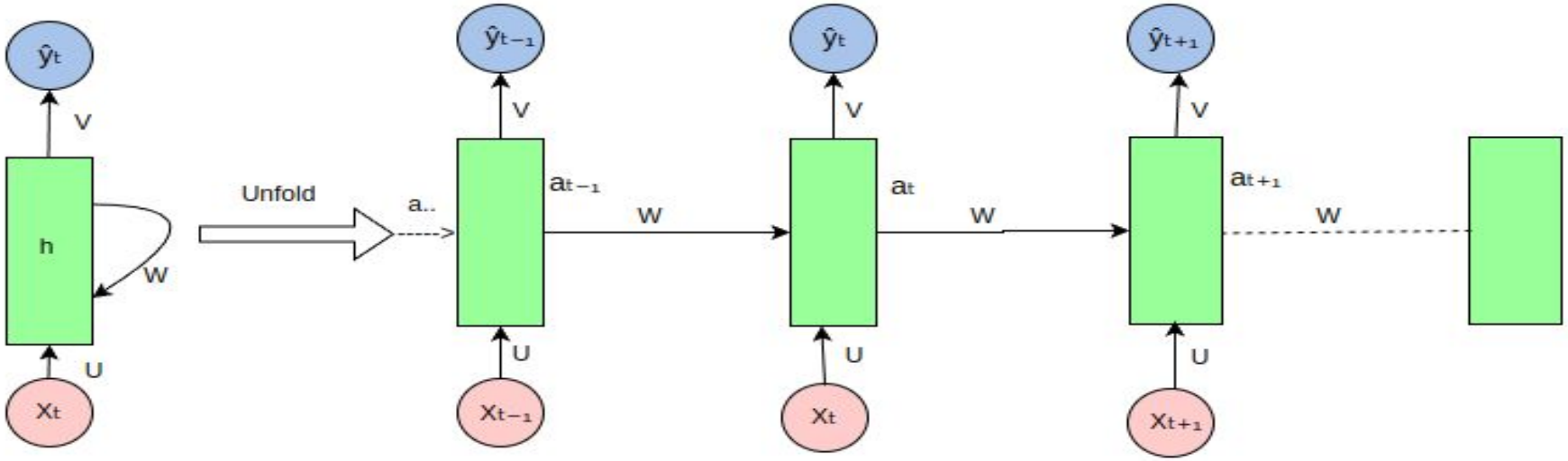
Llama 2

By
Roshan Prasad
Regd No 23364

Natural Language Processing

- RNN



Challenges

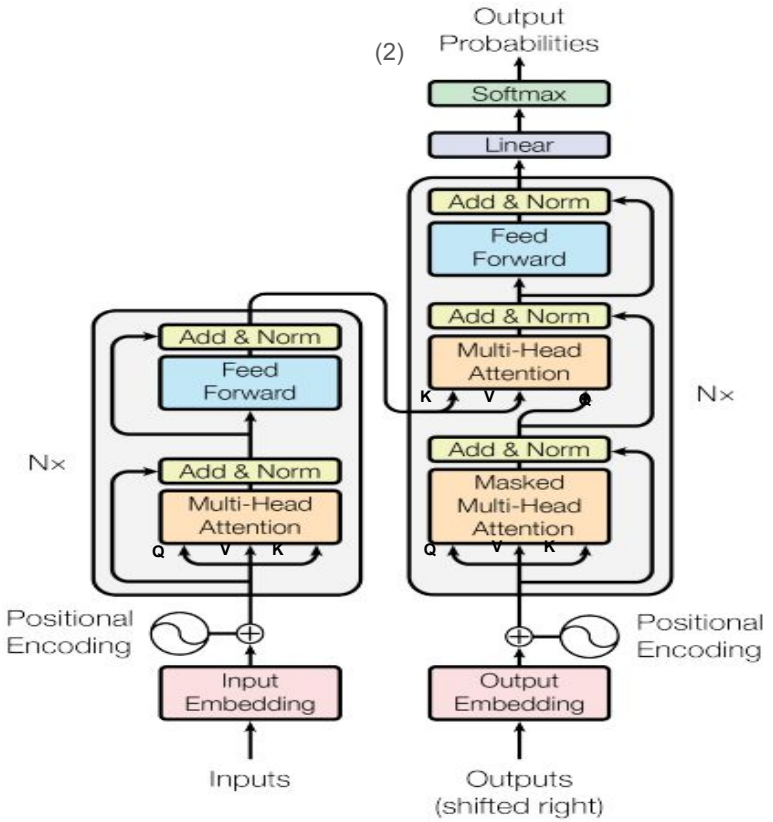
- Vanishing and Exploding Gradients
- Capturing Long-Term Dependencies or the Context of input
- Sequential Nature and Lack of Parallelism

Transformer

- Encoder-Decoder Structure
- Self-Attention
- Parallel Processing

Benefits

- Captures long-range dependencies
- Avoid Sequential Computation
- Context is preserved



Llama 2

- Llama 2 is a family of **Large Language Model(LLMs)**
- Llama 2 is an **auto-regressive language model**
- First Release , **July 18, 2023,** in partnership with **Microsoft**, **Meta** and **Open-source Large Language Models**
- Llama 2 pretrained models are trained on **2 trillion tokens** and have the double the context length than Llama 1
- Three models sizes were trained : **7, 13, 70 billion** parameters

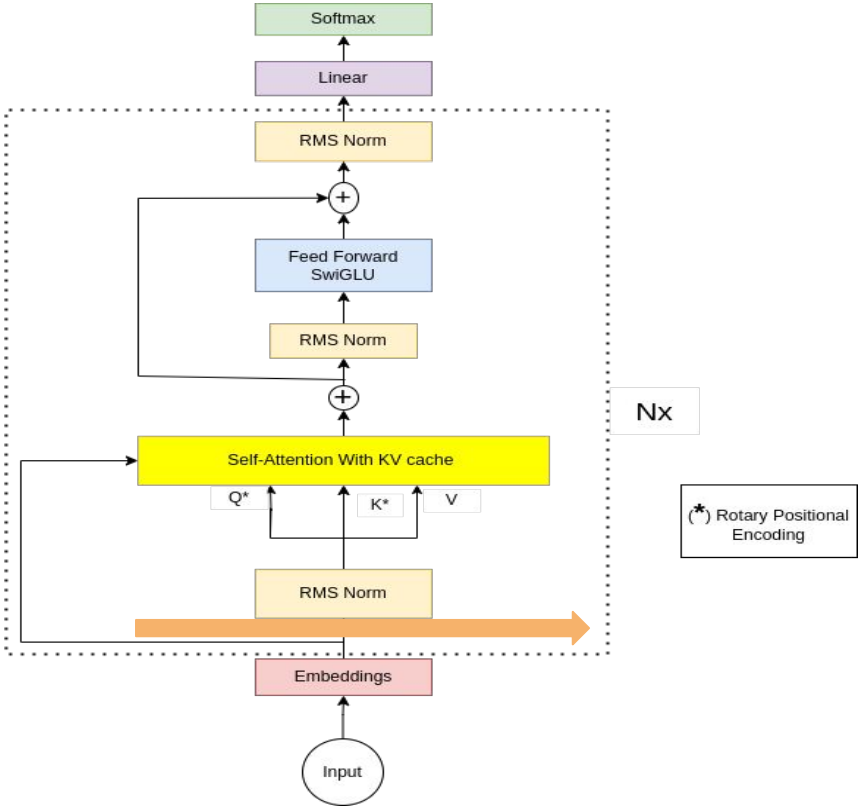
Llama 2 -can do

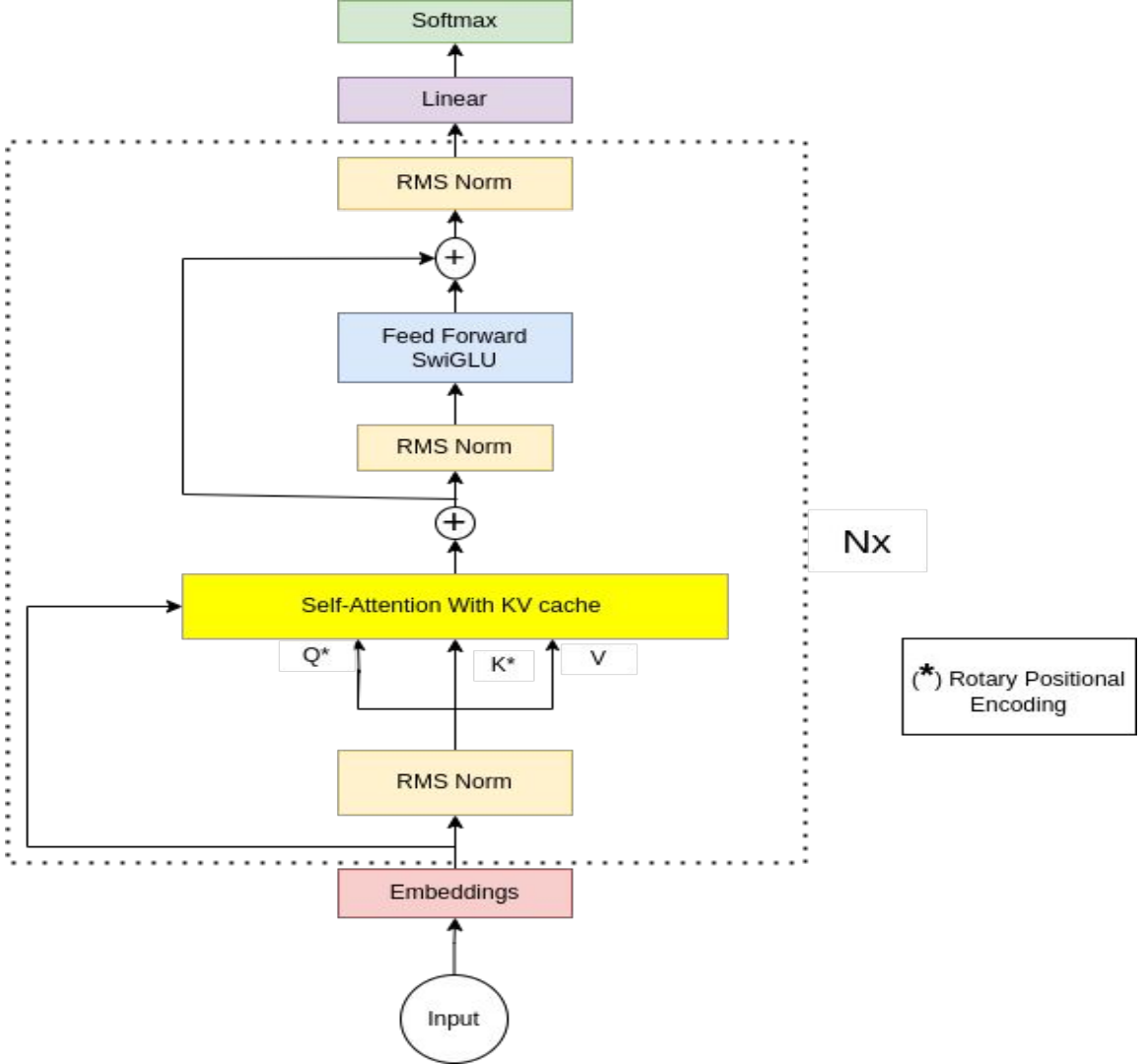
- Generate creative text
- Translate languages
- Answer your questions in an informative way
- Help with coding tasks.
- Generate dialogue for chatbots

Llama 2 -Improvements over Llama 1

- **Increased Training or Tokens:** trained on 40% more tokens
- **Longer context length : 4k tokens** from 2k tokens
- **Fine-Tuning for Dialogues:** The versions of Llama 2 that are fine-tuned (Labeled Llama 2-Chat) are aimed at being optimized for dialogue

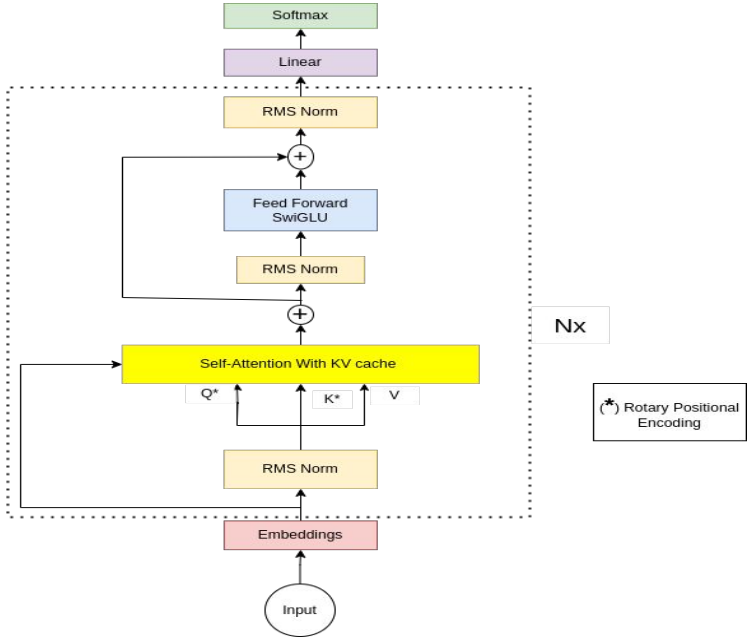
Llama 2 - Architecture




Softmax
Linear
RMS Norm
Feed Forward
SwiGLU
RMS Norm
Self-Attention With KV cache
Q*
K*
V
RMS Norm
Embeddings
Input
Nx
(*) Rotary Positional
Encoding

Next Token Prediction task :inference

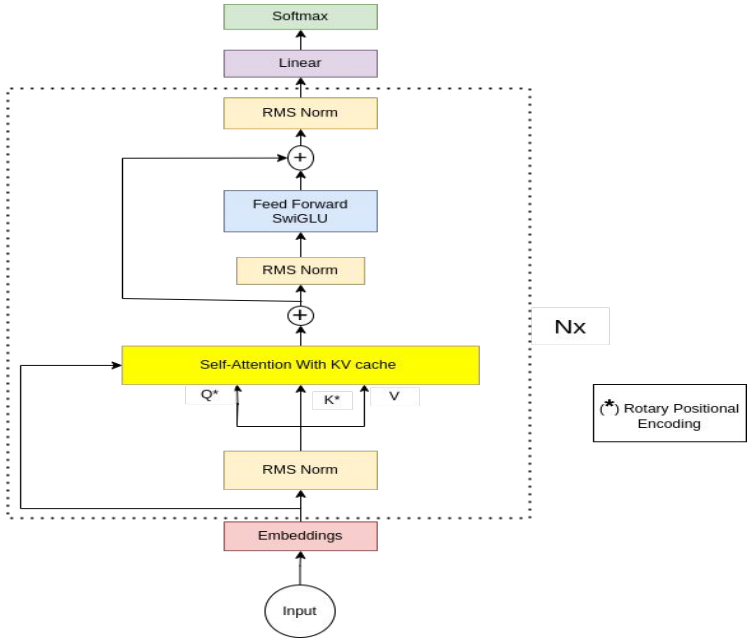
Input: [SOS]your cat is a good



Next Token Prediction task :inference

Output:your

Inference
T=1

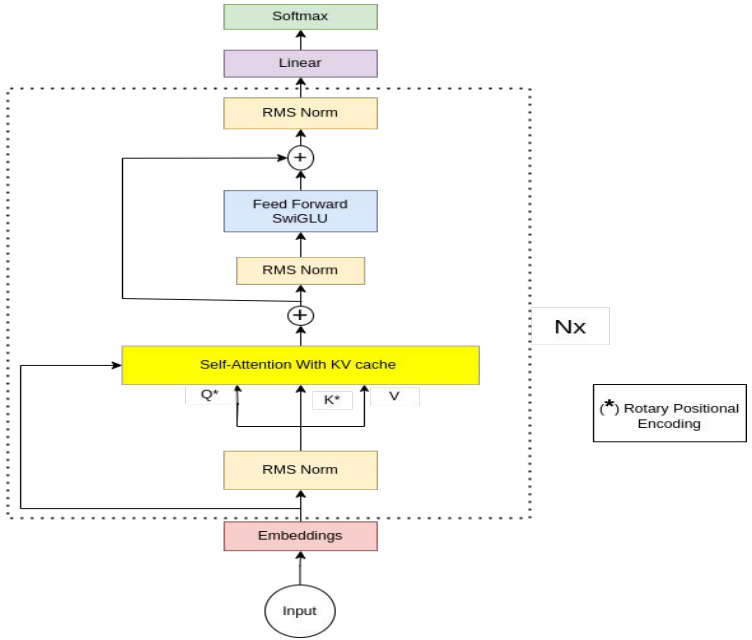


Input: [SOS]

Next Token Prediction task :inference

Output:your cat

Inference
T=2

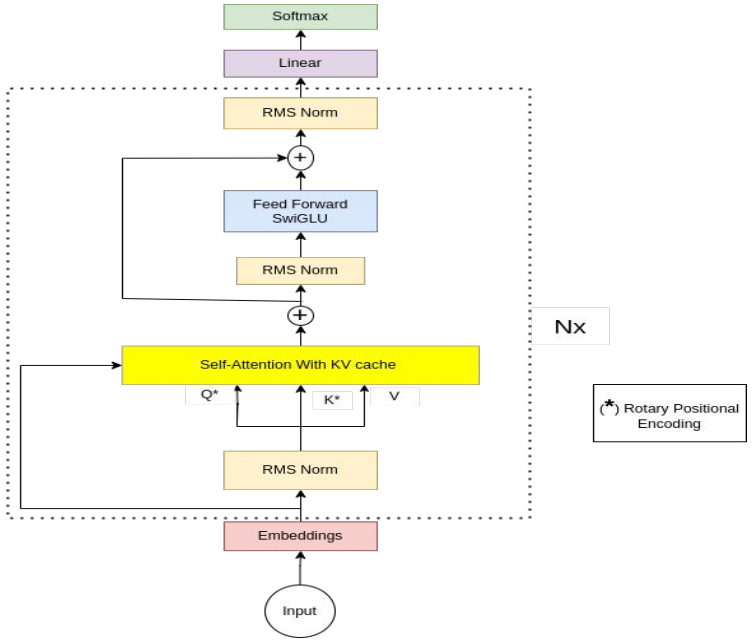


Input: [SOS] your

Next Token Prediction task :inference

Inference
T=3

Output:your cat is

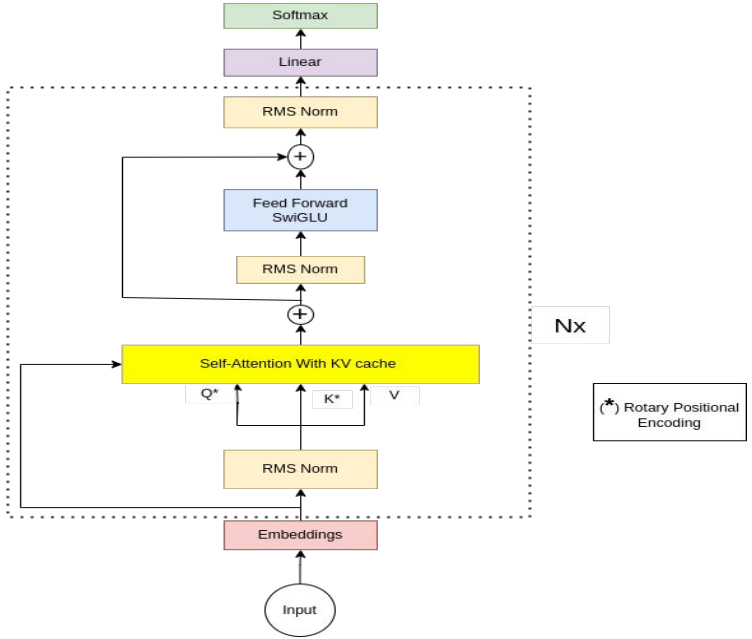


Input: [SOS] your cat

Next Token Prediction task :inference

Inference
T=4

Output:your cat is a

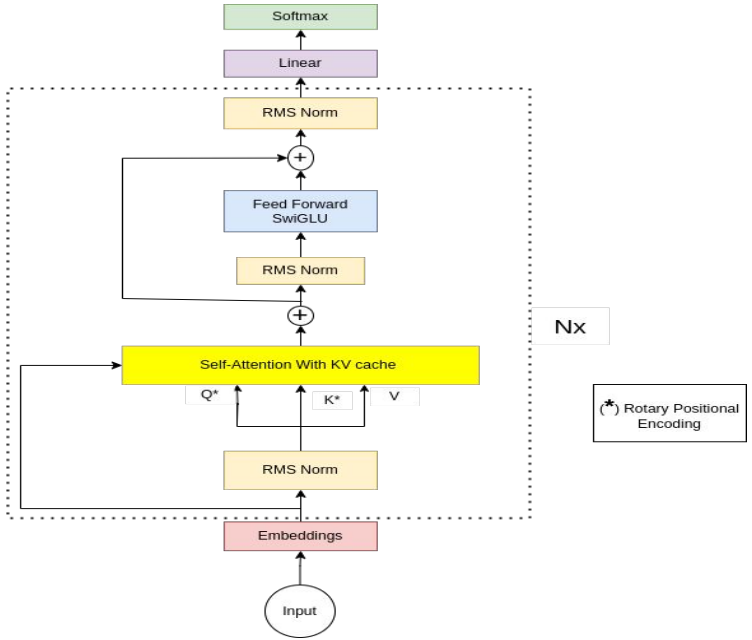


Input: [SOS] your cat is

Next Token Prediction task :inference

Inference
T=5

Output:your cat is a good

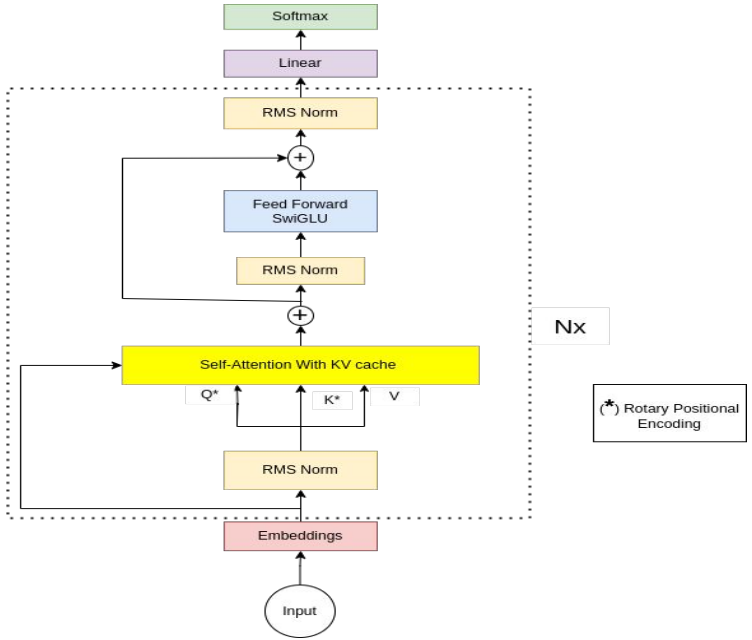


Input: [SOS]your cat is a

Next Token Prediction task :inference

Inference
T=6

Output:your cat is a good cat

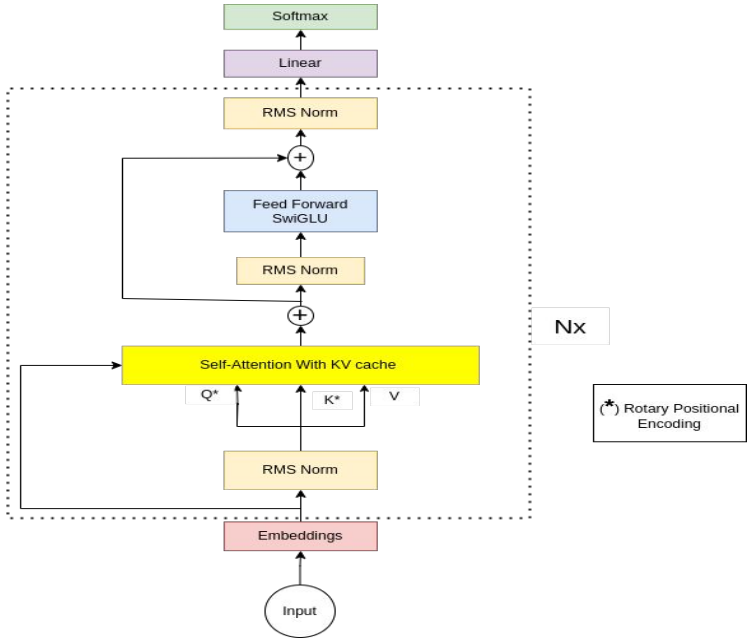


Input: [SOS]your cat is a good

Next Token Prediction task :inference

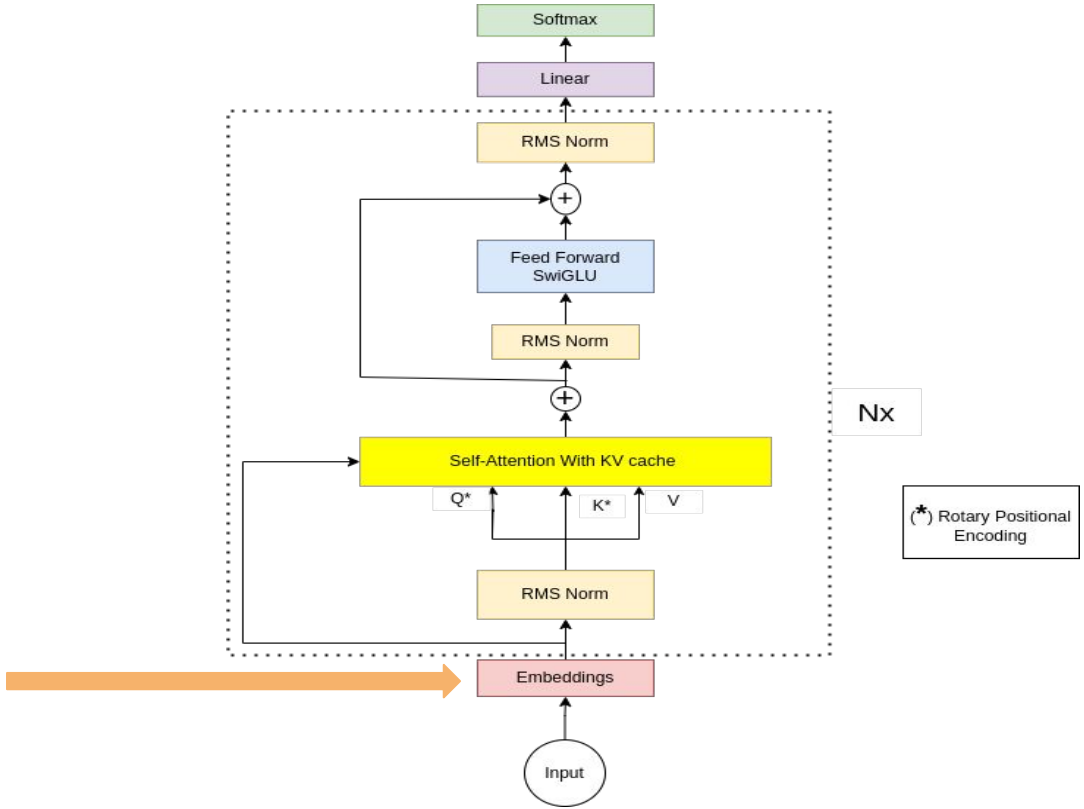
Inference
T=7

Output:your cat is a good cat [EOS]

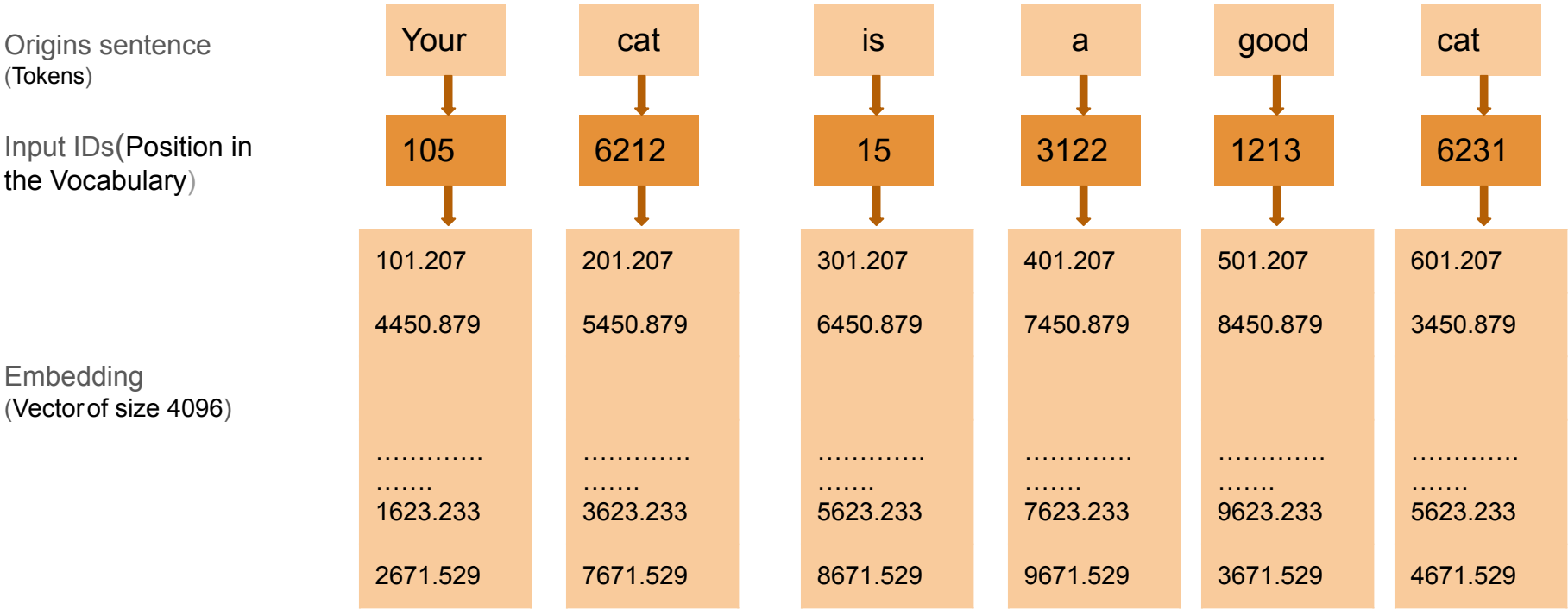


Input: [SOS]your cat is a good cat

Llama 2 - Architecture



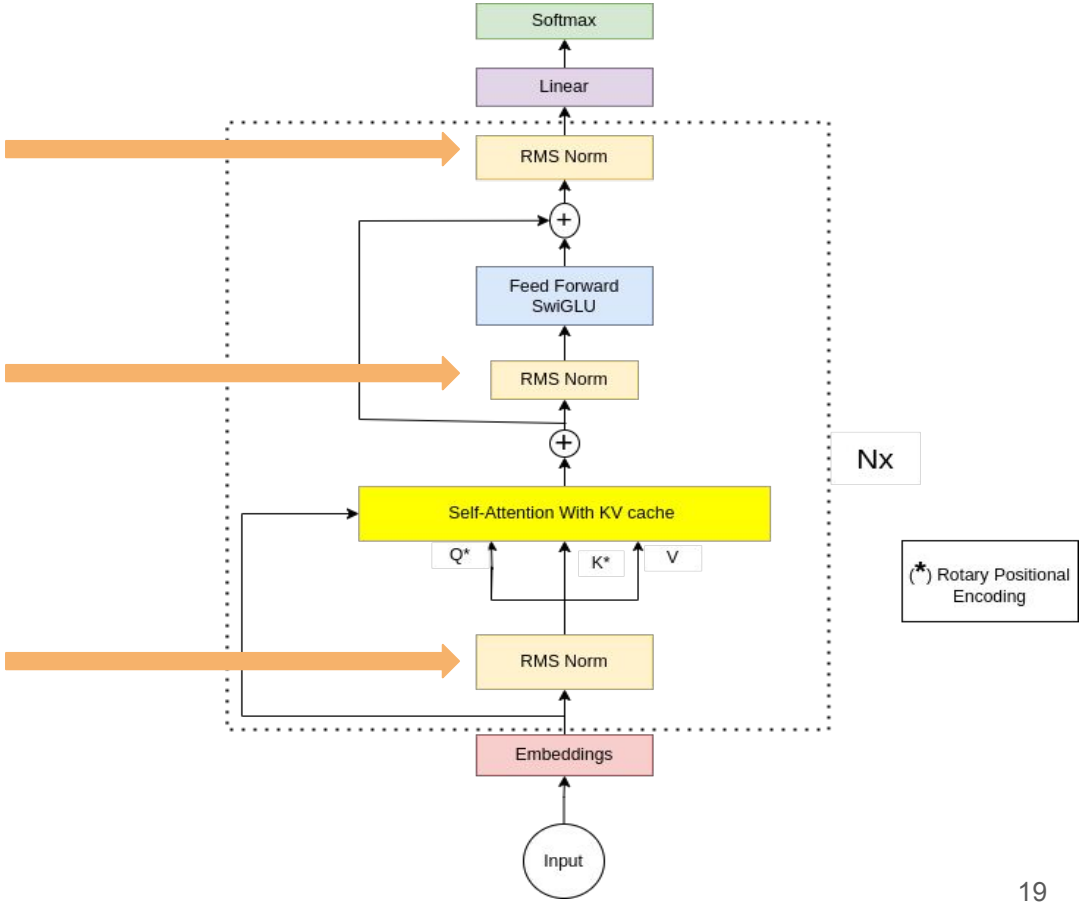
Embedding


Origins sentence
(Tokens)
Input IDs(Position in the Vocabulary)
Embedding
(Vectorof size 4096)
Your
cat
is
a
good
cat
105
6212
15
3122
1213
6231
101.207
4450.879
.....
1623.233
2671.529
201.207
5450.879
.....
3623.233
7671.529
301.207
6450.879
.....
5623.233
8671.529
401.207
7450.879
.....
7623.233
9671.529
501.207
8450.879
.....
9623.233
3671.529
601.207
3450.879
.....
5623.233
4671.529

RMS Norm

- In the batch normalization where we need to calculate the mean and variance of each Batch Normalization but in case of RMS Normalization we does not require to calculate the variance reducing the number of computation

$$\bar{a}_i = \frac{a_i}{\mathrm{RMS}(\mathbf{a})} g_i, \quad \text{where } \mathrm{RMS}(\mathbf{a}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} a_i^2}.$$



RMS Norm

Let us take a matrix of size (5,5)

13	26	35	43	51
61	77	54	25	31
27	17	67	16	79
31	57	61	32	72
14	17	12	63	33

RMS(ai)

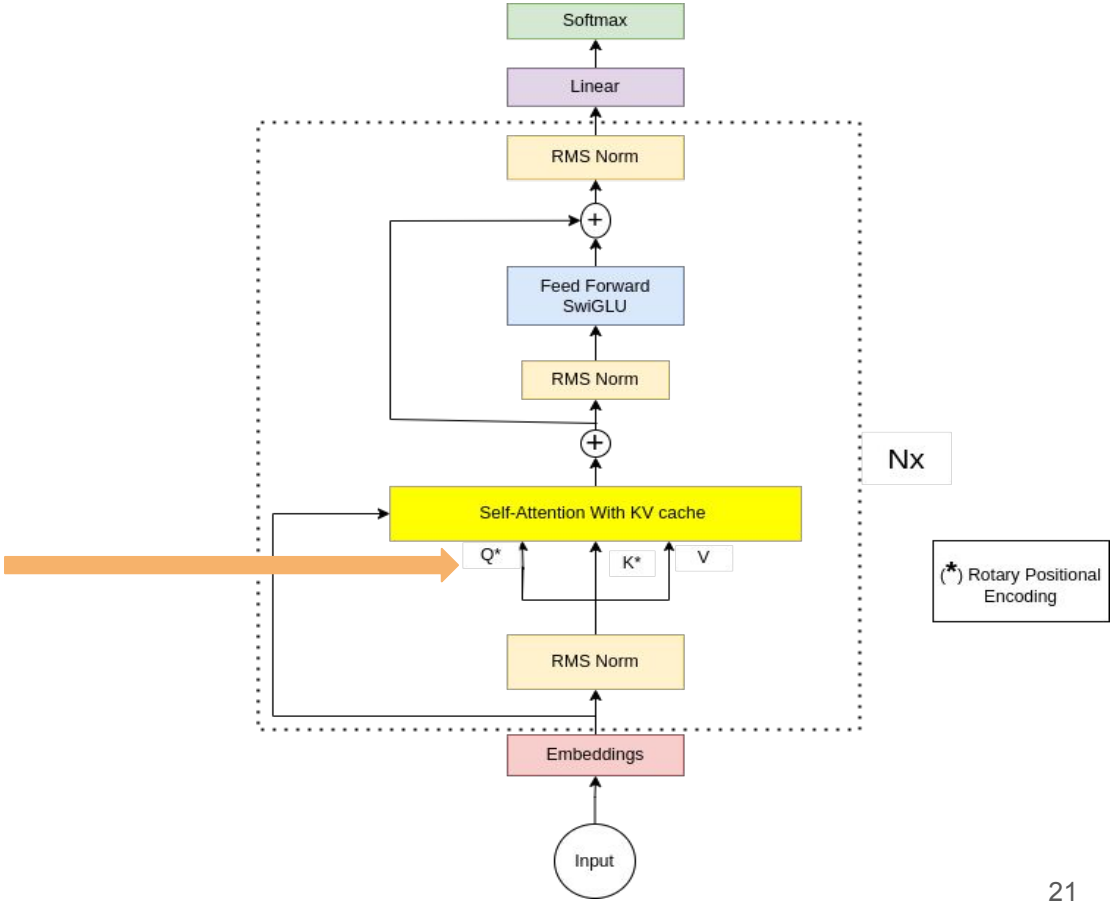
38.77
51.29
48.99
53.17
35.46

0.335 *g1	0.76 *g2	0.92 *g3	1.108 *g4	1.315 *g5
1.187 *g6	1.502 *g7	1.054 *g8	0.488 *g9	0.604 *g10
0.551 *g11	0.347 *g12	1.367 *g13	0.327 *g14	1.613 *g15
0.582 *g16	1.071 *g17	1.148 *g18	0.602 *g19	1.355 *g20
0.395 *g21	0.480 *g22	0.339 *g23	1.775 *g24	0.931 *g25

$$\bar{a}_i = \frac{a_i}{\mathrm{RMS}(\mathbf{a})} g_i, \quad \text{where } \mathrm{RMS}(\mathbf{a}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} a_i^2}.$$

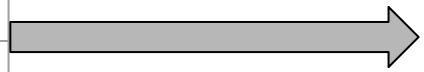
Each gi are learnable parameters

Rotary Positional Encoding



Rotary Positional Encoding

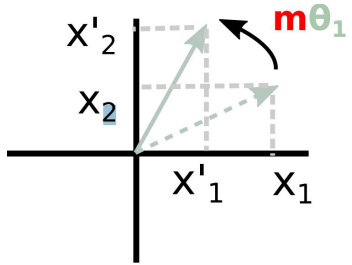
0.231	0.543	0.756	0.871	0.815
0.872	0.502	0.454	0.488	0.704
0.551	0.347	1.367	0.327	0.613
0.582	0.710	0.548	0.602	0.355
0.395	0.480	0.339	0.775	0.931



0.213	0.132	0.332	0.431	0.716
0.532	0.314	0.521	0.512	0.610
0.395	0.951	0.632	0.675	0.341
0.689	0.560	0.654	0.609	0.562
0.392	0.455	0.789	0.575	0.153

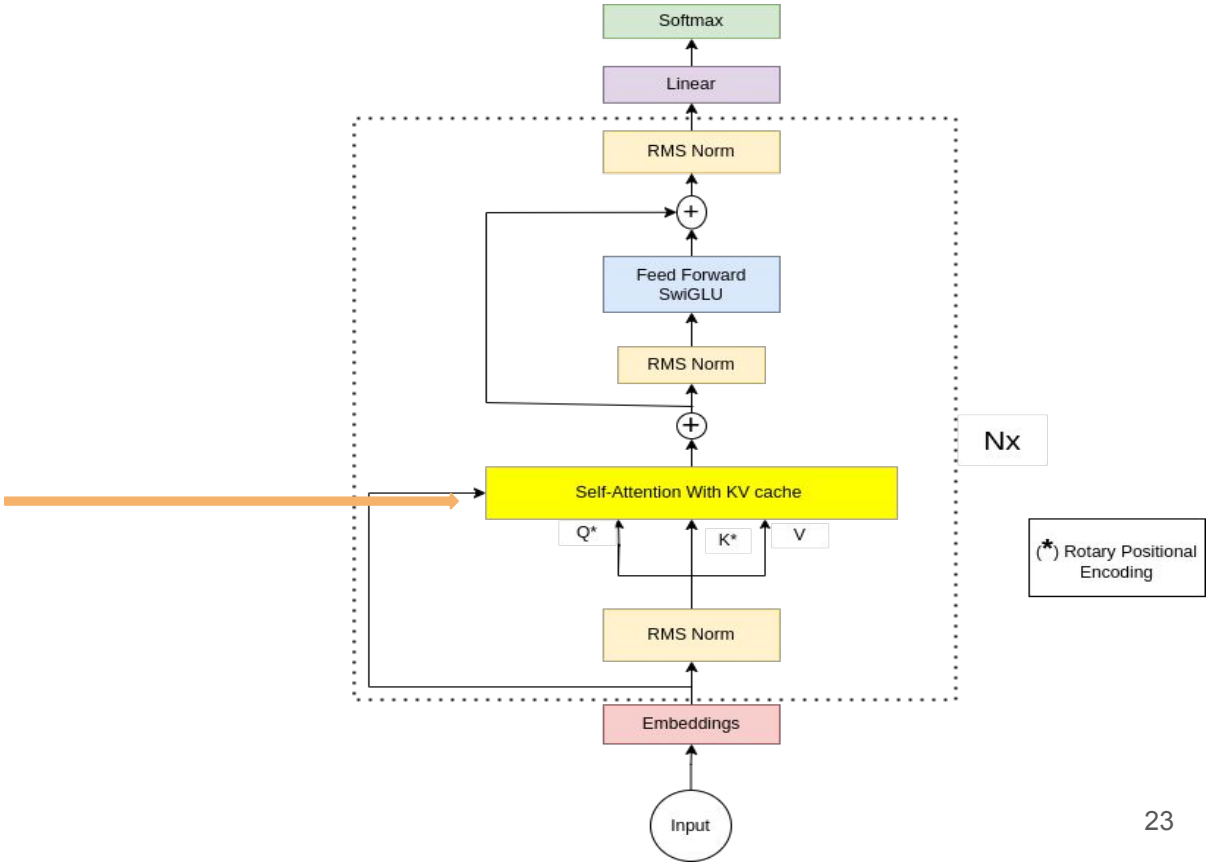
If we have the vector of size 2 then we multiply the given matrix to rotate it by angle .

$$\begin{pmatrix} \cos m\theta & -\sin m\theta \\ \sin m\theta & \cos m\theta \end{pmatrix}$$



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Self Attention using KV cache



Self-Attention using KV Cache

What is the Self Attention?

Qk/sqtr(4096)

	your	Cat	Is	A	Good	cat
you	0.23	0.13 43	0.23 4	0.12 3	0.23 4	0.45 2
cat	0.53 22	0.54 43	0.56 23	0.63 42	0.23 21	0.24 34
is	0.34 32	0.45 32	0.63 24	0.12 42	0.63 21	0.23 43
a	0.23 23	0.62 23	0.44 31	0.34 50	0.23 41	0.23 40
good	0.12 1	0.45 32	0.63 24	0.45 32	0.63 24	0.54 43
cat	0.23	0.23	0.23	0.23	0.23	0.23

X

V

(6,4096)

=

Attention

(6,4096)

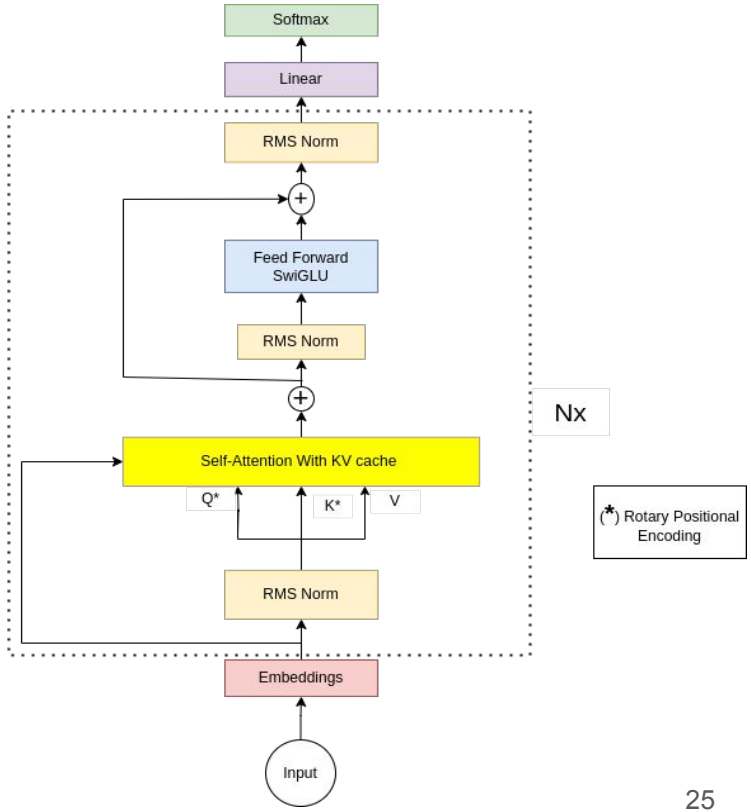
Self Attention using KV cache

Next Token Prediction task

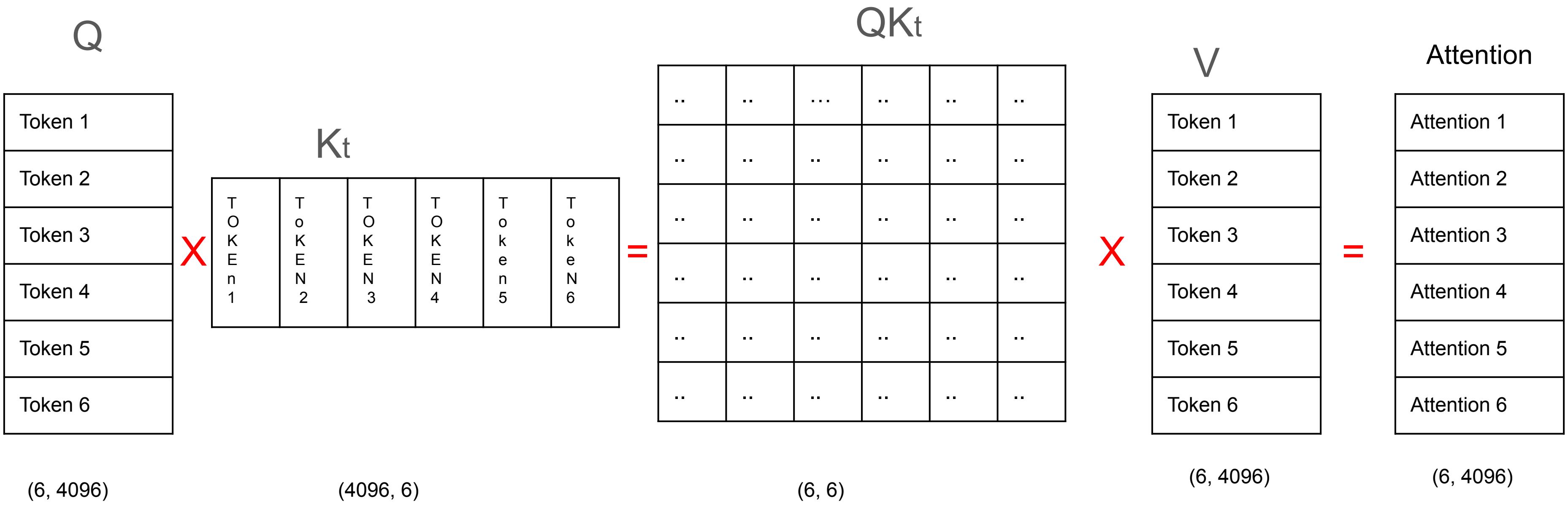
Let's take the example

Target : your cat is a good cat[EOS]

Input : [SOS] your cat is a good

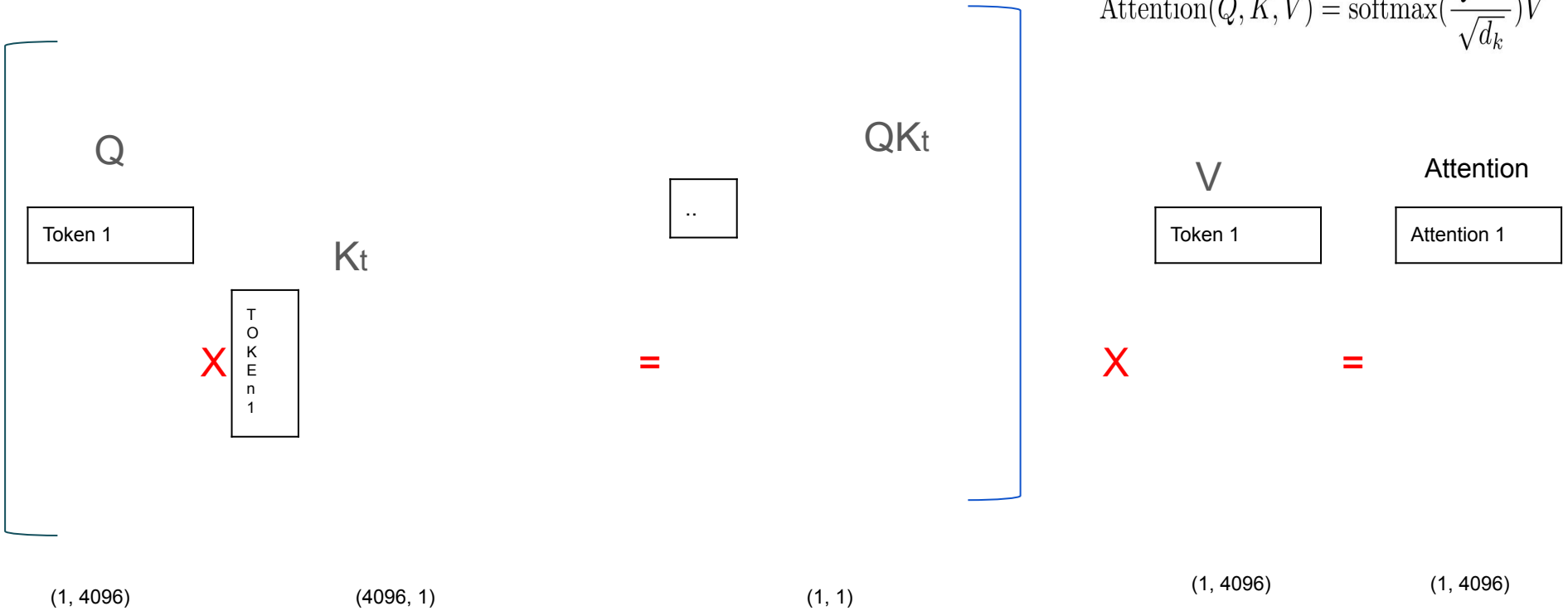


Self- Attention during Next Token prediction task



Self- Attention during Next Token prediction task

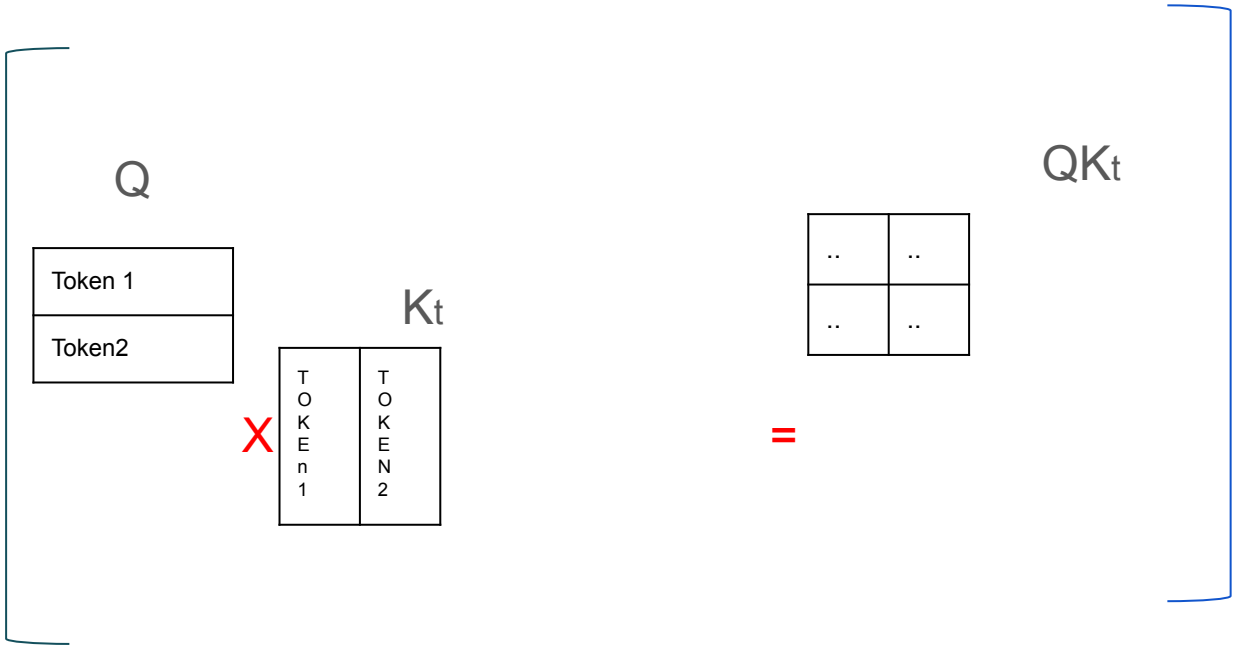
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



Inference T =1

Self- Attention during Next Token prediction task

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

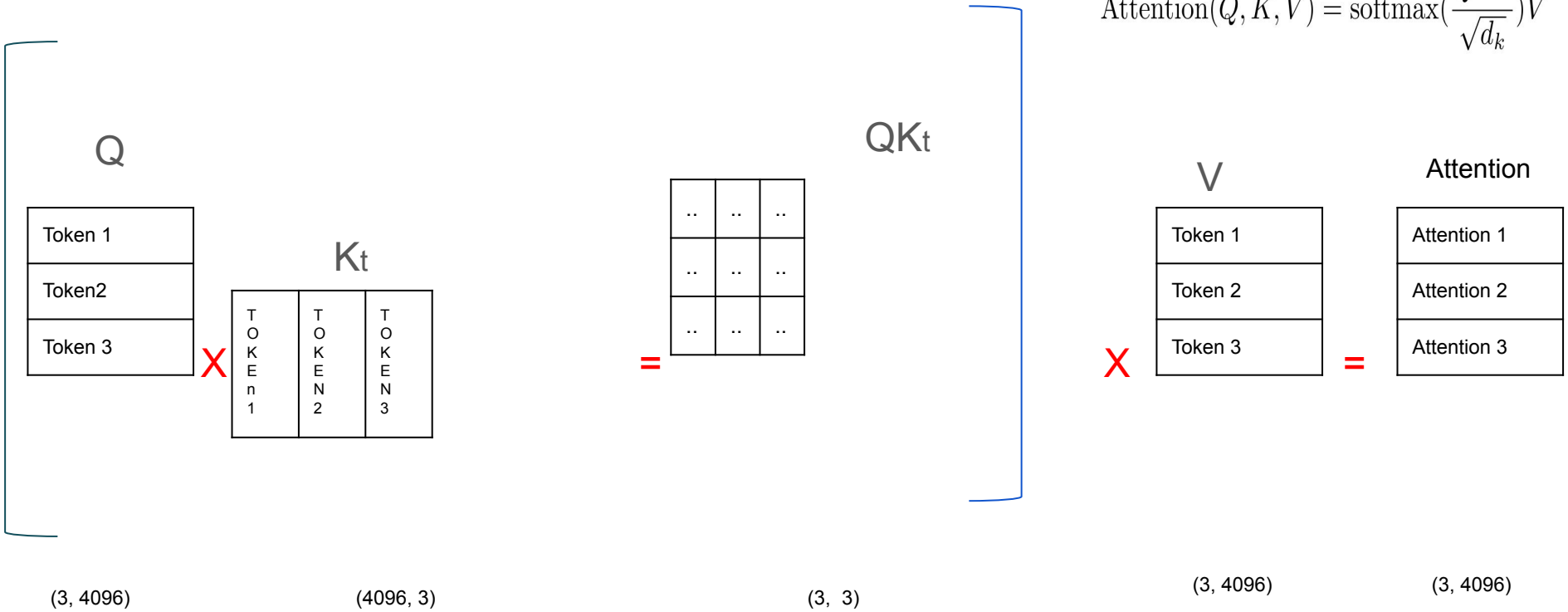


(2, 4096) (4096, 2) (2, 2) (2, 4096) (2, 4096)

Inference T =2

Self- Attention during Next Token prediction task

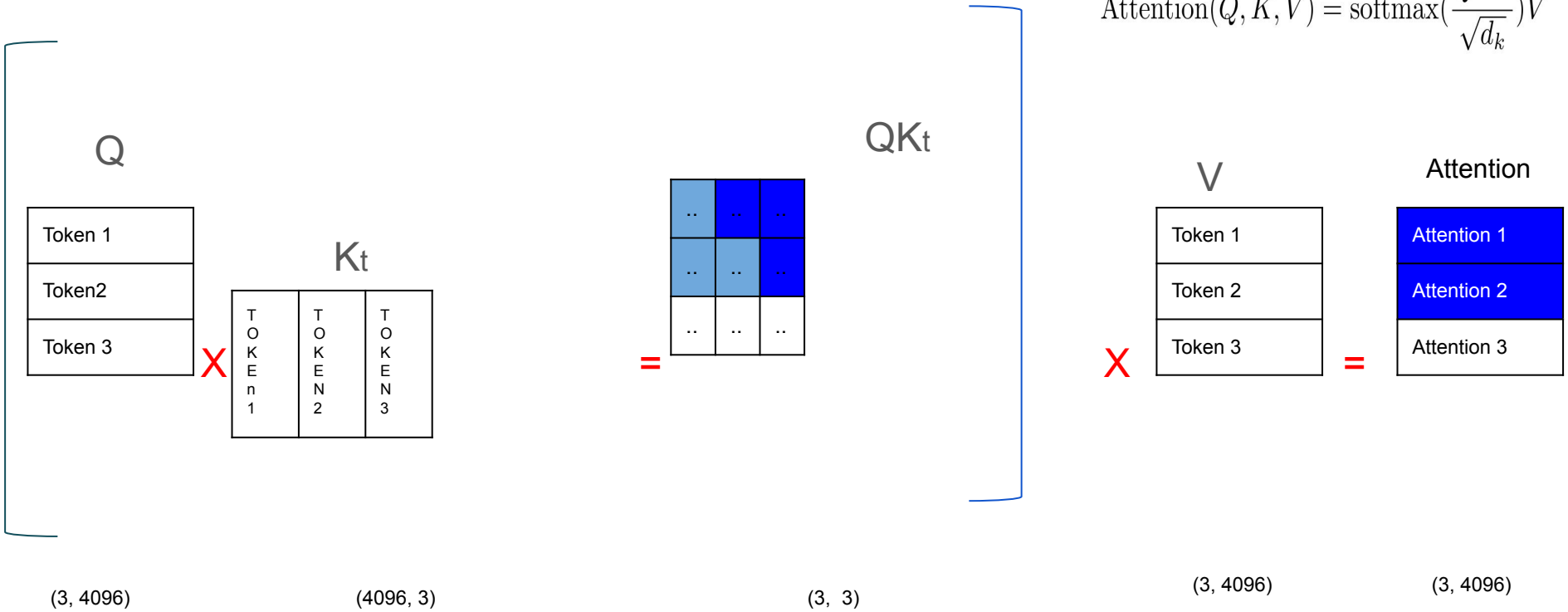
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



Inference T =3

Self- Attention during Next Token prediction task

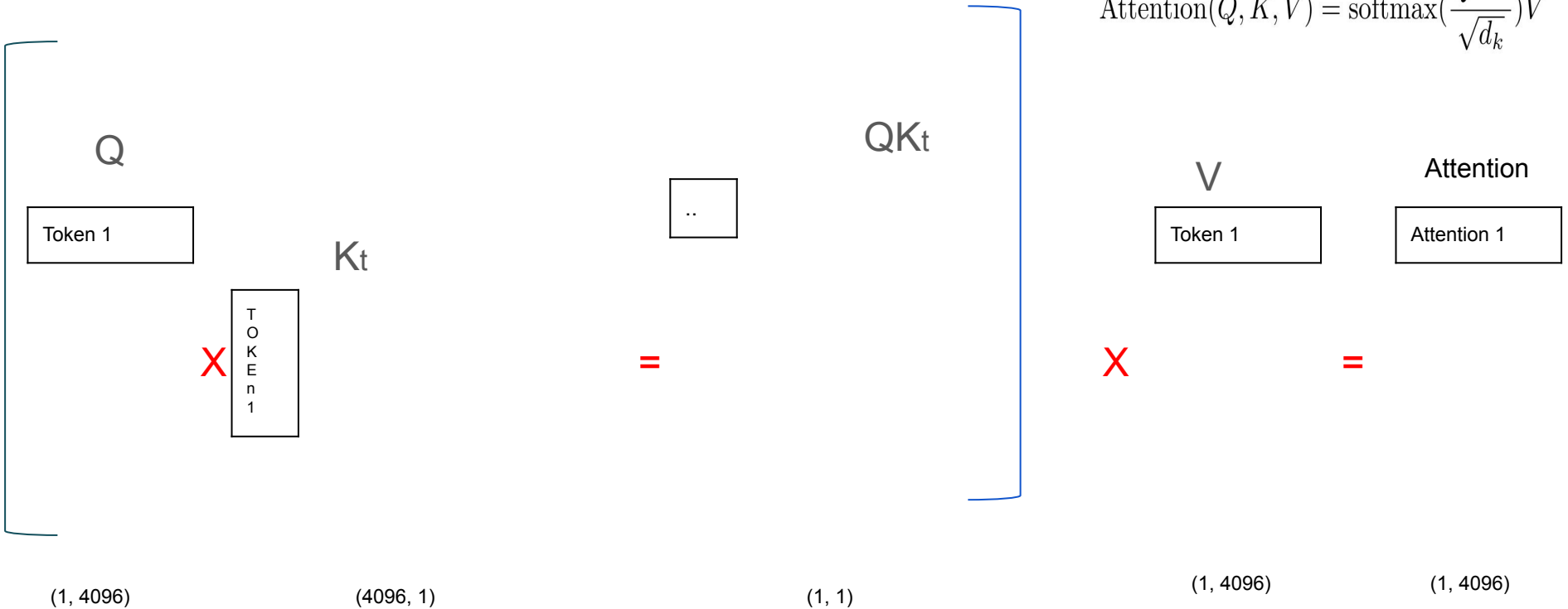
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



Inference T =3

Self- Attention with KV - Cache

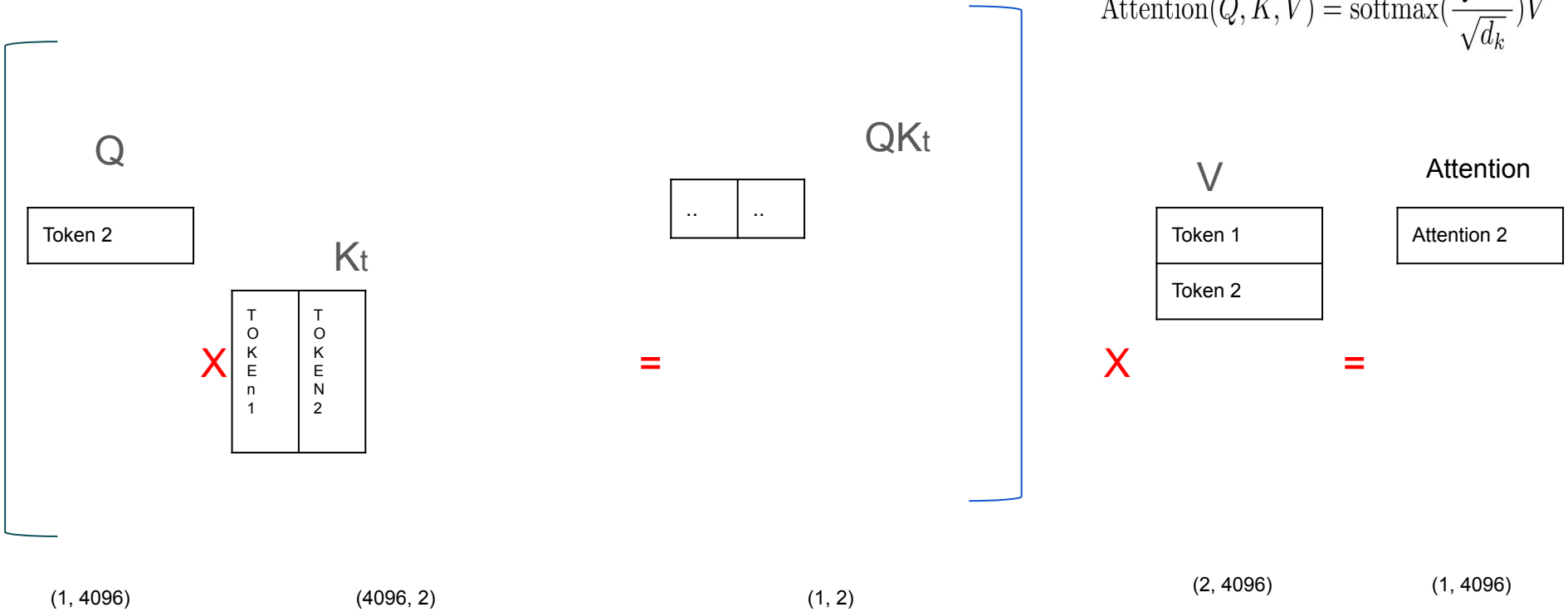
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



Inference T =1

Self- Attention with KV - Cache

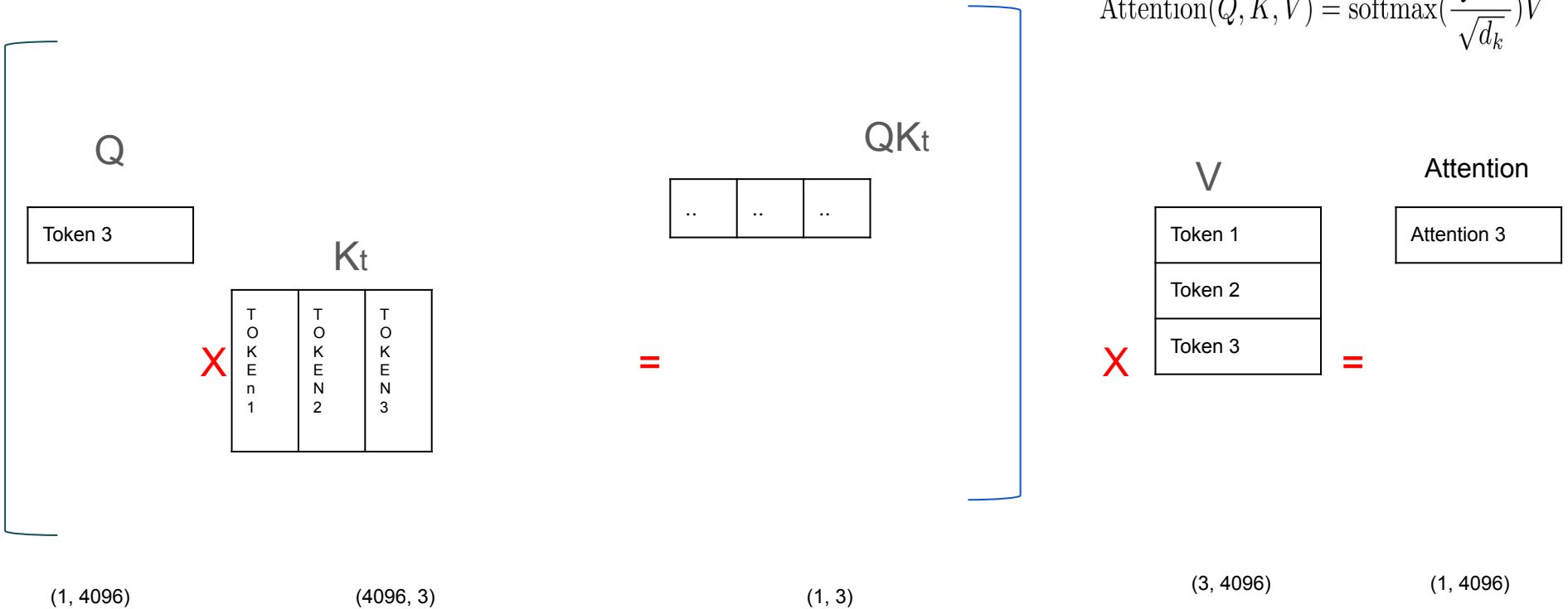
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



Inference T =2

Self- Attention with KV - Cache

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

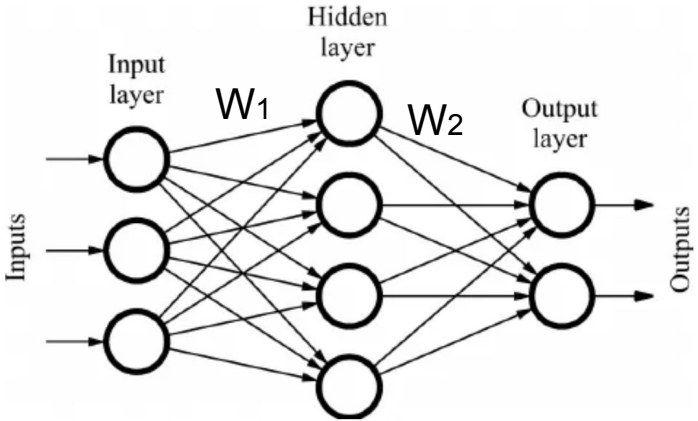


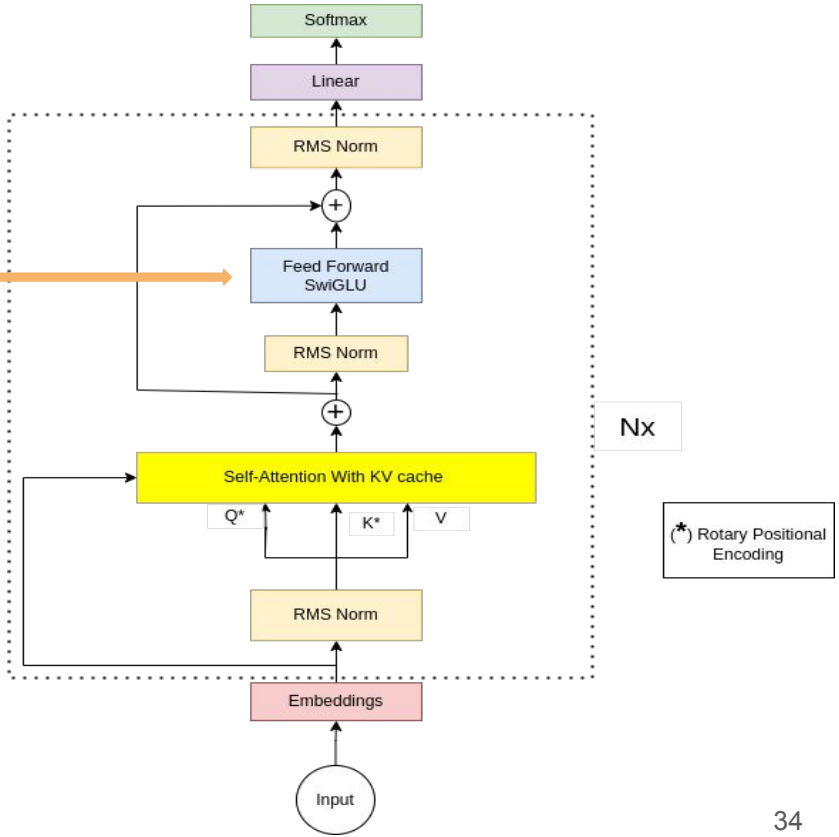
Inference T =3

Feed Forward Using SwiGLU

$$\mathrm{FFN}_{\mathrm{Swish}}(x, W_1, W_2) = \mathrm{Swish}_1(xW_1)W_2$$

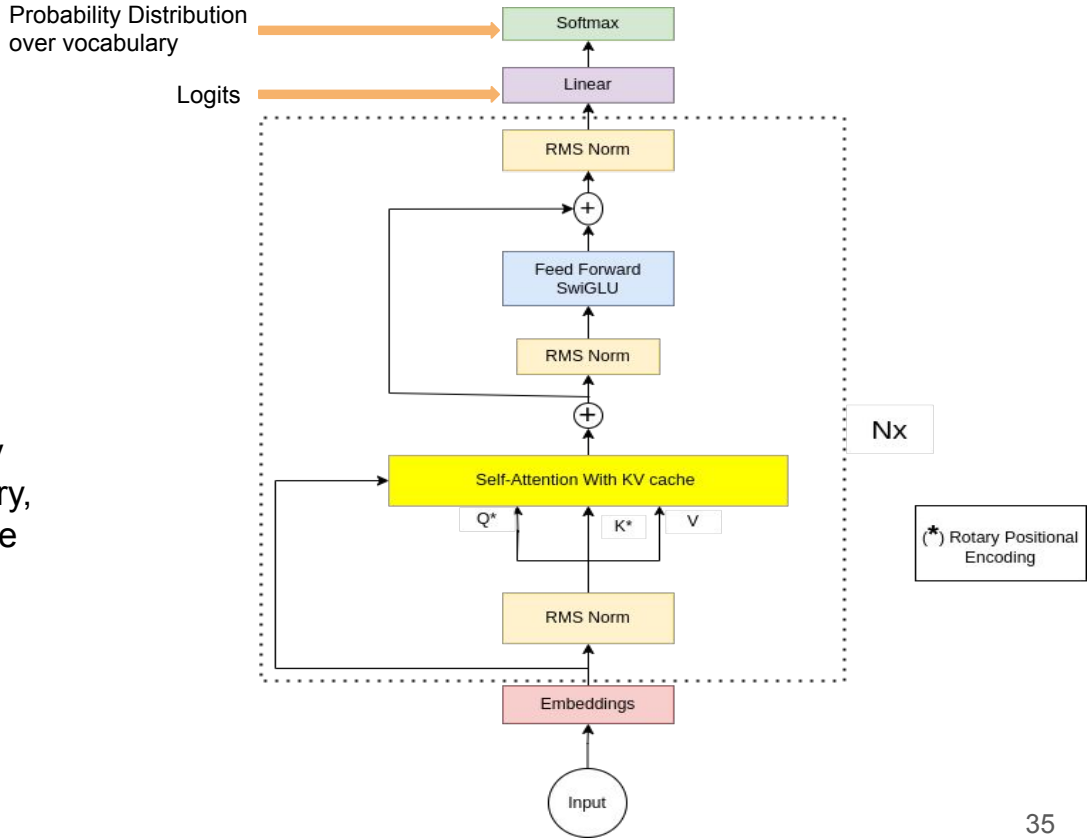
$$\mathrm{swish}(x) = x\,\mathrm{sigmoid}(\beta x) = \frac{x}{1 + e^{-\beta x}}$$





Logits and Softmax

- The output of the last linear layer in the Transformer model is called the **Logits** . The logits represent the unscaled “probabilities” , but they are not probabilities as they do not sum up to 1.
- THE **Softmax** scales the logits in such a way that they sum up to 1
- The output of the softmax is thus a probability distribution over all the words in the vocabulary, that is , each words in the vocabulary will have probability associated to it.



Llama 2 - Pretraining Functionality

- Trained using **AdamW Optimizer (β1=0.1, β2=0.95, eps=10^{-5})**
- **Supervised Fine Tuning** :pretrained model is adapted to a specific task using labeled dataset

Related Work

- Rapid LLM Advancement : July 23, 2024 **Llama 3.1 405B**

Thank You