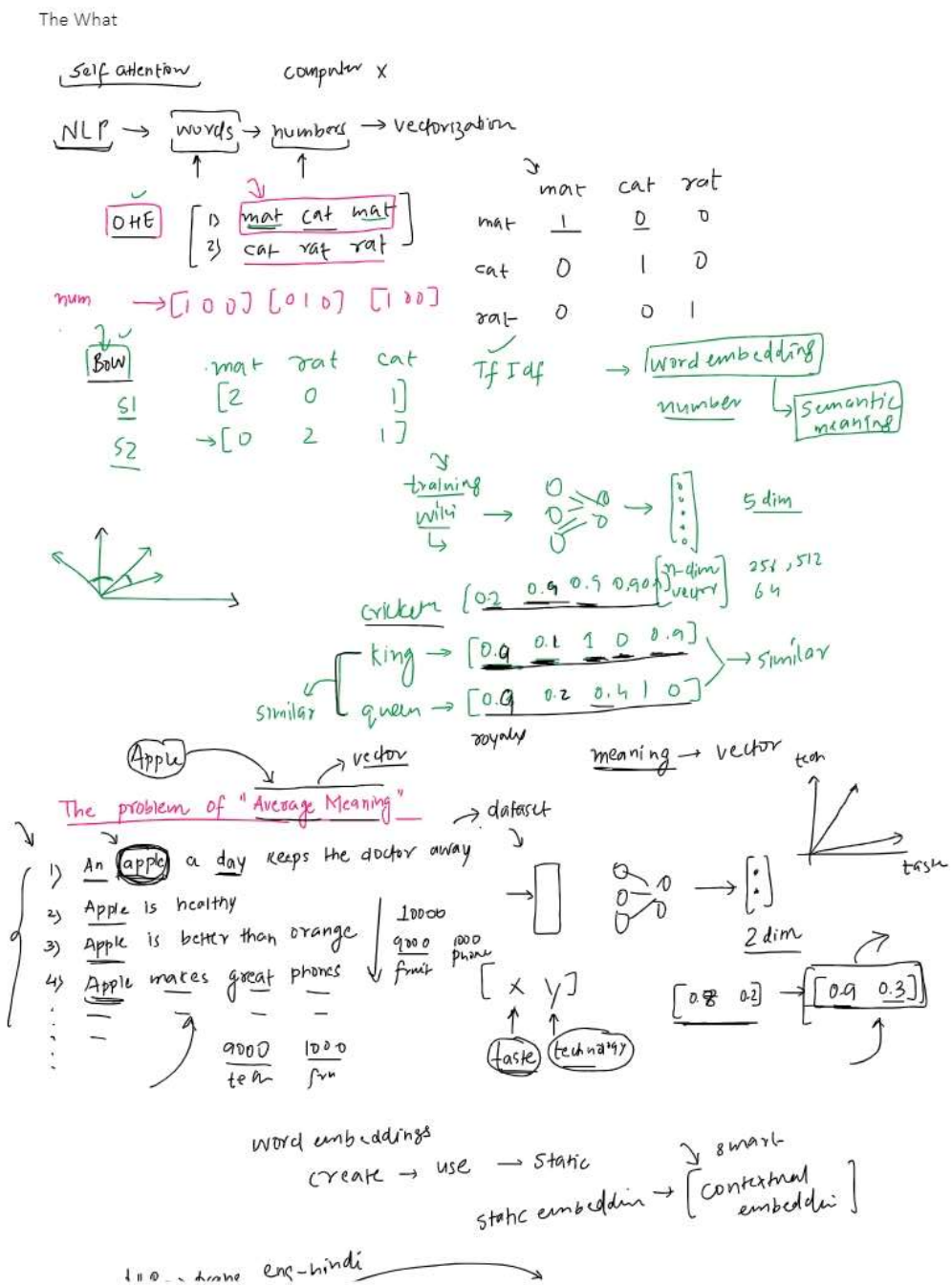


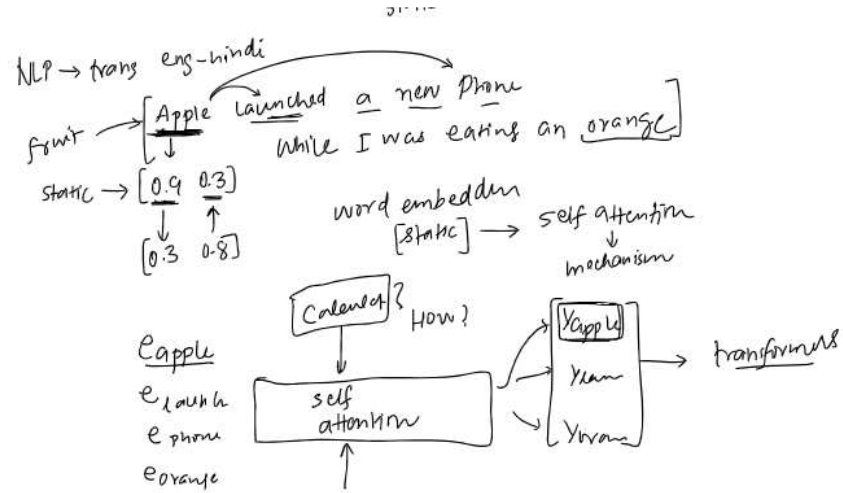
Future

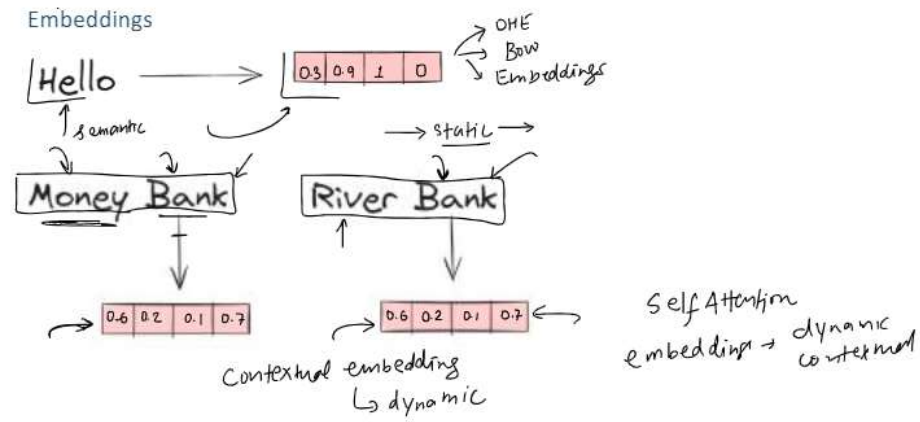
28 January 2024 02:09

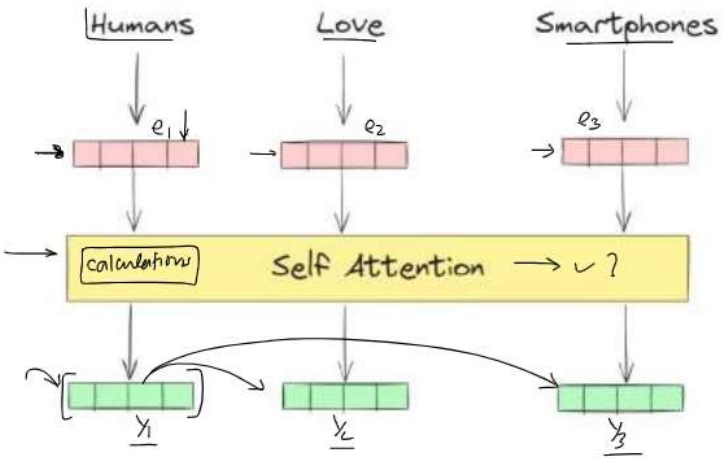


The What


Self attention
computer x
NLP → words → numbers → vectorization
OHE
1) mat cat mat
2) cat rat rat
num → [1 0 0] [0 1 0] [1 0 0]
BOW
mat rat cat
S1 [2 0 1]
S2 → [0 2 1]
mat cat rat
mat 1 0 0
cat 0 1 0
rat 0 0 1
Tf Idf → word embedding
number
semantic meaning
training
wiki
5 dim
cricket [0.2 0.9 0.9 0.90] n-dim vector
256, 512
64
king → [0.9 0.1 1 0 0.9]
queen → [0.9 0.2 0.4 1 0]
similar
similar
royalty
Apple
vector
meaning → vector
tech
taste
The problem of "Average Meaning"
dataset
1) An apple a day keeps the doctor away
2) Apple is healthy
3) Apple is better than orange
4) Apple makes great phones
10000
9000 fruit
1000 phone
9000 tech
1000 fruit
[x y]
taste
technology
2 dim
[0.8 0.2] → [0.9 0.3]
word embeddings
create → use → static
static embedding → [contextual embedding]
smart
eng-hindi


NLP → trans eng-hindi
fruit
Apple launched a new phone while I was eating an orange
static → [0.9 0.3]
[0.3 0.8]
word embedding [static] → self attention
mechanism
Calculate?
How?
e apple
e launch
e phone
e orange
self attention
Y apple
Y launch
Y phone
transformers


Embeddings
Hello
0.3 0.9 1 0
OHE
BOW
Embeddings
semantic
static
Money Bank
River Bank
0.6 0.2 0.1 0.7
0.6 0.2 0.1 0.7
Contextual embedding
dynamic
Self Attention
embeddings → dynamic contextual



First Principle Approach

02 February 2026 17:55

money bank grows

bank → bank

bank → 0.3 money + 0.7 bank + 0.1 grows

river bank flows

bank = 0.5 river + 0.4 bank + 0.1 flows

S1 word embe

money = 0.7 money + 0.2 bank + 0.1 grows

bank = 0.25 money + 0.7 bank + 0.05 grows

grows = 0.1 money + 0.2 bank + 0.7 grows

S2

river = 0.8 river + 0.15 bank + 0.05 flows

bank = 0.2 river + 0.78 bank + 0.02 flows

flows = 0.4 river + 0.01 bank + 0.59 flows

n dim

n dim

$$e^{(new)}_{money} = 0.7\, e_{money} + 0.2\, e_{bank} + 0.1\, e_{grows}$$

$$e^{(new)}_{bank} = 0.25\, e_{money} + 0.7\, e_{bank} + 0.05\, e_{grows}$$

$$e^{(new)}_{grow} = 0.1\, e_{money} + 0.2\, e_{bank} + 0.7\, e_{grows}$$

similarity

e_{money} e_{money}

e_{money} e_{bank}

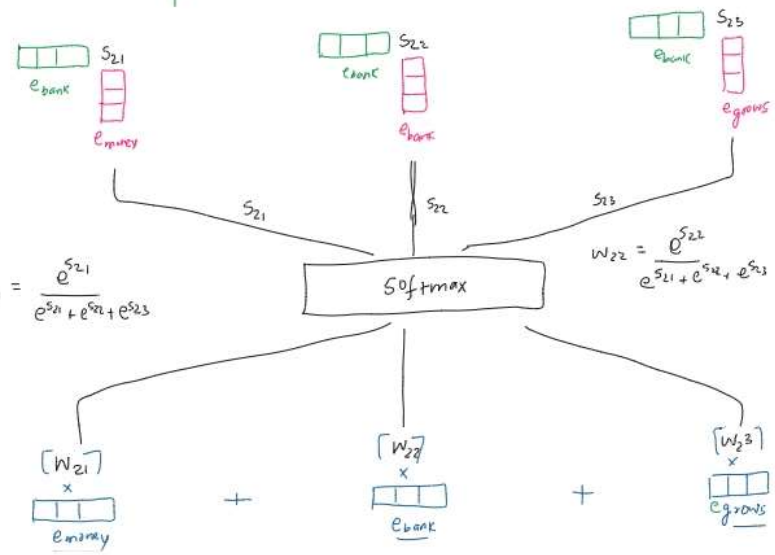
e_{money} e_{grow}

e_{grows} e_{money}

dot products

normalized

$$e^{(new)}_{bank} = [e_{bank} \cdot e^T_{money}]\, e_{money} + [e_{bank} \cdot e^T_{bank}]\, e_{bank} + [e_{bank} \cdot e^T_{grows}]\, e_{grows}$$



e_{money} + e_{bank} + e_{grows}

$e_{bank}^{(new)} = \boxed{Y_{bank}}$ → contextual word embedding

Points to consider

→ This operation is a parallel operation

→ There are no parameters involved

suf model

hello → e_1 → Y_1

how are you → e_1 e_2 e_3 → self mod → Y_1 Y_2 Y_3

general contextual embedding

machine

eng | hindi

how are you | कैसे हो

I am good | मैं अच्छा हूँ

piece of cake → [illegible] का टुकड़ा

X [task specific contextual embedding]

piece of cake → e_1 e_2 e_3 → Y_1 Y_2 Y_3 → [चुटकी का टुकड़ा]

break a leg → टाँग तोड़ दो

Progress

06 February 2024 00:42

words → machine

↓

embedding → semantic

→ Context

river bank money bank

→ contextual embeddings

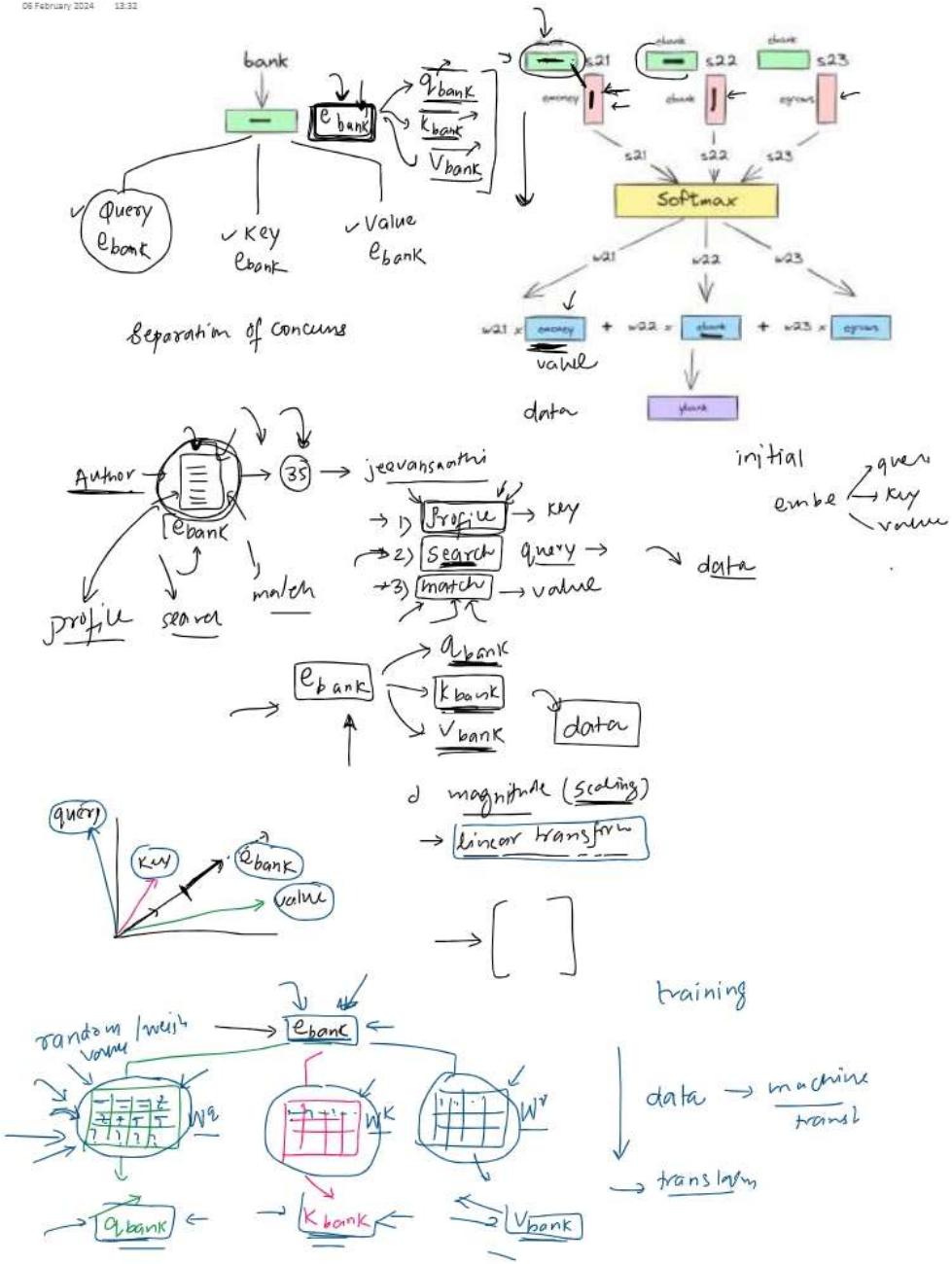
simple self model → parameters

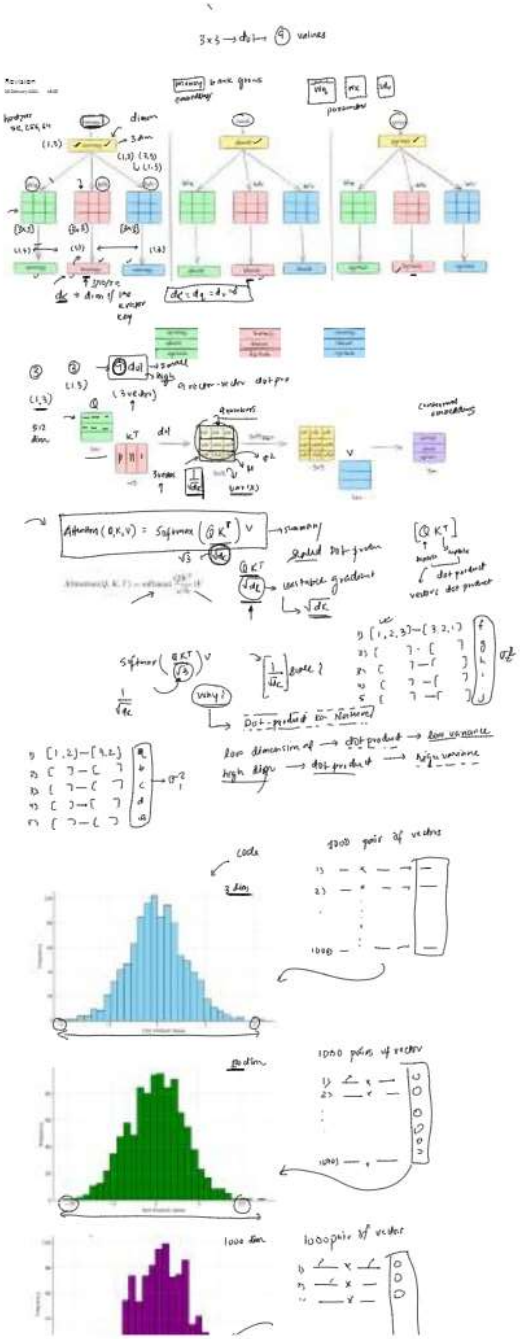
trainable parameters → general

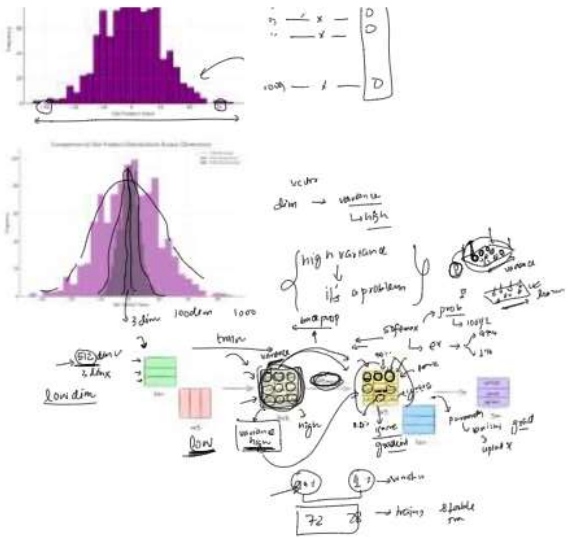
task-specific

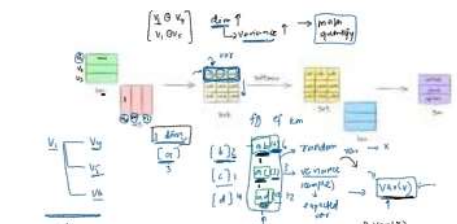
Query, Key & Value Vectors

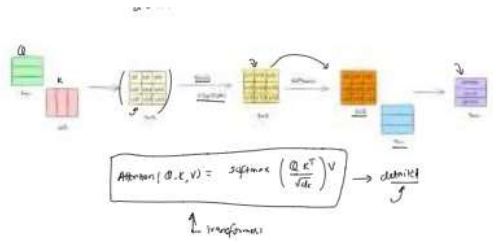
06 February 2024 13:32











cache c

$$\begin{cases} X \to Var(x) \\ Y \to c^2 Var(x) \end{cases}$$

$$\begin{cases} 1 \text{ dim} \to Var(x) \to Var(x) \\ 2 \text{ dim} \to 2Var(x) \to Var(x) \\ 3 \text{ dim} \to 3Var(x) \to Var(x) \\ \vdots \\ d \text{ dim} \to d\,Var(x) \to Var(x) \end{cases}$$

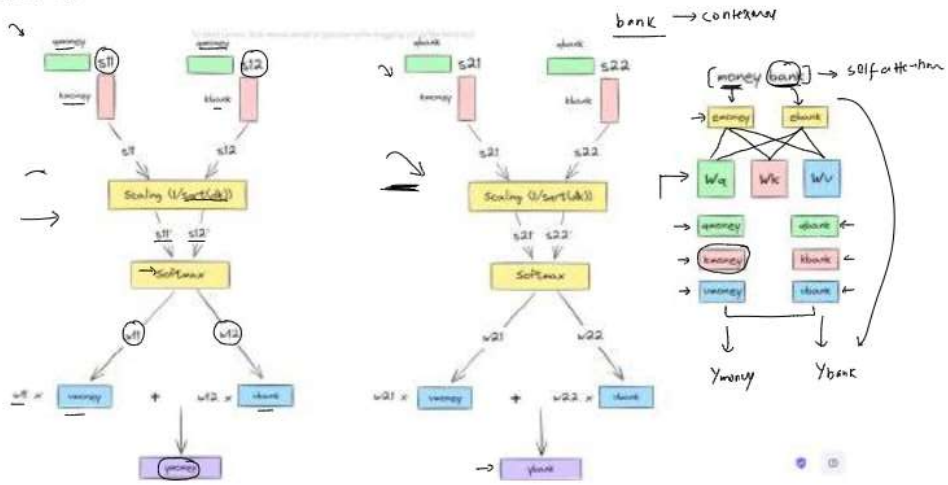
$[\ _\ _\ _\ _\ _\ ] \to \underbrace{Var(X)}_{d\ dim} \to$

What is d_k

28 February 2024 16:59

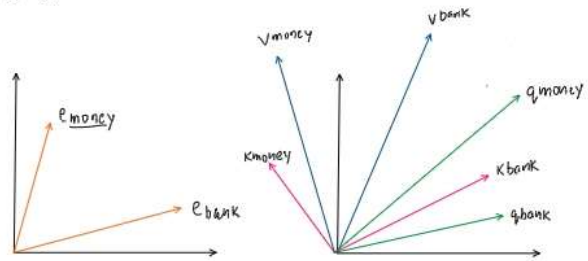
Recep

08 March 2024 15:04



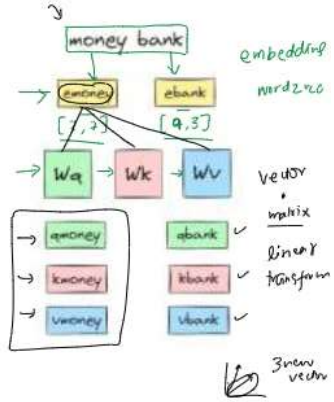
Geometric Intuition

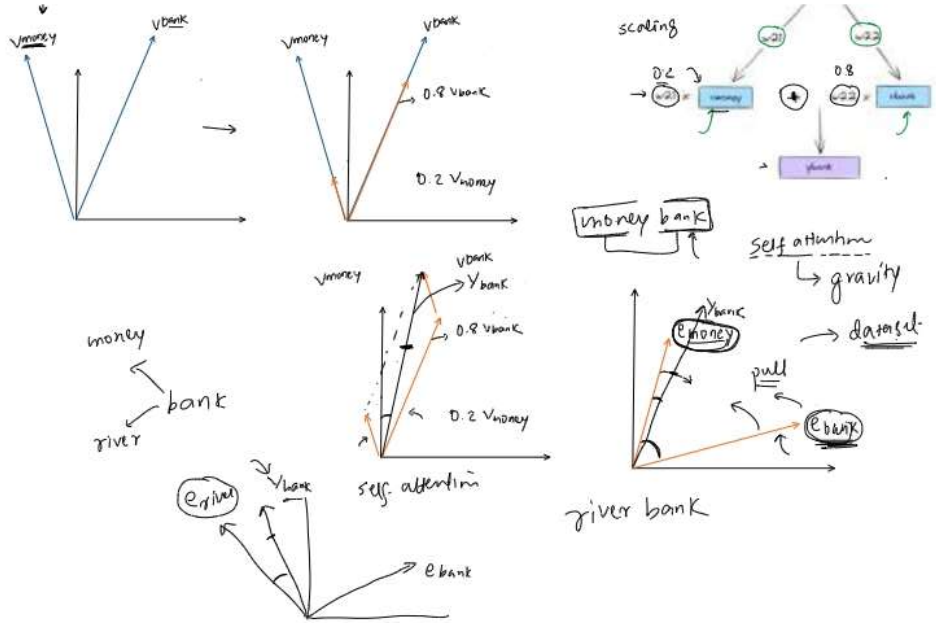
06 March 2024 15:18

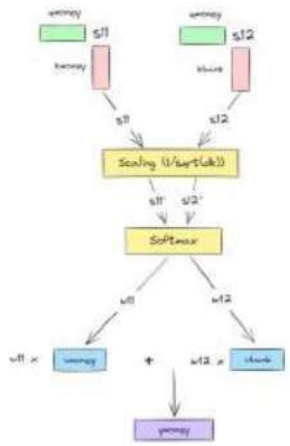


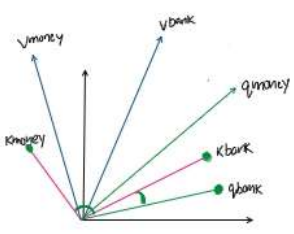
$$W_q = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \quad (2\times2) \qquad W_k = \begin{bmatrix} 3 & 4 \\ 5 & 1 \end{bmatrix} \quad (2\times2) \qquad W_v = \begin{bmatrix} 4 & 1 \\ 2 & 1 \end{bmatrix} \quad (2\times2)$$

* All values are hypothetical









[Dot Product]

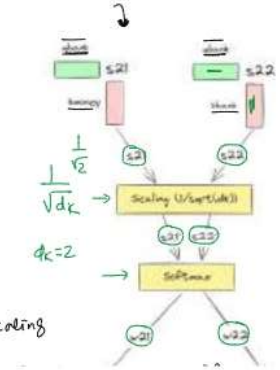
$S_{21} = \boxed{10}$ $\qquad S_{22} = \boxed{32}$

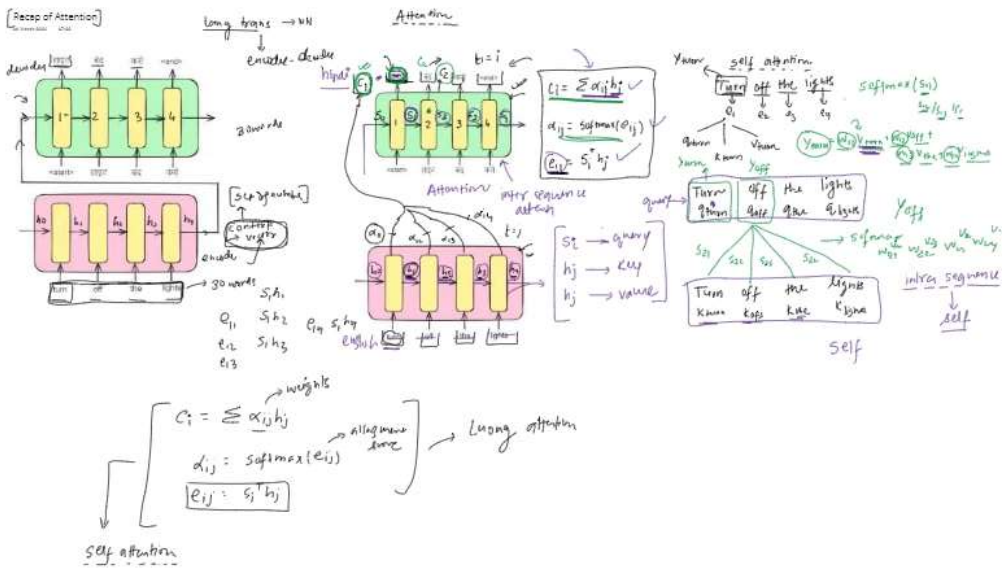
[Scaling]

$S'_{21} = \boxed{\frac{10}{\sqrt{2}}} = 7.09$ $\qquad S'_{22} = 22.69$

[Softmax]

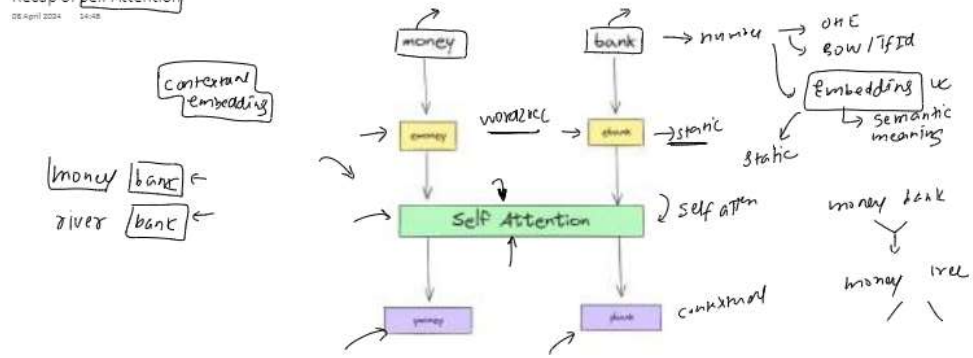
$w_{21} = \boxed{0.2}$ $\qquad w_{22} = \boxed{0.8}$

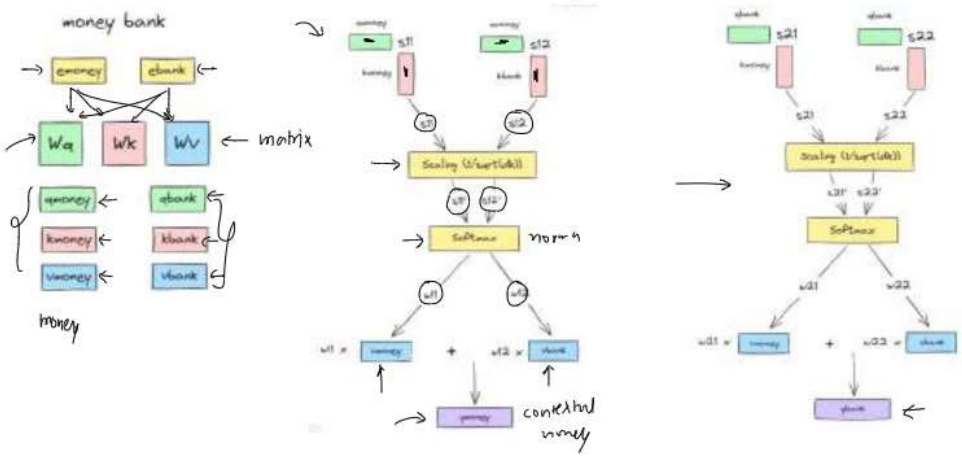




Recap of Self Attention

08 April 2024 14:48





Problem with Self Attention

08 April 2024 14:47

The man saw the astronomer with a telescope

Multihead attention

table

self

single perspective

	The	man	saw	the	astronomer	with	a	telescope
The	0							
man								
saw								
the								
astronomer								
with								
a								
telescope								

doc summarization

self attention

The future of AI in India presents a dynamic and promising landscape, marked by rapid advancements and a burgeoning ecosystem of innovation. With a robust talent pool of engineers and IT professionals, India is poised to become a significant player in the global AI arena. The government's proactive stance on AI, exemplified by initiatives like the National Strategy for Artificial Intelligence, aims to harness AI's potential across various sectors, including healthcare, education, agriculture, and urban infrastructure. Indian startups and tech giants are increasingly incorporating AI to solve complex societal challenges, improve efficiency, and enhance service delivery. Moreover, India's focus on ethical AI and data security aims to create a sustainable and responsible growth trajectory. As AI becomes more integrated into daily life and industry, India's unique blend of technological prowess, entrepreneurial spirit, and societal needs will likely shape a distinctive path in the AI domain, fostering innovation that is not only technologically advanced but also socially inclusive and impactful.

India is poised to become a key player in the AI domain, leveraging its skilled workforce and government initiatives to apply AI across various sectors like healthcare and education. With a focus on innovation, ethical AI, and data security, India aims to integrate AI to address societal challenges, enhance efficiency, and promote inclusive growth. This approach positions India to uniquely contribute to global AI advancements while ensuring sustainable and responsible development within its own borders.

India's AI future holds promise as it harnesses a burgeoning talent pool and government initiatives to pioneer AI-driven innovation. With a focus on sectors like healthcare and education, India aims to leverage AI for societal development. The emphasis on ethical AI and data security underscores India's commitment to responsible technological advancement. This approach positions India not only as a global AI hub but also as a trailblazer in addressing societal challenges through cutting-edge technology.

Multi-head Attention

The man saw the astronomer with a telescope

self attention → 1

self attention

2 self attention

contextual

1 self

2 self

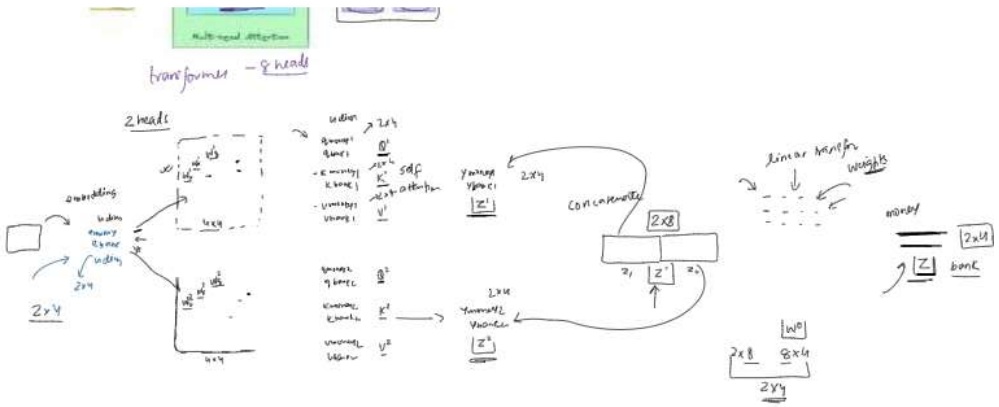
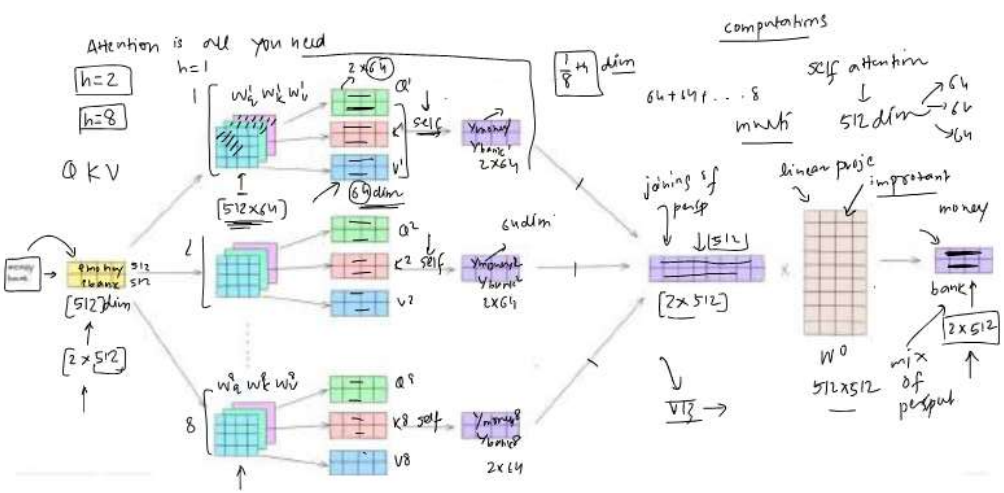
mini

2 self attention

2nd self attention

Multi-head Attention

8 heads



18 April 2024 16:56