Ans to the Question No-I



RNN (Recurrent Neural Networks) ⇒ Mainly for NLP

x — y — x

Not interested at ? → this time

in the ? → future

RNN auto completes it.

Google

Other usages is: translation, NER (Named Entity Recognition), Sentimental Analysis

Rafel bought a orange in 2nd July, 2000

NER: person — food — time

All these are sequence modelling problem.

→ *Important

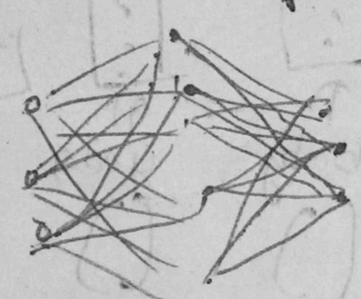
One hot encoding: Hello!

[ - - ] how

[ - - ] are

[ - - ] you?

Doing



তুমি কেমন আছো?

Simple ANN

sequence are important here. Fixed Neural Network is not going to work: (Doing ⇒ ? (error) Hello! ?)

Issues with ANN

# No fixed size of neurons

# Too much computations

# Parameters are not shared [Multiple sentences have same meaning]

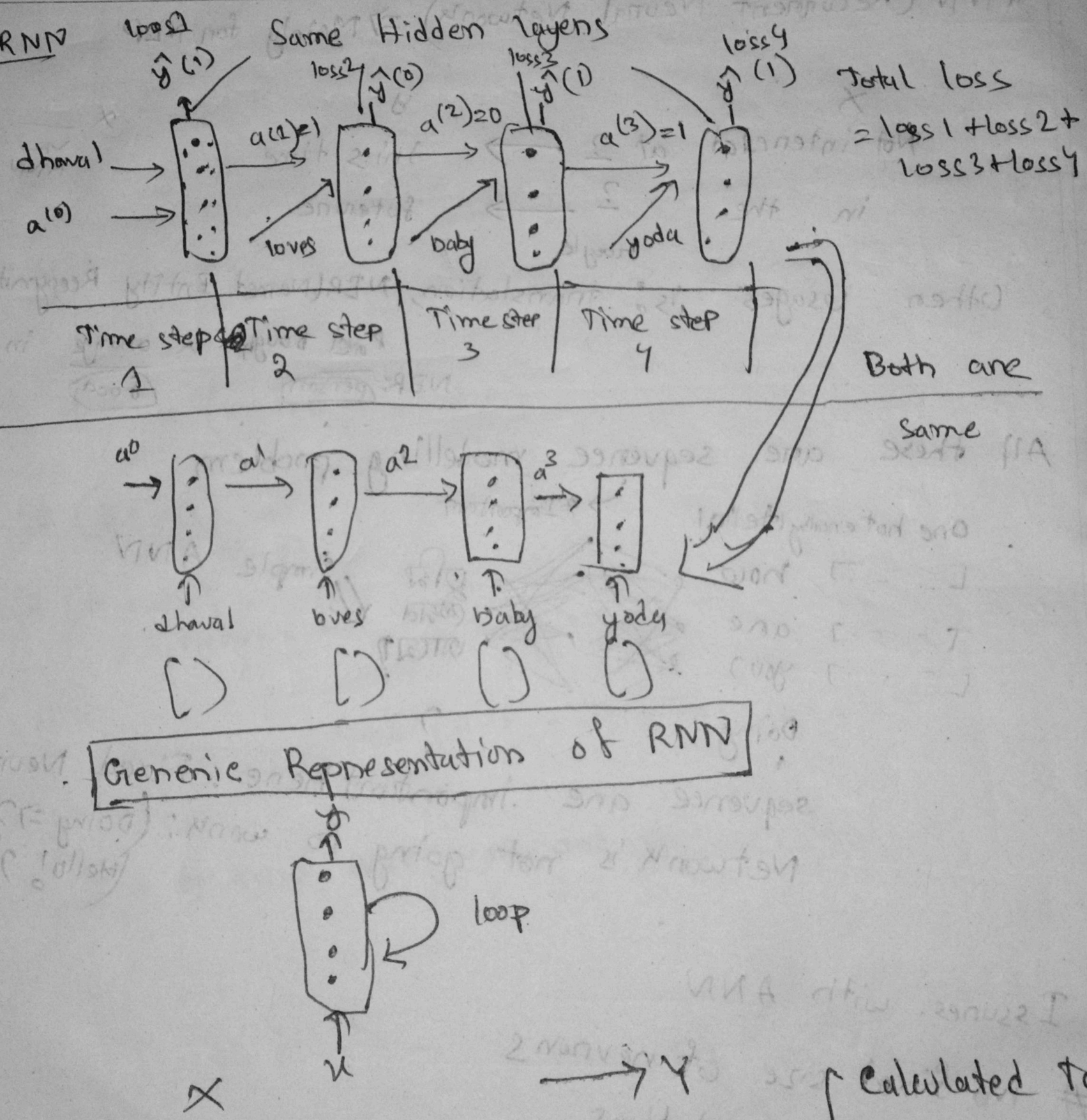
Sequence is not maintaining in ANN.

So we use RNN

Ans to the Question No-4



RNN




Total loss = loss1 + loss2 + loss3 + loss4

Both are same

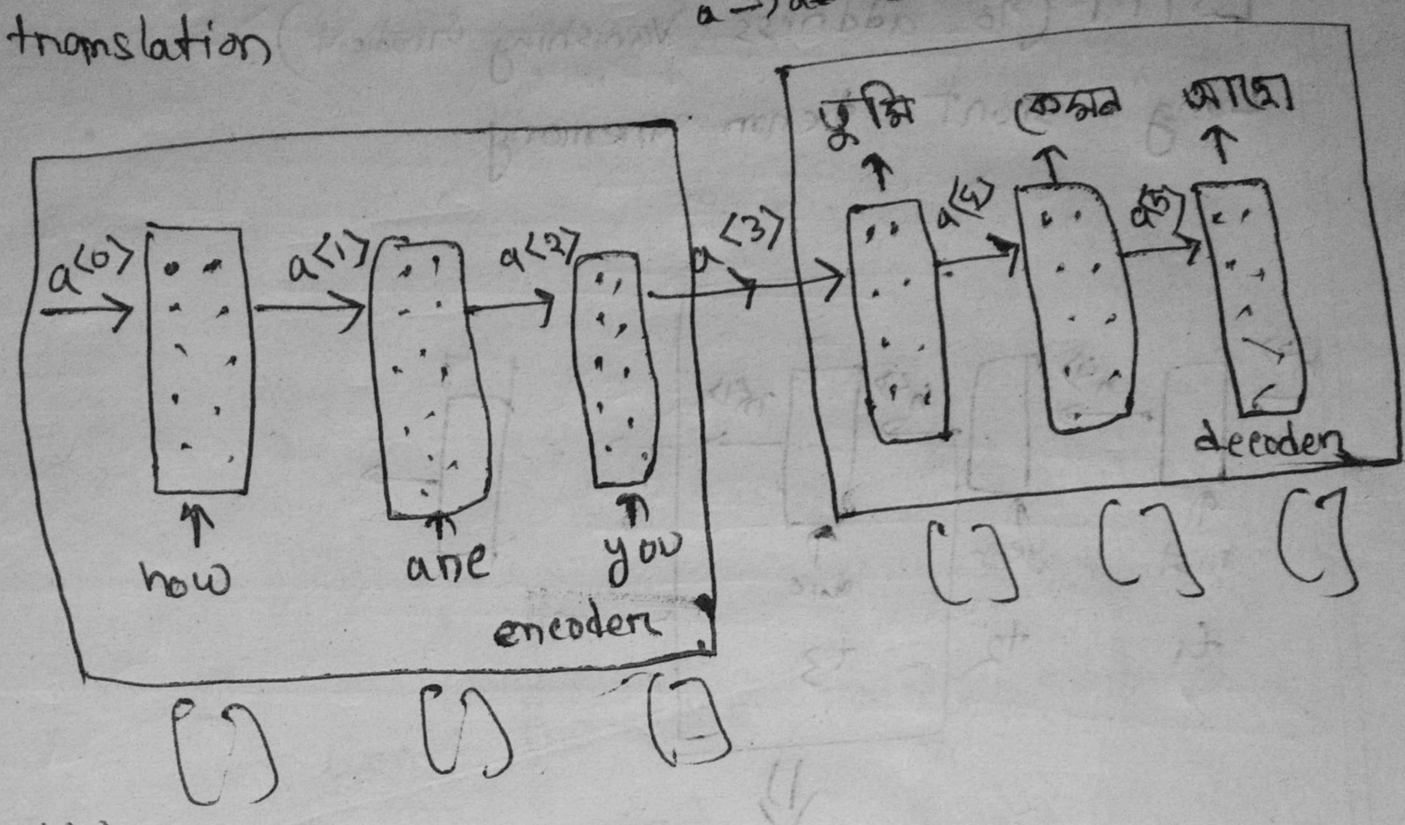
(Calculated total loss then, I do gradient Descent to reduce the losses)

Ans to the Question No-I



[ ] → hidden layer
a → ~~a~~ activation function

Language translation




Types of RNN

1. Many to Many $(x^{<1>}, x^{<2>} \dots x^{<t>} \Rightarrow \hat{y}^{<1>}, \hat{y}^{<2>}, \dots, y^{<t>})$
2. Many to One $(\langle x^{<1>}, x^{<2>} \dots x^{<t>} \Rightarrow \hat{y})$
3. One to Many $(x^{<1>} \Rightarrow \hat{y}^{<1>}, \hat{y}^{<2>}, \dots \hat{y}^{<t>})$

Vanishing Gradients

⇒ As ~~the~~ number of hidden layers grow, gradient becomes very small and weights will hardly change. This will hamper the learning process. (So use LSTM (Long short term memory))

Exploding Gradients

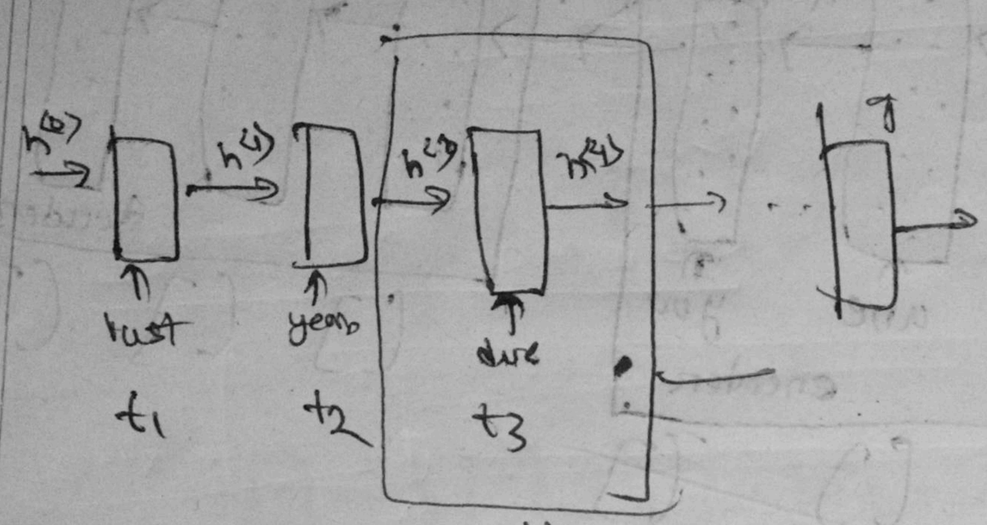
⇒ when individual derivatives are large, the final derivative will also ~~becom~~ become huge and weights would change drastically.

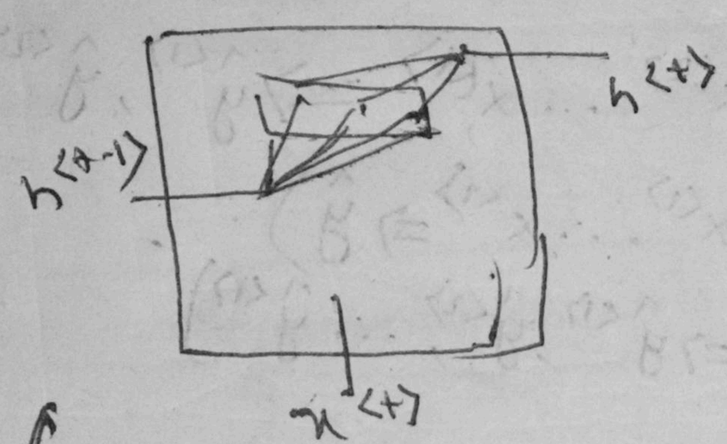
Ans to the Question NO-04



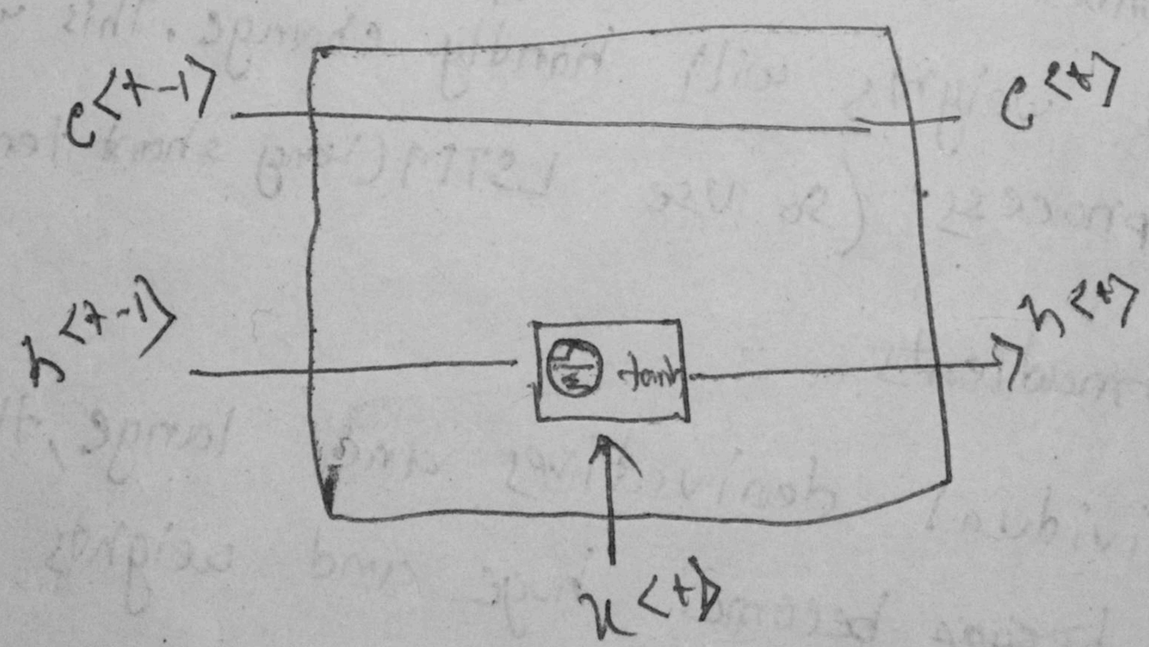
LSTM (to address Vanishing Gradient)

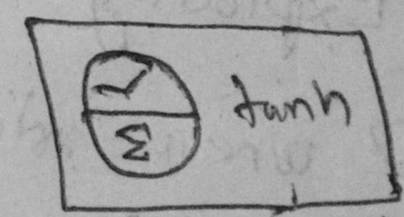
Long short term memory




(short Term memory cell)




hidden layers Neurons

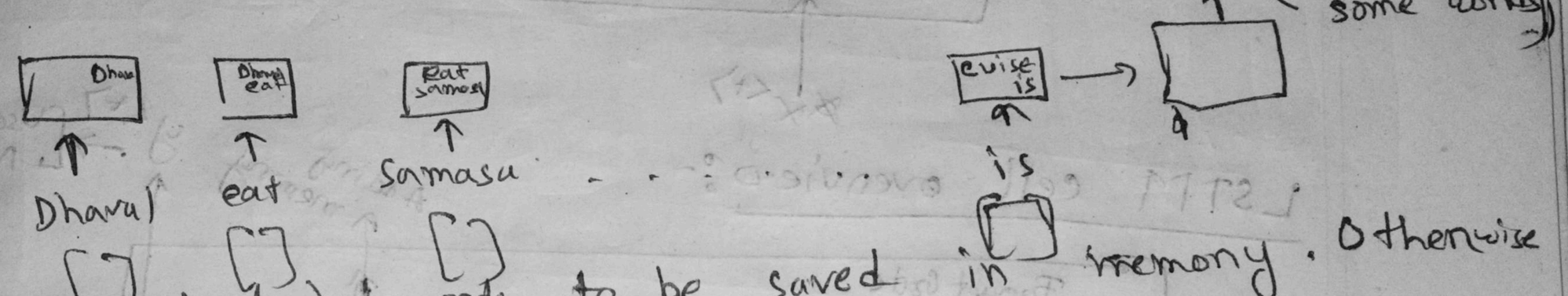
~~Sto~~ (short Term and long Term Memory)

Ans to the Question No-I



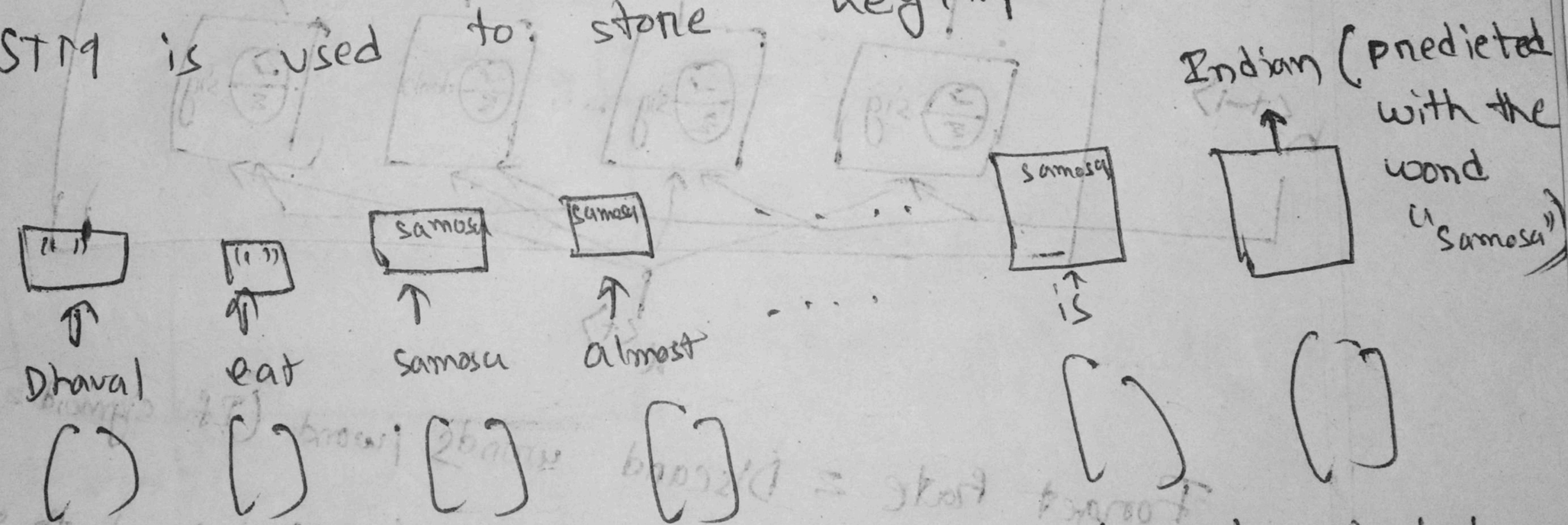
Sentence

Dhaval ~~e~~ eats samosa almost everyday it shouldn't be hard to guess that his favourite cuisine is Indian




So, Important keywords to be saved in memory. Otherwise RNN will not work. Therefore, we introduces LSTM, ~~Log~~ (Long short term memory)

LSTM is used to store key/important words

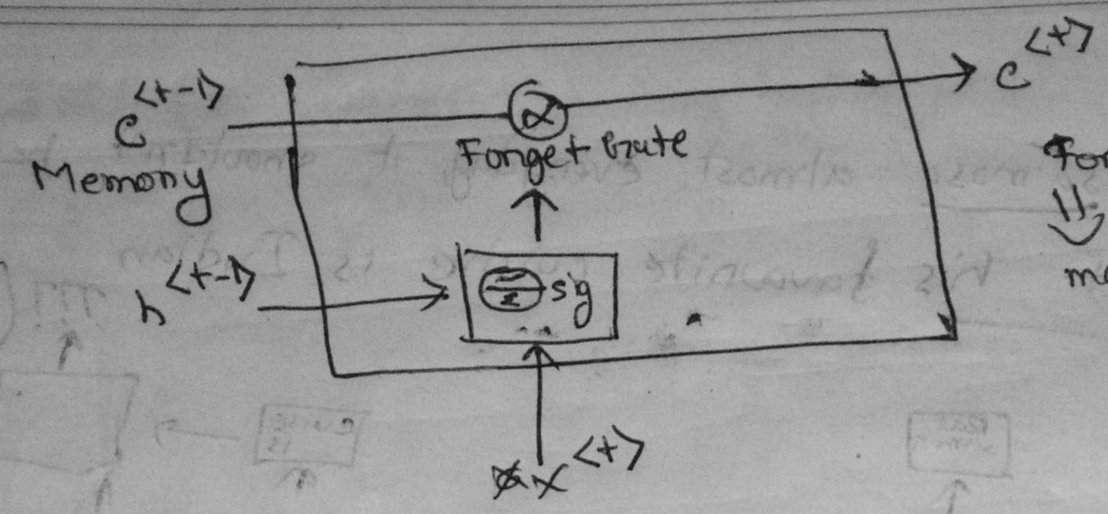



while training, RNN understands what to discard, what to store.

If the words gets biggen, It may ~~reduce~~ discard ~~any~~ some words. Again what to discard or store is decided by RNN

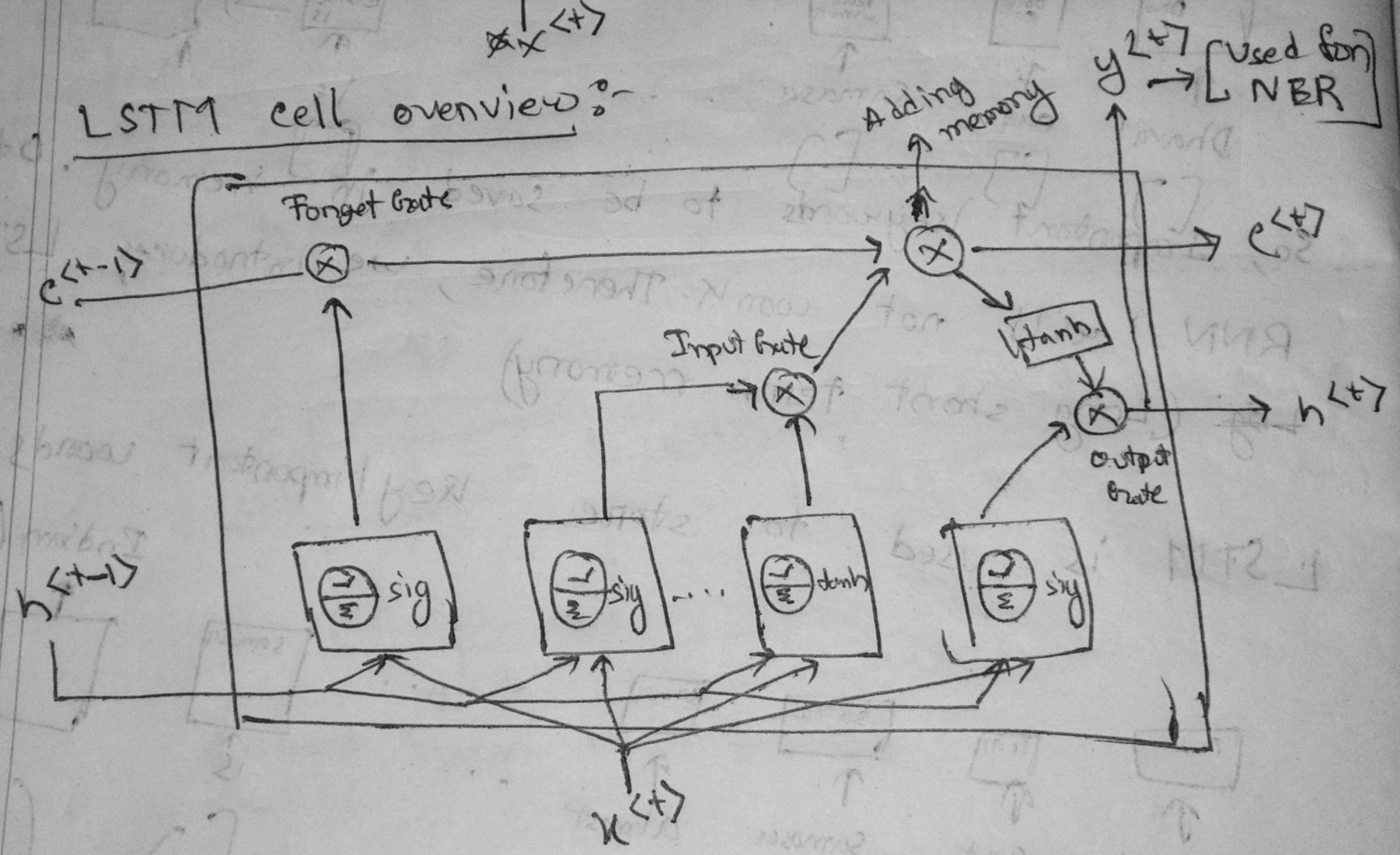
Ans to the Question No-04






Forget Gate
⇓
make a zero (0) vector.

LSTM cell overview:-




Forget Gate = Discard words / word - (If sigmoid = 0/1)

Input Gate = Add word (Multiplying both sigmoid and tanh

Output Gate = passes to next hidden state (tanh of previous gate)
[Take Long Term memory of tanh and multiply with sigmoid]

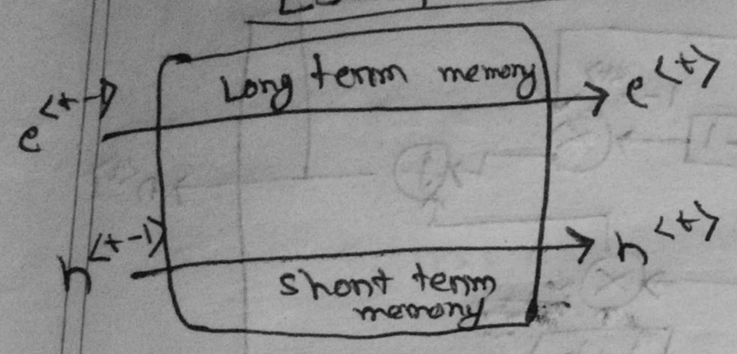
[Consider updating weights through Back propagation]

Ans to the Question No-1



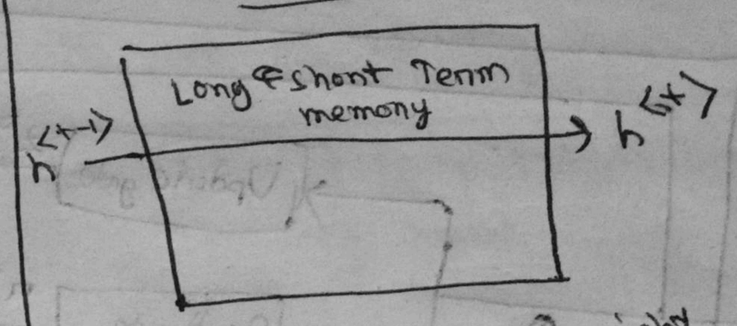
GRU (Gated Recurrent Unit)

LSTM



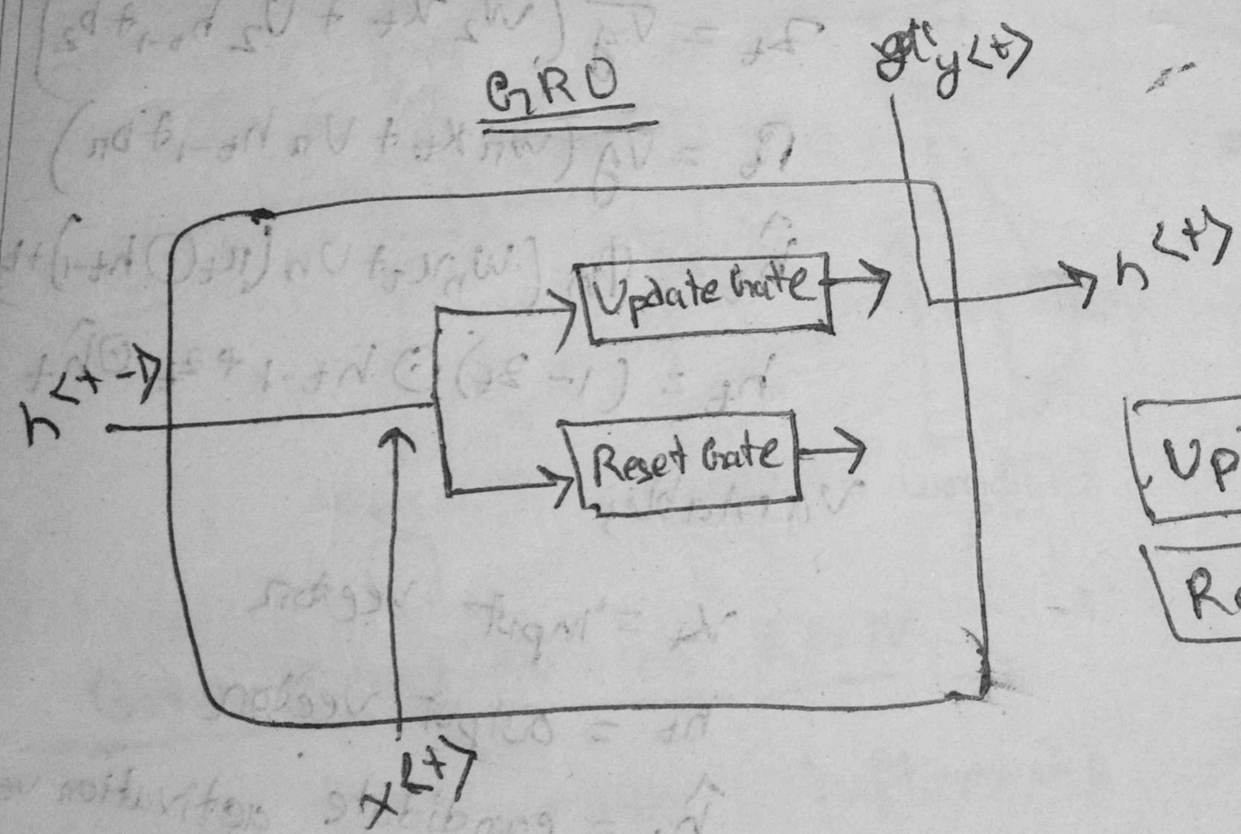

LSTM is heavyweight,
2 like cell state

GRU




GRU is light weight,
only hidden state

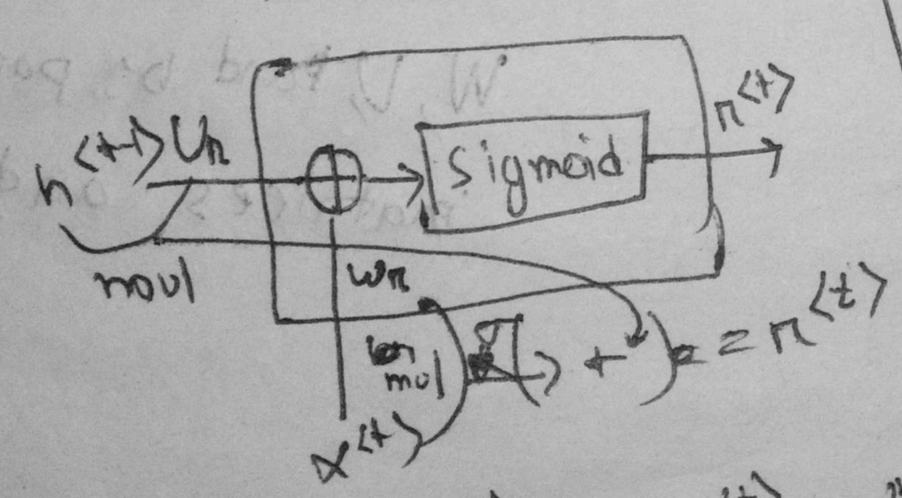
GRU




Update Gate [How much past memory to return]

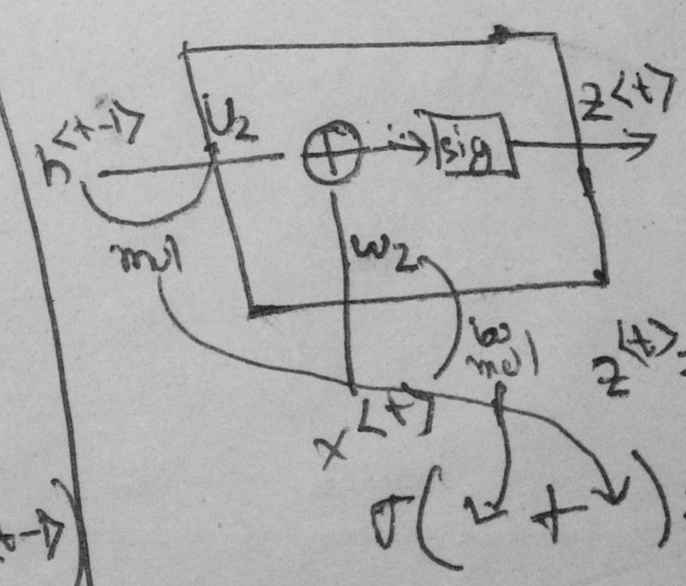
Reset Gate [How much past memory to forget]

Reset Gate



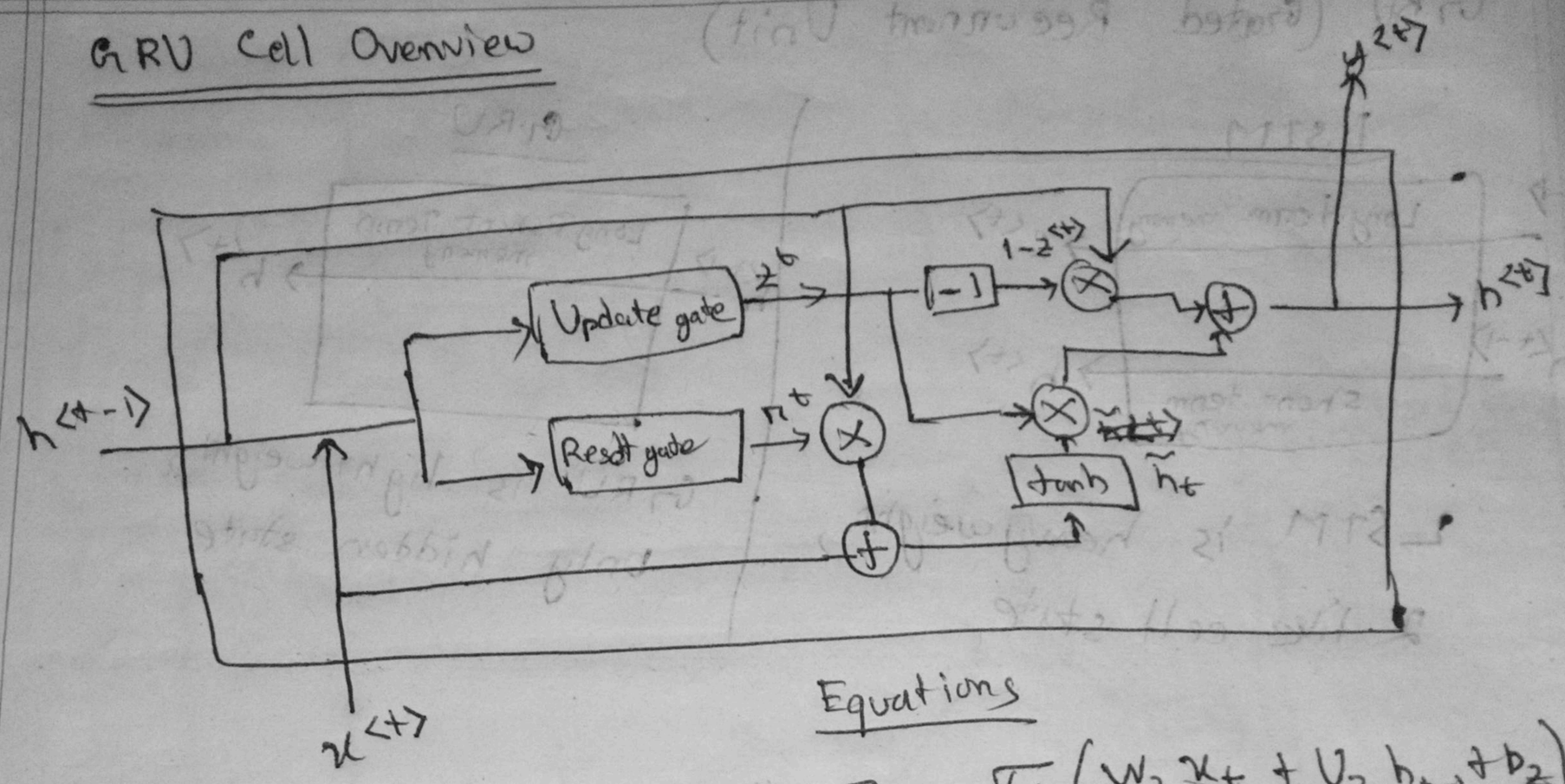

$r^{<t>} = \sigma(W_r x^{<t>} + U_r h^{<t-1>})$

Update Gate




$z^{<t>} = \sigma(W_z x^{<t>} + U_z h^{<t-1>})$

## GRU Cell Overview




### Equations

$$z_t = \sigma_g\left(W_z x_t + U_z h_{t-1} + b_z\right)$$

$$r_t = \sigma_g\left(W_r x_t + U_r h_{t-1} + b_r\right)$$

$$\hat{h}_t = \phi_h\left(W_h x_t + U_h\left(r_t \odot h_{t-1}\right) + b_h\right)$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t$$

### Variables

$x_t$ = input vector

$h_t$ = output vector

$\hat{h}_t$ = candidate activation vector

$z_t$ = Update gate vector

$r_t$ = Reset gate vector

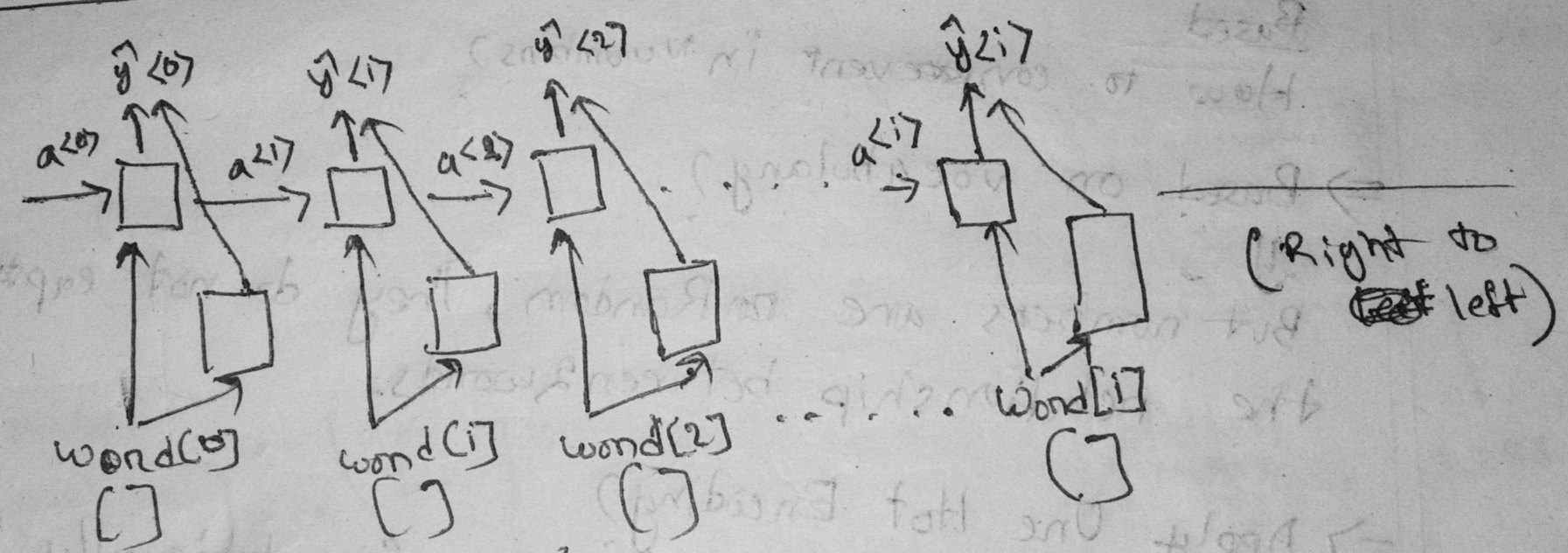
W, U, and b: parameter matrices and vector

Ans to the Question No-1



| LSTM | GRU |
|---|---|
| 3 Gates: Input, output, forget | 2 Gates: Reset and Update |
| More accurate on longer sequence, less efficient | More effient computation wise, getting more popular |
| 1995-1997 (Invention) | 2014 - (Invention) |

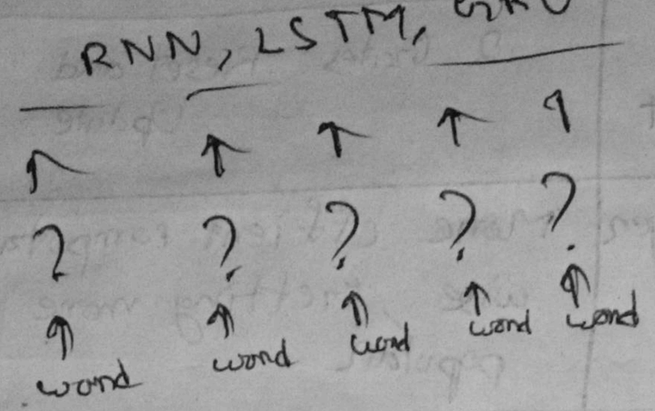
Bidirectional RNN




Both direction of RNN

Simple RNN has influences of previous words. Now Both direction of RNN will have next and previous words.

# Word Embedding




? ⇒ Vectorized form of word

Machine can not understand text. It understands numbers

Based
How to convert in Numbers?

⇒ Based on vocabulary?
⇓
But numbers are Random, They do not capture the relationship between 2 words.

⇒ Apply One Hot Encoding?
⇓
Do not capture relationship, Computationally inefficient

⇒ Word Embedding?
⇓
- captures the relationship between words.

Word → Features (Converted to Vector), therefore holds the relationship between words.

These are not hand crafted, Retrieved by Models.

Ans to the Question No-1



## How to calculate word embedding

By Model while training

1. Using ~~spersupervised~~ learning
2. Using self-supervised learning (Word2Vec, Glove)

take an NLP problem and try to ~~solo~~ solve it

By _____. ~~Get word~~ Now, get word embedding

as a side effect

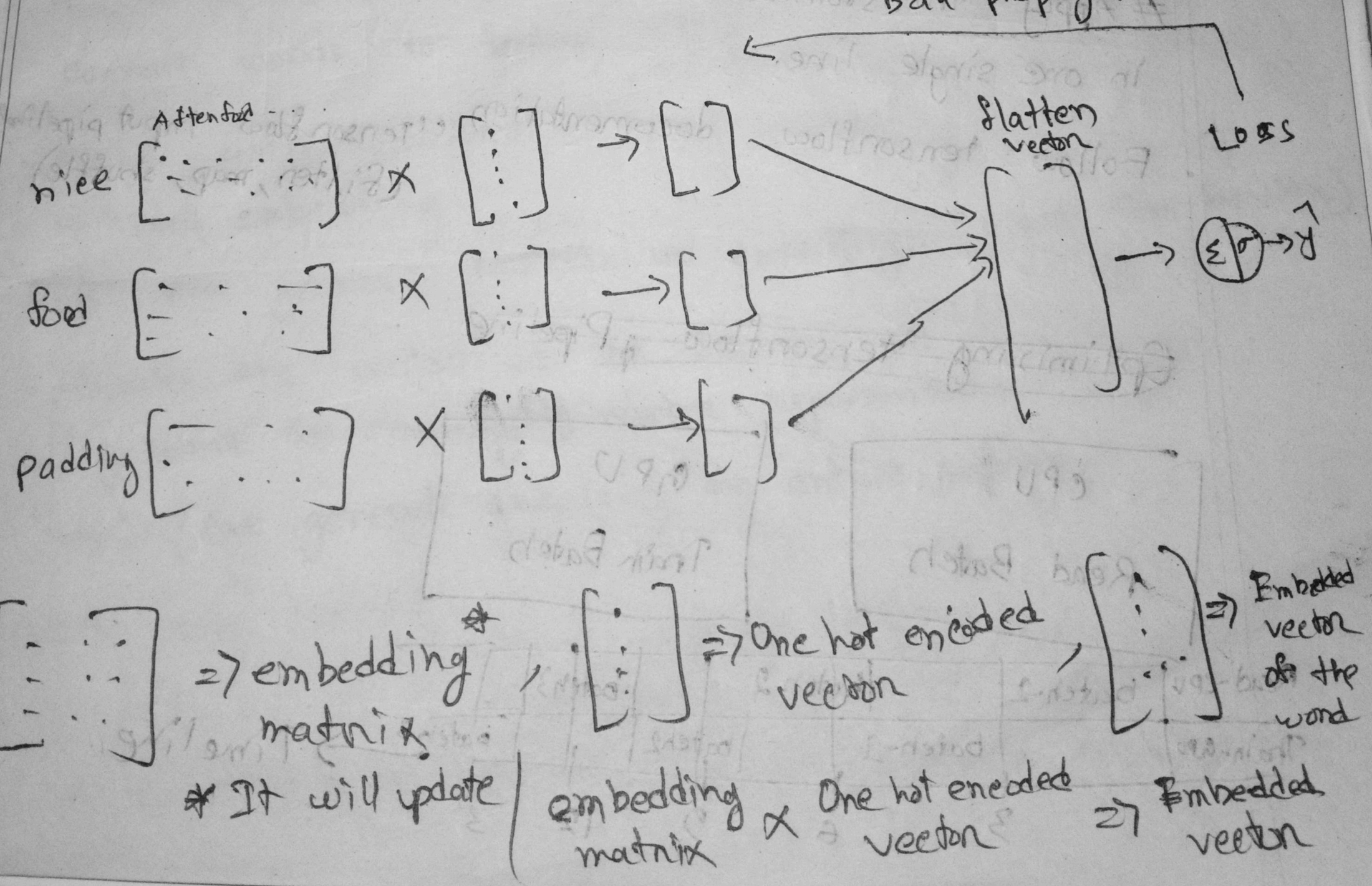
_____ (Yourself) ⇒ supervised

_____ (Model) ⇒ self supervised

Neural networks.

Sentence

Aften, food . . . poon , zona /



# Tensorflow Documentation

Tenson flow minnon stanagedy
⇓
Use distributed training on dgx on on any multi gpu system.
Follow tensonflow documentation
"Distributed training with tensonflow"

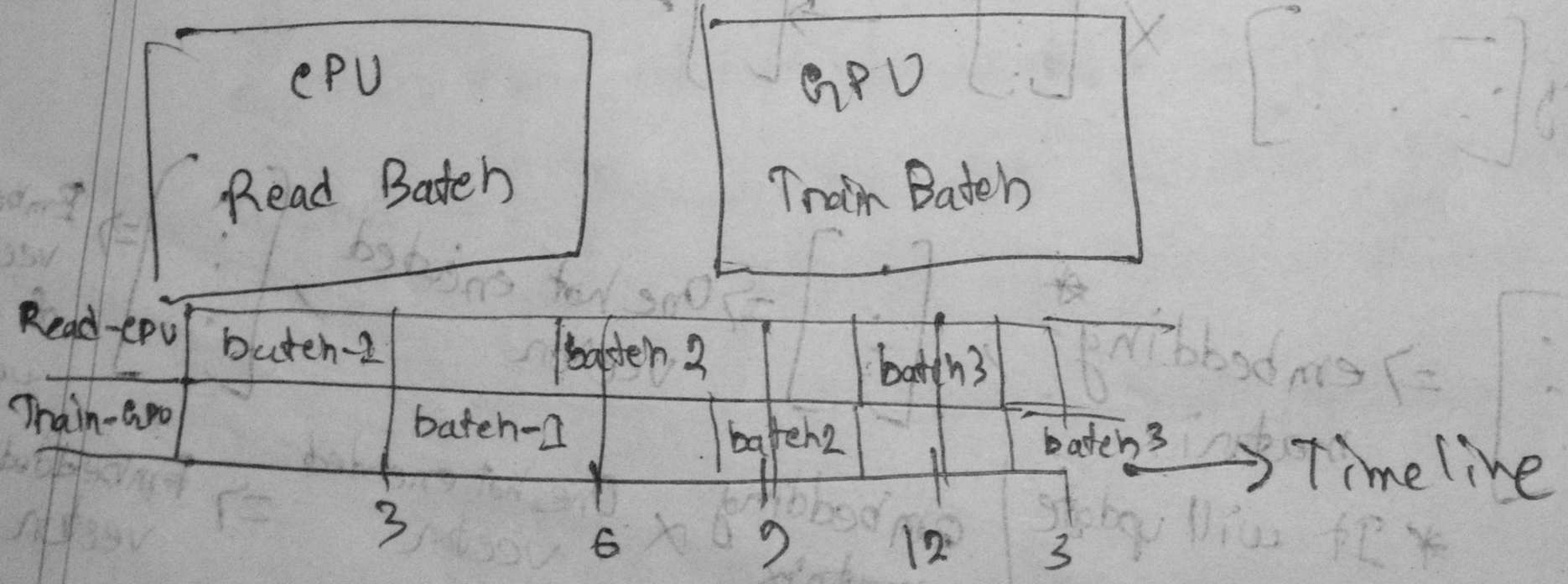
Tensonflow input pipeline (Build data pipeline)
(↳) create special data structure (batch's)
tf.data.Dataset
(↳) single line of code to eventhing tasks (filter, map)

**Benefits**

# Handle huge datasets by streaming them from disk using batching
# Apply transformation to make dataset ready for model in one single line.

Follow tensonflow documentation "Tenson flow input pipeline (filter, map, shuffle)

~~Optimizing tensonflow Pipeline~~



Ans to the Question No-1



Optimizing :=>

| | | | | | |
|---|---|---|---|---|---|
| Read - CPU | batch - 1 | batch - 2 | batch - 3 | | |
| Train - GPU | | batch-1 [ ] | batch-2 [ ] | batch-3 | |

3 6 9 12 15 → Timeline

tf.data.Dataset.prefetch(1)

tf.data.Dataset.prefetch(Autotune) => Tensorflow will decide

tf.data.Dataset.cache() [It saves close to 2x time]

↳ will not use map function again

## BERT

How to capture similarities between two words?

convert words to feature vectors

Issue with word2vec

# Fixed embedding

BERT can generate contextualized embeddings. (More than Word2vec)

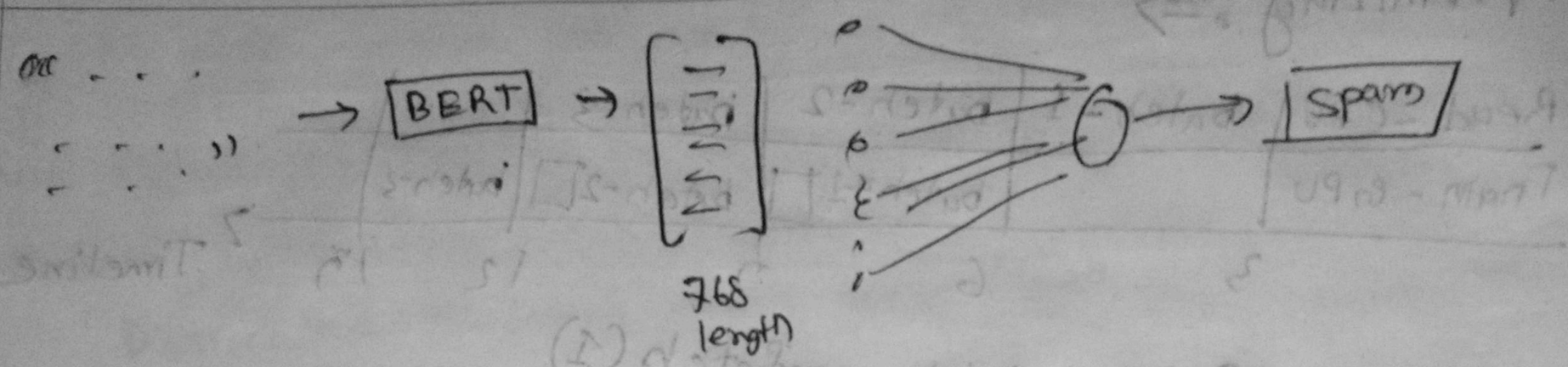
↳ looks the context of the statement and generate the meaningful number representation.

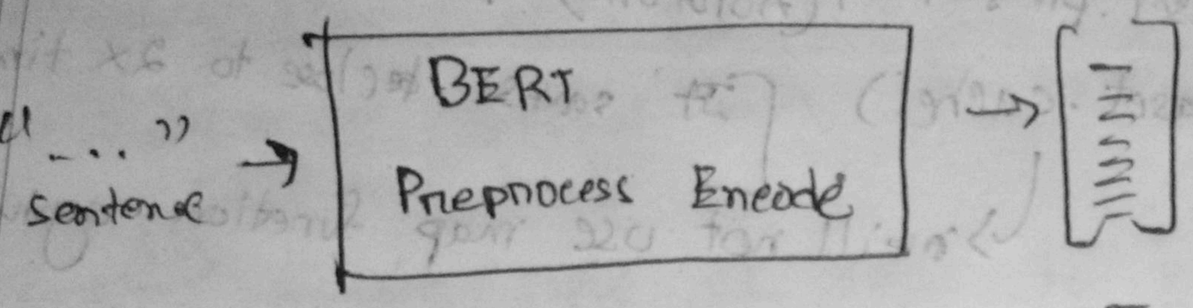
↳ It can generate embedding for entire sentence

BERT was trained ↳ Masked Language Model.

↳ Next sentence prediction

Bidirectional Encoder Representation from Transformers.



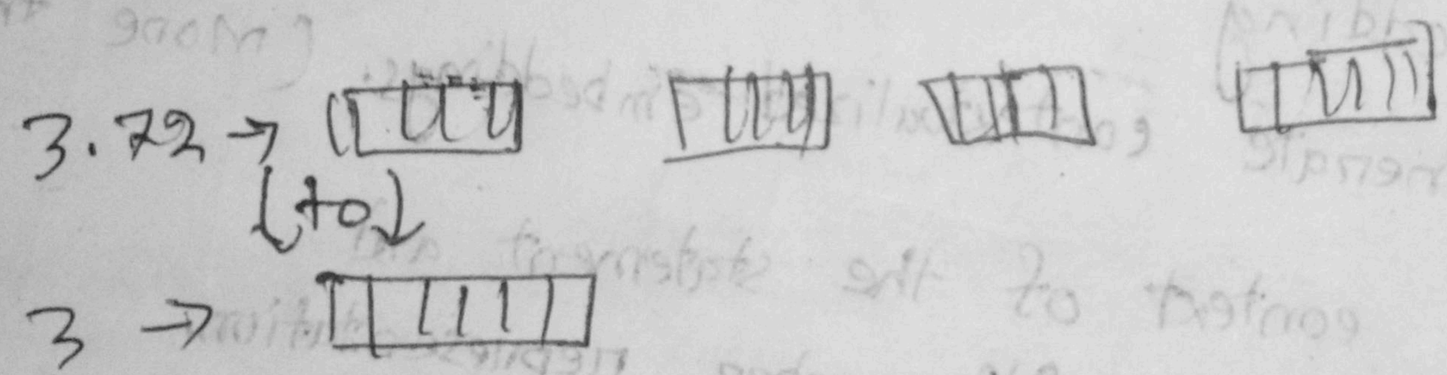




TEXT classification using BERT

# Tensorflow serving Tutorial.

## Quantization

It is a process of reducing model size so that it can run on EDGE devices.




It converts floating number to integer by taking floor values. Therefore reducing spaces

## Benefits

Run ML models efficiently on EDGE devices

Faster inference

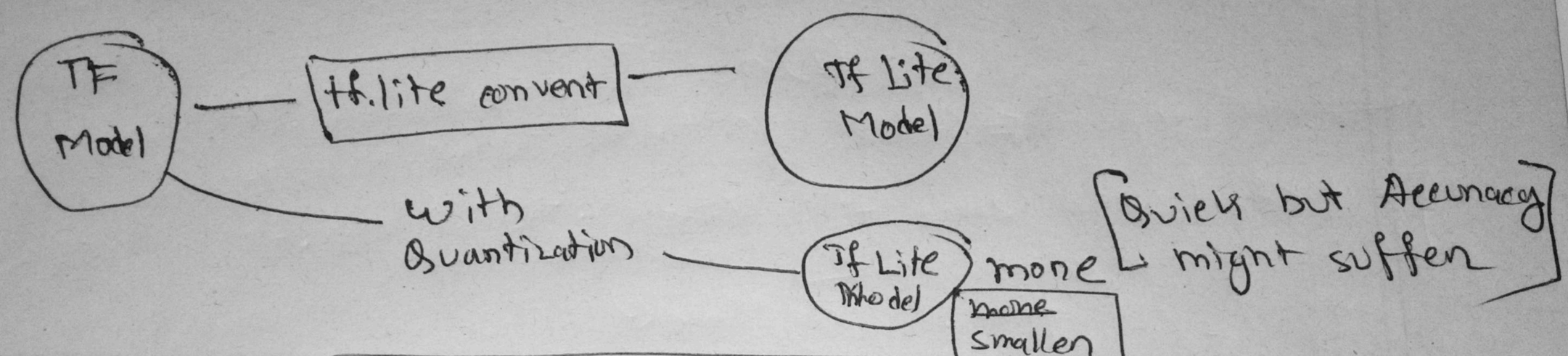
Ans to the Question No-1



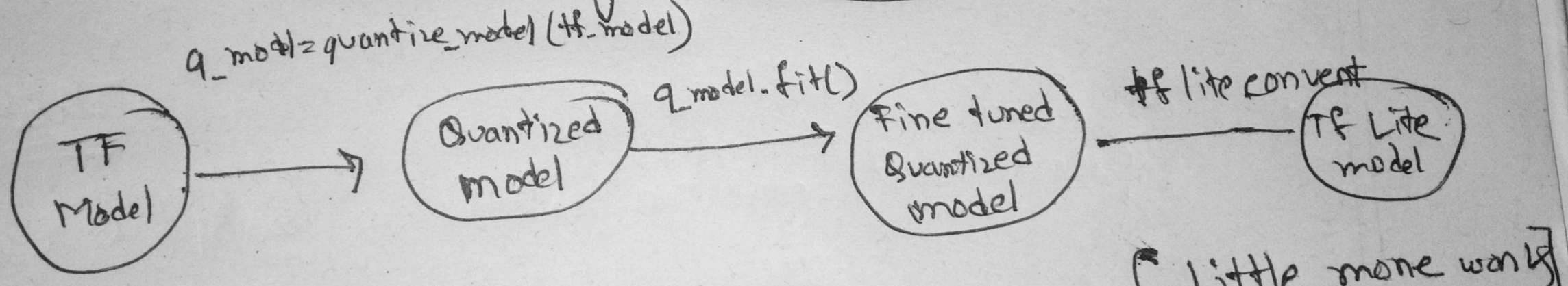
2 ways to perform quantization

→ Post Training Quantization

→ Quantization Aware Training




Post Training Quantization




Quantization aware Training