Generative AI

2025 / 11 / 09 – Part 1

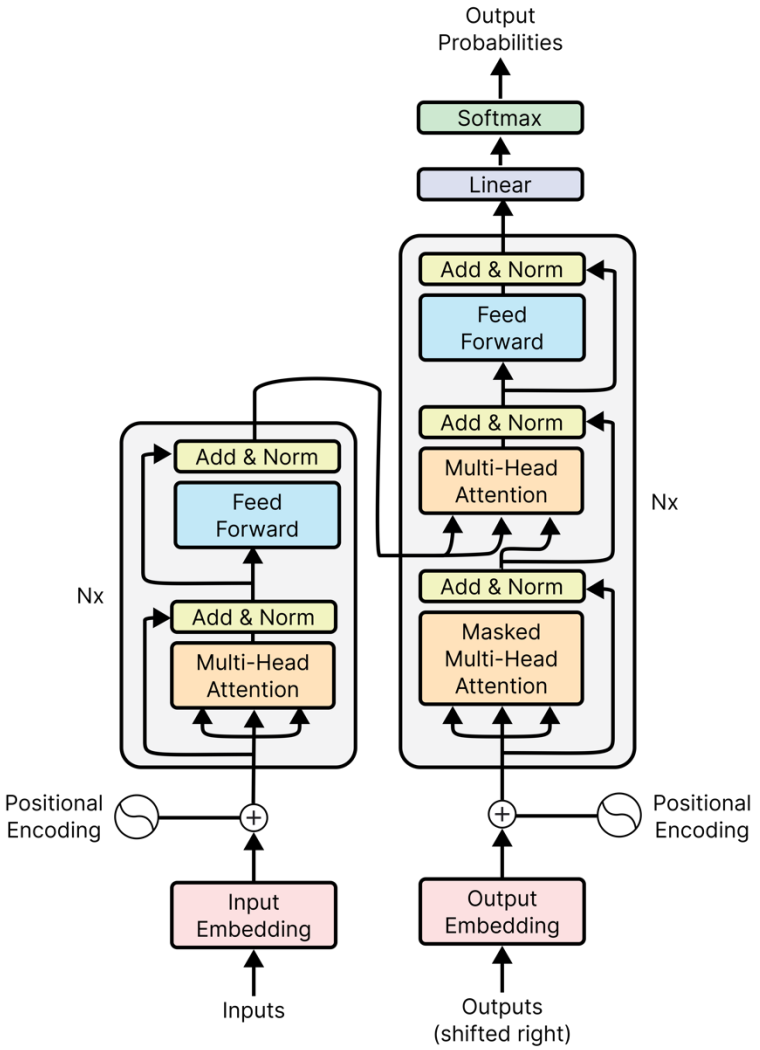
Training a Large Language Model

Johannes Oster

Recap of the

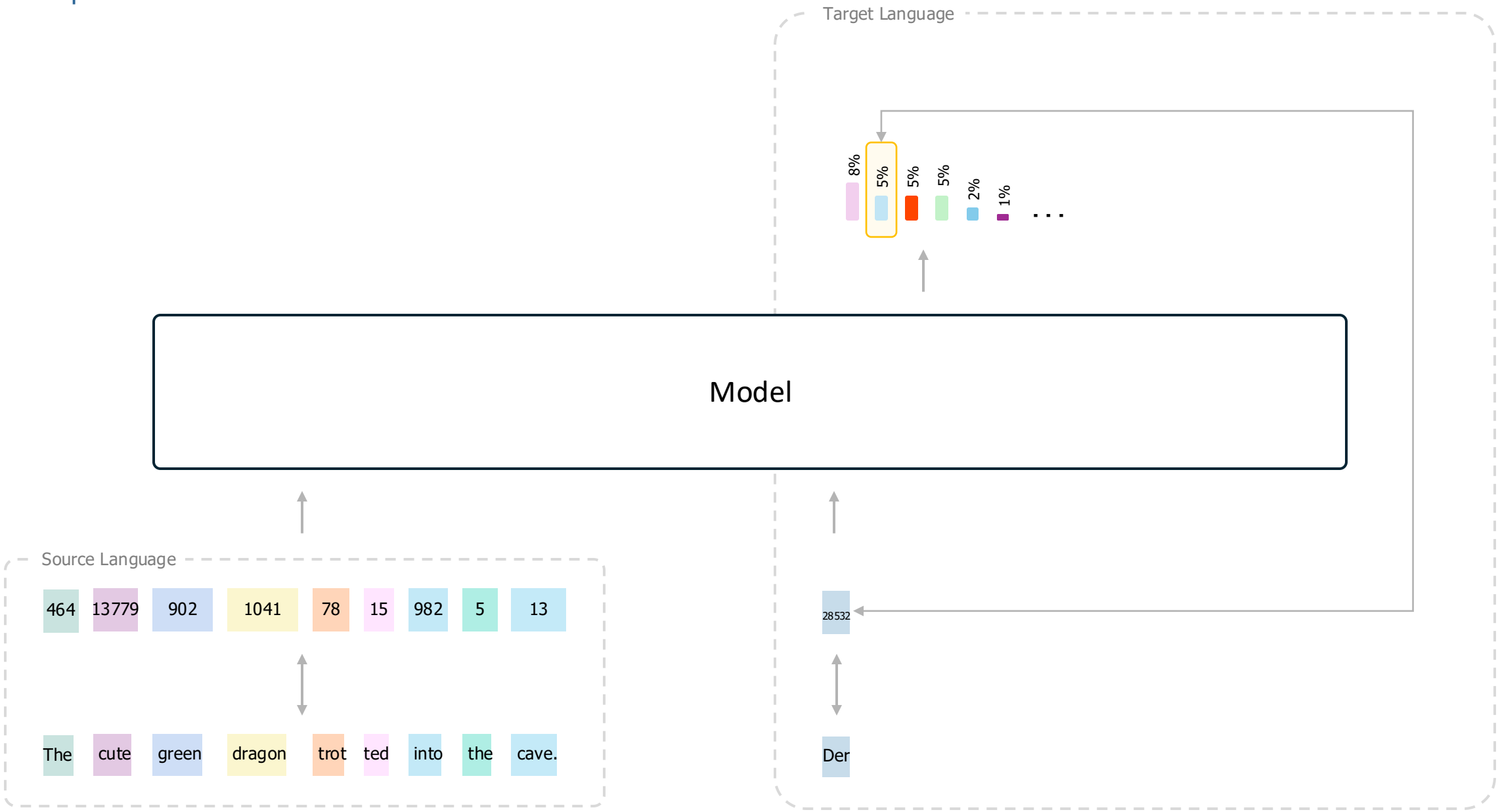
Transformer Architecture.

With reference to 'Attention Is All You Need' by A. Vaswani et al.[1]

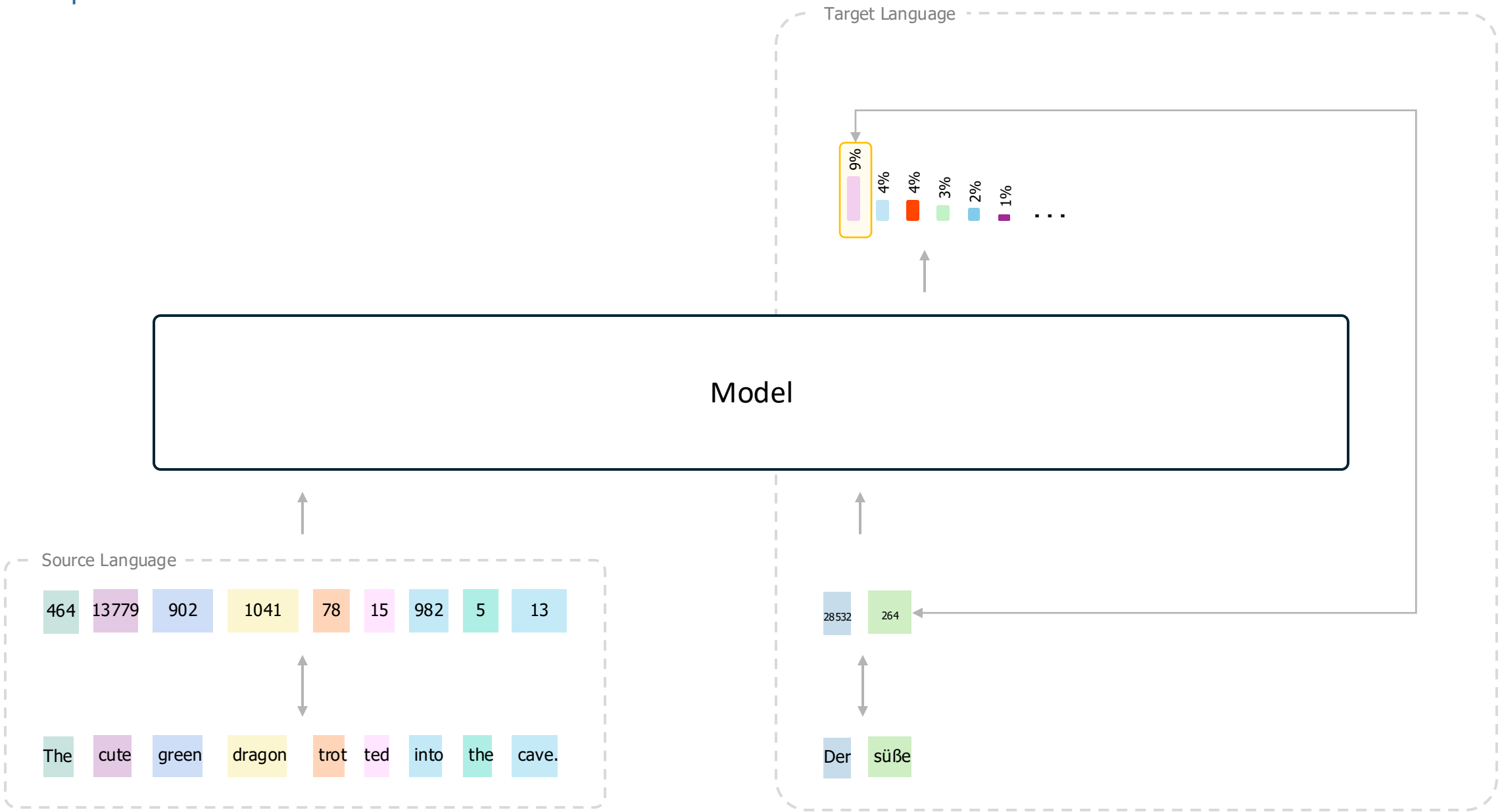


1. https://papers.nips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

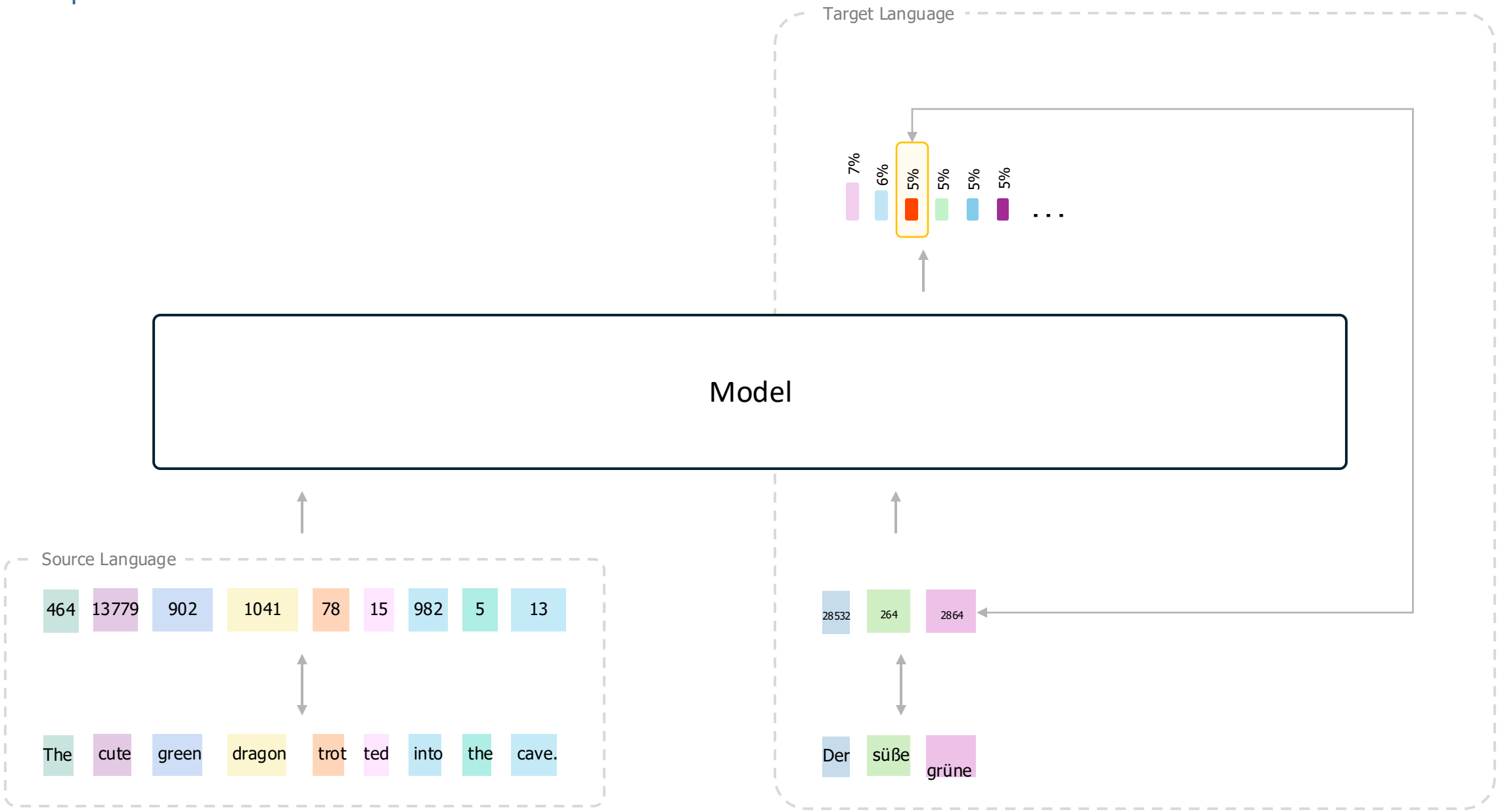
Recap: Transformer Architecture for Machine Translation



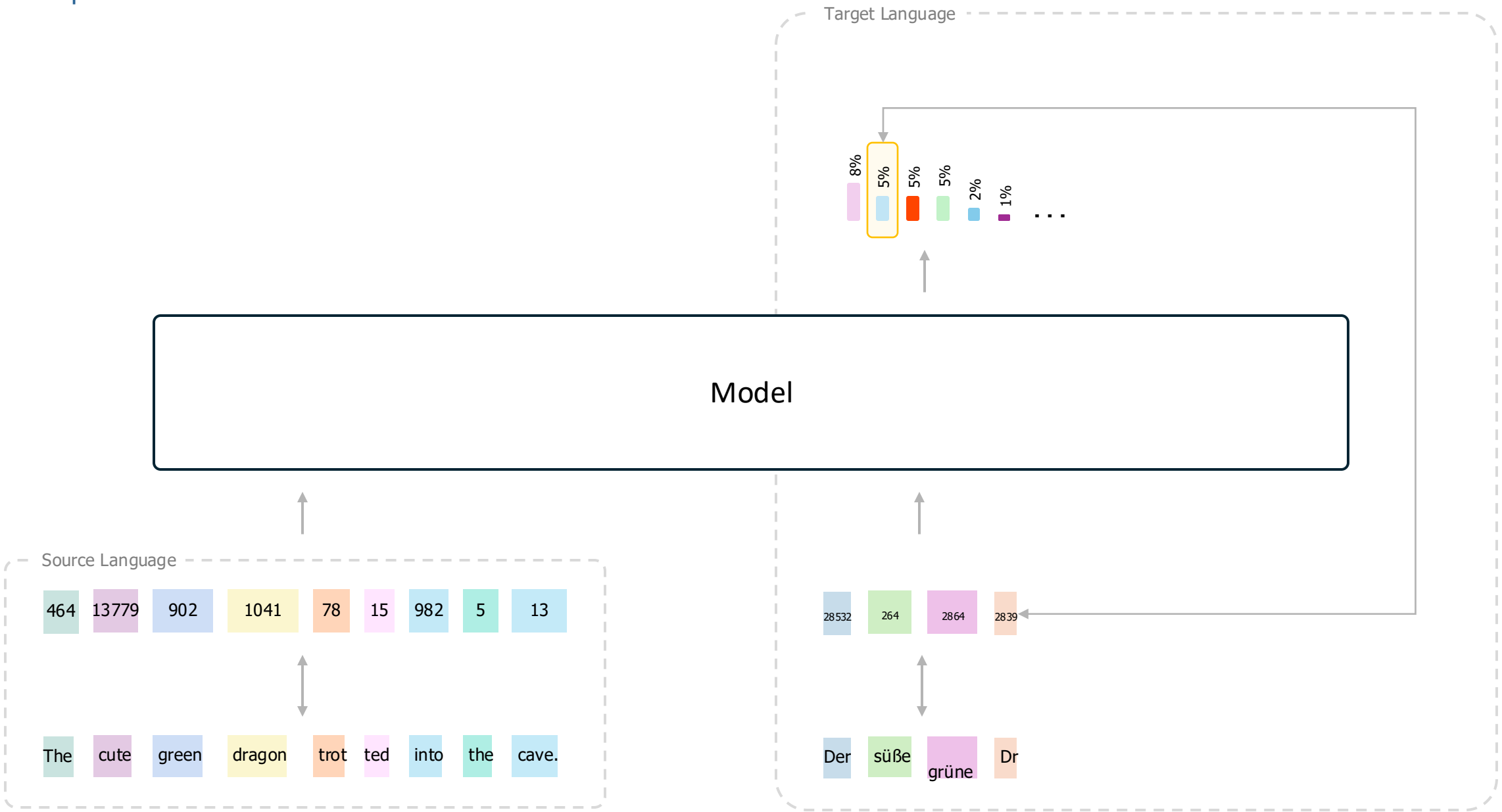
Recap: Transformer Architecture for Machine Translation



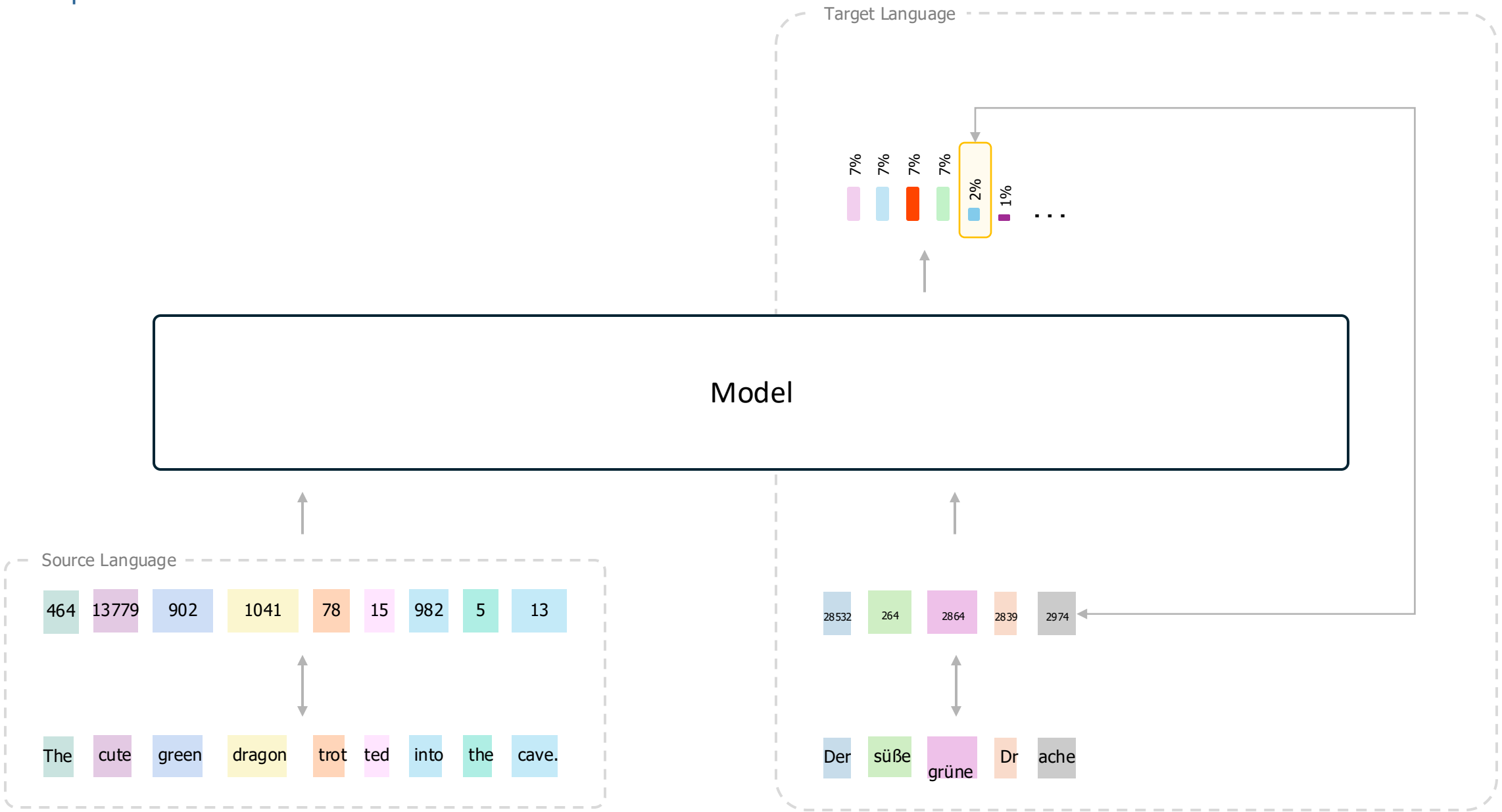
Recap: Transformer Architecture for Machine Translation



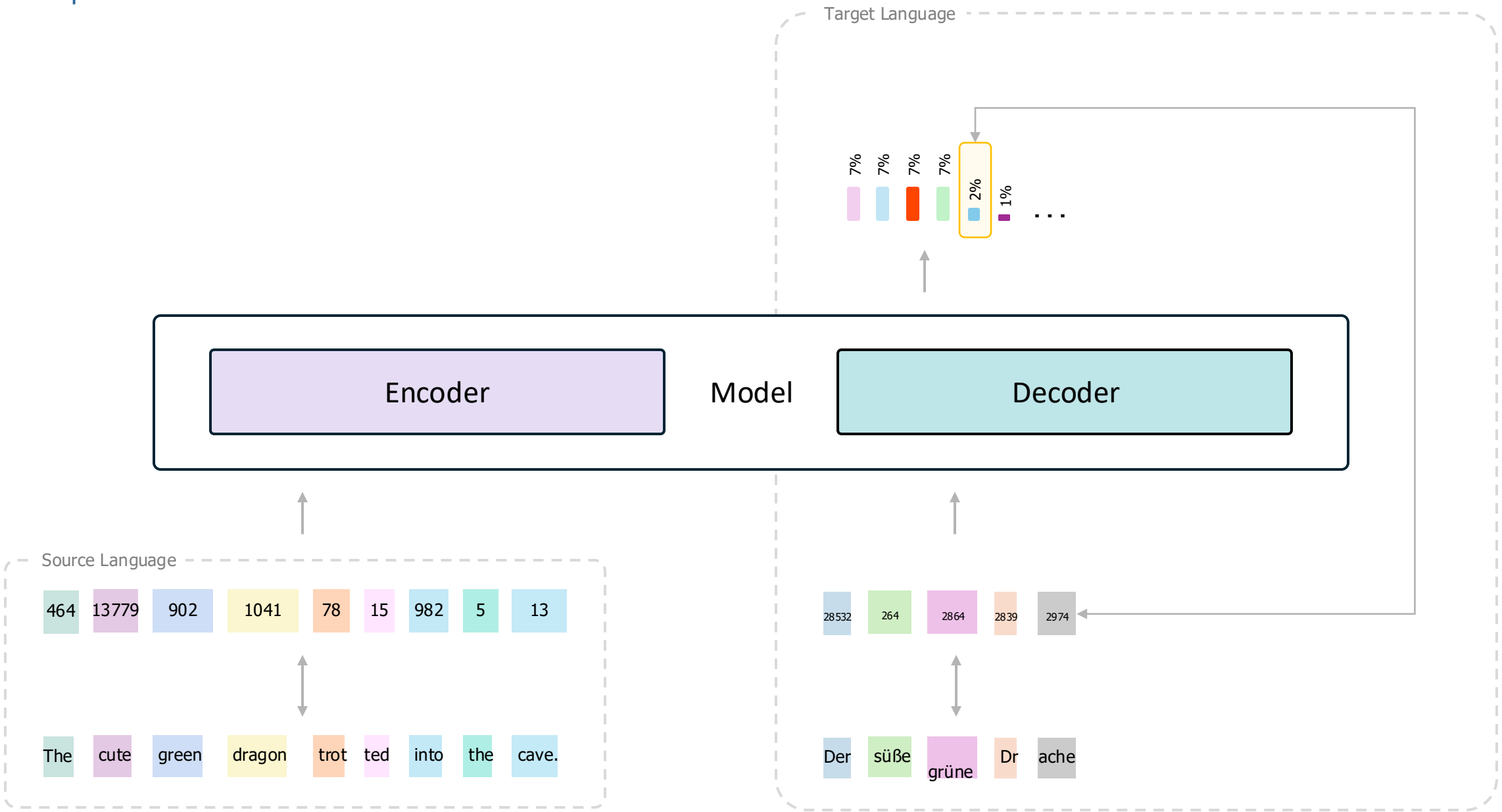
Recap: Transformer Architecture for Machine Translation



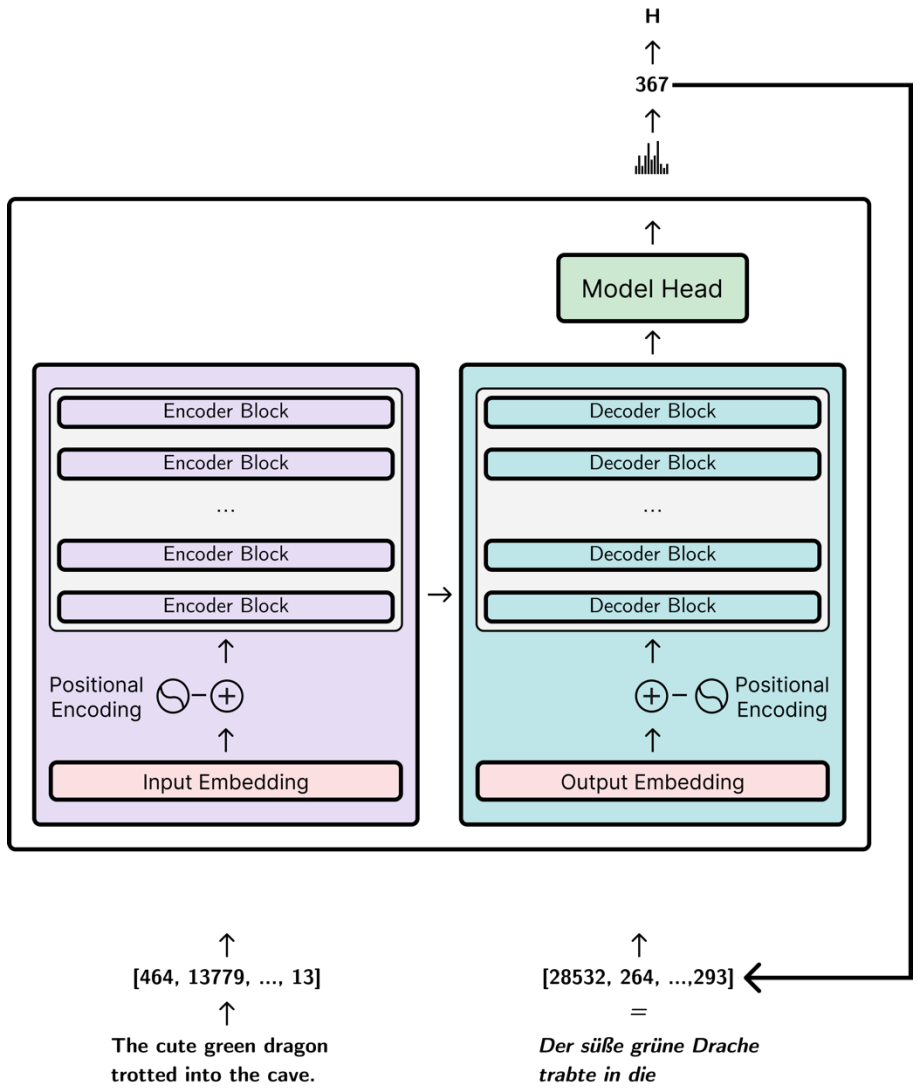
Recap: Transformer Architecture for Machine Translation



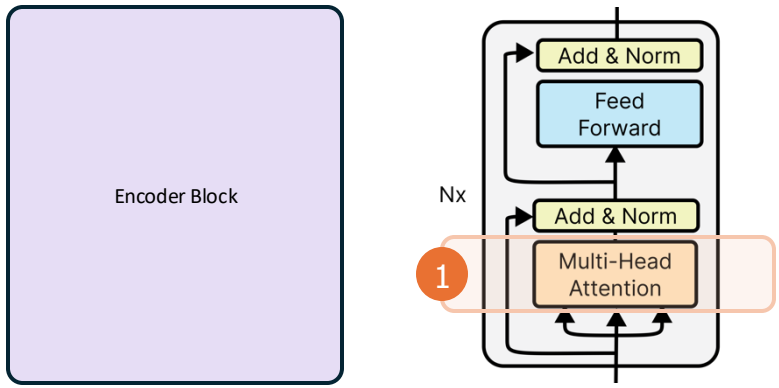
Recap: Transformer Architecture for Machine Translation

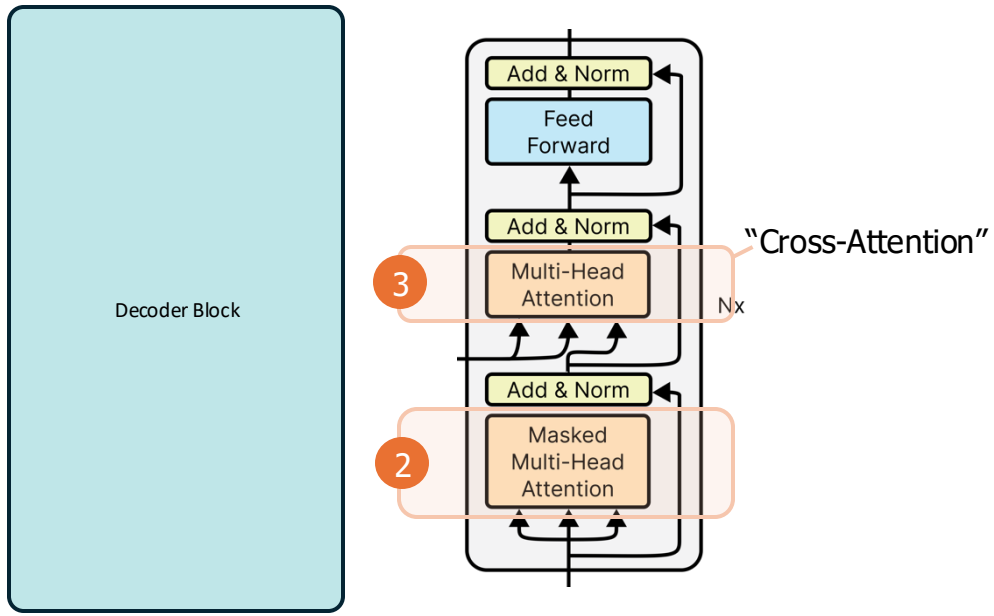


Recap: Transformer Architecture for Machine Translation

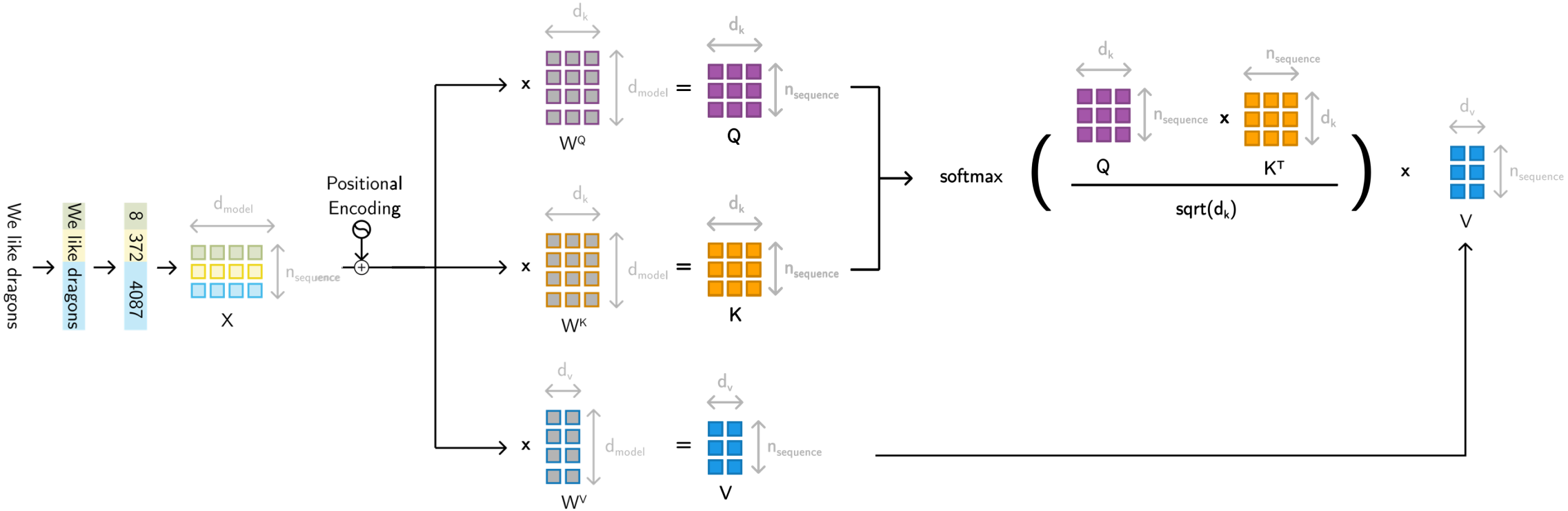


Recap: Encoder and Decoder Blocks

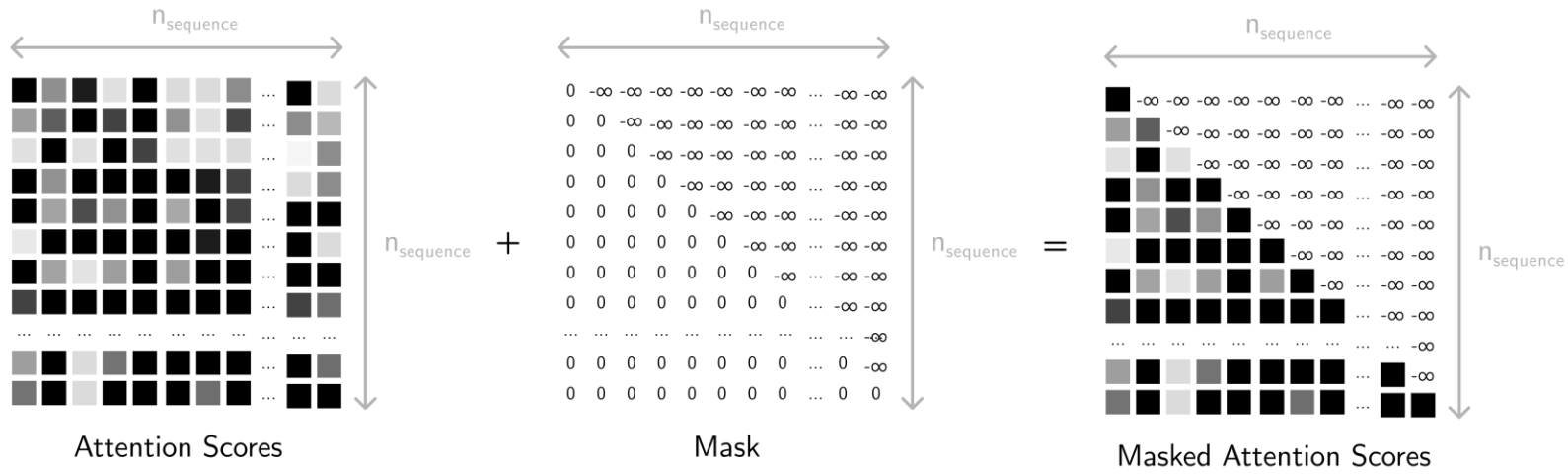


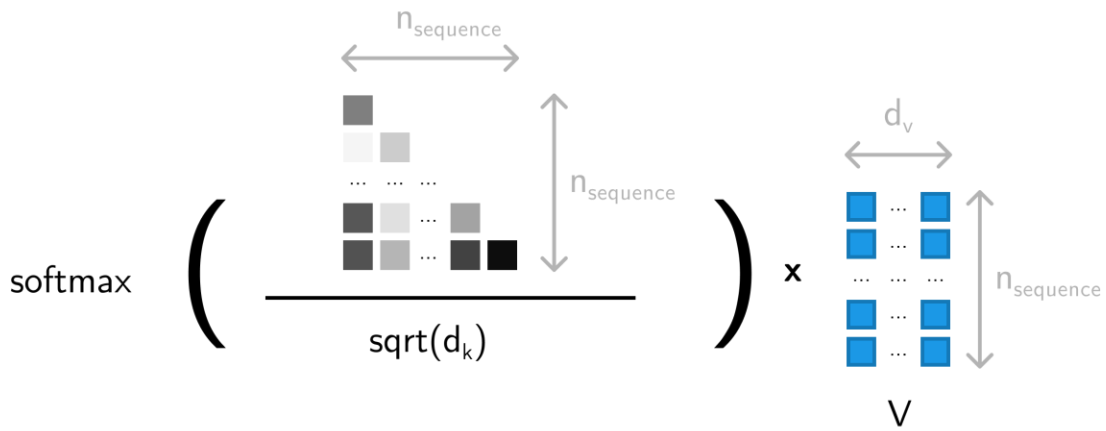


Recap: Attention Mechanism



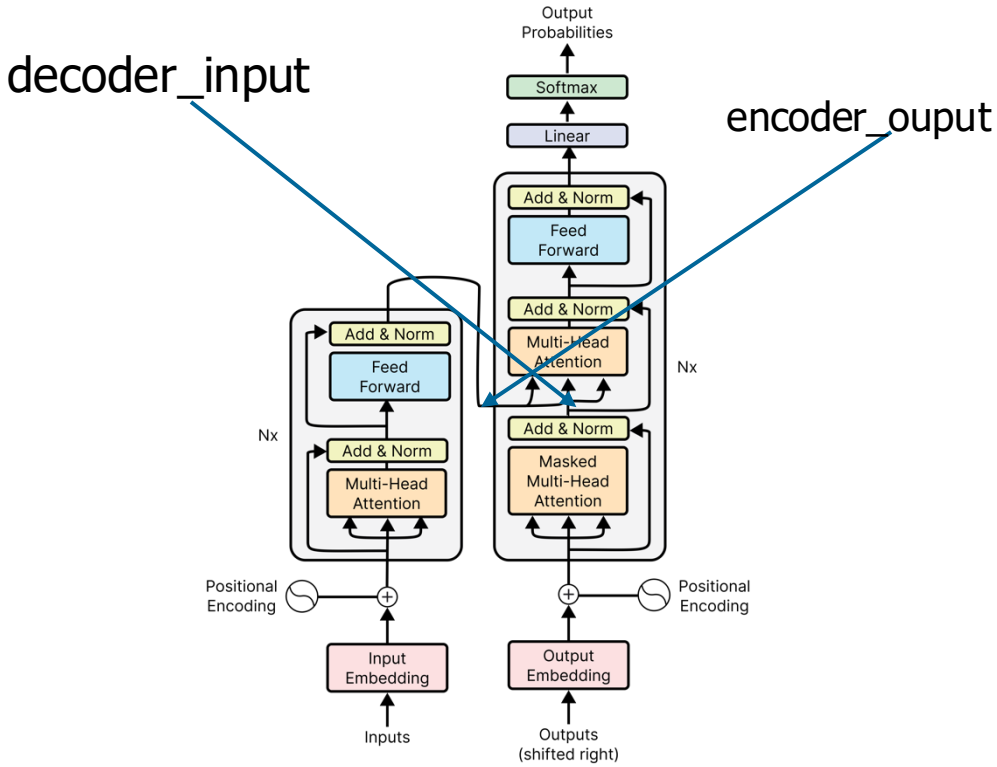
Recap: Masked Attention Mechanism





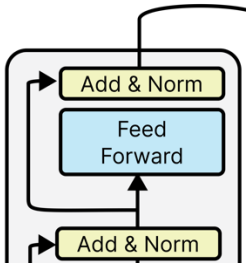
Recap: Cross-Attention

The query (Q) comes from the decoder layer. The key (K) and value (V) come from the encoder's output.




```
Q = W_Q(decoder_input)
K = W_K(encoder_output)
V = W_V(encoder_output)
```

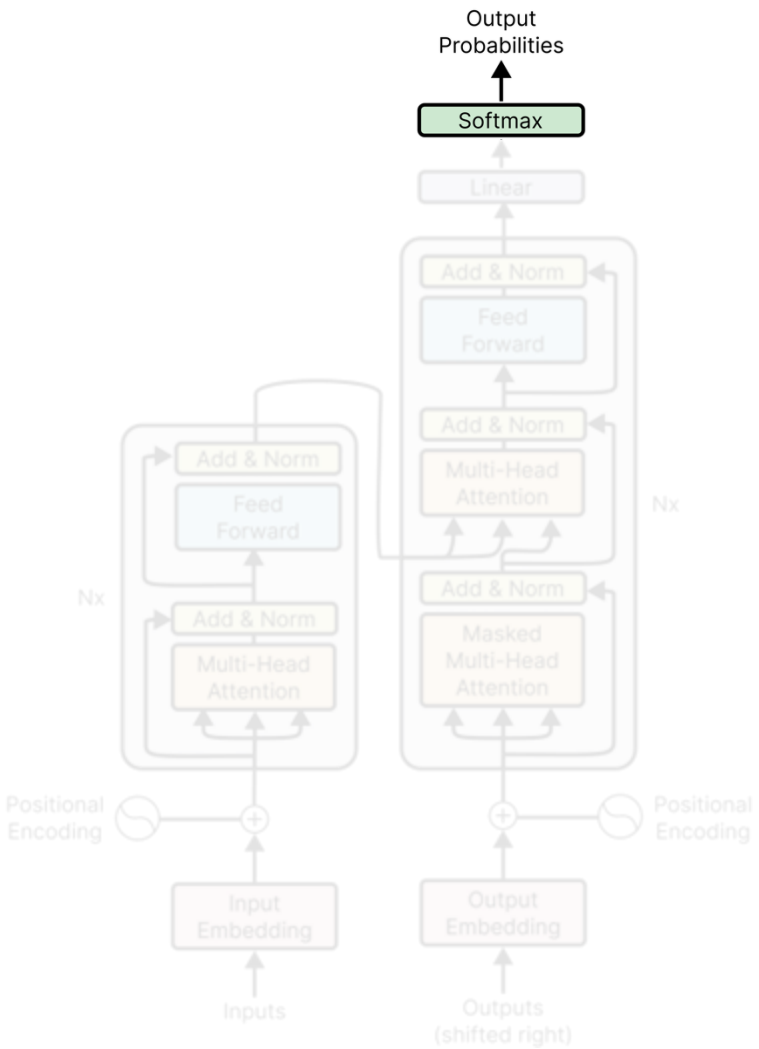
Recap: Feed Forward Networks, Residual Connections and Layer Norms

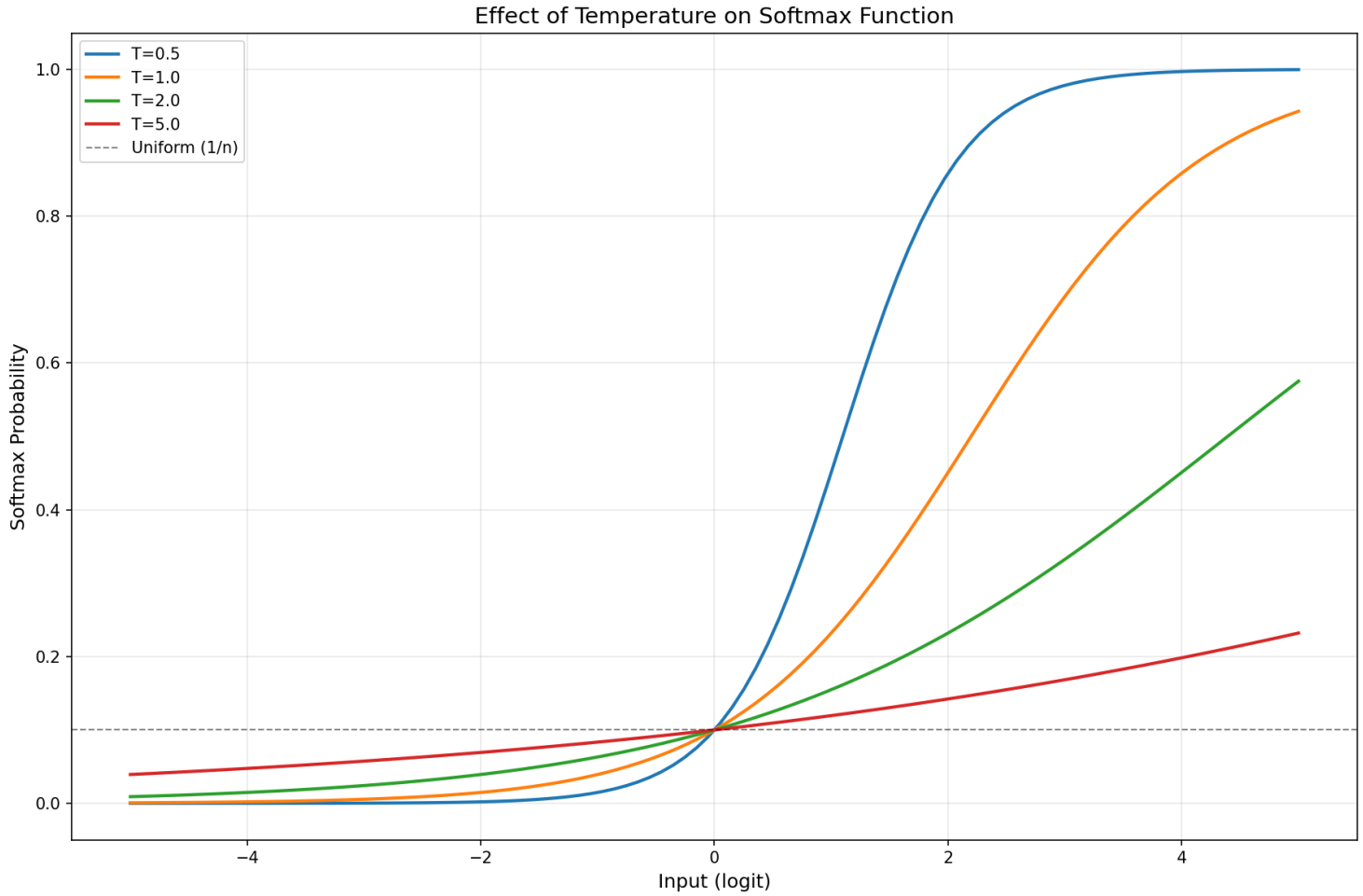



```
import torch
from torch import nn

d_model = 128
d_ff = 4*d_model
n_sequence = 200 # sample sequence length
x = torch.randn(n_sequence, d_model) # [n_sequence, d_model]

norm = nn.LayerNorm(d_model)
feed_forward = nn.Sequential(
      nn.Linear(d_model, dim_ff),
      nn.ReLU(),
      nn.Linear(dim_ff, d_model)
)

output = norm(x + feed_forward(x))
```

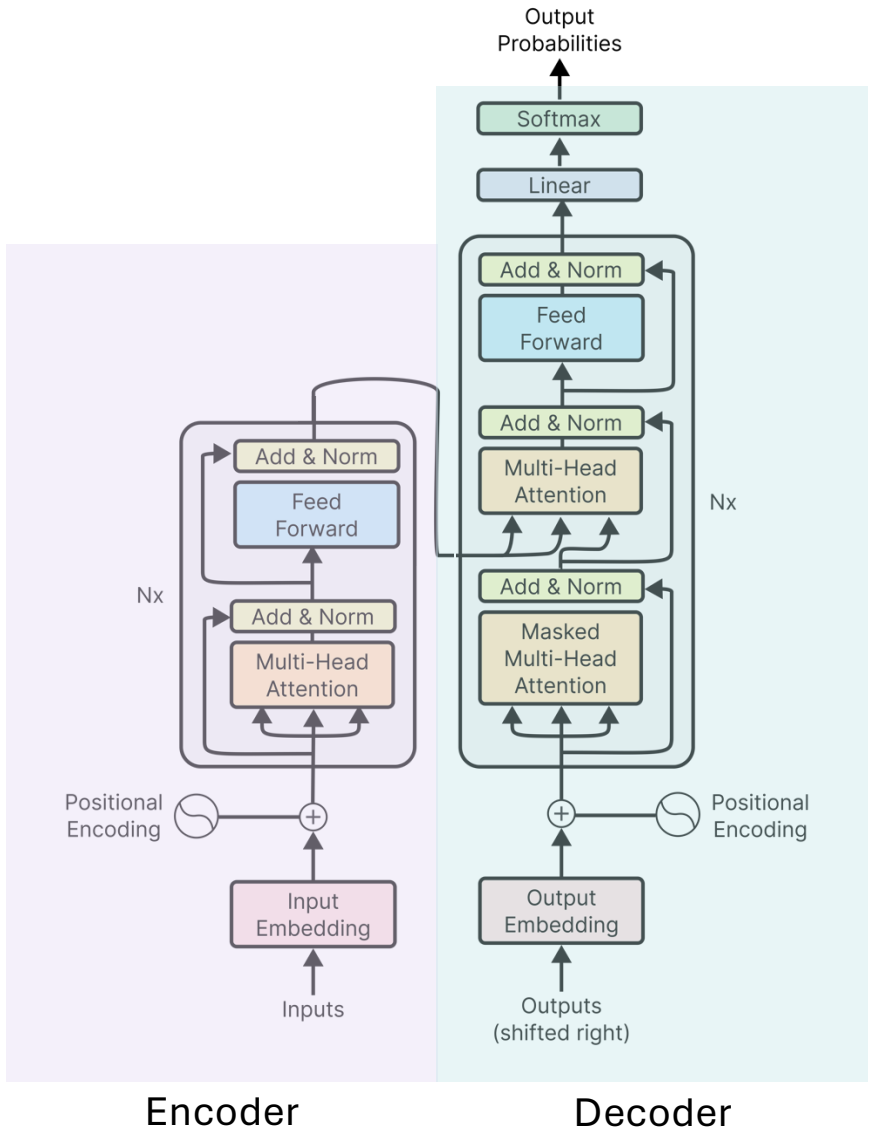
1. For details of the layer norm take a look at https://docs.pytorch.org/docs/stable/generated/torch.nn.LayerNorm.html

Recap: Final Softmax with Temperature Parameter

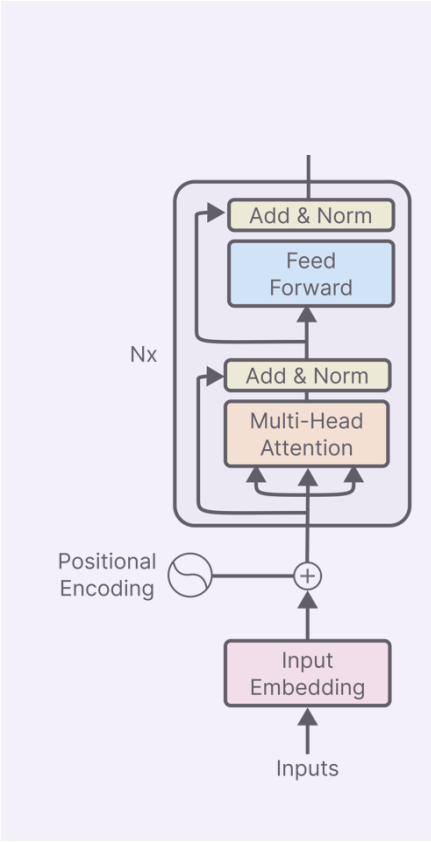




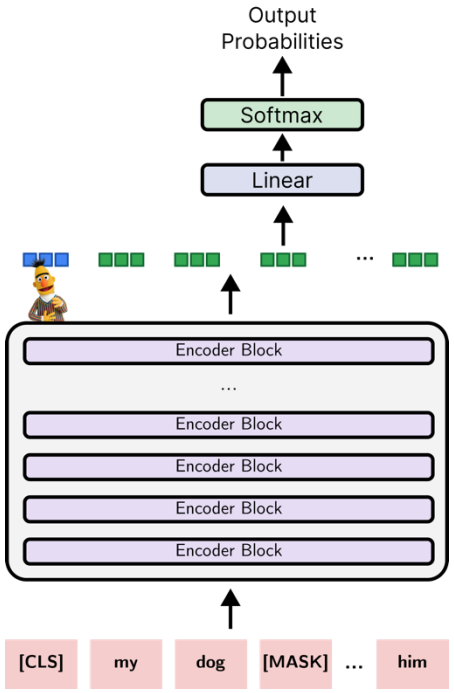
$$softmax(x_i) = \frac{e^{\frac{x_i}{T}}}{\sum_j e^{\frac{x_j}{T}}}$$


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Positional Encoding
Output Embedding
Outputs (shifted right)
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Input Embedding
Inputs
Encoder
Decoder

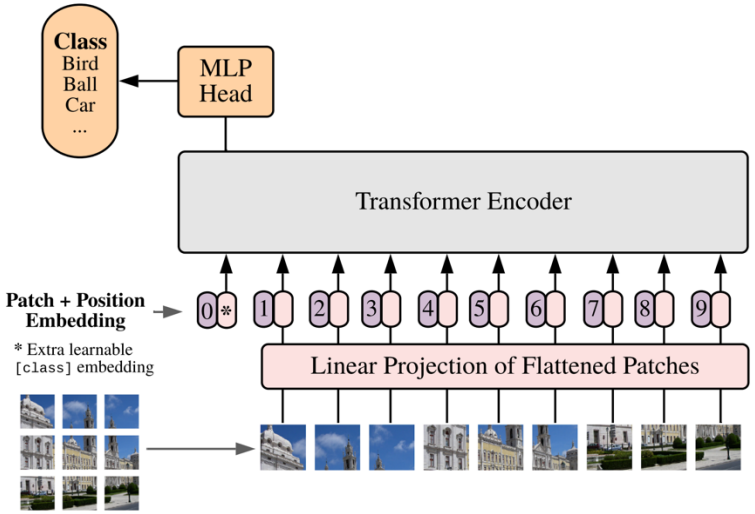
Encoder Only Models



Encoder

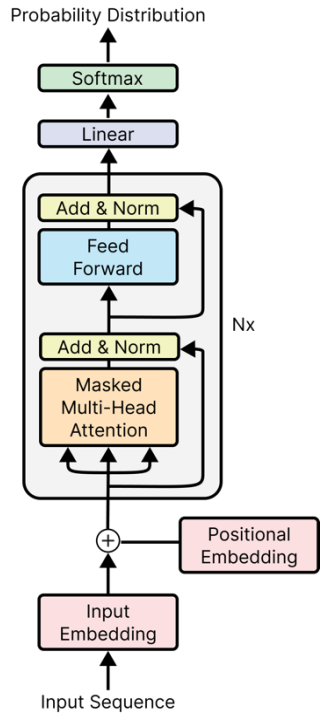


Encoder Only Transformers (e.g. BERT)[2]



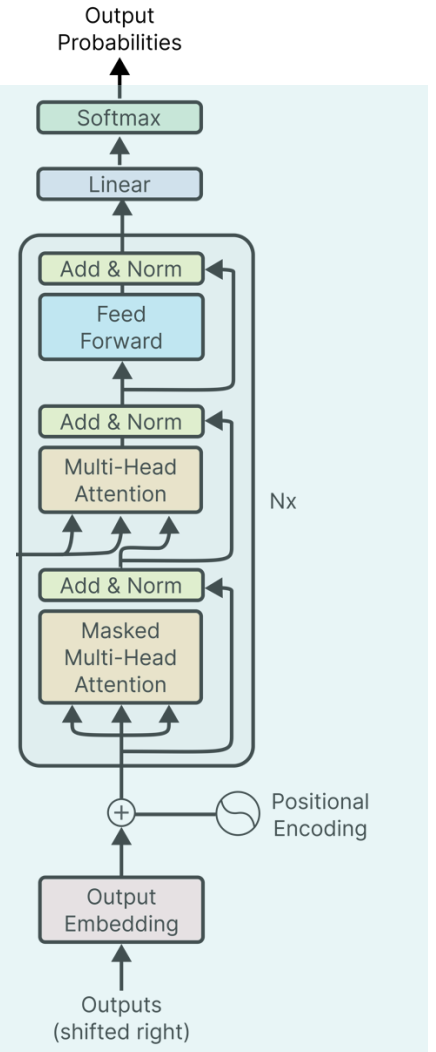
Encoder Only Vision Transformer

Decoder Only Models



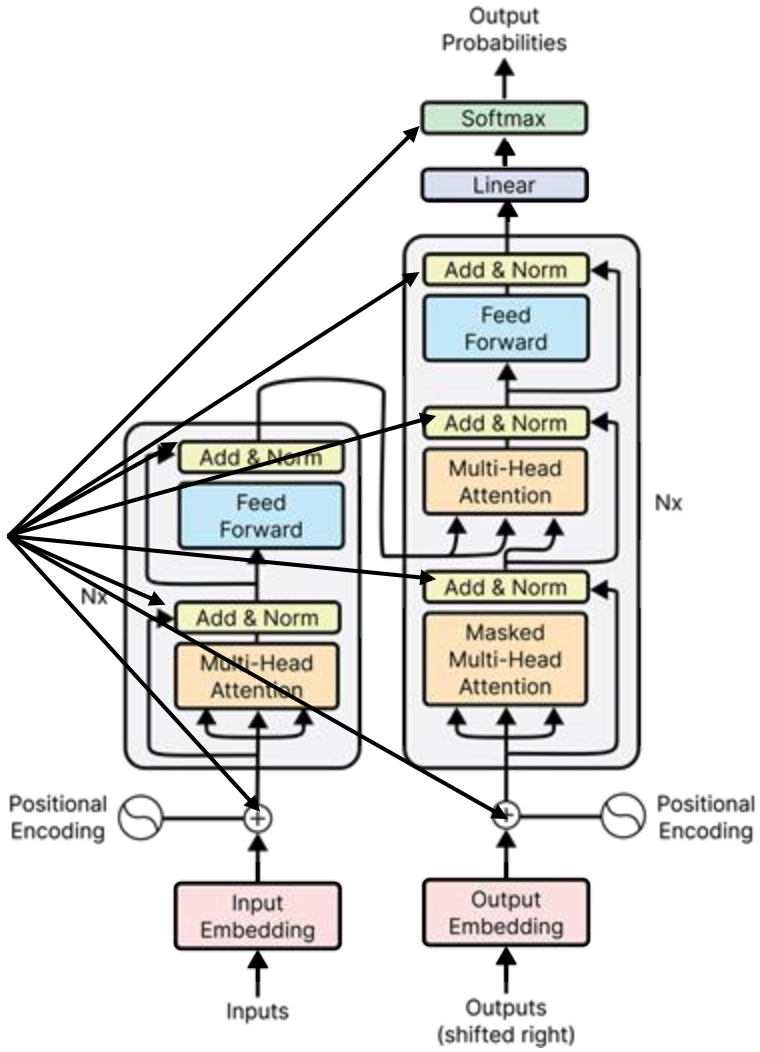
Decoder Only Transformers (e.g. GPT)[3]

[…] Pretty much any LLM you can think of […]

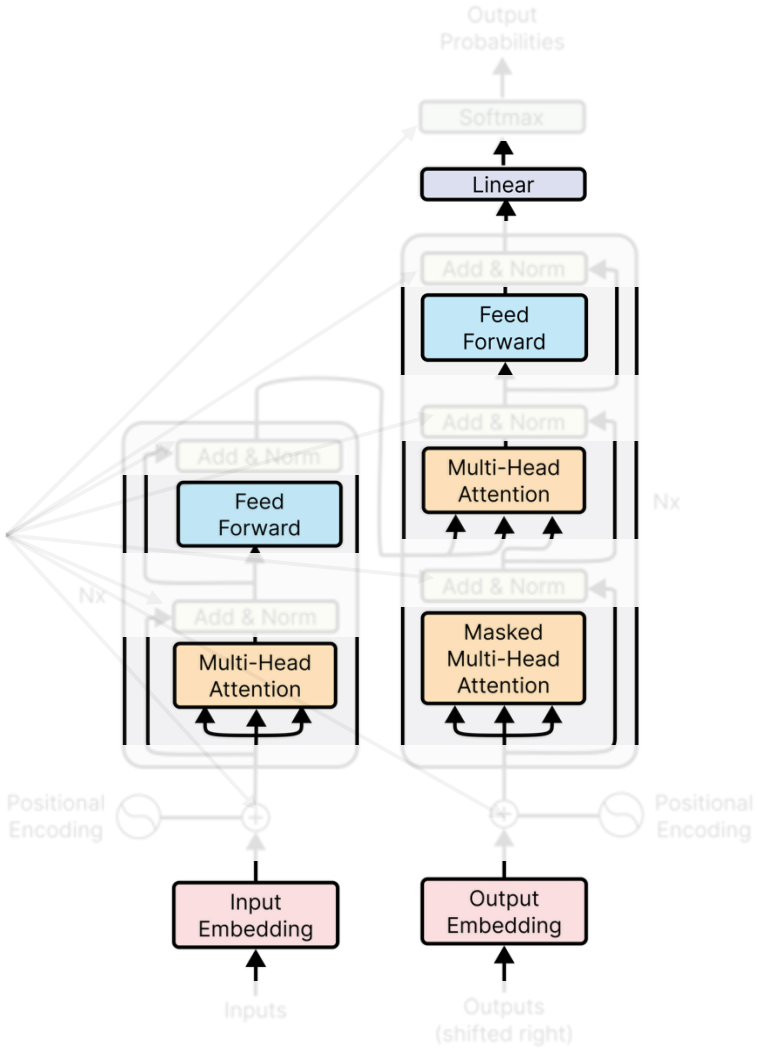


Decoder

Non-Learned Operations



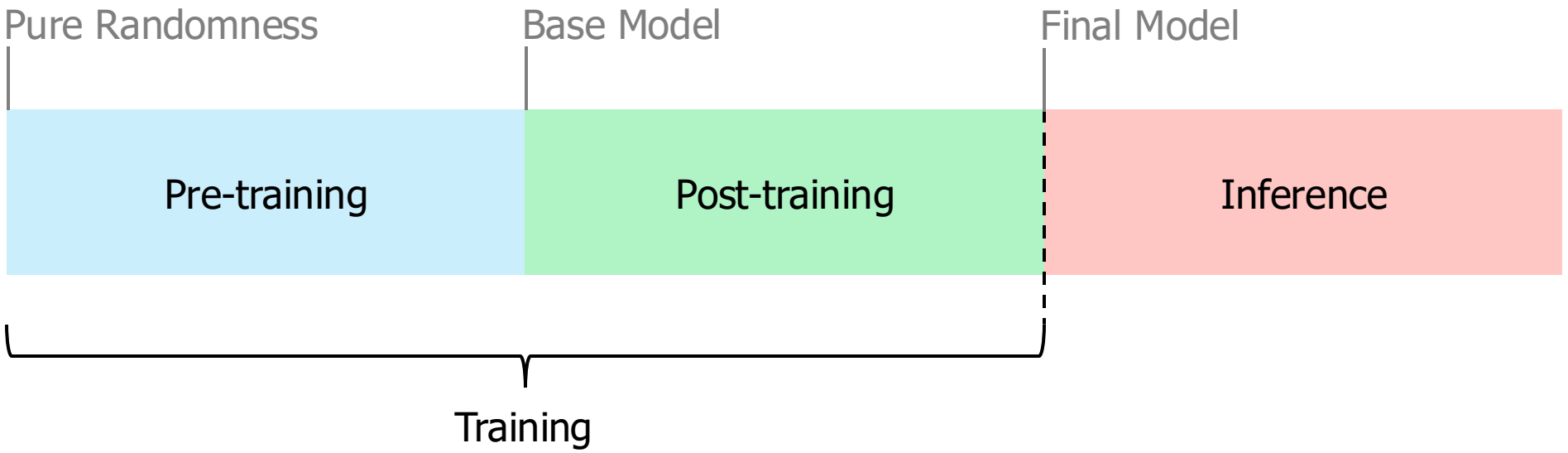
Static Non-Learned Operations



Learned Parameters

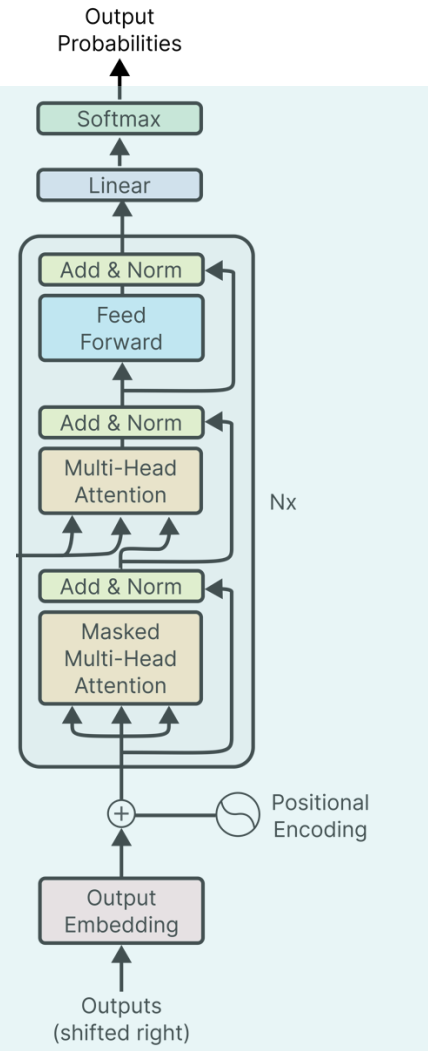
How Computers Learn To Read and Write

Training a Large Language Model


Pure Randomness
Base Model
Final Model
Pre-training
Post-training
Inference
Training

Training Of

Decoder Only Models



Decoder

Pre-training

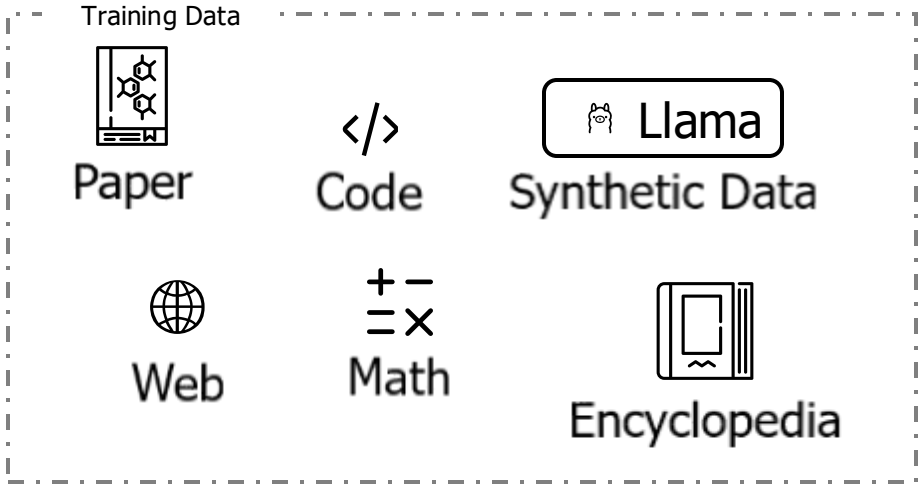
START

model.py ~500 lines of code defining the models architecture with randomly initialized weights

```
# Random weight initialization example
layer = nn.Linear(512, 256)
nn.init.normal_(layer.weight, mean=0.0, std=0.02)
```

METHODOLOGY

Self-Supervised Training



END RESULT

A *Base Model (Pre-trained Model)* that can recall factual knowledge and produce syntactically and semantically coherent text. "*Text completion on steroids"*

but

- can't follow instructions
- is anaware of external alignment objectives

START

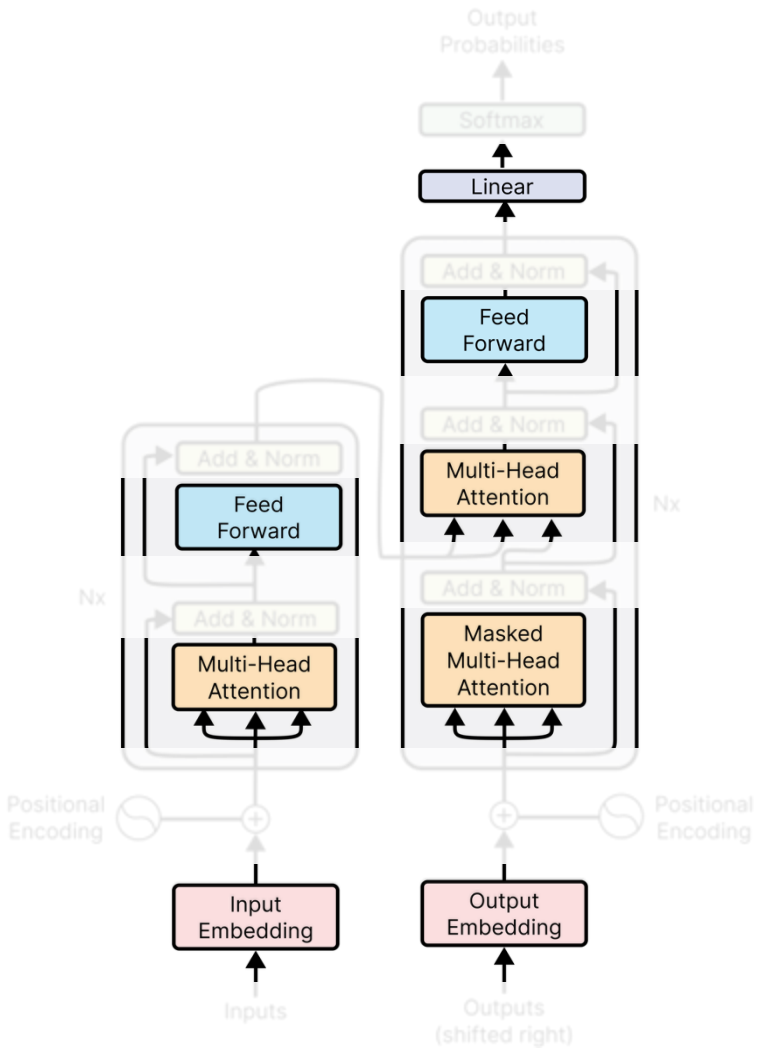
Base Model / Pre-trained Model

METHODOLOGY & END RESULT(S)

- Supervised Finetuning (SFT) / Instruction Tuning (IT)
- Preference Finetuning (PrefT)
- Reinforcement Learning with Verifiable Results (RLVR)

→ *Post-training strategies will be covered by Prof. Heitzinger*

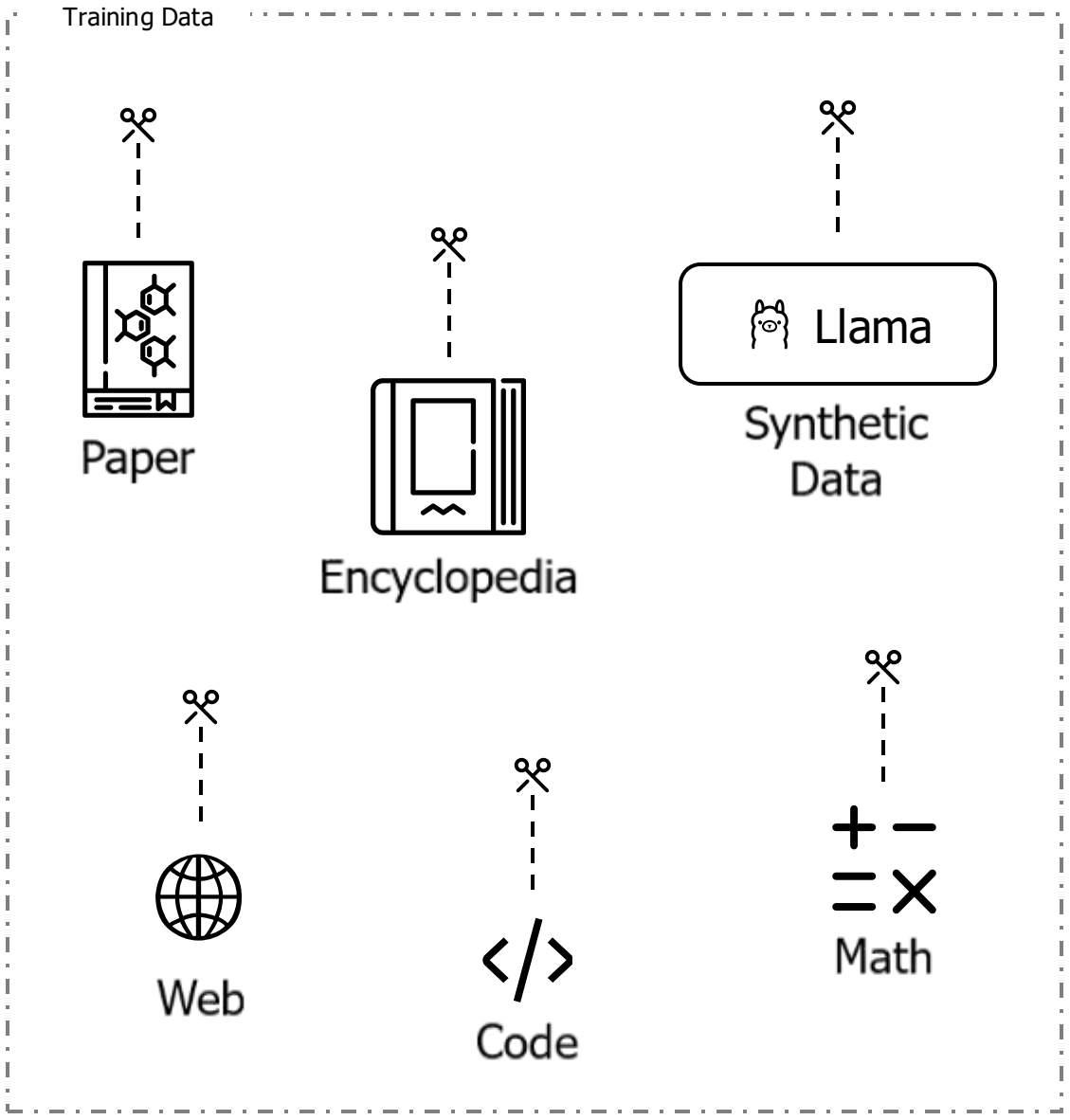
Pre-training



model.py ~500 lines of code defining the models architecture with randomly initialized weights

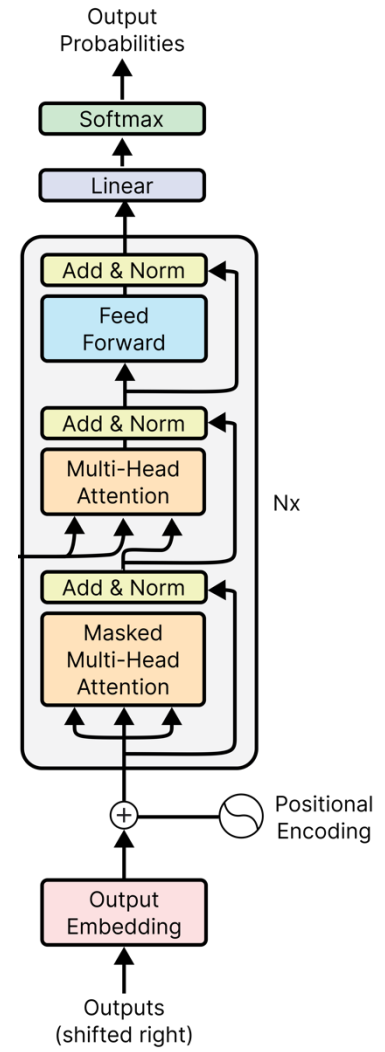
```
# Random weight initialization example
layer = nn.Linear(512, 256)
nn.init.normal_(layer.weight, mean=0.0, std=0.02)
```

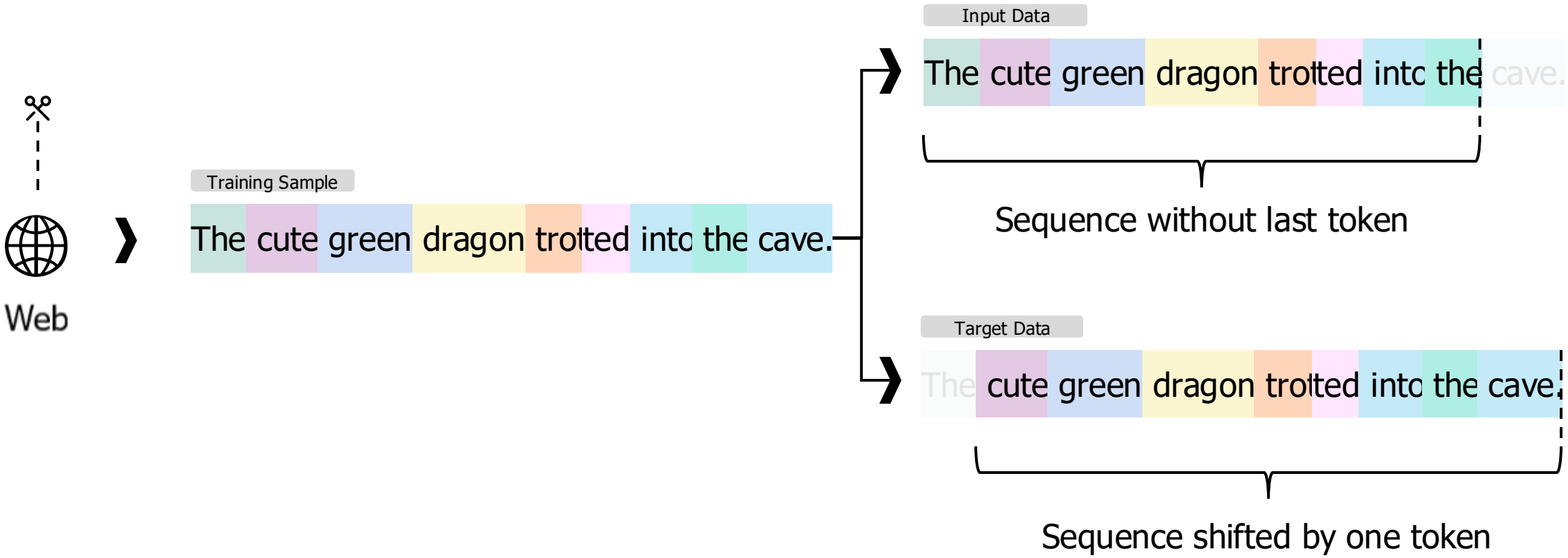
Pre-training

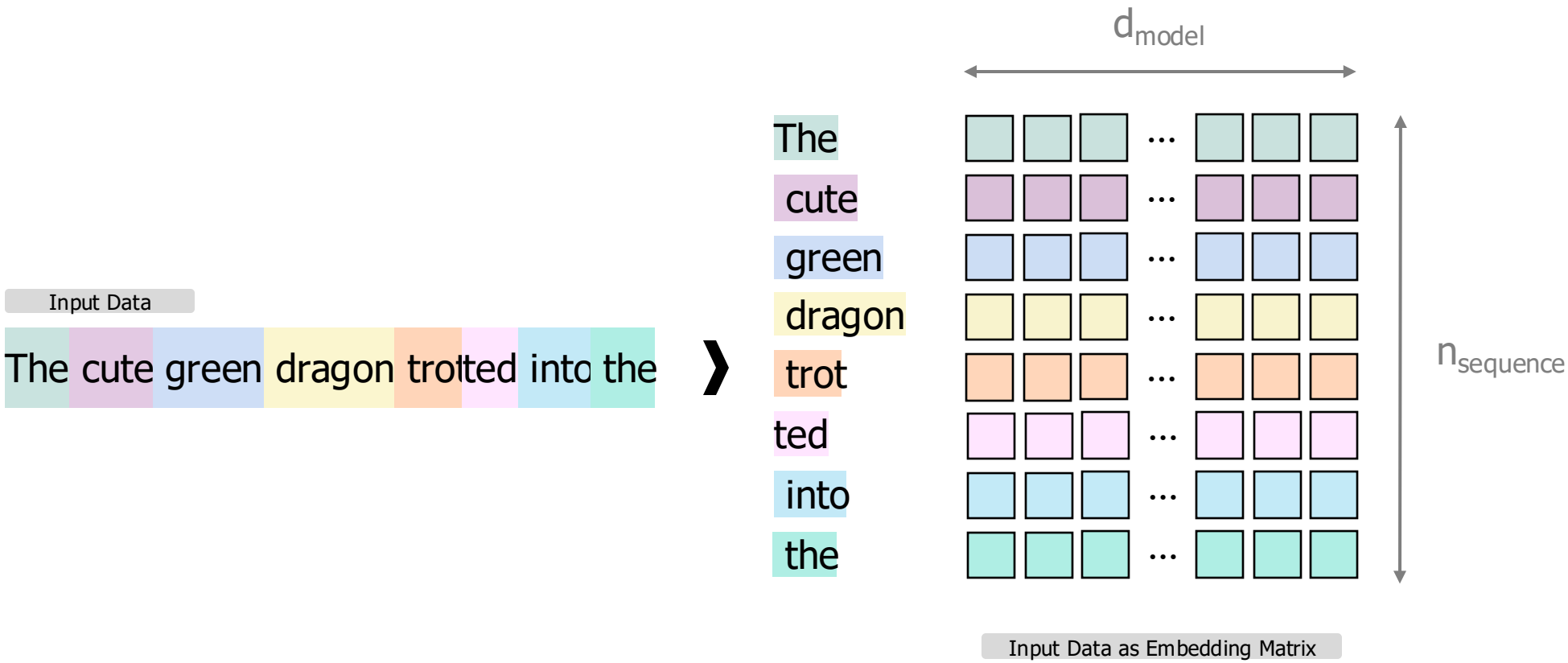


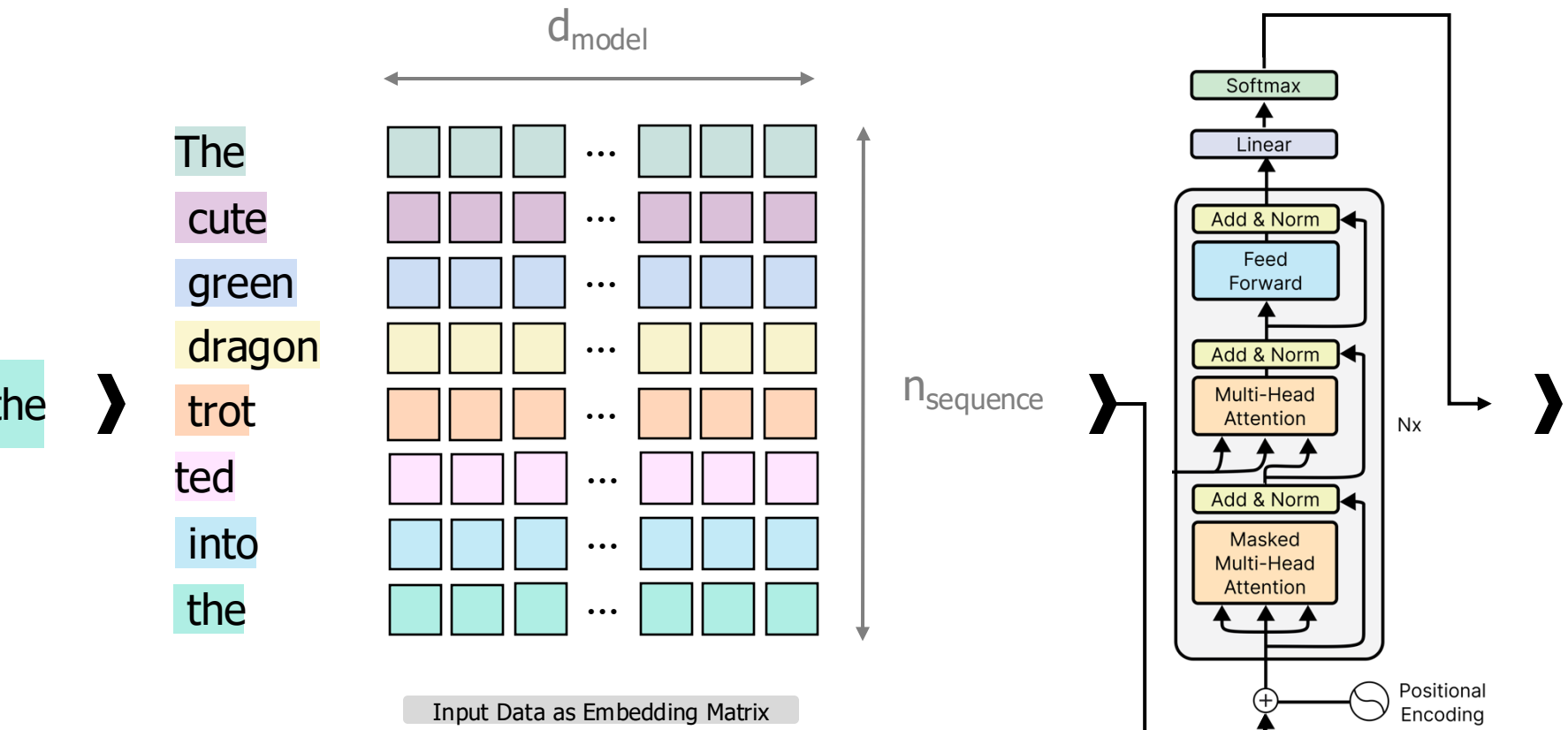
$15 * 10^{12}$

~ 15 000 000 000 000

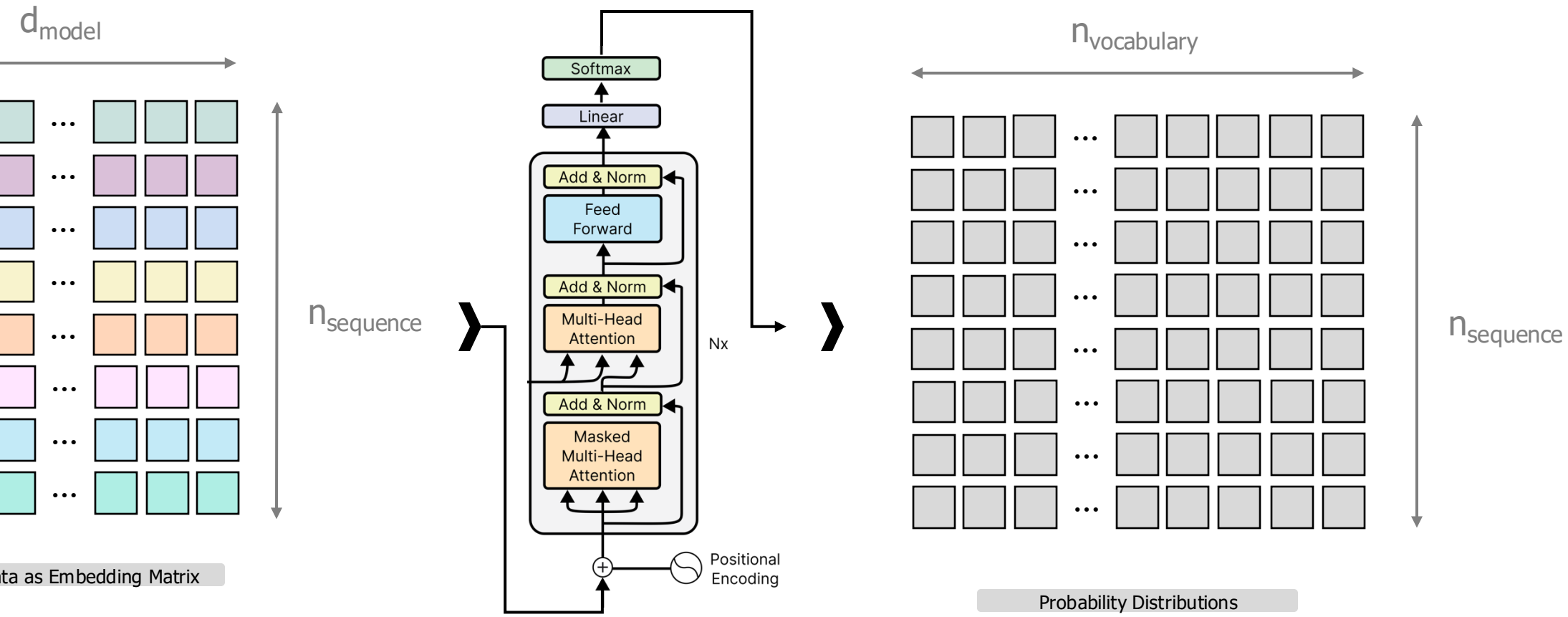


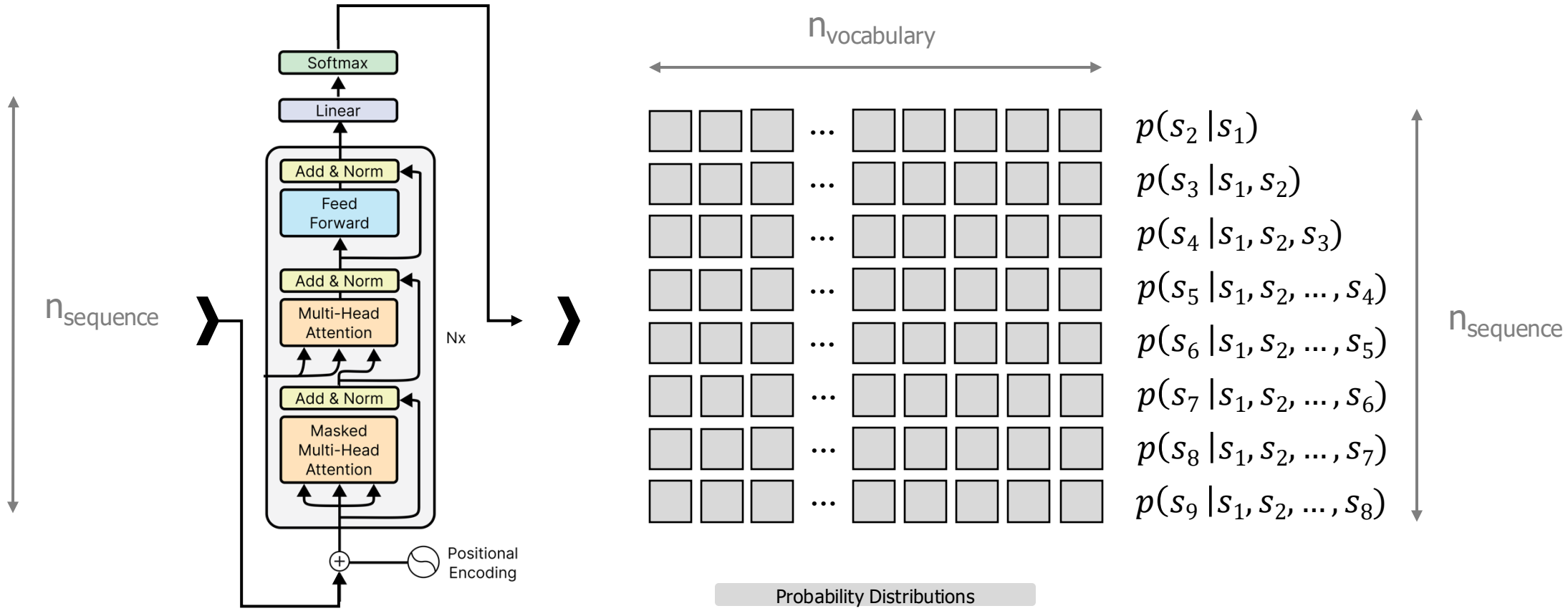

Web
Training Sample
The cute green dragon trotted into the cave.
Input Data
The cute green dragon trotted into the cave.
Sequence without last token
Target Data
The cute green dragon trotted into the cave.
Sequence shifted by one token

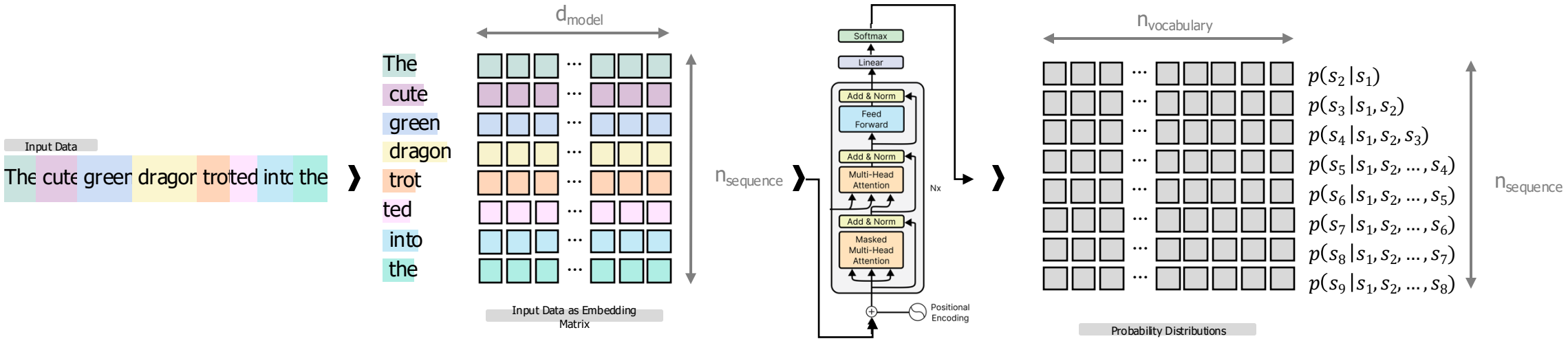

Input Data
The cute green dragon trotted into the
d model
The
cute
green
dragon
trot
ted
into
the
n sequence
Input Data as Embedding Matrix

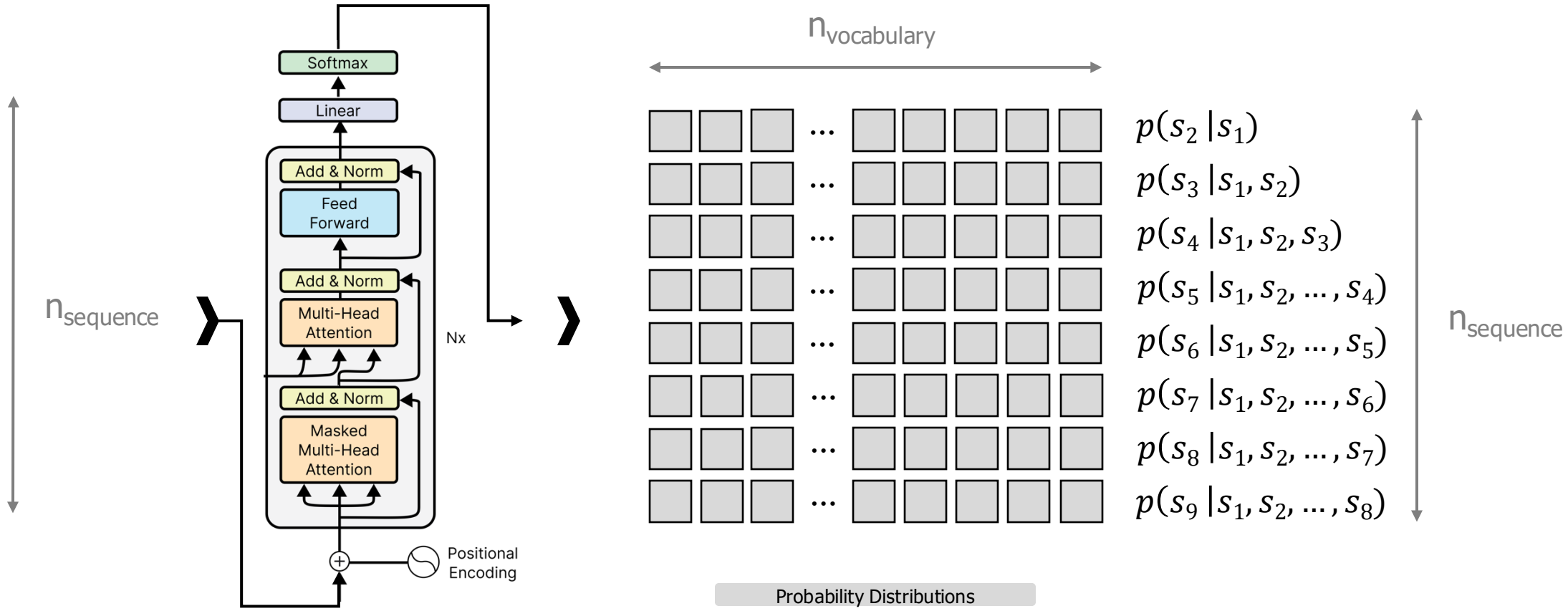

d model
The
cute
green
dragon
trot
ted
into
the
n sequence
Input Data as Embedding Matrix

Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Positional Encoding


d_{model}
$n_{sequence}$
ta as Embedding Matrix
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Positional Encoding
$n_{vocabulary}$
$n_{sequence}$
Probability Distributions


$n_{sequence}$
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Positional Encoding
$n_{vocabulary}$
$p(s_2 | s_1)$
$p(s_3 | s_1, s_2)$
$p(s_4 | s_1, s_2, s_3)$
$p(s_5 | s_1, s_2, \ldots, s_4)$
$p(s_6 | s_1, s_2, \ldots, s_5)$
$p(s_7 | s_1, s_2, \ldots, s_6)$
$p(s_8 | s_1, s_2, \ldots, s_7)$
$p(s_9 | s_1, s_2, \ldots, s_8)$
$n_{sequence}$
Probability Distributions


Input Data
The cute green dragor troted intc the
The
cute
green
dragon
trot
ted
into
the
d_{model}
Input Data as Embedding Matrix
$n_{sequence}$
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Positional Encoding
$n_{vocabulary}$
$p(s_2|s_1)$
$p(s_3|s_1,s_2)$
$p(s_4|s_1,s_2,s_3)$
$p(s_5|s_1,s_2,\ldots,s_4)$
$p(s_6|s_1,s_2,\ldots,s_5)$
$p(s_7|s_1,s_2,\ldots,s_6)$
$p(s_8|s_1,s_2,\ldots,s_7)$
$p(s_9|s_1,s_2,\ldots,s_8)$
$n_{sequence}$
Probability Distributions


$n_{sequence}$
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Positional Encoding
$n_{vocabulary}$
$p(s_2 | s_1)$
$p(s_3 | s_1, s_2)$
$p(s_4 | s_1, s_2, s_3)$
$p(s_5 | s_1, s_2, \ldots, s_4)$
$p(s_6 | s_1, s_2, \ldots, s_5)$
$p(s_7 | s_1, s_2, \ldots, s_6)$
$p(s_8 | s_1, s_2, \ldots, s_7)$
$p(s_9 | s_1, s_2, \ldots, s_8)$
$n_{sequence}$
Probability Distributions

Recap: Learning Objective

The goal of generative language modelling is to capture the "true distribution of language": $p(X)$

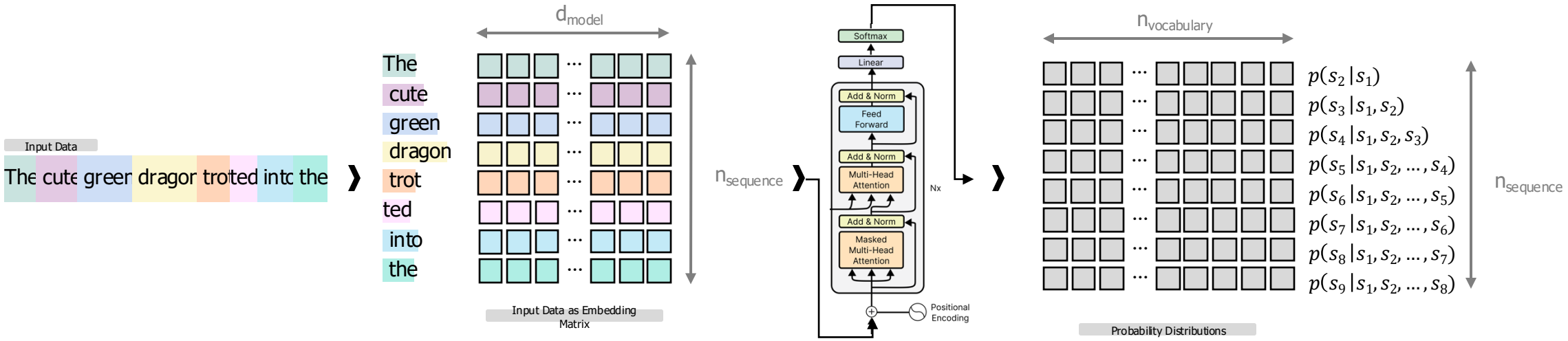
A few definitions:

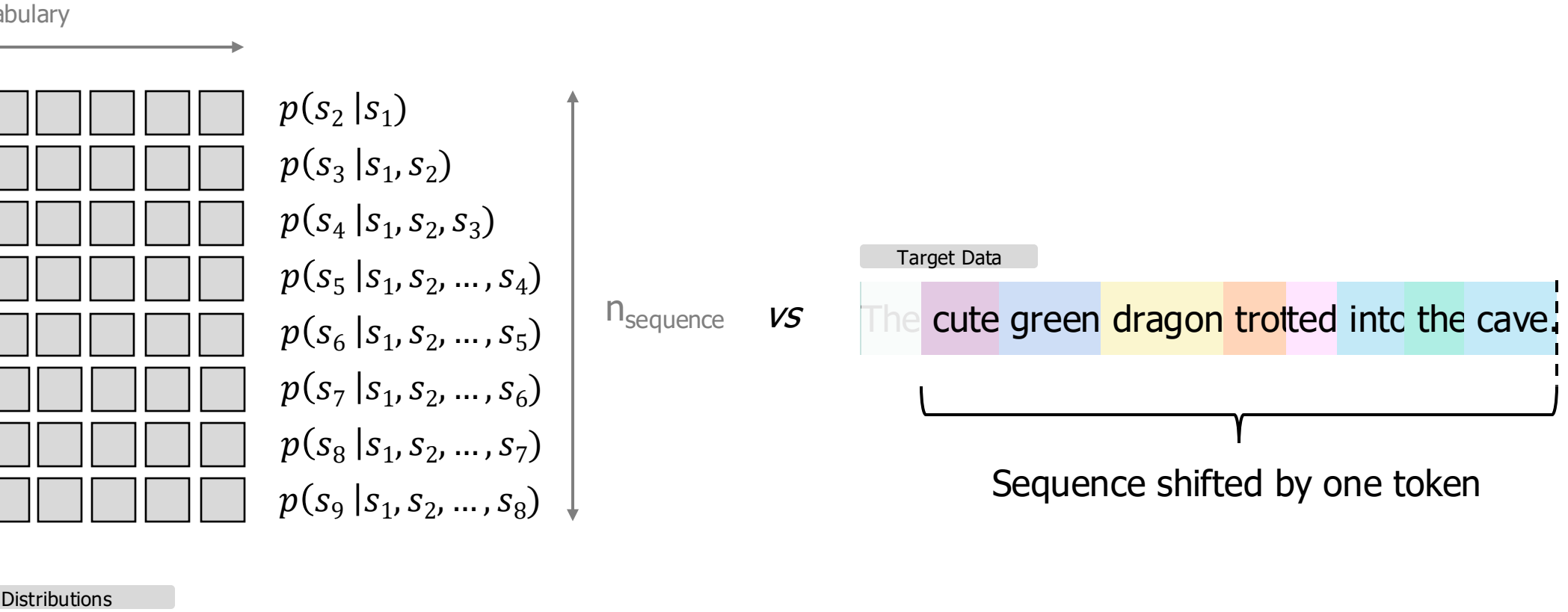
- V … vocabulary of unique tokens $s_i \in V$
- n … maximum sequence length
- X … space of all possible token sequences

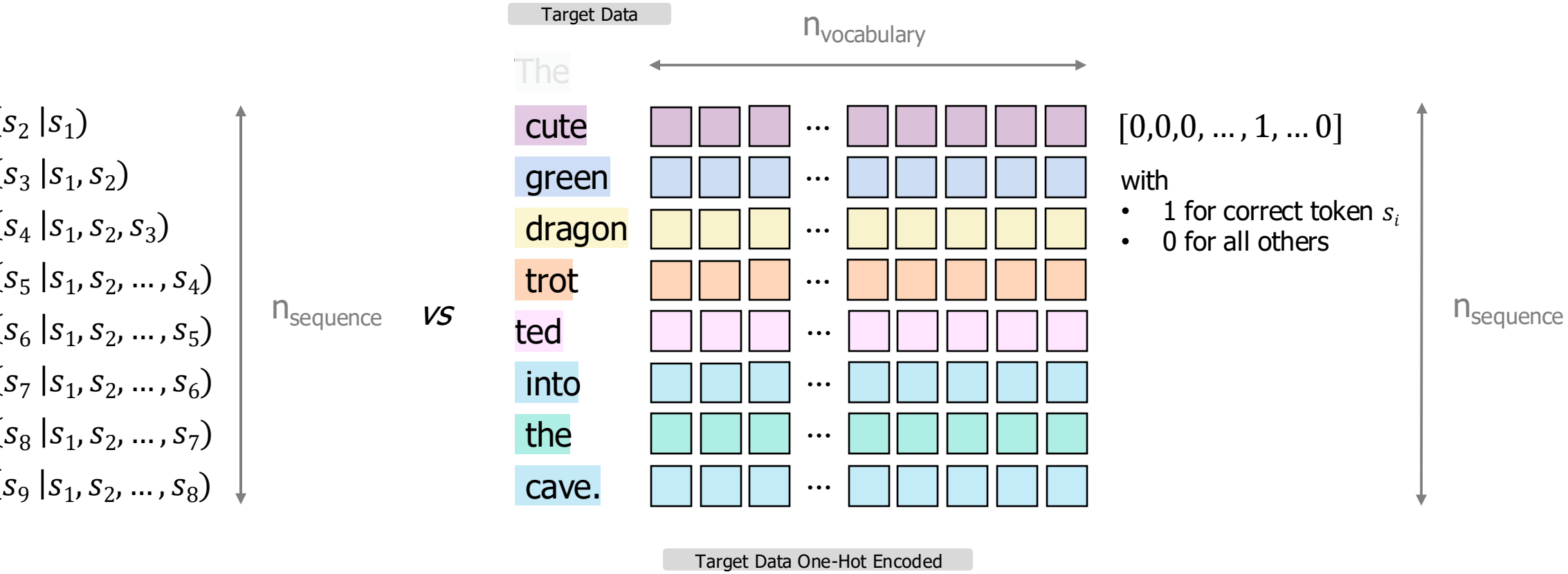
  $X = \{(s_1, s_2, \dots, s_k) | s_i \in V, 0 \leq k \leq n, k \in \mathbb{Z}\}$ where k is the sequence length
- $x \in X$ … a particular sequence of tokens
- The probability of a given sequence x

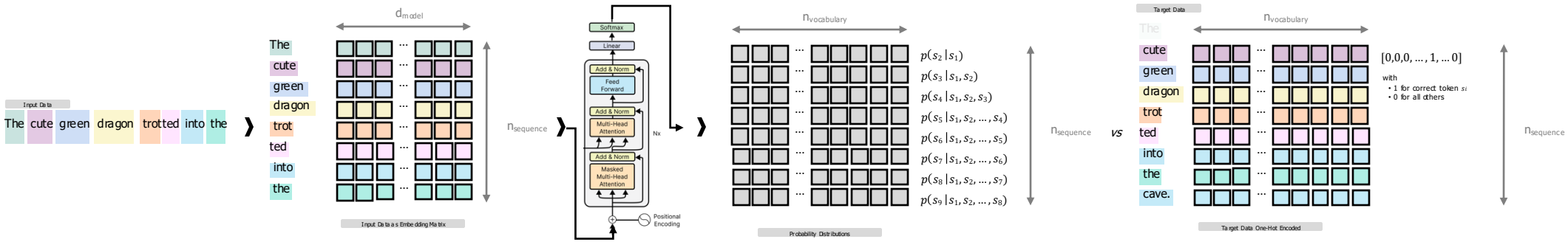
$$p(x) = p(s_1) \cdot p(s_2 | s_1) \cdot p(s_3 | s_1, s_2) \cdot \dots \cdot p(s_k | s_1, \dots, sk_{-1})$$
$$= p(s_1) \cdot \prod_{i=2}^{k} p(si | s_1, \dots, si_{-1})$$

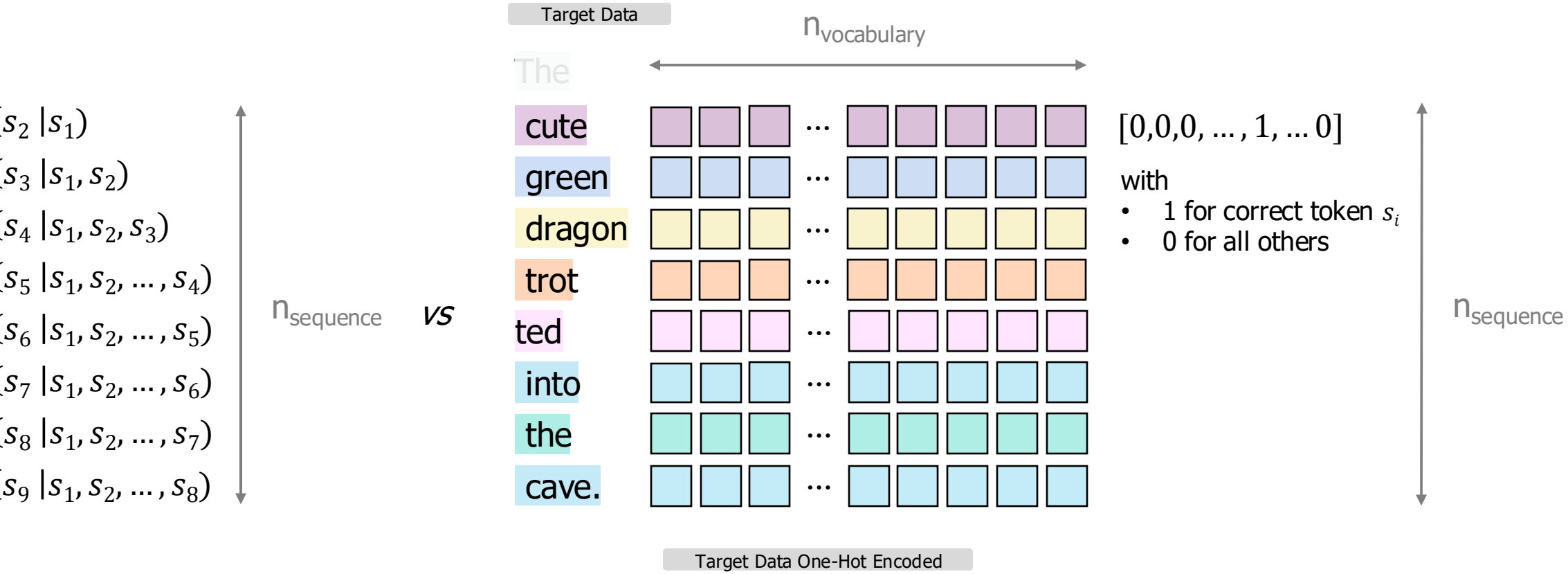
The probability of a sequence is the product of the probability of the first token and the conditional probabilities of each subsequent token given all previous tokens.


Input Data
The cute green dragor troted intc the
The
cute
green
dragon
trot
ted
into
the
d_{model}
Input Data as Embedding Matrix
$n_{sequence}$
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Positional Encoding
$n_{vocabulary}$
$p(s_2|s_1)$
$p(s_3|s_1,s_2)$
$p(s_4|s_1,s_2,s_3)$
$p(s_5|s_1,s_2,\ldots,s_4)$
$p(s_6|s_1,s_2,\ldots,s_5)$
$p(s_7|s_1,s_2,\ldots,s_6)$
$p(s_8|s_1,s_2,\ldots,s_7)$
$p(s_9|s_1,s_2,\ldots,s_8)$
$n_{sequence}$
Probability Distributions


abulary
$p(s_2 | s_1)$
$p(s_3 | s_1, s_2)$
$p(s_4 | s_1, s_2, s_3)$
$p(s_5 | s_1, s_2, \ldots, s_4)$
$p(s_6 | s_1, s_2, \ldots, s_5)$
$p(s_7 | s_1, s_2, \ldots, s_6)$
$p(s_8 | s_1, s_2, \ldots, s_7)$
$p(s_9 | s_1, s_2, \ldots, s_8)$
Distributions
$n_{sequence}$
vs
Target Data
The cute green dragon trotted into the cave.
Sequence shifted by one token


Target Data
$n_{vocabulary}$
The
cute
green
dragon
trot
ted
into
the
cave.
$(s_2 | s_1)$
$(s_3 | s_1, s_2)$
$(s_4 | s_1, s_2, s_3)$
$(s_5 | s_1, s_2, \dots, s_4)$
$(s_6 | s_1, s_2, \dots, s_5)$
$(s_7 | s_1, s_2, \dots, s_6)$
$(s_8 | s_1, s_2, \dots, s_7)$
$(s_9 | s_1, s_2, \dots, s_8)$
$n_{sequence}$
vs
$[0,0,0, \dots, 1, \dots 0]$
with
• 1 for correct token s_i
• 0 for all others
$n_{sequence}$
Target Data One-Hot Encoded


Input Data
The cute green dragon trotted into the
The
cute
green
dragon
trot
ted
into
the
d_{model}
$n_{sequence}$
Input Data as Embedding Matrix
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Positional Encoding
$n_{vocabulary}$
$p(s_2|s_1)$
$p(s_3|s_1, s_2)$
$p(s_4|s_1, s_2, s_3)$
$p(s_5|s_1, s_2, \ldots, s_4)$
$p(s_6|s_1, s_2, \ldots, s_5)$
$p(s_7|s_1, s_2, \ldots, s_6)$
$p(s_8|s_1, s_2, \ldots, s_7)$
$p(s_9|s_1, s_2, \ldots, s_8)$
Probability Distributions
$n_{sequence}$
vs
Target Data
The
cute
green
dragon
trot
ted
into
the
cave.
$n_{vocabulary}$
$[0,0,0, \ldots, 1, \ldots 0]$
with
• 1 for correct token s_i
• 0 for all others
Target Data One-Hot Encoded
$n_{sequence}$


Target Data
$n_{vocabulary}$
The
cute
green
dragon
trot
ted
into
the
cave.
$(s_2 | s_1)$
$(s_3 | s_1, s_2)$
$(s_4 | s_1, s_2, s_3)$
$(s_5 | s_1, s_2, \dots, s_4)$
$(s_6 | s_1, s_2, \dots, s_5)$
$(s_7 | s_1, s_2, \dots, s_6)$
$(s_8 | s_1, s_2, \dots, s_7)$
$(s_9 | s_1, s_2, \dots, s_8)$
$n_{sequence}$
vs
$[0,0,0, \dots, 1, \dots 0]$
with
• 1 for correct token s_i
• 0 for all others
$n_{sequence}$
Target Data One-Hot Encoded

Recap: Loss Function

The probability of a sequence is the product of the probability of the first token and the conditional probabilities of each subsequent token given all previous tokens.

A few more definitions:

- V_i … the random variable for a token at position i, and s_i is the actual token observed at that position.
- $p(V_i \,|s_1, \ldots, si_{-1})$ … the probability distribution over all possible tokens that will come next.
- y_i … a one-hot encoded vector of dimension equal to the vocabulary size holding 1 for the correct token at position i and 0 otherwise. This acts as the ground truth.

$$L = -\sum_i \mathrm{y}_i \log\big(p(V_i|s_1, \ldots, si_{-1})\big) = -\sum_i \log\big(p(s_i|s_1, \ldots, si_{-1})\big)$$

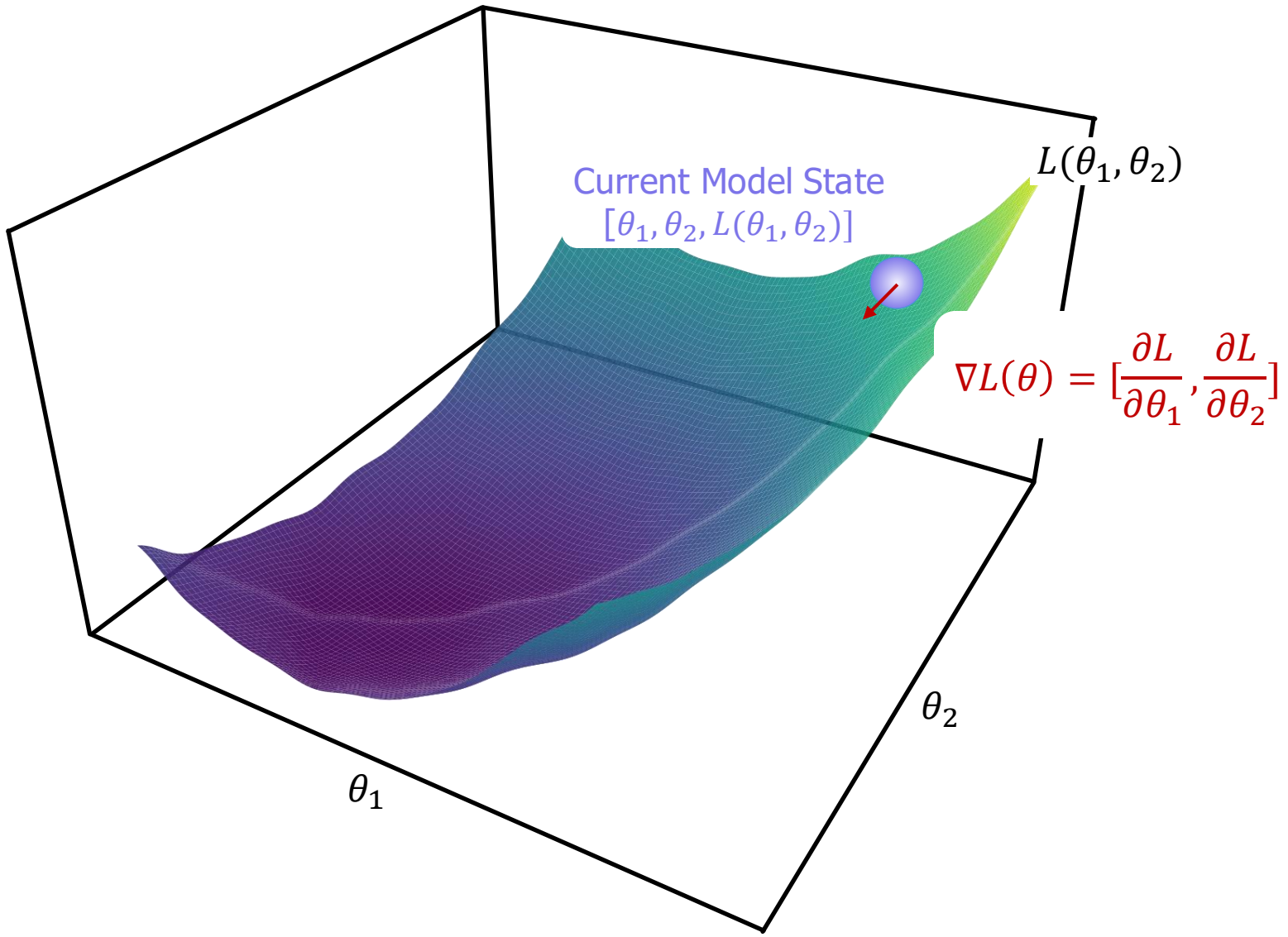
The loss function is defined as the negative log-likelihood of the correct tokens across all positions in the sequence, given the preceding tokens.

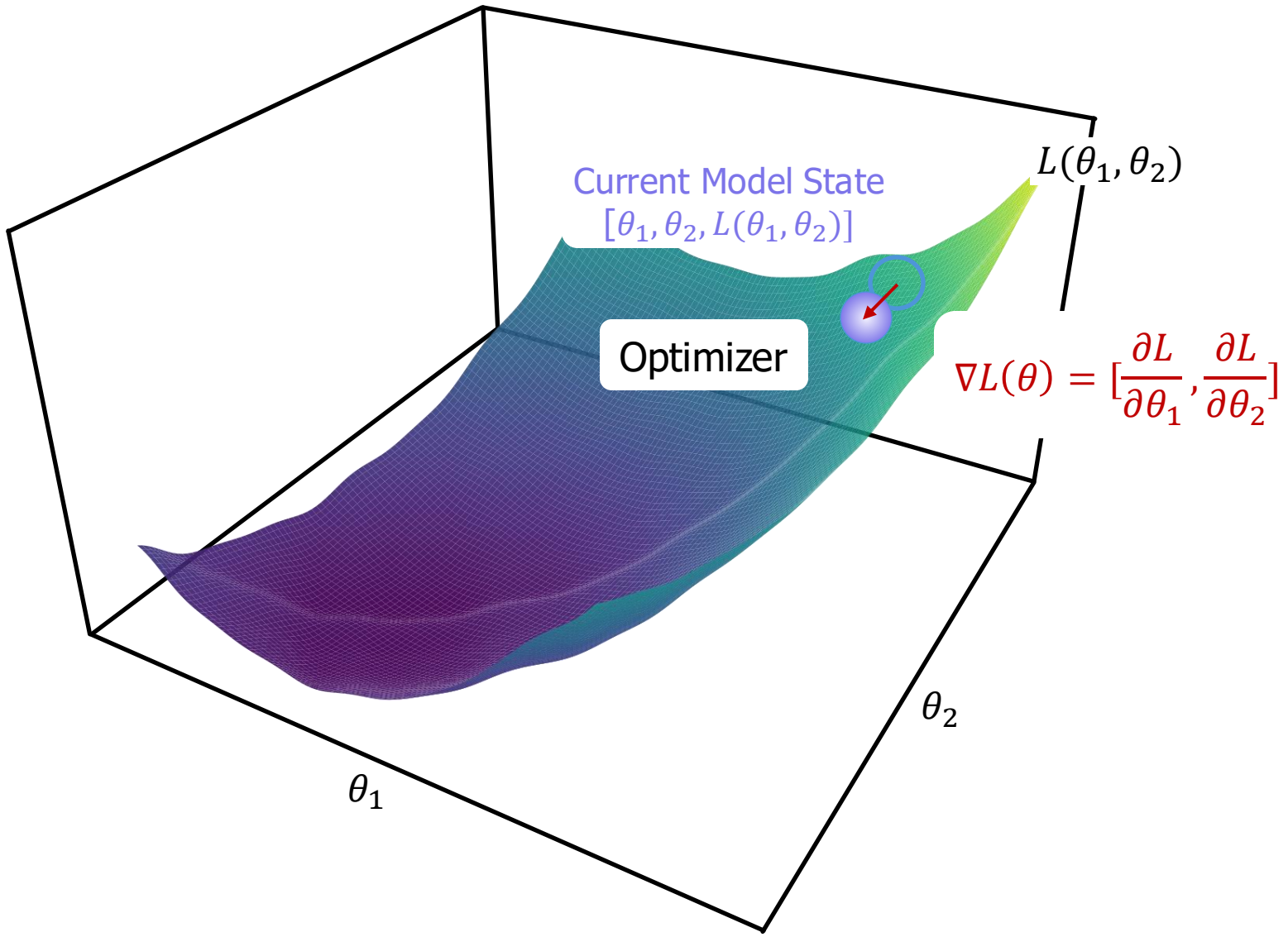
🎯 $L = -\sum_i y_i \log\big(p(V_i|s_1, \ldots, si_{-1})\big) = -\sum_i \log\big(p(s_i|s_1, \ldots, si_{-1})\big)$

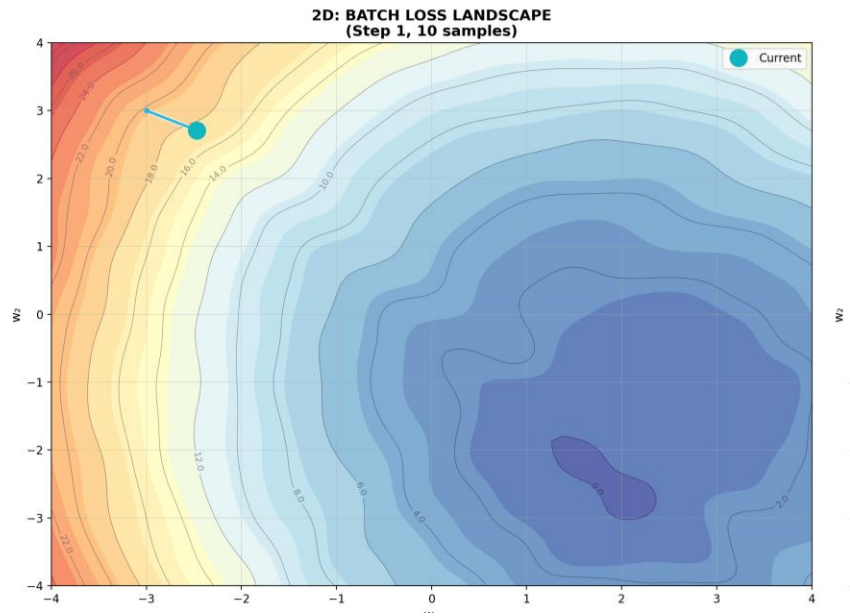
→ $\nabla L(\theta) = [\frac{\partial L}{\partial \theta_1}, \frac{\partial L}{\partial \theta_2}, \ldots, \frac{\partial L}{\partial \theta_n}]$

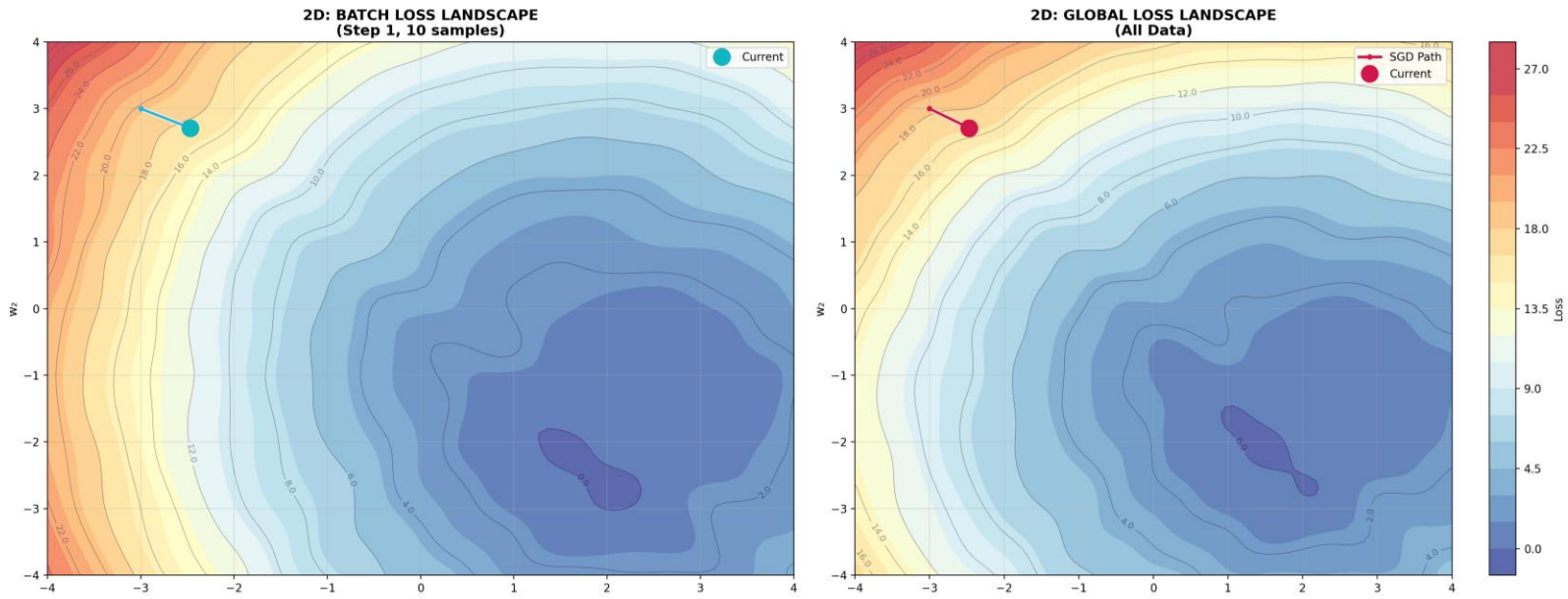
"How does the loss change in respect to parameter $\theta_1, \ldots \theta_n$"

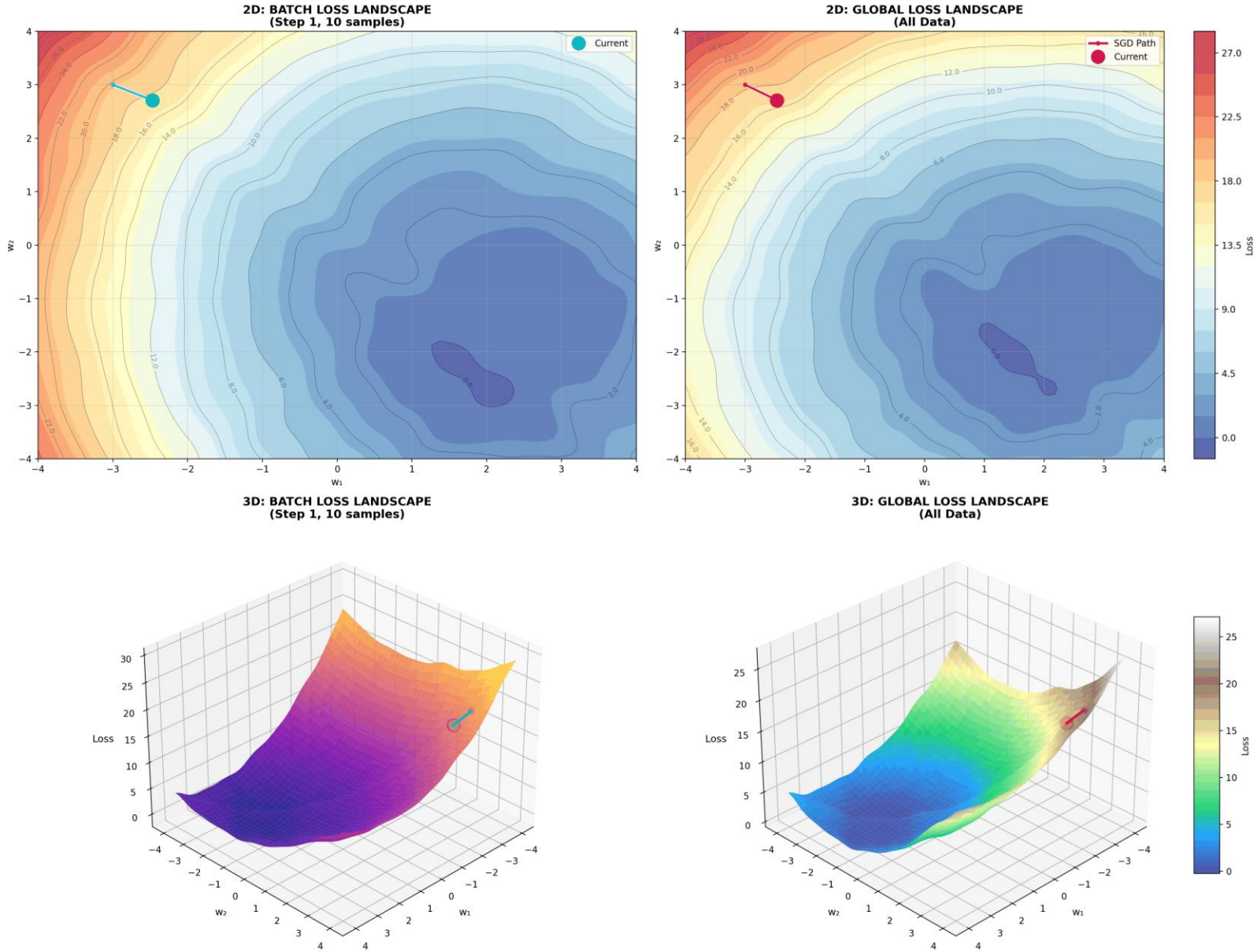
→ Optimizer (e.g. SDG) adapts weights based on gradient of loss


Current Model State
$[\theta_1, \theta_2, L(\theta_1, \theta_2)]$
$L(\theta_1, \theta_2)$
$\nabla L(\theta) = [\frac{\partial L}{\partial \theta_1}, \frac{\partial L}{\partial \theta_2}]$
θ_1
θ_2


Current Model State
$[\theta_1, \theta_2, L(\theta_1, \theta_2)]$
Optimizer
$L(\theta_1, \theta_2)$
$\nabla L(\theta) = [\frac{\partial L}{\partial \theta_1}, \frac{\partial L}{\partial \theta_2}]$
θ_1
θ_2


2D: BATCH LOSS LANDSCAPE
(Step 1, 10 samples)
Current
w₂
w₂
28.0
24.0
22.0
20.0
18.0
16.0
14.0
12.0
10.0
8.0
6.0
4.0
2.0
−4
−3
−2
−1
0
1
2
3
4


2D: BATCH LOSS LANDSCAPE
(Step 1, 10 samples)
Current
w₂
2D: GLOBAL LOSS LANDSCAPE
(All Data)
SGD Path
Current
w₂
Loss
27.0
22.5
18.0
13.5
9.0
4.5
0.0

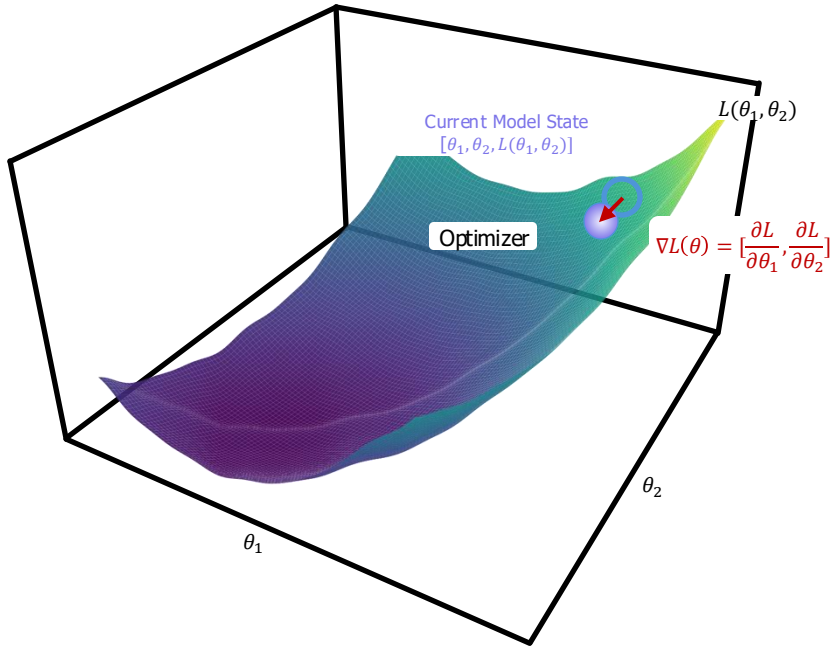

2D: BATCH LOSS LANDSCAPE
(Step 1, 10 samples)
Current
2D: GLOBAL LOSS LANDSCAPE
(All Data)
SGD Path
Current
Loss
w₁
w₂
3D: BATCH LOSS LANDSCAPE
(Step 1, 10 samples)
3D: GLOBAL LOSS LANDSCAPE
(All Data)

$$L = -\sum_i y_i \log\left(p(V_i|s_1, \ldots, si_{-1})\right) = -\sum_i \log\left(p(s_i|s_1, \ldots, si_{-1})\right)$$

$$\nabla L(\theta) = \left[\frac{\partial L}{\partial \theta_1}, \frac{\partial L}{\partial \theta_2}, \ldots, \frac{\partial L}{\partial \theta_n}\right]$$

"How does the loss change in respect to parameter $\theta_1, \ldots \theta_n$"

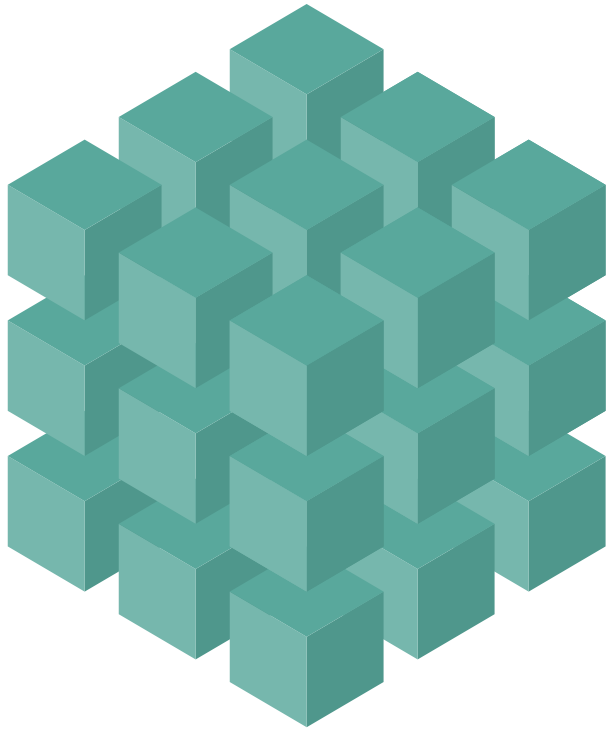
Optimizer (e.g. SDG, AdamW) adapts weights based on gradient of loss

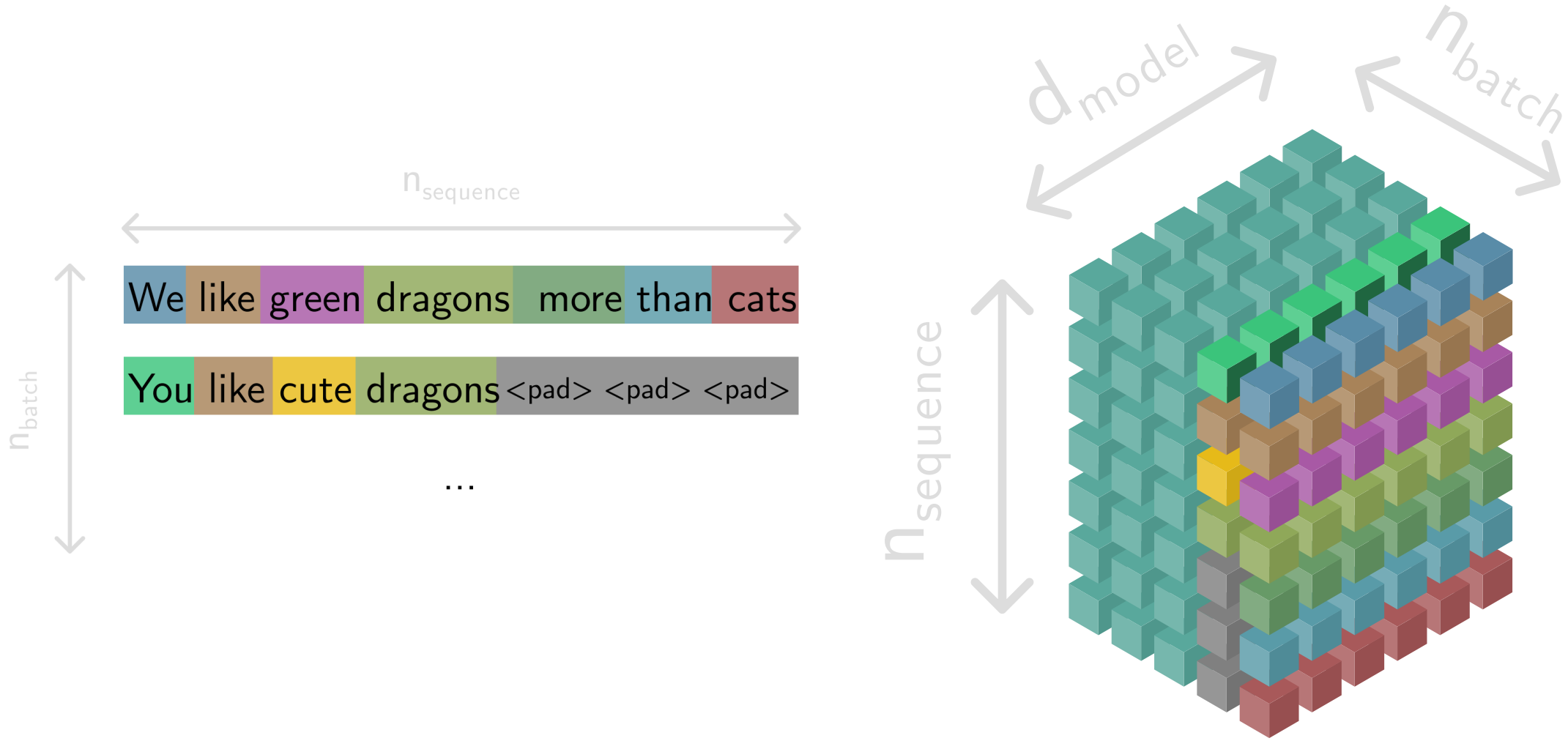


Side Note

Batching

Simultaneously process multiple training samples



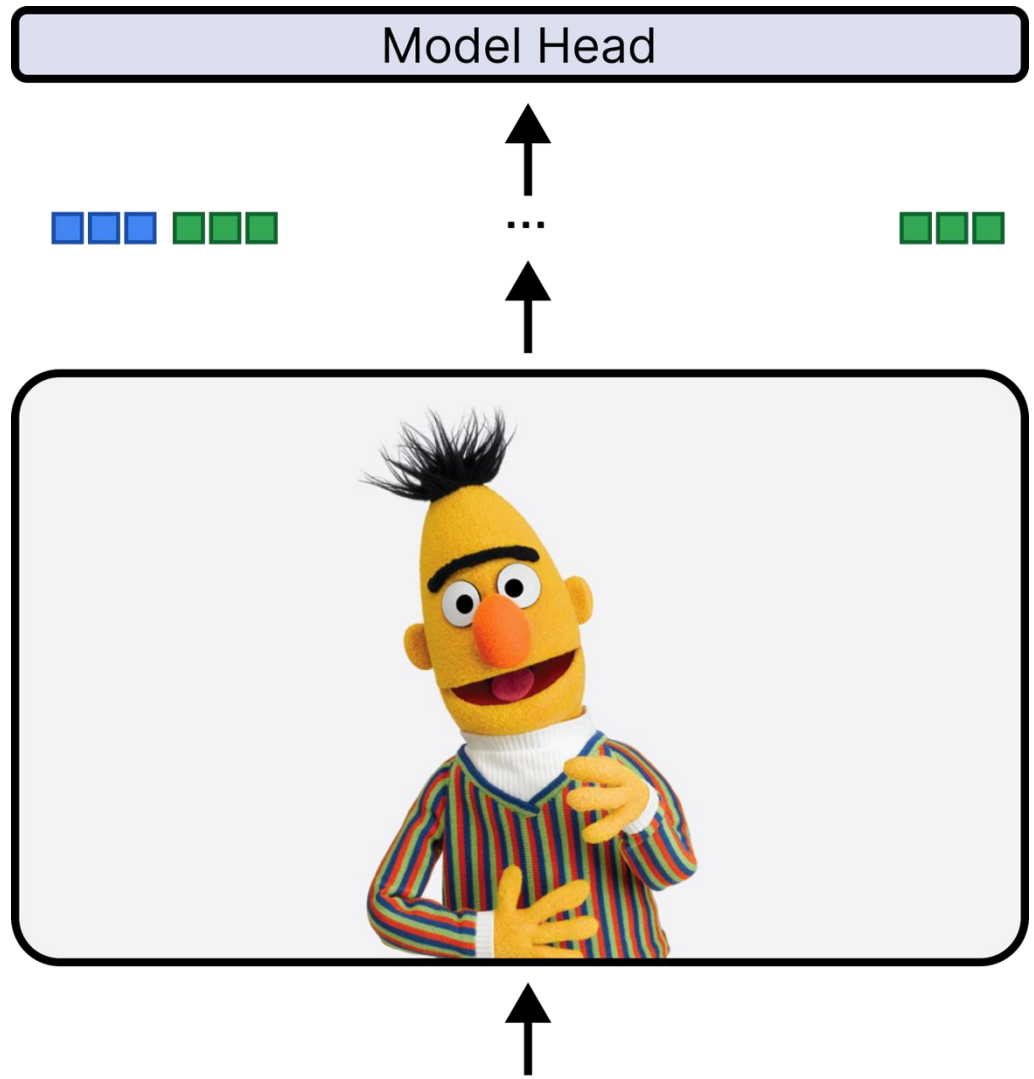

$n_{sequence}$
n_{batch}
We like green dragons more than cats
You like cute dragons <pad> <pad> <pad>
...
d_{model}
n_{batch}
$n_{sequence}$

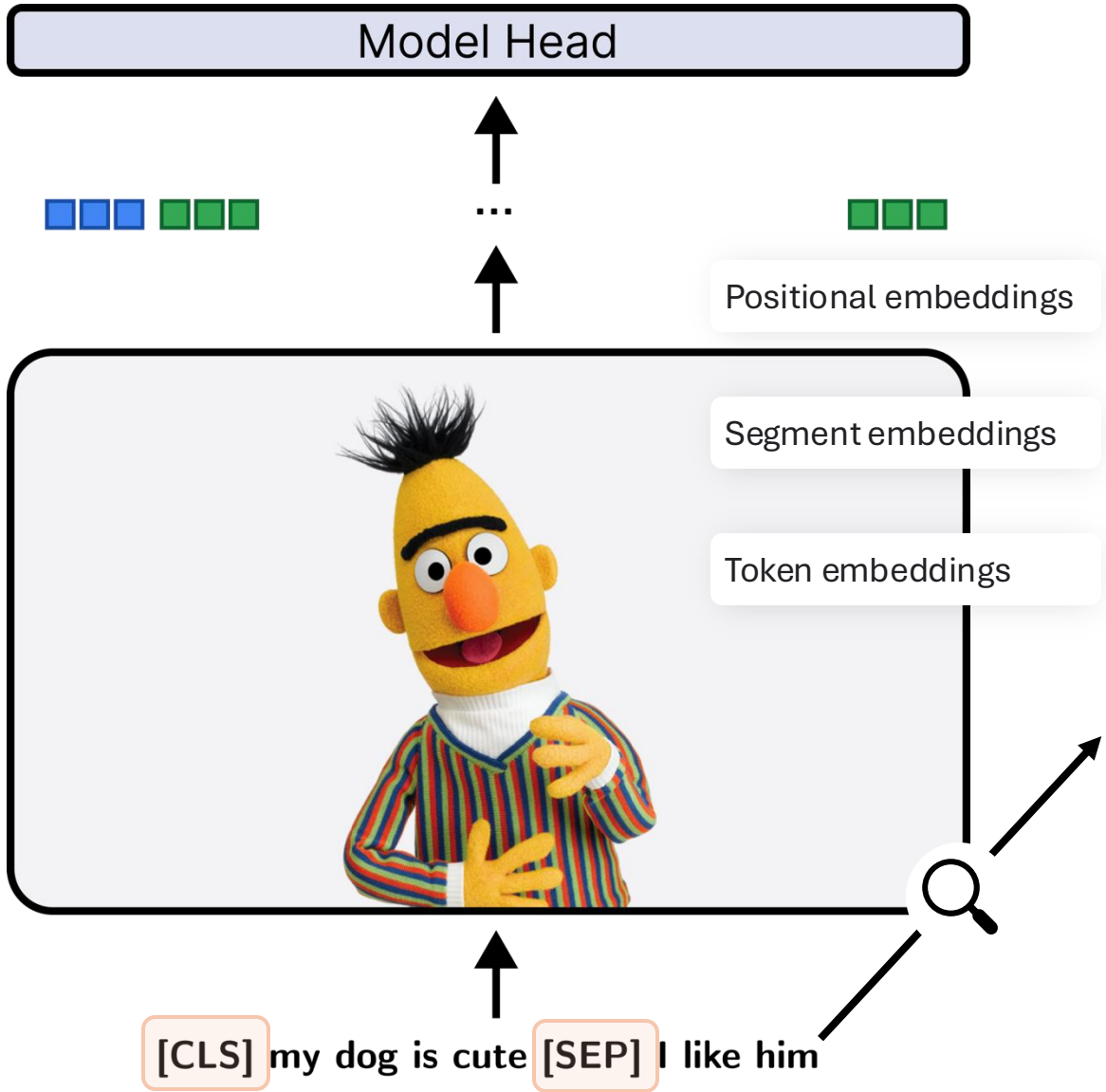
Training Objective For

Encoder Only Models

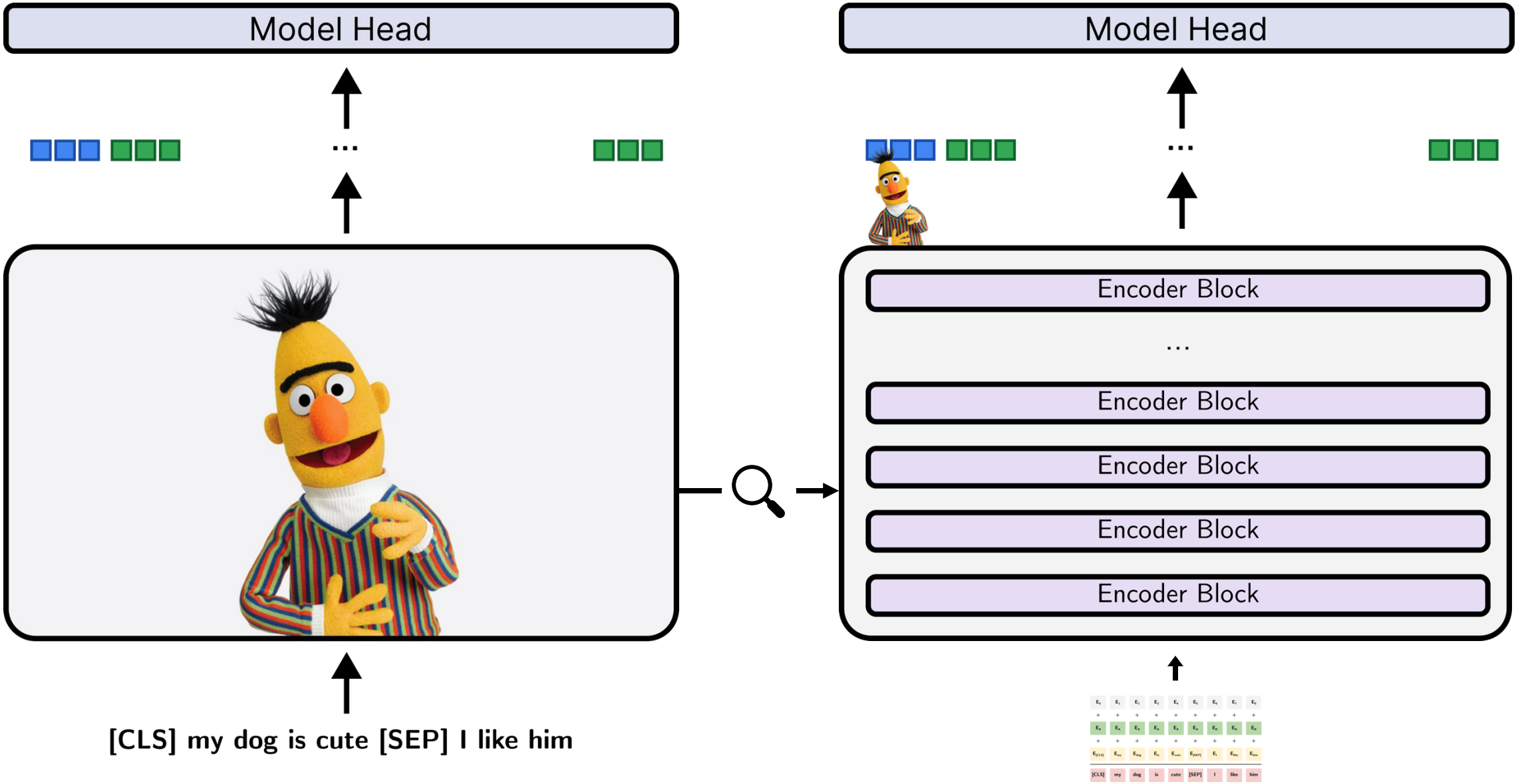
In reference to BERT - Pre-training of Deep Bidirectional Transformers for Language Understanding[1]

1https://arxiv.org/abs/1810.04805

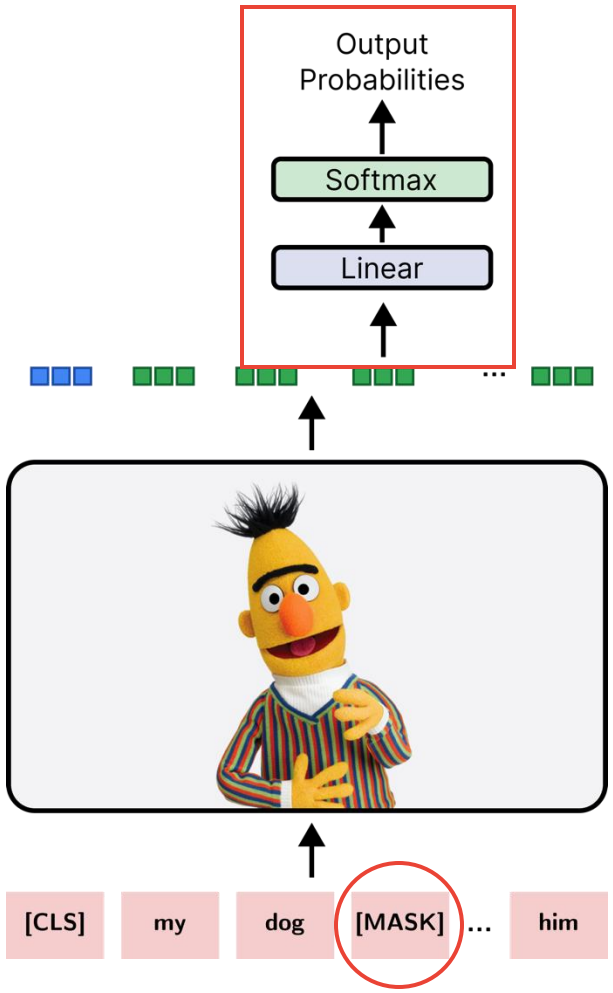

Model Head
...
[CLS] my dog is cute [SEP] I like him



Positional embeddings	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_9	[n_sequence, d_model]
+	+	+	+	+	+	+	+	+	+	
Segment embeddings	E_A	E_A	E_A	E_A	E_A	E_A	E_B	E_B	E_B	[n_sequence, d_model]
+	+	+	+	+	+	+	+	+	+	
Token embeddings	$E_{[CLS]}$	E_{my}	E_{dog}	E_{is}	E_{cute}	$E_{[SEP]}$	E_I	E_{like}	E_{him}	[n_sequence, d_model]
	[CLS]	my	dog	is	cute	[SEP]	I	like	him	[n_sequence]

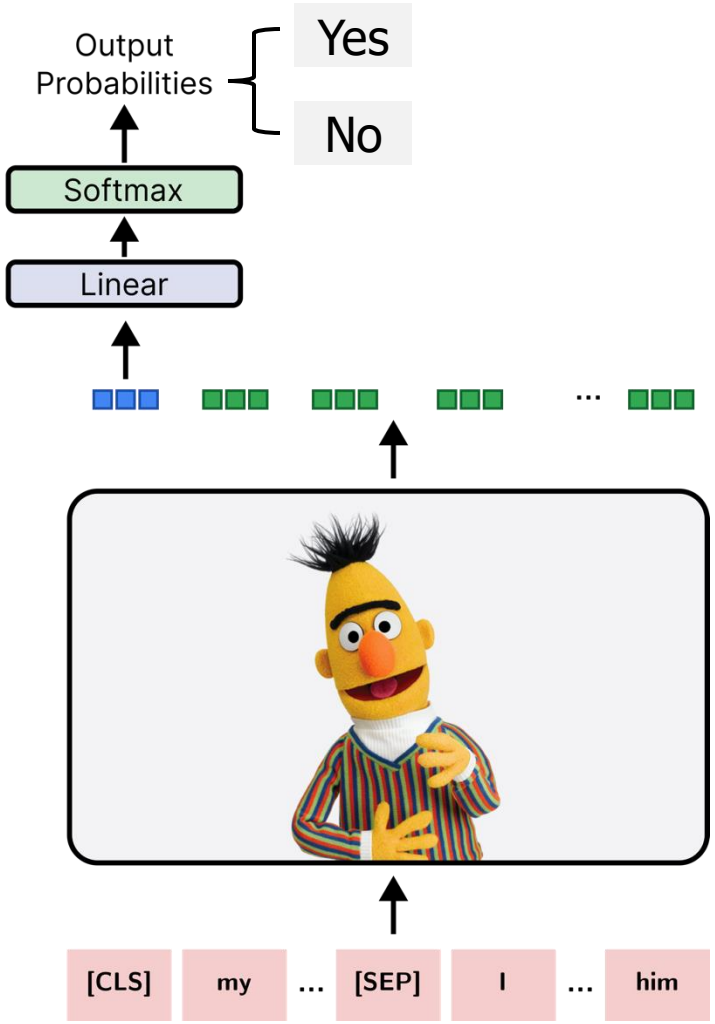

Model Head
Model Head
Encoder Block
Encoder Block
Encoder Block
Encoder Block
Encoder Block
[CLS] my dog is cute [SEP] I like him

Masked Token Prediction

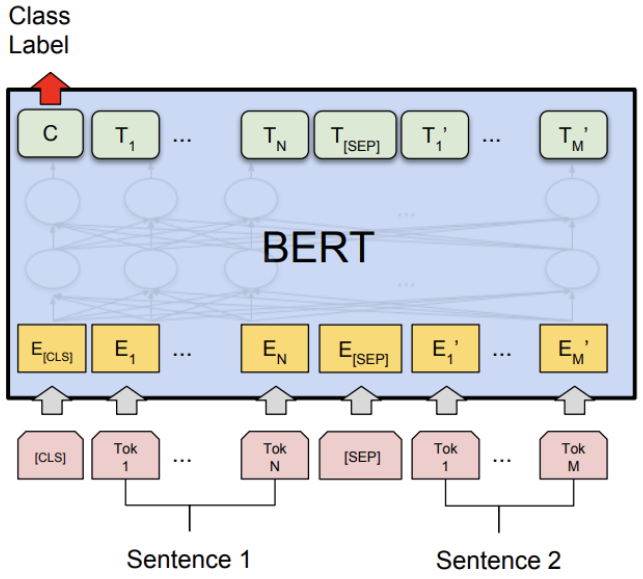


- 80% of the time: Replace the word with the *[MASK]* token
  To learn how words relate to their surrounding context to make accurate predictions.
- 10% of the time: Replace the token with a random token
  To prevent the model from simply learning that it should pay special attention to [MASK] tokens.
- 10% of the time: Keep the word unchanged
  To bias the representation towards the actual observed word.

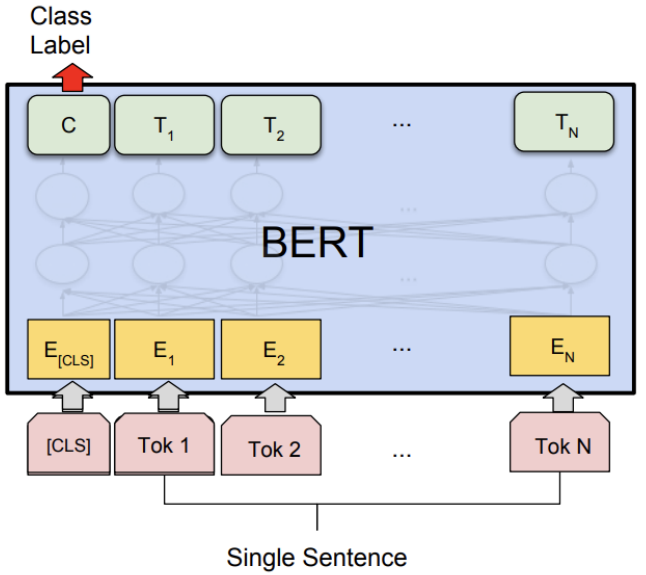
Next Sentence Prediction



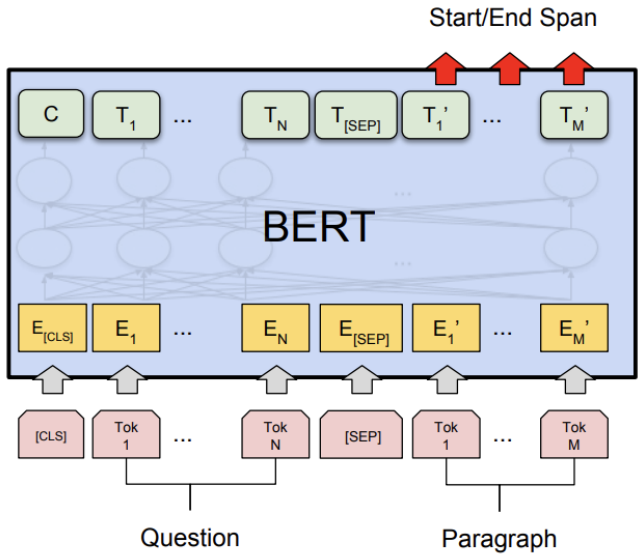
- 50% of the time actual next sentence is used 50% of the time it is a random sentence from the corpus
- [CLS] related output is used for binary classifcation



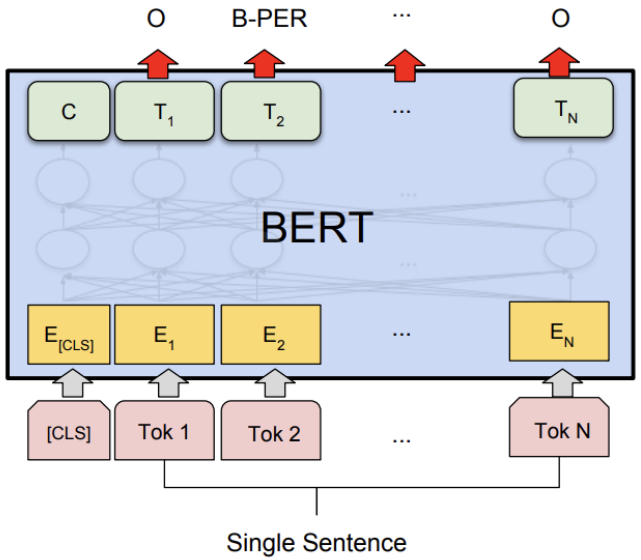
Sentence Pair Classification



Single Sentence Classification Tasks



Question Answering Tasks



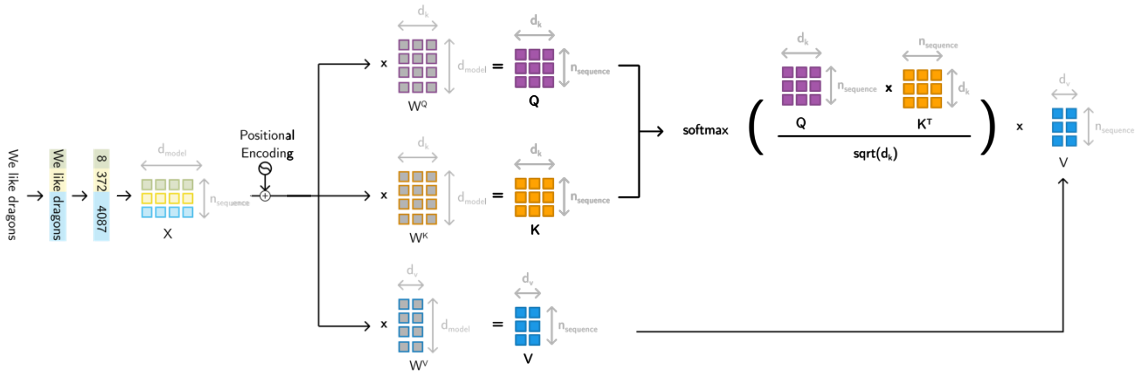
Single Token Tagging Tasks

What is so special about

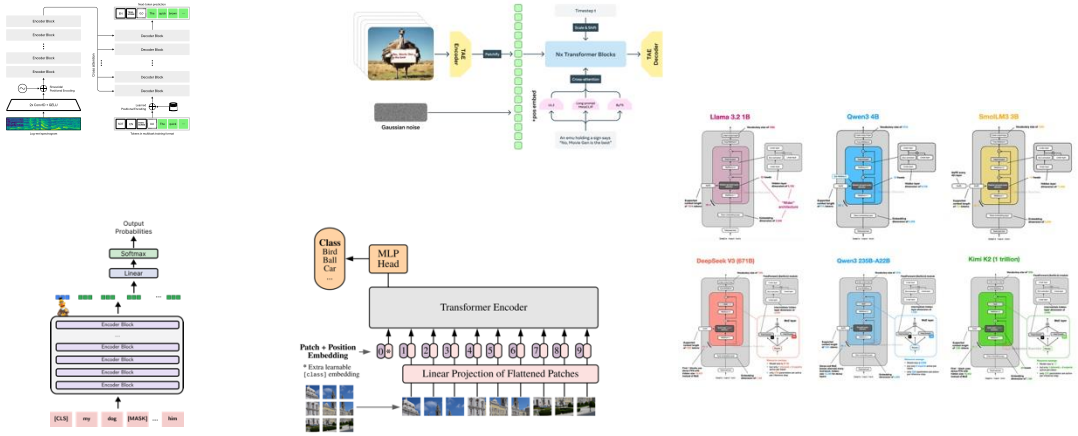
The Transformer Architecture?

1https://arxiv.org/abs/1810.04805

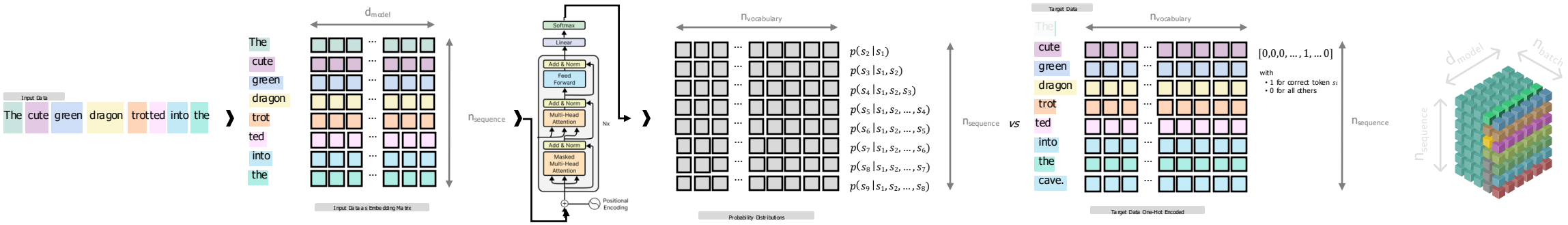
1. Long-range dependencies in sequences



3. Versatility



2. Fully parallelizable training paradigm



Further Resource

Allen Institute for AI

Pre-training: OLMo, OLMo 2, OLMoE
Post-training: Tülu 3
https://allenai.org/language-models

Annotated Transformer Implementation
https://nlp.seas.harvard.edu/annotated-transformer/

HuggingFace – LLM Course
https://huggingface.co/learn/llm-course

Jay Alammar – The Illustrated Transformer
https://jalammar.github.io/illustrated-transformer/

Sebastian Raschka – Build an LLM from Scratch
https://www.youtube.com/@SebastianRaschka