May 20th, 2025
Ay 119

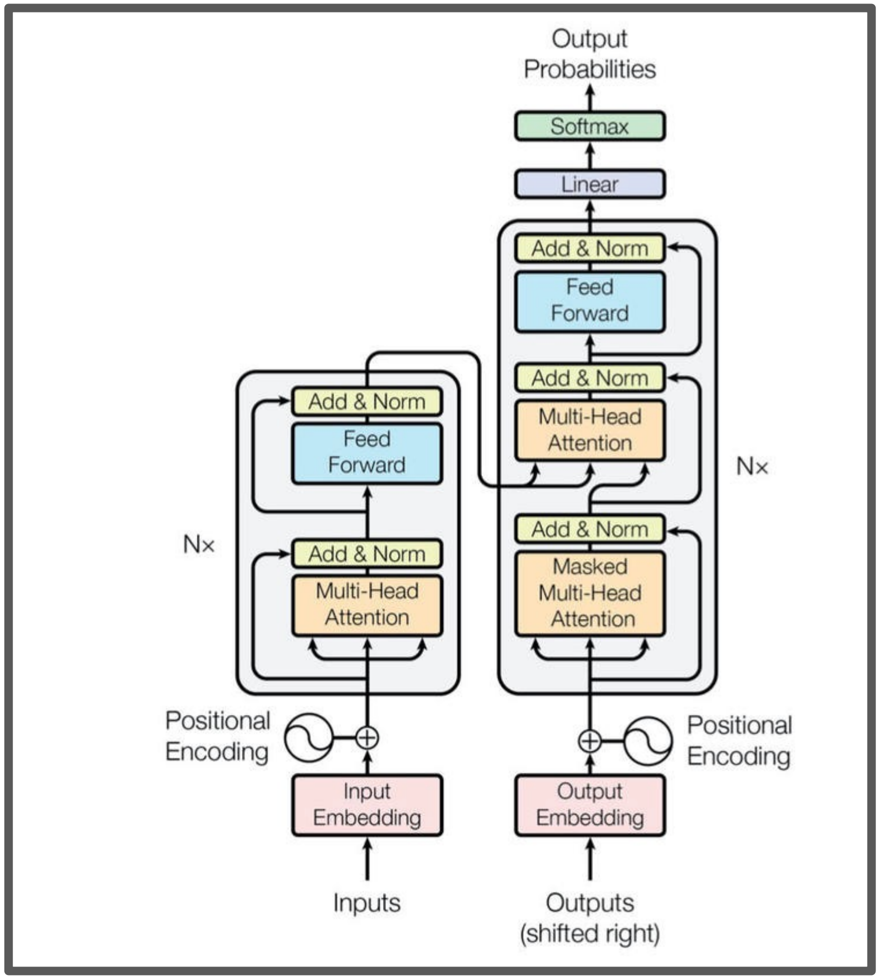
Advanced Deep Learning:

A Fleetingly Brief Introduction to LLMs

Matthew J. Graham
and Pavlos Protopapas

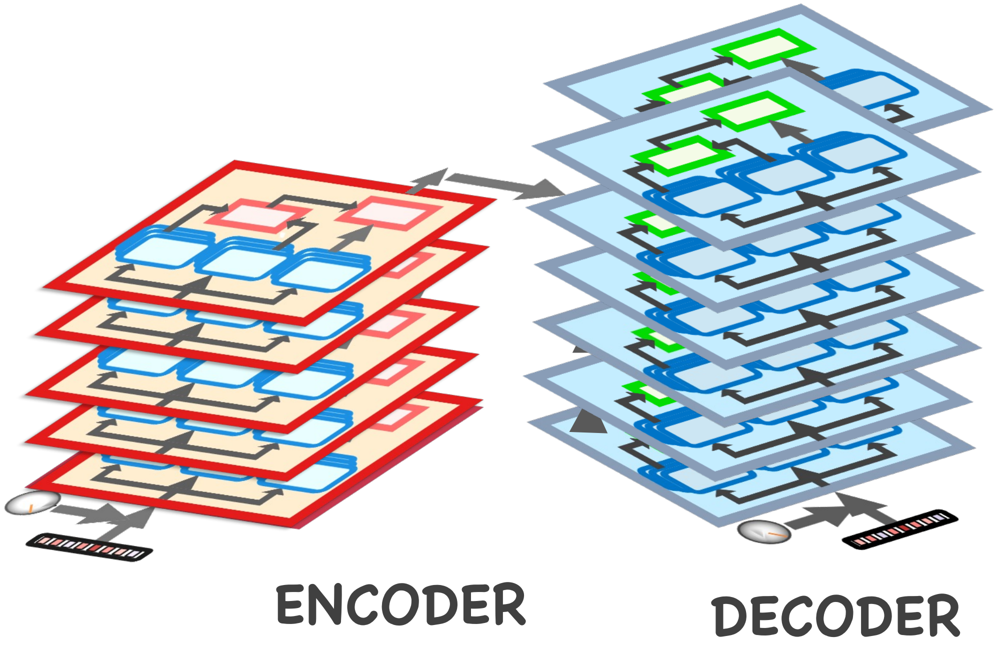
Deep learning: recap from last time

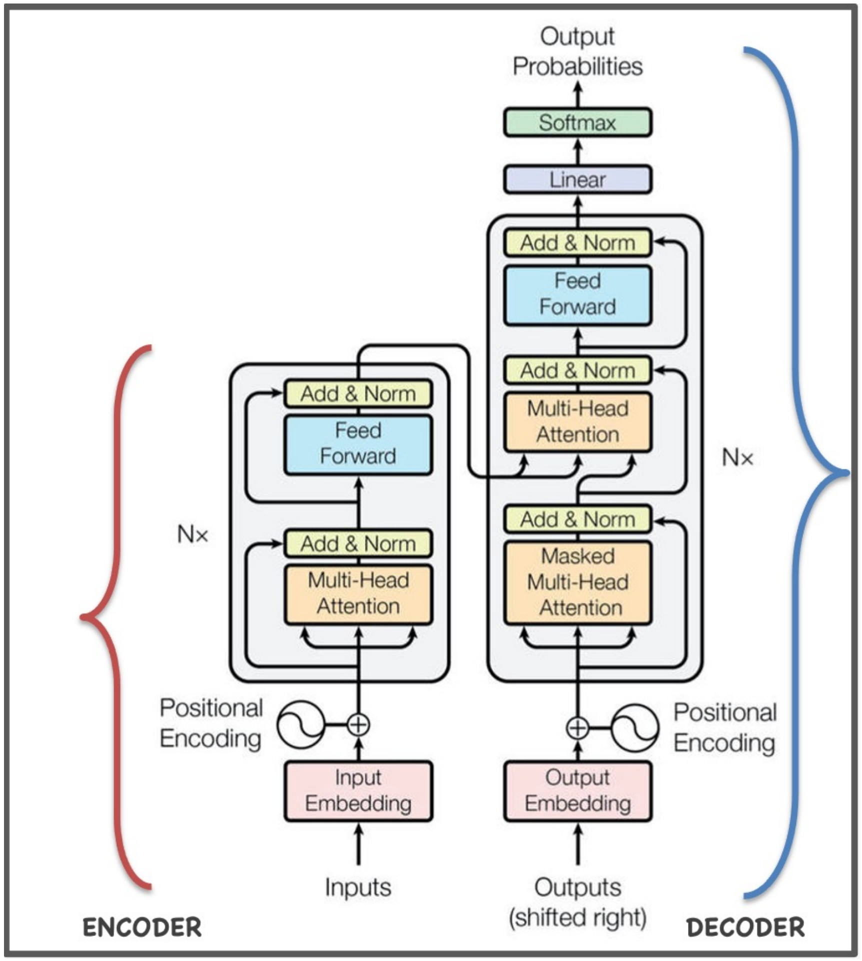
- Sequential data
- Tokenization
- Embeddings
- Attention
- Positional encoding
- Transformers



Transformers

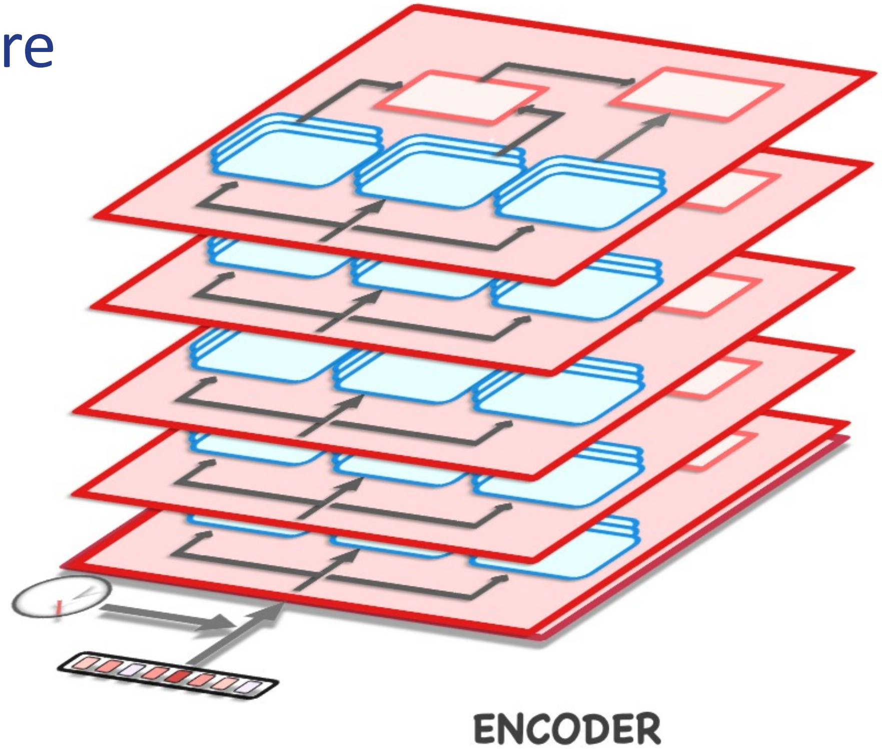
- They consist of an encoder-decoder architecture, using multi-head attention blocks





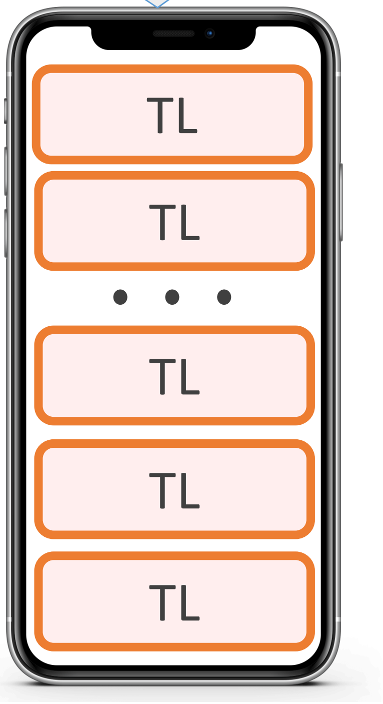
Transformers -> BERT

- Instead of an encoder-decoder architecture, what if we just use the encoder?
- This led to the new architecture called **B**idirectional **E**ncoder **R**epresentations from **T**ransformers or BERT
- BERT set new records in LM tasks
- BERT's code and models are open source and adaptable
- BERT is ideal for transfer learning



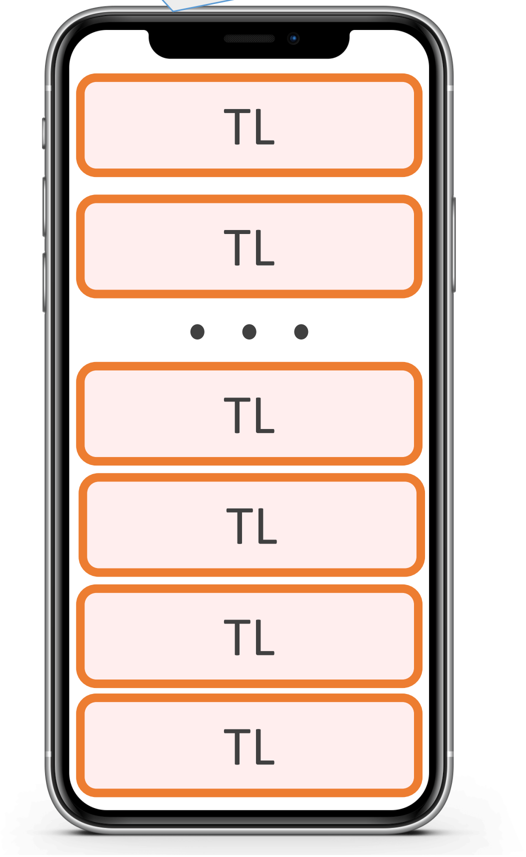
Flavors of BERT

12 TRANSFORMER LAYERS (TL) WITH 12 HEADS
768 HIDDEN UNITS



BERT BASIC

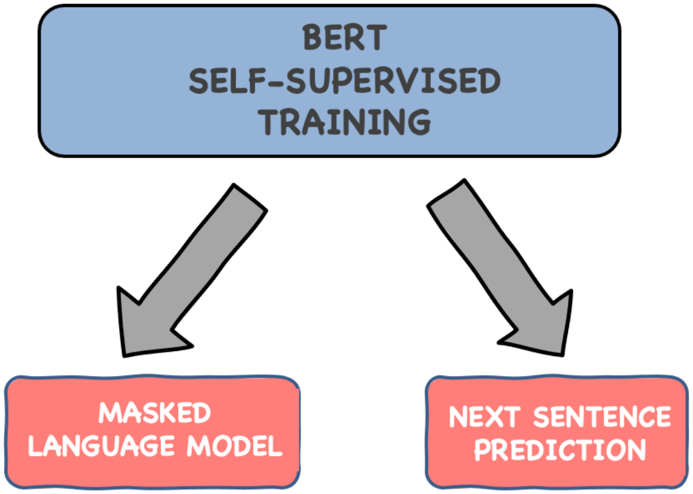
24 TRANSFORMER LAYERS (TL) EACH WITH 16 HEADS
1024 HIDDEN UNITS



BERT LARGE

Training BERT

- Two phases: pretraining and fine tuning
- Phase 1:
  - If BERT takes the entire sequence as an input, how can we train it as a LM => Masked language modeling (MLM)
  - How can we classify if two chosen sentences follow each other or are out of sequence if the output is also a sequence of tokens => Next sentence prediction (NSP)



BERT SELF-SUPERVISED TRAINING

MASKED LANGUAGE MODEL

NEXT SENTENCE PREDICTION

Training BERT: Masked language modeling

[CLS] Donald was hit by a bus as they were crossing the street [SEP]

[CLS] Donald was hit by a [MASK] as they were crossing the street [SEP]

[CLS] Donald was hit by a [MASK] as they were crossing the street [SEP]

[CLS] Donald was hit by a [MASK] as they were crossing the potato [SEP]

- Input sequences to BERT must have the following special tokens:
  - [CLS] – Classifier token
  - [MASK] – Masking token
  - [SEP] – Separator token
- Loss function only evaluated on masked tokens
- Final layer (Dense + Softmax) predicts masked word
- 15% of input tokens selected for prediction:
  - 80% are masked
  - 10% are the same words
  - 10% are replaced with randomly replaced words

Training BERT: Masked language modeling

[CLS] Donald was hit by a bus as they were crossing the street [SEP]

[CLS] Donald was hit by a [MASK] as they were crossing the street [SEP]

[CLS] Donald was hit by a [MASK] as they were crossing the street [SEP]

[CLS] Donald was hit by a [MASK] as they were crossing the potato [SEP]

- Input sequences to BERT must have the following special tokens:
  - [CLS] – Classifier token
  - [MASK] – Masking token
  - [SEP] – Separator token
- Loss function only evaluated on masked tokens
- Final layer (Dense + Softmax) predicts masked word
- 15% of input tokens selected for prediction:
  - 80% are masked
  - 10% are the same words
  - 10% are replaced with randomly replaced words

Training BERT: Next Sentence Prediction

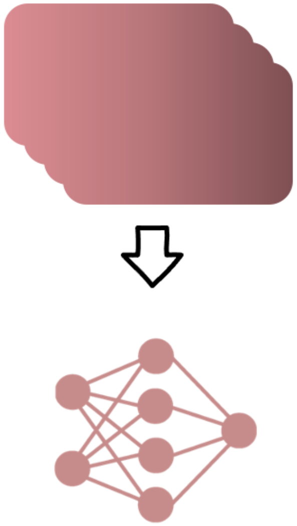
- Given two sentences A and B, is B likely to be the sentence that follows A or not?
- Form training pairs:
  - [CLS] A [SEP] B [SEP]
- Segment embeddings ensures words from A/B are assigned to the appropriate vector
- 50% of time B is the actual next sentence and 50% of time it is a random sentence from the corpus
- Binary classification: IsNext / NotNext
- NSP helps document level coherence and context-aware reasoning across sentences

Training BERT: Pre-training issues

- Base BERT has 109,482,240 trainable parameters requiring a very large data set for training
- BERT Large was pretrained on BooksCorpus which contain 800 millions words and Wikipedia containing 2.5 billion words = 40 TB
- BERT was trained for 40 epochs on 16 TPUs = 4 days / $18,000

Let’s talk about transfer learning

Train a large ML model on a large general dataset

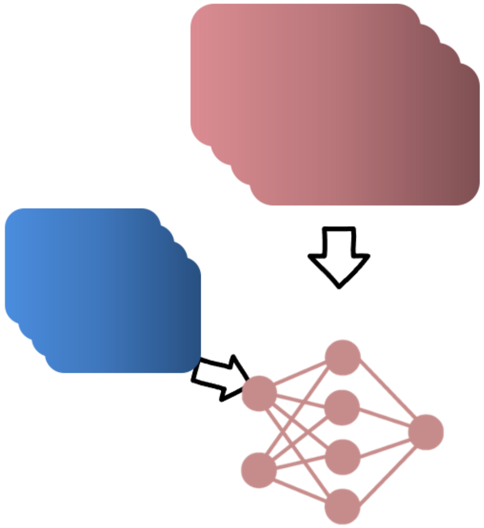


Let’s talk about transfer learning

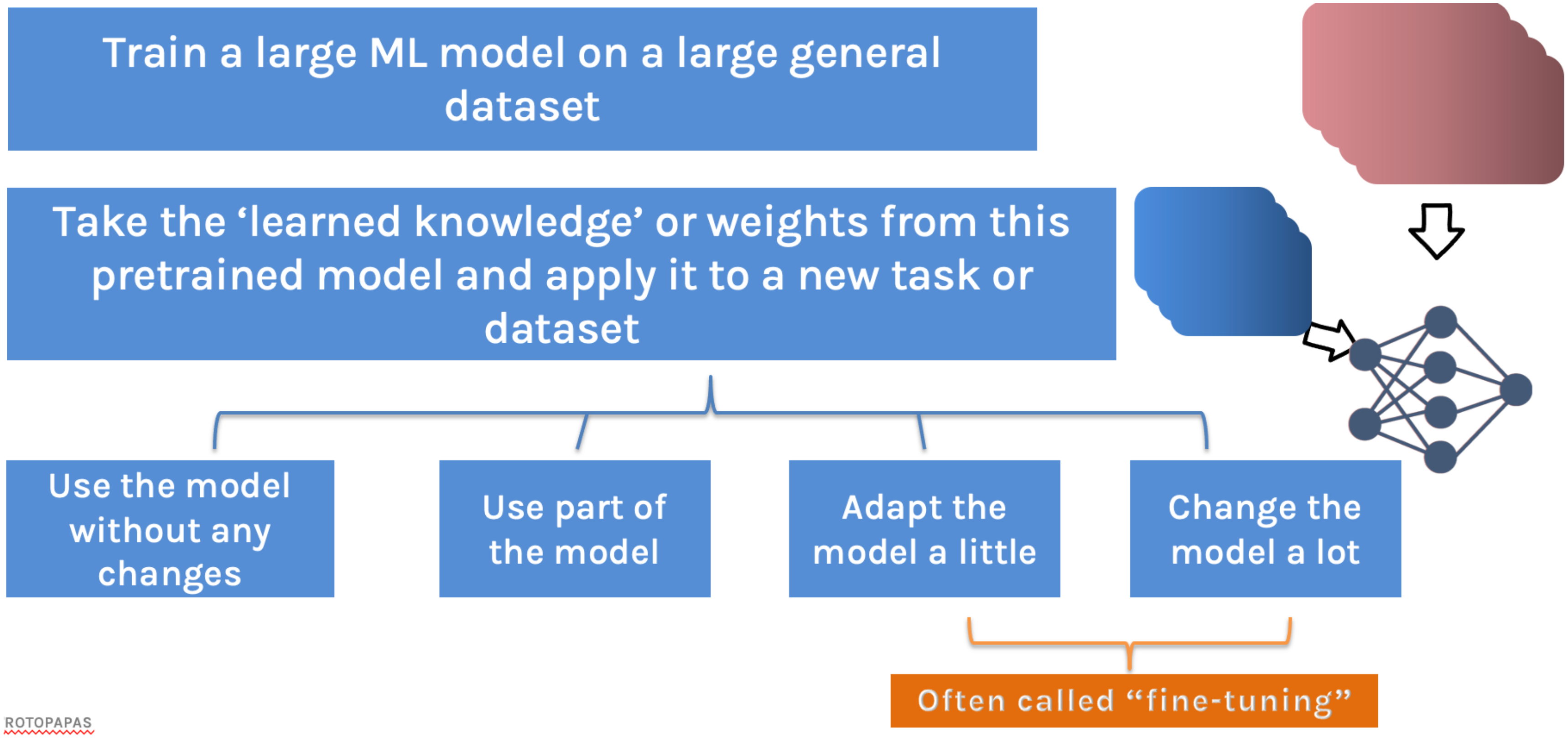
Train a large ML model on a large general dataset

Take the ‘learned knowledge’ or weights from this pretrained model and apply it to a new task or dataset

This can be achieved in a few ways…



Let’s talk about transfer learning



Train a large ML model on a large general dataset

Take the ‘learned knowledge’ or weights from this pretrained model and apply it to a new task or dataset

- Use the model without any changes
- Use part of the model
- Adapt the model a little
- Change the model a lot

Often called “fine-tuning”

ROTOPAPAS

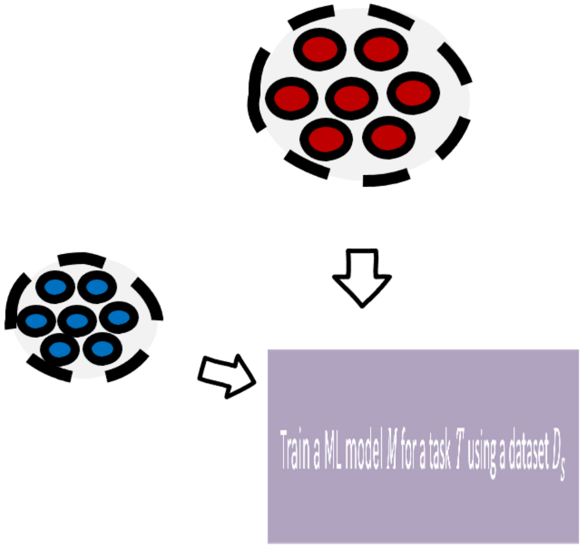
Let’s talk about transfer learning

Train a ML model M for a task T using a dataset D_s

For example, using the Wikipedia task to train a language model for next word prediction.

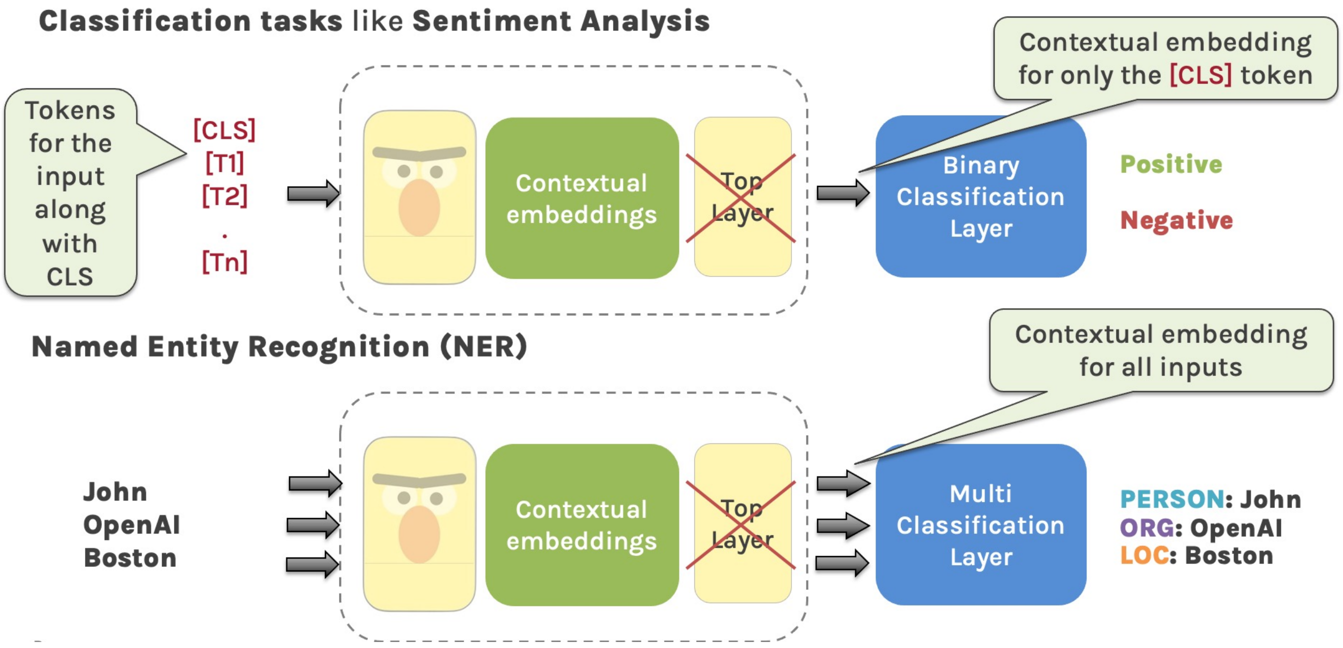
Use model M on a new dataset D_T for a similar task T_n

For example, using this model for Part of Speech tagging on a smaller dataset.



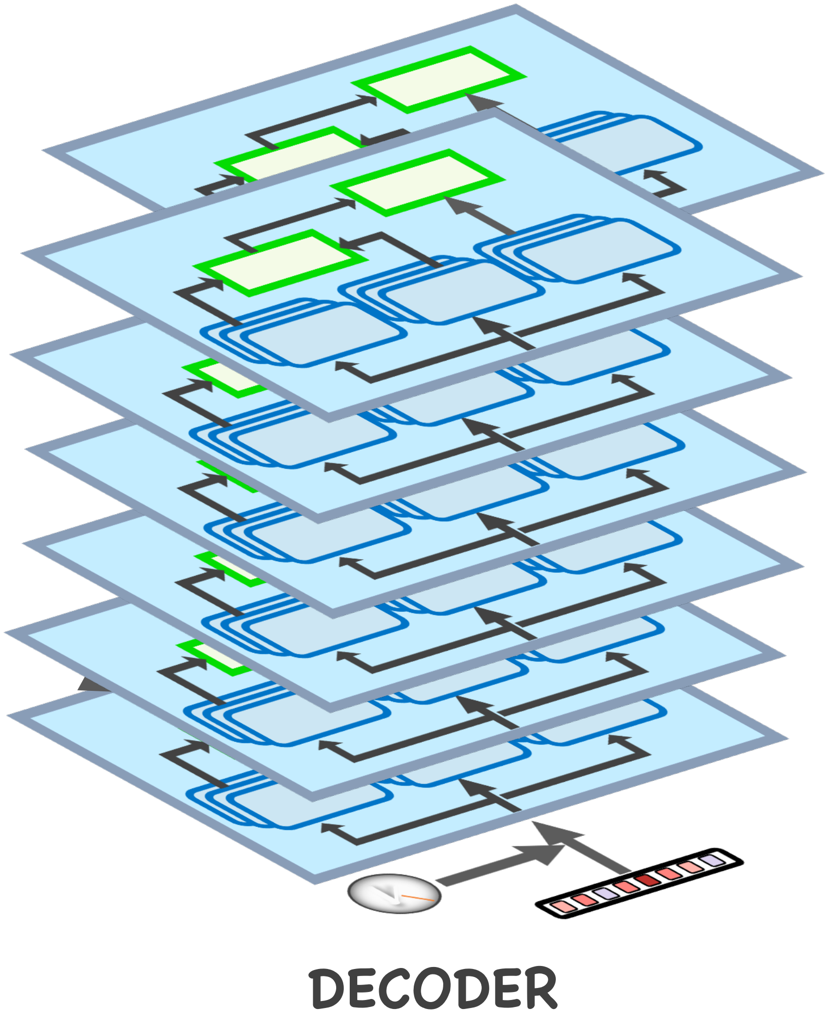
Let's talk about fine tuning BERT

- The process of providing a model with labelled examples to update its weight and improve performance on a specific task
- MLM is used to fine tune BERT
  - Load pretrained weights trained on a very large corpus and then fine tune to a domain specific dataset
  - Remove BERT's top layer, extract the contextual embeddings, and pass them through a new dense layer

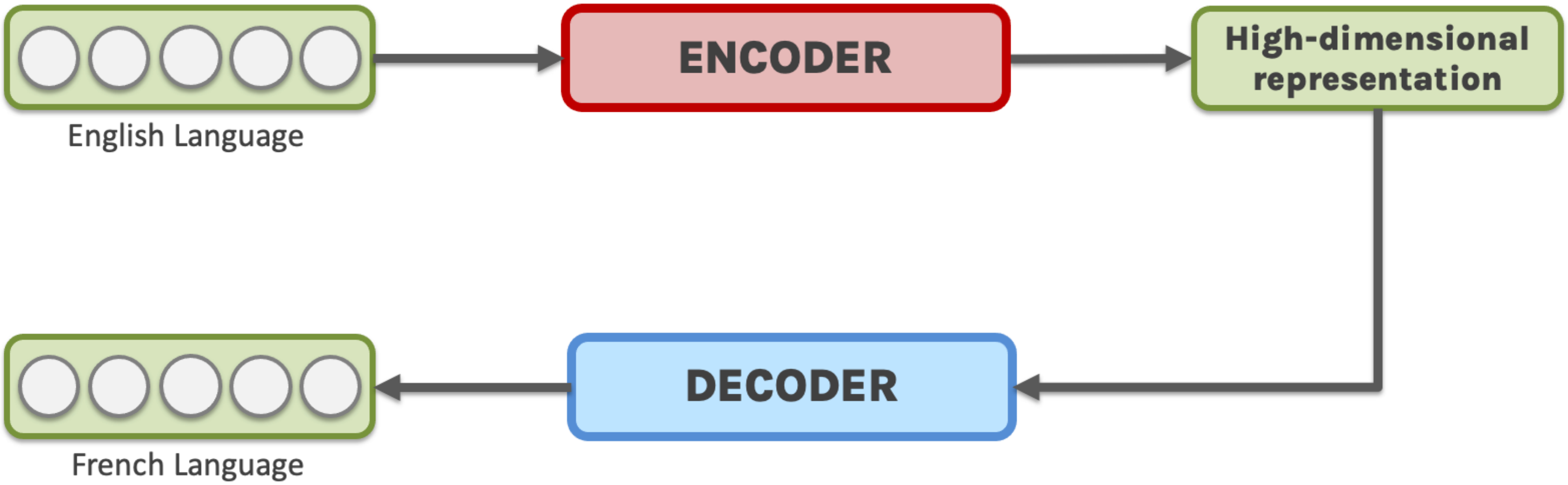

Classification tasks like Sentiment Analysis
Tokens for the input along with CLS
[CLS]
[T1]
[T2]
.
[Tn]
Contextual embeddings
Top Layer
Contextual embedding for only the [CLS] token
Binary Classification Layer
Positive
Negative
Named Entity Recognition (NER)
John
OpenAI
Boston
Contextual embeddings
Top Layer
Contextual embedding for all inputs
Multi Classification Layer
PERSON: John
ORG: OpenAI
LOC: Boston

Transformers -> GPT

- Instead of an encoder-decoder architecture, what if we just use the decoder?
- This led to the new architecture called **Generative Pre-trained Transformer** or GPT
- BERT is autoencoding but GPT is autoregressive

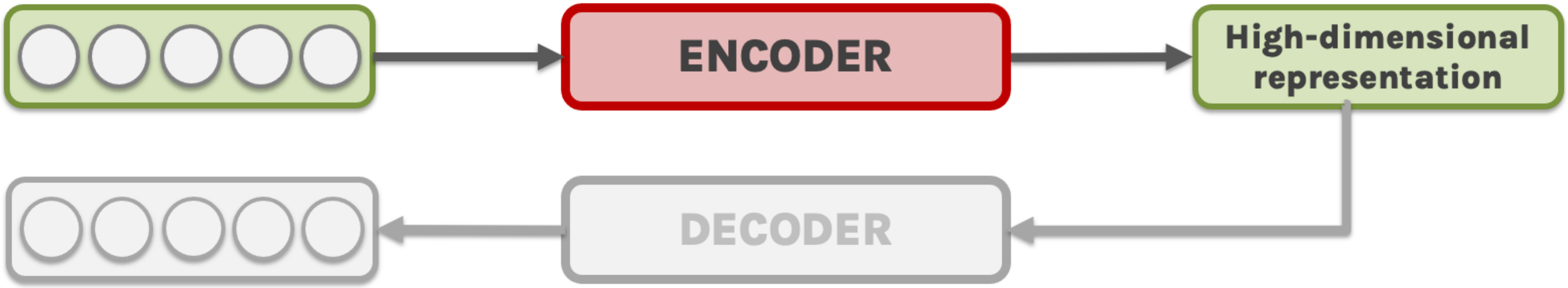


Recap: Transformer



Auto-encoding model

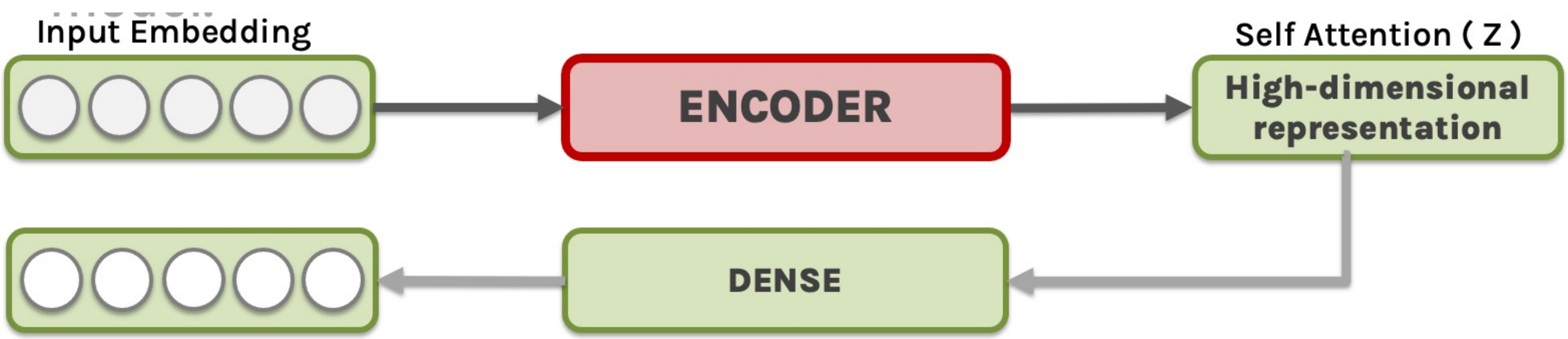
- Now if we take just the encoder blocks and train that part of the network separately, we have an autoencoding model



- The aim of an autoencoding model is to learn a representation of the data. One way to do this is to train the model on a reconstruction task.

Auto-encoding model

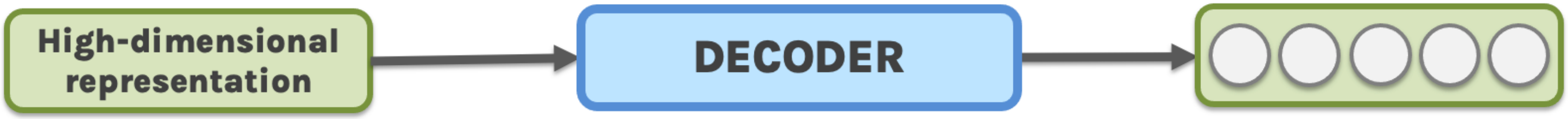
- Now if we take just the encoder blocks and train that part of the network separately, we have an autoencoding model



- The aim of an autoencoding model is to learn a representation of the data. One way to do this is to train the model on a reconstruction task.

Auto-regressive model

- Now if we take just the decoder blocks and train that part of the network separately, we have an autoregressive model



- A model is said to be autoregressive if it predicts future values based on past values.

Explaining BERT

- BERT is a denoising autoencoder model

- A denoising autoencoder model takes a noisy or corrupted input and attempts to *reconstruct* the original information
- For BERT, this mean reconstructing the masked words
- Uses bidirectional attention (sees past and future words)
- Trained to “understand text” by creating useful representations of text (i.e., embeddings)
- Embeddings used in downstream NLP tasks

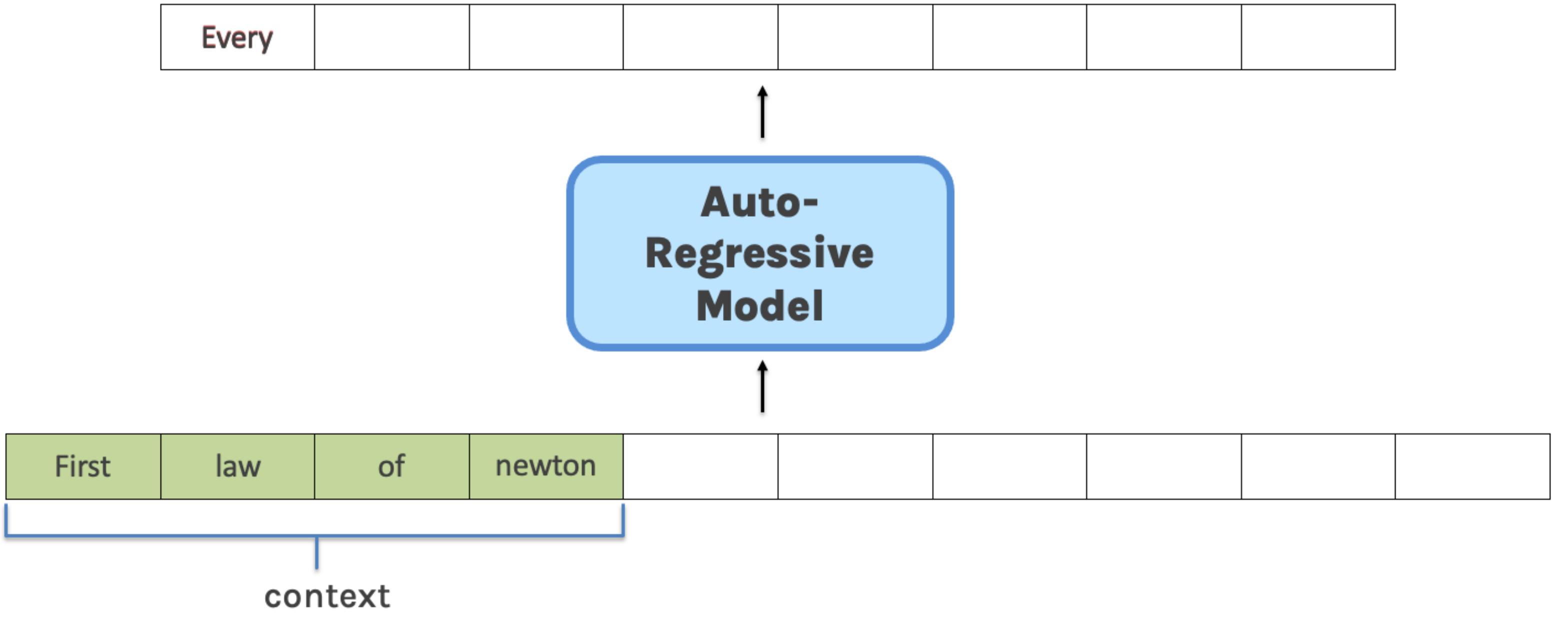
Explaining GPT

- GPT is an autoregressive model

- An autoregressive model predicts the next value in a sequence. In our case, these are language tokens.
- Uses “causal” attention (sees only past words)
- Can generate fluent text step-by-step
- Commonly used for chatbots, story writing, college assignments, etc.

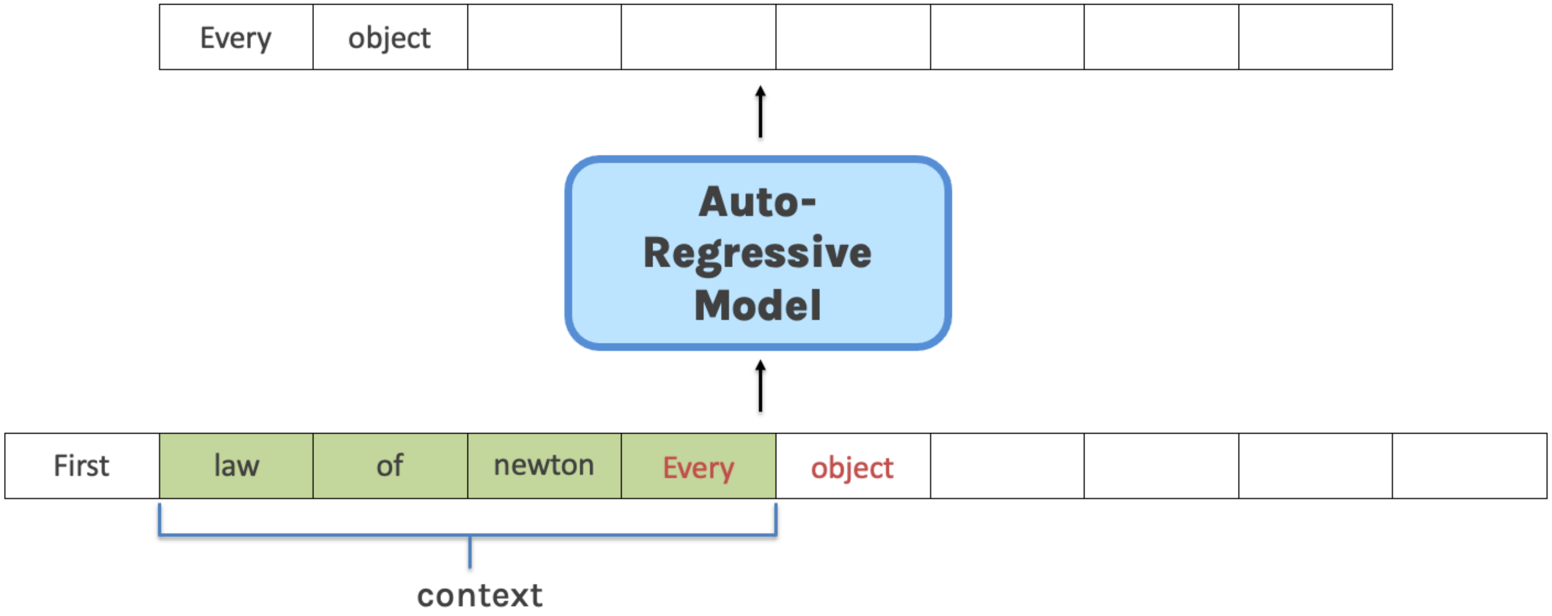
Autoregressive model

- Unidirectional processing and a fixed length sliding window are characteristic of a general auto-regressive model



Autoregressive model

- Unidirectional processing and a fixed length sliding window are characteristic of a general auto-regressive model



Every | object

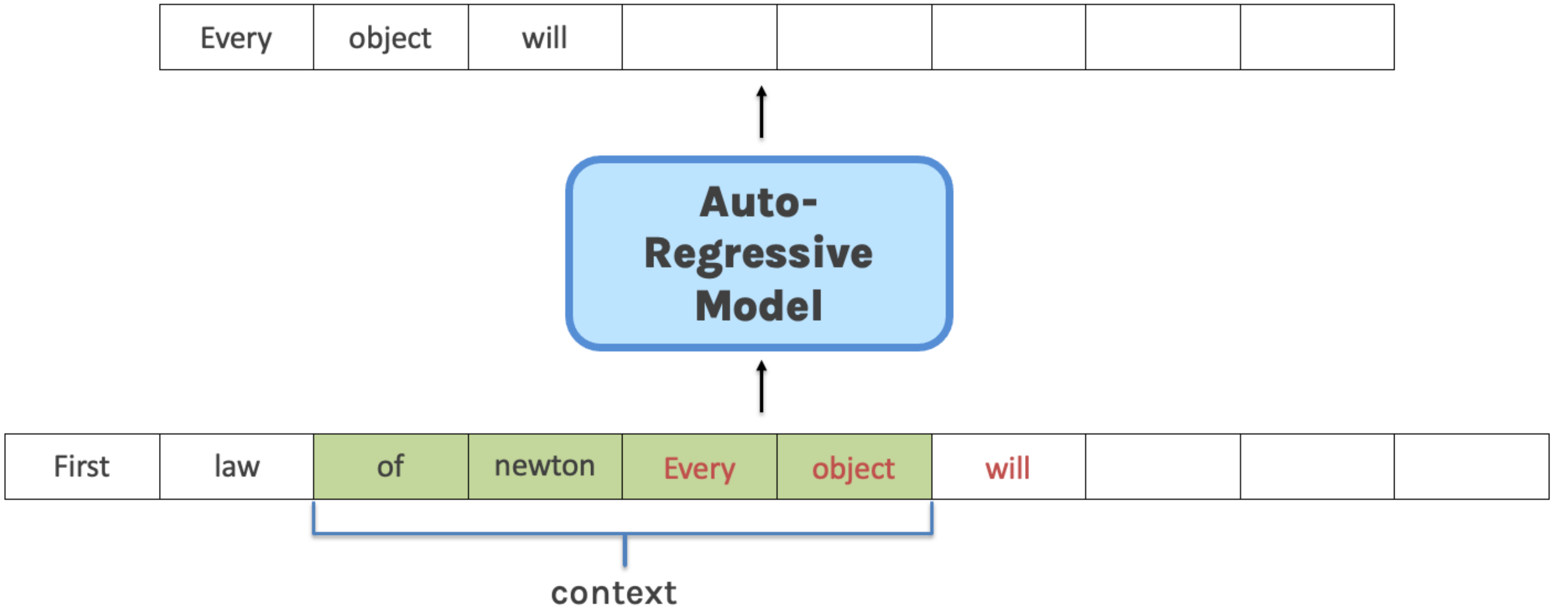
Auto-Regressive Model

First | law | of | newton | Every | object

context

Autoregressive model

- Unidirectional processing and a fixed length sliding window are characteristic of a general auto-regressive model



Every	object	will					

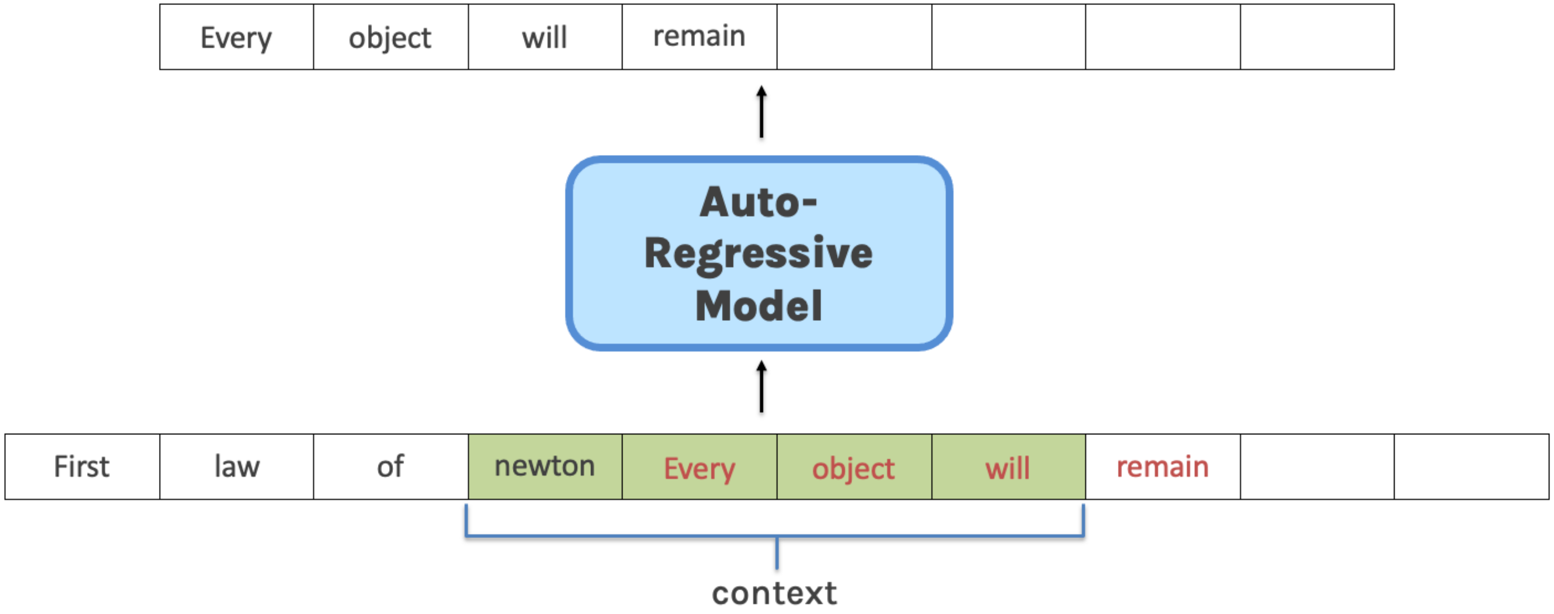
Auto-Regressive Model

First	law	of	newton	Every	object	will			

context

Autoregressive model

- Unidirectional processing and a fixed length sliding window are characteristic of a general auto-regressive model



Every	object	will	remain				

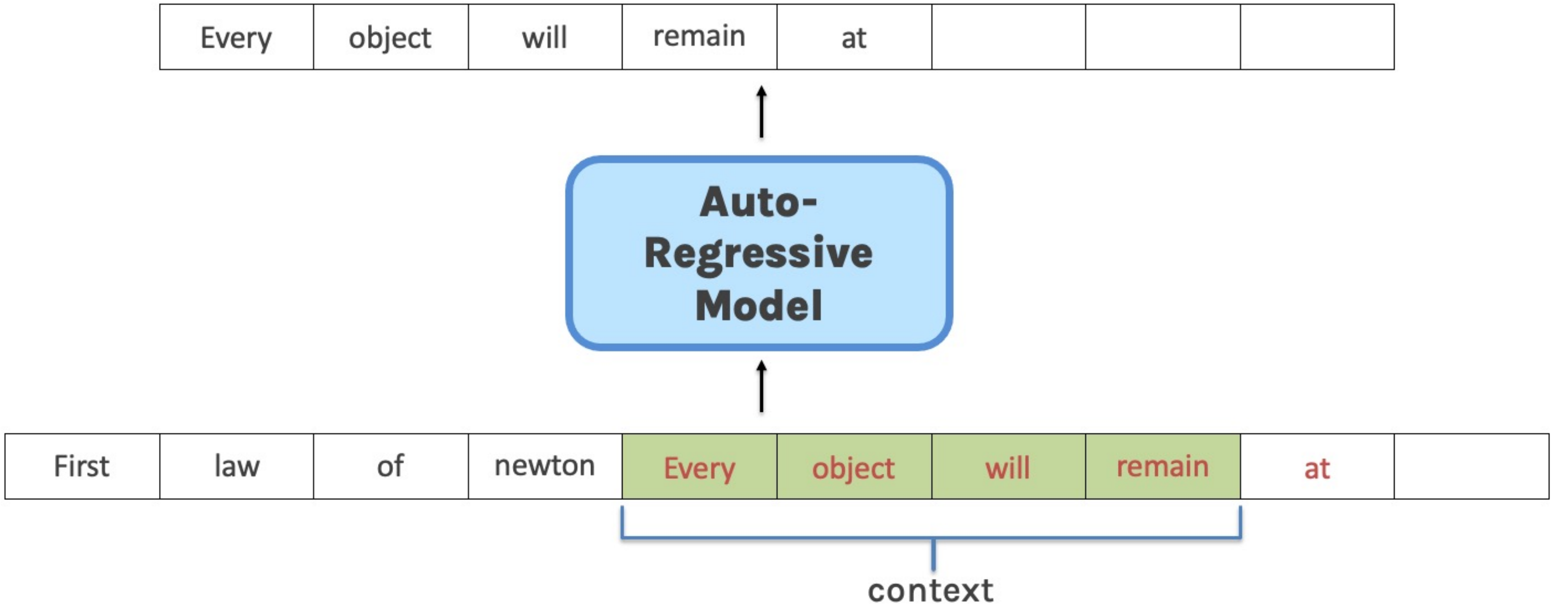
Auto-Regressive Model

First	law	of	newton	Every	object	will	remain		

context

Autoregressive model

- Unidirectional processing and a fixed length sliding window are characteristic of a general auto-regressive model



Every | object | will | remain | at

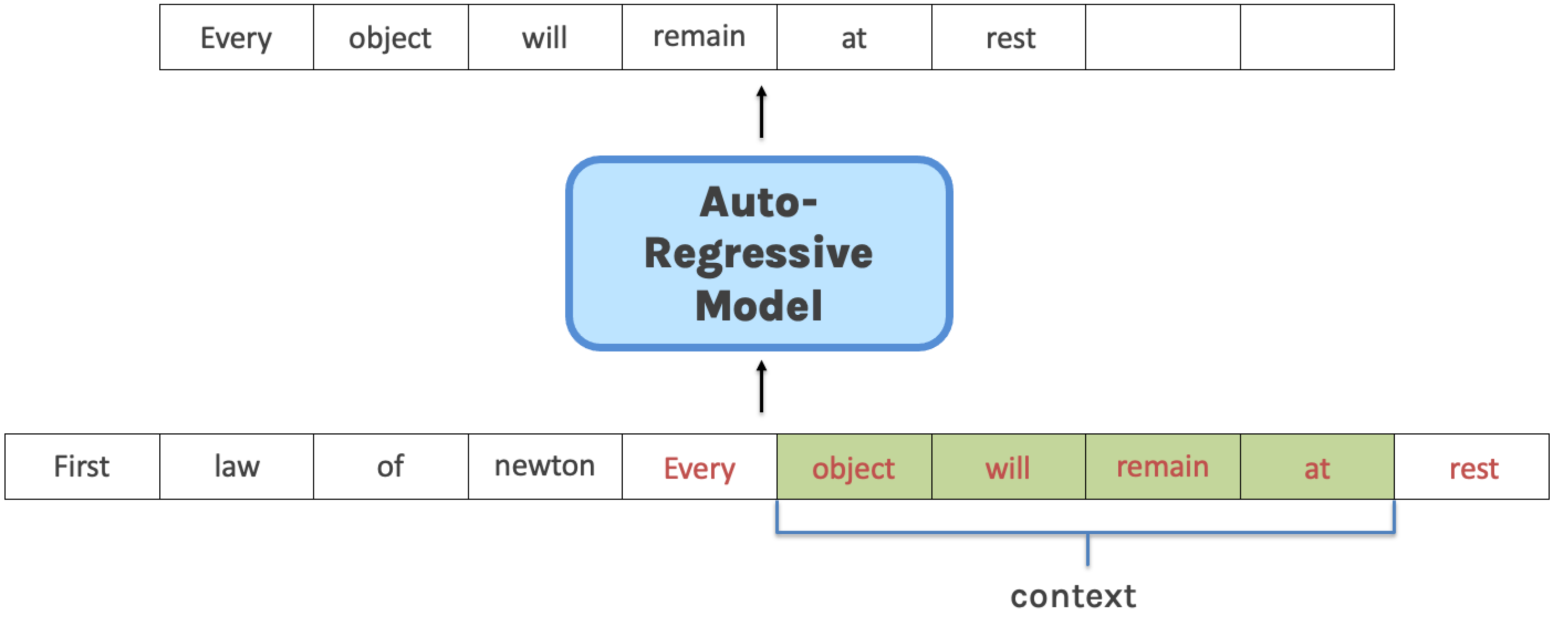
Auto-Regressive Model

First | law | of | newton | Every | object | will | remain | at

context

Autoregressive model

- Unidirectional processing and a fixed length sliding window are characteristic of a general auto-regressive model



Every	object	will	remain	at	rest		

Auto-Regressive Model

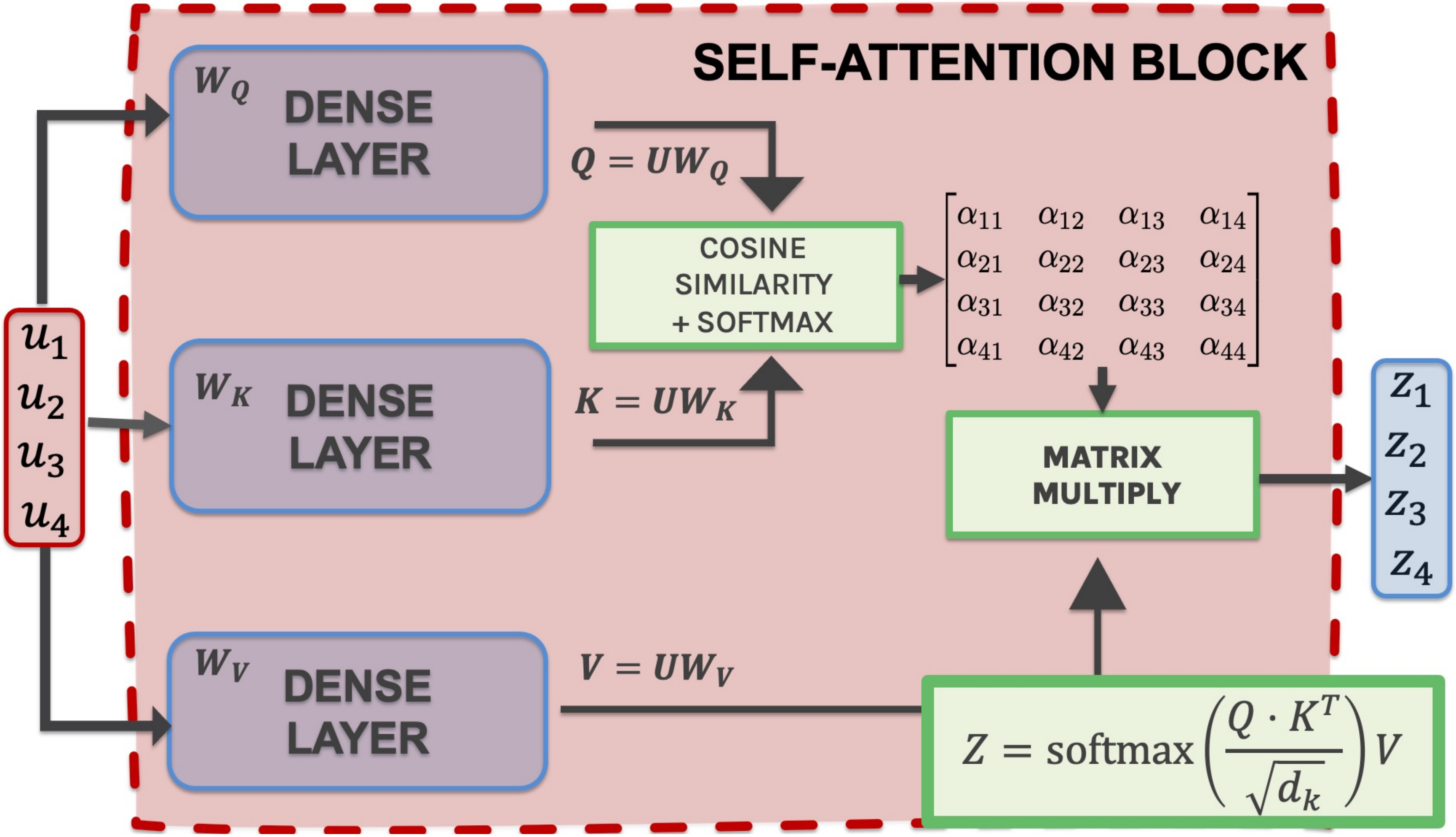
First	law	of	newton	Every	object	will	remain	at	rest

context

Training an autoregressive model

- Since every prediction is dependent on the previous one, this should be trained sequentially?
- But increasing sizes of transformer models makes this highly inefficient
- So we need to prevent the model from having a sneak peak into the future when training
- We achieve this by modifying the self-attention calculation

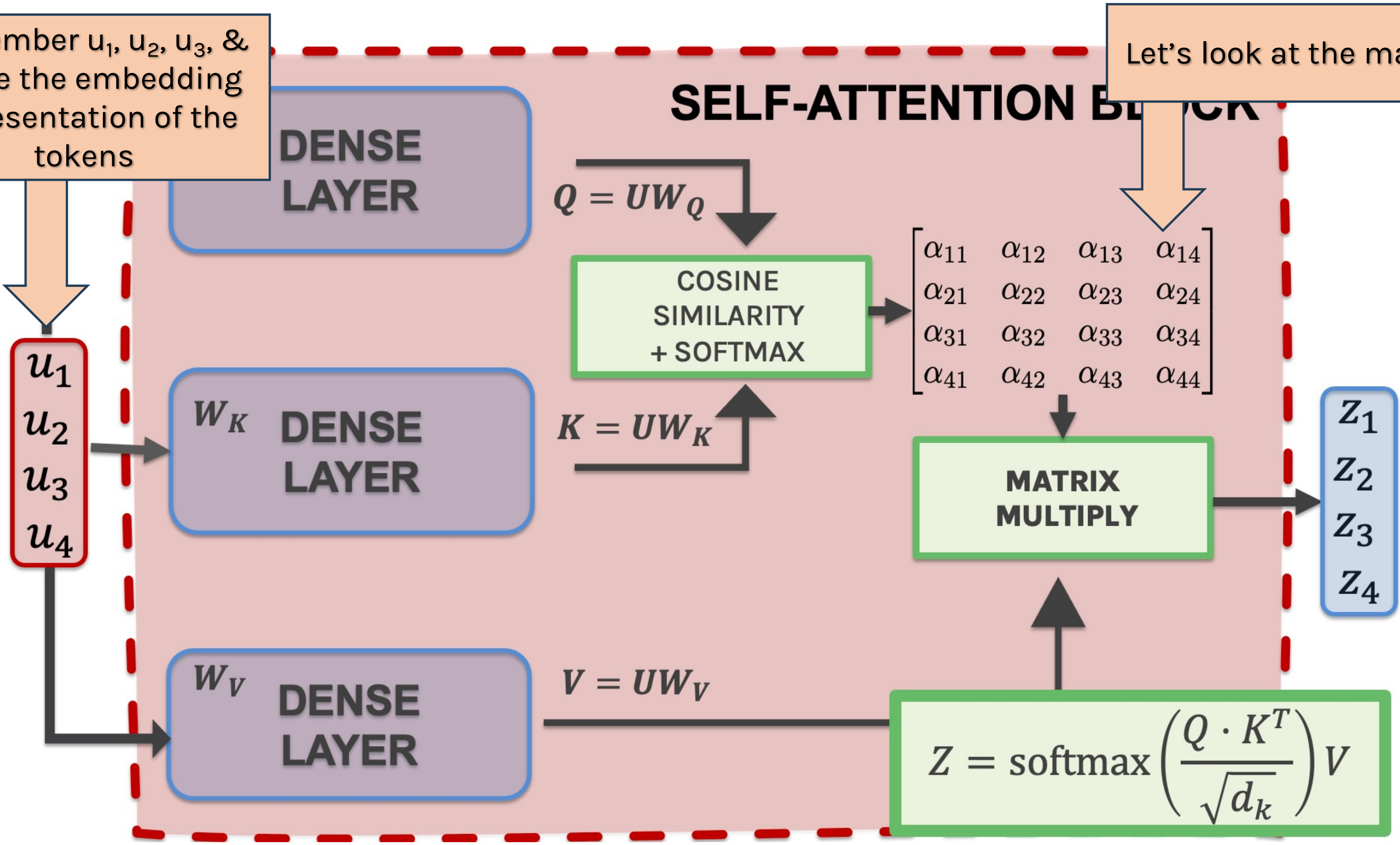
Recall the self-attention block


SELF-ATTENTION BLOCK
W_Q DENSE LAYER
W_K DENSE LAYER
W_V DENSE LAYER
u_1 u_2 u_3 u_4
$Q = UW_Q$
$K = UW_K$
$V = UW_V$
COSINE SIMILARITY + SOFTMAX
$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix}$
MATRIX MULTIPLY
$Z = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right)V$
z_1 z_2 z_3 z_4

Recall the self-attention block

Remember u_1, u_2, u_3, & u_4 are the embedding representation of the tokens

Let's look at the matrix



$$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix}$$

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

Recall the self-attention block

$$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix}$$

Recall the self-attention block

α_{12} is the similarity score for the first transformed token with the second transformed token

$$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix}$$

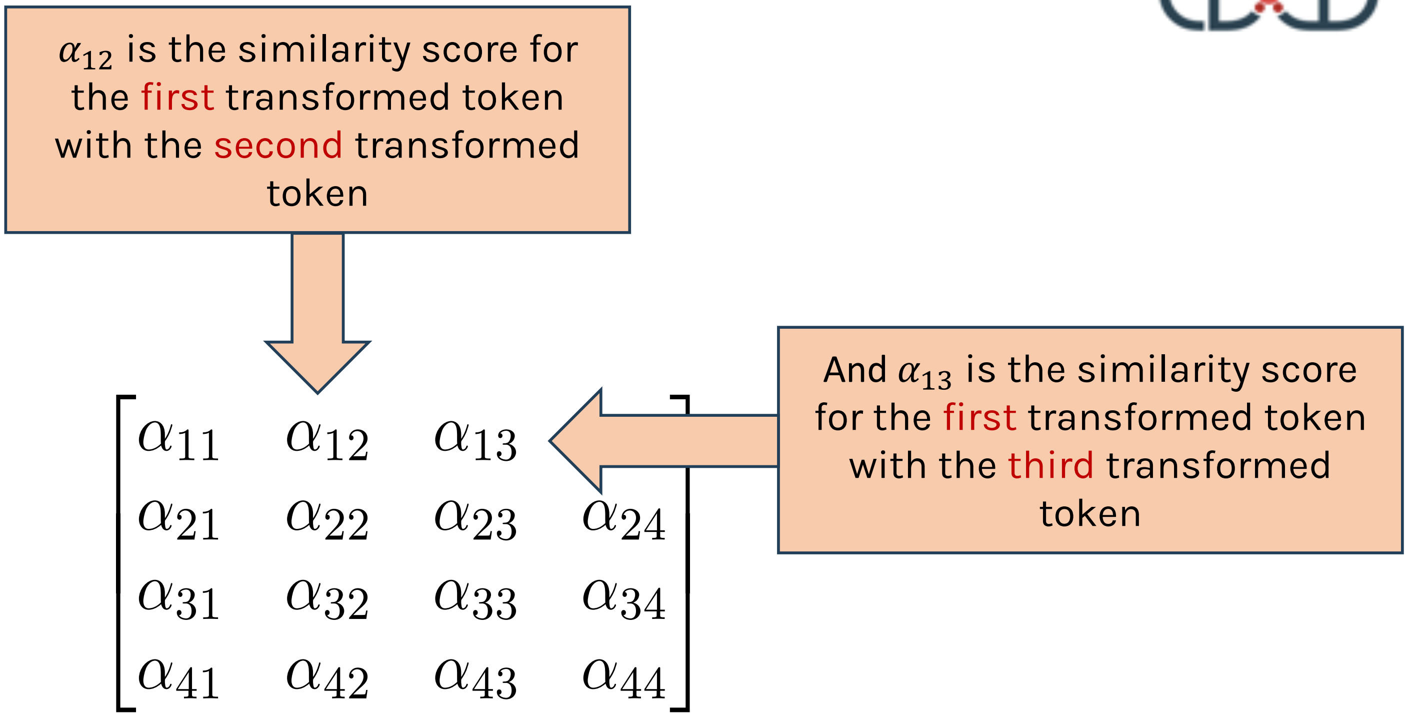
Recall the self-attention block

α_{12} is the similarity score for the first transformed token with the second transformed token

$$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix}$$

And α_{13} is the similarity score for the first transformed token with the third transformed token

Recall the self-attention block



α_{12} is the similarity score for the first transformed token with the second transformed token

And α_{13} is the similarity score for the first transformed token with the third transformed token

$$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix}$$

So the model is looking at the future here in calculating these terms and we need to prevent this

Masked Self-Attention

$$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \alpha_{11} & 0 & 0 & 0 \\ \alpha_{21} & \alpha_{22} & 0 & 0 \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & 0 \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix}$$

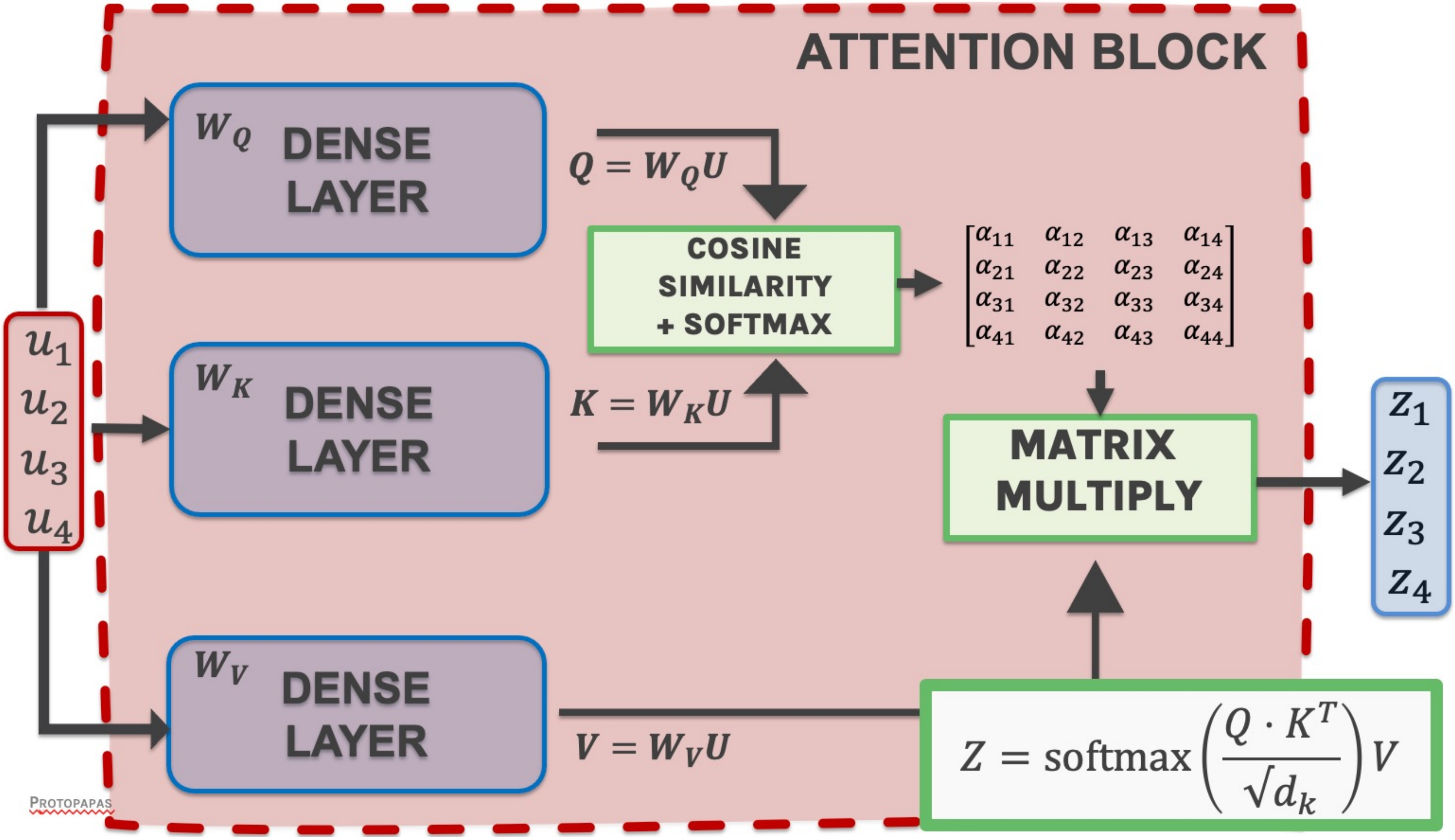
We will also need to modify the softmax function

Masked Self-Attention

So when this multiplied by the values, we get the desired outcome:

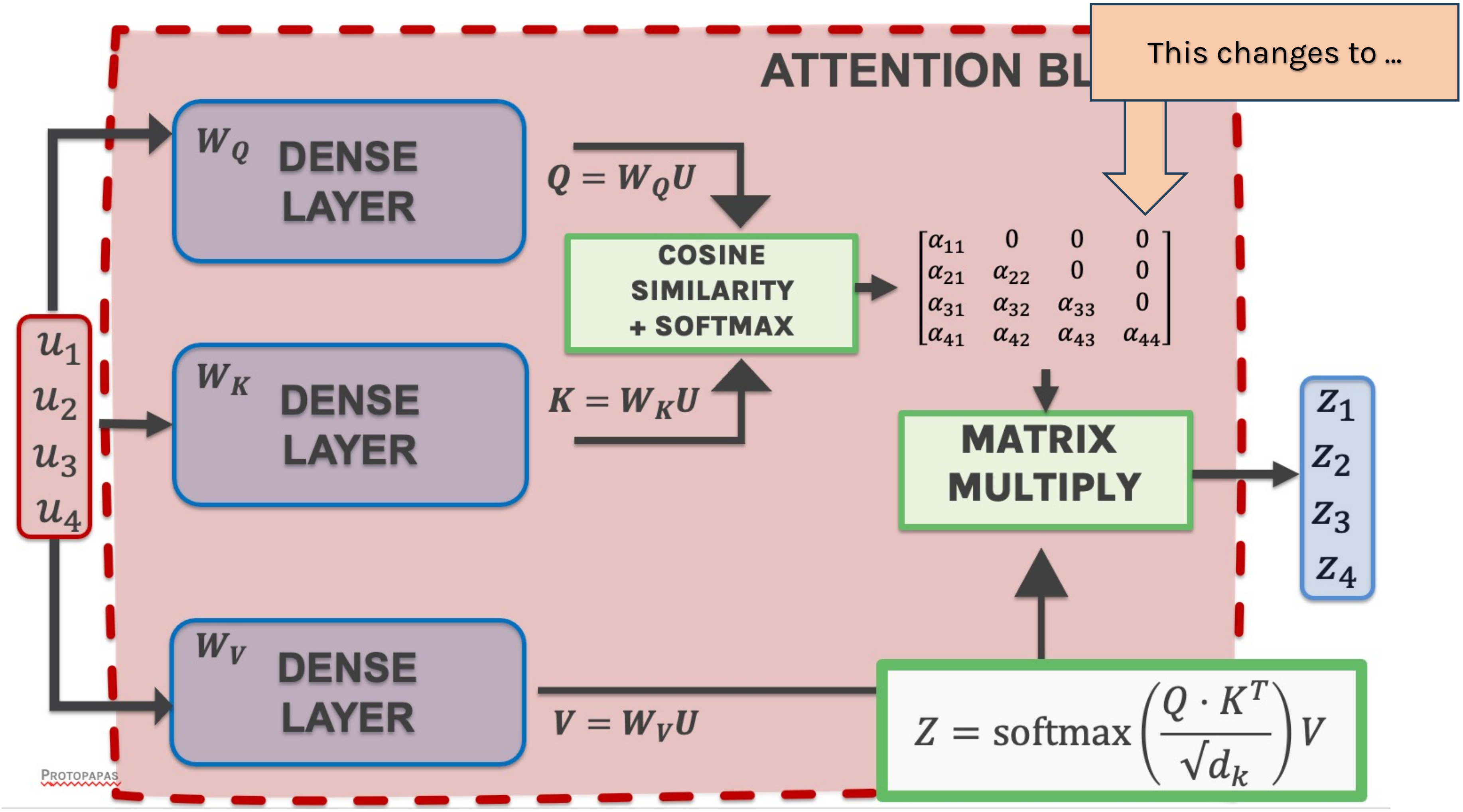
$$\begin{bmatrix}\alpha_{11} & 0 & 0 & 0\\ \alpha_{21} & \alpha_{22} & 0 & 0\\ \alpha_{31} & \alpha_{32} & \alpha_{33} & 0\\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44}\end{bmatrix} \times \begin{bmatrix}V_1\\ V_2\\ V_3\\ V_4\end{bmatrix} = \begin{bmatrix}\alpha_{11}V_1\\ \alpha_{21}V_1 + \alpha_{22}V_2\\ \alpha_{31}V_1 + \alpha_{32}V_2 + \alpha_{33}V_3\\ \alpha_{41}V_1 + \alpha_{42}V_2 + \alpha_{43}V_3 + \alpha_{44}V_4\end{bmatrix}$$

The self-attention block


ATTENTION BLOCK
u_1
u_2
u_3
u_4
W_Q DENSE LAYER
W_K DENSE LAYER
W_V DENSE LAYER
$Q = W_Q U$
$K = W_K U$
$V = W_V U$
COSINE SIMILARITY + SOFTMAX
$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix}$
MATRIX MULTIPLY
z_1
z_2
z_3
z_4
PROTOPAPAS

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

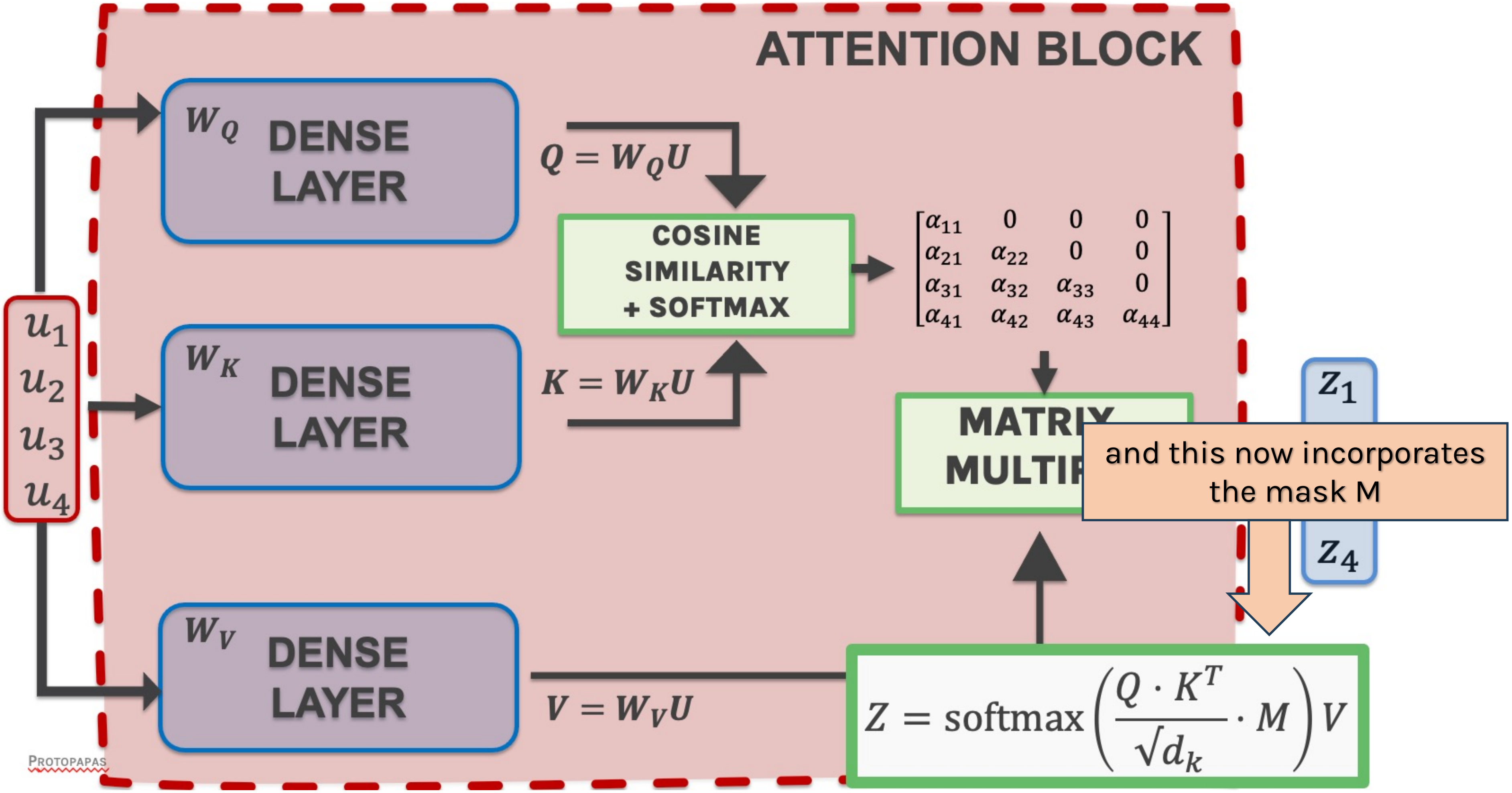
The self-attention block


ATTENTION BL
W_Q DENSE LAYER
W_K DENSE LAYER
W_V DENSE LAYER
$Q = W_Q U$
$K = W_K U$
$V = W_V U$
COSINE SIMILARITY + SOFTMAX
$\begin{bmatrix} \alpha_{11} & 0 & 0 & 0 \\ \alpha_{21} & \alpha_{22} & 0 & 0 \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & 0 \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix}$
MATRIX MULTIPLY
u_1 u_2 u_3 u_4
z_1 z_2 z_3 z_4
PROTOPAPAS

This changes to …

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

The self-attention block

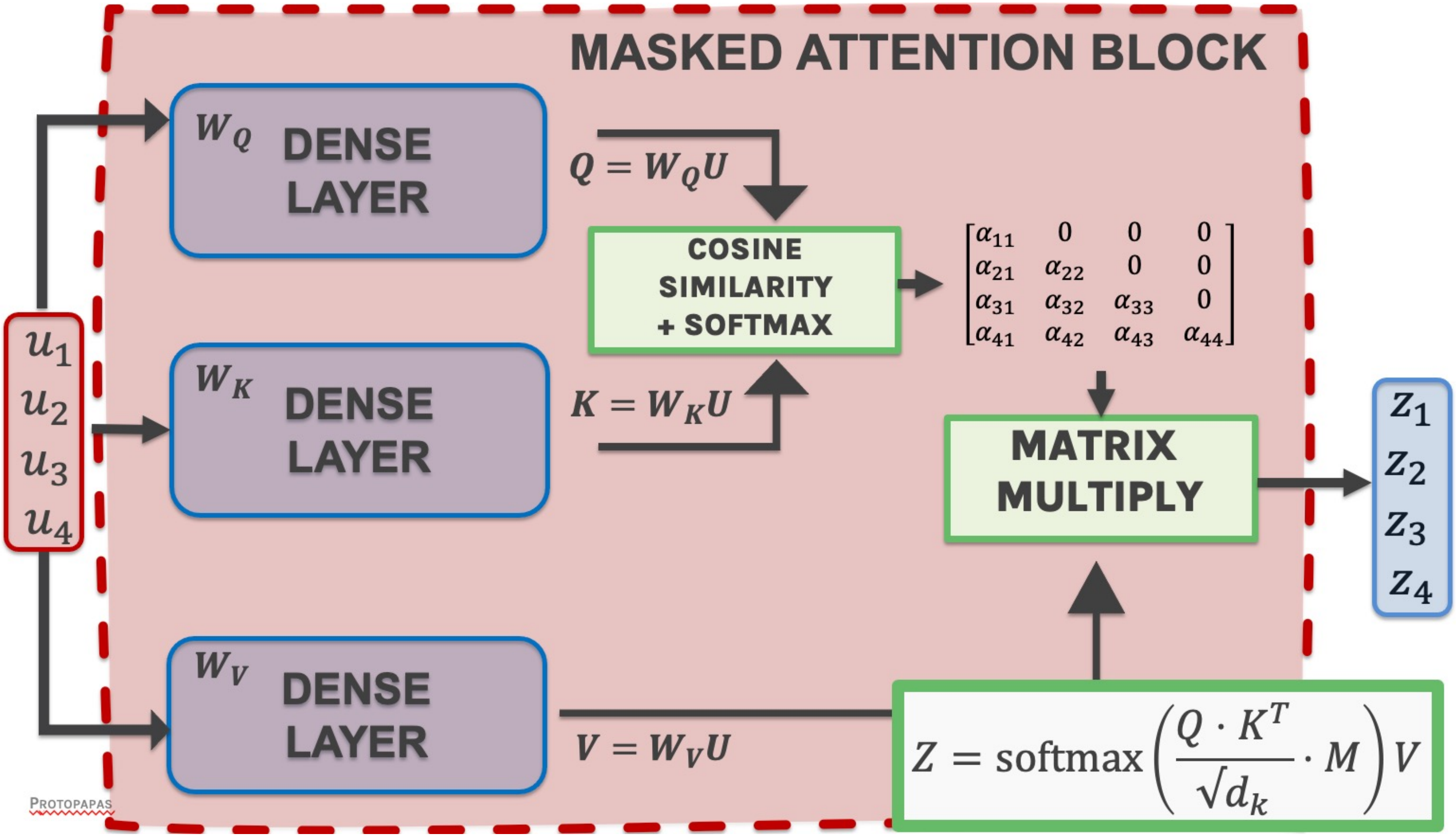


$$\begin{bmatrix} \alpha_{11} & 0 & 0 & 0 \\ \alpha_{21} & \alpha_{22} & 0 & 0 \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & 0 \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix}$$

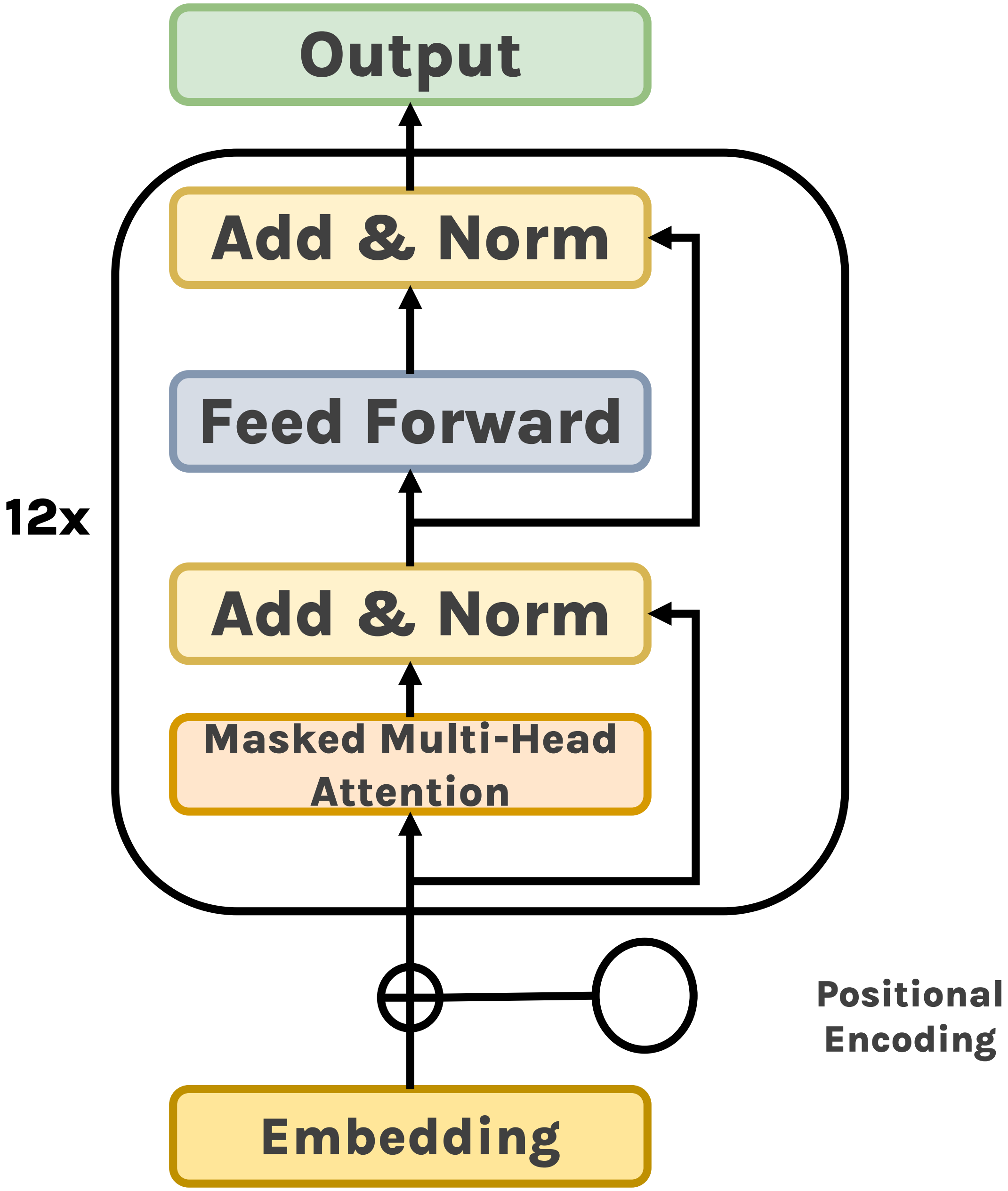
$$Z = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}} \cdot M\right) V$$

and this now incorporates the mask M

The masked self-attention block

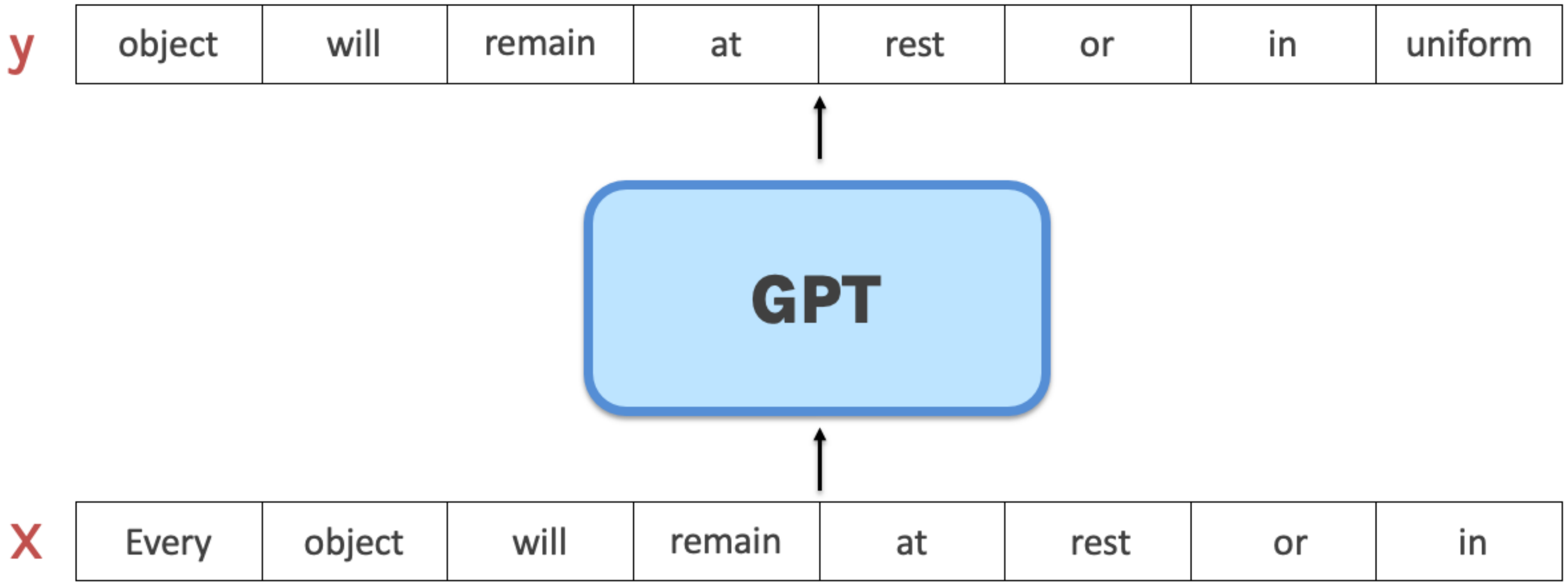

MASKED ATTENTION BLOCK
u_1
u_2
u_3
u_4
W_Q DENSE LAYER
W_K DENSE LAYER
W_V DENSE LAYER
$Q = W_Q U$
$K = W_K U$
$V = W_V U$
COSINE SIMILARITY + SOFTMAX
$\begin{bmatrix} \alpha_{11} & 0 & 0 & 0 \\ \alpha_{21} & \alpha_{22} & 0 & 0 \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & 0 \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix}$
MATRIX MULTIPLY
$Z = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}} \cdot M\right) V$
z_1
z_2
z_3
z_4
PROTOPAPAS

GPT: the complete picture



GPT: the complete picture

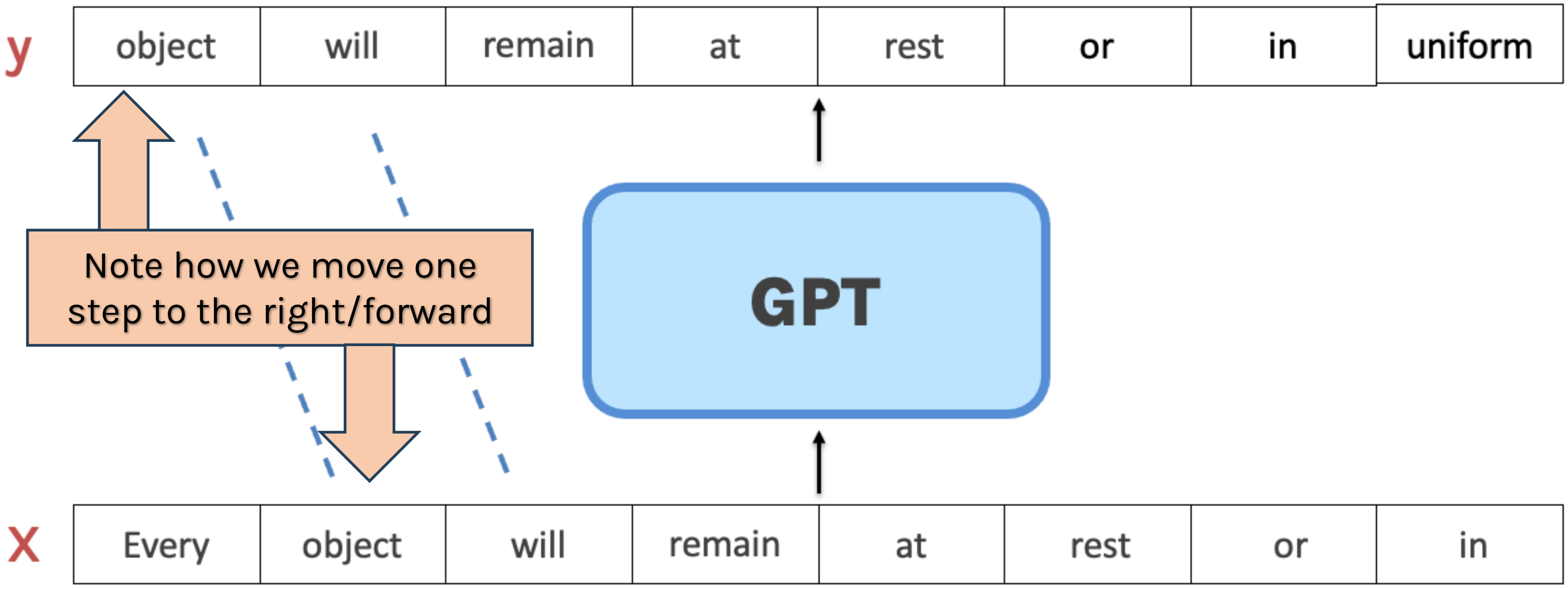
Since we have now ensured that the model cannot extract information from the future, it can be trained just any other LM to do next word prediction:



y	object	will	remain	at	rest	or	in	uniform
x	Every	object	will	remain	at	rest	or	in

GPT: the complete picture

Since we have now ensured that the model cannot extract information from the future, it can be trained just any other LM to do next word prediction:



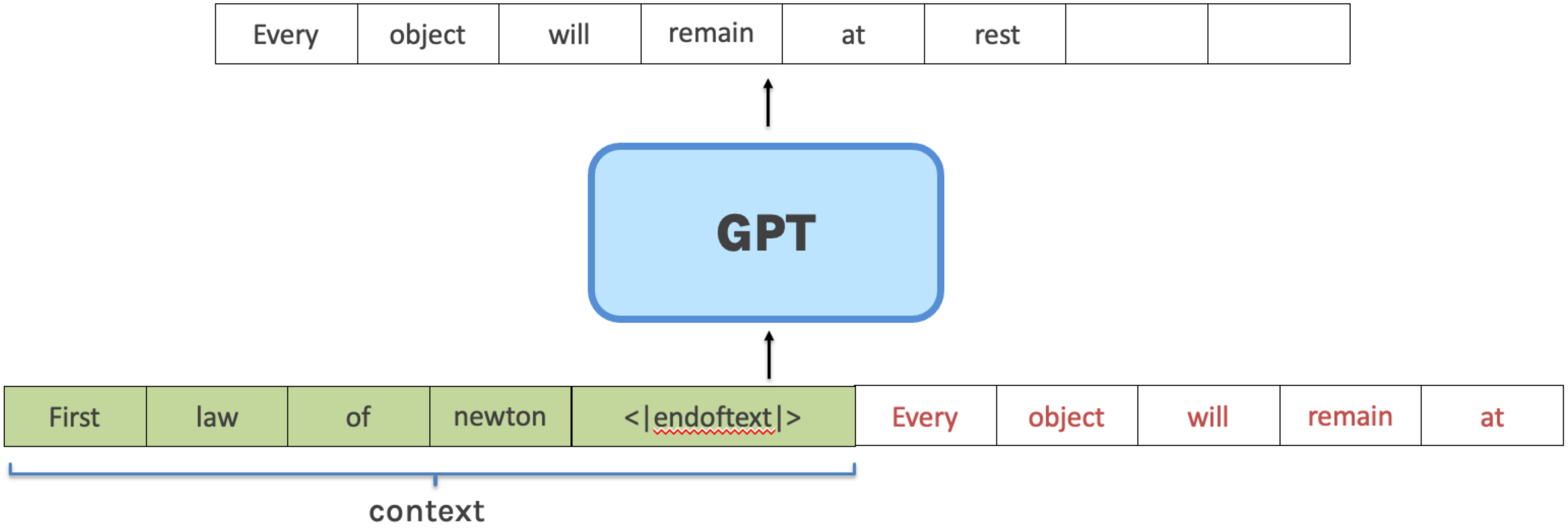
y	object	will	remain	at	rest	or	in	uniform
x	Every	object	will	remain	at	rest	or	in

Note how we move one step to the right/forward

GPT: Inferencing

GPT can generate text in two ways:

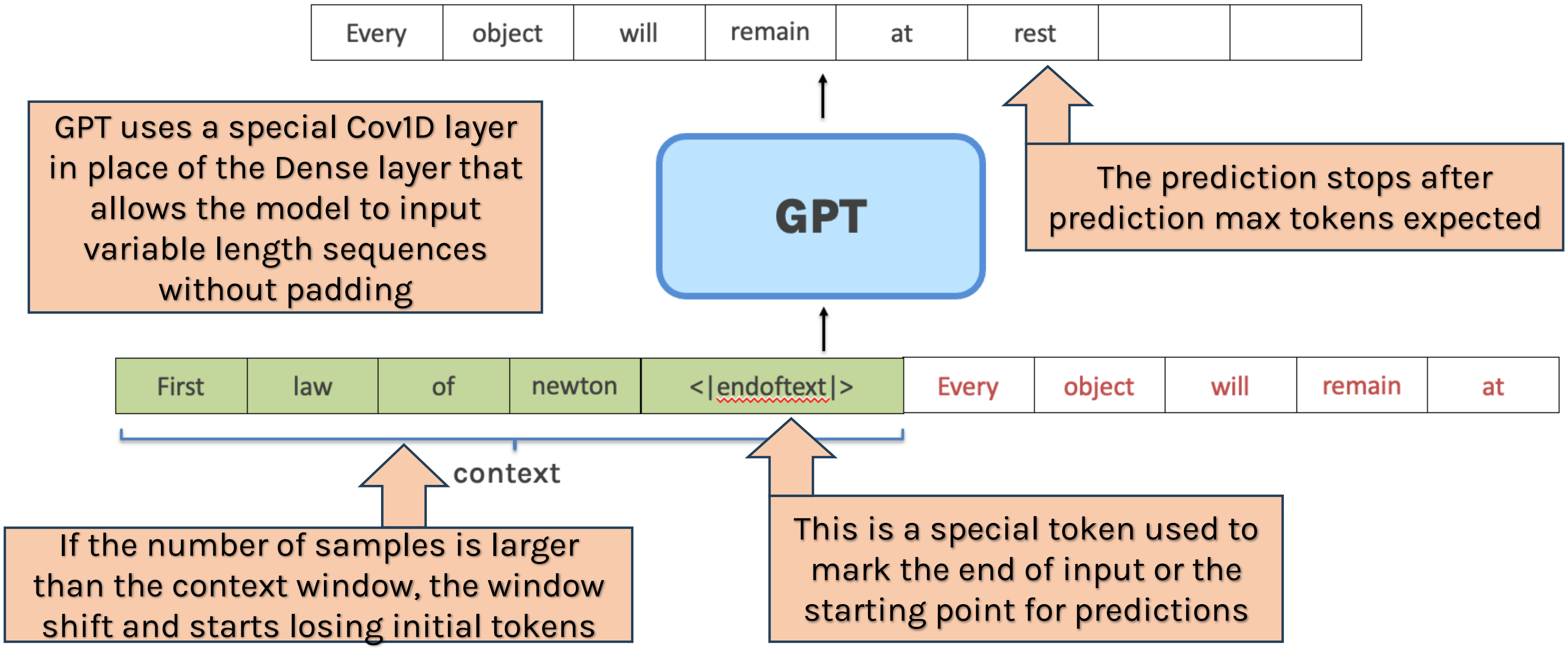
1. Conditional: We provide a starting prompt (context) and the model continues to generate text on that topic.



GPT: Inferencing

GPT can generate text in two ways:

1. Conditional: We provide a starting prompt (context) and the model continues to generate text on that topic.



GPT uses a special Cov1D layer in place of the Dense layer that allows the model to input variable length sequences without padding

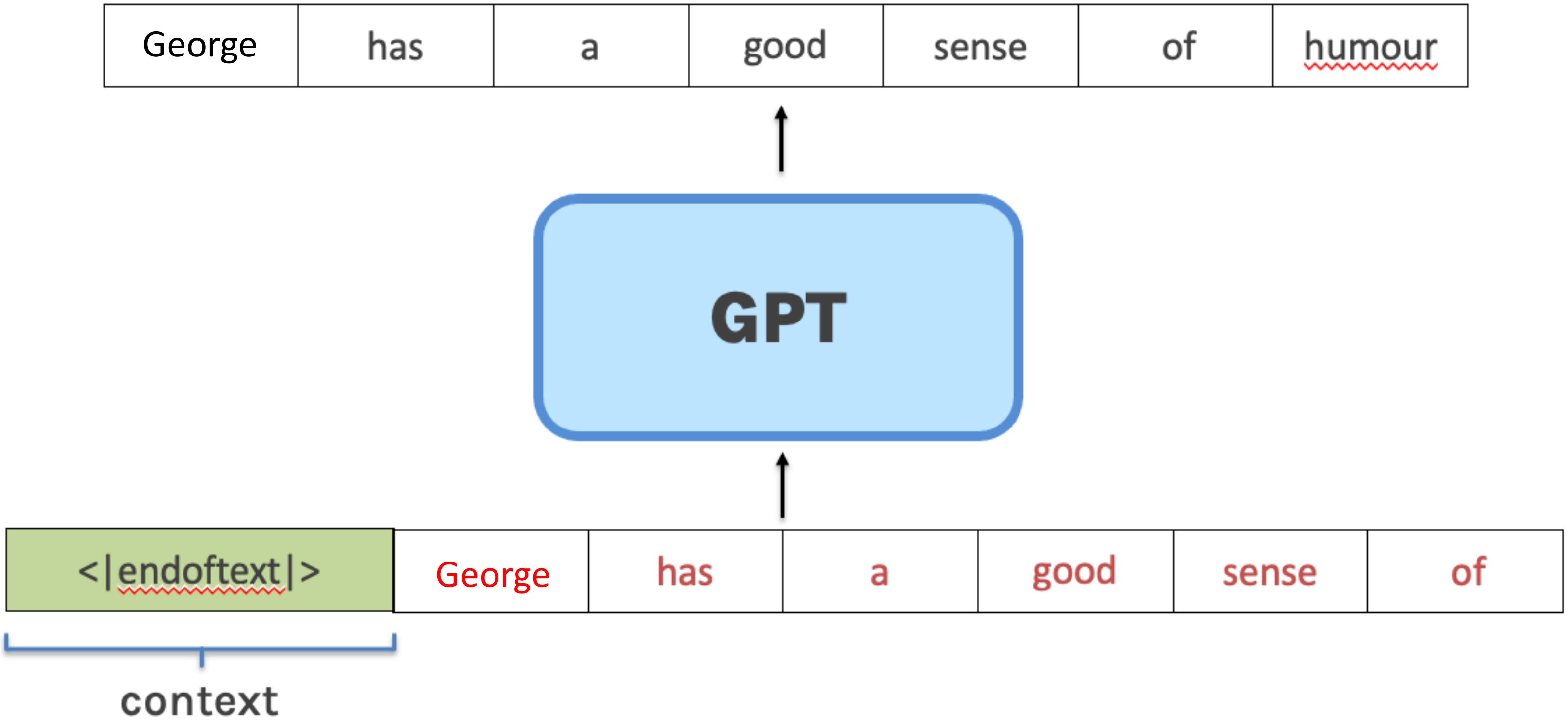
The prediction stops after prediction max tokens expected

If the number of samples is larger than the context window, the window shift and starts losing initial tokens

This is a special token used to mark the end of input or the starting point for predictions

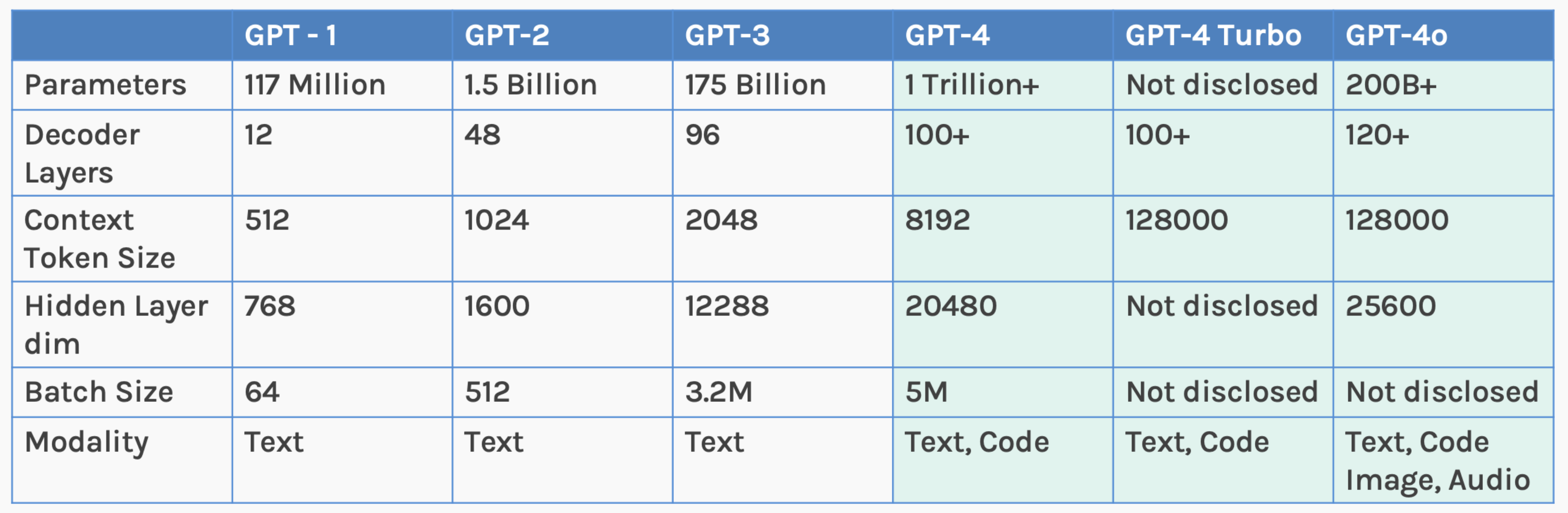
GPT: Inferencing

2. Unconditional: We start with </endoftext/> which is a special start token to have the model generate words on a topic of its own choosing. It draws on the prior distribution over text that is has previously learnt:



Evolution of GPT

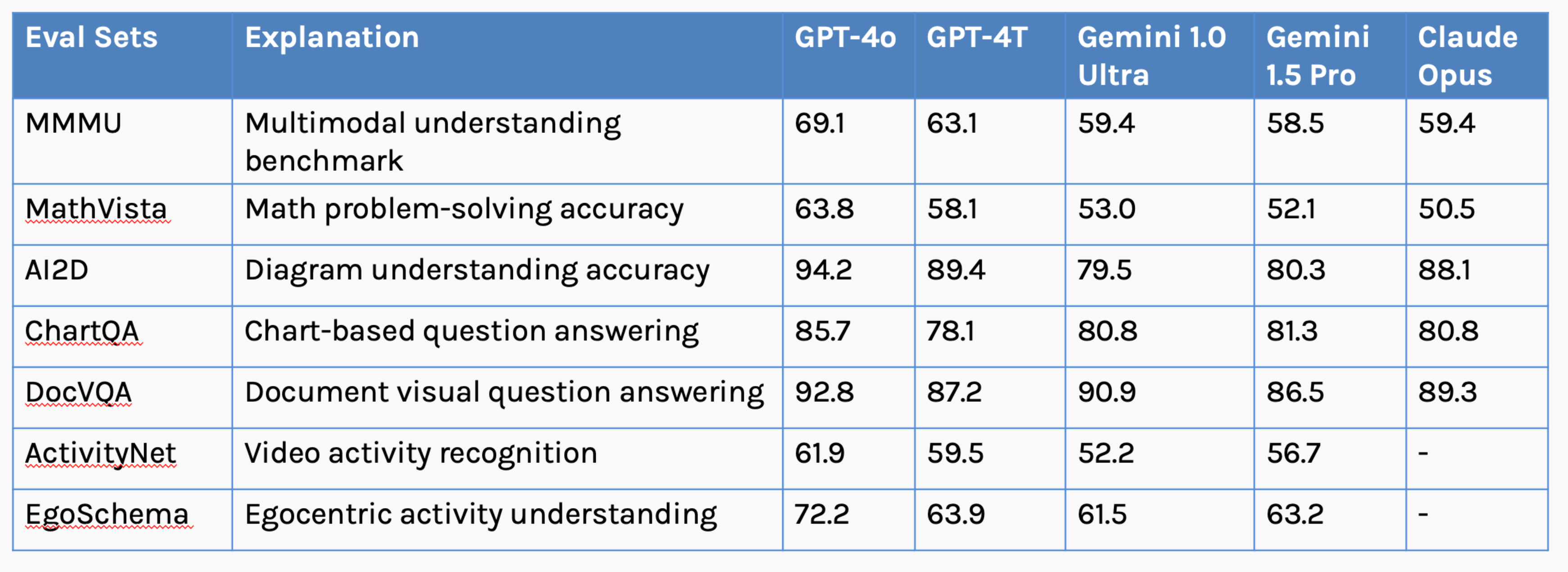
Since its inception in 2018, GPT has received some majpr upgrades in terms of its performance, thanks to the increasing number of parameters with each iteration.



	GPT - 1	GPT-2	GPT-3	GPT-4	GPT-4 Turbo	GPT-4o
Parameters	117 Million	1.5 Billion	175 Billion	1 Trillion+	Not disclosed	200B+
Decoder Layers	12	48	96	100+	100+	120+
Context Token Size	512	1024	2048	8192	128000	128000
Hidden Layer dim	768	1600	12288	20480	Not disclosed	25600
Batch Size	64	512	3.2M	5M	Not disclosed	Not disclosed
Modality	Text	Text	Text	Text, Code	Text, Code	Text, Code Image, Audio

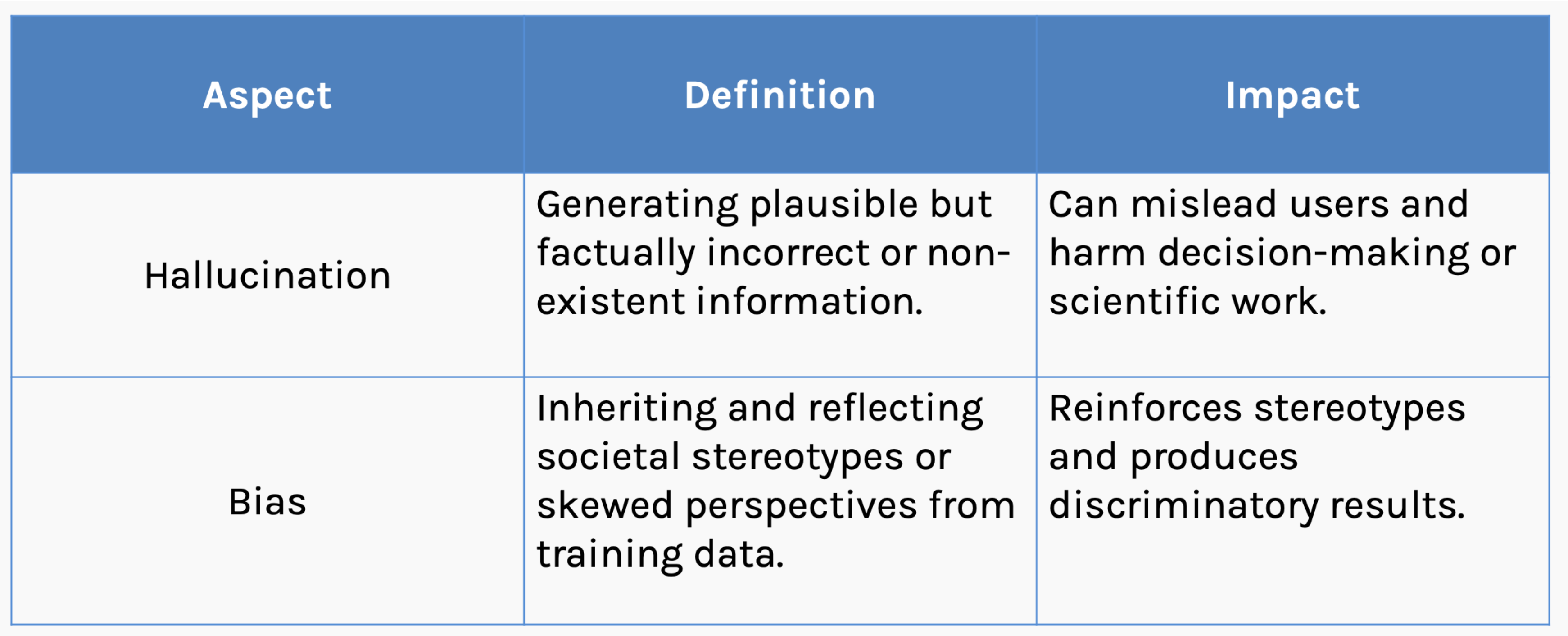
GPT-4 fine-tunes the model with **R**einforcement **L**earning from **H**uman **F**eedback (RLHF)

GPT-4o: Capabilities



Eval Sets	Explanation	GPT-4o	GPT-4T	Gemini 1.0 Ultra	Gemini 1.5 Pro	Claude Opus
MMMU	Multimodal understanding benchmark	69.1	63.1	59.4	58.5	59.4
MathVista	Math problem-solving accuracy	63.8	58.1	53.0	52.1	50.5
AI2D	Diagram understanding accuracy	94.2	89.4	79.5	80.3	88.1
ChartQA	Chart-based question answering	85.7	78.1	80.8	81.3	80.8
DocVQA	Document visual question answering	92.8	87.2	90.9	86.5	89.3
ActivityNet	Video activity recognition	61.9	59.5	52.2	56.7	-
EgoSchema	Egocentric activity understanding	72.2	63.9	61.5	63.2	-

Limitations

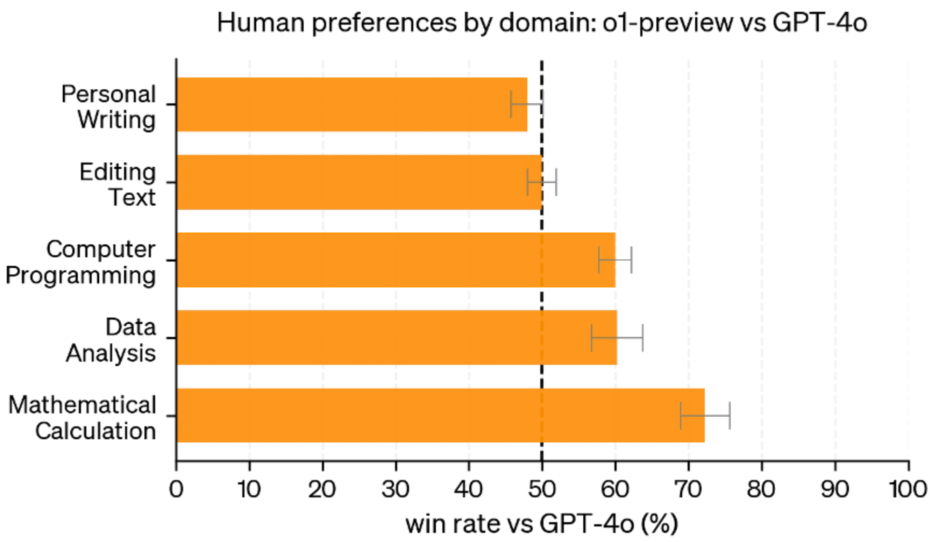


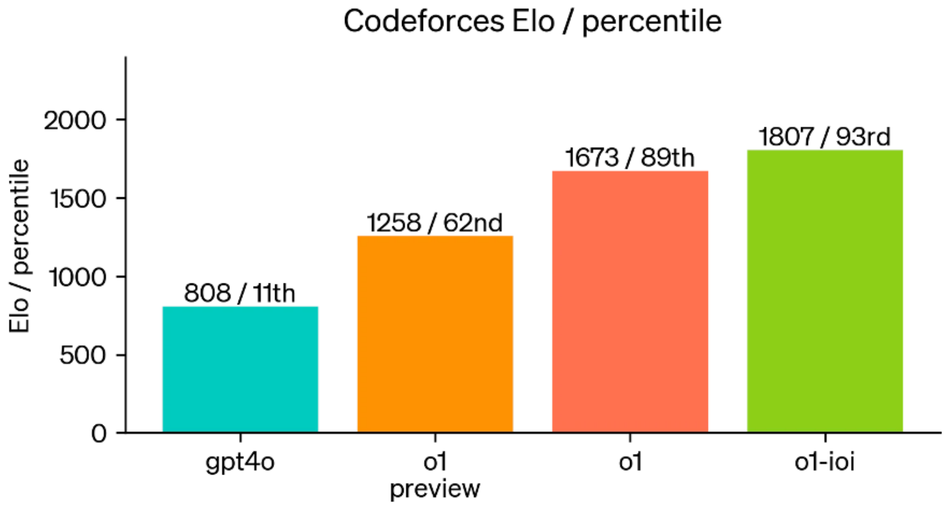
Aspect	Definition	Impact
Hallucination	Generating plausible but factually incorrect or non-existent information.	Can mislead users and harm decision-making or scientific work.
Bias	Inheriting and reflecting societal stereotypes or skewed perspectives from training data.	Reinforces stereotypes and produces discriminatory results.

OpenAI o1

The latest model after GPT-4o is OpenAI o1.

The main difference in the new models is the integration of chain-of-thought in the architecture and training.

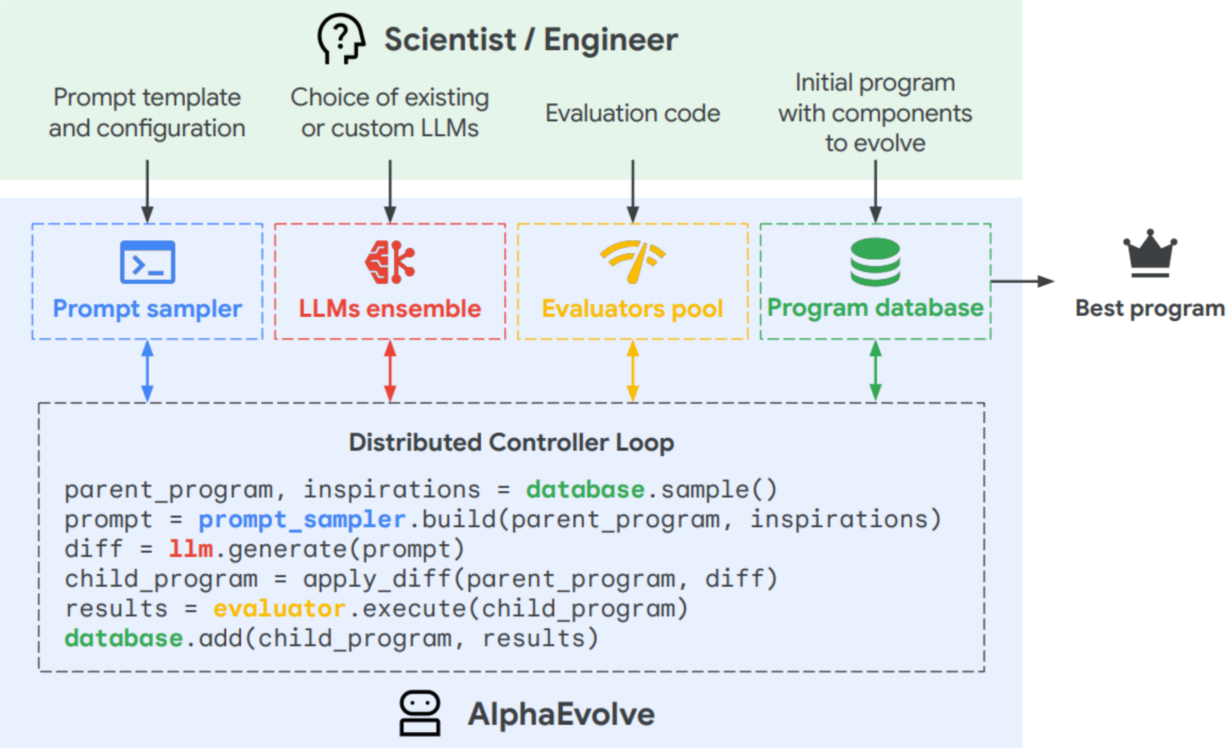




CoT encourages the model to explicitly break problems down into intermediate steps – ”explain its working”. GPT-4o can do this in a multimodal setting

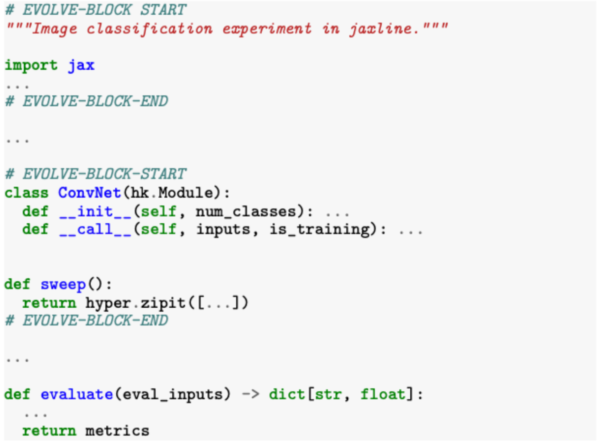
AlphaEvolve

- An evolutionary superoptimization coding agent powered by DeepMind Gemini LLMs
- Best suited for algorithm improvement problems – requires an automated evaluator


Scientist / Engineer
Prompt template and configuration
Choice of existing or custom LLMs
Evaluation code
Initial program with components to evolve
Prompt sampler
LLMs ensemble
Evaluators pool
Program database
Best program
Distributed Controller Loop
AlphaEvolve

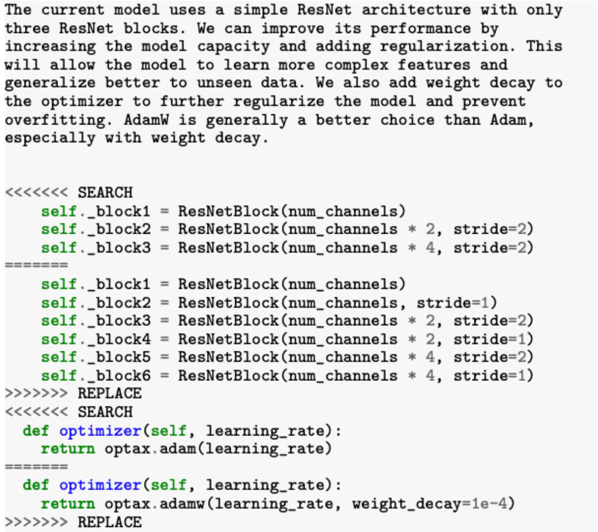
```
parent_program, inspirations = database.sample()
prompt = prompt_sampler.build(parent_program, inspirations)
diff = llm.generate(prompt)
child_program = apply_diff(parent_program, diff)
results = evaluator.execute(child_program)
database.add(child_program, results)
```

AlphaEvolve



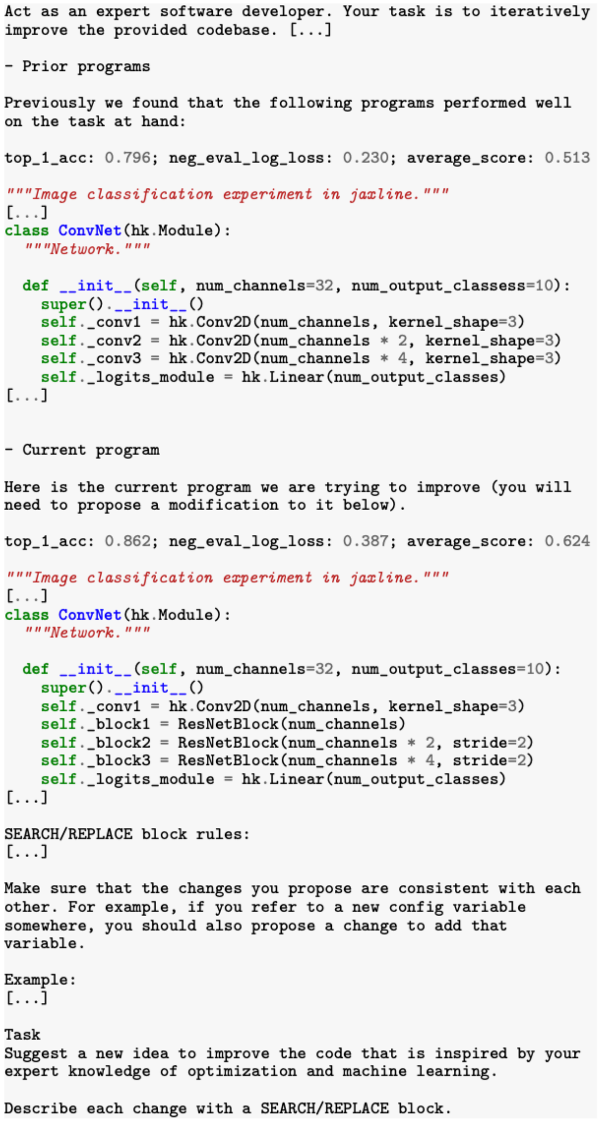
```
# EVOLVE-BLOCK START
"""Image classification experiment in jaxline."""

import jax
...
# EVOLVE-BLOCK-END

...

# EVOLVE-BLOCK-START
class ConvNet(hk.Module):
  def __init__(self, num_classes): ...
  def __call__(self, inputs, is_training): ...


def sweep():
  return hyper.zipit([...])
# EVOLVE-BLOCK-END

...

def evaluate(eval_inputs) -> dict[str, float]:
  ...
  return metrics
```

(a)



```
The current model uses a simple ResNet architecture with only
three ResNet blocks. We can improve its performance by
increasing the model capacity and adding regularization. This
will allow the model to learn more complex features and
generalize better to unseen data. We also add weight decay to
the optimizer to further regularize the model and prevent
overfitting. AdamW is generally a better choice than Adam,
especially with weight decay.


<<<<<<< SEARCH
    self._block1 = ResNetBlock(num_channels)
    self._block2 = ResNetBlock(num_channels * 2, stride=2)
    self._block3 = ResNetBlock(num_channels * 4, stride=2)
=======
    self._block1 = ResNetBlock(num_channels)
    self._block2 = ResNetBlock(num_channels, stride=1)
    self._block3 = ResNetBlock(num_channels * 2, stride=2)
    self._block4 = ResNetBlock(num_channels * 2, stride=1)
    self._block5 = ResNetBlock(num_channels * 4, stride=2)
    self._block6 = ResNetBlock(num_channels * 4, stride=1)
>>>>>>> REPLACE
<<<<<<< SEARCH
  def optimizer(self, learning_rate):
    return optax.adam(learning_rate)
=======
  def optimizer(self, learning_rate):
    return optax.adamw(learning_rate, weight_decay=1e-4)
>>>>>>> REPLACE
```

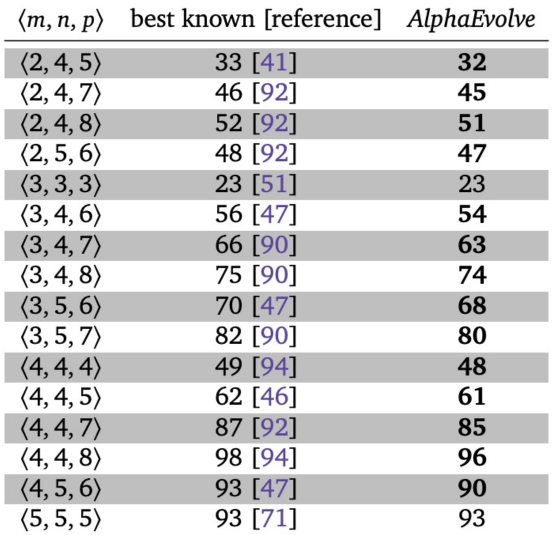
(c)



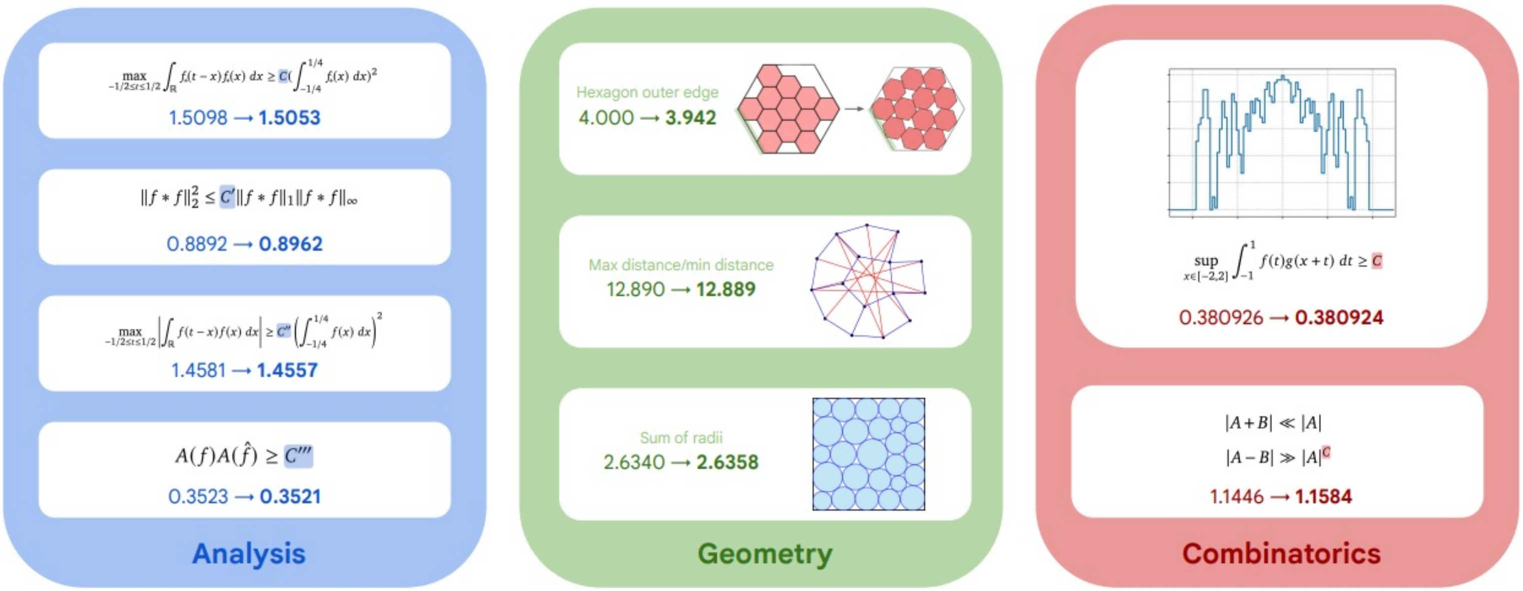
```
Act as an expert software developer. Your task is to iteratively
improve the provided codebase. [...]

- Prior programs

Previously we found that the following programs performed well
on the task at hand:

top_1_acc: 0.796; neg_eval_log_loss: 0.230; average_score: 0.513

"""Image classification experiment in jaxline."""
[...]
class ConvNet(hk.Module):
  """Network."""

  def __init__(self, num_channels=32, num_output_classess=10):
    super().__init__()
    self._conv1 = hk.Conv2D(num_channels, kernel_shape=3)
    self._conv2 = hk.Conv2D(num_channels * 2, kernel_shape=3)
    self._conv3 = hk.Conv2D(num_channels * 4, kernel_shape=3)
    self._logits_module = hk.Linear(num_output_classes)
[...]


- Current program

Here is the current program we are trying to improve (you will
need to propose a modification to it below).

top_1_acc: 0.862; neg_eval_log_loss: 0.387; average_score: 0.624

"""Image classification experiment in jaxline."""
[...]
class ConvNet(hk.Module):
  """Network."""

  def __init__(self, num_channels=32, num_output_classes=10):
    super().__init__()
    self._conv1 = hk.Conv2D(num_channels, kernel_shape=3)
    self._block1 = ResNetBlock(num_channels)
    self._block2 = ResNetBlock(num_channels * 2, stride=2)
    self._block3 = ResNetBlock(num_channels * 4, stride=2)
    self._logits_module = hk.Linear(num_output_classes)
[...]

SEARCH/REPLACE block rules:
[...]

Make sure that the changes you propose are consistent with each
other. For example, if you refer to a new config variable
somewhere, you should also propose a change to add that
variable.

Example:
[...]

Task
Suggest a new idea to improve the code that is inspired by your
expert knowledge of optimization and machine learning.

Describe each change with a SEARCH/REPLACE block.
```

(b)

AlphaEvolve



$\langle m, n, p\rangle$	best known [reference]	*AlphaEvolve*
$\langle 2, 4, 5\rangle$	33 [41]	**32**
$\langle 2, 4, 7\rangle$	46 [92]	**45**
$\langle 2, 4, 8\rangle$	52 [92]	**51**
$\langle 2, 5, 6\rangle$	48 [92]	**47**
$\langle 3, 3, 3\rangle$	23 [51]	23
$\langle 3, 4, 6\rangle$	56 [47]	**54**
$\langle 3, 4, 7\rangle$	66 [90]	**63**
$\langle 3, 4, 8\rangle$	75 [90]	**74**
$\langle 3, 5, 6\rangle$	70 [47]	**68**
$\langle 3, 5, 7\rangle$	82 [90]	**80**
$\langle 4, 4, 4\rangle$	49 [94]	**48**
$\langle 4, 4, 5\rangle$	62 [46]	**61**
$\langle 4, 4, 7\rangle$	87 [92]	**85**
$\langle 4, 4, 8\rangle$	98 [94]	**96**
$\langle 4, 5, 6\rangle$	93 [47]	**90**
$\langle 5, 5, 5\rangle$	93 [71]	93



$$\max_{-1/2\le t\le 1/2}\int_{\mathbb{R}} f_t(t-x)f_t(x)\,dx \ge C\left(\int_{-1/4}^{1/4} f_t(x)\,dx\right)^2$$

1.5098 → **1.5053**

$$\|f*f\|_2^2 \le C'\|f*f\|_1\|f*f\|_\infty$$

0.8892 → **0.8962**

$$\max_{-1/2\le t\le 1/2}\left|\int_{\mathbb{R}} f(t-x)f(x)\,dx\right| \ge C''\left(\int_{-1/4}^{1/4} f(x)\,dx\right)^2$$

1.4581 → **1.4557**

$$A(f)A(\hat{f}) \ge C'''$$

0.3523 → **0.3521**

Analysis

Hexagon outer edge

4.000 → **3.942**

Max distance/min distance

12.890 → **12.889**

Sum of radii

2.6340 → **2.6358**

Geometry

$$\sup_{x\in[-2,2]}\int_{-1}^{1} f(t)g(x+t)\,dt \ge C$$

0.380926 → **0.380924**

$$|A+B| \ll |A|$$

$$|A-B| \gg |A|^C$$

1.1446 → **1.1584**

Combinatorics