-ve

$z_1 = w_1 x_1 + w_2 x_2 + b_1 < 0$

η high → -ve

normalize → 0-1

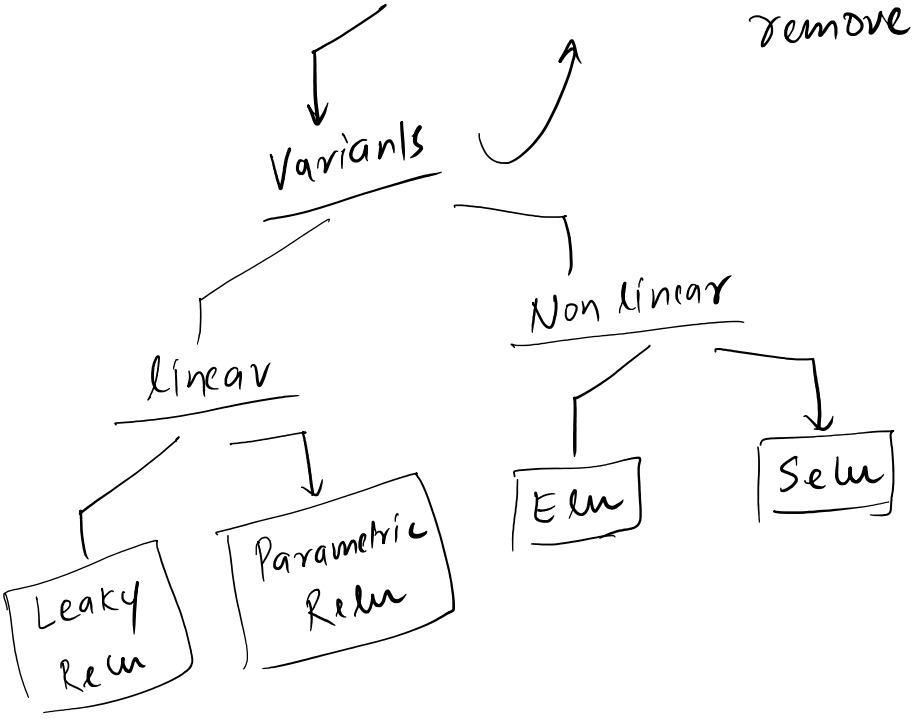
b_1 → 50% → dead → not recover

Solutions

→ Set low learning rate

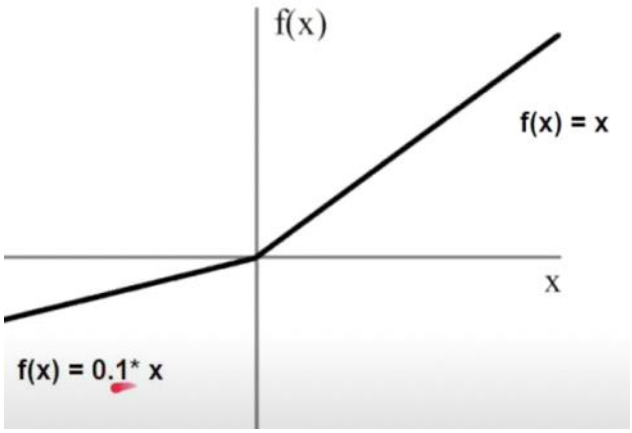
→ bias → +ve value → 0.01

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



Leaky Relu

08 June 2022 22:53



$f'(z) = \max(0.01z, z)$

$z \geq 0 \rightarrow z$

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

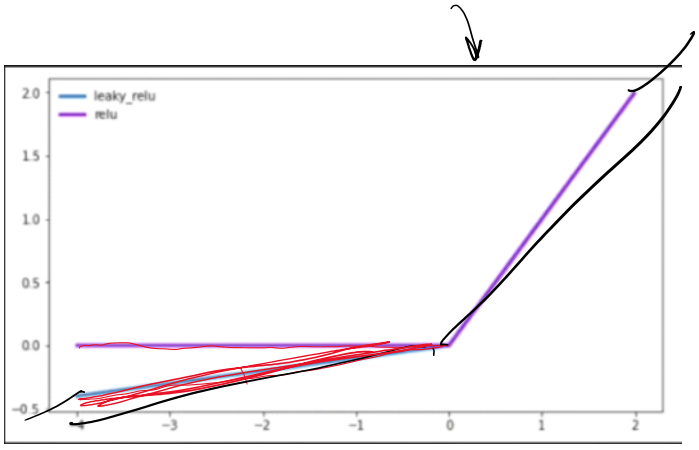
$w_1 = w_1 - \eta \frac{\partial L}{\partial w_1}$

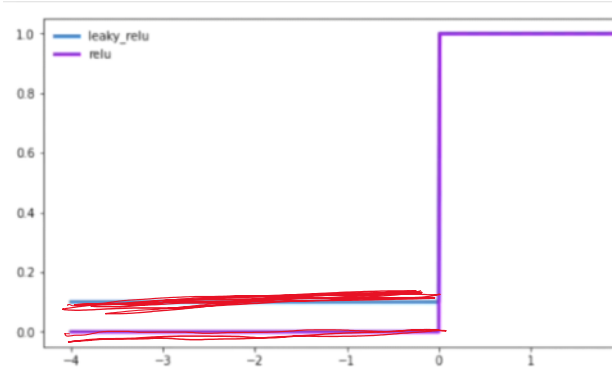
-ve (z) $\rightarrow \neq 0$ 0.01 z

↳ derivative

$f'(z)$ $z \geq 0 \rightarrow 1$

$z < 0 \rightarrow 0.01$





Advantages

→ Non-saturated → Unbounded

→ Easily computed

→ No dying relu problem

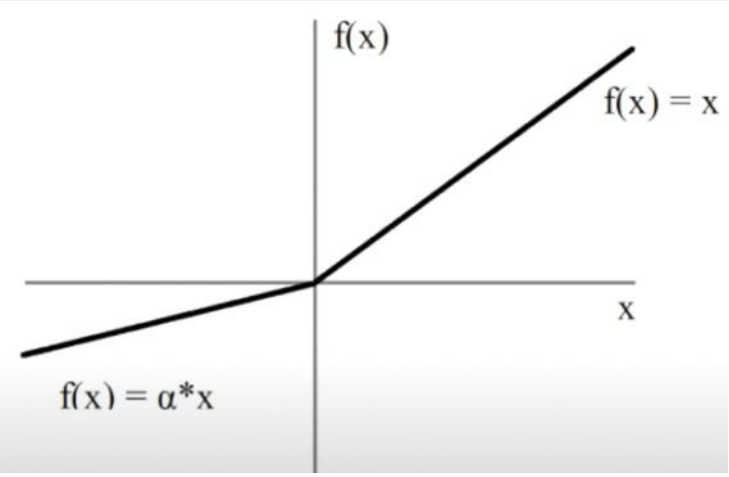
→ Close to 0 centered

-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

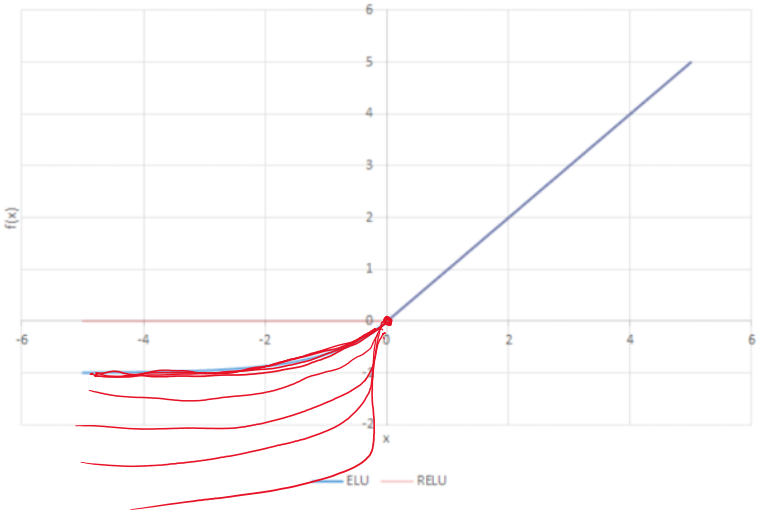
depend on data

flexibility

Elu - Exponential Linear Unit

09 June 2022 00:29

performance better Relu

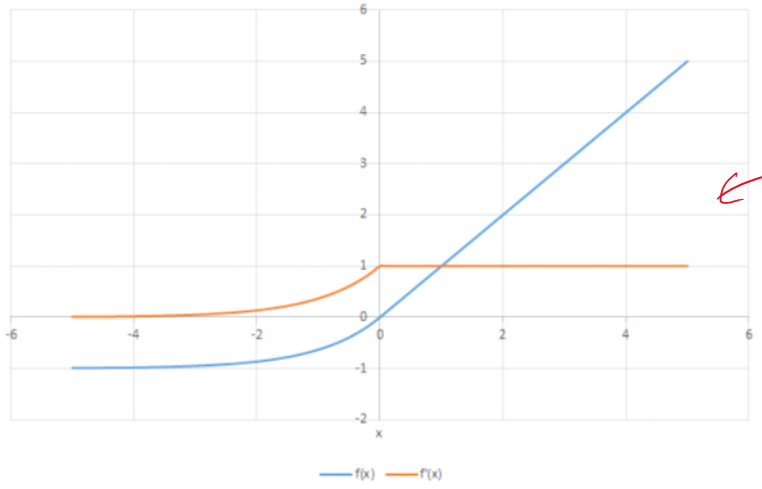


$$\mathrm{ELU}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{if } x < 0 \end{cases}$$

constant

0.1 to 0.3

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



Advan

→ Close zero centered
↳ convergence faster

→ Better generalized

→ Dying relu

→ Always continous as well diffren

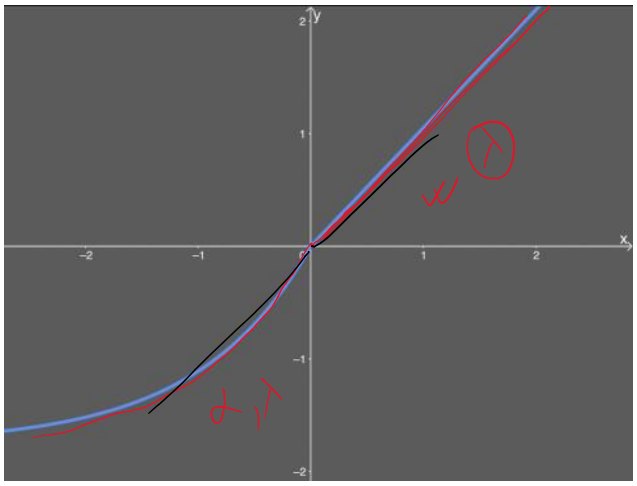
Disadv

→ computation expensive (ex)

Selu - Scaled Exponential Linear Unit

09 June 2022 00:29

Recent → new → 90+ pages



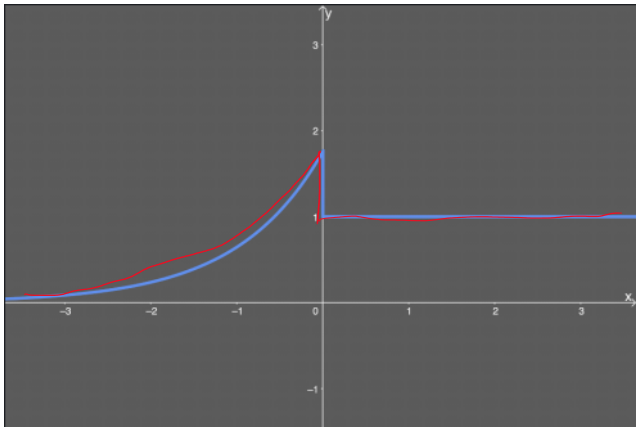
$$\mathrm{SELU}(x) = \lambda \begin{cases} x & \text{if } x > 0 \\ \alpha e^x - \alpha & \text{if } x \leq 0 \end{cases}$$

$$a \approx 1.6732632423543772848170429916717$$

$$\lambda \approx 1.0507009873554804934193349852946$$

fixed ✓ trainbu ✗

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



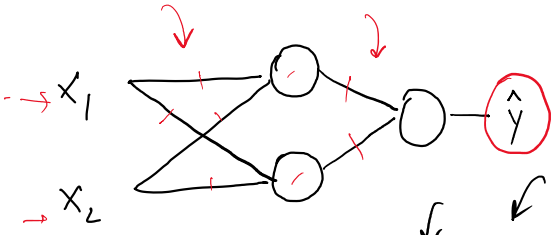
self-normalizing → activation → normalized

$m = 0 \quad \sigma = 1$

converge faster

Why Weight Initialization is Important?

22 June 2022 12:48



→ Vanishing GP
→ Exploding GP
→ Slow convergence

Wrong WI techniques

↓

What to do?

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

Sigmoid Activ

Wrong weight init

What not to do?

22 June 2022 12:49

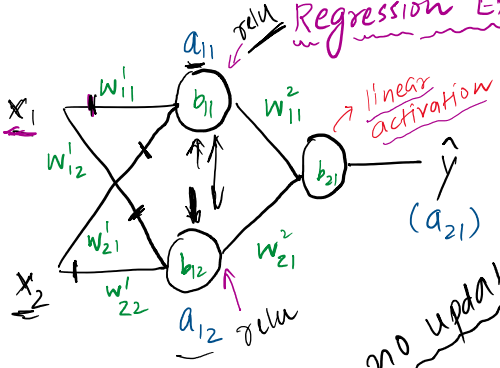
Case1 → Zero Initialization

relu
tanh
sigmoid

$W = 0, b = 0$

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

Regression Example



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

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

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

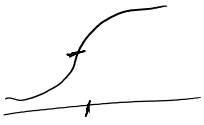
$z_{12} = w^1_{12} x_1 + w^1_{22} x_2 + b_{12} = 0$

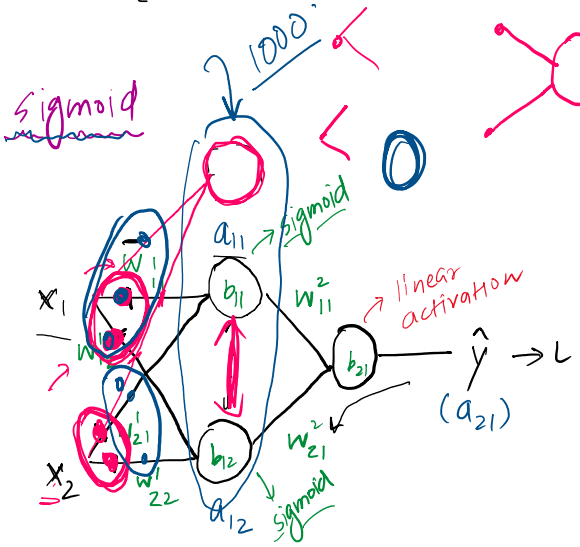
$a_{11} = a_{12}$ → equal and 0

no update

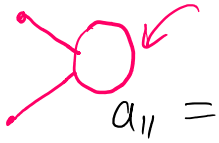
$$w^1_{11} = w^1_{11} - \eta \frac{\partial L}{\partial w^1_{11}}$$

$W = 0$ → No training will place





0



0

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

$z_{11} =$

$a_{12} = 0.5$

$a_{11} = a_{12}$

Non-linear problem

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

$a_{11} = a_{12}$

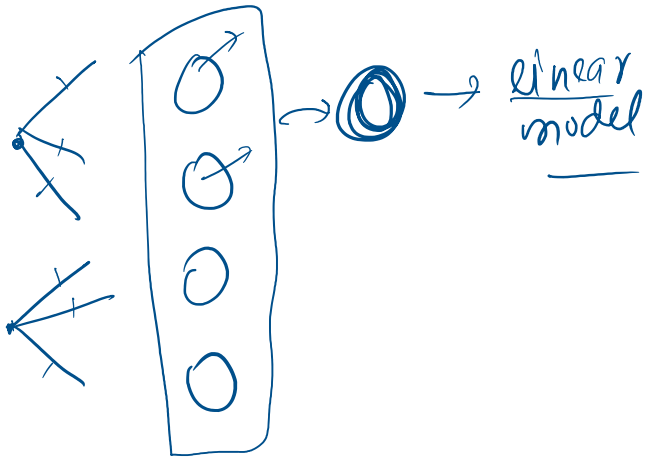
$z_{11} = z_{12}$

$$\frac{\partial L}{\partial w^1_{11}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{11}} \frac{\partial a_{11}}{\partial z_{11}} x_1$$

$$\frac{\partial L}{\partial w^1_{12}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{12}} \frac{\partial a_{12}}{\partial z_{12}} x_1$$

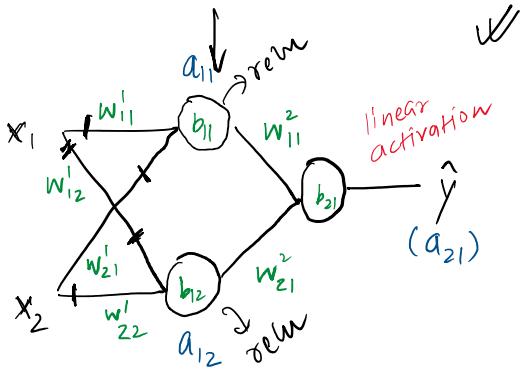
$$\frac{\partial L}{\partial w^1_{21}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{11}} \frac{\partial a_{11}}{\partial z_{11}} x_2$$

$$\frac{\partial L}{\partial w^1_{22}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{12}} \frac{\partial a_{12}}{\partial z_{12}} x_2$$



Case2 → Non-0 constant value

$w = 0.5 \quad b = 0.5$ sigmoid → 0



$a_{11} = \max(0, z_{11})$ → non-zero

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

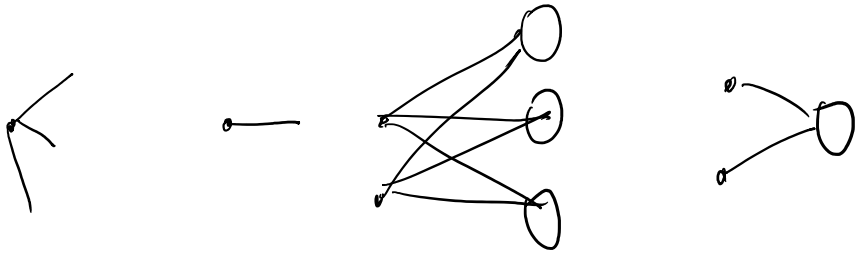
$a_{12} = \max(0, z_{12}) \neq 0$

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

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

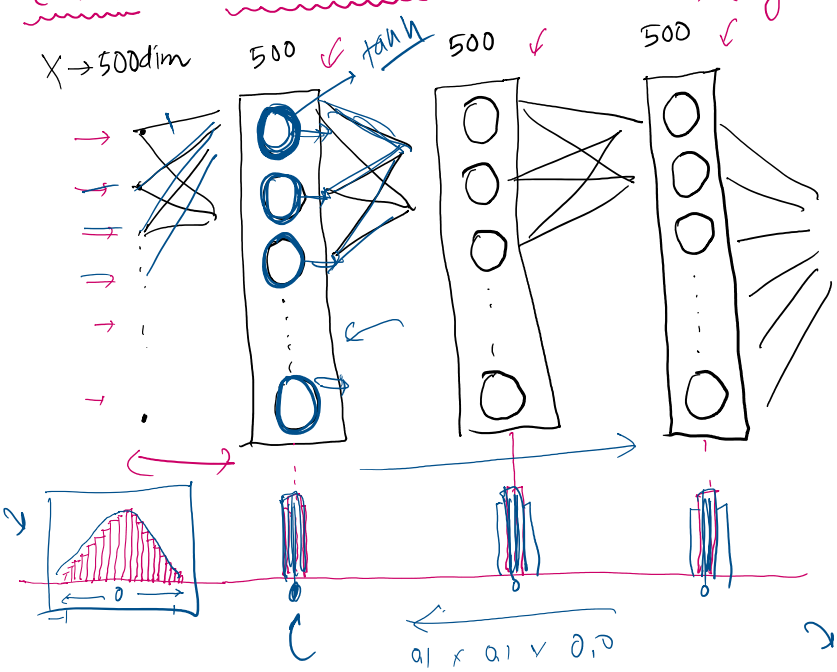
$$\boxed{\frac{\partial L}{\partial w^1_{11}}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{11}} \frac{\partial a_{11}}{\partial z_{11}} x_1 \qquad \boxed{\frac{\partial L}{\partial w^1_{12}}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{12}} \frac{\partial a_{12}}{\partial z_{12}} x_1$$

$$\frac{\partial L}{\partial w^1_{21}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{11}} \frac{\partial a_{11}}{\partial z_{11}} x_2 \qquad \frac{\partial L}{\partial w^1_{22}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{12}} \frac{\partial a_{12}}{\partial z_{12}} x_2$$



random init

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



a1 x a1 x a1 x 0.0

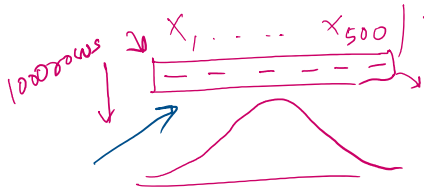
small small ≈ 0

$$W = W - \eta \frac{\partial L}{\partial W} \approx 0$$

- vanishing gradient problem

Relu → small values → slow convergence

(1000, 500)



weight init → random → small values

np.random.randn (500, 500) * 0.01

bias = 0

$\sum w_i x_i$

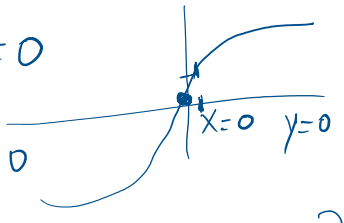
$x_i \rightarrow$ 0-1 → -1 to 1

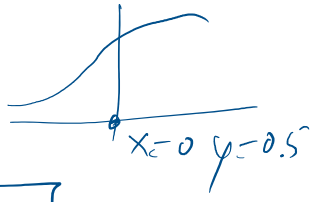
0.0007

small number tanh

Z is a small

very close to 0





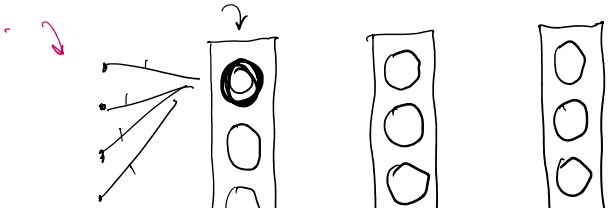
VGP → deep

small weights → 0.0007

tanh → VGP

sigmoid → VGP

Random Initialization (Large values)

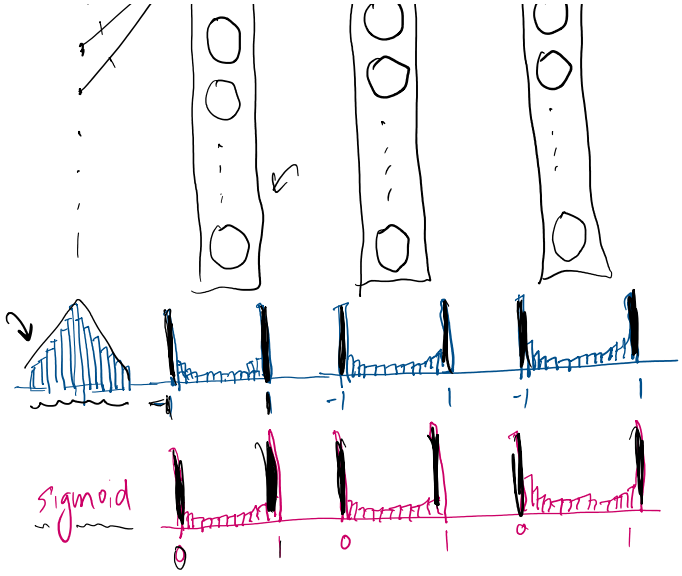


np.random.randn (500, 500) → [0-1]

$\sum w_i x_i$ → 500 → 100 → 500

0-1

tanh / sigmoid



0-1 tanh/sigmoid

$g(z)$ → saturation → saturate

$\sum w_i x_i \rightarrow 100$

$\sum w_i x_i \rightarrow$ 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) → -0.01 to 0.01

Vanishing grad

Slow converge

np.random.randn(2,2) → -3 to 3

What can be done

→ Heuristics (Jugad) → practical soln

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

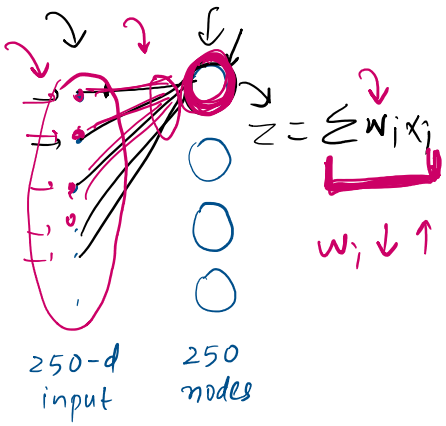
Keras

formulas

-1 to 1 0-1

-0.01 to 0.01

Intuition



$z = \sum w_i x_i$

$w_i \downarrow \uparrow$

small ← np.random.randn(250,250) * 0.01

250 → -250 to 250

large np.random.randn(250,250) * 1 → -3 to 3

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

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

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 ← relu

$\sqrt{\frac{2}{fan\_in}}$

Uniform Distribution

Xavier Uniform

$[-limit, limit]$

$limit = \sqrt{\frac{6}{(fan\_in + fan\_out)}}$

Keras

He Uniform

$[-limit, limit]$ $\quad limit = \sqrt{\frac{6}{fan\_in}}$

Weight Initialization Techniques

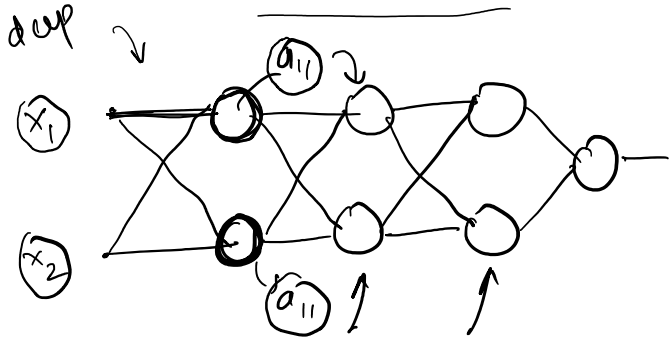
22 June 2022 12:49

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.

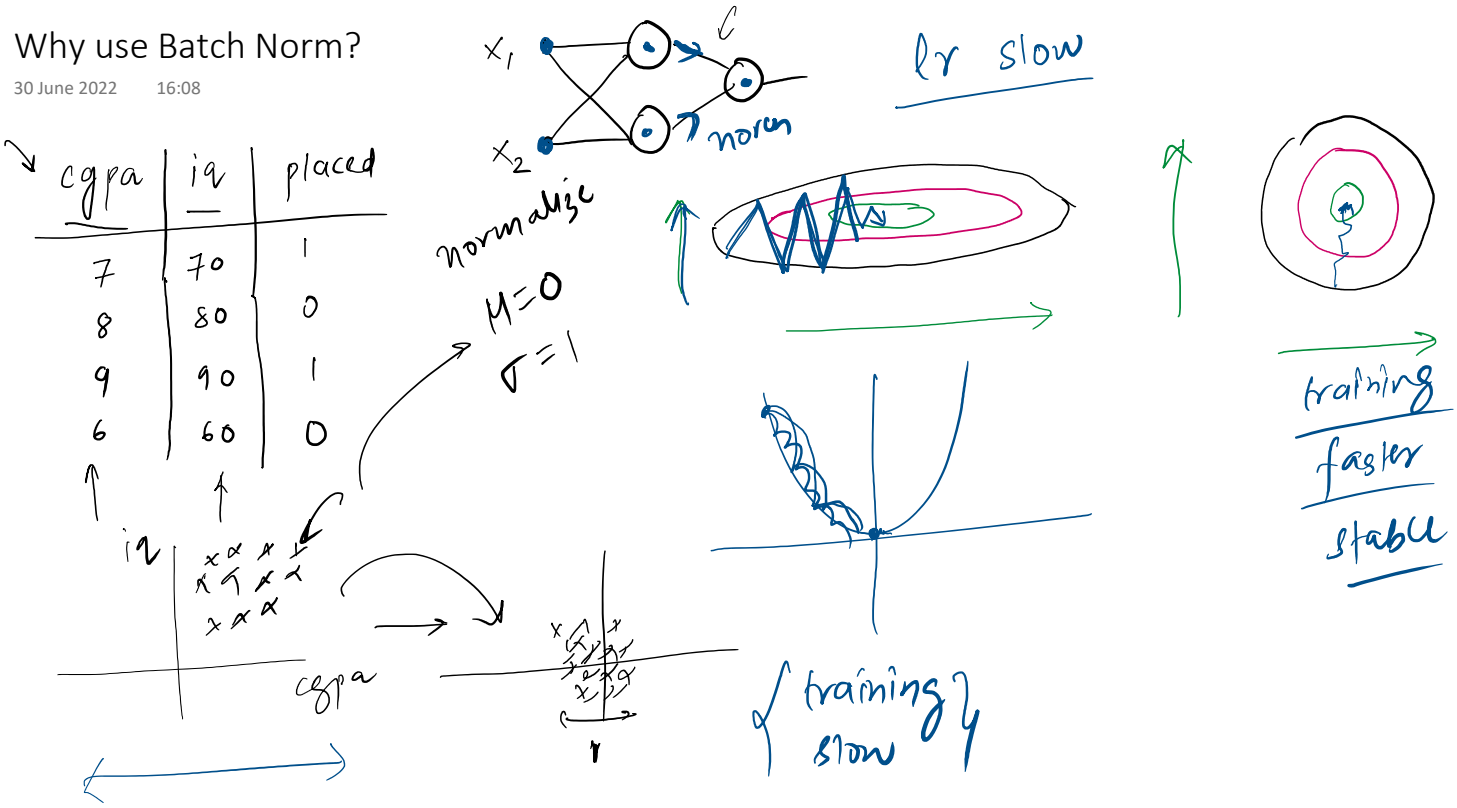


normalized
m=0
sd=1

normalize
m=0 sd=1

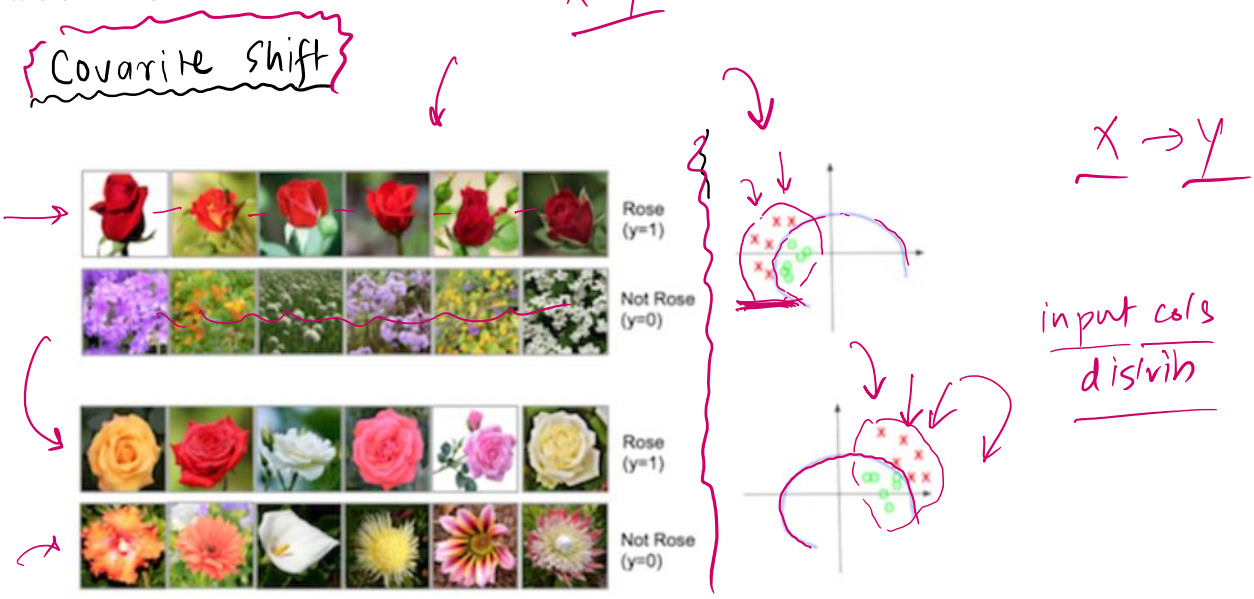
Why use Batch Norm?

30 June 2022 16:08



Internal Covariate Shift

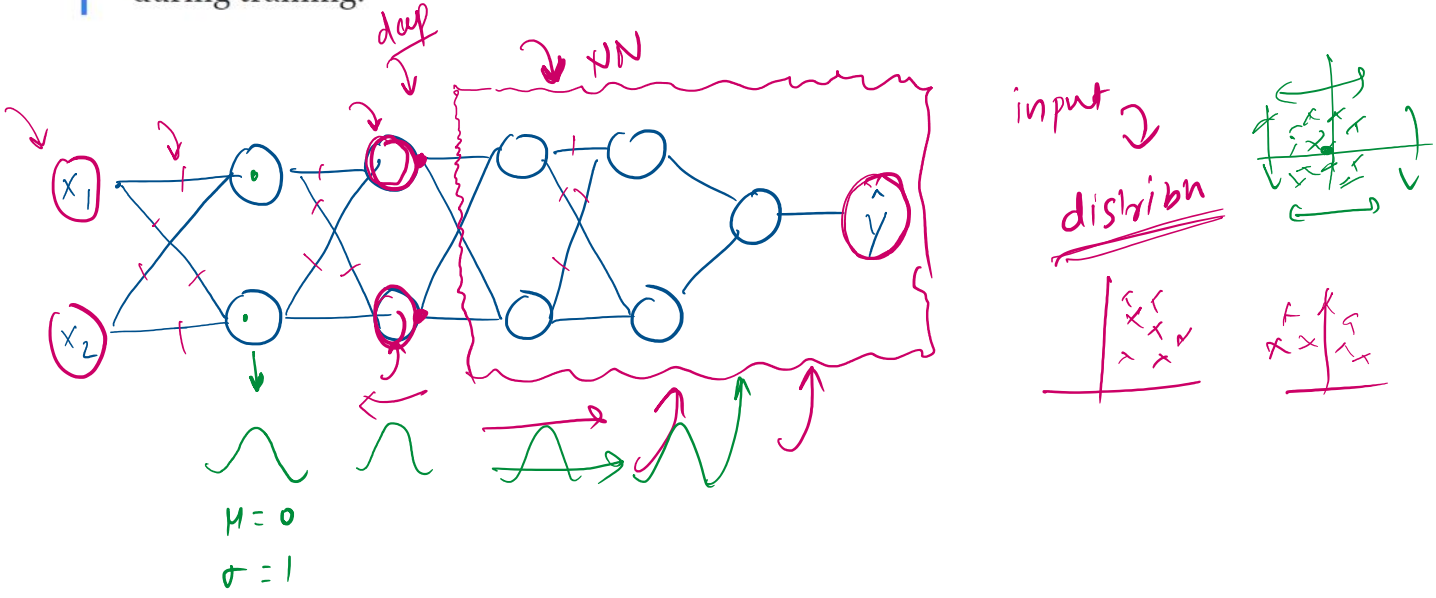
27 June 2022 11:02



learning rate bias parameters 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.




bees
knees
cheese
fleas!

Batch Norm - The How

27 June 2022 11:02

cgpa | iq | placed → cgpa

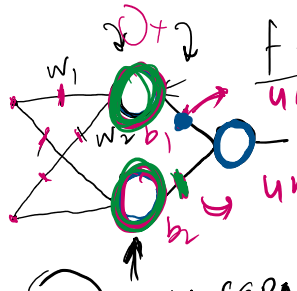
8.9	100	1
6.2	89	0
9.1	91	0
7.7	76	1
⋮	⋮	⋮
6.7	91	0

→ iq

batch_size : 4

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

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



fp

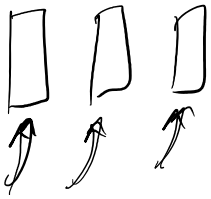
1 number

1 number

→ Mini-Batch GD

→ Layer by layer

normalize $\mu=0$ $\sigma=1$



$z_{11} = w_1 cgpa + w_2 iq + b$

$g(z_{11}) = a_{11}$

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

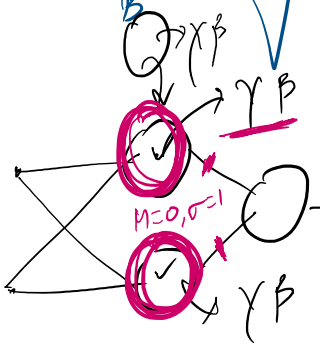
(2) $z_{11} \rightarrow g(z_{11}) = a_{11} \rightarrow a_{11}^N$

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

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

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

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



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

0-1

$\mu=0$, $\sigma=1$

learnable parameters w, b

$\gamma=1$ $\beta=0$

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

$z_{11} \rightarrow$ normal

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

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

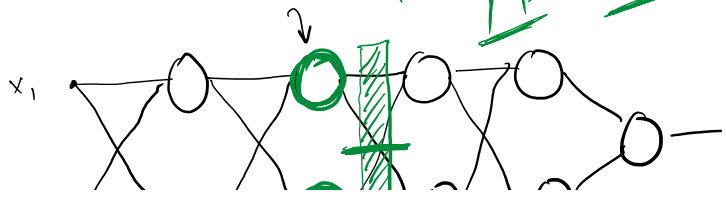
flexibility

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

Batch Norm → layer

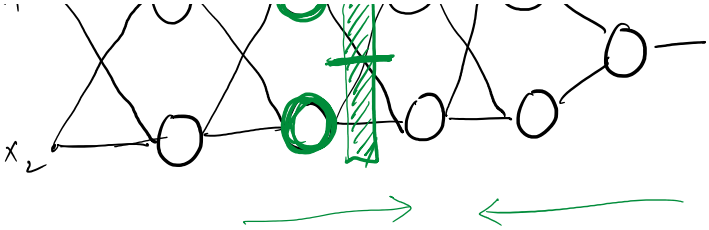
4 learnable γ β

γ β



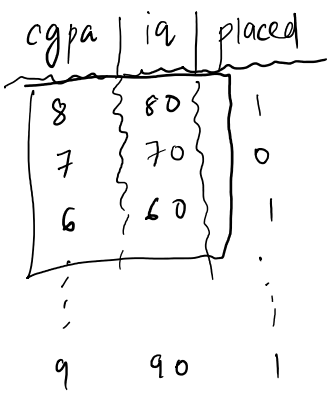
$$\gamma = \gamma - \eta \frac{\partial L}{\partial \gamma}$$

Keras



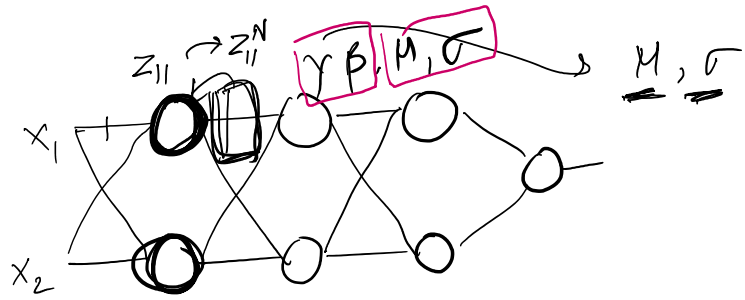
Batch Norm during test

27 June 2022 11:03



100 students

b = 4



25 times

1 batch

2

σ

4 parameters

2 learnable

γ β

Non-learn

μ, σ

testing

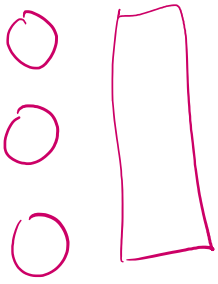
query point

Exp weighted moving avg.

EWMA

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

Ex EWMA



$3 \times 4 = 12$

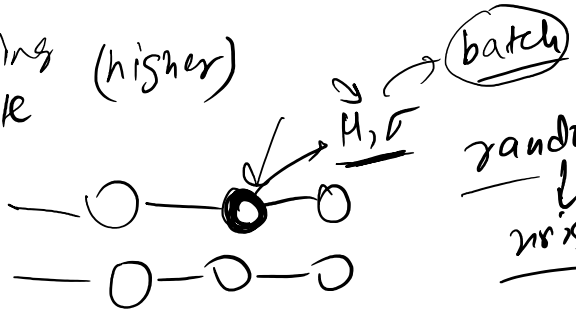
6 6

Advantages

27 June 2022 11:02

1) Stable → hyperpara → wider range of values

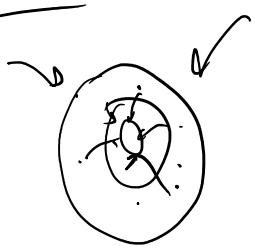
2) faster → learning rate (higher)



3) Regularizer → dropout

randomness ↓ rise

overfitting



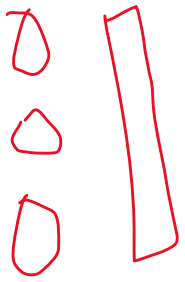
4) Weight init impact reduce

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'))
```

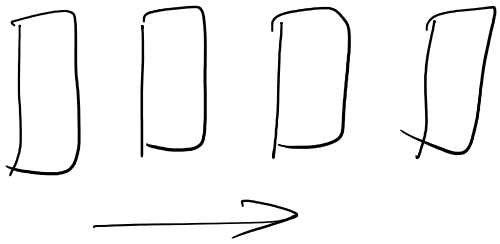


Introduction

05 July 2022 09:57

Performance of NN

↳ how the speed the training



{Optimizers} ✓

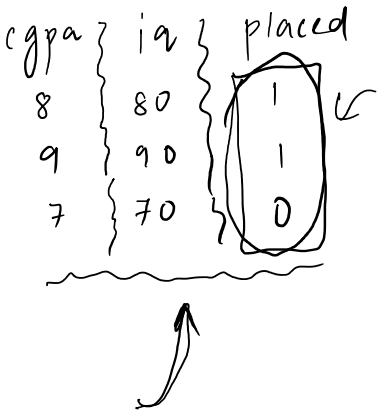
→ Weigh init

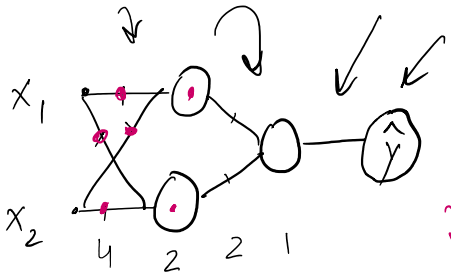
→ Batch Norm

→ Activation function

Role of Optimizer

05 July 2022 10:01





optimization

2 min

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

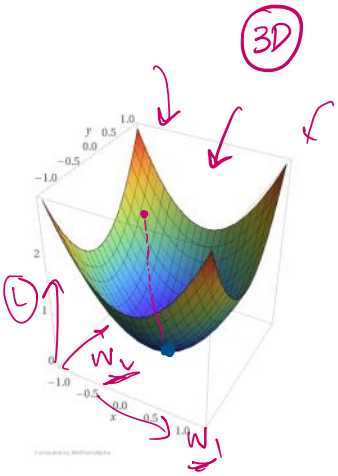
9 parameters

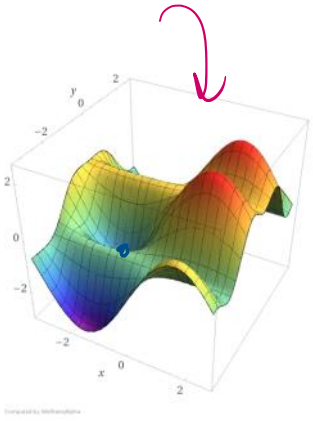
C/L()

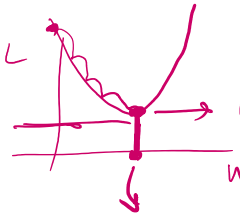
9 para

2 weights

L 3D L(.,.,.,.,.) 10D







Gradient Descent

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

Types of Optimizers

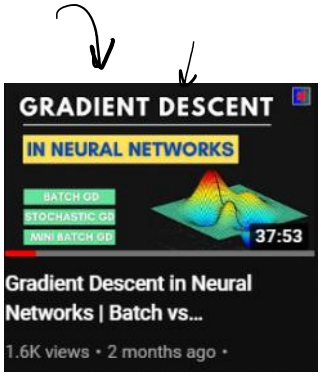
05 July 2022 10:02

- Batch GD
- Stochastic GD
- Mini batch GD

(10) ← (10)

epochs = 10

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



Gradient Descent in Neural Networks | Batch vs...

1.6K views • 2 months ago •

500 rows → pred

↓

loss → weight update

10 × 500 → 5000

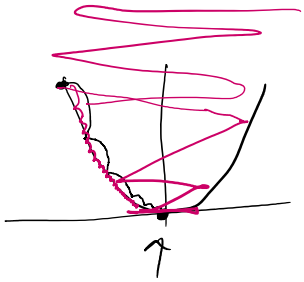
batch size = 100

5 batch → 10 × 5 times

Challenges

05 July 2022 10:02

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



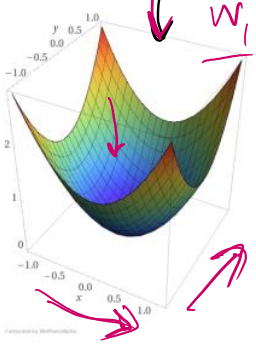
1) learning rate

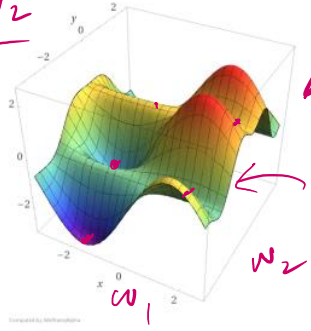
2) learning rate scheduling → pre define

3)

(η) same both the directions

w_1 w_2





(9) weights and bias (9 sep learn

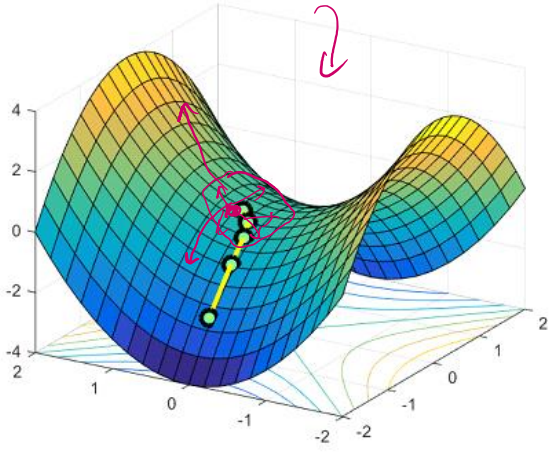
10-D → 10 directions

complex minima

sub optimum

4) local minima

5) saddle point

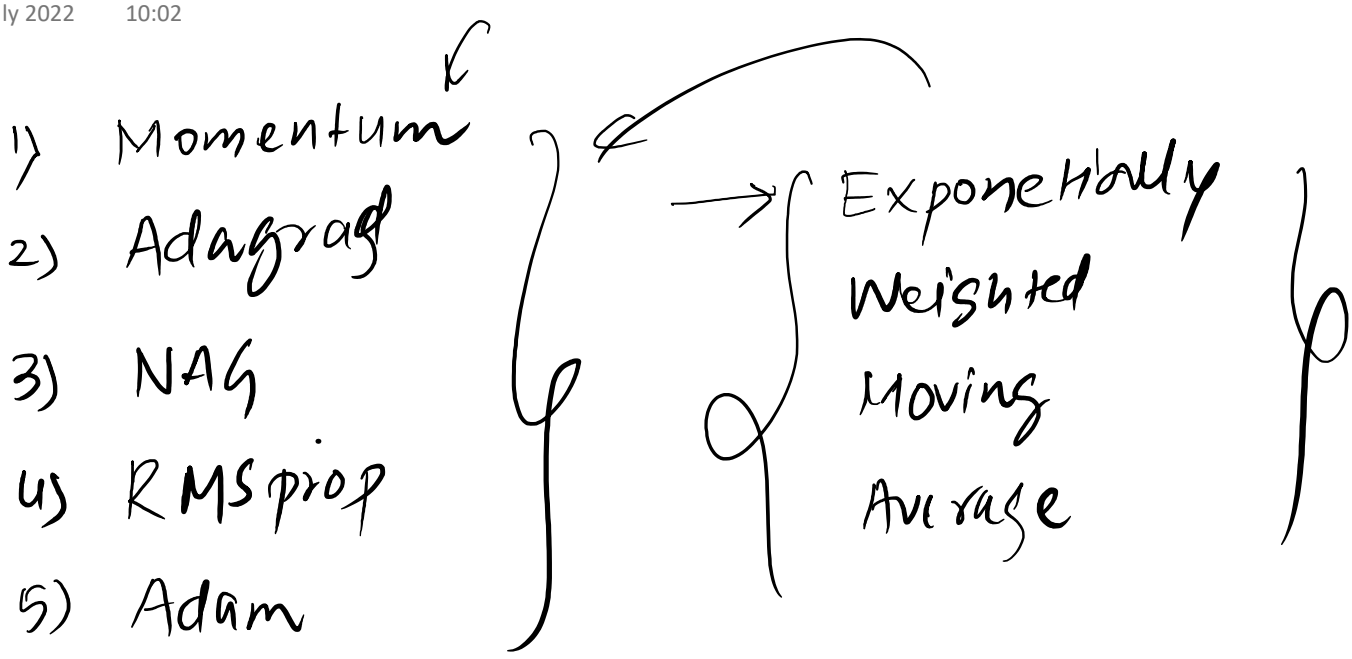


$$\frac{\partial L}{\partial w} = 0$$

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

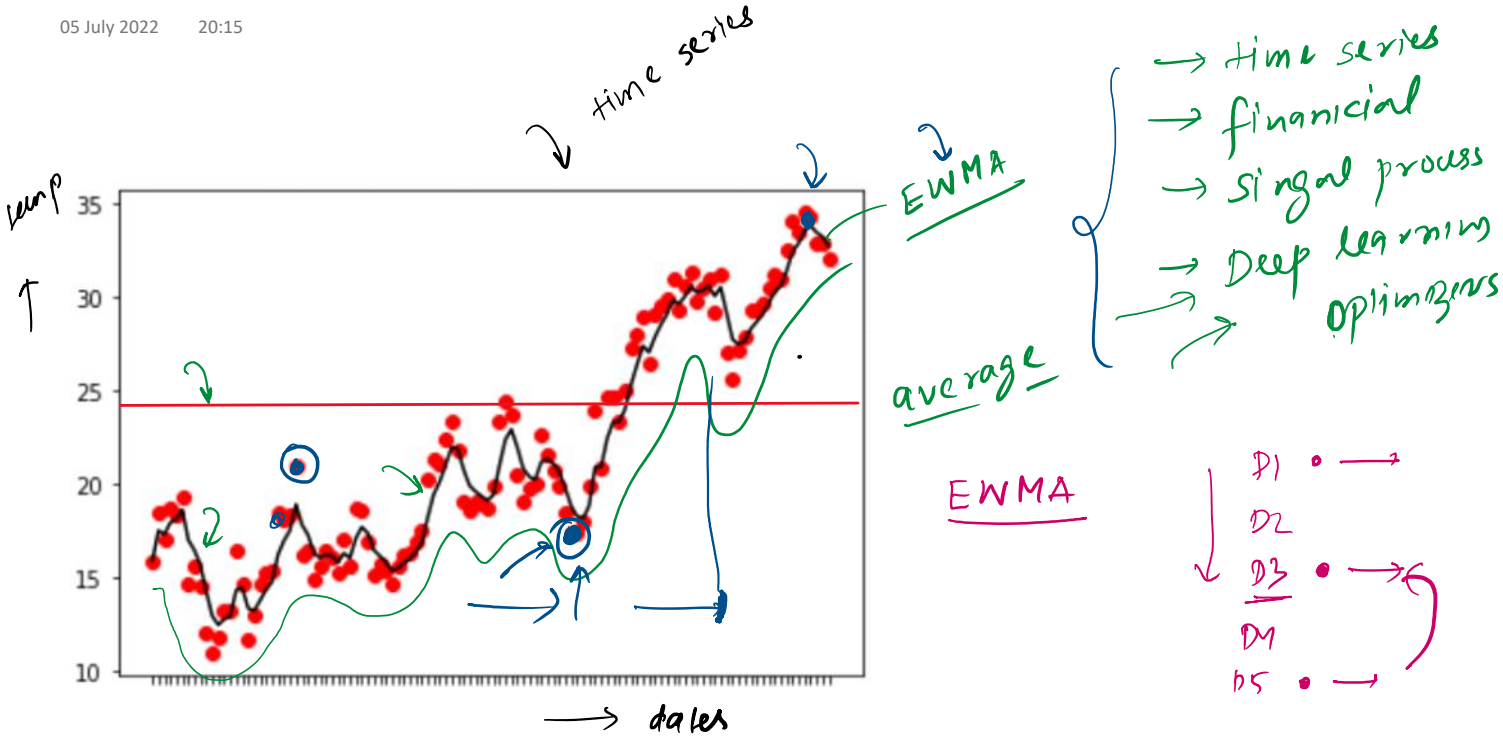
What next?

05 July 2022 10:02



The What

05 July 2022 20:15



Mathematical Formulation

05 July 2022 20:16

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

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

$\boxed{\beta = 0.9}$

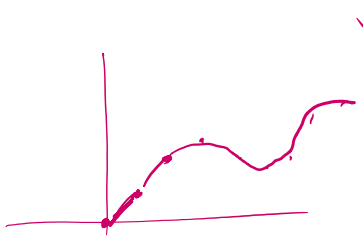
index	temp (θ)
D1	25
D2	13
D3	17
D4	31
→ D5	43

$V_0 = 0$ $\quad V_0 = \theta_0$

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

$= 1.3$

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



$\beta = 0.5$

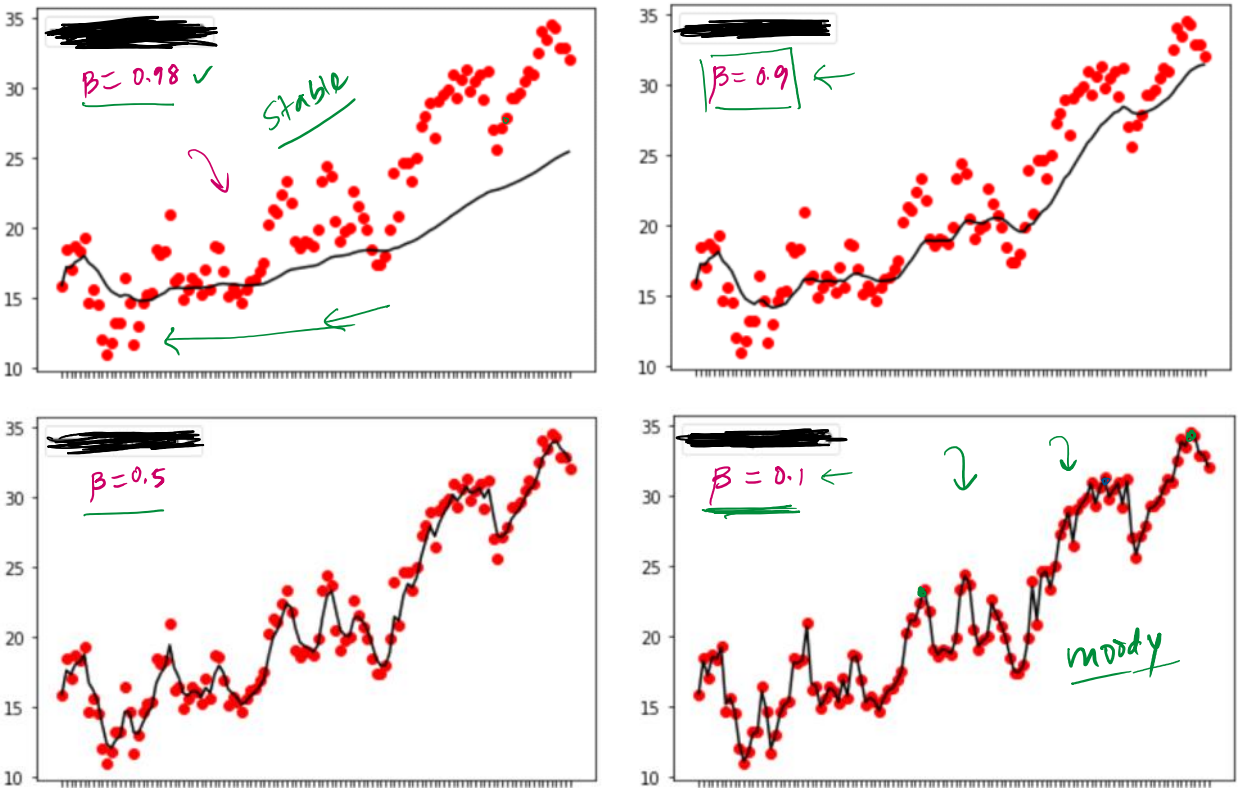
$\frac{1}{1-0.5} = \frac{1}{0.5}$

2 days

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

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

Effect of β



Mathematical Intuition

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

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

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

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

$V_0 = 0$

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

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

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

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

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

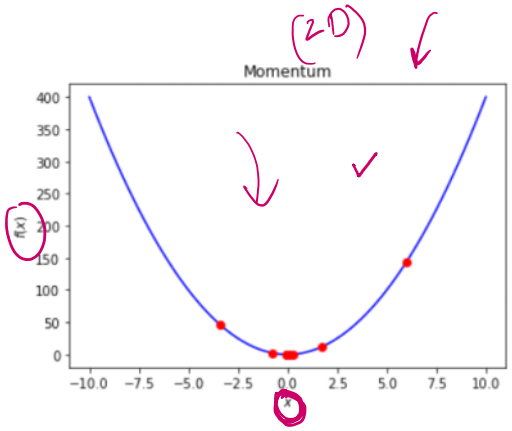
$V_4 = \beta^3(1-\beta)\theta_1 + \beta^2(1-\beta)\theta_2 + \beta(1-\beta)\theta_3 + (1-\beta)\theta_4$

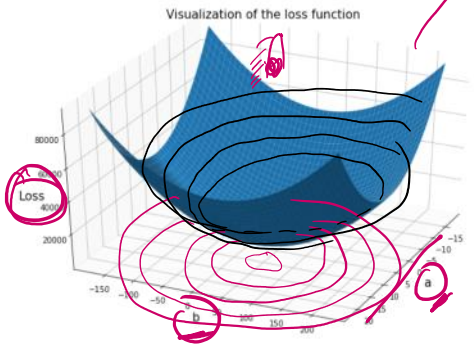
$= (1-\beta)[\beta^3\theta_1 + \beta^2\theta_2 + \beta\theta_3 + \theta_4]$

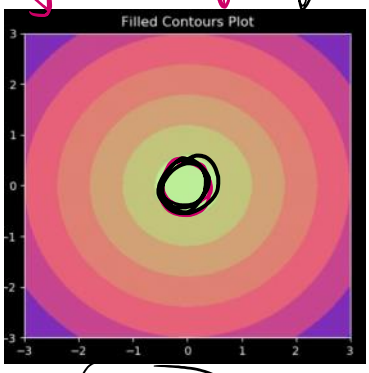
$\beta^3 < \beta^2 < \beta$ $\quad 0 < \beta < 1$

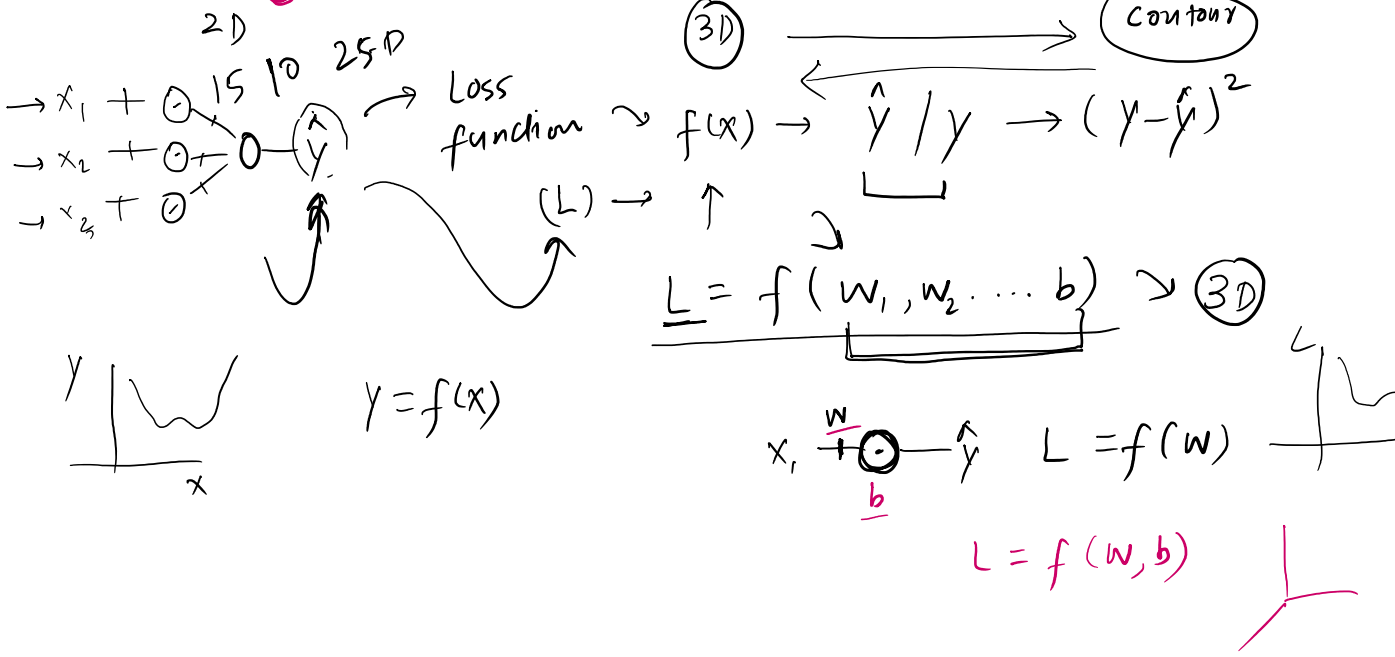
Understanding Graphs

20 July 2022 12:03







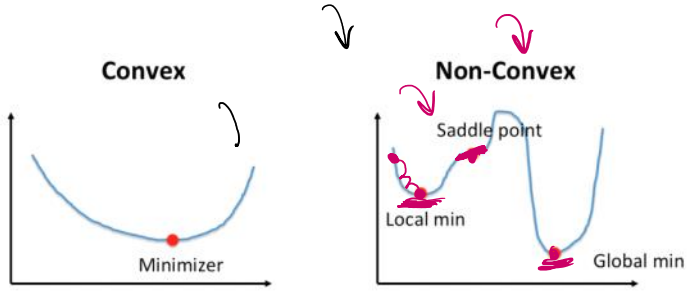


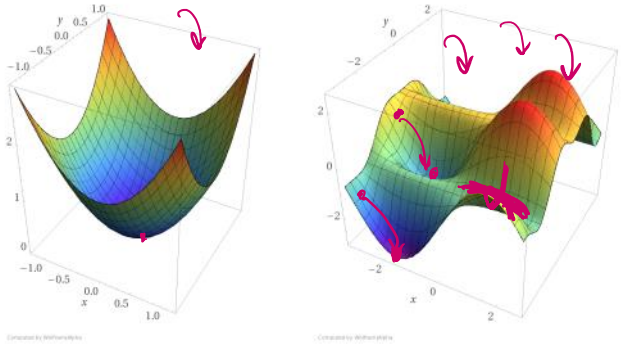
y = f(x)

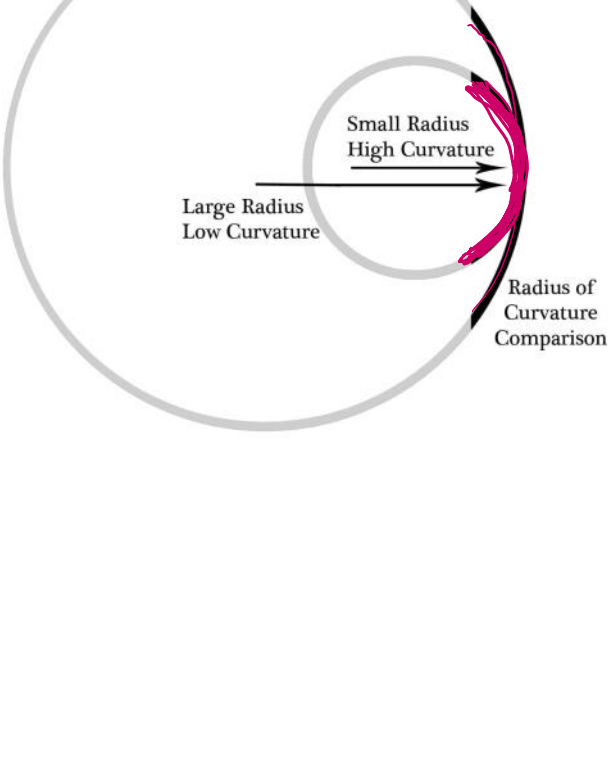
L = f(w, b)

Convex Vs Non-Convex Optimization

20 July 2022 13:06

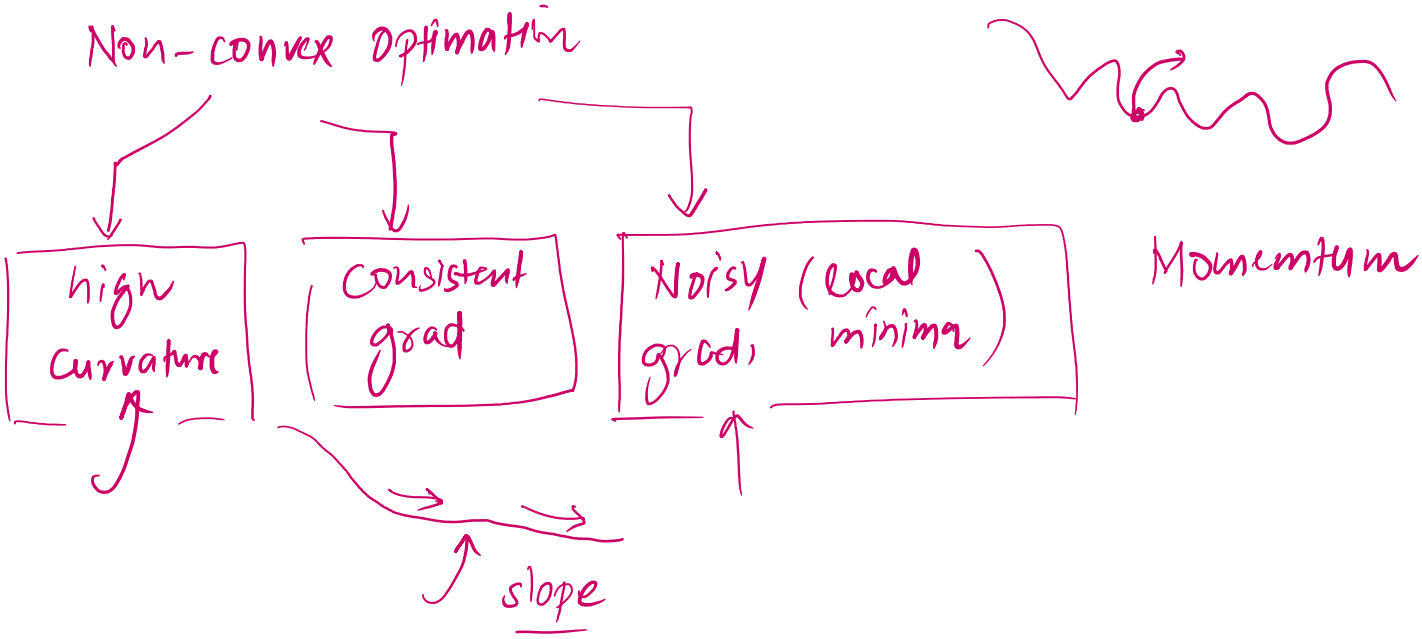






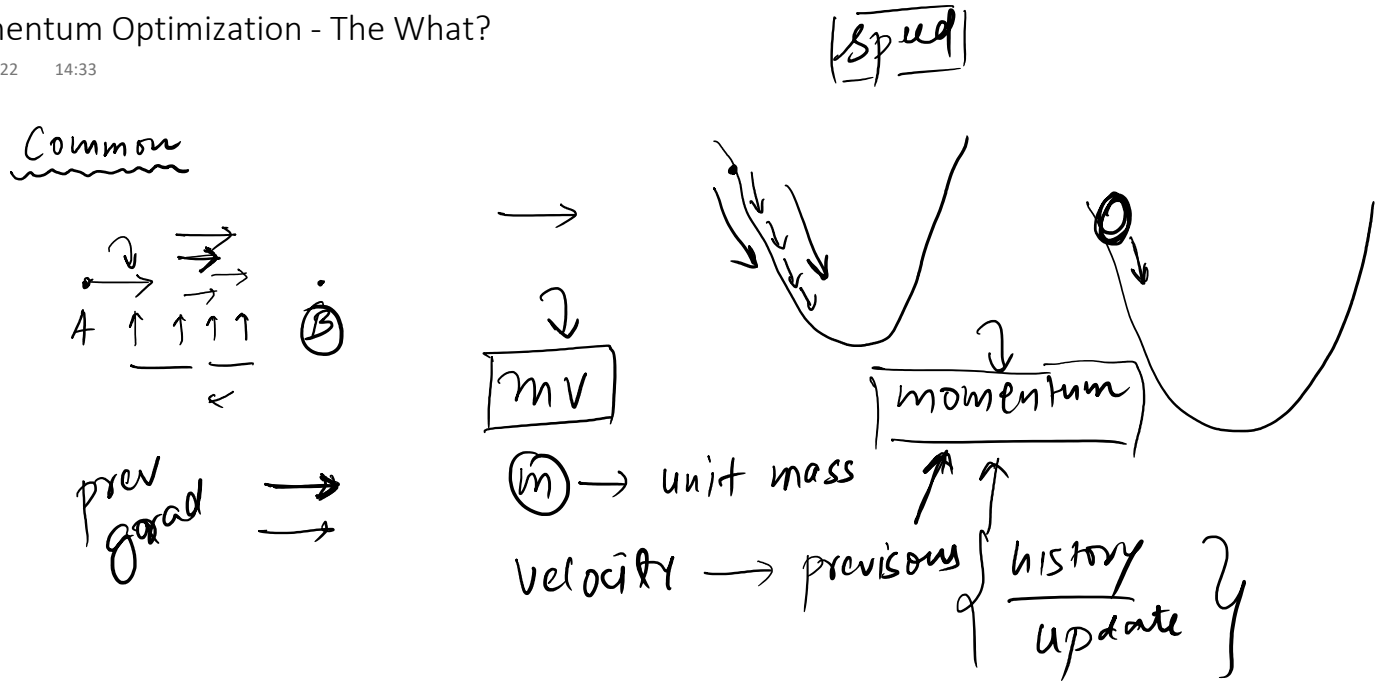
Momentum Optimization - The Why?

20 July 2022 14:28



Momentum Optimization - The What?

20 July 2022 14:33



Momentum Optimization - Mathematics(The How?)

20 July 2022 14:56

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

v → velocity

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

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

acceleration

momentum

0.9

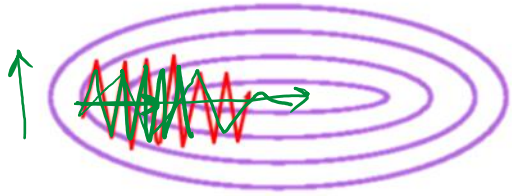
$0 < \beta < 1$

history of past velo- use

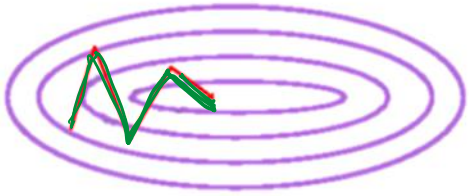
prev velocity

v_{t-2}

v_{t-3}



SGD



SGD momentum

faster convergence

Effect of beta

20 July 2022 15:09

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

SGD

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

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

$\beta = 0$ momentum → SGD

$\beta = 0$

decaying factor

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

0.9

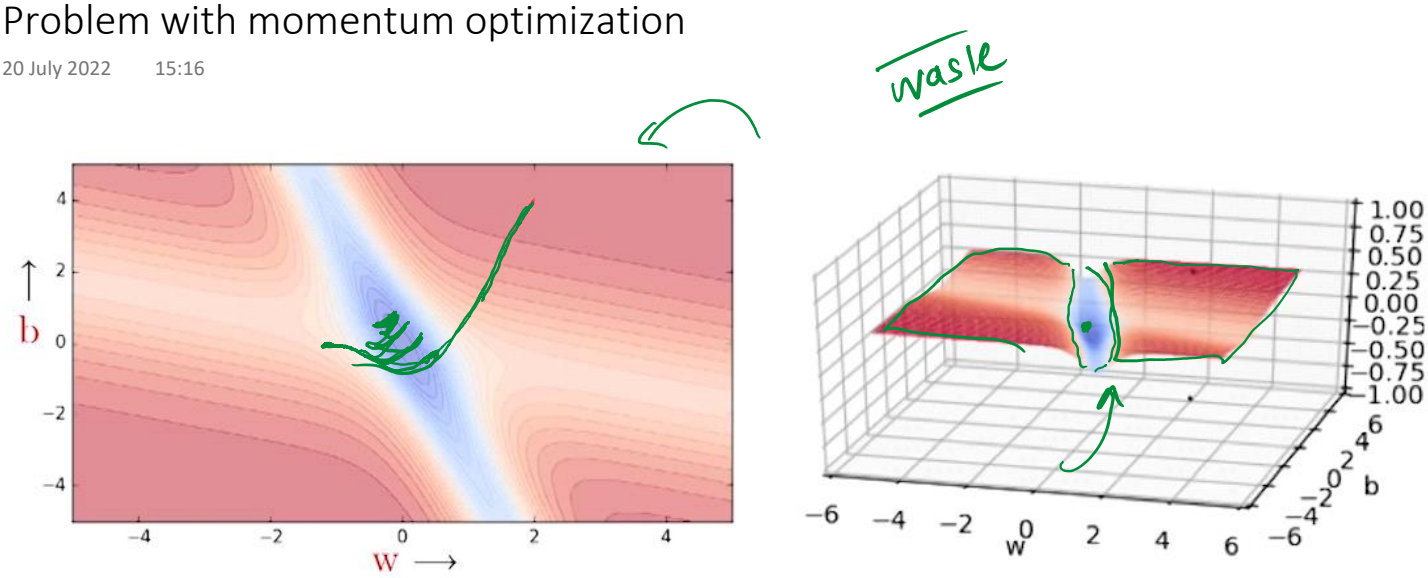
current avg past 10 velocities

up1 up2 . . up3

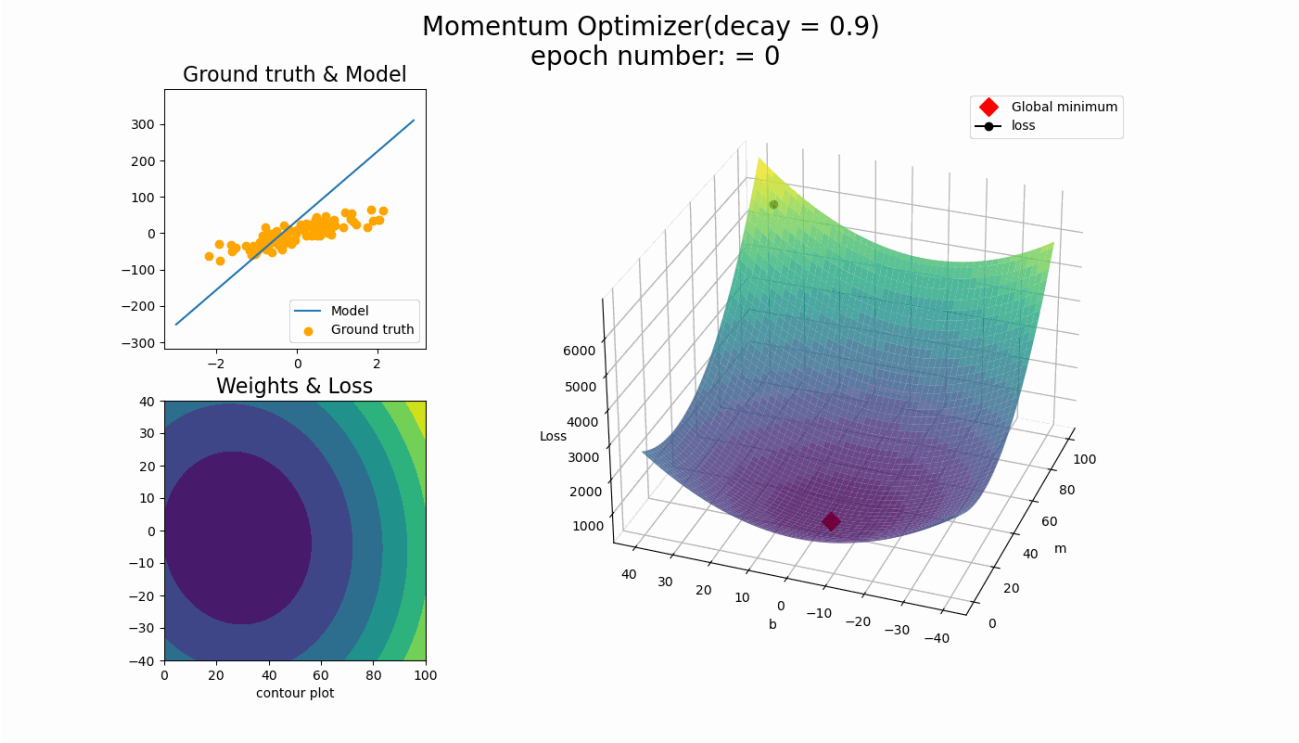
$\beta = 1$

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

Momentum

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

where

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

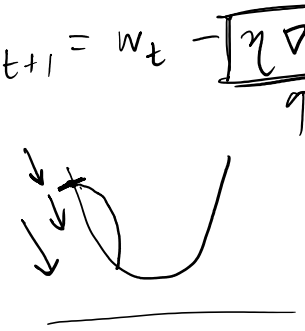
$\beta \rightarrow$ decay factor

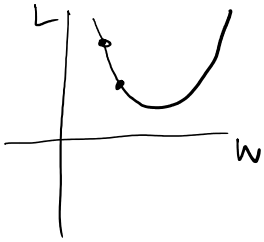
$$w_{t+1} = w_t - (\beta v_{t-1} + \eta \nabla w_t)$$

past velocity

gradient at that point

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

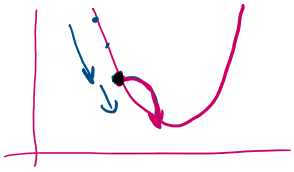




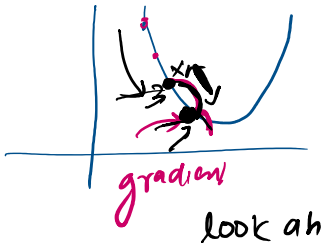
NAG

Nesterov Accelerated Gradient

better



history of velocity + gradient at that point

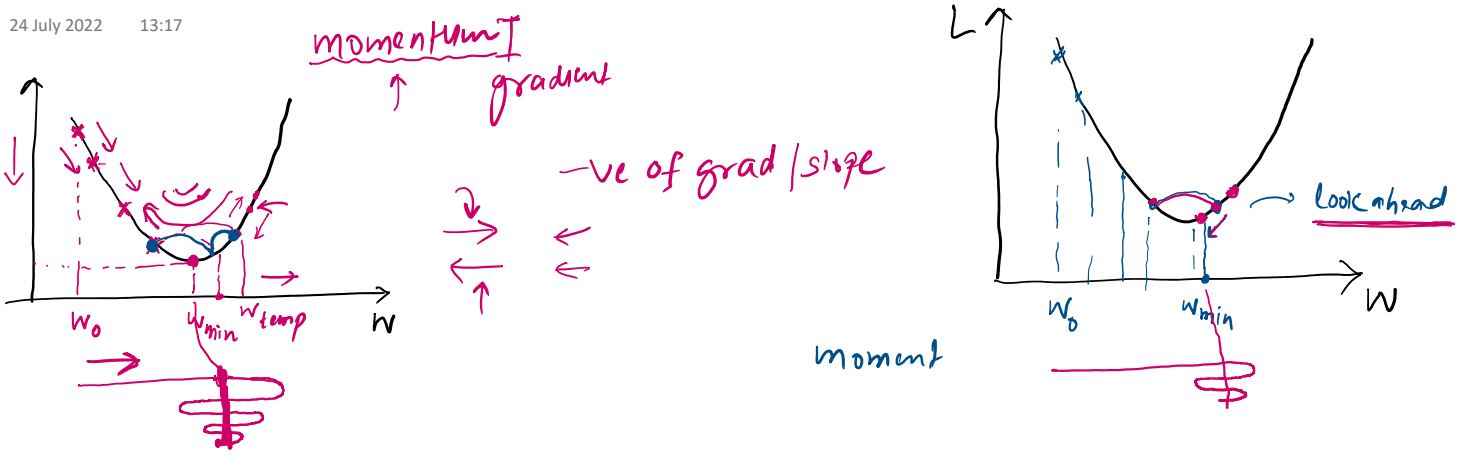


look ahead

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

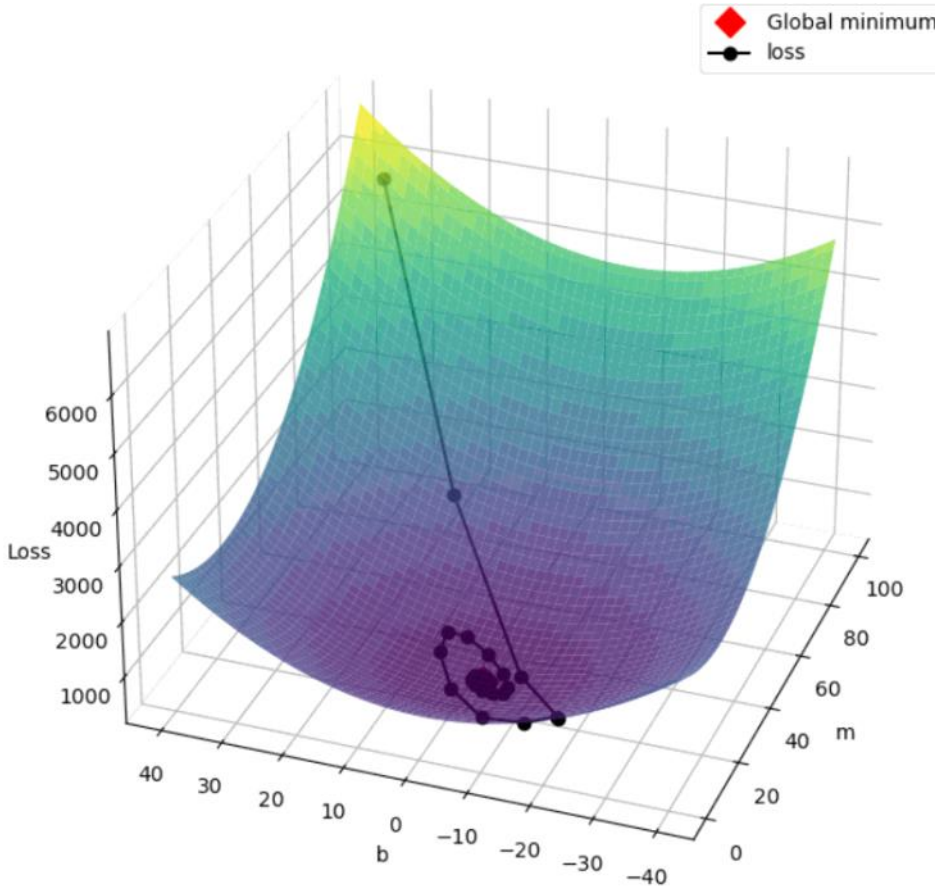
Geometric Intuition

24 July 2022 13:17



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

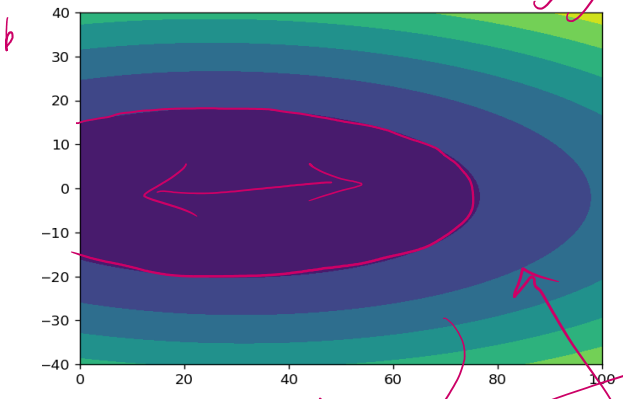
→ input features → scale
↳ different

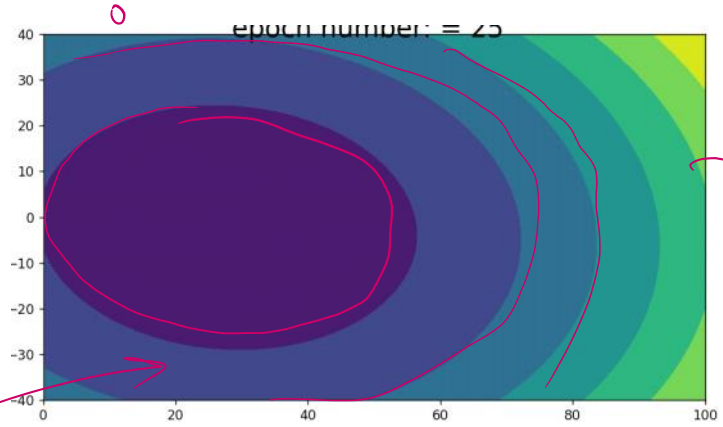
⇒ features → sparse → 0 (most)
↳ Adagrad

iq | cgpa | from iit | package
0 → 0,0,0,
1 →

Elongated bowl problem

sparse
x | y
0
0
0

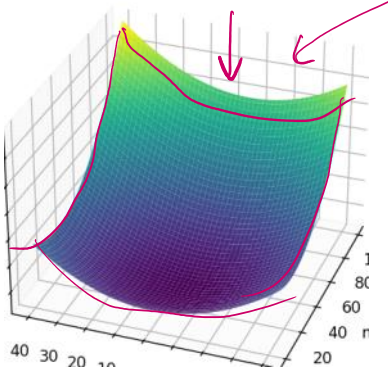


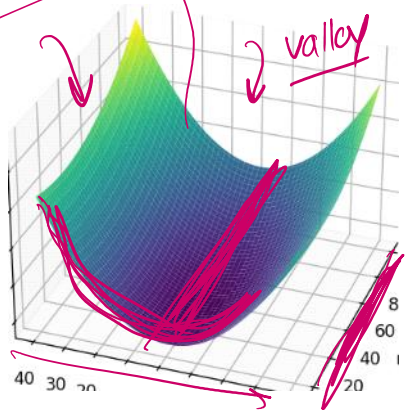


→ m

m

x | y





valley

loss → w,b

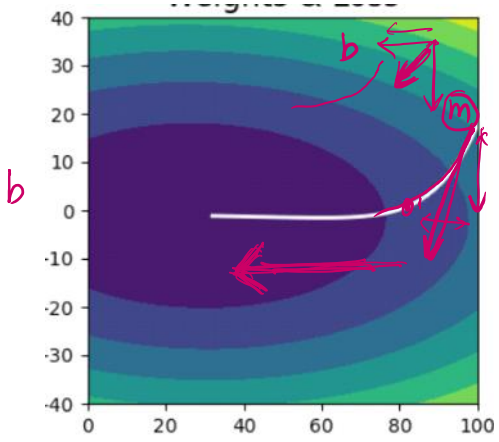
Bgd → momentum → NAG

Adagrad

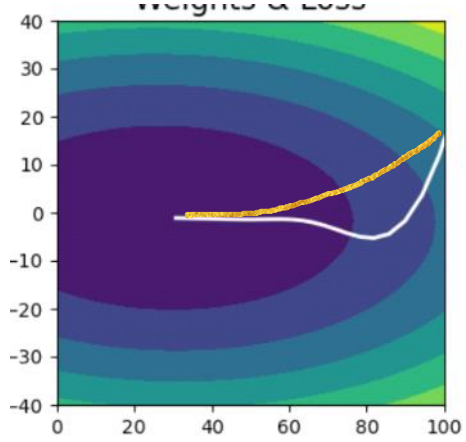
elongated
↓
slope → 1axis – changing
2nd axis constant

How optimizers behave(Why?)

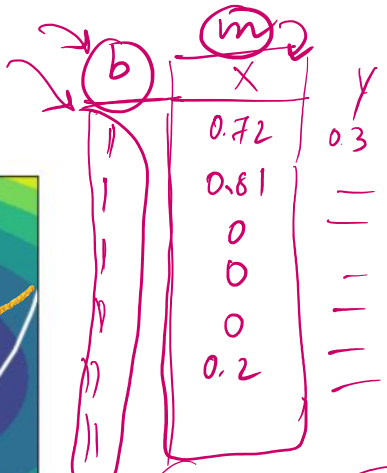
02 August 2022 18:44

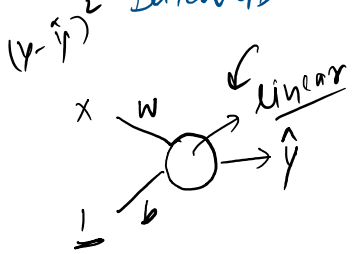


Batch GD



Momentum





BGD → sparspe

for i in epochs:

$$w = w - \eta\left[\frac{\partial L}{\partial w}\right]$$

$$b = b - \eta\left[\frac{\partial L}{\partial b}\right]$$ → big update in every epoch

$$\left[\frac{\partial L}{\partial w}\right] = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w}$$

$$= -2(y-\hat{y})[X]$$ add

$$\frac{\partial L}{\partial b} = [-2(y-\hat{y}) * 1]$$ bias num

100 rows → small number

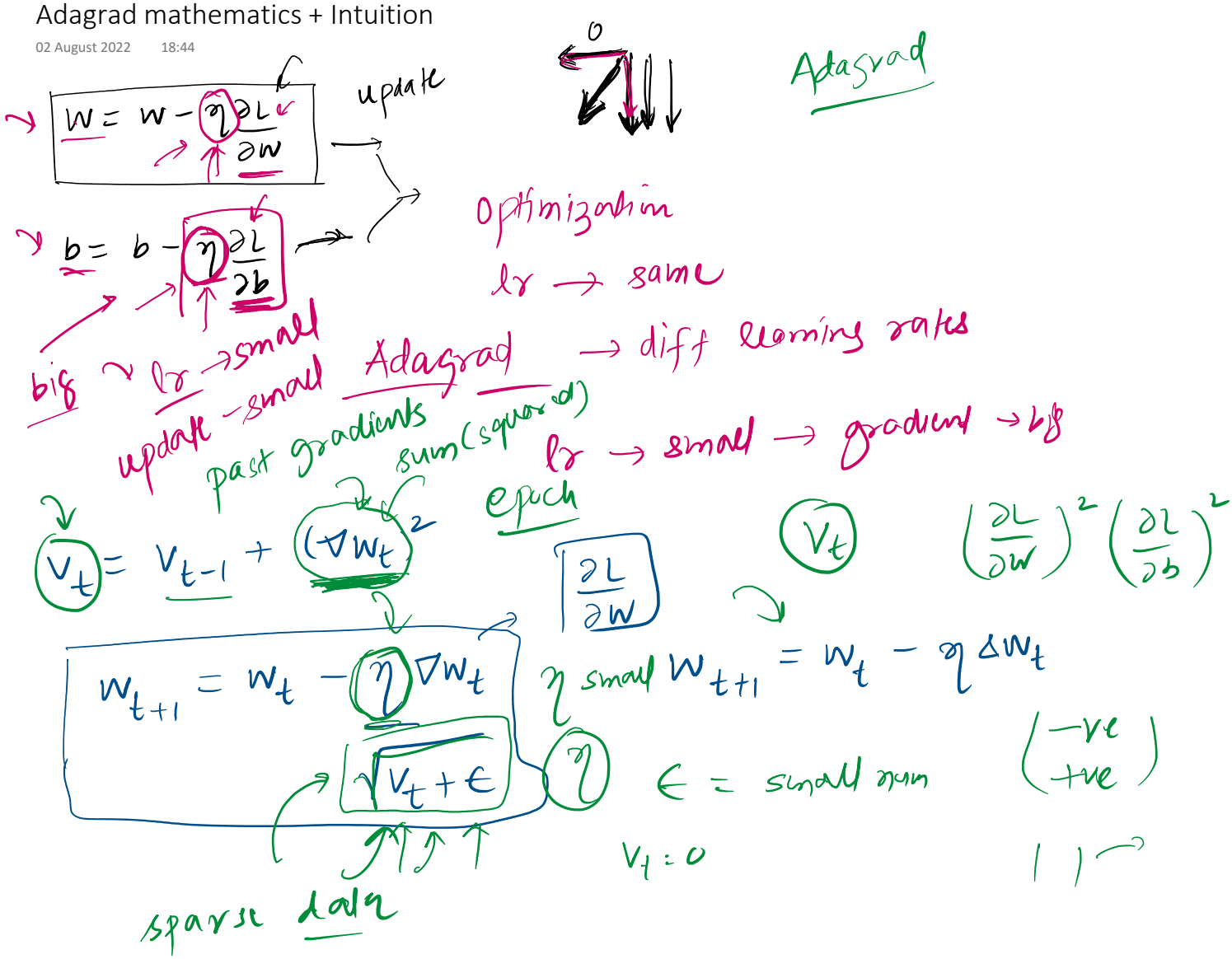
X sparse → small in every epoch

sparse col → small

non sparse → normal/big

Adagrad mathematics + Intuition

02 August 2022 18:44



Adagrad Demo

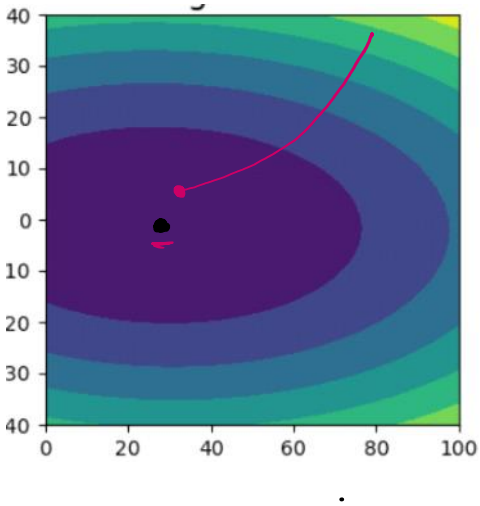
02 August 2022 18:44

Keras Implementation

02 August 2022 18:44

Disadvantage

02 August 2022 18:43



NN → use X → Linear regress

small
η → decrease

$\sqrt{v_t}$

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

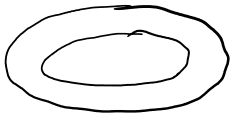
↓

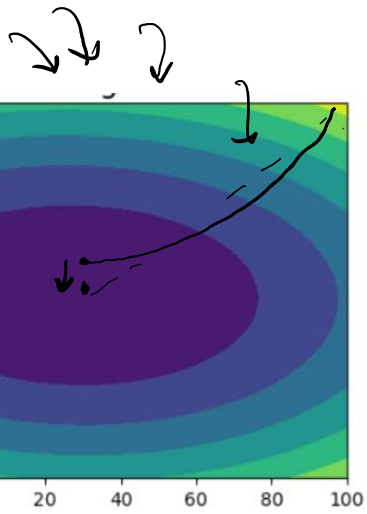
Adagrad ← sparse data

BAD → momentum

Disadvantage → learning rate ↓

Update → small





Mathematical Formulation

03 August 2022 14:04

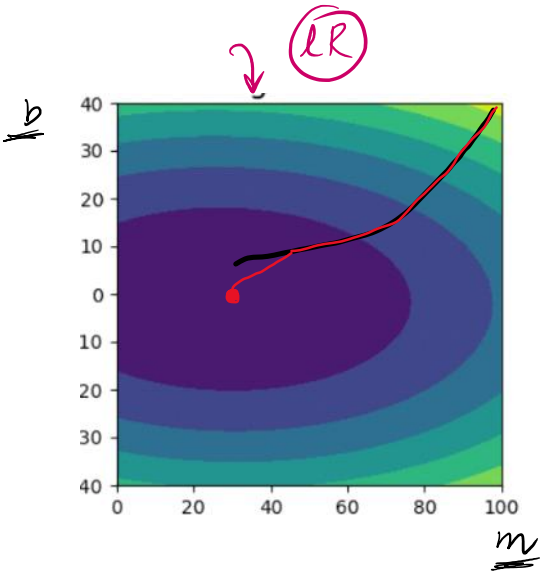
epochs

Adagrad

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

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

(b) small



b | x | m | y
1 | 0 | 0 | 0
1 | 0 | 0 | 0
1 | 0 | 0 | .72
1 | . | . | .
1 | , | , | ,

↓ b move

non-sparse

RMSprop

$v_t = \beta v_{t-1} + (1-\beta)(\nabla w_t)^2$ — exp decaying avg

$\beta = 0.95$

old epoch

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

→ $v_0 = 0$

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

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

→ $v_3 = 0.95 \times 0.95 \times 0.05 (\nabla w_1)^2 + 0.95 \times 0.05 (\nabla w_2)^2 + 0.05 (\nabla w_3)^2$

smaller ← epoch 1 | epoch 2 | epoch 3 ← larger

v_t shoot → η small

Adagrad → NN → non-convex optimization

↳ convex optimiz → linear reg

↘ convex → xmax
optimizs eg

global minima RMSprop

Disadvantage

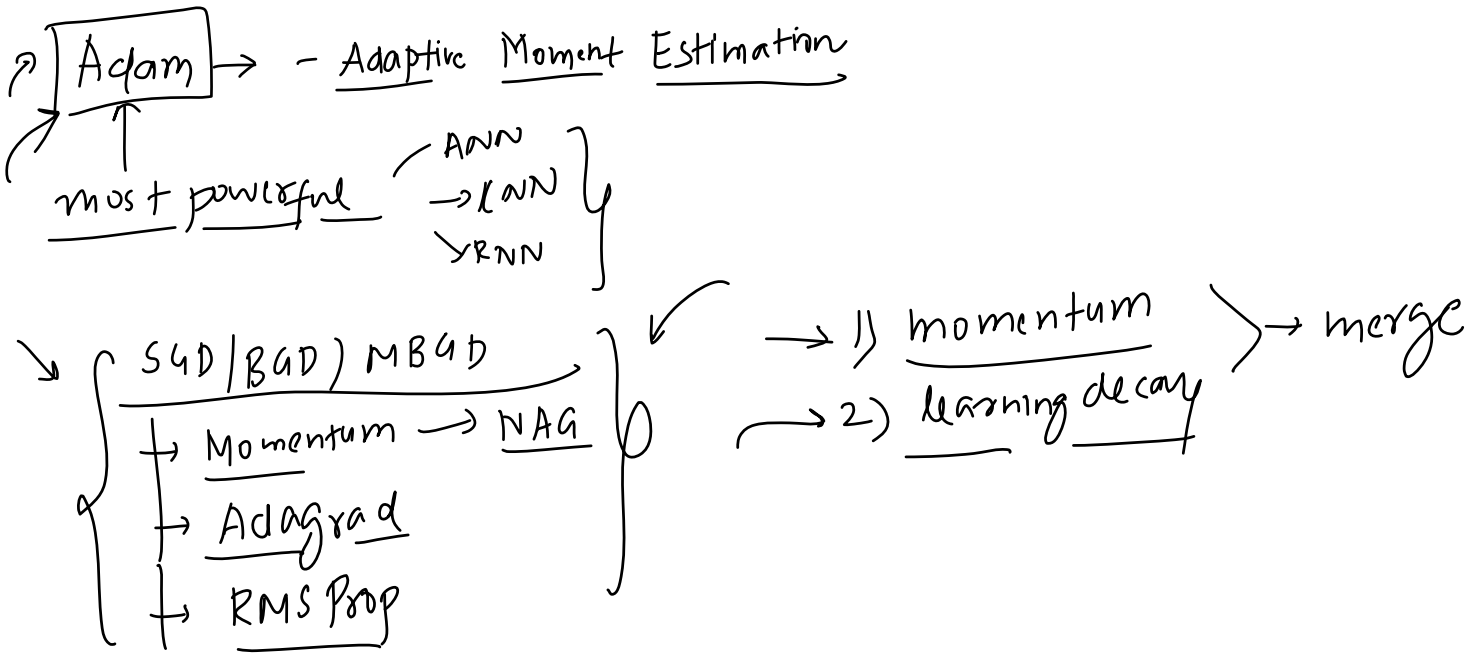
03 August 2022 14:54

No,
Optimization

↓
Adam

Introduction

04 August 2022 10:45



Mathematical Formulation

04 August 2022 10:45

Keras

Bias correction

epoch #

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{v_t + \epsilon}} * m_t \rightarrow \hat{m}_t = \frac{m_t}{1-\beta_1^t} \rightarrow \hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$\beta_1 = 0.9$

$\beta_2 = 0.99$ → Keras

where

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

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

$\eta = 0.1$

0.01

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

$v_0 = 0$

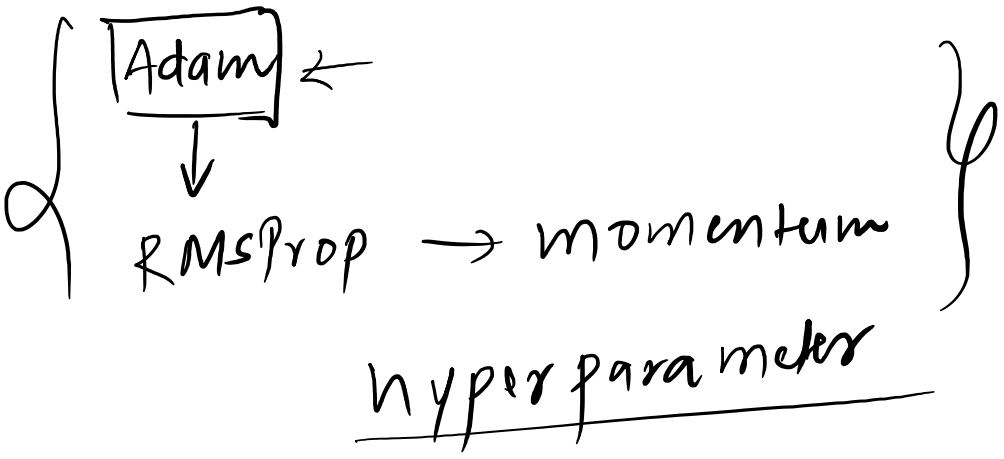
$0 \rightarrow$

Demo

04 August 2022 10:45

Verdict

04 August 2022 10:45

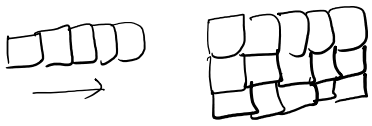


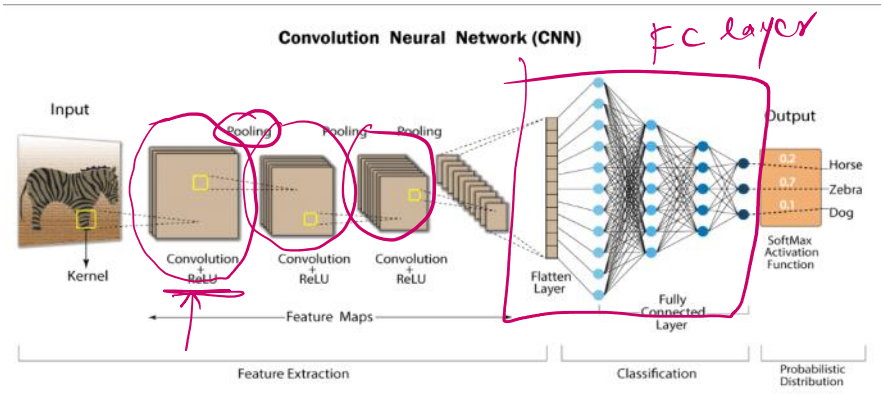
What is a CNN?

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 ↓ convolution ⊗ matrix mult.
CNN ↗
ANN →

Inspirations
↳ visual cortex

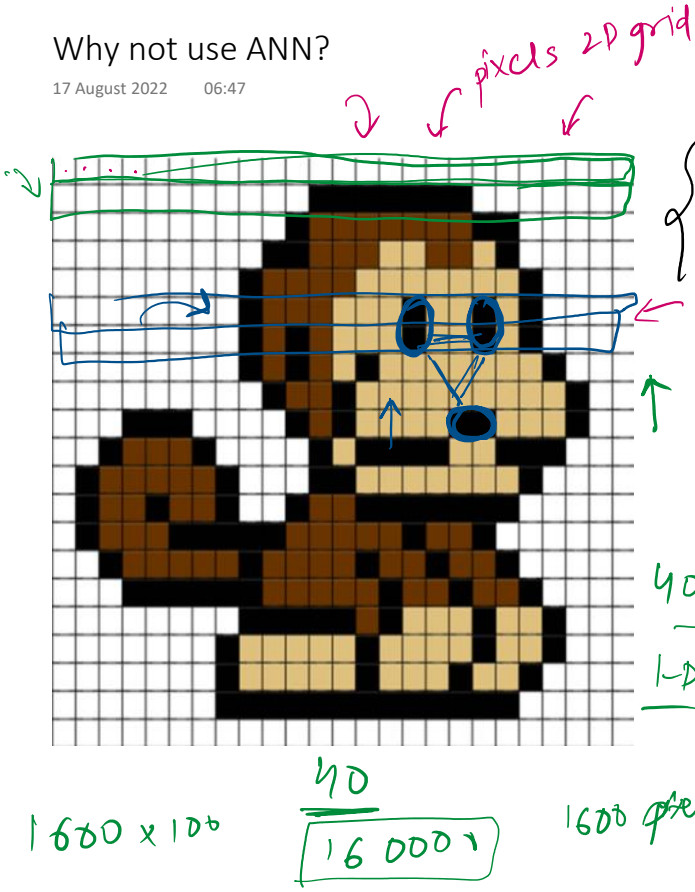
1998 Yann Lecun → AT&T →
↳ Microsoft → OCR, hand writing recog } CNN

{ facial recog, self driving } CNN RNN

Why not use ANN?

17 August 2022 06:47

CNN >> ANN

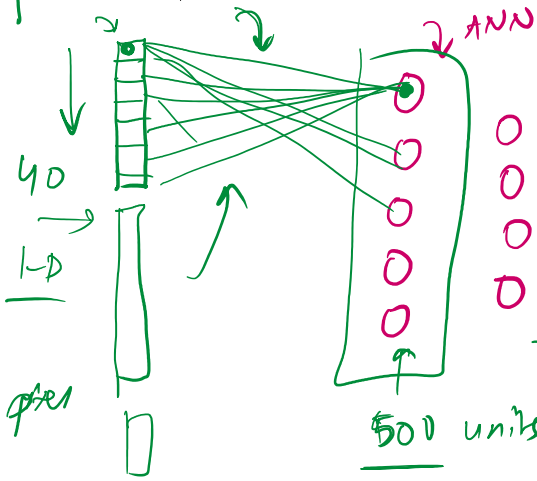


1. High Computation Cost ✓
2. Overfitting →
3. Loss of imp info like spatial arrangement of pixels

1D

40 x 40 image

MNIST → ANN → 98%



ANN

1000 x 1000

Overfitting

1000 x 1000 x 500

1600 x 100

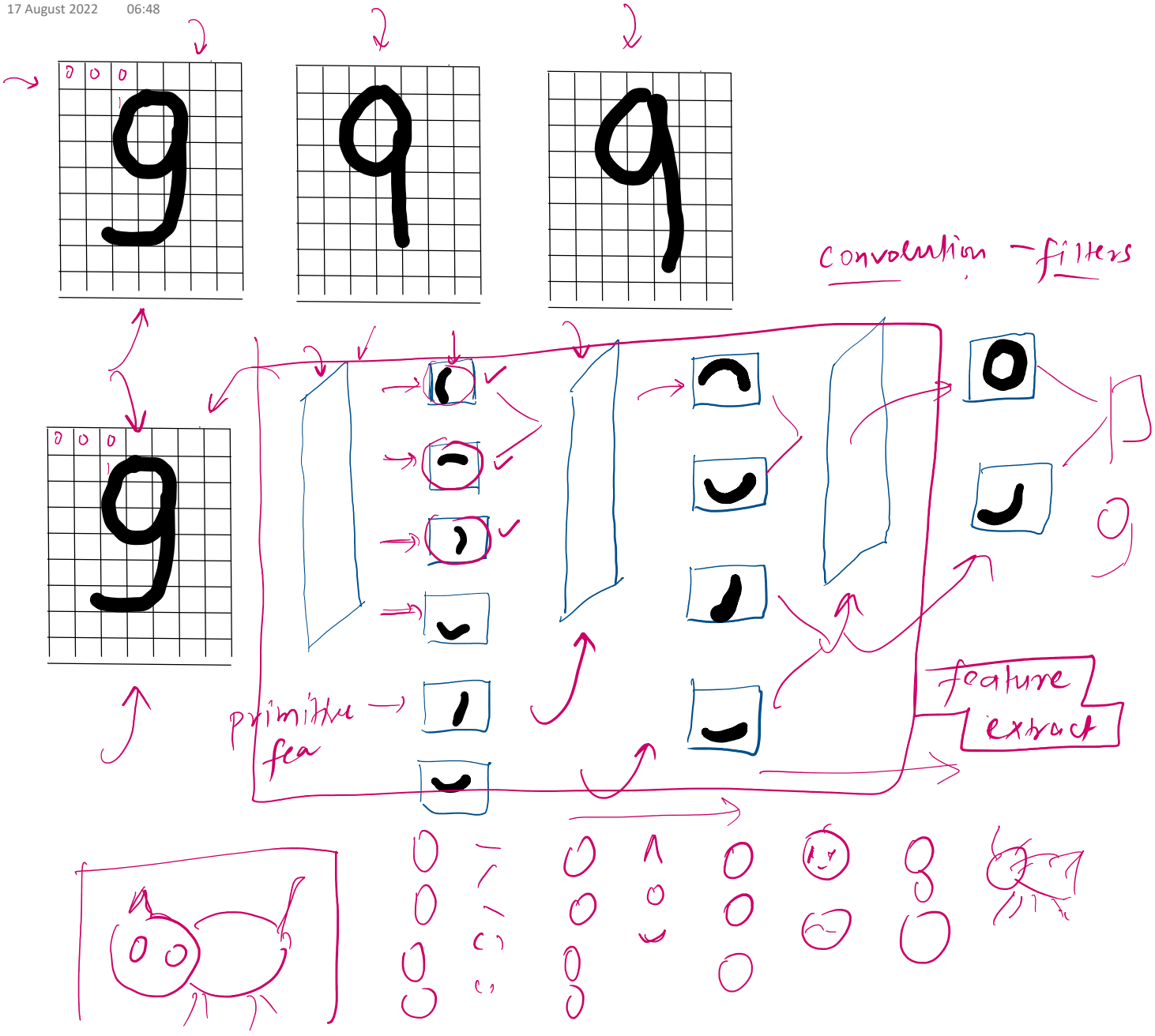
40

16000

1600 pixels

CNN Intuition

17 August 2022 06:48

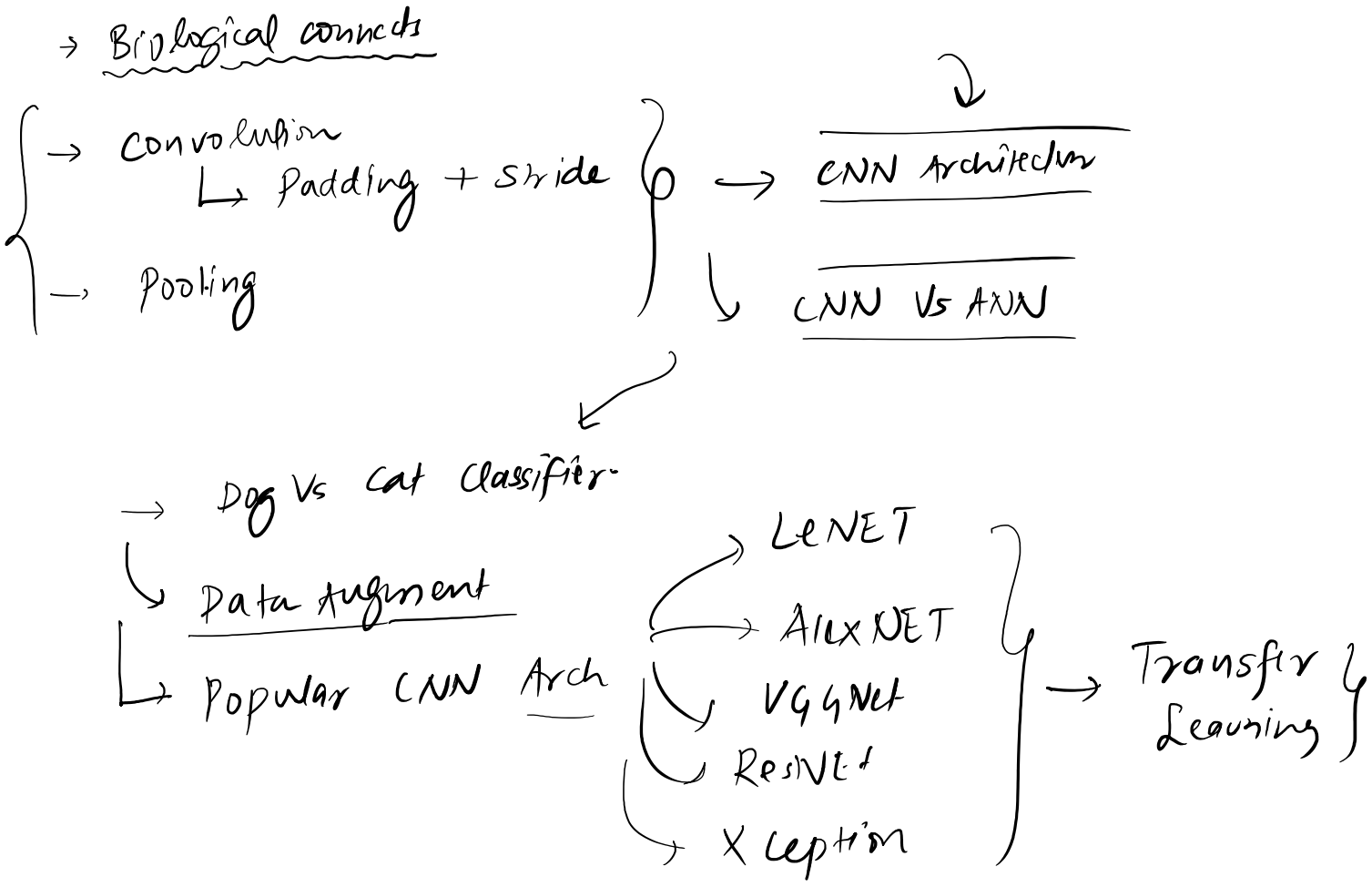


CNN Applications

17 August 2022 06:48

Roadmap

17 August 2022 06:48



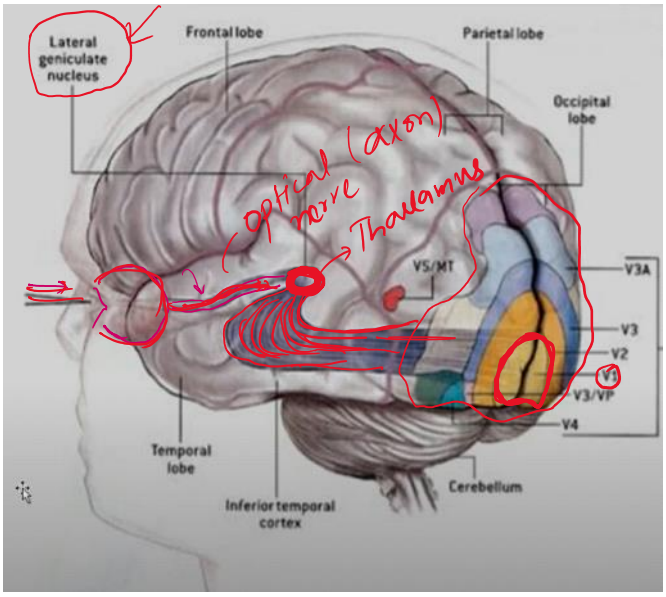
Human Visual Cortex

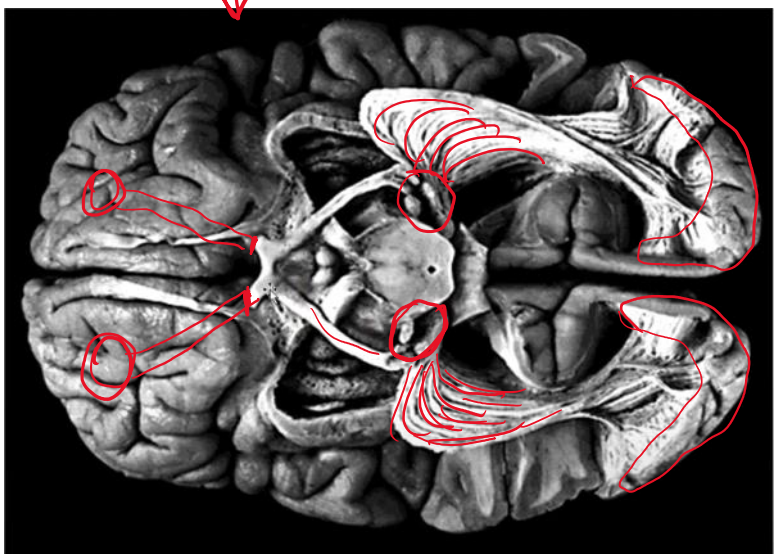
18 August 2022 16:05

2D sheet

primary Vcortex

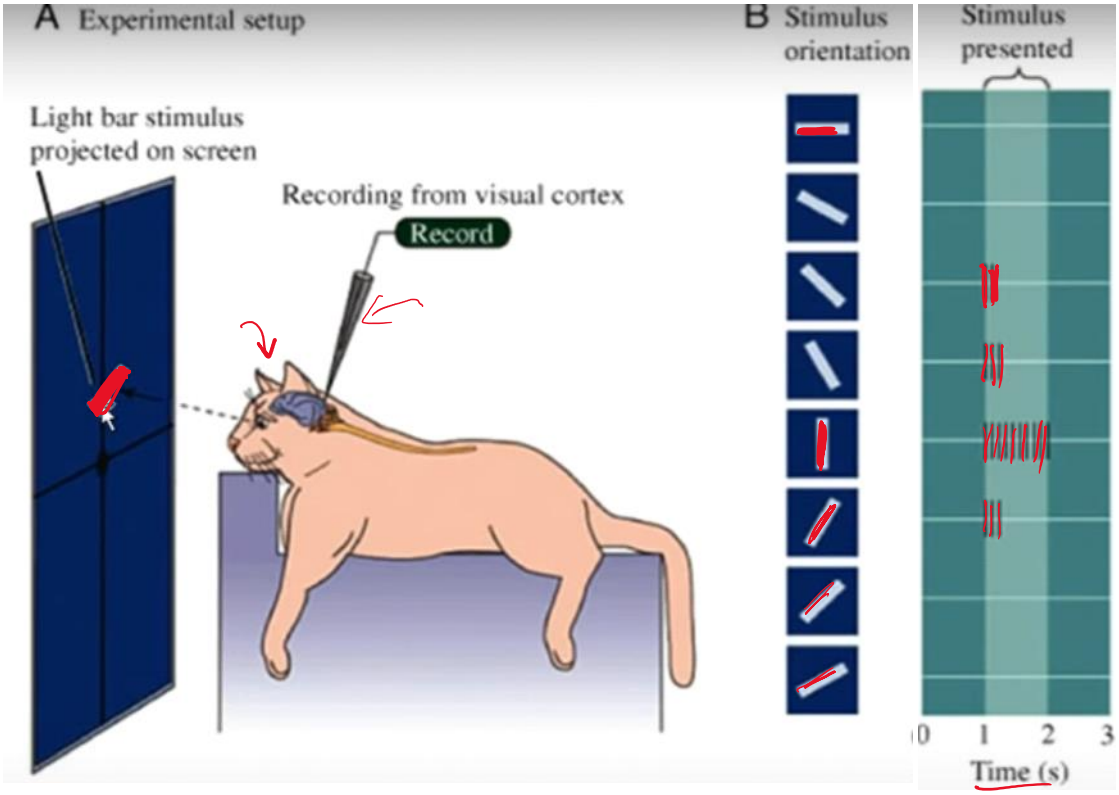
actual brain

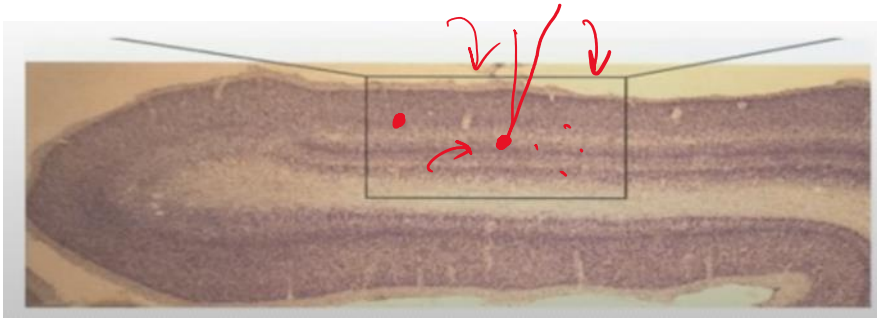




The Experiment

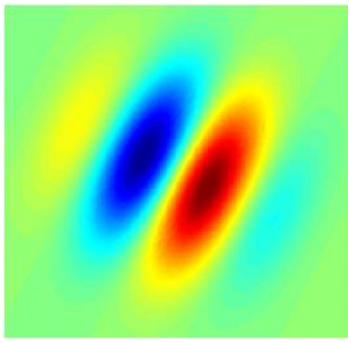
18 August 2022 16:09





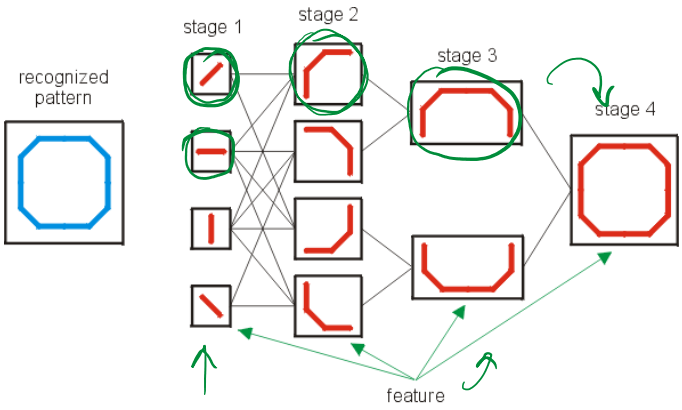
Conclusion

18 August 2022 17:57

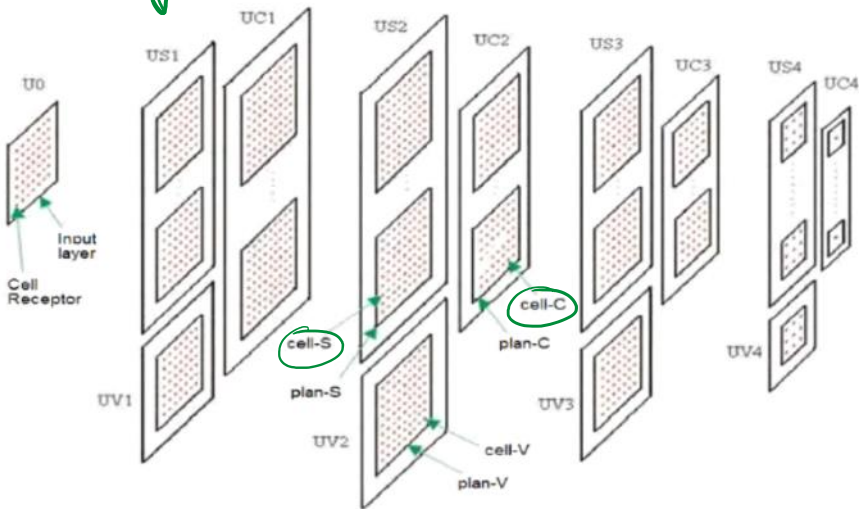


Development

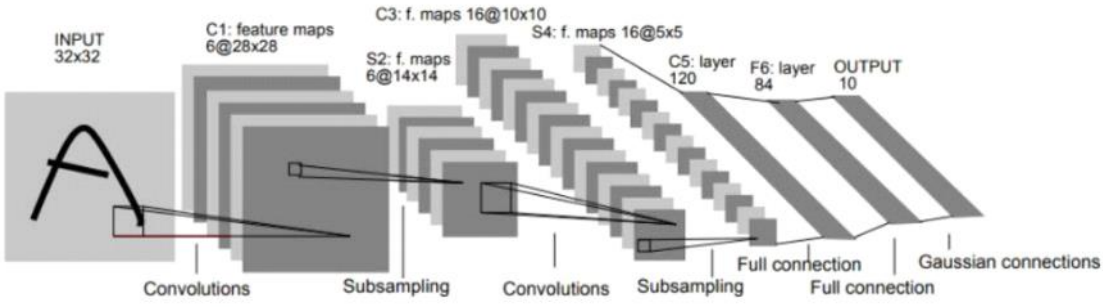
18 August 2022 16:15



Neocognitron → Fukushima



Yann Lecun → CNN → Backprop Convolution

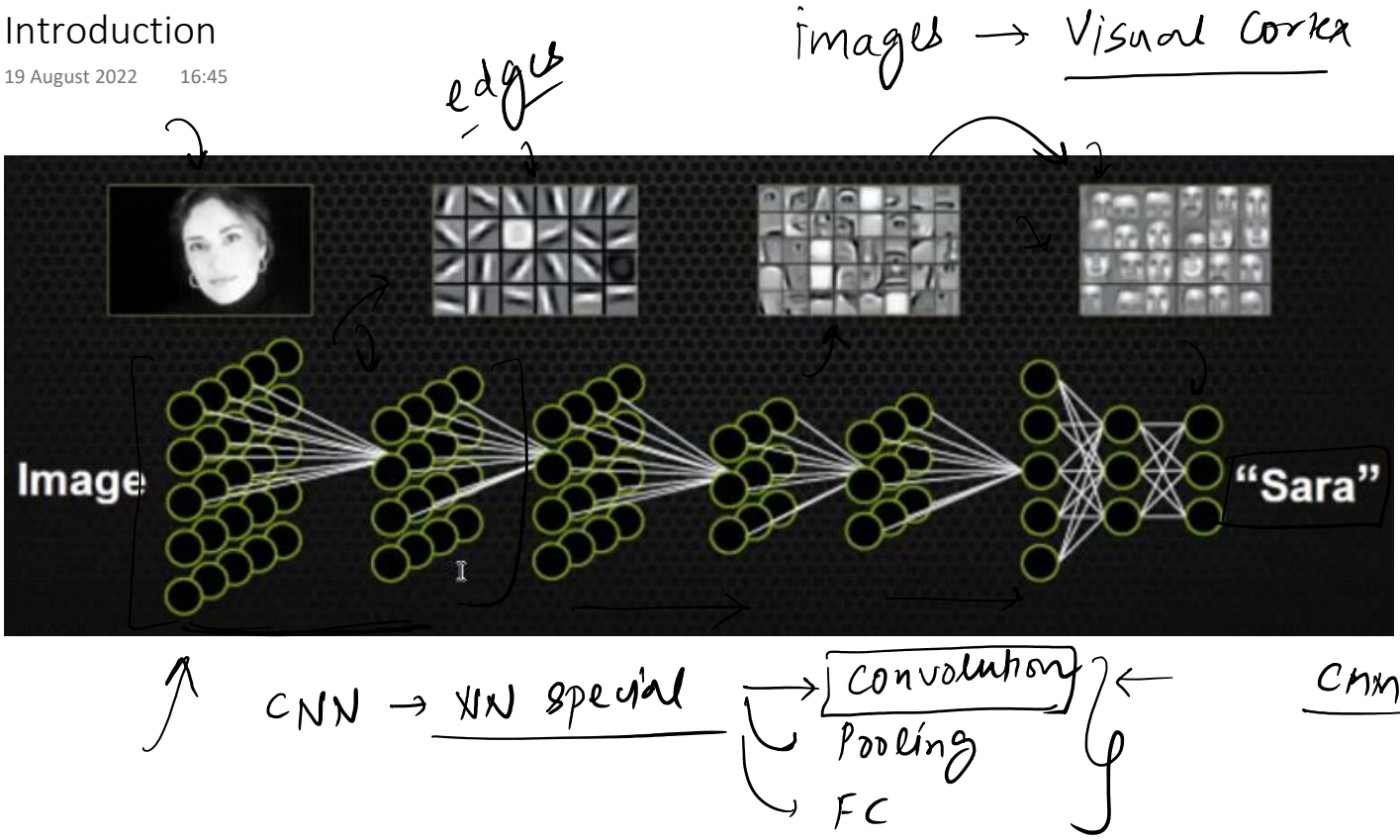


early CNN

2012 → AlexNET → ImageNET → CNNs

Introduction

19 August 2022 16:45



Basics of Images

19 August 2022 16:53

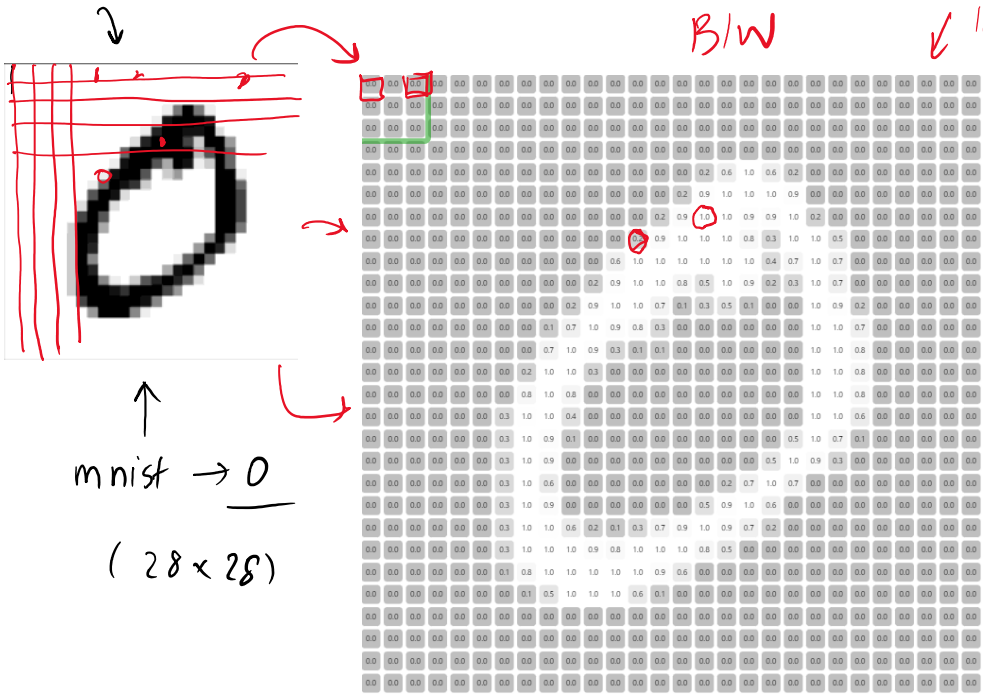
0-255 → 0-1

B/W

1 channel

Greyscale → B/W

RGB → Colored images



28 x 28 pixels

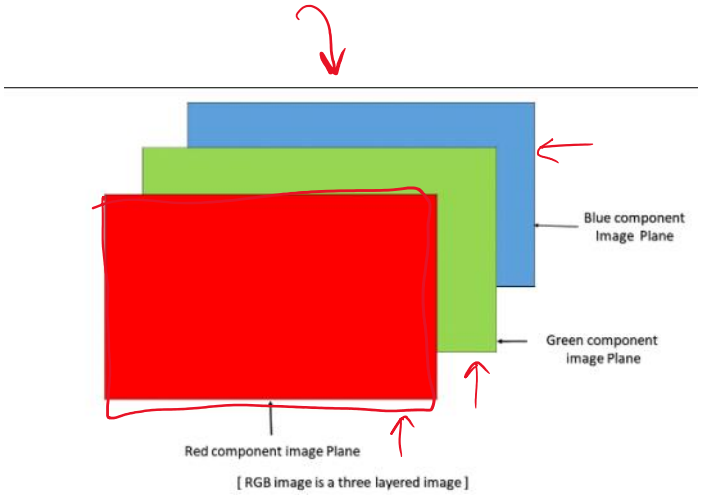
2D arrays →

2D numpy arrays

mnist → 0

(28 x 28)

3 channels → Red Green Blue (primary)

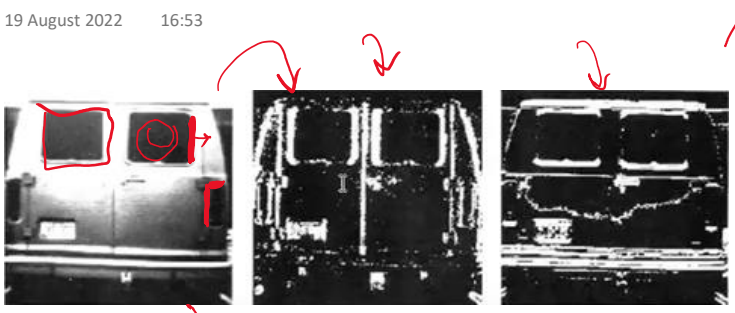


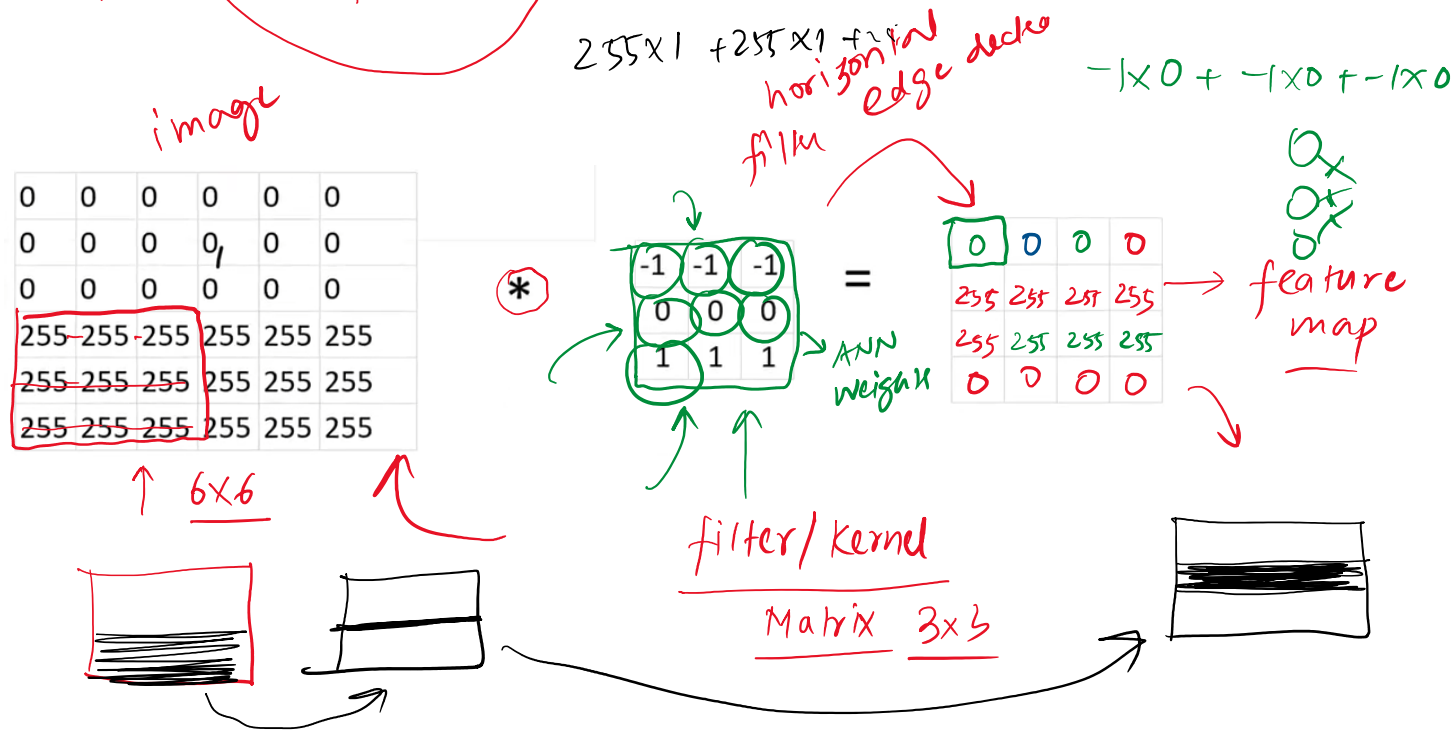
228 * 228 * 3

3 channels

Edge Detection (Convolution Operation)

19 August 2022 16:53





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

=

(6 x 6) (3x3) (4x4)

(28x28) → (26x26)

nxn

(3x3) → ? (26x26)

mxm → (n-m+1) x (n-m+1)

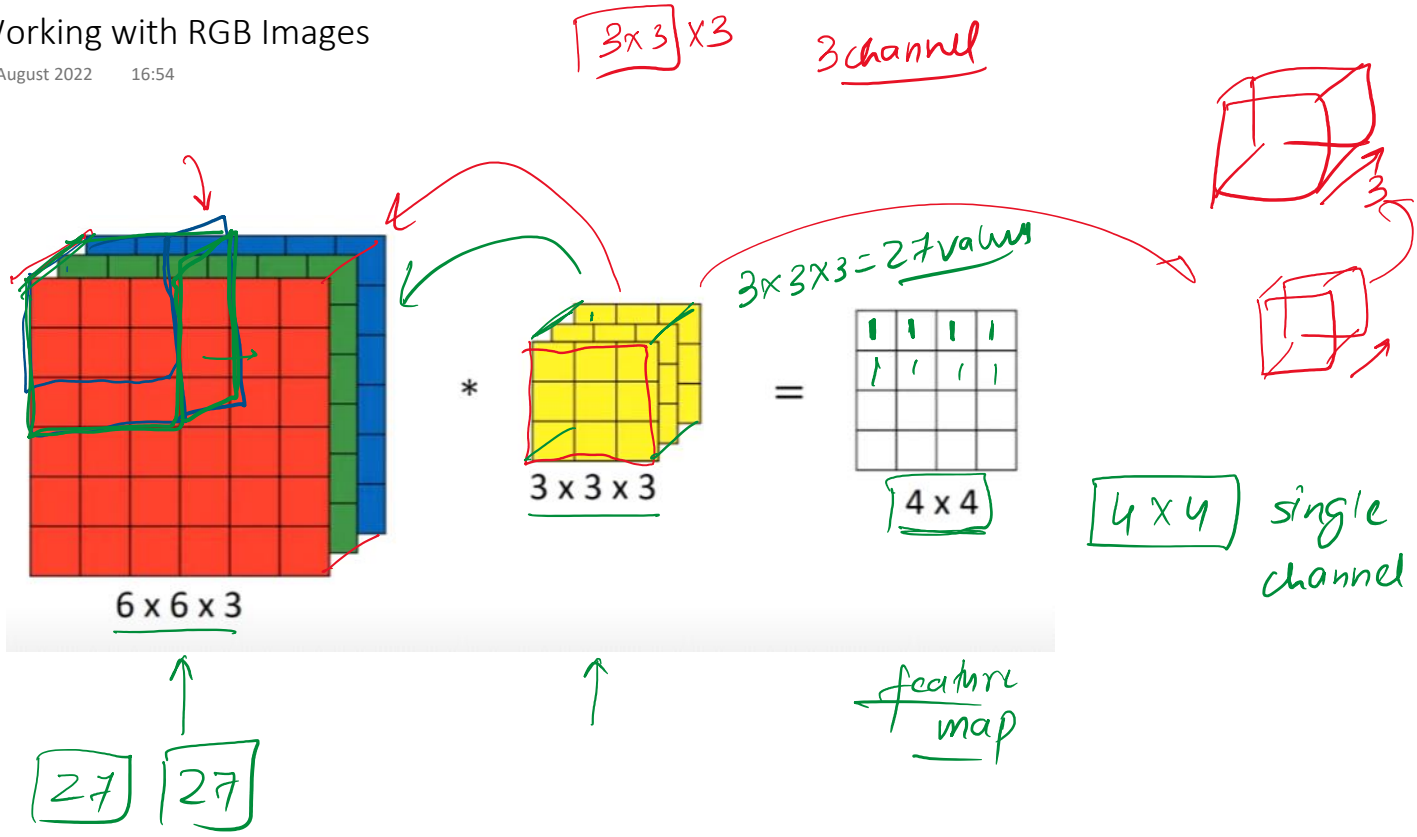
(64x64) (3x3) → (62x62)

Demo

19 August 2022 16:54

Working with RGB Images

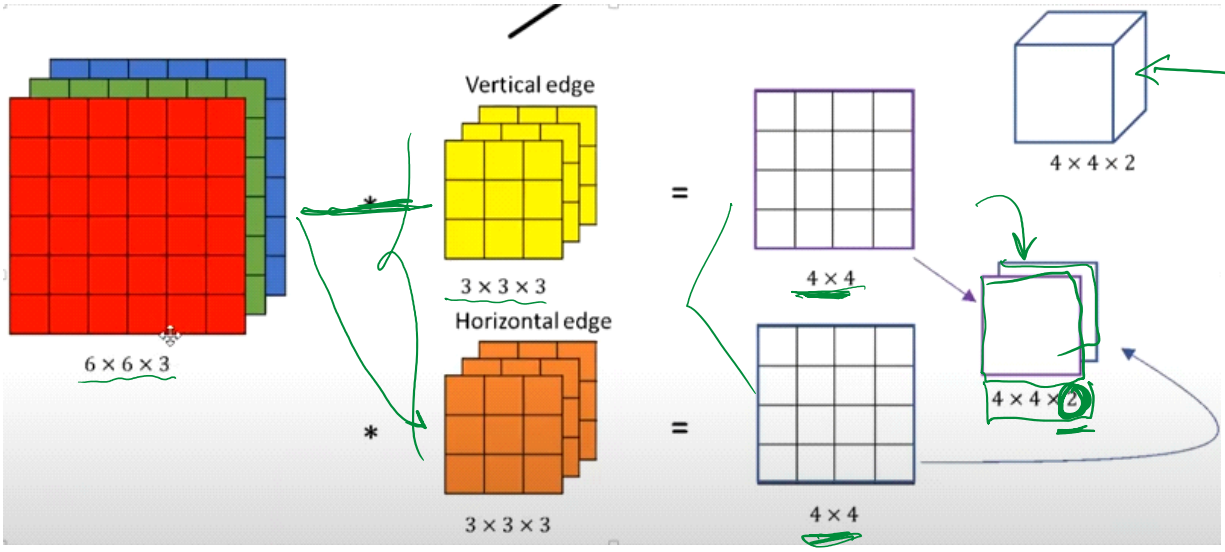
19 August 2022 16:54



$m \times m \times c$ $n \times n \times c$ $\longrightarrow$ $(m-n+1)(m-n+1)$ single channel

Multiple Filters

23 August 2022 08:24



#filters used

Image taken from Andrew NG's lecture

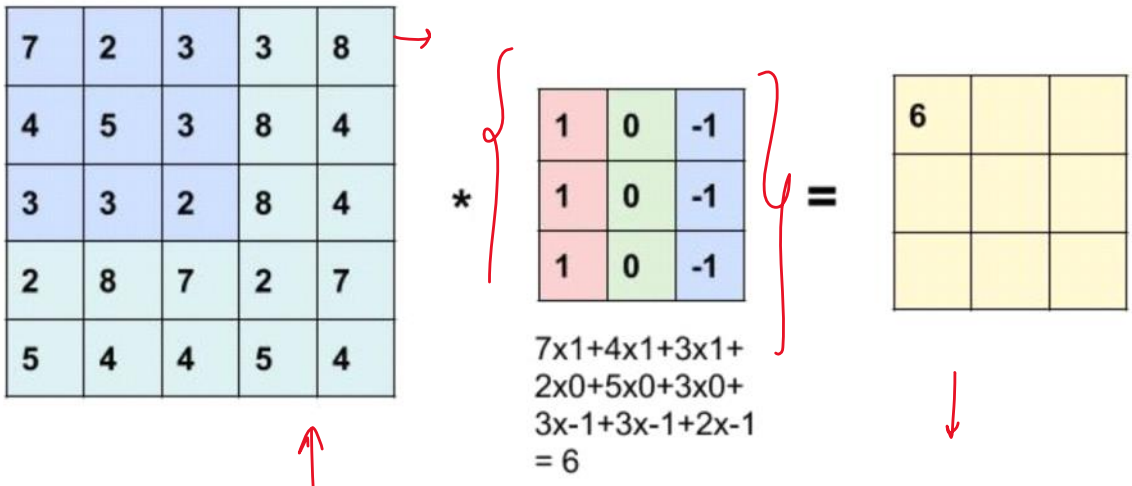
10 → 4 × 4 × 10

Problem with Convolution

26 August 2022 14:25

What is Padding?

26 August 2022 14:26



$n-f+1=n$

$n \times n$ → 5×5 → 7×7

$f \times f$ → 3×3

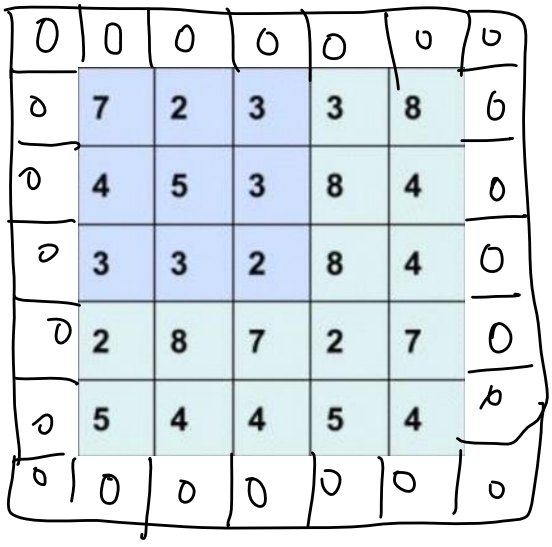
$\rightarrow (n-f+1)(n-f+1)$

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

$n-f+1 = 5$

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

$n=7$



7×7

convolution

Zero padding

$5 \times 5 \longrightarrow 3 \times 3$

$5 \times 5 \longrightarrow 3 \times 3$

$(n - f + 1)$

↓

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

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

$= 7 - 3 + 1 = 5$

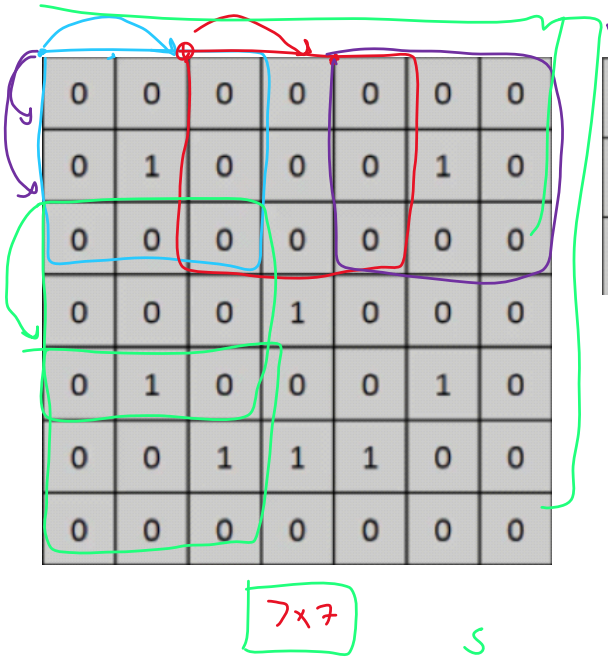
Keras

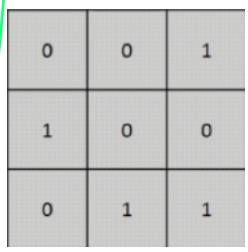
Valid

Same

Strides

27 August 2022 08:23





Stride = 1

(1,1)

3 x3

right

bottom

→ result feature

Stride = (2,2)

Stride = 2 →

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

2+1 = 3

s

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

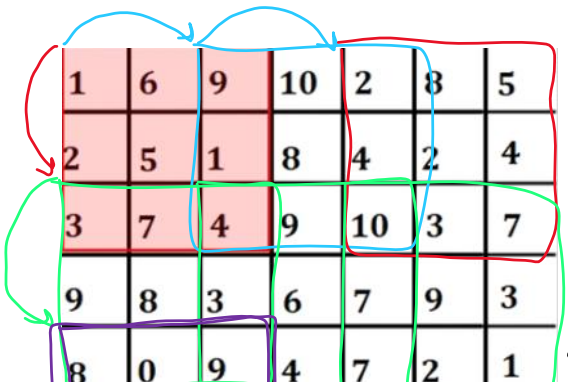
$\left[\frac{n+2p-f}{2} + 1\right]$ → strided convolution

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

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

Special case

Stride = 2



2x3

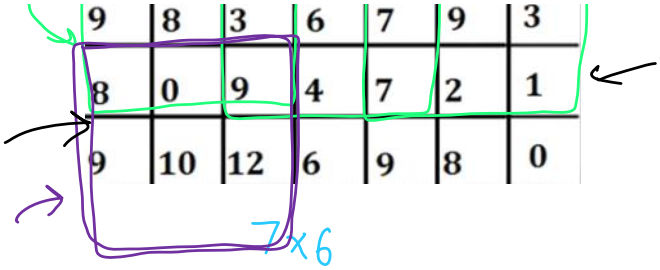
$\frac{6 \times 7}{3 \times 3}$

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

19 → 1

1.1 → 1

floor



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

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

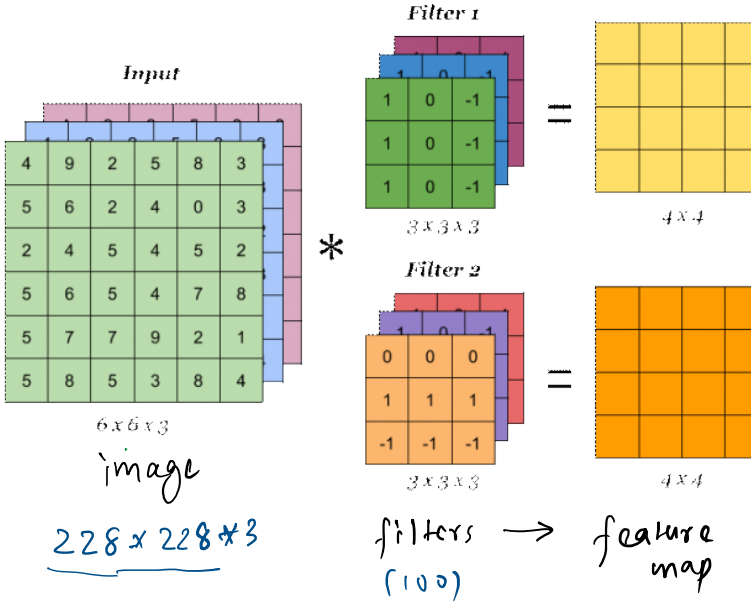
$$\frac{28+2-3}{2}+1$$

$$\frac{13.5}{}+1$$

$$13+1=14$$

The Problem with Convolution

01 September 2022 09:55



stride=2

1) Memory issue
2) Translation variance

reduce size of feature map

pooling

image

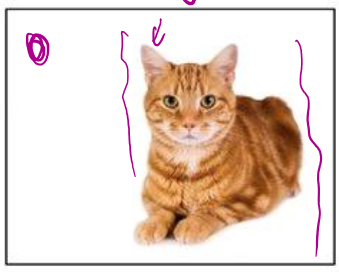
228 x 228 * 3

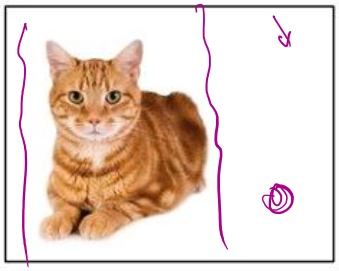
filters (100) → feature map → (226 * 226) * 100 * 32 → 19 MB

1 training → 100

1.5 GB

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

*

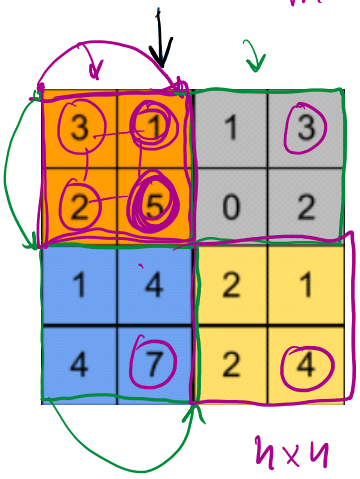
-1	-1	-1
0	0	0
1	1	1

= relu ((4x4))

Maxpooling
Minpooling
Avg pooling
L2 pooling
Global pooling

feature map (non-linear)

max pooling



4x4

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

receptive field

5	3
7	4

2x2

feature map

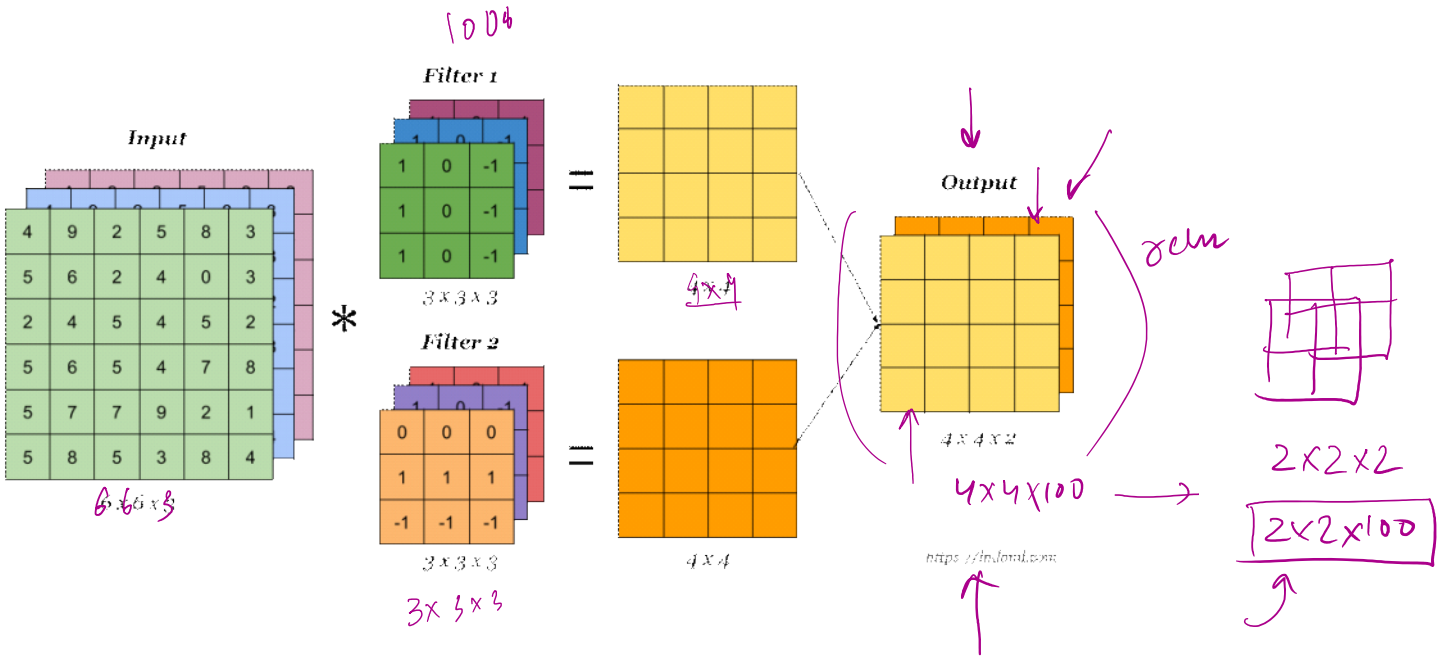
low level details eliminat

Demo

01 September 2022 09:57

Pooling on Volumes

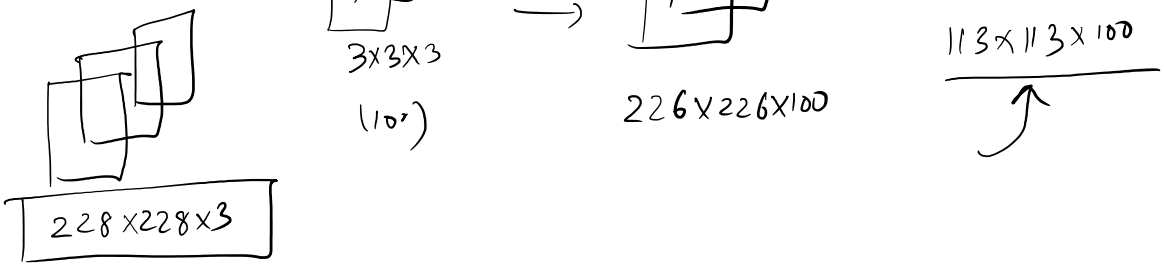
01 September 2022 09:56



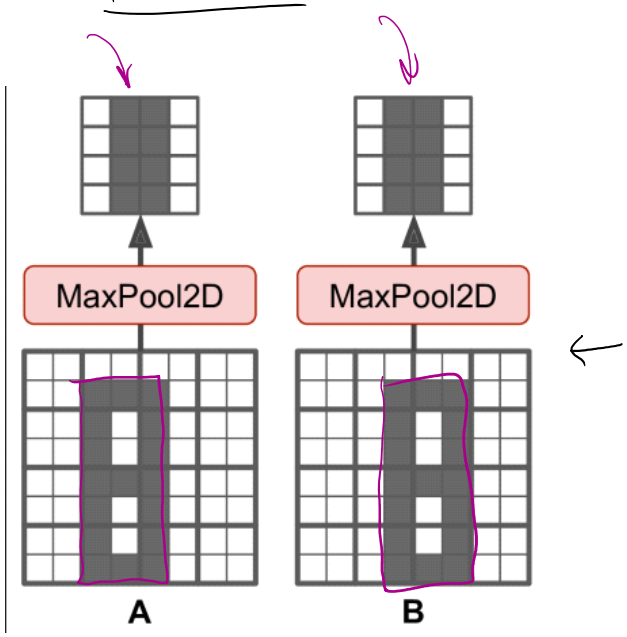
Advantages of Pooling

01 September 2022 09:56

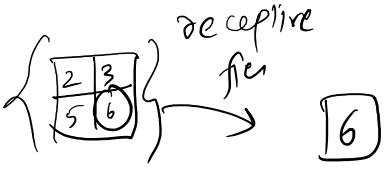
1) reduced size

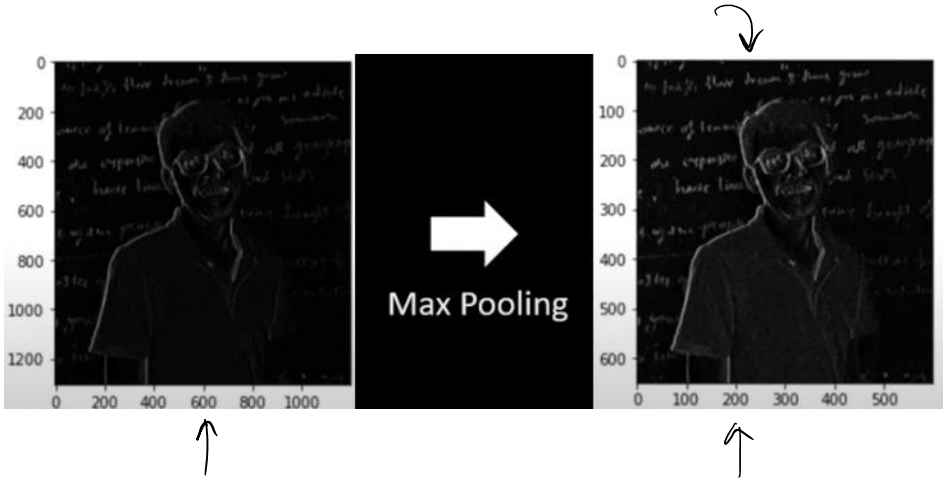


2) Translation invariance

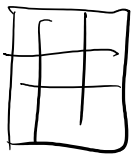


3) Enhanced features (only in case of Max pooling)



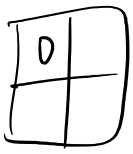


4) No need of training min



max pooling → aggregate

avg pooling



faster

→ (2x2)

→ 2

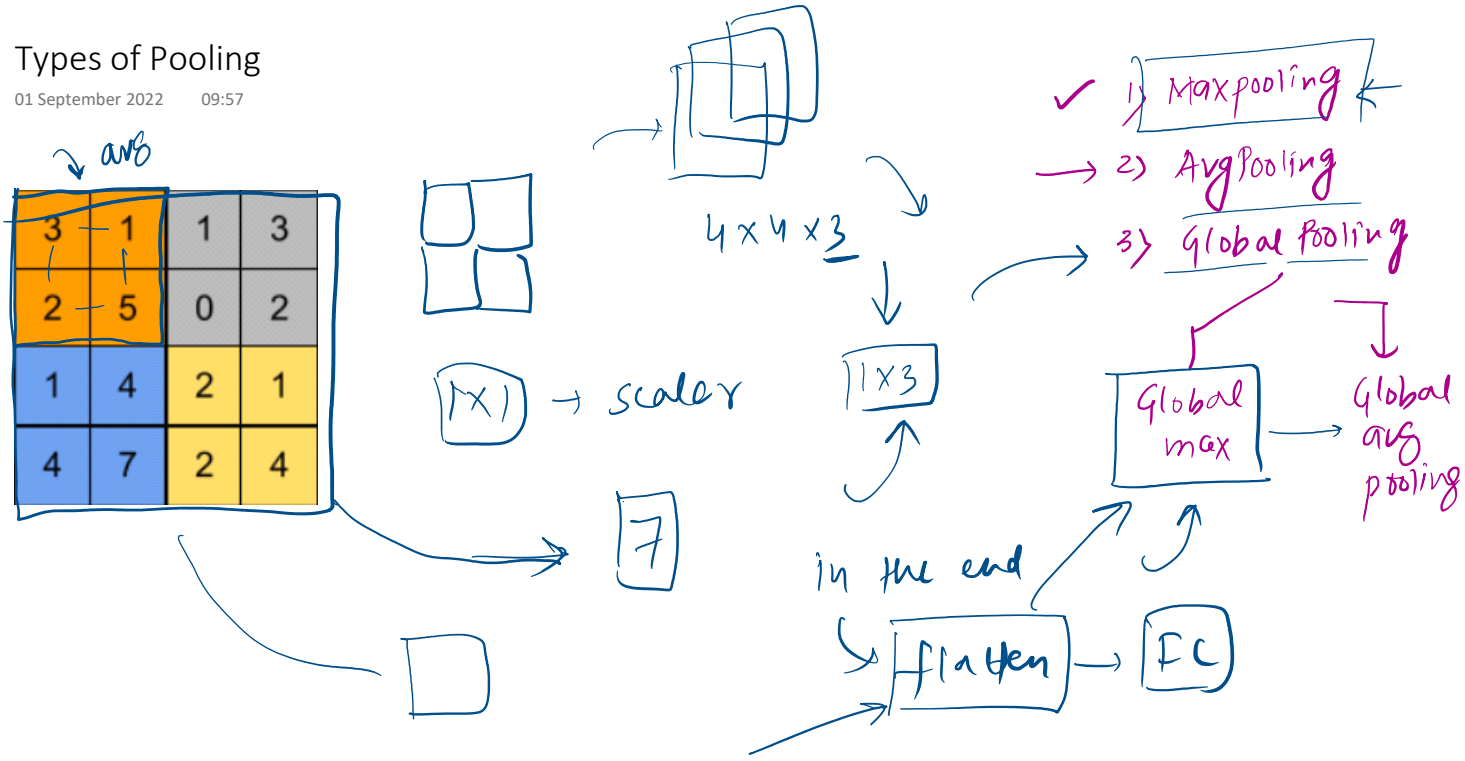
→ type

Keras Code

01 September 2022 09:56

Types of Pooling

01 September 2022 09:57



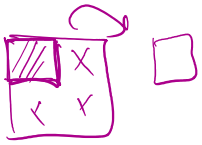
Disadvantages of Pooling

01 September 2022 09:57

lose a lot of info

4x4 → 16

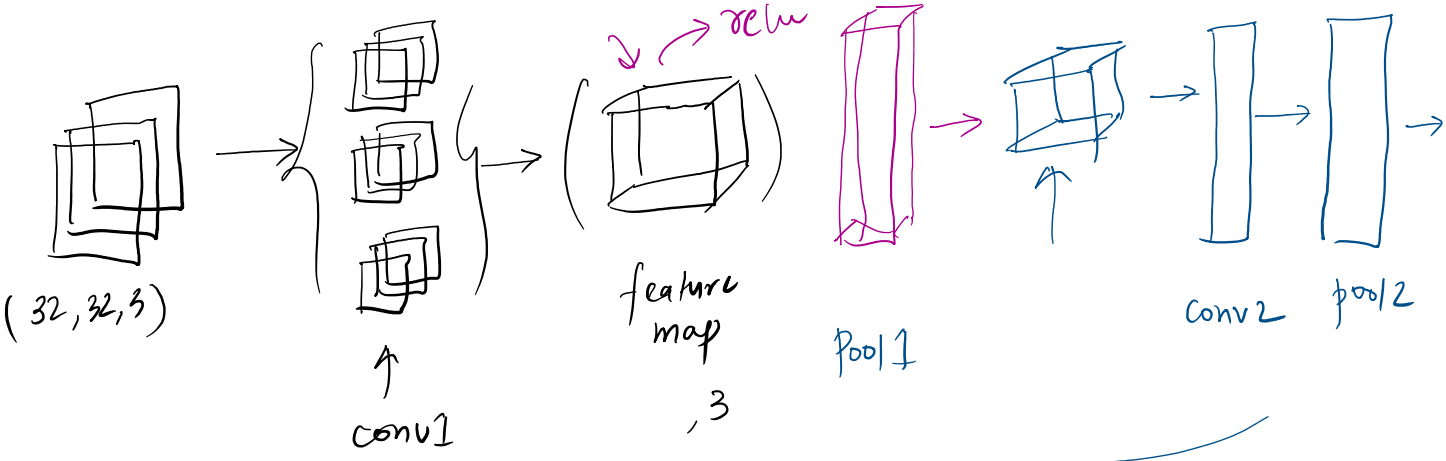
2x2 → 4

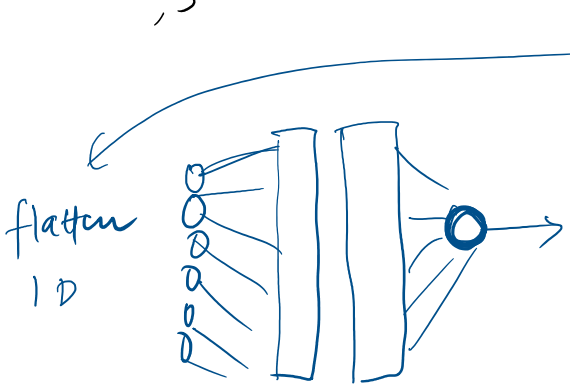


CNN Architecture

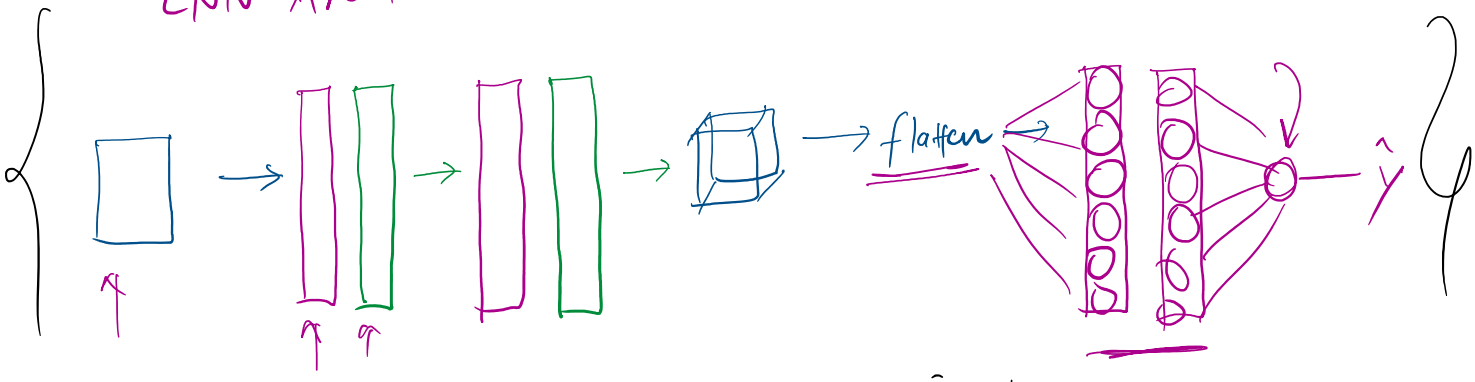
02 September 2022 10:55

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

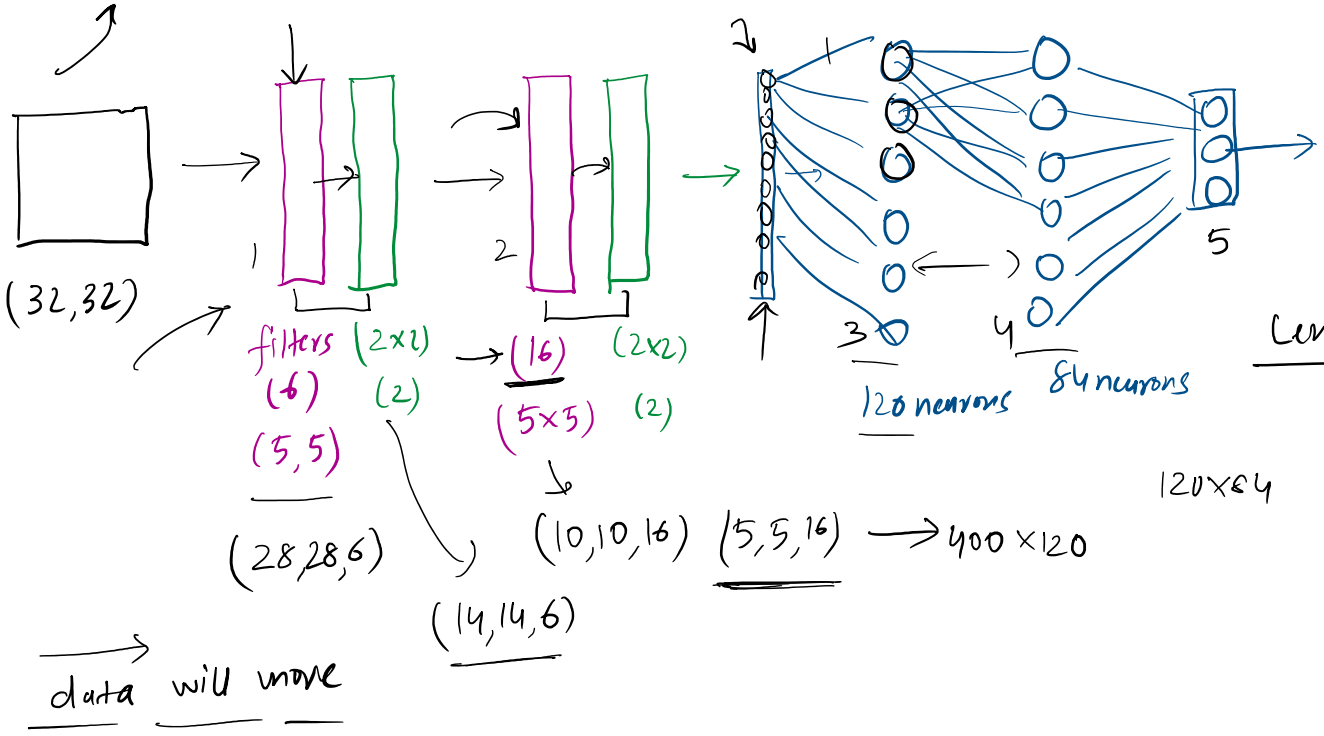
3) GoogLeNET

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

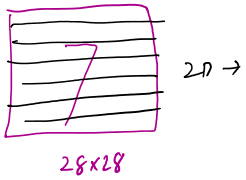
Keras Code

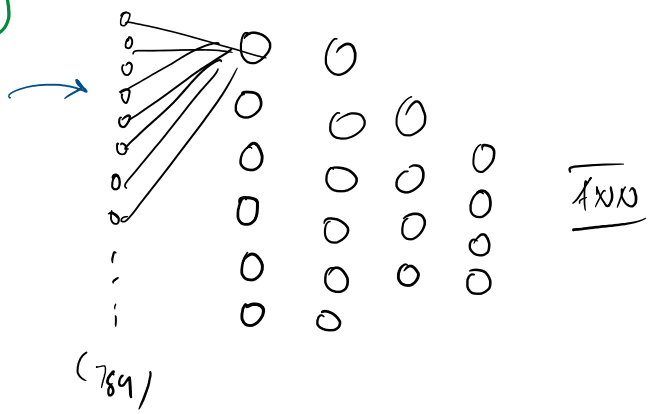
02 September 2022 14:58

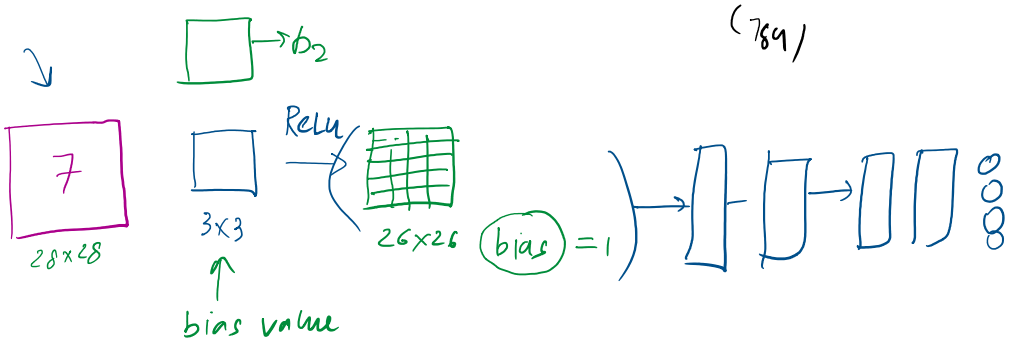
CNN Vs ANN

06 September 2022 10:00

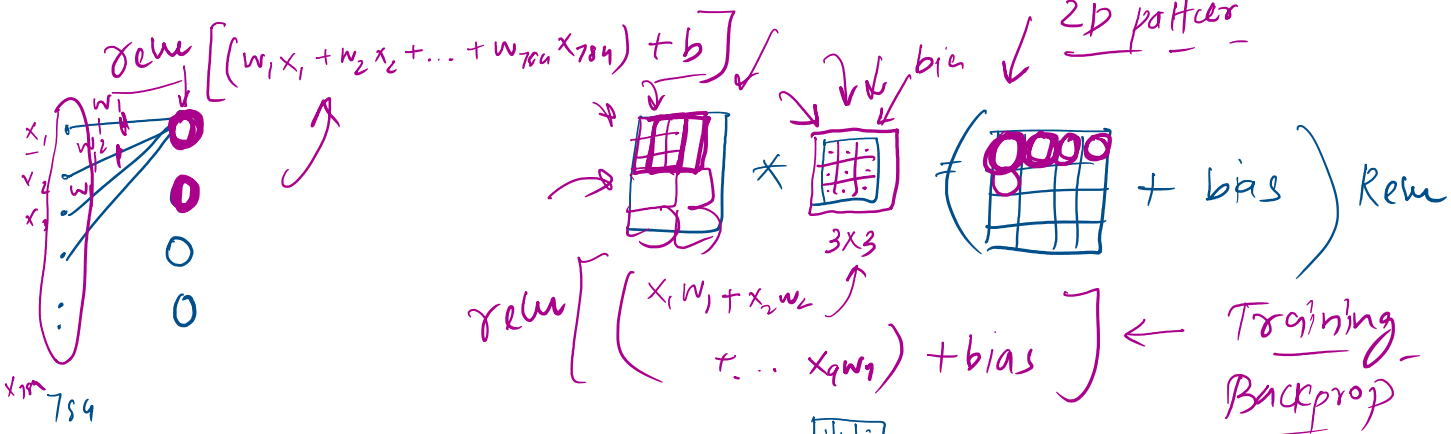
1) Computation Cost → ↓↓
2) Overfitting →
3) loss of imp features like spatial arrangement of pixels

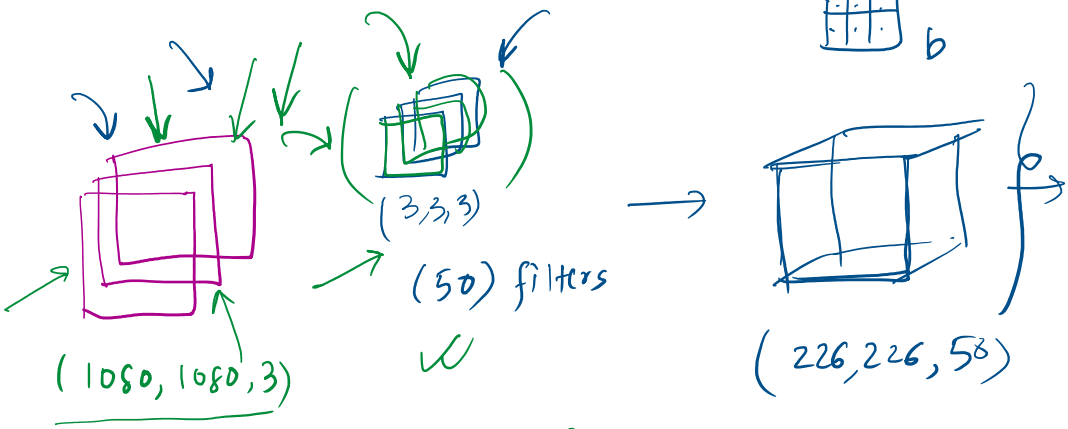






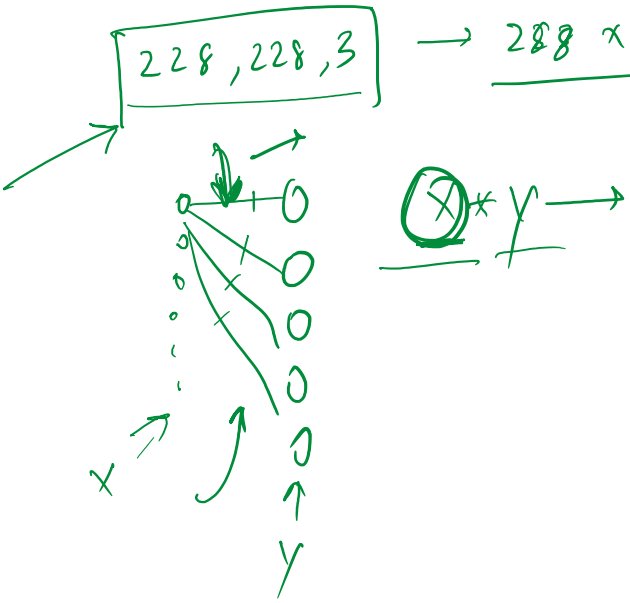
similarity ANN CNN





learnable parameters 1400

1350 wts
+50
1400



288 x228x3

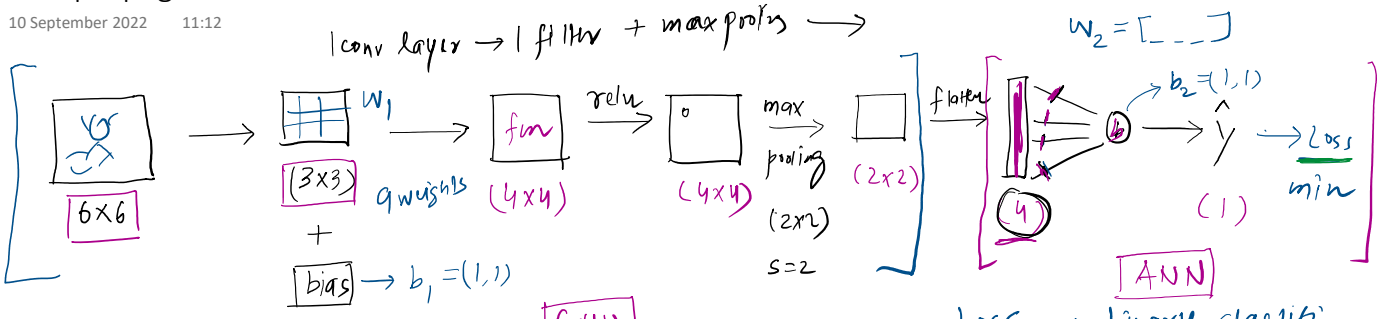
1080 X 1080 x3

overfitting AND

Backprop in CNN

Backpropagation in CNN

10 September 2022 11:12



Trainable Parameters

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

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

Logical Flow

Loss → binary classif

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

$A_2 = \hat{y}$

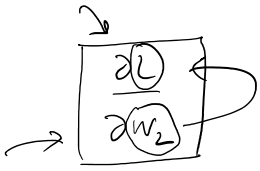
Forward Prop

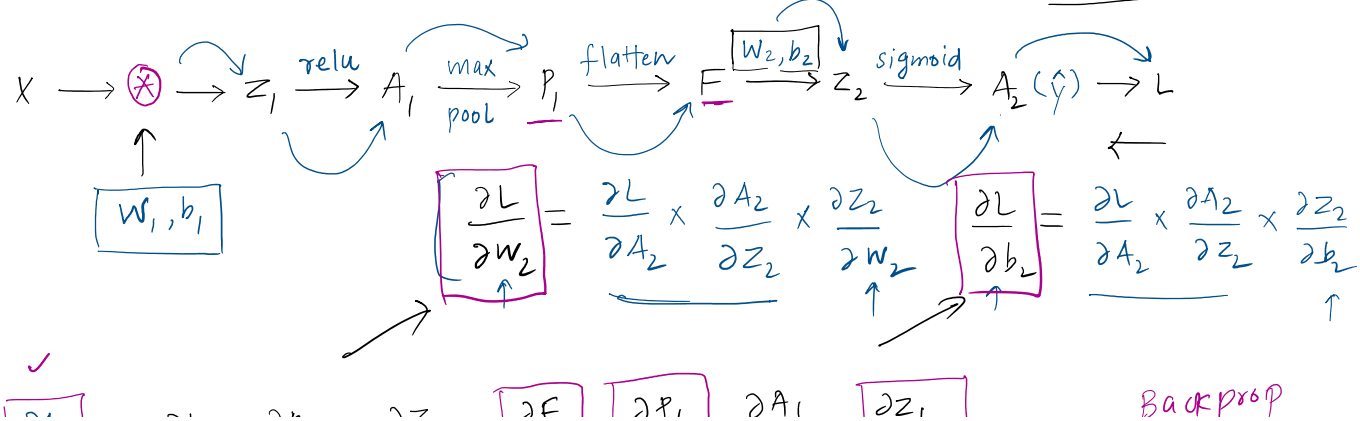
$$Z_1 = Conv(X, W_1) + b_1$$
$$A_1 = relu(Z_1)$$
$$P_1 = maxpool(A_1)$$
$$F = flatten(P_1)$$
$$Z_2 = W_2 F + b_2$$
$$A_2 = \sigma(Z_2)$$

Gradient Descent

$$W_1 = W_1 - \eta \frac{\partial L}{\partial W_1} \quad \checkmark$$
$$W_2 = W_2 - \eta \frac{\partial L}{\partial W_2} \quad \checkmark$$
$$b_1 = b_1 - \eta \frac{\partial L}{\partial b_1} \quad \checkmark$$
$$b_2 = b_2 - \eta \frac{\partial L}{\partial b_2} \quad \checkmark$$

Loss is minimized

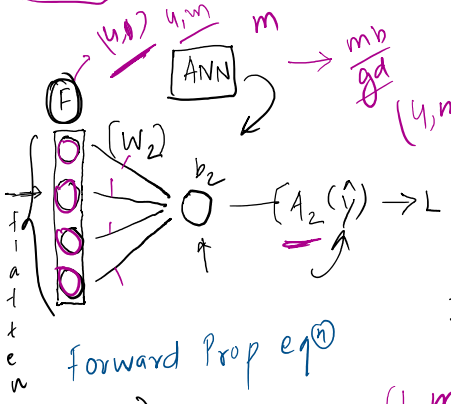




$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial A_2} \times \frac{\partial A_2}{\partial Z_2} \times \frac{\partial Z_2}{\partial F} \times \frac{\partial F}{\partial P_1} \times \frac{\partial P_1}{\partial A_1} \times \frac{\partial A_1}{\partial Z_1} \times \frac{\partial Z_1}{\partial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

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

Backprop
- → Convolution
- → Flatten
- → Max pooling



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

Forward Prop eqn

$$Z_2 = W_2 F + b_2$$
$$A_2 = \sigma(Z_2)$$

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

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

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

$$\frac{\partial A_2}{\partial Z_2} = \sigma(Z_2)[1-\sigma(Z_2)] = a_2[1-a_2]$$

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

W_2 update

Shape =

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

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

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

$\frac{\partial L}{\partial w_2} = (A_2 - Y) F'$ | $\frac{\partial L}{\partial b_2} =$

$(1,m) - (1,m)$

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

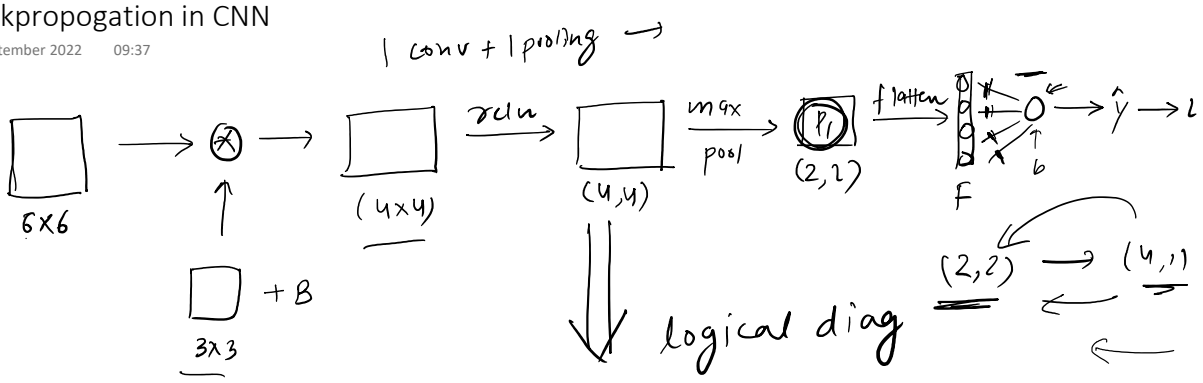
1 image

batch of image

$W_2 \rightarrow (1,4)$

Backpropogation in CNN

15 September 2022 09:37



Forward Prop

$Z_1 = conv(X, W_1) + b_1$

$A_1 = relu(Z_1)$

$P_1 = maxpool(A_1)$

$F = flatten(P_1)$

flatten

$Z_2 = W_2 F + b_2$

$A_2 = \sigma(Z_2)$

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

$(A_2 - Y)$

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

5

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

1 2 3 4 6

reshape(P_1.shape)

6 derivatives

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

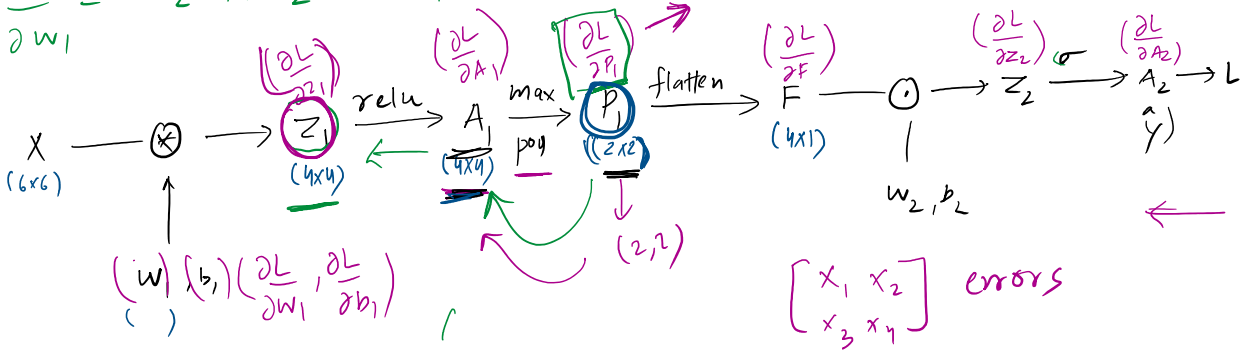
Shape ? → (F)

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

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

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

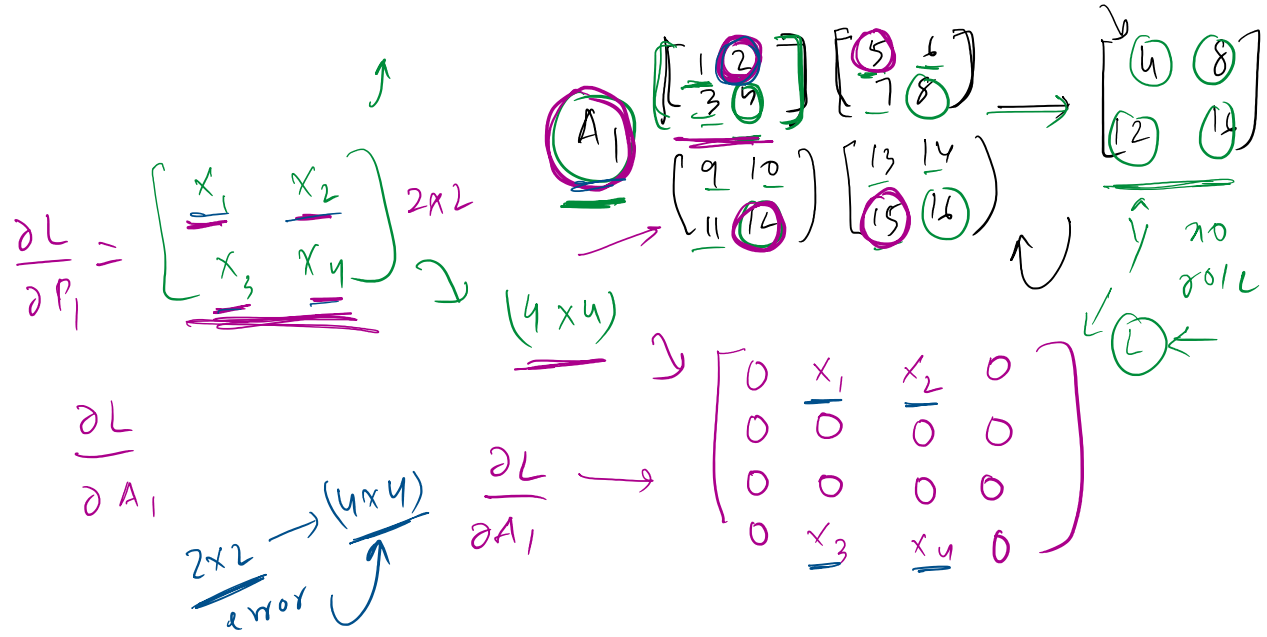
$\begin{bmatrix} x_1 & x_2 \\ x_3 & x_4 \end{bmatrix}$

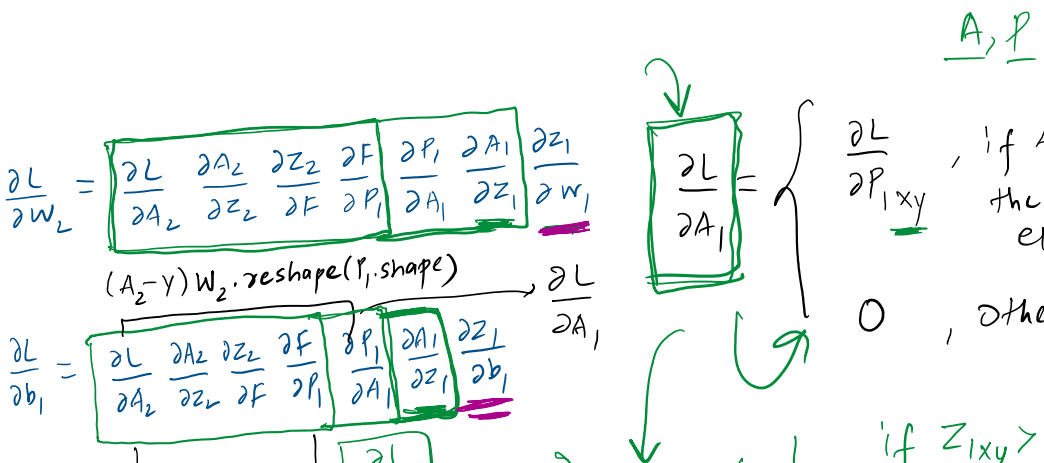


$\frac{\partial L}{\partial A_1} = (4,4)$

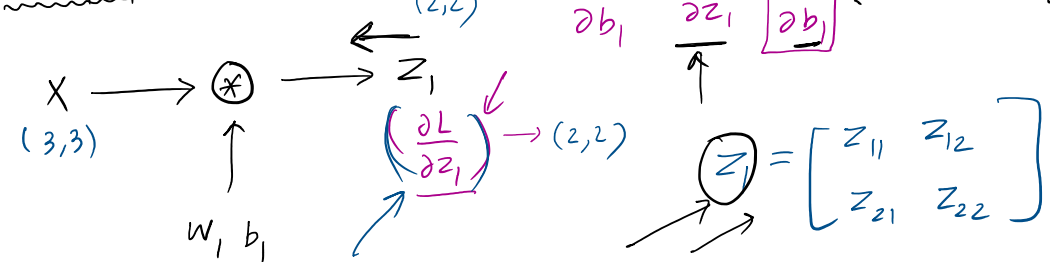
$\frac{\partial L}{\partial A_1} = (4,4)$

Flatten → no trainable parameters

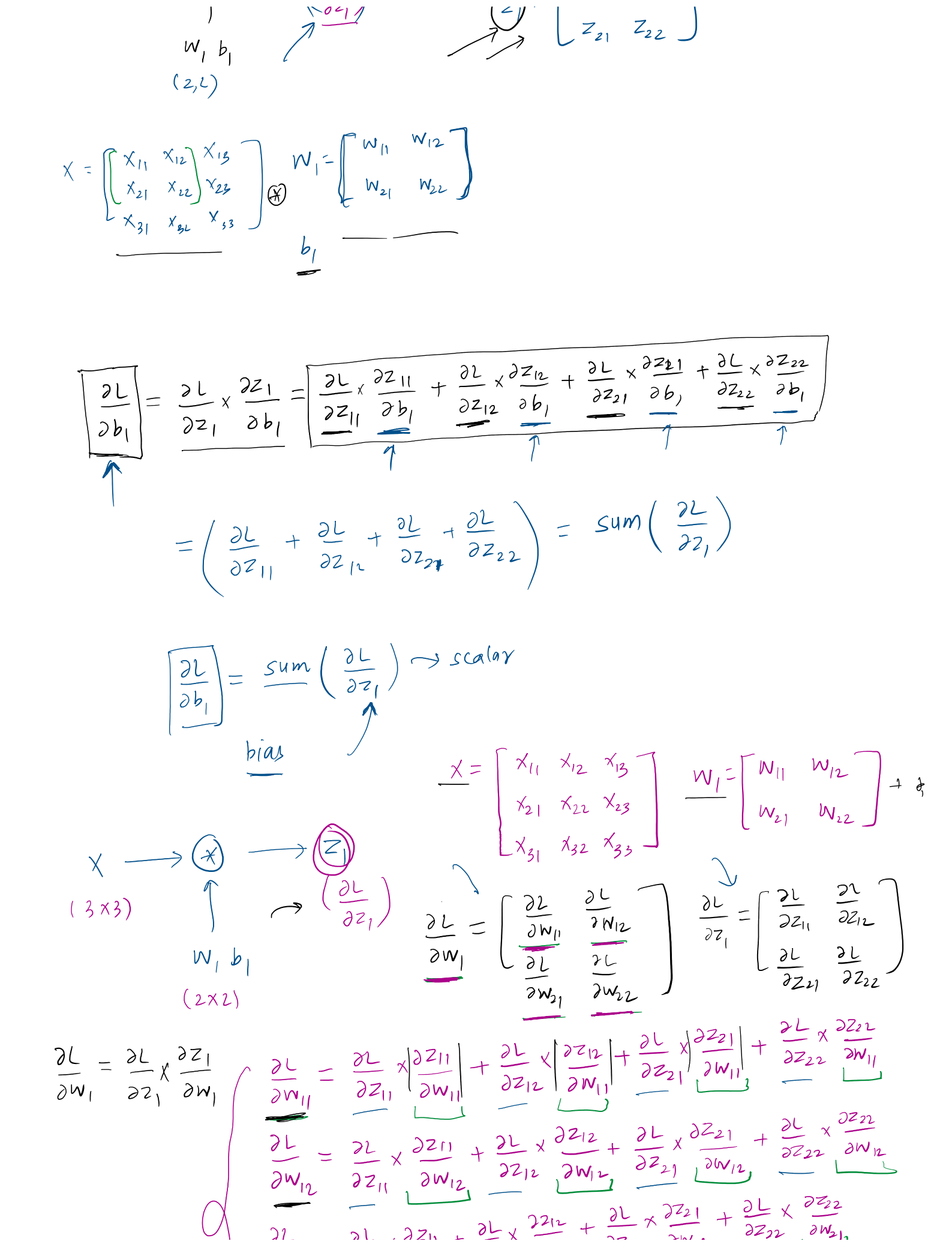




Backprop on Convolution



Day 43 Backpropogation in CNN Part 2 Page 200


W_1 b_1
(2,2)
Z_{21} Z_{22}
$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}$ $W_1 = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$
b_1
$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial z_1} \times \frac{\partial z_1}{\partial b_1} = \frac{\partial L}{\partial z_{11}} \times \frac{\partial z_{11}}{\partial b_1} + \frac{\partial L}{\partial z_{12}} \times \frac{\partial z_{12}}{\partial b_1} + \frac{\partial L}{\partial z_{21}} \times \frac{\partial z_{21}}{\partial b_1} + \frac{\partial L}{\partial z_{22}} \times \frac{\partial z_{22}}{\partial b_1}$
$= \left(\frac{\partial L}{\partial z_{11}} + \frac{\partial L}{\partial z_{12}} + \frac{\partial L}{\partial z_{21}} + \frac{\partial L}{\partial z_{22}} \right) = sum\left(\frac{\partial L}{\partial z_1} \right)$
$\frac{\partial L}{\partial b_1} = sum\left(\frac{\partial L}{\partial z_1} \right) \rightarrow$ scalar
bias
$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}$ $W_1 = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$ + b
X → (*) → Z_1
(3x3)
W_1 b_1
(2x2)
$\left(\frac{\partial L}{\partial z_1} \right)$
$\frac{\partial L}{\partial W_1} = \begin{bmatrix} \frac{\partial L}{\partial w_{11}} & \frac{\partial L}{\partial w_{12}} \\ \frac{\partial L}{\partial w_{21}} & \frac{\partial L}{\partial w_{22}} \end{bmatrix}$ $\frac{\partial L}{\partial z_1} = \begin{bmatrix} \frac{\partial L}{\partial z_{11}} & \frac{\partial L}{\partial z_{12}} \\ \frac{\partial L}{\partial z_{21}} & \frac{\partial L}{\partial z_{22}} \end{bmatrix}$
$\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial z_1} X \frac{\partial z_1}{\partial W_1}$
$\frac{\partial L}{\partial w_{11}} = \frac{\partial L}{\partial z_{11}} \times \frac{\partial z_{11}}{\partial w_{11}} + \frac{\partial L}{\partial z_{12}} \times \frac{\partial z_{12}}{\partial w_{11}} + \frac{\partial L}{\partial z_{21}} \times \frac{\partial z_{21}}{\partial w_{11}} + \frac{\partial L}{\partial z_{22}} \times \frac{\partial z_{22}}{\partial w_{11}}$
$\frac{\partial L}{\partial w_{12}} = \frac{\partial L}{\partial z_{11}} \times \frac{\partial z_{11}}{\partial w_{12}} + \frac{\partial L}{\partial z_{12}} \times \frac{\partial z_{12}}{\partial w_{12}} + \frac{\partial L}{\partial z_{21}} \times \frac{\partial z_{21}}{\partial w_{12}} + \frac{\partial L}{\partial z_{22}} \times \frac{\partial z_{22}}{\partial w_{12}}$
$\frac{\partial L}{\partial w_{21}} = \frac{\partial L}{\partial z_{11}} \times \frac{\partial z_{11}}{\partial w_{21}} + \frac{\partial L}{\partial z_{12}} \times \frac{\partial z_{12}}{\partial w_{21}} + \frac{\partial L}{\partial z_{21}} \times \frac{\partial z_{21}}{\partial w_{21}} + \frac{\partial L}{\partial z_{22}} \times \frac{\partial z_{22}}{\partial w_{21}}$

$$\frac{\partial L}{\partial w_{21}} = \frac{\partial L}{\partial z_{11}} \times \frac{\partial z_{11}}{\partial w_{21}} + \frac{\partial L}{\partial z_{12}} \times \frac{\partial z_{12}}{\partial w_{21}} + \frac{\partial L}{\partial z_{21}} \times \frac{\partial z_{21}}{\partial w_{21}} + \frac{\partial L}{\partial z_{22}} \times \frac{\partial z_{22}}{\partial w_{21}}$$

$$\frac{\partial L}{\partial w_{22}} = \frac{\partial L}{\partial z_{11}} \times \frac{\partial z_{11}}{\partial w_{22}} + \frac{\partial L}{\partial z_{12}} \times \frac{\partial z_{12}}{\partial w_{24}} + \frac{\partial L}{\partial z_{21}} \times \frac{\partial z_{21}}{\partial w_{22}} + \frac{\partial L}{\partial z_{32}} \times \frac{\partial z_{22}}{\partial w_{22}}$$

$$\frac{\partial L}{\partial w_{11}} = \frac{\partial L}{\partial z_{11}} x_{11} + \frac{\partial L}{\partial z_{12}} x_{12} + \frac{\partial L}{\partial z_{21}} x_{21} + \frac{\partial L}{\partial z_{22}} x_{22}$$

$$\frac{\partial L}{\partial w_{12}} = \frac{\partial L}{\partial z_{11}} x_{12} + \frac{\partial L}{\partial z_{12}} x_{13} + \frac{\partial L}{\partial z_{21}} x_{22} + \frac{\partial L}{\partial z_{22}} x_{23}$$

$$\frac{\partial L}{\partial w_{21}} = \frac{\partial L}{\partial z_{11}} x_{21} + \frac{\partial L}{\partial z_{12}} x_{22} + \frac{\partial L}{\partial z_{21}} x_{31} + \frac{\partial L}{\partial z_{22}} x_{32}$$

$$\frac{\partial L}{\partial w_{22}} = \frac{\partial L}{\partial z_{11}} x_{22} + \frac{\partial L}{\partial z_{12}} x_{23} + \frac{\partial L}{\partial z_{21}} x_{32} + \frac{\partial L}{\partial z_{22}} x_{33}$$

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

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

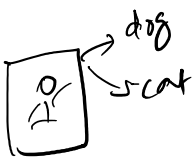
$$\frac{\partial L}{\partial w_1} = \text{conv}\left(X, \frac{\partial L}{\partial z_1}\right)$$

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

Why use Pretrained models?

03 October 2022 12:52

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



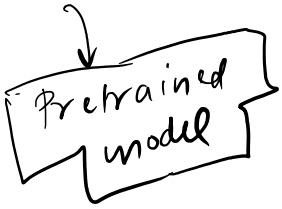
labelling

10000 photos

manually

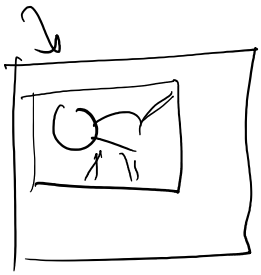
pinterest

2) Time → model → training



ImageNET Dataset

03 October 2022 12:36



Visual Database of images (Why What and How)

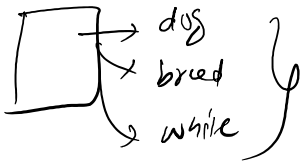
Why

2006 → Fei fei Li → model and algorithms

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

1 million images → to label → bounding box

↳ object local



How → crowd sourcing → Amazon mechanical Turk

↳ Dataset → Deep learning

ImageNET challenge