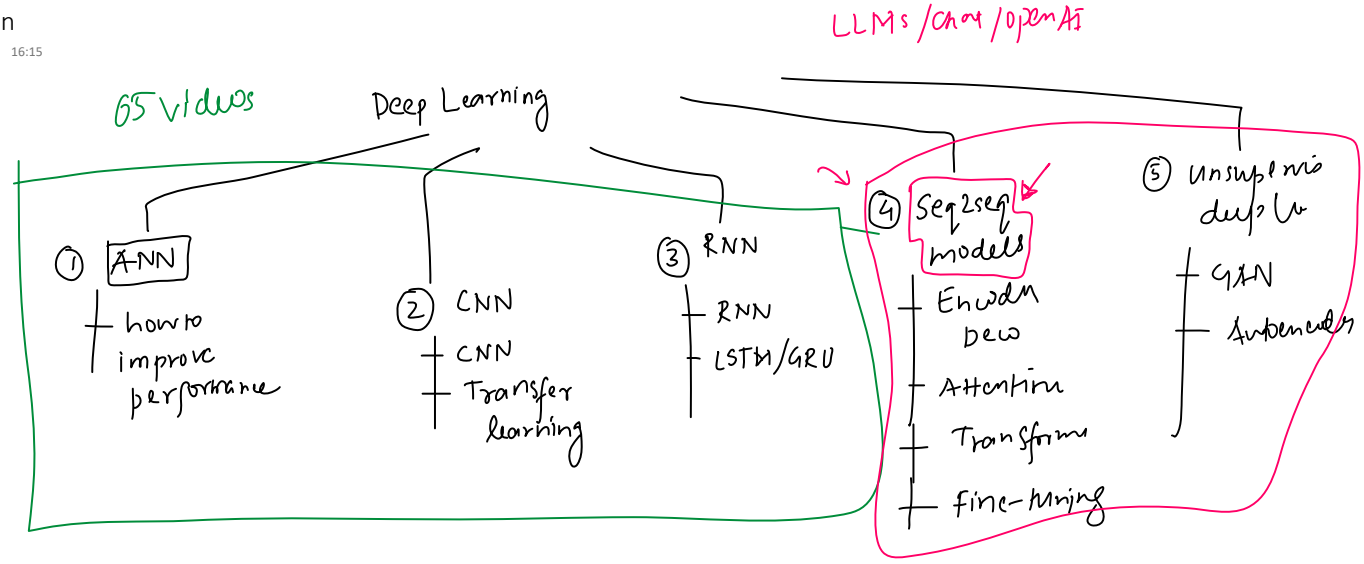
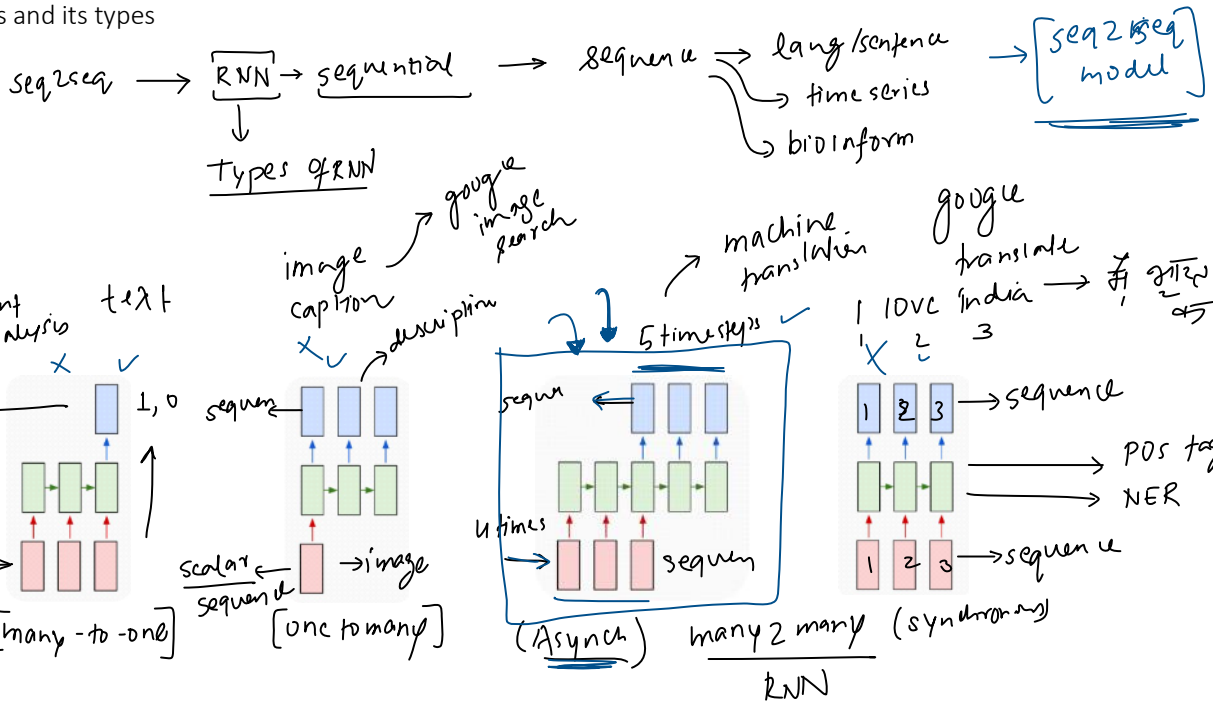
Introduction

16 November 2023 16:15



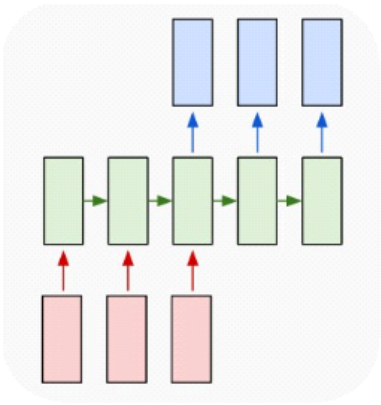
Sequence Tasks and its types

16 November 2023 16:16



Seq2Seq Tasks

16 November 2023 16:16



NLP

Seq2seq → machine trans

1) text summari → seq → seq

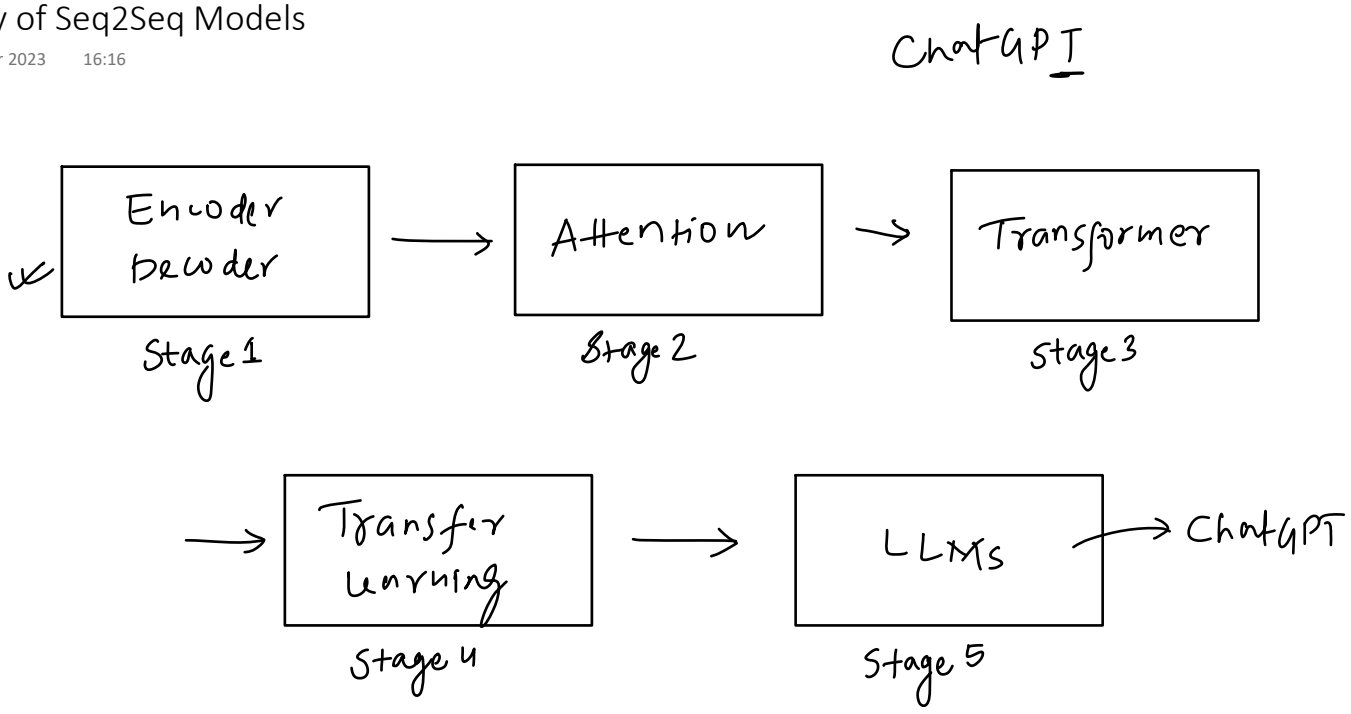
2) Question answer → seq → seq (knowledge base)

3) chatbot → input (text) → output (text)

4) speech-to-text → seq seq → seq2seq → chatgpt

History of Seq2Seq Models

16 November 2023 16:16



Stage 1 - Encoder Decoder Architecture

18 November 2023 16:16

2014 Seminal

Sequence to Sequence Learning with Neural Networks

Ilya Sutskever
Google
ilyasu@google.com

Oriol Vinyals
Google
vinyals@google.com

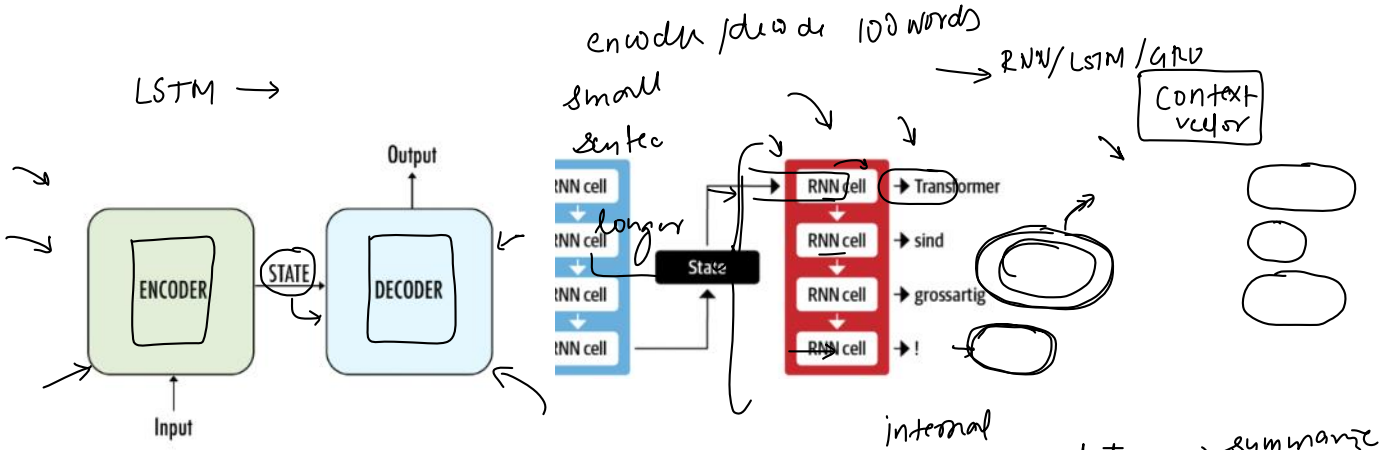
Quoc V. Le
Google
qvl@google.com

Abstract

Deep Neural Networks (DNNs) are powerful models that have achieved excellent performance on difficult learning tasks. Although DNNs work well whenever large labeled training sets are available, they cannot be used to map sequences to sequences. In this paper, we present a general end-to-end approach to sequence learning that makes minimal assumptions on the sequence structure. Our method uses a multilayered Long Short-Term Memory (LSTM) to map the input sequence to a vector of a fixed dimensionality, and then another deep LSTM to decode the target sequence from the vector. Our main result is that on an English to French translation task from the WMT'14 dataset, the translations produced by the LSTM achieve a BLEU score of 34.8 on the entire test set, where the LSTM's BLEU score was penalized on out-of-vocabulary words. Additionally, the LSTM did not have difficulty on long sentences. For comparison, a phrase-based SMT system achieves a BLEU score of 33.3 on the same dataset. When we used the LSTM to rerank the 1000 hypotheses produced by the aforementioned SMT system, its BLEU score increases to 36.5, which is close to the previous best result on this task. The LSTM also learned sensible phrase and sentence representations that are sensitive to word order and are relatively invariant to the active and the passive voice. Finally, we found that reversing the order of the words in all source sentences (but not target sentences) improved the LSTM's performance markedly, because doing so introduced many short term dependencies between the source and the target sentence which made the optimization problem easier.

Ilya Sutskever → cofounder OpenAI

seq2seq
↓
diff
↳ encoder decoder



seq2seq → input → output
machine translation
eng → hindi

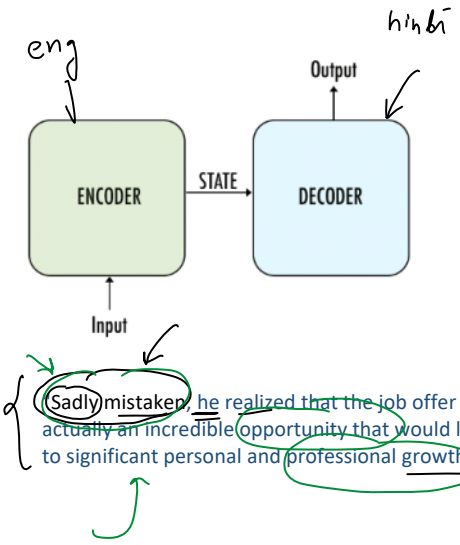
I love India → मैं भारत से प्यार करता हूँ

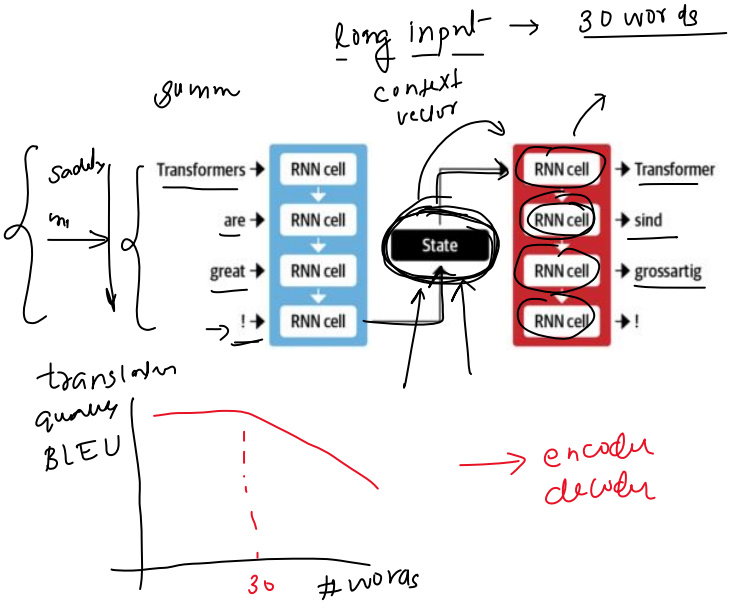
process word by word → compress

Attention mechanism

Stage 2 - Attention Mechanism

20 November 2023 10:59



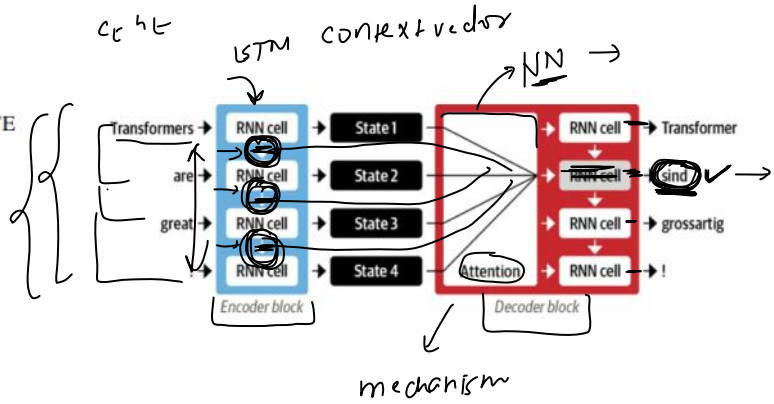


2015 Attention

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio***
Université de Montréal



ABSTRACT

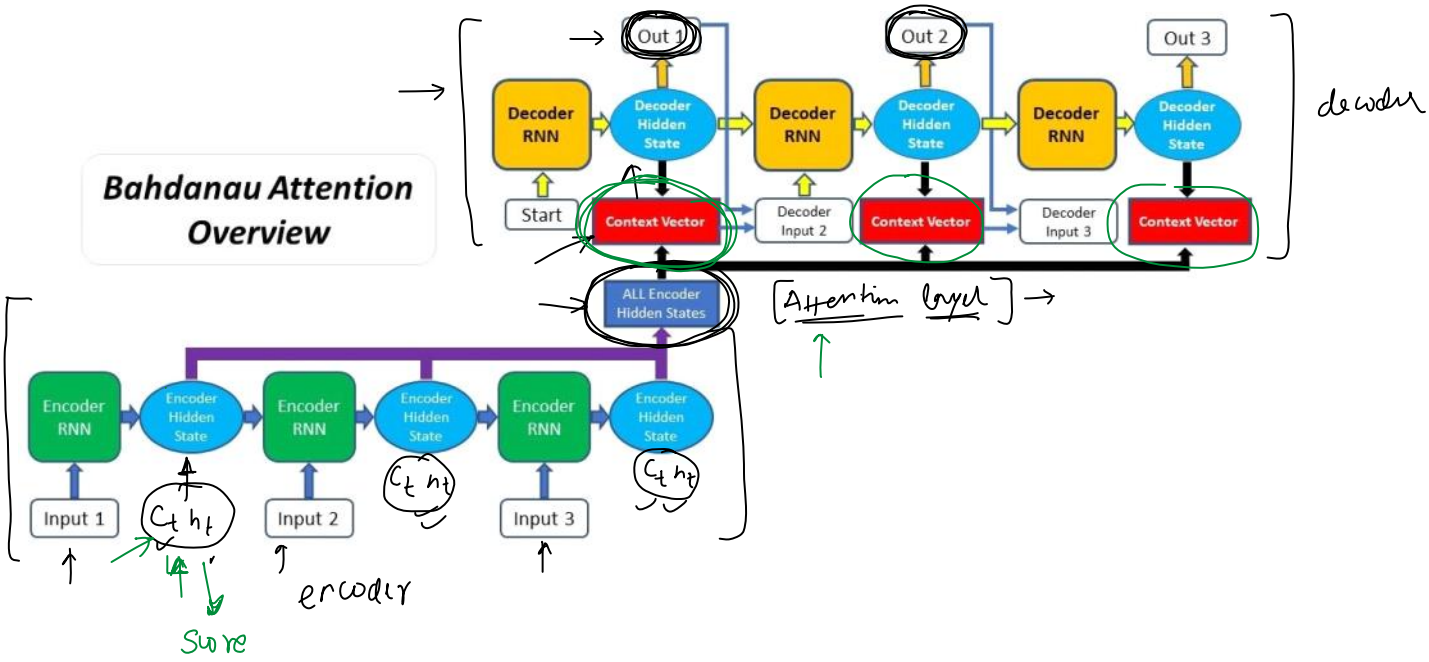
Neural machine translation is a recently proposed approach to machine translation. Unlike the traditional statistical machine translation, the neural machine translation aims at building a single neural network that can be jointly tuned to maximize the translation performance. The models proposed recently for neural machine translation often belong to a family of encoder–decoders and encode a source sentence into a fixed-length vector from which a decoder generates a translation. In this paper, we conjecture that the use of a fixed-length vector is a bottleneck in improving the performance of this basic encoder–decoder architecture, and propose to extend this by allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word, without having to form these parts as a hard segment explicitly. With this new approach, we achieve a translation performance comparable to the existing state-of-the-art phrase-based system on the task of English-to-French translation. Furthermore, qualitative analysis reveals that the (soft-)alignments found by the model agree well with our intuition.

1 INTRODUCTION

Neural machine translation is a newly emerging approach to machine translation, recently proposed by Kalchbrenner and Blunsom (2013), Sutskever *et al.* (2014) and Cho *et al.* (2014b). Unlike the traditional phrase-based translation system (see, e.g., Koehn *et al.*, 2003) which consists of many small sub-components that are tuned separately, neural machine translation attempts to build and train a single, large neural network that reads a sentence and outputs a correct translation.

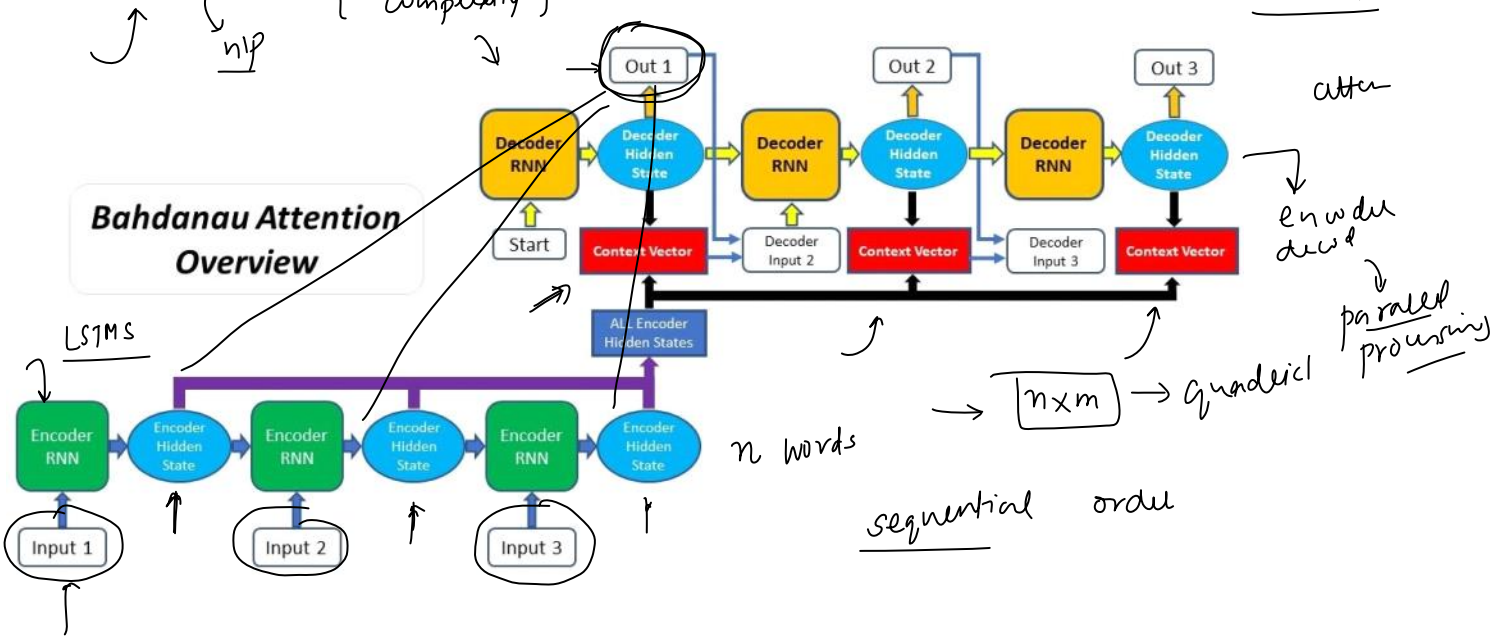
Most of the proposed neural machine translation models belong to a family of *encoder–decoders* (Sutskever *et al.*, 2014; Cho *et al.*, 2014a), with an encoder and a decoder for each language, or involve a language-specific encoder applied to each sentence whose outputs are then compared (Hermann and Blunsom, 2014). An encoder neural network reads and encodes a source sentence into a fixed-length vector. A decoder then outputs a translation from the encoded vector. The whole encoder–decoder system, which consists of the encoder and the decoder for a language pair, is jointly trained to maximize the probability of a correct translation given a source sentence.

A potential issue with this encoder–decoder approach is that a neural network needs to be able to compress all the necessary information of a source sentence into a fixed-length vector. This may make it difficult for the neural network to cope with long sentences, especially those that are longer than the sentences in the training corpus. Cho *et al.* (2014b) showed that indeed the performance of a basic encoder–decoder deteriorates rapidly as the length of an input sentence increases.


Bahdanau Attention Overview
Out 1
Out 2
Out 3
Decoder RNN
Decoder Hidden State
Start
Context Vector
Decoder Input 2
Decoder Input 3
ALL Encoder Hidden States
Encoder RNN
Encoder Hidden State
Input 1
Input 2
Input 3
decoder
Attention layer
encoder
Score

Stage 3 - Transformers

20 November 2023 12:18



Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

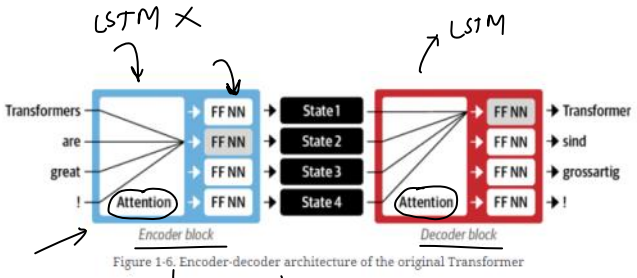


Figure 1-6. Encoder-decoder architecture of the original Transformer

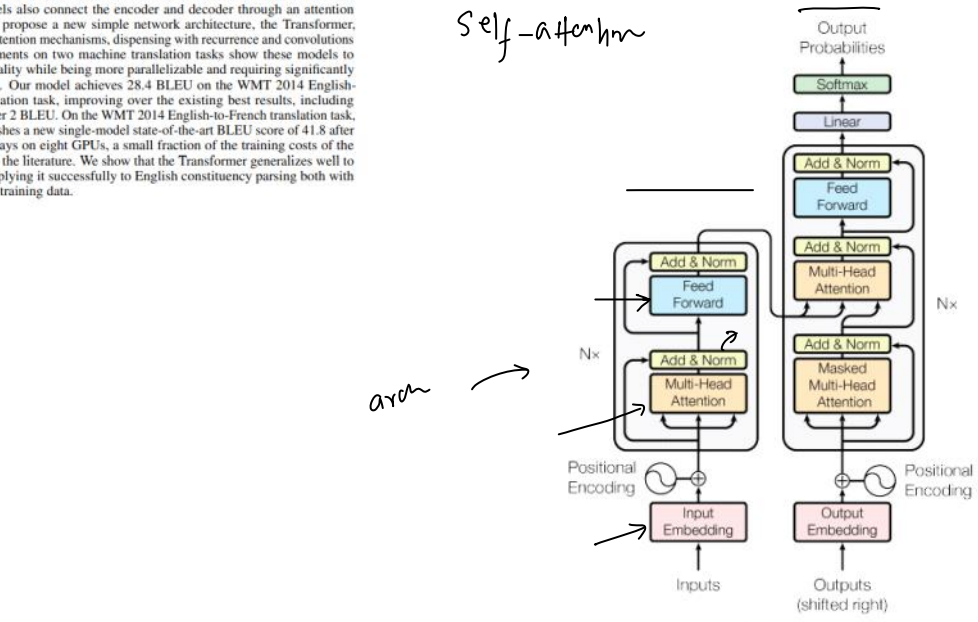
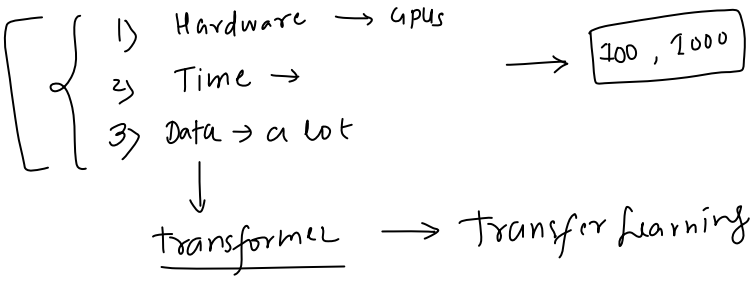


Figure 1: The Transformer - model architecture.

Stage 4 - Transfer Learning

20 November 2023 15:39



ULMFiT

Universal Language Model Fine-tuning for Text Classification

Jeremy Howard*
fast.ai
University of San Francisco
j@fast.ai

Sebastian Ruder*
Insight Centre, NUI Galway
Aylien Ltd., Dublin
sebastian@ruder.io

2018 → vision

transfer → NLP tasks

ULMFit

primary pretrain

Abstract

Inductive transfer learning has greatly impacted computer vision, but existing approaches in NLP still require task-specific modifications and training from scratch. We propose Universal Language Model Fine-tuning (ULMFiT), an effective transfer learning method that can be applied to any task in NLP, and introduce techniques that are key for fine-tuning a language model. Our method significantly outperforms the state-of-the-art on six text classification tasks, reducing the error by 18-24% on the majority of datasets. Furthermore, with only 100 labeled examples, it matches the performance of training from scratch on 100× more data. We open-source our pretrained models and code[1].

1 Introduction

Inductive transfer learning has had a large impact on computer vision (CV). Applied CV models (including object detection, classification, and segmentation) are rarely trained from scratch, but instead are fine-tuned from models that have been pretrained on ImageNet, MS-COCO, and other datasets (Sharif Razavian et al., 2014; Long et al., 2015a; He et al., 2016; Huang et al., 2017).

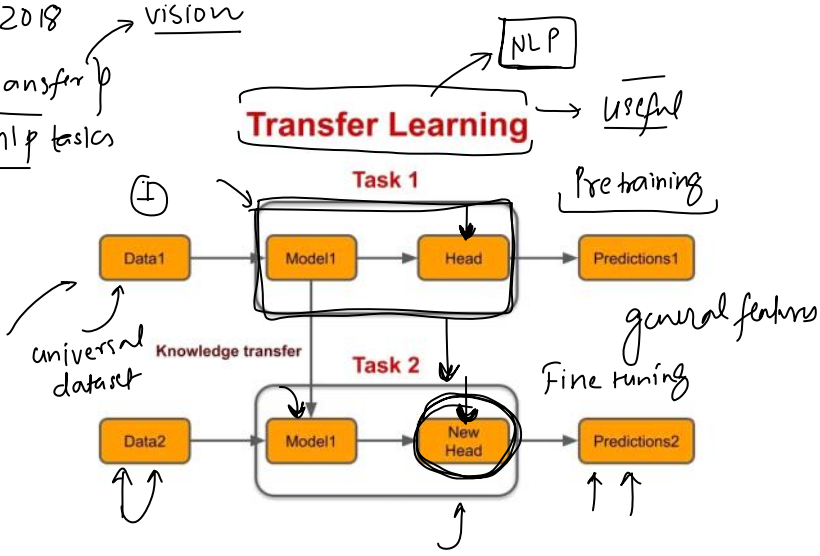
Text classification is a category of Natural Language Processing (NLP) tasks with real-world applications such as spam, fraud, and bot detection (Jindal and Liu, 2007; Ngai et al., 2011; Chu et al., 2012), emergency response (Caragea et al., 2011), and commercial document classification, such as for legal discovery (Roitblat et al., 2010).

While Deep Learning models have achieved state-of-the-art on many NLP tasks, these models are trained from scratch, requiring large datasets, and days to converge. Research in NLP focused mostly on *transductive* transfer (Blitzer et al., 2007). For *inductive* transfer, fine-tuning pretrained word embeddings (Mikolov et al., 2013), a simple transfer technique that only targets a model's first layer, has had a large impact in practice and is used in most state-of-the-art models. Recent approaches that concatenate embeddings derived from other tasks with the input at different layers (Peters et al., 2017; McCann et al., 2017; Peters et al., 2018) still train the main task model from scratch and treat pretrained embeddings as fixed parameters, limiting their usefulness.

In light of the benefits of pretraining (Erhan et al., 2010), we should be able to do better than *randomly initializing* the remaining parameters of our models. However, inductive transfer via fine-tuning has been unsuccessful for NLP (Mou et al., 2016). Dai and Le (2015) first proposed fine-tuning a language model (LM) but require millions of in-domain documents to achieve good performance, which severely limits its applicability.

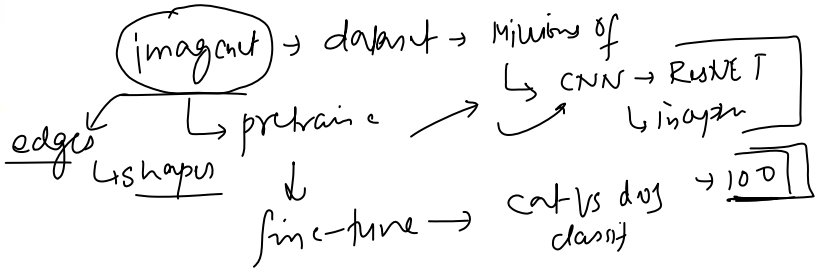
We show that not the idea of LM fine-tuning but our lack of knowledge of how to train them effectively has been hindering wider adoption. LMs overfit to small datasets and suffered catastrophic forgetting when fine-tuned with a classifier. Compared to CV, NLP models are typically more shallow and thus require different fine-tuning methods.

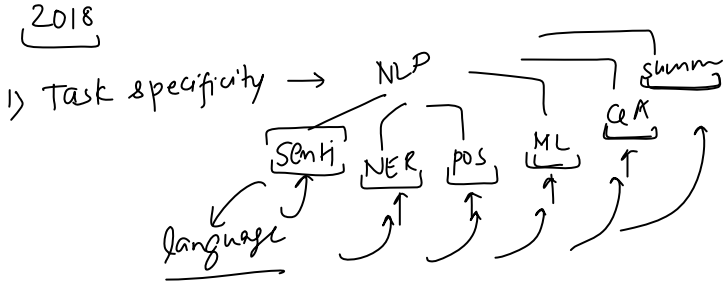
We propose a new method, Universal Language Model Fine-tuning (ULMFiT) that addresses these issues and enables robust inductive transfer learning for any NLP task, akin to fine-tuning ImageNet



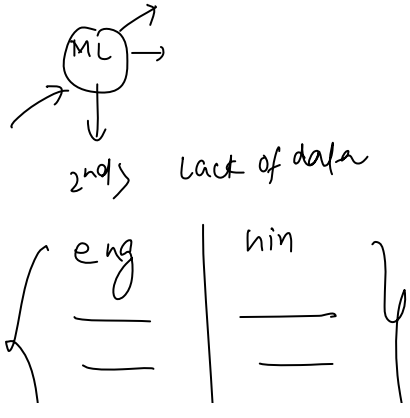
Transfer learning (TL) is a technique in which knowledge learned from a task is re-used in order to boost performance on a related task.

For example, for image classification, knowledge gained while learning to recognize cars could be applied when trying to recognize trucks.

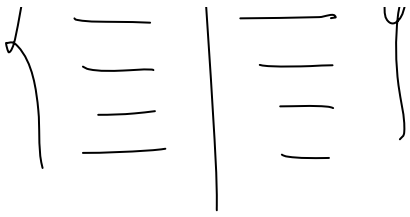




NLP → transfer learning



NLP → transfer learning



Pretraining → Machine transl X

→ language modelling → pretraining → sucessful → (2)

NLP task → NLP/DL model next word pred

I live in india. and the capital is New Delhi

Language modelling as a Pretraining task

→ unsupervised pretrain task

1) Rich feature learning

The hotel was exceptionally clean, yet the service was bad / pathetic

→ know trans

↓

text classi / quest. | textsumm | NER | pos

mt (labw → supervised labeled

eng | nin → unsupervised task

2) Huge avail of data

pdf → dataset labelling

[ULMFiT]

fine tuning

X transformer

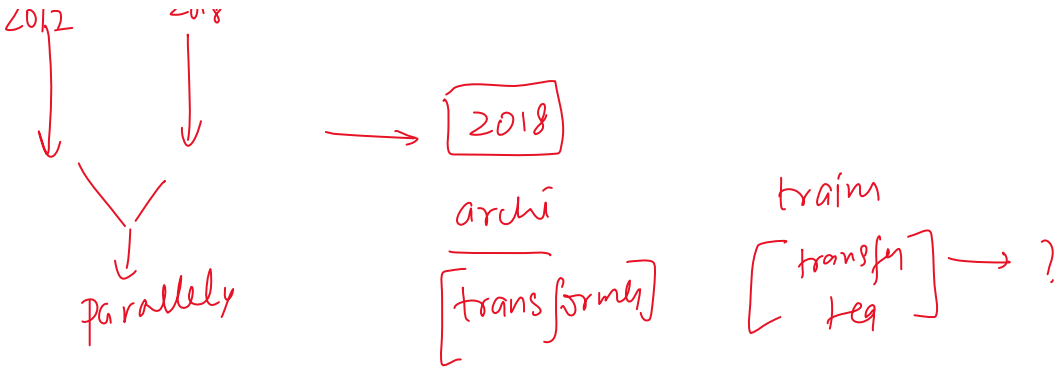
AWD LSTM → wikipedia → classifier → imdb / yelp / new data → model → test → State of the art

Unsupervised pretrain Language modelling

Scratch → 10000 rows

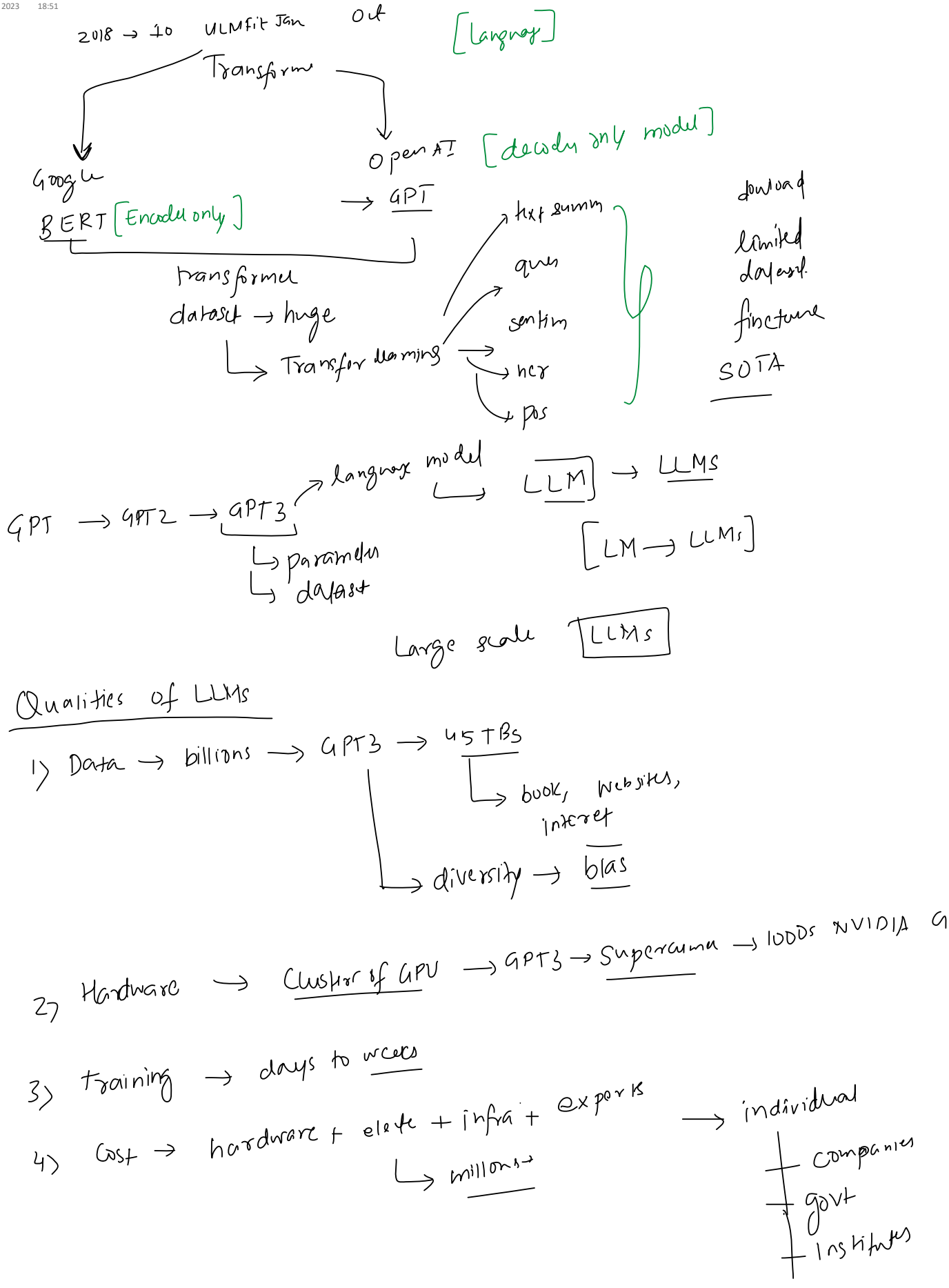
100 row → better →

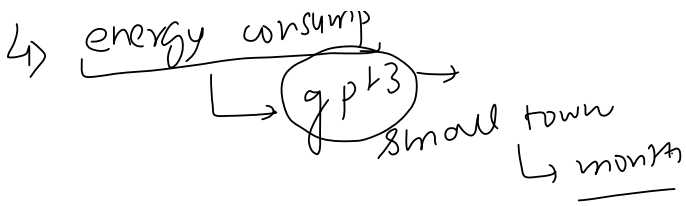
2012 2018


2012
2018
parallely
2018
archi
transformer
train
transfer
eq
?

Stage 5 - LLMs

20 November 2023 18:51




energy consump
gpt3
small town
month

The Grand Finale - ChatGPT

22 November 2023 10:13

GPT3 → chatGPT

[diff]

GPT → model

[ChatGPT] → application

Chat-bot

NOV

hp → chatgpt → Bard, Jasper

intel → GPT → Bard, Jasper

dell

ASW

GPT3 → [ChatGPT]

1) RLHF → Reinf lea from human feed

- 1 Supervised Fine tuning → dataset → labeled
- 2 Reinforce → prompt produces
  - responses
  - human → respone rank

2) Incorporate safety and ethical guideline

- minize biases

3) improvent in contextual point

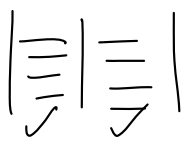
- context retain → maintain context → dialogu Conver

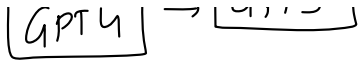
4) Dialogue specific training

- conversation
- better unders → dialogue lang → patterns

5) ChatGPT continous imp → human feedback usu

train → refining → GPT4 → GPT3


train → gpt


GPT4

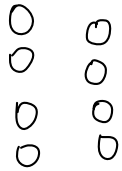
Seq2Seq Data

08 December 2023 17:12

Spatial placement

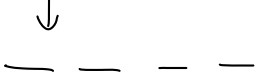
DNN

1> Tabular → [ANN] ×



Encoder Decoder Archi

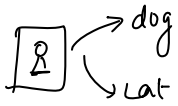
3> Sequential → textual
→ timeseries



RNN

LSTMs GRUs

2> Image data → → CNN ×



dog

cat

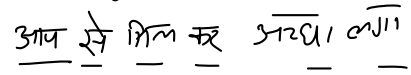
[2d grid]

4> Seq2seq data difficult

[input seq → output seq]

machine translation

nice to meet you



input → sent → eng → variable length

output → sent → hindi → variable len

3 words eng → 3 words

variable length

[lstm/gru] → input

[output]

Before Starting

08 December 2023 19:24

Prerequisite

- RNN / LSTM

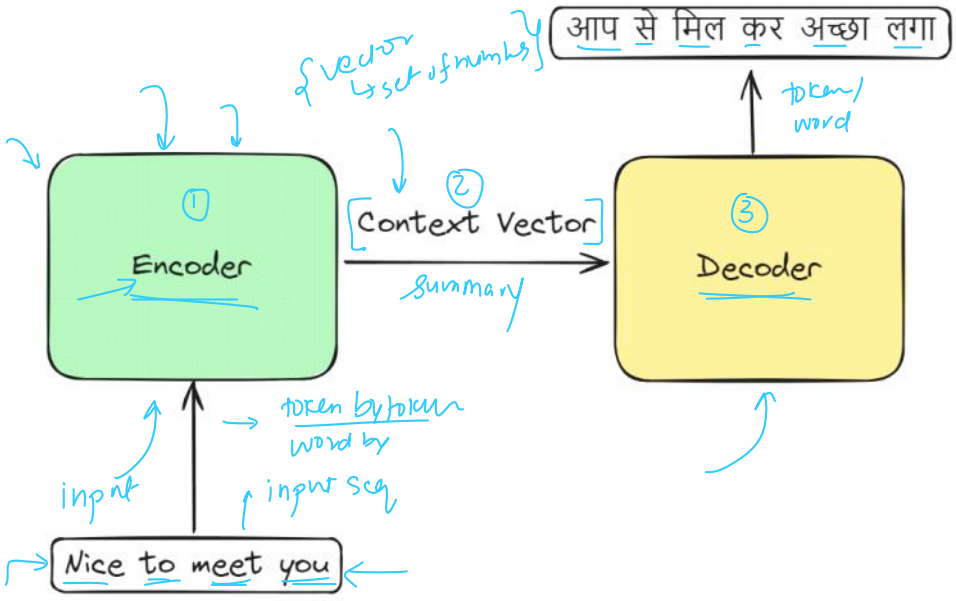
Plan of attack

→ simple version
→ deep
→ improvements

High Level Overview

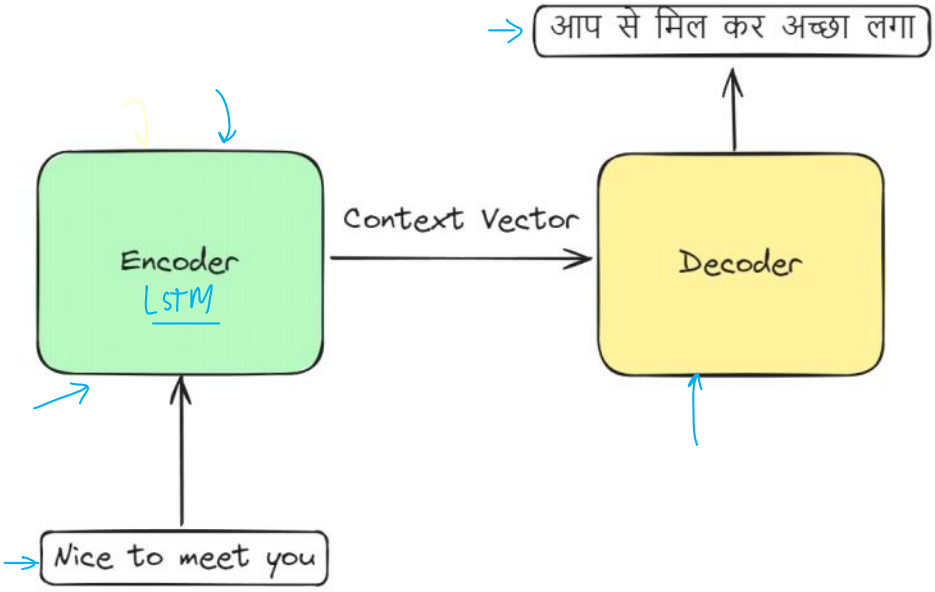
08 December 2023 17:14

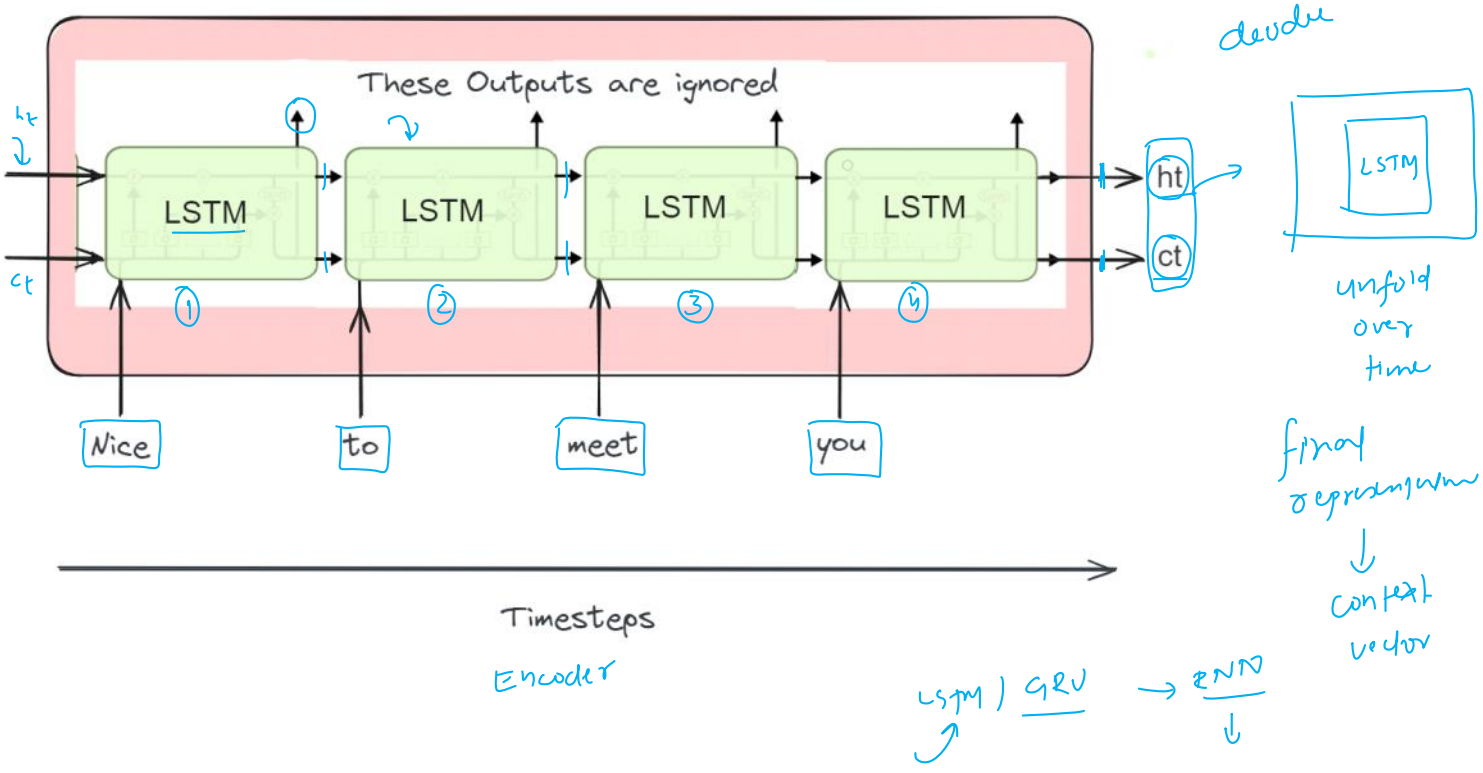
Machine tran → exam

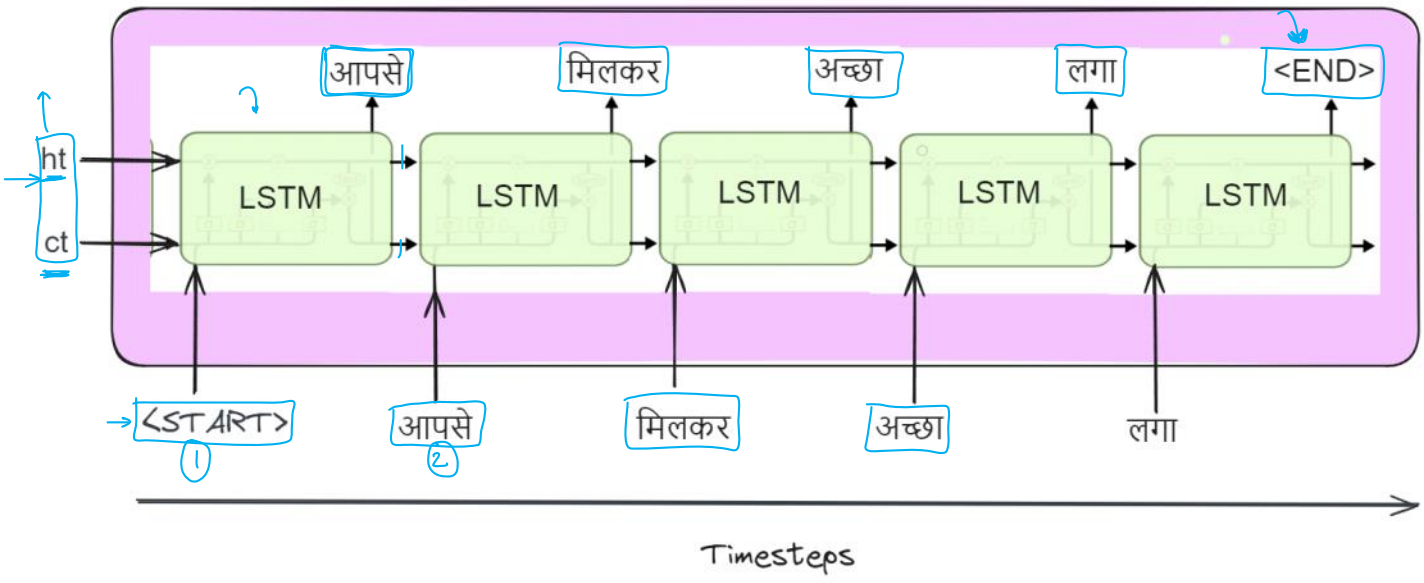


What's under the hood?

08 December 2023 17:14

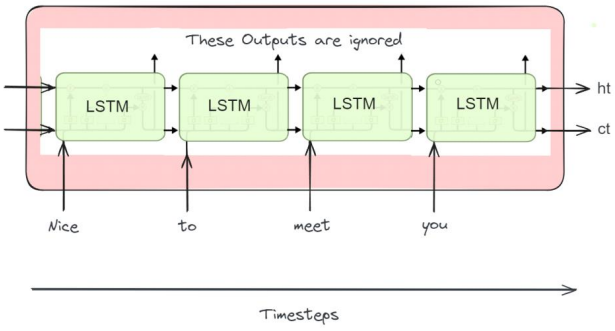


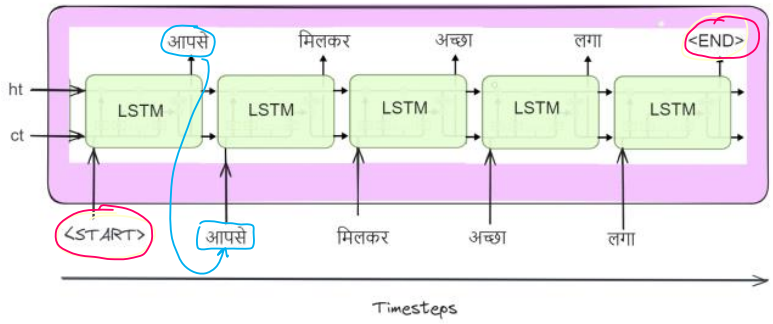


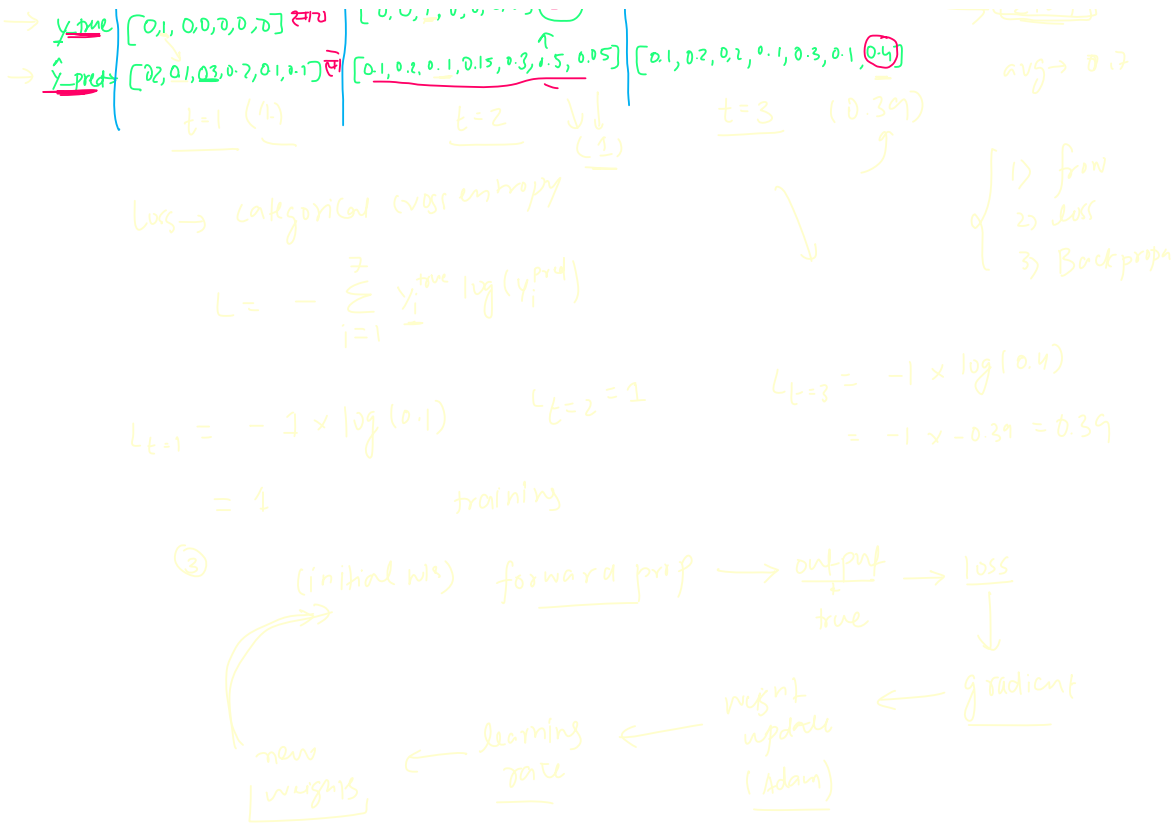

ht
ct
LSTM
LSTM
LSTM
LSTM
LSTM
आपसे
मिलकर
अच्छा
लगा
<END>
<START>
आपसे
मिलकर
अच्छा
लगा
1
2
Timesteps

Training the Architecture using Backpropagation

08 December 2023 17:15

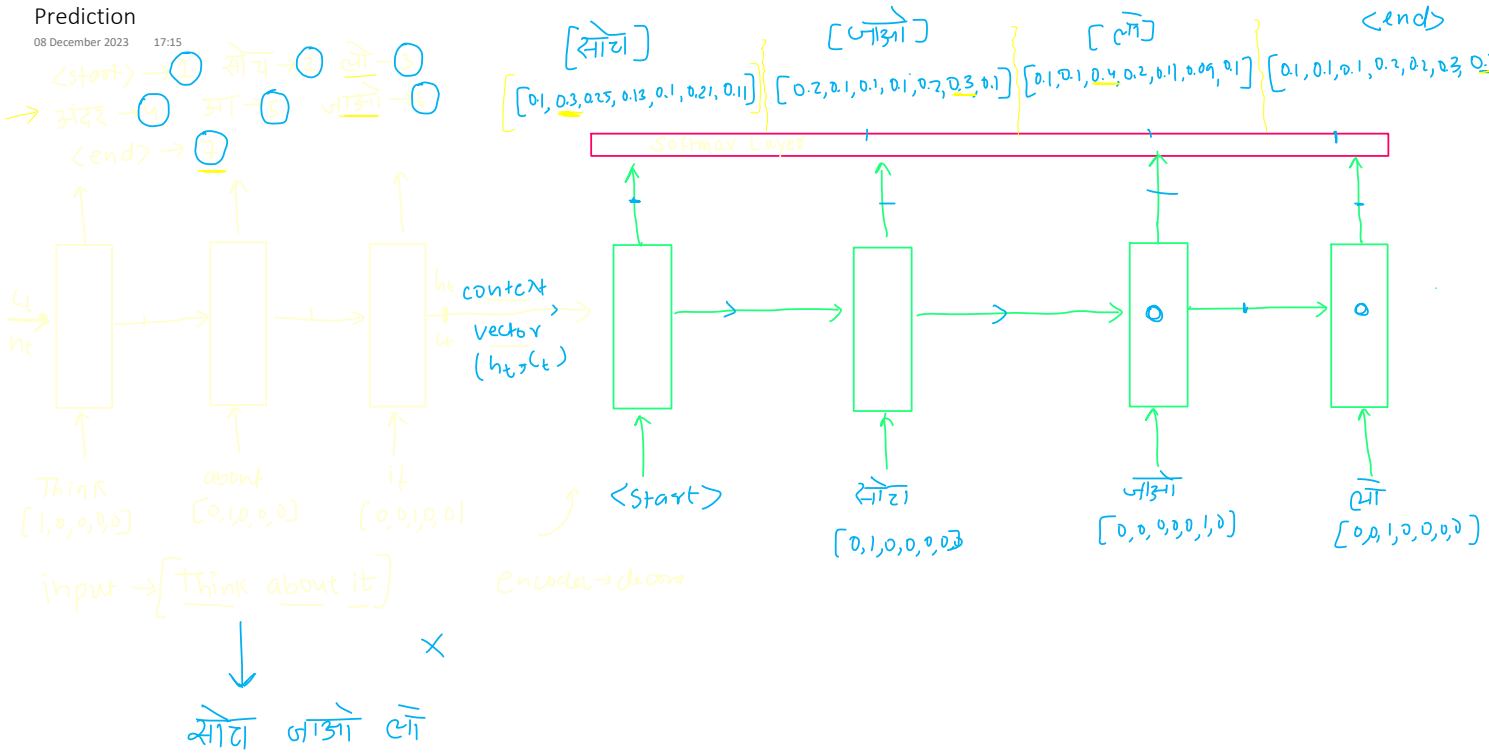






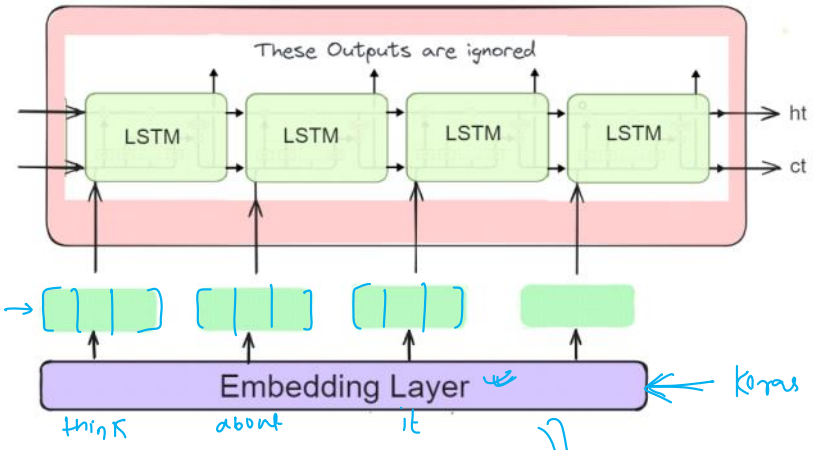
Prediction

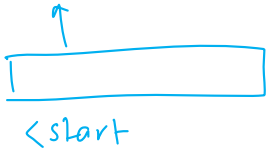
08 December 2023 17:15



Improvement 1 - Embeddings

08 December 2023 17:16





encoder
decoder

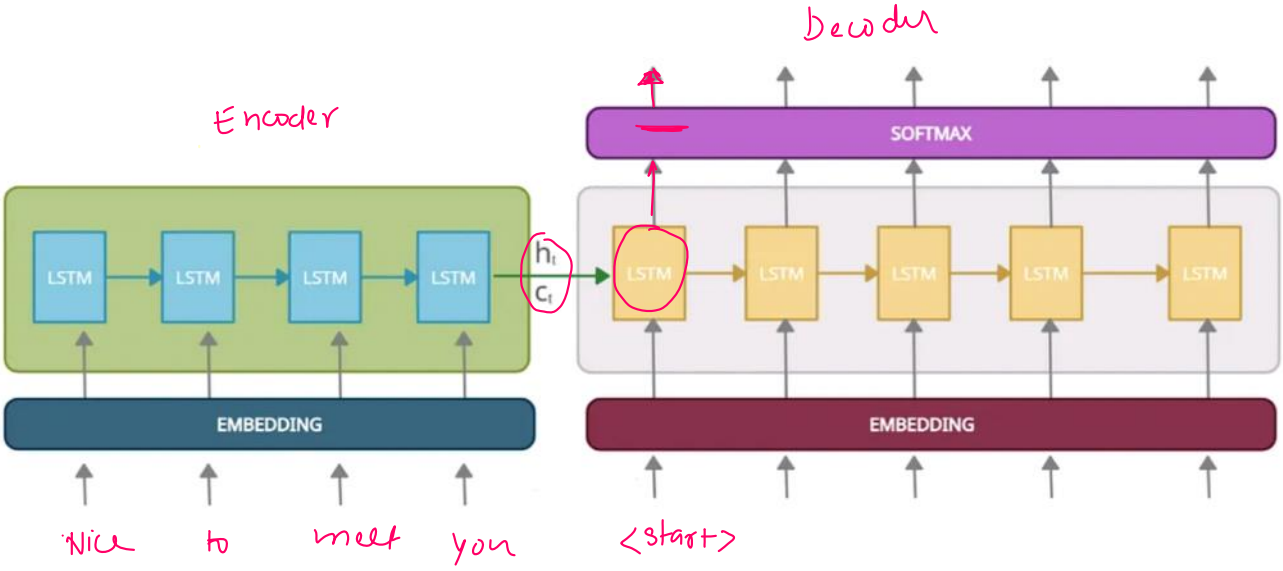
Keras

3 dim

low dimen / dense
summary of word

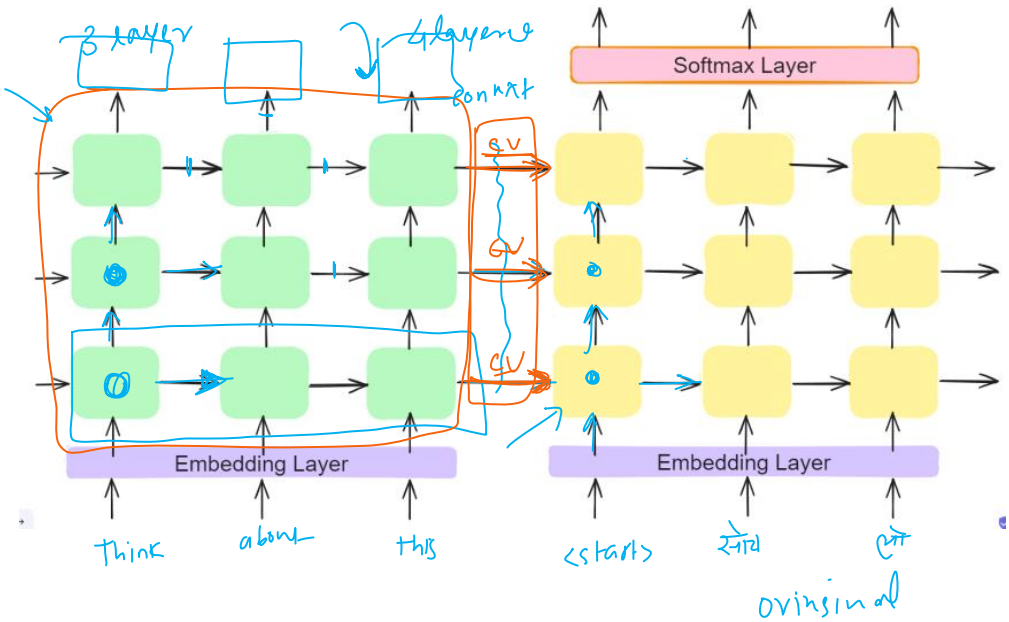
pretrained → Word2vec / Glove

train



Improvement 2 - Deep LSTMs

08 December 2023 17:16

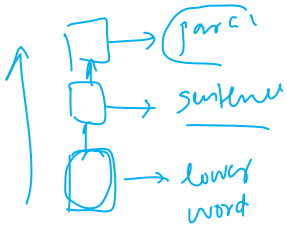


long para single layer

3> deep learning NN learning

1) long term dependency

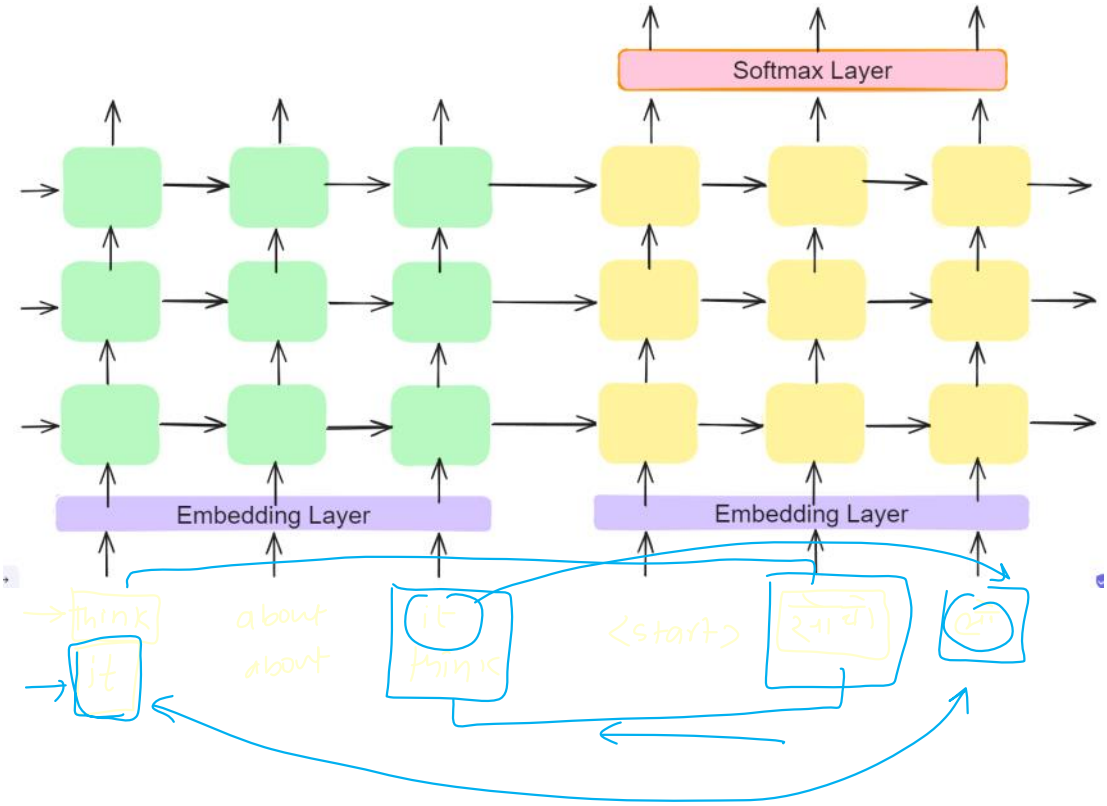
2) layered representation



the phone battery is bad but the overall is great

Improvement 3 - Reversing the Input

08 December 2023 17:16



The Sutskever Architecture

08 December 2023 17:18

Application to Translation: The model focused on translating English to French, demonstrating the effectiveness of sequence-to-sequence learning in neural machine translation.

Special End-of-Sentence Symbol: Each sentence in the dataset was terminated with a unique end-of-sentence symbol ("<EOS>"), enabling the model to recognize the end of a sequence.

Dataset: The model was trained on a subset of 12 million sentences, comprising 348 million French words and 304 million English words, taken from a publicly available dataset.

Vocabulary Limitation: To manage computational complexity, fixed vocabularies for both languages were used, with 160,000 most frequent words for English and 80,000 for French. Words not in these vocabularies were replaced with a special "UNK" token.

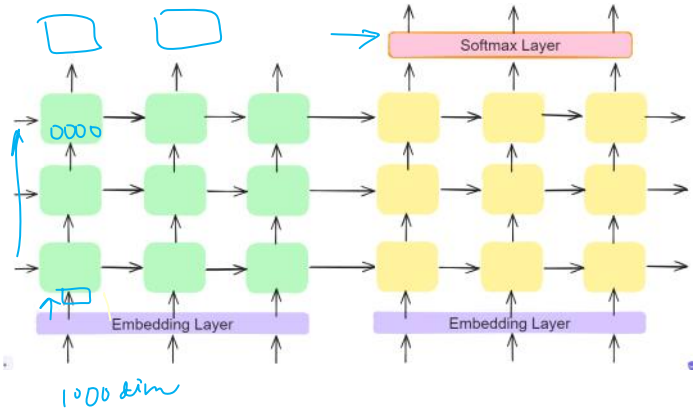
Reversing Input Sequences: The input sentences (English) were reversed before feeding them into the model, which was found to significantly improve the model's learning efficiency, especially for longer sentences.

Word Embeddings: The model used a 1000-dimensional word embedding layer to represent input words, providing dense, meaningful representations of each word.

Architecture Details: Both the input (encoder) and output (decoder) models had 4 layers, with each layer containing 1,000 units, showcasing a deep LSTM-based architecture.

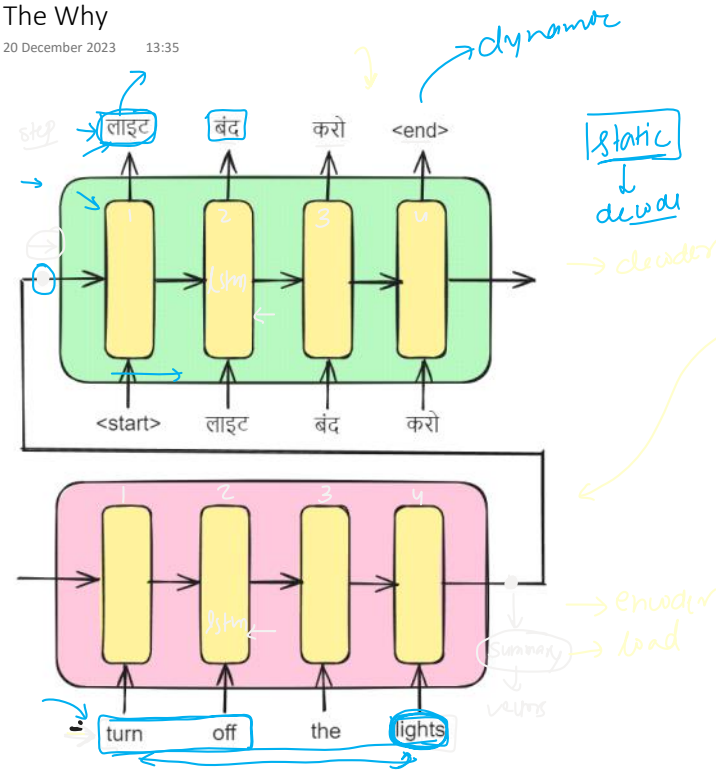
Output Layer and Training: The output layer employed a Softmax function to generate the probability distribution over the target vocabulary. The model was trained end-to-end with these settings.

Performance - BLEU Score: The model achieved a BLEU score of 34.81, surpassing the baseline Statistical Machine Translation (SMT) system's score of 33.30 on the same dataset, marking a significant advancement in neural machine translation.



The Why

20 December 2023 13:35

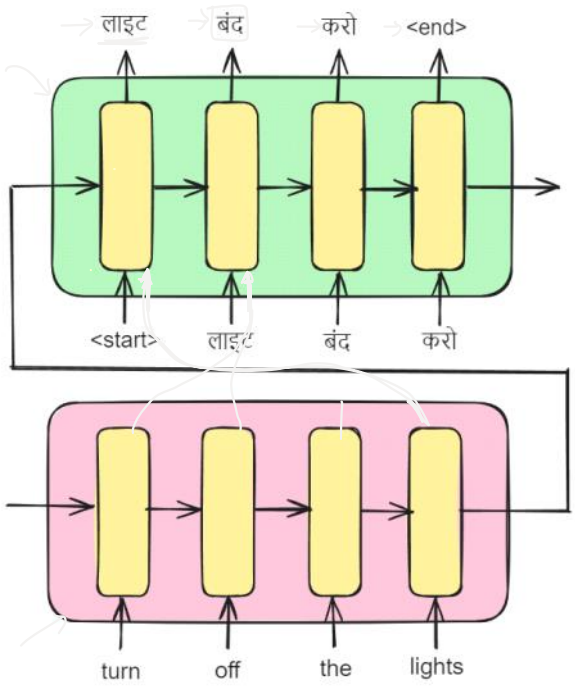


Once upon a time in a small Indian village, a mischievous monkey stole a turban from a sleeping barber, wore it to a wedding, danced with the bewildered guests, accidentally got crowned the 'Banana King' by the local kids, and ended up leading a vibrant, impromptu parade of laughing villagers, cows, and street dogs, all while balancing a stack of mangoes on its head, creating a hilariously unforgettable spectacle and an amusing legend that the village still chuckles about every monsoon season.

The Solution

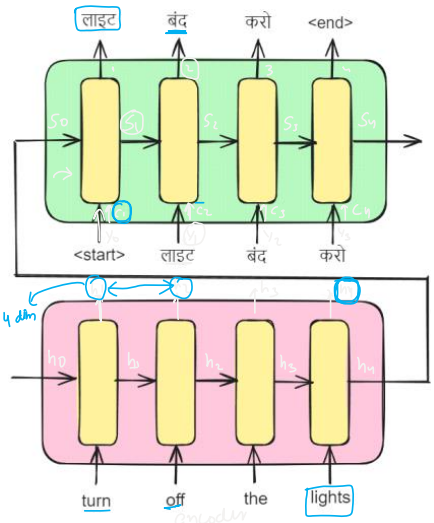
20 December 2023 17:32

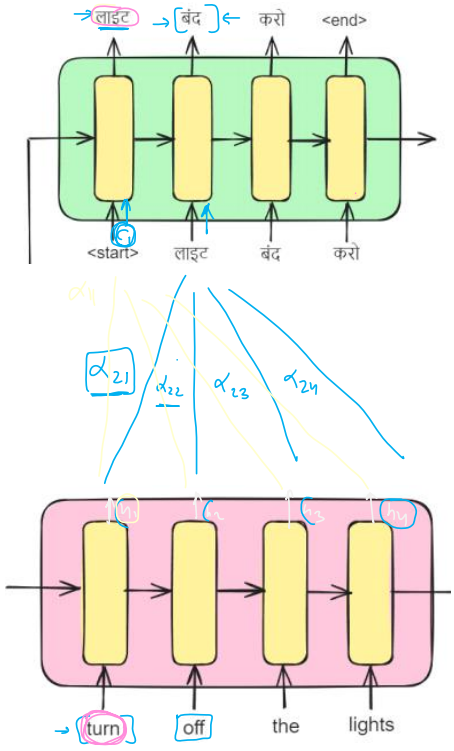
Once upon a time in a small Indian village, a mischievous monkey stole a turban from a sleeping barber, wore it to a wedding, danced with the bewildered guests, accidentally got crowned the 'Banana King' by the local kids, and ended up leading a vibrant, impromptu parade of laughing villagers, cows, and street dogs, all while balancing a stack of mangoes on its head, creating a hilariously unforgettable spectacle and an amusing legend that the village still chuckles about every monsoon season.



The What

21 December 2023 06:04

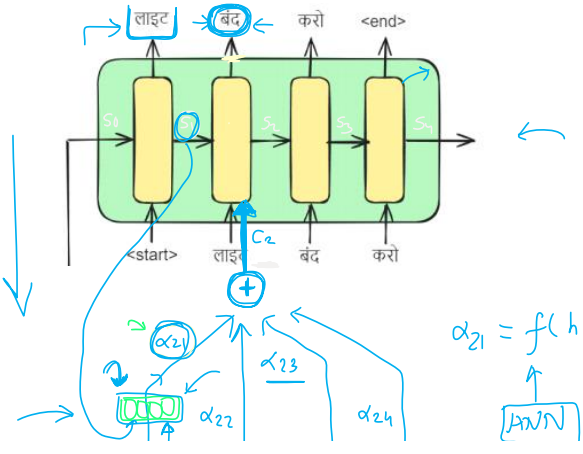




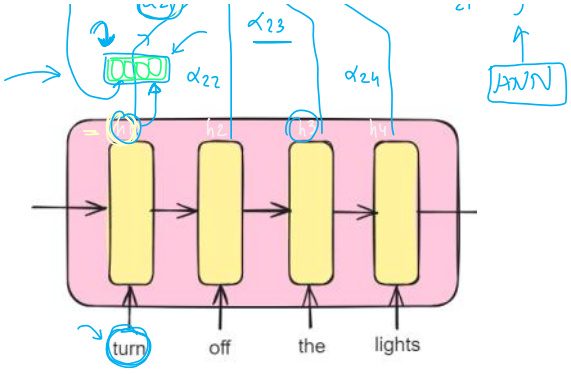
$$c_1 = \alpha_{11} h_1 + \alpha_{12} h_2 + \alpha_{13} h_3 + \alpha_{14} h_4$$

$$c_2 = \alpha_{21} h_1 + \alpha_{22} h_2 + \alpha_{23} h_3 + \alpha_{24} h_4$$

$$c_i = \sum_j \alpha_{ij} h_j$$



$$\alpha_{21} = f(h_1, s_1)$$



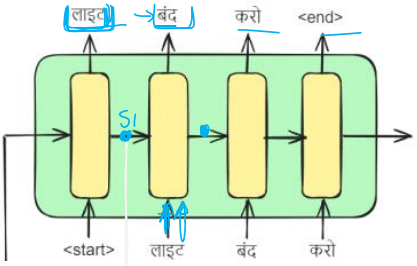
$\alpha_{21} \rightarrow f(h_1, s_1)$

$\alpha_{ij} = f(h_j, s_{i-1})$

ANN

unique function approx

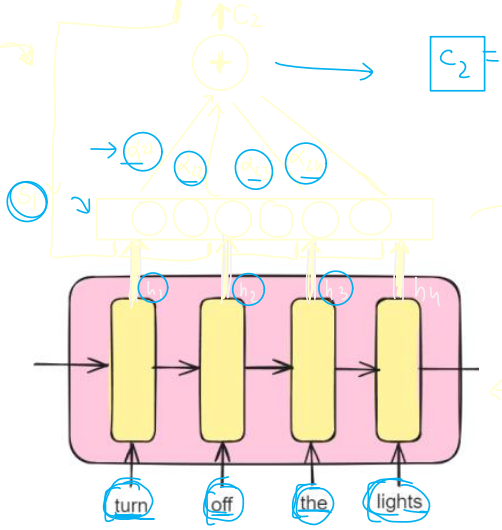
What function?



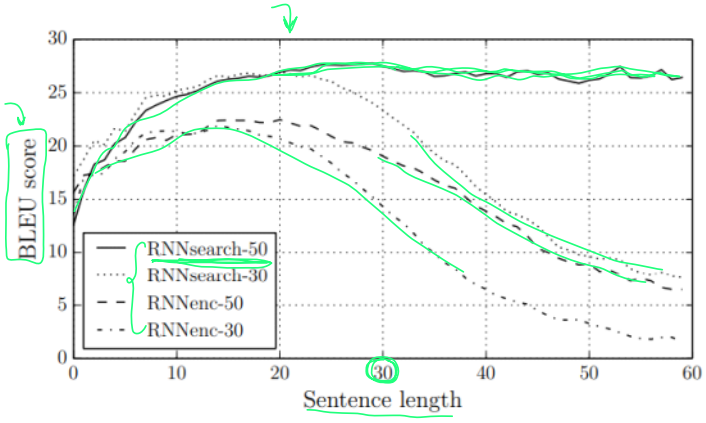
$c_2 = \alpha_{21} h_1 + \alpha_{22} h_2 + \alpha_{23} h_3 + \alpha_{24} h_4$

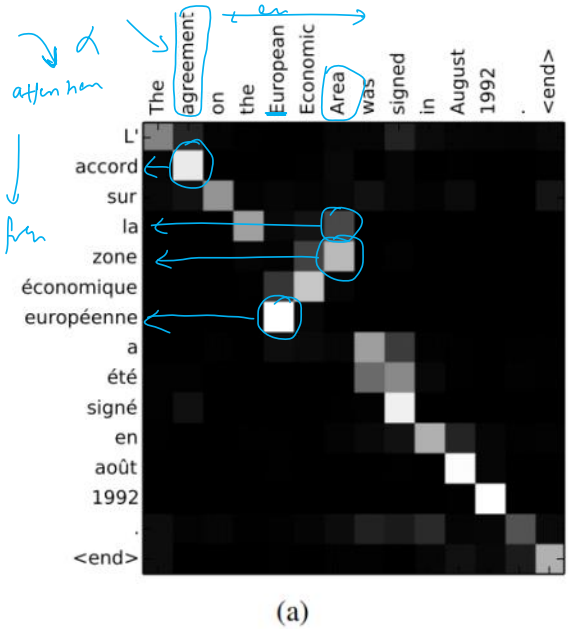
Time Distributed Dense layer

encoder: bi LSTM



eng-french

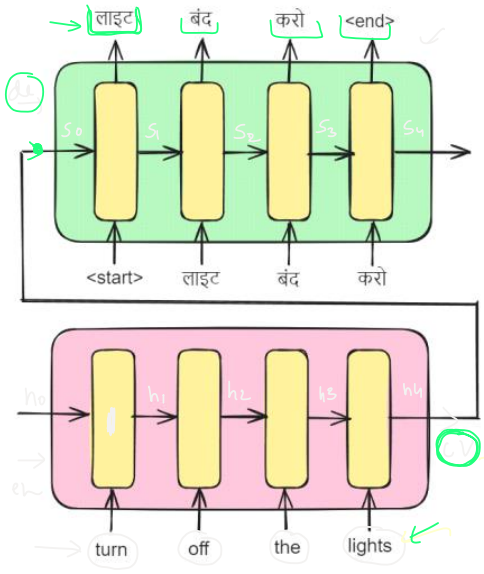




(a)

Recap

16 January 2024 16:10



turn off the lights → लाइट बंद करो

[NMT]

1) Encoder-Decoder

sentence > 30 words

paragraph

document

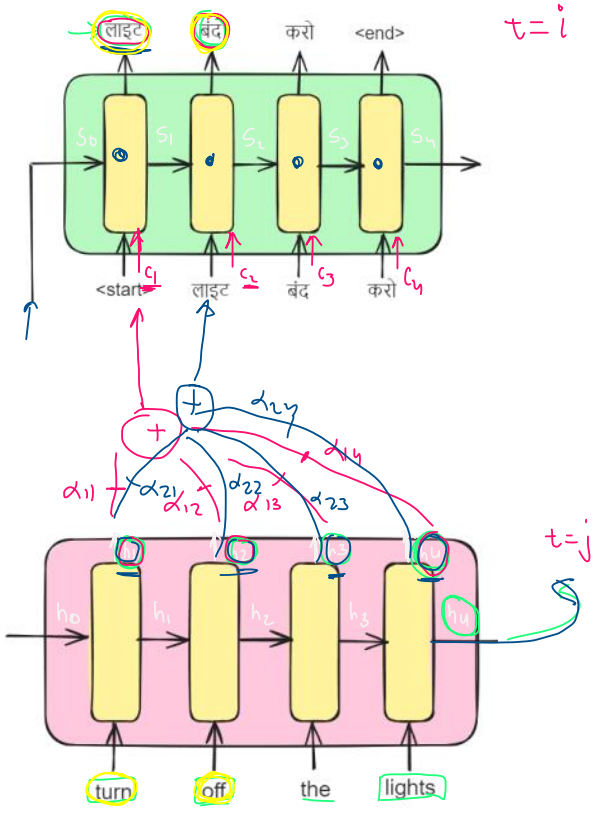
bilstm

stacked lstm

translation

bottleneck → Attention mechanism

set of words



t=i

c_1 c_2 c_3 c_4

4 x 4 = 16

weighted sum

$$c_i = \sum_{j=1}^{4} \alpha_{ij} h_j$$

α → alignment score

$c_1 = \alpha_{11} h_1 + \alpha_{12} h_2 + \alpha_{13} h_3 + \alpha_{14} h_4$

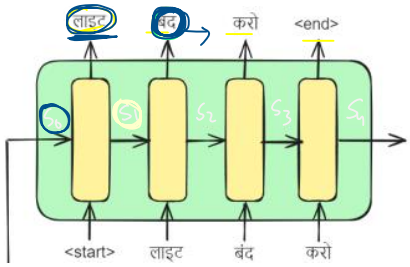
$c_2 = \alpha_{21} h_1 + \alpha_{22} h_2 + \alpha_{23} h_3 + \alpha_{24} h_4$

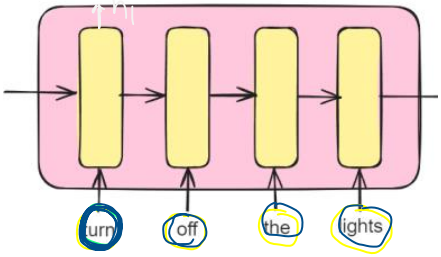
α → find out

- Bahdanau architecture attention
- Luong attention

Bahdanau Attention

16 January 2024 16:11





given

alignment score

$c_i = \sum \alpha_{ij} h_j$

α alignment

α_{11} → लाइट → turn

α_{12} → लाइट → off

decoder → prev hidden state

$\alpha_{11} = f(h_1, s_0)$ $\alpha_{21} = f(h_1, s_1)$

$\alpha_{ij} = f(h_j, s_{i-1})$ → main funct → the encoder hidden

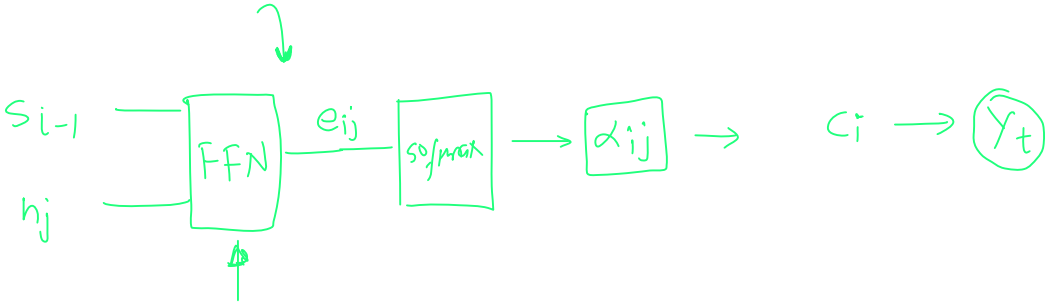
approximate

α_{ij} → FFN → ANN → UFA

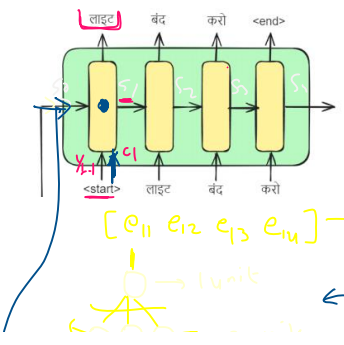
softmax

e_{ij}

s_{i-1} h_j



s_0 y_{t-1} c_1 → dshm → y_t (लाइट) $[s_1]$



$i=1$ $t=2$

encoder sentence

h_1 h_2 h_3 h_4

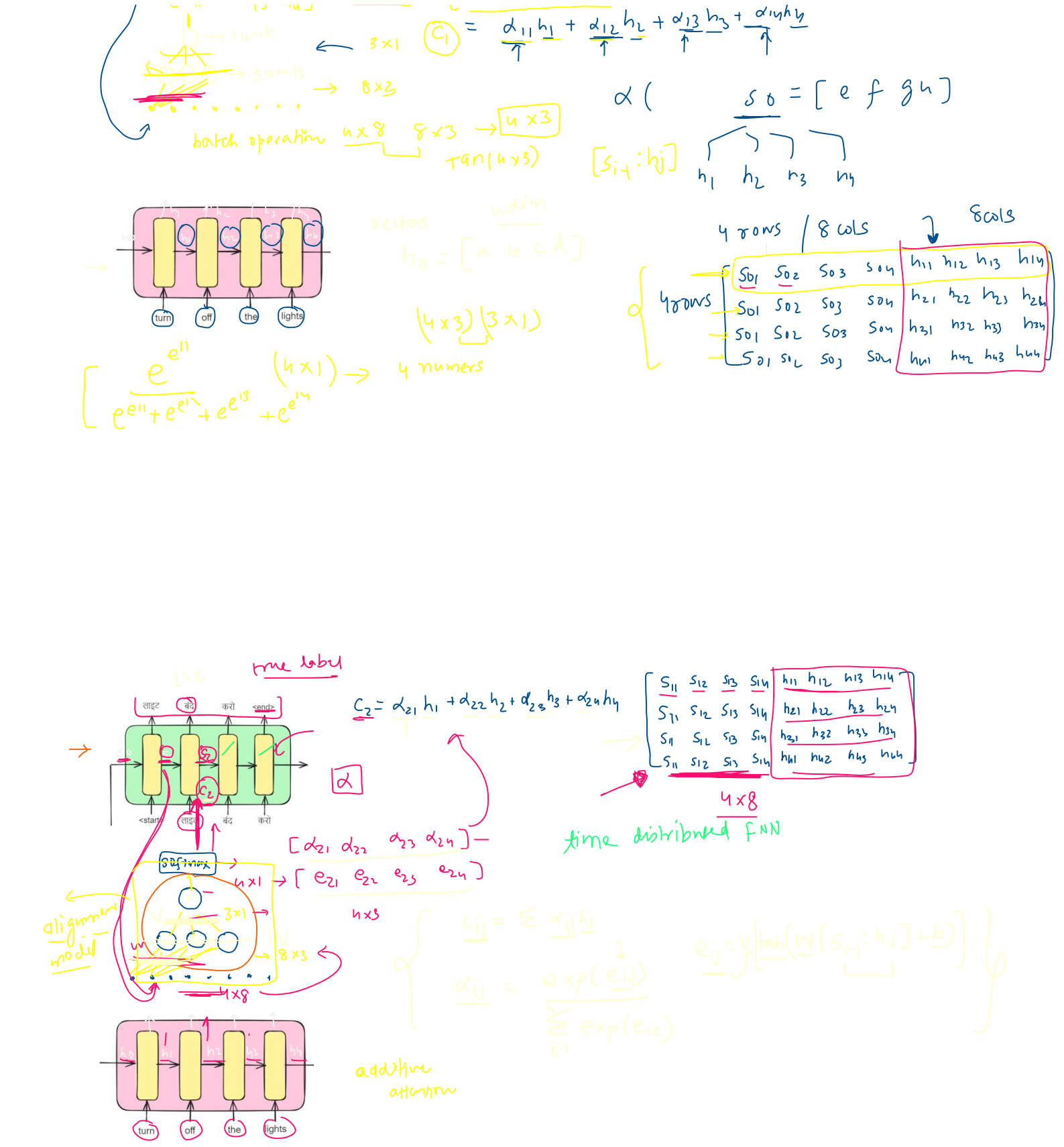
s_{i-1} h_j

$[e_{11}\ e_{12}\ e_{13}\ e_{14}]$ → softmax → $[\alpha_{11}\ \alpha_{12}\ \alpha_{13}\ \alpha_{14}]$

$c_1 = \sum \alpha_{1j} h_j$

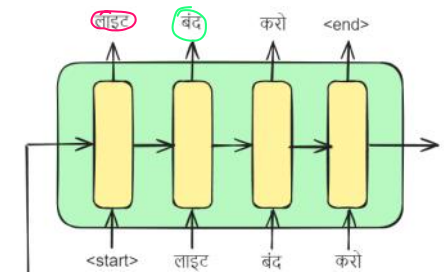
$c_1 = \alpha_{11} h_1 + \alpha_{12} h_2 + \alpha_{13} h_3 + \alpha_{14} h_4$

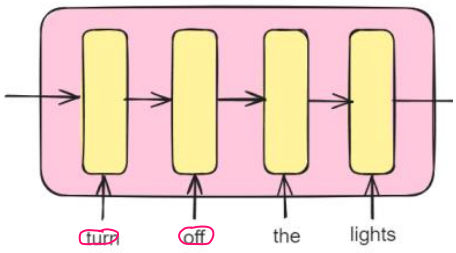
3×1

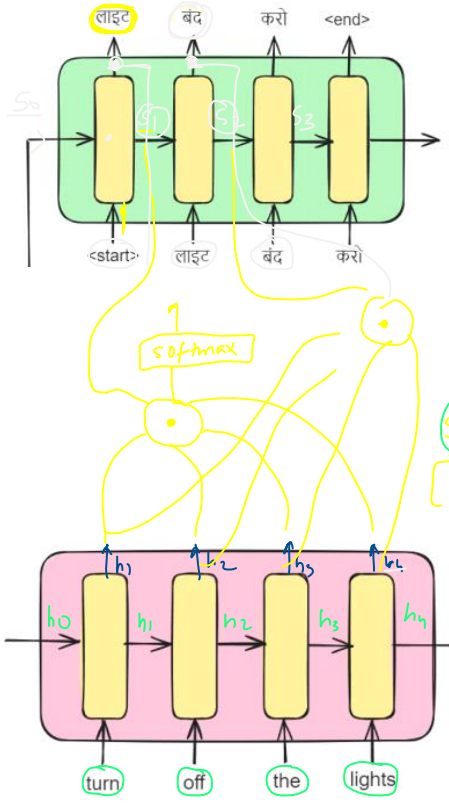

turn
off
the
lights
<start>
<end>
softmax
batch operation
tan(4x3)
4 rows / 8 cols
8 cols
4 rows
time distributed FNN
time lobul
alignment module
additive attention

Luong Attention

17 January 2024 00:09



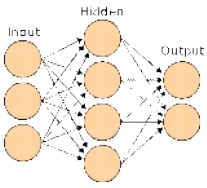




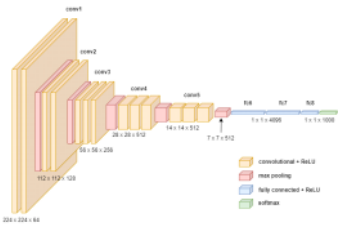
What is Transformer?

27 January 2024 18:41

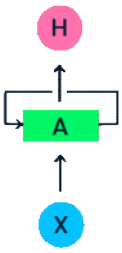
→ self attention scalable



ANN
tabular



CNN
Image

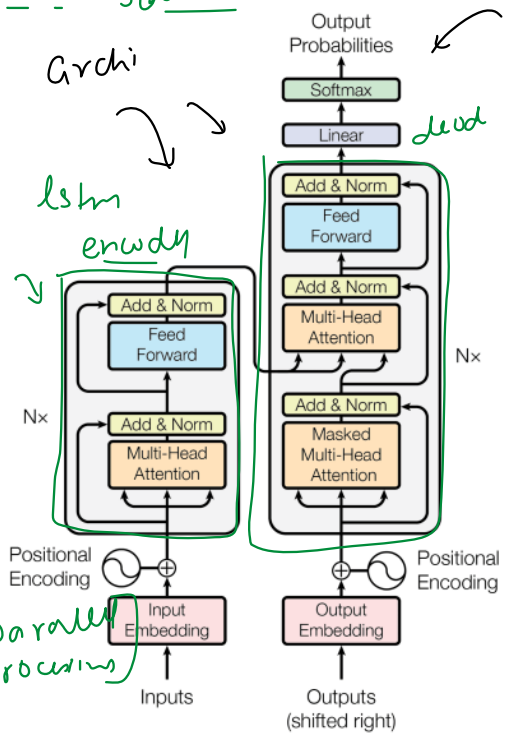


RNN
sequence
↳ text

Archi

lstm

encoder



→ [transformer] → seq2seq task

NN arch

- machine translation
- question ans
- text summary

[parallel processing]

Attention Is All You Need

2017

Google Brain

Deep learning

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

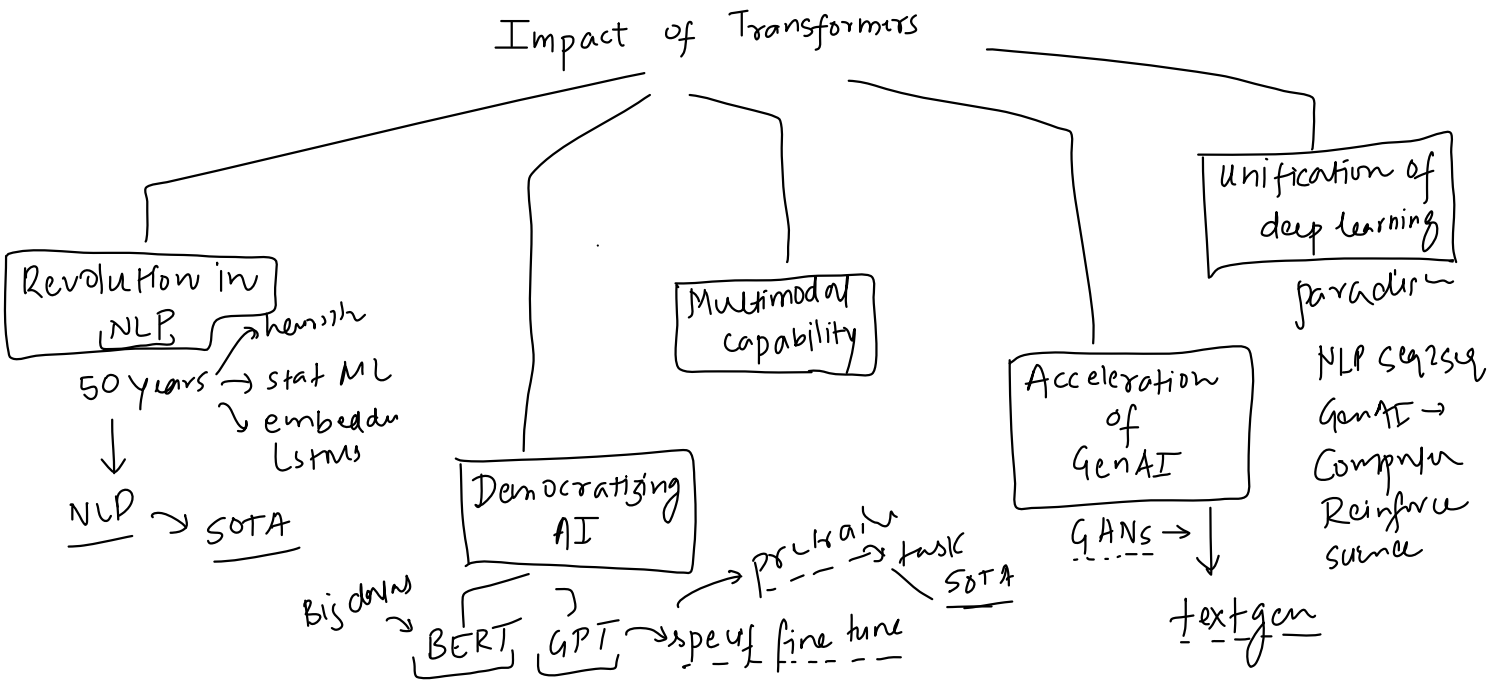
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

Impact of Transformers

27 January 2024 20:03



The Origin Story!

27 January 2024 22:38

2014-15 → seq2seq machine

Sequence to Sequence Learning with Neural Networks

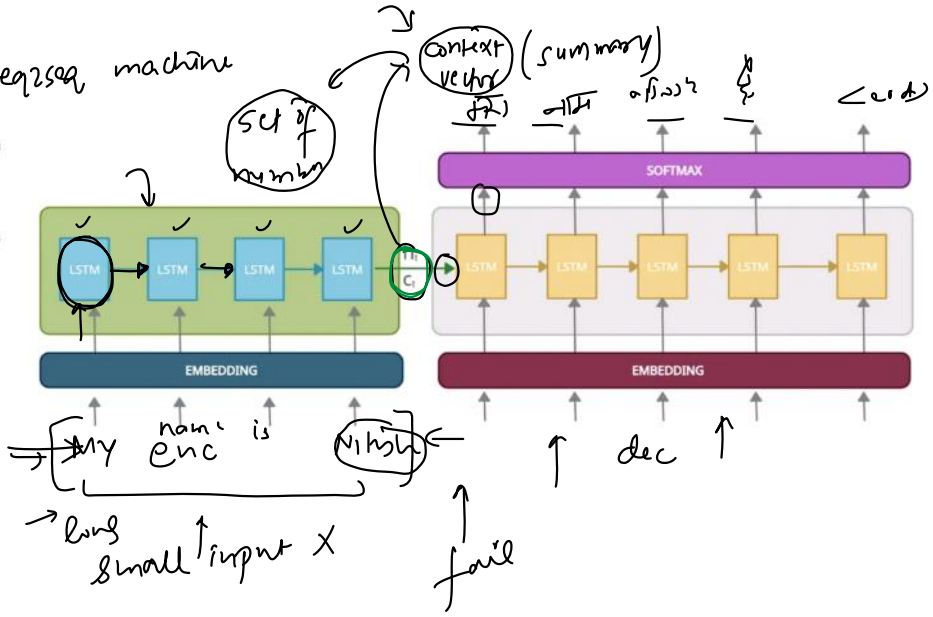
Ilya Sutskever
Google
ilyasu@google.com

Oriol Vinyals
Google
vinyals@google.com

Quoc V. Le
Google
qvl@google.com

Abstract

Deep Neural Networks (DNNs) are powerful models that have achieved excellent performance on difficult learning tasks. Although DNNs work well whenever large labeled training sets are available, they cannot be used to map sequences to sequences. In this paper, we present a general end-to-end approach to sequence learning that makes minimal assumptions on the sequence structure. Our method uses a multilayered Long Short-Term Memory (LSTM) to map the input sequence to a vector of a fixed dimensionality, and then another deep LSTM to decode the target sequence from the vector. Our main result is that on an English to French translation task from the WMT'14 dataset, the translations produced by the LSTM achieve a BLEU score of 34.8 on the entire test set, where the LSTM's BLEU score was penalized on out-of-vocabulary words. Additionally, the LSTM did not have difficulty on long sentences. For comparison, a phrase-based SMT system achieves a BLEU score of 33.3 on the same dataset. When we used the LSTM to rerank the 1000 hypotheses produced by the aforementioned SMT system, its BLEU score increases to 36.5, which is close to the previous best result on this task. The LSTM also learned sensible phrase and sentence representations that are sensitive to word order and are relatively invariant to the active and the passive voice. Finally, we found that reversing the order of the words in all source sentences (but not target sentences) improved the LSTM's performance markedly, because doing so introduced many short term dependencies between the source and the target sentence which made the optimization problem easier.



NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

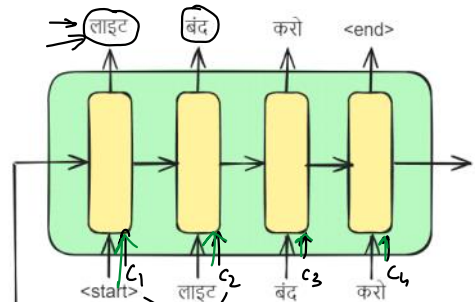
KyungHyun Cho Yoshua Bengio*
Université de Montréal

→ attention → concept

translation sent >30 words

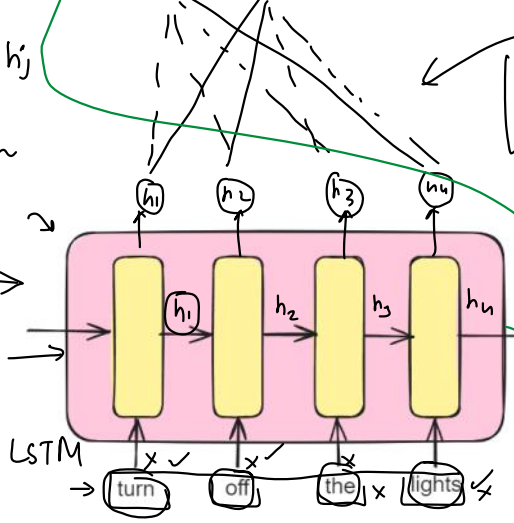
ABSTRACT

Neural machine translation is a recently proposed approach to machine translation. Unlike the traditional statistical machine translation, the neural machine translation aims at building a single neural network that can be jointly tuned to maximize the translation performance. The models proposed recently for neural machine translation often belong to a family of encoder–decoders and encode a source sentence into a fixed-length vector from which a decoder generates a translation. In this paper, we conjecture that the use of a fixed-length vector is a bottleneck in improving the performance of this basic encoder–decoder architecture, and propose to extend this by allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word, without having to form these parts as a hard segment explicitly. With this new approach, we achieve a translation performance comparable to the existing state-of-the-art phrase-based system on the task of English-to-French translation. Furthermore, qualitative analysis reveals that the (soft-)alignments found by the model agree well with our intuition.



$c_i = \sum \alpha_{ij} h_j$

attention weights



scratch model → time and effort → dataset → money

sequential training → slow → huge dataset

Transfer learning → CNN → finetune → dog vs cat class → ImageNET

pretrain fine tuning

research → incremental → stream → time travel

→ RNN/self → parallely
→ stable
→ hyperfine

NLP → BERT, GPT → finetune

Attention Is All You Need → 2017

Output Probabilities

Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

Jakob Uszkoreit*
Google Research
usz@google.com

Llion Jones*
Google Research
llion@google.com

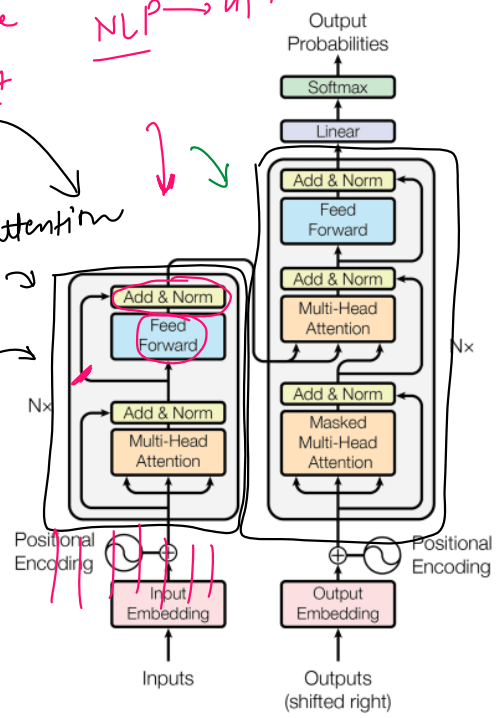
Aidan N. Gomez* †
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser*
Google Brain
lukaszkaiser@google.com

Illia Polosukhin* ‡
illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.



The Timeline

28 January 2024 00:55

2000 - 2014 → RNNs / LSTMs

2014 → Attention

2017 → Transformer

2018 → BERT / GPT (Transfer learning)

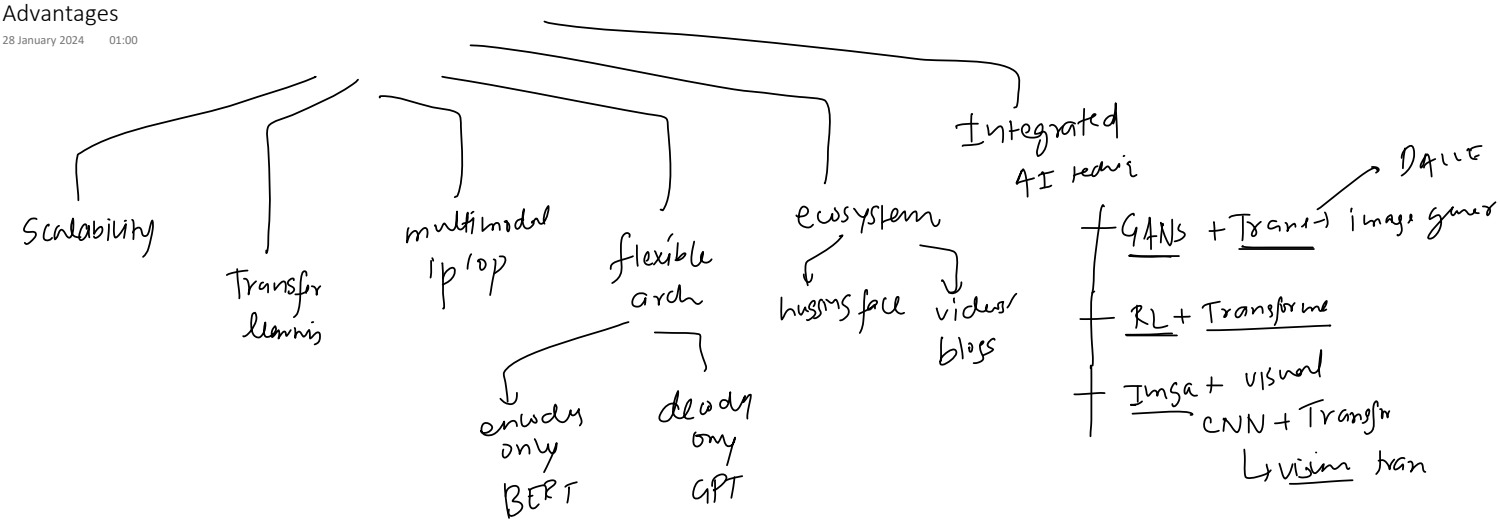
2018 - 2020 → Vision Transformer / Alphafold-2

2021 → Gen AI

2022 → ChatGPT / Stable Diffusion

Advantages

28 January 2024 01:00

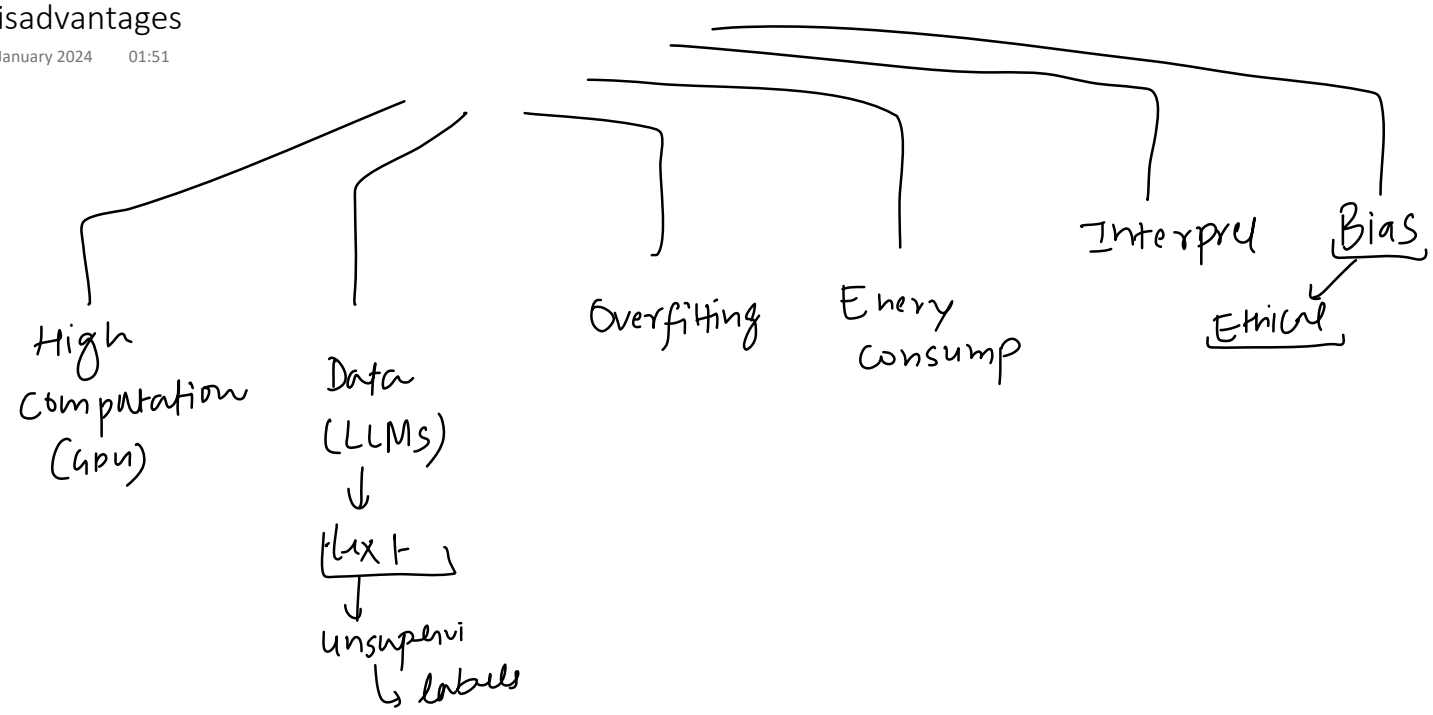


Famous Applications

28 January 2024 01:17

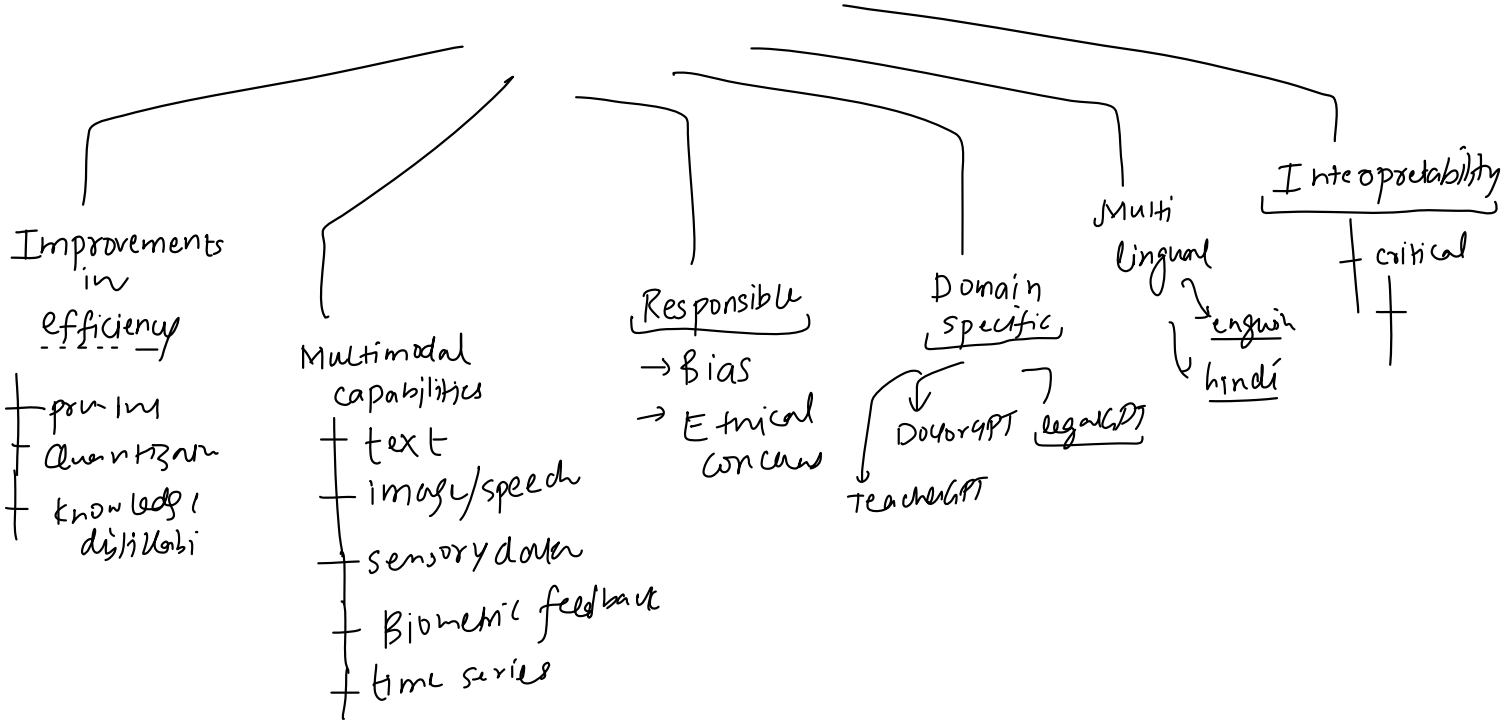
Disadvantages

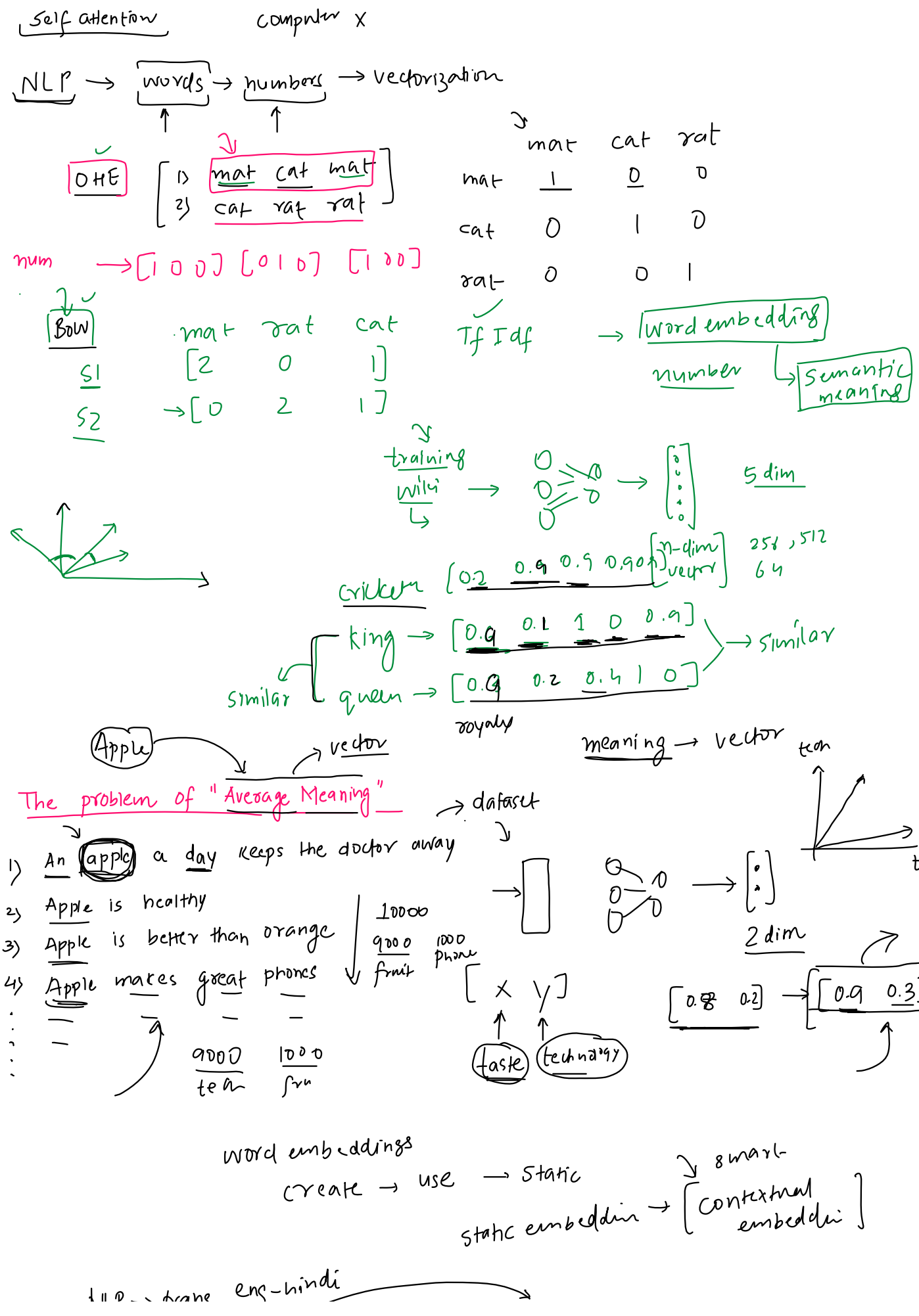
28 January 2024 01:51

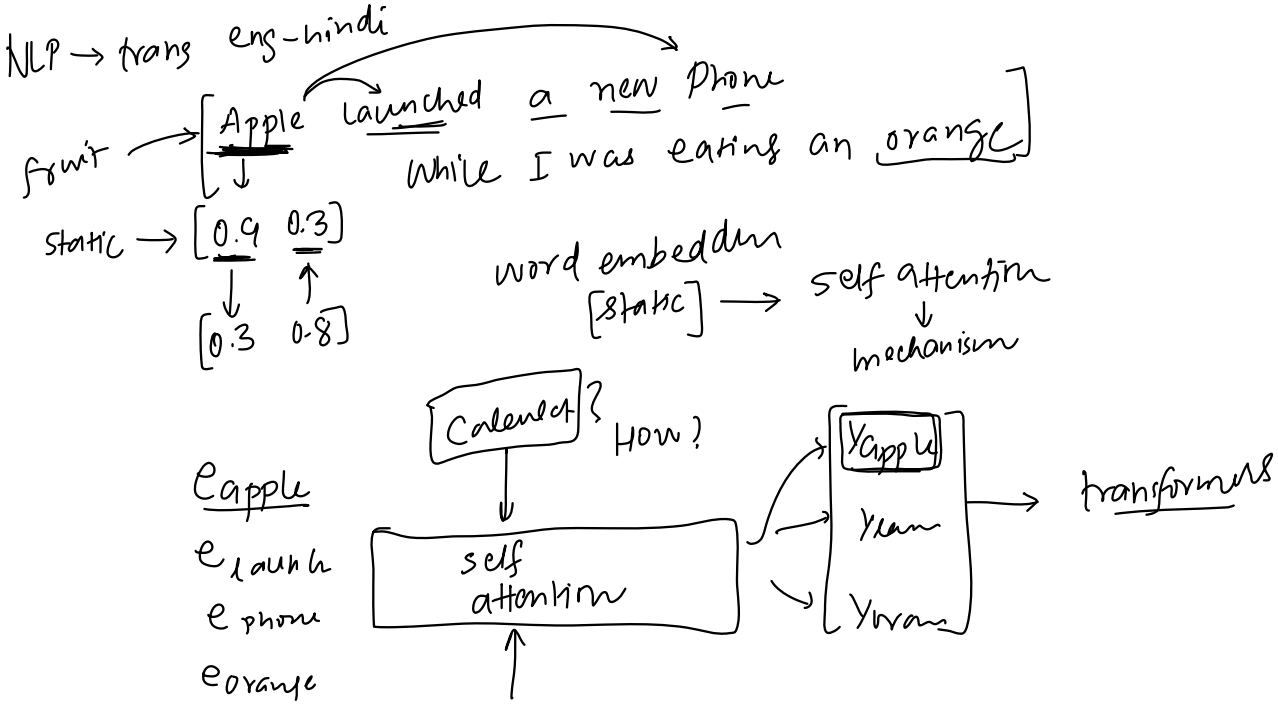


Future

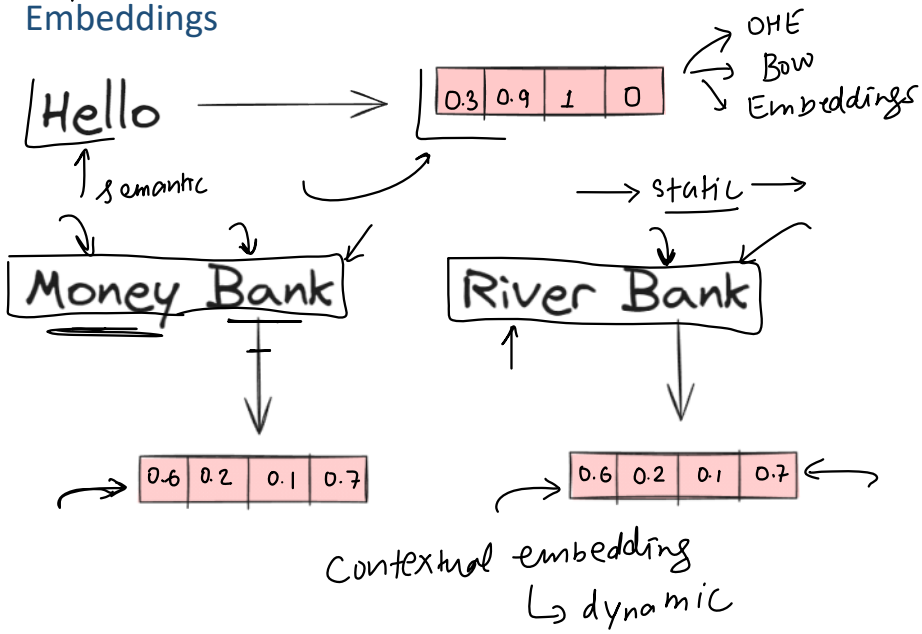
28 January 2024 02:09

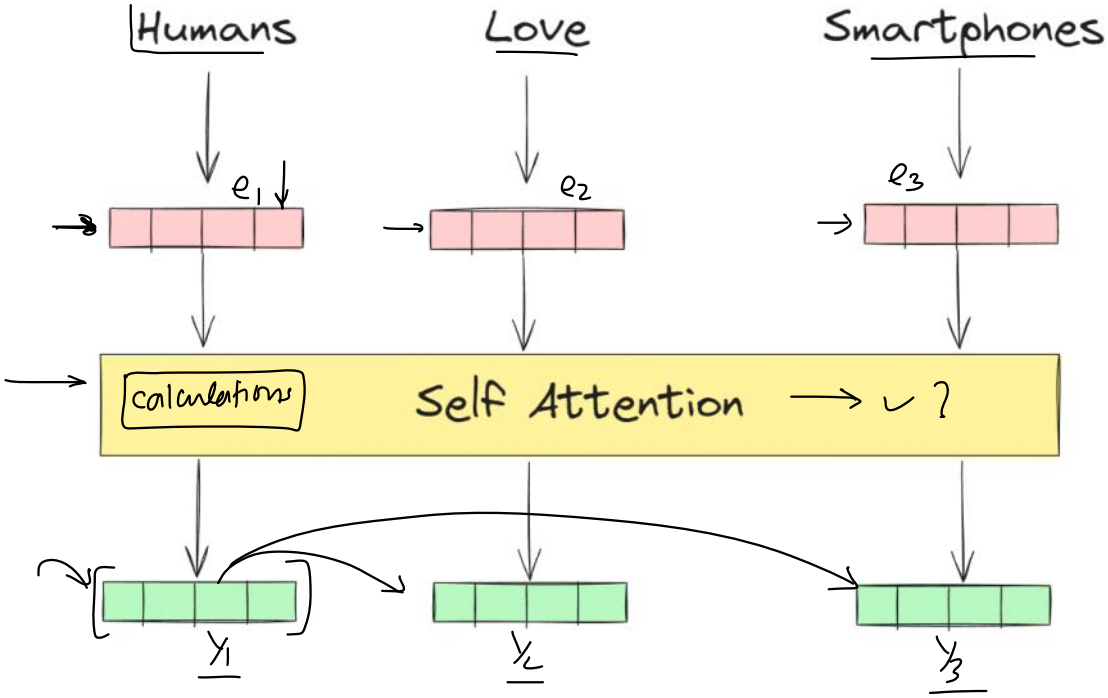



Self attention
computer X
NLP → words → numbers → vectorization
OHE
1) mat cat mat
2) cat rat rat
num → [1 0 0] [0 1 0] [1 0 0]
mat cat rat
mat 1 0 0
cat 0 1 0
rat 0 0 1
BOW
mat rat cat
S1 [2 0 1]
S2 → [0 2 1]
Tf Idf → word embedding
number → semantic meaning
training
wiki
5 dim
n-dim vector
258, 512
64
Cricket [0.2 0.9 0.5 0.90]
king → [0.9 0.1 1 0 0.9]
queen → [0.9 0.2 0.4 1 0]
similar
similar
royalty
Apple → vector
meaning → vector
The problem of "Average Meaning"
dataset
1) An apple a day keeps the doctor away
2) Apple is healthy
3) Apple is better than orange
4) Apple makes great phones
9000 tea
1000 fruit
10000
9000 fruit
1000 phone
[x y]
taste
technology
2 dim
[0.8 0.2] → [0.9 0.3]
word embeddings
create → use → static
static embeddin → 8 mark- contextual embeddi
eng-hindi


NLP → trans eng-hindi
fruit
Apple launched a new phone
While I was eating an orange
static → [0.9 0.3]
[0.3 0.8]
word embeddin [static] → self attention mechanism
Calendar? How?
eapple
elaunch
ephone
eorange
self attention
yapple
yfruit
yorange
transformers

Embeddings


Hello
0.3 0.9 1 0
OHE
Bow
Embeddings
semantic
static
Money Bank
River Bank
0.6 0.2 0.1 0.7
0.6 0.2 0.1 0.7
Contextual embedding
↳ dynamic
Self Attention
embedding → dynamic contextual


Humans
Love
Smartphones
e1
e2
e3
calculations
Self Attention
y1
y2
y3

First Principle Approach

04 February 2024 17:55

money bank grows

bank → bank

bank → 0.3 money + 0.7 bank + 0.1 grows

river bank flows

bank = 0.5 river + 0.4 bank + 0.1 flows

S1 word embe

money = 0.7 money + 0.2 bank + 0.1 grows

bank = 0.25 money + 0.7 bank + 0.05 grows

grows = 0.1 money + 0.2 bank + 0.7 grows

S2

river = 0.8 river + 0.15 bank + 0.05 flows

bank = 0.2 river + 0.78 bank + 0.02 flows

flows = 0.4 river + 0.01 bank + 0.59 flows

n dim

$e^{(new)}_{money} = 0.7\, e_{money} + 0.2\, e_{bank} + 0.1\, e_{grows}$

$e^{(new)}_{bank} = 0.25\, e_{money} + 0.7\, e_{bank} + 0.05\, e_{grows}$

$e^{(new)}_{grows} = 0.1\, e_{money} + 0.2\, e_{bank} + 0.7\, e_{grows}$

similarity

e_{money} e_{money}

e_{money} e_{bank}

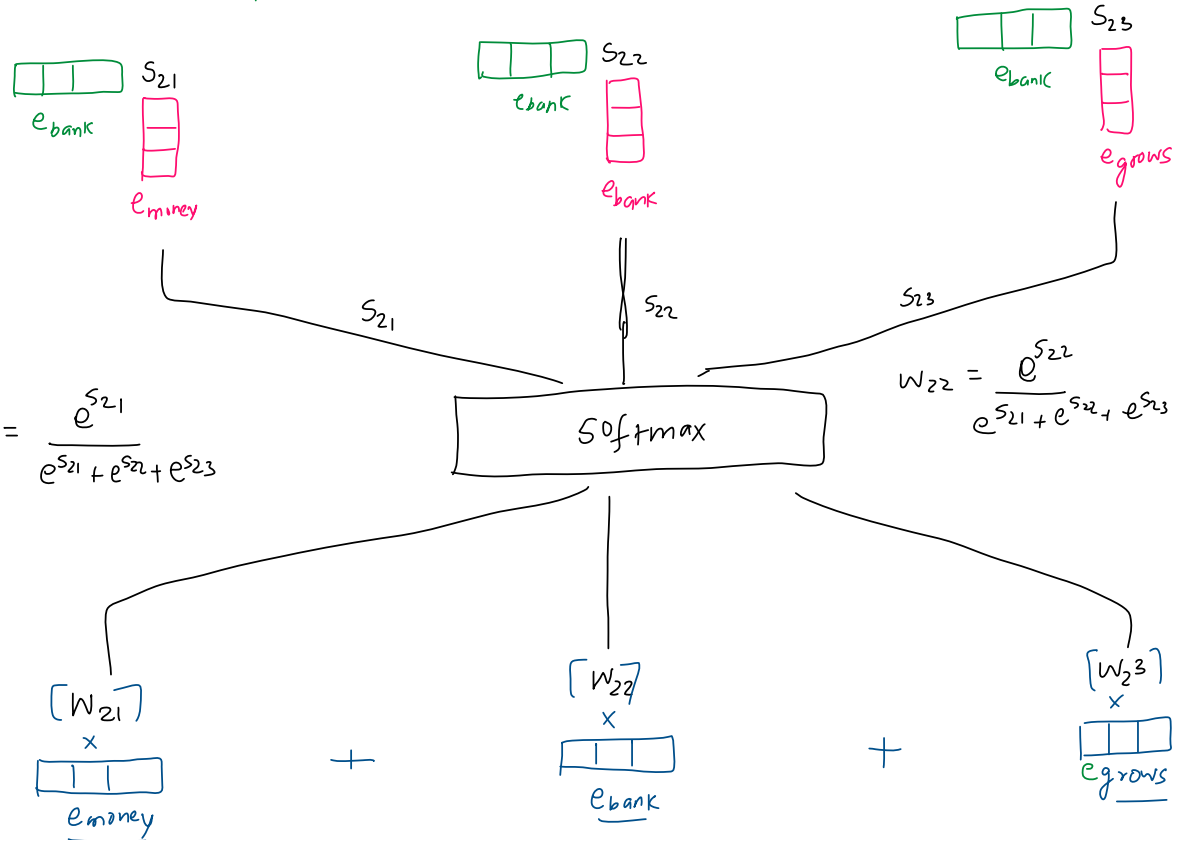
e_{money} e_{grow}

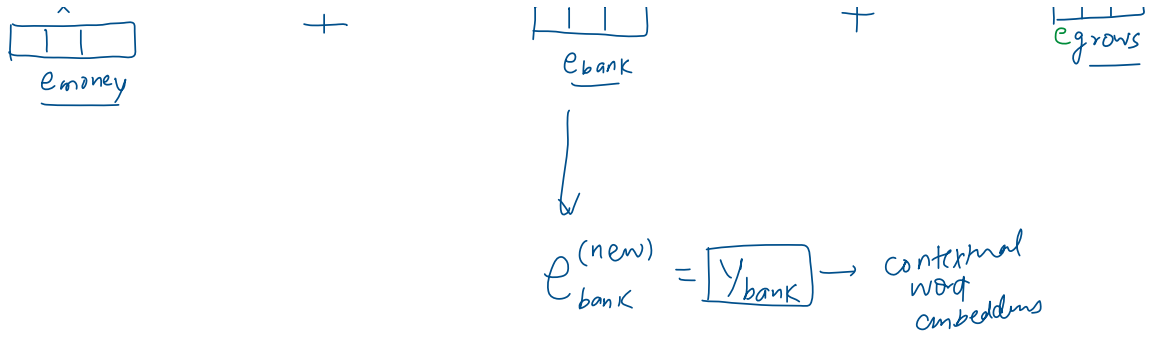
e_{grows} e_{money}

dot products

(1) normalised

$$e^{(new)}_{bank} = [e_{bank} \cdot e^{T}_{money}]\, e_{money} + [e_{bank} \cdot e^{T}_{bank}]\, e_{bank} + [e_{bank} \cdot e^{T}_{grows}]\, e_{grows}$$

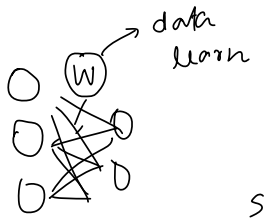




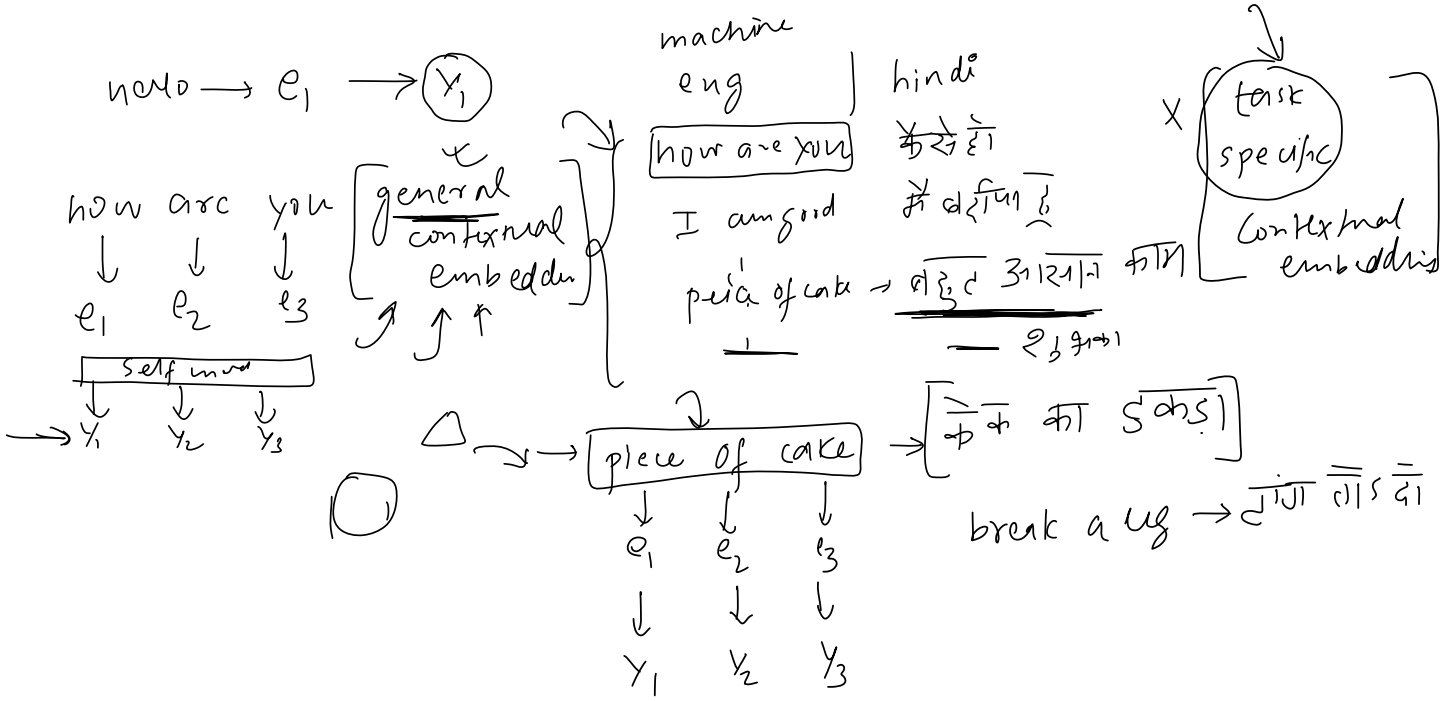
Points to consider

→ This operation is a parallel operation

→ There are no parameters involved

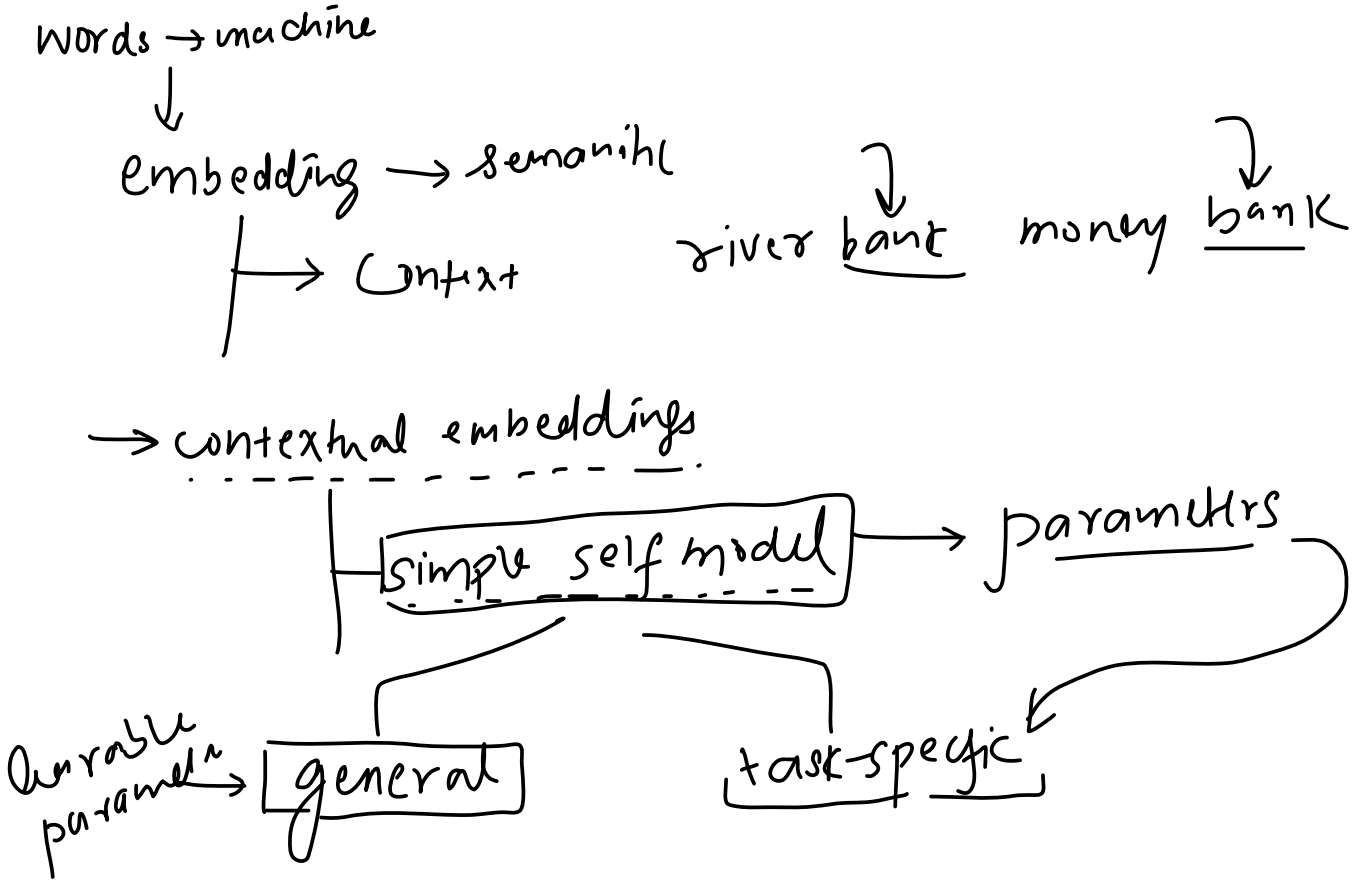


stuf model



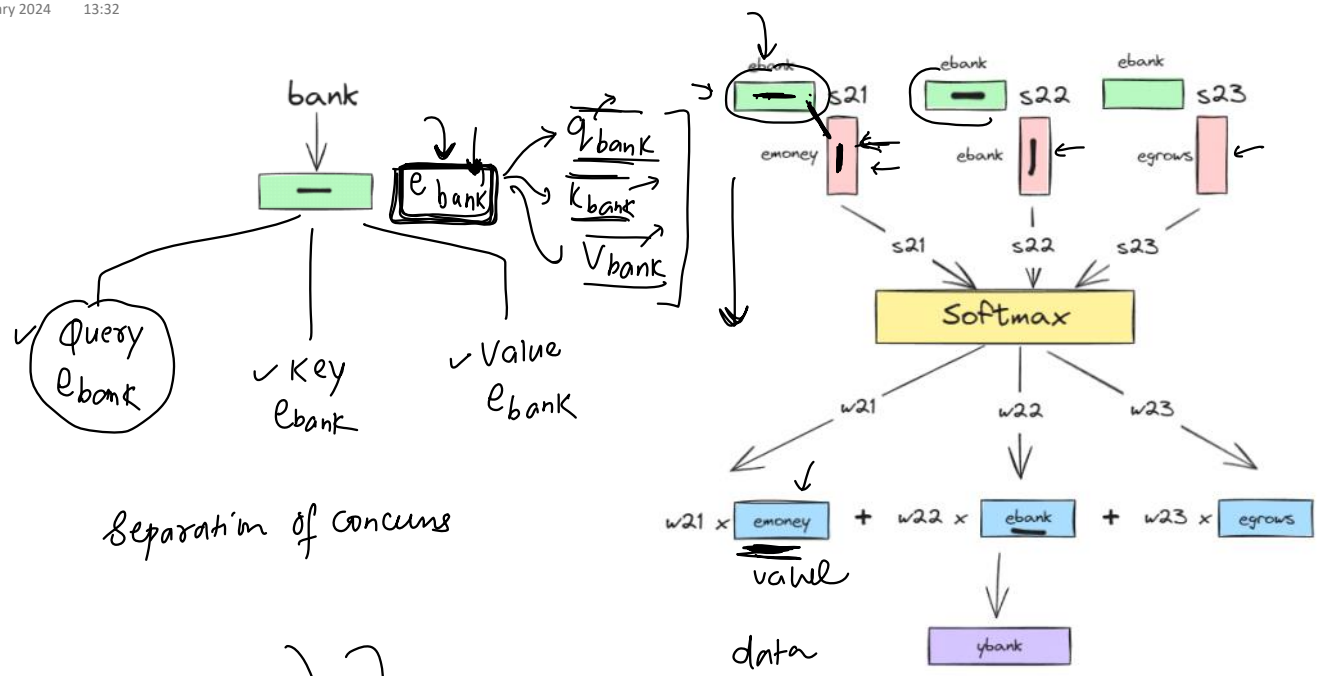
Progress

06 February 2024 00:42

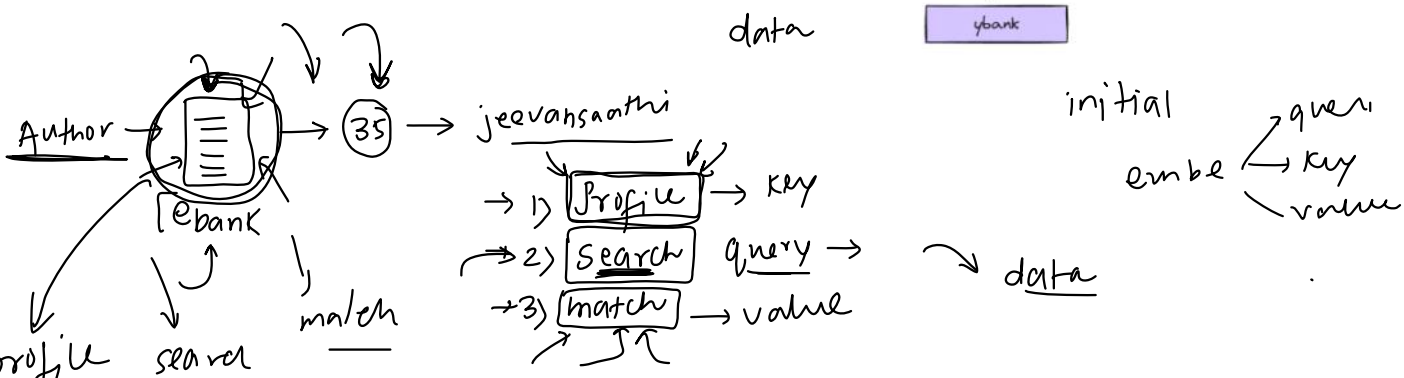


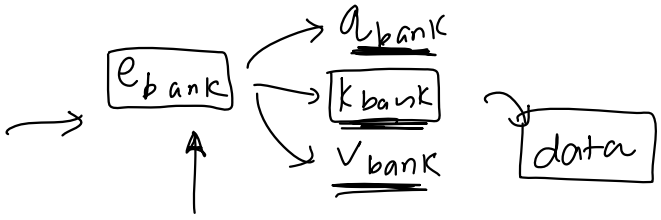
Query, Key & Value Vectors

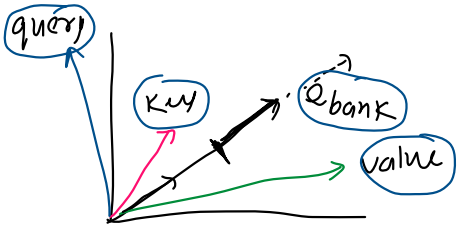
06 February 2024 13:32



Separation of concerns



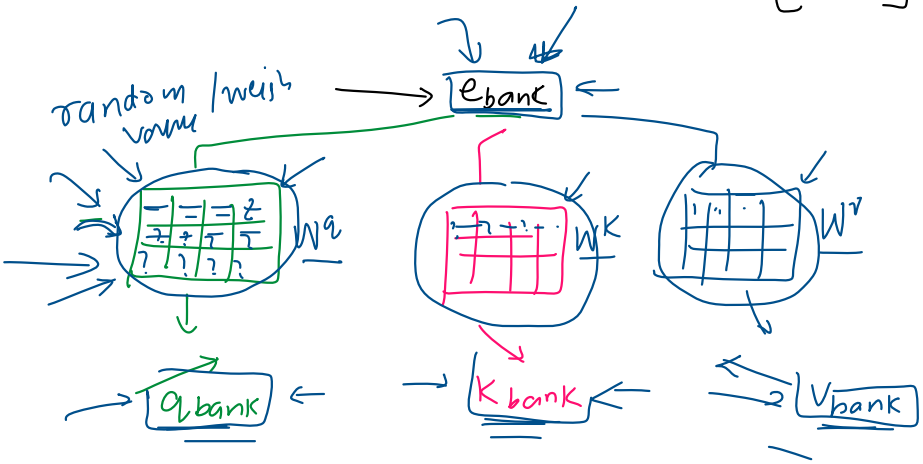




d magnitude (scaling)

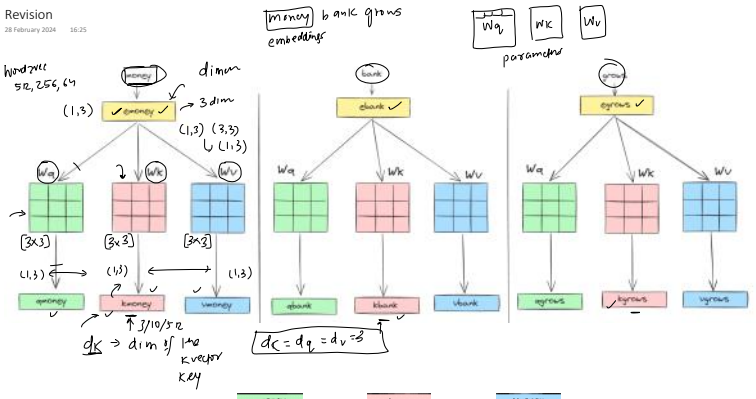
→ linear transform

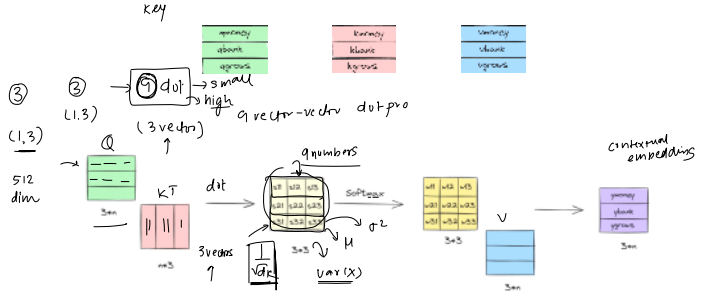
→ [ ]

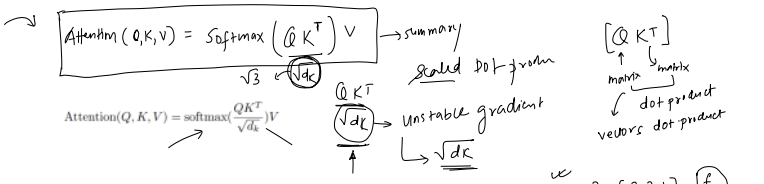


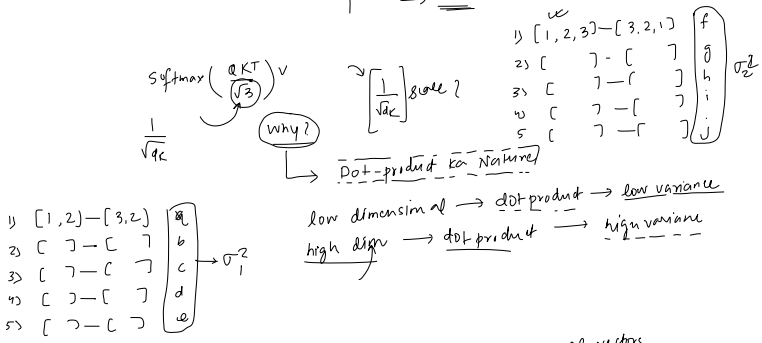
3 x 3 → dot → (9) values

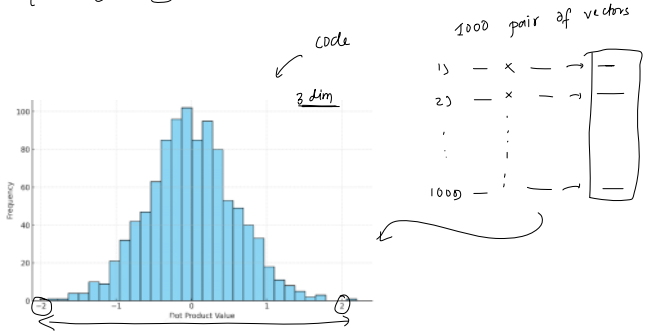
Revision
28 February 2024 16:25

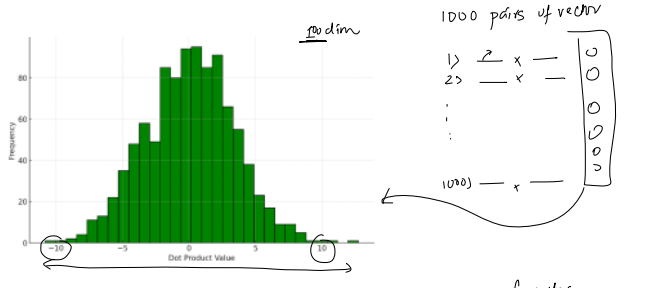


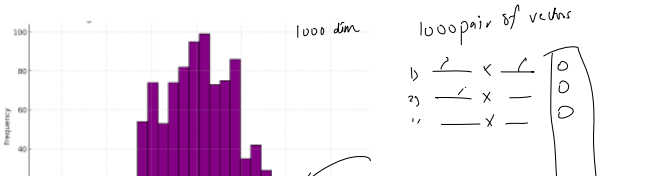


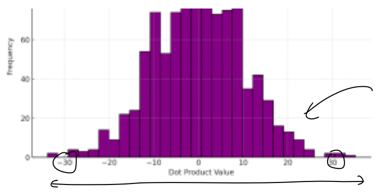


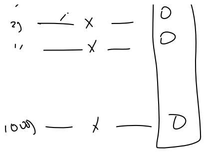


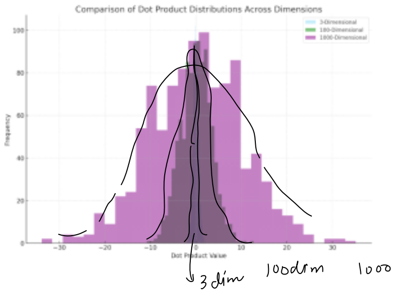




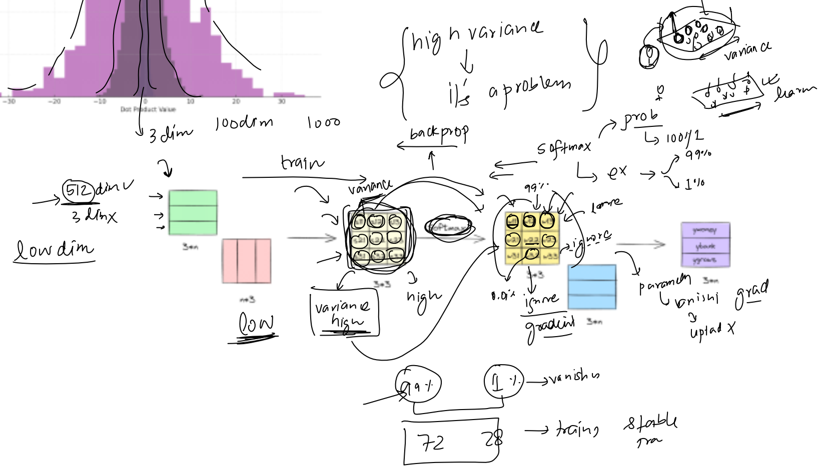








vector dim → variance ↳ high



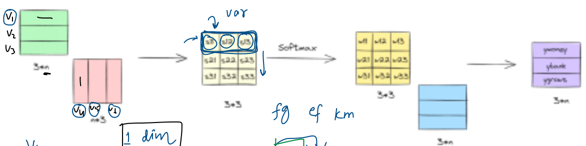
each c

{ X → Var(X)
Y → cX ↳ c²Var(Y) }

If you have a random variable X with a variance of $\text{Var}(X)$, and you create a new variable Y by scaling X with a constant c, so that $Y = cX$, the variance of Y ($\text{Var}(Y)$) is related to the variance of X by the square of the scaling factor c. Mathematically, this relationship is expressed as:

$$\text{Var}(Y) = c^2\text{Var}(X)$$

dim ↑ ↳ variance ↑ → math quantify



V_1 — V_4, V_5, V_6

2dim

1 dim [a] 3

[b] 2, [c] 1, [d] 4 → ab 6, ac 3, ad 12 → random var → X

variance (sample) → expected var → Var(X)

fg of km

X → Var(X)

cX → c²Var(X)

Y → Var(Y) → 2Var(X)

$\frac{Y}{\sqrt{2}} \to \frac{1}{2}\text{Var}(Y) \to \frac{1}{2}\cdot 2\,\text{Var}(X) \to \text{Var}(X)$

[a b] [c d] → $\frac{ac+bd}{\sqrt{2}}$; [e f] → $\frac{ae+bf}{\sqrt{2}}$; [g h] → $\frac{ag+bh}{\sqrt{2}}$ → Y → Var(Y)

2 Var(X)

Var(Y) ≃ 2Var(X)

Var(Y) > Var(X) Yes No

2dim $\frac{1}{\sqrt{d_k}}$

3 dim [a b c] [d e f] → $\frac{ad+be+cf}{\sqrt{3}}$; [g h i] → $\frac{ag+bh+ci}{\sqrt{3}}$; [k l m] → $\frac{ak+bl+cm}{\sqrt{3}}$ → Z

Var(Z) > Var(Y) > Var(X)

Var(Z) ≃ 3Var(X)

Z → Var(Z) ≃ 3 Var(X)

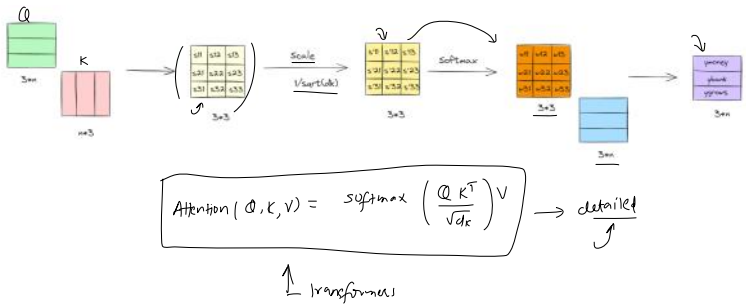
$\frac{1}{\sqrt{3}}Z \to \left(\frac{1}{\sqrt{3}}\right)^2\text{Var}(Z) = \frac{1}{3}\cdot 3\,\text{Var}(X)$

d dim → d Var(X) ↳ variance dim = 1

dim → variance linear

1 dim → Var(X) → Var(X)
2 dim → 2Var(X) → Var(X)
3 dim → 3 Var(X) → Var(X)
⋮
d dim → d Var(X) → Var(X)

[- - - - -] → Var(X) →
d dim



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

→ detailed

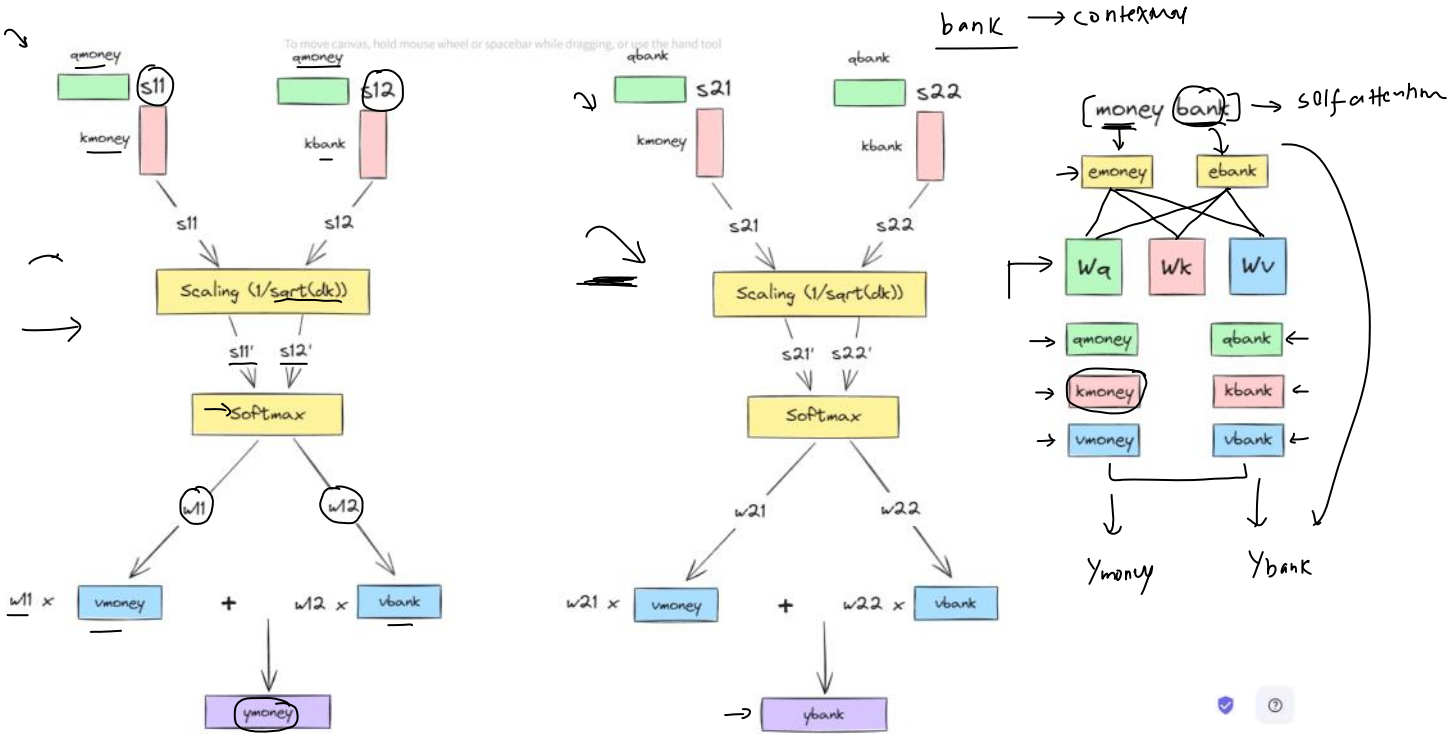
↑ transformers

What is d_k

28 February 2024 16:59

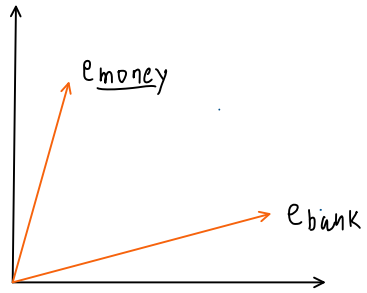
Recep

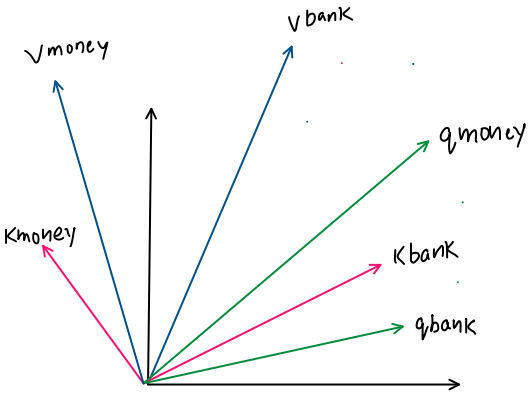
08 March 2024 15:14



Geometric Intuition

08 March 2024 15:16



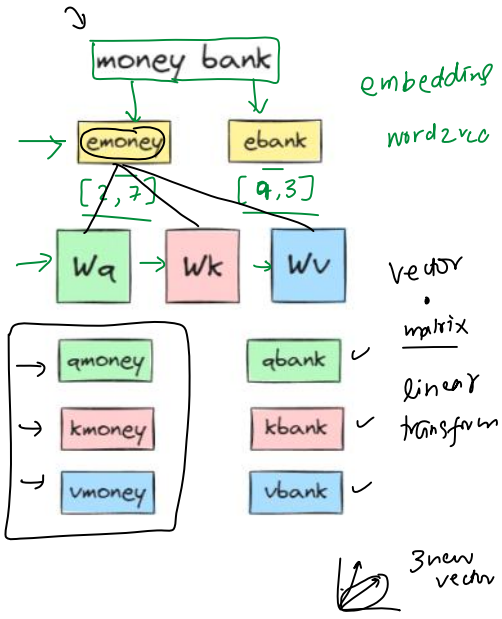


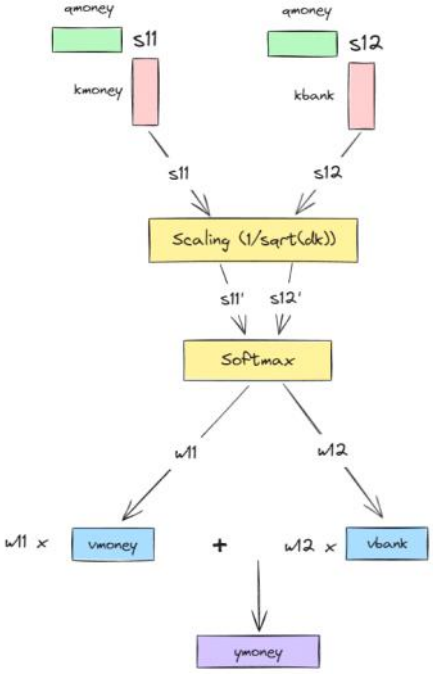
$W_q = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$ (2x2)

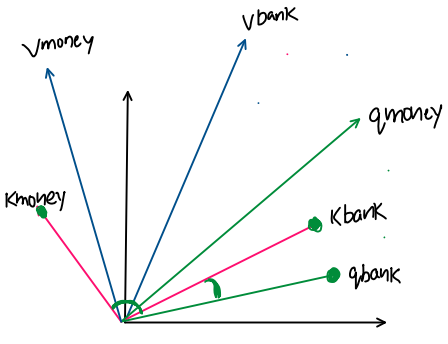
$W_k = \begin{bmatrix} 3 & 4 \\ 5 & 1 \end{bmatrix}$ (2x2)

$W_v = \begin{bmatrix} 4 & 1 \\ 2 & 1 \end{bmatrix}$ (2x2)

* All values are hypothetical







[Dot Product]

$S_{21} = 10$ $\quad S_{22} = \frac{32}{\sqrt{2}}$

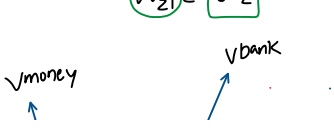
[Scaling]

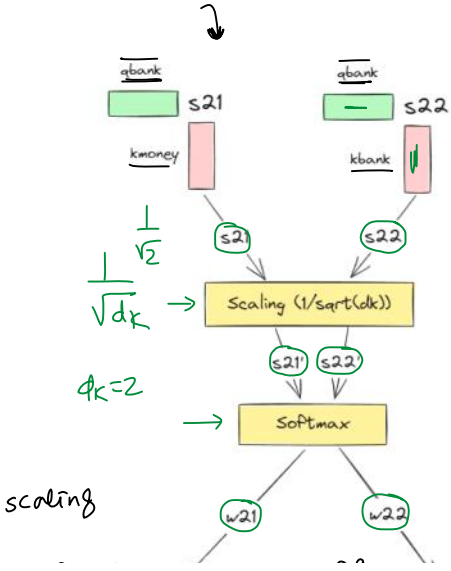
$S'_{21} = \frac{10}{\sqrt{2}} = 7.09$ $\quad S'_{22} = 22.69$

[Softmax]

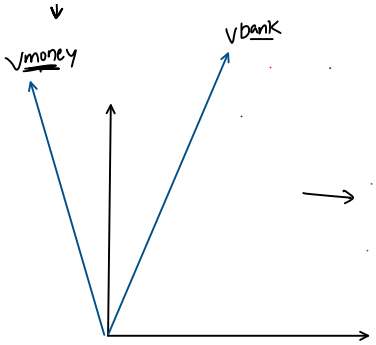
$W_{21} = 0.2$ $\quad W_{22} = 0.8$

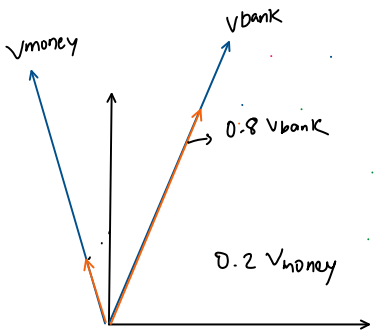
1

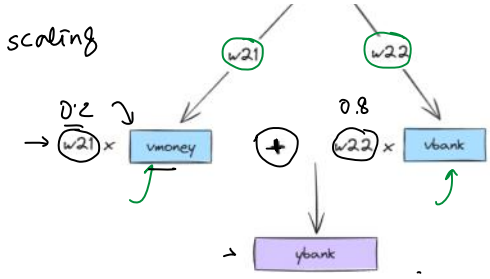


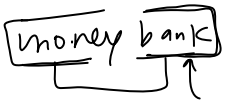


scaling

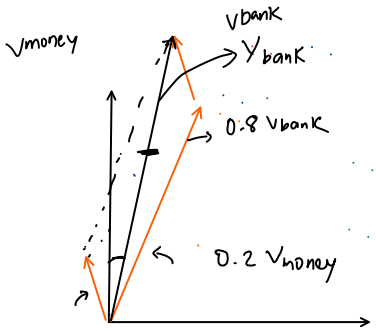


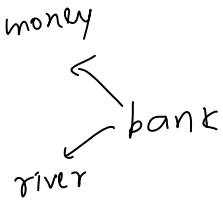


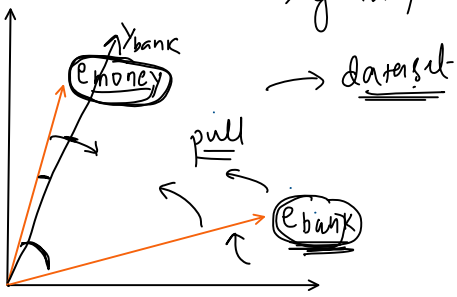




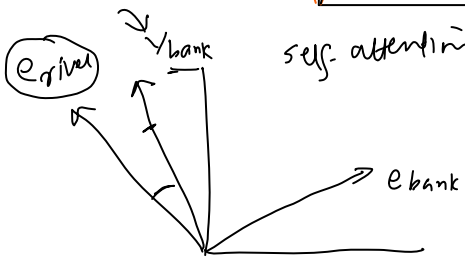
self attention
↳ gravity







dataset



self attention

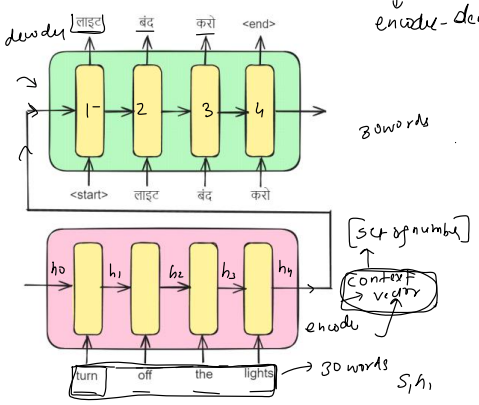
river bank

Recap of Attention

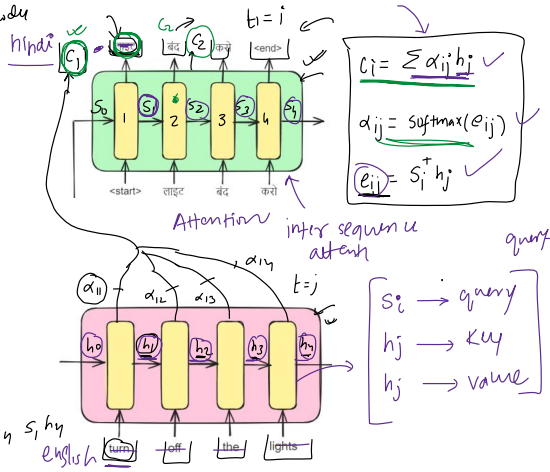
11 March 2024 17:09

long trans → NN

encoder-decoder



Attention



$c_i = \sum \alpha_{ij} h_j$ ✓

$\alpha_{ij} = \text{softmax}(e_{ij})$ ✓

$e_{ij} = s_i^T h_j$ ✓

s_i → query

h_j → key

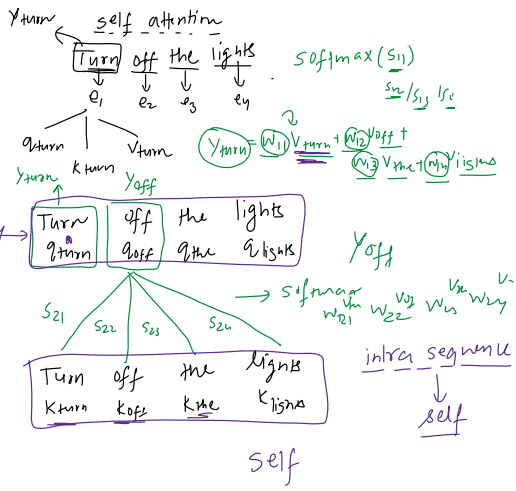
h_j → value

e_{11} $s_1 h_1$

e_{12} $s_1 h_2$

e_{13} $s_1 h_3$

e_{14} $s_1 h_4$



softmax(s11)

$Y_{turn} = W_{11} V_{turn} + W_{12} V_{off} + W_{13} V_{the} + W_{14} V_{lights}$

intra sequence

self

$c_i = \sum \alpha_{ij} h_j$ (weights)

$\alpha_{ij} = \text{softmax}(e_{ij})$ (alignment score)

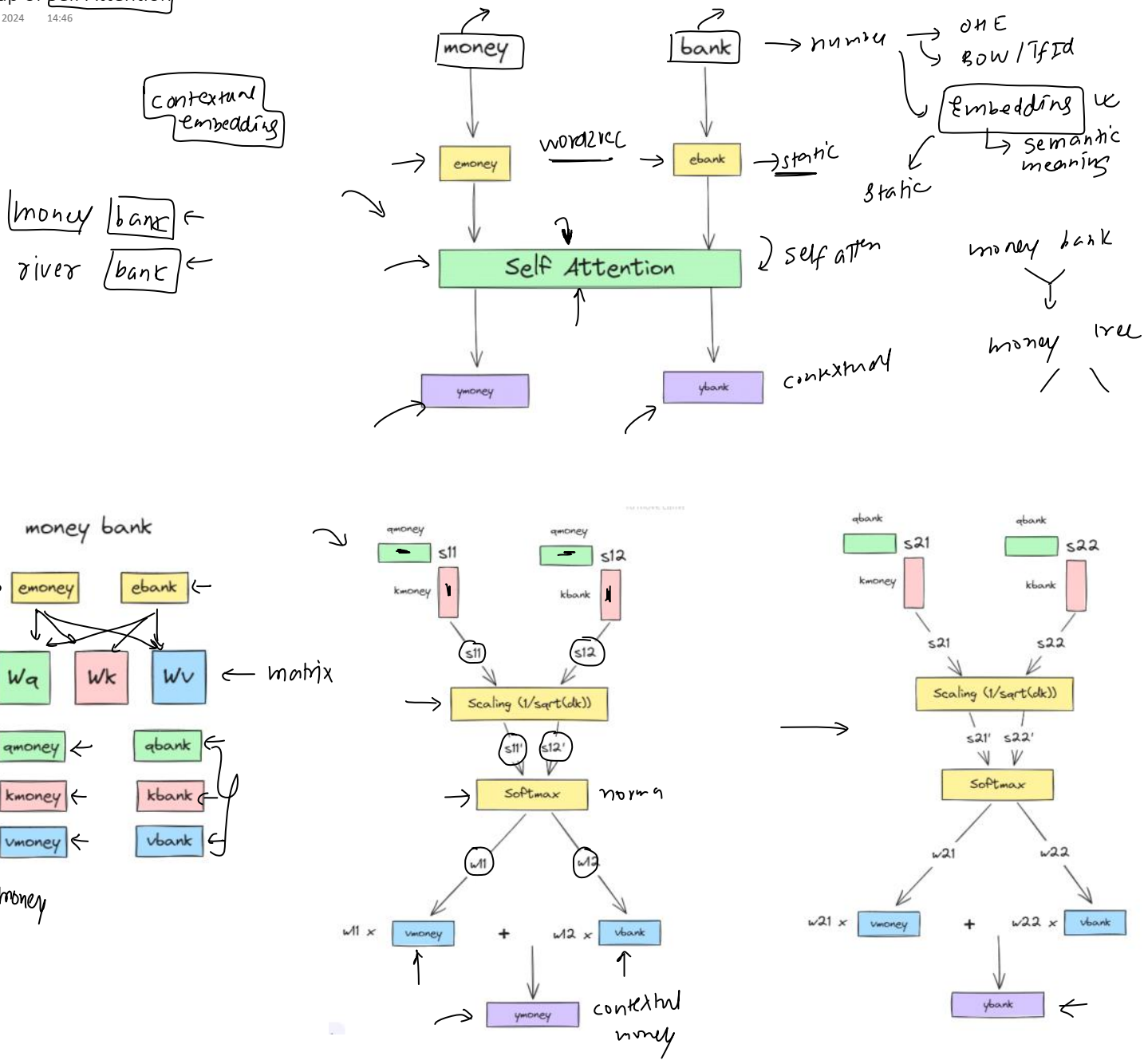
$e_{ij} = s_i^T h_j$

→ Luong attention

self attention

Recap of Self Attention

08 April 2024 14:46

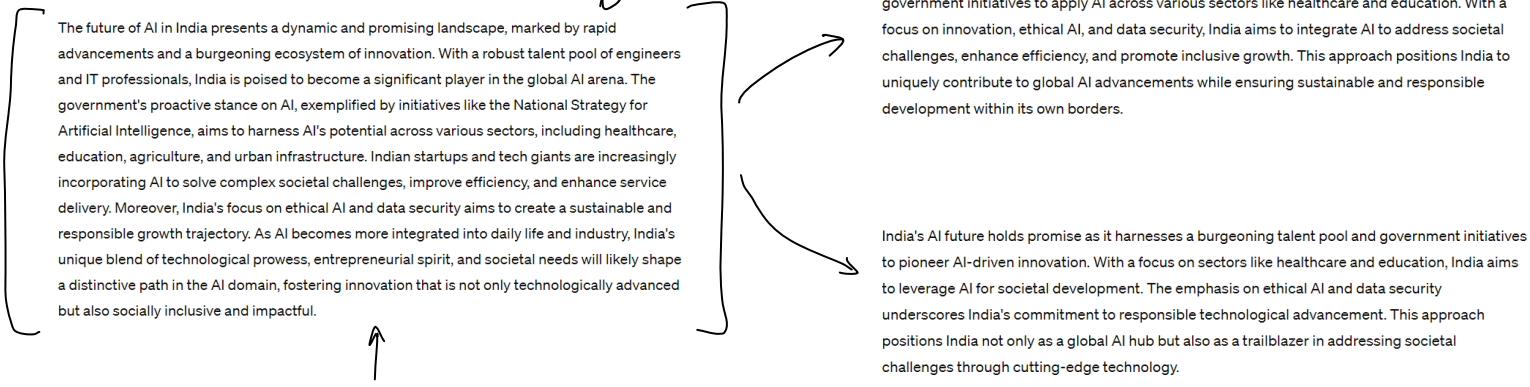


Problem with Self Attention

08 April 2024 14:47

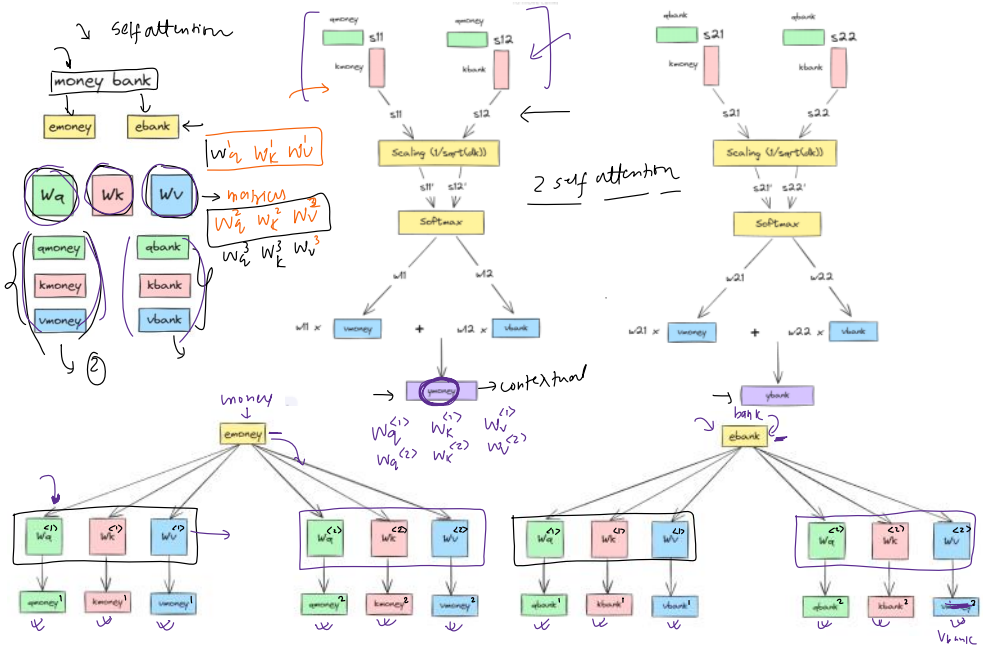
The man saw the astronomer with a telescope

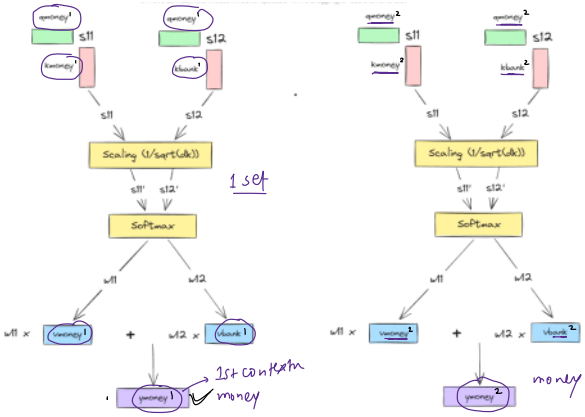
	The	man	saw	the	astronomer	with	a	telescope
The								
man								
saw								
the								
astronomer								
with								
a								
telescope								

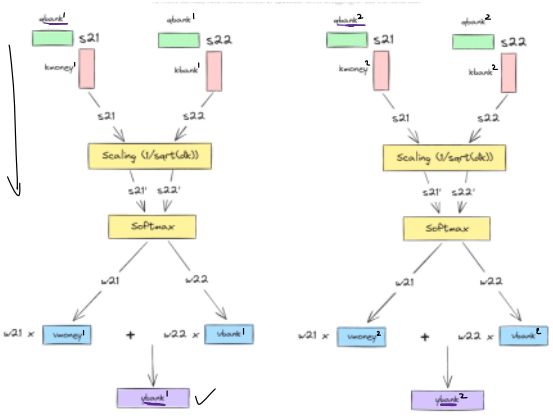


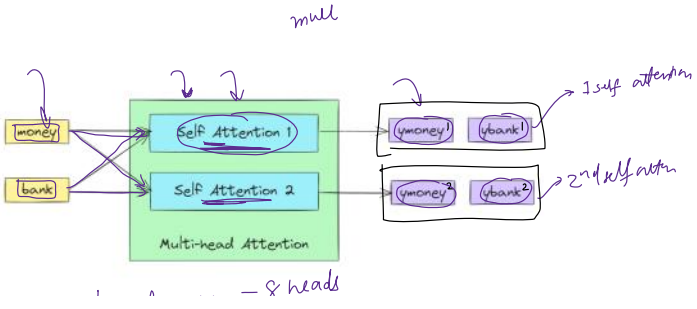
The man saw the astronomer with a telescope

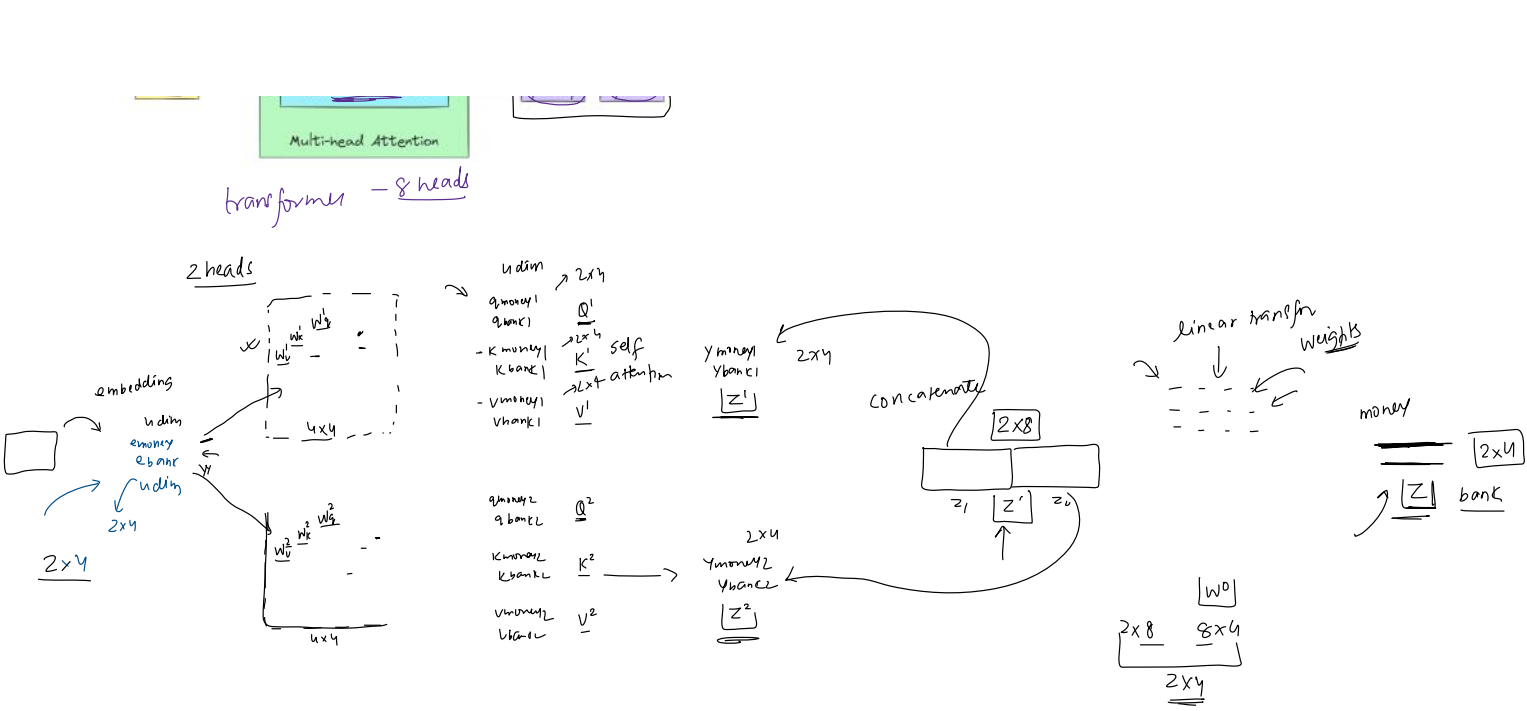
self attention → (2)

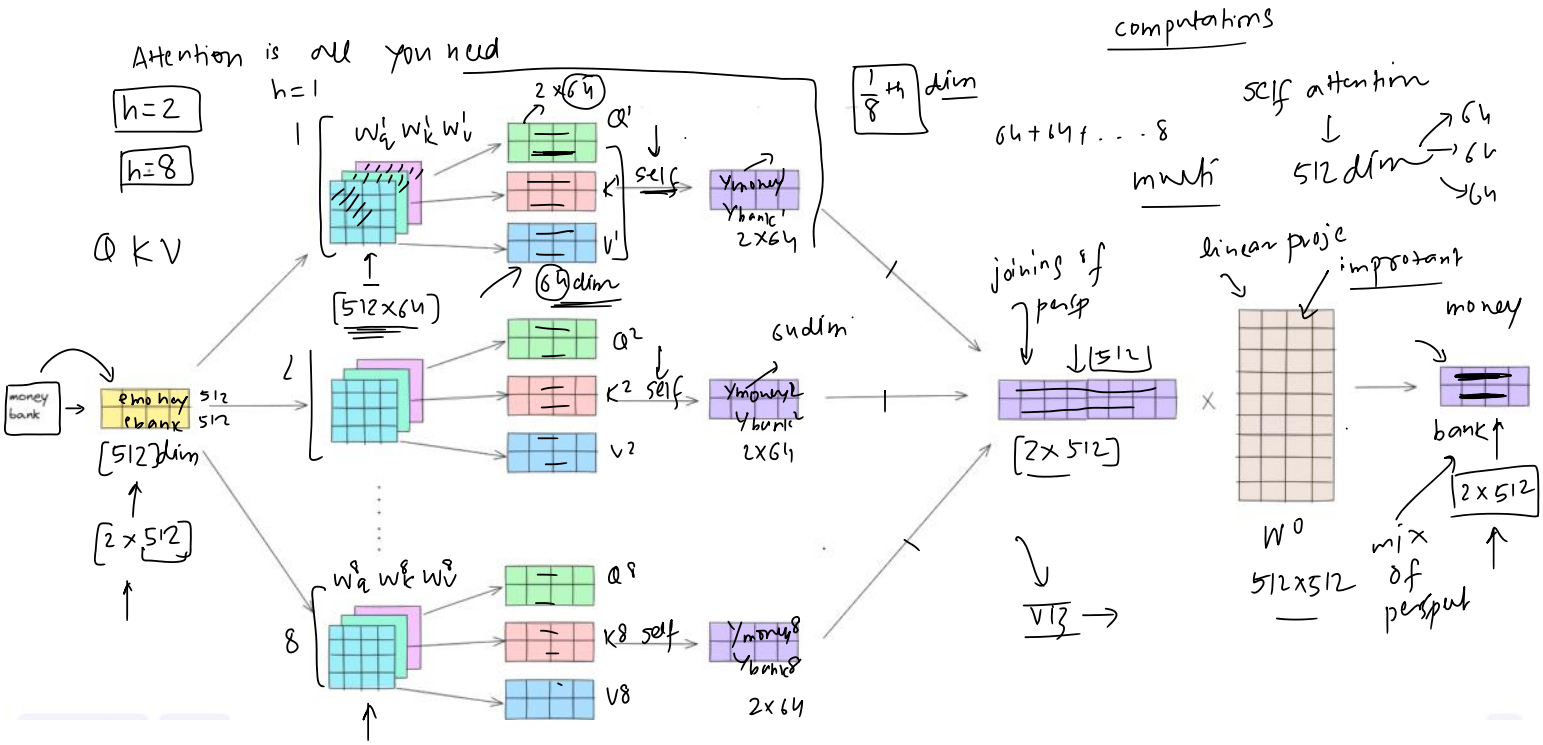








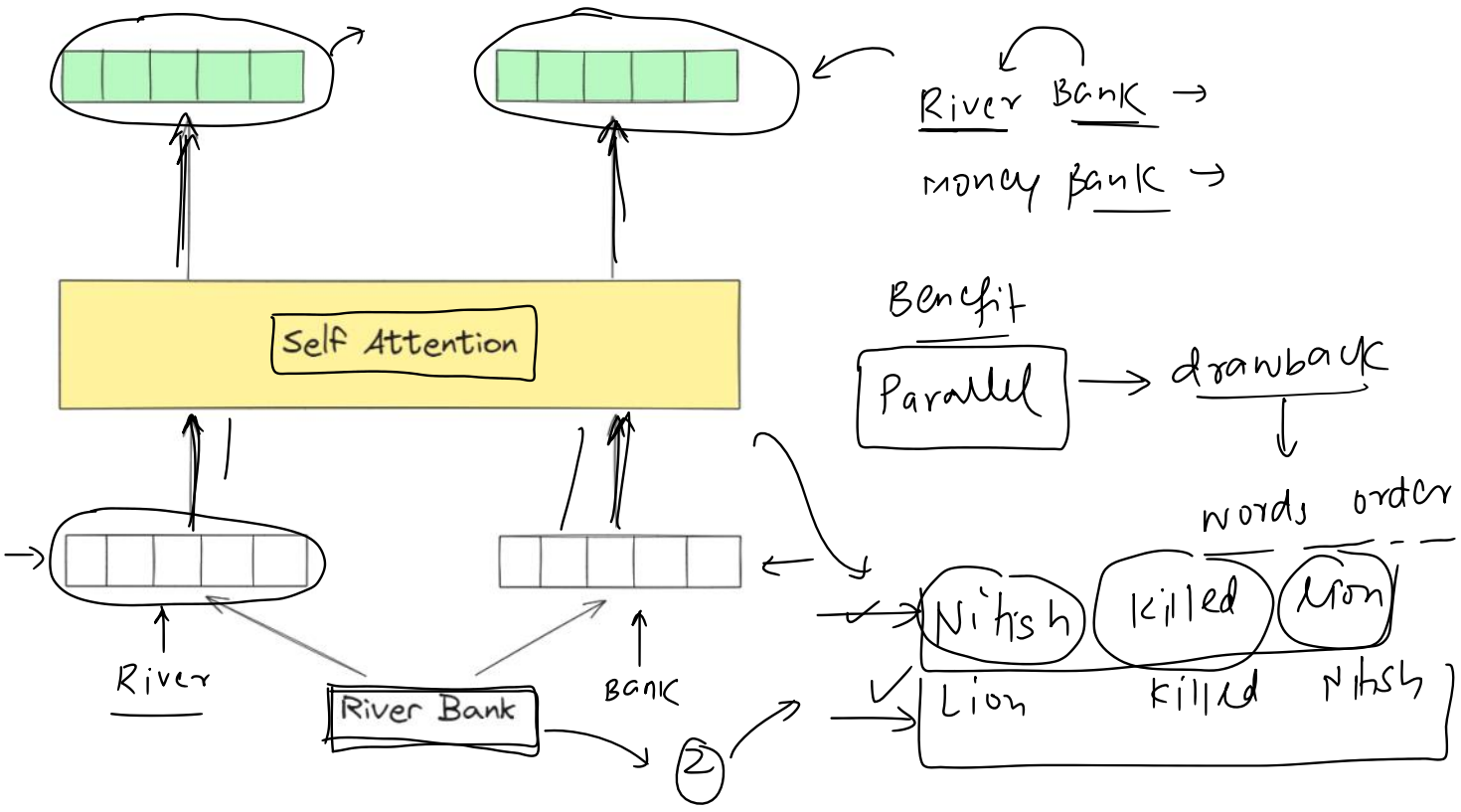

Multi-head Attention
transformer – 8 heads
2 heads
embedding
4dim
money
bank
adding
2x4
4x4
4dim
2x4
q money1
q bank1
Q1
2x4
K money1
K bank1
K1
self attention
2x4
V money1
V bank1
V1
q money2
q bank2
Q2
K money2
K bank2
K2
V money2
V bank2
V2
4x4
y money1
y bank1
Z1
2x4
y money2
y bank2
Z2
2x4
concatenate
2x8
z1
z'
z2
linear transform
weights
money
bank
2x4
Z
W0
2x8
8x4
2x4

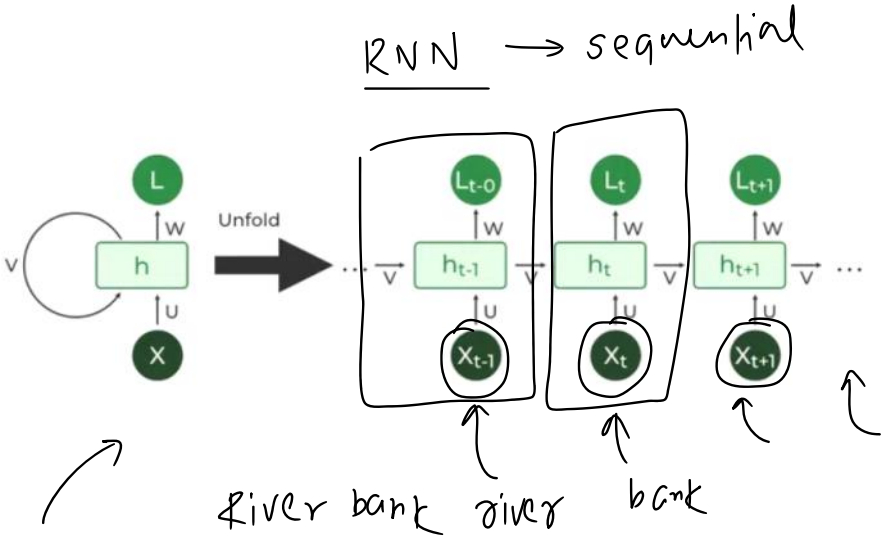

Attention is all you need
computations
h=2
h=8
Q K V
h=1
W_q^1 W_k^1 W_v^1
2x64
Q1
K1
V1
self
Y_{money}
Y_{bank}
2x64
[512x64]
64 dim
money bank
emoney 512
ebank 512
[512]dim
[2x512]
Q2
K2 self
V2
64dim
W_q^8 W_k^8 W_v^8
Q8
K8 self
V8
$\frac{1}{8}$th dim
64+64+...8
multi
joining of persp
[512]
[2x512]
V12
linear proje
important
W^0
512x512
mix of perspt
money
bank
[2x512]
self attention
512 dim
64
64
64

The Why

23 May 2024 14:33

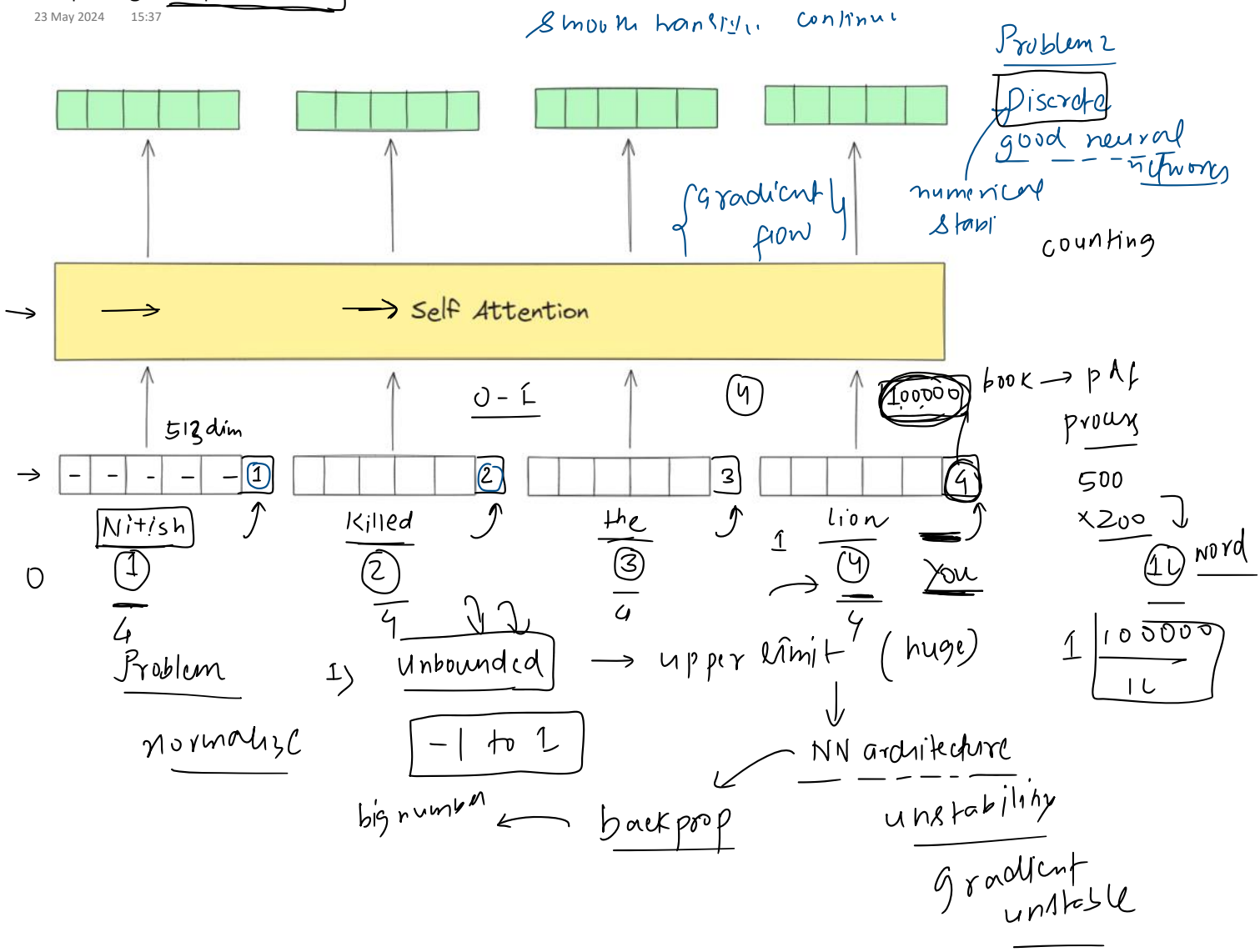
Self-Attention

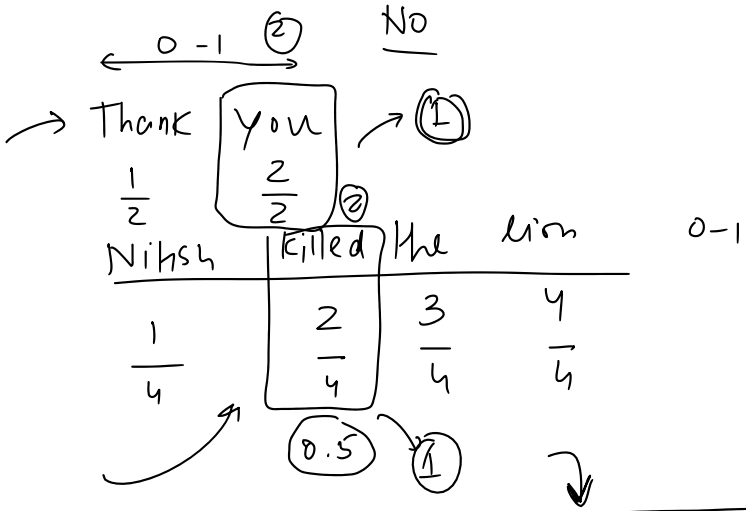




Proposing a simple solution

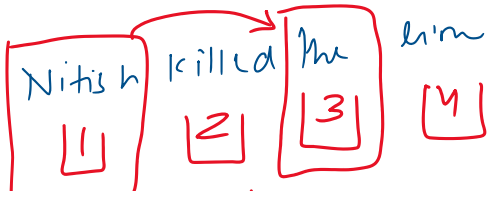
23 May 2024 15:37





Problem 3 → can't relative positionings

3 - 1 = 2



1 2 3 4

absolute positim / reative positim

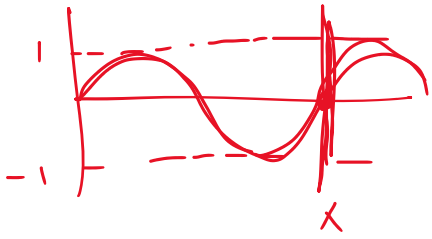
discrete → periodic → relative positioning

Problem

→ unbounded (bounded) ✓✓

→ discrete (continuous) → functim → trigonometric → sine

→ relative posith (periodic) ✓



positional encodn → sine → better solutim

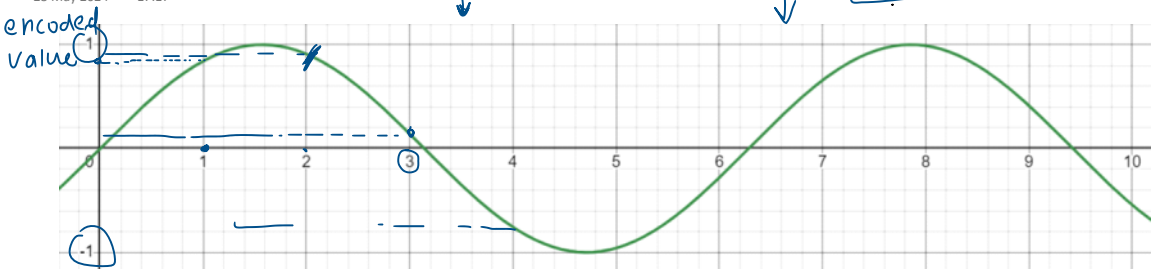
The sine function as a solution

23 May 2024 17:17

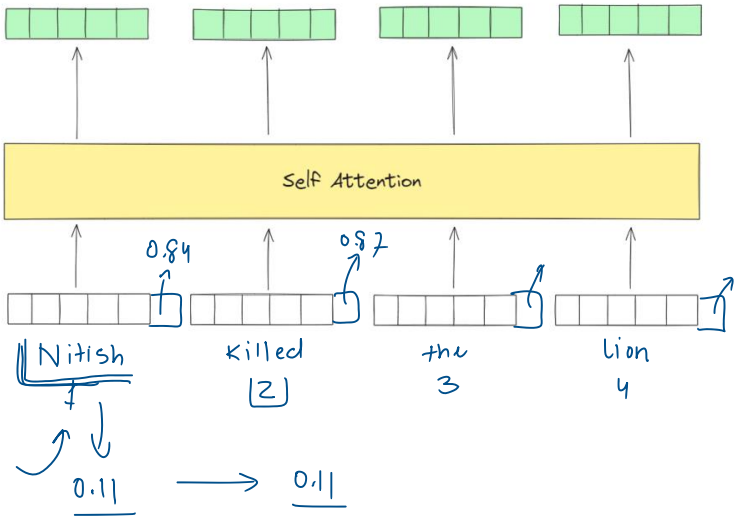
periodic

BIS problem.

$y = \sin(pos)$



1) unbounded
2) discrete
3) relational positioning



encoded value

$y = \sin(position)$

$y = \sin(1) = 0.84$

$y = \sin(2) = 0.87$

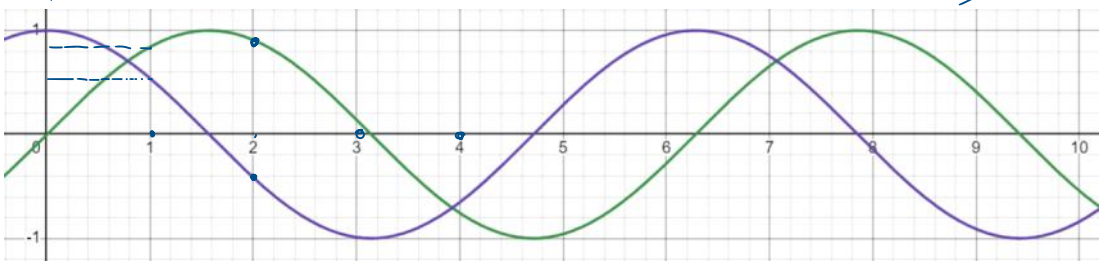
$y = \sin(3) = 0.14$

$y = \sin(4) = -0.90$

$y = \sin(1) = 0.84$

$y = \cos(1) = 0.5$

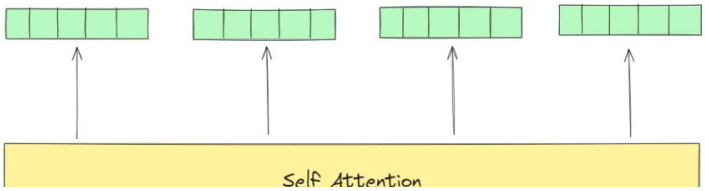
$y = \sin(pos)$ $y = \cos(pos)$

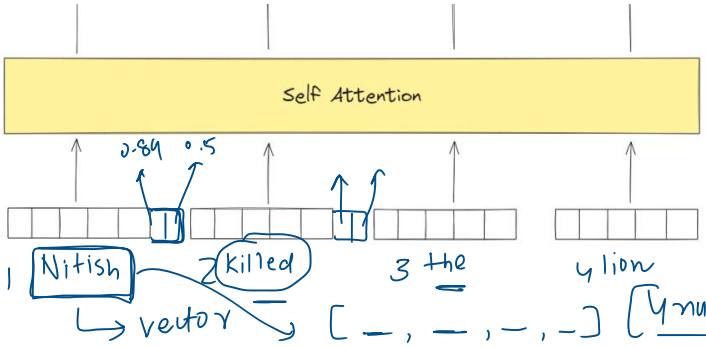


Nitish → [0.84, 0.5]

$y = \sin(2)$ $y = \cos(2)$

[0.87, -0.4]





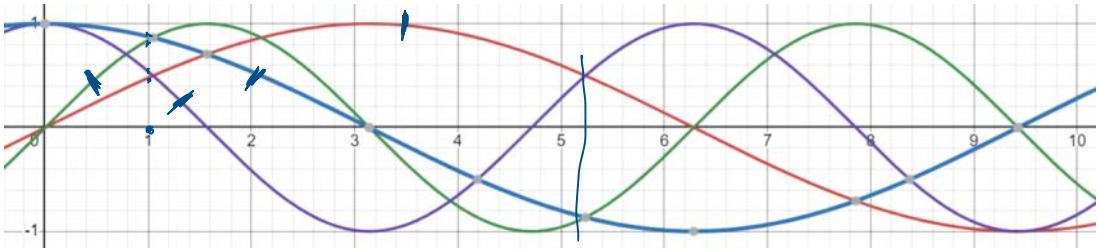
[0.87, -0.4] ↘

scalar → vector

$y = \sin(pos)$ $y = \cos(pos)$

$y = \sin(pos/2)$ $y = \cos(pos/2)$

↳ vector [_ , _ , _ , _] [4 number]

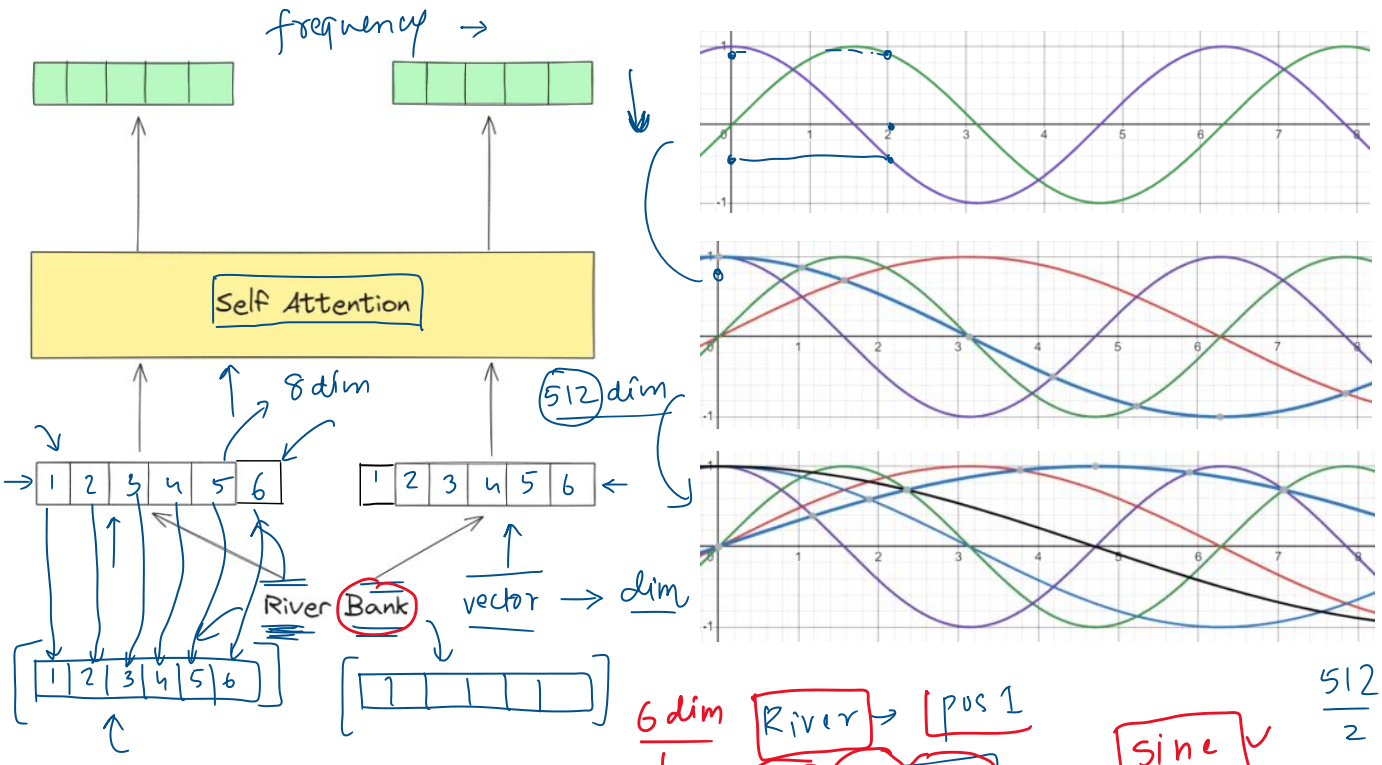


pos't 6 number vector

$y = \sin(pos/3)$ $y = \cos(pos/3)$

Positional Encoding

23 May 2024 20:35



embd (6 dim) + pos encod (6 dim) = 6 dim vector

6 dim → 2

River → pos 1

→ positional encodings

sine cosine

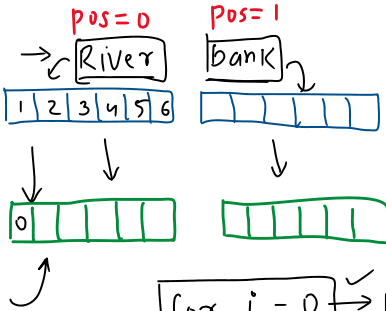
positional vector

512 / 2

concatenation → addition

↳ 6 dim 6 dim

↳ 12 dim → parameters increase → training double up



pos → position

pos=0 pos=1

d_{model} → dim of embedding

$d_{model} = 6$

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$

i = 0, 1, 2

i = 0 — $d_{model}/2$ + 1 → 0 - 6/2 3 → 0 - 3

for i = 0 → 0 - 2

for i = 0 → pos=0

$PE(0,0) = \sin(0/10000^{0}) = 0$

$PE(0,1) = \cos(0/10000^{0}) = 1$

2i+1

$PE(1,0) = \sin(1/10000^{0}) = 0.84$

$PE(1,1) = \cos(1/10000^{0}) = 0.54$

for i = 1

for i = 1

$1/3) = 0.04$

for i=1 ✓

$PE(0,2) = \sin\left(0 / 10000^{1/3}\right) = 0$

$PE(0,3) = \cos\left(0 / 10000^{1/3}\right) = 1$

for i=1

$PE(1,2) = \sin\left(1 / 10000^{1/3}\right) = 0.04$

$PE(1,3) = \cos\left(1 / 100000^{1/3}\right) = 0.99$

for i=2 ✓

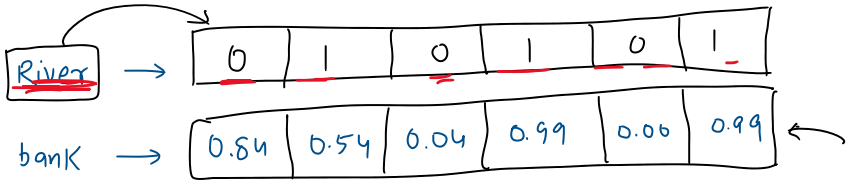
$PE(0,4) = \sin\left(0 / 10000^{2/3}\right) = 0$

$PE(0,5) = \cos\left(0 / 10000^{2/3}\right) = 1$

for i=2

$PE(1,4) = \sin\left(1 / 10000^{2/3}\right) = 0.00$

$PE(1,5) = \cos\left(1 / 10000^{2/3}\right) = 0.99$

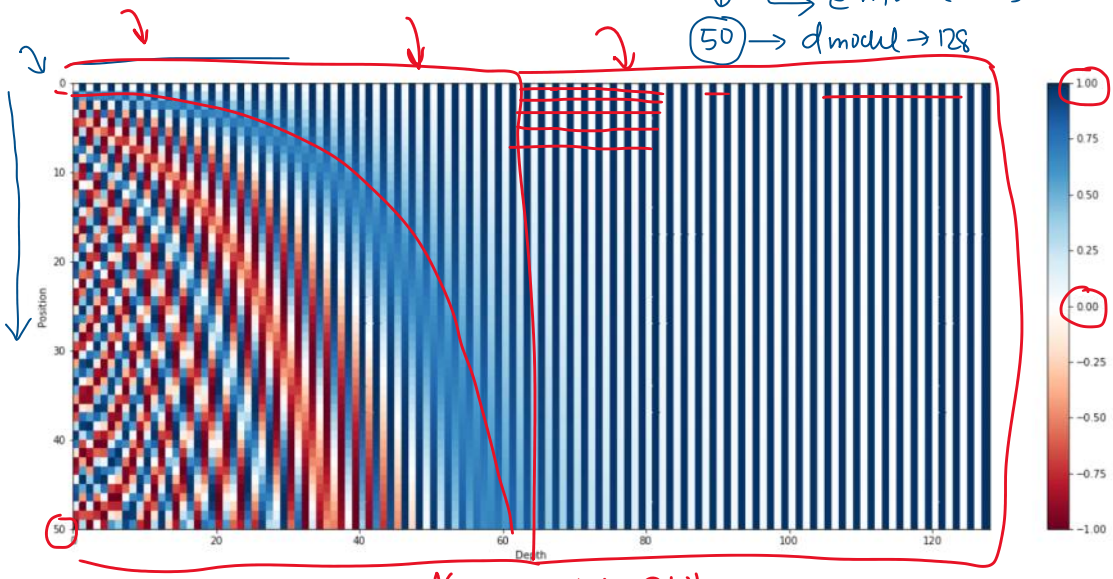


Interesting Observations

24 May 2024 02:06

sentence → 50 words → embedding → 128 dmodel = 128

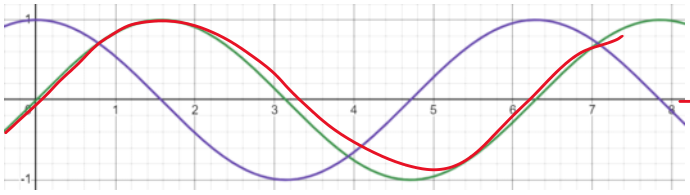
50 → dmodel → 128

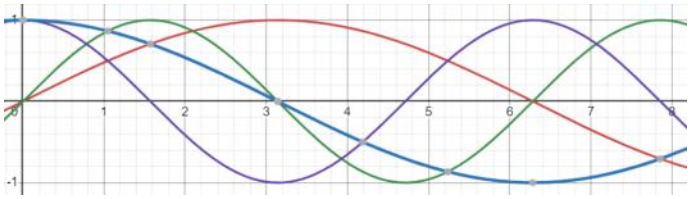


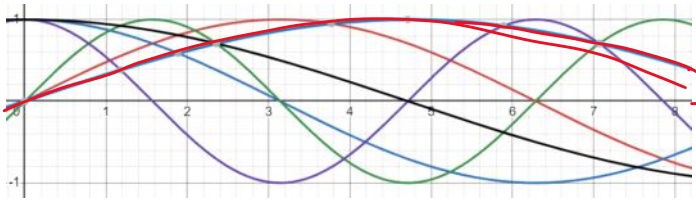
100

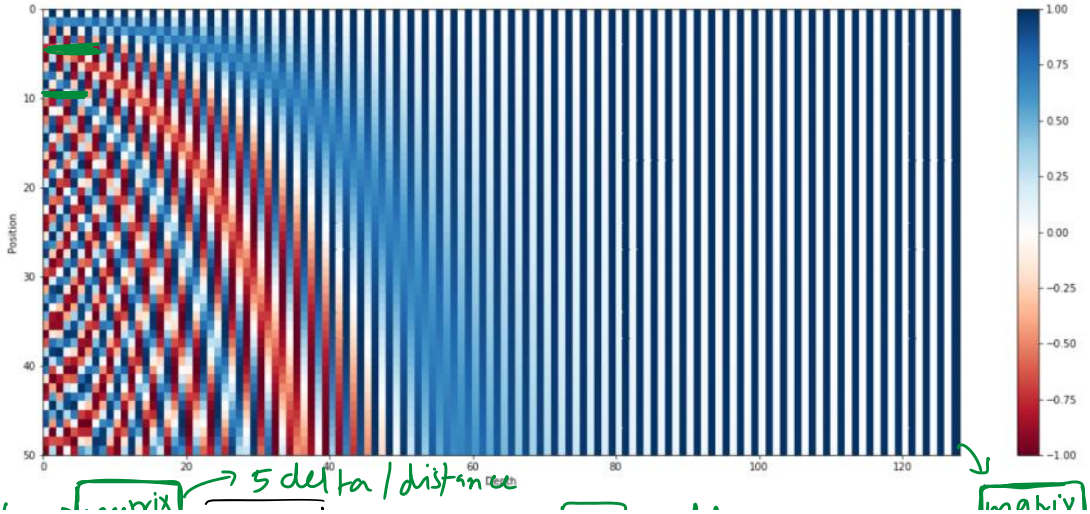
binary encoding 4bit 8bit

0 :	0	0	0	0	8 :	1	0	0	0
1 :	0	0	0	1	9 :	1	0	0	1
2 :	0	0	1	0	10 :	1	0	1	0
3 :	0	0	1	1	11 :	1	0	1	1
4 :	0	1	0	0	12 :	1	1	0	0
5 :	0	1	0	1	13 :	1	1	0	1
6 :	0	1	1	0	14 :	1	1	1	0
7 :	0	1	1	1	15 :	1	1	1	1



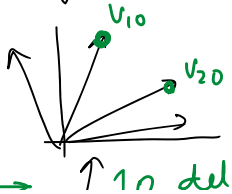






5 delta / distance

50 words
↓
50 vectors
↓
128 dim



10 delta

$V_5 \rightarrow$ matrix $\rightarrow$ 5 delta / distance

property

$\hookrightarrow V_{10}$

$V_{12} \rightarrow V_{17}$ $V_{21} \rightarrow V_{26}$

$V_{10} \rightarrow$ linear transformation $\rightarrow$ matrix $\rightarrow$ 10 delta / distance

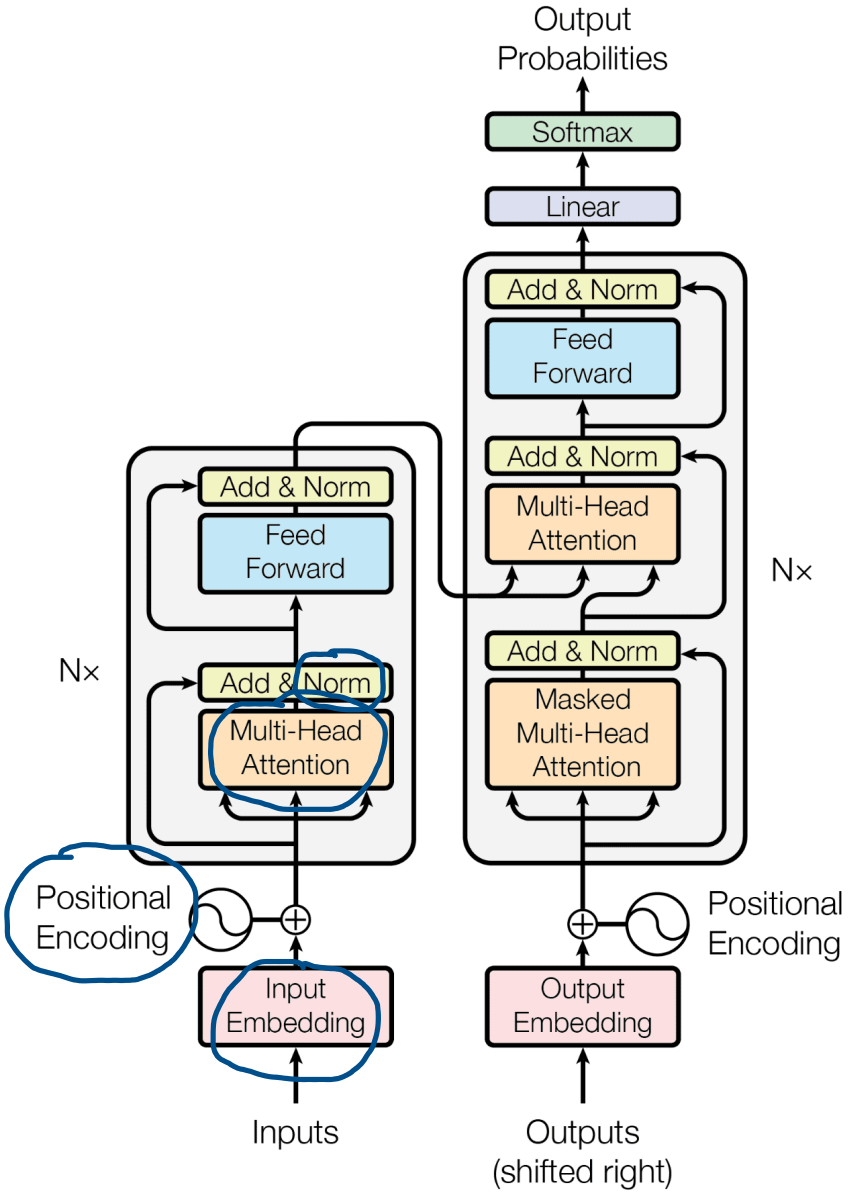
$\hookrightarrow V_{20}$ for each delta $V_{30} \rightarrow V_{40}$

$V_{40} \rightarrow V_{50}$

linear / matrix

Agenda

07 June 2024 02:03

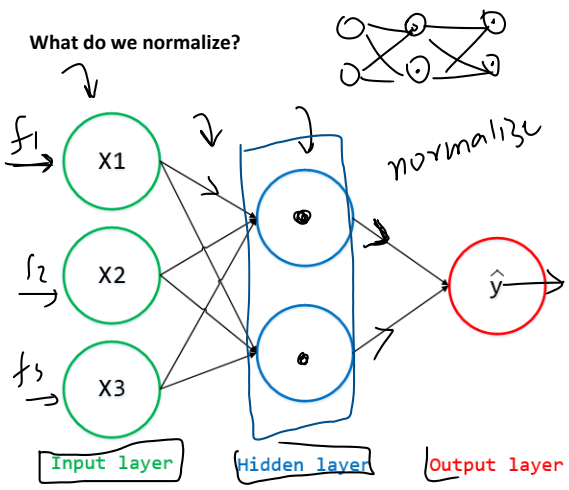


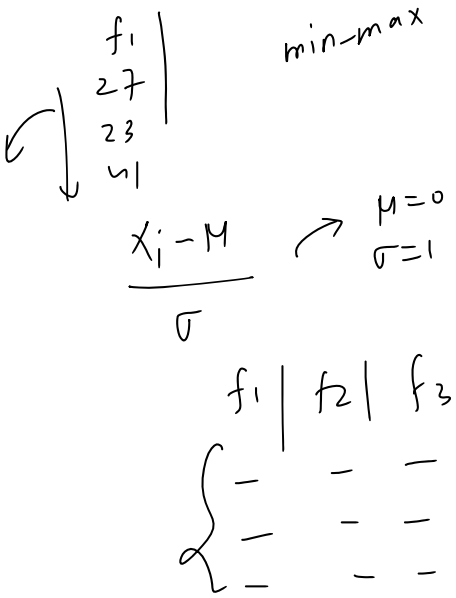
What is Normalization

05 June 2024 10:32

Normalization in deep learning refers to the process of transforming data or model outputs to have specific statistical properties, typically a mean of zero and a variance of one.

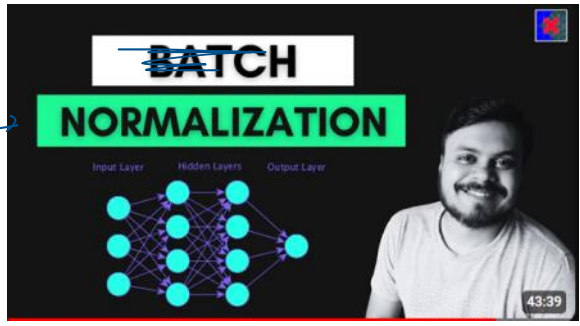
What do we normalize?

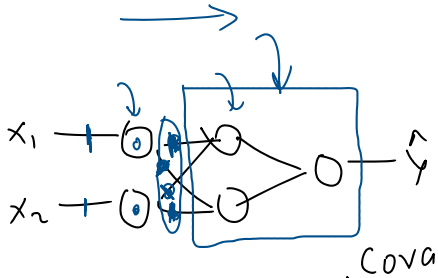


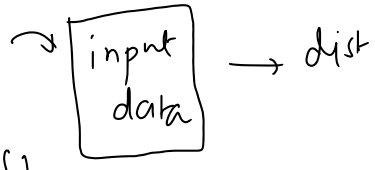


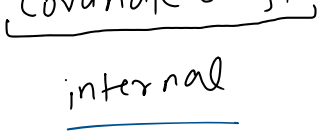
Benefits of Normalization in Deep Learning

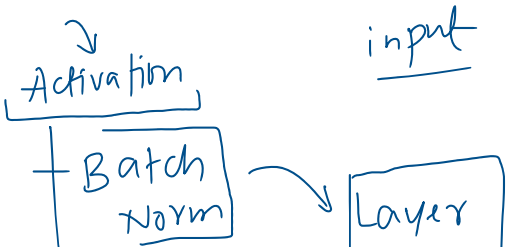
- Improved Training Stability:
  - Normalization helps to stabilize and accelerate the training process by reducing the likelihood of extreme values that can cause gradients to explode or vanish.
- Faster Convergence:
  - By normalizing inputs or activations, models can converge more quickly because the gradients have more consistent magnitudes. This allows for more stable updates during backpropagation.
- Mitigating Internal Covariate Shift:
  - Internal covariate shift refers to the change in the distribution of layer inputs during training. Normalization techniques, like batch normalization, help to reduce this shift, making the training process more robust.
- Regularization Effect:
  - Some normalization techniques, like batch normalization, introduce a slight regularizing effect by adding noise to the mini-batches during training. This can help to reduce overfitting.

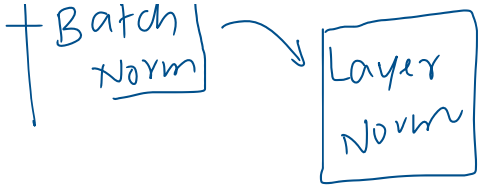






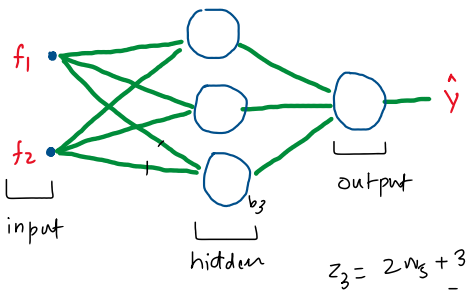





Batch
Norm
Layer
Norm

Batch Norm(Revision)

05 June 2024 10:39



f_1	f_2	z_1	z_2	z_3
2	3	7	5	4
1	1	2	3	4
5	4	1	2	3
6	1	7	5	6
7	1	3	3	4

$z_3 = 2w_5 + 3w_6 + b_3 = 4$

$z_1 = 2w_1 + 3w_2 + b_1 = 7$

$z_2 = 2w_3 + 3w_4 + b_2 = 5$

$\frac{7-\mu_1}{\sigma_1} = 0.86\,\gamma_1 + \beta_1 = 0.86$ ($\gamma_1 = 1$, $\beta_1 = 0$)

$\frac{2-\mu_1}{\sigma_1} = 0.71\,\gamma_1 + \beta_1 = 0.71$

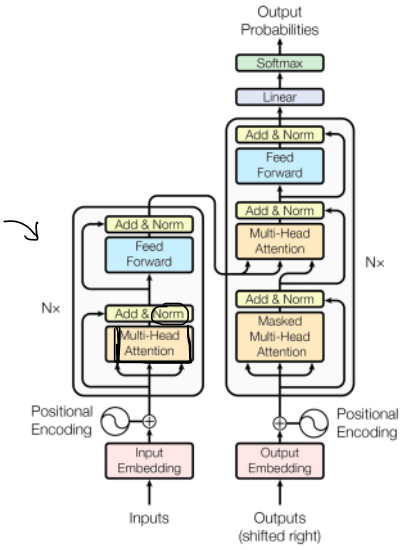
$\frac{5-\mu_2}{\sigma_2} = -0.21\,\gamma_2 + \beta_2 = -0.21$

$\frac{4-\mu_3}{\sigma_3} = 0.12\,\gamma_3 + \beta_3 = 0.12$

Why don't we use Batch Norm in Transformers?

05 June 2024 10:40

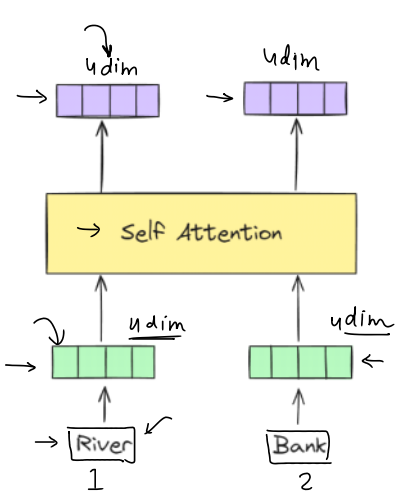
↳ self attention



→ Batch Norm

↑

Self - Attention



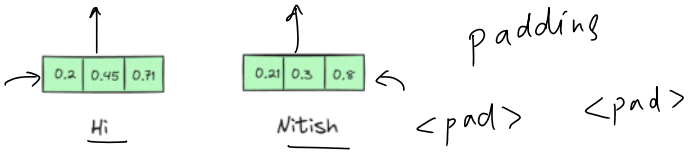
Batches

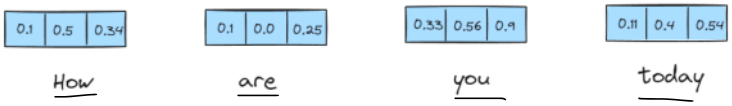
dim = 3

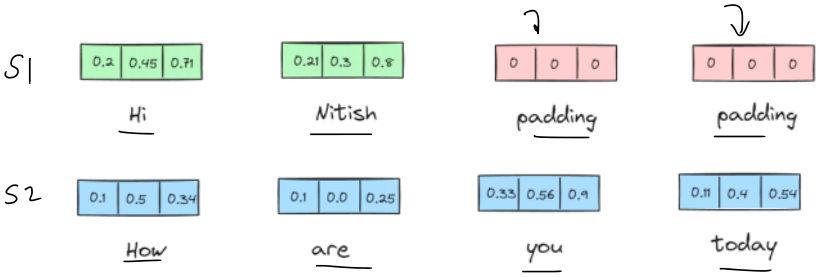
	Review	Sentiment
r1	Hi Nitish	1
r2	How are you today	0
r3	I am good	0
r4	You?	1

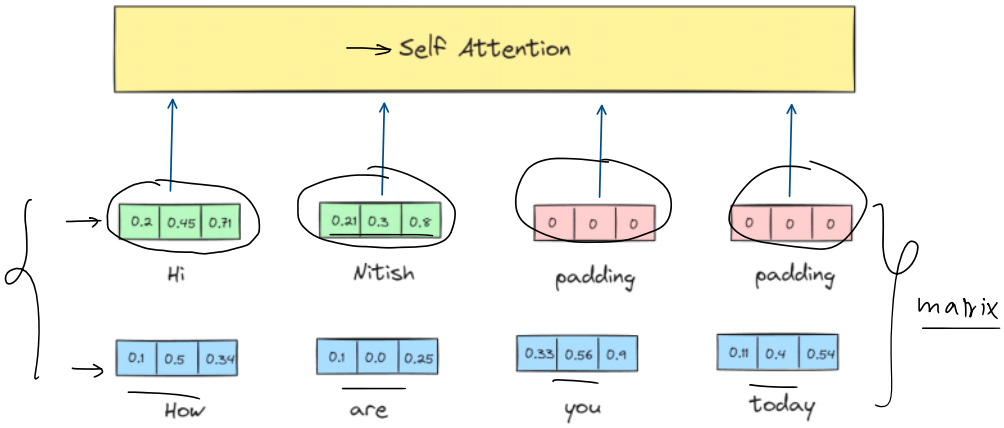
Embedding dimension - 3

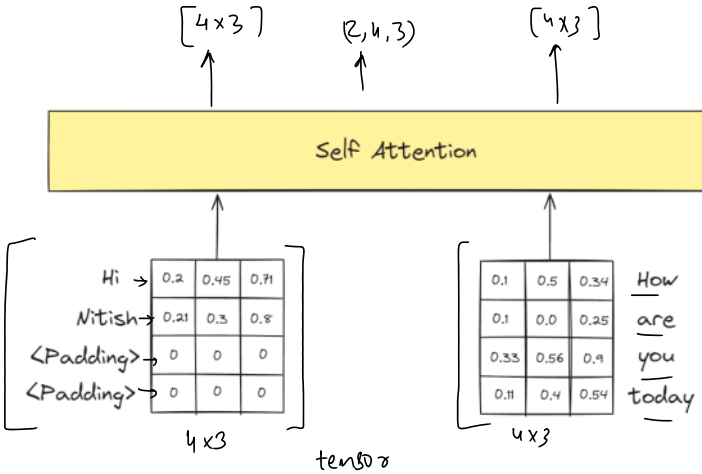
Batch Size - 2

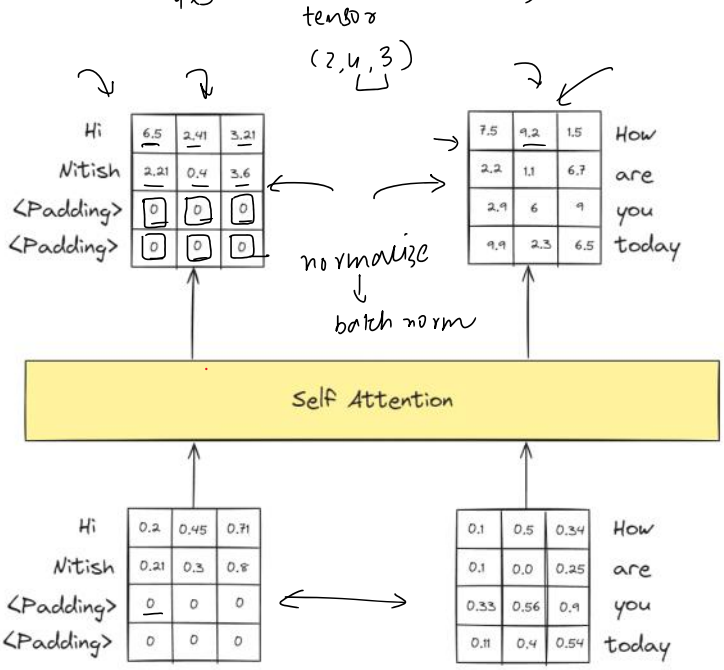


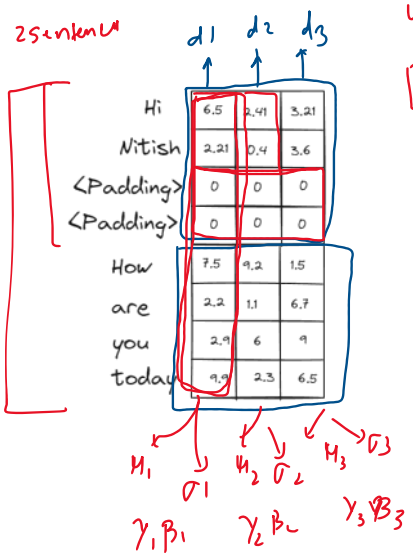





Self Attention
0.2 0.45 0.71
Hi
0.21 0.3 0.8
Nitish
0 0 0
padding
0 0 0
padding
matrix
0.1 0.5 0.34
How
0.1 0.0 0.25
are
0.33 0.56 0.9
you
0.11 0.4 0.54
today


[4x3]
(2,4,3)
[4x3]
Self Attention
Hi 0.2 0.45 0.71
Nitish 0.21 0.3 0.8
<Padding> 0 0 0
<Padding> 0 0 0
4x3
0.1 0.5 0.34 How
0.1 0.0 0.25 are
0.33 0.56 0.9 you
0.11 0.4 0.54 today
4x3

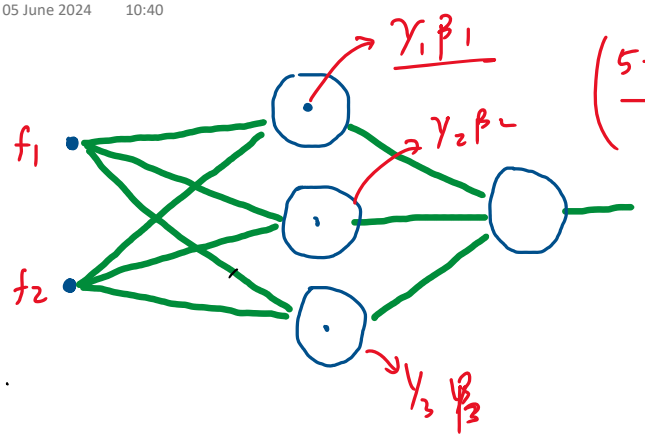

tensor
(2,4,3)
Hi 6.5 2.41 3.21
Nitish 2.21 0.4 3.6
<Padding> 0 0 0
<Padding> 0 0 0
7.5 9.2 1.5 How
2.2 1.1 6.7 are
2.9 6 9 you
9.9 2.3 6.5 today
normalize
batch norm
Self Attention
Hi 0.2 0.45 0.71
Nitish 0.21 0.3 0.8
<Padding> 0 0 0
<Padding> 0 0 0
0.1 0.5 0.34 How
0.1 0.0 0.25 are
0.33 0.56 0.9 you
0.11 0.4 0.54 today


2 sentences
d1 d2 d3
4 words
32
100 words
30 words
Hi 6.5 2.41 3.21
Nitish 2.21 0.4 3.6
<Padding> 0 0 0
<Padding> 0 0 0
How 7.5 9.2 1.5
are 2.2 1.1 6.7
you 2.9 6 9
today 9.9 2.3 6.5
μ1 σ1 μ2 σ2 μ3 σ3
γ1 β1 γ2 β2 γ3 β3

Layer Norm

05 June 2024 10:40

across batch across feature



$\left(\frac{5-\mu_4}{\sigma_4}\right)\gamma_2+\beta_2$

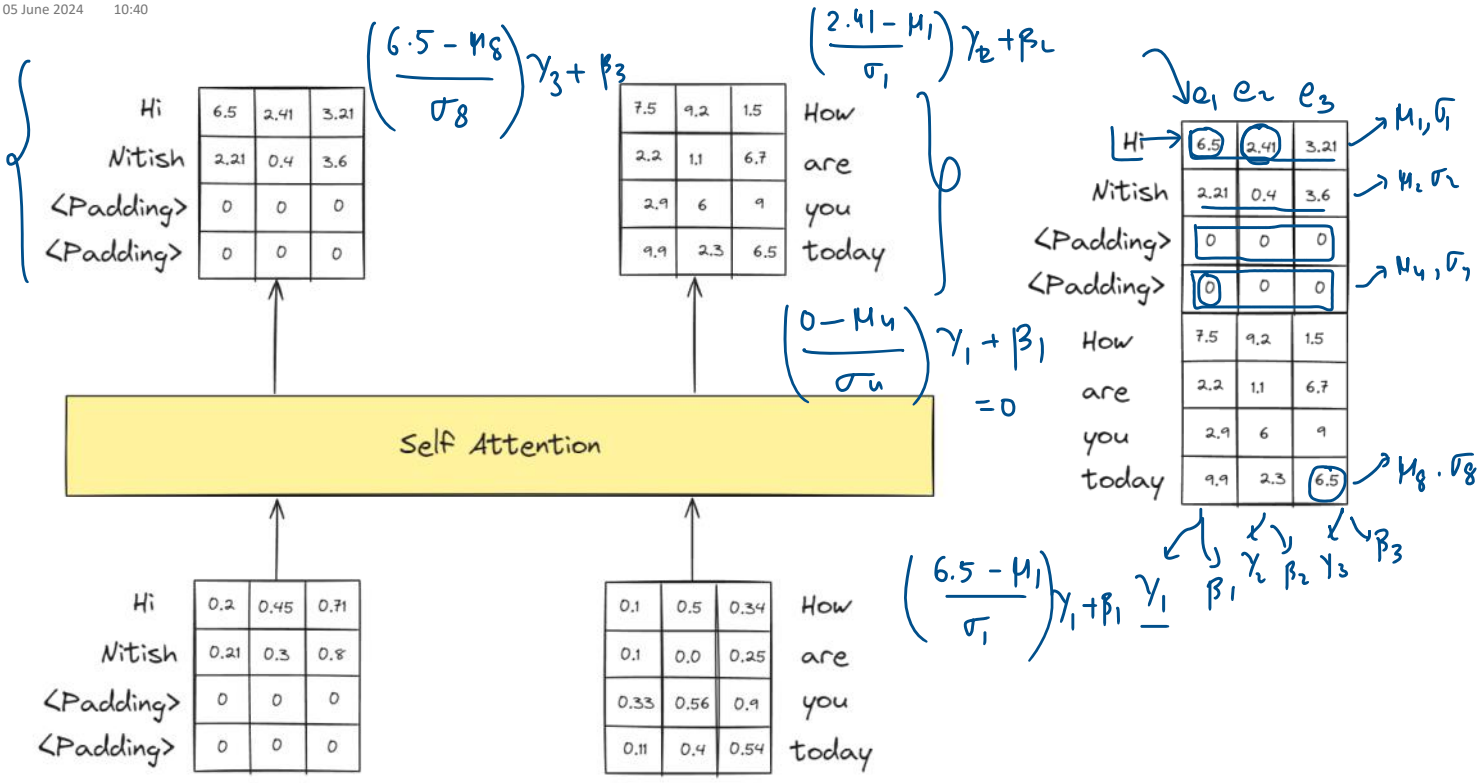
f_1	f_2	z_1	z_2	z_3	
2	3	7	5	4	μ_1, σ_1
1	1	2	3	4	μ_2, σ_2
5	4	1	2	3	
6	1	7	5	6	
7	1	3	3	4	μ_5, σ_5

$\left(\frac{2-\mu_2}{\sigma_2}\right)\gamma_1+\beta_1$ $\left(\frac{4-\mu_1}{\sigma_1}\right)\gamma_3+\beta_3$

$\left(\frac{5-\mu_1}{\sigma_1}\right)\gamma_2+\beta_2$ $\frac{7-\mu_1}{\sigma_1} = 0.3\gamma_1+\beta_1$

Layer Norm in Transformers

05 June 2024 10:40



Recap

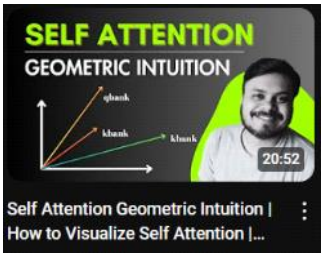
09 July 2024 08:47

Introduction to Transformers | Transformers Part 1

What is Self Attention | Transformers Part 2 | CampusX

Self Attention in Transformers | Deep Learning | Simple Explanatio...

Scaled Dot Product Attention | Why do we scale Self Attention?



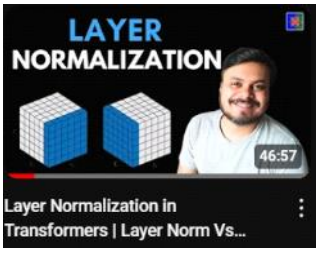
Self Attention Geometric Intuition | How to Visualize Self Attention |...

Why is Self Attention called "Self"? | Self Attention Vs Luong Attention i...

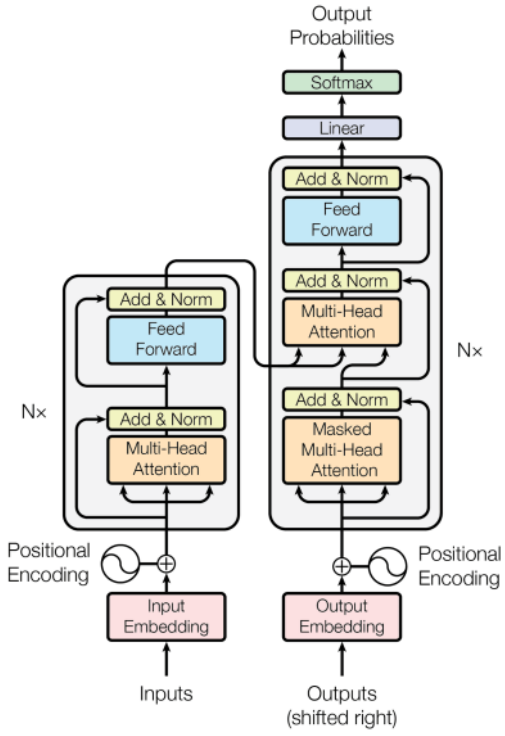
What is Multi-head Attention in Transformers | Multi-head Attentio...

Positional Encoding in Transformers | Deep Learning | CampusX

7hrs



Layer Normalization in Transformers | Layer Norm Vs...



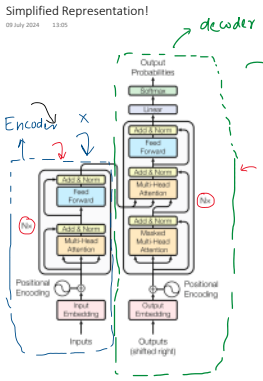
complex

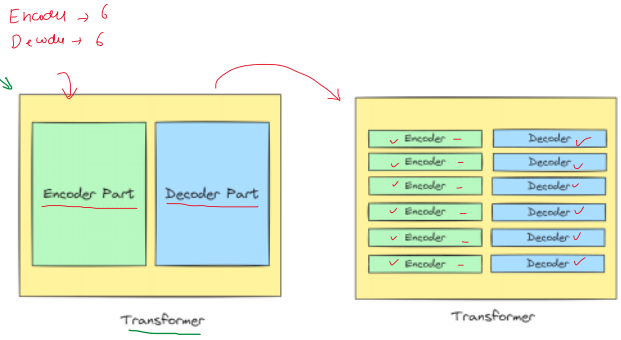
encoder

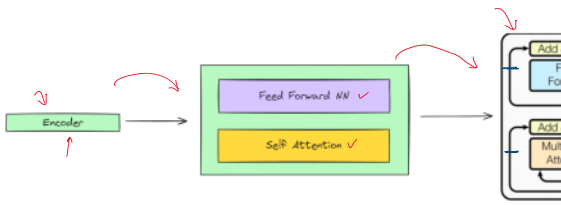
Decoder

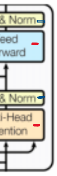
Simplified Representation!

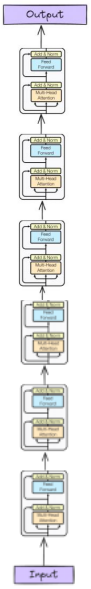
09 July 2024 13:05





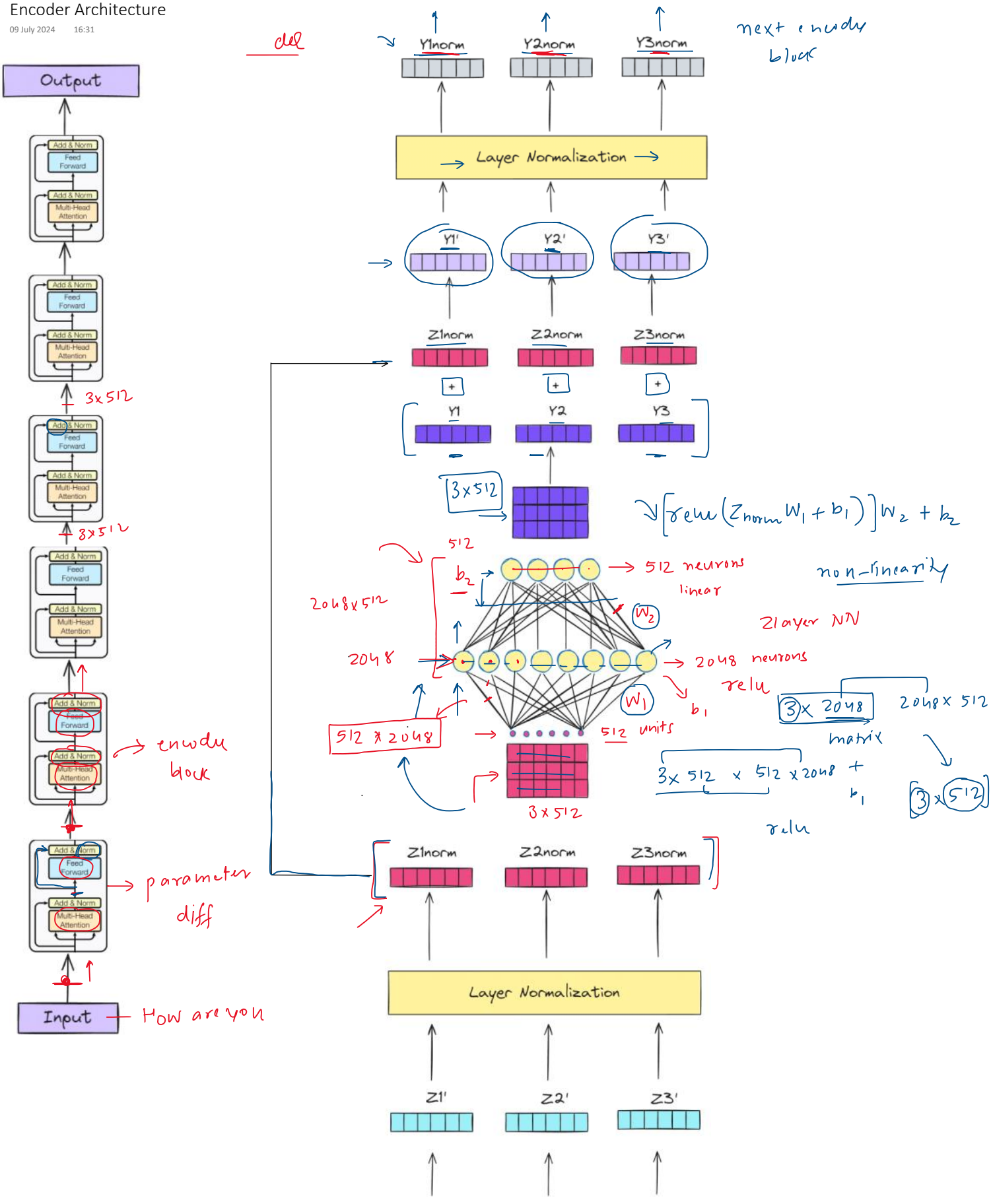


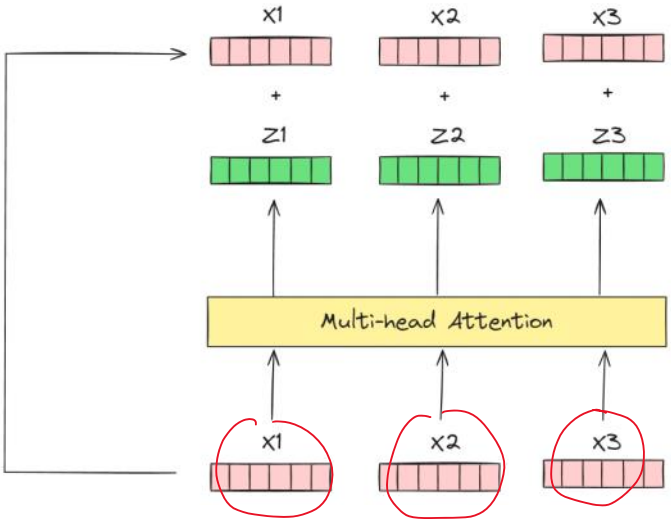

& Norm
eed
ward
& Norm
i-Head
ention

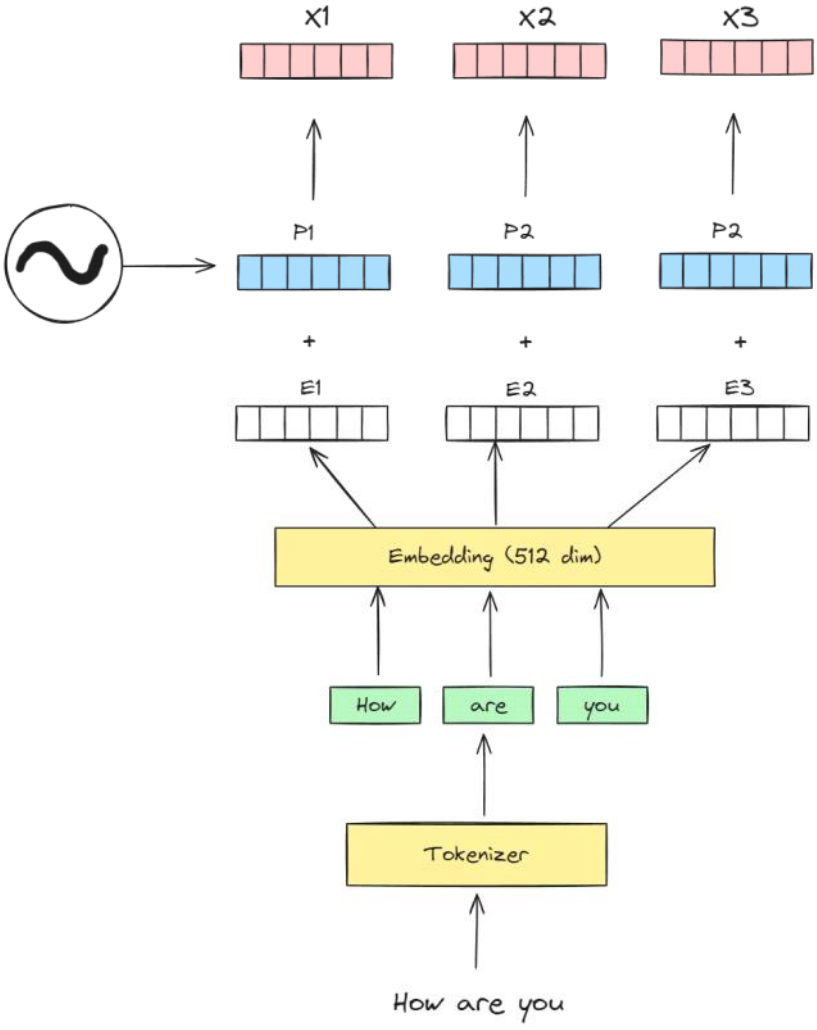

Output
Input

Encoder Architecture

09 July 2024 16:31




X1
X2
X3
+
+
+
Z1
Z2
Z3
Multi-head Attention
X1
X2
X3


X1
X2
X3
P1
P2
P2
+
+
+
E1
E2
E3
Embedding (512 dim)
How
are
you
Tokenizer
How are you

Some questions

09 July 2024 20:48

1. Why use residual connections? → Kaggle → comment
2. Why use a FFNN?
3. Why use 6 encoder blocks?

alternate

gradient flow

relu

non-linearity

scratch pytorch

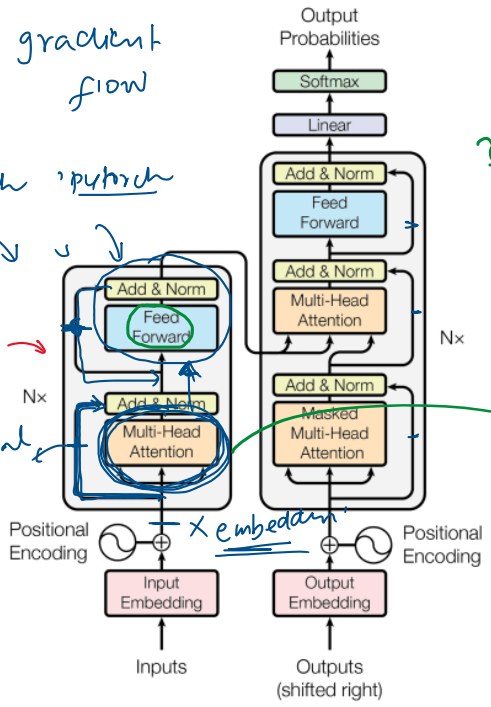
language → complex

1) stable training — deep NN

ResNet

vanishing Grad

residual



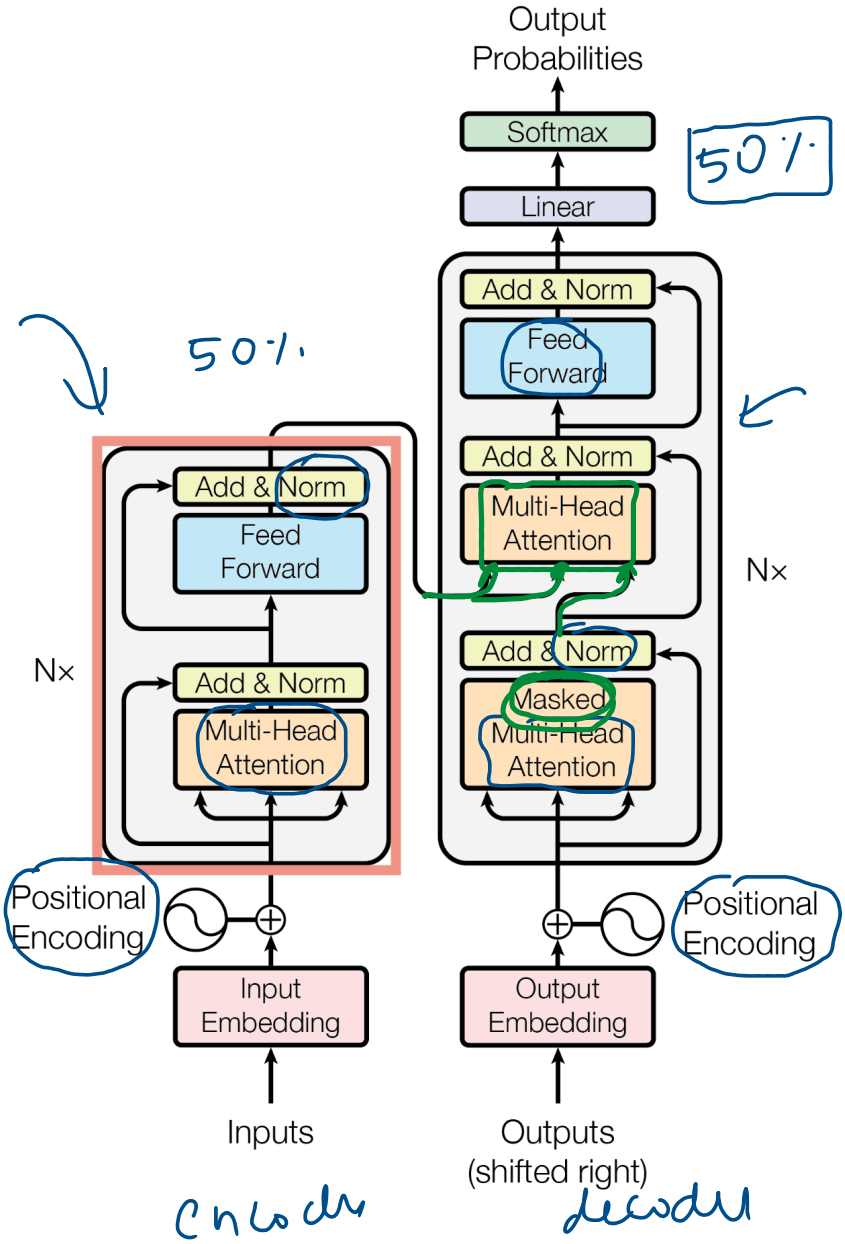
linear

× embedding

2)

Recap

23 July 2024 17:09

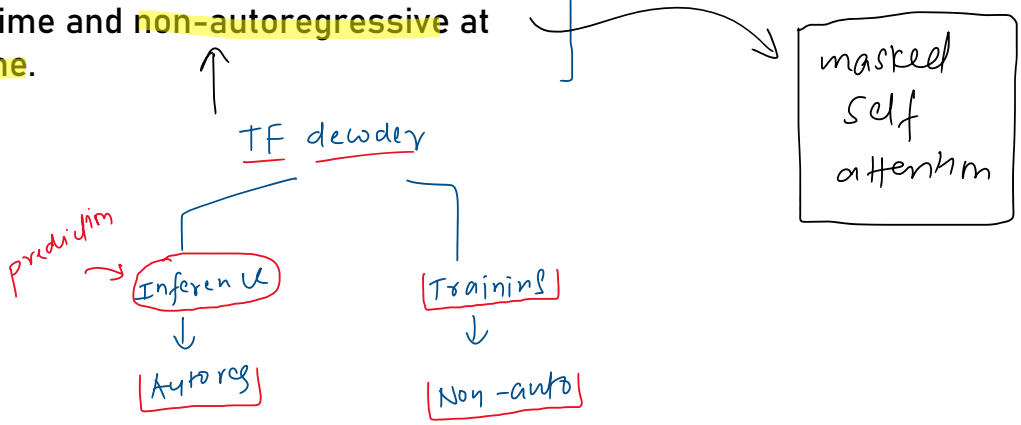


Autoregressive models

23 July 2024 17:39

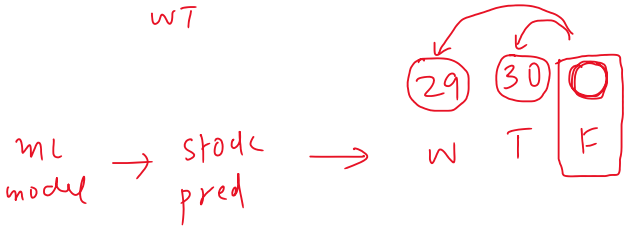
economics → Time series

The Transformer decoder is autoregressive at inference time and non-autoregressive at training time.

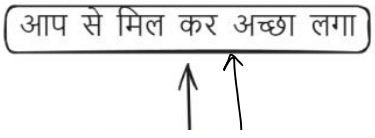


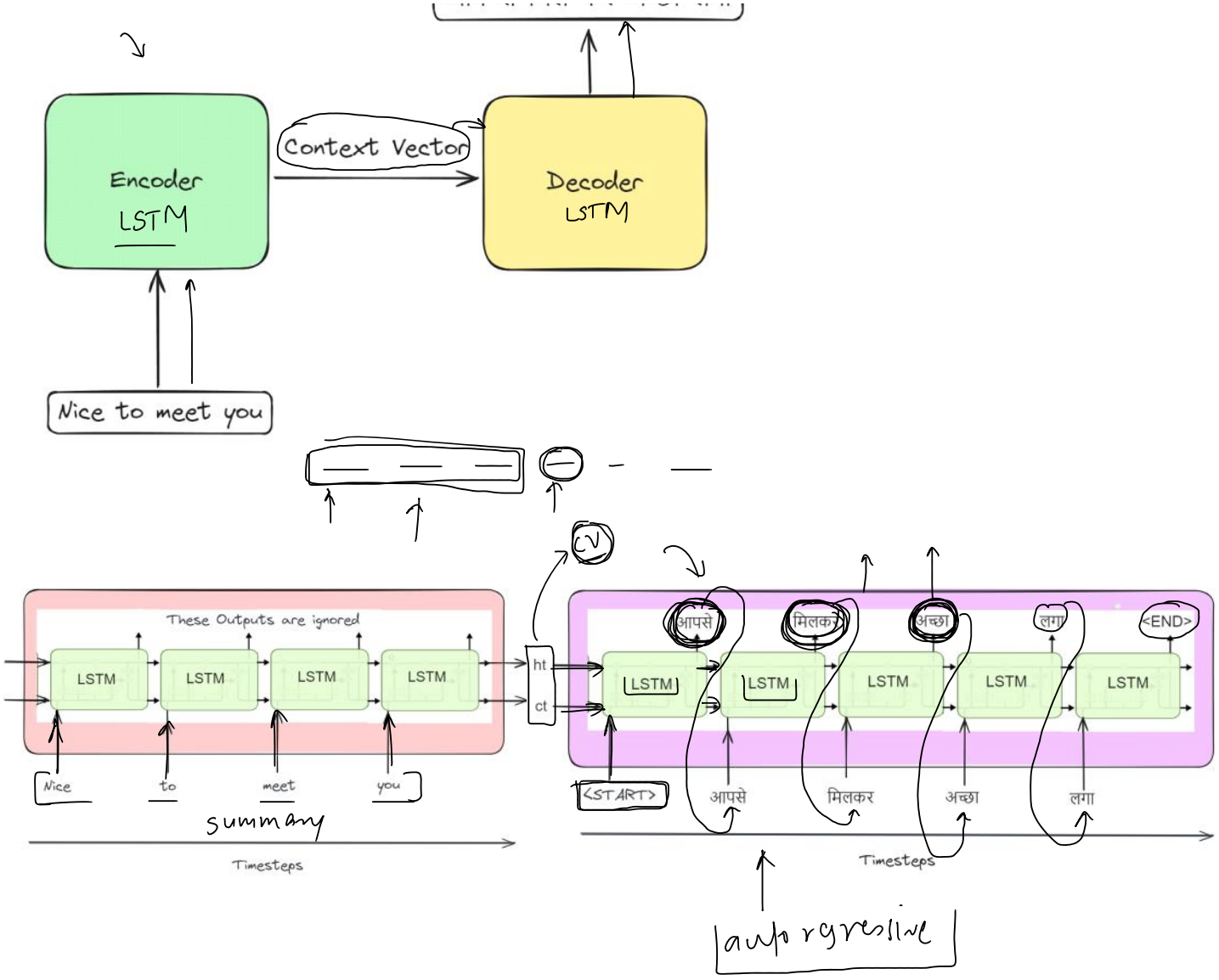
In the context of deep learning, autoregressive models are a class of models that generate data points in a sequence by conditioning each new point on the previously generated points.

F WT



↳ Encoder - Decoder → Ilya




Encoder
LSTM
Context Vector
Decoder
LSTM
Nice to meet you
These Outputs are ignored
LSTM
LSTM
LSTM
LSTM
ht
ct
Nice
to
meet
you
summary
Timesteps
CV
आपसे
मिलकर
अच्छा
लगा
<END>
<START>
Timesteps
auto regressive

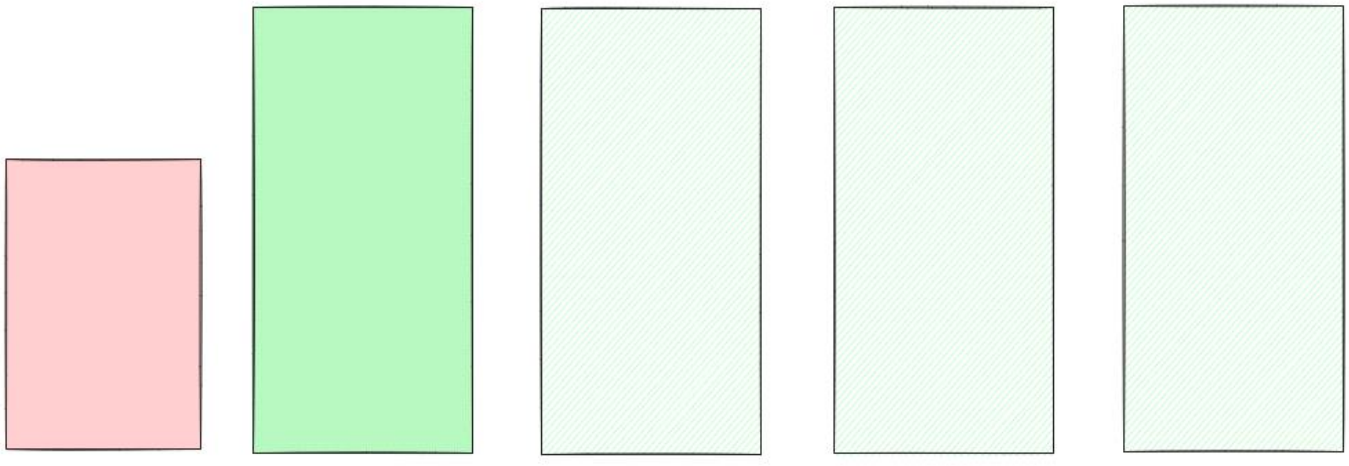
Transformer as an Autoregressive Model

23 July 2024 23:16

The Transformer decoder is autoregressive at inference time and non-autoregressive at training time.

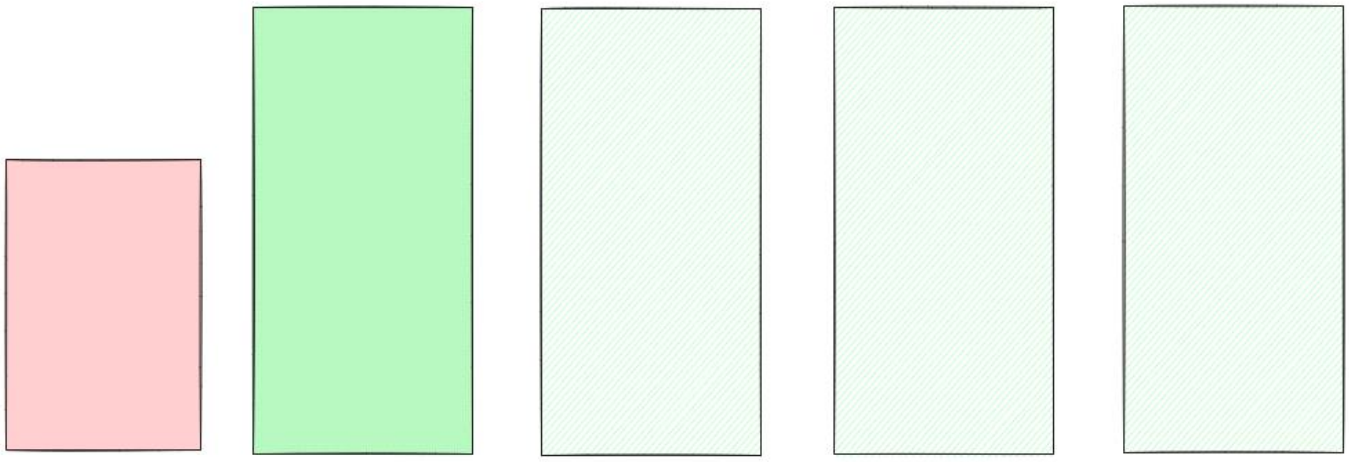
Inference

Query Sentence -> I am fine



Training

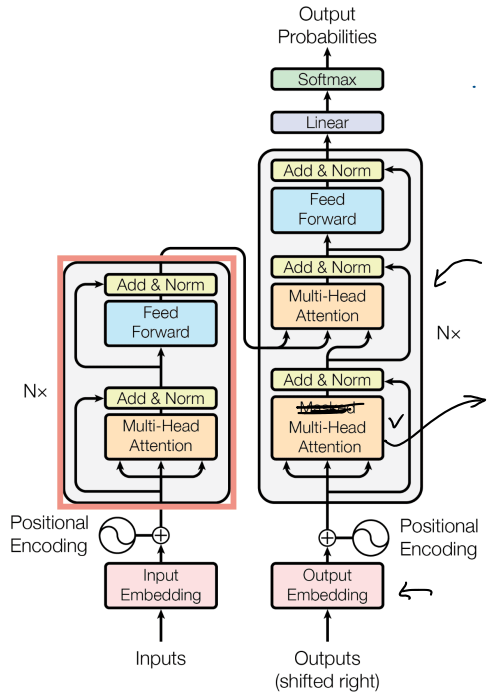
S.No	English Sentence	Hindi Sentence
1	How are you?	आप कैसे हैं
2	Congratulations	बधाई हो
3	Thank you	धन्यवाद



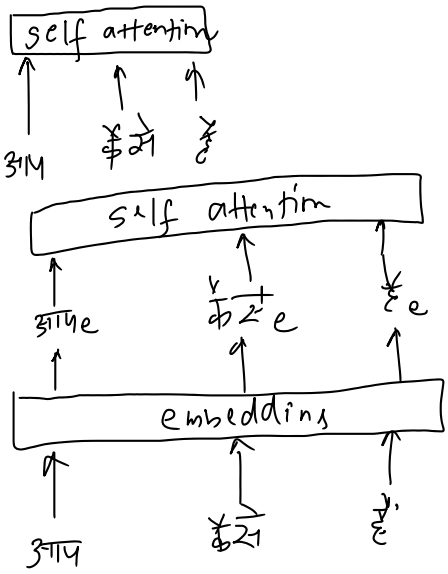
The problem in parallelizing

25 July 2024 22:55

S.No	English Sentence	Hindi Sentence
1	How are you?	आप कैसे हैं
2	Congratulations	बधाई हो
3	Thank you	धन्यवाद



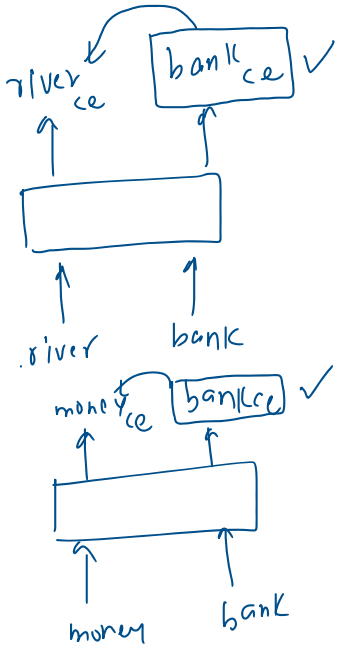
self attention



word ce

self attenti

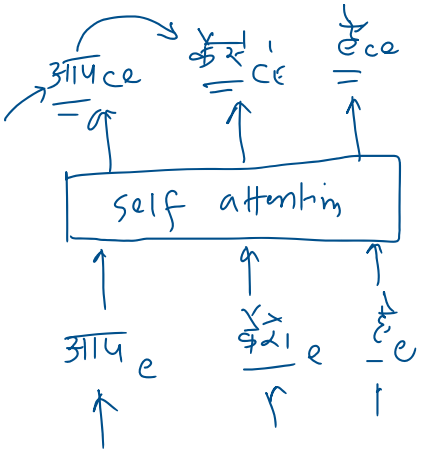
word e



river bank

money bank

Data leakage

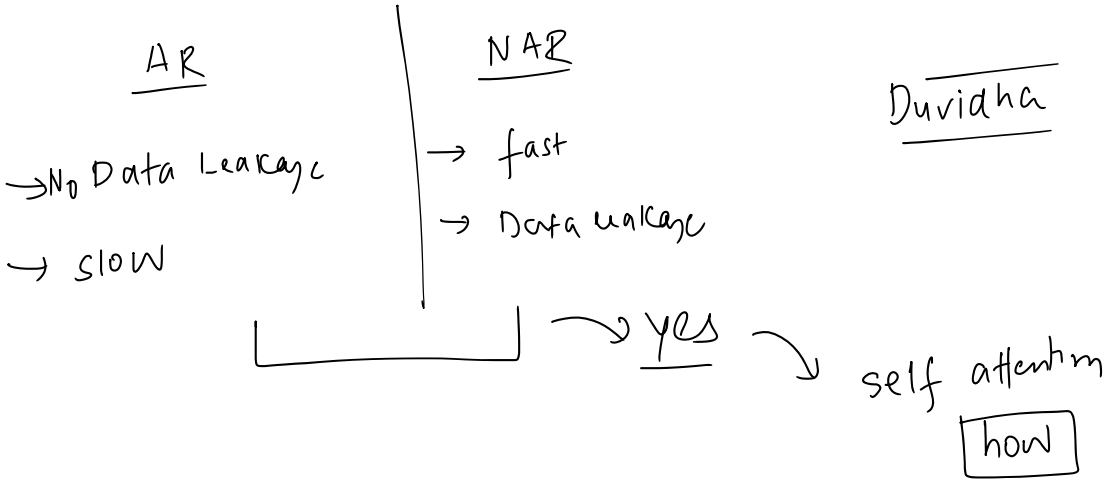


Cheating

current token value

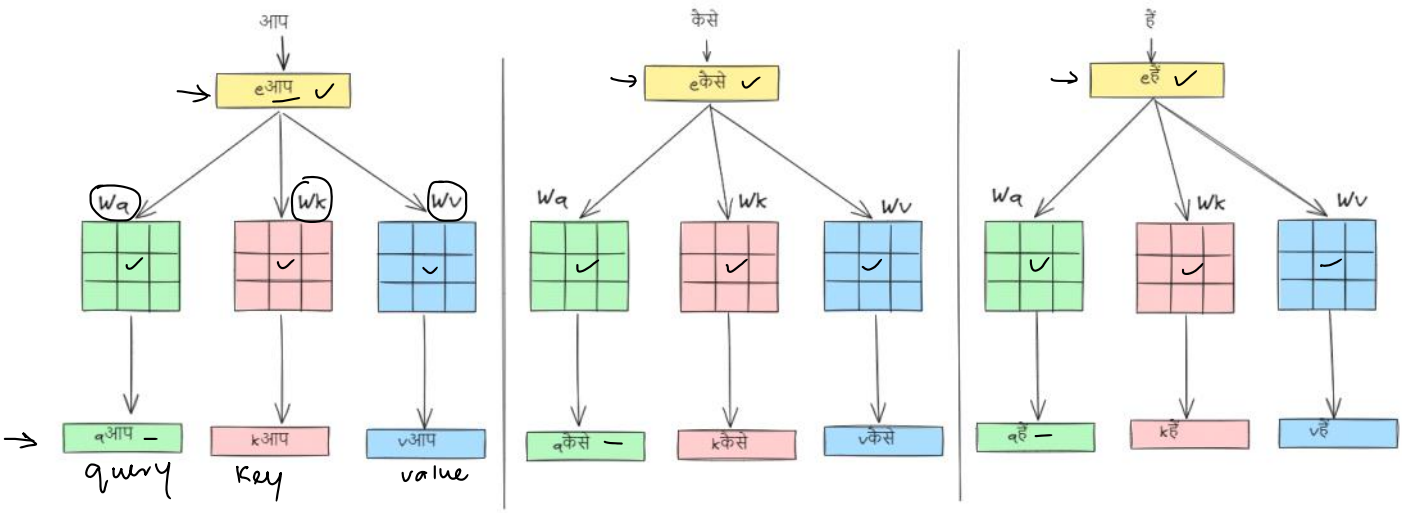
↓

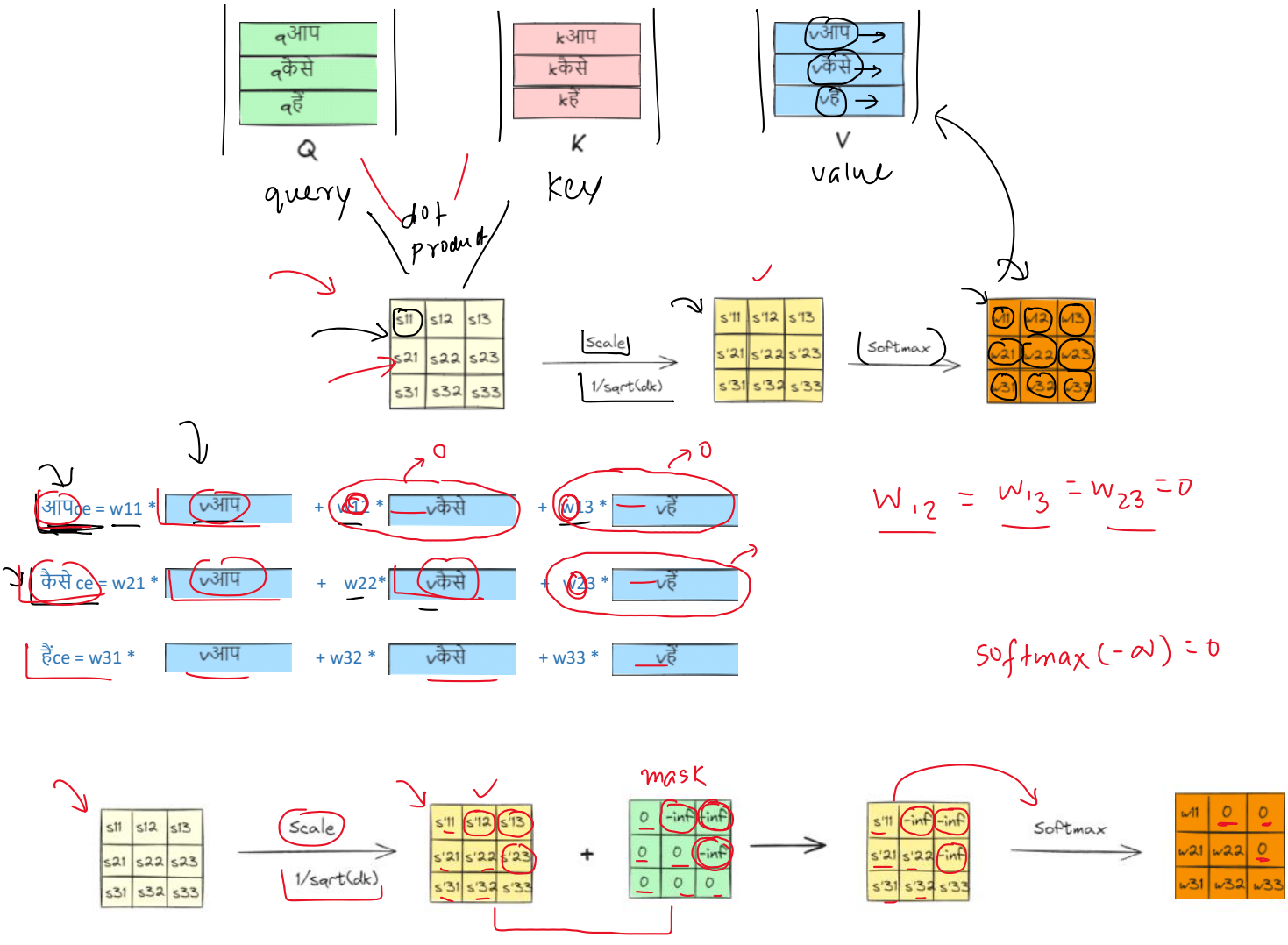
future token value


AR
NAR
Duvidha
→ No Data Leakage
→ slow
→ fast
→ Data Leakage
yes
self attention
how

Finding the answer

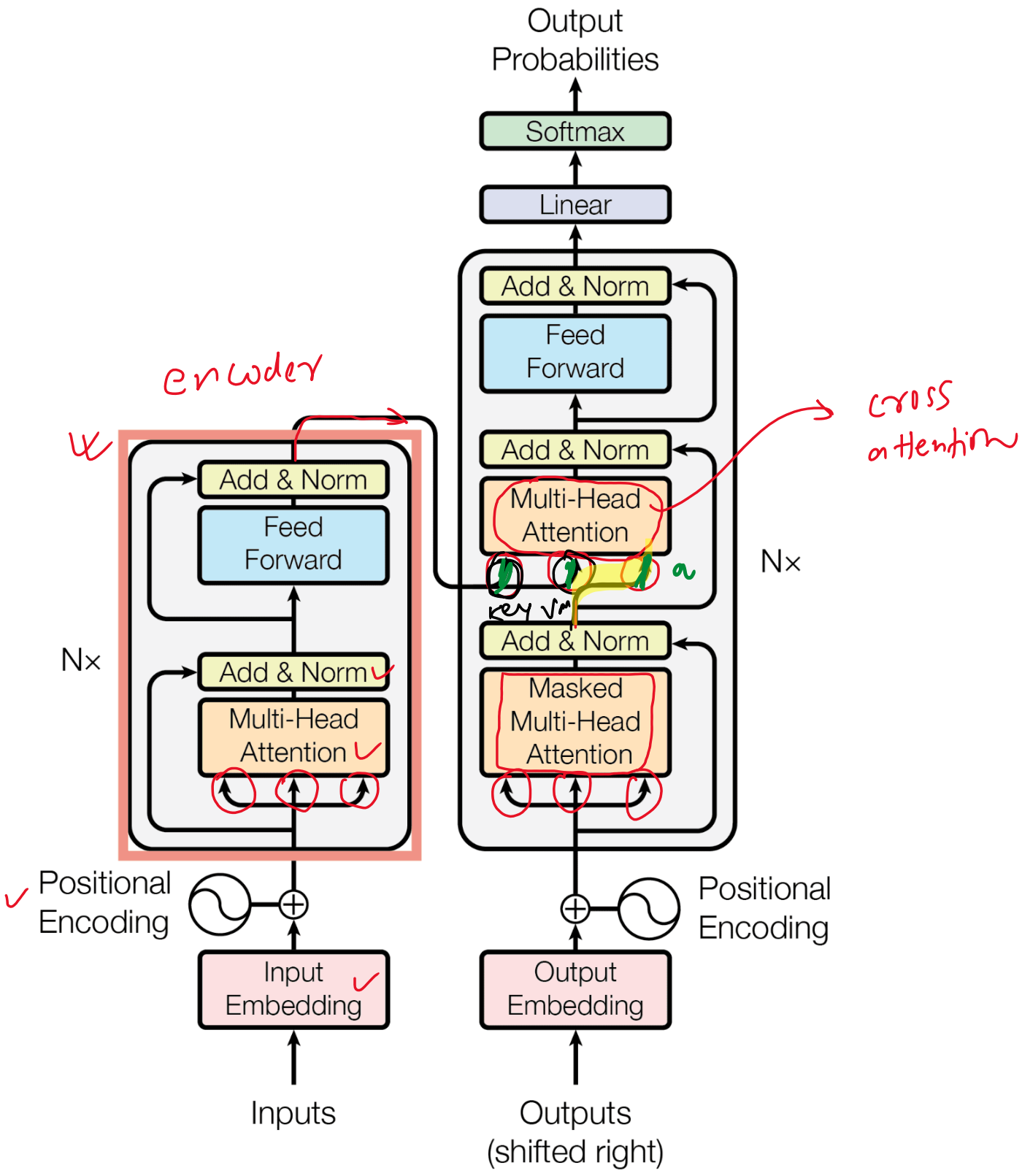
26 July 2024 00:21





Plan of Action

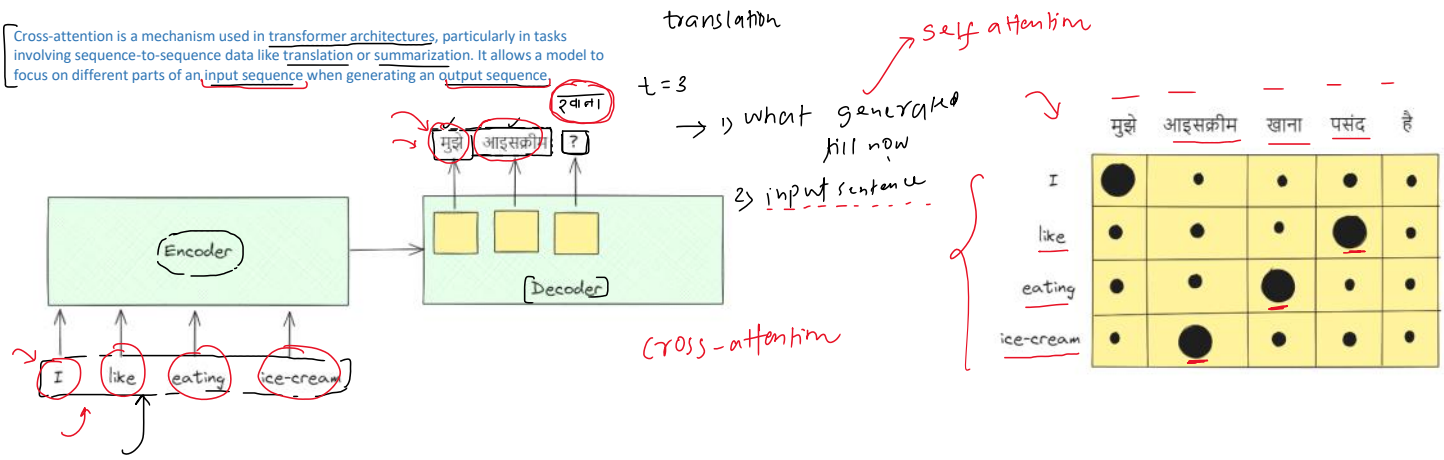
12 August 2024 17:53



What is Cross Attention

12 August 2024 17:53

Cross-attention is a mechanism used in transformer architectures, particularly in tasks involving sequence-to-sequence data like translation or summarization. It allows a model to focus on different parts of an input sequence when generating an output sequence.



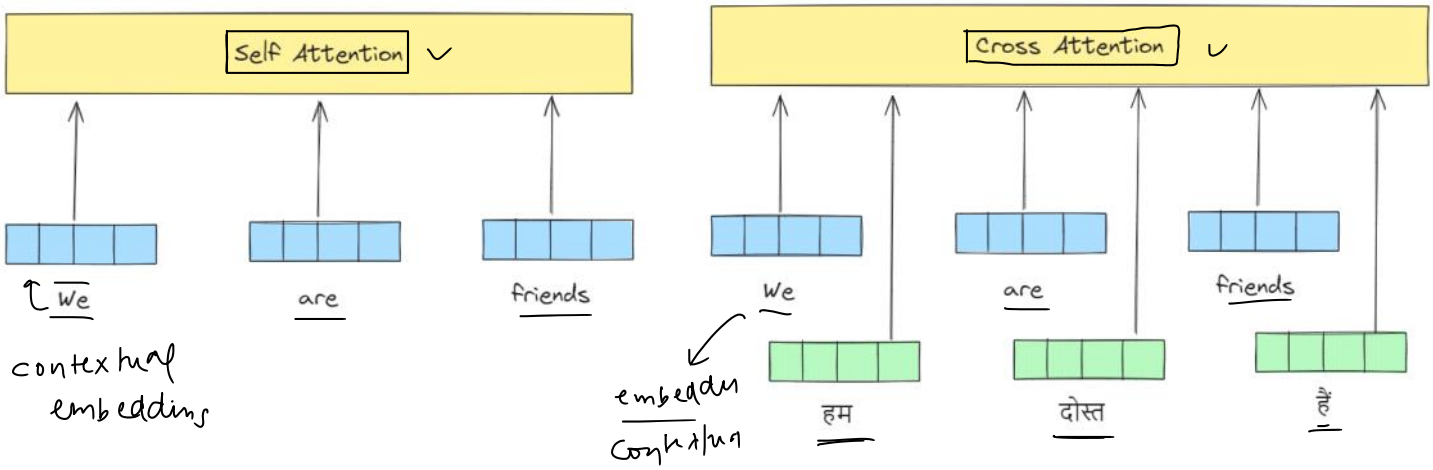
Cross Attention is conceptually very similar to Self-Attention

Self-Attention Vs Cross Attention

1. The input
2. The processing
3. The output

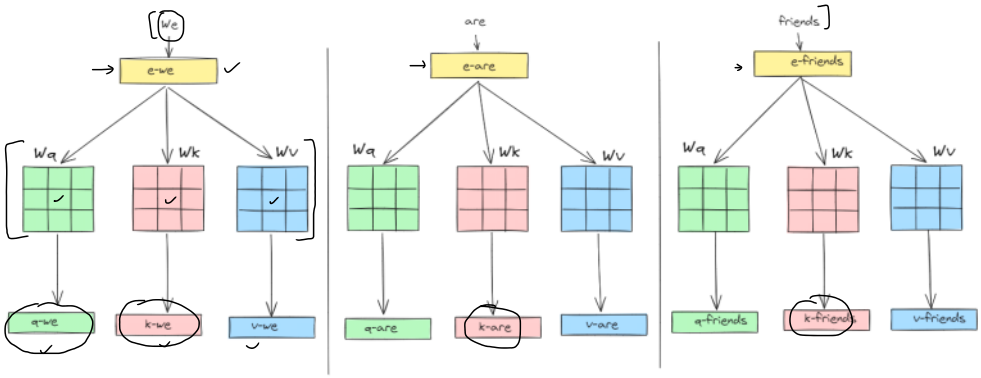
Self-Attention Vs Cross Attention (Input)

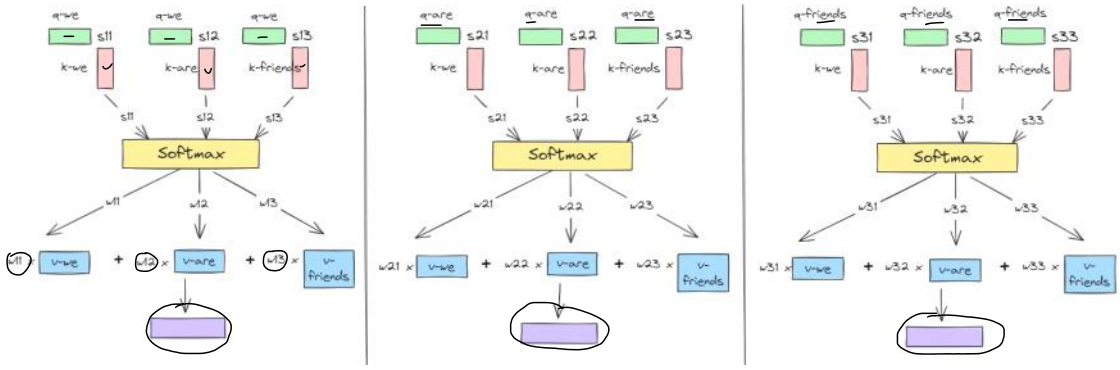
13 August 2024 08:22

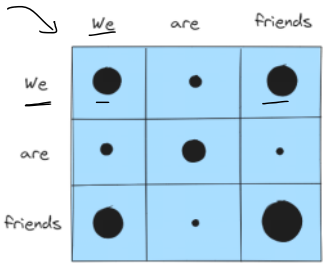


Self-Attention Vs Cross Attention (Processing)

12 August 2024 17:54

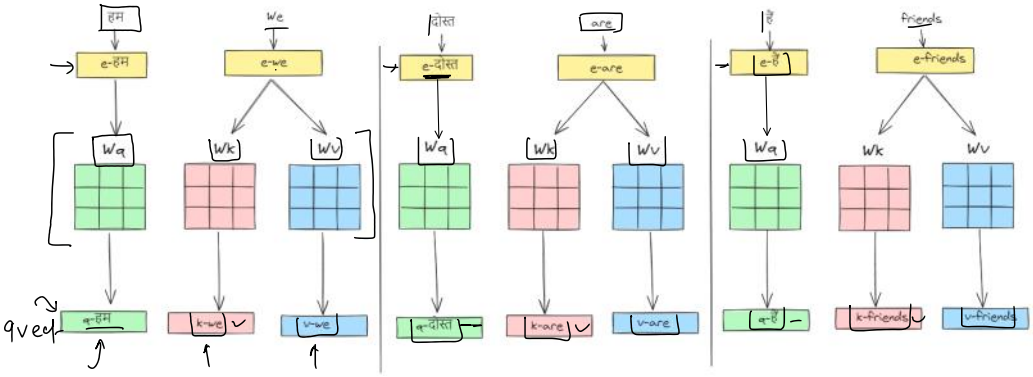


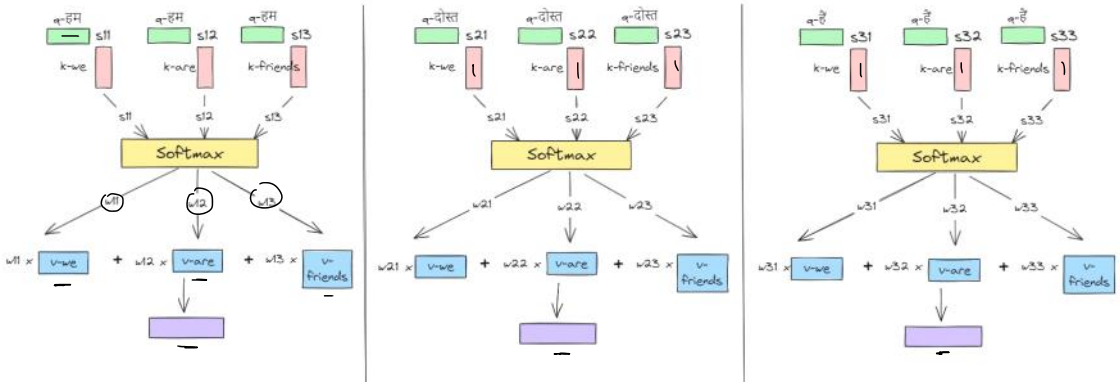


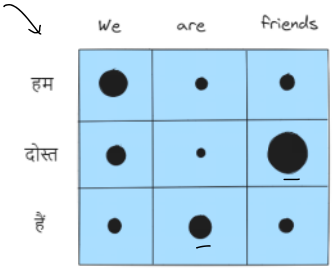


query → output seq eng - hindi

key and value - input seq

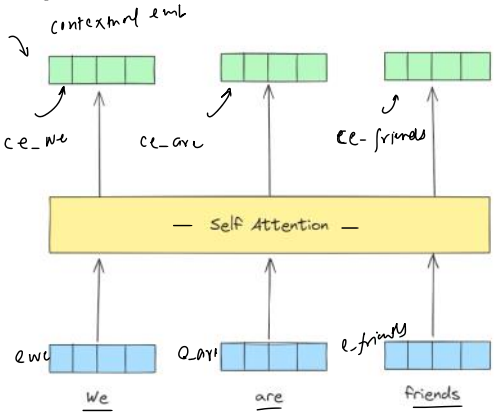


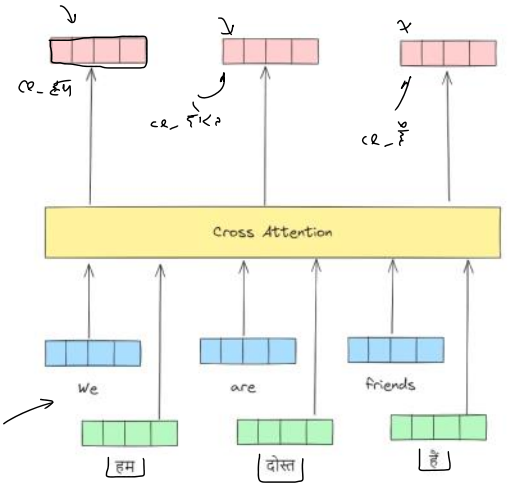




Self-Attention Vs Cross Attention [Output]

12 August 2024 18:05





$$\text{ce\_हम} = 0.5 \times \text{e\_we} + 0.3 \times \text{e\_are} + 0.2 \times \text{e\_friends}$$

$$\text{ce\_दोस्त} = 0.2 \times \text{e\_we} + 0.2 \times \text{e\_are} + 0.6 \times \text{e\_friends}$$

$$\text{ce\_हैं} = 0.3 \times \text{e\_we} + 0.4 \times \text{e\_are} + 0.3 \times \text{e\_friends}$$

$$\text{ce\_we} = 0.8 \times \text{e\_we} + 0.1 \times \text{e\_are} + 0.1 \times \text{e\_friends}$$

$$\text{ce\_are} = 0.15 \times \text{e\_we} + 0.75 \times \text{e\_are} + 0.1 \times \text{e\_friends}$$

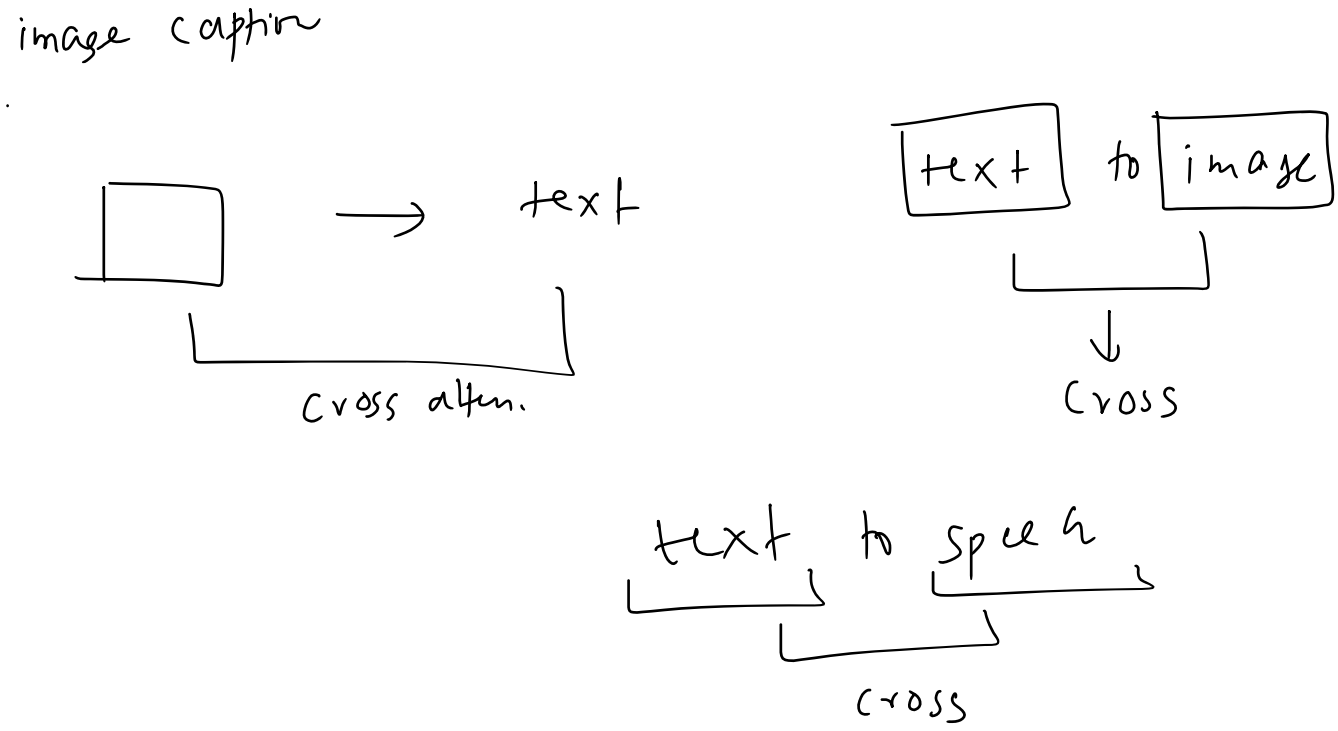
$$\text{ce\_friends} = 0.2 \times \text{e\_we} + 0.1 \times \text{e\_are} + 0.7 \times \text{e\_friends}$$

Cross Attention Vs Bahdanau/Luong Attention

12 August 2024 18:06

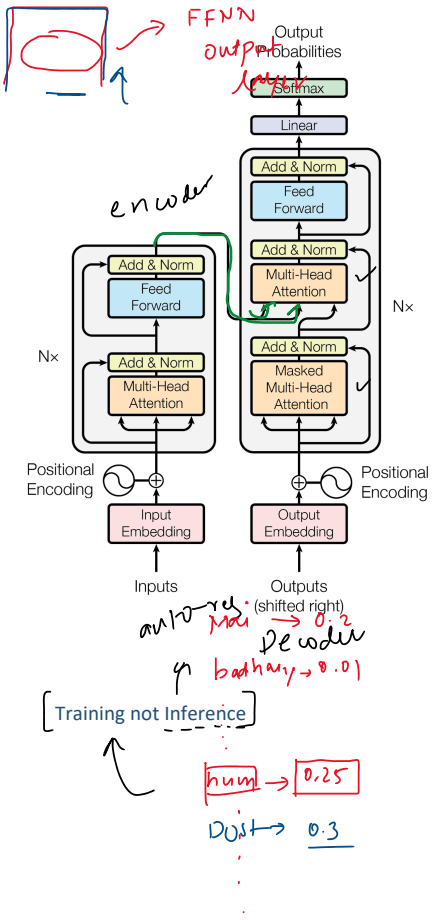
Use-cases

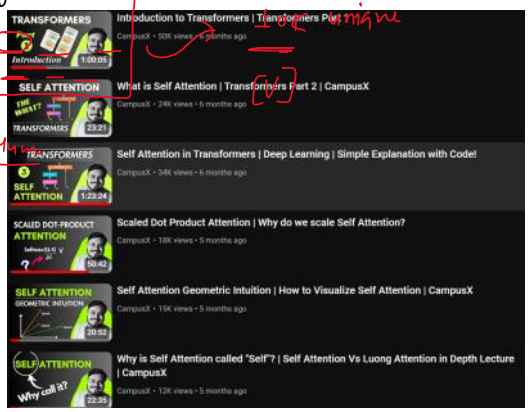
12 August 2024 18:07

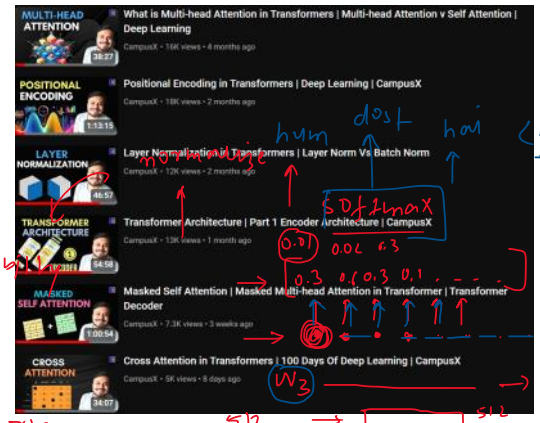


Plan of Attack

22 August 2024 15:36

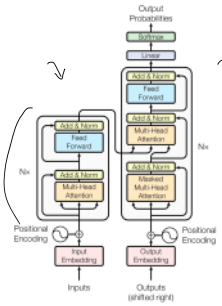


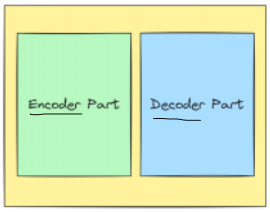




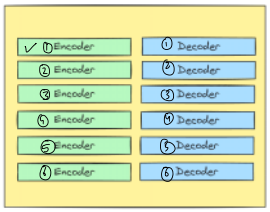
Simplified View

22 August 2024 00:33

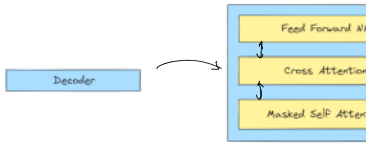


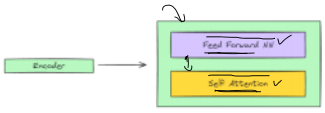


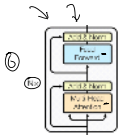
Transformer

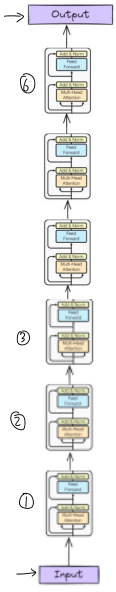


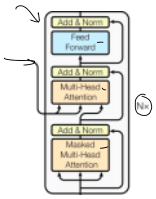
Transformer

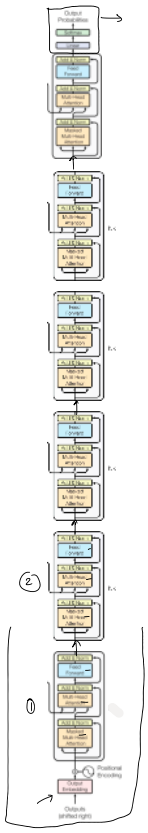








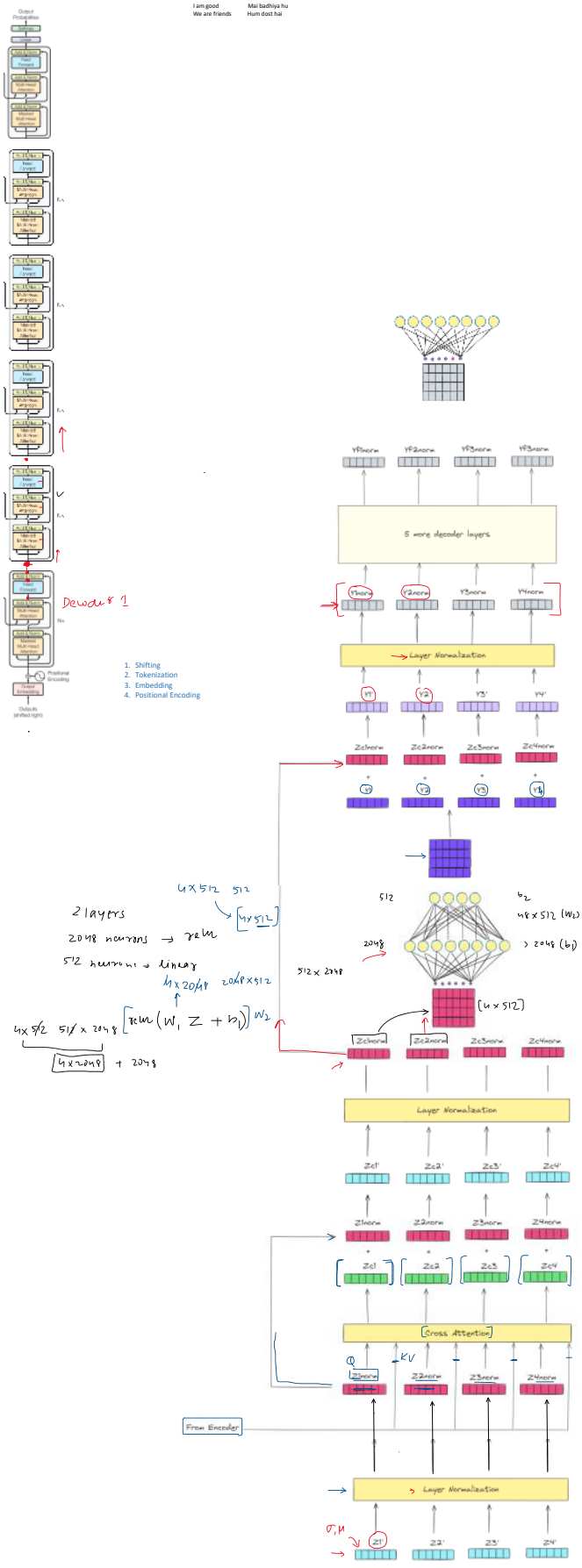

Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx

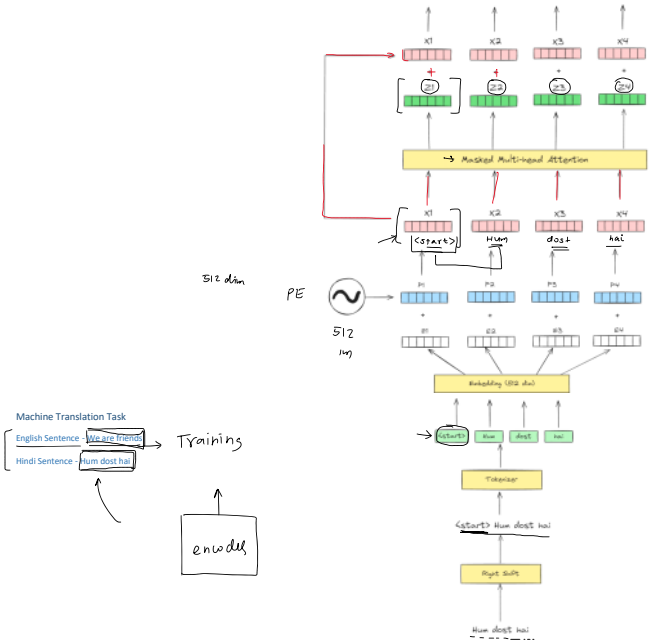

Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Nx
Nx
2
0
Positional Encoding
Output Embedding
Outputs (shifted right)

Decoder Architecture

22 August 2024 00:56

Eng	Hindi
I am good	Mai badhiya hu
We are friends	Hum dost hai

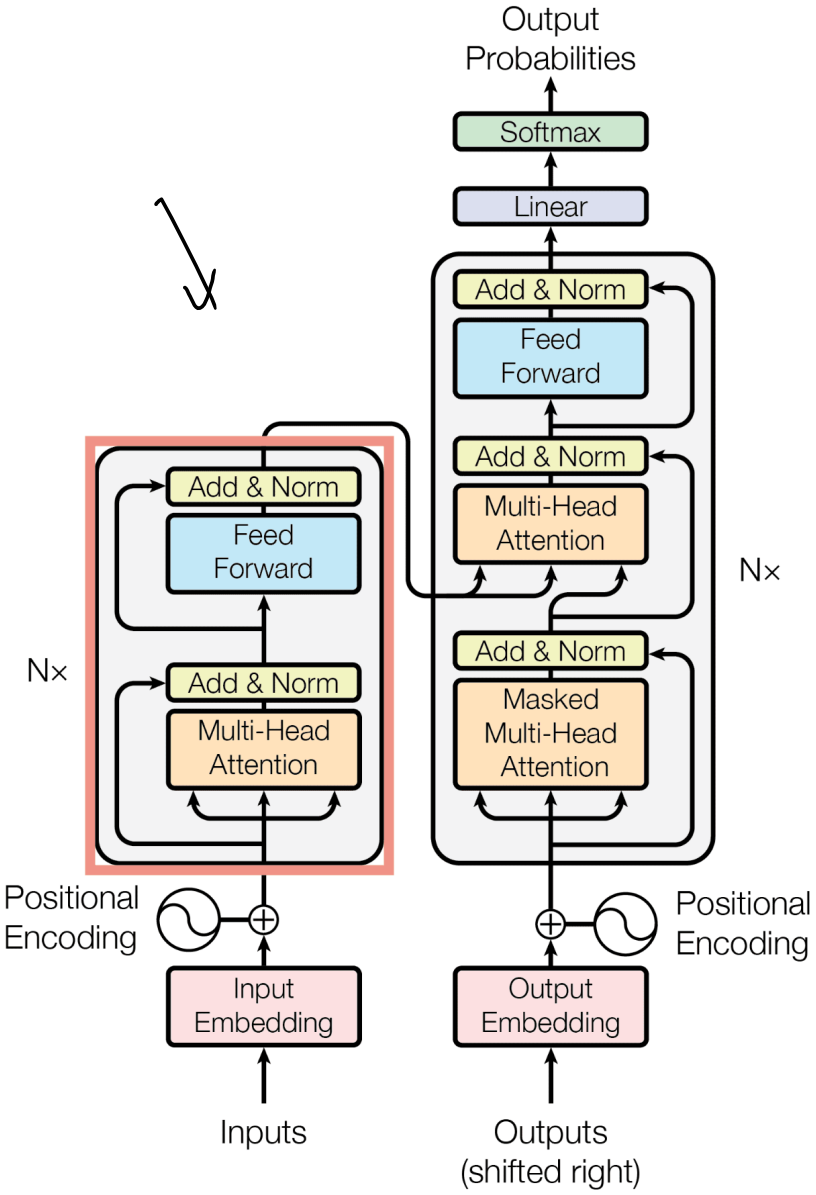



Machine Translation Task
English Sentence - We are friends
Hindi Sentence - Hum dost hai
Training
encoder
512 dim
PE
512
Masked Multi-head Attention
<start>
Hum
dost
hai
Embedding (512 dim)
Tokenizer
<start> Hum dost hai
Right Shift
Hum dost hai

Plan of Attack

03 September 2024 00:37

training

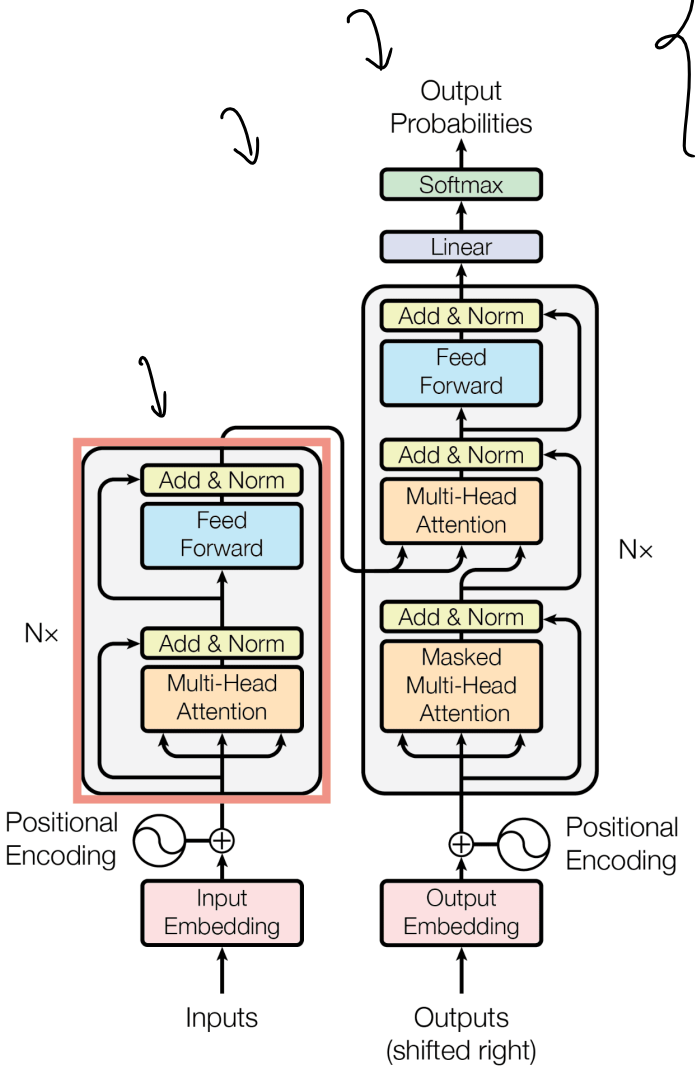


The Setup

03 September 2024 00:39

Llama

S no	English Sentence	Hindi Sentence
1	How are you	Aap kaise ho
2	Thank you	Dhanyawad
3	You are welcome	Apka swagat hai



Query Sentence

We are friends → ?

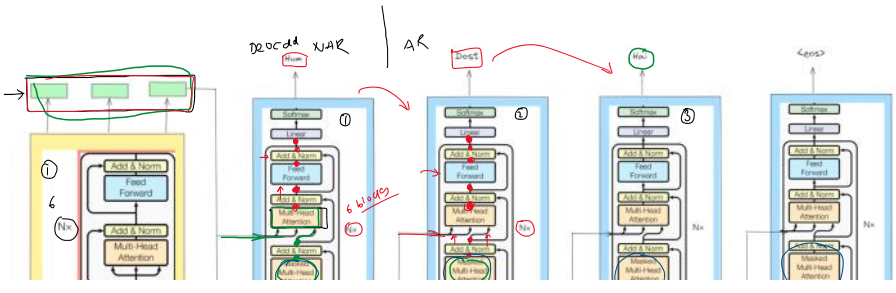
Transformer during Inference

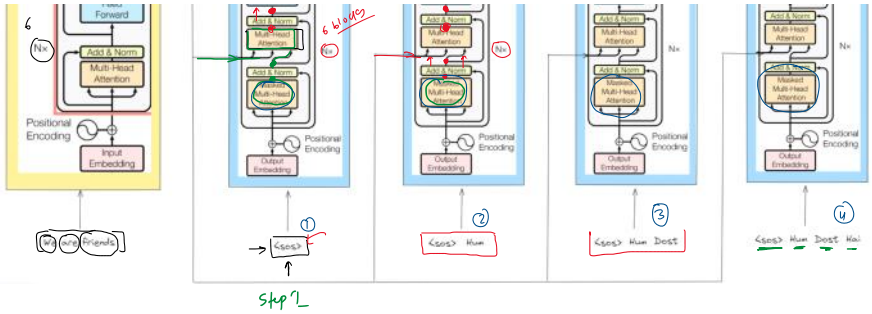
03 September 2024 00:50

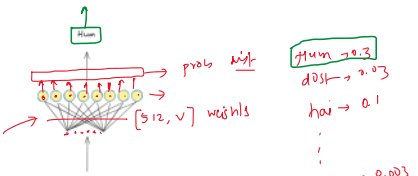
Training | Inference

Encoder same same

R





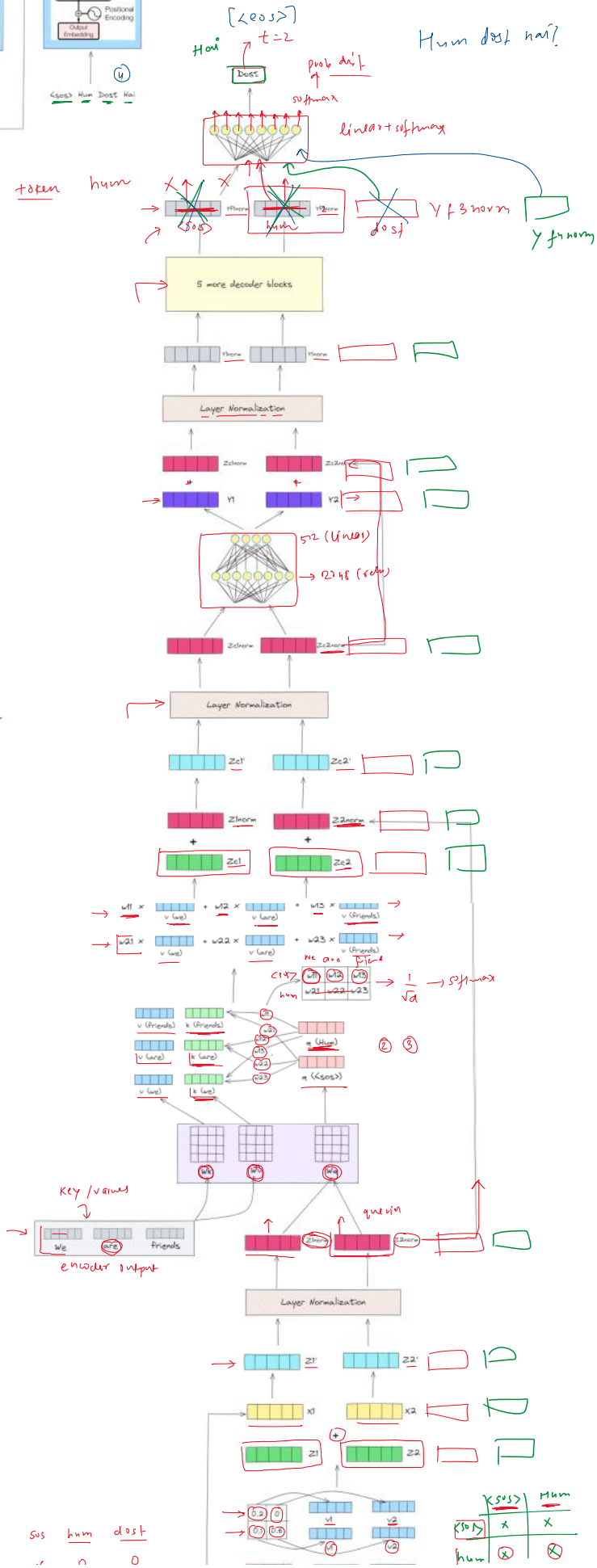


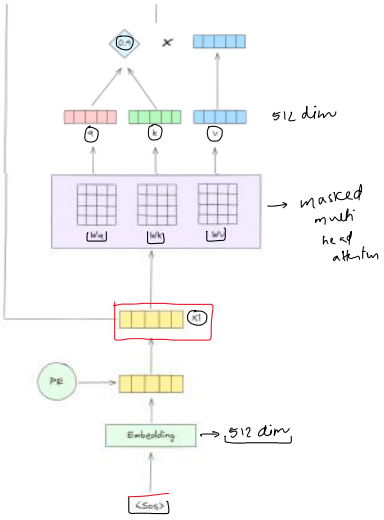
(1, 512) (512, 2048)

(1, 2048) + b

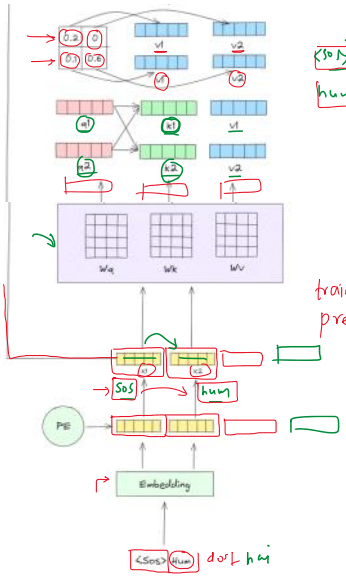
relu((1, 2048)) × (2048, 512)

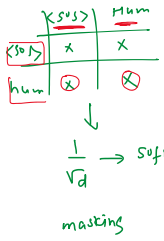
(1, 512)





	sos	hum	dost
sos	X	0	0
hum	X	X	0
dost	X	X	X





training → masking
pred → masking X