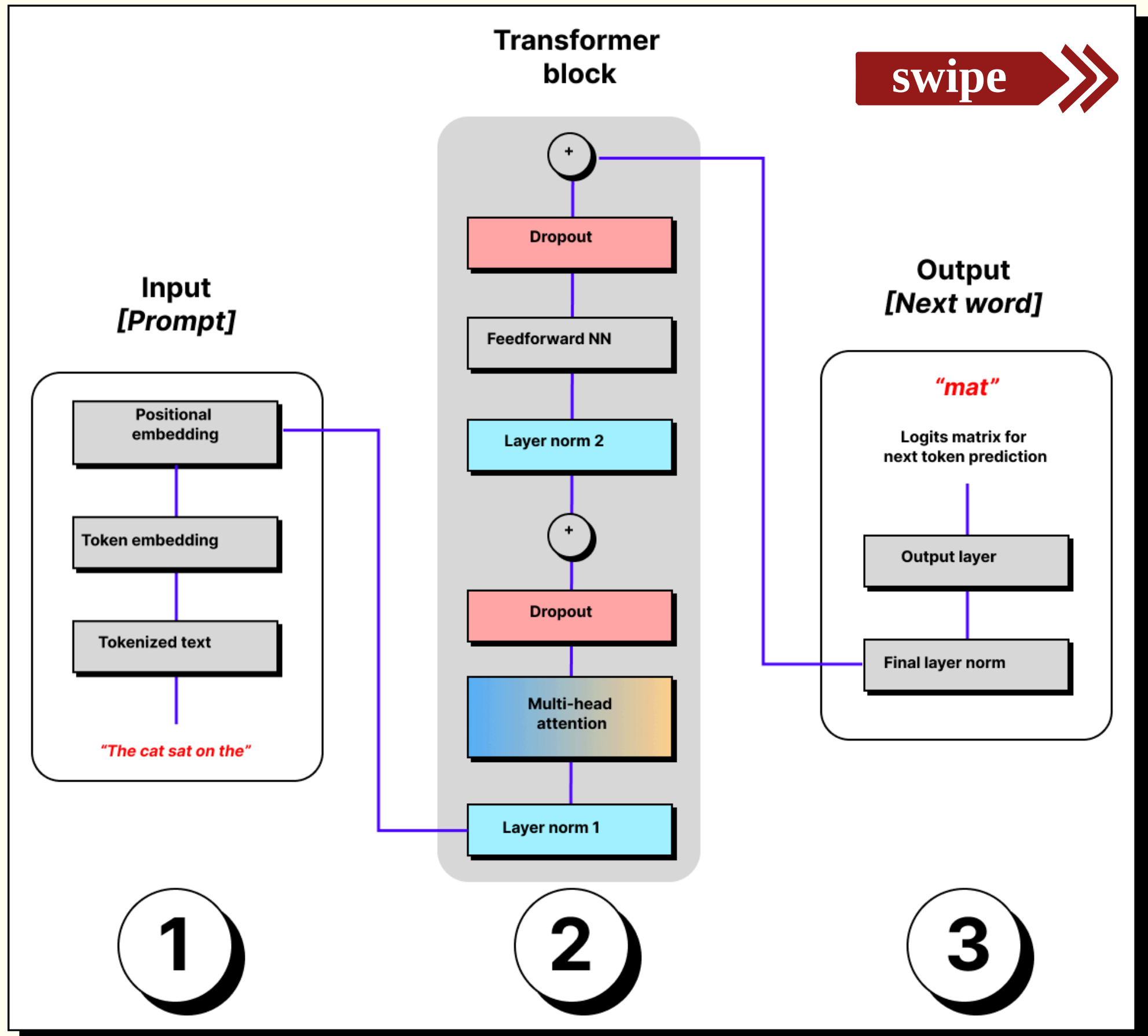
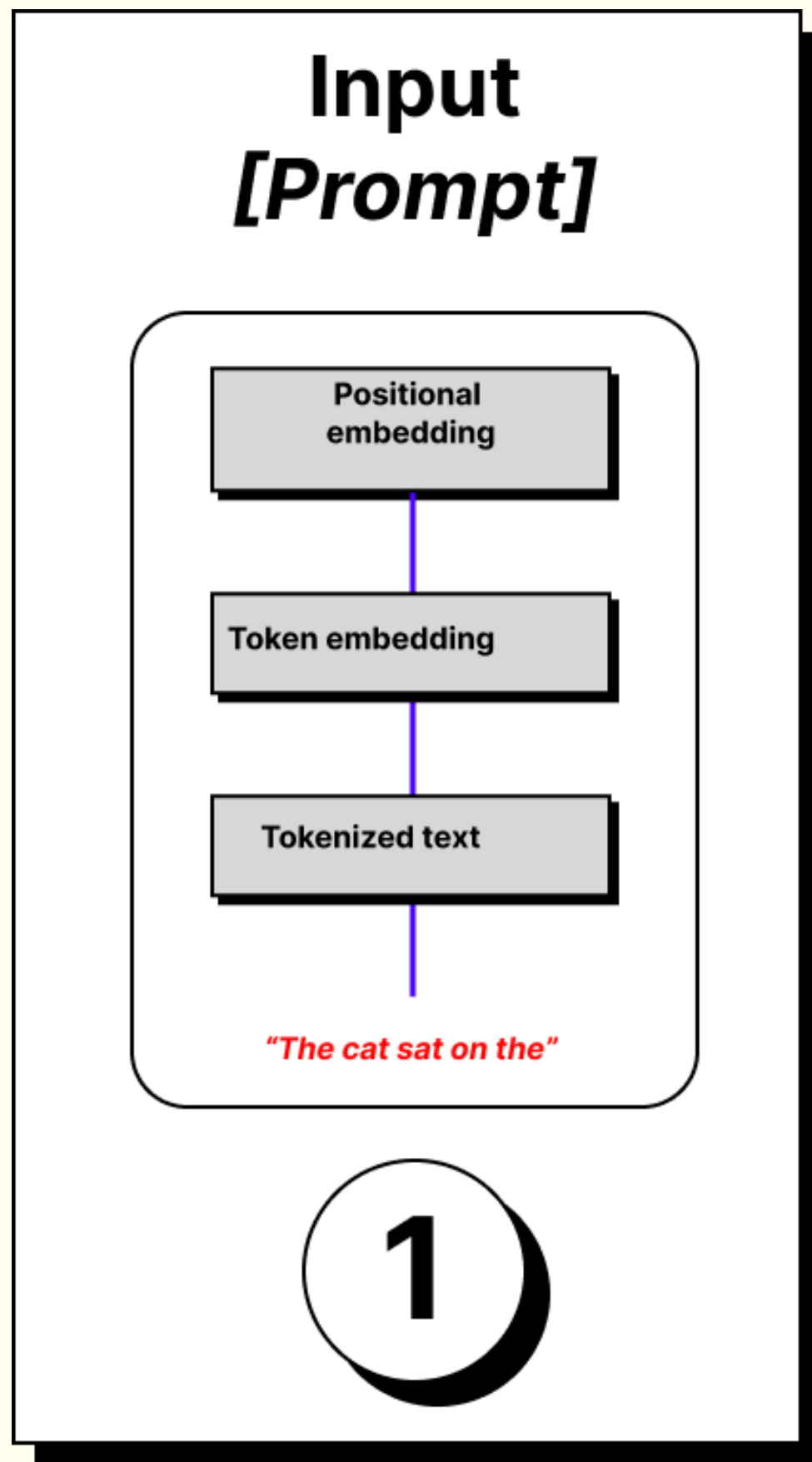


Transformer architecture

In 3 parts

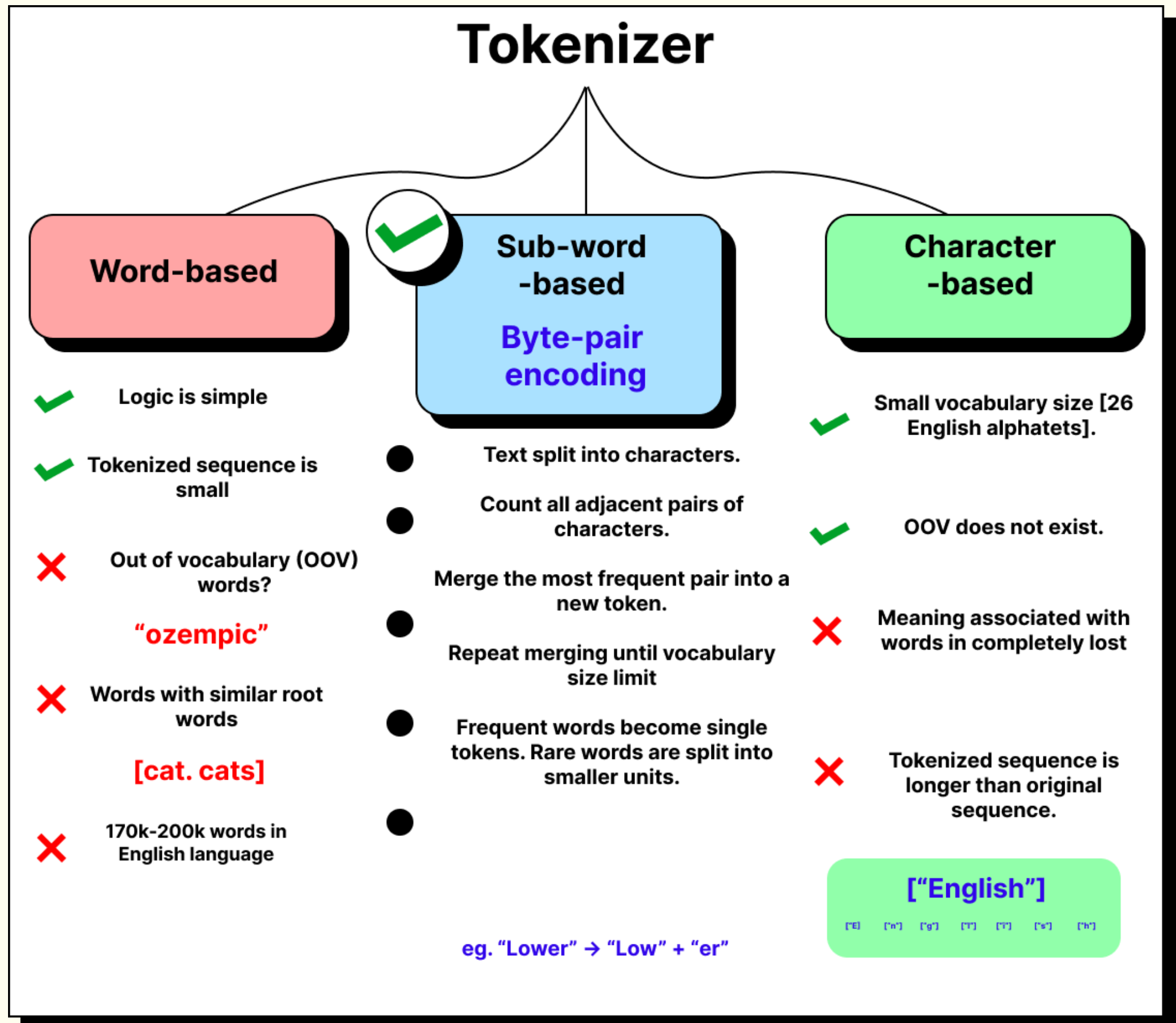


Part 1: Input



Part 1: Input

Tokenizer options

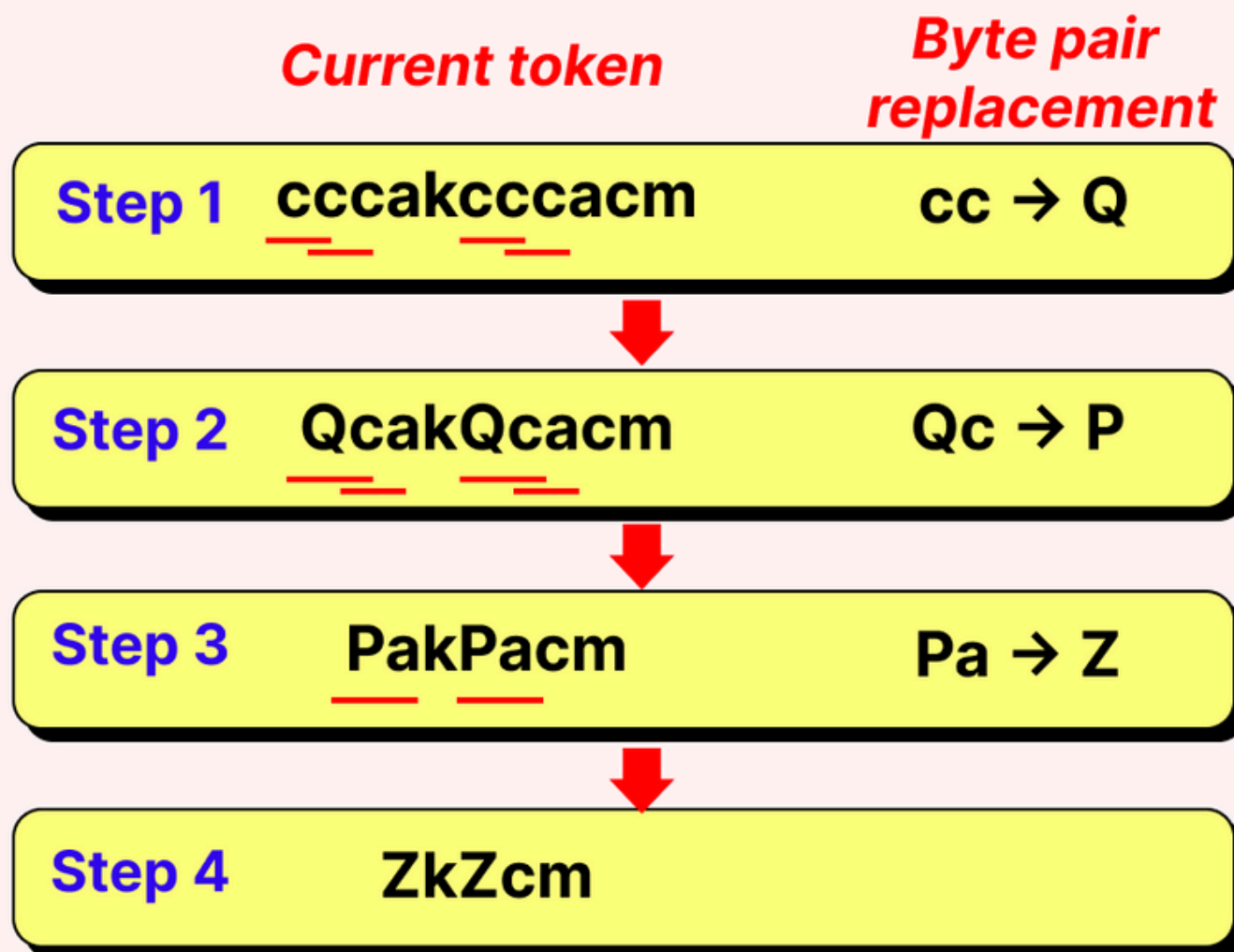


Part 1: Input

Tokenizer used

Byte pair encoding [BPE]

Example token = ["cccakccacm"]



- The most frequent pair is merged into a new character
- Continue till desired vocabulary size or no more frequent pairs
- Allows the tokenizer to build a vocabulary of subword units
- Helps in handling rare words, out-of-vocabulary (OOV) words



Part 1: Input

Simplified tokenization example

Input

Sentence is split into tokens

The cat sat on the

Now consider one token

cat

What happens to this token?



Part 1: Input

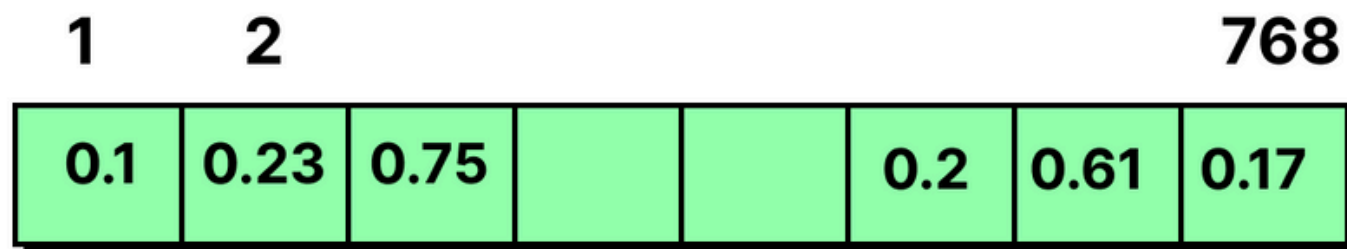
Identify token vectors from vocabulary

Dictionary of token IDs

a → 1
b → 2
⋮
cat → 3428

mat → 7568
⋮
<end> → 50257

Token embedding in 768 dimensions



Is noun?

Is gender?

Is verb?

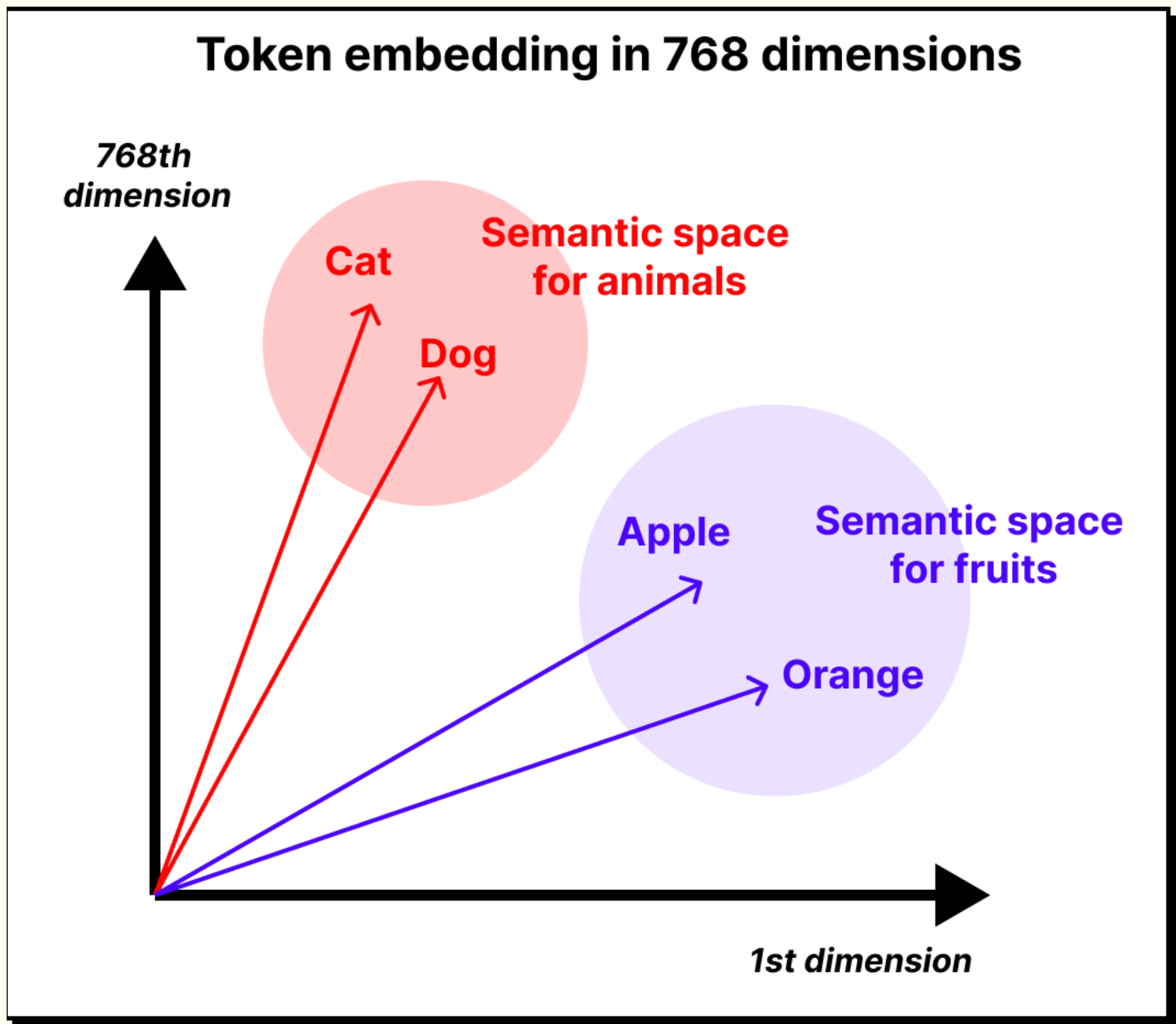
Is animal?

Is emotion?



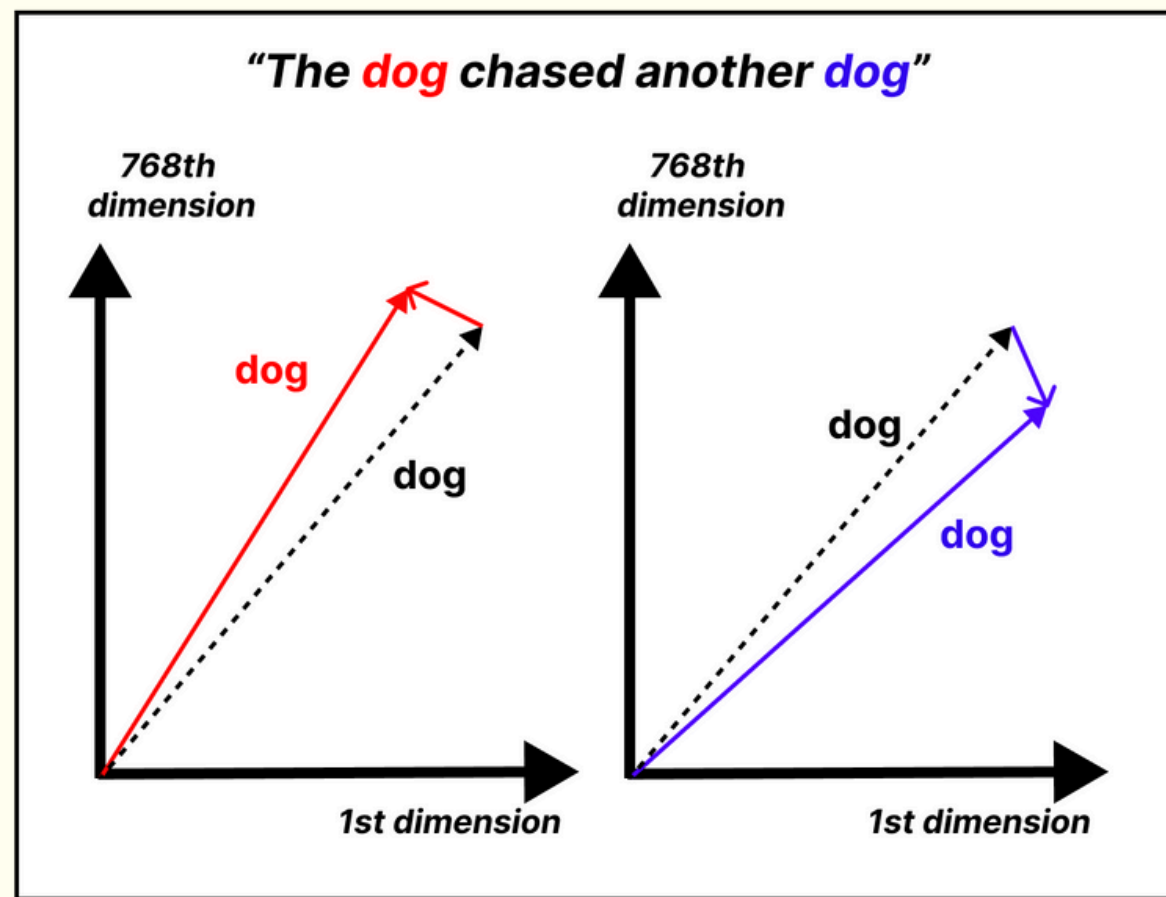
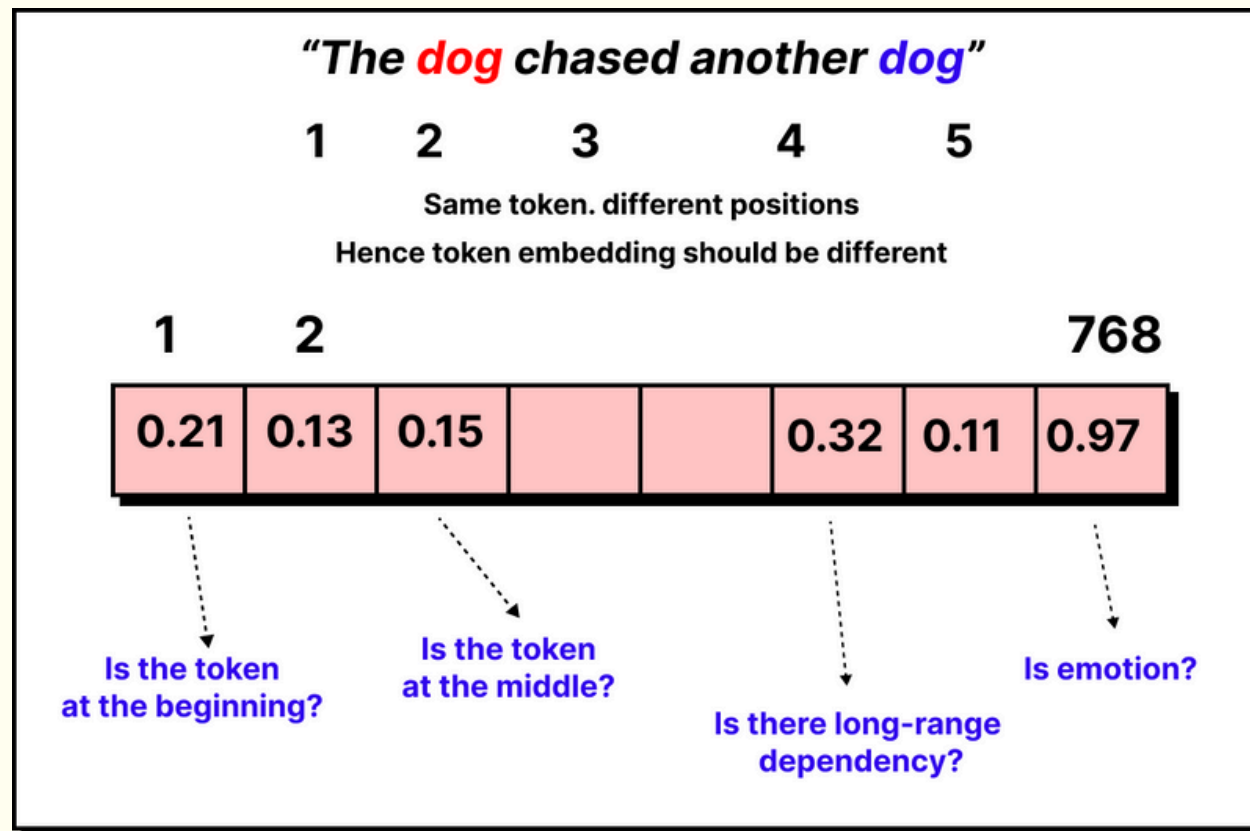
Part 1: Input

Token vectors carry semantic meaning



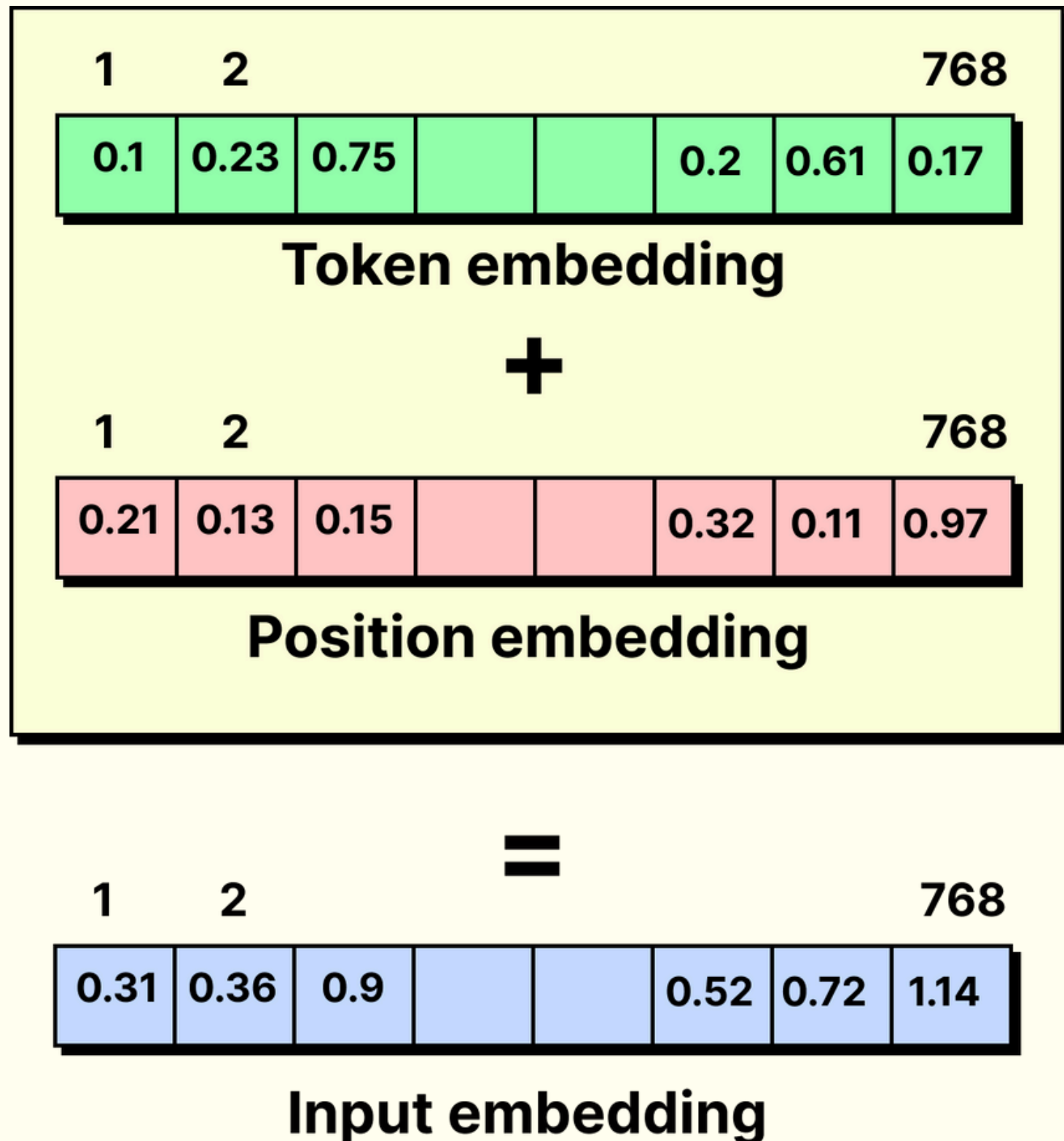
Part 1: Input

It is important to consider token position

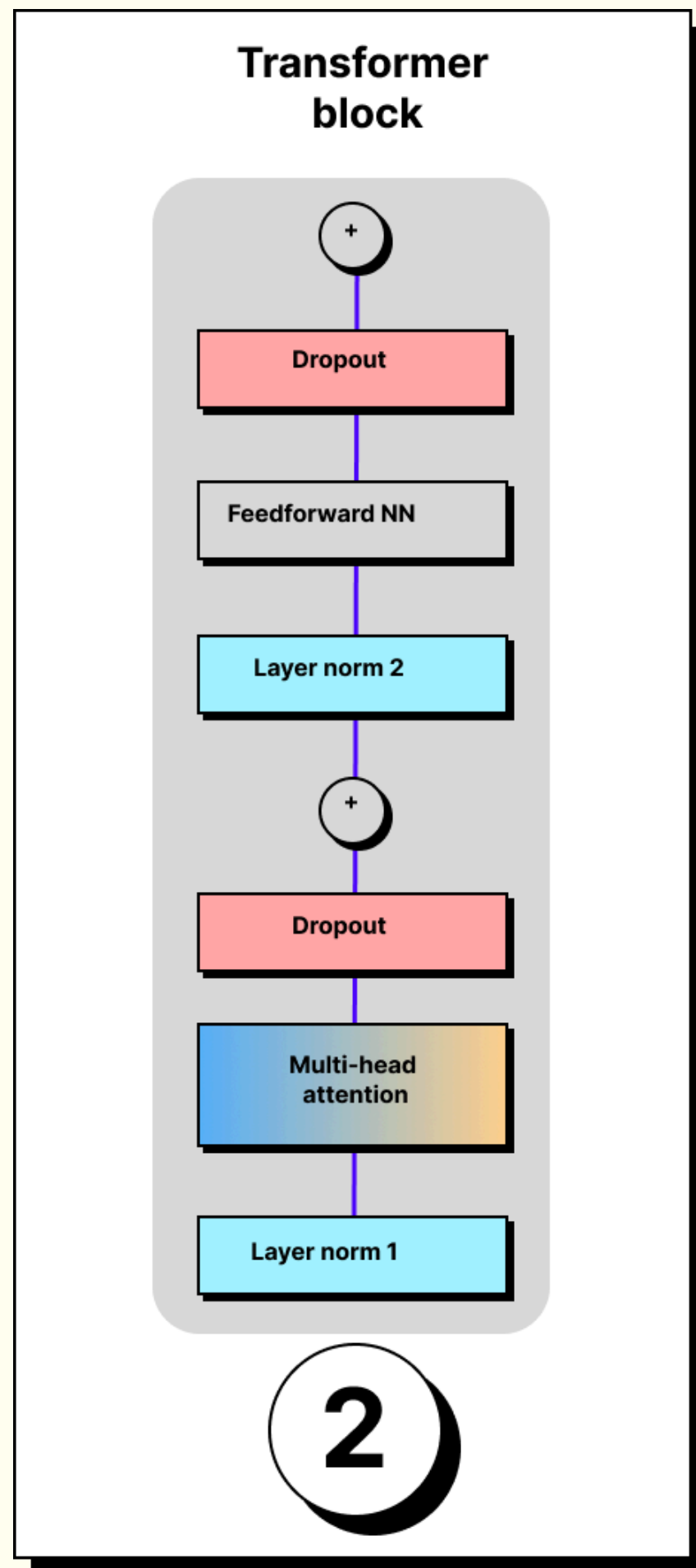


Part 1: Input

Add position embedding to create input embedding

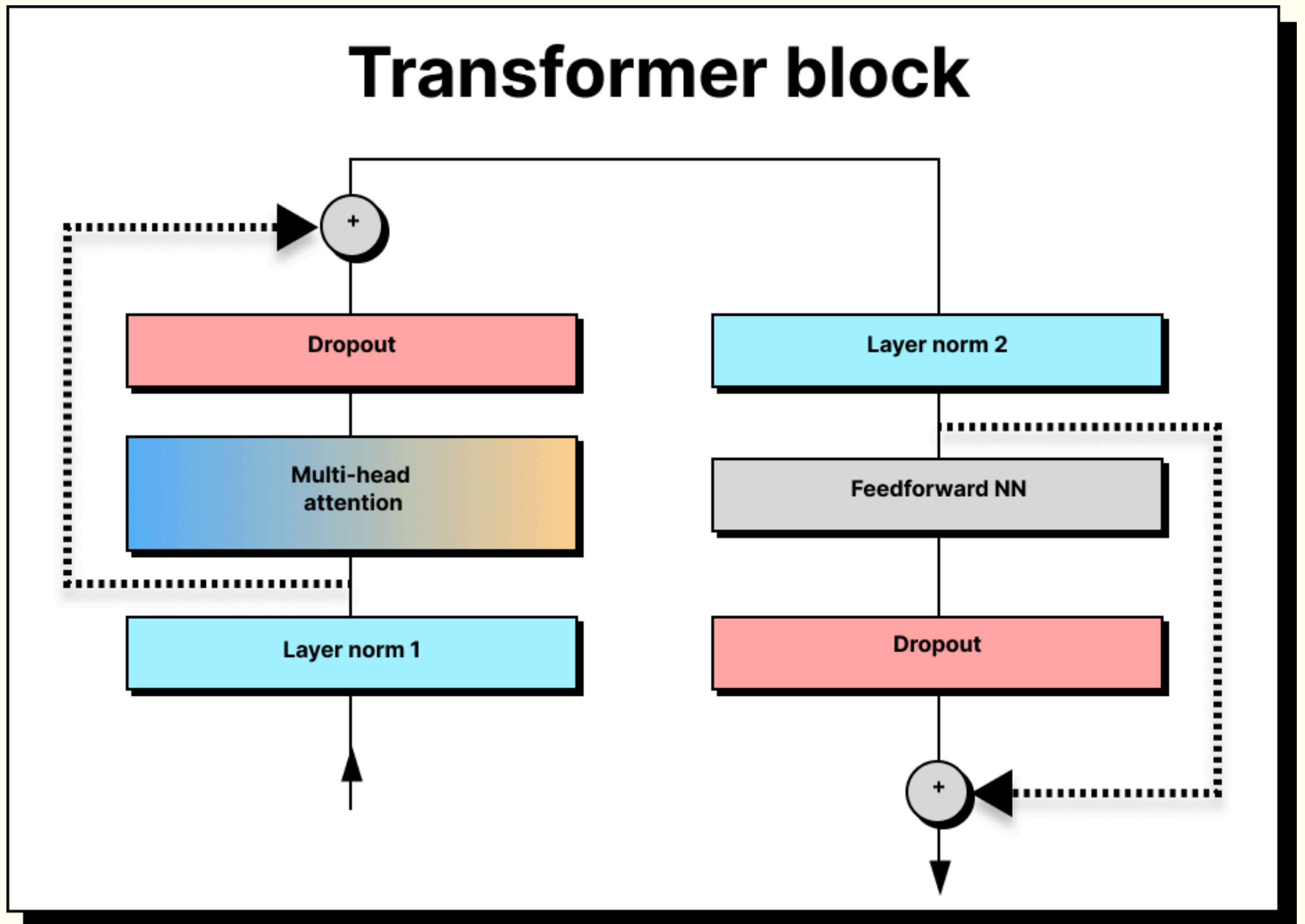


Part 2: Transformer block



Part 2: Transformer block

We will focus on overlooked aspects



Part 2: Transformer block

Layer normalization

**Layer normalization -
applied for each token**

Eg. token: $x=[2.0.-1.0.3.0.0.0]$

$$\text{mean}(x)=[2.0+(-1.0)+3.0+0.0]/4=4.0/4=1.0$$

$$\begin{aligned}\text{variance}(x) &= [(2-1)^2+(-1-1)^2+(3-1)^2+(0-1)^2]/4 \\ &= [1+4+4+1]/4=10/4=2.5\end{aligned}$$

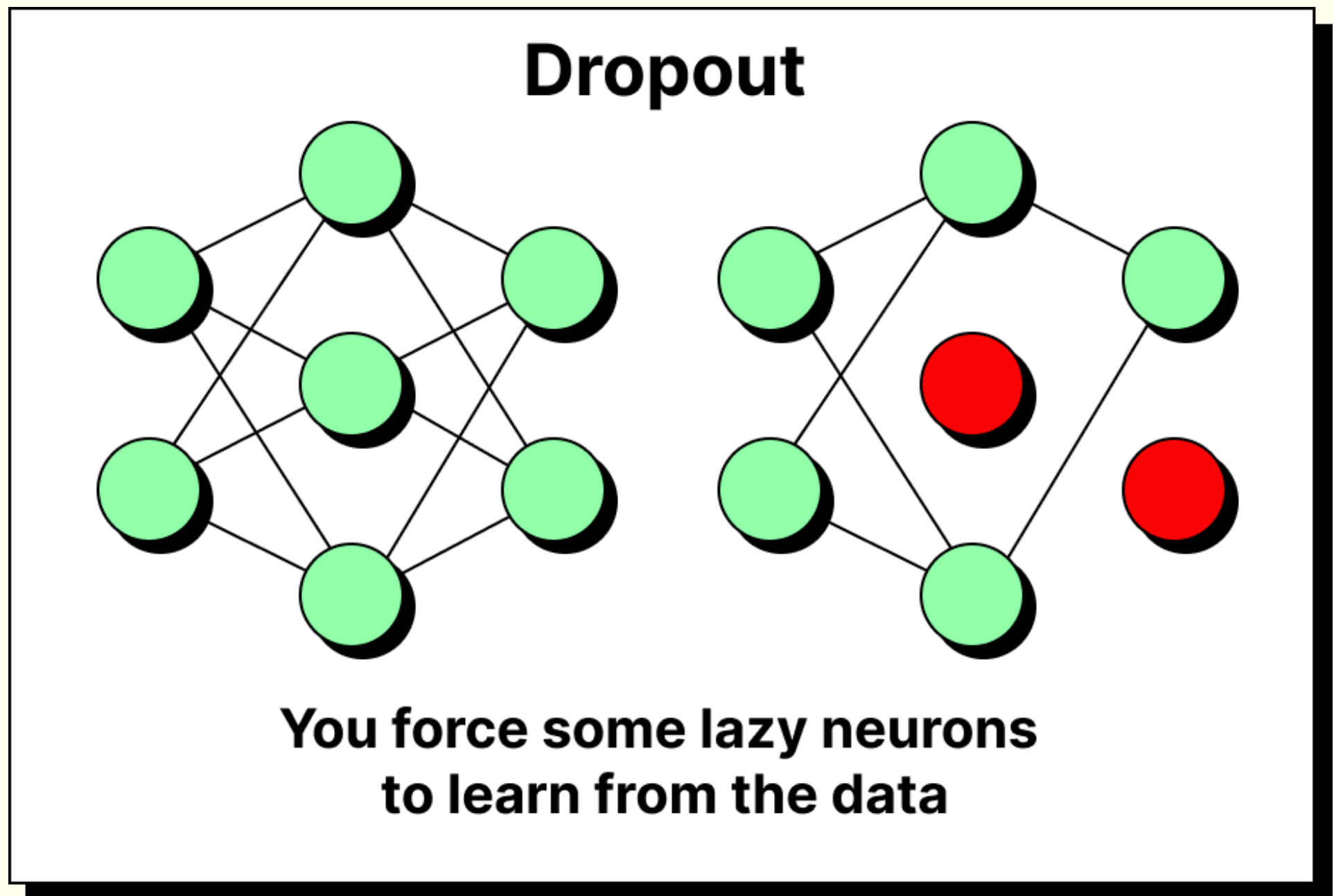
$$\text{std}(x)=\text{sqrt}(2.5)\approx 1.58$$

$$x = [x-\text{mean}(x)]/\text{std}(x) \approx [0.63.-1.26.1.26.-0.63]$$



Part 2: Transformer block

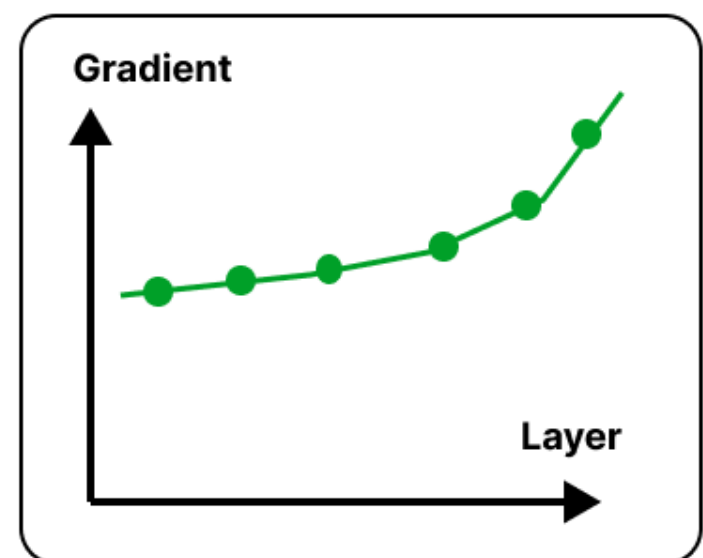
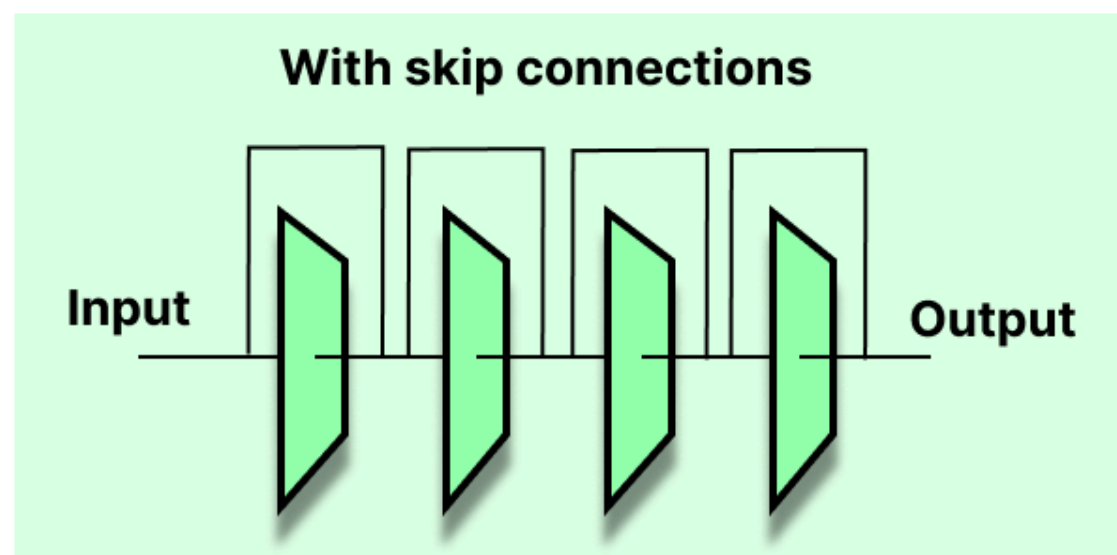
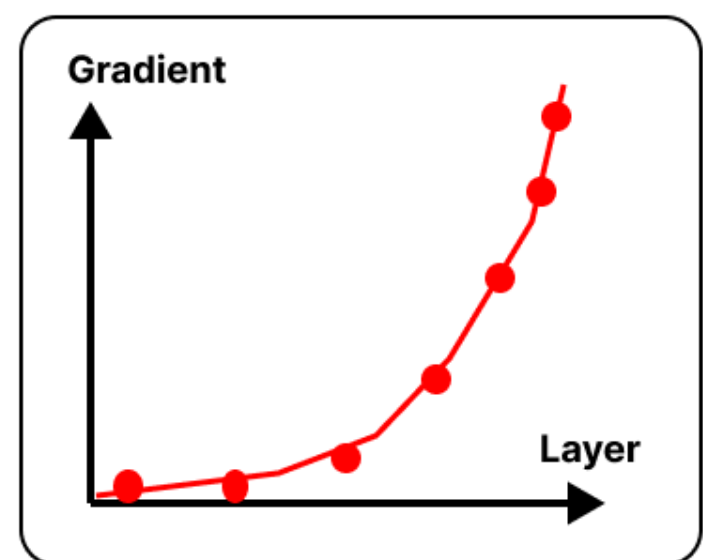
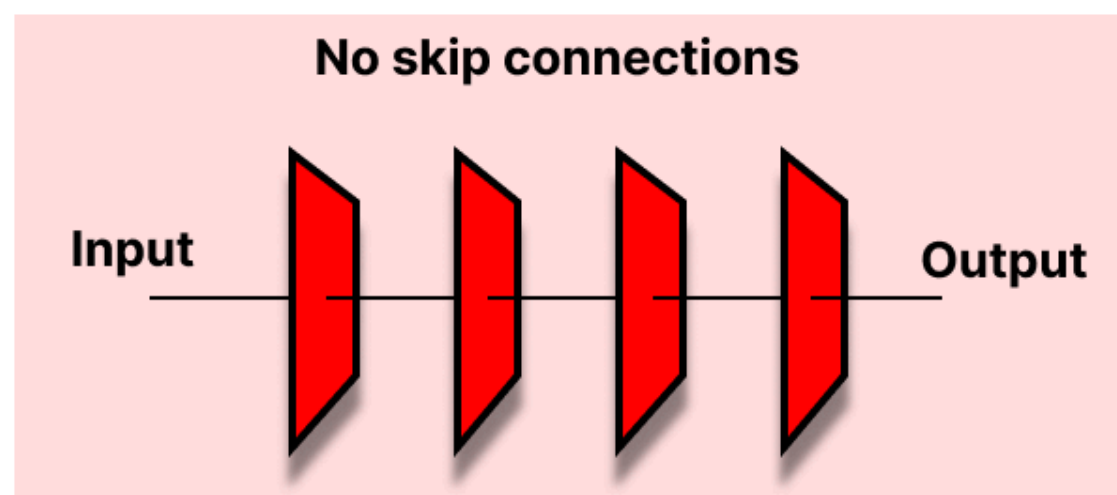
Dropout for preventing overfit



Part 2: Transformer block

Skip connection to prevent vanishing gradient

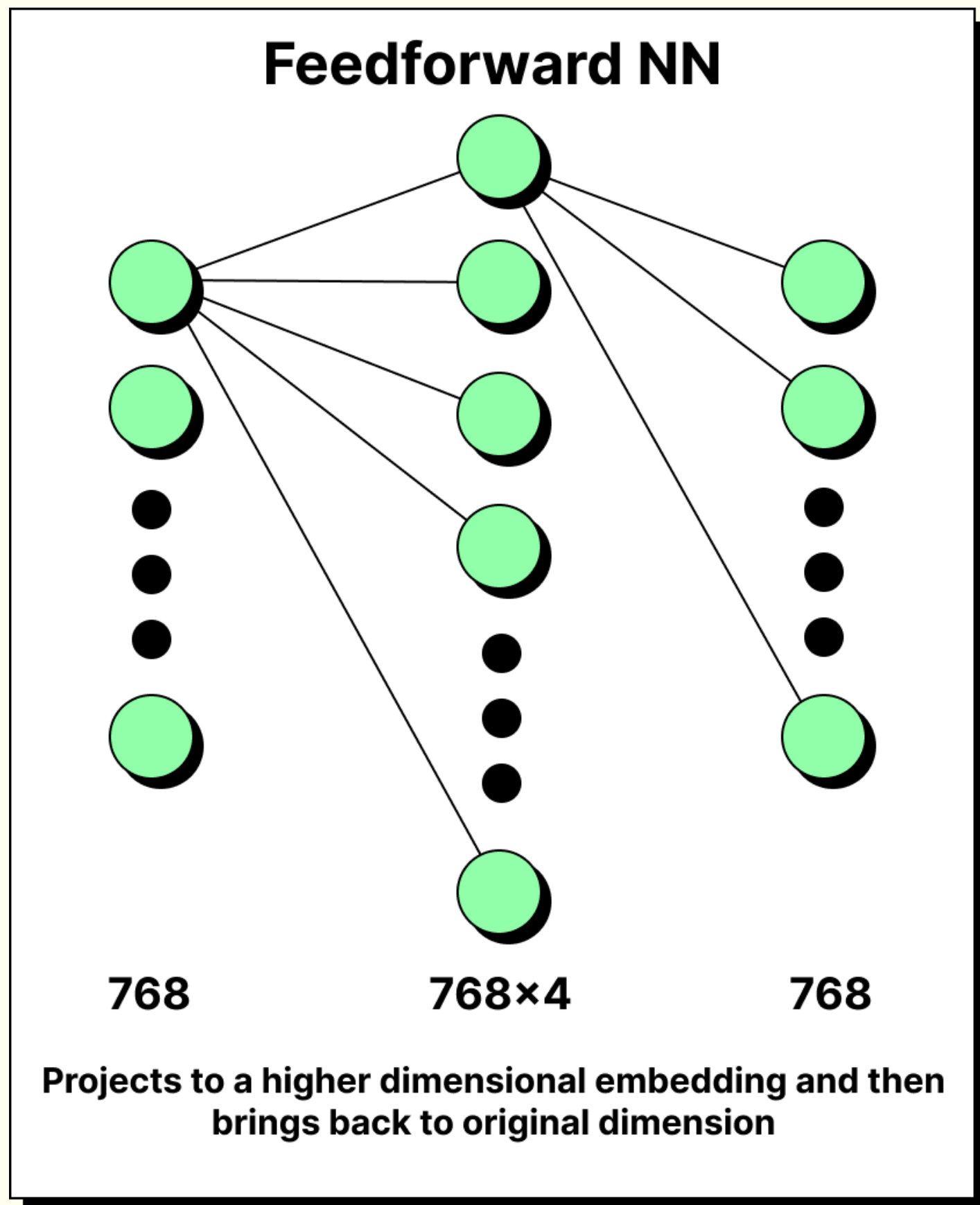
Skip connections



Solves vanishing gradient problem

Part 2: Transformer block

Feedforward NN to enhance the embedding



Part 2: Transformer block

Multihead attention



**Multi-head
attention**

Will be discussed in detail in another article

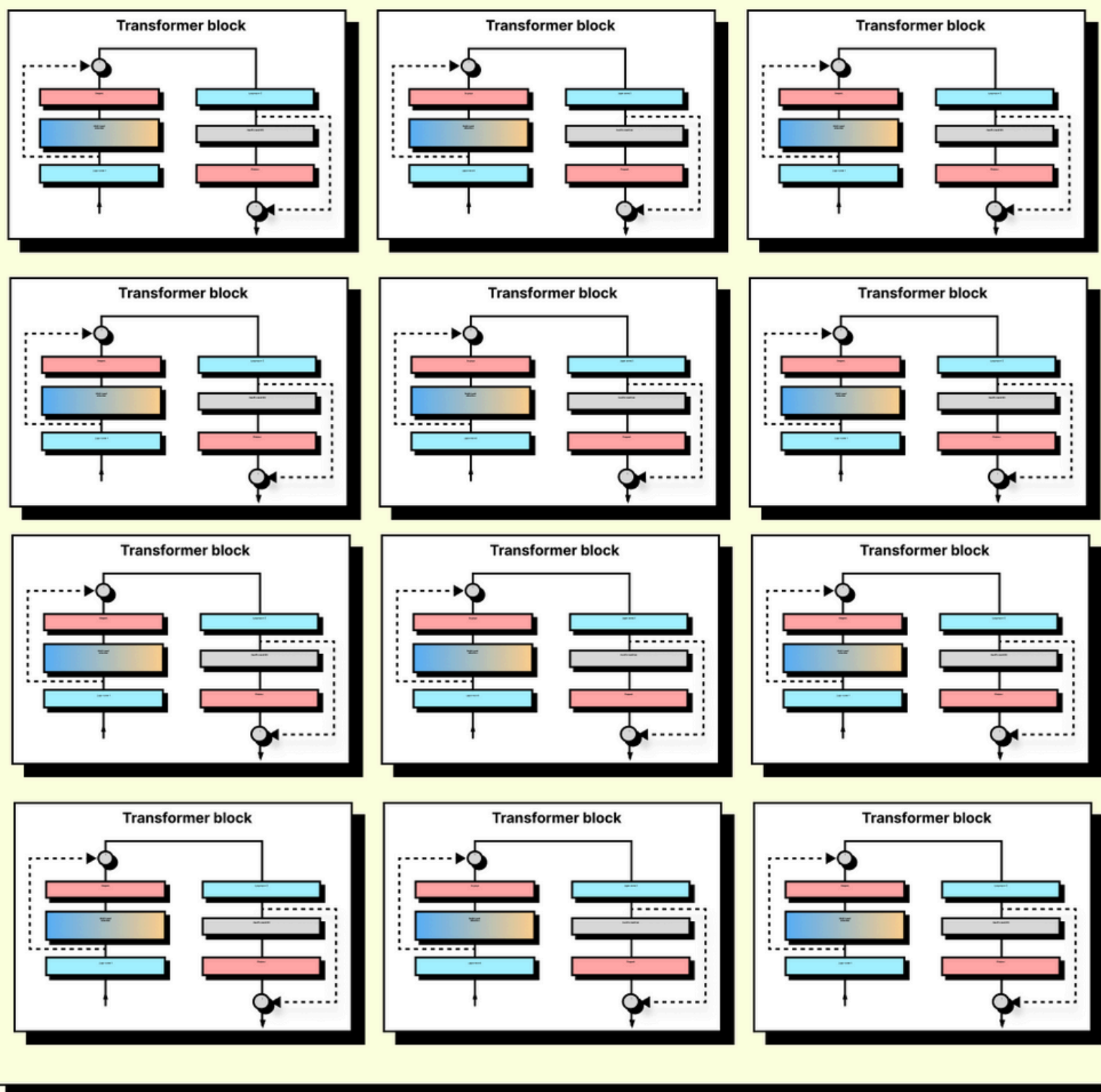


vision-transformer.vizzuara.ai

Part 2: Transformer block

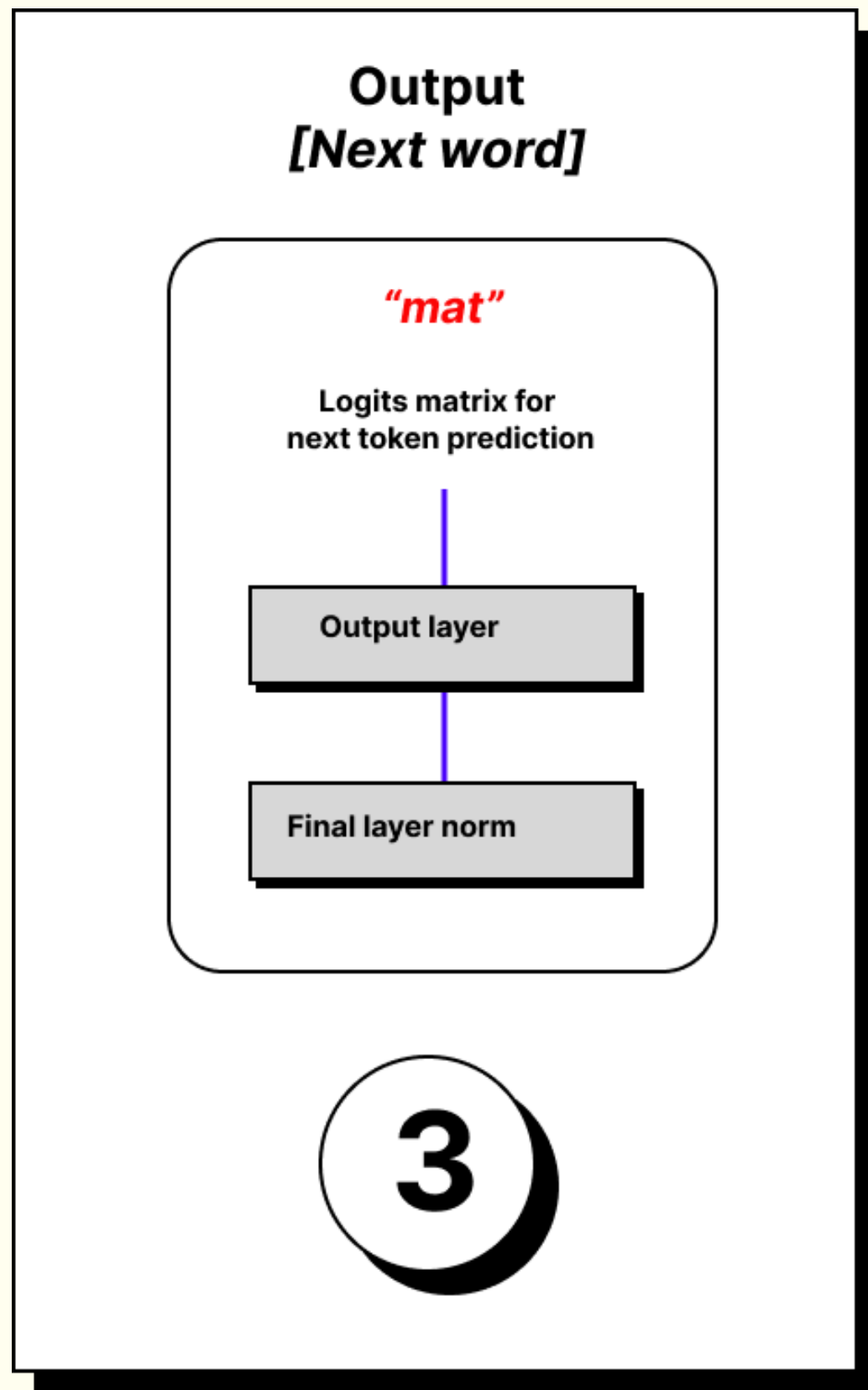
GPT-2

**Each token has to pass through
12 transformer blocks**



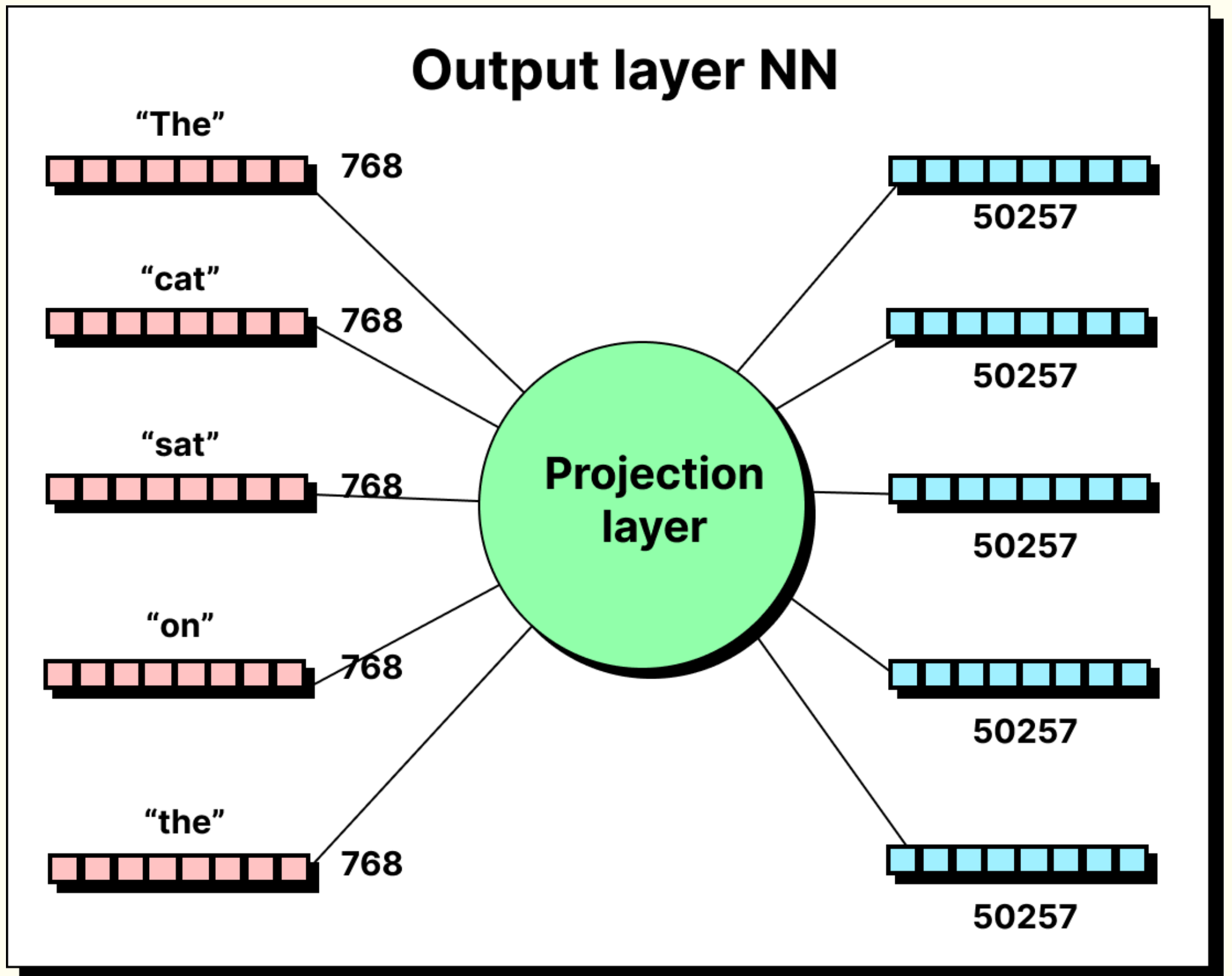
vision-transformer.vizuara.ai

Part 3: Output



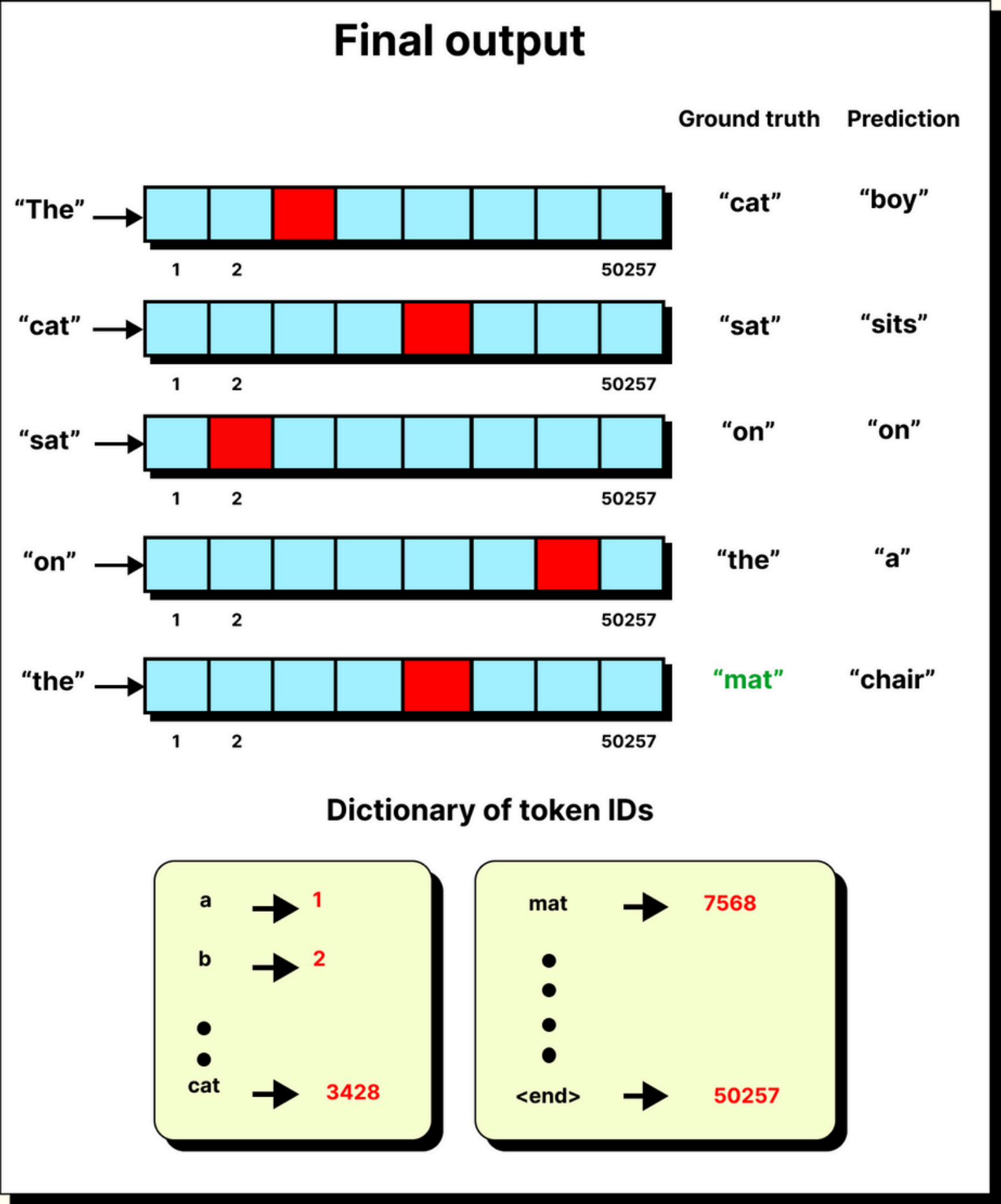
Part 3: Output

Projection layer NN



Part 3: Output

Final output



Overall

