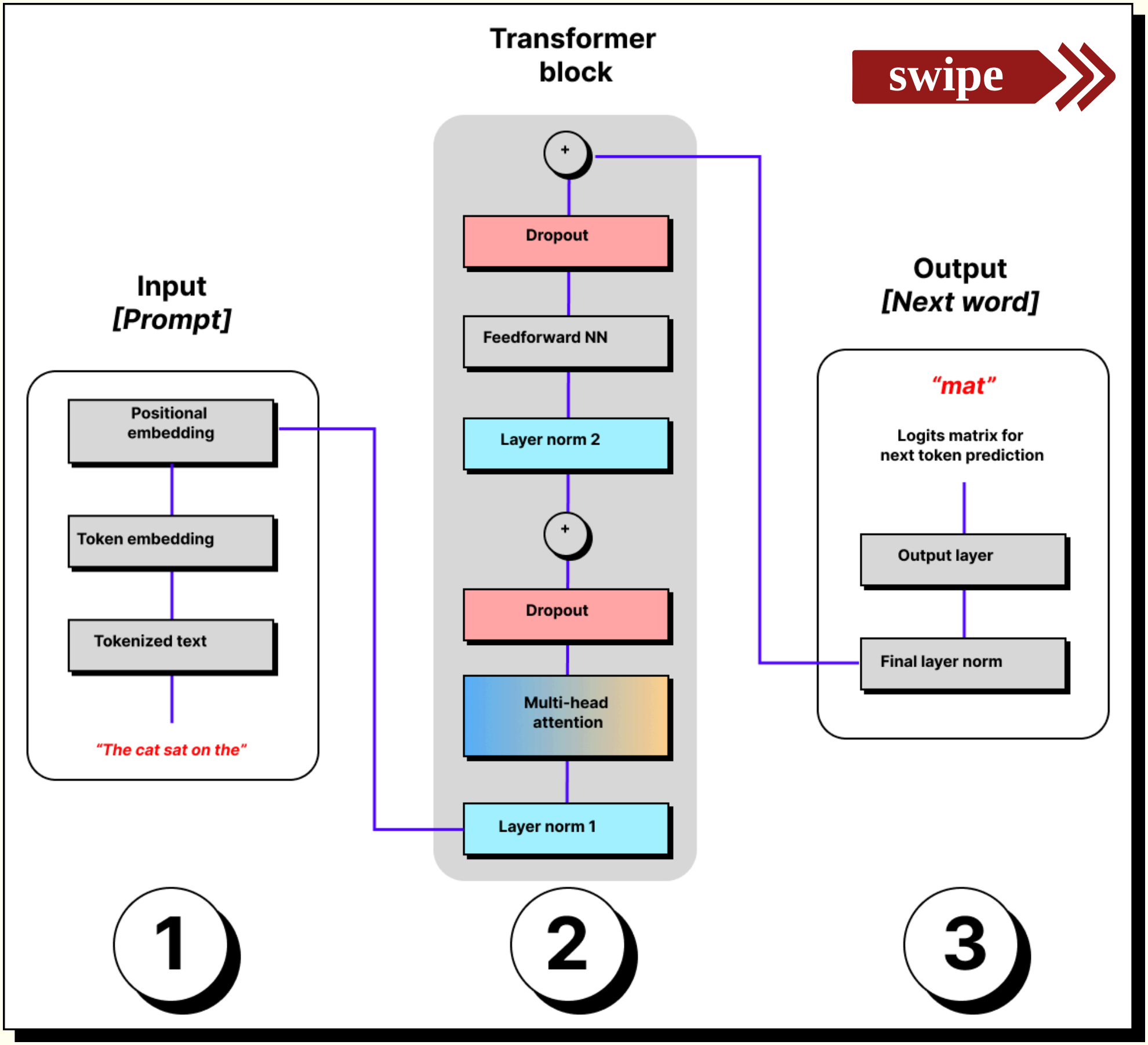
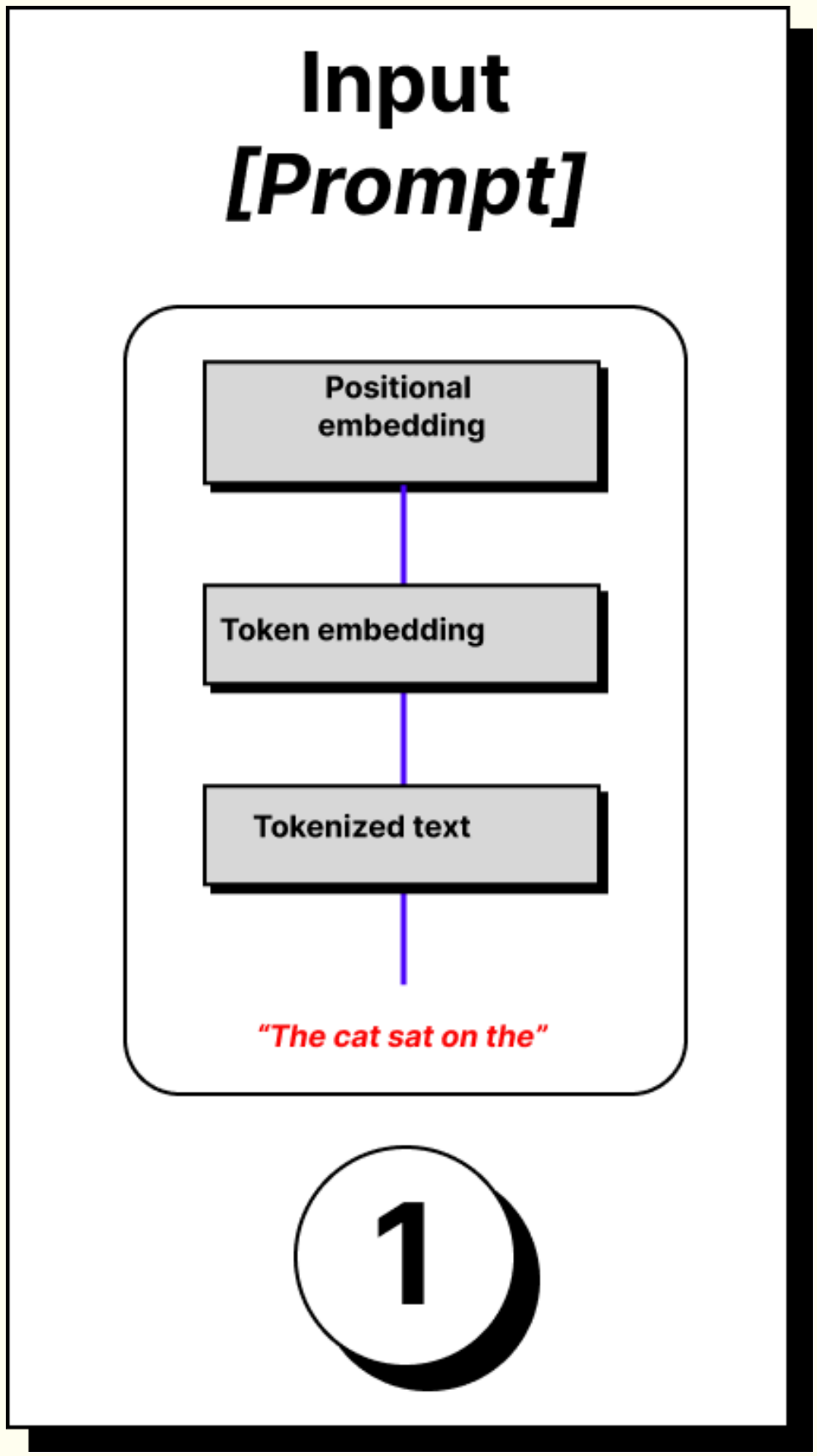
Transformer architecture

In 3 parts



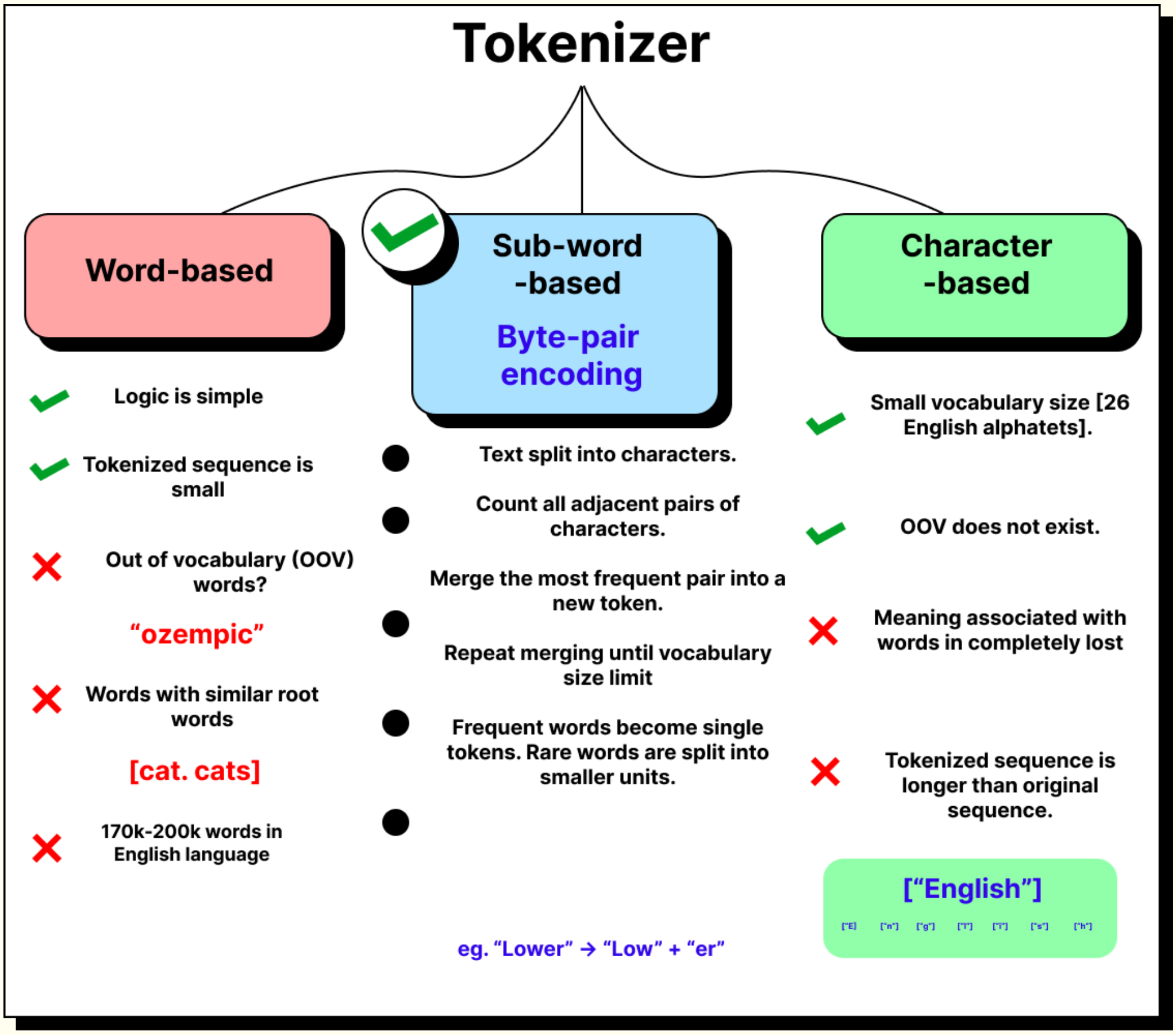
vision-transformer.vizuara.ai

Part 1: Input


Input
[Prompt]
Positional embedding
Token embedding
Tokenized text
"The cat sat on the"
1

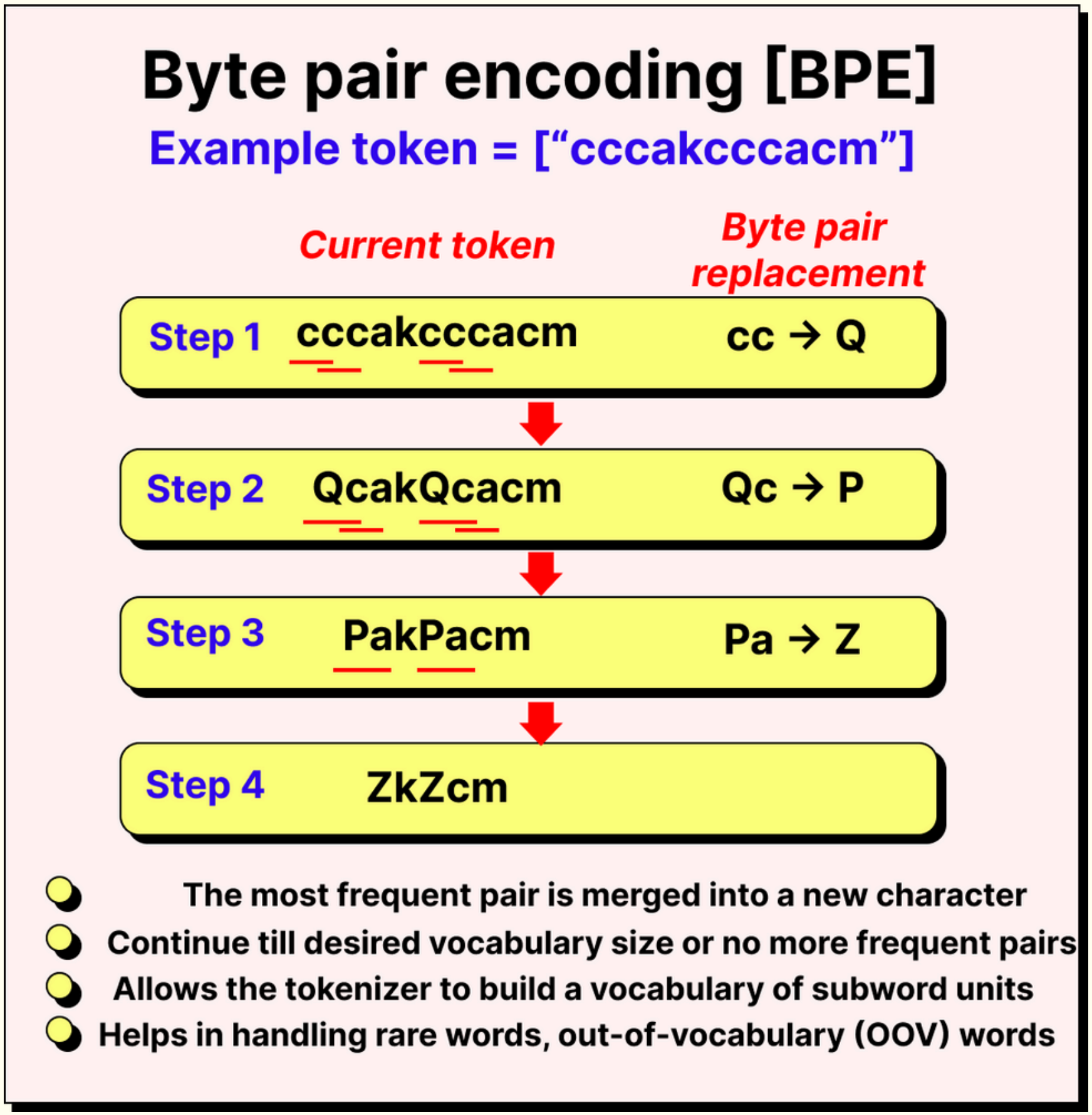
Part 1: Input

Tokenizer options



Part 1: Input

Tokenizer used

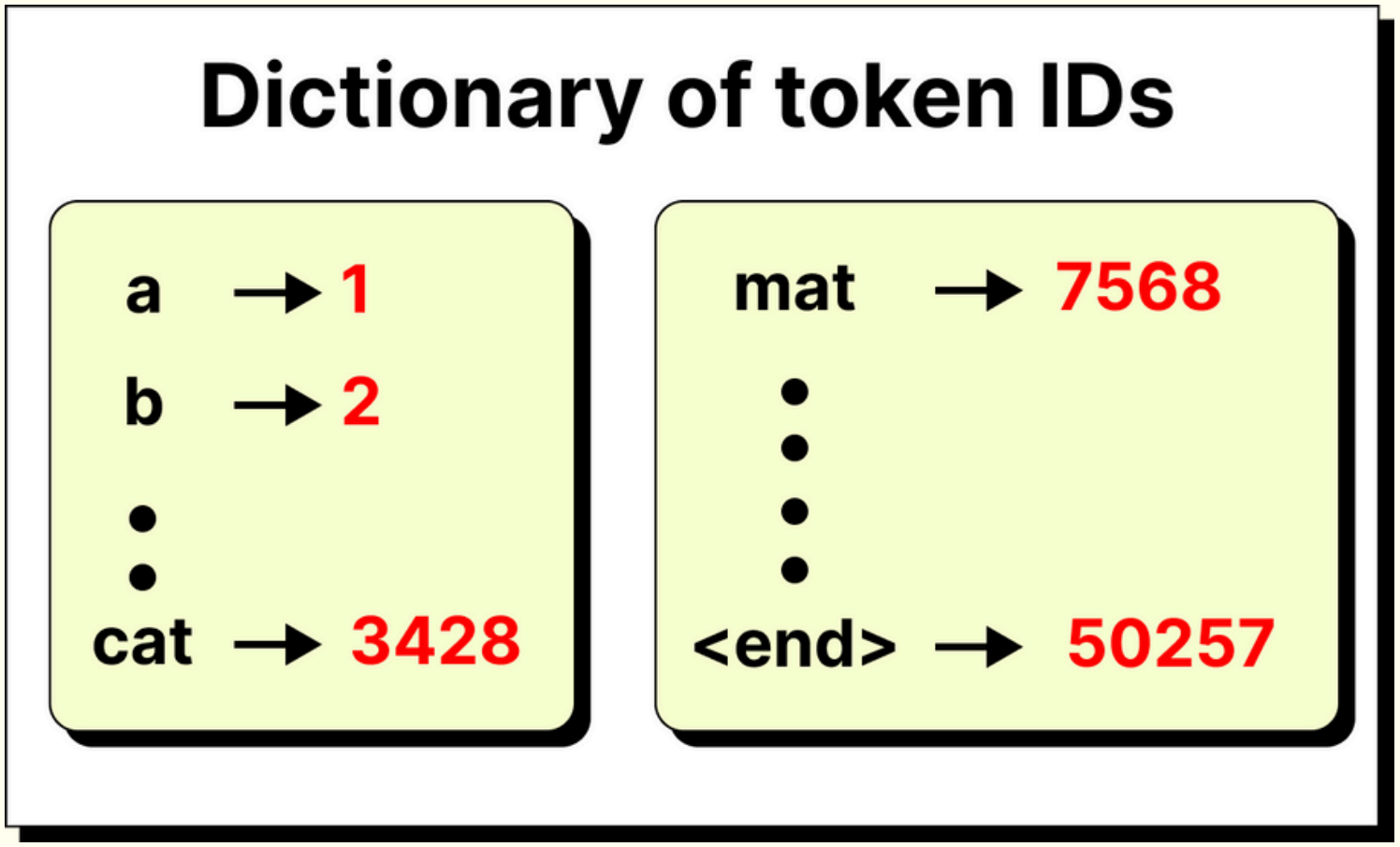


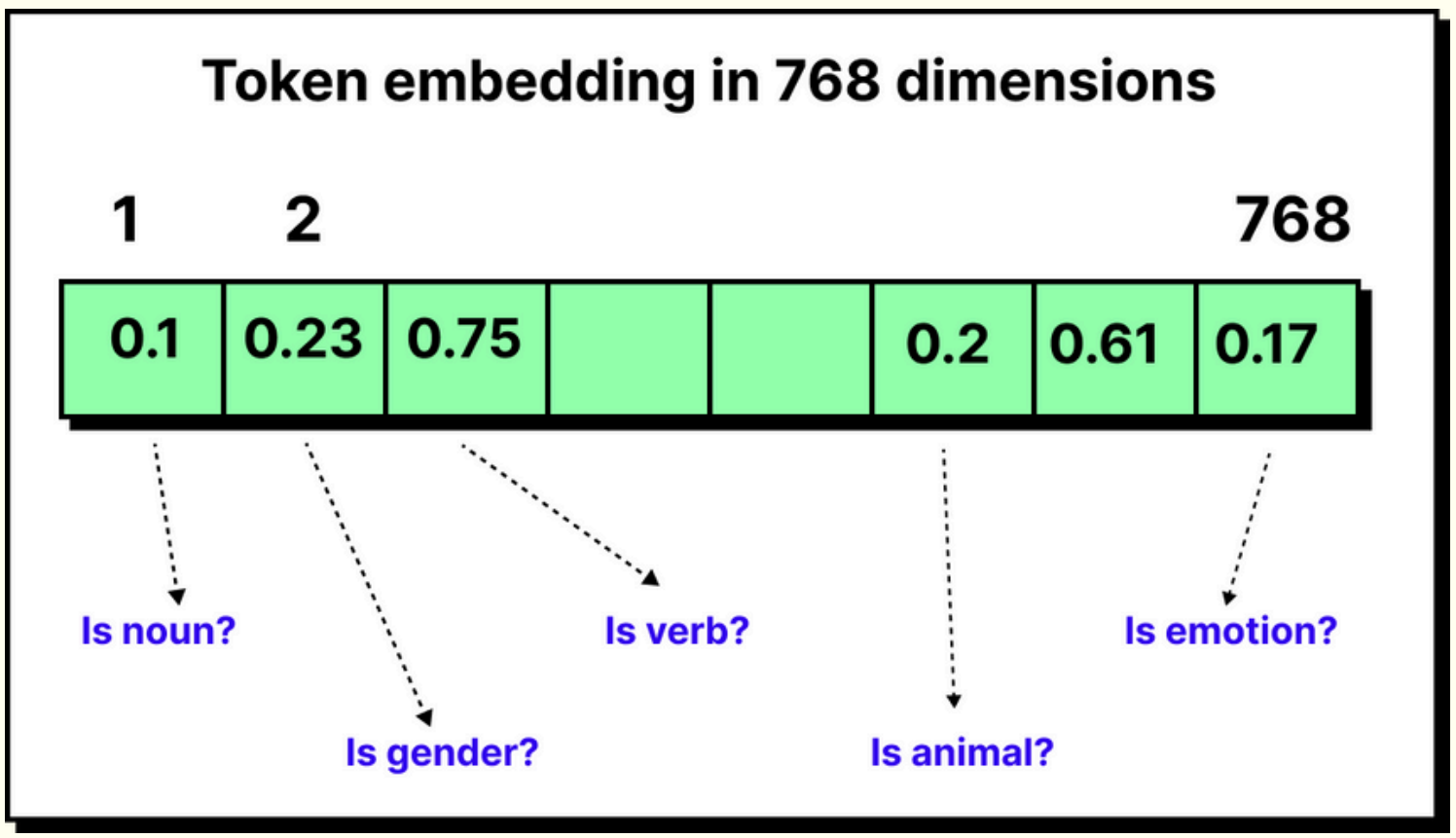
Part 1: Input

Simplified tokenization example

Part 1: Input

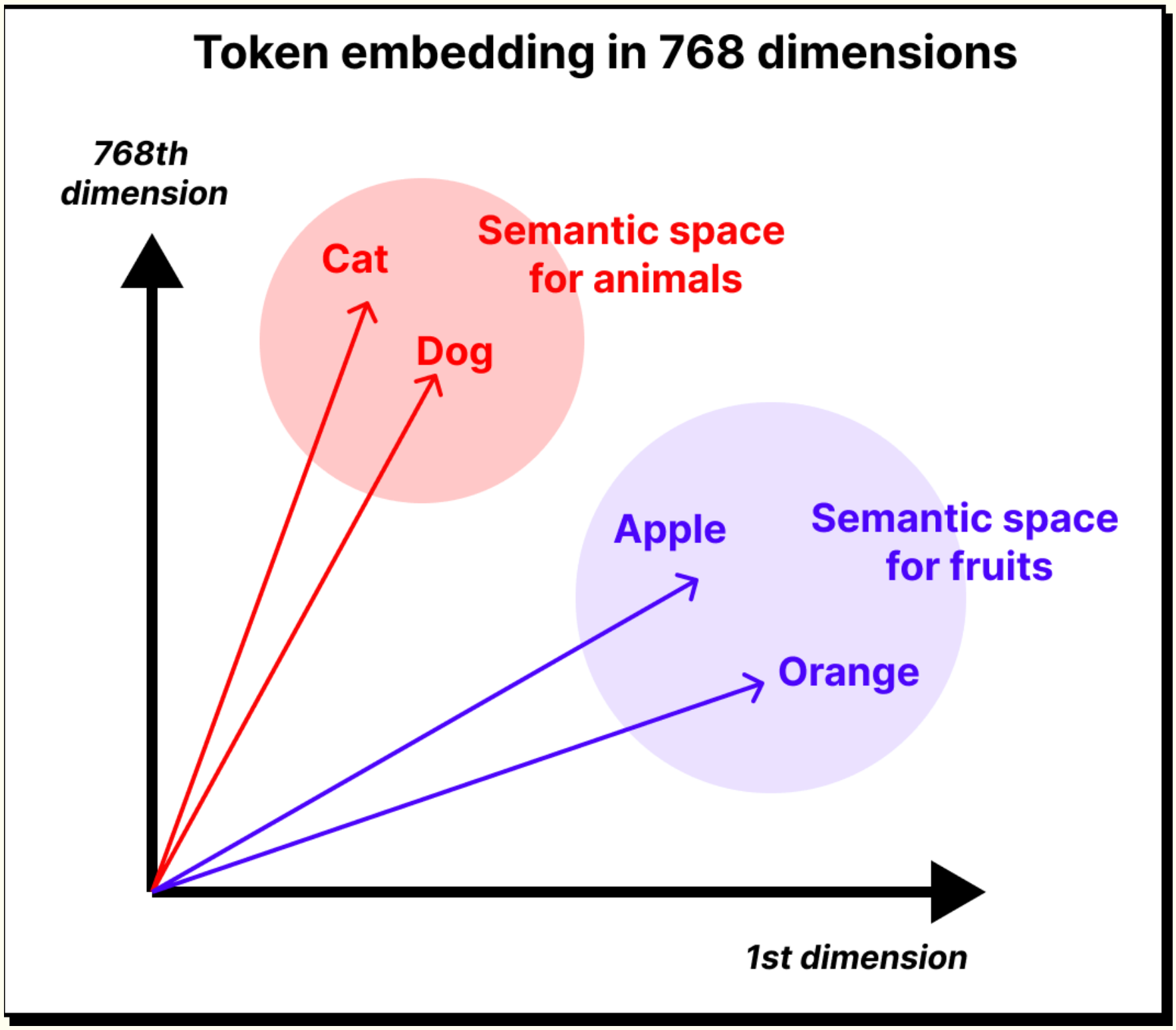
Identify token vectors from vocabulary





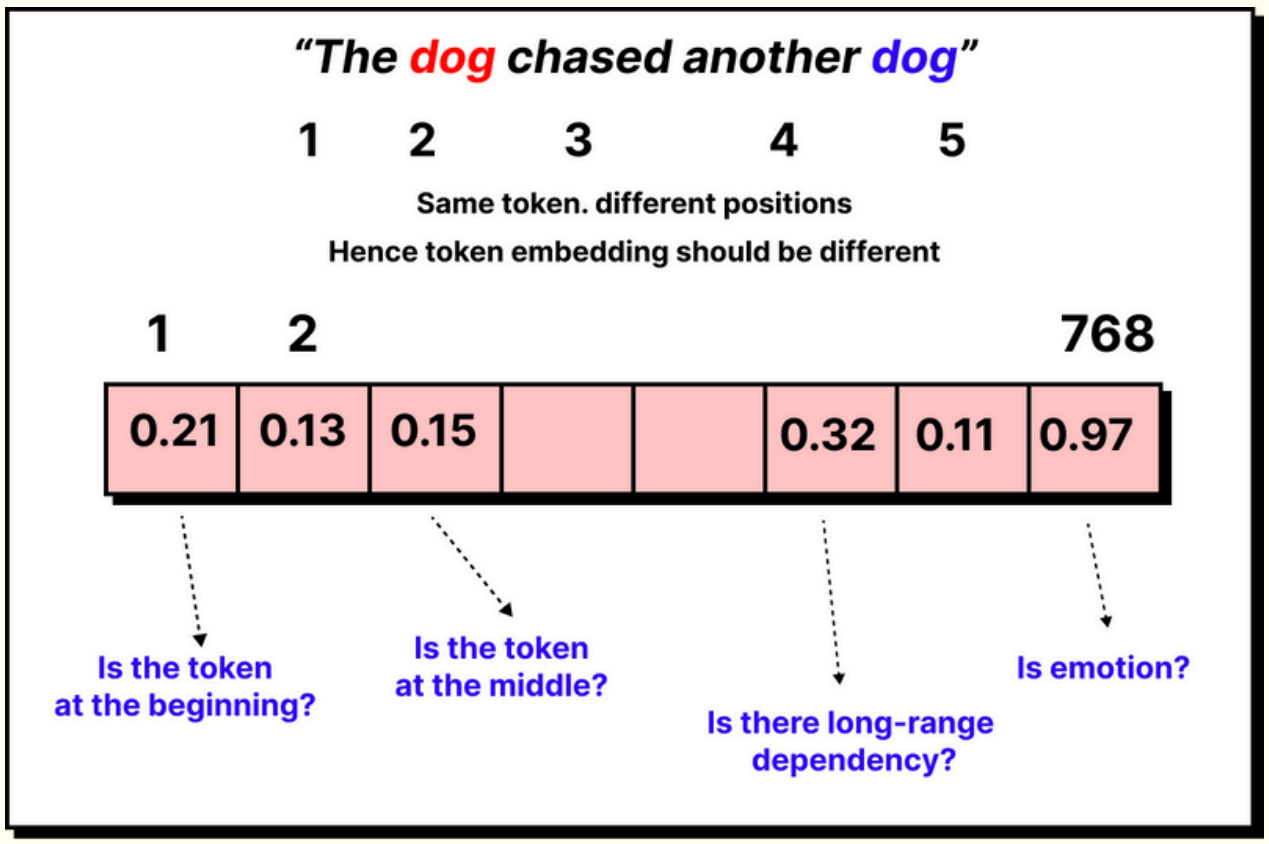
Part 1: Input

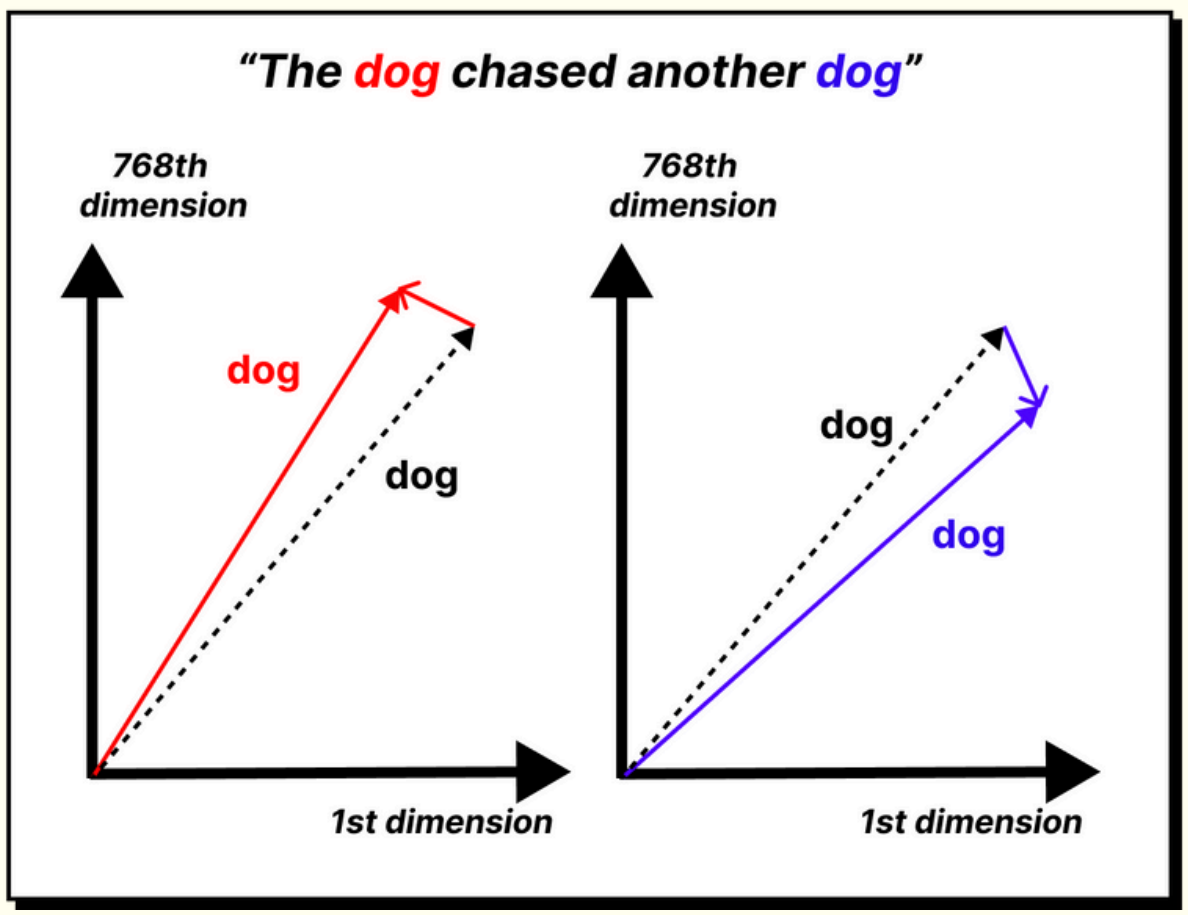
Token vectors carry semantic meaning



Part 1: Input

It is important to consider token position

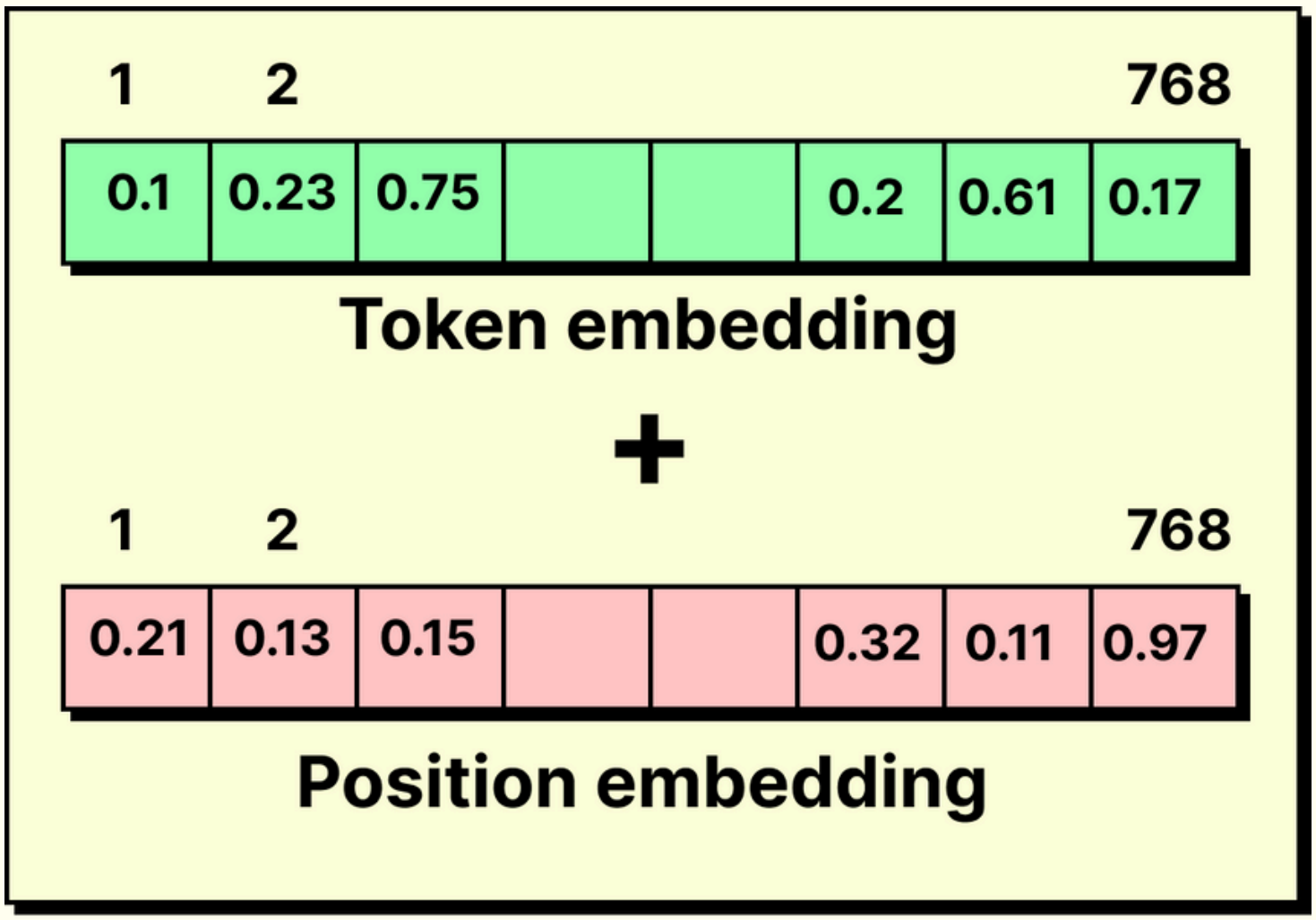


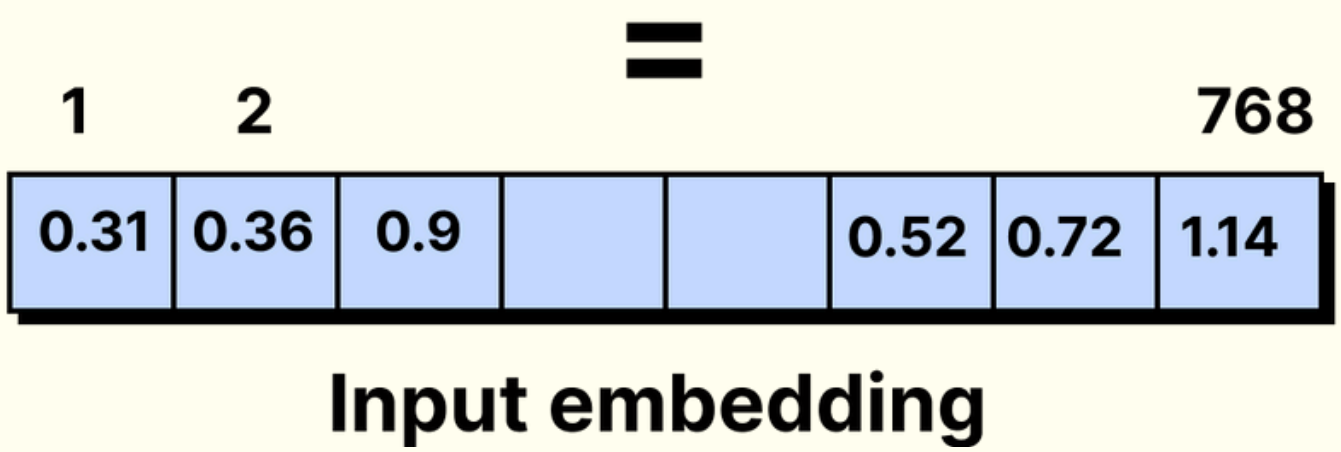


vision-transformer.vizuara.ai

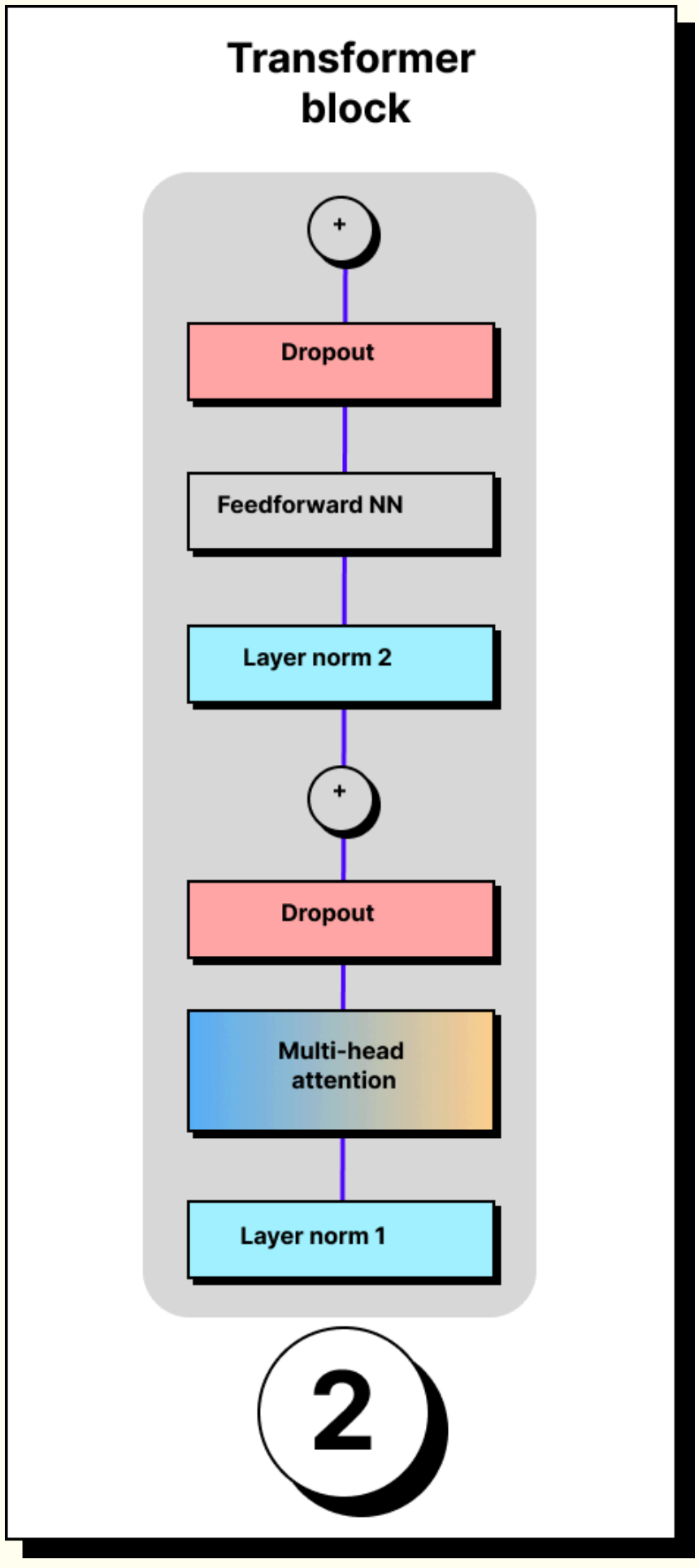
Part 1: Input

Add position embedding to create input embedding



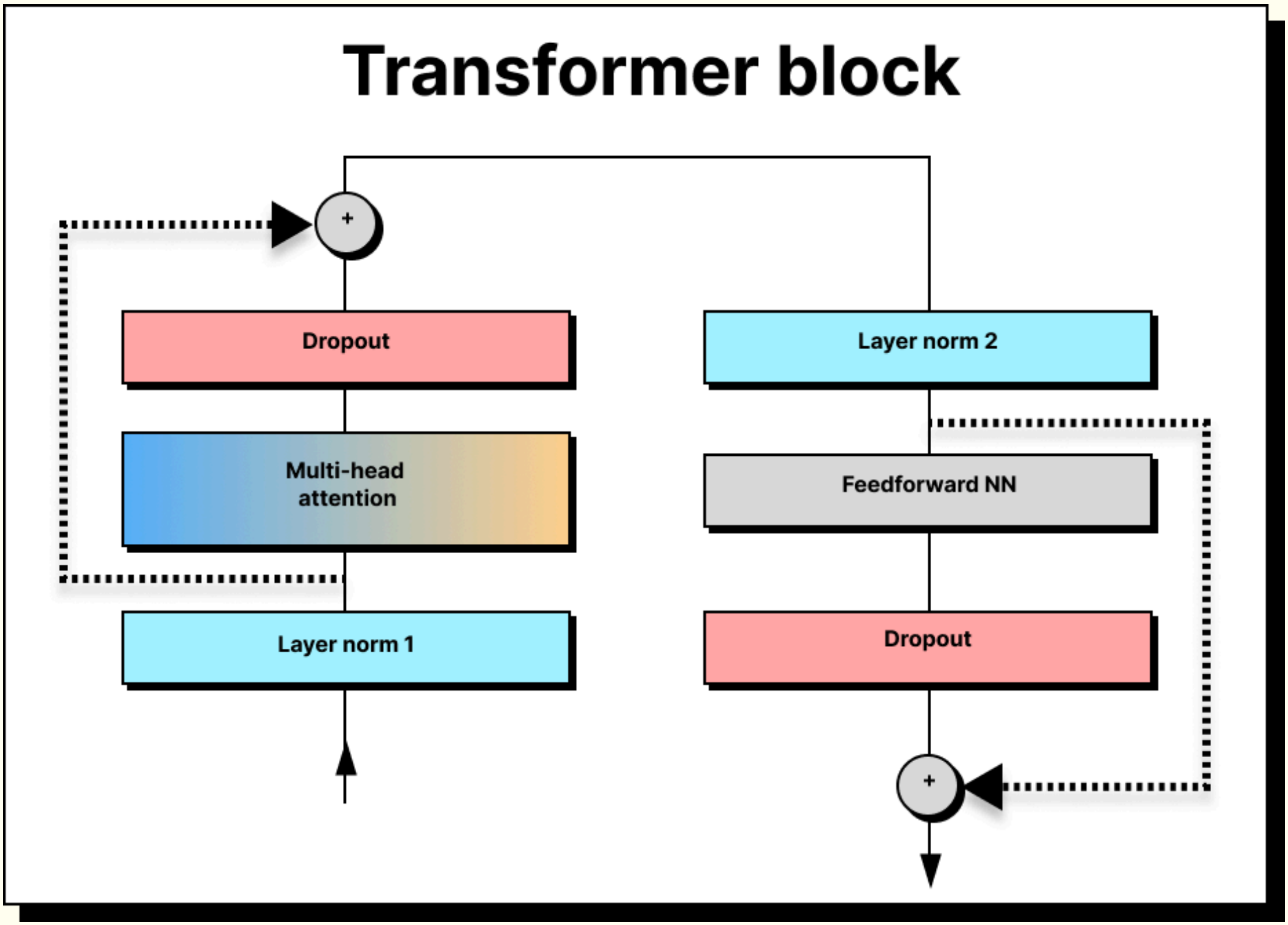


Part 2: Transformer block



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]

mean(x)=[2.0+(−1.0)+3.0+0.0]/4=4.0/4=1.0

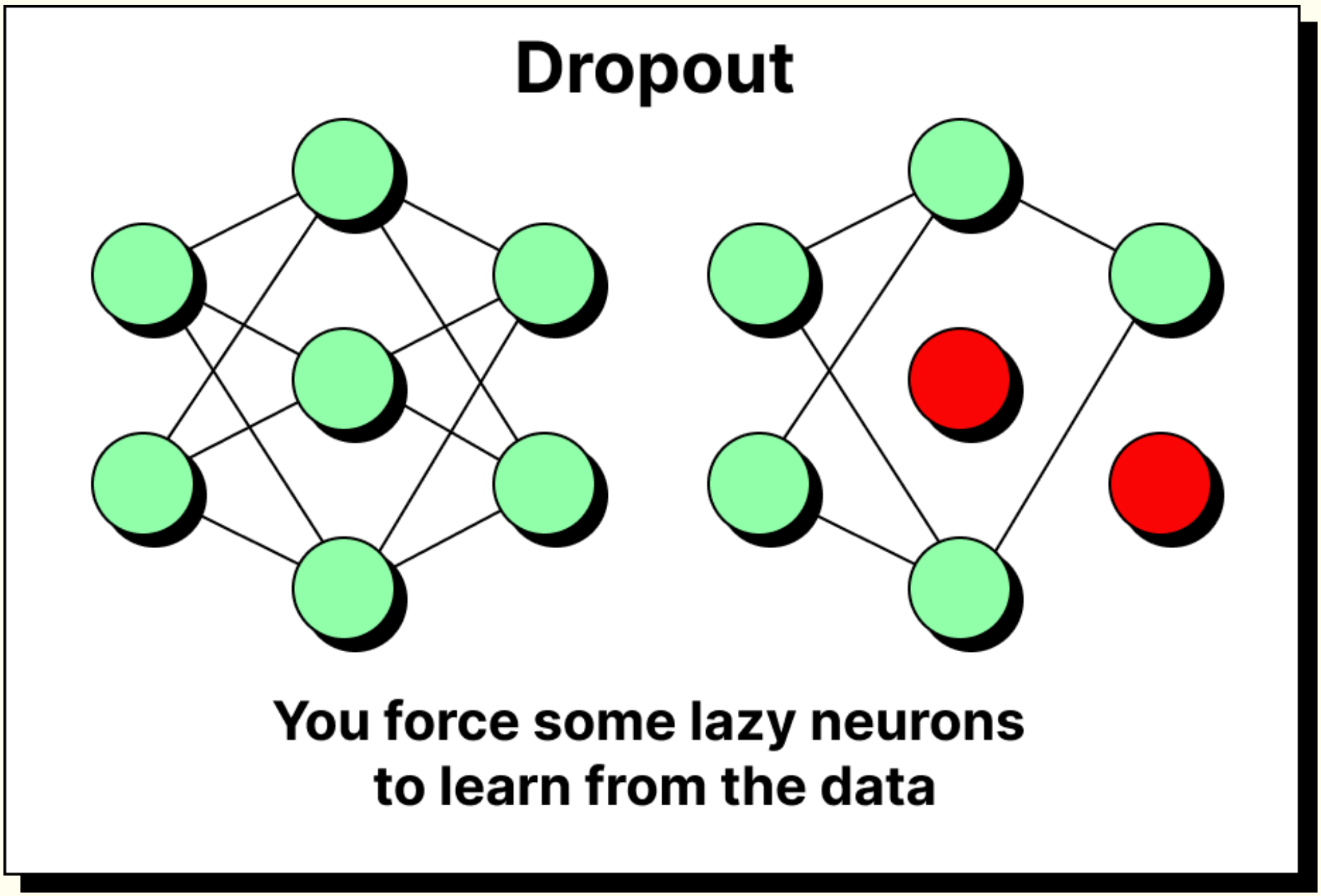
variance(x)=[(2−1)^2+(−1−1)^2+(3−1)^2+(0−1)^2]/4
=[1+4+4+1]/4=10/4=2.5

std(x)=sqrt(2.5)≈1.58

x = [x-mean(x)]/std(x) ≈ [0.63.−1.26.1.26.−0.63]

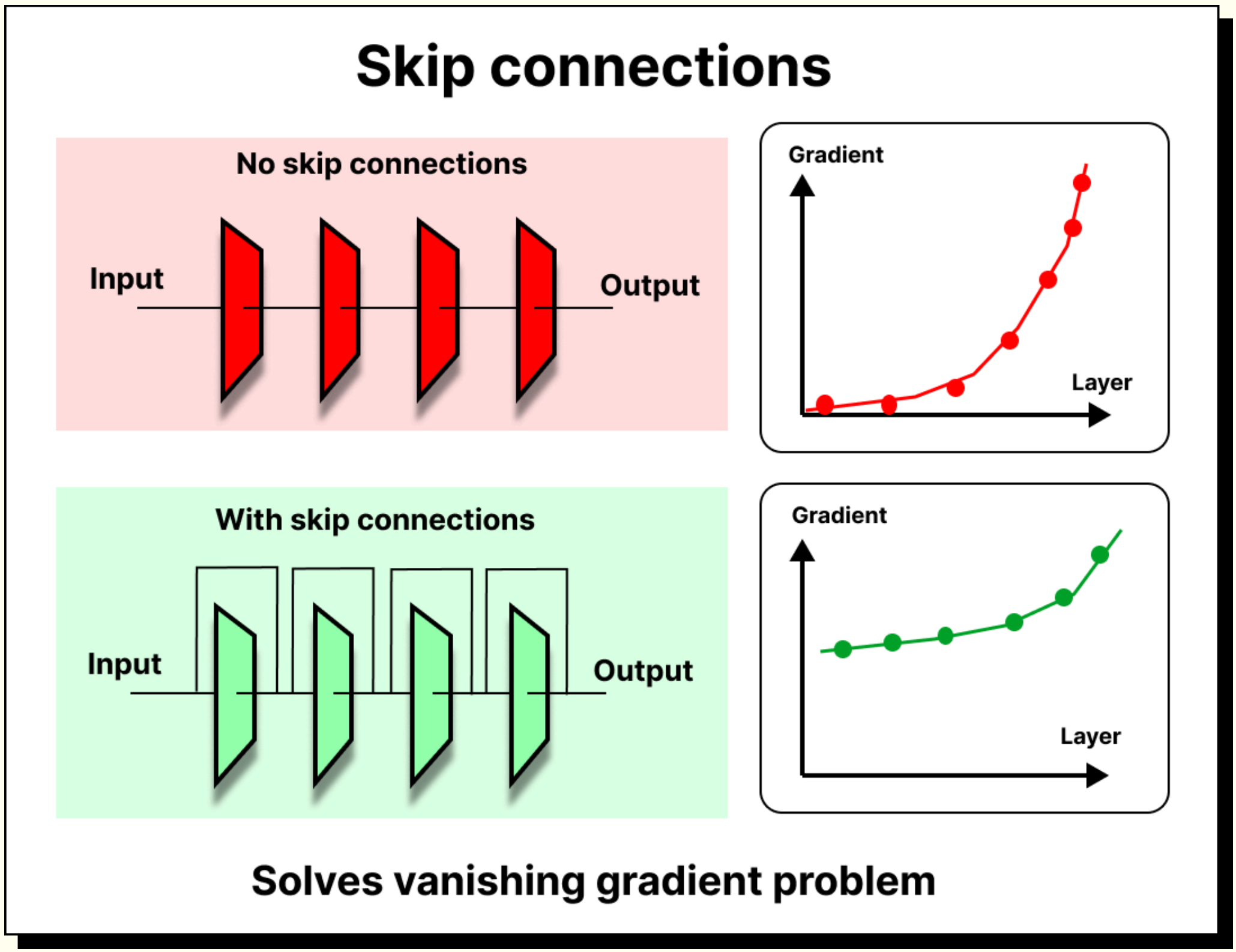
Part 2: Transformer block

Dropout for preventing overfit



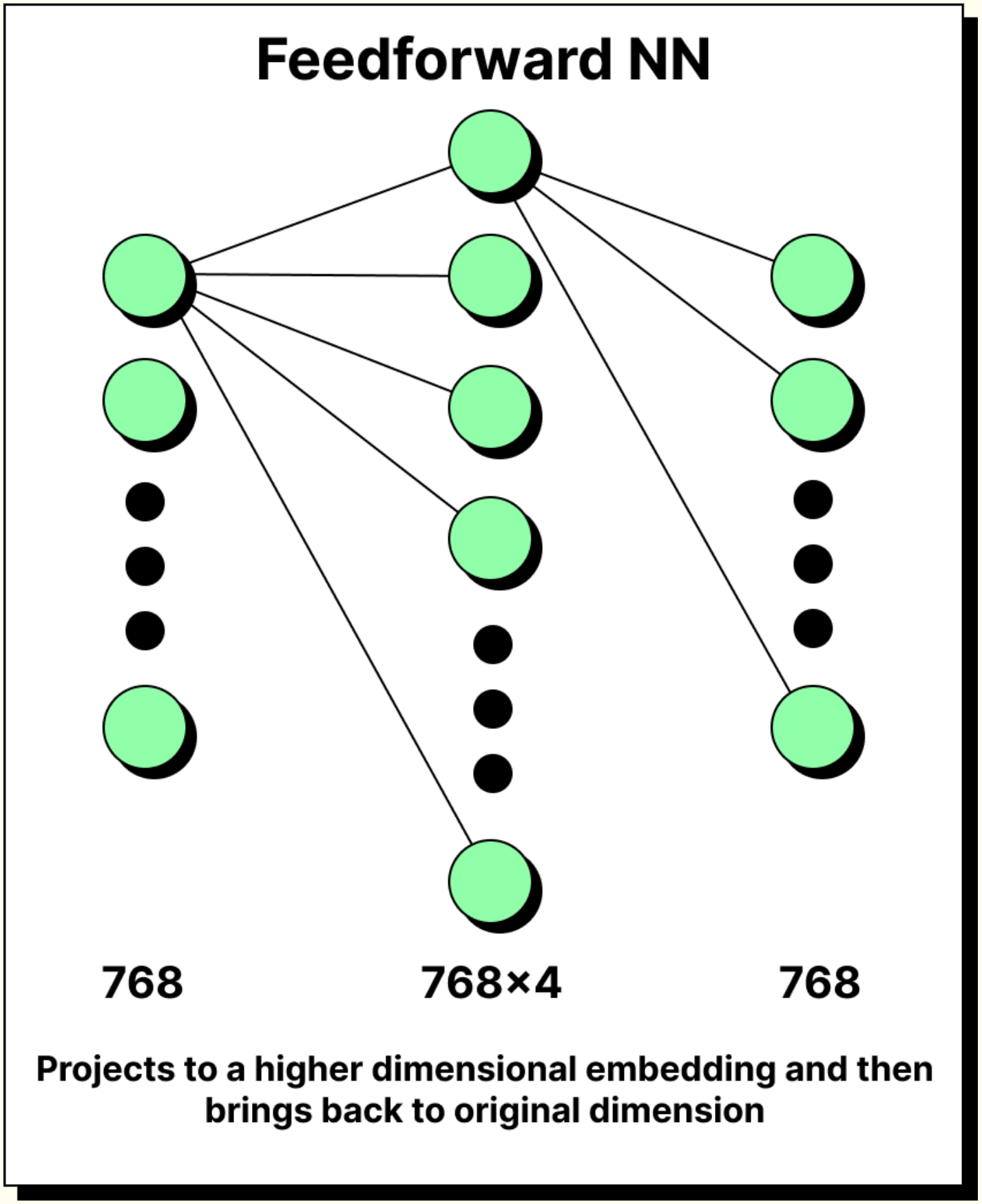
Part 2: Transformer block

Skip connection to prevent vanishing gradient



Part 2: Transformer block

Feedforward NN to enhance the embedding



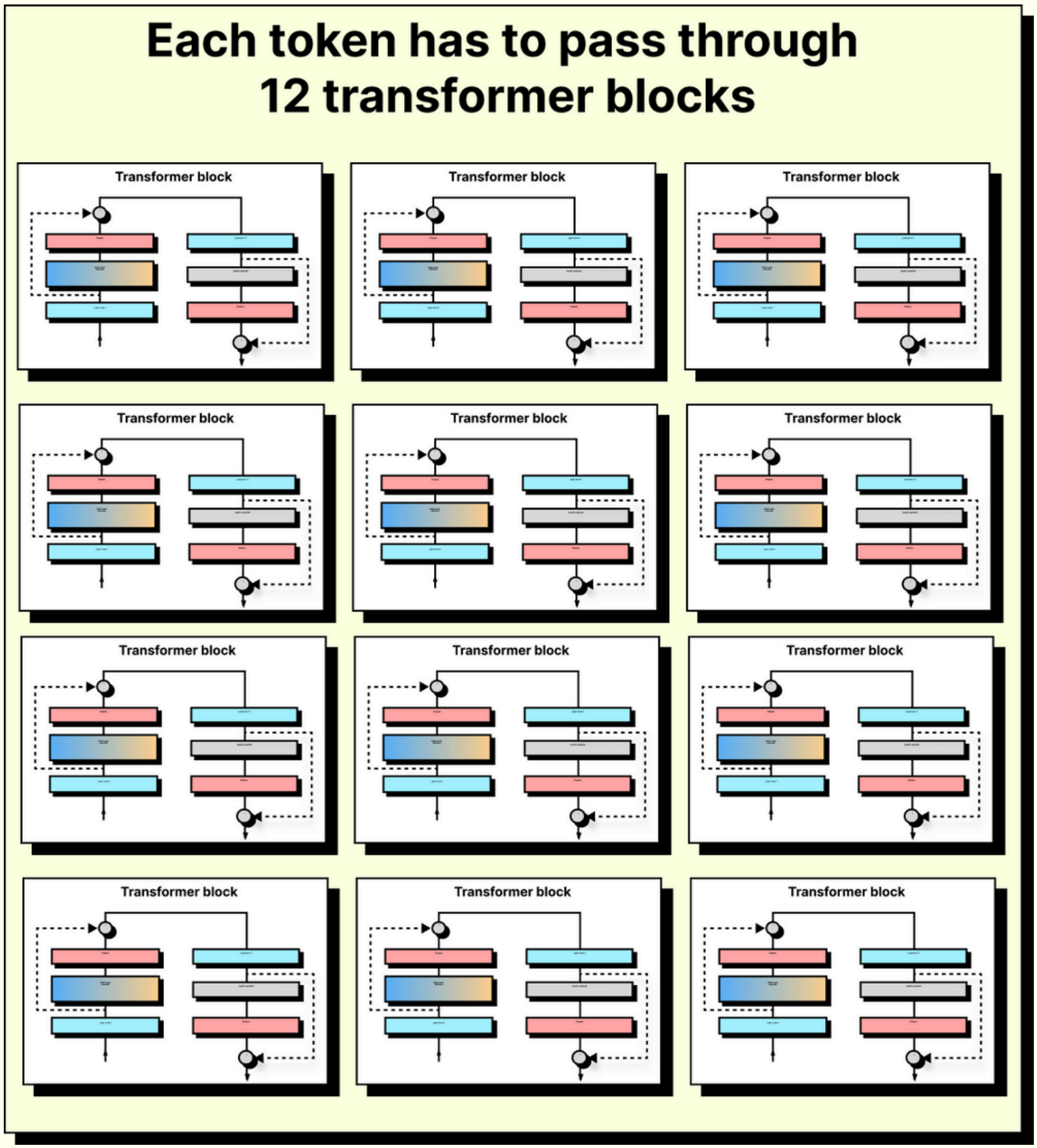
Part 2: Transformer block

Multihead attention

Will be discussed in detail in another article

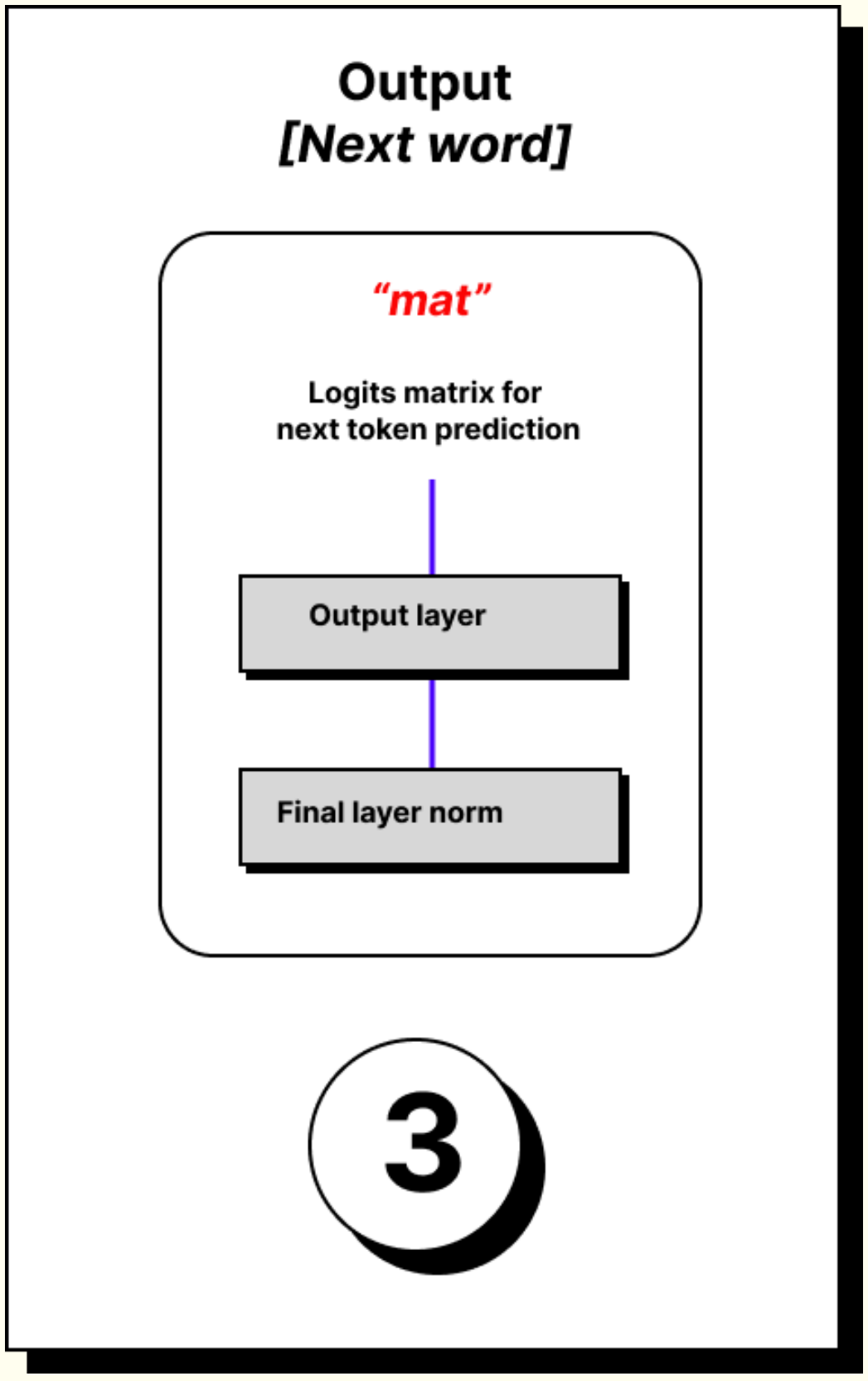
Part 2: Transformer block

GPT-2



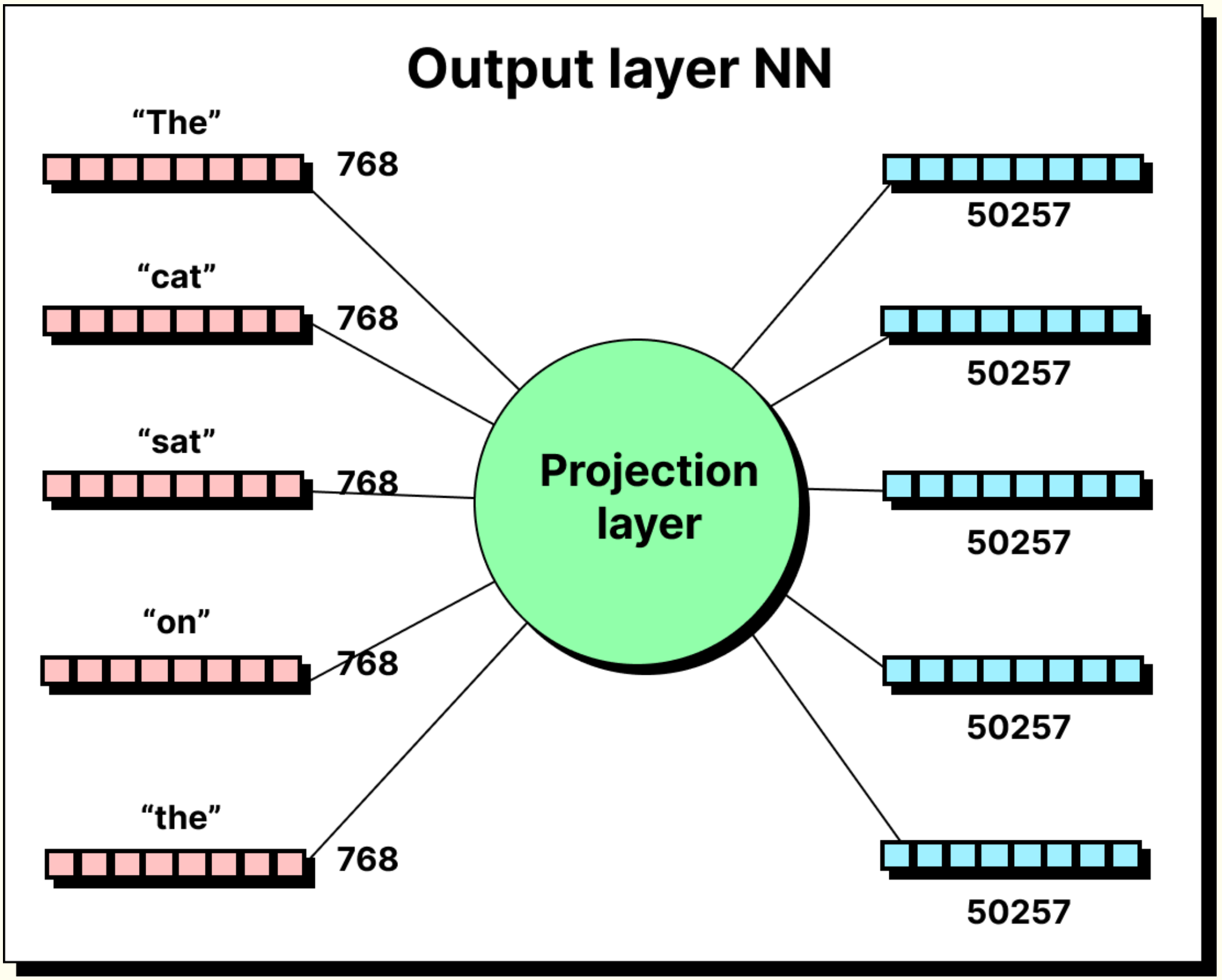
vision-transformer.vizuara.ai

Part 3: Output



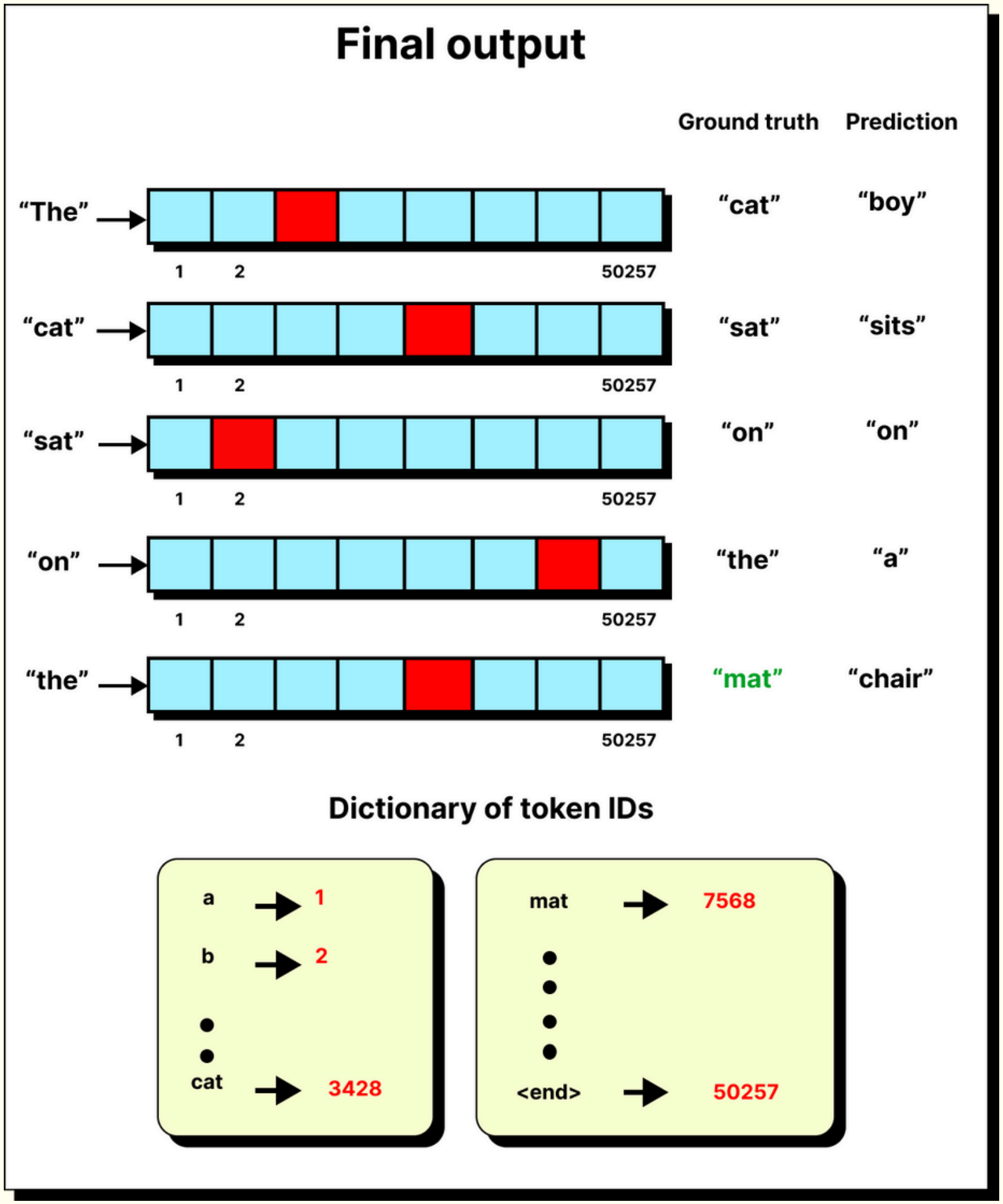
Part 3: Output

Projection layer NN

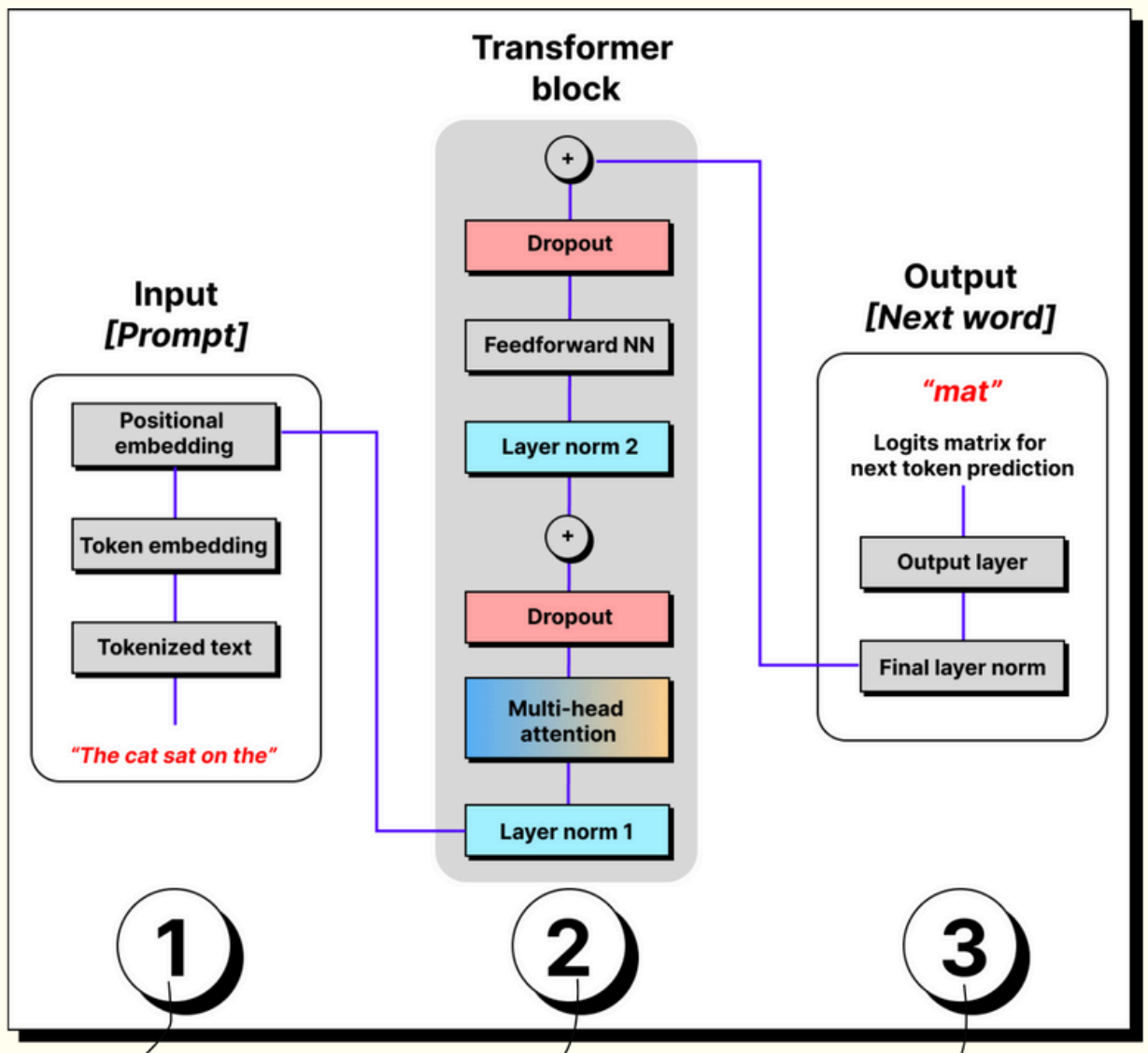


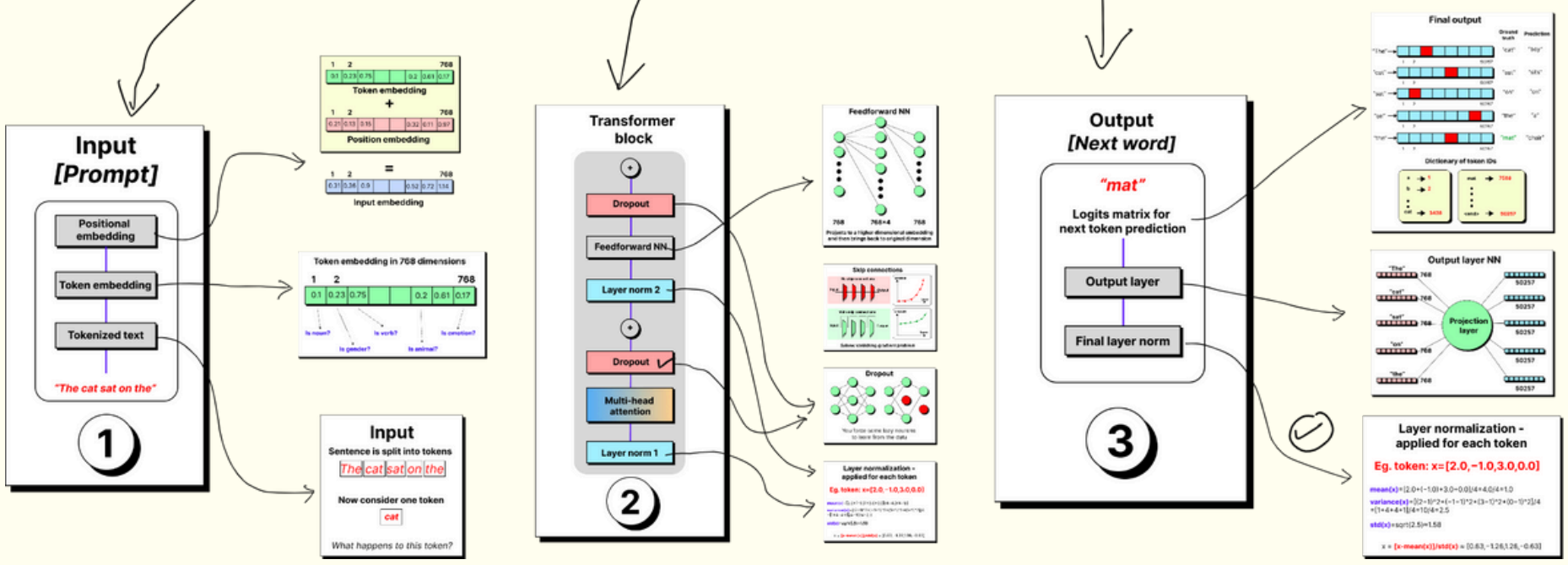
Part 3: Output

Final output



Overall





vision-transformer.vizuara.ai