Agenda

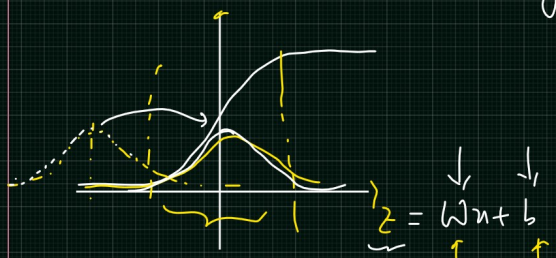
1) Batch Normalization.
2) Transfer learning.

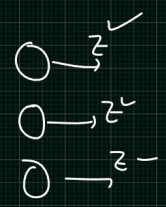
Std NN can suffer vanishing & exploding gradient

due to,

ICS: Internal Covariate shift

↳ Change in distribution of network activation o/p





due to change in n/w parameter (W, b) during training

Act^{n} f^{n}., Wt. init,

⟶ It has been experimentally proven:-

{ Le Cun et. al. 1998 & Wiesler & Ney 2011 }

The network converges faster if inputs to the layer are whitened / normalized

linearly transformed to zero mean & unit variance } Std. Normal distribution

→ Expectation :

Fix the distribution for each layer {Dynamically}

⇓

Reduced ICS

⇓

Faster convergence

⇓

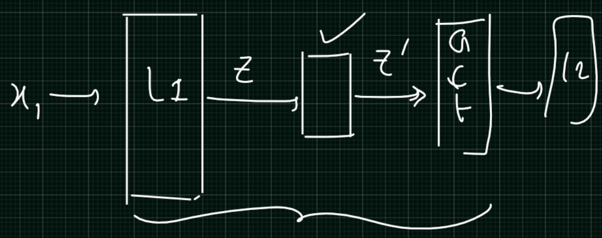
Faster training

⇓

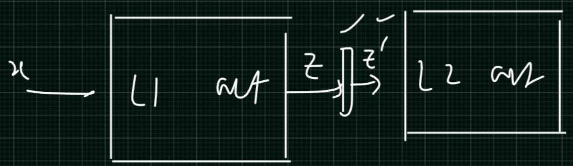
Less use of Resources

⇓

Save $ ₹



or

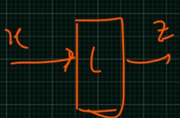


Batch Normalization {BN}

└ In 2015, Sergey Ioffe & Christian Szegedy

└ proposed an extra set of operation to perform before or after activation layer.

Algorithm : - {While Training}



1) $\mu_B = \frac{1}{m_B} \sum_{i=1}^{m_B} x^{(i)}$ ← batch mean

$x_{i/p}, z_i$

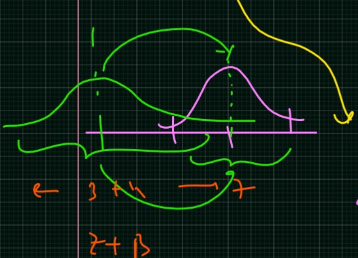
m_B → batch size

$x^{(i)}$ → data

2) $\sigma_B^2 = \frac{1}{m_B} \sum_{i=1}^{m_B} (x^{(i)} - \mu_B)^2$ ← Batch Variance

3) $\hat{x}^{(i)} = \frac{x^{(i)} - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$ ← Normalizing $\hat{x}^{(i)}$

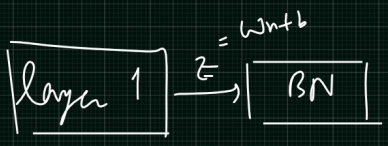
└→ smoothing term $\epsilon = 10^{-7}$ } to avoid zero division error



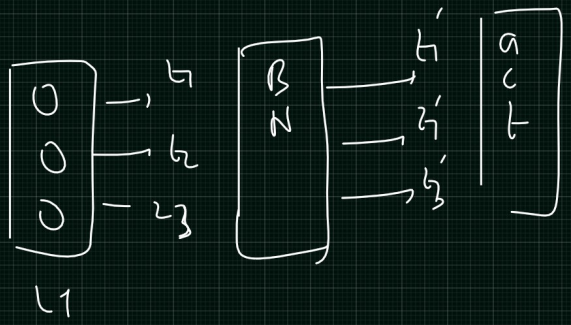
4) $z^{(i)} = \gamma \otimes \hat{x}^{(i)} + \beta$

└ Shifting } Learnable parameter

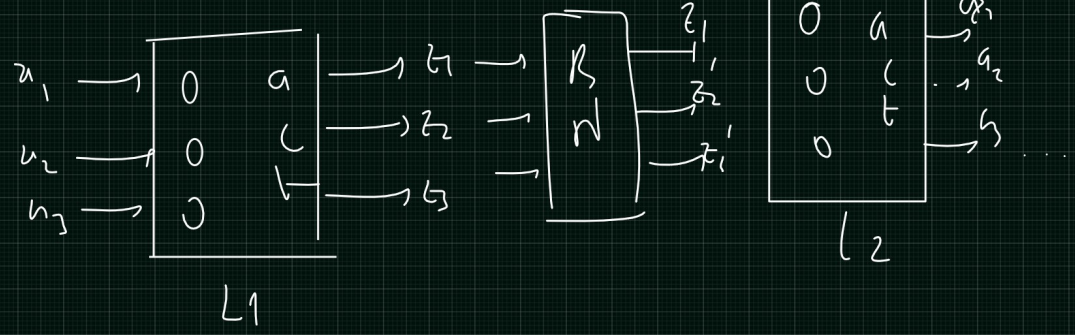
Scaling



I.



II



NN

(W, b)

$\frac{\partial e}{\partial W}, \frac{\partial e}{\partial b}$

$W \leftarrow W - \eta \frac{\partial e}{\partial W}$

BN

(γ, β)

$\frac{\partial e}{\partial \gamma}, \frac{\partial e}{\partial \beta}$

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

After training

μ is calculated from all μ_B

σ^2 is calculated from all σ_B^2

} Calculated once

While prediction

$$\begin{cases} (i) & \hat{x}^{(i)} = \dfrac{x^{(i)} - \mu}{\sqrt{\sigma^2 + \epsilon}} \\ (ii) & z^{(i)} = \gamma \otimes \hat{x}^{(i)} + \beta \end{cases}$$

More Observation :-

→ γ & β → 2 new trainable parameters

→ μ & σ^2 → 2 additional parameters (non trainable)
They are calculated using moving average internally.

extra parameters = 4 → $\gamma, \beta, \mu, \sigma$
trainable = 2 γ, β
non trainable 2 μ, σ

→ Now Cost function is :-

$C(W, b, \gamma, \beta)$

⇓
updated by backpropagation

→ Advantages :-

i) You don't need scaling of data if you are using BN as a first layer.

ii) Although it adds 2 extra parameters to train still it converges to the solution faster

Without BN	With BN
90%	90%
100 steps	70 steps

iii) It solves unstable gradient problem to some extent

iv) It doesn't depends on choice of activation f^n and Wt. initialization. get affected by

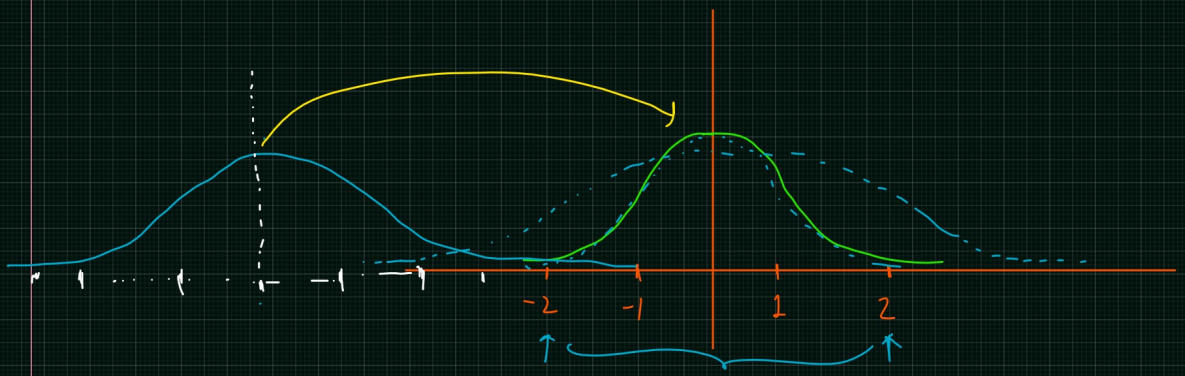
→ Disadvantage :-

i) It increases the complexity of n/w.

ii) No. of trainable parameter increased

iii) Complexity => runtime penalty => slow prediction.

iv) Training time increased but convergence is fast



Transfer Learning :-

↓ DL

(i) [12th Std.] →

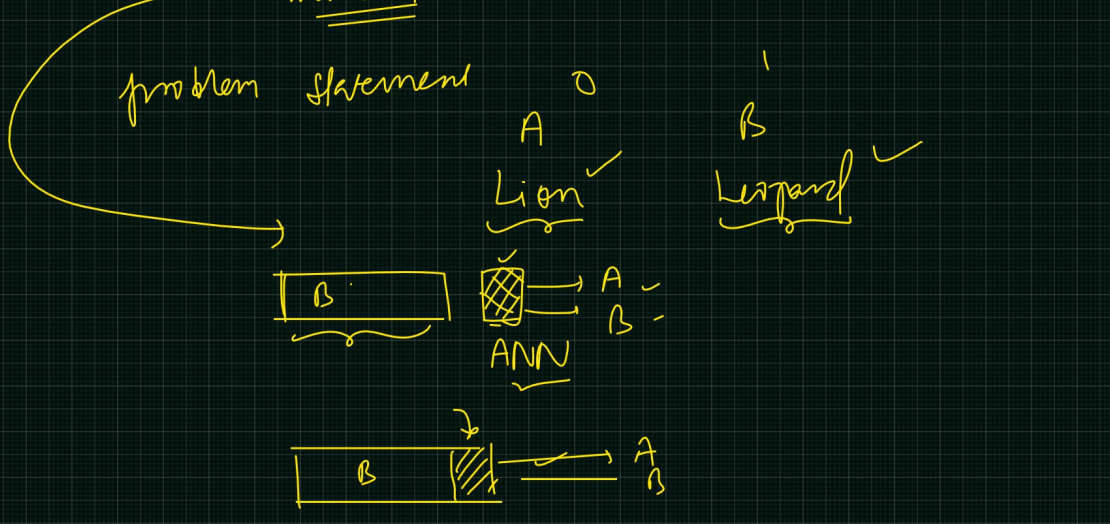
(ii) [Expert People] →

pre trained model

model A → Sea animals

model B → Land animals → 1000

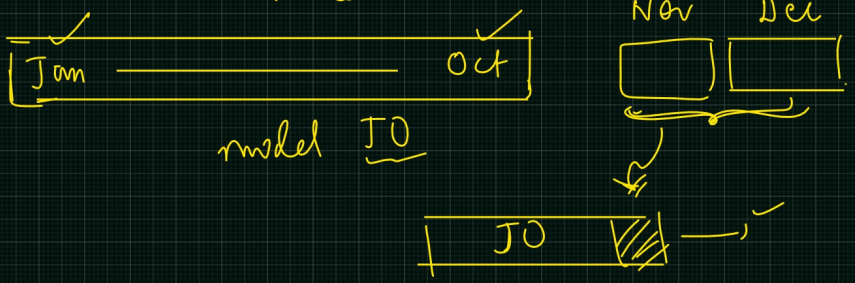
problem statement



Regression

morning data

model



model JO

19

19th → 75% → 85%

18 19th →

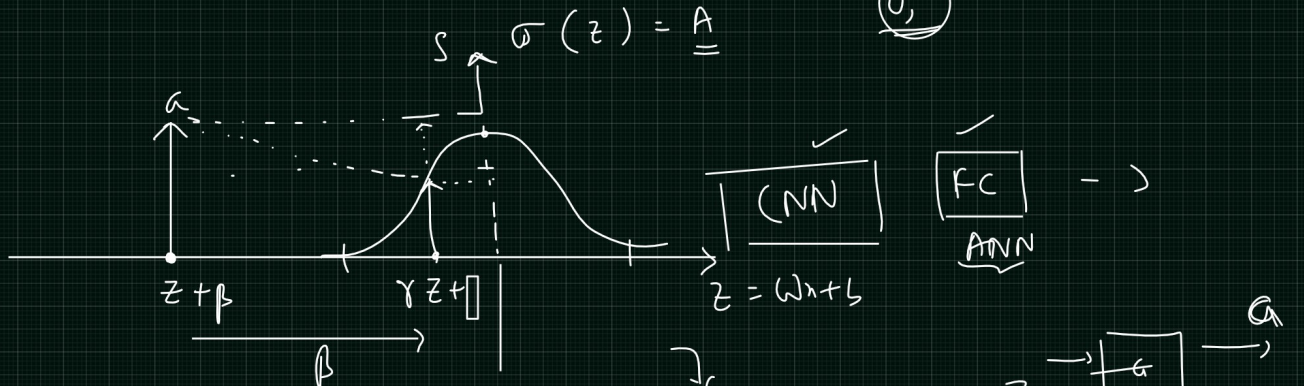
19th → 90%

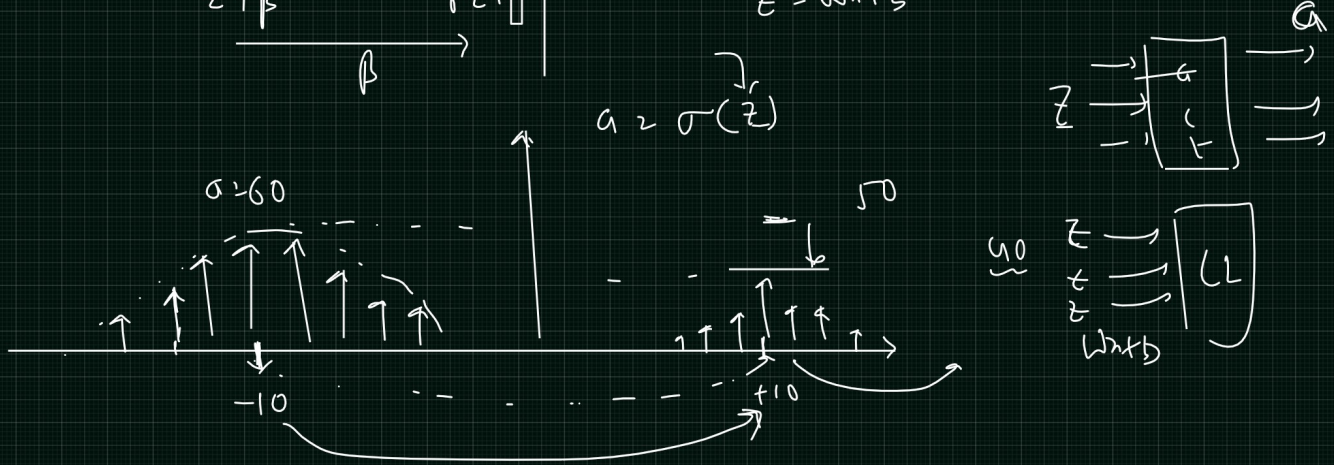
18th 19th → 9

NLP

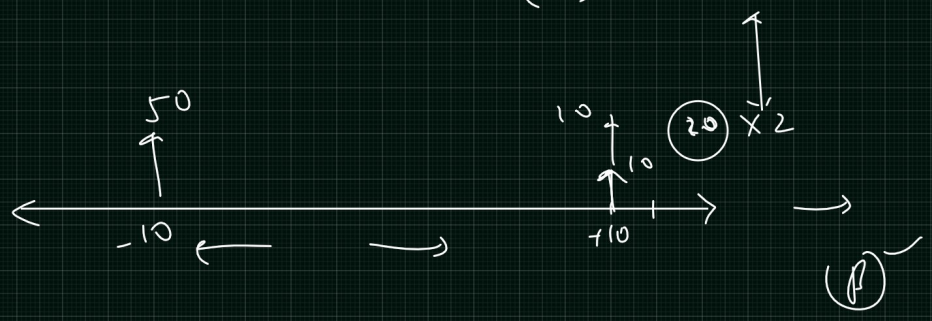
CV

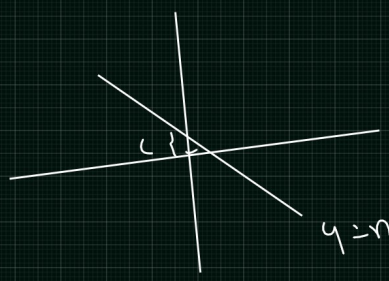
RNN

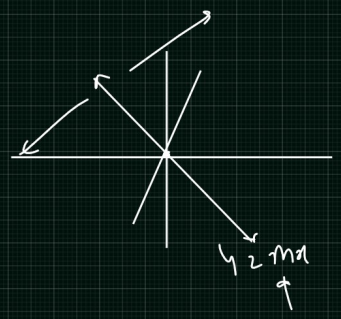


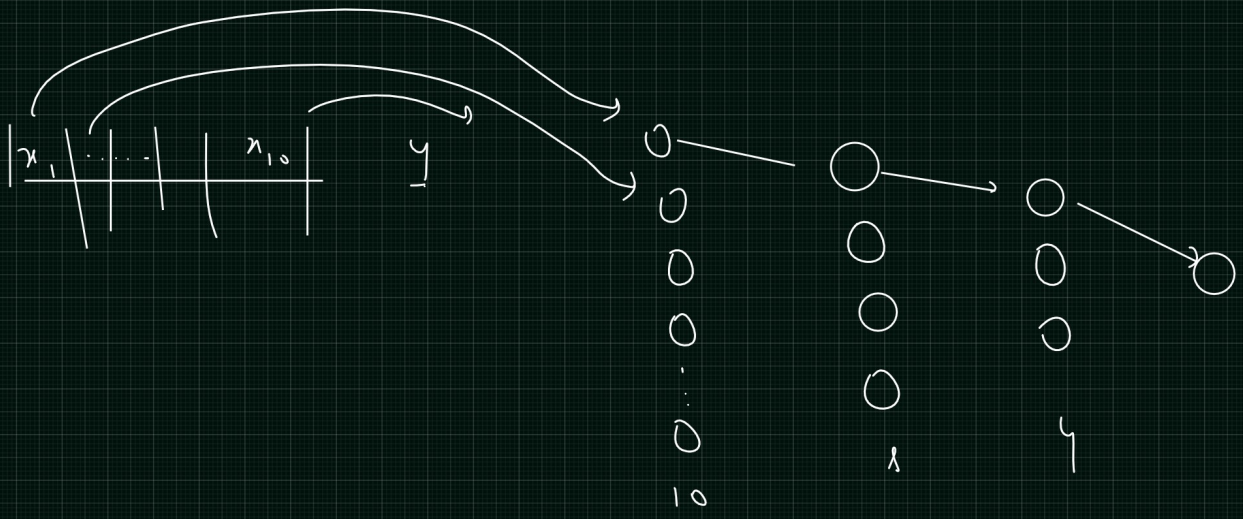


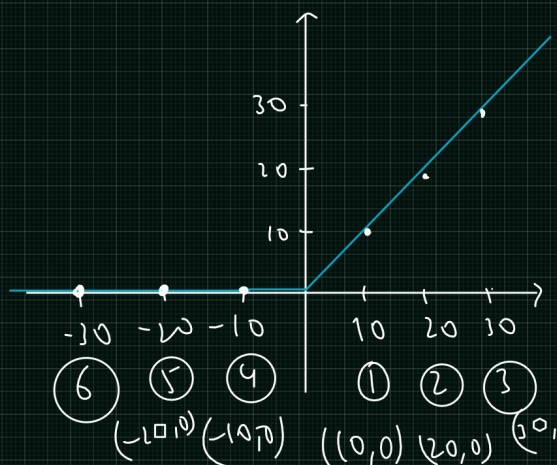
$\sigma\left(\frac{1}{z}\right) = a$











ReLU = $\max(z, 0)$

$= \begin{cases} z & \geq 0 \\ 0 & < 0 \end{cases}$

ReLU