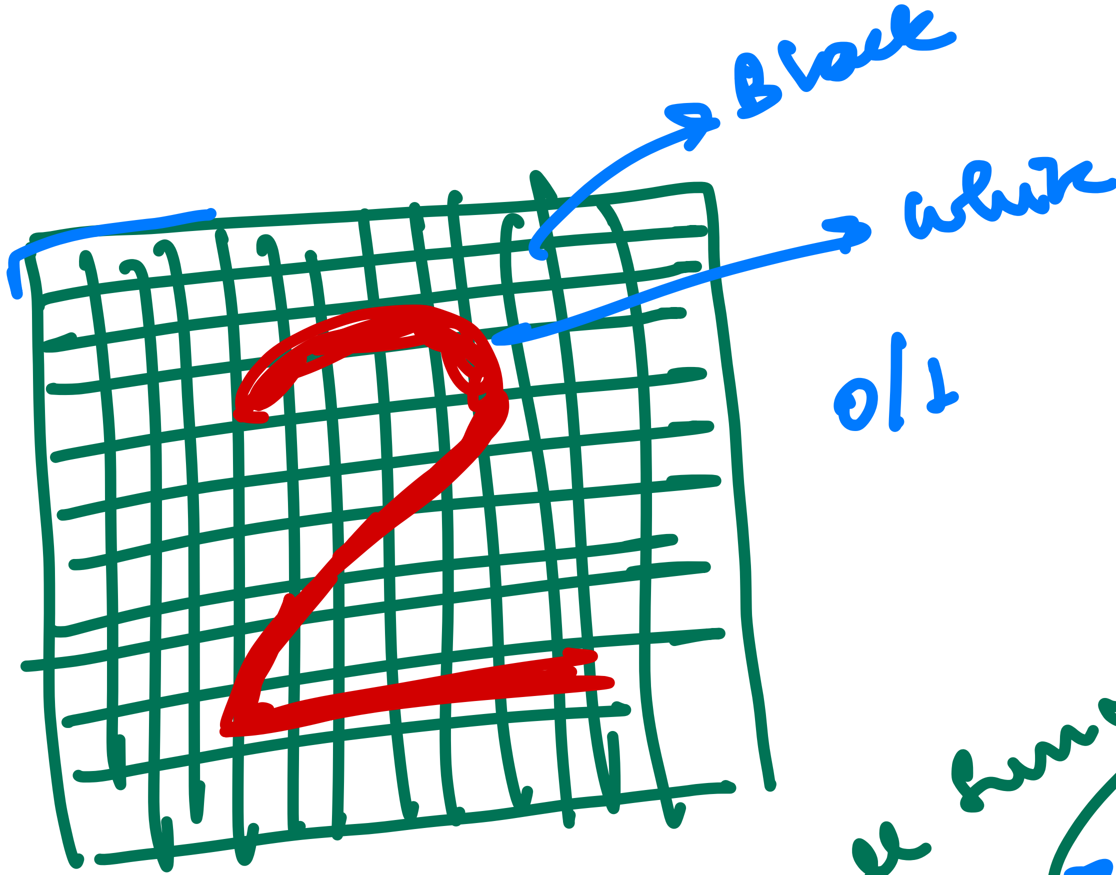
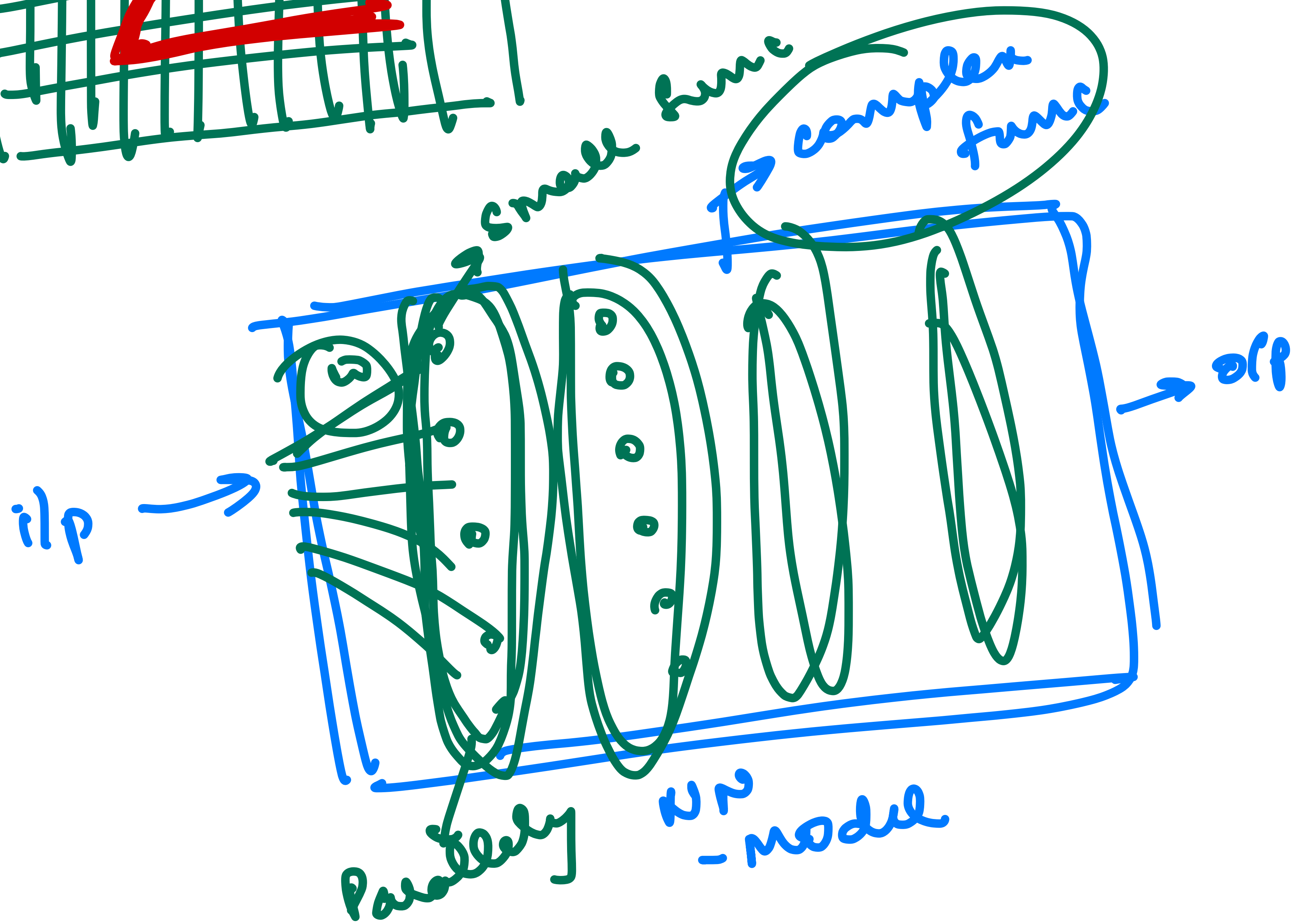
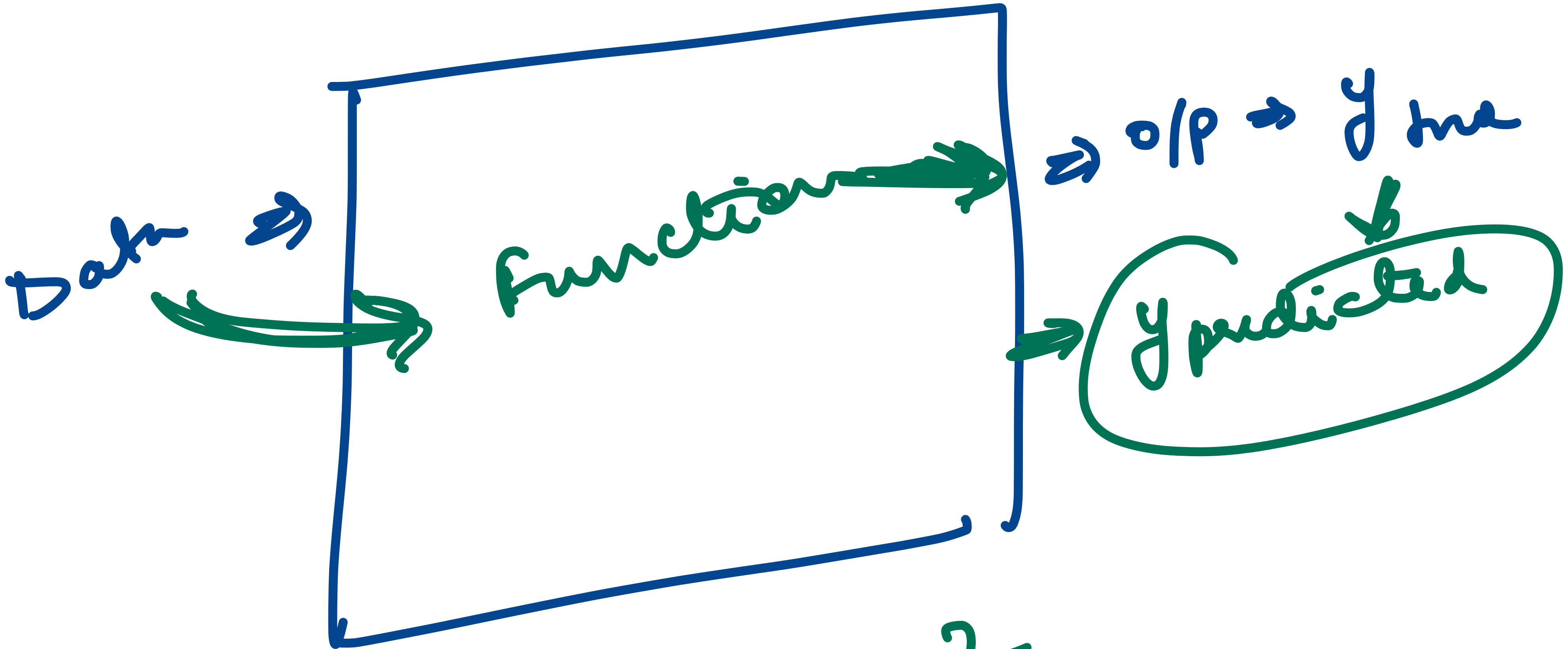

Black
white
0/1


small func
complex func
o/p
i/p
Parallely
NN
-model



Cost

$$\text{Cost} = \underbrace{\left(y_{true} - y_{predicted}\right)^2}_{\text{Minimize}}$$

Error

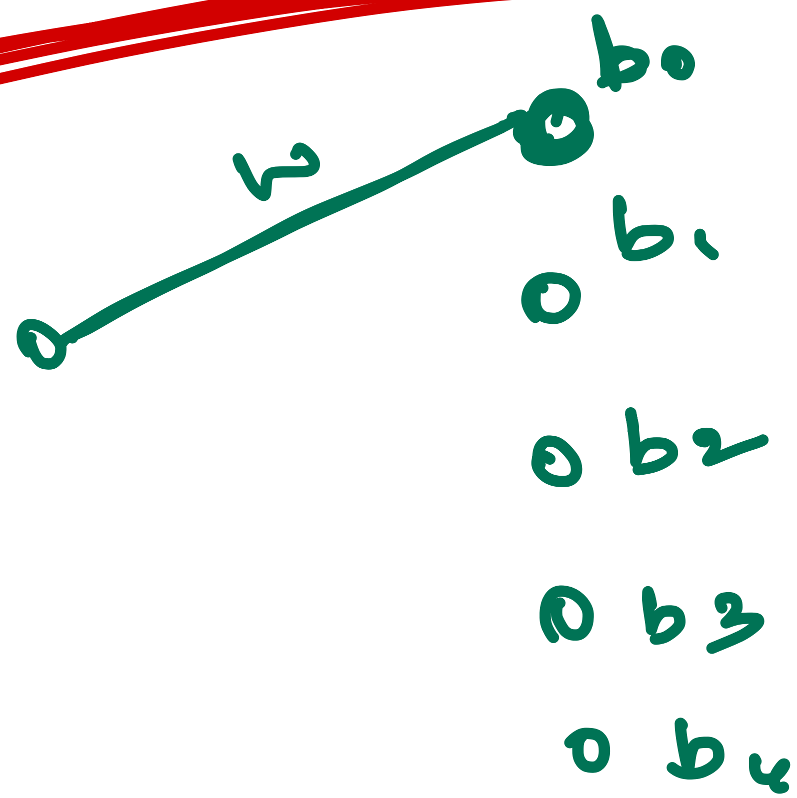
$w \rightarrow$ weight
$b \rightarrow$ bias

} Parameters

Parameters → weight w → for every connection

Bias b → func should not always pass through origin

learnt from data



GPT3 → 175 B

GPT4 → 1 Trillion

Llama → > 100B

NN → computational model inspired by human brain

→ layers of interconnected nodes

i/p layer → hidden layers → o/p layer

perform computations and extract patterns

$$w = w - \alpha \frac{\partial c}{\partial w} \quad , \quad b = b - \alpha \frac{\partial c}{\partial b}$$

$\downarrow$

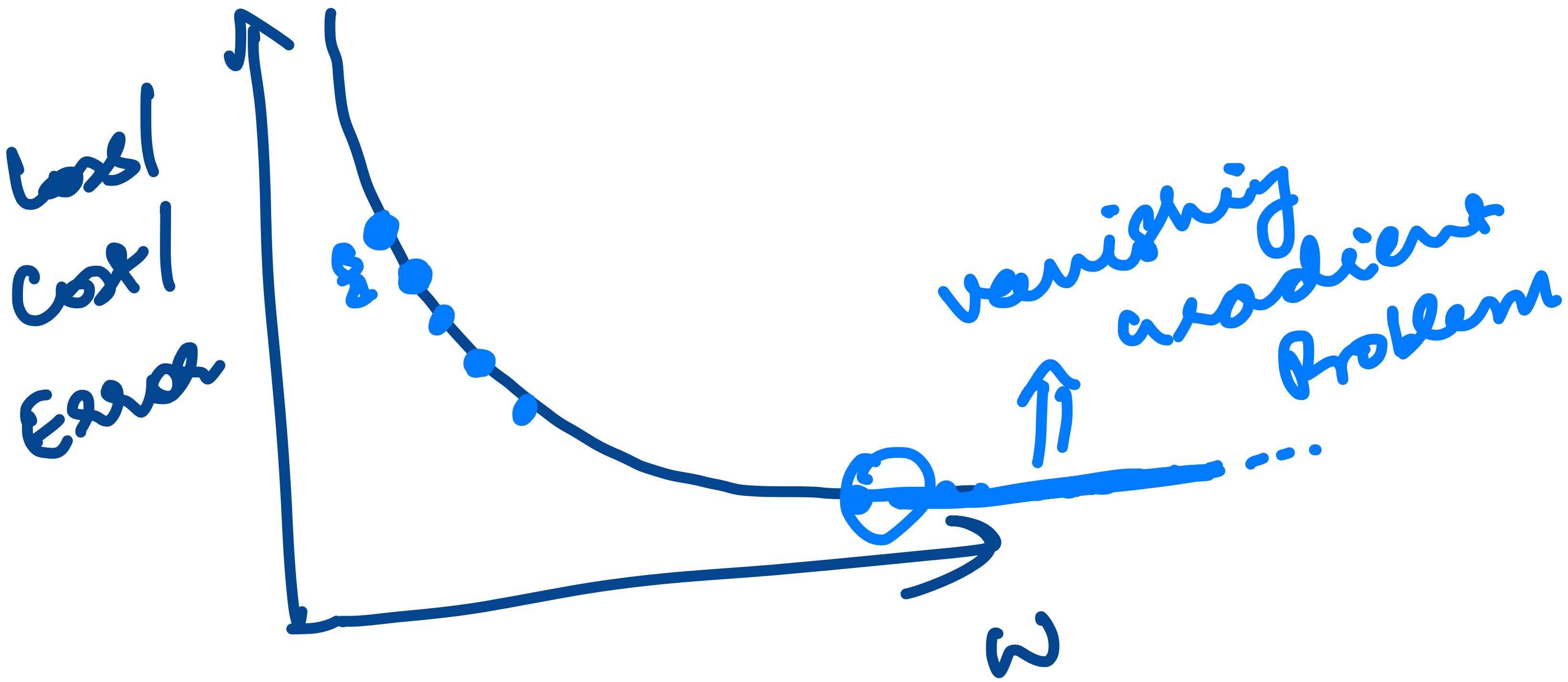
0.1

0.0001

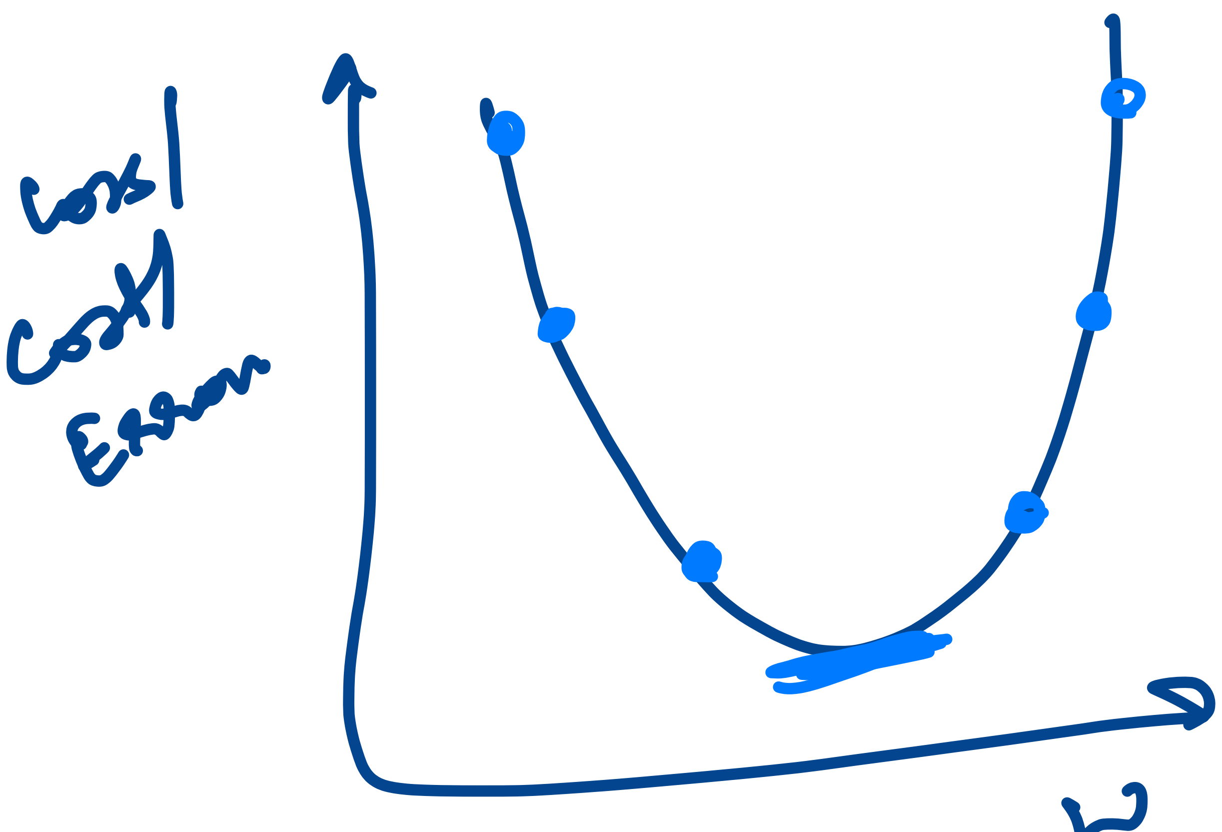
0.00001

vanishing Gradient Problem

Exploding Gradient Problem



→ overfitting



Aim → Generalisation

Goal → Model should perform well on (seen) training data & testing data (unseen)

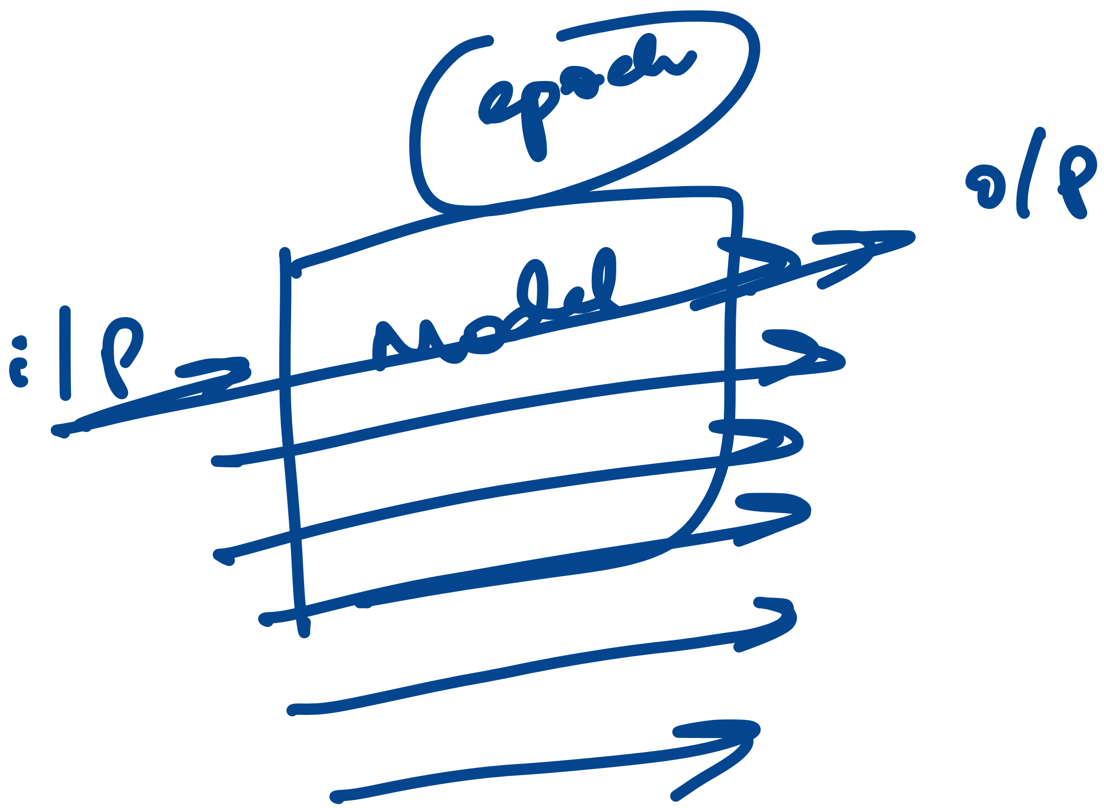
	Overfitting	Underfitting
		Summary
DSA Interview	Remembers the questions without understanding	
Face Recognition	Hair, hair length, Glasses, Background, color, clothes	oval, 2 eyes, nose
Spam Detection	Exact Mails	"win" "free"
	STOP Sign 	Anything that is Red
Self Driving car		

⇒ Training data

Loss ⇒ minimum

⇒ Testing

Loss ⇒ More



Training Data ×

Loss ↑

Testing Data ×

Loss ↑

Hyper Parameters of Model

No. of epochs ?
No. of hidden layers ?
weights Initialisation ?
Bias Initialisation ?

$$W = W - \alpha \frac{\partial C}{\partial W}$$

learning Rate ?

0.01, 0.001
0.00001 → learning slow

No. of neurons per layers

→ Manually set
→ Not learnt from data

Trial and Error

→ Settings of Model
→ Set before training

Gradients → weight, Bias

Algos → $w = w - \alpha \frac{\partial c}{\partial w}$

learning rate

Stochastic Gradient Descent

Mini Batch GD

Adaptive learning method

Adam

:

No. of doses v.s health

$\underbrace{\text{No. of doses}}_{x}$ v.s $\underbrace{\text{health}}_{y}$

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

$$z = wx + b$$

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

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

$$\frac{\partial c}{\partial w} = \frac{\partial c}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial w}$$

$$= -2(y - \hat{y}) \, \frac{\sigma(z)(1-\sigma(z))}{\sigma'(z)} \, x$$

$$\frac{\partial c}{\partial b} = \frac{\partial c}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial b}$$

$$= -2(y - \hat{y}) \, \sigma(z)(1-\sigma(z))$$

Loss Function /
Cost Function /
Error Function

→ y true, $\hat{y}$ predicted

Regression

MSE → Mean Squared Error

$(y-\hat{y})^2$ ✓

MAE → Mean Absolute Error

Huber loss

Classification

Binary Cross Entropy

Categorical Cross Entropy

Sparse Categorical Entropy

Adam

Training Steps

1. **Forward Pass**

   i/p to o/p

   calculate y predicted

$L = \text{loss}(y_{true}, y_{pred})$

$\Downarrow$

loss Function / Cost Function /
Error Function

2. **Gradient Computation / Backward Propogation**

$\frac{\partial L}{\partial w}$, $\frac{\partial L}{\partial b}$ use chain Rule

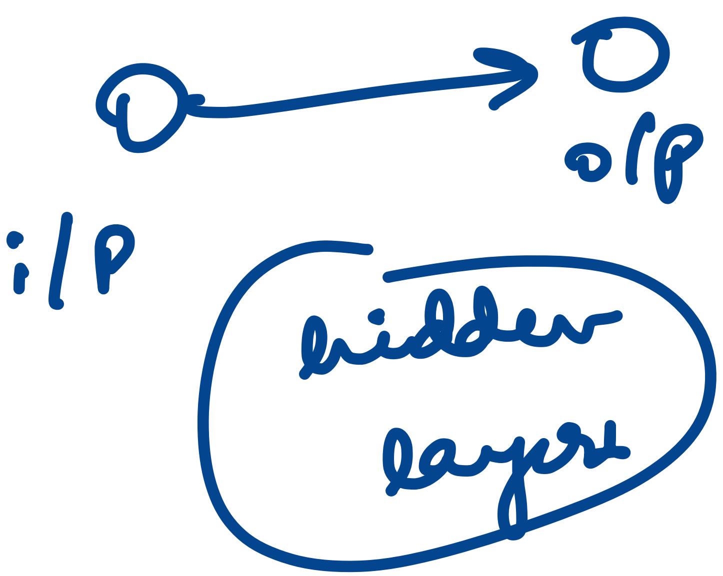
③ Optimization | update weights & biases

Optimization Algo

$$w = w - \alpha \frac{dC}{dw}$$

$$b = b - \alpha \frac{dC}{db}$$

Gradient Descent.



Activation

value that a neuron/node produces after applying activation function

→ Input layer ⇒ Activation = i/p values

→ Hidden layers ⇒ $\sigma(wx+b)$ ⇒ Activation

σ ↓ Activation functions

→ Output layer ⇒ final prediction/probability

For a particular neuron → i/ps are $x_1, x_2, x_3 \dots$

$w_1 \; w_2 \; w_3 \dots$

$$z = w_1x_1 + w_2x_2 + w_3x_3 \dots + \underline{b}$$

Activation funcs

Sigmoid $\frac{1}{1+e^{-z}}$

Relu $\max(0, z)$

Tanh $= \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$

Softmax $= \frac{e^{z_i}}{\sum_j e^{z_j}}$

1. Loss function / cost / Error
2. Optimization algo
3. Activation function

28 pixels × 28 pixels

784 pixels

0 0

1 0

2 0

⋮

781 0

782 0

783 0

I/p Layer

0 0

0 1

0 2

0 3

0 4

0 5

0 6

0 7

0 8

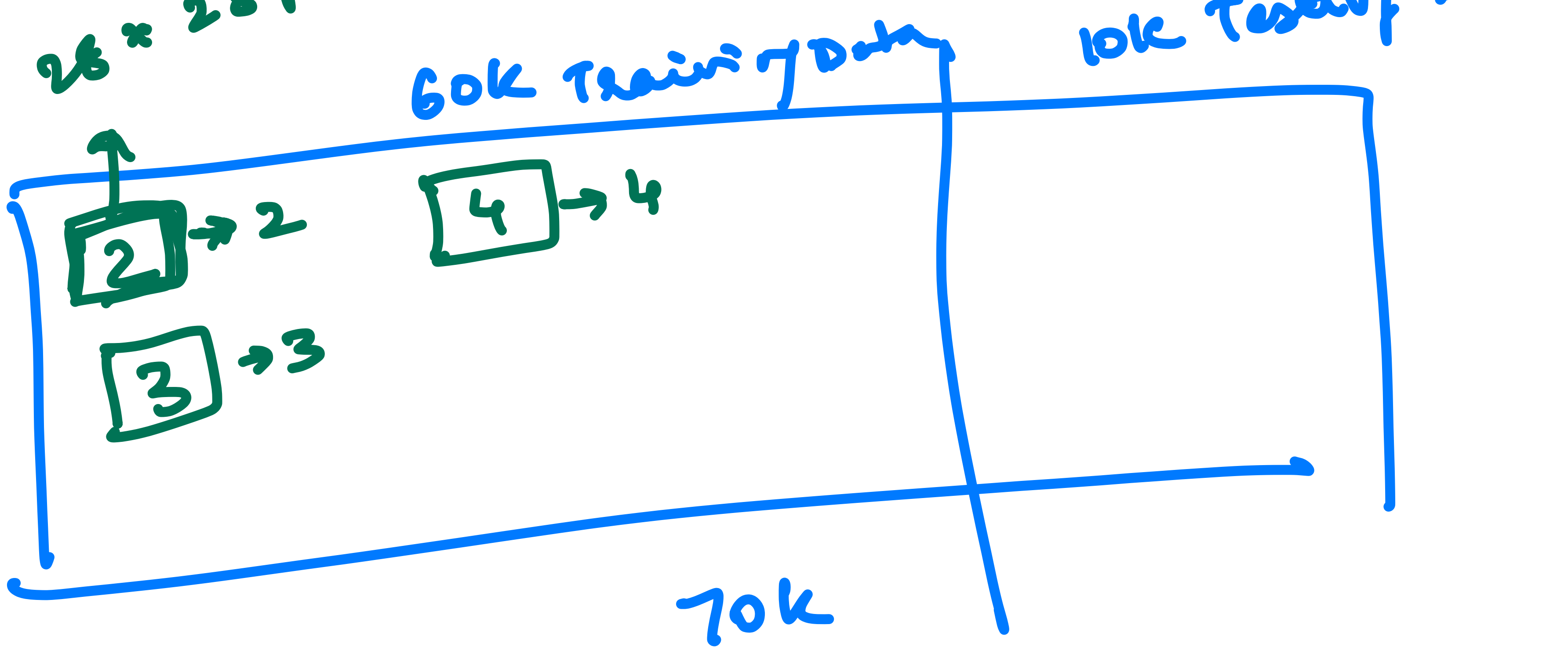
0 9

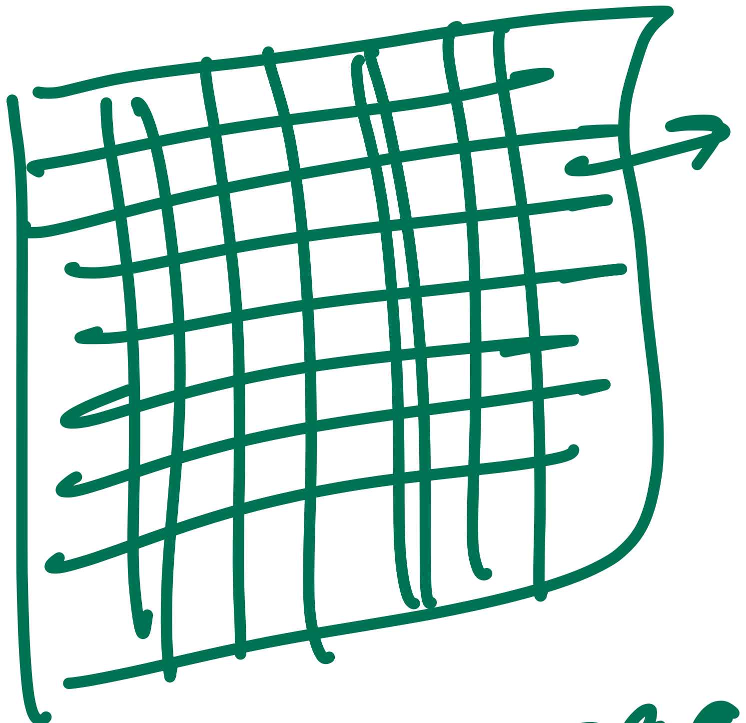
Tensorflow
PyTorch } → Training, Creating, Testing Models

Keras → wrapper on Tensorflow
API

TF → DataSets → MNIST → handwritten

load_data





0 - 255

0 → Black

255 → white

60k

x_train → 28 × 28

y_train → label

single digit no

0-9

10k

x_test → 28 × 28

y_test → label