Introduction to LLMs

Let’s level set this…

By systems people…

…for systems people

By systems people…
(and the Internet)

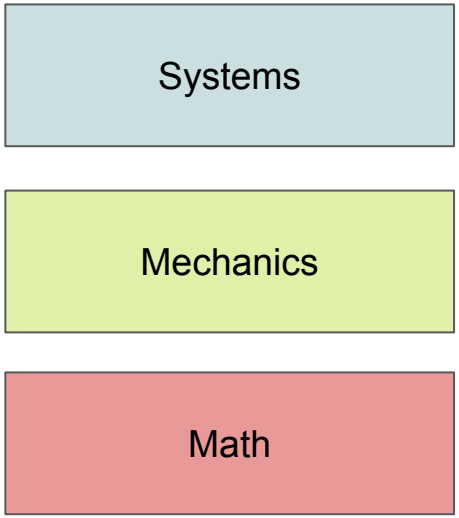
…for systems people

Systems

Mechanics

Math

https://www.kingarthurbaking.com/recipes/tiramisu-recipe



Flash Attention
(NeurIPS)

(powers
training)

Paged Attention
OSDI

(powers
inference)

Tomorrow

Systems

3 - Inference

4 - Dist. Training

Mechanics

1 - LLM Core

2 - Pytorch/GPUS

Today

Math

In terms of GPUs,

we have no GPUs

```
$ pip install torch transformers numpy scipy

$ hf download gpt2 model.safetensors

$ wget
https://huggingface.co/gpt2/resolve/main/pytorch_model.bin
```

Interactive Hands-on

No AI was used in the making of these slides.

… but can be used in the consumption of these slides.

LLM Core

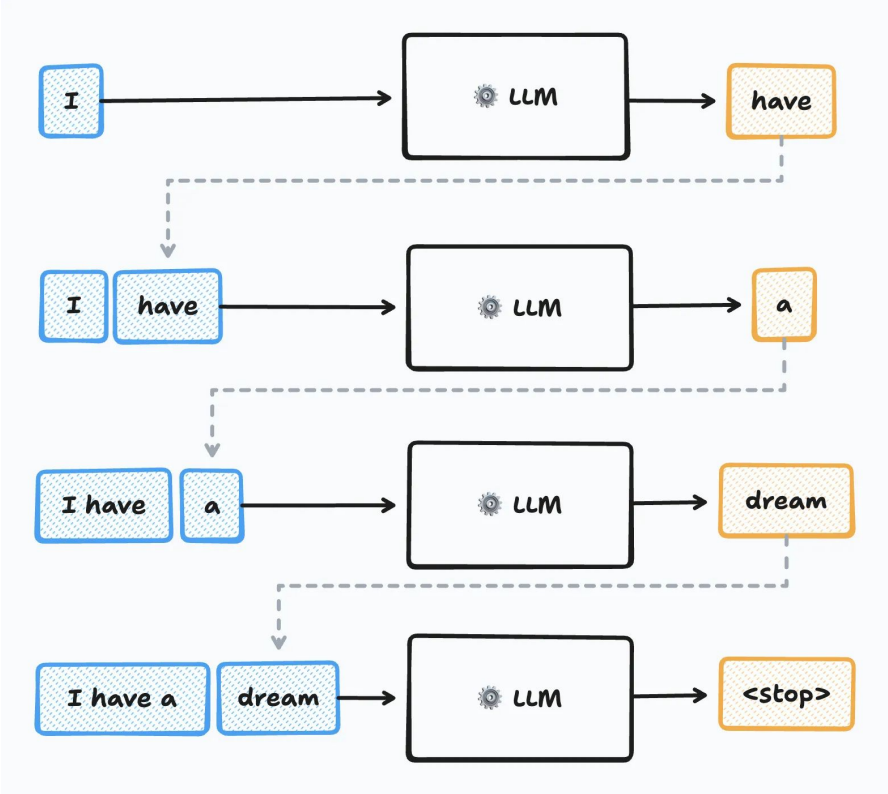
GenAI models!

Gemini

Language Models

- Probability distribution over words
  - *“IIT Bombay is an institute in the city of …….”*
- Enable machines to understand, generate human language
- What all can they do?
  - QnA
  - Content Generation
  - Text Summarization
  - Sentiment Analysis
  - Conversation
  - Translation
  - Code Generation

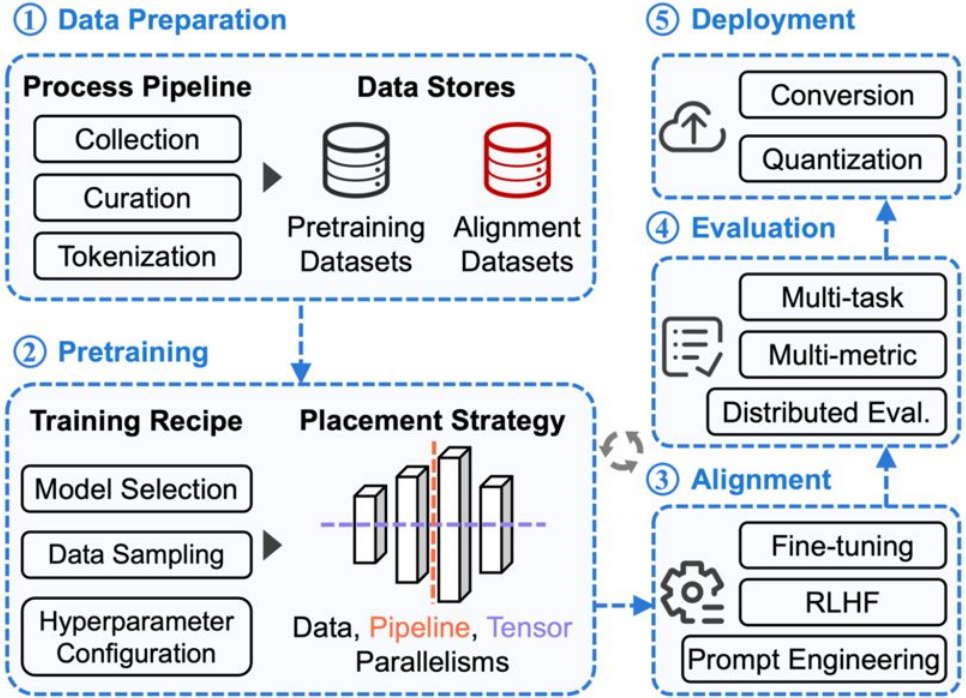
Causal Language Modelling or Next Token Prediction



https://pm.dartus.fr/posts/2025/how-llm-generate-text/

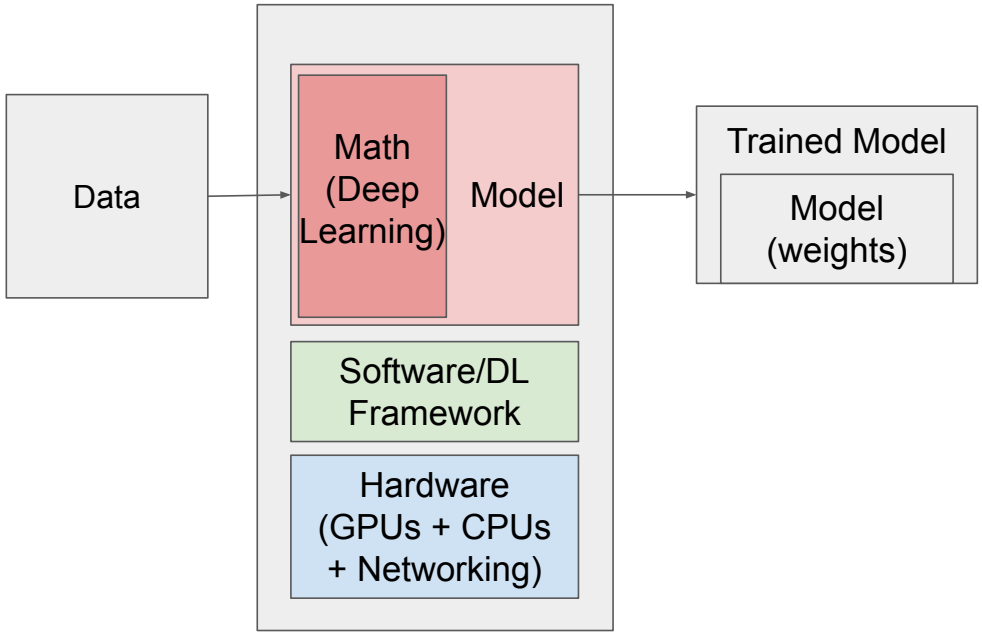
Model Creation Pipeline

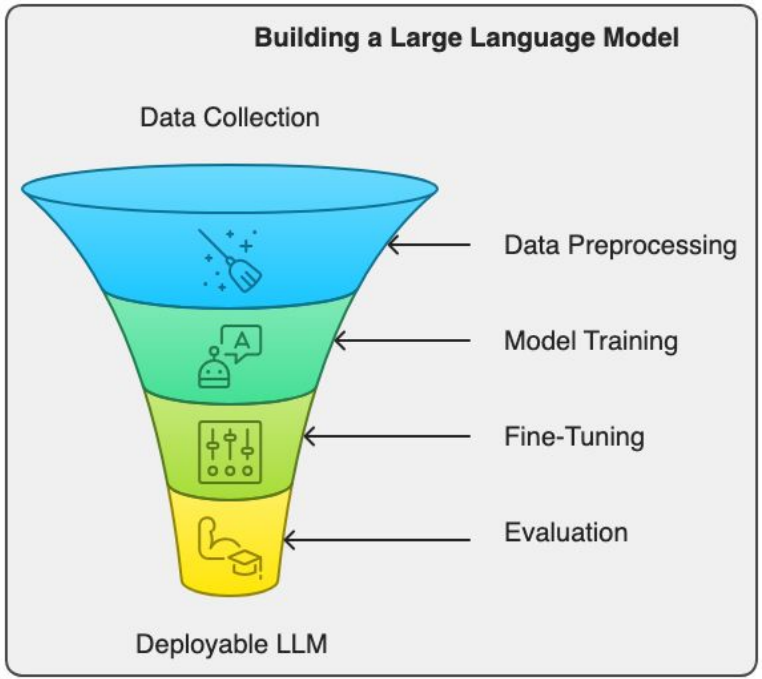
- **Data Preparation:** pretraining data, alignment data
- **Pretraining:** self-supervised training on large-scale curated data
- **Alignment:** Prompt Engineering, fine-tuning
- **Evaluation:** accuracy, fairness, and toxicity
- **Deployment:** Quntantization, model-parallelism and memory management



https://www.usenix.org/publications/loginonline/understanding-workload-characteristics-large-language-model-development

What does it take to train an LLM?





Training and Fine tuning

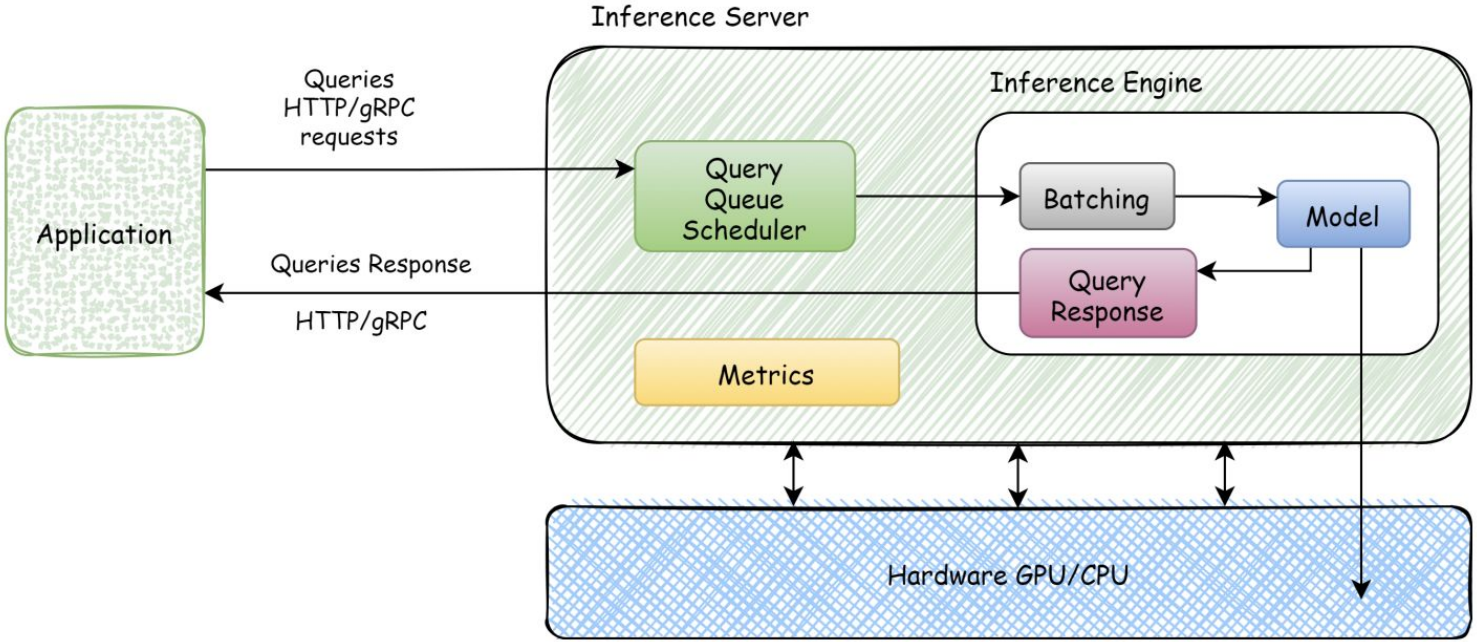
https://code-b.dev/blog/large-language-model

How do we use the models?

Model (weights)

Inference Platform

Hardware



Inference Server + Engine

https://ranjankumar.in/large-language-models-llms-inference-and-serving/

A partial timeline

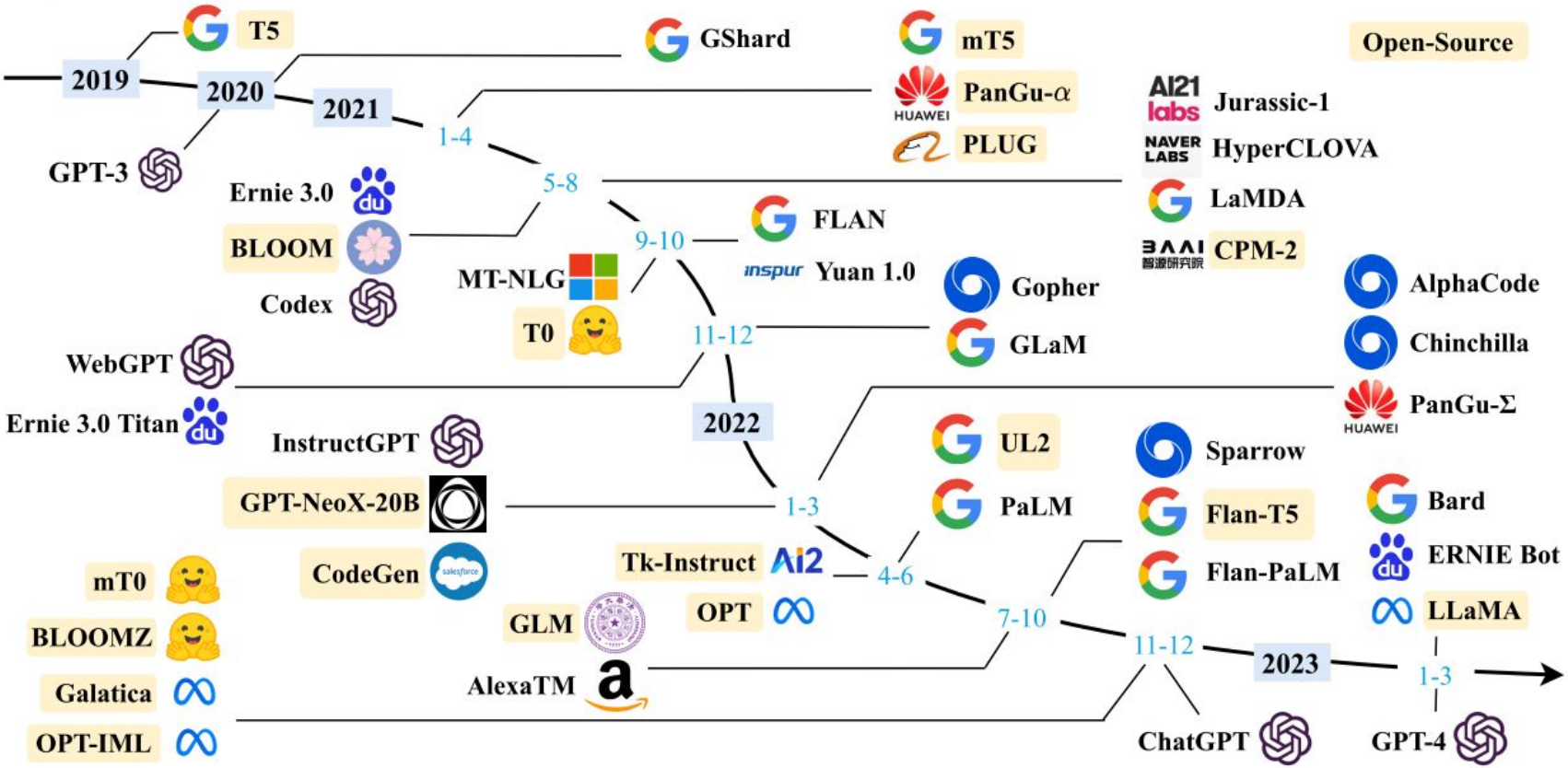
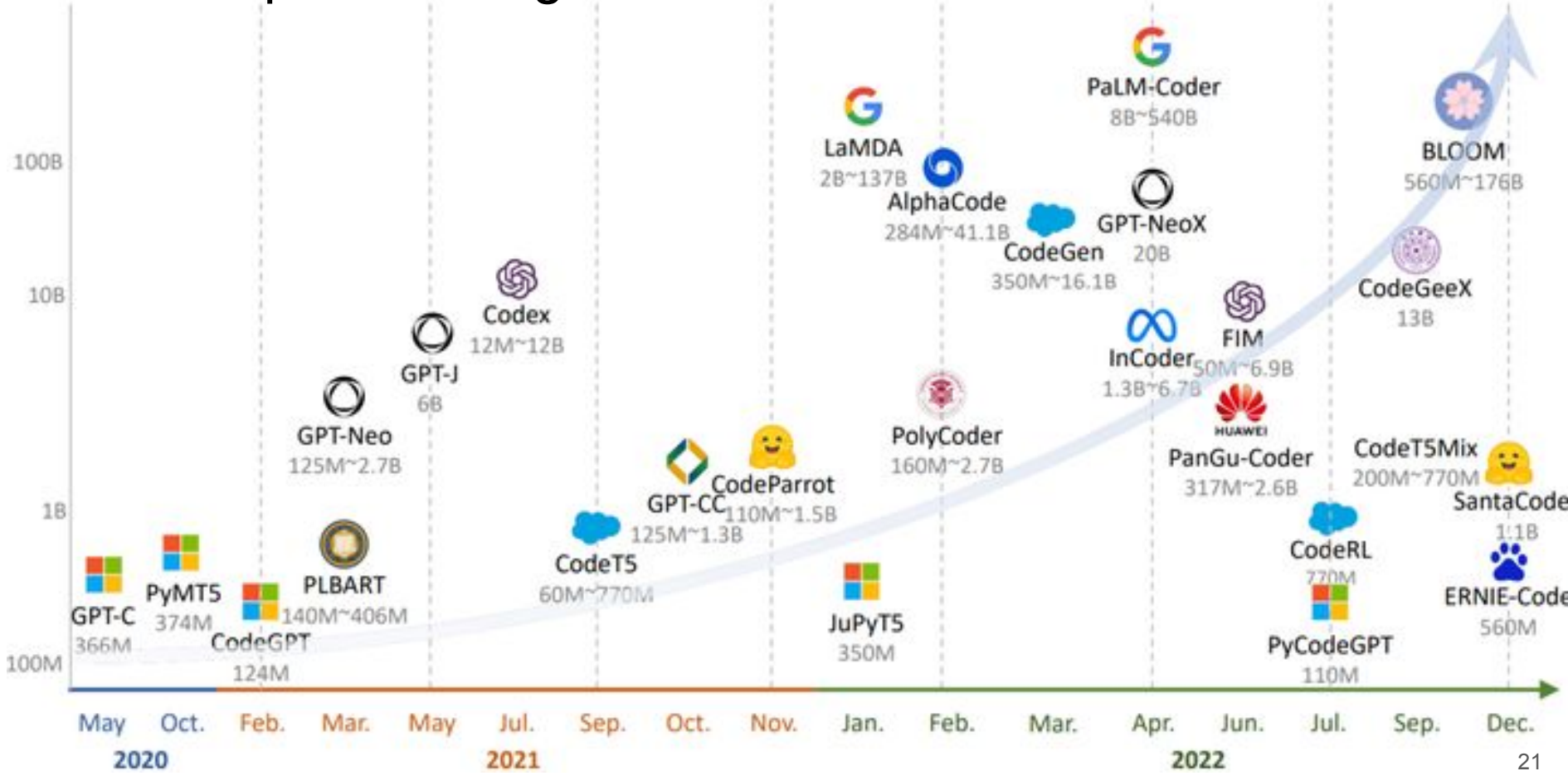


Fig. 1. A timeline of existing large language models (having a size larger than 10B) in recent years. We mark the open-source LLMs in yellow color.

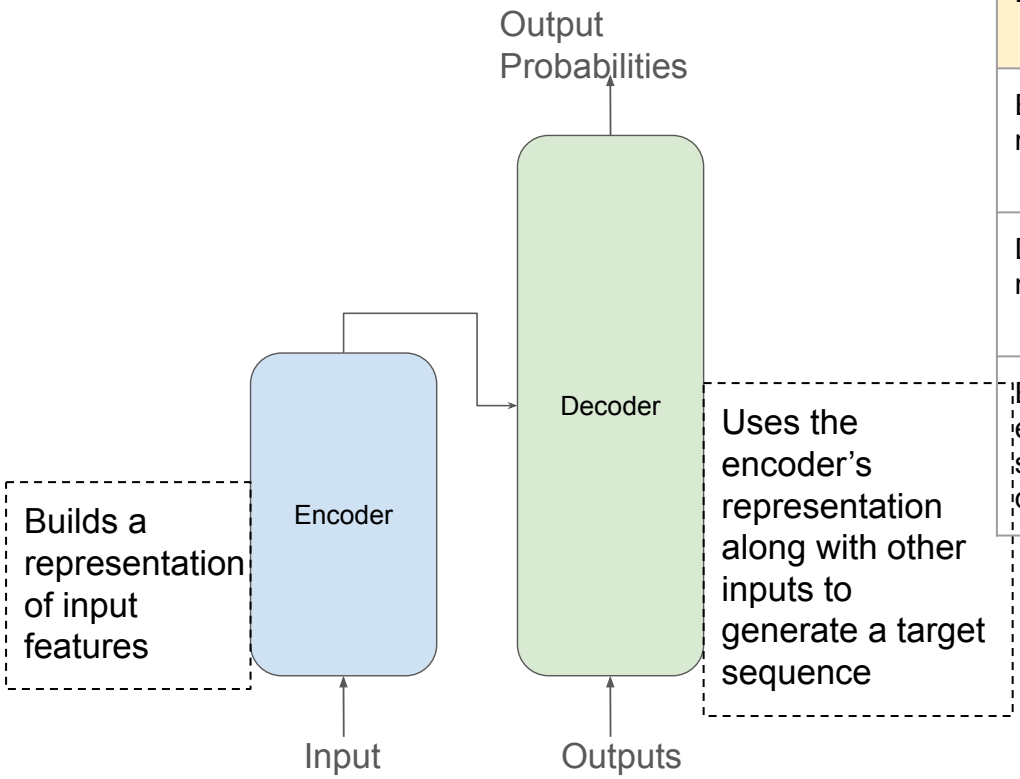
https://dnacap.fund/insights/exploring-the-landscape-of-large-language-models

Sizes keep increasing



https://dnacap.fund/insights/exploring-the-landscape-of-large-language-models

Types of LLM Models



Model type	Used for	Example tasks	Example Models
Encoder-only models	Understanding input	Classification, embeddings	BERT, RoBERTa, SBERT
Decoder-only models	Generative tasks	Text generation	GPT, LLaMA, Mistral
Encoder-decoder models/ sequence-to-sequence models	Generative tasks that require an input	Translation, summarization, QA	T5, BART, MarianMT

Each of these parts can be used independently depending on the task

https://huggingface.co/learn/llm-course/chapter1/4

Transformer: Base architecture of all LLMs today!!

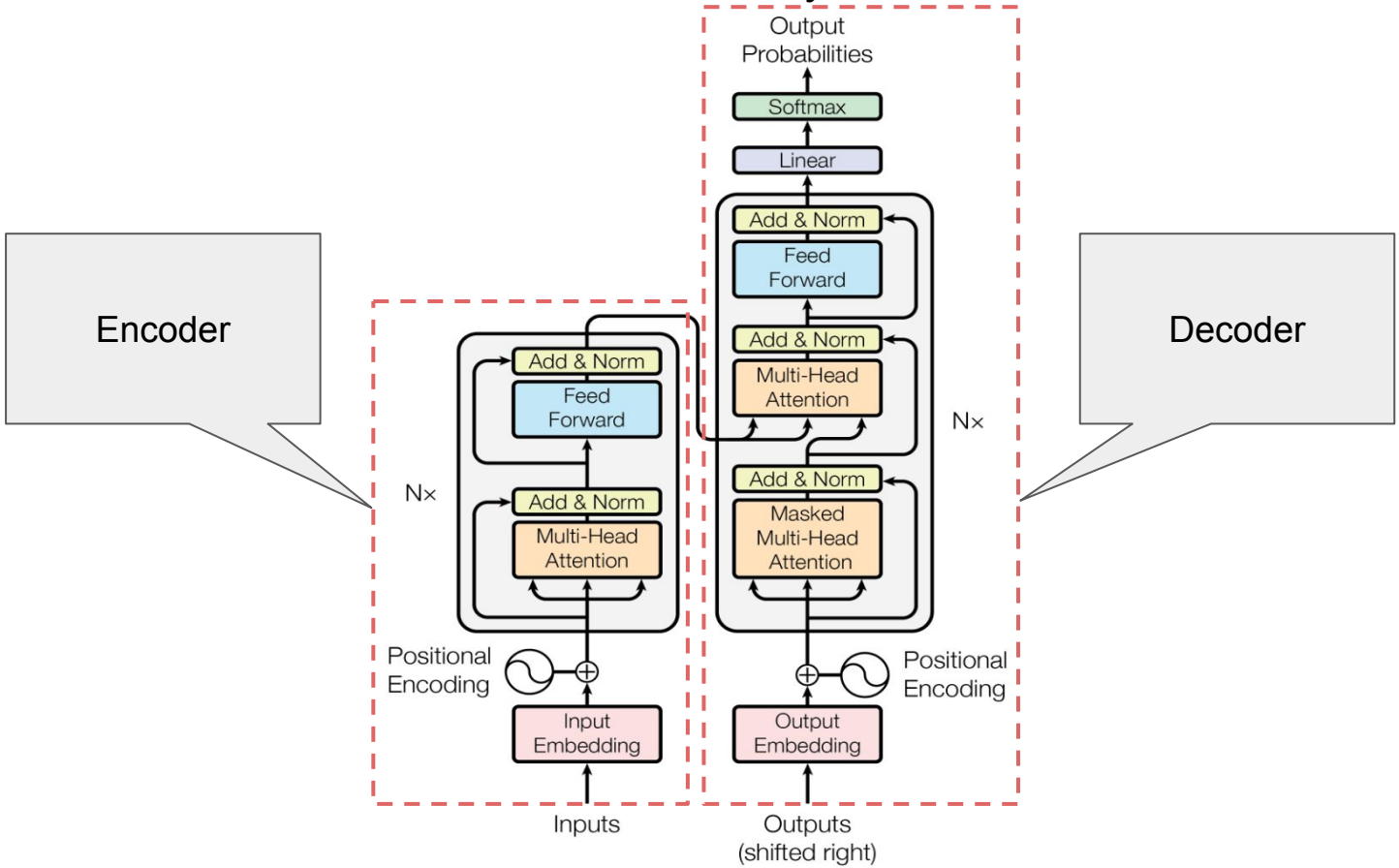


Figure 1: The Transformer - model architecture.

Image source: Attention Is All You Need

How to use an LLM: First Look

What is hugging face? 

- Global hub for Large Language models
  - Spaces (try the models)
  - Models: 1.8M models
  - git clone <modelurl>

https://huggingface.co/

Hands-on: the simplest hello world

```
from transformers import pipeline
pipe = pipeline(model="gpt2")
```

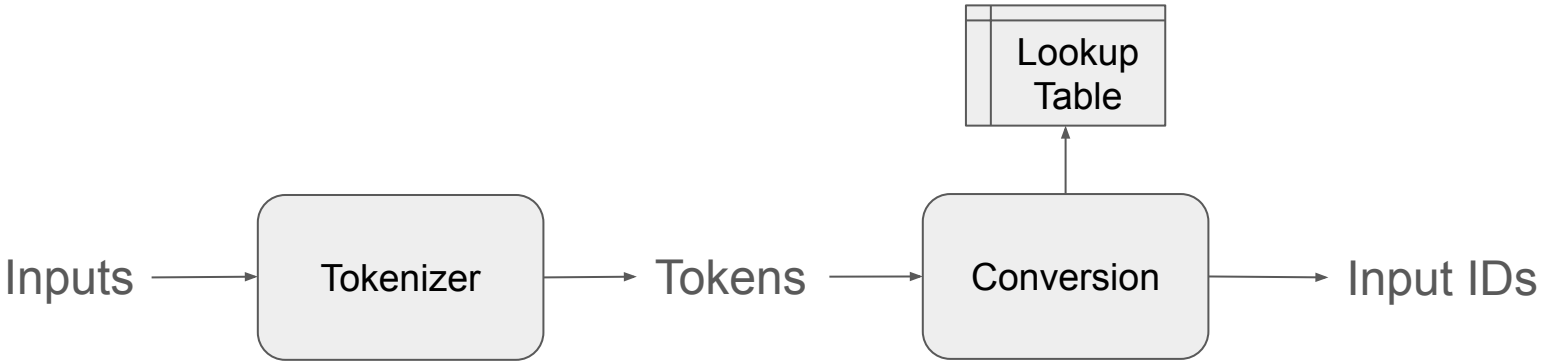
```
pipe("Paris is the capital of")
```

-> HF Pipeline
-> HF Model
-> Pytorch Model

First, let’s look at inputs

Tokenizer

- Prepare inputs for models:



When running inference on a model, it is necessary to use the same tokenizer to tokenize the inputs as that was used for pretraining the model.

Tokenizer

- Types
  - Word-level

We are learning systems for LLMs.

[‘We’, ‘are’, ‘learning’, ‘systems’, ‘for’, ‘LLMs’, ‘.’]

Problems

1. Separate tokens for different versions of same word - e.g., ‘system’ and ‘systems’
2. Results in large vocabulary
3. Limiting vocabulary results in bad performance

Tokenizer

- Types
  - Character-level

We are learning systems for LLMs.

['W', 'e', ' ', 'a', 'r', 'e', ' ', 'l', 'e', 'a', 'r', 'n', 'i', 'n', 'g', ' ', 's', 'y', 's', 't', 'e', 'm', 's', ' ', 'f', 'o', 'r', ' ', 'L', 'L', 'M', 's', '.']

Problems

1. Tokens contain minimal semantic information individually.
2. Produces longer sequences, limiting the effective input size of language models.

Tokenizer

- Types
  - Subword-level
    - Breaks text into units that are smaller than words but larger than characters.
    - Frequently-used words stored as entire tokens
    - Rare or unknown words are split into smaller, meaningful chunks ("cats" → "cat" + "s").

Tokenizer code snippet

```
from transformers import AutoTokenizer
tok = AutoTokenizer.from_pretrained("gpt2")
text = “Hello, how are you?”
print(tok(text))
print(tok.tokenize(text))
```

Display input_ids

Display tokens

1. https://huggingface.co/openai-community/gpt2/blob/main/tokenizer.json
2. https://huggingface.co/openai-community/gpt2/blob/main/vocab.json

Tokenizer hands-on

Tokenize the following:

1. cat
2. superman
3. spiderman

   Examine the number of tokens

See the tokenizer config.json / vocab.json:
https://huggingface.co/openai-community/gpt2/tree/main

What is a model?

1. Architecture
2. Configuration
3. Code (in a DL framework)
4. Weights

Also known as checkpoints

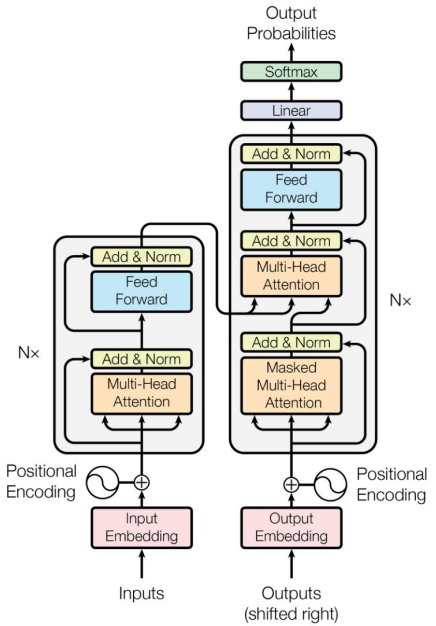
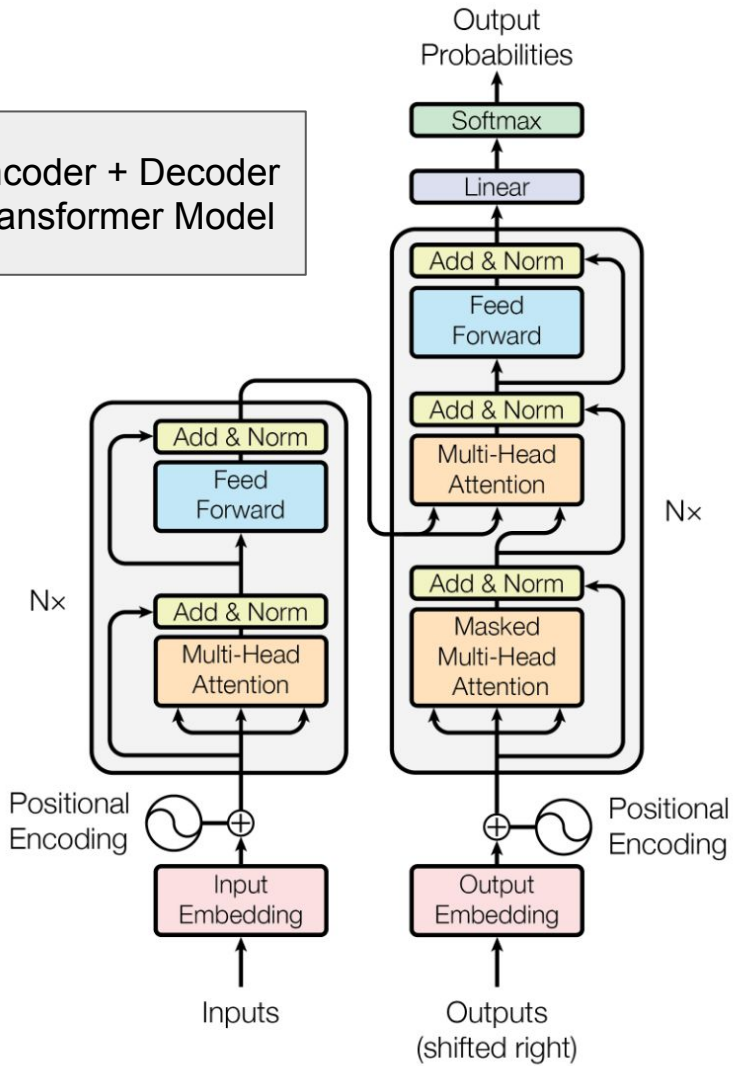


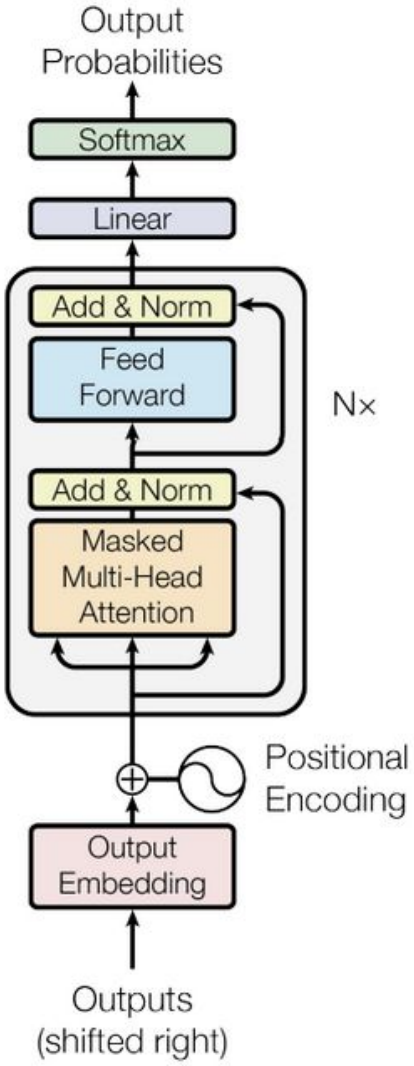
Figure 1: The Transformer - model architecture.

Obtained after extensive training

Encoder + Decoder Transformer Model



GPT2 (Decoder Only)



Try it out: config/dummy model

```
from transformers import
GPT2Config, GPT2LMHeadModel

config = GPT2Config()


m1 = GPT2LMHeadModel(config)
m1
m1.lm_head.weight
```



```
GPT2Config {
  "activation_function": "gelu_new",
  "attn_pdrop": 0.1,
  "bos_token_id": 50256,
  "embd_pdrop": 0.1,
  "eos_token_id": 50256,
  "initializer_range": 0.02,
}
```

```
GPT2LMHeadModel(
  (transformer): GPT2Model(
    (wte): Embedding(50257, 768)
    (wpe): Embedding(1024, 768)
    (drop): Dropout(p=0.1,
inplace=False)
    (h): ModuleList(
      (0-11): 12 x GPT2Block(
…)
)
```

Try it out: loading weights

```
from transformers import AutoModelForCausalLM

m2 = AutoModelForCausalLM.from_pretrained("gpt2")

m2.lm_head.weight
```

Check the weights of the model

Try for another model (big model)

```
from transformers import AutoConfig

config =  AutoConfig.from_pretrained("Qwen/Qwen3-32B")
print(config)
```

Don’t do these!

```
from transformers import AutoModelForCausalLM

model = AutoModelFromCausalLM.from_config(config)
model
```

~70 GB

```
from transformers import AutoModelForCausalLM

model = AutoModelFromCausalLM.from_pretrained(config)
```

Let’s download the weights - cutting out HuggingFace

wget https://huggingface.co/gpt2/resolve/main/pytorch_model.bin

README.md Safe	8.09 kB	Add note that this is the smallest versi…	almost 3 years ago
config.json Safe	665 Bytes	Update config.json	over 5 years ago
flax_model.msgpack	498 MB xet	add gpt2	over 4 years ago
generation_config.json Safe	124 Bytes	Update generation_config.json	over 2 years ago
merges.txt Safe	456 kB	Update merges.txt	over 6 years ago
model.safetensors Safe	548 MB xet	Upload model.safetensors with huggingfac…	almost 3 years ago
pytorch_model.bin Safe pickle	548 MB xet	Update pytorch_model.bin	over 6 years ago
rust_model.ot	703 MB xet	Update rust_model.ot	over 5 years ago
tf_model.h5	498 MB xet	Update tf_model.h5	almost 6 years ago
tokenizer.json Safe	1.36 MB	Update tokenizer.json	almost 5 years ago

Loading and examining the State Dict

```
import torch

mw = torch.load(“pytorch_model.bin”)

list(mw.keys())[:10]

mw['wte.weight']
```

```
['wte.weight',
 'wpe.weight',
 'h.0.ln_1.weight',
 'h.0.ln_1.bias',
 'h.0.attn.bias',
 'h.0.attn.c_attn.weight',
 'h.0.attn.c_attn.bias',
 'h.0.attn.c_proj.weight',
 'h.0.attn.c_proj.bias',
 'h.0.ln_2.weight']
```

Using these weights

Use nn.Module.load_state_dict

More on this later, for now:
We have an hierarchy of modules

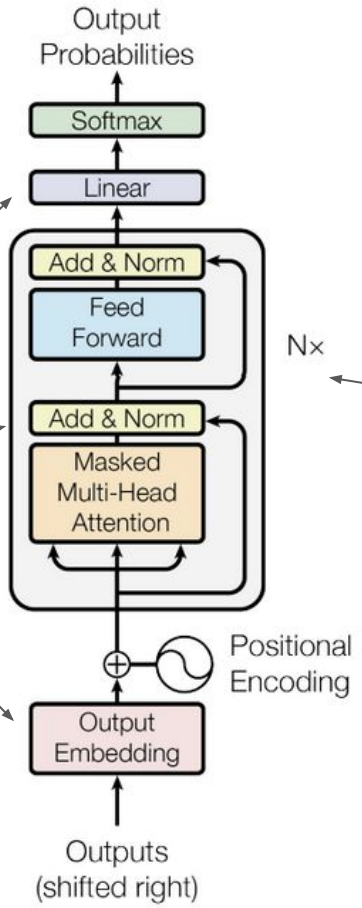
```
def extract(sd, prefix):
  prefix = prefix + “.”
  return {k.split(prefix)[1]: v for k, v in sd.items()
if k.startswith(prefix)}
```

HF
(Application)

Pytorch prereqs: nn.Module

Pytorch
(DL Framework)

All components and models are nn.Modules

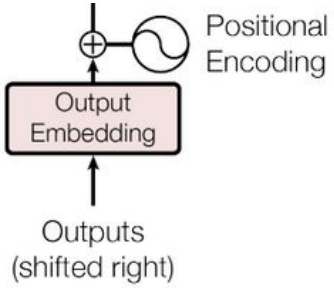


nn.Module

torch.nn.ModuleList

Each Module has a “forward” method

Embedding



- Transformation of input tokens into a high-dimensional vector space
  - Capture semantic relationship between tokens
  - Dimensionality:
    - Smaller vectors (lower dimensions) – more efficient to keep in memory or to process,
    - Bigger vectors (higher dimensions) – capture intricate relationships, prone to overfitting.

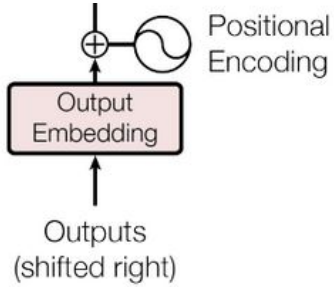
```
import torch.nn as nn
embedding_layer = nn.Embedding(10000, 128)

input_indices = torch.tensor([10, 25, 300])

output_embeddings =
embedding_layer(input_indices)
```

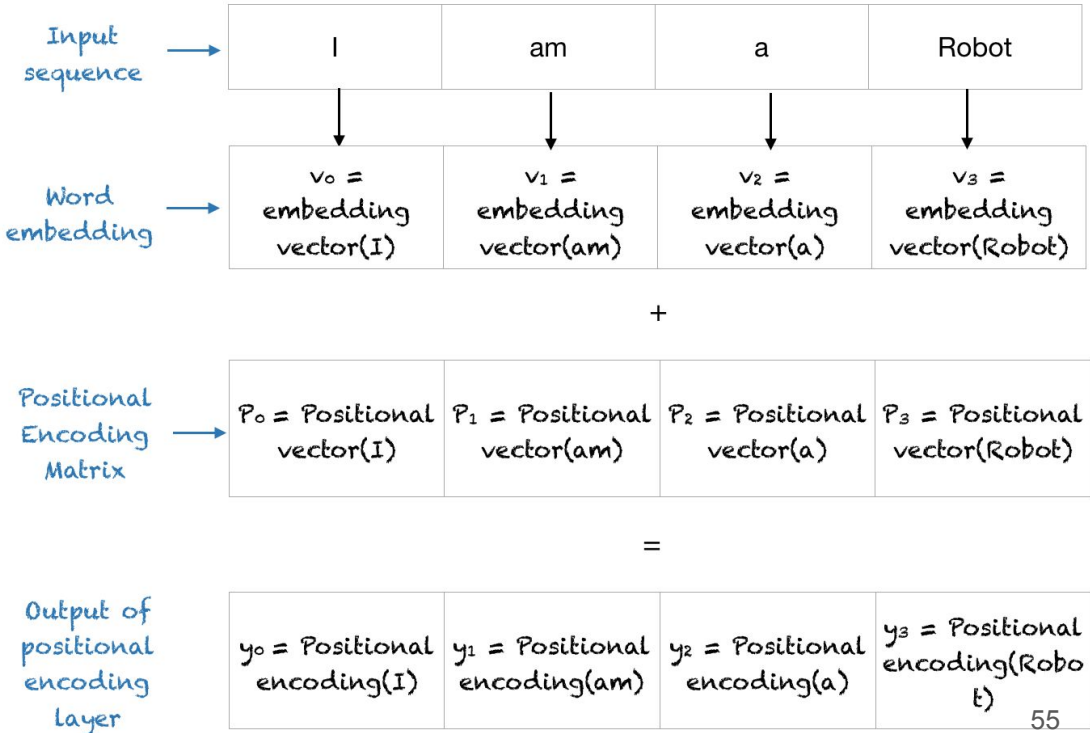
```
output_embeddings.shape
torch.Size([3, 128])
```

What will happen if you try to embed “20000”?

Embedding

- Positional Encoding
  - Information about the positions of the tokens added into embeddings to specify the order
  - Eg: RoPE used in Llama





https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/

Embedding - using the real weights

```
l = torch.nn.Embedding(50257, 768)
l.load_state_dict(extract(mw, “wte”))

<All keys matched successfully>

apple = l(torch.tensor(tok("apple")["input_ids"]))
fruit = l(torch.tensor(tok("fruit")["input_ids"]))
man = l(torch.tensor(tok("man")["input_ids"]))
woman = l(torch.tensor(tok("man")["input_ids"]))
```

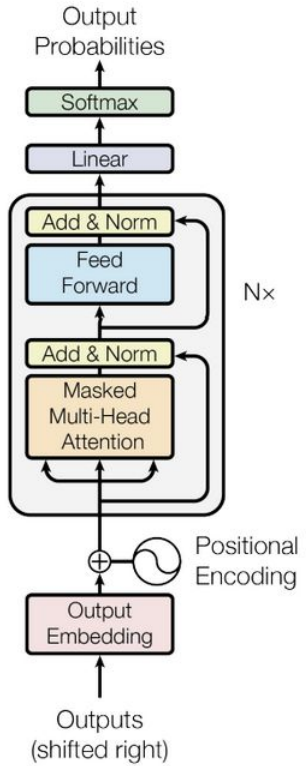
```
>>> torch.cosine_similarity(man, apple)
tensor([0.1757], grad_fn=<SumBackward1>)
>>> torch.cosine_similarity(fruit, apple)
tensor([0.3985], grad_fn=<SumBackward1>)
>>> torch.cosine_similarity(man, woman)
tensor([0.5863], grad_fn=<SumBackward1>)
```

Linear

```
linear_layer = torch.nn.Linear(10, 20)

input = torch.randn(5, 10)
output = linear_layer(input)

print(linear_layer.weight)
print(linear_layer.bias)
```

Y = WX + B

GPT2 uses Conv1D instead of linear

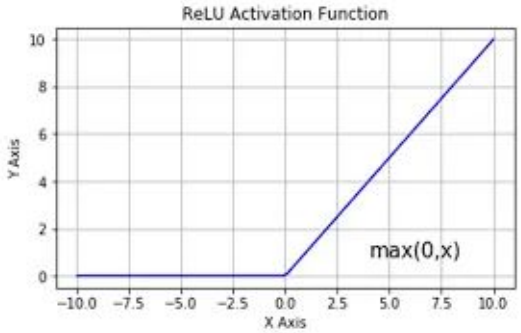


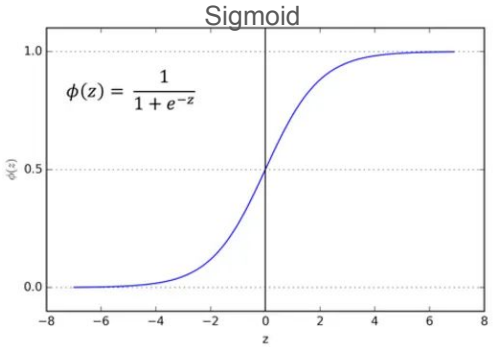
https://medium.com/@muhammadraflyindrawan/understanding-multilayer-perceptron-the-foundation-of-modern-neural-networks-5b8bf757db99

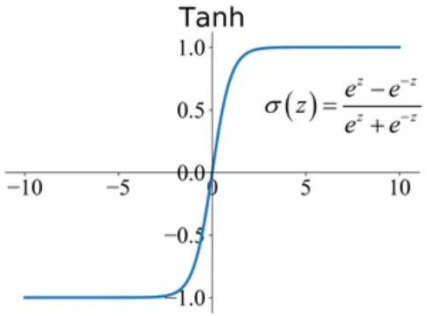
Activations

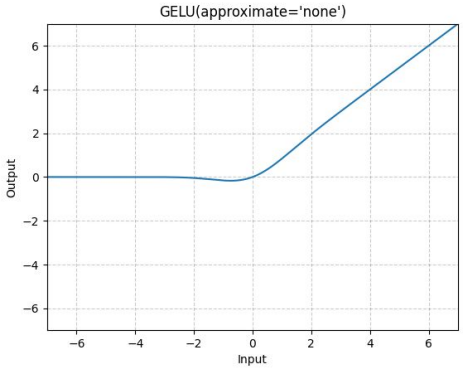
1. RELU
2. Sigmoid
3. Tanh
4. GELU

..







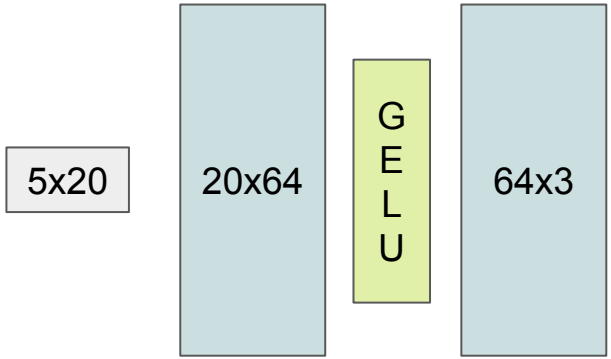


Y = WX + B
Can only capture linear relations

https://medium.com/@muhammadraflyindrawan/understanding-multilayer-perceptron-the-foundation-of-modern-neural-networks-5b8bf757db99

MLP layer - standalone

```
class MLP(nn.Module):
    def __init__(self, dim1, dim2, dim3):
        super().__init__()
        self.fc1 = nn.Linear(dim1, dim2)
        self.activation = nn.GELU()
        self.fc2 = nn.Linear(dim2, dim3)

    def forward(self, x):
        x = self.fc1(x)
        x = self.activation(x)
        x = self.fc2(x)
        return x

mlp = MLP(20, 64, 3)
x = torch.randn(5, 20)
output = mlp(x)
```



https://medium.com/@muhammadraflyindrawan/understanding-multilayer-perceptron-the-foundation-of-modern-neural-networks-5b8bf757db99

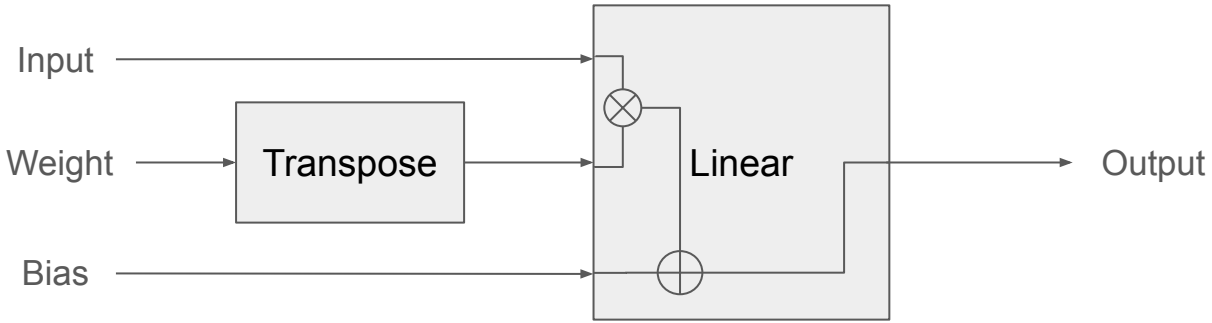
MLP layer in GPT2

```
from torch import nn
from transformers.pytorch_utils import Conv1D

class GptMLP(nn.Module):
    def __init__(self):
        super().__init__()
        self.c_fc = Conv1D(768 * 4, 768)
        self.gelu = torch.nn.GELU()
        self.c_proj = Conv1D(768, 768 * 4)
        self.dropout = torch.nn.Dropout(p=0.1, inplace=False)

    def forward(self, x):
        x = self.c_fc(x)
        x = self.gelu(x)
        x = self.c_proj(x)
        x = self.dropout(x)
        return x
```

```
>>> extract(mw, “h.0.mlp”).keys()
dict_keys(['c_fc.weight', 'c_fc.bias',
'c_proj.weight', 'c_proj.bias'])
>>> m = GptMLP()
>>> m.load_state_dict(extract(mw, “h.0.mlp”))
<All keys matched successfully>
```

Conv1D

- Convolution, but represents Cross Correlation in this context.
- Think of it as a simple transposed Linear layer

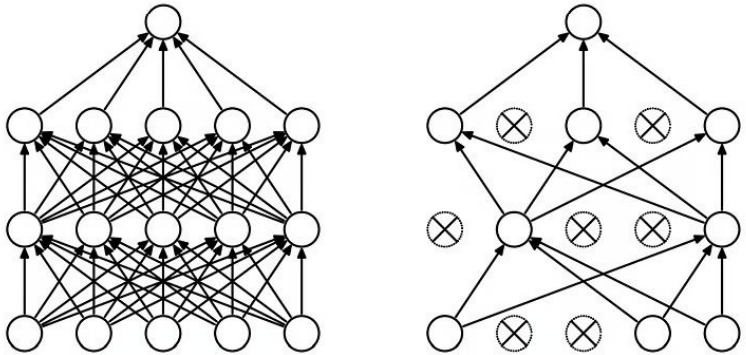


Dropout

- Regularization technique to prevent overfitting/improve generalization

```
m = nn.Dropout(p=0.2)
input = torch.randn(5, 2)
output = m(input)
```

- A certain percentage of values are ignored or “dropped out " during each forward and backward pass, making the model’s architecture dynamically different for each training batch.

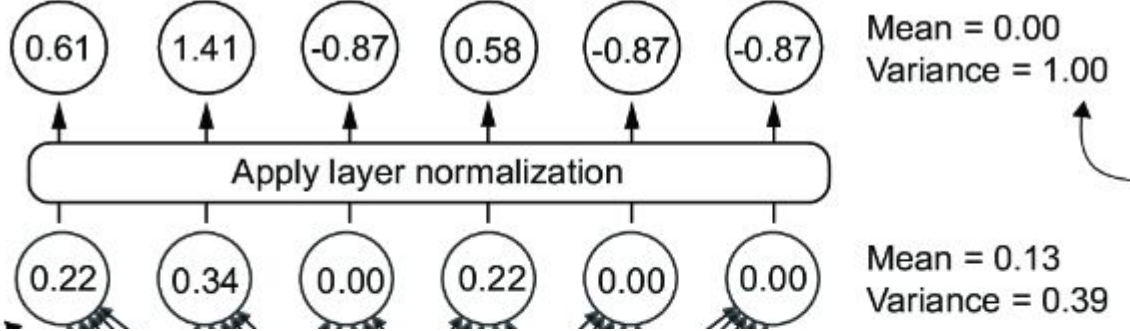


(a) Standard Neural Net (b) After applying dropout.

- Only used during training

https://arxiv.org/abs/1207.0580
https://www.datacamp.com/tutorial/dropout-regularization-using-pytorch-guide

LayerNorm



$$\text{LayerNorm}(x) = \gamma \cdot \frac{x - \mu}{\sqrt{\sigma^2}} + \beta$$

where μ is the mean of x, σ^2 is the variance of x, and γ and β are learnable parameters.

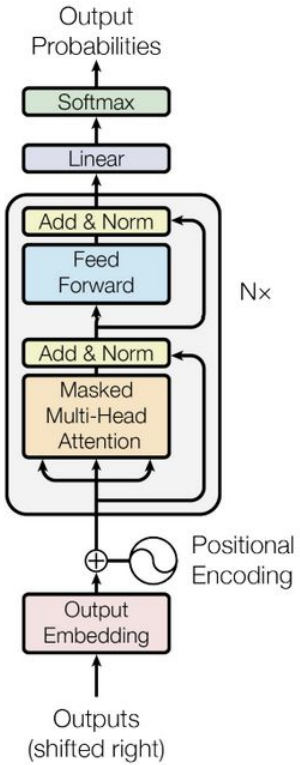
- Normalization technique that standardizes the inputs across the feature dimension of each individual sample
- Mean and variance are calculated across all dimensions of each input sample

```
layer_norm = torch.nn.LayerNorm(10)

x = torch.randn(3, 10)
out = layer_norm(x)
```

https://livebook.manning.com/wiki/categories/llm/layer+normalization

https://jaykmody.com/blog/gpt-from-scratch#layer-normalization

https://arxiv.org/pdf/1607.06450



Masked . Multi-head . **Attention**

The heart of the transformer!!!

Attention

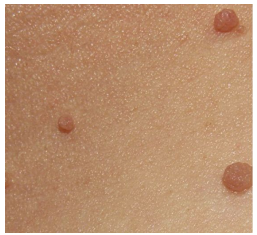
What attention needs to do?

American shrew **mole**

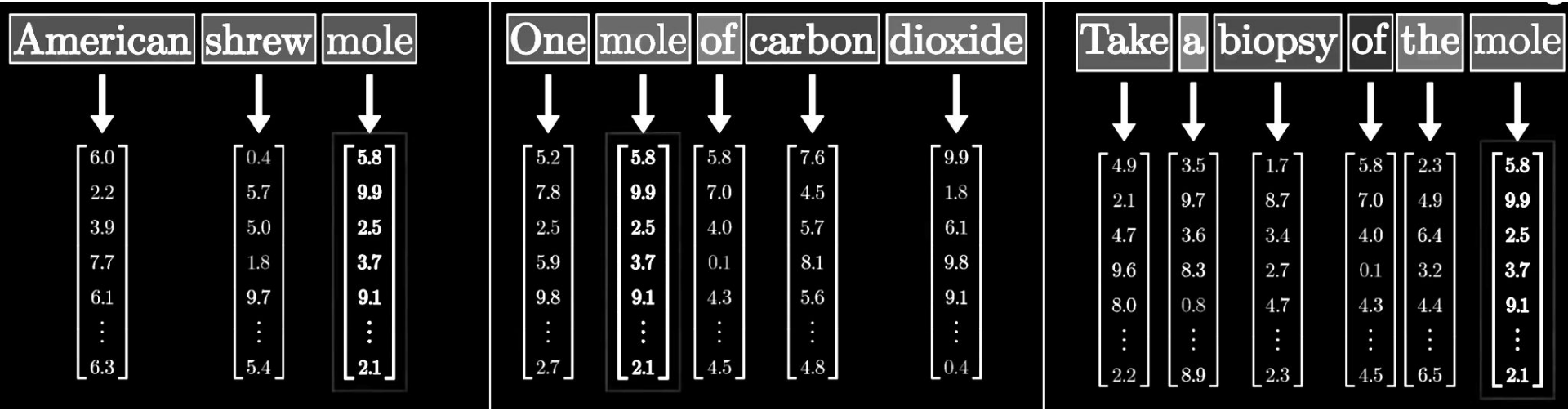
Differs based on context

One **mole** of carbon dioxide 6.02×10^{23}

Take a biopsy of the **mole**



Attention

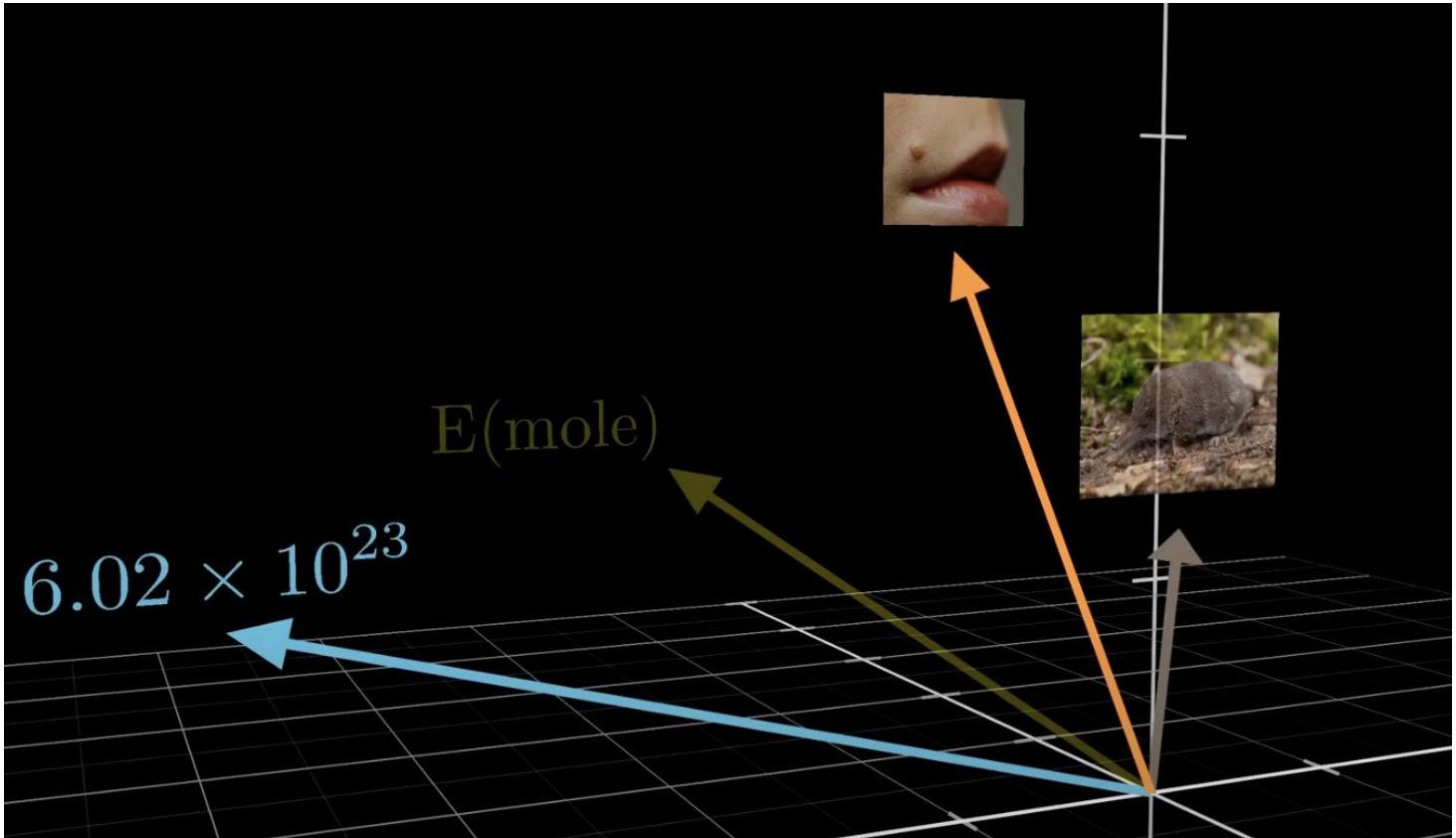


Token Embedding is a table lookup

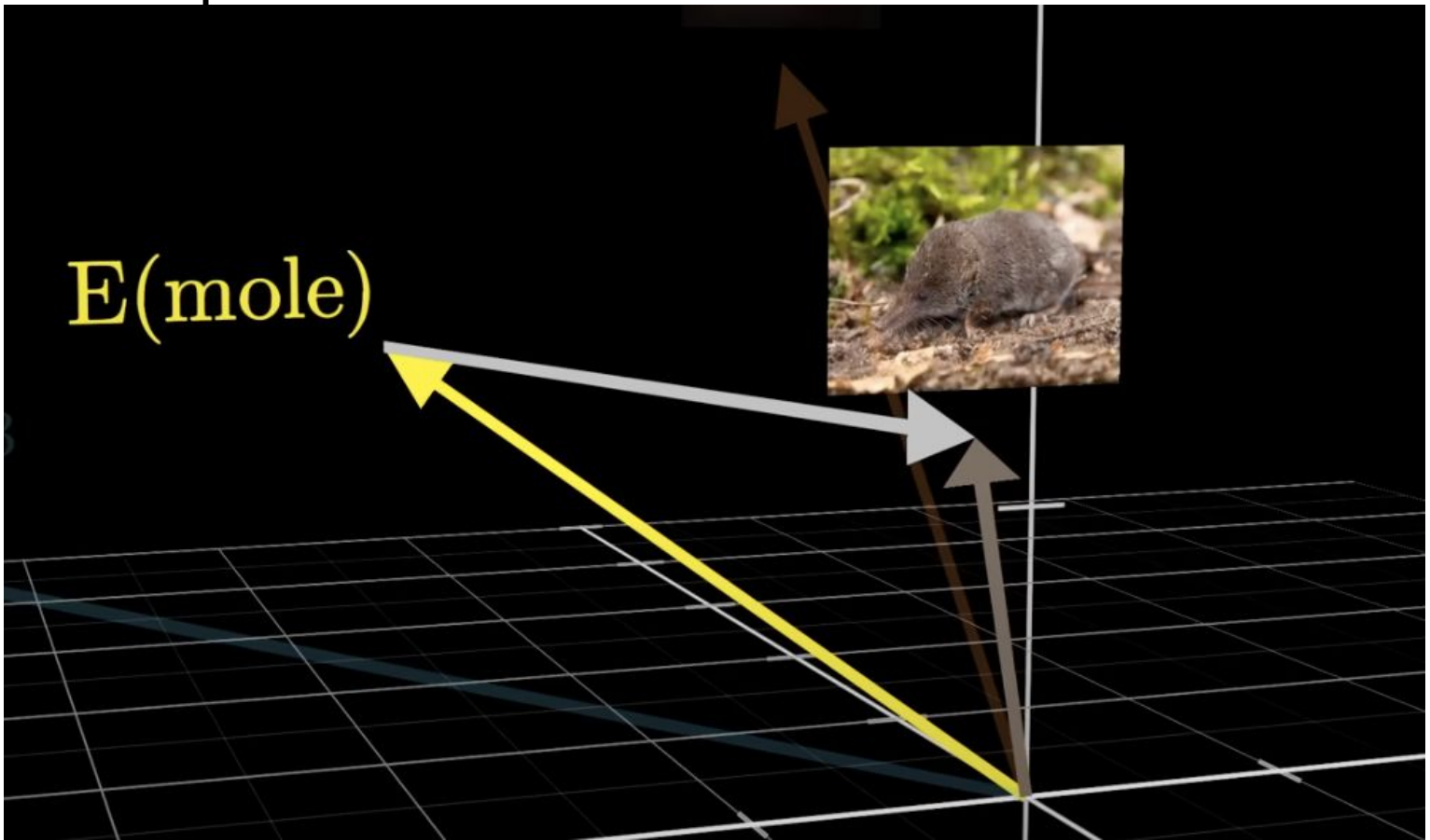
Attention helps pass the context information from other vectors

**Subsequent slides on attention from https://youtu.be/eMlx5fFNoYc?si=9lYxz2a8dyimrm78

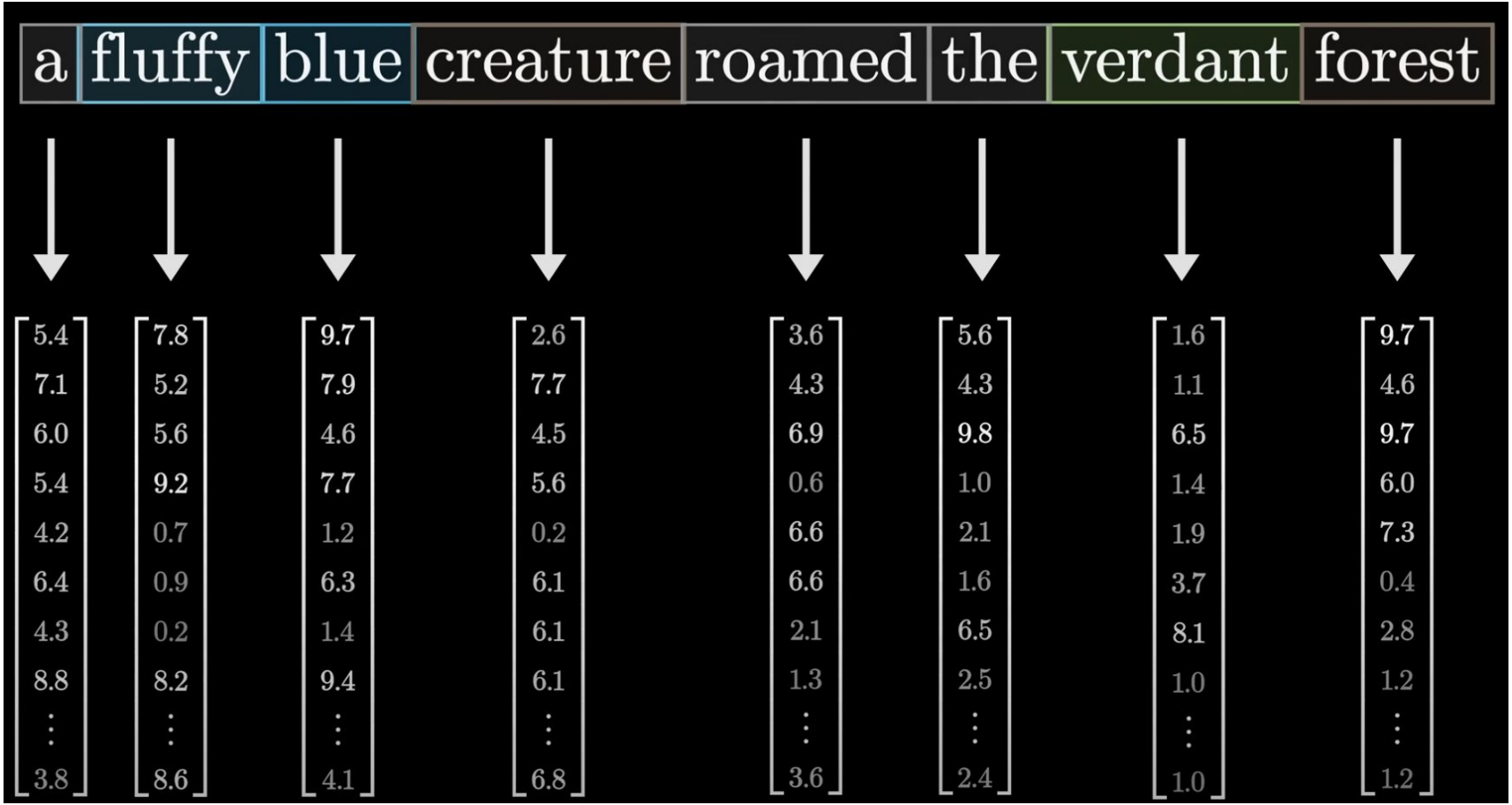
In vector space

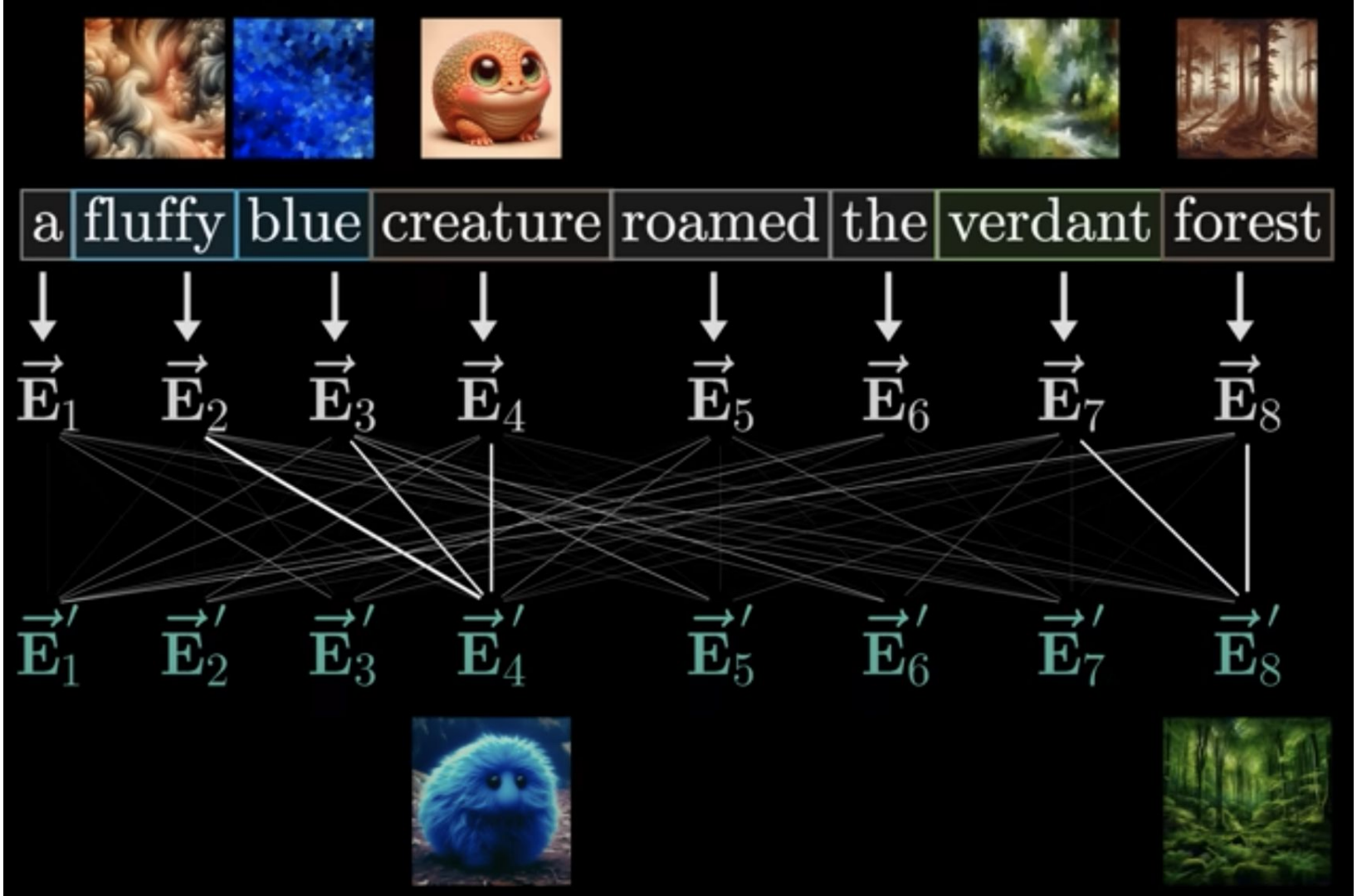


In vector space

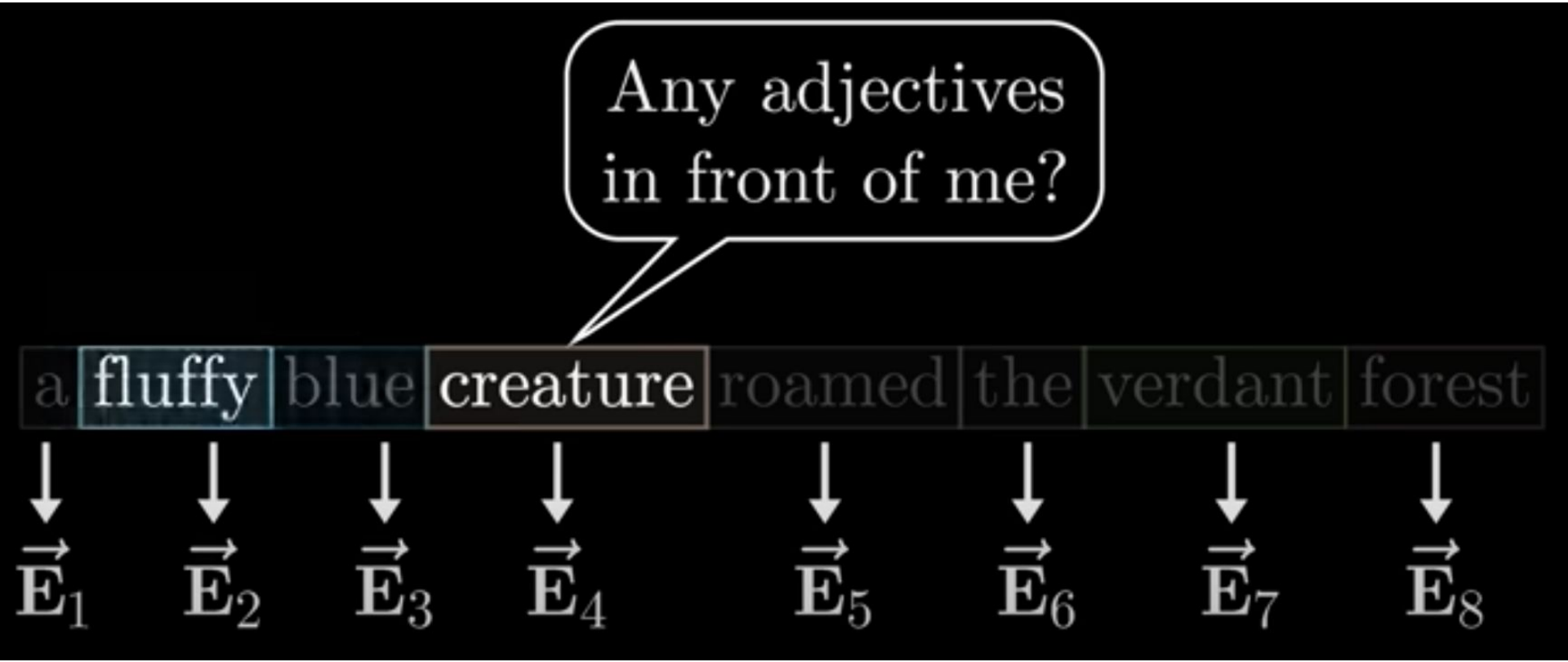


a fluffy blue creature roamed the verdant forest

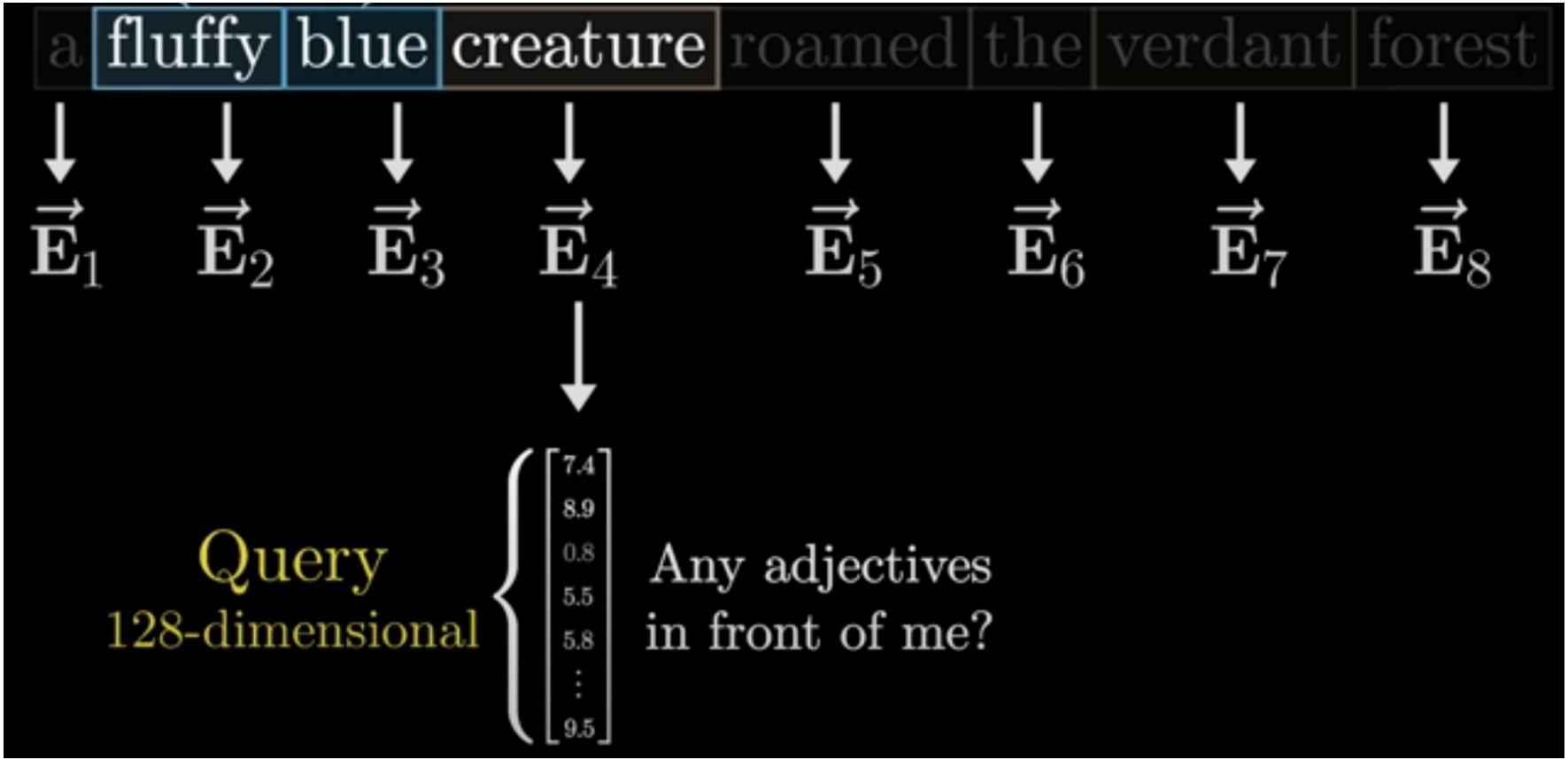

a
fluffy
blue
creature
roamed
the
verdant
forest
5.4 7.1 6.0 5.4 4.2 6.4 4.3 8.8 3.8
7.8 5.2 5.6 9.2 0.7 0.9 0.2 8.2 8.6
9.7 7.9 4.6 7.7 1.2 6.3 1.4 9.4 4.1
2.6 7.7 4.5 5.6 0.2 6.1 6.1 6.1 6.8
3.6 4.3 6.9 0.6 6.6 6.6 2.1 1.3 3.6
5.6 4.3 9.8 1.0 2.1 1.6 6.5 2.5 2.4
1.6 1.1 6.5 1.4 1.9 3.7 8.1 1.0 1.0
9.7 4.6 9.7 6.0 7.3 0.4 2.8 1.2 1.2


a fluffy blue creature roamed the verdant forest
$\vec{E}_1$ $\vec{E}_2$ $\vec{E}_3$ $\vec{E}_4$ $\vec{E}_5$ $\vec{E}_6$ $\vec{E}_7$ $\vec{E}_8$
$\vec{E}_1'$ $\vec{E}_2'$ $\vec{E}_3'$ $\vec{E}_4'$ $\vec{E}_5'$ $\vec{E}_6'$ $\vec{E}_7'$ $\vec{E}_8'$

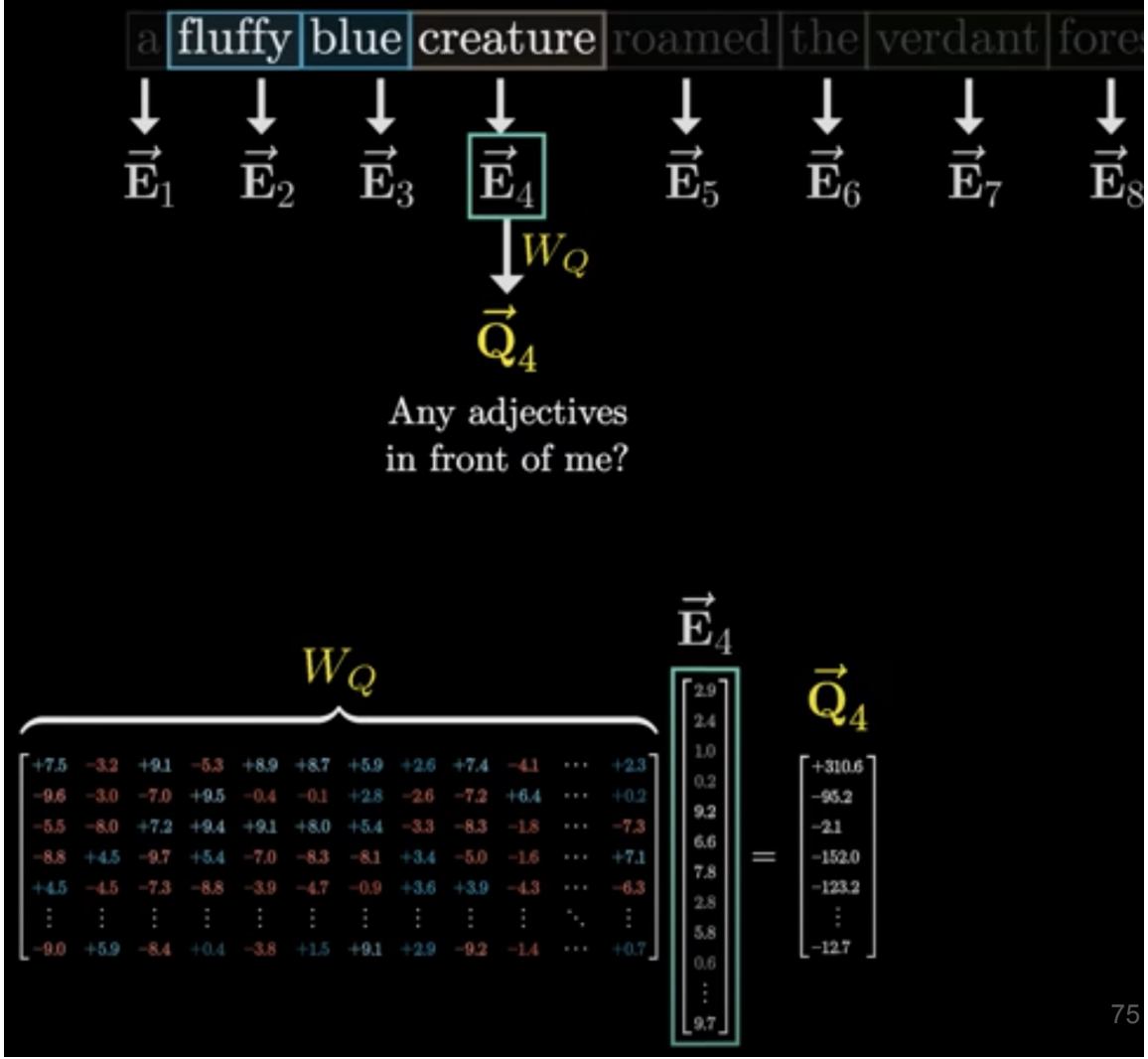
Query


Any adjectives in front of me?
a fluffy blue creature roamed the verdant forest
$\vec{E}_1$ $\vec{E}_2$ $\vec{E}_3$ $\vec{E}_4$ $\vec{E}_5$ $\vec{E}_6$ $\vec{E}_7$ $\vec{E}_8$

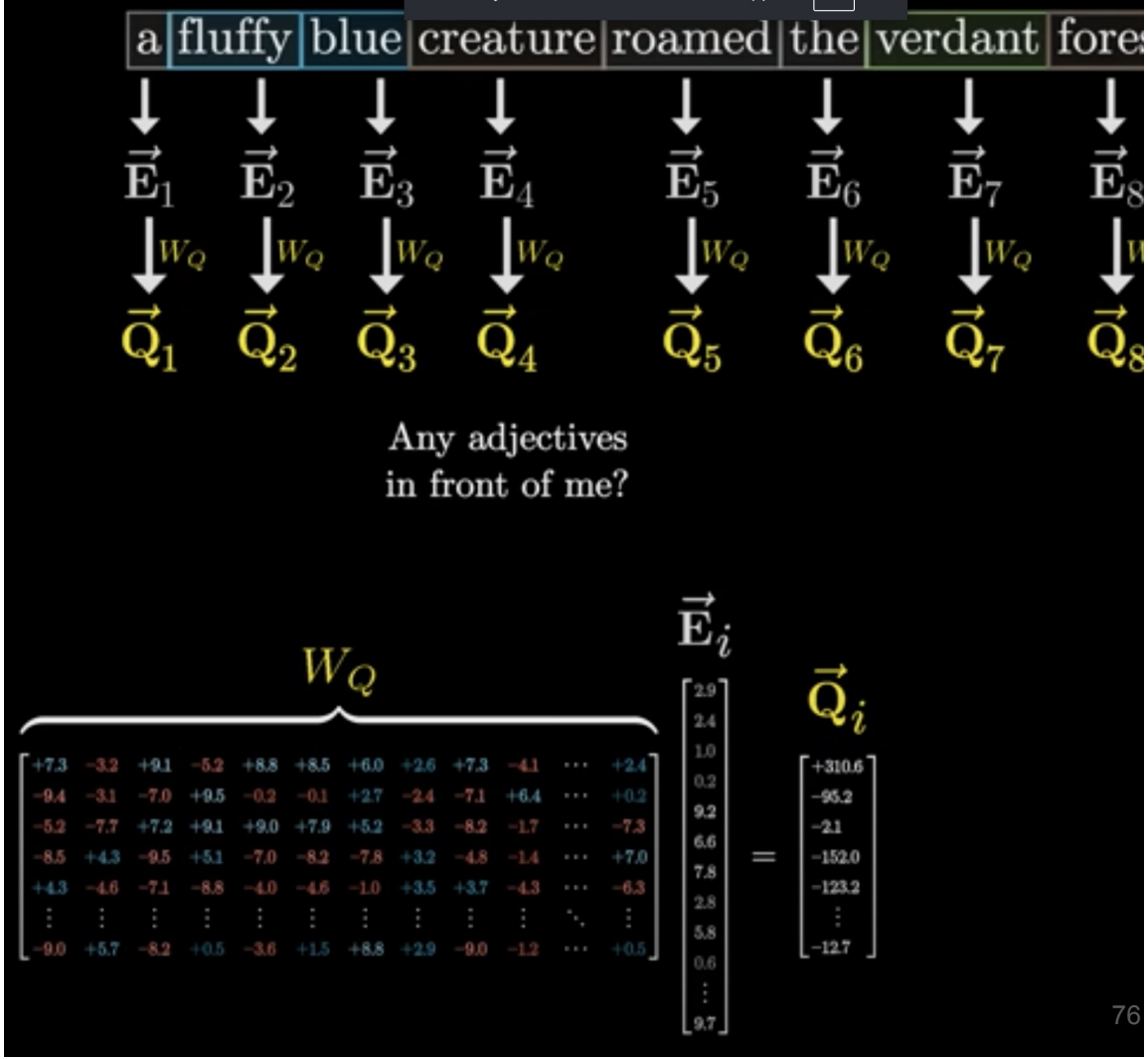
Query


a fluffy blue creature roamed the verdant forest
$\vec{E}_1$ $\vec{E}_2$ $\vec{E}_3$ $\vec{E}_4$ $\vec{E}_5$ $\vec{E}_6$ $\vec{E}_7$ $\vec{E}_8$
Query
128-dimensional
7.4
8.9
0.8
5.5
5.8
9.5
Any adjectives in front of me?

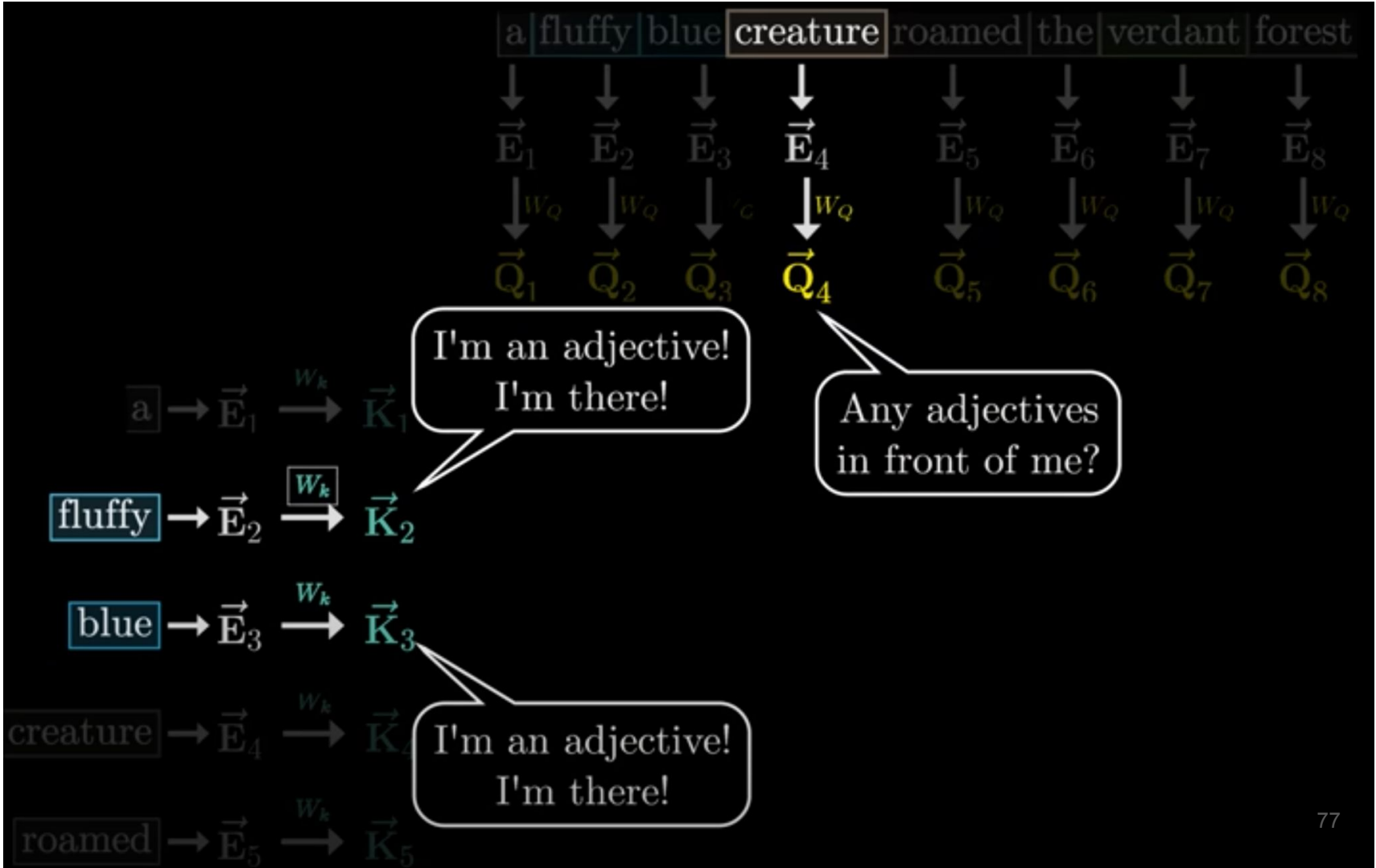
How is query calculated?

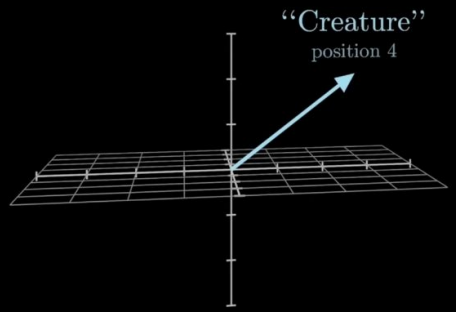


How is query calculated?

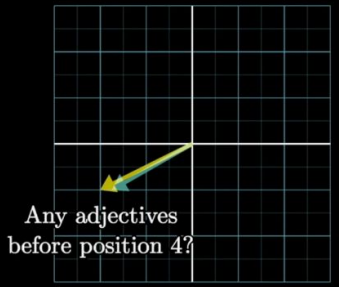


Key


a fluffy blue creature roamed the verdant forest
$\vec{E}_1$ $\vec{E}_2$ $\vec{E}_3$ $\vec{E}_4$ $\vec{E}_5$ $\vec{E}_6$ $\vec{E}_7$ $\vec{E}_8$
W_Q
$\vec{Q}_1$ $\vec{Q}_2$ $\vec{Q}_3$ $\vec{Q}_4$ $\vec{Q}_5$ $\vec{Q}_6$ $\vec{Q}_7$ $\vec{Q}_8$
I'm an adjective! I'm there!
Any adjectives in front of me?
a → $\vec{E}_1$ W_k $\vec{K}_1$
fluffy → $\vec{E}_2$ W_k $\vec{K}_2$
blue → $\vec{E}_3$ W_k $\vec{K}_3$
creature → $\vec{E}_4$ W_k $\vec{K}_4$
I'm an adjective! I'm there!
roamed → $\vec{E}_5$ W_k $\vec{K}_5$



W_Q



How well does key and query match?

a → $\vec{E}_1$ → W_k → $\vec{K}_1$

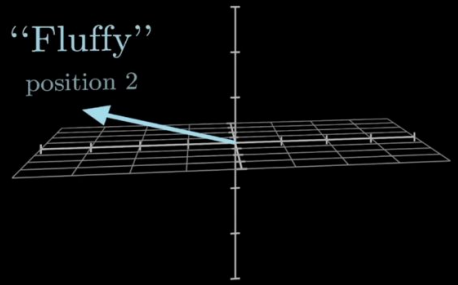
fluffy → $\vec{E}_2$ → W_k → $\vec{K}_2$

blue → $\vec{E}_3$ → W_k → $\vec{K}_3$

creature → $\vec{E}_4$ → W_k → $\vec{K}_4$

roamed → $\vec{E}_5$ → W_k → $\vec{K}_5$

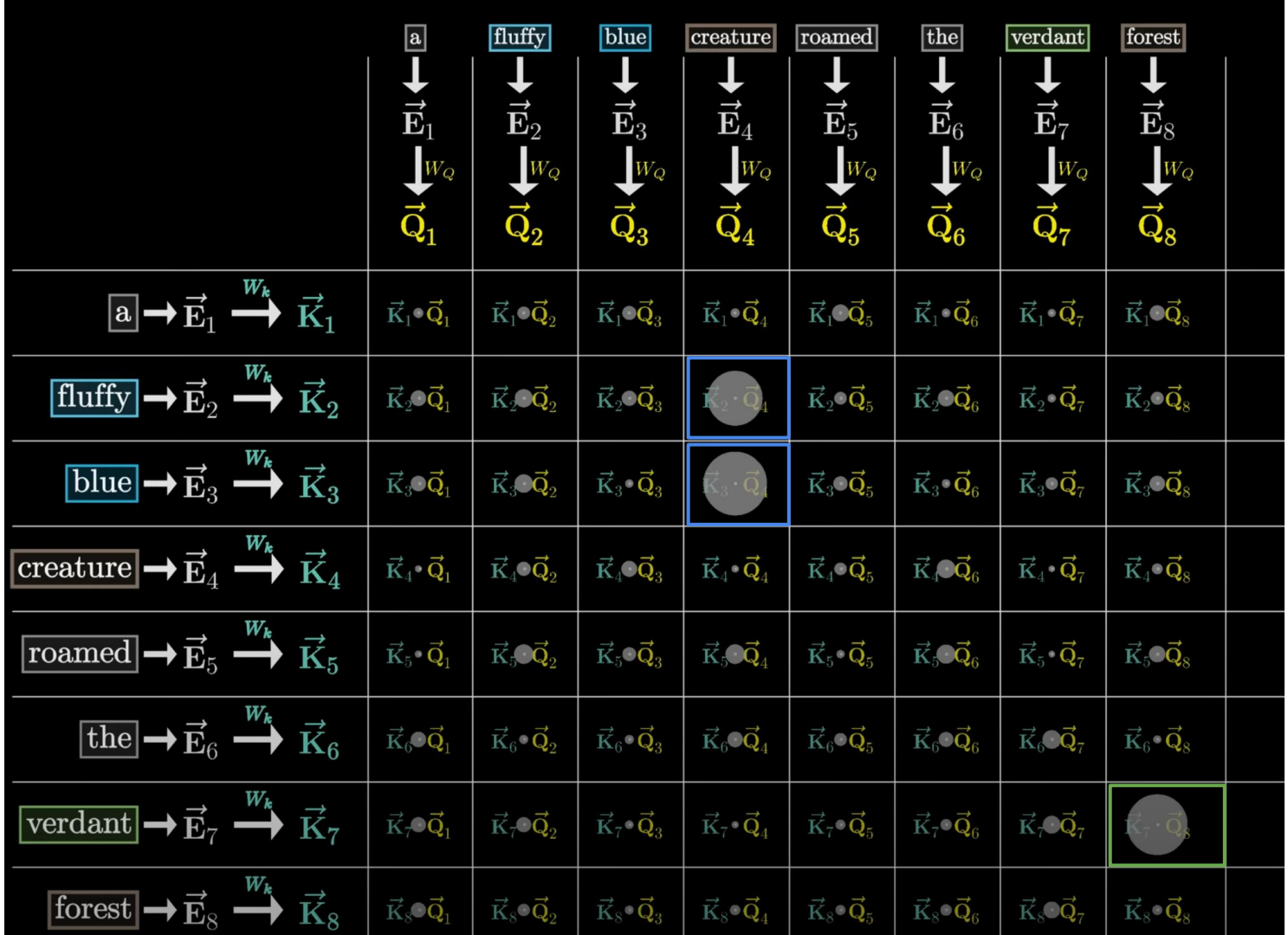
Embedding space
12,288-dimensional



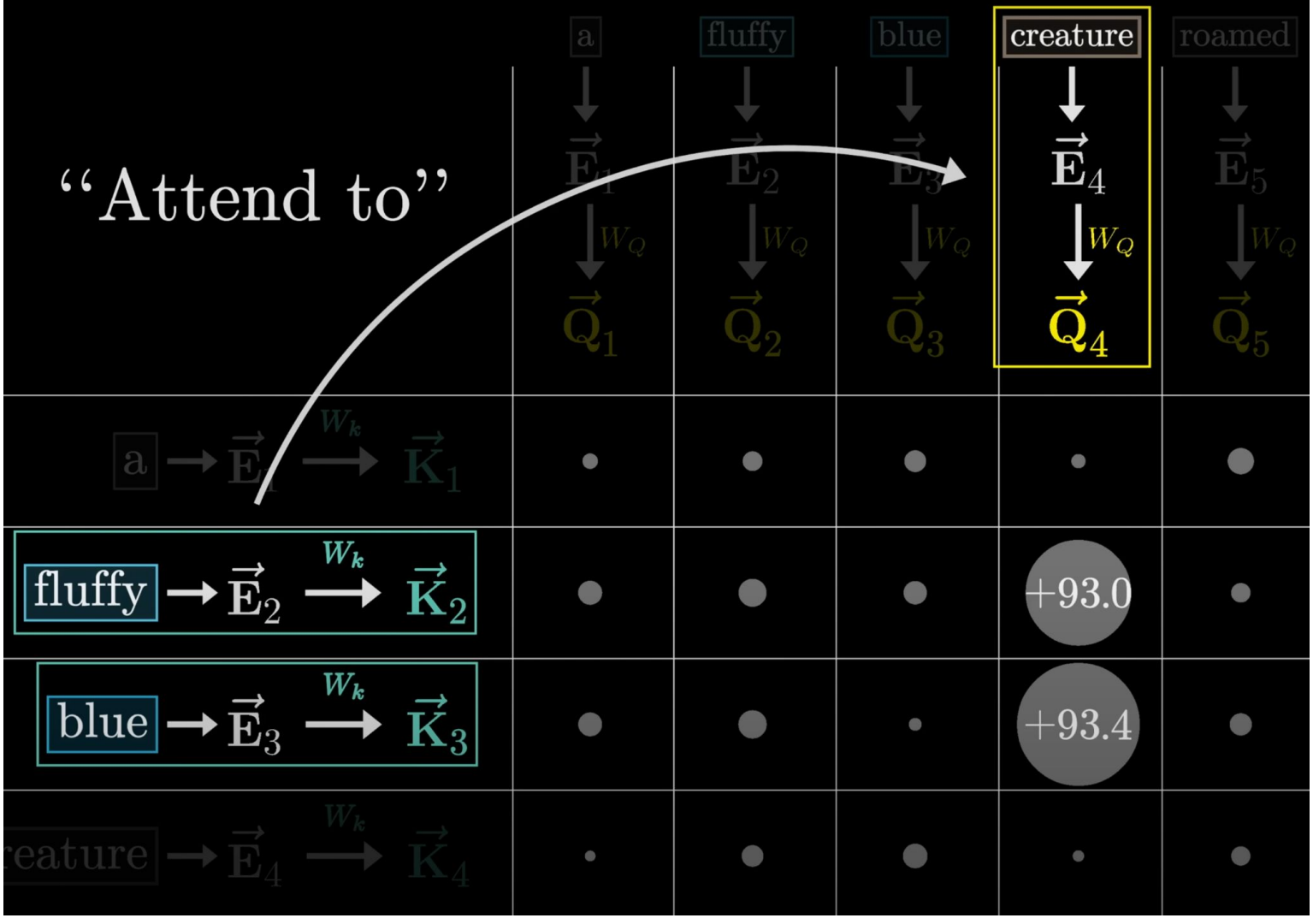
W_K

Query/Key space
128-dimensional

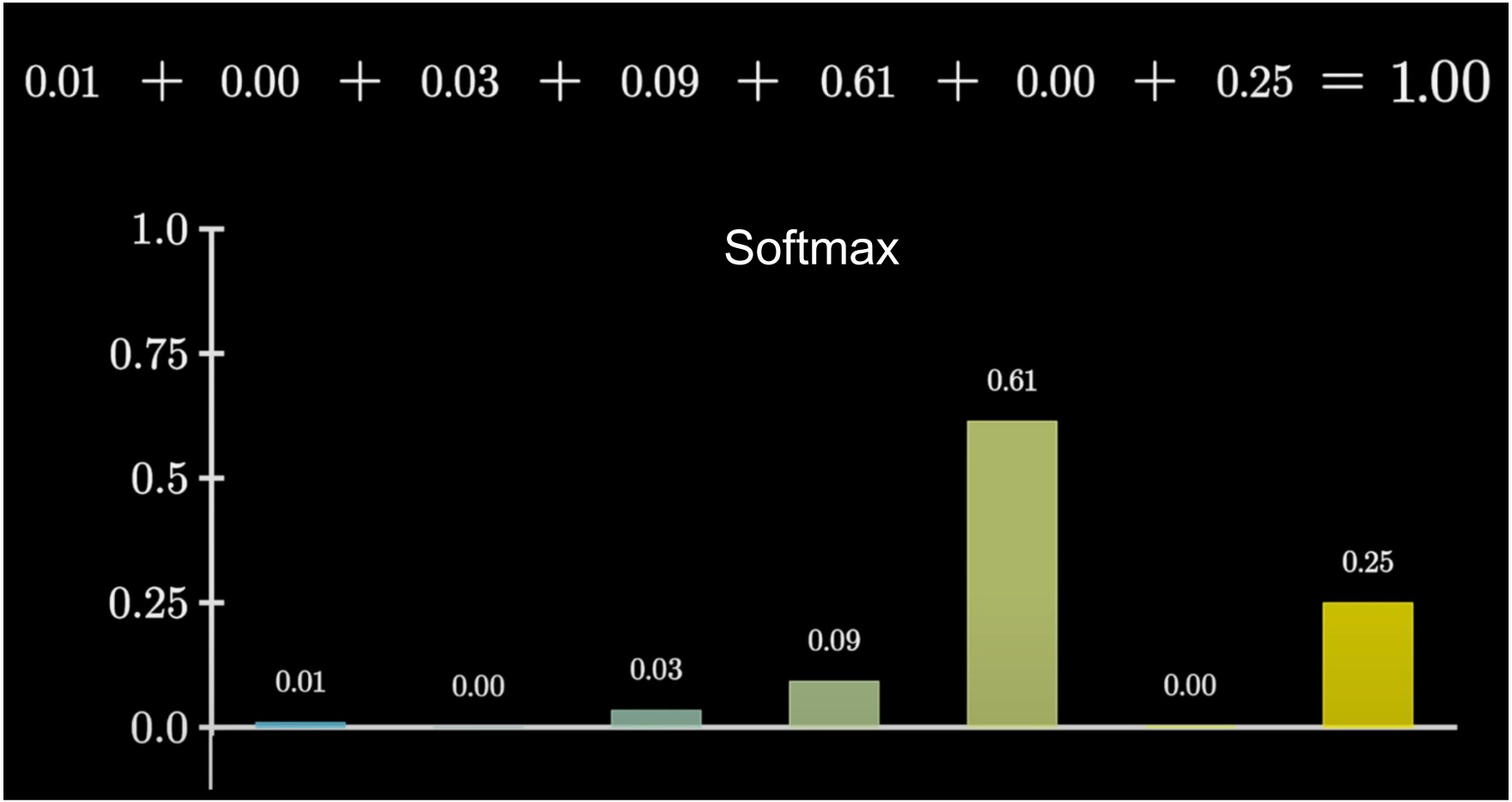

a
fluffy
blue
creature
roamed
the
verdant
forest
$\vec{E}_1$ $\vec{E}_2$ $\vec{E}_3$ $\vec{E}_4$ $\vec{E}_5$ $\vec{E}_6$ $\vec{E}_7$ $\vec{E}_8$
W_Q
$\vec{Q}_1$ $\vec{Q}_2$ $\vec{Q}_3$ $\vec{Q}_4$ $\vec{Q}_5$ $\vec{Q}_6$ $\vec{Q}_7$ $\vec{Q}_8$
a → $\vec{E}_1$ W_k → $\vec{K}_1$
fluffy → $\vec{E}_2$ W_k → $\vec{K}_2$
blue → $\vec{E}_3$ W_k → $\vec{K}_3$
creature → $\vec{E}_4$ W_k → $\vec{K}_4$
roamed → $\vec{E}_5$ W_k → $\vec{K}_5$
the → $\vec{E}_6$ W_k → $\vec{K}_6$
verdant → $\vec{E}_7$ W_k → $\vec{K}_7$
forest → $\vec{E}_8$ W_k → $\vec{K}_8$
$\vec{K}_1 \cdot \vec{Q}_1$ $\vec{K}_1 \cdot \vec{Q}_2$ $\vec{K}_1 \cdot \vec{Q}_3$ $\vec{K}_1 \cdot \vec{Q}_4$ $\vec{K}_1 \cdot \vec{Q}_5$ $\vec{K}_1 \cdot \vec{Q}_6$ $\vec{K}_1 \cdot \vec{Q}_7$ $\vec{K}_1 \cdot \vec{Q}_8$
$\vec{K}_2 \cdot \vec{Q}_1$ $\vec{K}_2 \cdot \vec{Q}_2$ $\vec{K}_2 \cdot \vec{Q}_3$ $\vec{K}_2 \cdot \vec{Q}_4$ $\vec{K}_2 \cdot \vec{Q}_5$ $\vec{K}_2 \cdot \vec{Q}_6$ $\vec{K}_2 \cdot \vec{Q}_7$ $\vec{K}_2 \cdot \vec{Q}_8$
$\vec{K}_3 \cdot \vec{Q}_1$ $\vec{K}_3 \cdot \vec{Q}_2$ $\vec{K}_3 \cdot \vec{Q}_3$ $\vec{K}_3 \cdot \vec{Q}_4$ $\vec{K}_3 \cdot \vec{Q}_5$ $\vec{K}_3 \cdot \vec{Q}_6$ $\vec{K}_3 \cdot \vec{Q}_7$ $\vec{K}_3 \cdot \vec{Q}_8$
$\vec{K}_4 \cdot \vec{Q}_1$ $\vec{K}_4 \cdot \vec{Q}_2$ $\vec{K}_4 \cdot \vec{Q}_3$ $\vec{K}_4 \cdot \vec{Q}_4$ $\vec{K}_4 \cdot \vec{Q}_5$ $\vec{K}_4 \cdot \vec{Q}_6$ $\vec{K}_4 \cdot \vec{Q}_7$ $\vec{K}_4 \cdot \vec{Q}_8$
$\vec{K}_5 \cdot \vec{Q}_1$ $\vec{K}_5 \cdot \vec{Q}_2$ $\vec{K}_5 \cdot \vec{Q}_3$ $\vec{K}_5 \cdot \vec{Q}_4$ $\vec{K}_5 \cdot \vec{Q}_5$ $\vec{K}_5 \cdot \vec{Q}_6$ $\vec{K}_5 \cdot \vec{Q}_7$ $\vec{K}_5 \cdot \vec{Q}_8$
$\vec{K}_6 \cdot \vec{Q}_1$ $\vec{K}_6 \cdot \vec{Q}_2$ $\vec{K}_6 \cdot \vec{Q}_3$ $\vec{K}_6 \cdot \vec{Q}_4$ $\vec{K}_6 \cdot \vec{Q}_5$ $\vec{K}_6 \cdot \vec{Q}_6$ $\vec{K}_6 \cdot \vec{Q}_7$ $\vec{K}_6 \cdot \vec{Q}_8$
$\vec{K}_7 \cdot \vec{Q}_1$ $\vec{K}_7 \cdot \vec{Q}_2$ $\vec{K}_7 \cdot \vec{Q}_3$ $\vec{K}_7 \cdot \vec{Q}_4$ $\vec{K}_7 \cdot \vec{Q}_5$ $\vec{K}_7 \cdot \vec{Q}_6$ $\vec{K}_7 \cdot \vec{Q}_7$ $\vec{K}_7 \cdot \vec{Q}_8$
$\vec{K}_8 \cdot \vec{Q}_1$ $\vec{K}_8 \cdot \vec{Q}_2$ $\vec{K}_8 \cdot \vec{Q}_3$ $\vec{K}_8 \cdot \vec{Q}_4$ $\vec{K}_8 \cdot \vec{Q}_5$ $\vec{K}_8 \cdot \vec{Q}_6$ $\vec{K}_8 \cdot \vec{Q}_7$ $\vec{K}_8 \cdot \vec{Q}_8$


a
fluffy
blue
creature
roamed
the
verdant
forest
$\vec{E}_1$ $\vec{E}_2$ $\vec{E}_3$ $\vec{E}_4$ $\vec{E}_5$ $\vec{E}_6$ $\vec{E}_7$ $\vec{E}_8$
W_Q
$\vec{Q}_1$ $\vec{Q}_2$ $\vec{Q}_3$ $\vec{Q}_4$ $\vec{Q}_5$ $\vec{Q}_6$ $\vec{Q}_7$ $\vec{Q}_8$
a → $\vec{E}_1$ W_k → $\vec{K}_1$
fluffy → $\vec{E}_2$ W_k → $\vec{K}_2$
blue → $\vec{E}_3$ W_k → $\vec{K}_3$
creature → $\vec{E}_4$ W_k → $\vec{K}_4$
roamed → $\vec{E}_5$ W_k → $\vec{K}_5$
the → $\vec{E}_6$ W_k → $\vec{K}_6$
verdant → $\vec{E}_7$ W_k → $\vec{K}_7$
forest → $\vec{E}_8$ W_k → $\vec{K}_8$
$\vec{K}_i \cdot \vec{Q}_j$

	a	fluffy	blue	creature	roamed	the	verdant	forest
	$\vec{E}_1 \downarrow W_Q \vec{Q}_1$	$\vec{E}_2 \downarrow W_Q \vec{Q}_2$	$\vec{E}_3 \downarrow W_Q \vec{Q}_3$	$\vec{E}_4 \downarrow W_Q \vec{Q}_4$	$\vec{E}_5 \downarrow W_Q \vec{Q}_5$	$\vec{E}_6 \downarrow W_Q \vec{Q}_6$	$\vec{E}_7 \downarrow W_Q \vec{Q}_7$	$\vec{E}_8 \downarrow W_Q \vec{Q}_8$
a $\rightarrow \vec{E}_1 \xrightarrow{W_k} \vec{K}_1$	+0.7	−83.7	−24.7	−27.8	−5.2	−89.3	−45.2	−36.1
fluffy $\rightarrow \vec{E}_2 \xrightarrow{W_k} \vec{K}_2$	−73.4	+2.9	−5.4	+93.0	−48.2	−87.3	−49.7	+7.8
blue $\rightarrow \vec{E}_3 \xrightarrow{W_k} \vec{K}_3$	−53.4	−5.7	+1.8	+93.4	−55.6	−56.0	−26.1	−62.1
creature $\rightarrow \vec{E}_4 \xrightarrow{W_k} \vec{K}_4$	−21.5	−29.7	−56.1	+4.9	−32.4	−92.3	−9.5	−28.1
roamed $\rightarrow \vec{E}_5 \xrightarrow{W_k} \vec{K}_5$	−20.1	−40.9	−87.8	−55.4	+0.6	−64.7	−96.7	−18.9
the $\rightarrow \vec{E}_6 \xrightarrow{W_k} \vec{K}_6$	−87.9	−33.3	−22.6	−31.4	+5.5	+0.6	−4.6	−96.8
verdant $\rightarrow \vec{E}_7 \xrightarrow{W_k} \vec{K}_7$	−41.2	−55.5	−42.3	−59.8	−79.0	−97.9	+3.7	+93.8
forest $\rightarrow \vec{E}_8 \xrightarrow{W_k} \vec{K}_8$	−58.9	−75.5	−91.1	−90.6	−75.6	−89.0	−70.8	+4.7


a
fluffy
blue
creature
roamed
"Attend to"
$\vec{E}_1$
$\vec{E}_2$
$\vec{E}_3$
$\vec{E}_4$
$\vec{E}_5$
W_Q
$\vec{Q}_1$
$\vec{Q}_2$
$\vec{Q}_3$
$\vec{Q}_4$
$\vec{Q}_5$
a → $\vec{E}_1$ W_k → $\vec{K}_1$
fluffy → $\vec{E}_2$ W_k → $\vec{K}_2$
blue → $\vec{E}_3$ W_k → $\vec{K}_3$
reature → $\vec{E}_4$ W_k → $\vec{K}_4$
+93.0
+93.4

$$0.01 + 0.00 + 0.03 + 0.09 + 0.61 + 0.00 + 0.25 = 1.00$$

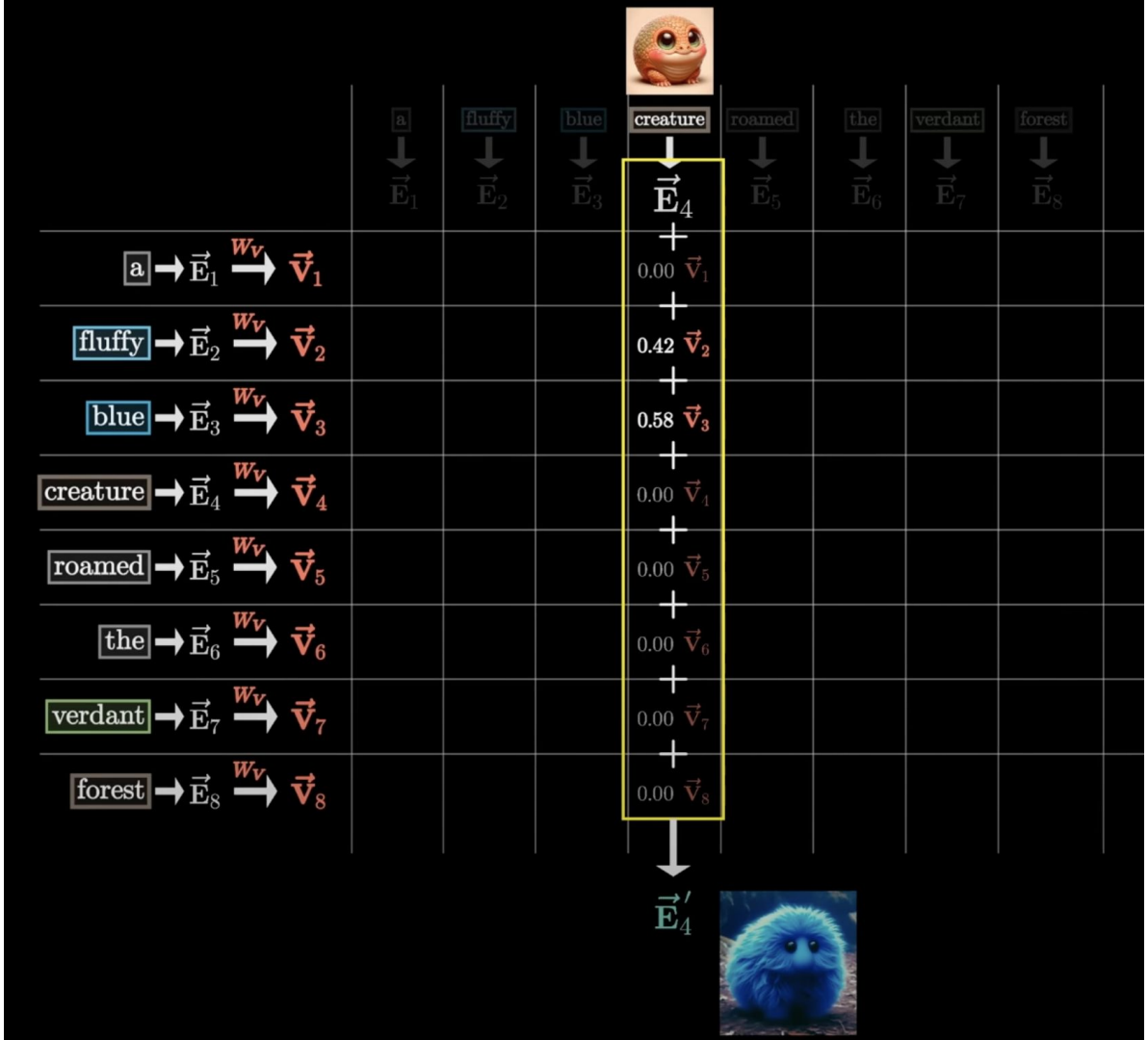


Attention Pattern

Which words are relevant to which other words

	a	fluffy	blue	creature	roamed	the	verdant	forest
	$\vec{E}_1 \xrightarrow{W_Q} \vec{Q}_1$	$\vec{E}_2 \xrightarrow{W_Q} \vec{Q}_2$	$\vec{E}_3 \xrightarrow{W_Q} \vec{Q}_3$	$\vec{E}_4 \xrightarrow{W_Q} \vec{Q}_4$	$\vec{E}_5 \xrightarrow{W_Q} \vec{Q}_5$	$\vec{E}_6 \xrightarrow{W_Q} \vec{Q}_6$	$\vec{E}_7 \xrightarrow{W_Q} \vec{Q}_7$	$\vec{E}_8 \xrightarrow{W_Q} \vec{Q}_8$
a $\vec{E}_1 \xrightarrow{W_K} \vec{K}_1$	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
fluffy $\vec{E}_2 \xrightarrow{W_K} \vec{K}_2$	0.00	1.00	0.00	0.42	0.00	0.00	0.00	0.00
blue $\vec{E}_3 \xrightarrow{W_K} \vec{K}_3$	0.00	0.00	1.00	0.58	0.00	0.00	0.00	0.00
creature $\vec{E}_4 \xrightarrow{W_K} \vec{K}_4$	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
roamed $\vec{E}_5 \xrightarrow{W_K} \vec{K}_5$	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00
the $\vec{E}_6 \xrightarrow{W_K} \vec{K}_6$	0.00	0.00	0.00	0.00	0.99	1.00	0.00	0.00
verdant $\vec{E}_7 \xrightarrow{W_K} \vec{K}_7$	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00
forest $\vec{E}_8 \xrightarrow{W_K} \vec{K}_8$	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Values

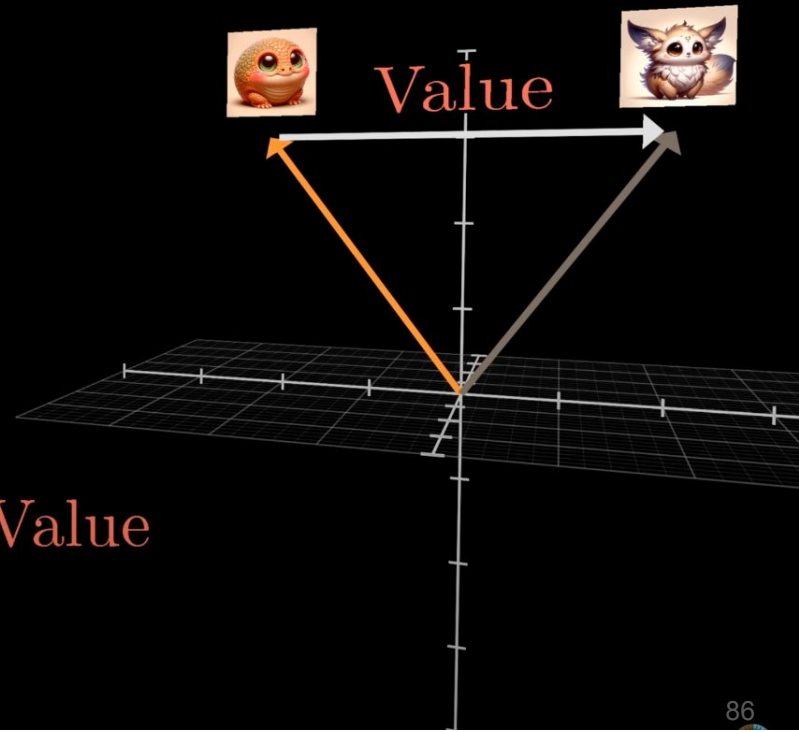

a
fluffy
blue
creature
roamed
the
verdant
forest
$\vec{E}_1$ $\vec{E}_2$ $\vec{E}_3$ $\vec{E}_4$ $\vec{E}_5$ $\vec{E}_6$ $\vec{E}_7$ $\vec{E}_8$
a → $\vec{E}_1$ W_V → $\vec{V}_1$
fluffy → $\vec{E}_2$ W_V → $\vec{V}_2$
blue → $\vec{E}_3$ W_V → $\vec{V}_3$
creature → $\vec{E}_4$ W_V → $\vec{V}_4$
roamed → $\vec{E}_5$ W_V → $\vec{V}_5$
the → $\vec{E}_6$ W_V → $\vec{V}_6$
verdant → $\vec{E}_7$ W_V → $\vec{V}_7$
forest → $\vec{E}_8$ W_V → $\vec{V}_8$
$\vec{E}_4$ +
0.00 $\vec{V}_1$ +
0.42 $\vec{V}_2$ +
0.58 $\vec{V}_3$ +
0.00 $\vec{V}_4$ +
0.00 $\vec{V}_5$ +
0.00 $\vec{V}_6$ +
0.00 $\vec{V}_7$ +
0.00 $\vec{V}_8$
$\vec{E}_4'$

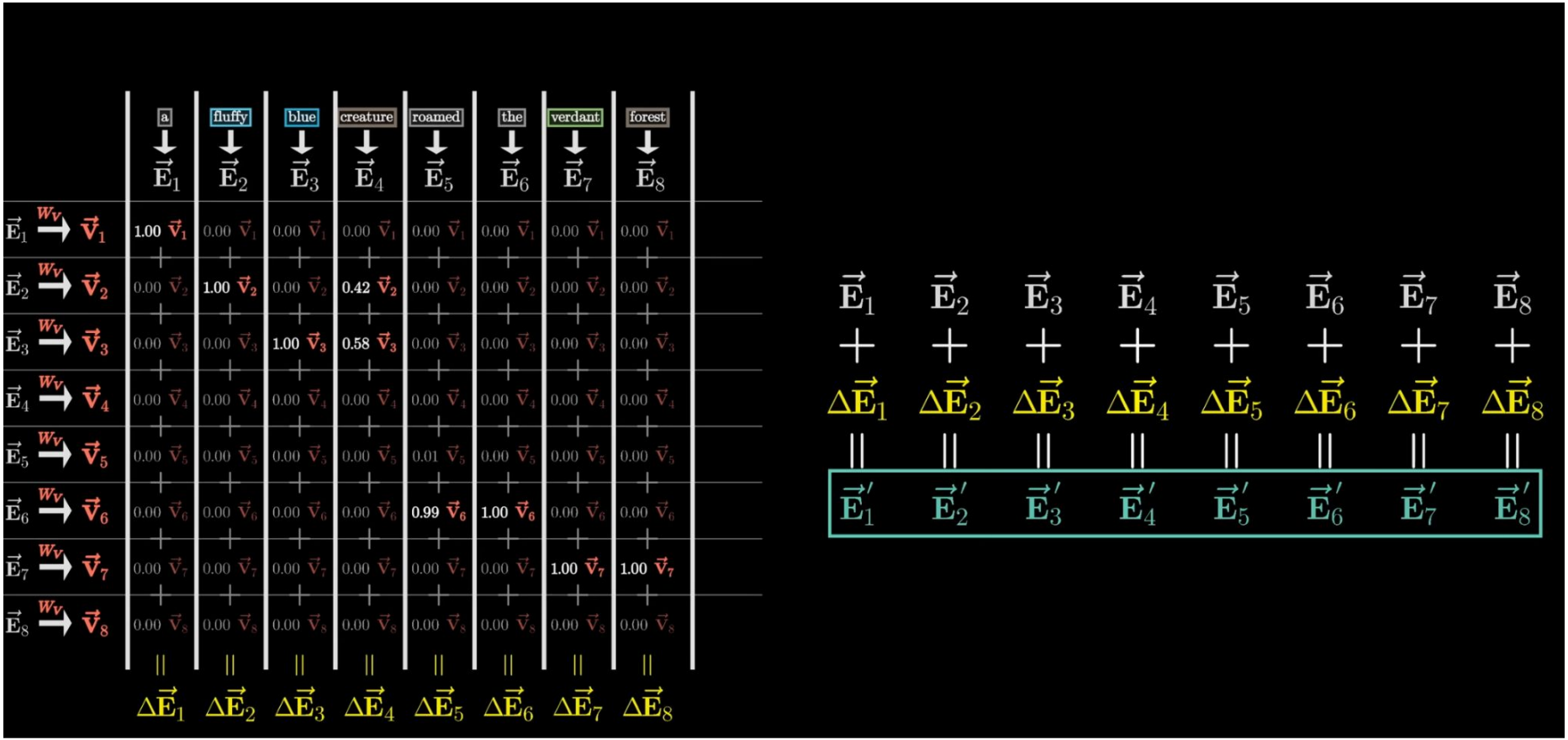
fluffy creature

$$W_V \begin{bmatrix} +9.2 \\ -2.3 \\ +5.8 \\ +0.6 \\ +1.3 \\ +8.4 \\ \vdots \\ -8.2 \end{bmatrix} = \begin{bmatrix} -52.4 \\ +89.3 \\ -80.2 \\ -17.8 \\ +7.3 \\ +223.8 \\ \vdots \\ -41.0 \end{bmatrix}$$

$$W_V = \begin{bmatrix} -3.6 & -1.7 & -8.6 & +3.8 & +1.3 & -4.6 & \cdots & -8.0 \\ +1.5 & +8.5 & -3.6 & +3.3 & -7.3 & +4.3 & \cdots & -6.3 \\ +1.7 & -9.5 & +6.5 & -9.8 & +3.5 & -4.6 & \cdots & +9.2 \\ -5.0 & +1.5 & +1.8 & +1.4 & -5.5 & +9.0 & \cdots & +6.9 \\ +3.9 & -4.0 & +6.2 & -2.0 & +7.5 & +1.6 & \cdots & +3.8 \\ +4.5 & +0.0 & +9.0 & +2.9 & -1.5 & +2.1 & \cdots & -3.9 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ +1.5 & +3.0 & +3.0 & -1.4 & +7.9 & -2.6 & \cdots & +7.8 \end{bmatrix}$$

Value




a
fluffy
blue
creature
roamed
the
verdant
forest
$\vec{E}_1$ $\vec{E}_2$ $\vec{E}_3$ $\vec{E}_4$ $\vec{E}_5$ $\vec{E}_6$ $\vec{E}_7$ $\vec{E}_8$
W_V
$\vec{E}_1 \to \vec{V}_1$: 1.00 $\vec{V}_1$ 0.00 $\vec{V}_1$ 0.00 $\vec{V}_1$ 0.00 $\vec{V}_1$ 0.00 $\vec{V}_1$ 0.00 $\vec{V}_1$ 0.00 $\vec{V}_1$ 0.00 $\vec{V}_1$
$\vec{E}_2 \to \vec{V}_2$: 0.00 $\vec{V}_2$ 1.00 $\vec{V}_2$ 0.00 $\vec{V}_2$ 0.42 $\vec{V}_2$ 0.00 $\vec{V}_2$ 0.00 $\vec{V}_2$ 0.00 $\vec{V}_2$ 0.00 $\vec{V}_2$
$\vec{E}_3 \to \vec{V}_3$: 0.00 $\vec{V}_3$ 0.00 $\vec{V}_3$ 1.00 $\vec{V}_3$ 0.58 $\vec{V}_3$ 0.00 $\vec{V}_3$ 0.00 $\vec{V}_3$ 0.00 $\vec{V}_3$ 0.00 $\vec{V}_3$
$\vec{E}_4 \to \vec{V}_4$: 0.00 $\vec{V}_4$ 0.00 $\vec{V}_4$ 0.00 $\vec{V}_4$ 0.00 $\vec{V}_4$ 0.00 $\vec{V}_4$ 0.00 $\vec{V}_4$ 0.00 $\vec{V}_4$ 0.00 $\vec{V}_4$
$\vec{E}_5 \to \vec{V}_5$: 0.00 $\vec{V}_5$ 0.00 $\vec{V}_5$ 0.00 $\vec{V}_5$ 0.00 $\vec{V}_5$ 0.01 $\vec{V}_5$ 0.00 $\vec{V}_5$ 0.00 $\vec{V}_5$ 0.00 $\vec{V}_5$
$\vec{E}_6 \to \vec{V}_6$: 0.00 $\vec{V}_6$ 0.00 $\vec{V}_6$ 0.00 $\vec{V}_6$ 0.00 $\vec{V}_6$ 0.99 $\vec{V}_6$ 1.00 $\vec{V}_6$ 0.00 $\vec{V}_6$ 0.00 $\vec{V}_6$
$\vec{E}_7 \to \vec{V}_7$: 0.00 $\vec{V}_7$ 0.00 $\vec{V}_7$ 0.00 $\vec{V}_7$ 0.00 $\vec{V}_7$ 0.00 $\vec{V}_7$ 0.00 $\vec{V}_7$ 1.00 $\vec{V}_7$ 1.00 $\vec{V}_7$
$\vec{E}_8 \to \vec{V}_8$: 0.00 $\vec{V}_8$ 0.00 $\vec{V}_8$ 0.00 $\vec{V}_8$ 0.00 $\vec{V}_8$ 0.00 $\vec{V}_8$ 0.00 $\vec{V}_8$ 0.00 $\vec{V}_8$ 0.00 $\vec{V}_8$
$\Delta\vec{E}_1$ $\Delta\vec{E}_2$ $\Delta\vec{E}_3$ $\Delta\vec{E}_4$ $\Delta\vec{E}_5$ $\Delta\vec{E}_6$ $\Delta\vec{E}_7$ $\Delta\vec{E}_8$
$\vec{E}_1$ $\vec{E}_2$ $\vec{E}_3$ $\vec{E}_4$ $\vec{E}_5$ $\vec{E}_6$ $\vec{E}_7$ $\vec{E}_8$
$\Delta\vec{E}_1$ $\Delta\vec{E}_2$ $\Delta\vec{E}_3$ $\Delta\vec{E}_4$ $\Delta\vec{E}_5$ $\Delta\vec{E}_6$ $\Delta\vec{E}_7$ $\Delta\vec{E}_8$
$\vec{E}'_1$ $\vec{E}'_2$ $\vec{E}'_3$ $\vec{E}'_4$ $\vec{E}'_5$ $\vec{E}'_6$ $\vec{E}'_7$ $\vec{E}'_8$

Representing attention

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

Attention is all you need

CNN, RNN, LSTM, GAN,
Test time data,
Early stopping,
Data augmentation,
Dropout, Batch norm,
Gradient clipping

Attention

https://www.instagram.com/p/DHyTTONT4dl/

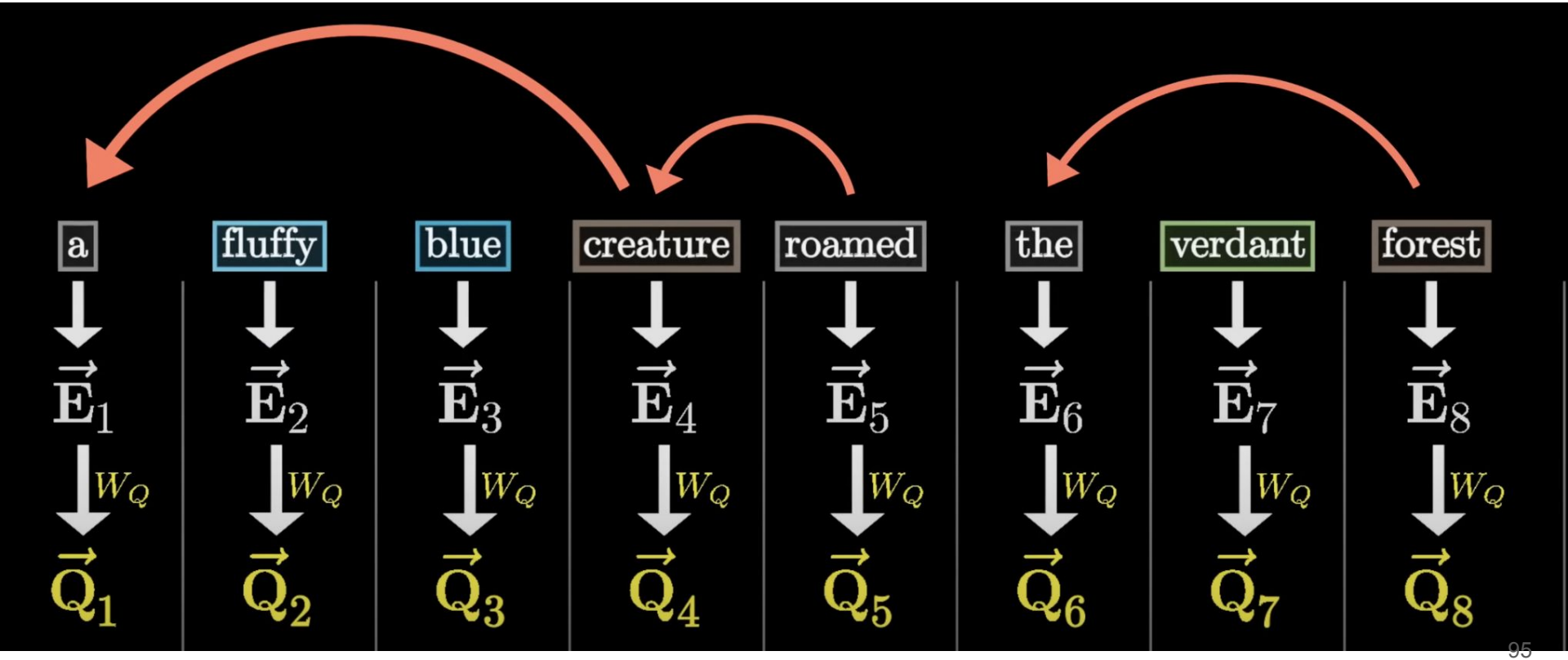
Attention hands-on

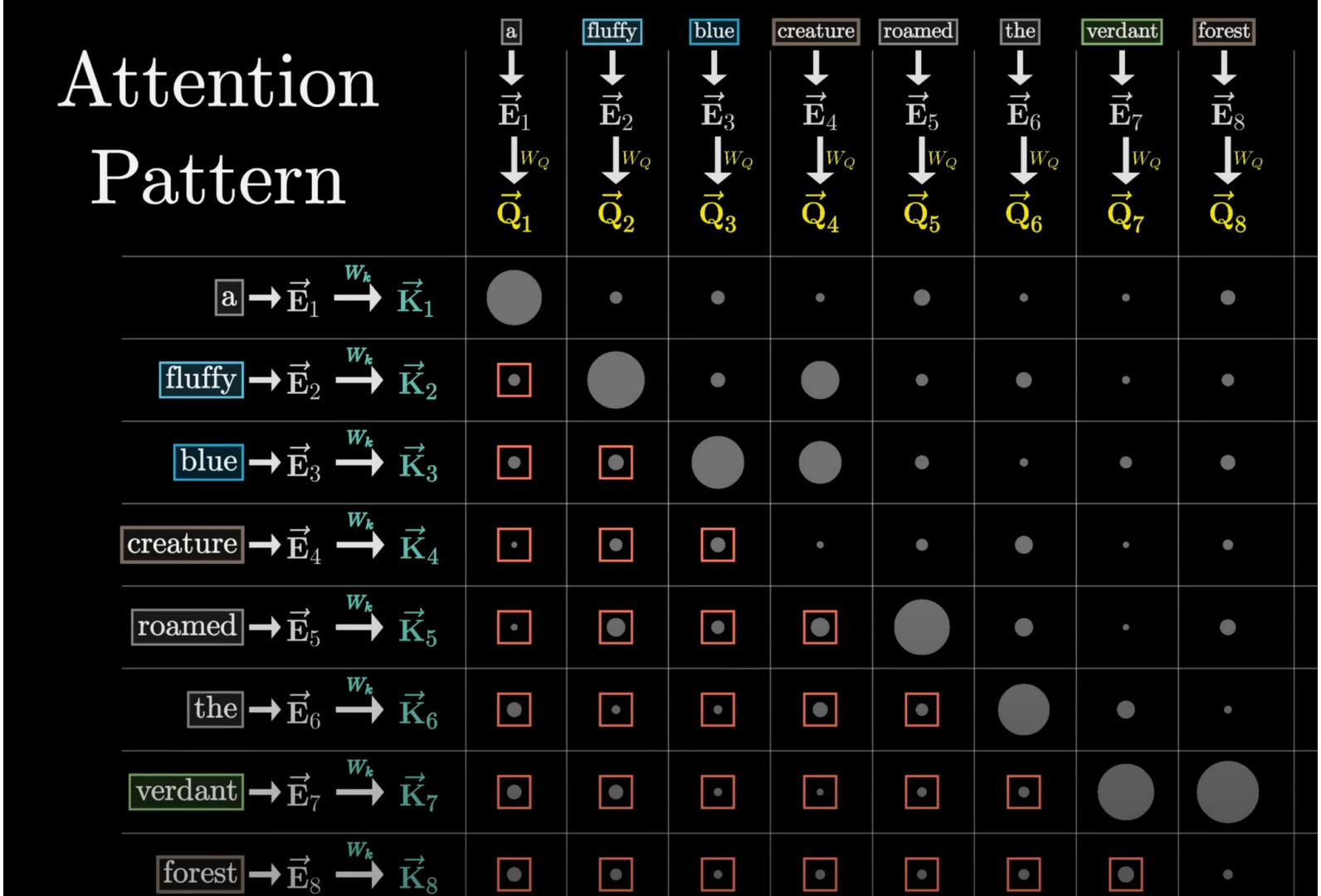
Let’s write our own attention implementation

For correctness, we will compare with Pytorch’s Scaled Dot Product Attention (SDPA) implementation

Masked . Multi-head . **Attention**

We don’t want what comes next to effect the previous words attention




Attention Pattern
a
fluffy
blue
creature
roamed
the
verdant
forest
$\vec{E}_1$ $\vec{E}_2$ $\vec{E}_3$ $\vec{E}_4$ $\vec{E}_5$ $\vec{E}_6$ $\vec{E}_7$ $\vec{E}_8$
W_Q
$\vec{Q}_1$ $\vec{Q}_2$ $\vec{Q}_3$ $\vec{Q}_4$ $\vec{Q}_5$ $\vec{Q}_6$ $\vec{Q}_7$ $\vec{Q}_8$
a → $\vec{E}_1$ W_K → $\vec{K}_1$
fluffy → $\vec{E}_2$ W_K → $\vec{K}_2$
blue → $\vec{E}_3$ W_K → $\vec{K}_3$
creature → $\vec{E}_4$ W_K → $\vec{K}_4$
roamed → $\vec{E}_5$ W_K → $\vec{K}_5$
the → $\vec{E}_6$ W_K → $\vec{K}_6$
verdant → $\vec{E}_7$ W_K → $\vec{K}_7$
forest → $\vec{E}_8$ W_K → $\vec{K}_8$

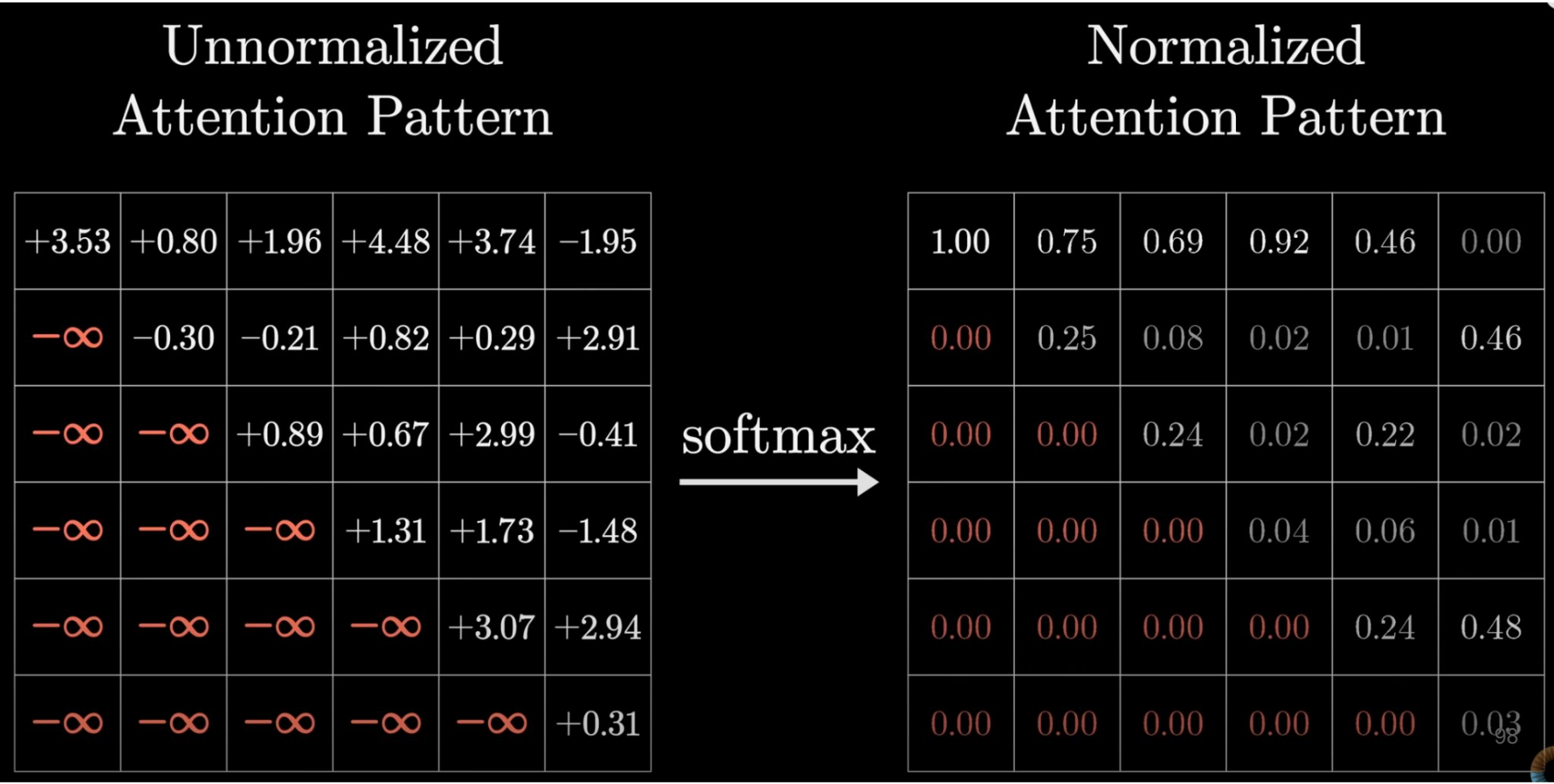
Unnormalized Attention Pattern

+3.53	+0.80	+1.96	+4.48	+3.74	−1.95
+1.90	−0.30	−0.21	+0.82	+0.29	+2.91
+1.52	+0.24	+0.89	+0.67	+2.99	−0.41
+0.63	−1.71	−5.11	+1.31	+1.73	−1.48
+4.54	−2.91	+0.09	−0.37	+3.07	+2.94
+0.31	+0.76	−1.78	−3.96	−0.70	+0.31

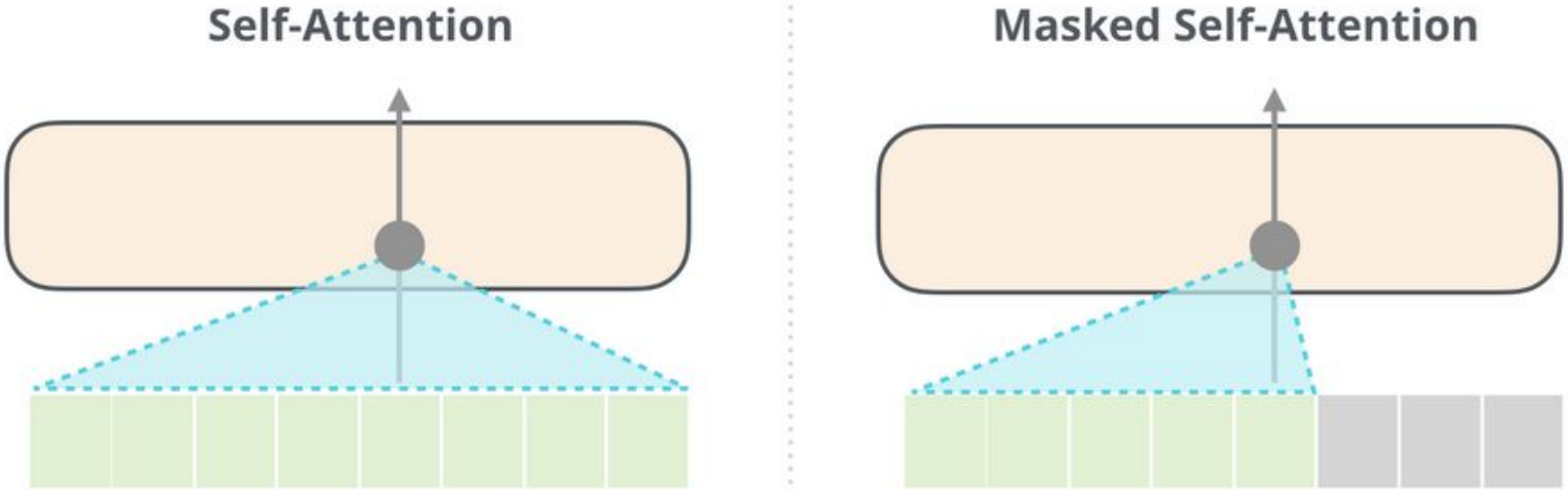
softmax

Normalized Attention Pattern

Attention Mask



Attention Mask



https://jalammar.github.io/illustrated-gpt2/

Masked . Multi-head . **Attention**

Query

$$\begin{bmatrix} -3.7 & +3.9 & -2.4 & -6.3 & -9.4 & -8.6 & +3.6 & -0.9 & \cdots & +0.7 \\ +7.9 & +9.7 & -5.6 & +3.2 & -4.7 & -9.5 & +5.1 & -3.6 & \cdots & -2.3 \\ +1.7 & +6.6 & +2.6 & +7.4 & -4.5 & +5.9 & -6.2 & +9.0 & \cdots & +3.7 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -5.6 & +8.9 & +4.6 & -4.9 & -5.7 & +0.4 & -9.4 & -5.8 & \cdots & -1.5 \end{bmatrix}$$

Key

$$\begin{bmatrix} -2.5 & -0.7 & -4.4 & +1.7 & +7.2 & -7.6 & +0.3 & -7.3 & \cdots & +4.3 \\ -2.1 & +1.3 & -6.3 & -7.0 & -0.2 & -2.9 & +8.7 & +5.3 & \cdots & +4.9 \\ +8.0 & -8.2 & +1.0 & +1.7 & +9.1 & -4.1 & -5.1 & -7.9 & \cdots & -9.6 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ +8.5 & +3.4 & +5.6 & -4.3 & +1.7 & -8.6 & -0.3 & +9.5 & \cdots & +7.5 \end{bmatrix}$$

How many parameters?

Value

$$\begin{bmatrix} -3.2 & +9.1 & -5.3 & +8.9 & +8.7 & +5.9 & +2.6 & +7.4 & \cdots & -4.1 \\ +6.9 & +2.3 & -9.6 & -3.0 & -7.0 & +9.5 & -0.4 & -0.1 & \cdots & +2.8 \\ -2.6 & -7.2 & +6.4 & -6.1 & +0.2 & -5.5 & -8.0 & +7.2 & \cdots & +9.4 \\ +9.1 & +8.0 & +5.4 & -3.3 & -8.3 & -1.8 & -5.3 & -7.3 & \cdots & -8.8 \\ +4.5 & -9.7 & +5.4 & -7.0 & -8.3 & -8.1 & +3.4 & -5.0 & \cdots & -1.6 \\ +1.1 & +7.1 & +4.5 & -4.5 & -7.3 & -8.8 & -3.9 & -4.7 & \cdots & -0.9 \\ +3.6 & +3.9 & -4.3 & -2.4 & -6.3 & +5.7 & -8.8 & +3.9 & \cdots & +5.5 \\ +5.5 & -4.8 & -2.5 & +1.7 & -4.5 & -2.6 & -6.0 & -0.8 & \cdots & -9.0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ +5.9 & -8.4 & +0.4 & -3.8 & +1.5 & +9.1 & +2.9 & -9.2 & \cdots & -1.4 \end{bmatrix}$$

Query

$$128 \times 12{,}288 = 1{,}572{,}864$$

$$128 \left\{ \overbrace{\begin{bmatrix} -3.7 & +3.9 & -2.4 & -6.3 & -9.4 & -8.6 & +3.6 & -0.9 & \cdots & +0.7 \\ +7.9 & +9.7 & -5.6 & +3.2 & -4.7 & -9.5 & +5.1 & -3.6 & \cdots & -2.3 \\ +1.7 & +6.6 & +2.6 & +7.4 & -4.5 & +5.9 & -6.2 & +9.0 & \cdots & +3.7 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -5.6 & +8.9 & +4.6 & -4.9 & -5.7 & +0.4 & -9.4 & -5.8 & \cdots & -1.5 \end{bmatrix}}^{12{,}288} \right.$$

Key

$$128 \times 12{,}288 = 1{,}572{,}864$$

$$128 \left\{ \overbrace{\begin{bmatrix} -2.5 & -0.7 & -4.4 & +1.7 & +7.2 & -7.6 & +0.3 & -7.3 & \cdots & +4.3 \\ -2.1 & +1.3 & -6.3 & -7.0 & -0.2 & -2.9 & +8.7 & +5.3 & \cdots & +4.9 \\ +8.0 & -8.2 & +1.0 & +1.7 & +9.1 & -4.1 & -5.1 & -7.9 & \cdots & -9.6 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ +8.5 & +3.4 & +5.6 & -4.3 & +1.7 & -8.6 & -0.3 & +9.5 & \cdots & +7.5 \end{bmatrix}}^{12{,}288} \right.$$

Value

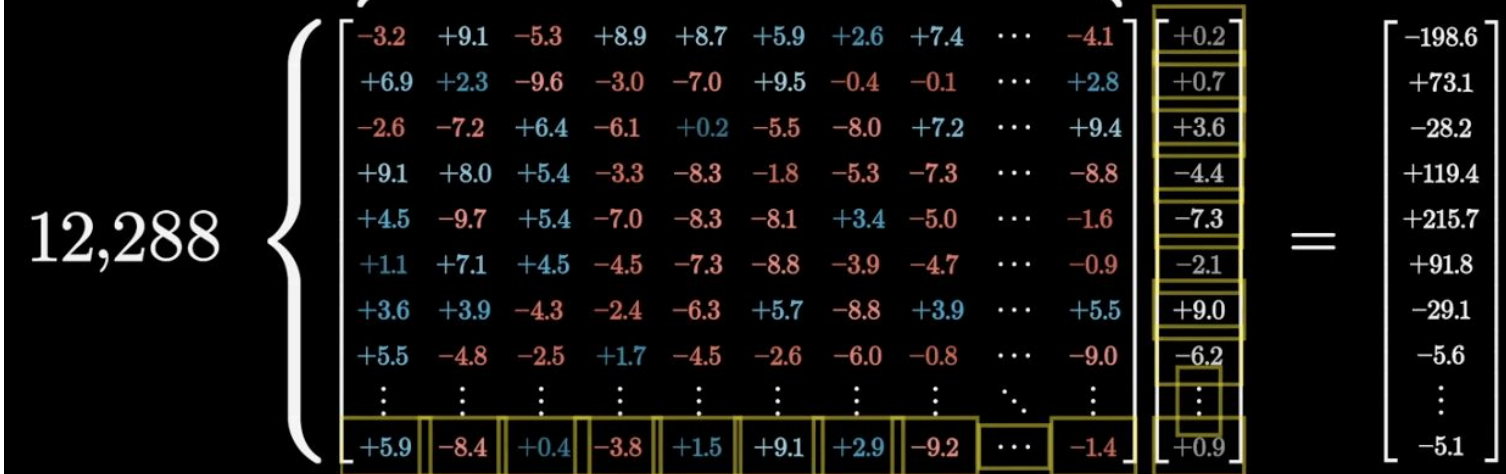
$$12{,}288 \times 12{,}288 = 150{,}994{,}944$$

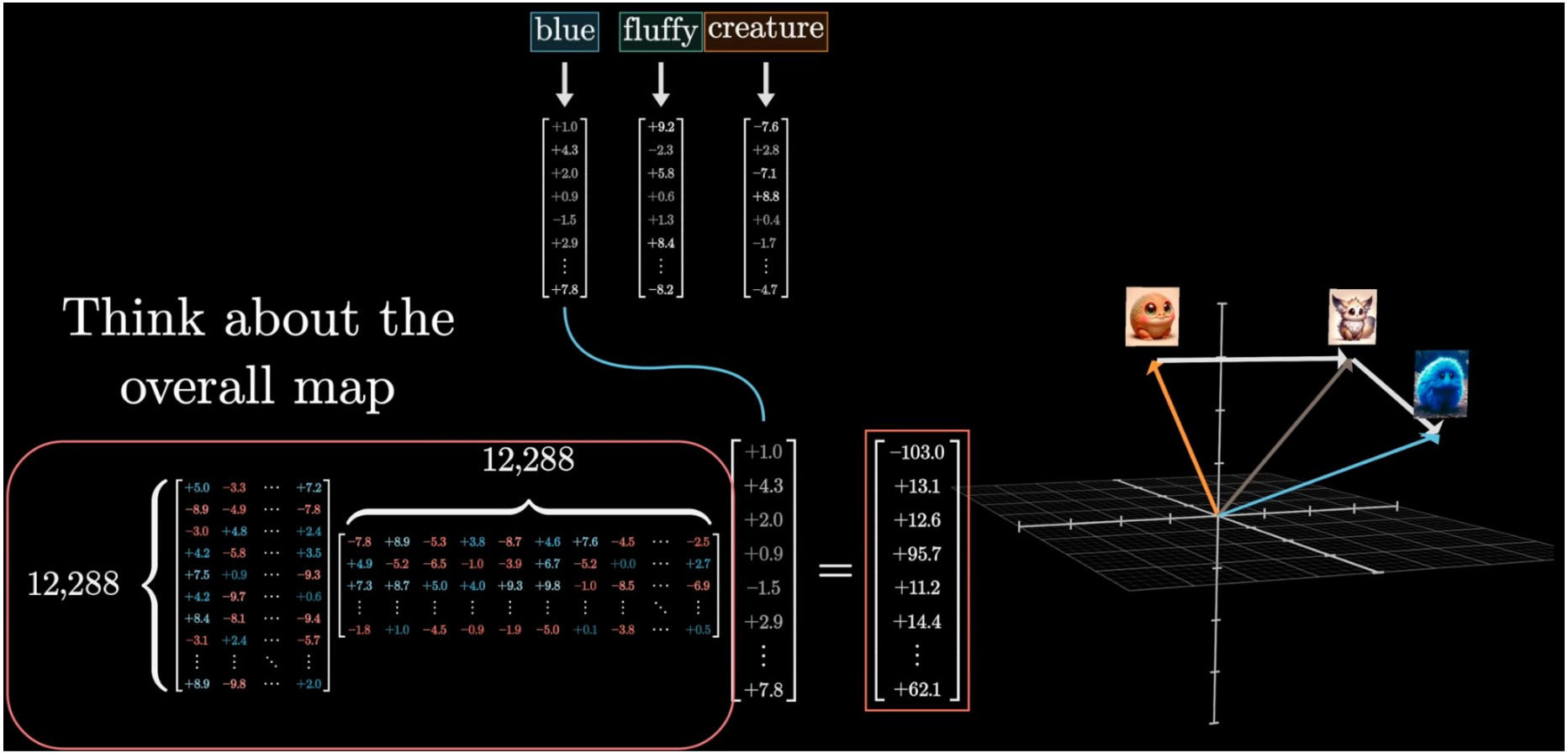
d_input
12,288

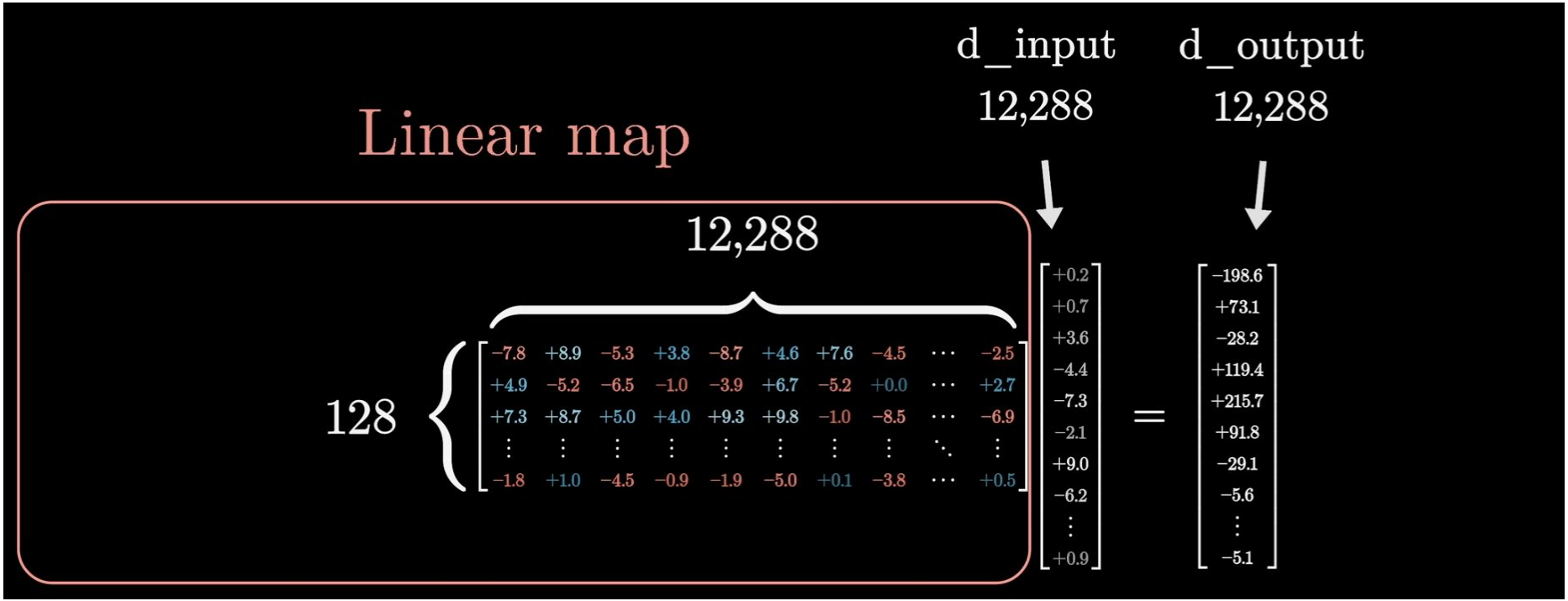
d_output
12,288

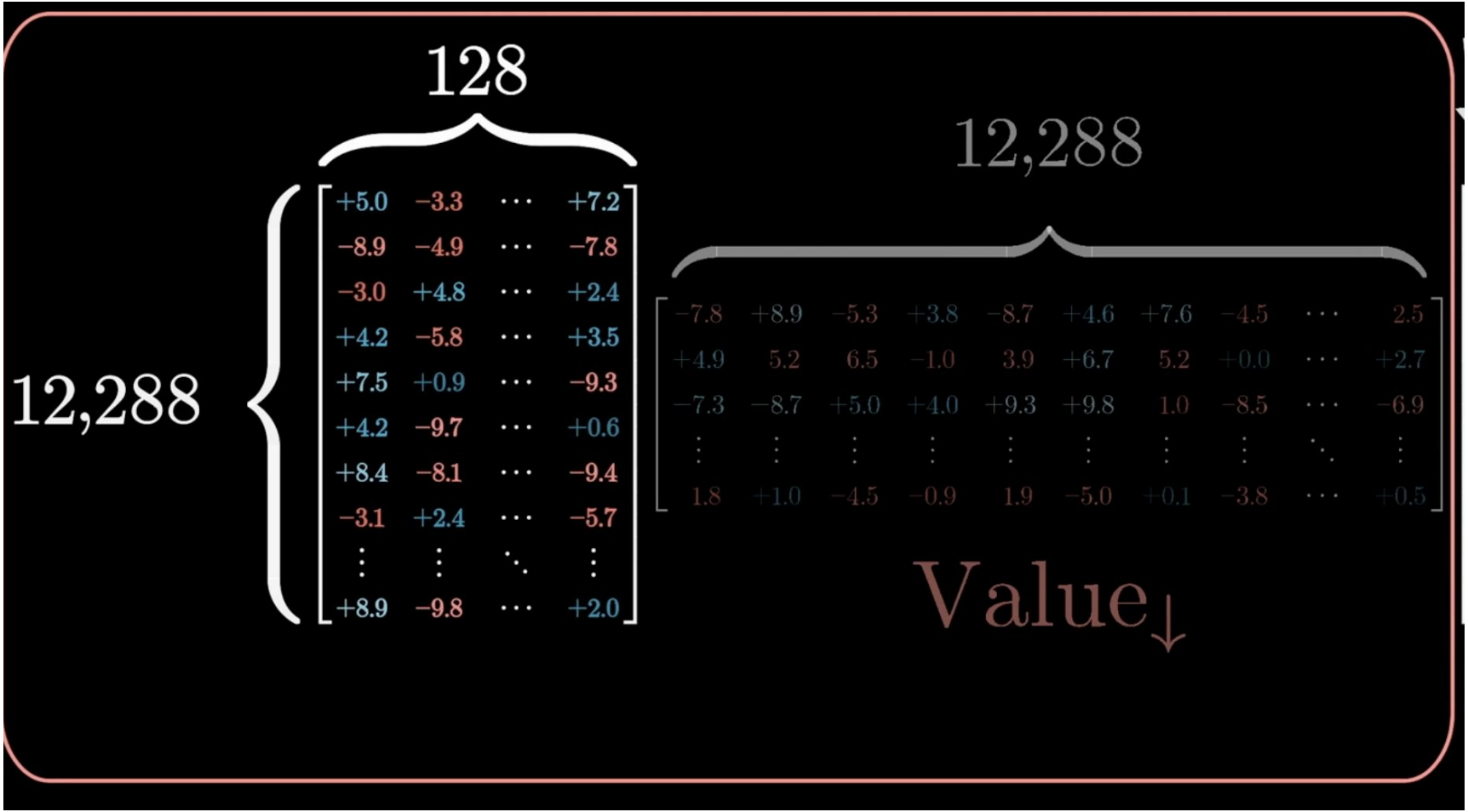
12,288

12,288



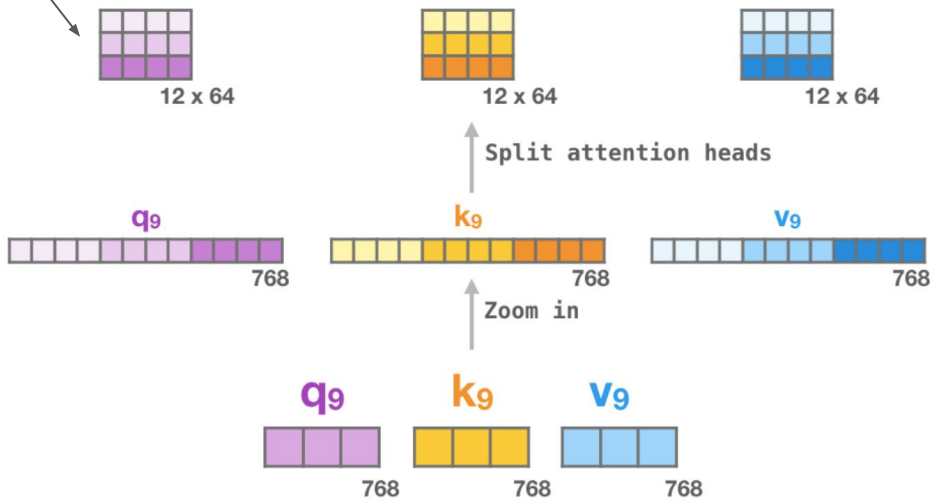

blue
fluffy
creature
Think about the overall map
12,288
12,288
=


d_input
12,288
d_output
12,288
Linear map
12,288
128
−7.8 +8.9 −5.3 +3.8 −8.7 +4.6 +7.6 −4.5 ⋯ −2.5
+4.9 −5.2 −6.5 −1.0 −3.9 +6.7 −5.2 +0.0 ⋯ +2.7
+7.3 +8.7 +5.0 +4.0 +9.3 +9.8 −1.0 −8.5 ⋯ −6.9
−1.8 +1.0 −4.5 −0.9 −1.9 −5.0 +0.1 −3.8 ⋯ +0.5
+0.2 +0.7 +3.6 −4.4 −7.3 −2.1 +9.0 −6.2 ⋮ +0.9
=
−198.6 +73.1 −28.2 +119.4 +215.7 +91.8 −29.1 −5.6 ⋮ −5.1


128
12,288
12,288
+5.0 −3.3 ⋯ +7.2
−8.9 −4.9 ⋯ −7.8
−3.0 +4.8 ⋯ +2.4
+4.2 −5.8 ⋯ +3.5
+7.5 +0.9 ⋯ −9.3
+4.2 −9.7 ⋯ +0.6
+8.4 −8.1 ⋯ −9.4
−3.1 +2.4 ⋯ −5.7
+8.9 −9.8 ⋯ +2.0
−7.8 +8.9 −5.3 +3.8 −8.7 +4.6 +7.6 −4.5 ⋯ 2.5
+4.9 5.2 6.5 −1.0 3.9 +6.7 5.2 +0.0 ⋯ +2.7
−7.3 −8.7 +5.0 +4.0 +9.3 +9.8 1.0 −8.5 ⋯ −6.9
1.8 +1.0 −4.5 −0.9 1.9 −5.0 +0.1 −3.8 ⋯ +0.5
Value↓

Multi head attention

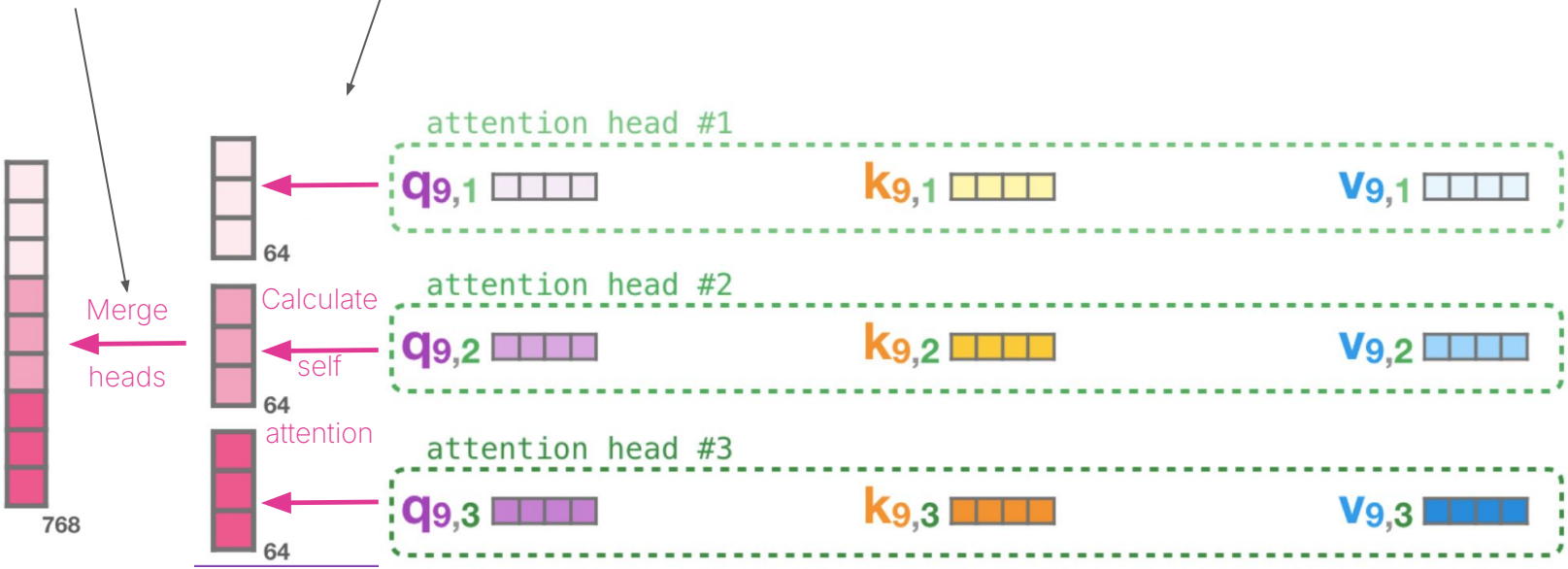
1. q, k and v are now split into a number of smaller pieces, called “heads”.



https://jalammar.github.io/illustrated-gpt2/#part-2-illustrated-self-attention

Multi head attention

2. Attention is calculated per head
3. Finally, we “merge” the heads by concatenating the results



https://jalammar.github.io/illustrated-gpt2/#part-2-illustrated-self-attention

How torch sdpa looks internally

```
T = q.shape[2]
att = (q @ k.transpose(-2, -1)) * (1.0 / math.sqrt(k.size(-1)))
att = att.masked_fill(self.bias[:,:,:T,:T] == 0, float('-inf'))
att = F.softmax(att, dim=-1)
y = att @ v # (B, nh, T, T) x (B, nh, T, hs) -> (B, nh, T, hs)
return y
```

The 2 sides of attention

It is THE engine of the current LLM wave

LLMs wouldn’t be what they are without the Attention block.

It is very costly - O(n2)

Where n is sequence length.

Longer and longer context becomes harder.

Attention is a lot more than this!

Sliding Window

Don't focus on all tokens

Linear

Switch from Quadratic form to an approximate linear form

Hybrid

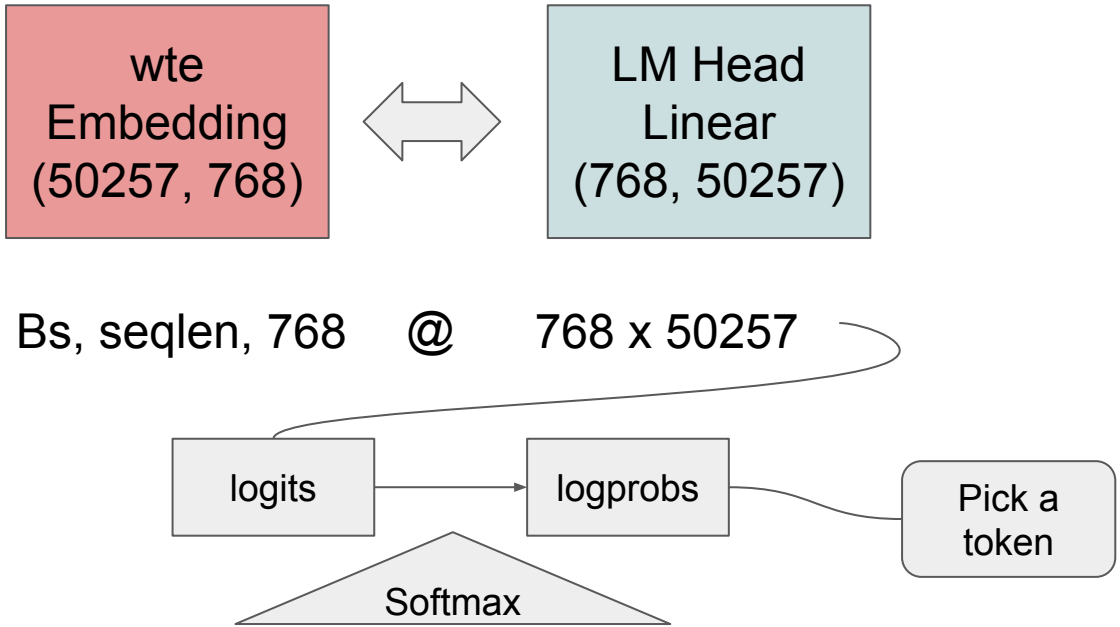
Mix and match various Attention blocks in a single architecture

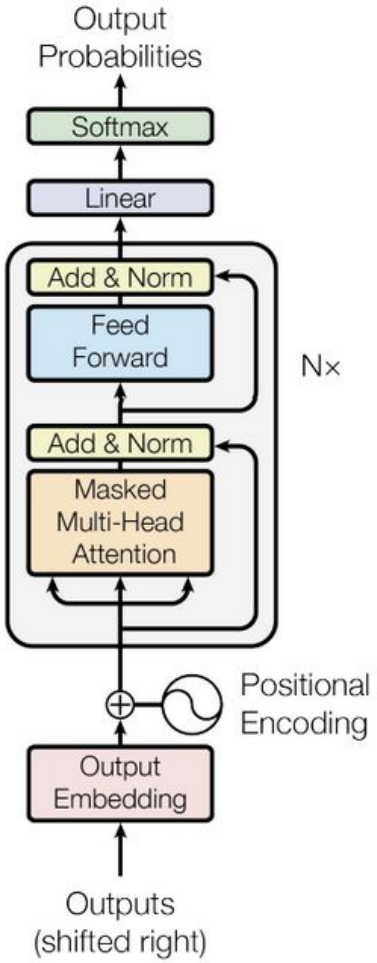
Example of latest research work here:
Jet-Nemotron from Nvidia, 21'st Aug, 2025:
https://www.arxiv.org/pdf/2508.15884

Search for the best attention configuration for a task + hardware

LM Head and Softmax

The last Linear layer is called LMHead





Let’s write our own GPT2 Model: Putting the layers together

2 step exercise

Write the module definitions of all the components of the GPT2 Model and load the weights.

Win condition:

No error in loading from the state dict.

```
import torch
mw =
torch.load("pytorch_model.bin")
list(mw.keys())
```

Write the forward definitions of all the components.

Win condition:

When running `generate.py`, you are able to generate proper English sentences.

Step 1: Create the modules (fill in gpt2.py)

Complete the init for all modules using the provided empty classes and hints

1. Use the weight state dict keys to get the class field names.
2. Use the comments as hints for the type (LayerNorm, Conv1D, MLP, Embedding or another Module)
3. Use the dict values shapes as the size for the Conv/Embedding/LayerNorm modules

It works if you are able to load weights without error.

Final answer is in: solutions/gpt_with_init.py

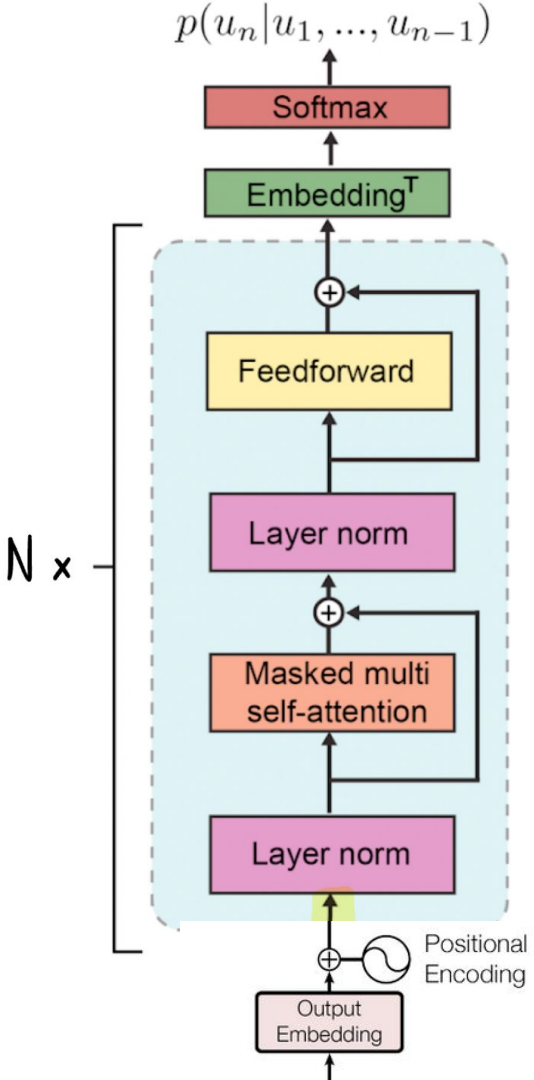
Step 2: Fill in the forwards

Now, write the forward for all the modules. Use the MLP you have already written.

This is how a single block should look.

Make the same calls in the way shown and make sure to add the residual.

If everything works, run generate.py which should produce good output.



Recap

Transformer architecture - GPT2

From here:

- Explore other features of HF ecosystem
- Explore other model architectures (ex. Advancements like MoE)
- Explore Hybrid architectures (Mamba attention)
- Explore Multi-modal LMs such as image, video, audio