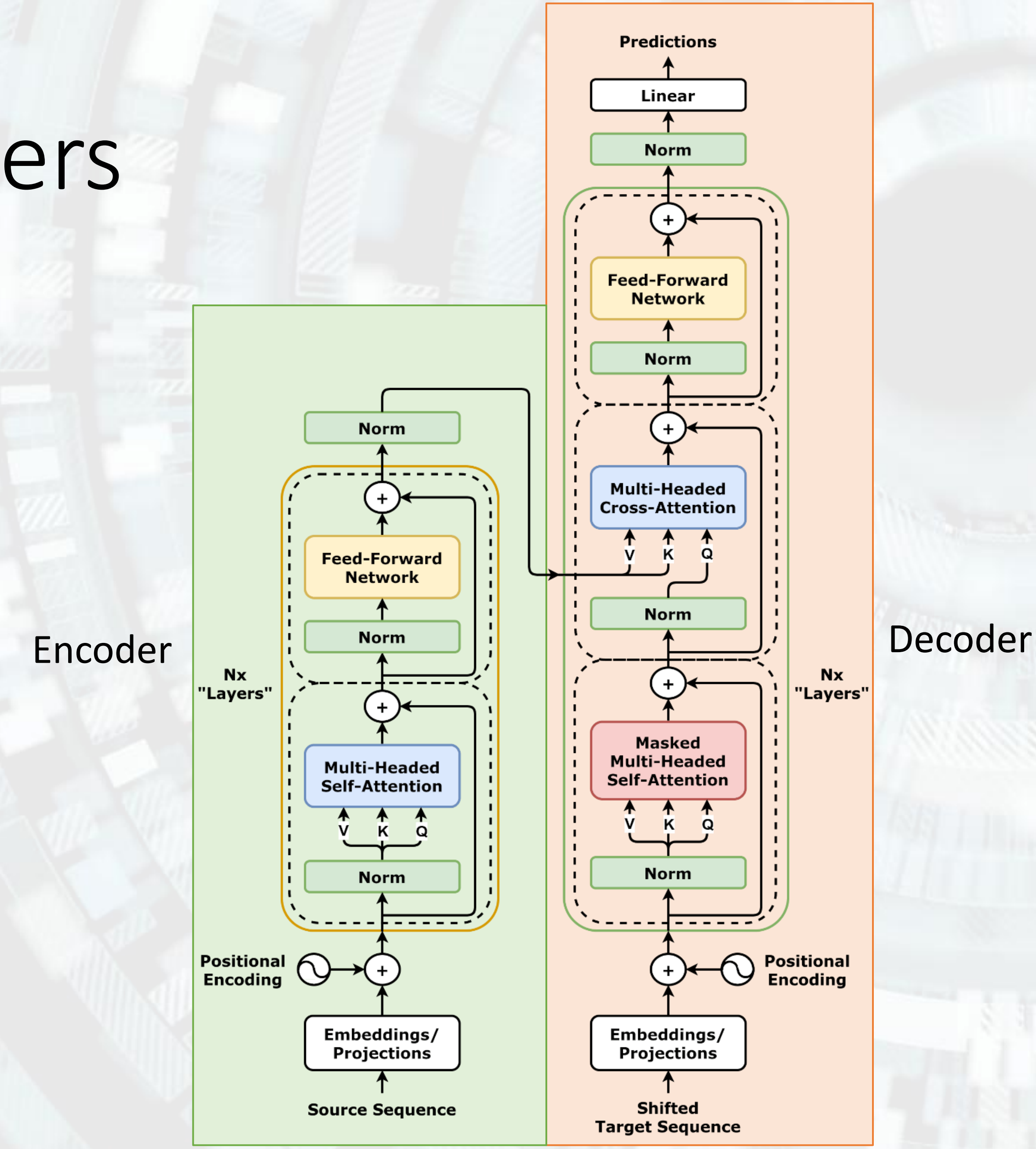
Deep Learning Fundamentals

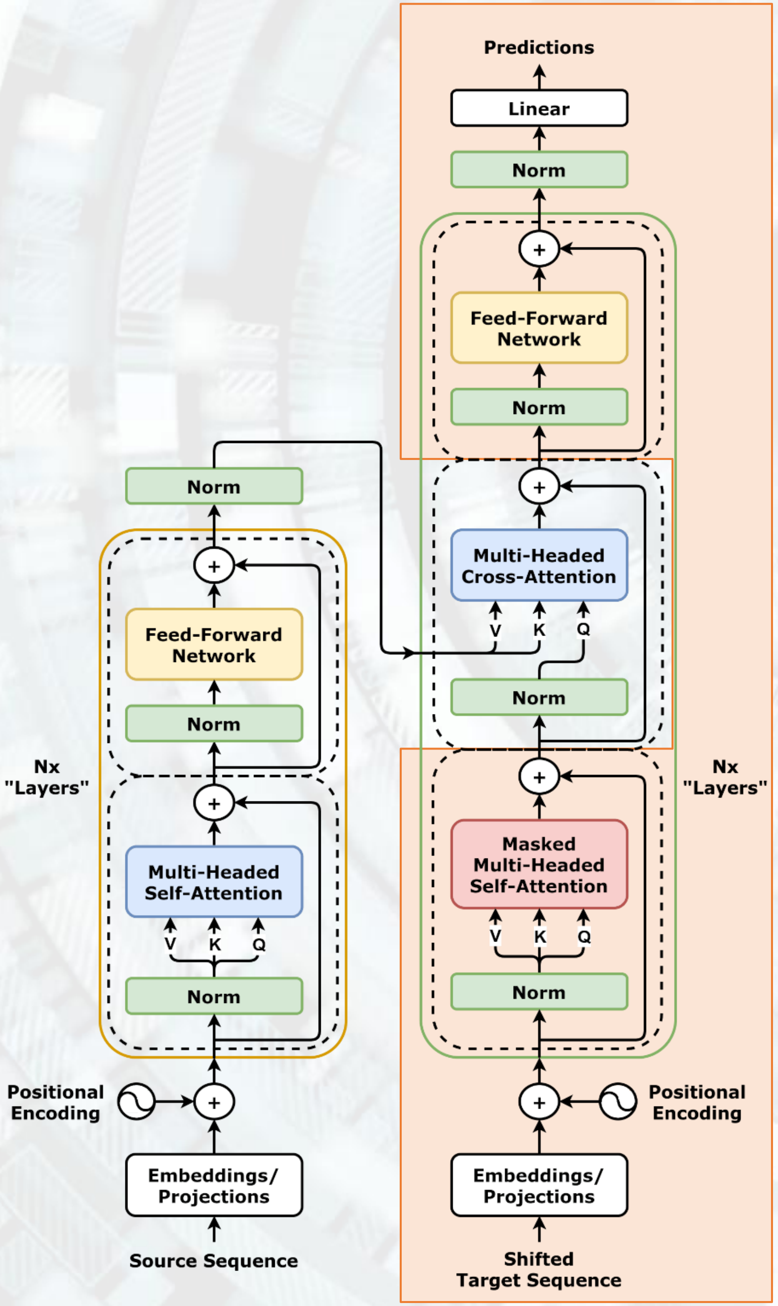
Mihai DOGARIU

www.mdogariu.aimultimedialab.ro

Transformers



GPT


Predictions
Linear
Norm
Feed-Forward Network
Norm
Multi-Headed Cross-Attention
V K Q
Norm
Masked Multi-Headed Self-Attention
V K Q
Norm
Nx "Layers"
Positional Encoding
Embeddings/ Projections
Shifted Target Sequence
Norm
Feed-Forward Network
Norm
Multi-Headed Self-Attention
V K Q
Norm
Nx "Layers"
Positional Encoding
Embeddings/ Projections
Source Sequence

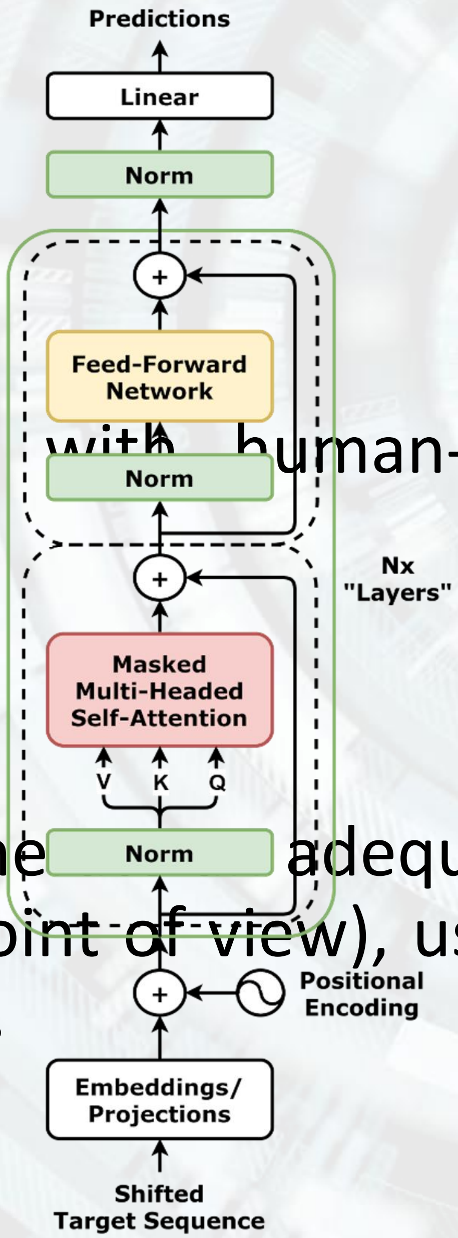
GPT

Goal (general):

- Solve language problems, with human-like performance.

Goal (specific):

- For a given text prompt, gene adequate response (from a semantic point of view), using a large language model (LLM).



GPT

- GPT = Generative Pre-trained Transformer
  - Is able to create new, arbitrarily long, text sequences;
  - It is originally trained on a very large text corpus (estimated ~10 trillion words for GPT-4);
  - It understands context through attention.
- Most popular Large Language Model (LLM)
  - Large:
    - 1.7-1.8 trillion parameters (GPT-4)
    - 175 billion parameters (GPT-3)

GPT

- Input – sequence of arbitrary length:

  Unexpectedly, the boy entered

- Output – a sequence of words that will continue the input sequence or respond to it:

  Unexpectedly, the boy entered the room to look at the clock.

GPT – recursive generation

- Input – sequence of arbitrary length:
  - Unexpectedly, the boy entered

- Output – a character/symbol/word/**token** that will continue the input sequence:
  - Unexpectedly, the boy entered the

GPT – recursive generation

- Input – the previous sequence, appended with the generated token:
  - Unexpectedly, the boy entered the

- Output – a character/symbol/word/**token** that will continue the input sequence:
  - Unexpectedly, the boy entered the room

GPT – recursive generation

- Input – the previous sequence, appended with the generated token:
  - Unexpectedly, the boy entered the room

- Output – a character/symbol/word/**token** that will continue the input sequence:
  - Unexpectedly, the boy entered the room to

GPT – recursive generation

- Input – the previous sequence, appended with the generated token:

  Unexpectedly, the boy entered the room to

- Output – a character/symbol/word/**token** that will continue the input sequence:

  Unexpectedly, the boy entered the room to look

GPT – recursive generation

- Input – the previous sequence, appended with the generated token:
  - Unexpectedly, the boy entered the room to look
- Output – a character/symbol/word/**token** that will continue the input sequence:
  - Unexpectedly, the boy entered the room to look at

GPT – recursive generation

- Input – the previous sequence, appended with the generated token:
  - Unexpectedly, the boy entered the room to look at

- Output – a character/symbol/word/**token** that will continue the input sequence:
  - Unexpectedly, the boy entered the room to look at the

GPT – recursive generation

- Input – the previous sequence, appended with the generated token:
  - Unexpectedly, the boy entered the room to look at the

- Output – a character/symbol/word/**token** that will continue the input sequence:
  - Unexpectedly, the boy entered the room to look at the clock

GPT – recursive generation

- Input – the previous sequence, appended with the generated token:

  Unexpectedly, the boy entered the room to look at the clock

- Output – a character/symbol/word/**token** that will continue the input sequence:

  Unexpectedly, the boy entered the room to look at the clock.

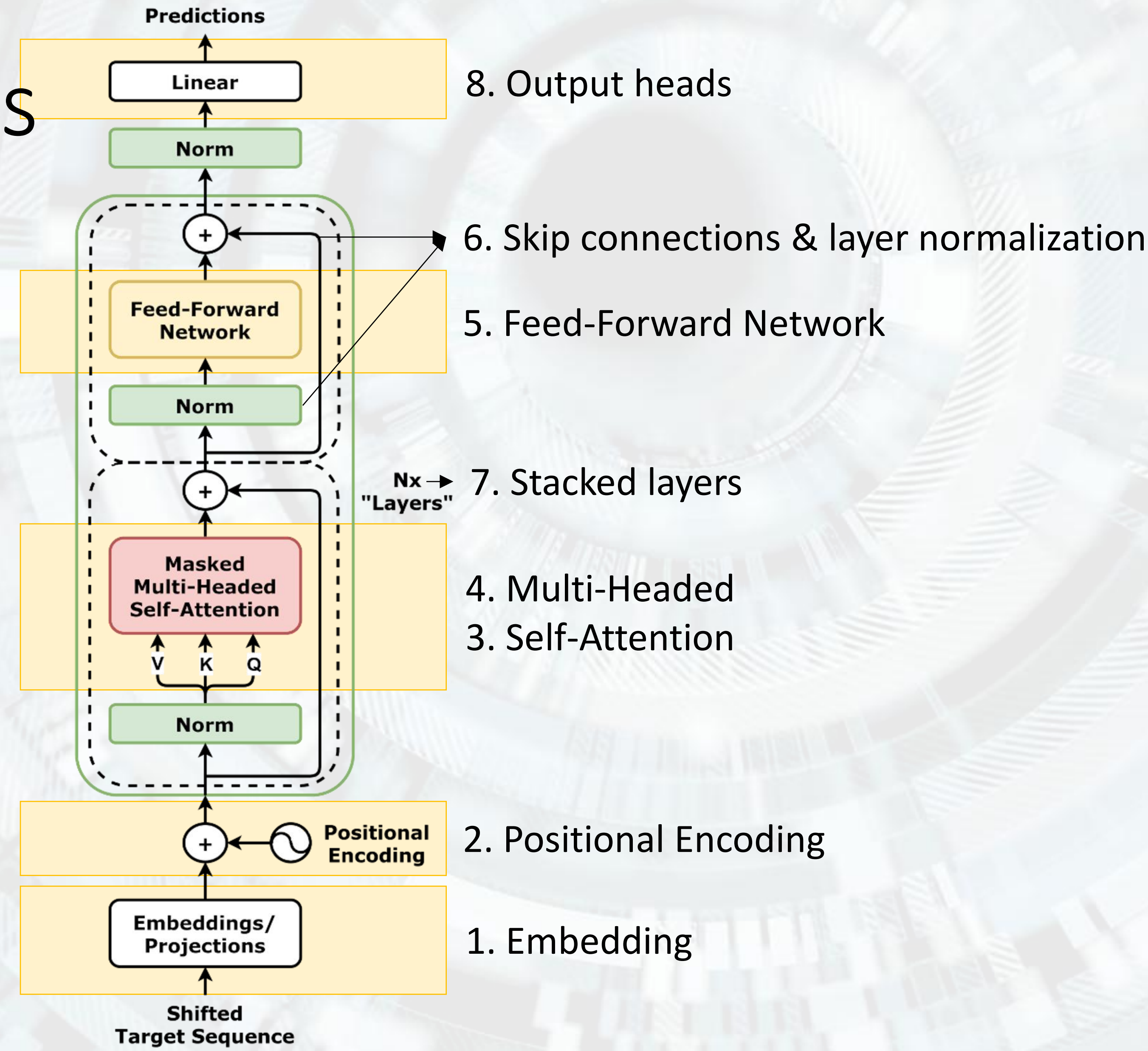
GPT – recursive generation

- Input – the previous sequence, appended with the generated token:
  - Unexpectedly, the boy entered the room to look at the clock.

- Output – a character/symbol/word/**token** that will continue the input sequence:
  - Unexpectedly, the boy entered the room to look at the clock. <END>

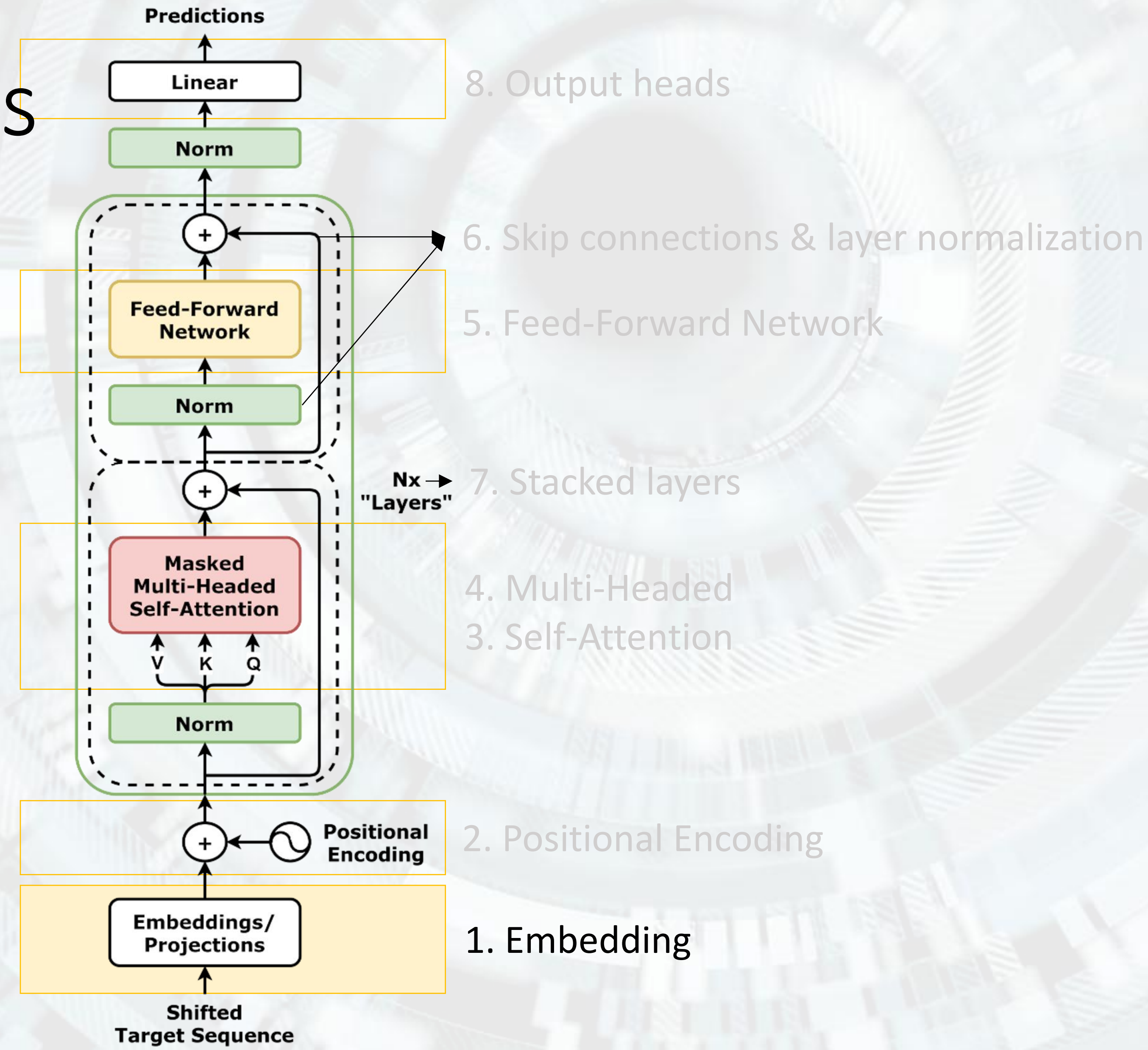
GPT - components

Step	Purpose
1. Input Embedding	Encode word meaning
2. Positional Encoding	Add word order information
3. Self-Attention	Learn contextual relationships
4. Multi-Head Attention	Capture different attention types
5. Feed-Forward Network	Non-linear transformation
6. Residual + LayerNorm	Stability and gradient flow
7. Stacking Layers	Increase model capacity
8. Output Layer	Predict next token / classification



GPT - components

Step	Purpose
1. Input Embedding	Encode word meaning
2. Positional Encoding	Add word order information
3. Self-Attention	Learn contextual relationships
4. Multi-Head Attention	Capture different attention types
5. Feed-Forward Network	Non-linear transformation
6. Residual + LayerNorm	Stability and gradient flow
7. Stacking Layers	Increase model capacity
8. Output Layer	Predict next token / classification



1. Embedding

Turns a text input sequence into numerical values. Consists of 2 processes: 1) tokenization and 2) token embedding.

1) Input sequence is divided into tokens (parts of words) using Byte Pair Encoding (BPE):

Unexpectedly, the boy entered

Tokens are represented with their position number in the token vocabulary (size: 50 257 tokens)

Token	Id
U	52
nexpected	42072
ly	306
,	11
the	262
boy	2933
entered	5982

1. Embedding

2) Token embedding: replace each token with its corresponding embedding vector from the embedding matrix (size: 768 values):

Token	Id
U	52
nexpected	42072
ly	306
,	11
the	262
boy	2933
entered	5982

D = 768 (embedding size)

aah	aardwark	aardwolf	aargh	ab	aback	abacterial	...	zygosis	zygote	zygotic	zyme	zymogen	zymosis	zzz
0.98	4.30	2.06	0.90	-1.53	2.92	-1.25	...	9.27	-2.33	5.83	0.58	1.36	8.51	-8.58
-8.26	-9.60	6.65	5.56	7.40	9.57	5.98	...	5.61	-7.63	2.80	-7.13	8.89	0.44	-1.71
-4.71	5.48	-0.88	1.37	-9.62	2.35	2.24	...	8.87	3.64	-2.81	-1.26	3.95	-8.80	3.34
3.41	-5.79	-7.42	-3.69	-2.73	1.40	-1.23	...	-7.96	-5.82	-6.77	3.06	-4.93	-0.67	-5.11
-6.82	-7.79	3.13	-7.24	-6.07	-2.63	6.42	...	6.76	-8.08	9.53	-0.63	9.54	2.10	4.79
-9.22	-4.34	-7.60	-4.08	-7.63	-3.64	-1.71	...	3.85	1.33	-4.69	0.46	-8.12	1.52	8.59
-3.63	3.35	-7.36	4.33	-4.21	-6.34	1.73	...	6.58	-9.91	3.56	-4.60	4.70	9.24	-5.02
1.52	1.84	1.45	-5.54	9.05	-1.06	6.93	...	-4.05	6.28	-2.07	7.62	1.63	7.63	3.85
:	:	:	:	:	:	:	:	:	:	:	:	:	:	:
1.82	1.49	3.06	3.04	-1.37	7.93	-2.65	-1.28	7.84	6.12	4.08	-8.00	8.39	4.28	9.98

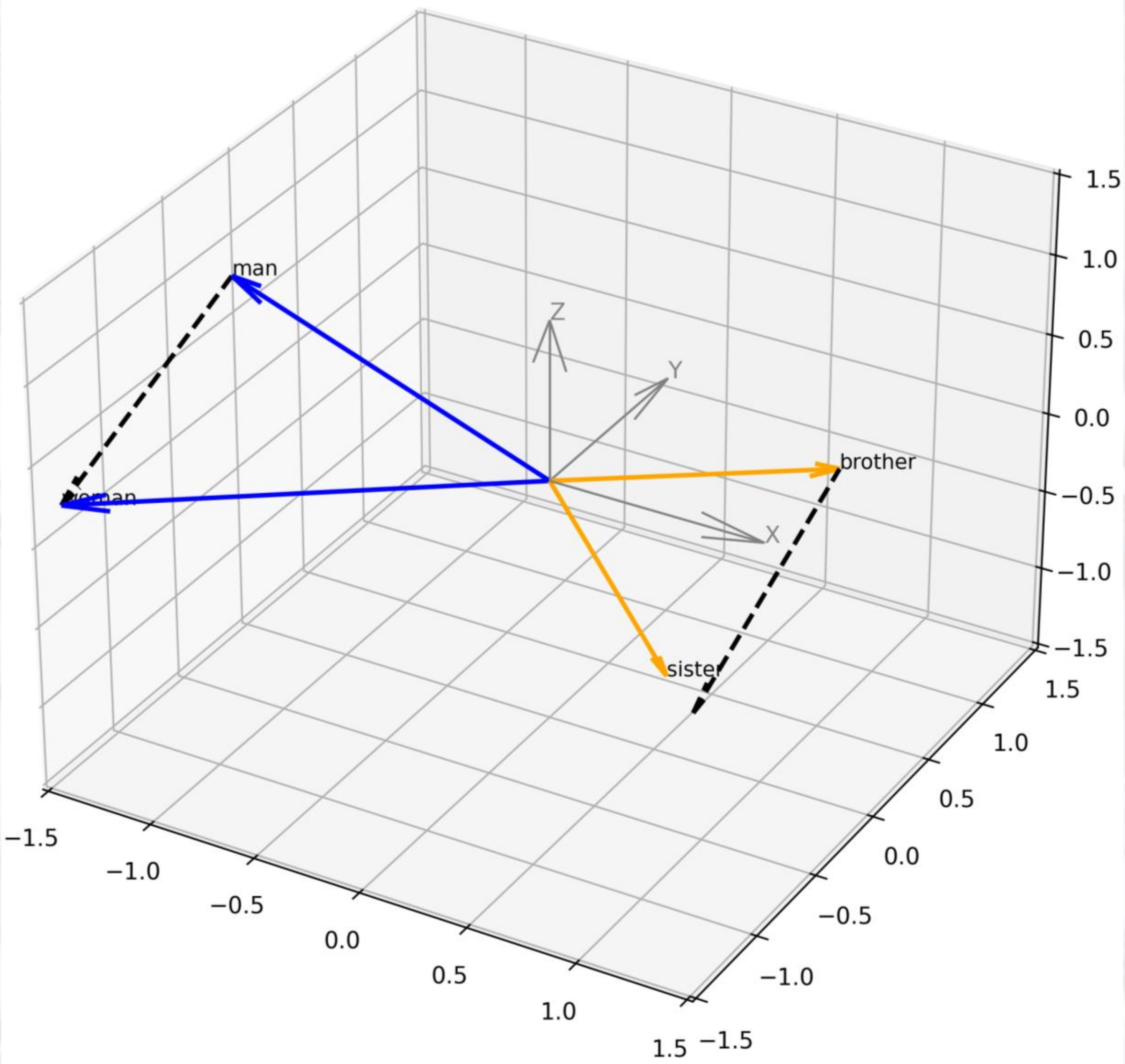
V = 50,257 (vocab size)

1. Embedding

Embeddings are a high dimensional representation of the token’s semantic meaning. They are learned and adjusted throughout training.

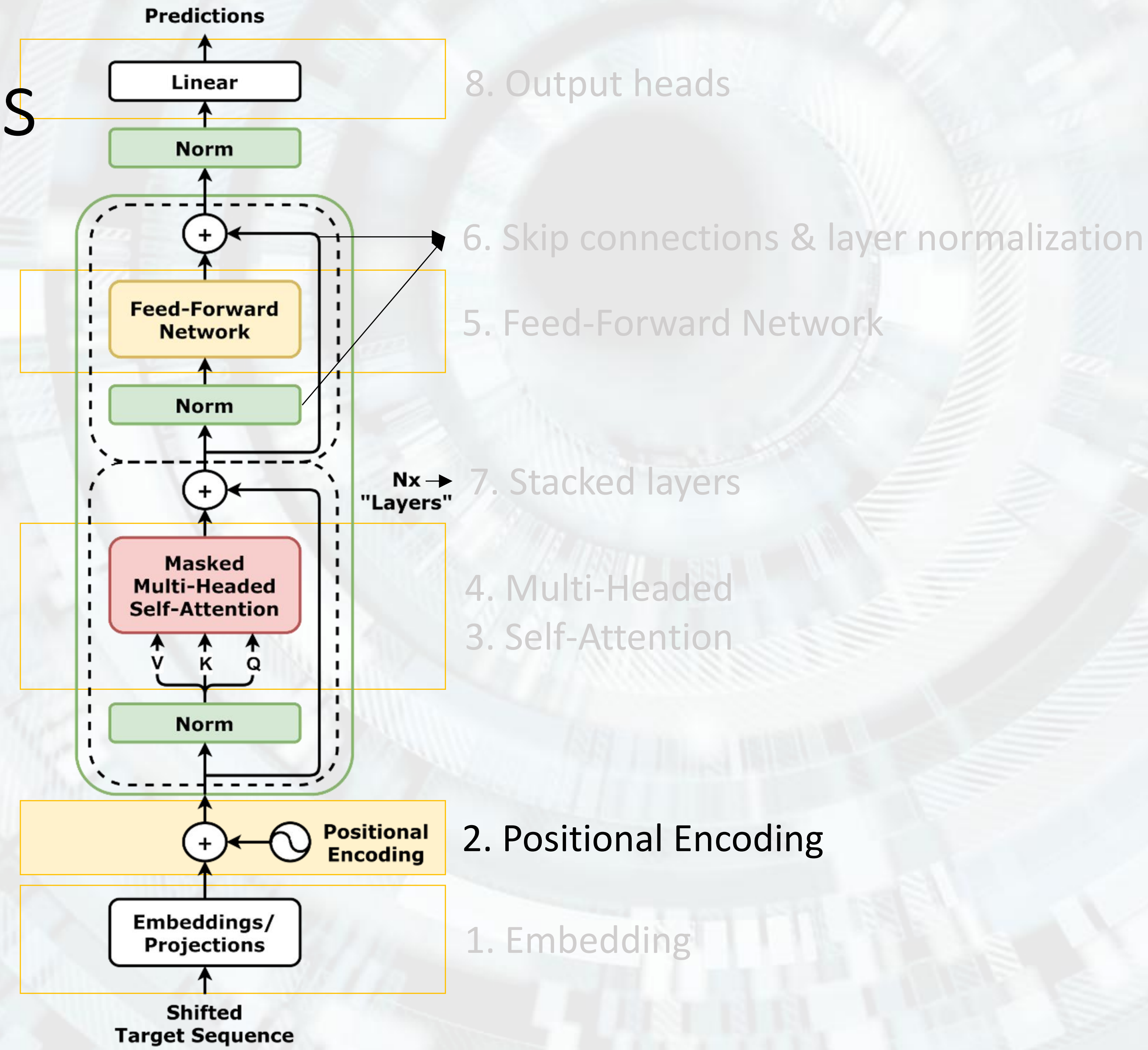
Semantic analogy:

[woman] – [man] + [brother] = [sister]



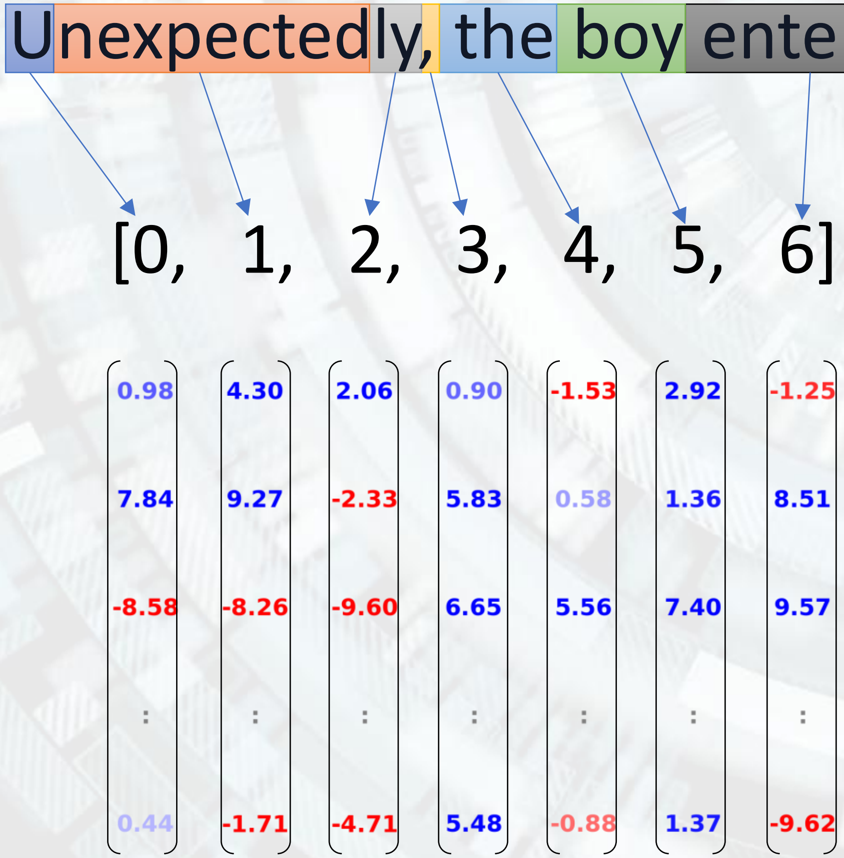
GPT - components

Step	Purpose
1. Input Embedding	Encode word meaning
2. Positional Encoding	Add word order information
3. Self-Attention	Learn contextual relationships
4. Multi-Head Attention	Capture different attention types
5. Feed-Forward Network	Non-linear transformation
6. Residual + LayerNorm	Stability and gradient flow
7. Stacking Layers	Increase model capacity
8. Output Layer	Predict next token / classification



2. Positional Encoding

Each token’s position in the sequence is encoded in a D = 768 embedding vector. The result is added with the token embedding.



Embedding size: 7 x 768

2. Positional Encoding

Is used to track the token’s positioning inside the sequence.

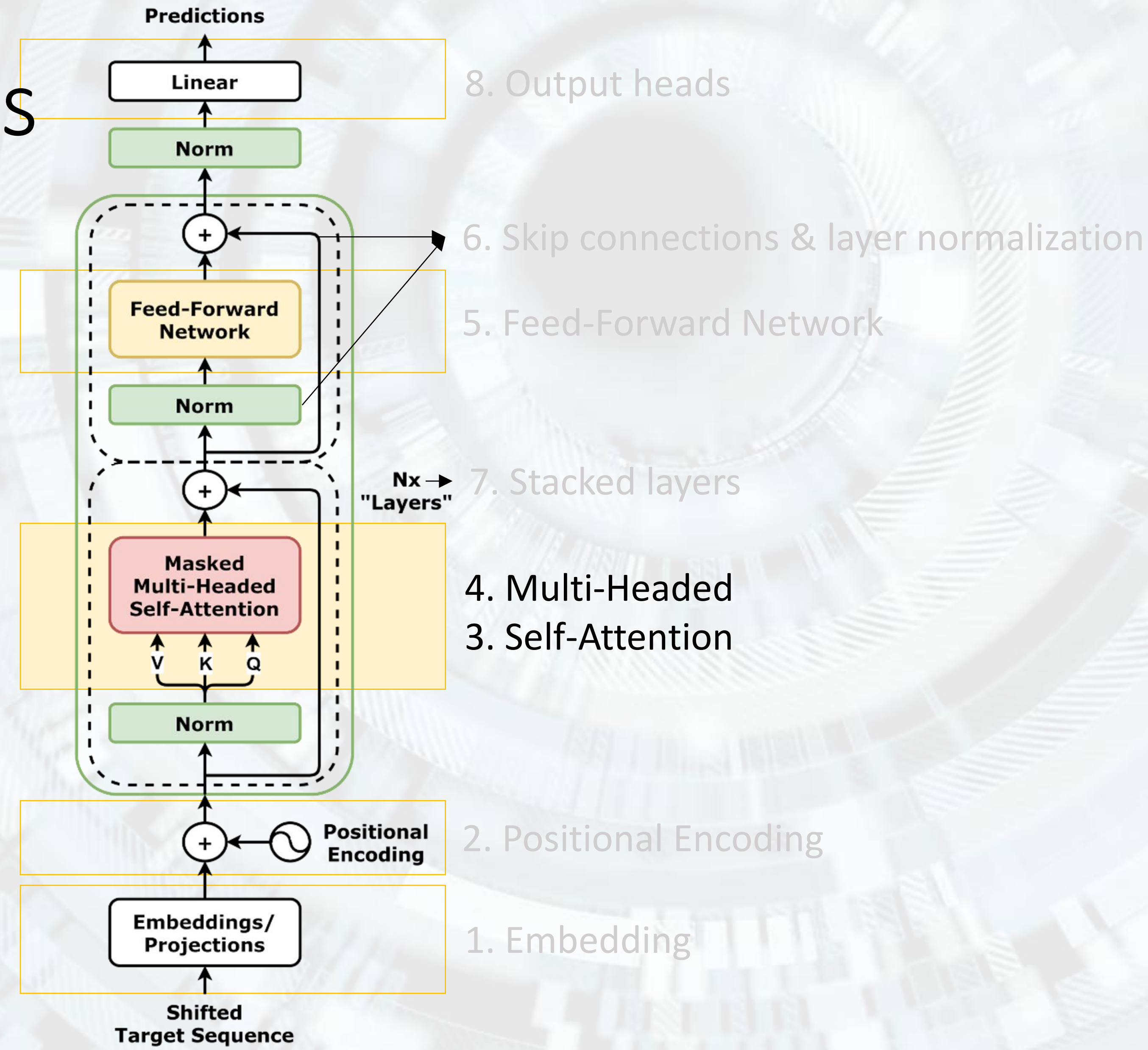
I love you more than anything.

is different from:

I love anything more than you.

GPT - components

Step	Purpose
1. Input Embedding	Encode word meaning
2. Positional Encoding	Add word order information
3. Self-Attention	Learn contextual relationships
4. Multi-Head Attention	Capture different attention types
5. Feed-Forward Network	Non-linear transformation
6. Residual + LayerNorm	Stability and gradient flow
7. Stacking Layers	Increase model capacity
8. Output Layer	Predict next token / classification



3. Self-attention – intuition

We need to see which embedding “attends” to which embedding. Attending = assigning importance to another token based on how relevant it is for understanding the current one.

Token A attends to token B.

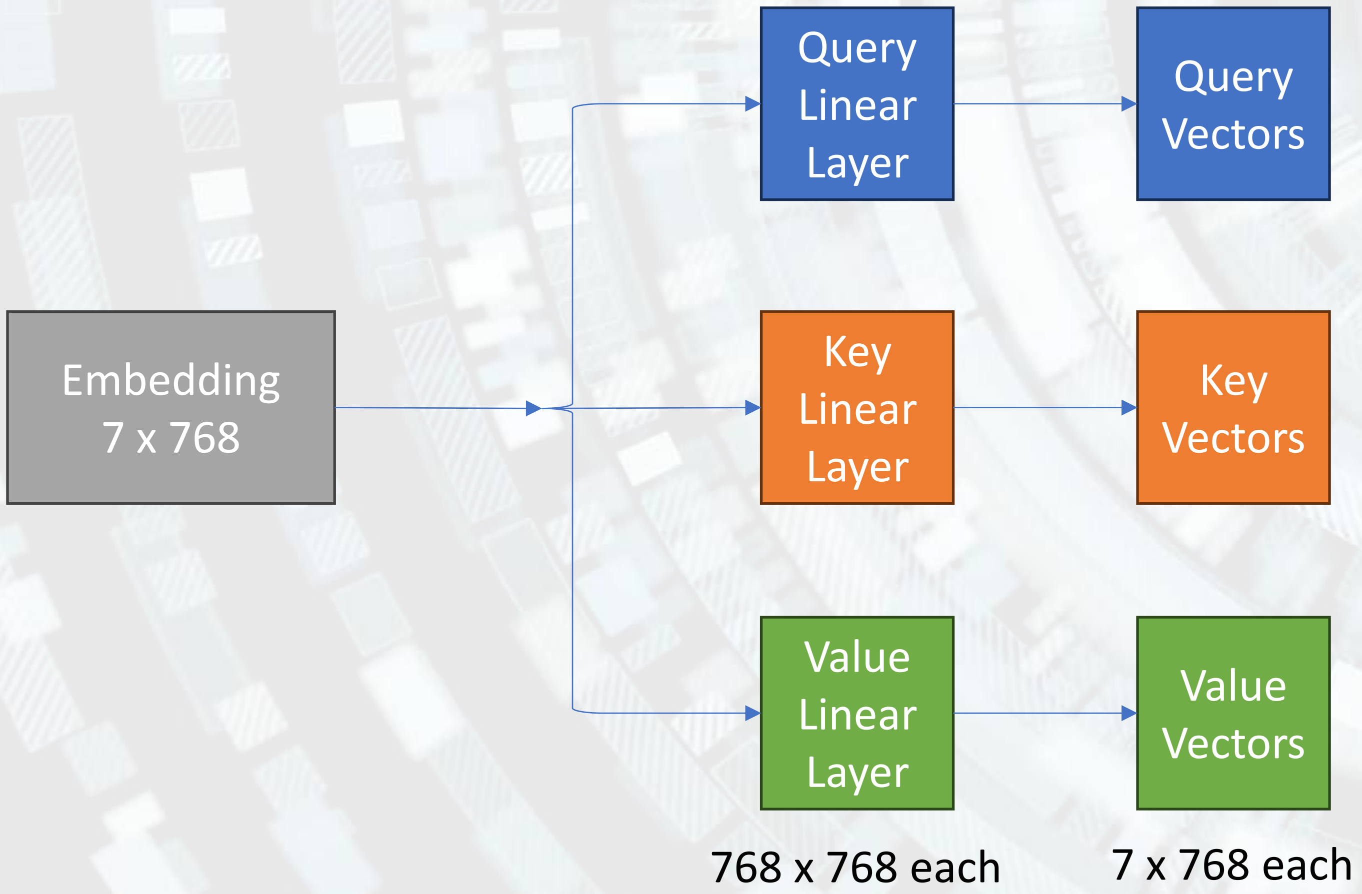
Is the same as saying

Token A is paying attention to Token B to help decide how to better represent itself.

Solution: ATTENTION!!!

3. Self-attention

Each token embedding is transformed in 3 vectors of the same length as the embedding: Query, Key, Value.



Queries represent the question that token A asks to find information about himself.

Keys are short/summarized descriptions of other tokens.

Values are the actual information content in the tokens.

3. Self-attention

If the query aligns (coincides in meaning) with a key, then it will assign more weight to that particular embedding.

Dot product		U	nexpected	ly	,	the	boy	entered
	$K_i \cdot Q_j$	K_U	$K_{nexpected}$	K_{ly}	$K_,$	K_{the}	K_{boy}	$K_{entered}$
U	Q_U	$K_U \cdot Q_U$	$K_{nexpected} \cdot Q_U$	$K_{ly} \cdot Q_U$	$K_, \cdot Q_U$	$K_{the} \cdot Q_U$	$K_{boy} \cdot Q_U$	$K_{entered} \cdot Q_U$
nexpected	$Q_{nexpted}$	$K_U \cdot Q_{nexpted}$	$K_{nexpected} \cdot Q_{nexpted}$	$K_{ly} \cdot Q_{nexpted}$	$K_, \cdot Q_{nexpted}$	$K_{the} \cdot Q_{nexpted}$	$K_{boy} \cdot Q_{nexpted}$	$K_{entered} \cdot Q_{nexpted}$
ly	Q_{ly}	$K_U \cdot Q_{ly}$	$K_{nexpected} \cdot Q_{ly}$	$K_{ly} \cdot Q_{ly}$	$K_, \cdot Q_{ly}$	$K_{the} \cdot Q_{ly}$	$K_{boy} \cdot Q_{ly}$	$K_{entered} \cdot Q_{ly}$
,	$Q_,$	$K_U \cdot Q_,$	$K_{nexpected} \cdot Q_,$	$K_{ly} \cdot Q_,$	$K_, \cdot Q_,$	$K_{the} \cdot Q_,$	$K_{boy} \cdot Q_,$	$K_{entered} \cdot Q_,$
the	Q_{the}	$K_U \cdot Q_{the}$	$K_{nexpected} \cdot Q_{the}$	$K_{ly} \cdot Q_{the}$	$K_, \cdot Q_{the}$	$K_{the} \cdot Q_{the}$	$K_{boy} \cdot Q_{the}$	$K_{entered} \cdot Q_{the}$
boy	Q_{boy}	$K_U \cdot Q_{boy}$	$K_{nexpected} \cdot Q_{boy}$	$K_{ly} \cdot Q_{boy}$	$K_, \cdot Q_{boy}$	$K_{the} \cdot Q_{boy}$	$K_{boy} \cdot Q_{boy}$	$K_{entered} \cdot Q_{boy}$
entered	$Q_{entered}$	$K_U \cdot Q_{entered}$	$K_{nexpected} \cdot Q_{entered}$	$K_{ly} \cdot Q_{entered}$	$K_, \cdot Q_{entered}$	$K_{the} \cdot Q_{entered}$	$K_{boy} \cdot Q_{entered}$	$K_{entered} \cdot Q_{entered}$

3. Self-attention

Attention runs only in the past ⬄ queries are not allowed to investigate future keys.

Dot product		U	nexpected	ly	,	the	boy	entered
	$K_i \cdot Q_j$	K_U	$K_{nexpected}$	K_{ly}	$K_,$	K_{the}	K_{boy}	$K_{entered}$
U	Q_U	$K_U \cdot Q_U$	$\cancel{K_{nexpected} \cdot Q_U}$	$\cancel{K_{ly} \cdot Q_U}$	$\cancel{K_, \cdot Q_U}$	$\cancel{K_{the} \cdot Q_U}$	$\cancel{K_{boy} \cdot Q_U}$	$\cancel{K_{entered} \cdot Q_U}$
nexpected	$Q_{nexpted}$	$K_U \cdot Q_{nexpted}$	$K_{nexpected} \cdot Q_{nexpted}$	$\cancel{K_{ly} \cdot Q_{nexpted}}$	$\cancel{K_, \cdot Q_{nexpted}}$	$\cancel{K_{the} \cdot Q_{nexpted}}$	$\cancel{K_{boy} \cdot Q_{nexpted}}$	$\cancel{K_{entered} \cdot Q_{nexpted}}$
ly	Q_{ly}	$K_U \cdot Q_{ly}$	$K_{nexpected} \cdot Q_{ly}$	$K_{ly} \cdot Q_{ly}$	$\cancel{K_, \cdot Q_{ly}}$	$\cancel{K_{the} \cdot Q_{ly}}$	$\cancel{K_{boy} \cdot Q_{ly}}$	$\cancel{K_{entered} \cdot Q_{ly}}$
,	$Q_,$	$K_U \cdot Q_,$	$K_{nexpected} \cdot Q_,$	$K_{ly} \cdot Q_,$	$K_, \cdot Q_,$	$\cancel{K_{the} \cdot Q_,}$	$\cancel{K_{boy} \cdot Q_,}$	$\cancel{K_{entered} \cdot Q_,}$
the	Q_{the}	$K_U \cdot Q_{the}$	$K_{nexpected} \cdot Q_{the}$	$K_{ly} \cdot Q_{the}$	$K_, \cdot Q_{the}$	$K_{the} \cdot Q_{the}$	$\cancel{K_{boy} \cdot Q_{the}}$	$\cancel{K_{entered} \cdot Q_{the}}$
boy	Q_{boy}	$K_U \cdot Q_{boy}$	$K_{nexpected} \cdot Q_{boy}$	$K_{ly} \cdot Q_{boy}$	$K_, \cdot Q_{boy}$	$K_{the} \cdot Q_{boy}$	$K_{boy} \cdot Q_{boy}$	$\cancel{K_{entered} \cdot Q_{boy}}$
entered	$Q_{entered}$	$K_U \cdot Q_{entered}$	$K_{nexpected} \cdot Q_{entered}$	$K_{ly} \cdot Q_{entered}$	$K_, \cdot Q_{entered}$	$K_{the} \cdot Q_{entered}$	$K_{boy} \cdot Q_{entered}$	$K_{entered} \cdot Q_{entered}$

3. Self-attention

Attention runs only in the past ⇔ queries are not allowed to investigate future keys => assign them –inf weight.

Dot product		U	nexpected	ly	,	the	boy	entered
$K_i \cdot Q_j$		K_U	$K_{nexpected}$	K_{ly}	$K_{,}$	K_{the}	K_{boy}	$K_{entered}$
U	Q_U	$K_U \cdot Q_U$	-inf	-inf	-inf	-inf	-inf	-inf
nexpected	$Q_{nexpted}$	$K_U \cdot Q_{nexpted}$	$K_{nexpected} \cdot Q_{nexpted}$	-inf	-inf	-inf	-inf	-inf
ly	Q_{ly}	$K_U \cdot Q_{ly}$	$K_{nexpected} \cdot Q_{ly}$	$K_{ly} \cdot Q_{ly}$	-inf	-inf	-inf	-inf
,	$Q_{,}$	$K_U \cdot Q_{,}$	$K_{nexpected} \cdot Q_{,}$	$K_{ly} \cdot Q_{,}$	$K_{,} \cdot Q_{,}$	-inf	-inf	-inf
the	Q_{the}	$K_U \cdot Q_{the}$	$K_{nexpected} \cdot Q_{the}$	$K_{ly} \cdot Q_{the}$	$K_{,} \cdot Q_{the}$	$K_{the} \cdot Q_{the}$	-inf	-inf
boy	Q_{boy}	$K_U \cdot Q_{boy}$	$K_{nexpected} \cdot Q_{boy}$	$K_{ly} \cdot Q_{boy}$	$K_{,} \cdot Q_{boy}$	$K_{the} \cdot Q_{boy}$	$K_{boy} \cdot Q_{boy}$	-inf
entered	$Q_{entered}$	$K_U \cdot Q_{entered}$	$K_{nexpected} \cdot Q_{entered}$	$K_{ly} \cdot Q_{entered}$	$K_{,} \cdot Q_{entered}$	$K_{the} \cdot Q_{entered}$	$K_{boy} \cdot Q_{entered}$	$K_{entered} \cdot Q_{entered}$

3. Self-attention

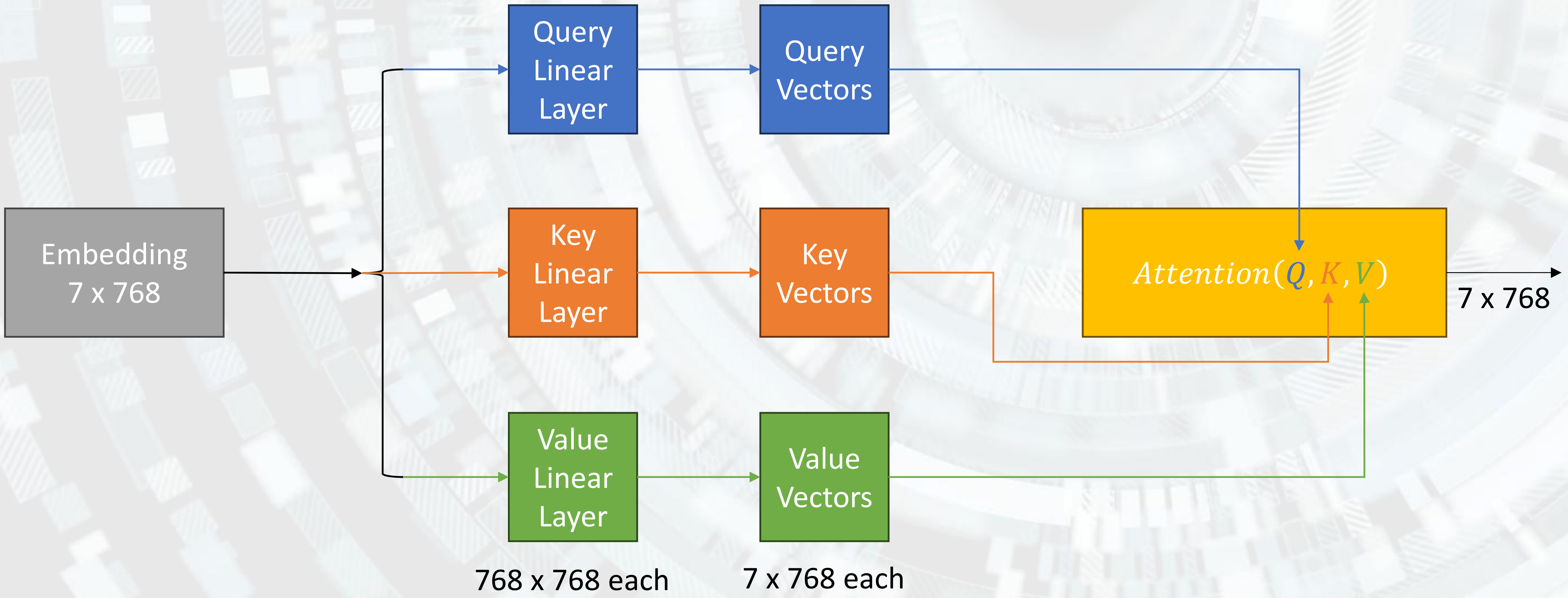
Scale and transform to probability distribution. Finally, retrieve the weighted values.

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

Turns the vectors into probability distributions

Dimension of the keys vector
Helps with stability

3. Self-attention



3. Self-attention

Outcome:

A wizard living in Hogwarts that was named Harry.

The Queen of Sussex followed along prince Harry.

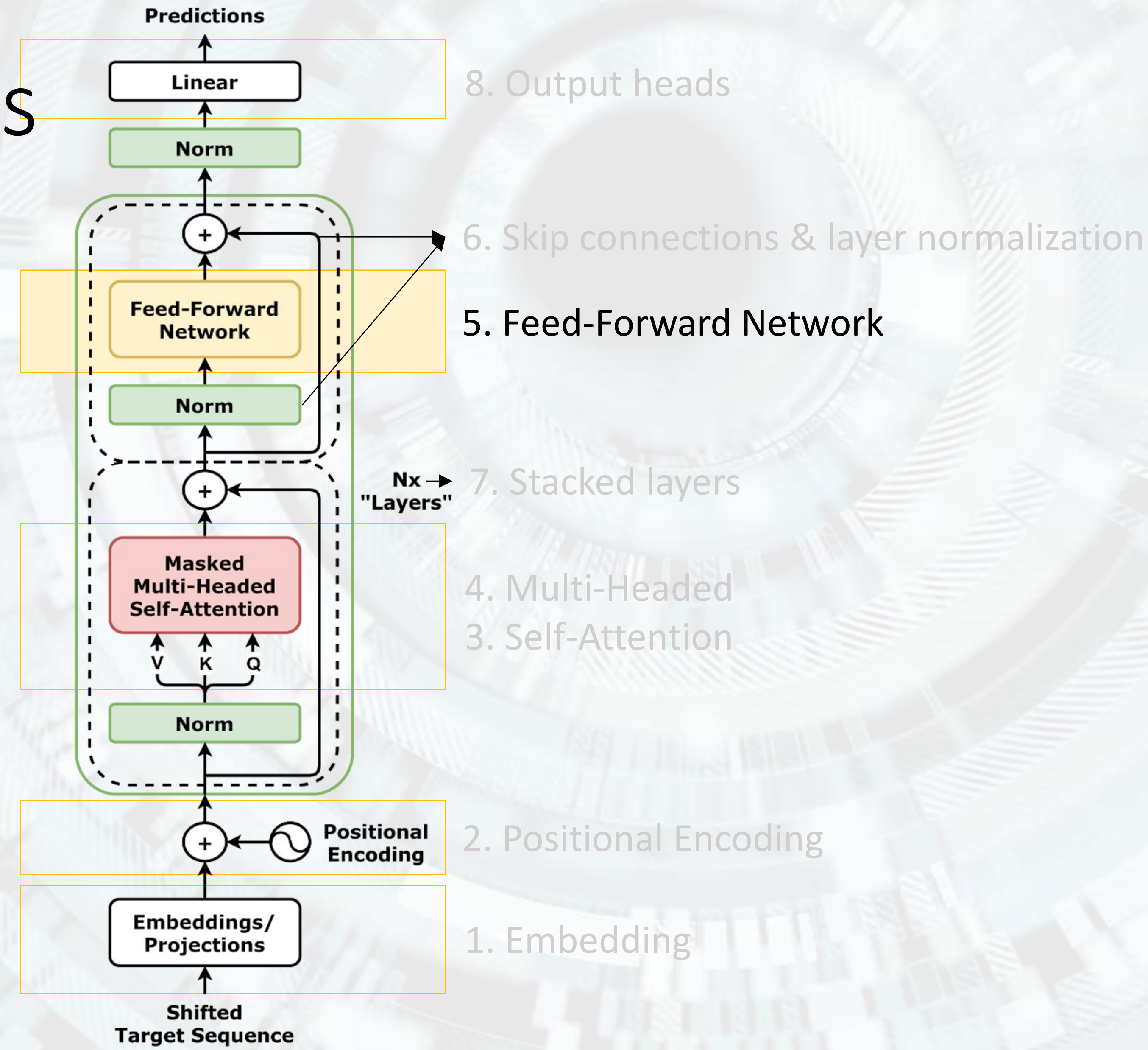
4. Multi-headed Self-attention

Repeat the attention block several (12) times in parallel to gain more knowledge on the context.

- Each head attends to **different parts** of the input.
- This adds **diversity** to the way the model understands context.
- Together, they create a **richer** representation of each token.

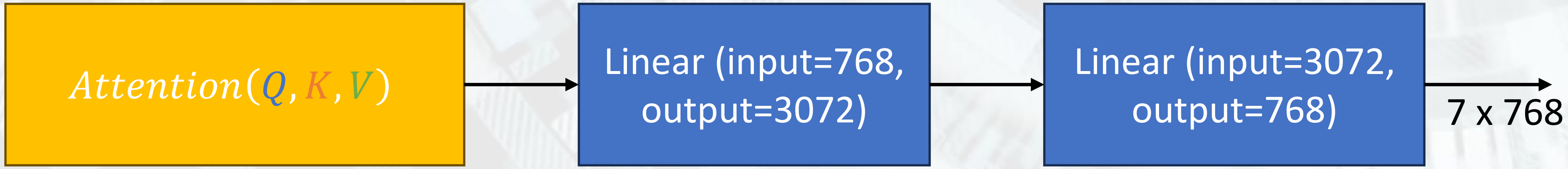
GPT - components

Step	Purpose
1. Input Embedding	Encode word meaning
2. Positional Encoding	Add word order information
3. Self-Attention	Learn contextual relationships
4. Multi-Head Attention	Capture different attention types
5. Feed-Forward Network	Non-linear transformation
6. Residual + LayerNorm	Stability and gradient flow
7. Stacking Layers	Increase model capacity
8. Output Layer	Predict next token / classification



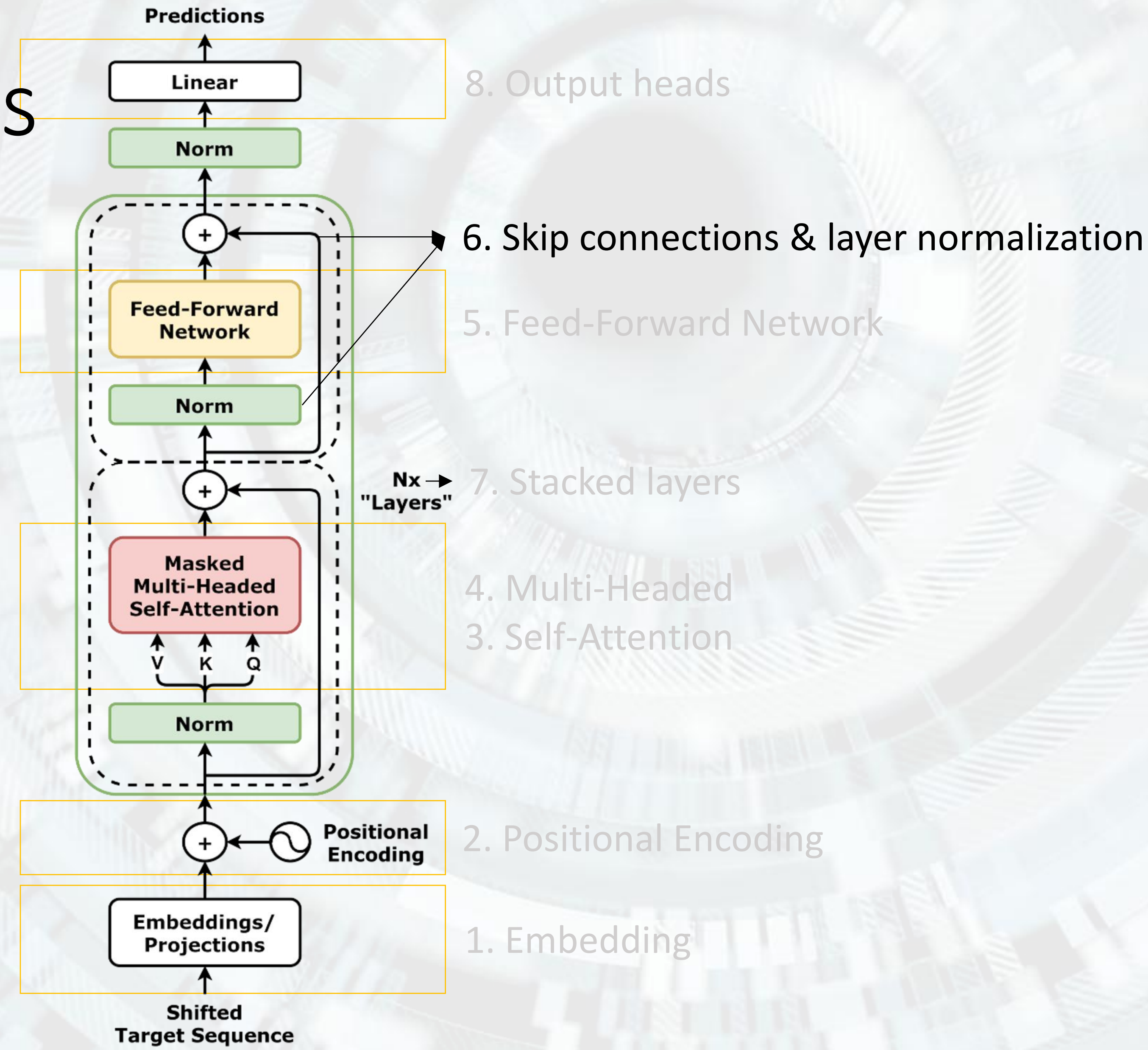
5. Feed-Forward Network

- Attention tells the model: “what other tokens should I focus on?” – brings **context**.
- FFN tells the model: “how should I process and transform that focused info?” – brings **insight**.
- It’s where the model can apply learned reasoning or feature extraction on a per-token basis.
- Runs on each token, individually – tokens do not communicate with each other at this step.



GPT - components

Step	Purpose
1. Input Embedding	Encode word meaning
2. Positional Encoding	Add word order information
3. Self-Attention	Learn contextual relationships
4. Multi-Head Attention	Capture different attention types
5. Feed-Forward Network	Non-linear transformation
6. Residual + LayerNorm	Stability and gradient flow
7. Stacking Layers	Increase model capacity
8. Output Layer	Predict next token / classification



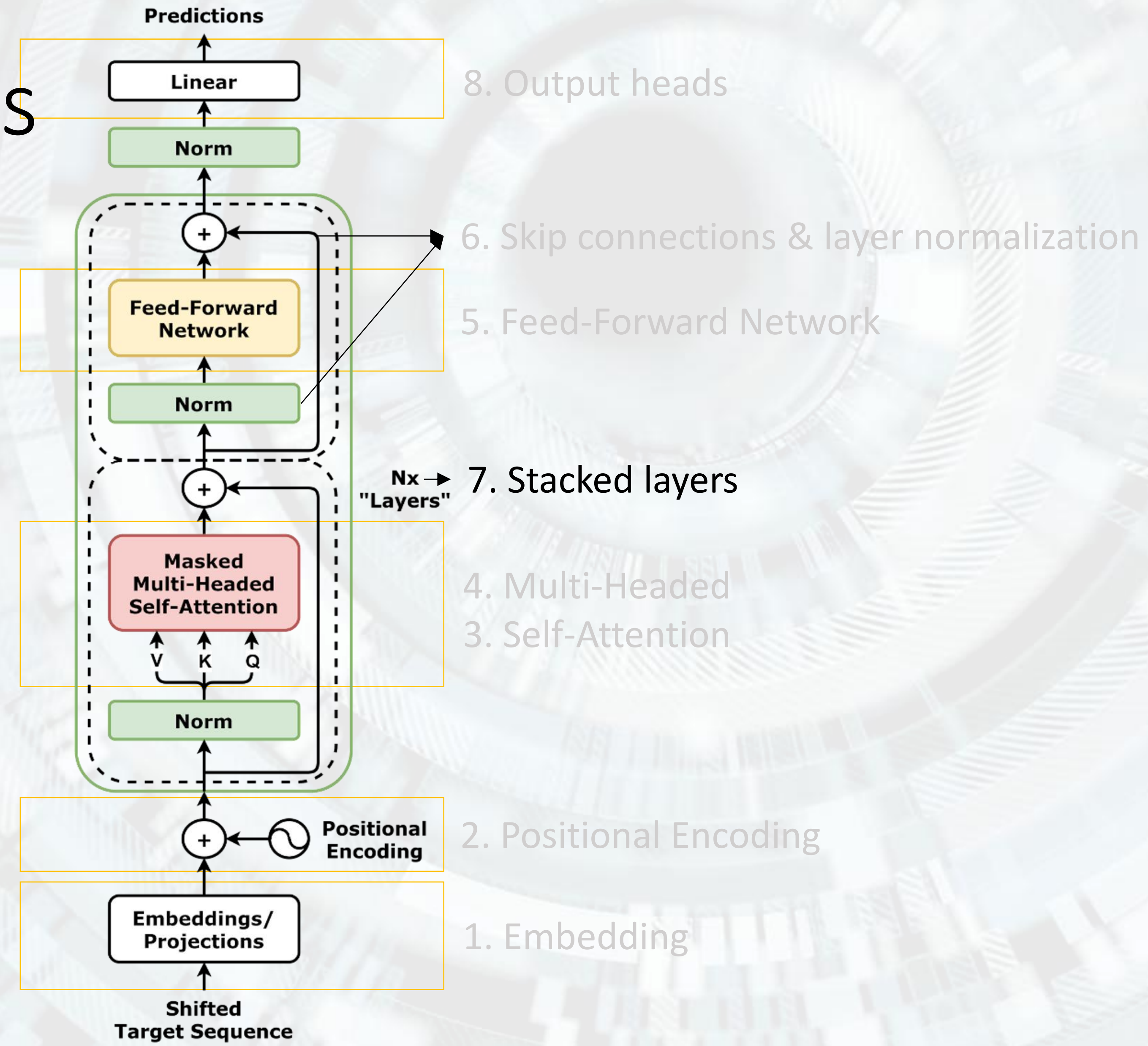
6. Residual + LayerNorm

- Residual connections preserve gradient => no vanishing gradient.

- LayerNorm preserves constant scale => no exploding gradient.

GPT - components

Step	Purpose
1. Input Embedding	Encode word meaning
2. Positional Encoding	Add word order information
3. Self-Attention	Learn contextual relationships
4. Multi-Head Attention	Capture different attention types
5. Feed-Forward Network	Non-linear transformation
6. Residual + LayerNorm	Stability and gradient flow
7. Stacking Layers	Increase model capacity
8. Output Layer	Predict next token / classification



7. Stacking layers

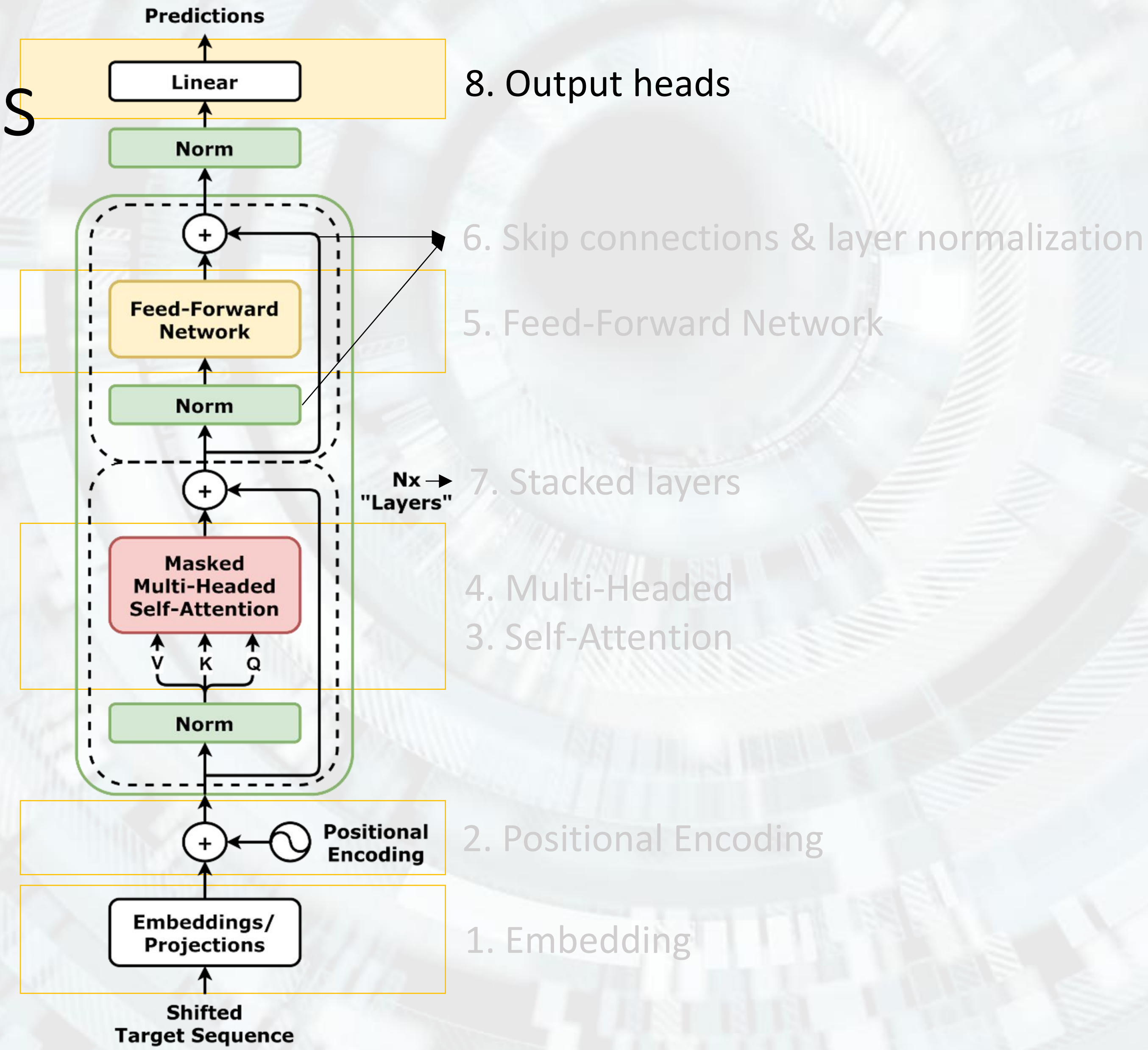
- Each Attention + FFN block is repeated several (again, 12) times

- This is also defined as **depth**.

- Depth helps the model:
  - Refine representations over multiple stages
  - Learn more complex patterns and hierarchies
  - Improve expressive power without just widening everything

GPT - components

Step	Purpose
1. Input Embedding	Encode word meaning
2. Positional Encoding	Add word order information
3. Self-Attention	Learn contextual relationships
4. Multi-Head Attention	Capture different attention types
5. Feed-Forward Network	Non-linear transformation
6. Residual + LayerNorm	Stability and gradient flow
7. Stacking Layers	Increase model capacity
8. Output Layer	Predict next token / classification



8. Output layer

- The output layer's input feature is of size 7 x 768 (num_tokens x dim_embedding)
- At the end, we will sample a token from the output probability distribution.

U	1 x 768
nexpected	1 x 768
ly	1 x 768
,	1 x 768
the	1 x 768
boy	1 x 768
entered	1 x 768

Used only during training

Linear (input=768, output=50257)

Unembedding layer

aah	-168.32
aardwark	-185.2
aardwolf	-170.85
aargh	-172.00
ab	-173.5
aback	-184.2
abacterial	-192.87
…	
zygosis	-179.82
zygote	-184.3
zygotic	-179.64
zyme	-173.89
zymogen	-184.76
zymosis	-184.78
zzz	-177.65

Top-k

the	-100.43
a	-111.87
his	-120.85
into	-122.00
an	-123.5
with	-134.2
.	-142.87
and	-145.8
my	-149.82
,	-154.3
through	-159.64
her	-163.89
in	-164.76
without	-164.78
our	-167.65

Softmax

the	78.74%
a	11%
his	4.39%
into	3.76%
an	1.13%
with	0%
.	0%
and	0%
my	0%
,	0%
through	0%
her	0%
in	0%
without	0%
our	0%

Let's test it out – unit #11