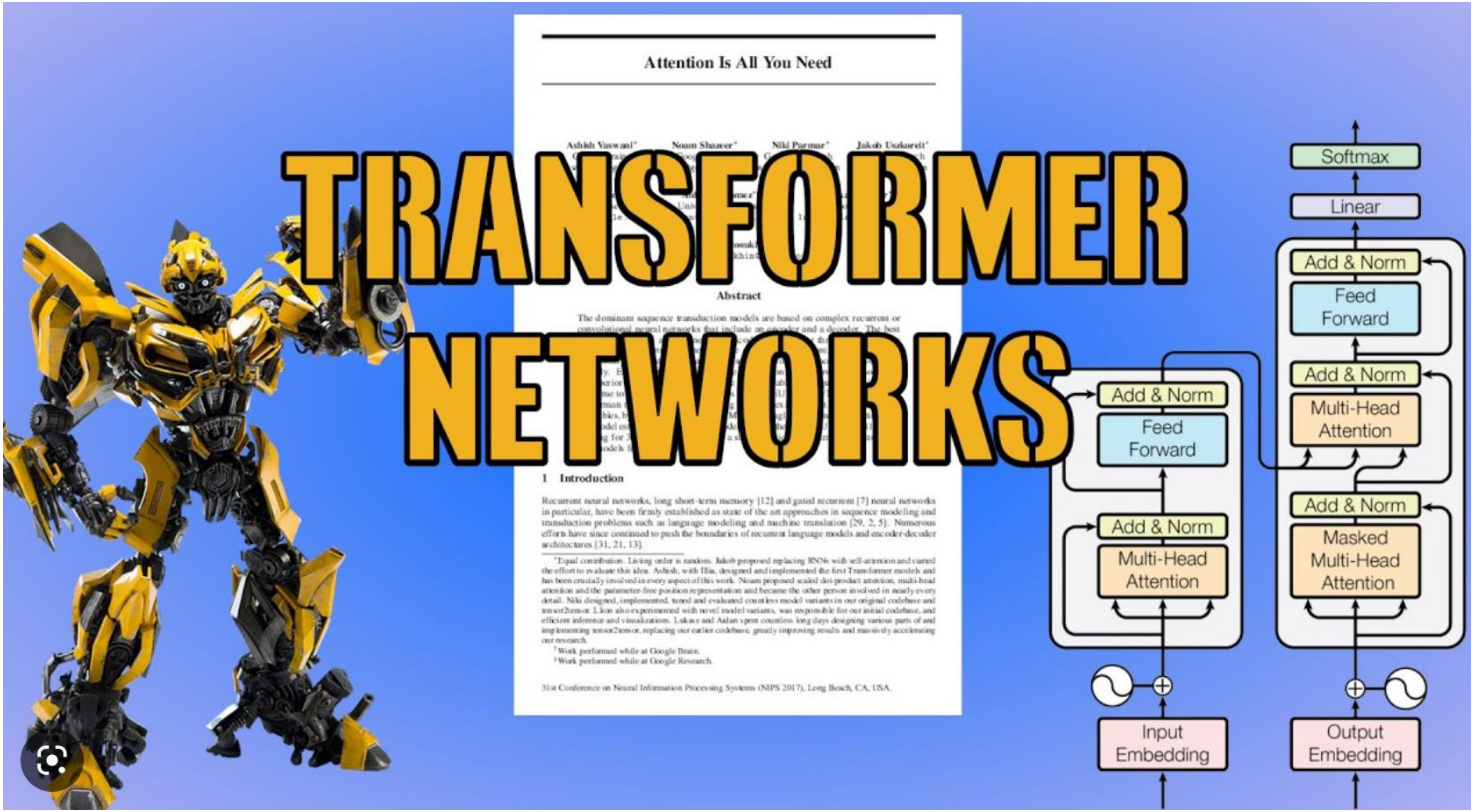
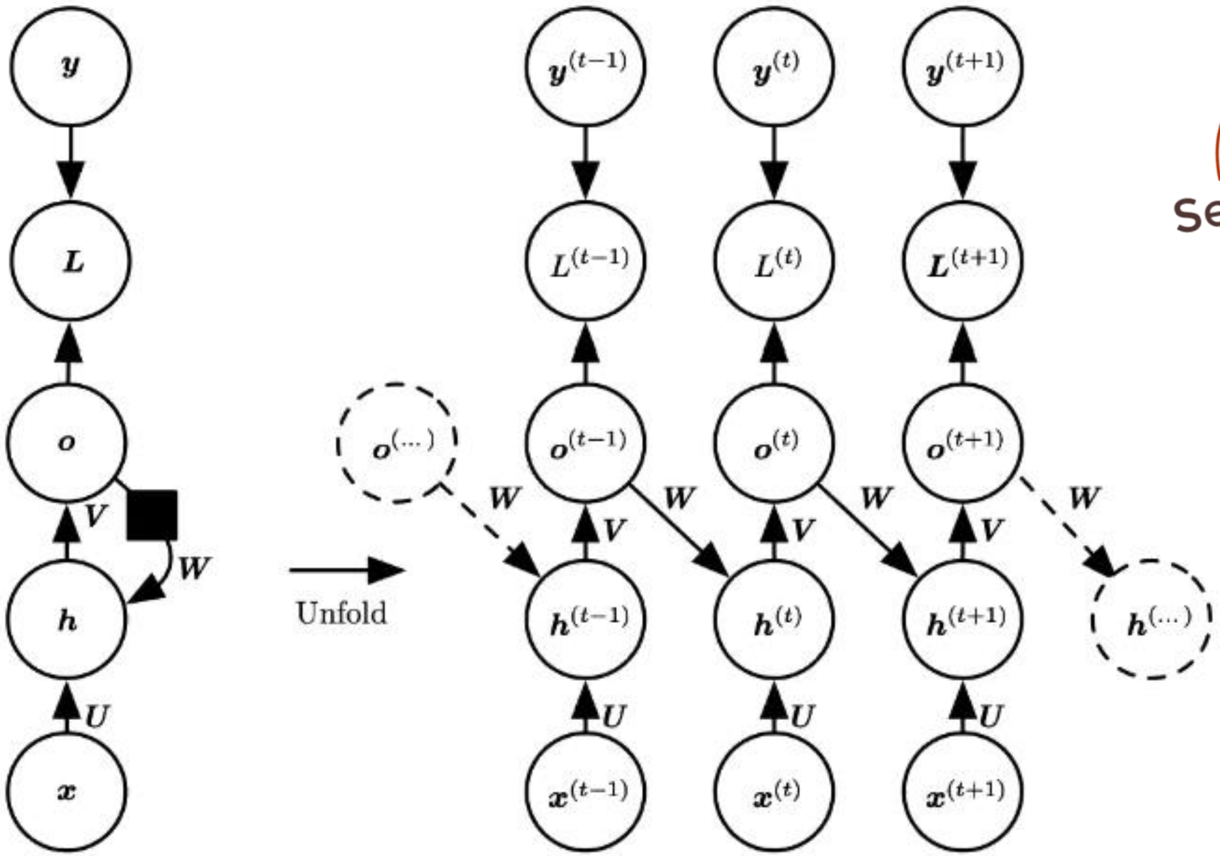
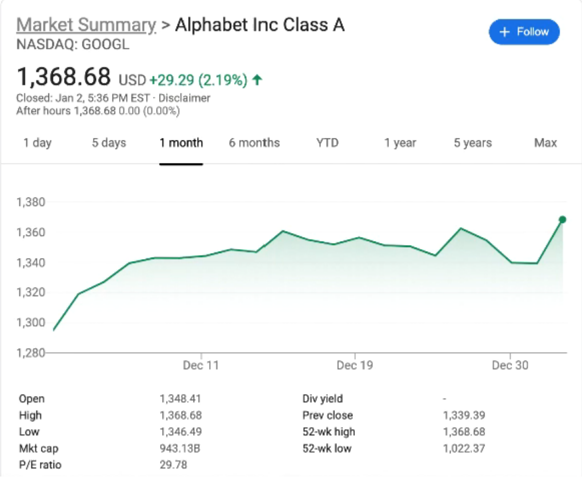

TRANSFORMER
NETWORKS
Attention Is All You Need
Ashish Vaswani*
Noam Shazeer*
Niki Parmar*
Jakob Uszkoreit*
Abstract
The dominant sequence transduction models are based on complex recurrent or
1 Introduction
Recurrent neural networks, long short-term memory [12] and gated recurrent [7] neural networks in particular, have been firmly established as state of the art approaches in sequence modeling and transduction problems such as language modeling and machine translation [29, 2, 5]. Numerous efforts have since continued to push the boundaries of recurrent language models and encoder-decoder architectures [31, 21, 13].
†Work performed while at Google Brain.
‡Work performed while at Google Research.
31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Input Embedding
Output Embedding

Recurrent Neural Networks



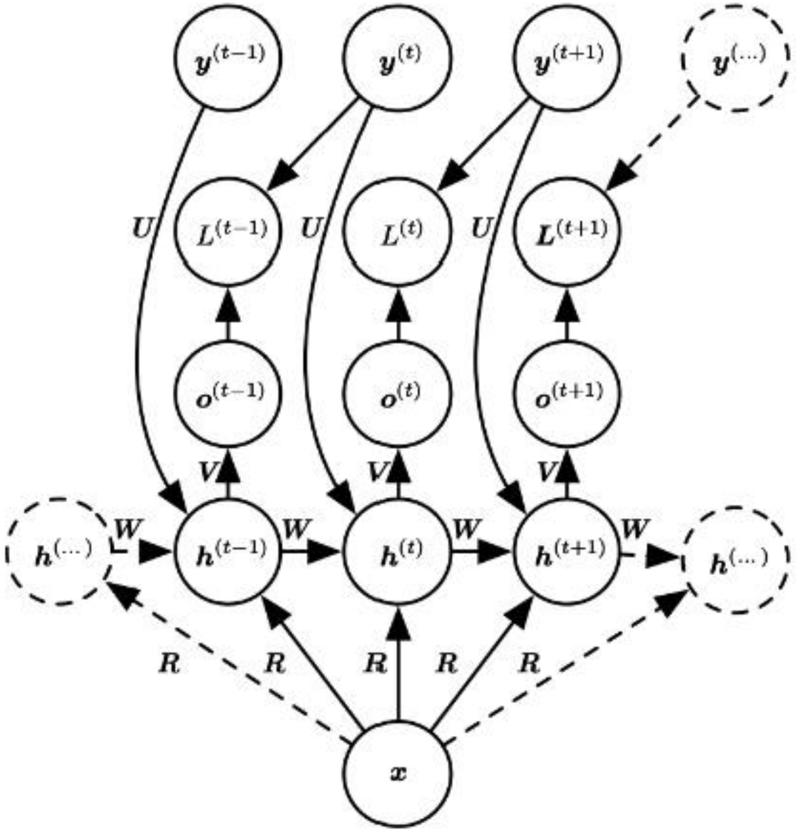
The unicorn is scotland's national animal

Sequences



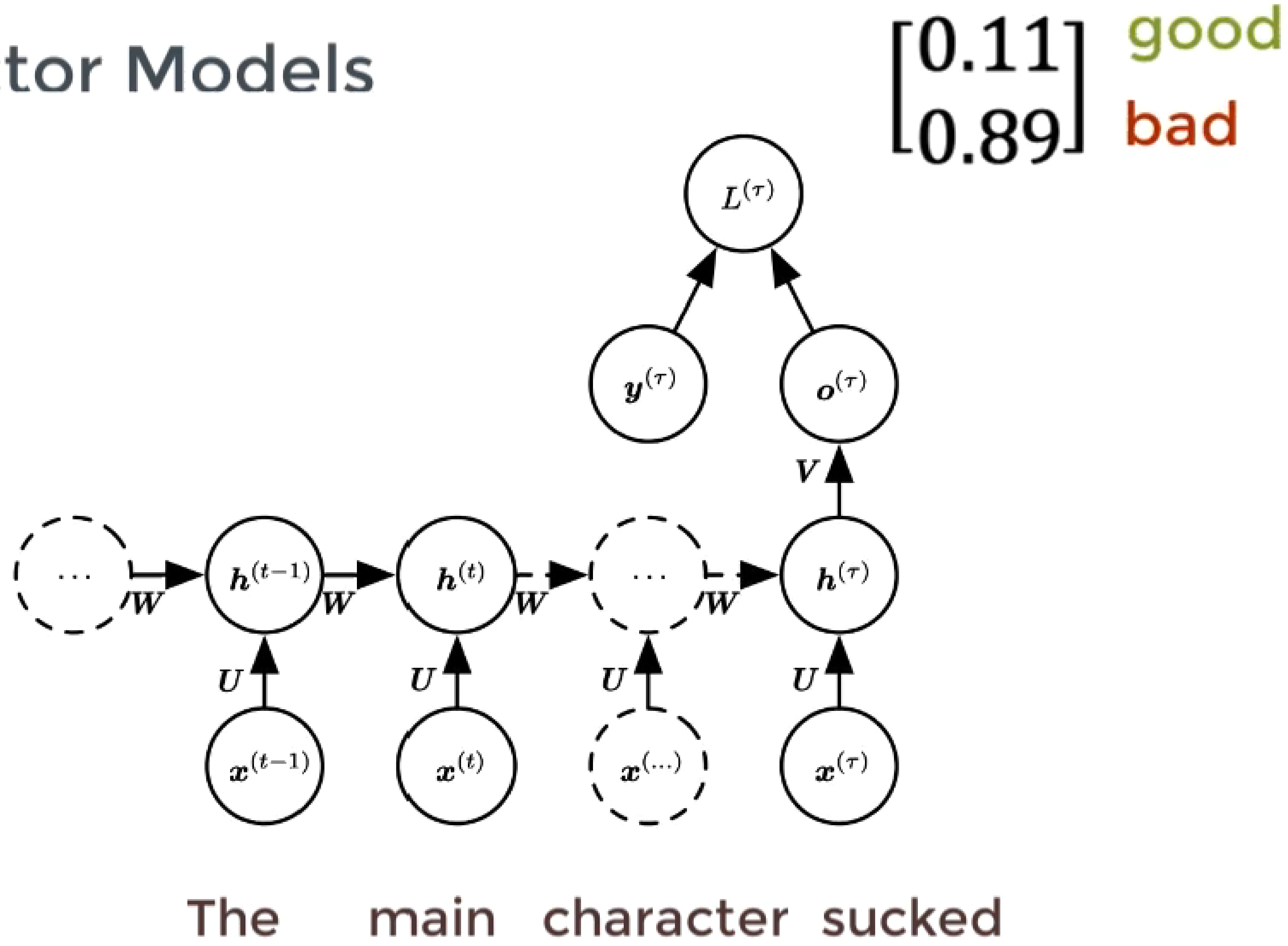
Recurrent Neural Networks

1. Vector-Sequence Models



Recurrent Neural Networks

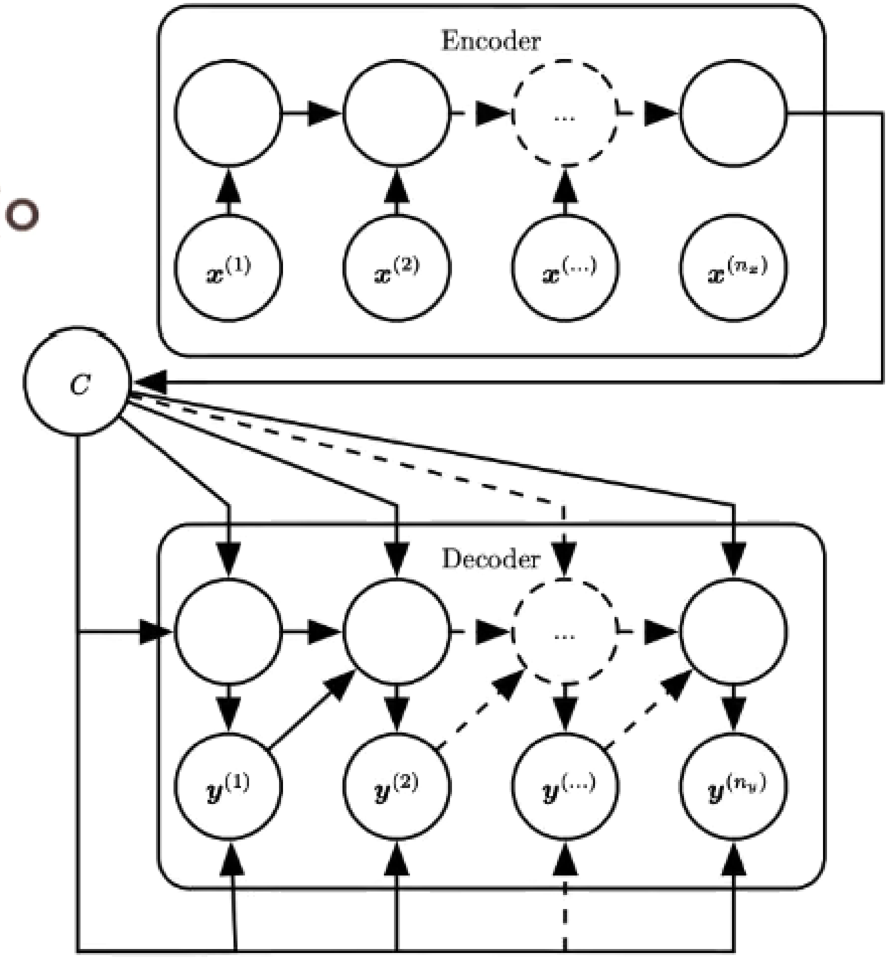
2. Sequence-Vector Models



Recurrent Neural Networks

3. Sequence-Sequence Models

El gran perro rojo



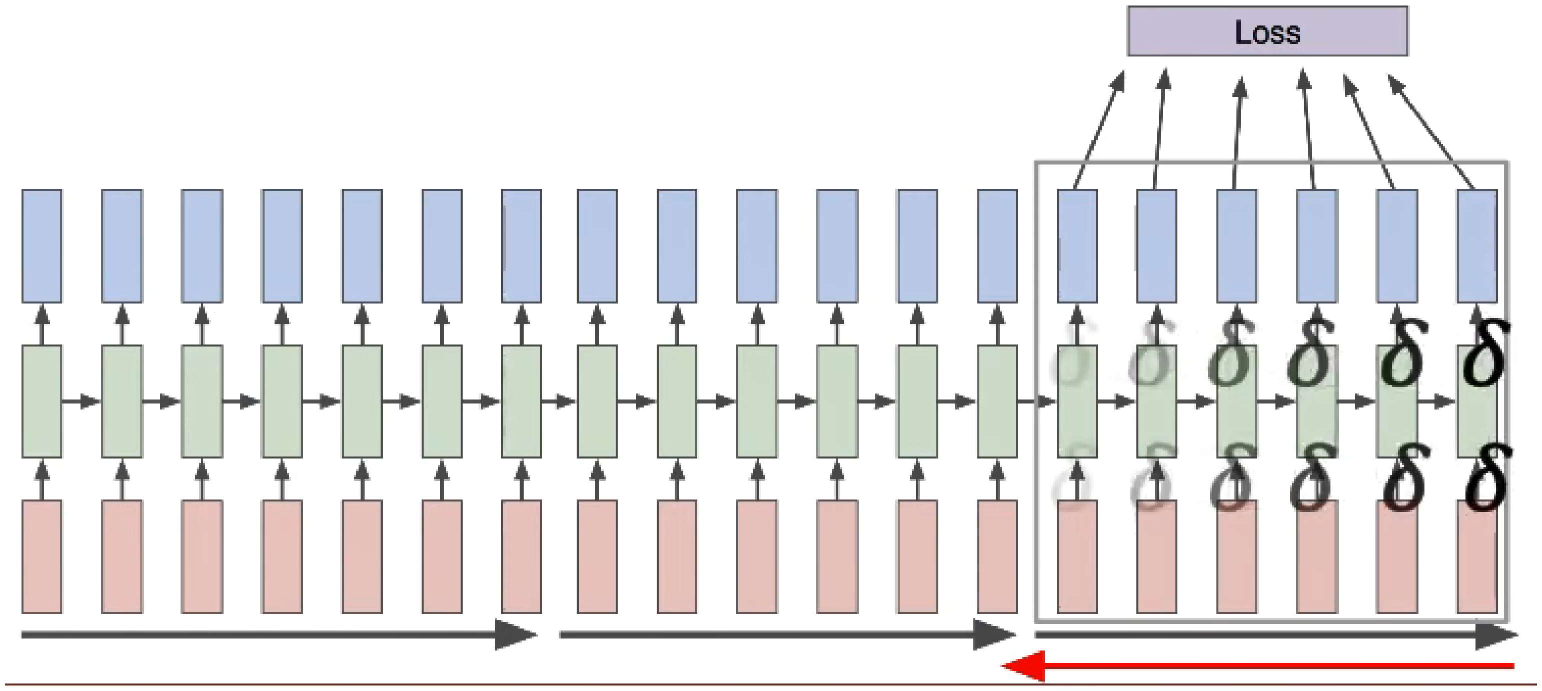
The big red dog

Recurrent Neural Networks

<u>Disadvantages</u>

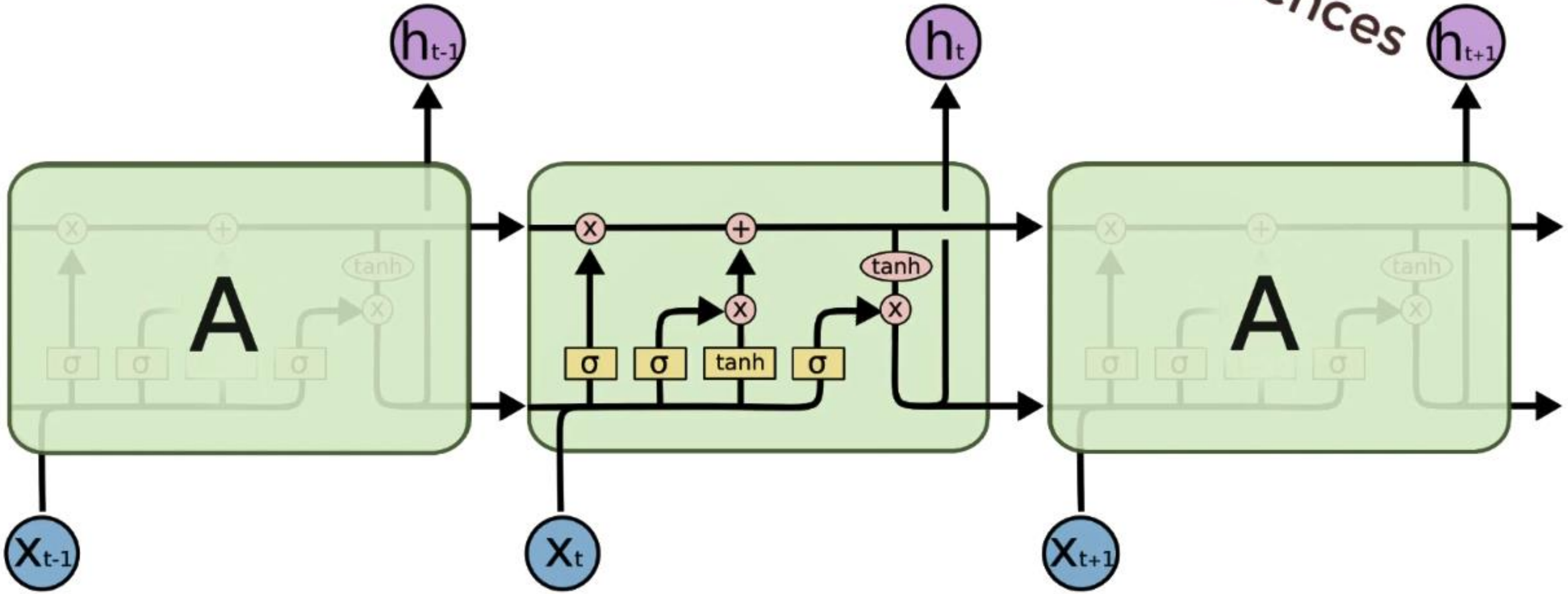
1. Slow to train.

2. Long sequences lead to vanishing/ exploding gradients



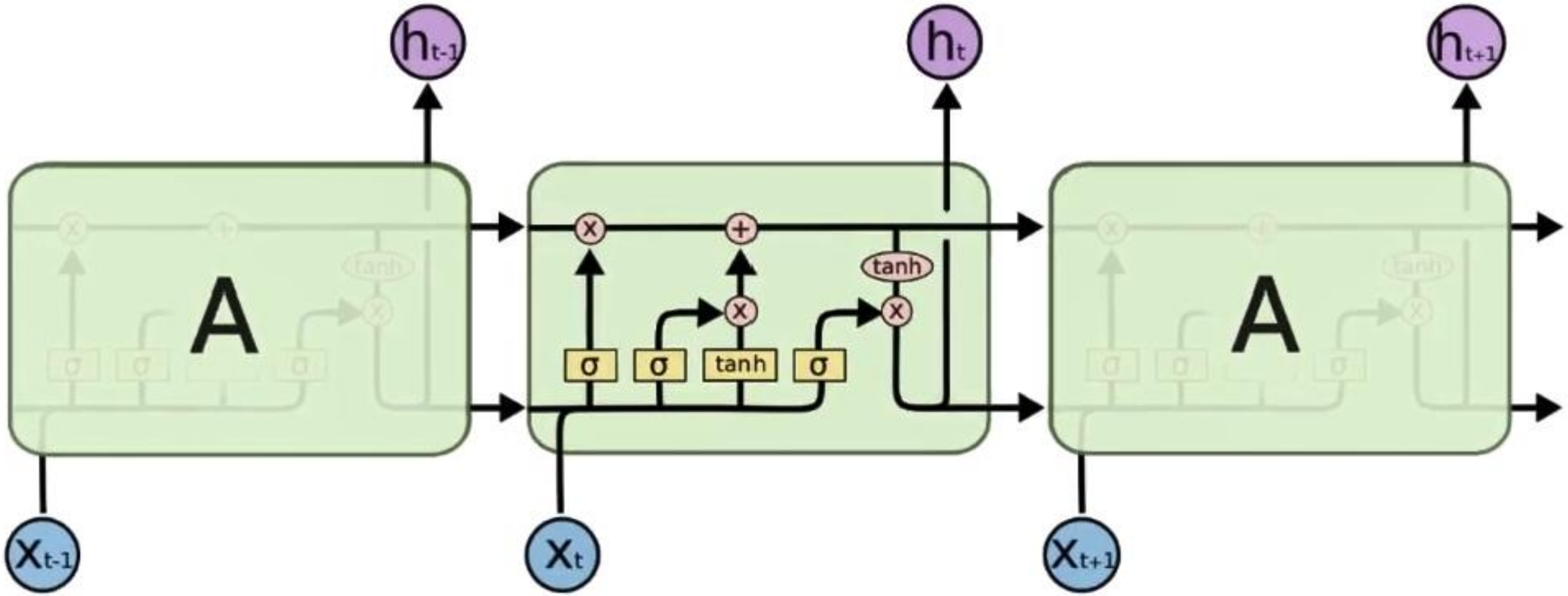
Truncated Backprop Through Time

LSTM Networks



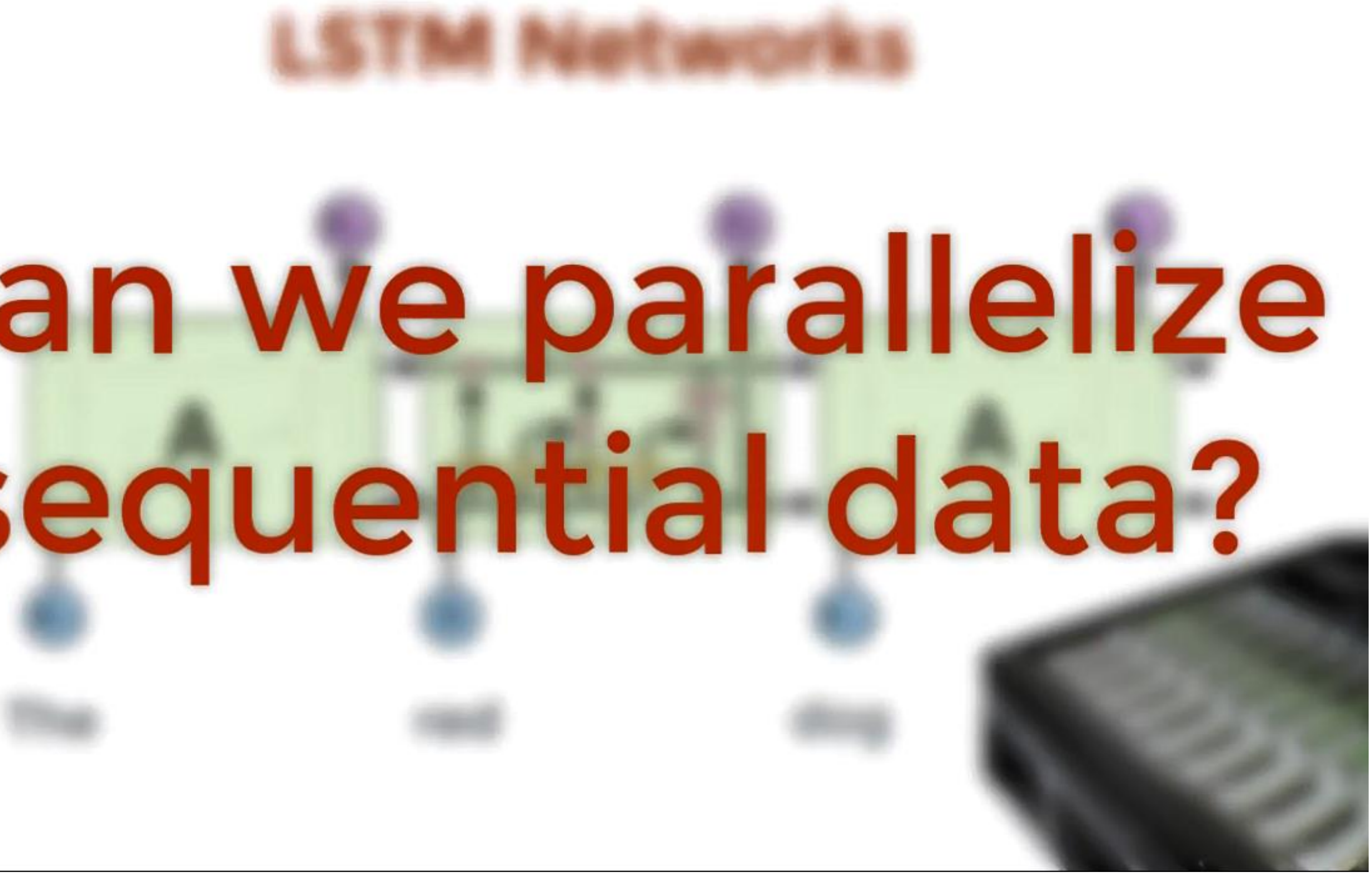
The repeating module in an LSTM contains four interacting layers.

LSTM Networks



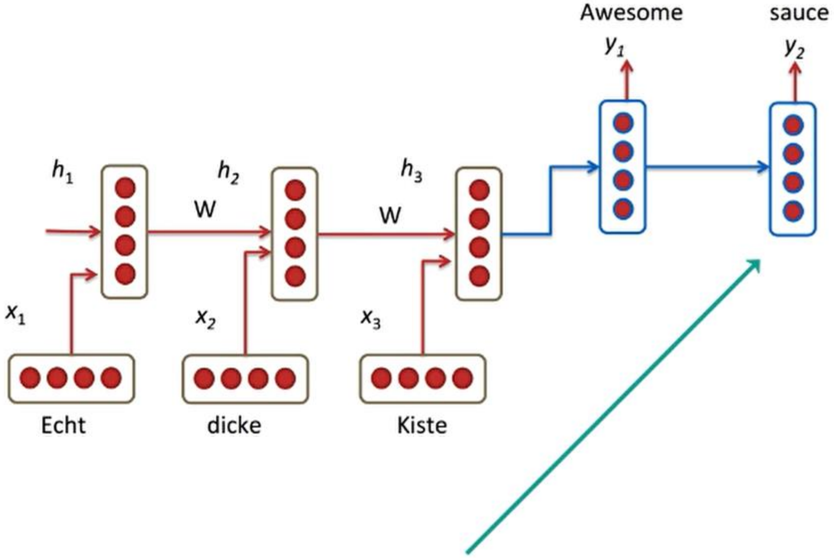
The red dog

Can we parallelize sequential data?



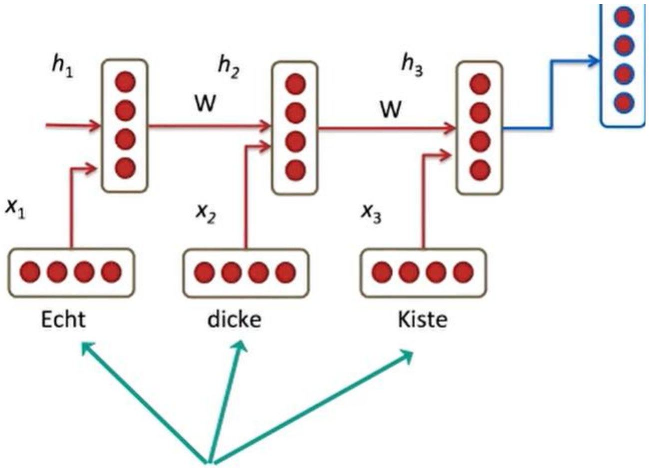
LSTM Networks

1. Slow



Timestep 5

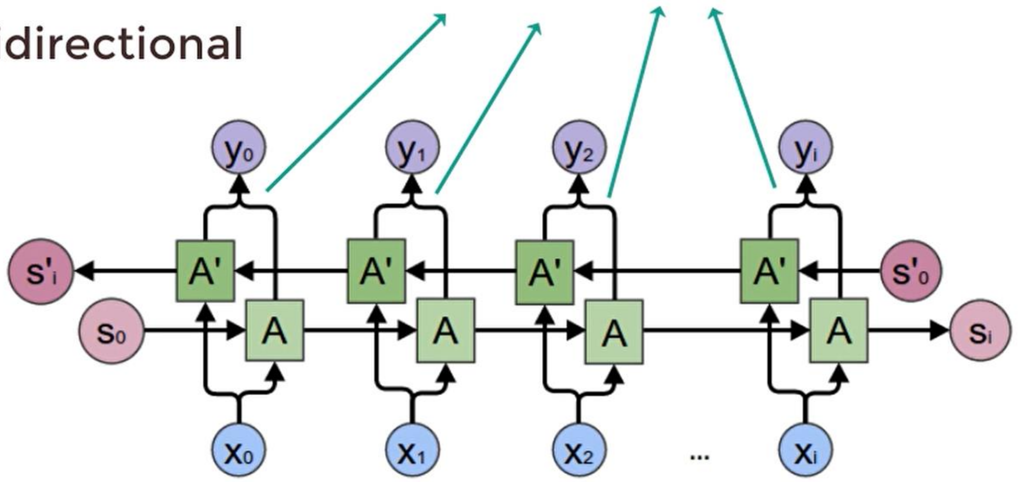
2. Not truly Bidirectional



Bidirectional

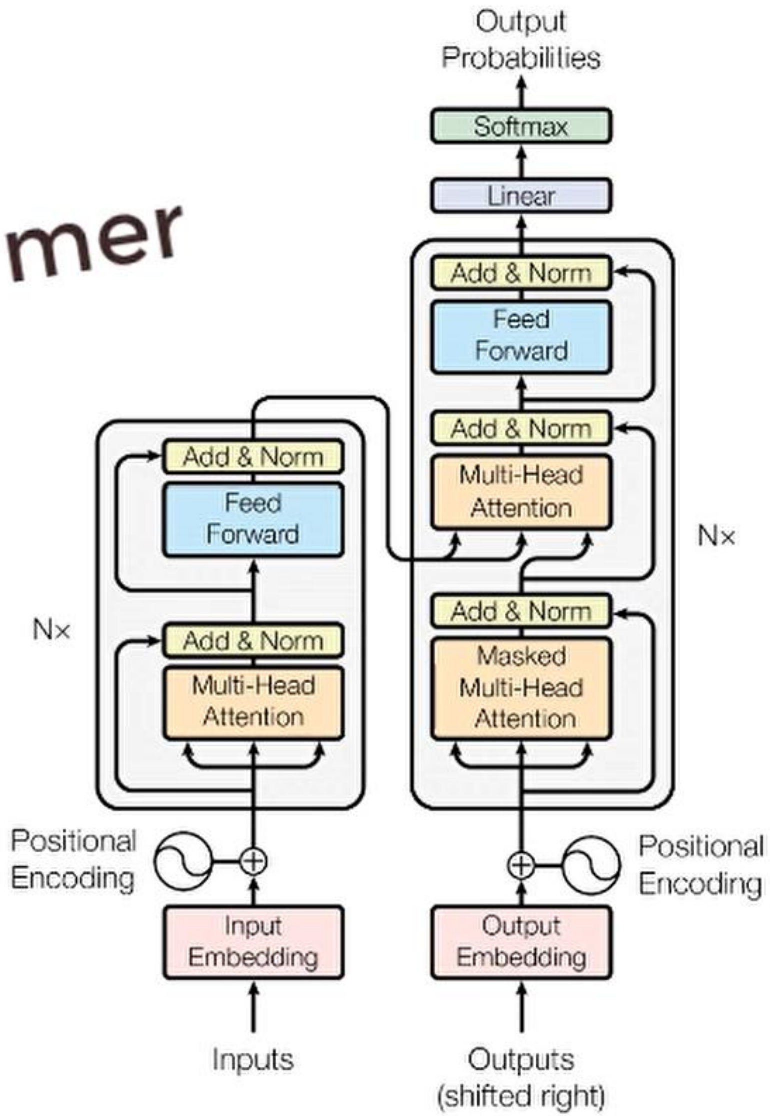
True meaning of source
words not entirely captured

Simple concatenation



LSTM Vs Transformer

Transformer



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

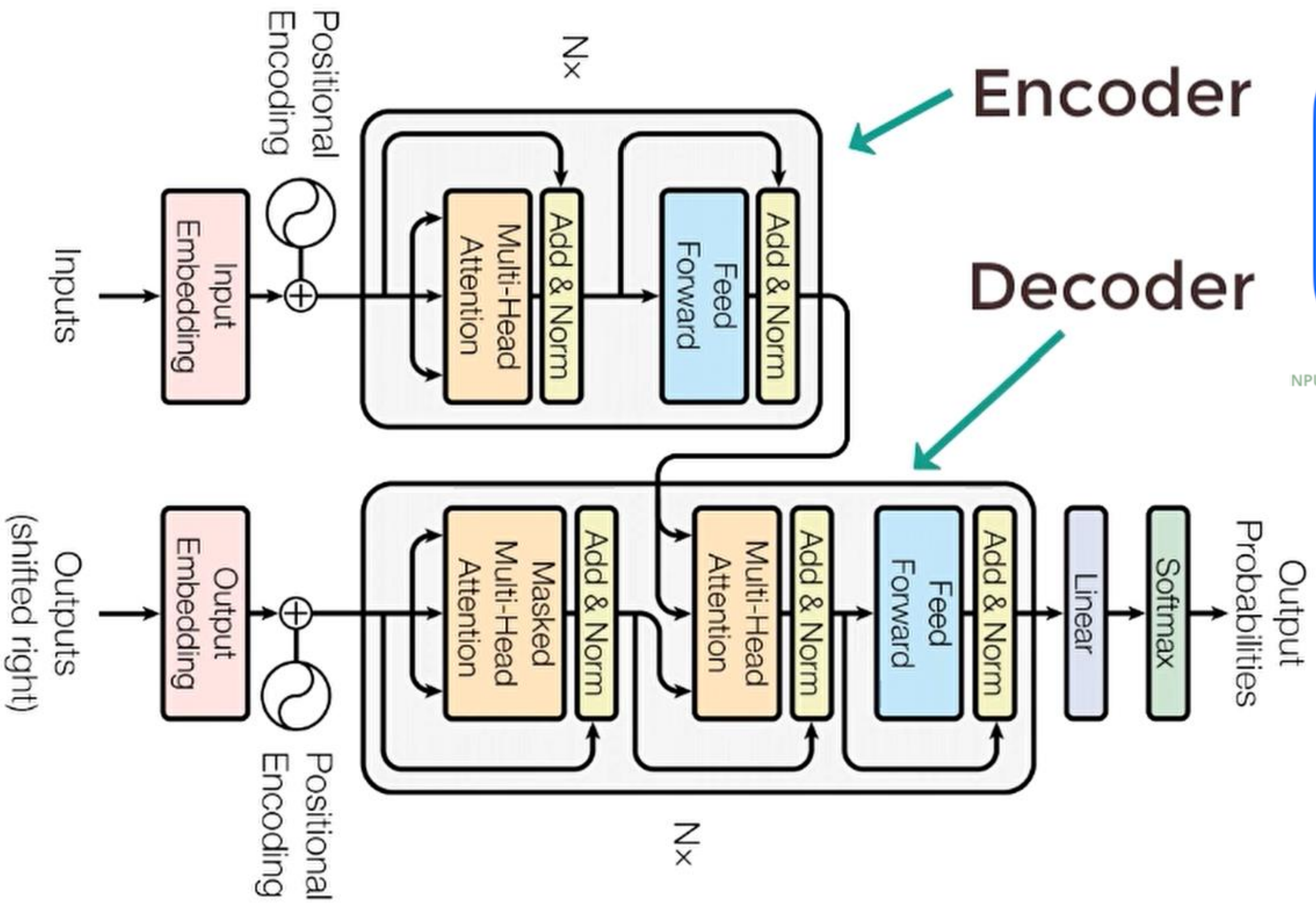
Łukasz Kaiser* Google Brain lukaszkaiser@google.com

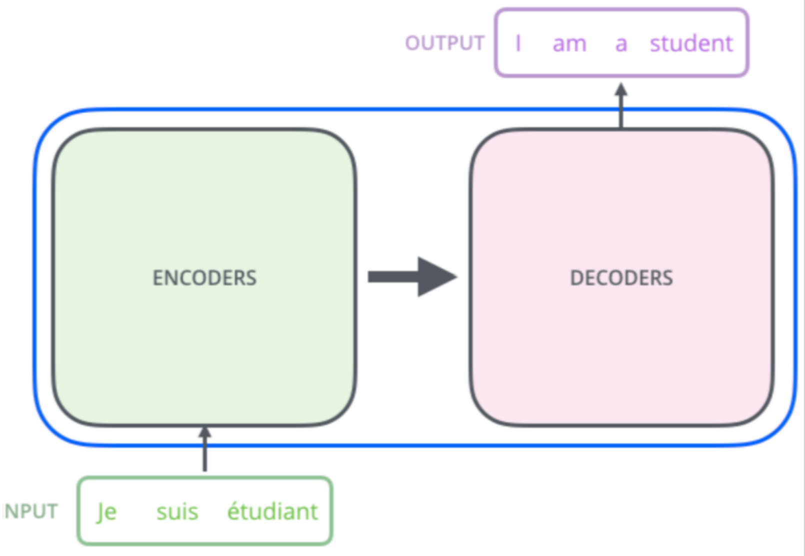
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

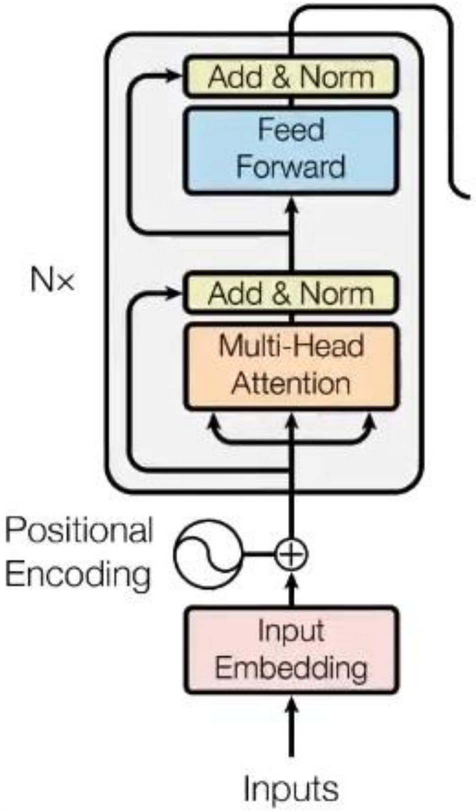
The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, 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.

Transformer Flow





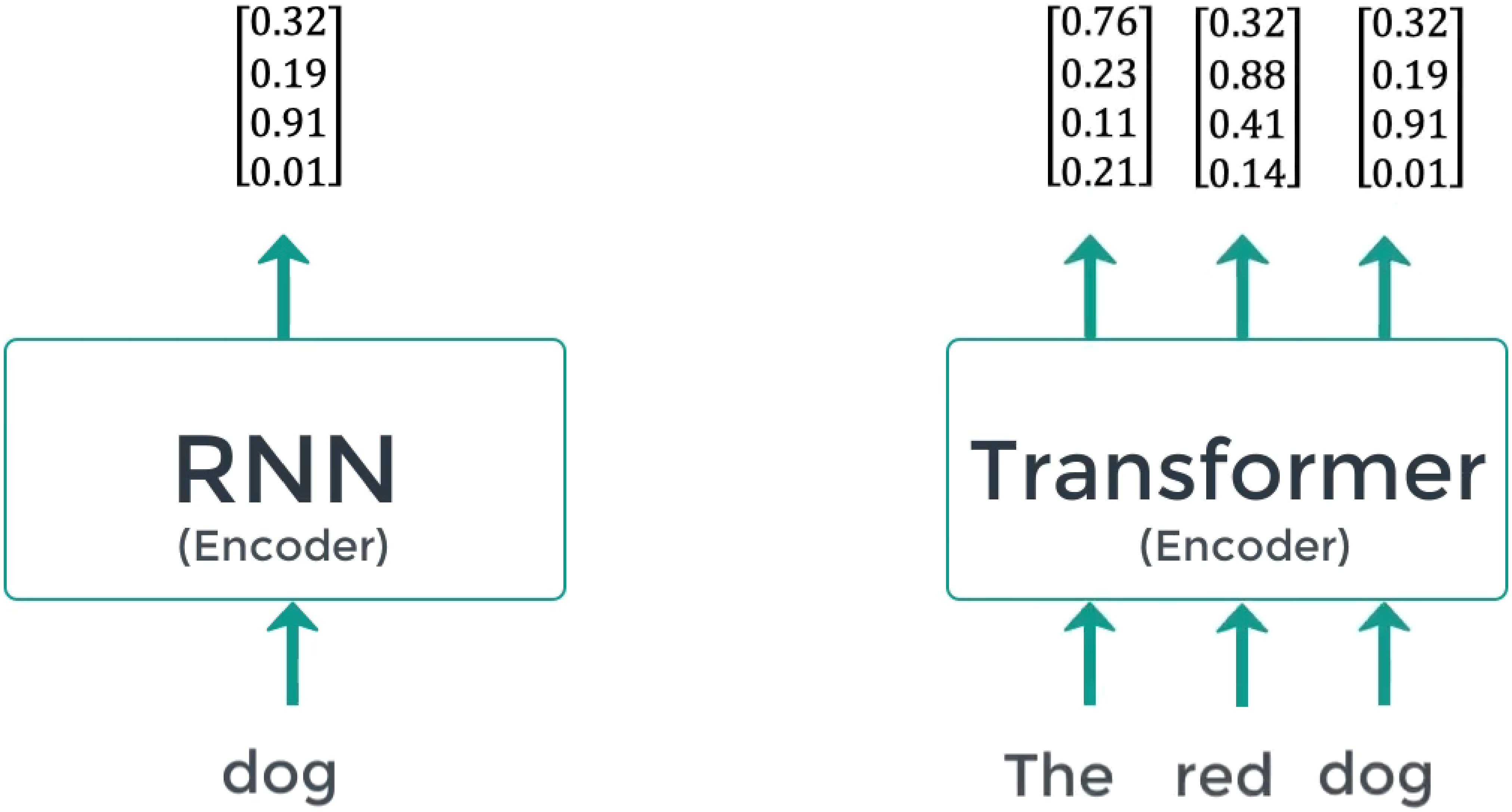
Transformers



$$\begin{bmatrix}0.76\\0.23\\0.11\\0.21\end{bmatrix} \quad \begin{bmatrix}0.32\\0.88\\0.41\\0.14\end{bmatrix} \quad \begin{bmatrix}0.32\\0.19\\0.91\\0.01\end{bmatrix}$$

The red dog

English-French Translation



Transformer Components

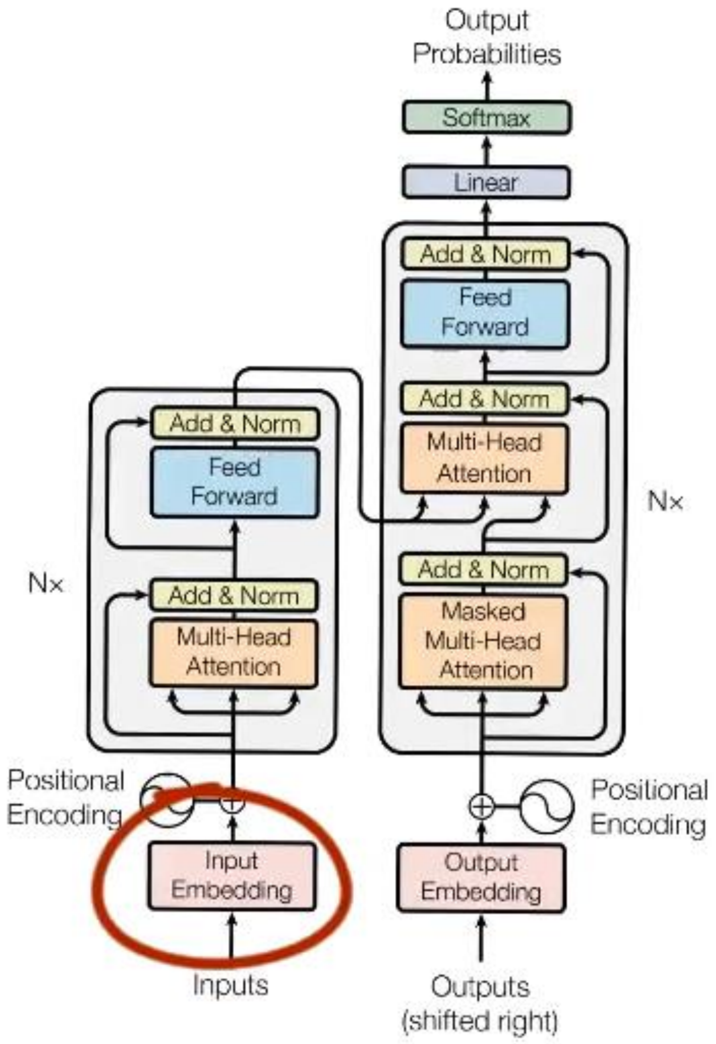
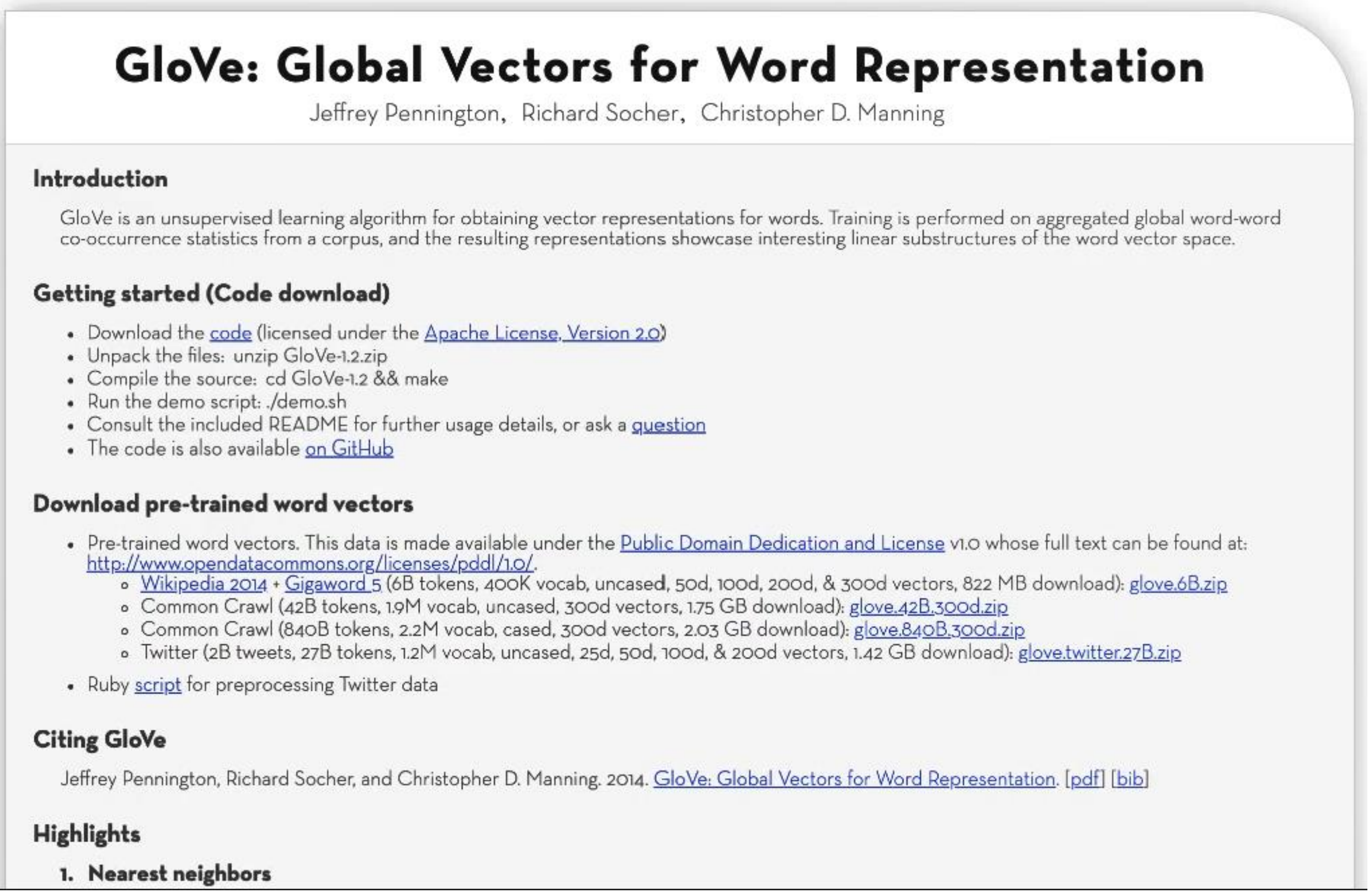


Figure 1: The Transformer - model architecture.

Transformer Components

Input Embedding

"Embedding Space"



GloVe: Global Vectors for Word Representation

Jeffrey Pennington, Richard Socher, Christopher D. Manning

Introduction

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

Getting started (Code download)

- Download the code (licensed under the Apache License, Version 2.0)
- Unpack the files: unzip GloVe-1.2.zip
- Compile the source: cd GloVe-1.2 && make
- Run the demo script: ./demo.sh
- Consult the included README for further usage details, or ask a question
- The code is also available on GitHub

Download pre-trained word vectors

- Pre-trained word vectors. This data is made available under the Public Domain Dedication and License v1.0 whose full text can be found at: http://www.opendatacommons.org/licenses/pddl/1.0/.
  - Wikipedia 2014 + Gigaword 5 (6B tokens, 400K vocab, uncased, 50d, 100d, 200d, & 300d vectors, 822 MB download): glove.6B.zip
  - Common Crawl (42B tokens, 1.9M vocab, uncased, 300d vectors, 1.75 GB download): glove.42B.300d.zip
  - Common Crawl (840B tokens, 2.2M vocab, cased, 300d vectors, 2.03 GB download): glove.840B.300d.zip
  - Twitter (2B tweets, 27B tokens, 1.2M vocab, uncased, 25d, 50d, 100d, & 200d vectors, 1.42 GB download): glove.twitter.27B.zip
- Ruby script for preprocessing Twitter data

Citing GloVe

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. [pdf] [bib]

Highlights

1. **Nearest neighbors**

Transformer Components

Input Embedding

dog →

→ $\begin{bmatrix} 0.37 \\ 0.99 \\ 0.01 \\ 0.08 \end{bmatrix}$

AJ's **dog** is a cutie

AJ looks like a **dog**

Transformer Components

<u>Positional Encoder</u> :vector that gives context based on position of word in sentence

AJ's dog is a cutie $\longrightarrow$ Position 2

AJ looks like a dog $\longrightarrow$ Position 5

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

Transformer Components

<u>Positional Encoder</u> :vector that gives context based on position of word in sentence

$$\begin{bmatrix} 0.37 \\ 0.99 \\ 0.01 \\ 0.08 \end{bmatrix} + \boxed{\text{Positional Encoding}} \rightarrow \begin{bmatrix} 0.42 \\ 0.84 \\ 0.12 \\ 0.81 \end{bmatrix}$$

Embedding of "Dog"

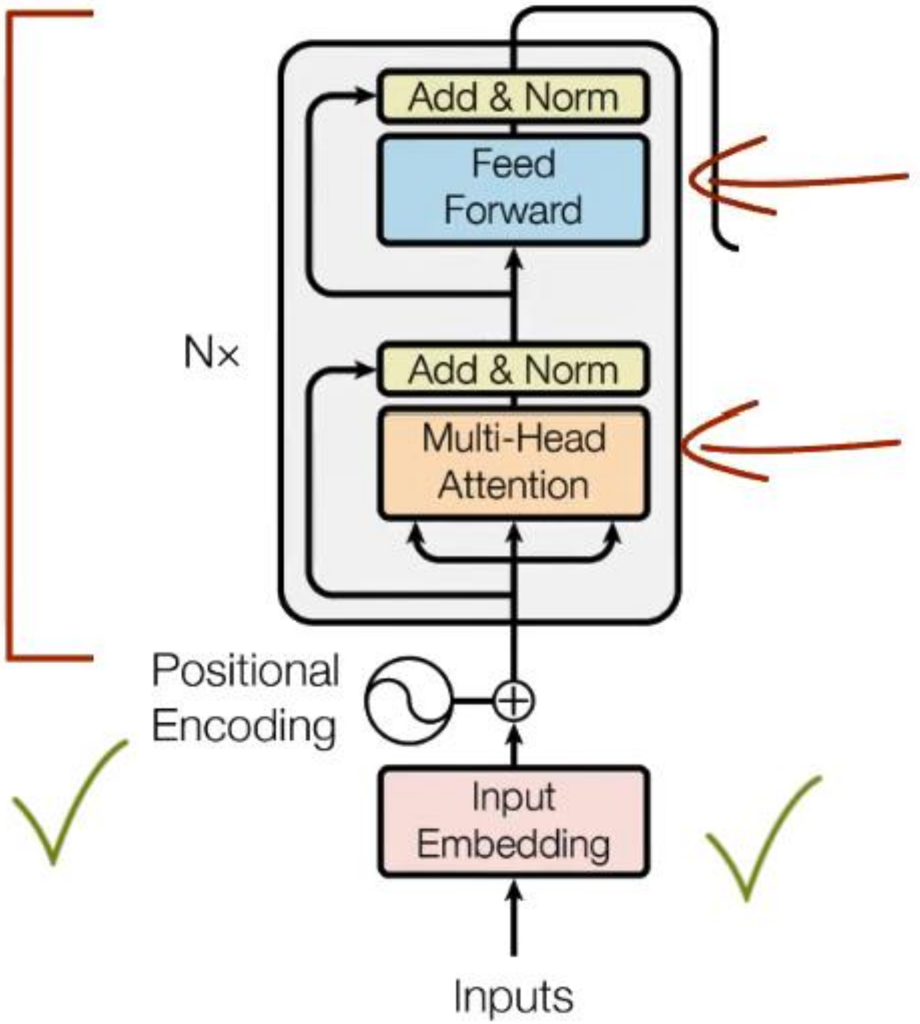
Vector Encoding of position in sentence

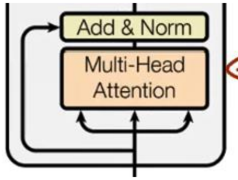
Embedding of Dog (with context info)

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

Transformer Components

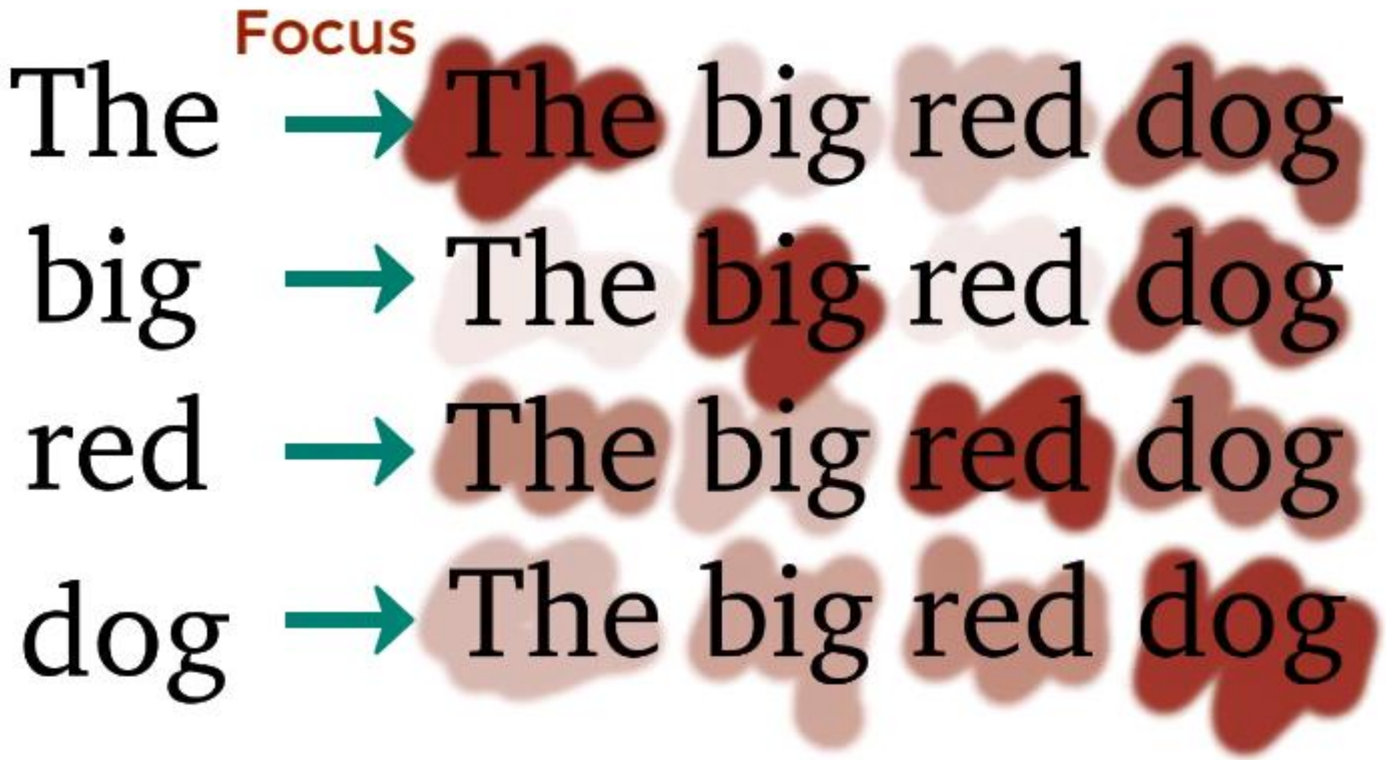
Encoder Block





Transformer Components

Attention : What part of the input should we focus?



Attention Vectors

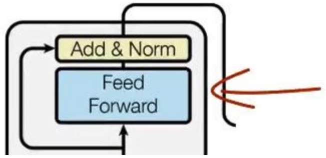
$[0.71 \quad 0.04 \quad 0.07 \quad 0.18]^T$

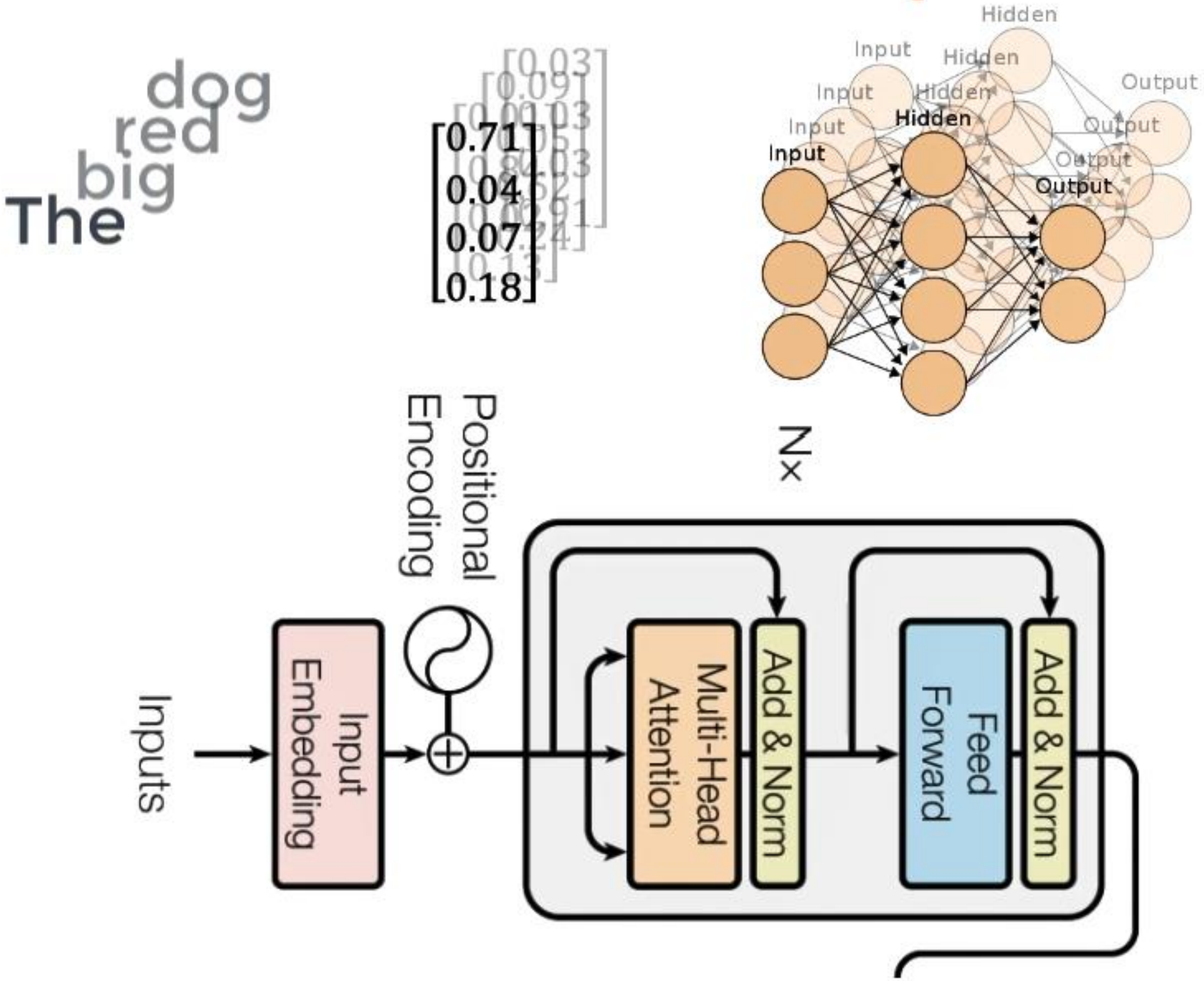
$[0.01 \quad 0.84 \quad 0.02 \quad 0.13]^T$

$[0.09 \quad 0.05 \quad 0.62 \quad 0.24]^T$

$[0.03 \quad 0.03 \quad 0.03 \quad 0.91]^T$

Transformer Components

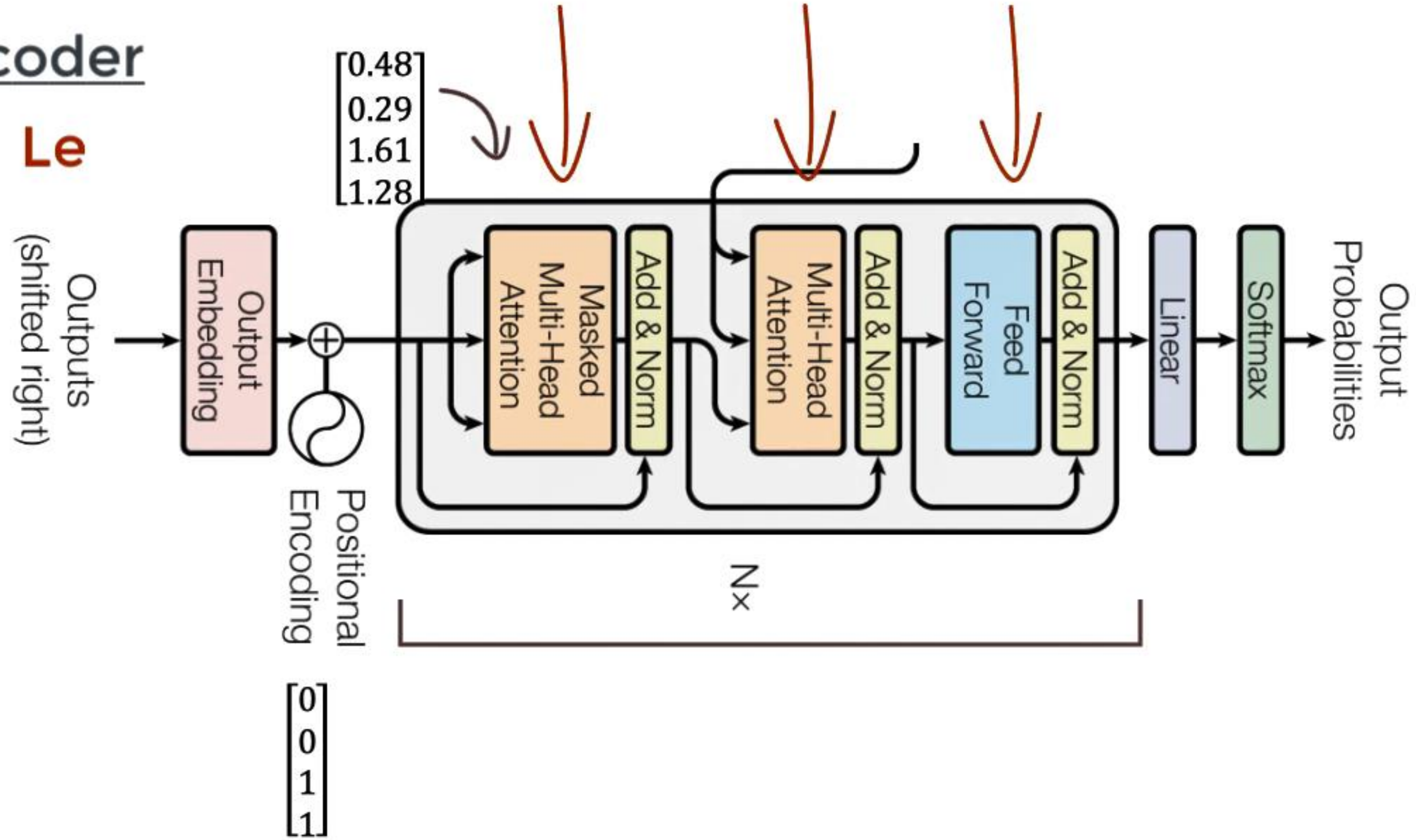




Transformer Components

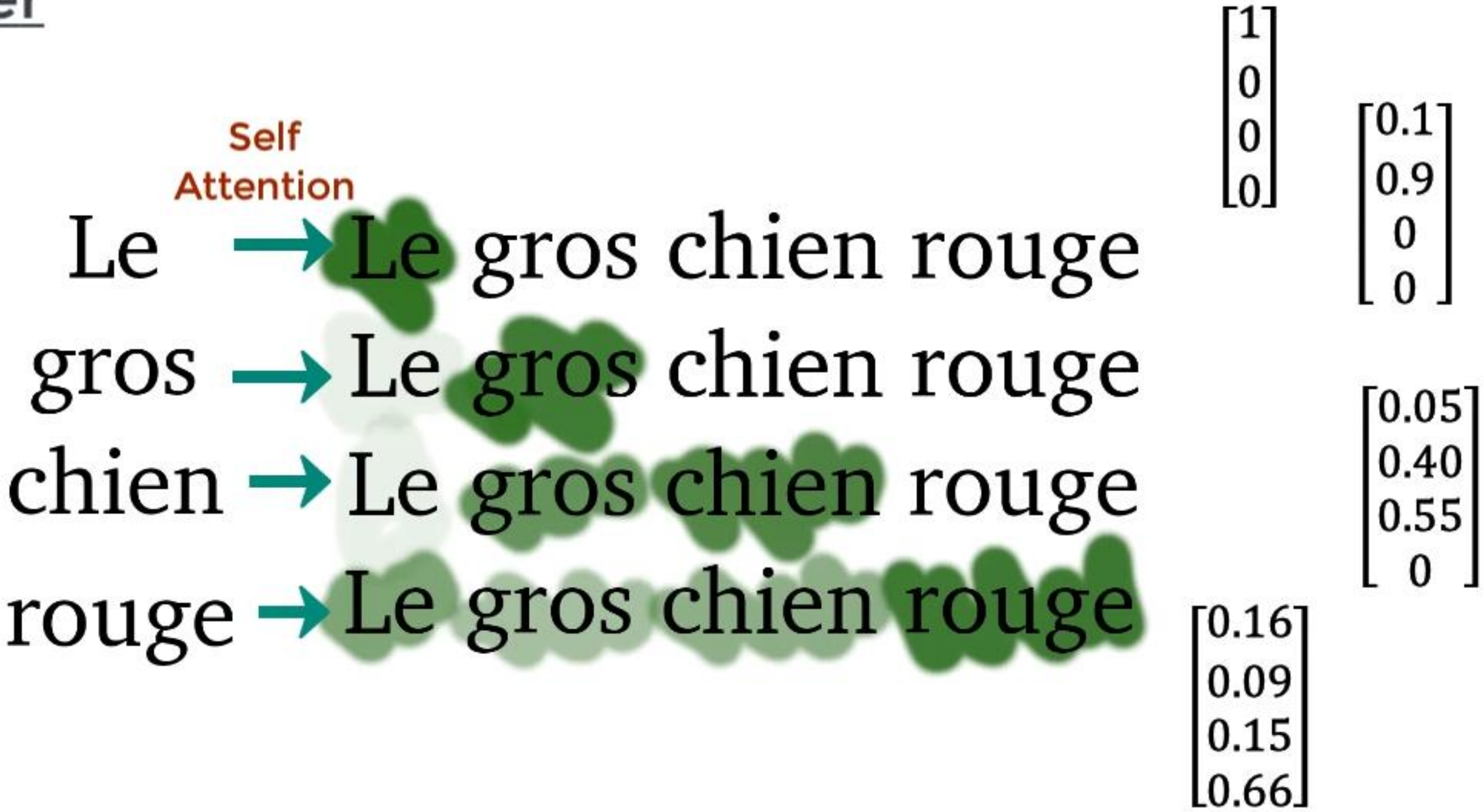
Decoder

Le



Transformer Components

Decoder



Transformer Components

Decoder

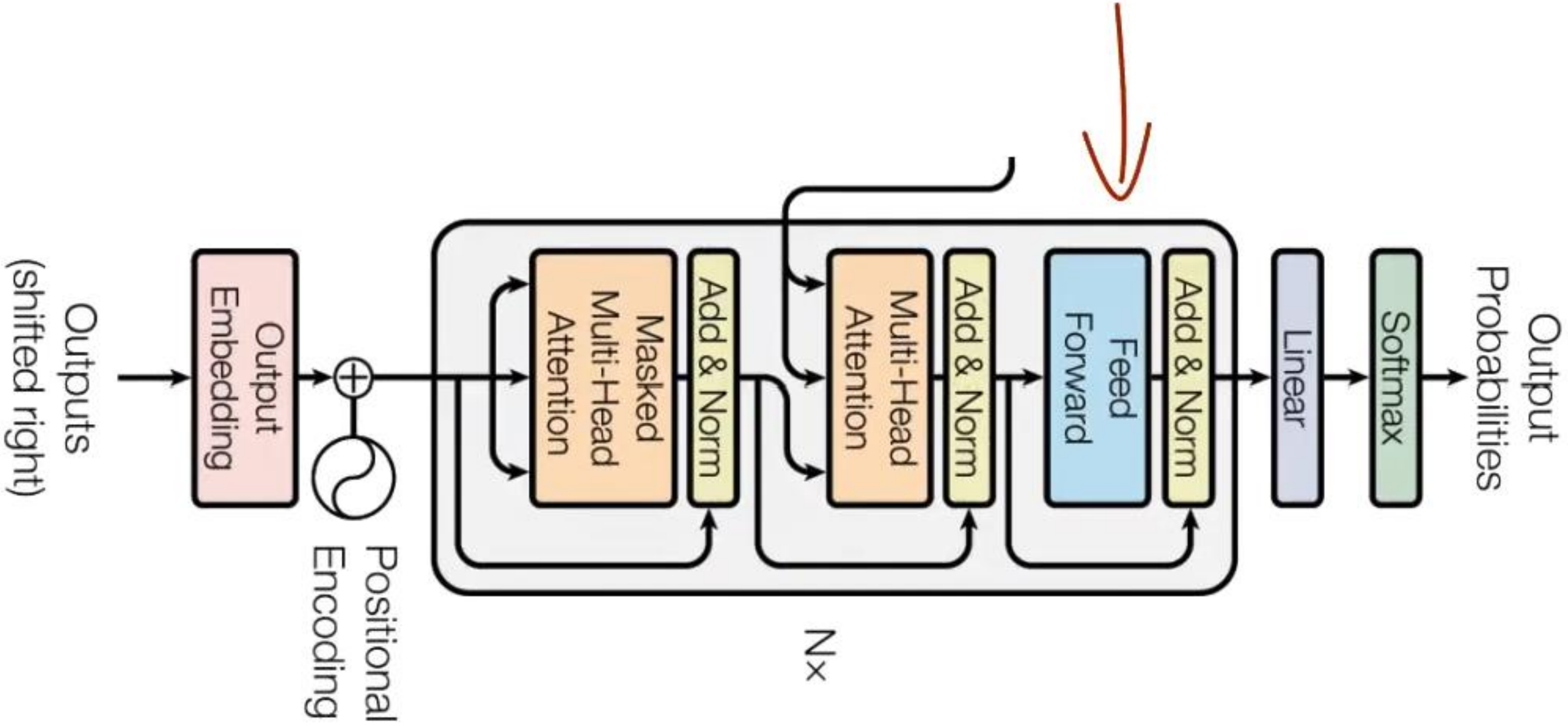
$\begin{bmatrix}1\\0\\0\\0\end{bmatrix}$	$\begin{bmatrix}0.1\\0.9\\0\\0\end{bmatrix}$	$\begin{bmatrix}0.05\\0.40\\0.55\\0\end{bmatrix}$	$\begin{bmatrix}0.16\\0.09\\0.15\\0.66\end{bmatrix}$
Le	gros	chien	rouge

Encoder-Decoder Attention

$\begin{bmatrix}0.71\\0.04\\0.07\\0.18\end{bmatrix}$	$\begin{bmatrix}0.01\\0.84\\0.02\\0.13\end{bmatrix}$	$\begin{bmatrix}0.09\\0.05\\0.62\\0.24\end{bmatrix}$	$\begin{bmatrix}0.03\\0.03\\0.03\\0.91\end{bmatrix}$
The	big	red	dog

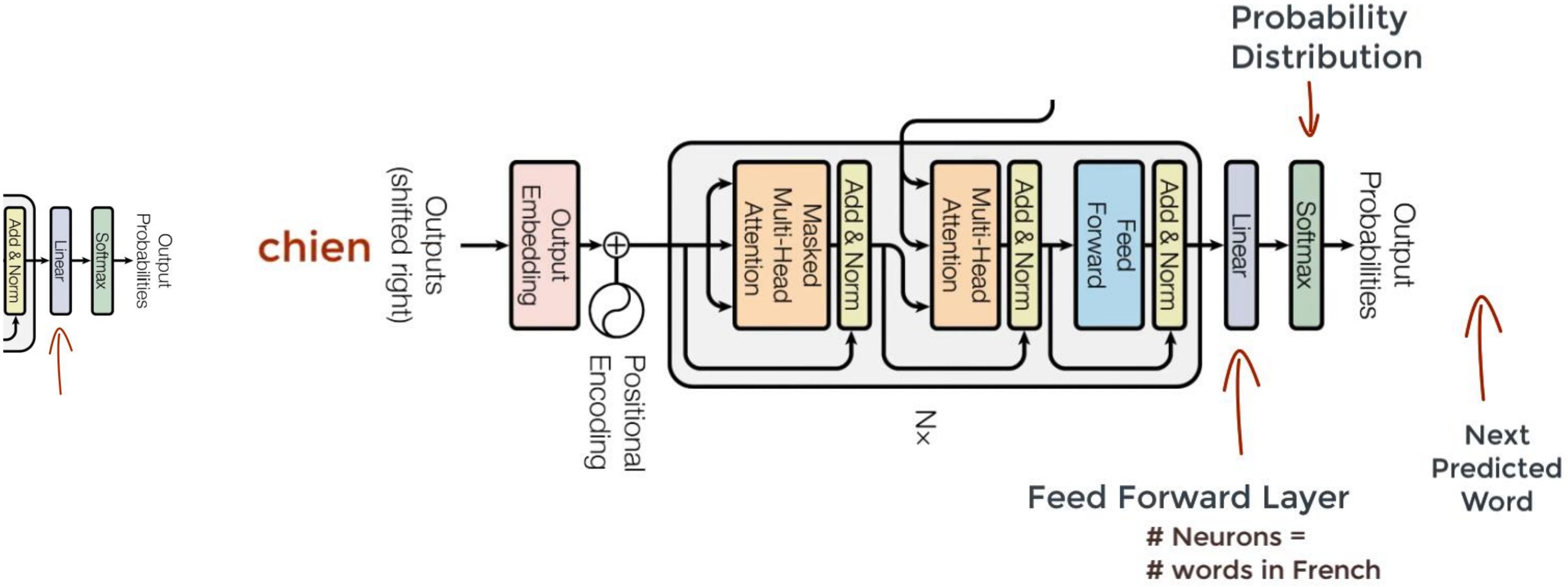
Transformer Components

Le



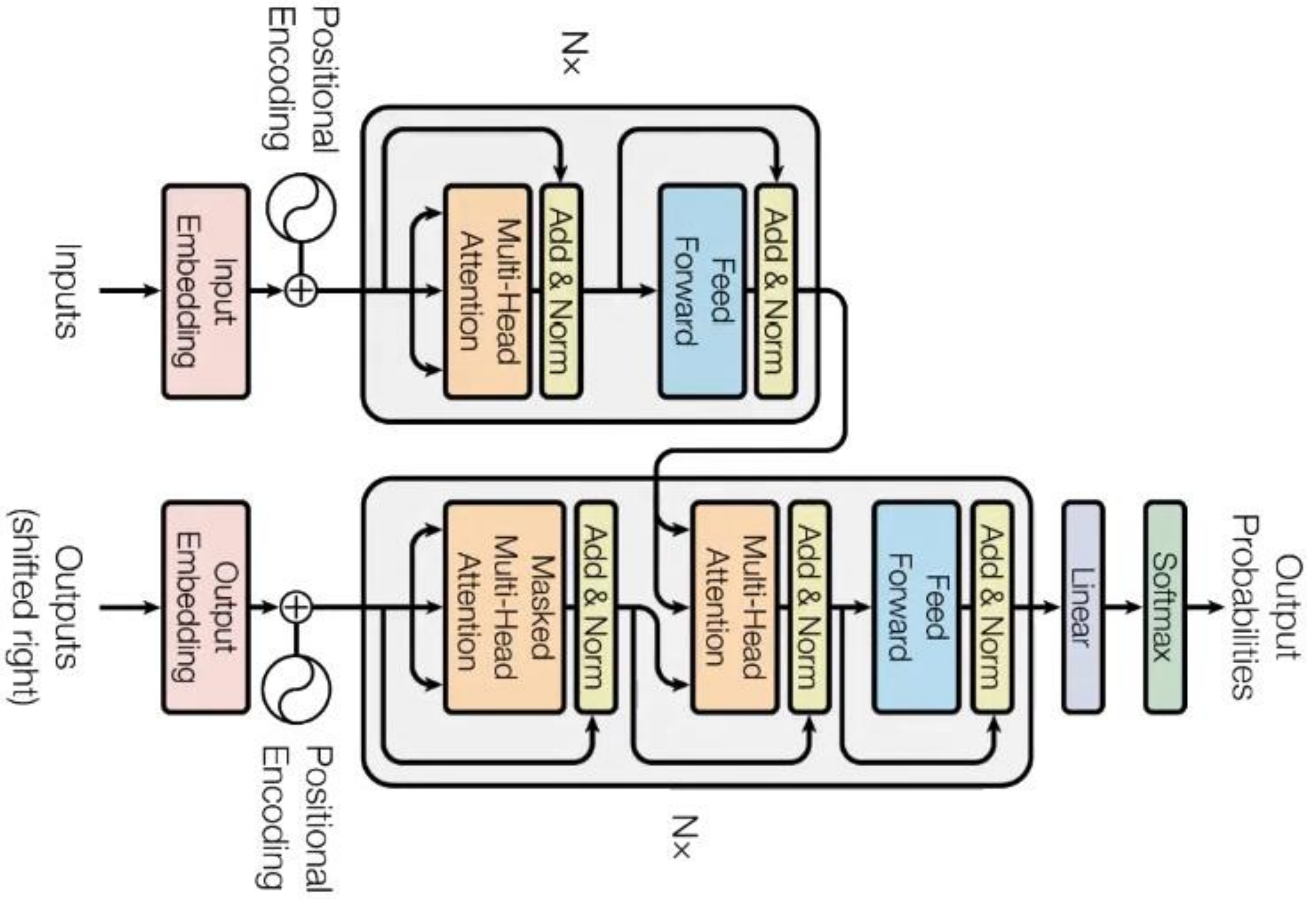
WE PASS EACH ATTENTION VECTOR TO THE FEED FORWARD UNIT

Transformer Components



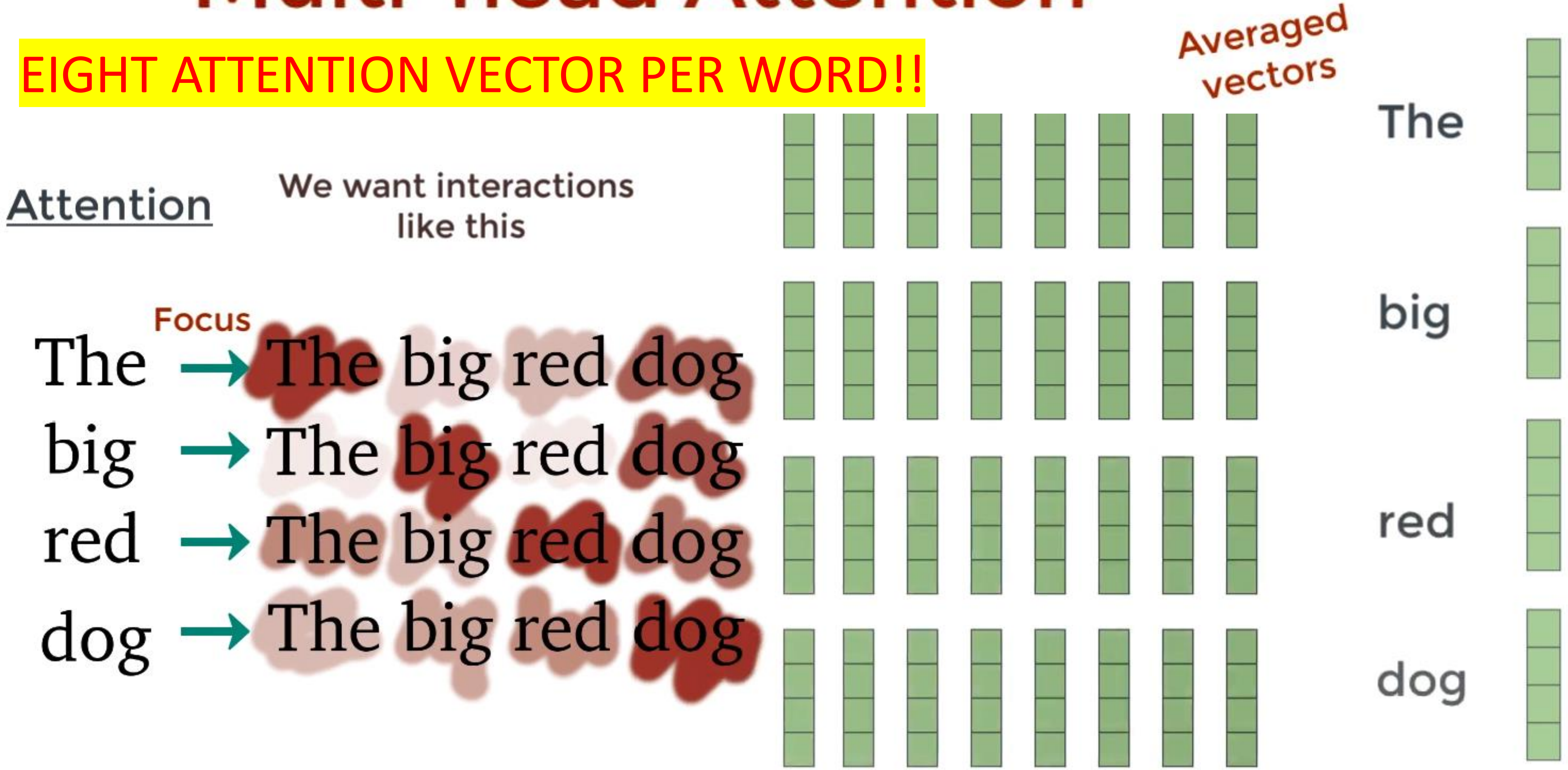
Transformer Components

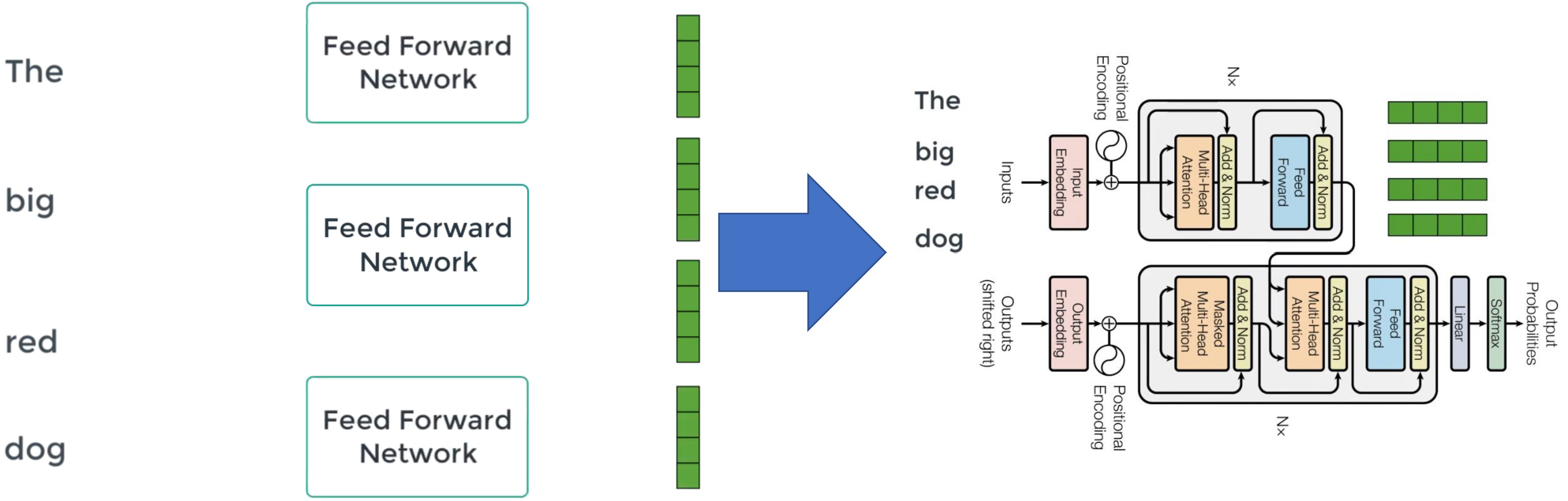
Next Pass



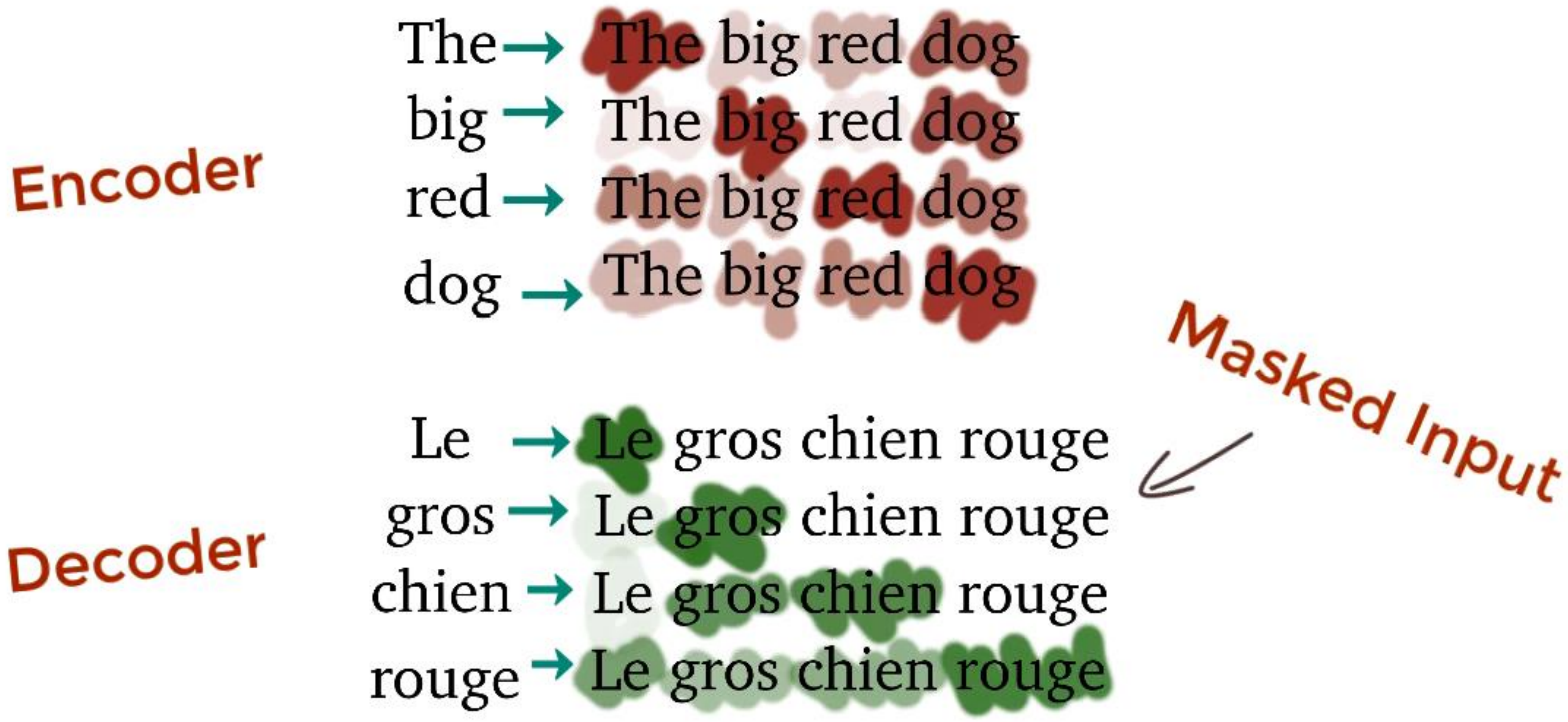
Multi-head Attention

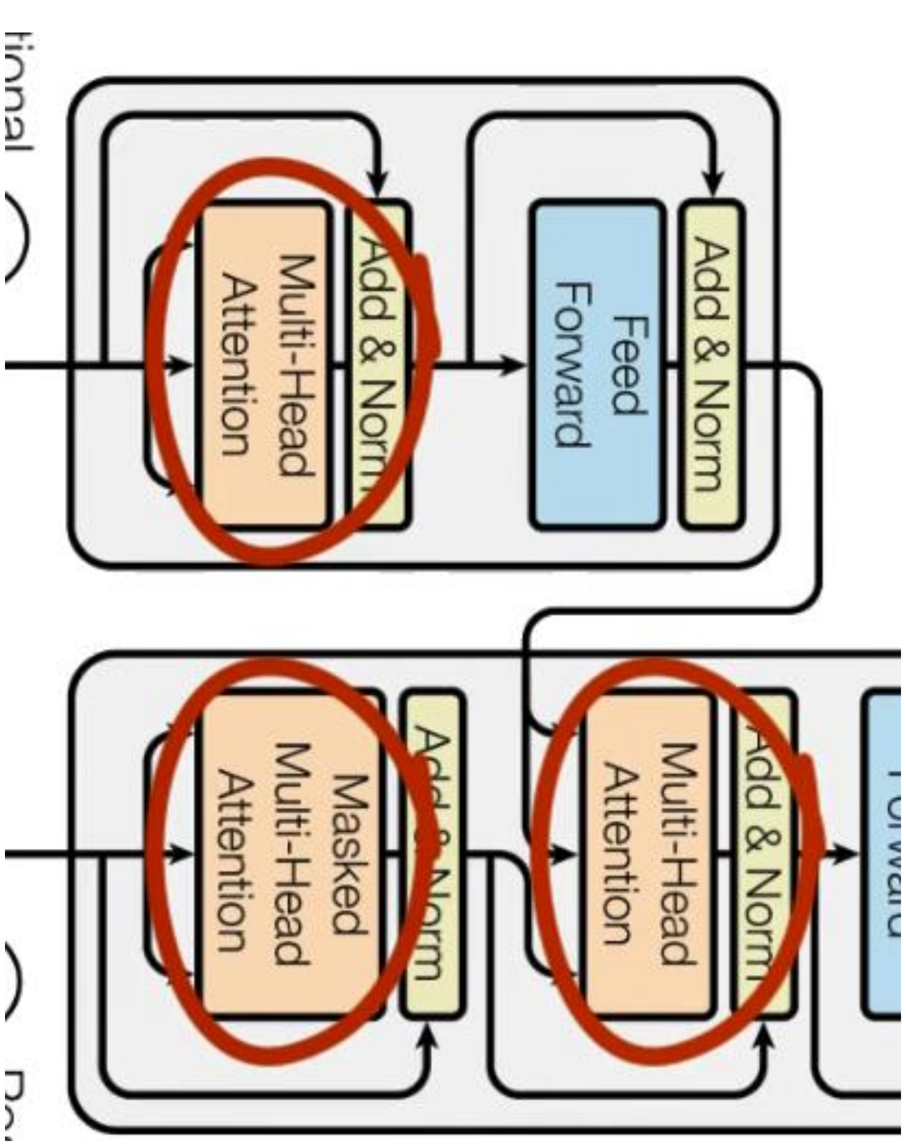
EIGHT ATTENTION VECTOR PER WORD!!



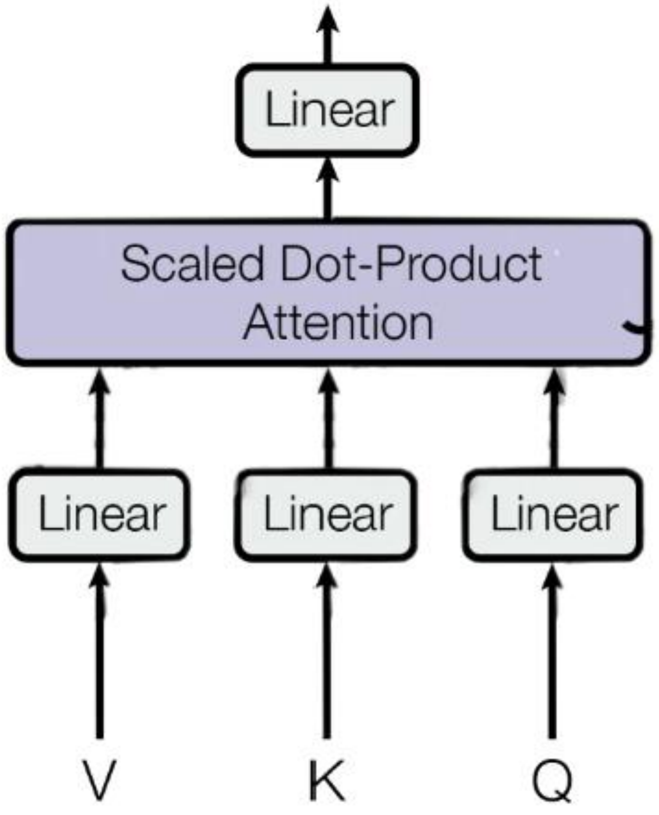

The
big
red
dog
Feed Forward Network
Feed Forward Network
Feed Forward Network
The
big
red
dog
Inputs
Input Embedding
Positional Encoding
Multi-Head Attention
Add & Norm
Feed Forward
Add & Norm
Nx
Outputs (shifted right)
Output Embedding
Positional Encoding
Masked Multi-Head Attention
Add & Norm
Multi-Head Attention
Add & Norm
Feed Forward
Add & Norm
Nx
Linear
Softmax
Output Probabilities

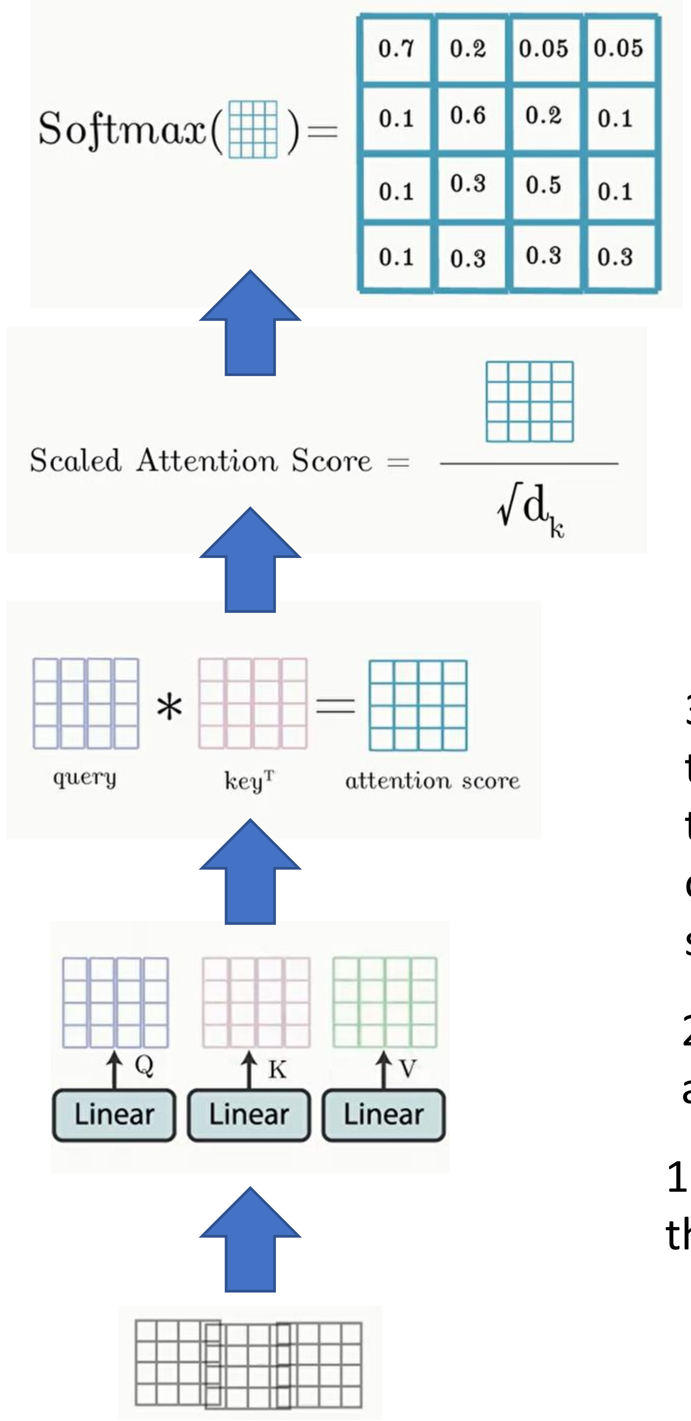
Multi-headed Attention





HOW MULTI-HEAD ATTENTION LOOKS LIKE ?





5) These scores are passed into a soft max layer which returns probability values between 0 and 1. Finally this SOFTMAX attention scores are used to multiply the value vector to produce a weighted output vector

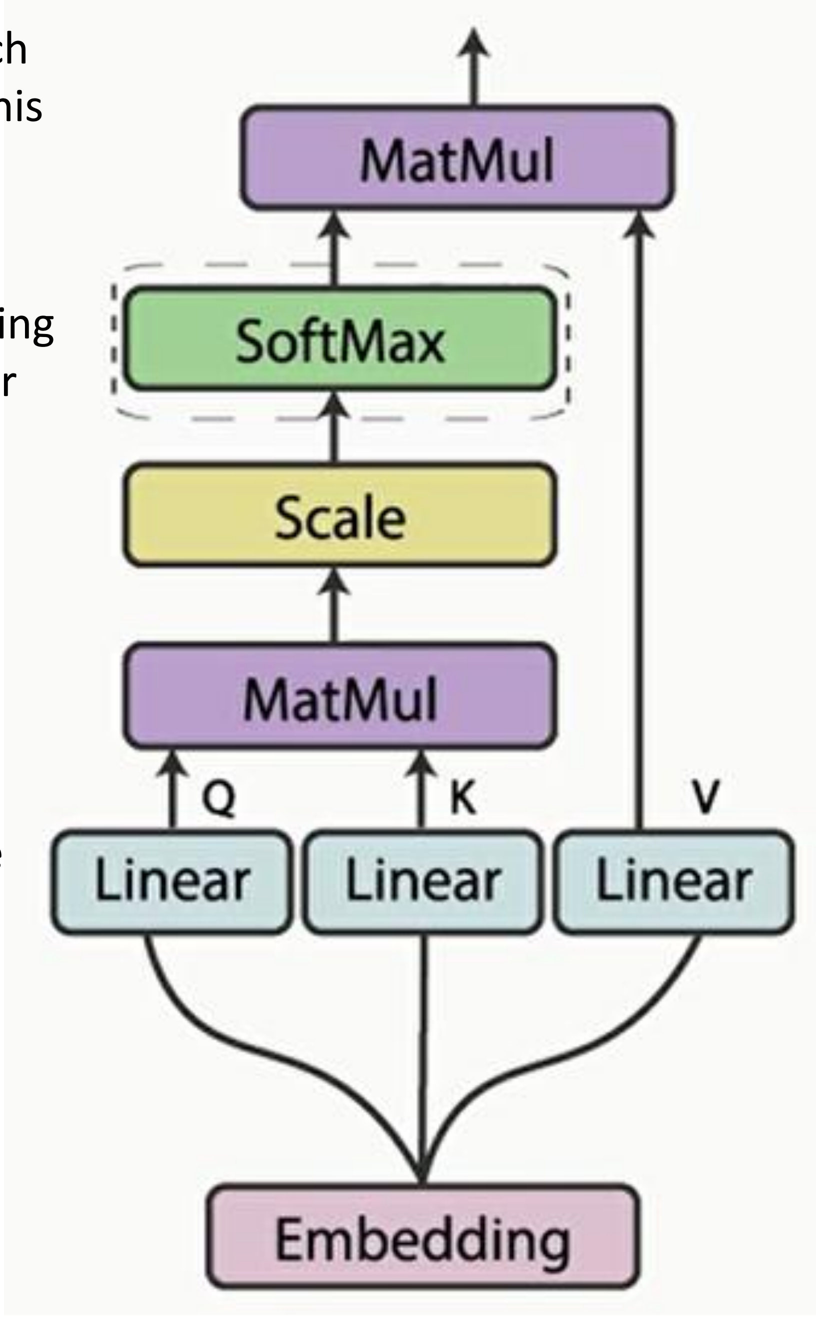
4) The scoring matrix then gets scaled down by dividing by the square root of the dimension of the key vector and this is to allow for more stable gradients as multiplying these values several times can have exploding effects

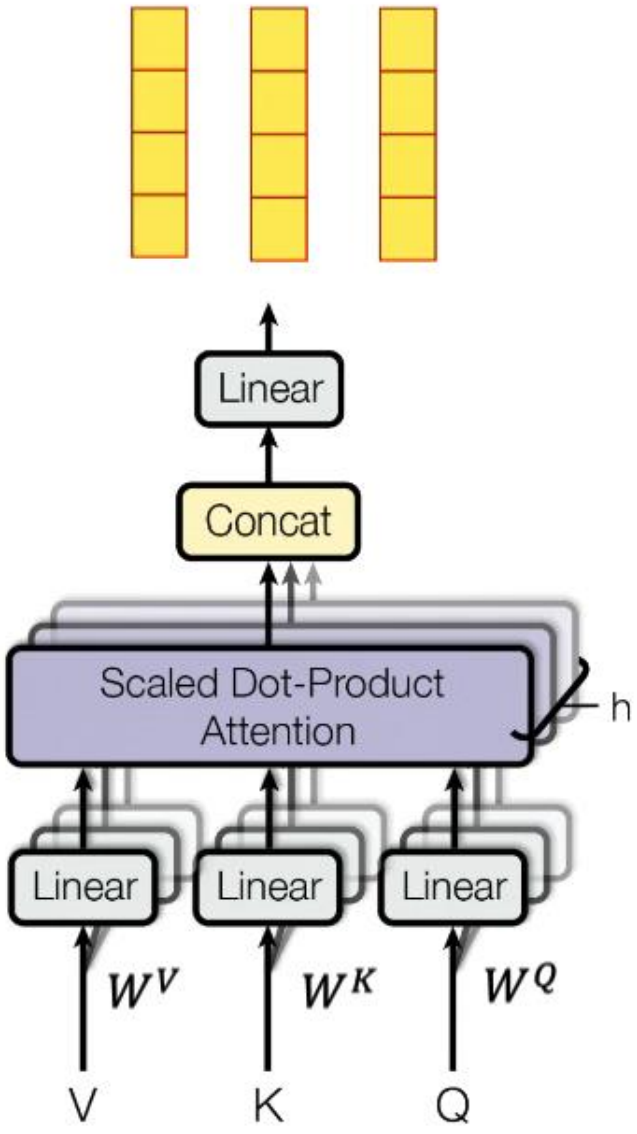
3) Dot multiplication is applied to the query and the transpose of the key to produce a scoring matrix and this scoring matrix is what determines the relevance of a word to other words in the matrix the higher the score the more relevant the word is

2) Create the ENCODER attention query Q key K and value V vectors

1) The positional input embeddings are feed into three distinct linear layers

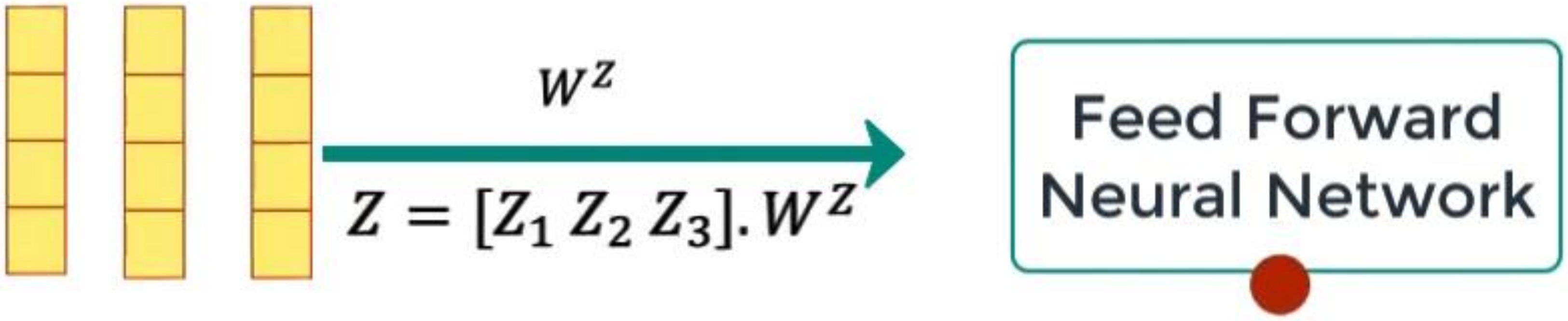
$$Attention(Q, K, V) = softmax_k\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$





MULTIPLE ATTENTION Z for every word!!

$$Z = softmax\left(\frac{Q.K^T}{\sqrt{Dimension\ of\ vector\ Q, K\ or\ V}}\right).V$$



Matrix Wz → OUTPUT “Z” IS STILL ATTENTION VECTOR PER WORD

AFTER MULTI-HEAD ATTENTION – WE ADD THE NORMALIZATION STEP