NLP+ Deep Learning(Generative AI)

Natural Language Processing – Deep Learning Modules

- Word Embedding
- LSTM-Long Short -Term Memory
- Sequence 2 Sequence
- Transformer

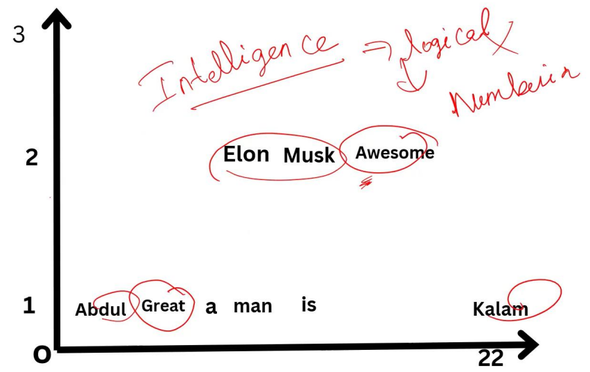
Abdul Kalam is a Great man

Elon Musk is Awesome

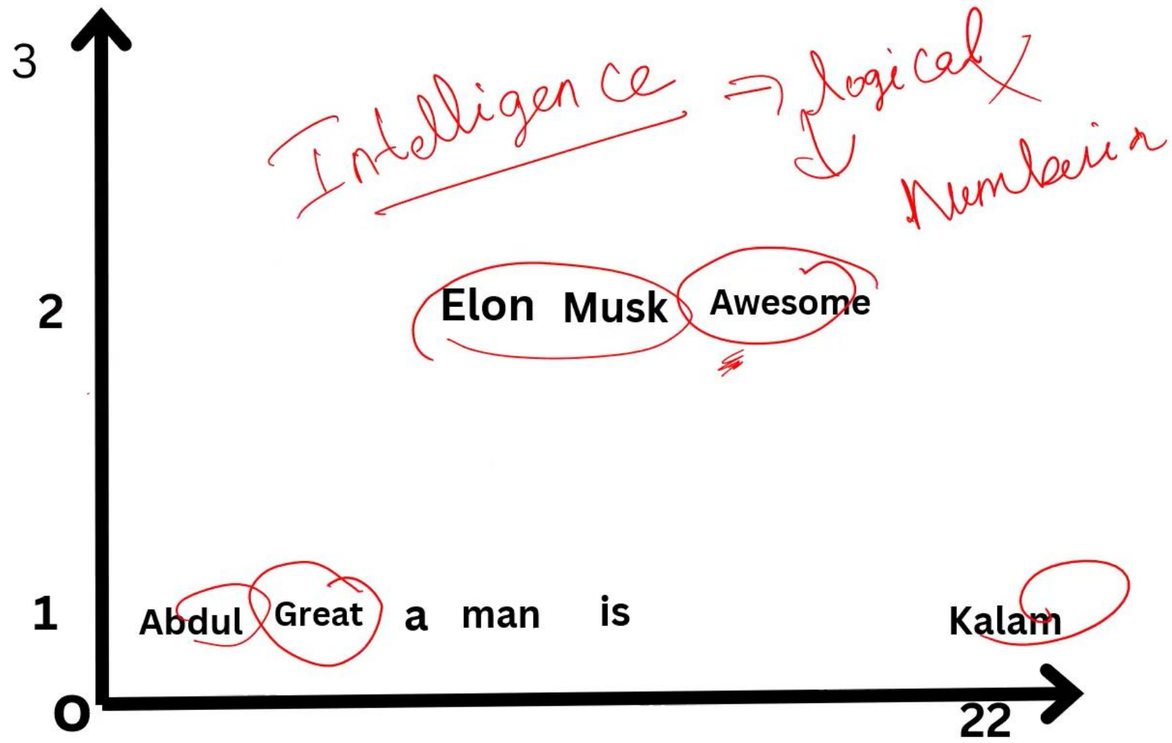
Abdul	1
Kalam	22
is	13
a	4
Great	5
man	6

Elon	11
Musk	12
is	13
Awesome	19

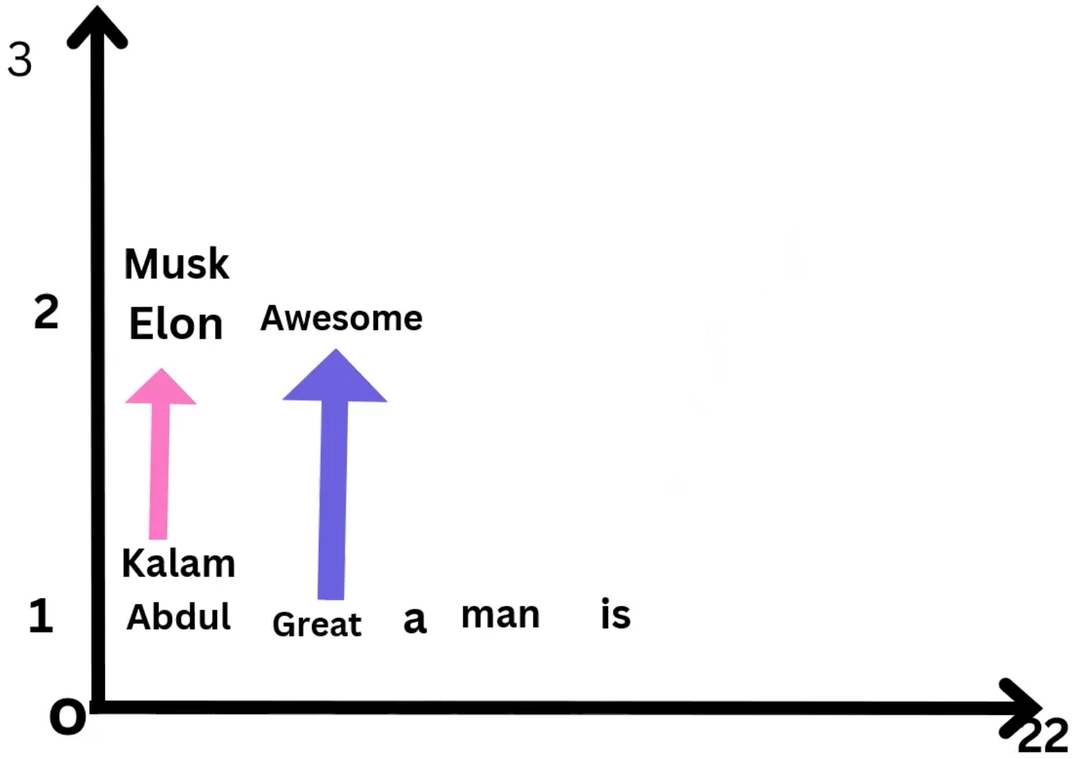
Why Word Embedding is needed?



Why Word Embedding is needed?

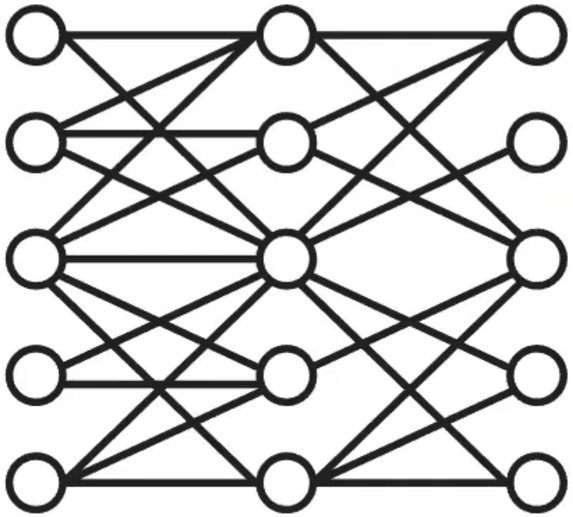


Why Word Embedding is needed?



Word Embeddings using Neural Network

Using Neural Network



We can create meaningful numbers

Word → Vec

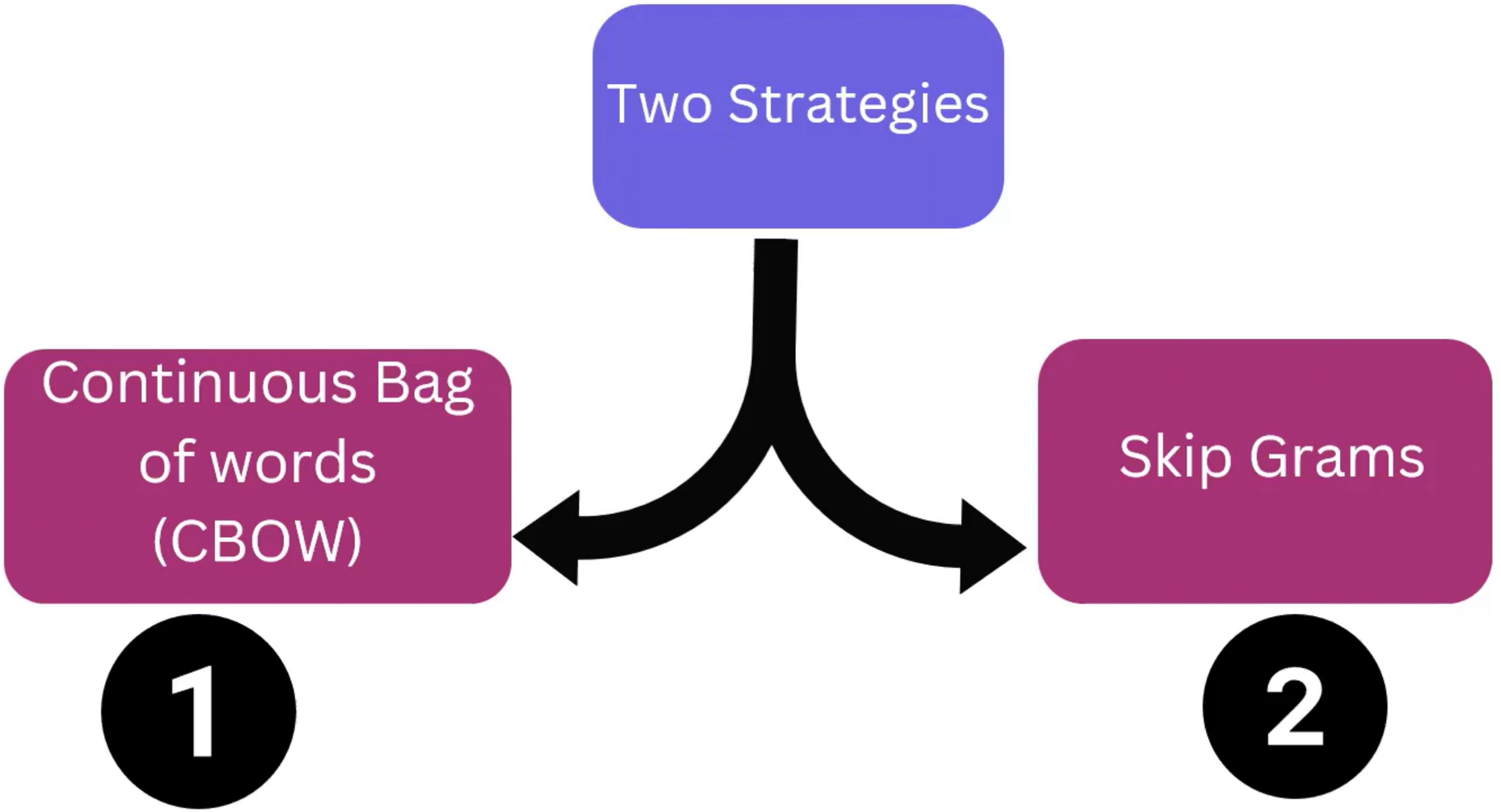
fast-fast

byte pair

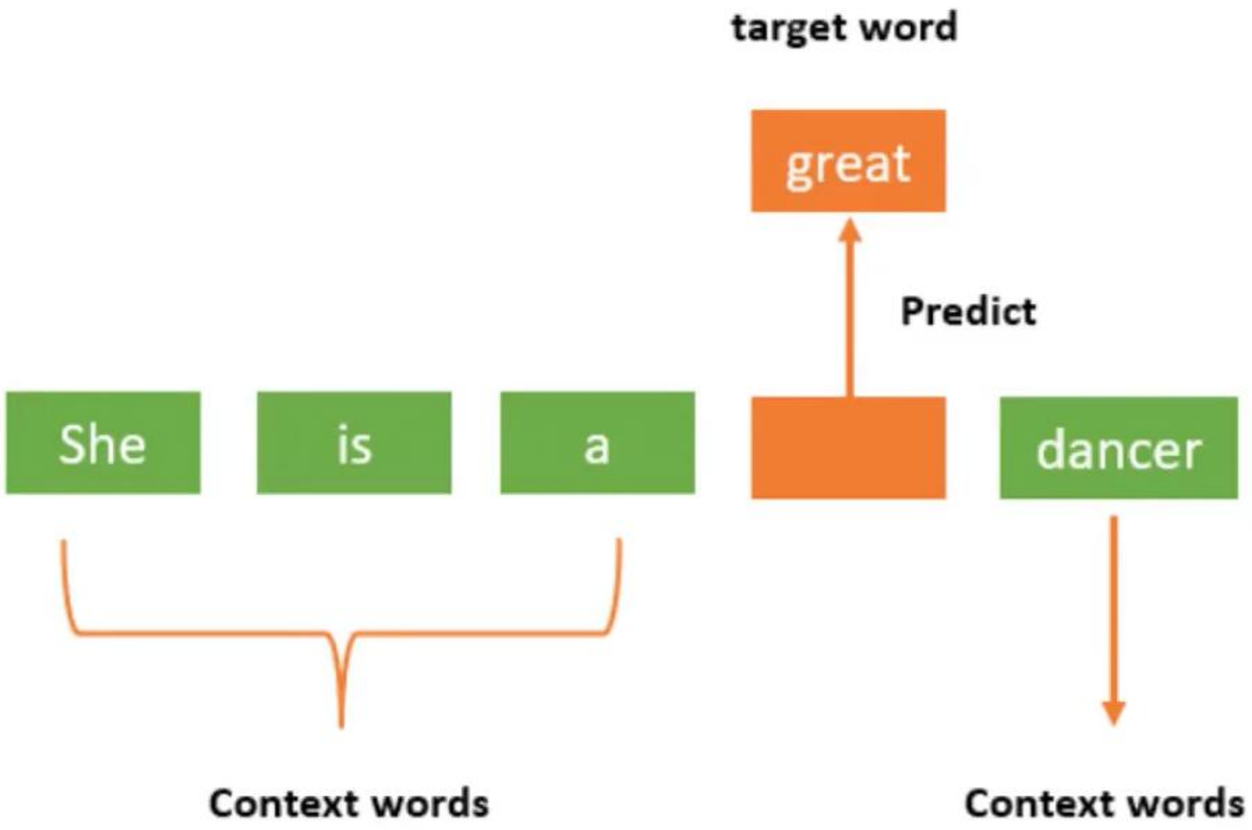
pre

Words → Meanings

Word Emebedding-Word2Vec



Word Emebedding-Word2Vec-CBOW



Continuous Bag of words (CBOW)

The quick brown fox jumps over the lazy dog

Context Word, Target Word, Window Size

Context Word: (-2,Traget Word,+2)

The quick brown fox jumps over the lazy dog
(-2) (-1) (T) (1) (2)

Window 3:
Context words ('The', 'quick', 'fox',jumps) -
Target word ('brown')

The quick brown fox jumps over the lazy dog
(-2)(-1) (T) (1) (2)

Window 1:
Context words ('N/A','N/A', quick, brown) -
Target word ('The')

Window 2
Context words(NA,the,brown, fox)
Target word("quick")

The quick brown fox jumps over the lazy dog
(-2) (-1) (T) (1) (2)

N/A,N/A, quick, brown

N/A, The, brown, fox

The, quick, fox, jumps

The

quick

brown

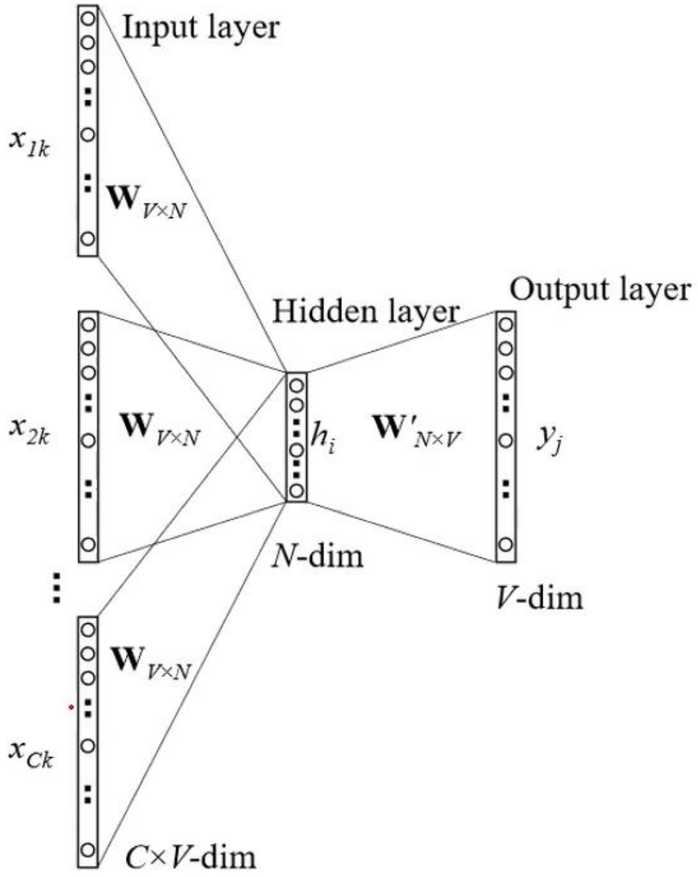


Figure 2: Continuous bag-of-word model

Encode Words as Indices

|

One-Hot Encoding

|

CBOW Model

|

Average Context Vectors

|

Predict Target Word Vector

|

Calculate Loss/Error

|

Backpropagation

|

Update Word Vectors

|

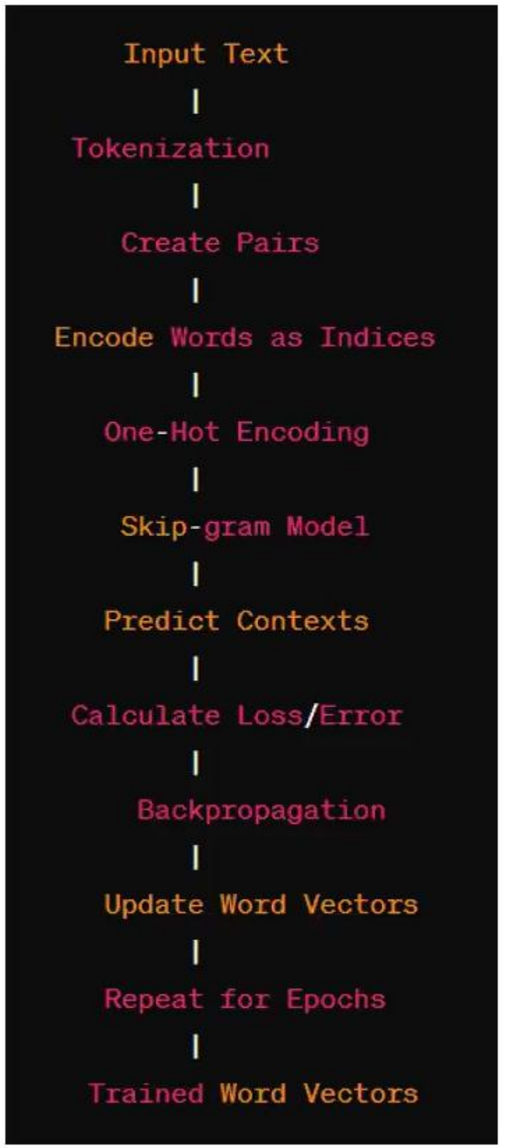
Repeat for Epochs

|

Trained Word Vectors

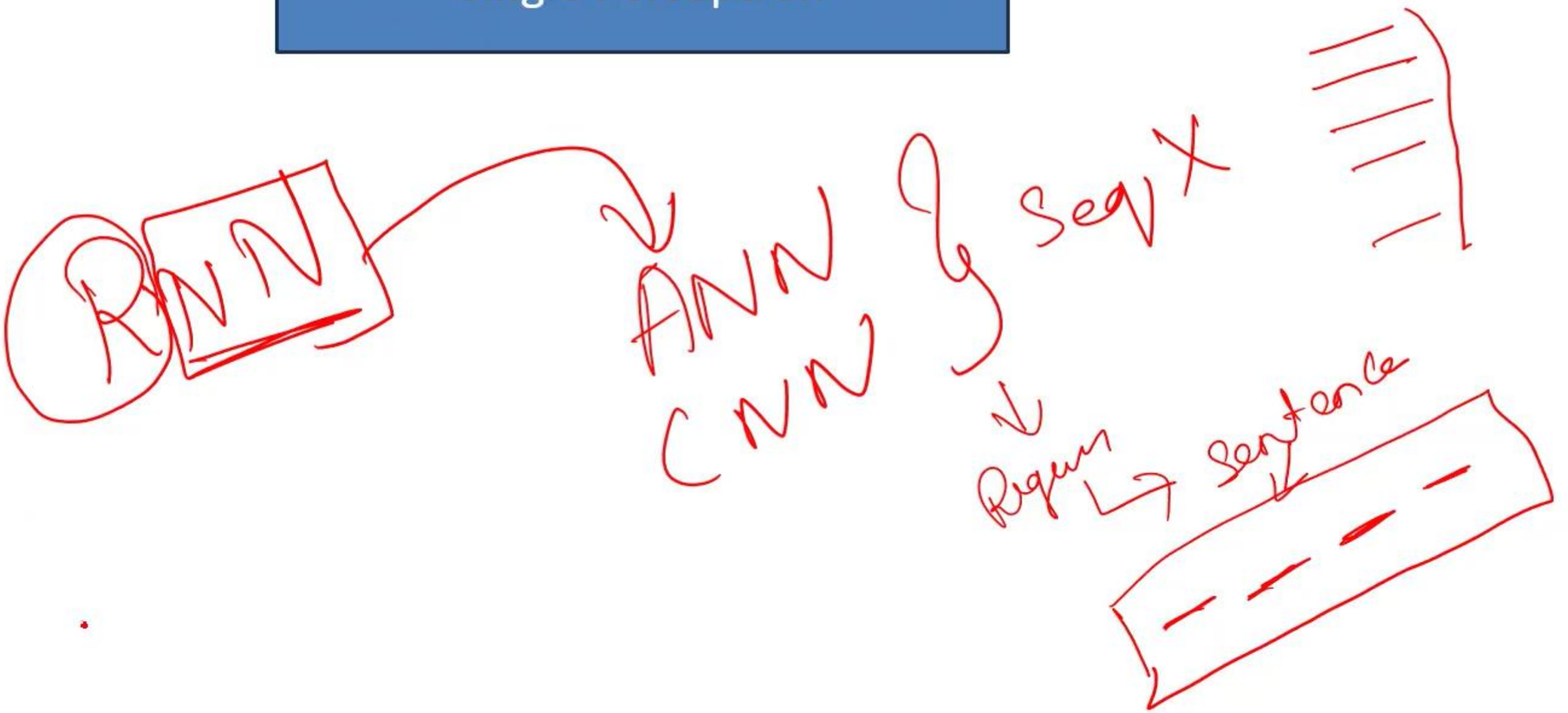
Word Emebedding-Word2Vec-Skip Gram

Input	Output
The	N/A,N/A, quick, brown
quick	N/A, The, brown, fox
brown	The, quick, fox, jumps

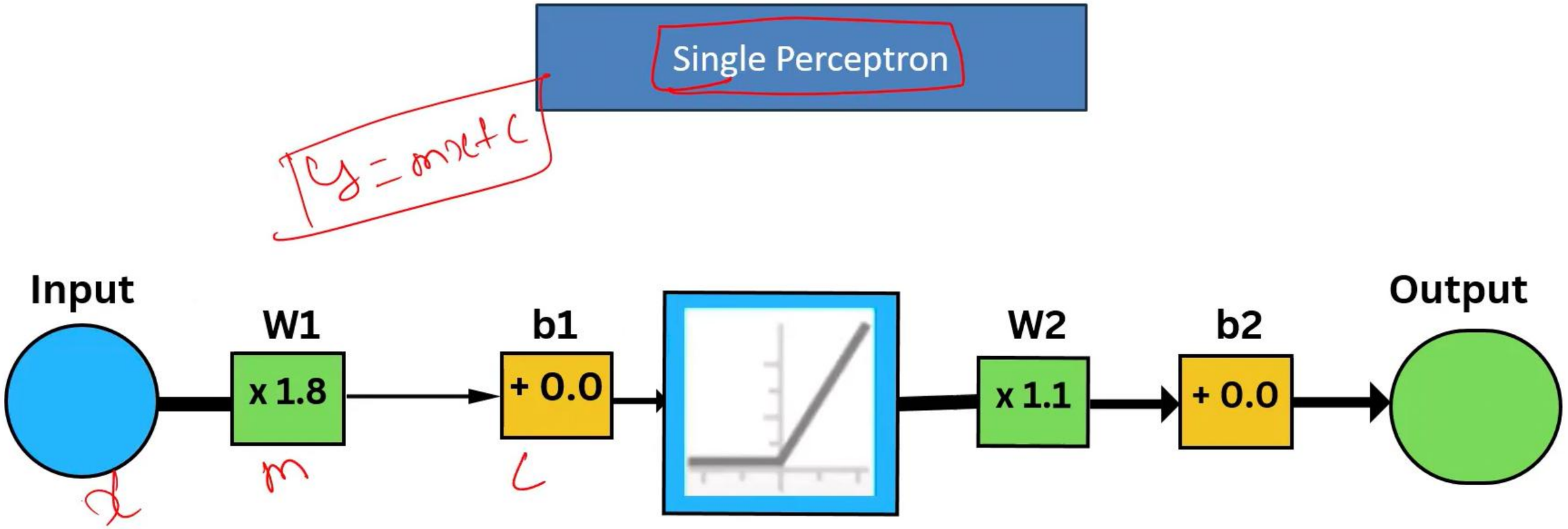


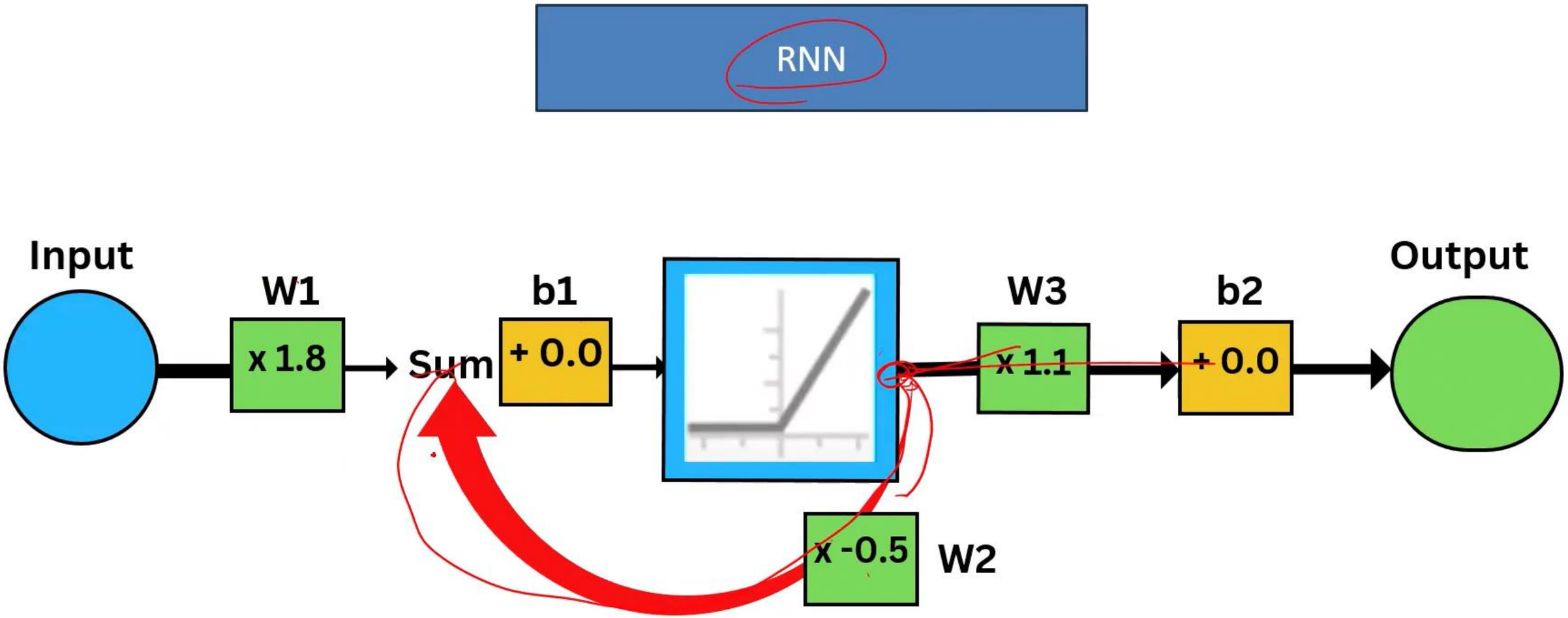
Long Short-Term Memory- (LSTM)

Single Perceptron

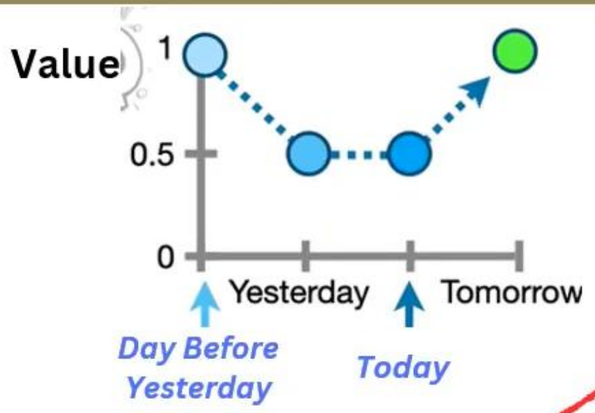


Long Short-Term Memory- (LSTM)

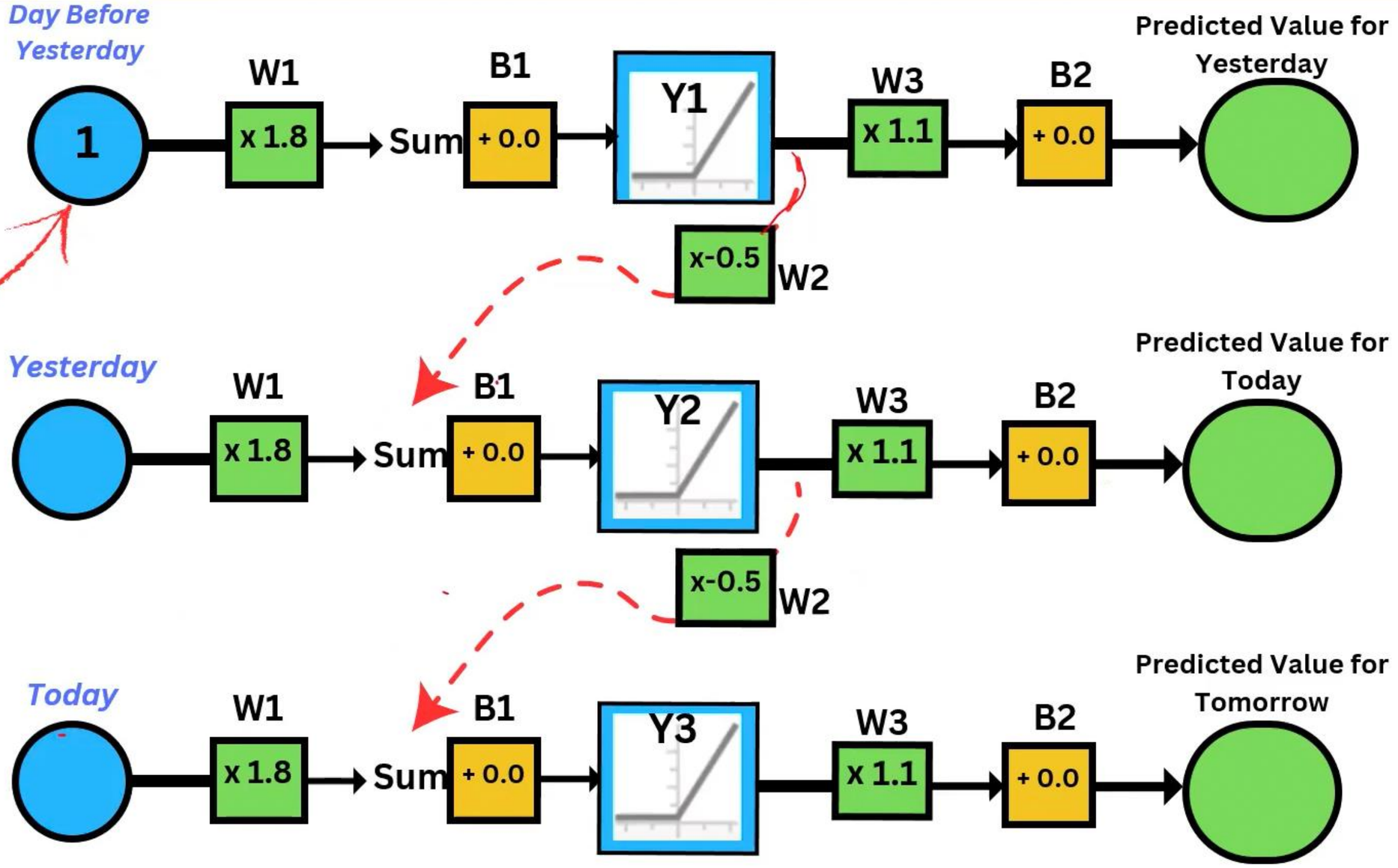



RNN
Input
W1
x 1.8
Sum
b1
+ 0.0
W3
x 1.1
b2
+ 0.0
Output
x -0.5
W2

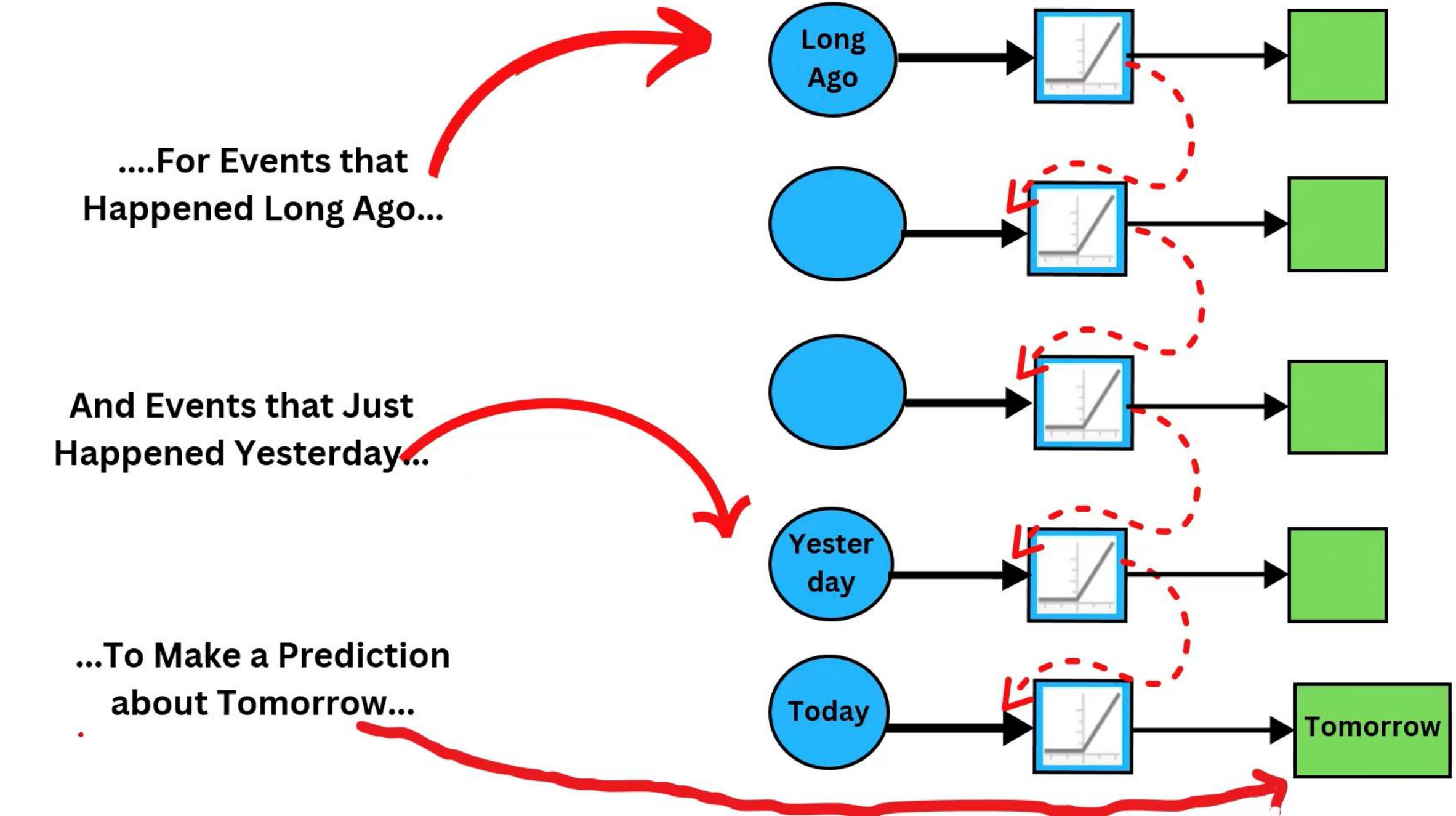
Long Short-Term Memory- (LSTM)



Unfold-RNN



Long Short-Term Memory- (LSTM)

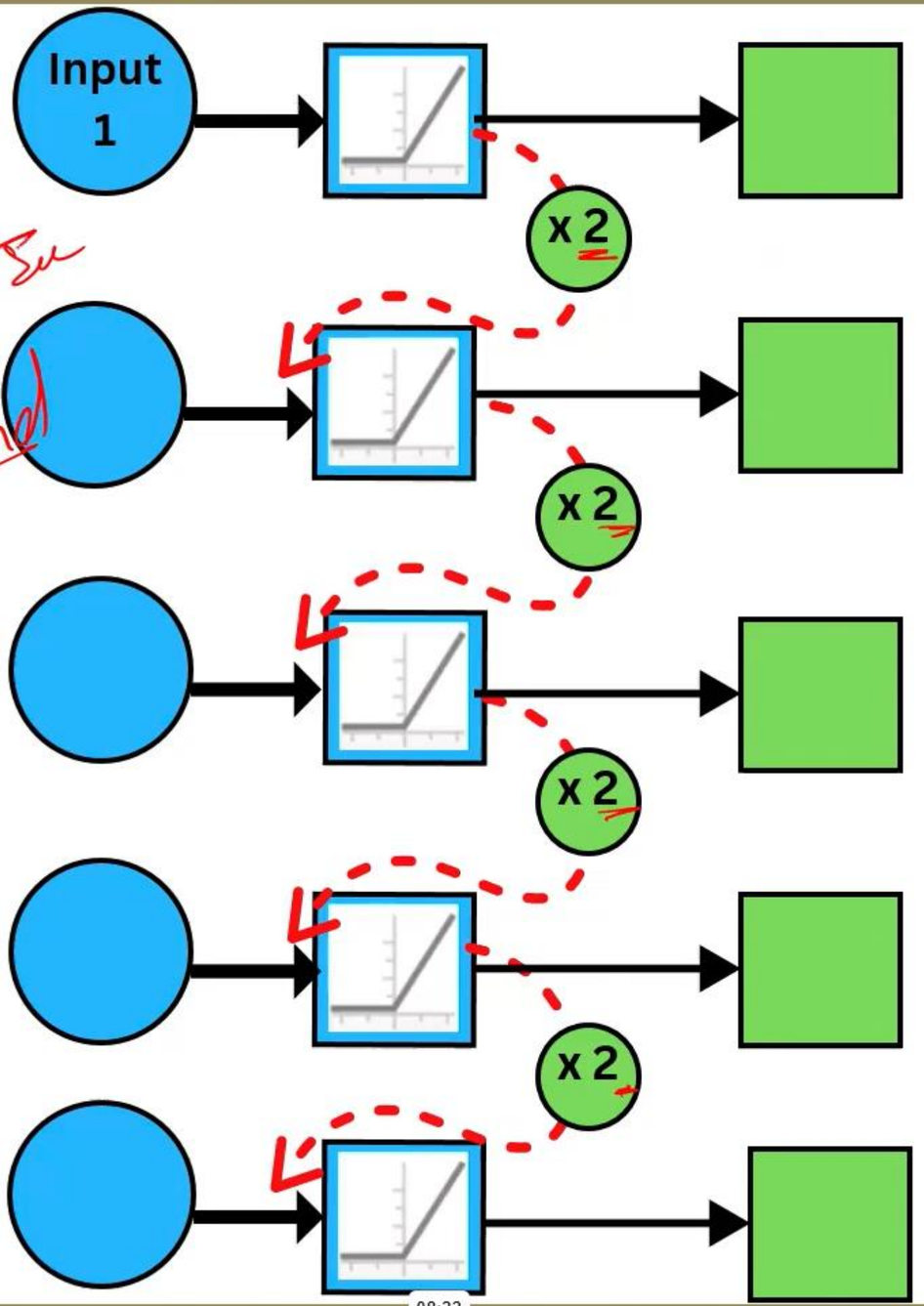


Exploding

2 to the 50th Power is a Huge Number

= Input $_1$ x 2^{50}

= Input $_1$ x A Huge Number



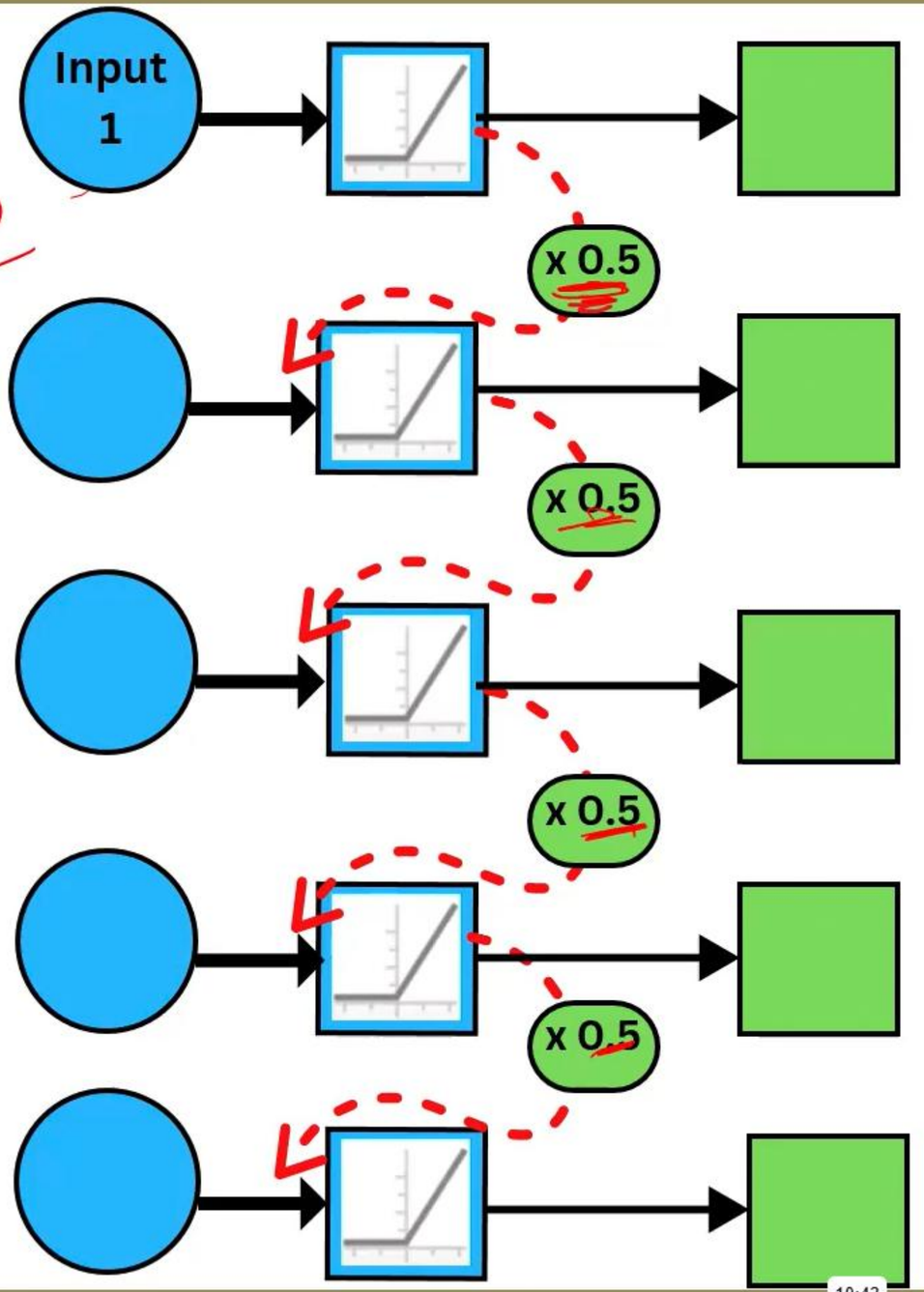
Long Short-Term Memory- (LSTM)

Vanishing

This Number Super Close to 0 would Cause the Gradient, Which We need for Gradient Descent

= Input 1 x 0.5

= Input 1 x A Number Super Close to 0



🔹 1. ReLU (Rectified Linear Unit) — *Most Popular*

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

✅ Used in **CNN, Deep Neural Networks**

✅ Fast & simple

❌ Problem: "Dying ReLU" for negative values

📌 **Use case:** Hidden layers in almost all modern models

🔹 2. Sigmoid

$$f(x) = \frac{1}{1 + e^{-x}}$$

✅ Output between **0 and 1**

✅ Good for **binary classification**

❌ Vanishing gradient problem

📌 **Use case:** Output layer for **binary classif ion**

🔹 3. Tanh (Hyperbolic Tangent)

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

✅ Output between **-1 and +1**

✅ Zero-centered

❌ Still suffers from vanishing gradients

📌 **Use case:** RNN, LSTM, NLP models

tanh – Positive Value

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

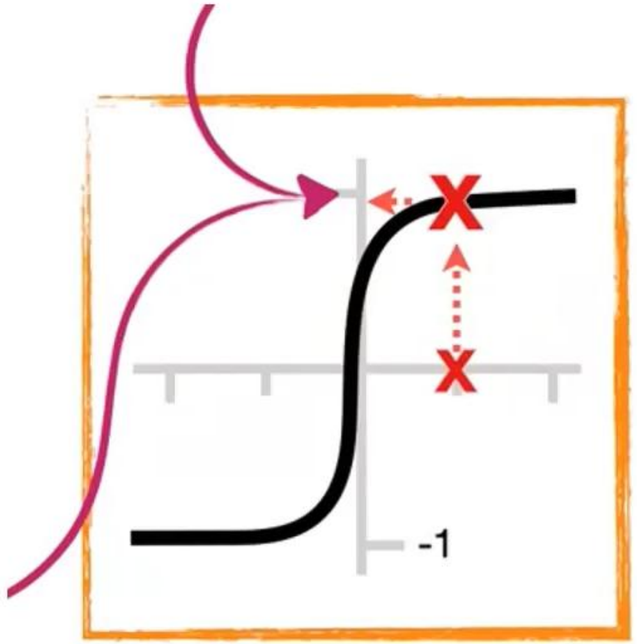
$$f(2)=\frac{e^{2}-e^{-2}}{e^{2}+e^{-2}}$$

$$=0.96$$

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

$$f(-5)=\frac{e^{-5}-e^{5}}{e^{-5}+e^{5}}$$

$$=-1$$



🔹 3. Tanh (Hyperbolic Tangent)

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

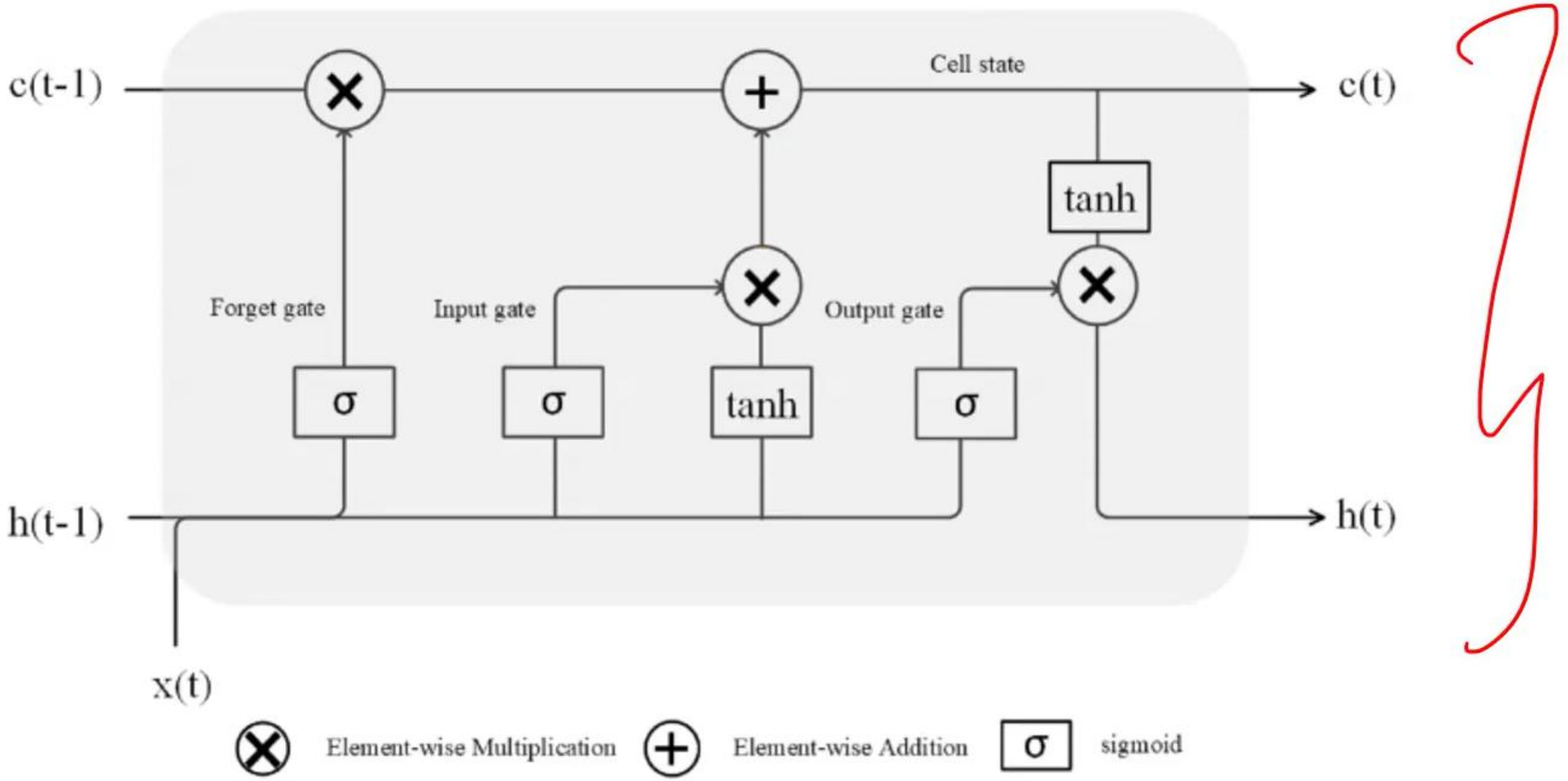
✅ Output between -1 and +1

✅ Zero-centered

❌ Still suffers from vanishing gradients

📌 **Use case:** RNN, LSTM, NLP models

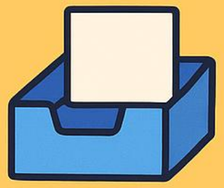
Long Short-Term Memory- (LSTM)-Case -1-Positive Value



LSTM MEMORY BOX

1. Forget Old Stuff

"Throw away things I don't need anymore!"



2. Add New Stuff

"Learn something new!"

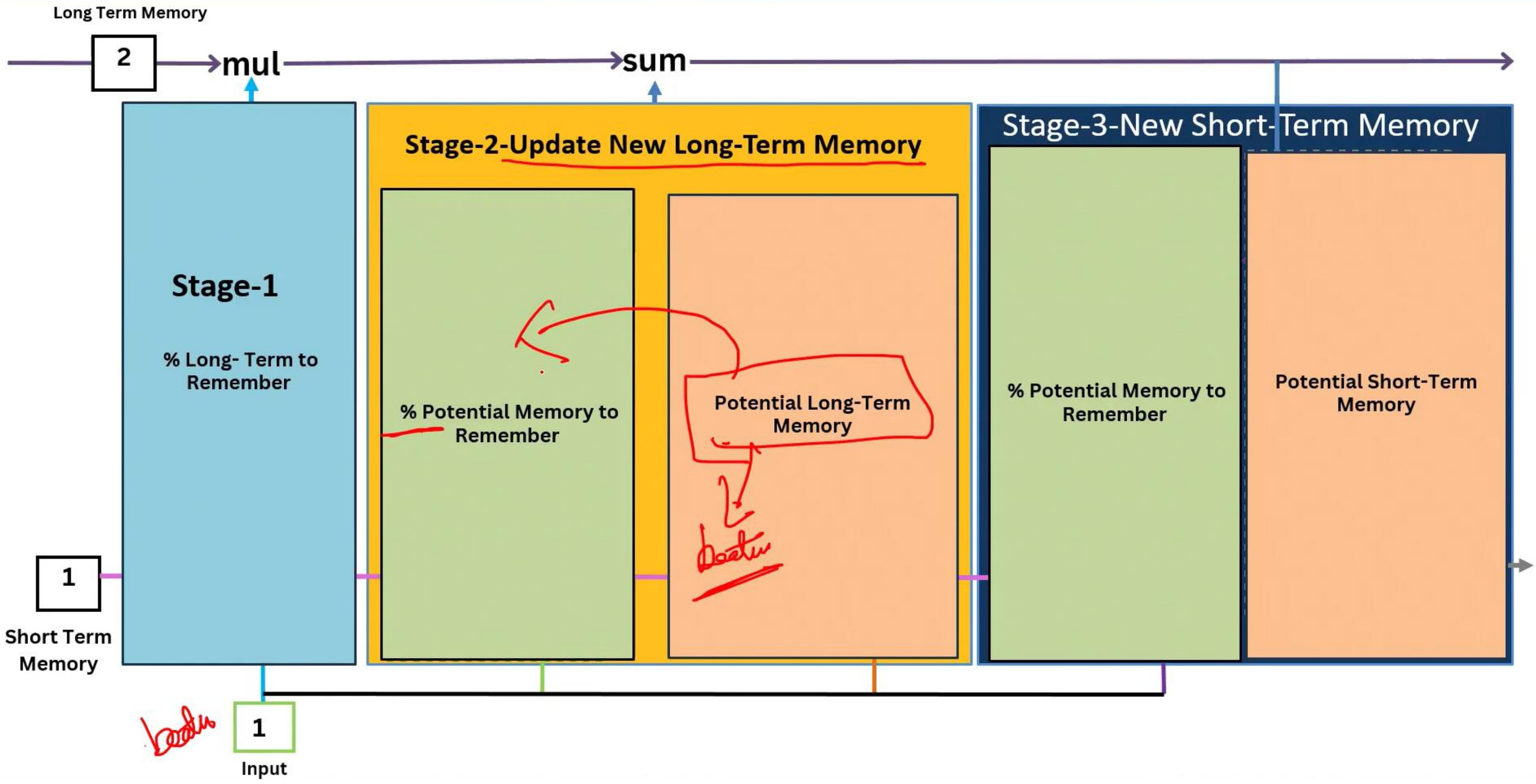
3. Mix Old + New

"Update my brain!"

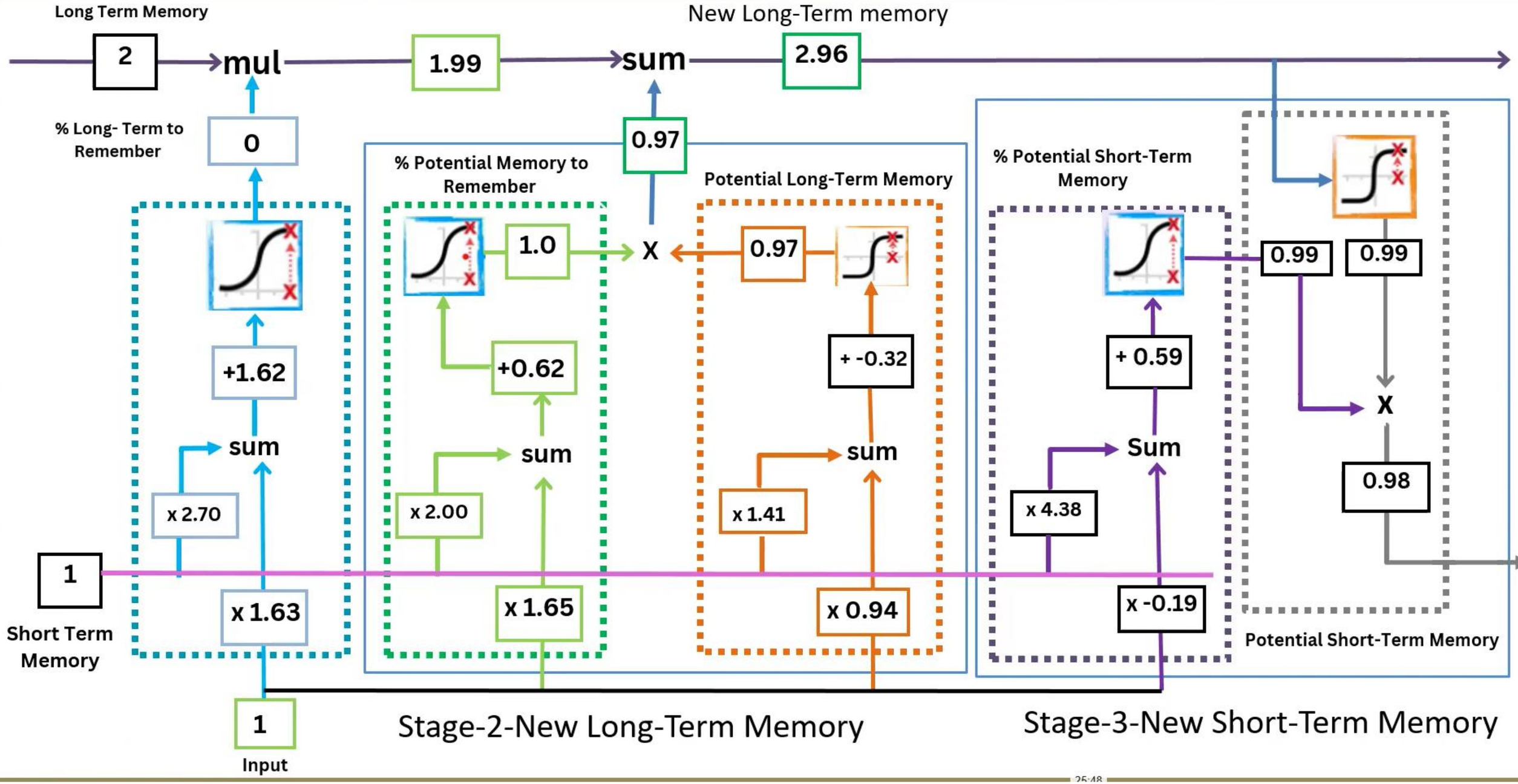
4. Output Answer

"Say something useful now!"

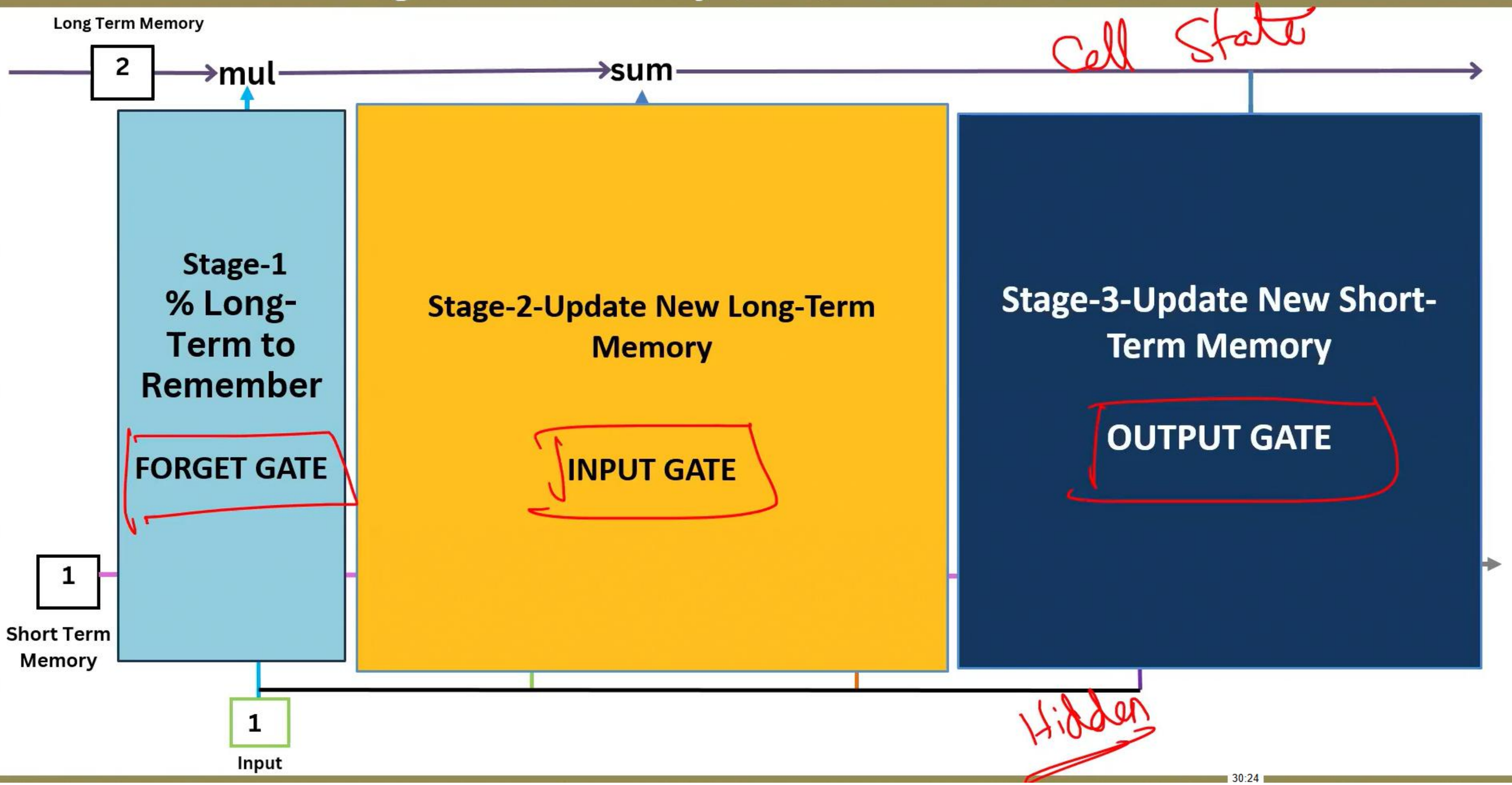
Long Short-Term Memory- (LSTM)-Case -1-Positive Value



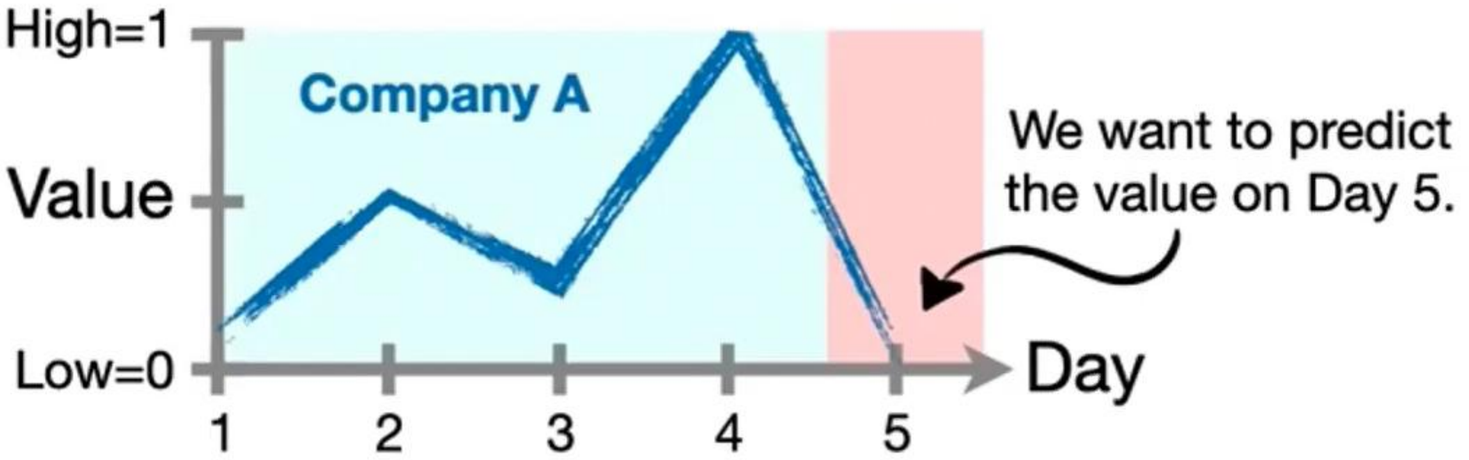
Long Short-Term Memory- (LSTM)-Case -1-Positive Value

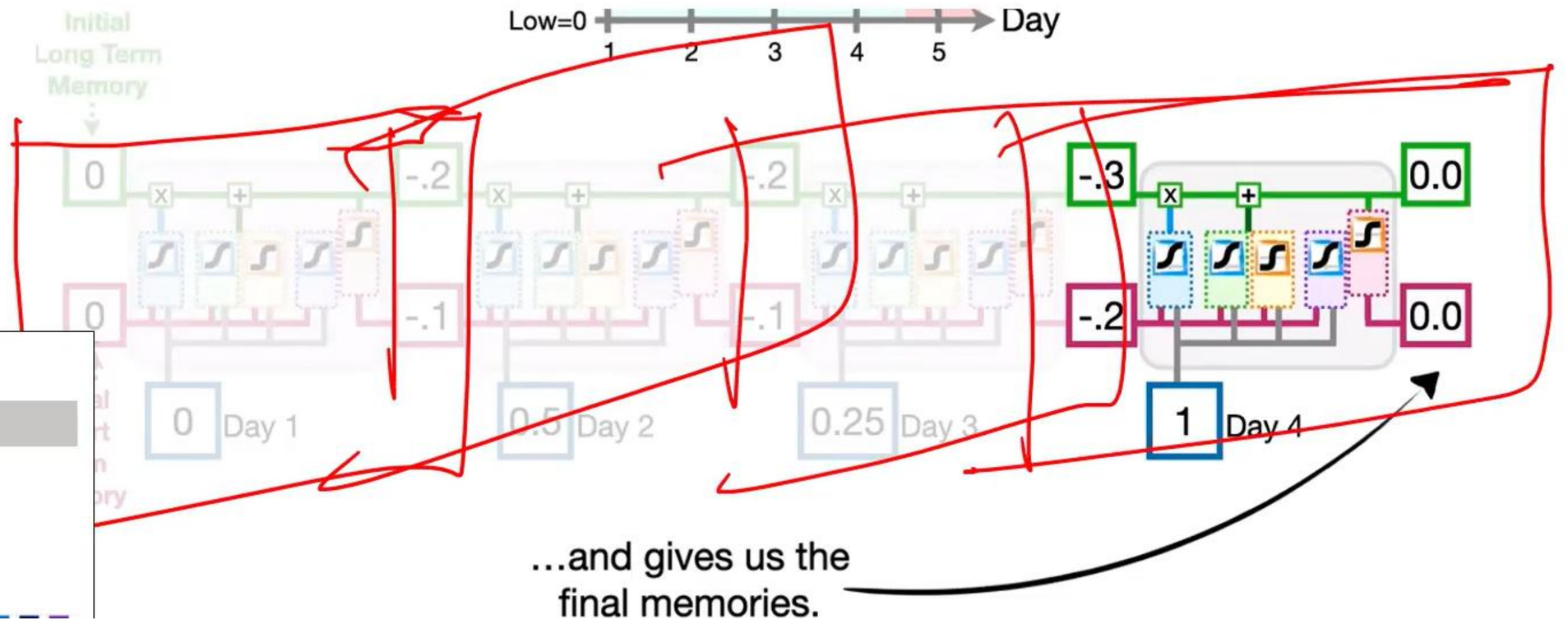


Long Short-Term Memory- (LSTM)-Case -1-Positive Value

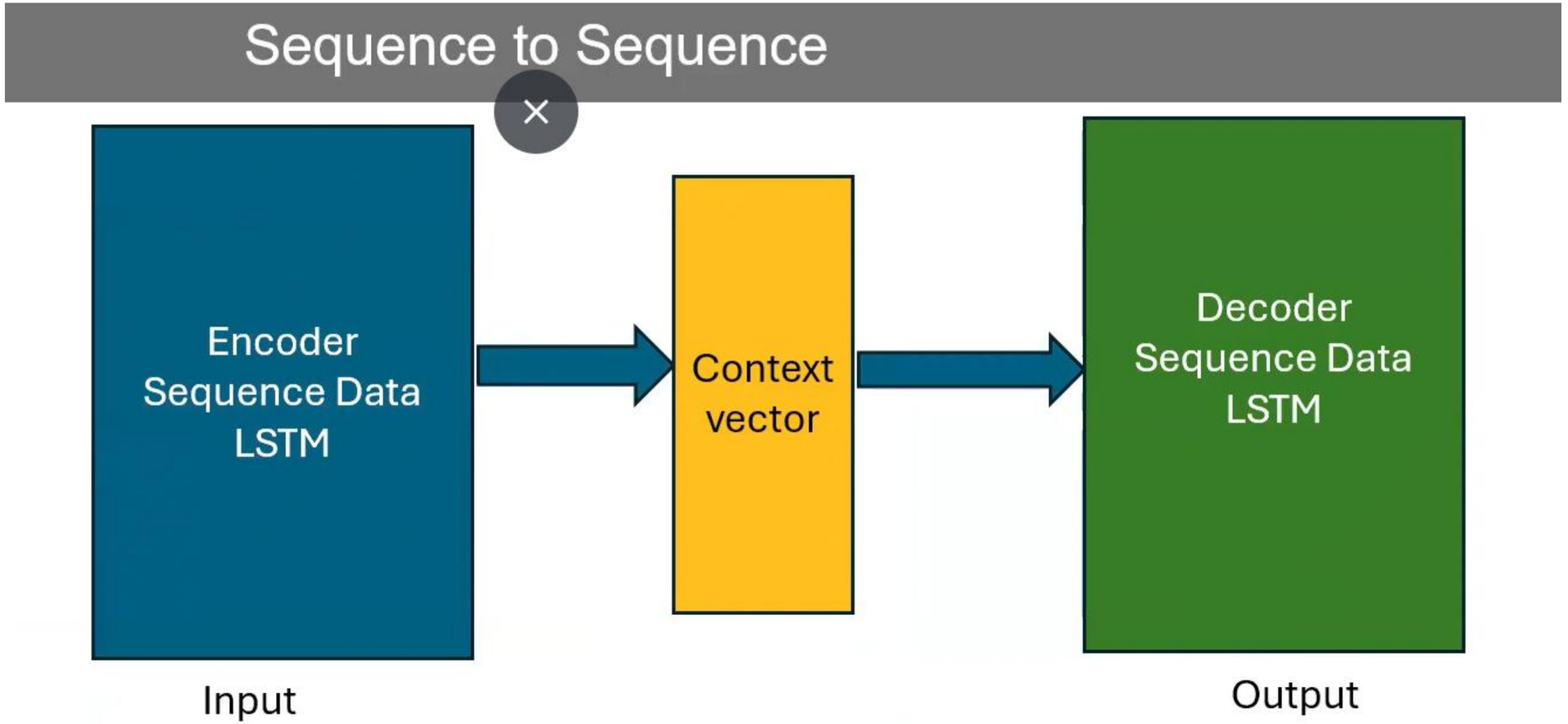


Long Short-Term Memory- (LSTM)

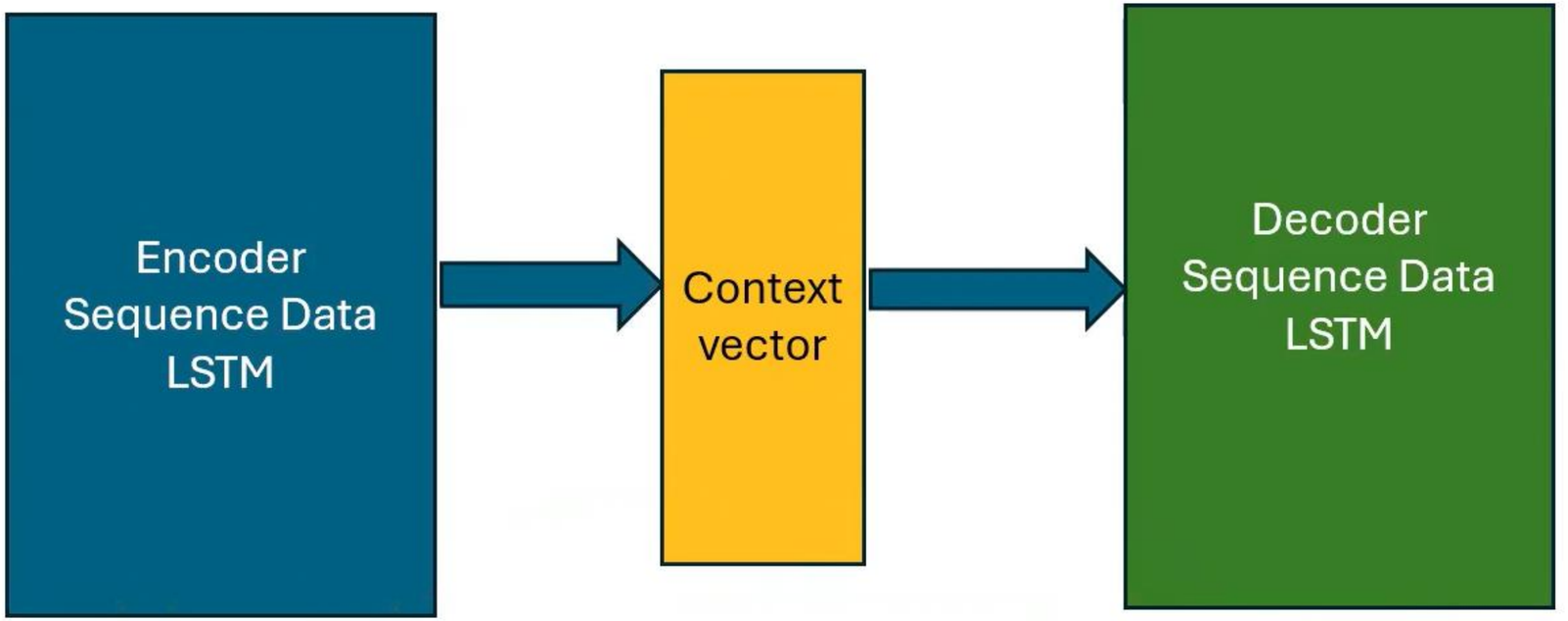




er Pointer
hlighter
ser
se All Ink on Slide


Sequence to Sequence
Encoder
Sequence Data
LSTM
Context
vector
Decoder
Sequence Data
LSTM
Input
Output

Sequence to Sequence



	Input	Output
Machine Translation	English	Tamil
Text Summarization	Paragraphs	Summarization
Question Answering	Questions	Answer

Tamil Cinema Famous Dialog Completion

input	Output
நா ஒரு தடவை சொன்னா	நூறு தடவா சொன்ன மாறி
ஒரு வாட்டி முடிவு பண்ணிட்டா	என் பேச்சை நானே கேட்கமாட்டேன்

Sequence to Sequence

உமி	எள்	கலி	காலை
0.2	0.0	0.3	0.6

- - - - Final word

நா	ஒரு	தடவை	சொன்னா
0.2	0.4	0.1	0.6
0.3	0.5	0.6	0.9
0.1	0.4	0.1	0.8
0.2	0.4	0.1	0.6
0.3	0.5	0.1	0.7

Word Embeddings

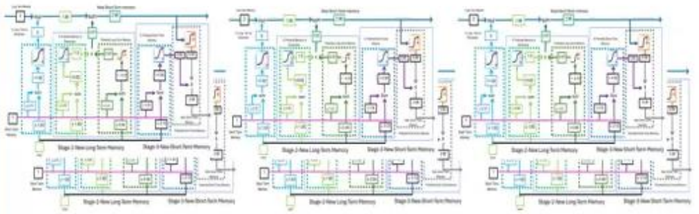
5 Features for Single

word- Example

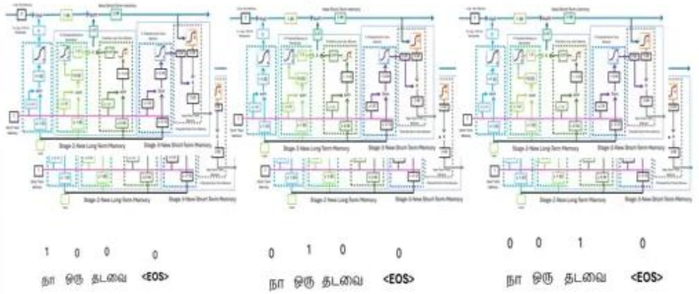
Sequence to Sequence

Encoder

Layer-2



Layer-1

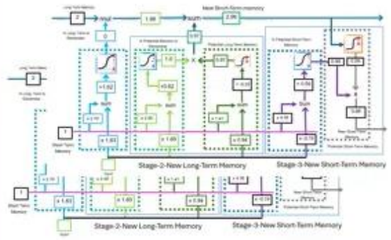


Context Vectors

Decoder

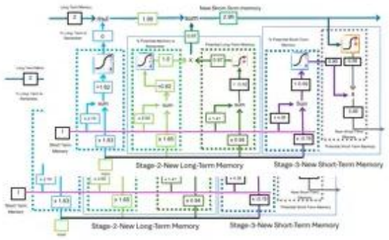
<SOS> நூறு தடவா சொன்ன
0 1 0 0

Softmax



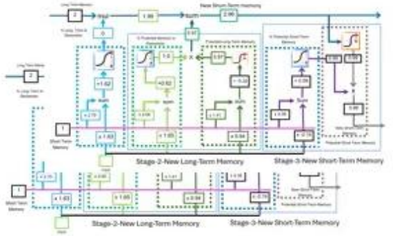
<SOS> நூறு தடவா சொன்ன
0 0 1 0

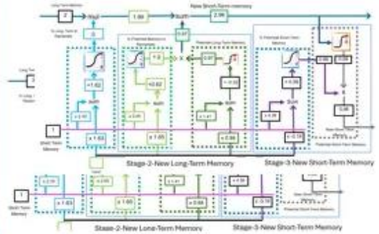
Softmax



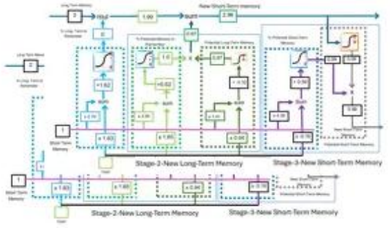
<SOS> நூறு தடவா சொன்ன
0 0 0 1

Softmax

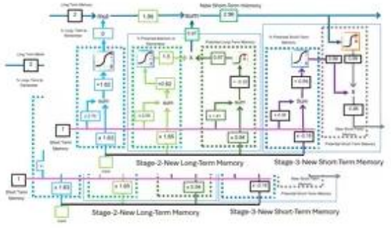




<SOS> நூறு தடவா சொன்ன
1 0 0 0

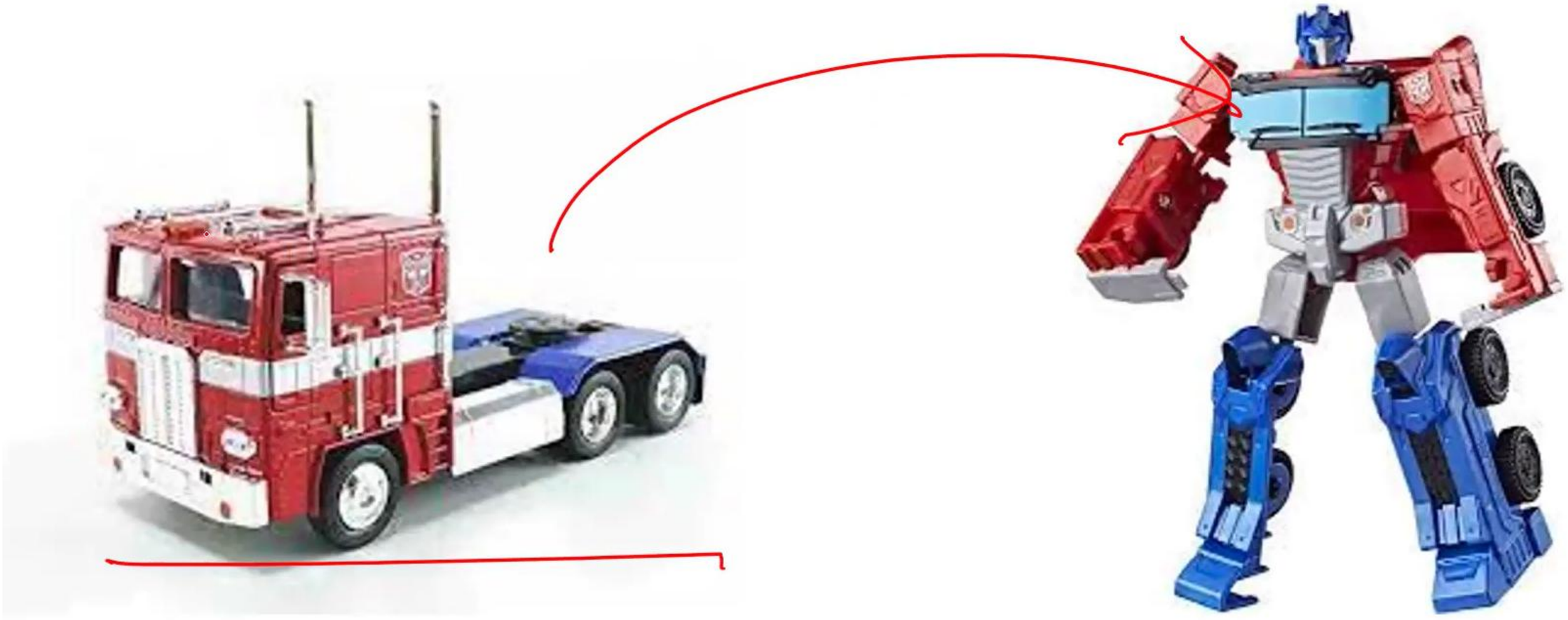


<SOS> நூறு தடவா சொன்ன
0 1 0 0



<SOS> நூறு தடவா சொன்ன
0 0 1 0

Why Transformer?



You

Complete the sentence " You are beautiful.... "

ChatGPT

"You are beautiful just the way you are, inside and out."

You

write python to handle large number rows

ChatGPT

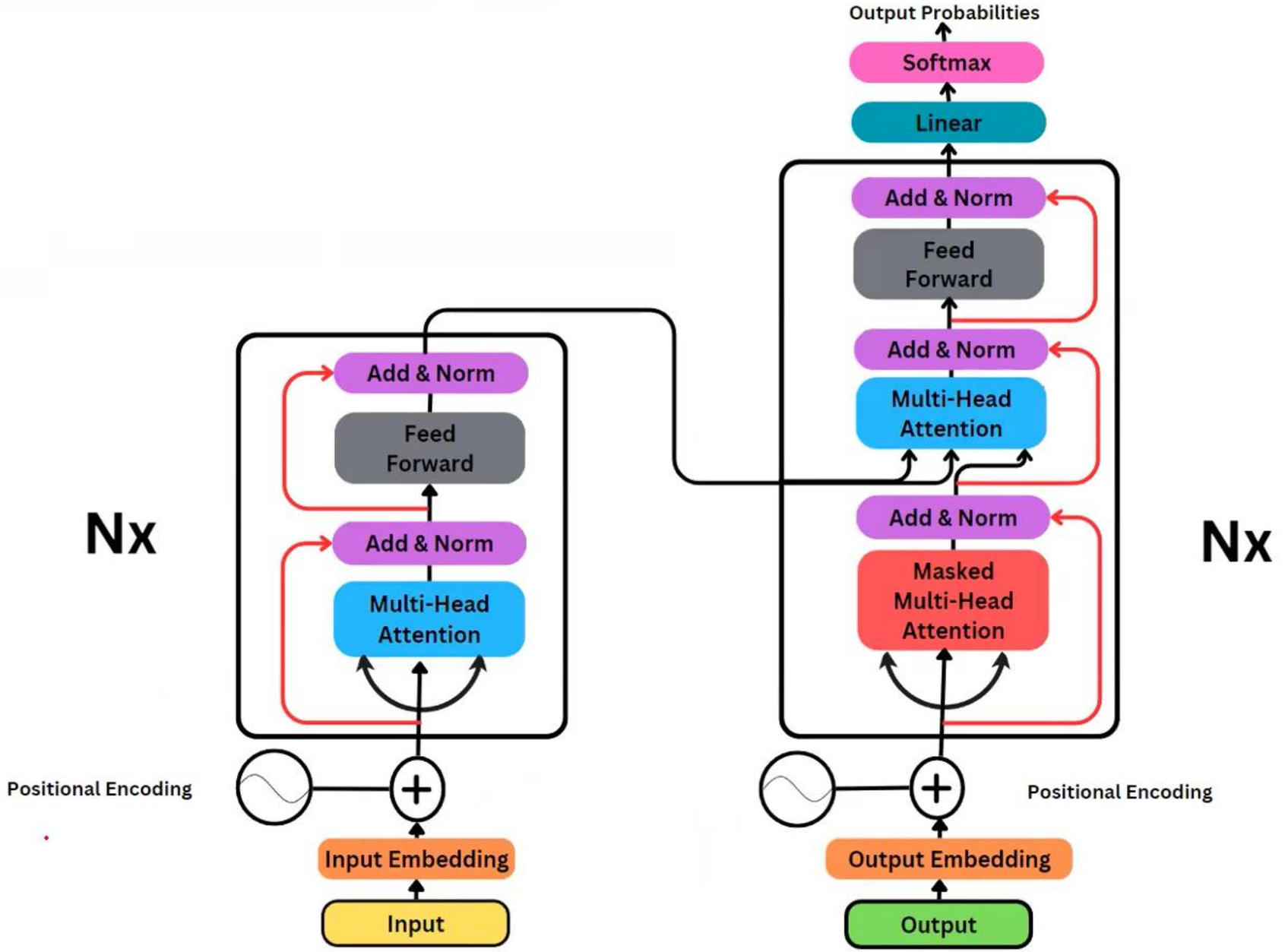
Handling large numbers of rows in Python often involves using efficient libraries like pandas or Dask for data manipulation and processing. Here's a basic example using pandas:

python Copy code

```python
import pandas as pd

# Assuming you have a large CSV file named 'data.csv'
# Load the data into a pandas DataFrame in chunks
chunk_size = 100000  # Adjust the chunk size as needed
data_chunks = pd.read_csv('data.csv', chunksize=chunk_size)
```

Why Transformer?



Why Transformer?

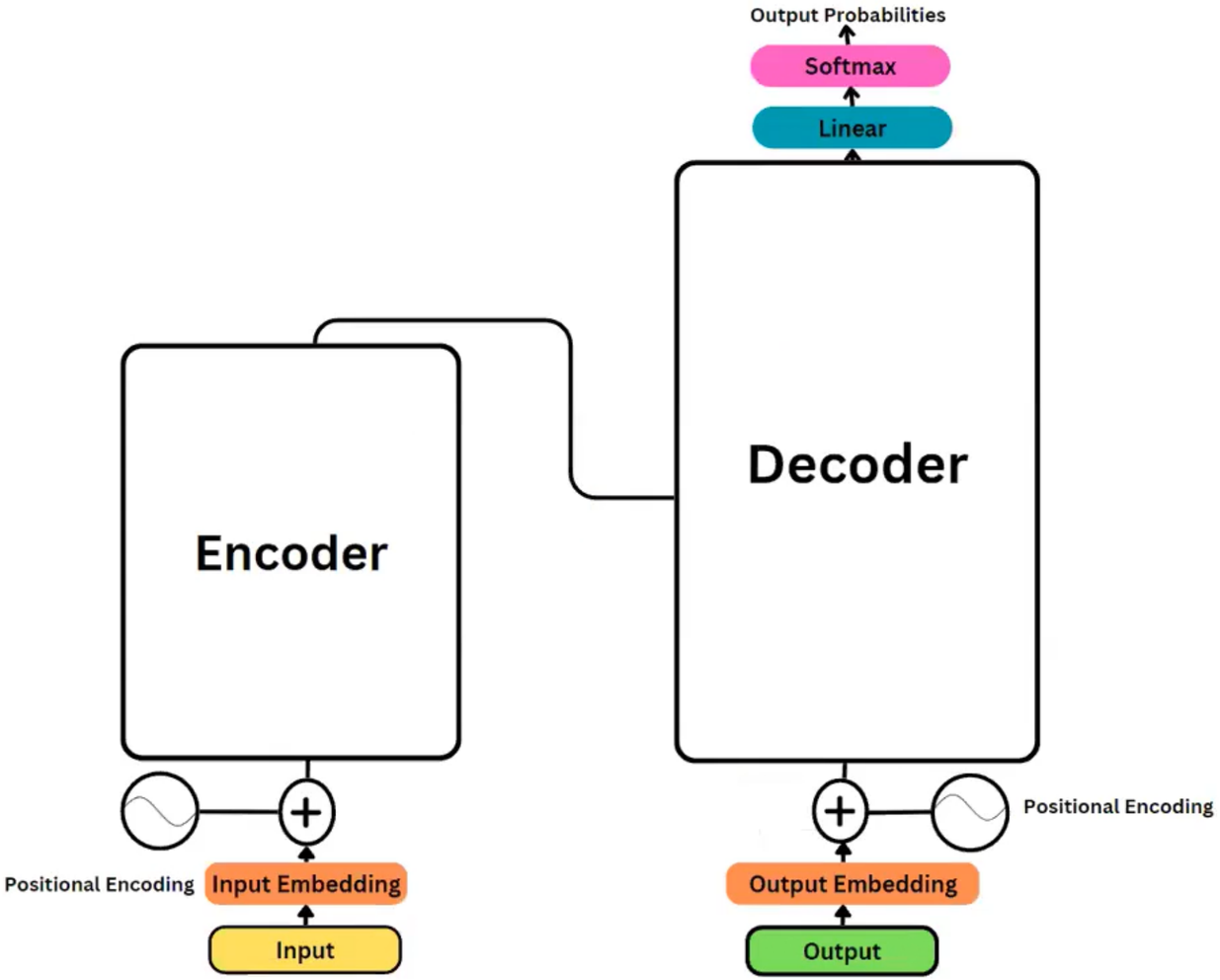
Supervised Learning

Unsupervised Learning

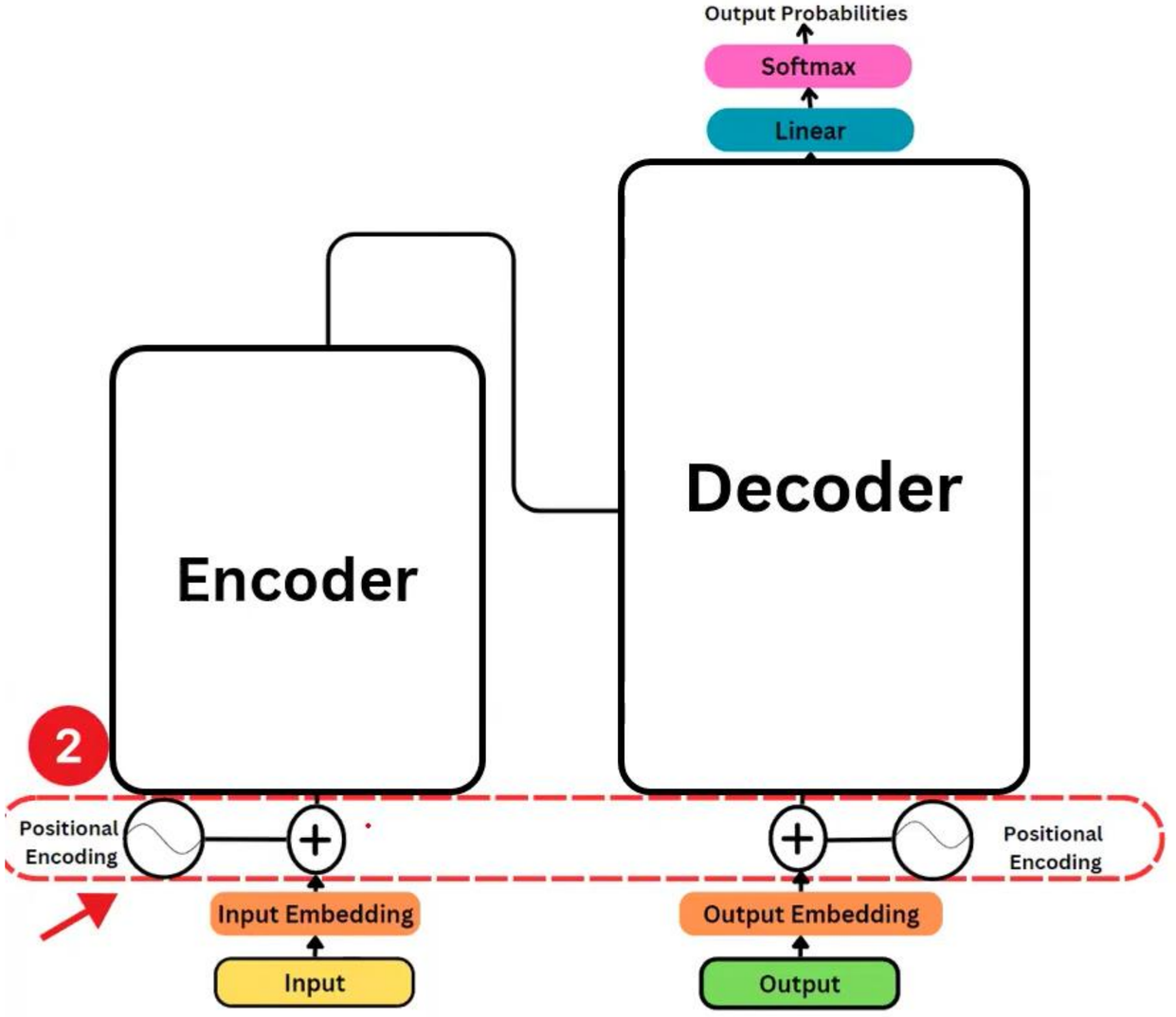
Tamil Cinema Famous Dialog Completion

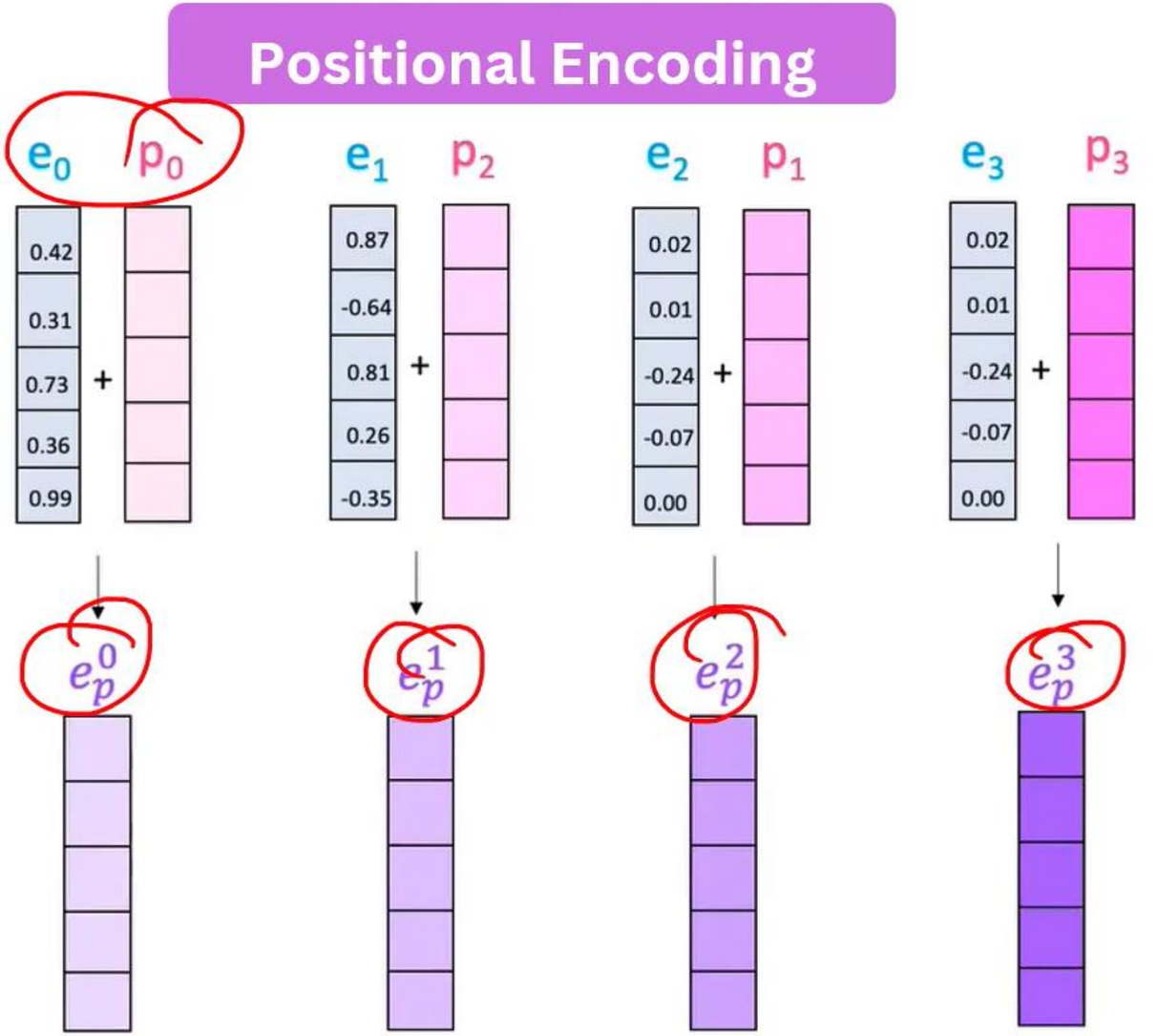
input	Output
நா ஒரு தடவை சொன்னா	நூறு தடவா சொன்ன மாறி
ஒரு வாட்டி முடிவு பண்ணிட்டா	என் பேச்சை நானே கேட்கமாட்டேன்

Why Transformer?

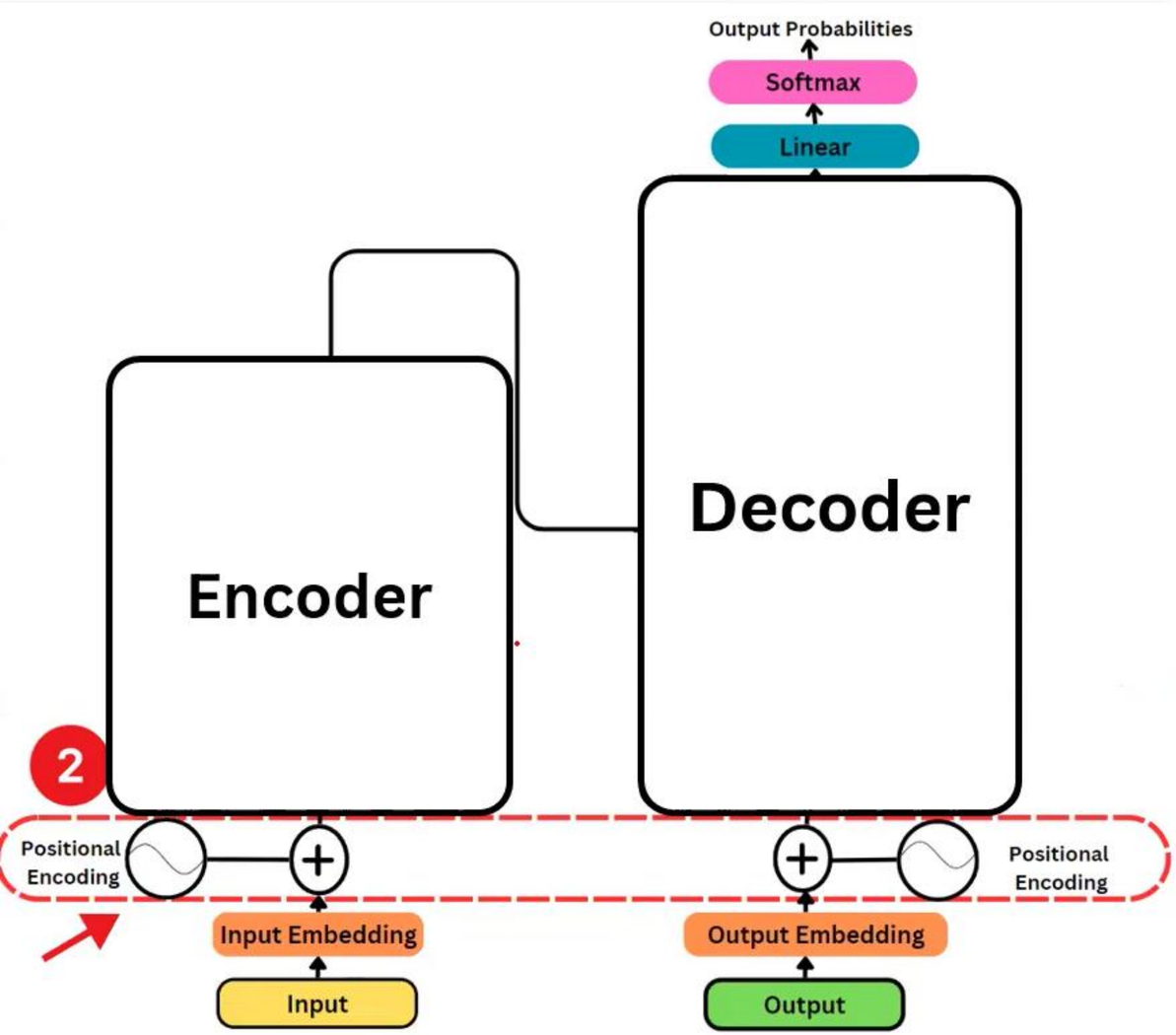


Input	Output
நா ஒரு தடவை சொன்னா	நூறு தடவா சொன்ன மாறி
ஒரு வாட்டி முடிவு பண்ணிட்டா	என் பேச்சை நானே கேட்கமாட்டேன்


Output Probabilities
Softmax
Linear
Encoder
Decoder
2
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Input
Output


Positional Encoding
e0 p0
e1 p2
e2 p1
e3 p3
0.42
0.31
0.73
0.36
0.99
0.87
-0.64
0.81
0.26
-0.35
0.02
0.01
-0.24
-0.07
0.00
0.02
0.01
-0.24
-0.07
0.00
+
e_p^0
e_p^1
e_p^2
e_p^3

Transformer-Positional Encoding



Positional Encoding -Option 2

Sentence 1

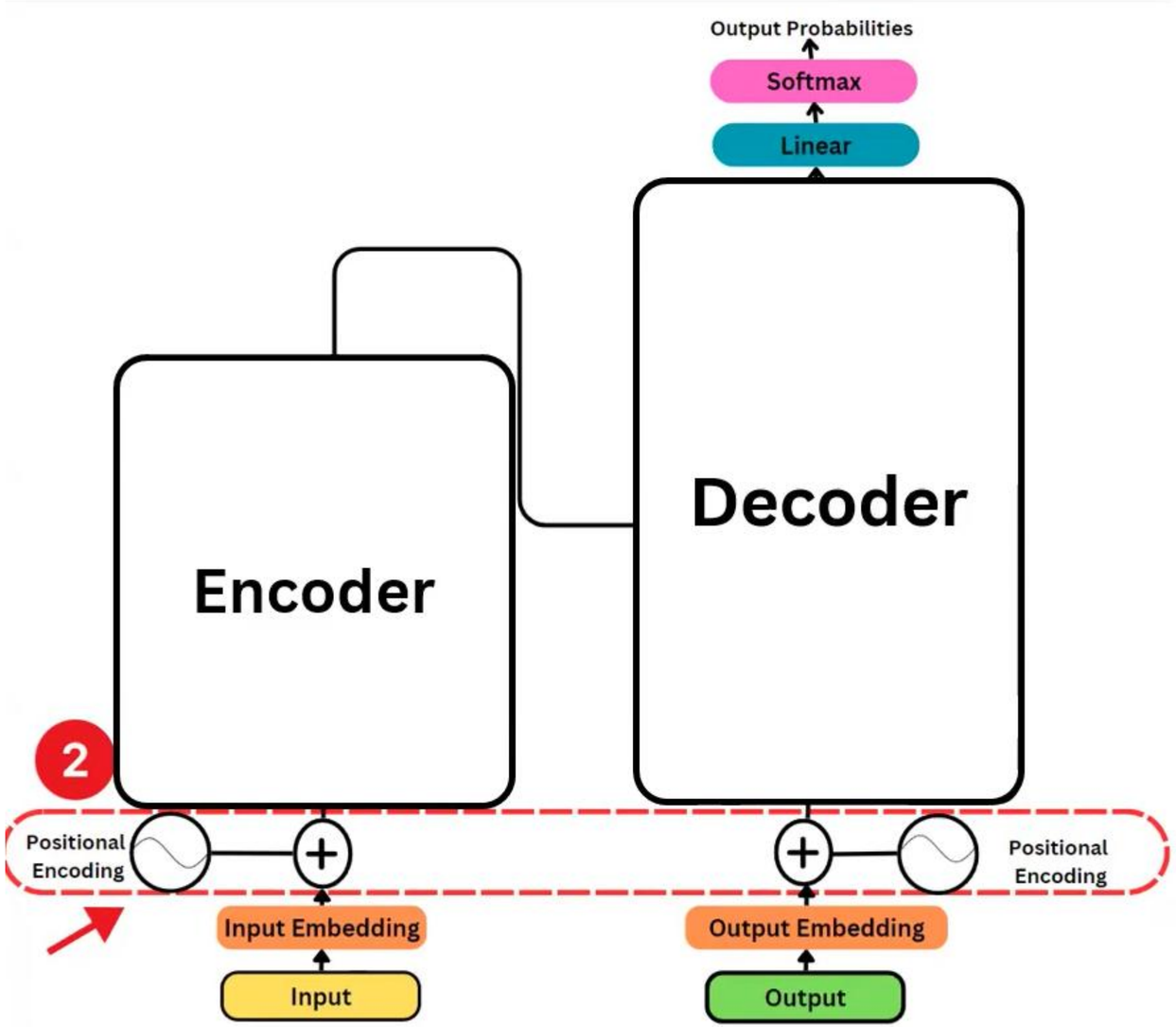
N=4

e_0	p_0	e_1	p_1	e_2	p_2	e_3	p_3
0.42	0	0.87	0.33	0.02	0.66	0.02	1
0.31	0	-0.64	0.33	0.01	0.66	0.01	1
0.73	0	0.81	0.33	-0.24	0.66	-0.24	1
0.36	0	0.26	0.33	-0.07	0.66	-0.07	1
0.99	0	-0.35	0.33	0.00	0.66	0.00	1

Sentence 2

N=5

e_0	p_0	e_1	p_1	e_2	p_2	e_4	p_4
0.42	0	0.87	0.20	0.02	0.40	0.02	1
0.31	0	-0.64	0.20	0.01	0.40	0.01	1
0.73	0	0.81	0.20	-0.24	0.40	-0.24	1
0.36	0	0.26	0.20	-0.07	0.40	-0.07	1
0.99	0	-0.35	0.20	0.00	0.40	0.00	1



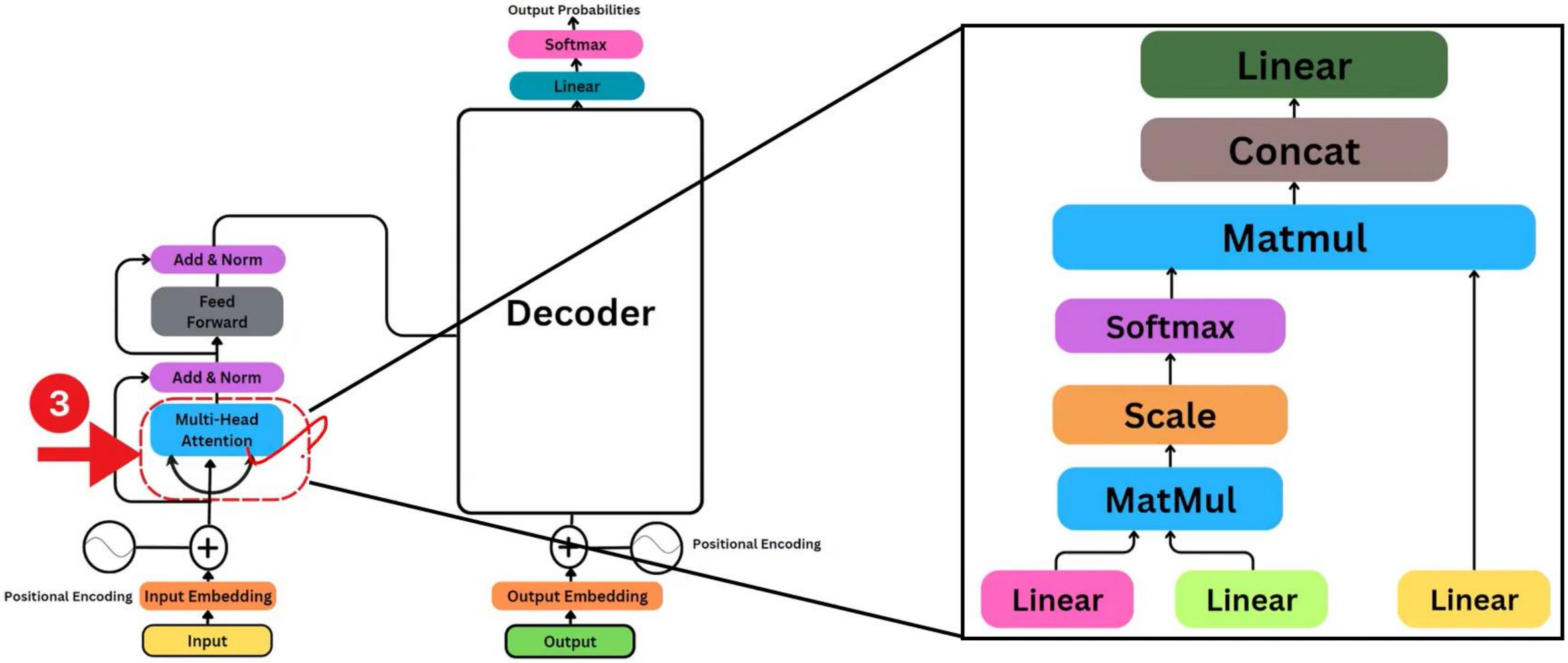
Positional Encoding -Option-3

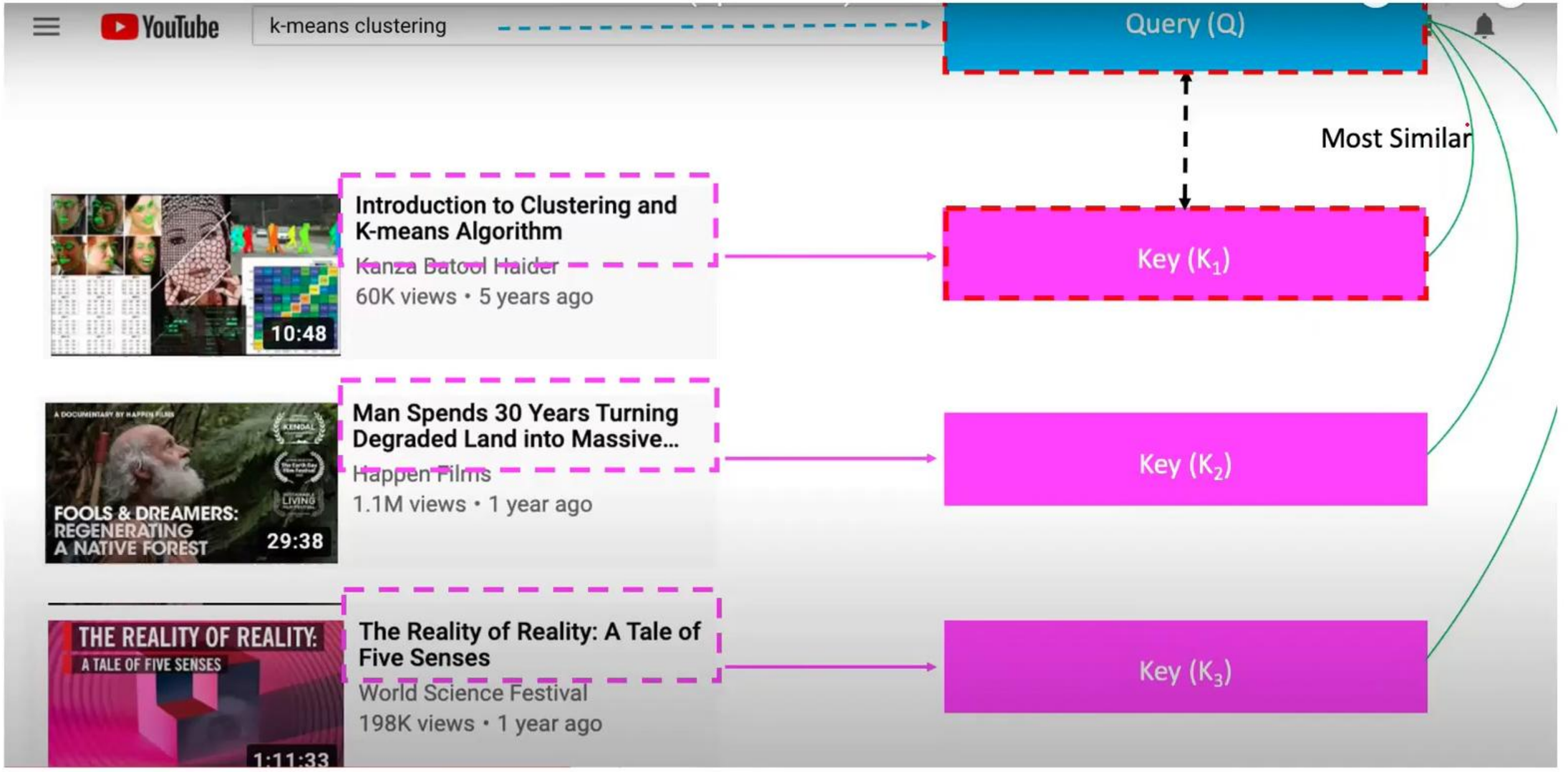
odd

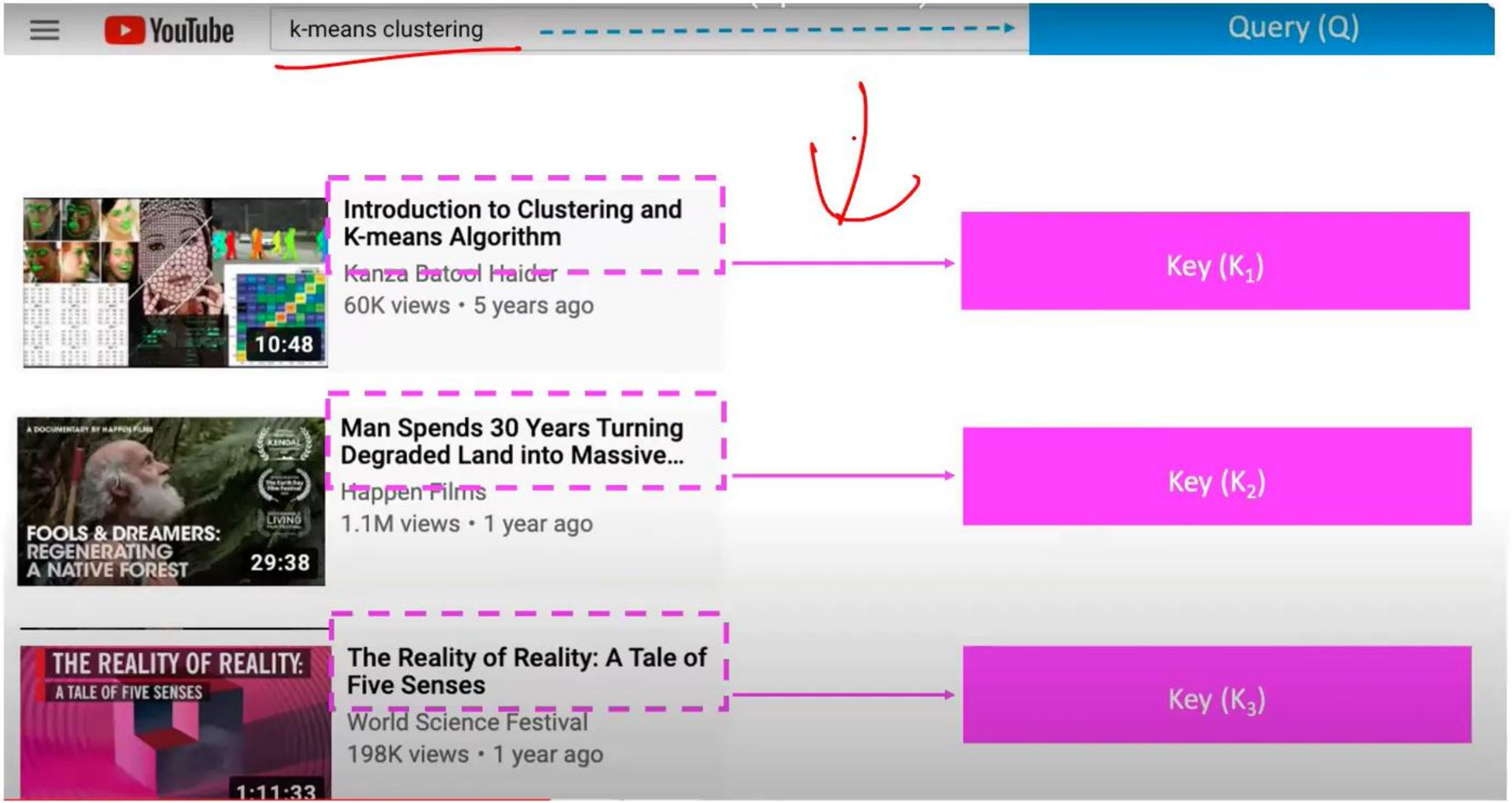
$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right) \quad ?$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d}}}\right) \quad ?$$

even

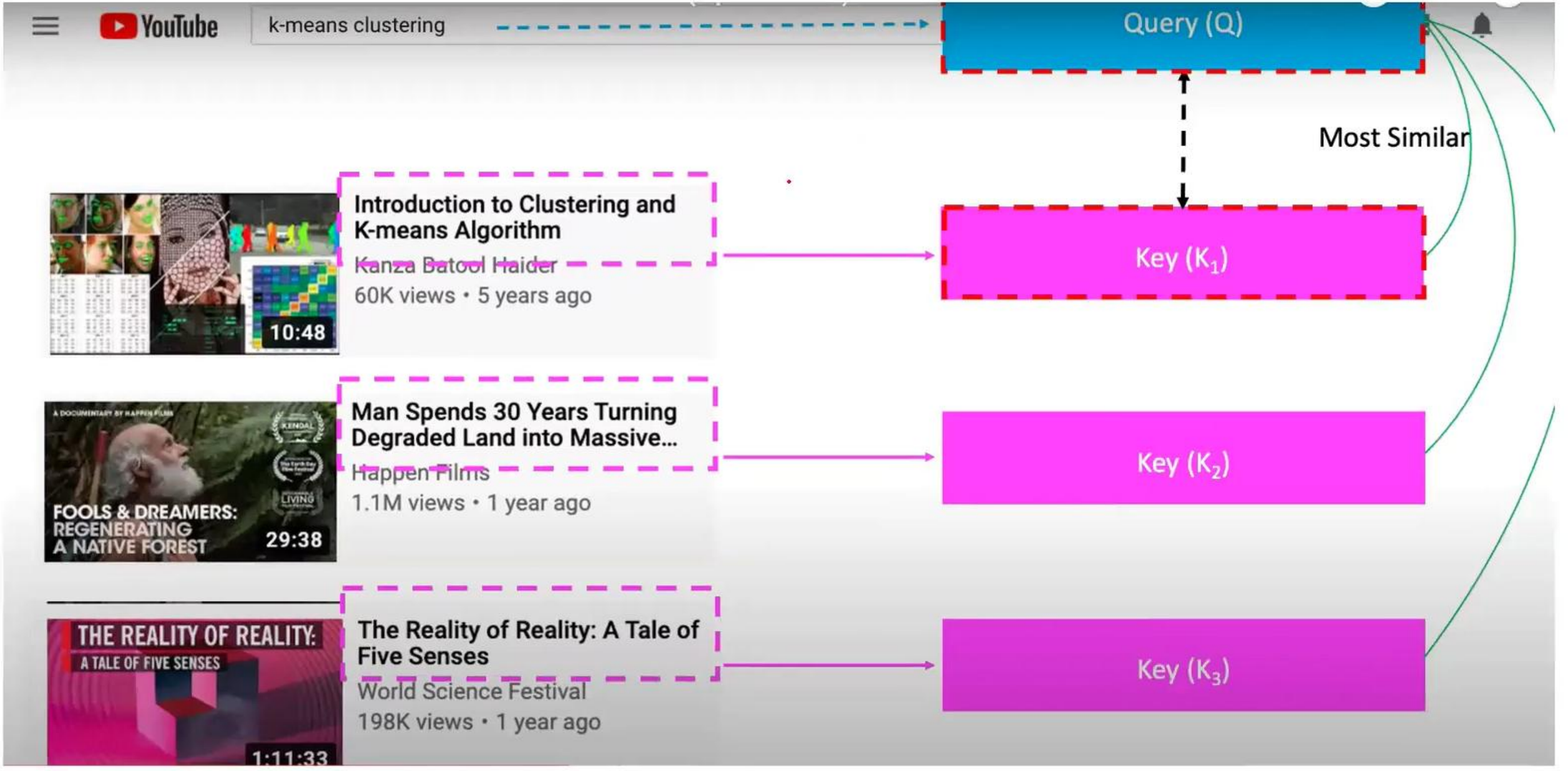

Output Probabilities
Softmax
Linear
Decoder
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
3
Positional Encoding
Input Embedding
Input
Positional Encoding
Output Embedding
Output
Linear
Concat
Matmul
Softmax
Scale
MatMul
Linear
Linear
Linear


YouTube
k-means clustering
Query (Q)
Most Similar
Introduction to Clustering and K-means Algorithm
Kanza Batool Haider
60K views • 5 years ago
10:48
Key (K_1)
Man Spends 30 Years Turning Degraded Land into Massive...
Happen Films
1.1M views • 1 year ago
FOOLS & DREAMERS: REGENERATING A NATIVE FOREST
29:38
Key (K_2)
The Reality of Reality: A Tale of Five Senses
World Science Festival
198K views • 1 year ago
THE REALITY OF REALITY:
A TALE OF FIVE SENSES
1:11:33
Key (K_3)

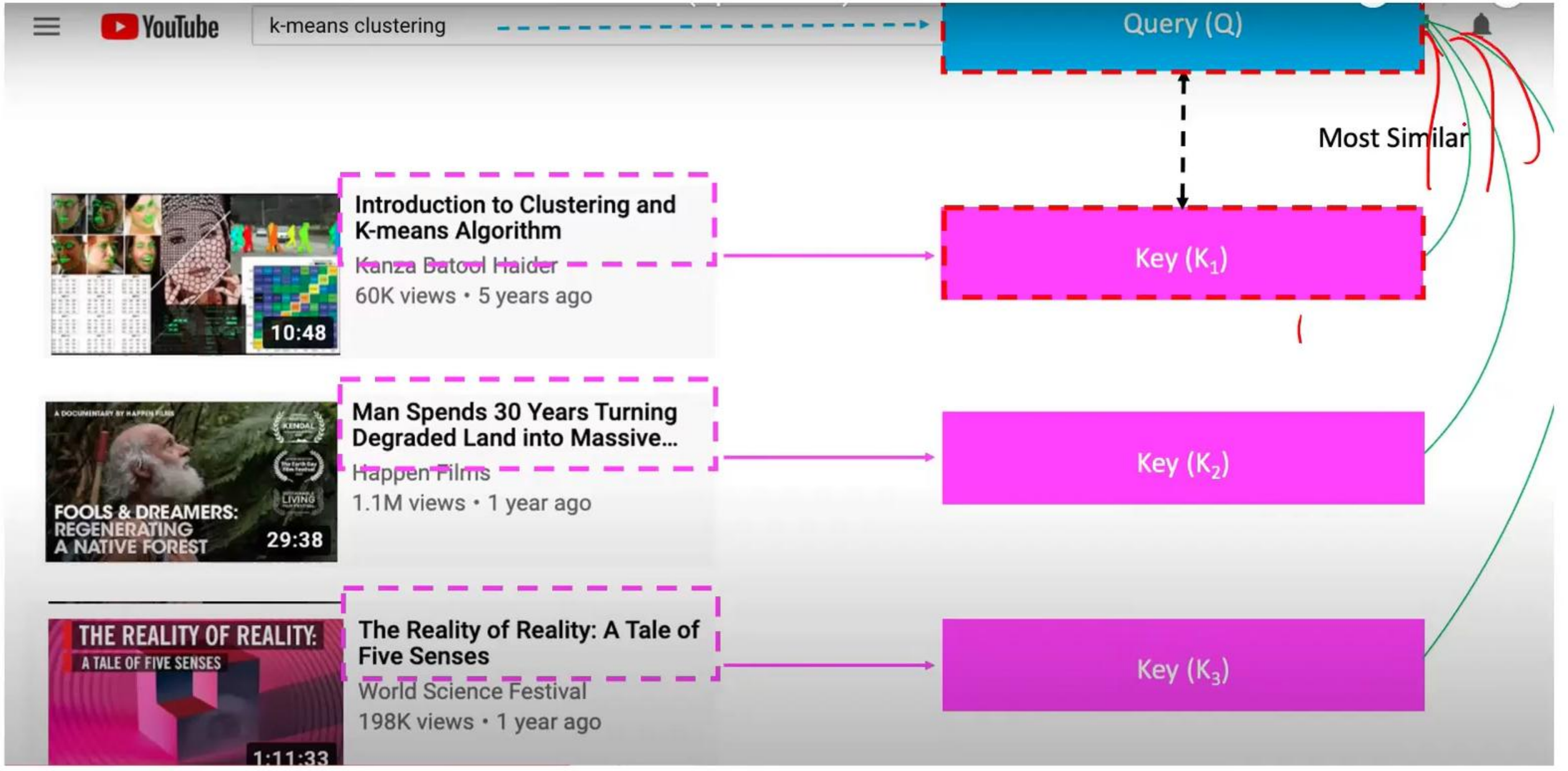

YouTube
k-means clustering
Query (Q)
10:48
Introduction to Clustering and K-means Algorithm
Kanza Batool Haider
60K views • 5 years ago
Key (K_1)
FOOLS & DREAMERS: REGENERATING A NATIVE FOREST
29:38
Man Spends 30 Years Turning Degraded Land into Massive...
Happen Films
1.1M views • 1 year ago
Key (K_2)
THE REALITY OF REALITY:
A TALE OF FIVE SENSES
1:11:33
The Reality of Reality: A Tale of Five Senses
World Science Festival
198K views • 1 year ago
Key (K_3)

Multi Head Attention

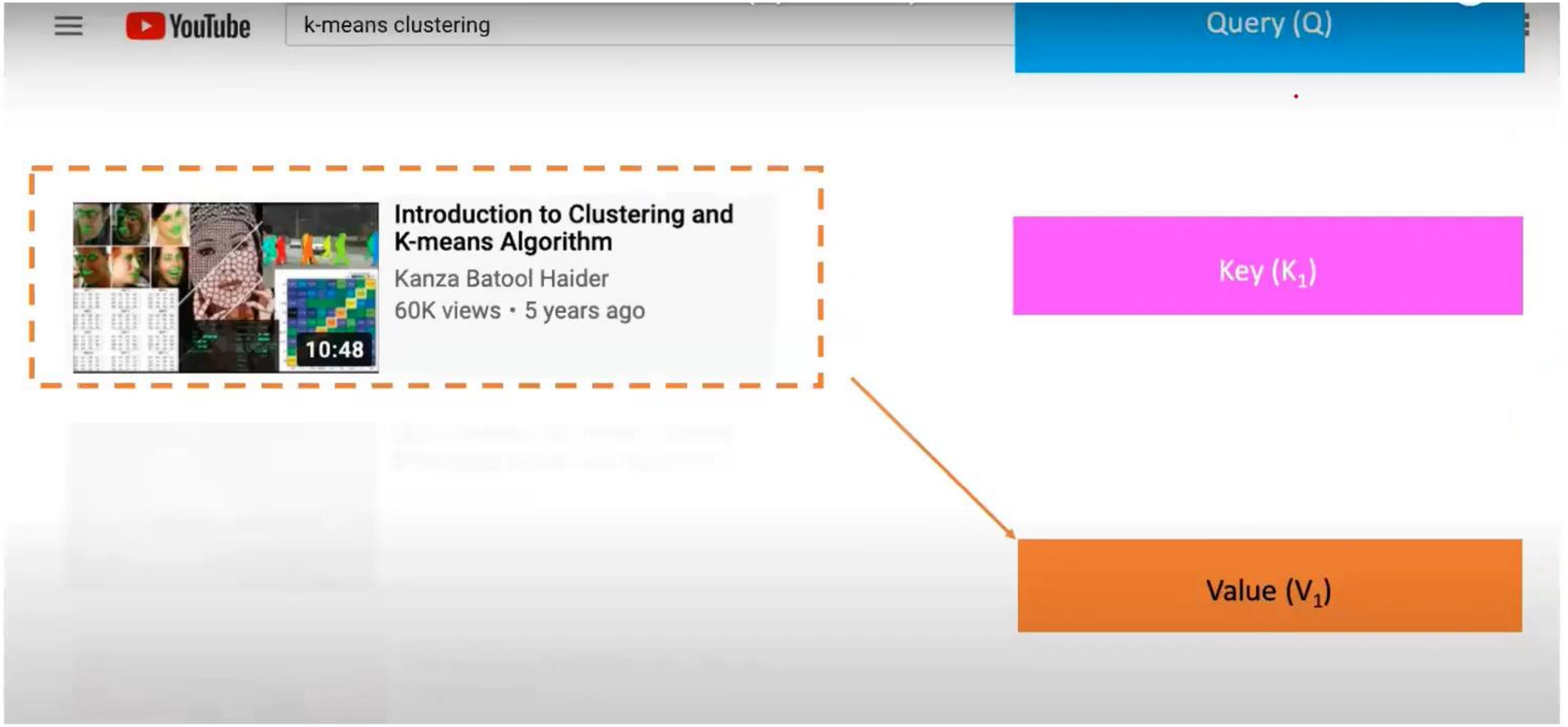
3



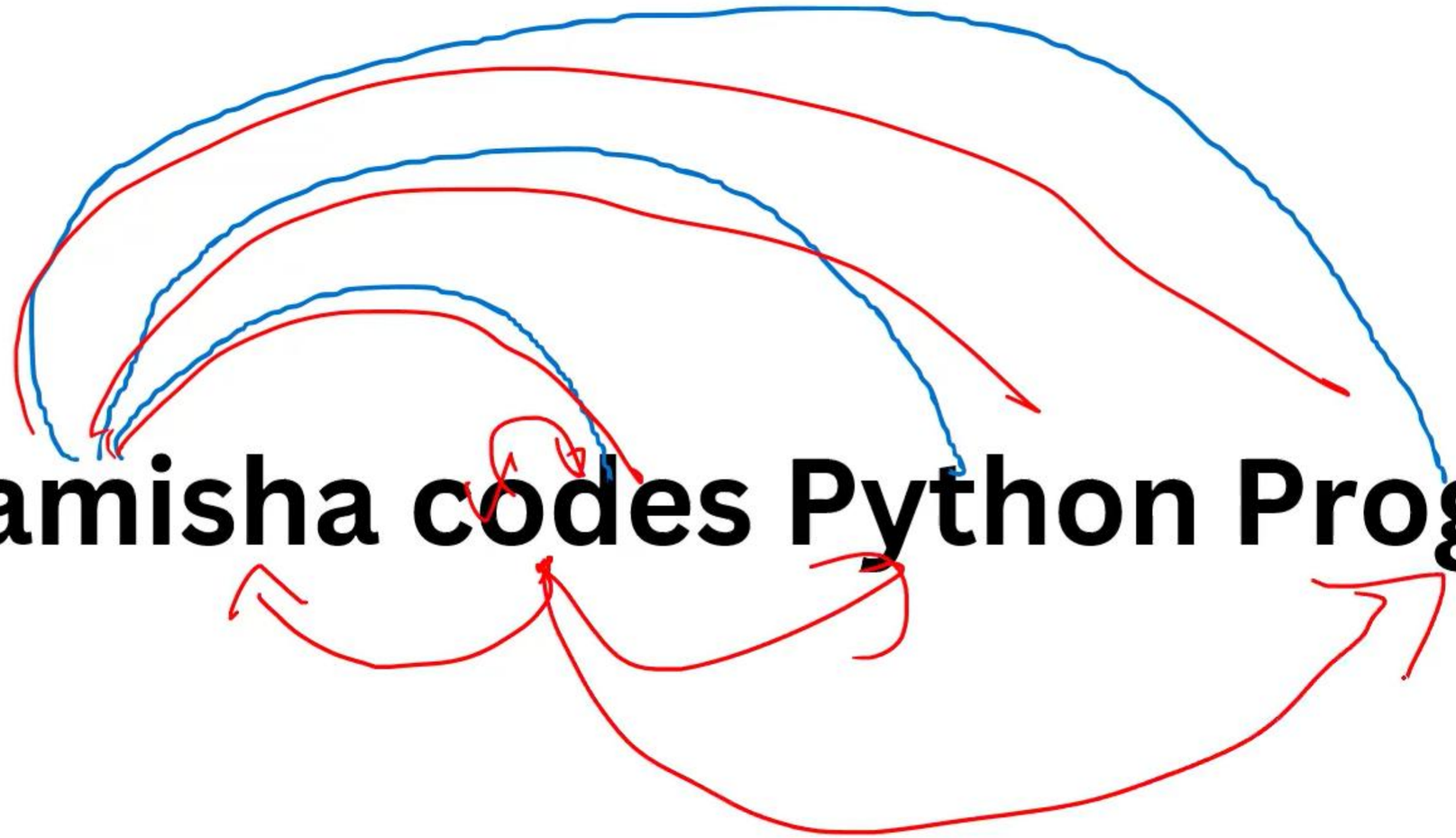
Multi Head Attention



Multi Head Attention



Why Transformer?



Ramisha codes Python Programming

Similarity?

-1 to +1

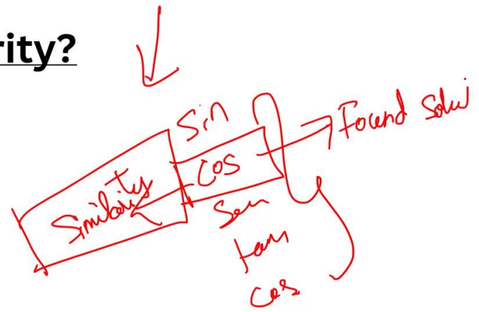
Most Dissimilar Most similar

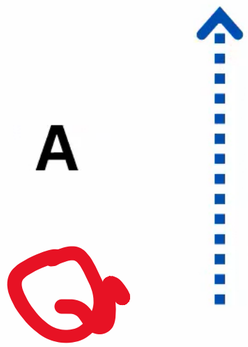
Query (Q)

Key (K1)

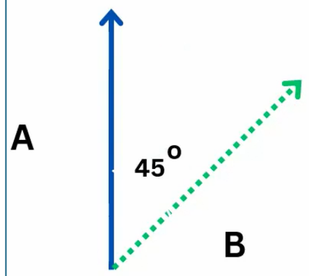
Similarity?

Similarity?



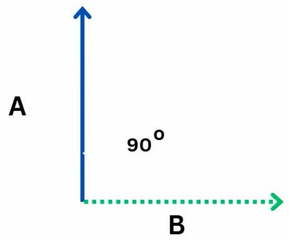


Cos (0) = +1

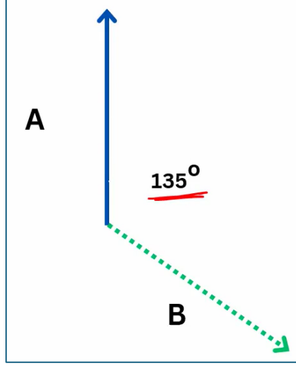


Cos (45) = +0.7

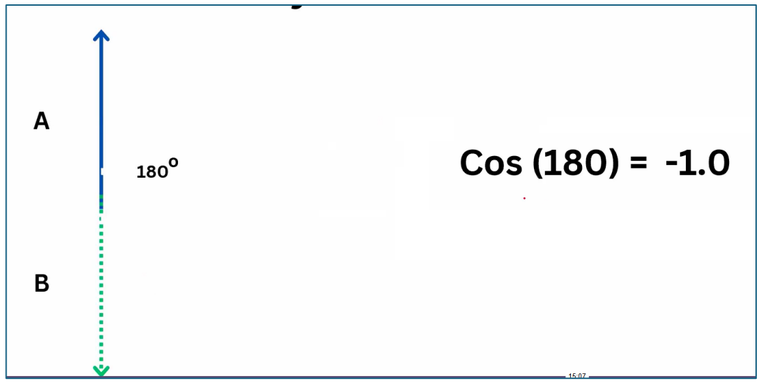
Cosine Similarity



Cos (90) = 0.0



Cos (135) = -0.7



Cos (180) = -1.0

-1 to **+1**

Most Dissimilar **Most similar**

$$\text{Cos}(A,B) = \frac{A \cdot B}{|A|\,|B|}$$

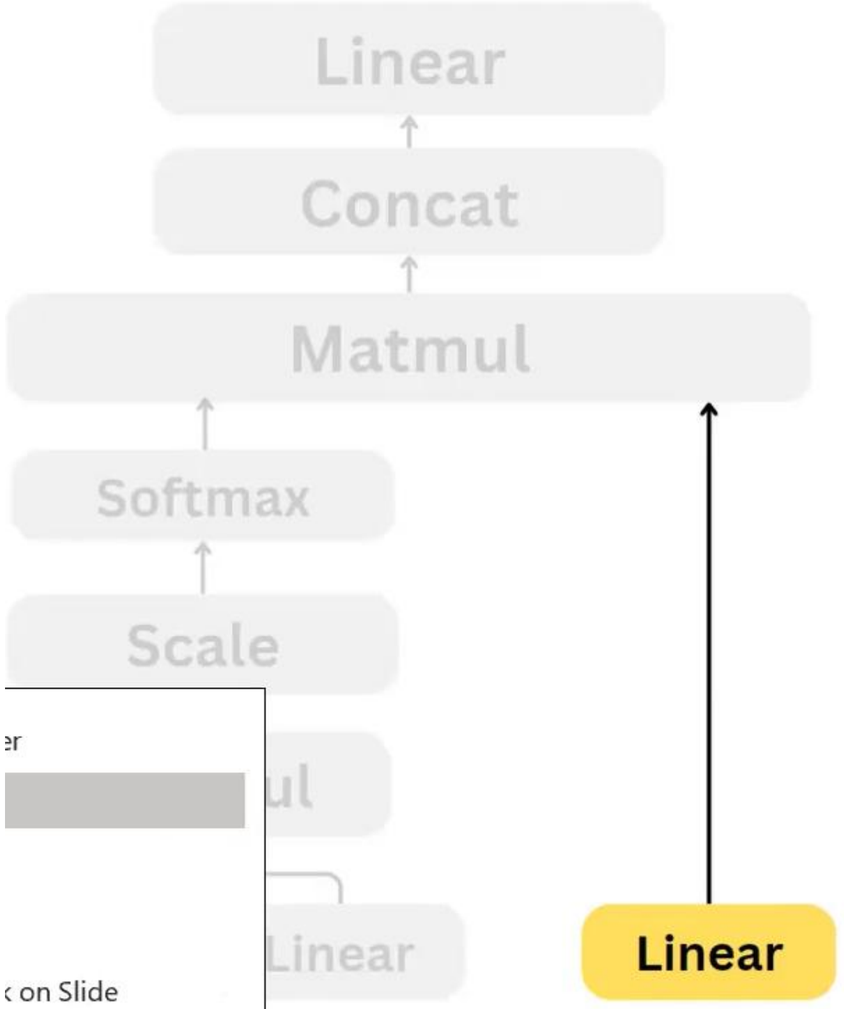
Similarity b/w Vectors

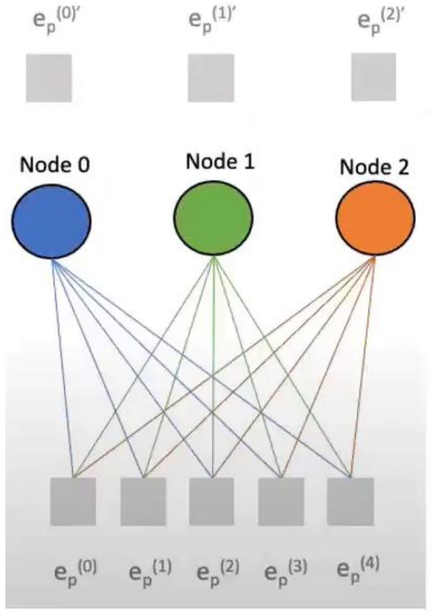
$$\text{Similarity}(A,B) = \frac{A \cdot B}{\text{Scaling}}$$

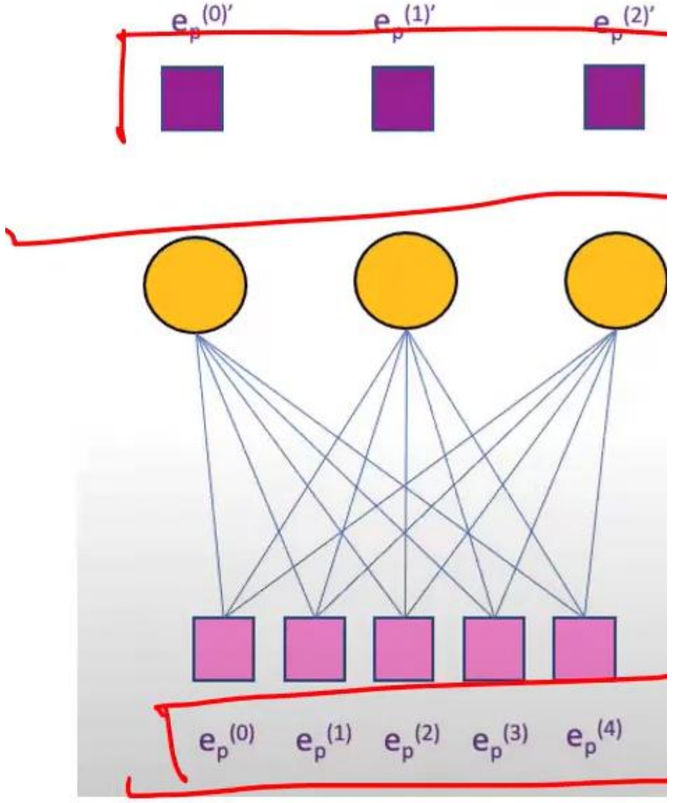
Similarity b/w Query and Key

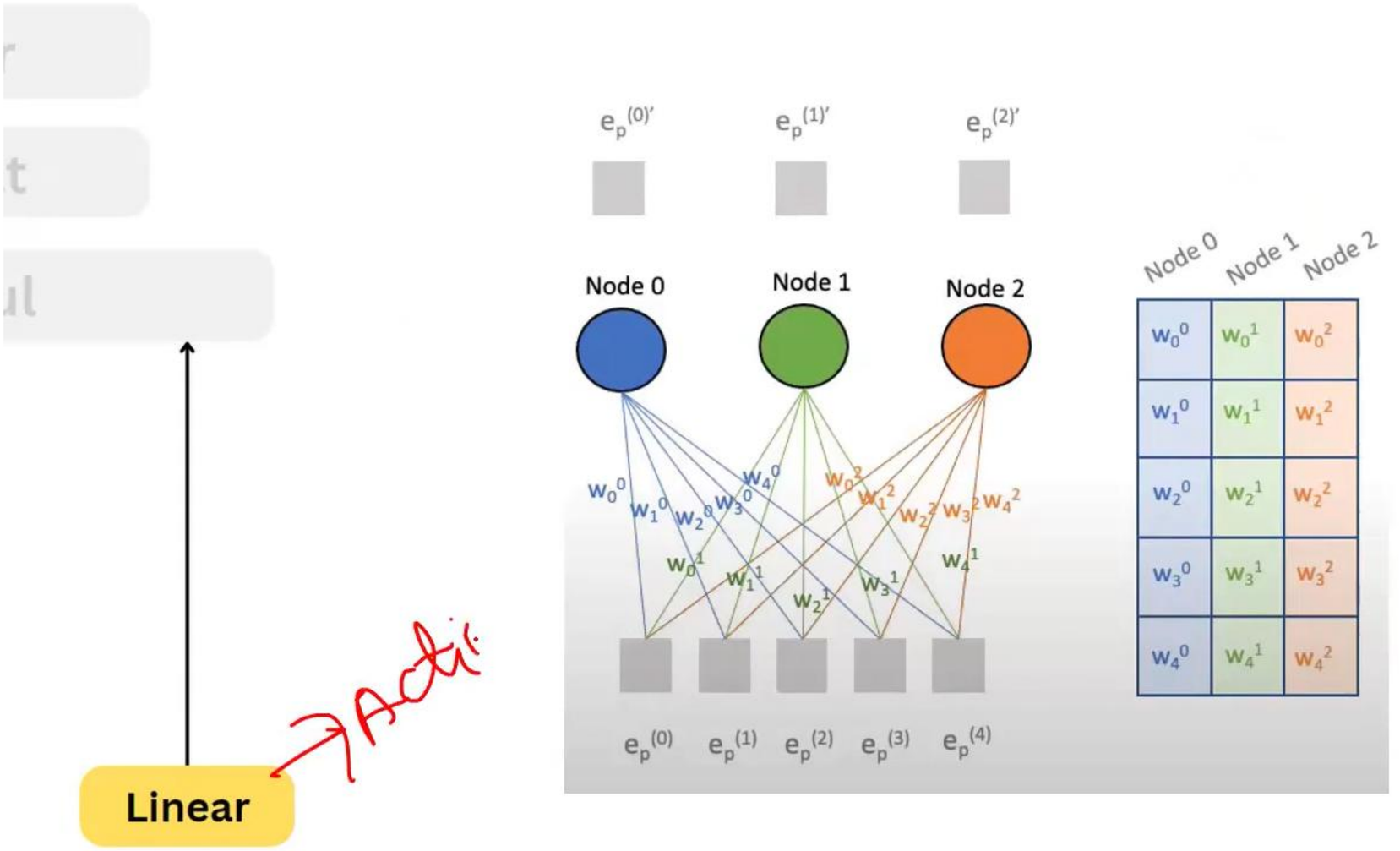
$$\text{Similarity}(Q,K) = \frac{Q \cdot K^{T}}{\text{Scaling}}$$

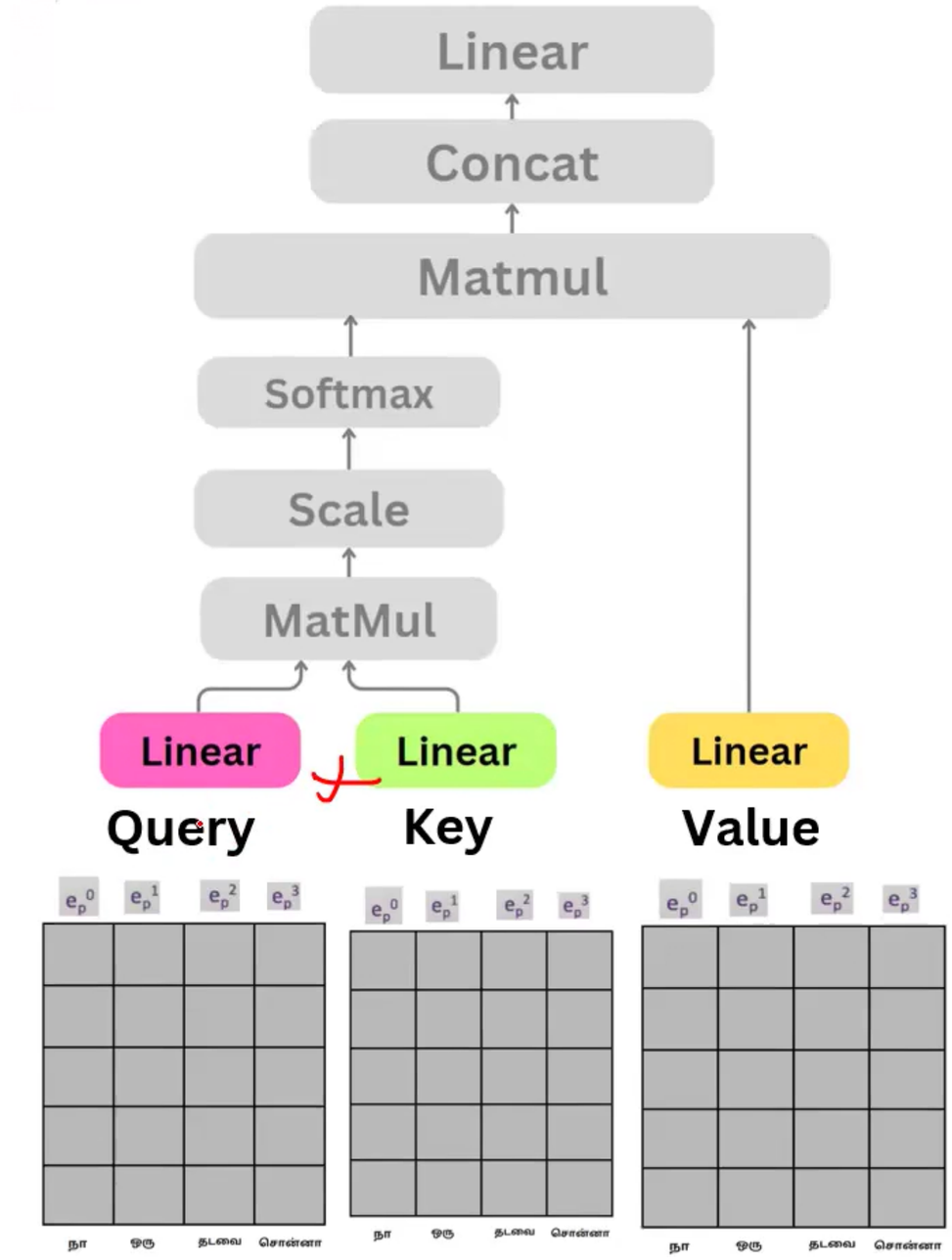
Multi Head Attention

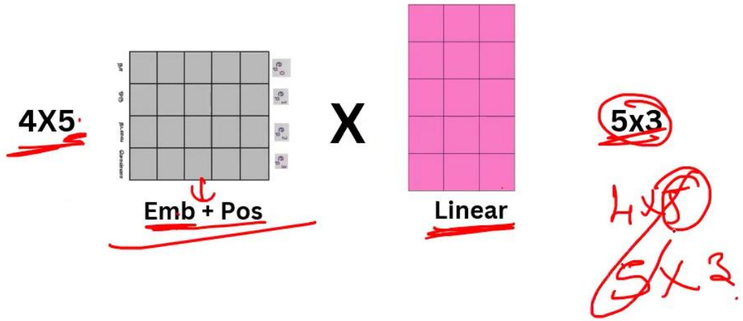


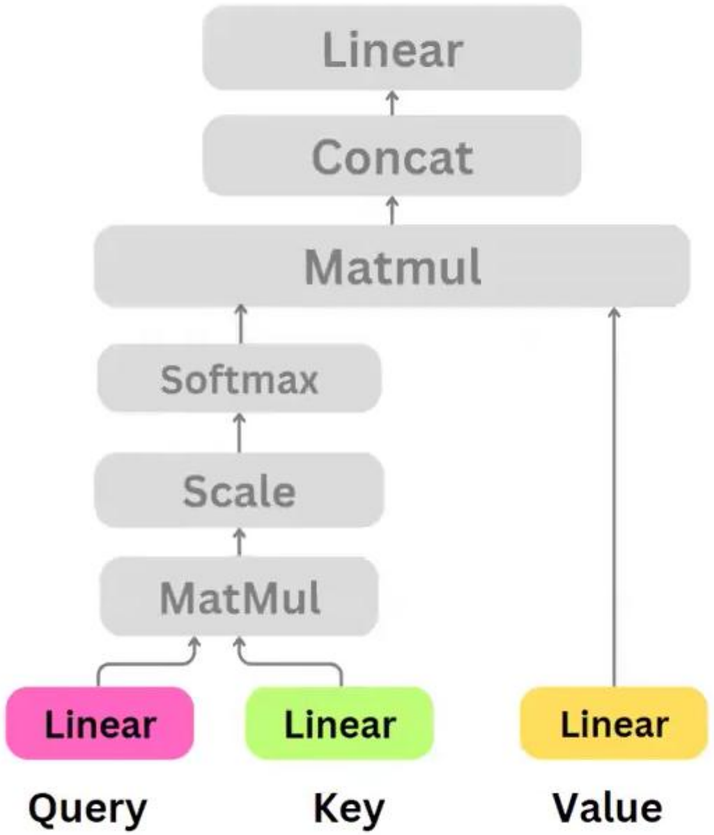


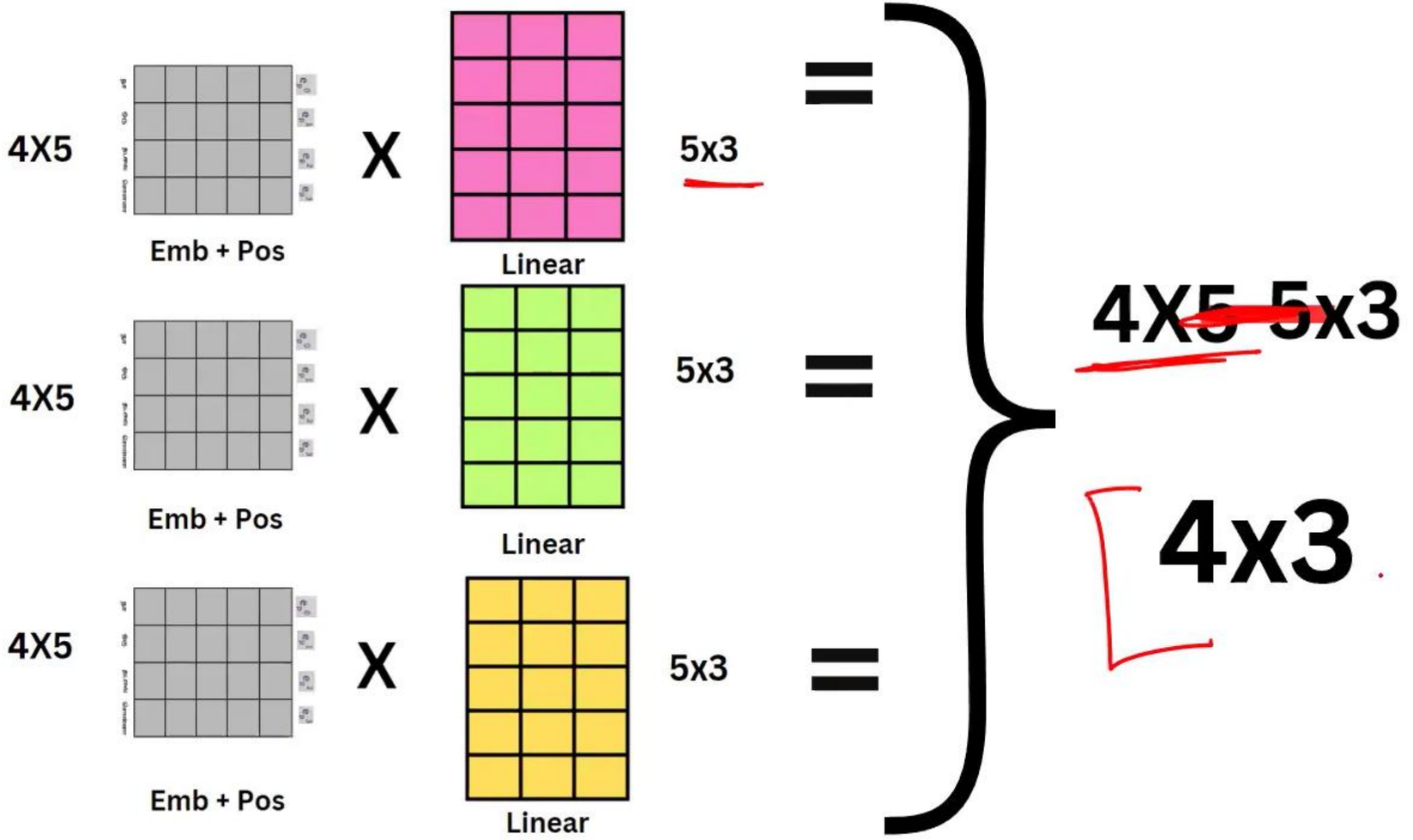


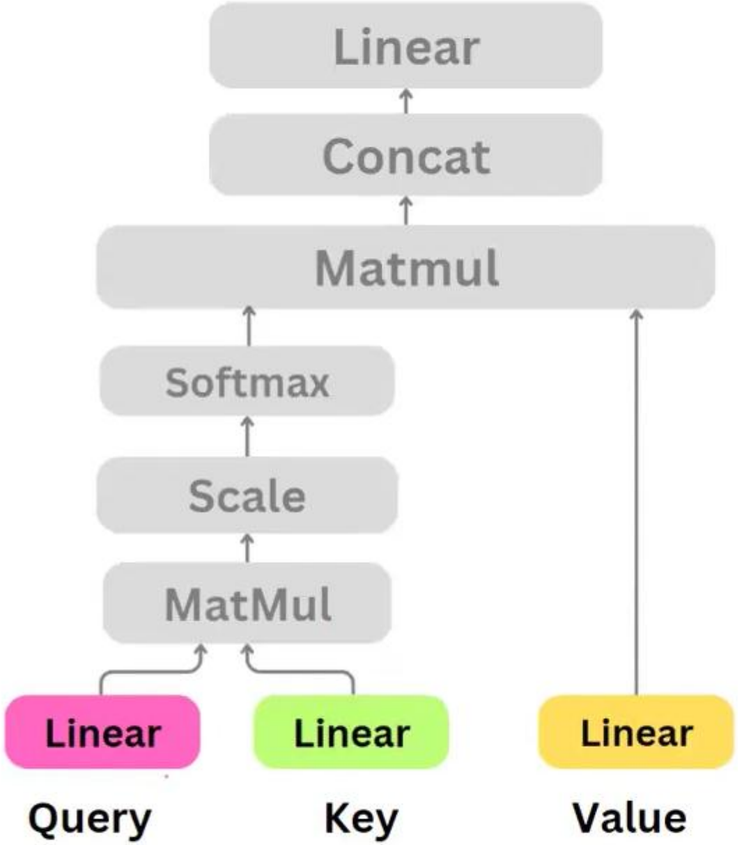

Linear
Node 0
Node 1
Node 2
Node 0
Node 1
Node 2

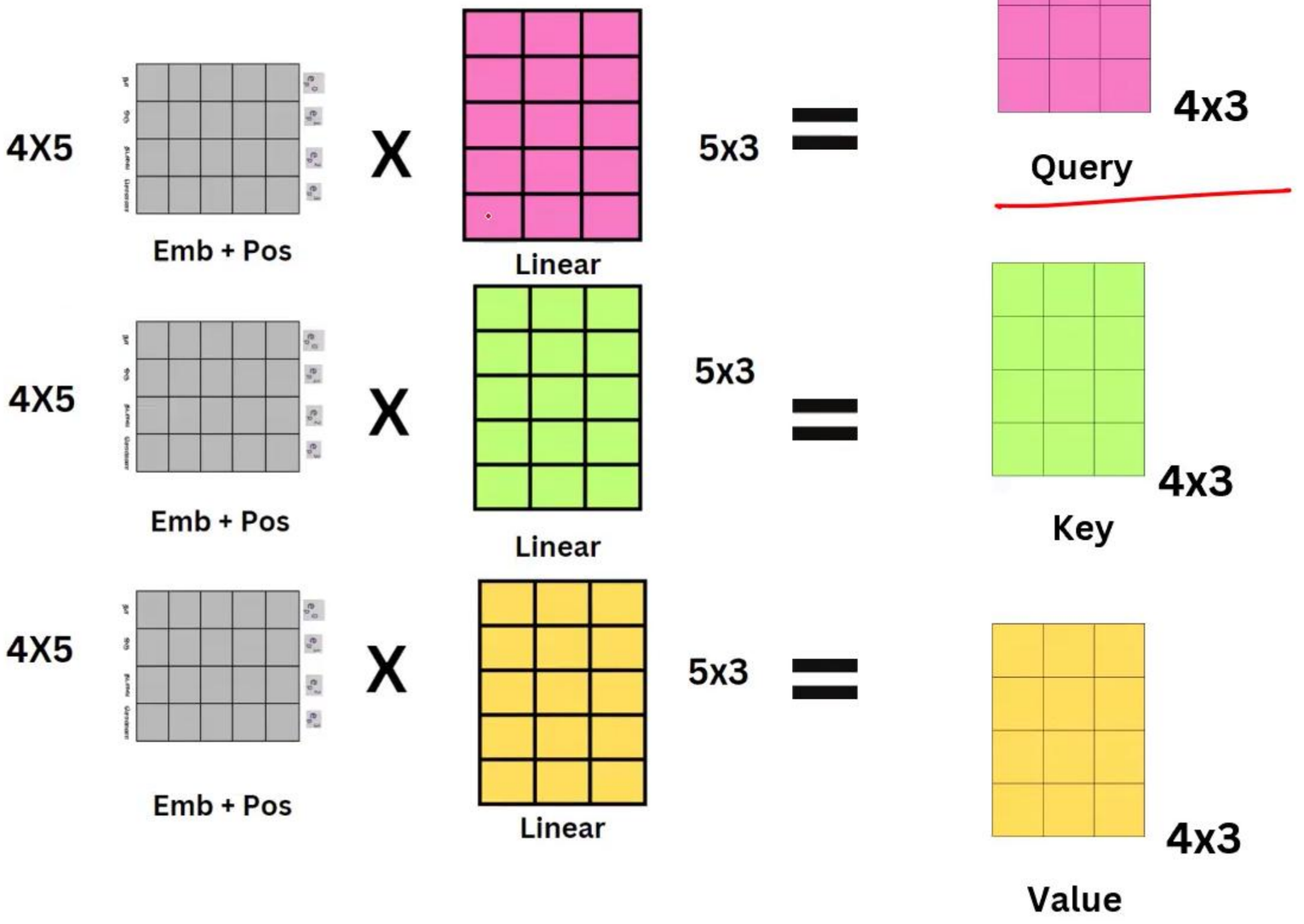

Linear
Concat
Matmul
Softmax
Scale
MatMul
Linear
Linear
Linear
Query
Key
Value
நா
ஒரு
தடவை
சொன்னா

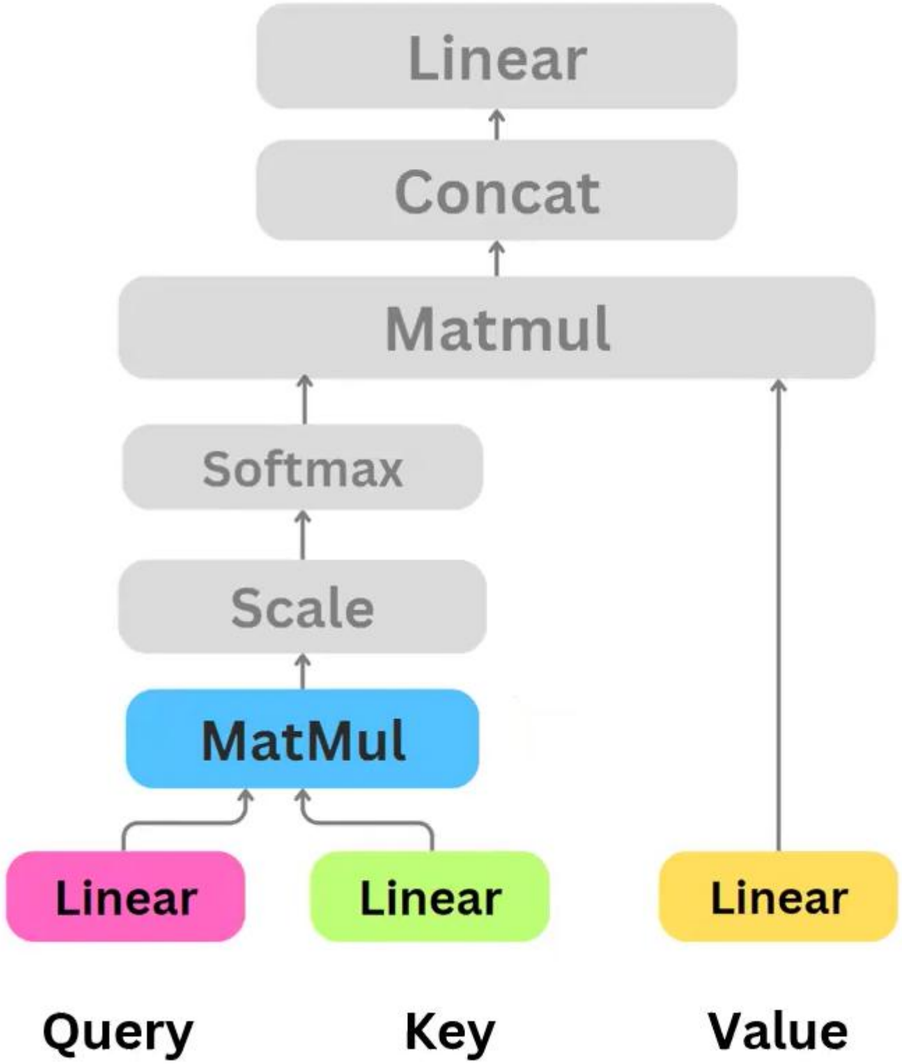

4X5
Emb + Pos
X
Linear
5x3

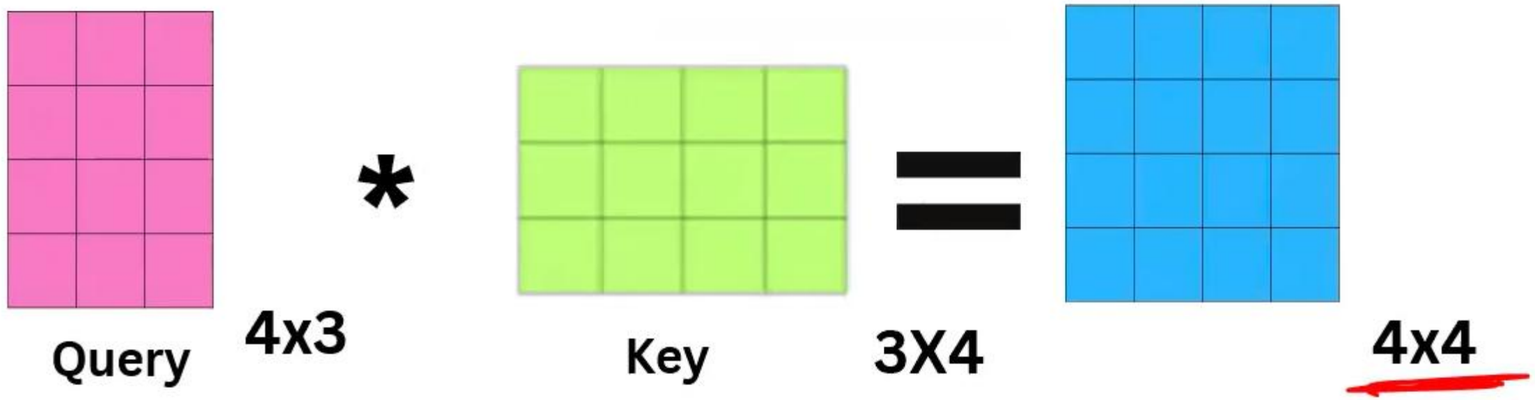

Linear
Concat
Matmul
Softmax
Scale
MatMul
Linear
Linear
Linear
Query
Key
Value


4X5
X
Emb + Pos
Linear
5x3
=
4X5
X
Emb + Pos
Linear
5x3
=
4X5
X
Emb + Pos
Linear
5x3
=
4X5 5x3
4x3


Linear
Concat
Matmul
Softmax
Scale
MatMul
Linear
Linear
Linear
Query
Key
Value


4X5
Emb + Pos
X
Linear
5x3
4x3
Query
4X5
Emb + Pos
X
Linear
5x3
4x3
Key
4X5
Emb + Pos
X
Linear
5x3
4x3
Value





$$Q \cdot K^T$$

Self Attention

	நா	ஒரு	தடவை	சொன்னா
நா	0.3	0.4	0.9	0.1
ஒரு	0.2	0.6	0.8	0.4
தடவை	0.4	0.3	0.5	0.3
சொன்னா	0.1	0.2	0.3	0.2

4x4

Attention Filter

*

Value/Original

=

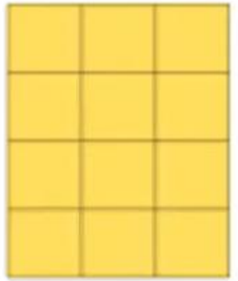
Filtered Important Features

	நா	ஒரு	தடவை	சொன்னா
நா	0.3	0.4	0.9	0.1
ஒரு	0.2	0.6	0.8	0.4
தடவை	0.4	0.3	0.5	0.3
சொன்னா	0.1	0.2	0.3	0.2

4x4

Attention Filter

*



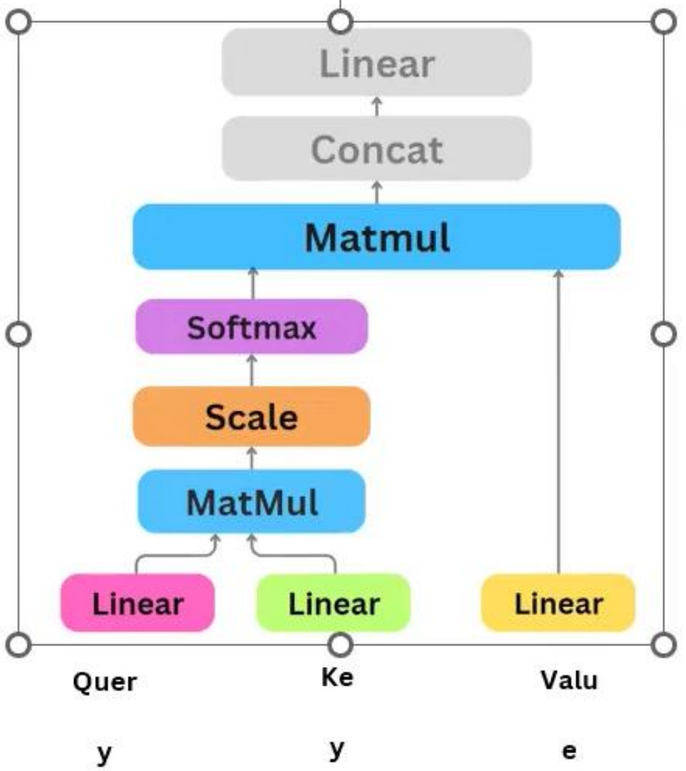
4x3

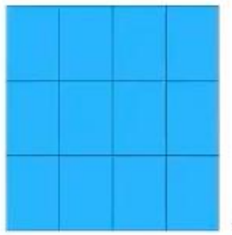
Value

=

4x3

Filtered Important Features





4x3

Filtered Important

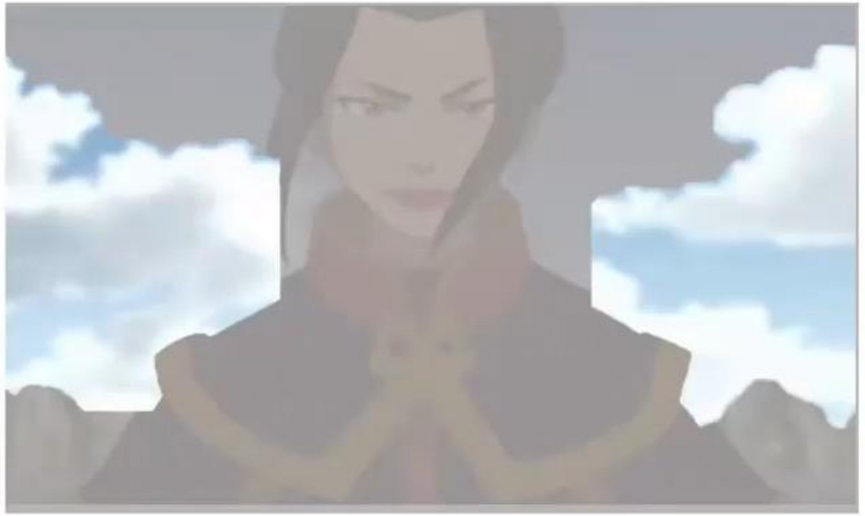
Features

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{Q.K^T}{\sqrt{d_k}}\right).V$$

Multi Head Attention-Similarity

Attention Filter 1

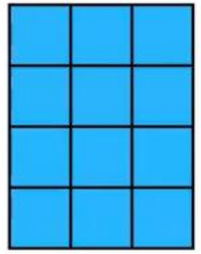
Attention Filter 2



Attention Filter 3

Multi Head Attention-Similarity

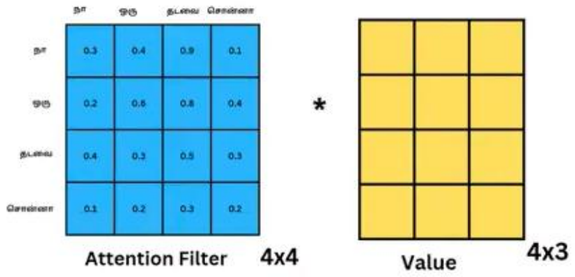
Head- 1

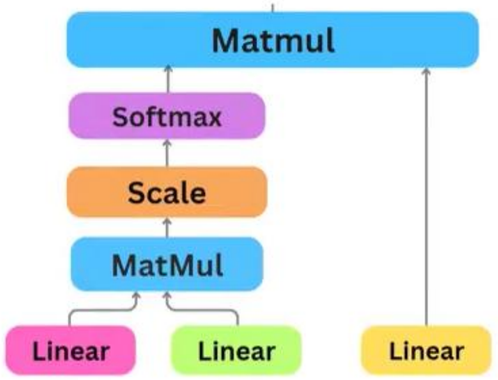


4x3

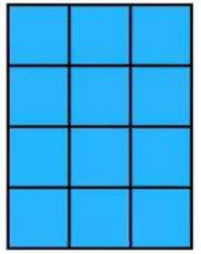
Attention Filter 1

Filtered Important Features





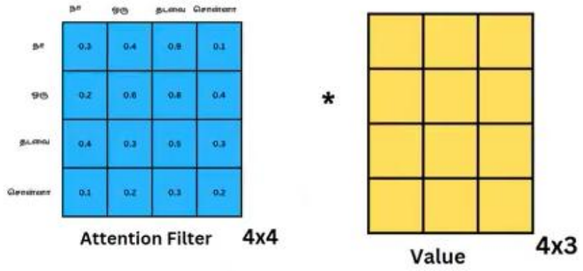
Head- 2

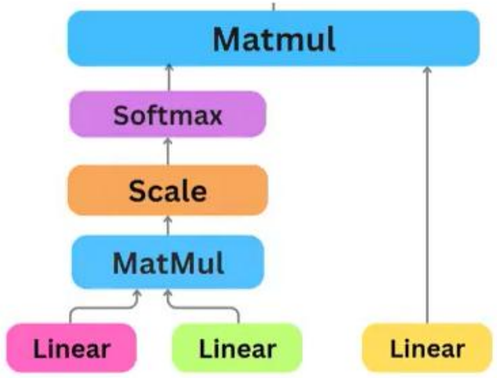


4x3

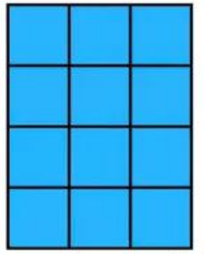
Attention Filter 2

Filtered Important Features





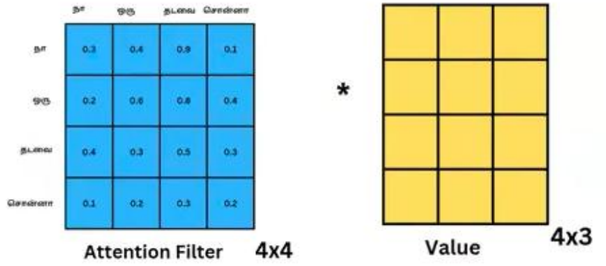
Head-3

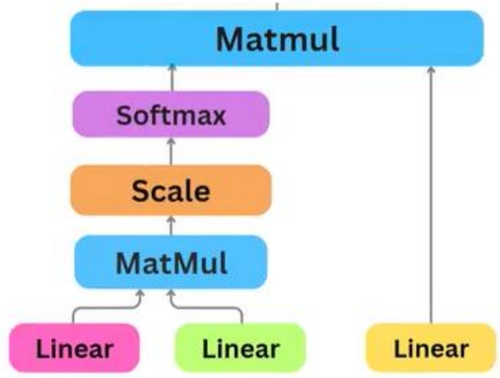


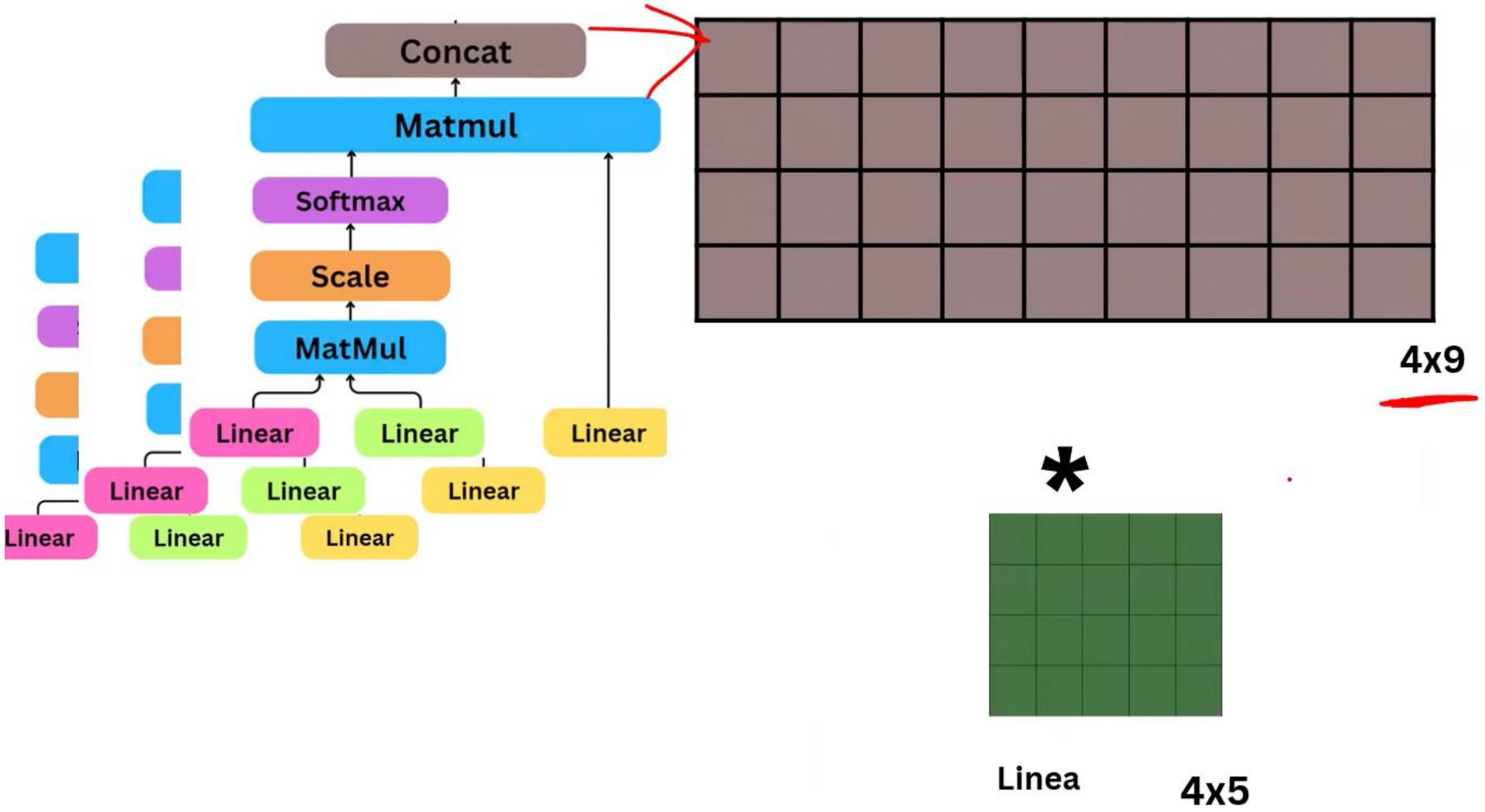
4x3

Attention Filter 3

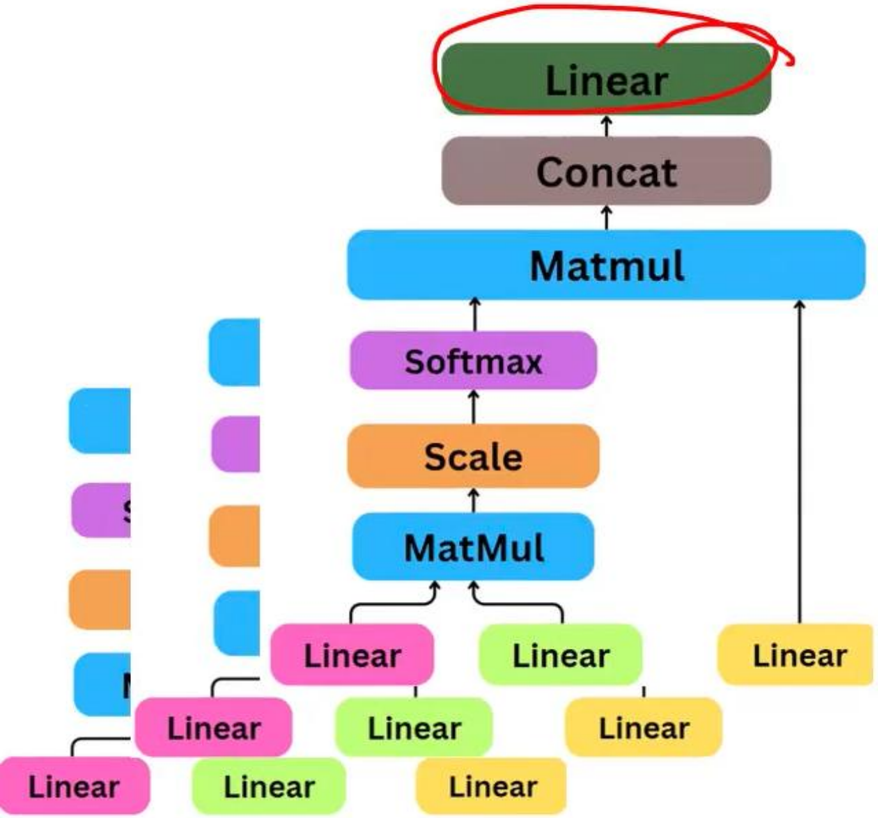
Filtered Important Features

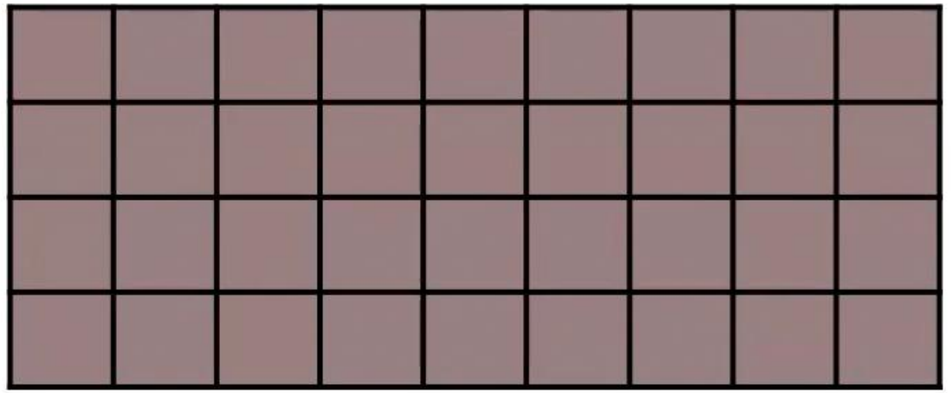





Concat
Matmul
Softmax
Scale
MatMul
Linear
Linear
Linear
Linear
Linear
Linear
Linear
Linear
Linear
4x9
*
Linea
4x5

Multi Head Attention-Similarity

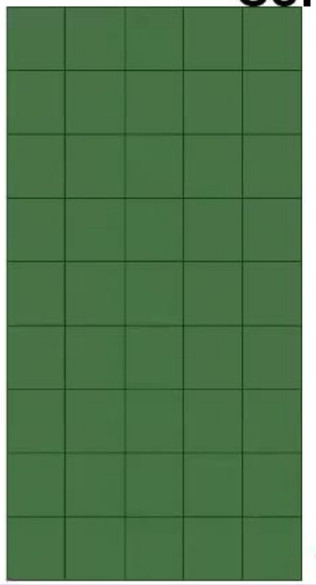




Concatenate Output 4x9

*

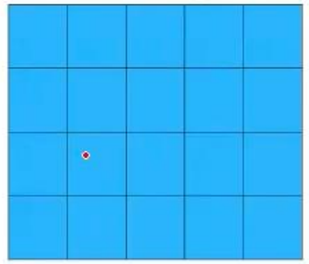
Linear



9x5

=

4x9 9x5 = 4x5



Multi-Head Attention Output 4x5