LLM Architecture & Training

Shreya Havaldar

Announcements

- HW 1 is due today!
  - Let me know if you have issues with EdStem or Gradescope.

- Office hours will be in AGH 300E. **Email me if you don’t have access to AGH. All SEAS students should have access.**

Auditing

- You are allowed to attend lectures, attend my office hours, and (optionally) complete the assignments and final project.

- You are NOT allowed to attend the TA's office hours, receive grades on your assignments, or get credit for the course.

- All lectures and assignments will be available on the website. If you would like access to the EdStem group, lmk after class!

Today’s Lecture

1. **Neural Networks**
2. Training
3. The Transformer

Neural networks are high-dimensional functions

- A neural network is a high-dimensional function $f(\Theta, x) \rightarrow y$
  - Goal: Map inputs to outputs.
  - LLM-specific output: A probability distribution over a vocabulary.

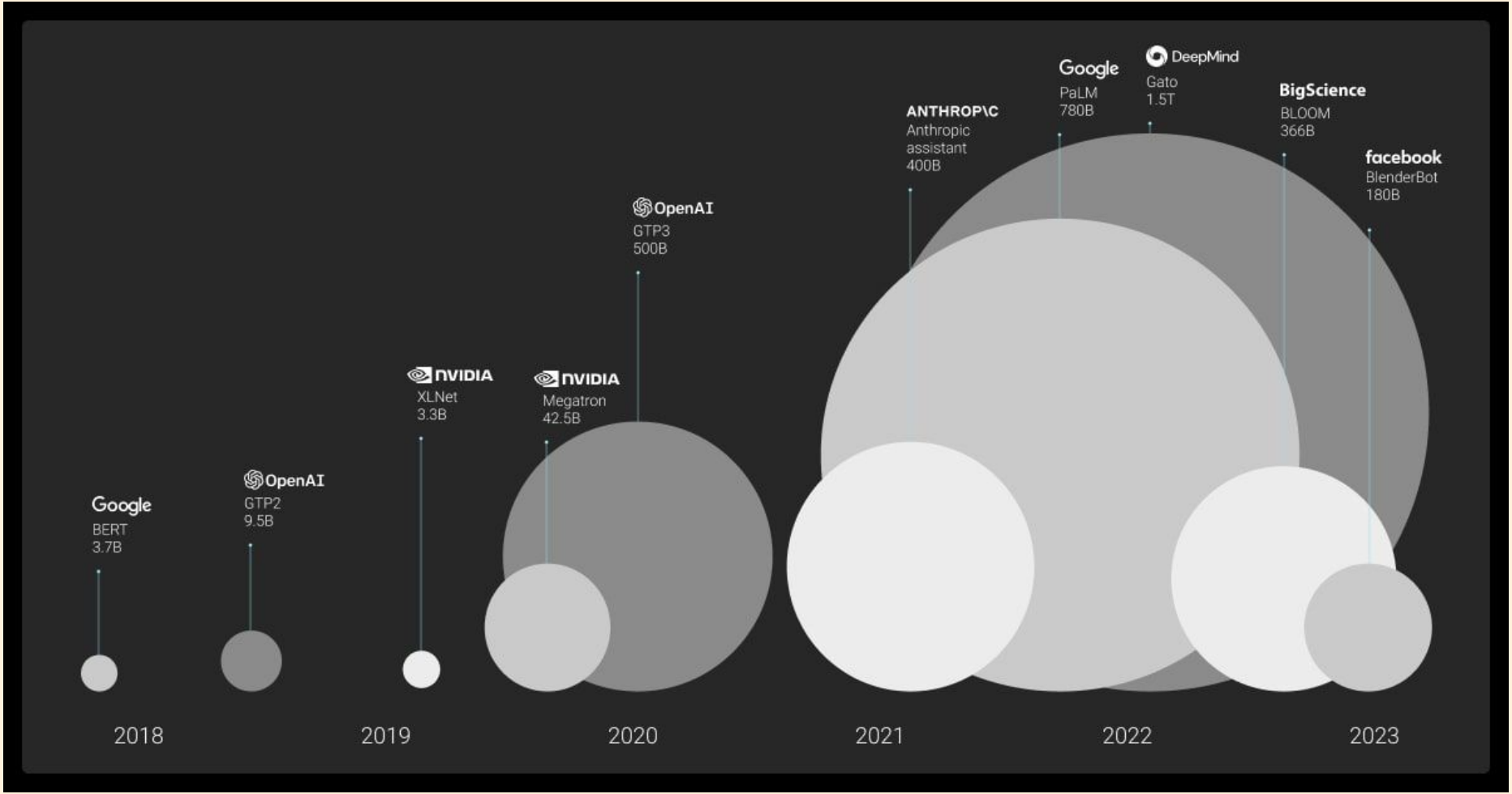
- Consist of:
  - Parameters (learned weights + biases)
  - Activation functions
  - Input, hidden, output layers

Parameters

Parameters store the knowledge acquired during training and do not change while the model is being used.

Weights: Numerical values representing the strength of connections between neurons, adjusted during learning

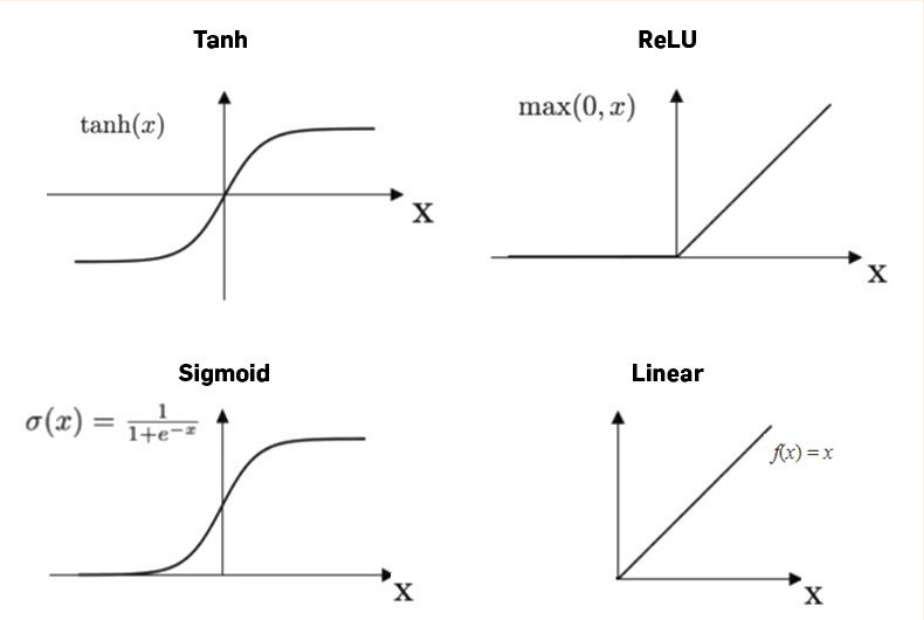
Biases: Additional parameters that help shift the activation function, giving the network more flexibility.


Google
BERT
3.7B
OpenAI
GTP2
9.5B
NVIDIA
XLNet
3.3B
NVIDIA
Megatron
42.5B
OpenAI
GTP3
500B
ANTHROP\C
Anthropic
assistant
400B
Google
PaLM
780B
DeepMind
Gato
1.5T
BigScience
BLOOM
366B
facebook
BlenderBot
180B
2018
2019
2020
2021
2022
2023

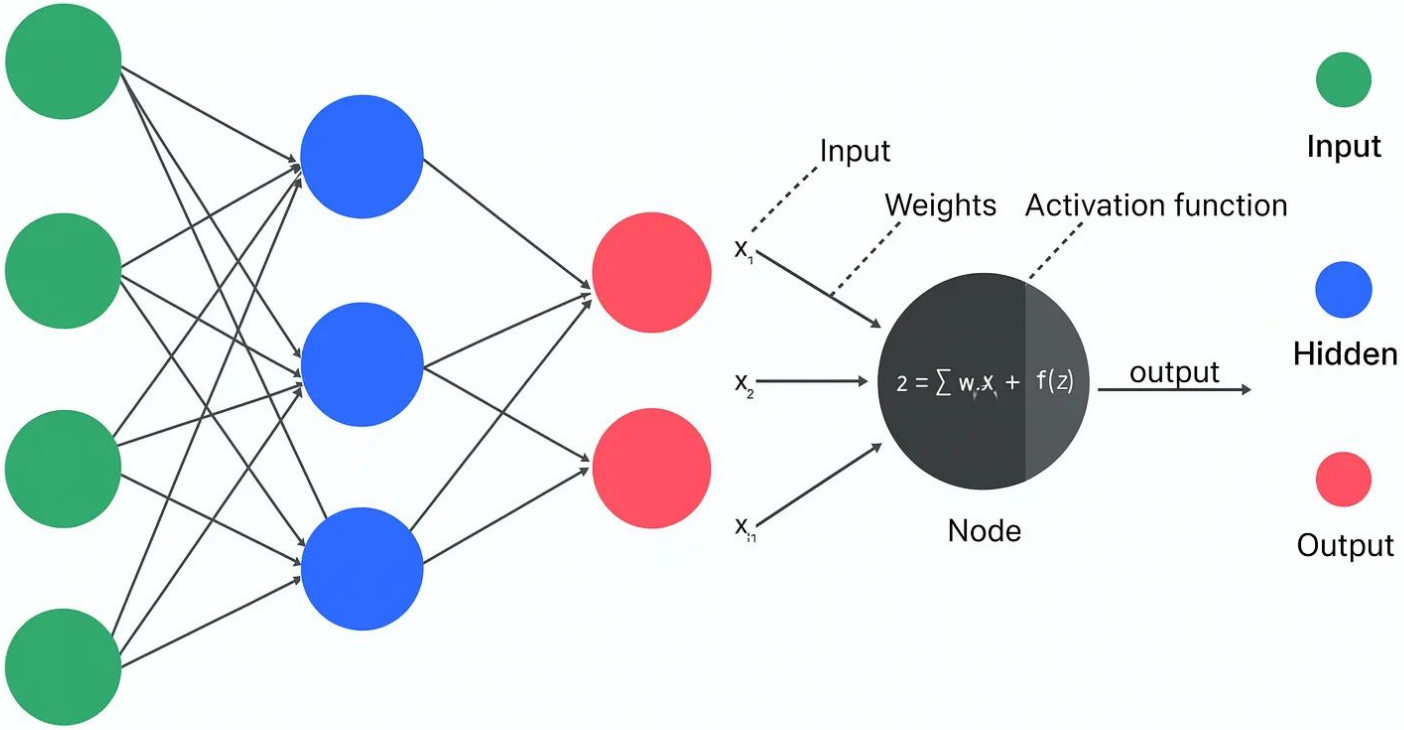
Activation Functions

A mathematical function that decides whether a neuron should "fire" (activate) and pass information to the next layer

- Introduces **non-linearity**
- Changes with every different input



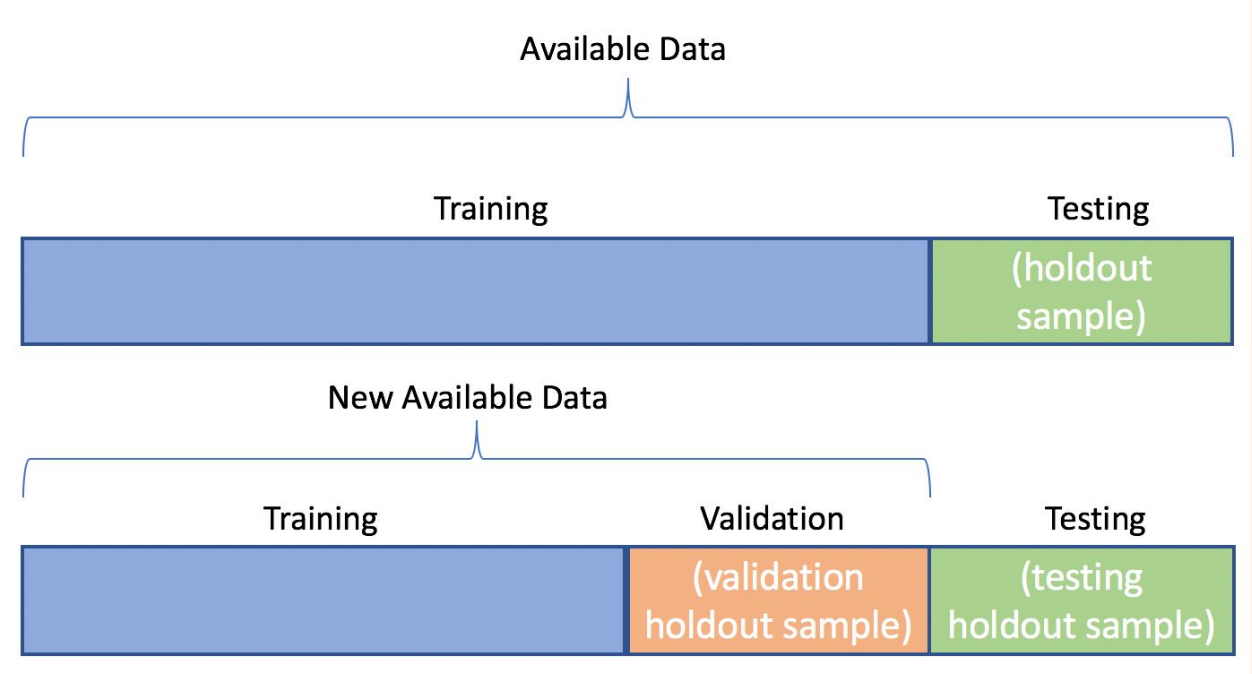
Neural Network Architecture



Today’s Lecture

1. Neural Networks
2. **Training**
3. The Transformer

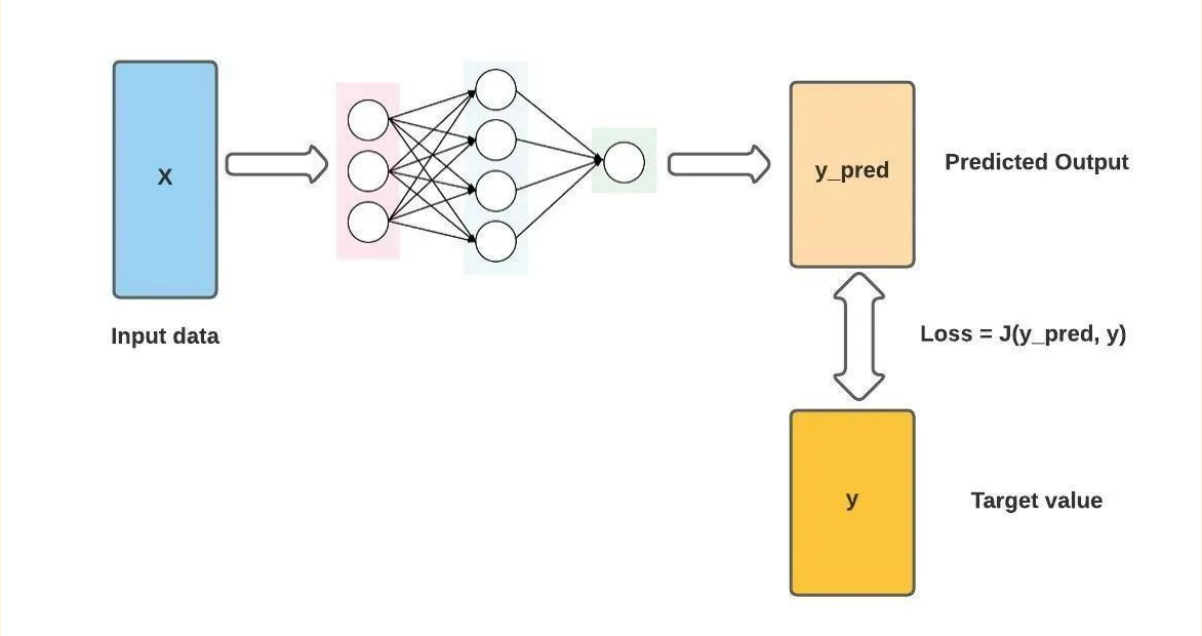
Train, Val, Test



The Loss Function

A scalar value measuring the discrepancy between prediction and ground truth.

Cost function: average loss over entire training set



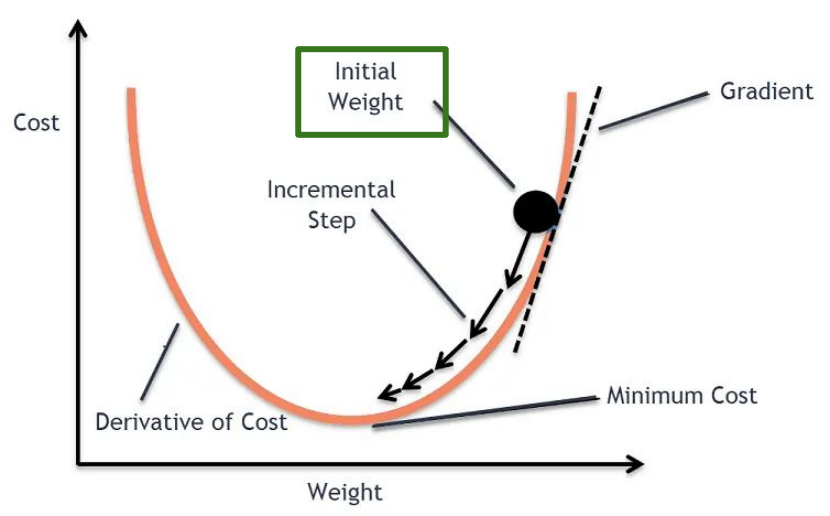
Gradient Descent

- An iterative optimization algorithm used in machine learning to find the minimum of a cost or loss function
  - Optimization problem in a high-dimensional landscape

- All training pipelines use gradient descent/ascent
  - Supervised fine-tuning
  - Instruction tuning
  - Preference optimization

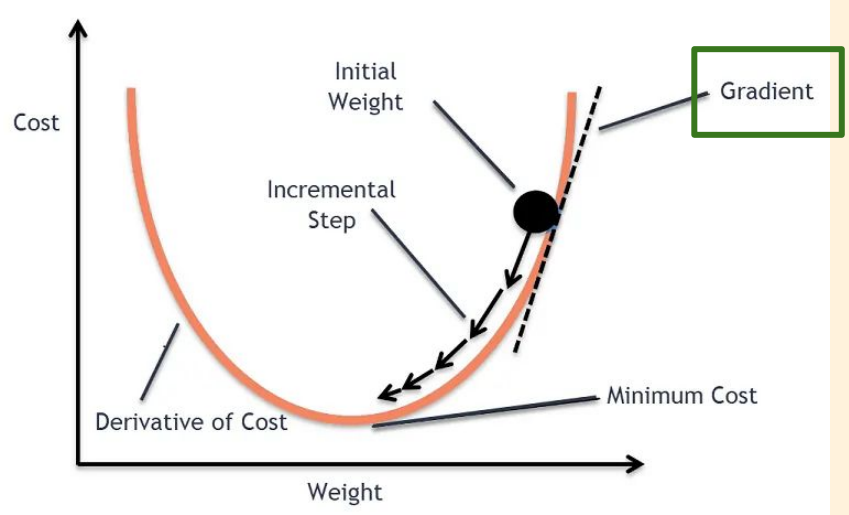
Gradient Descent

1. The algorithm begins with initial values for the model's parameters.



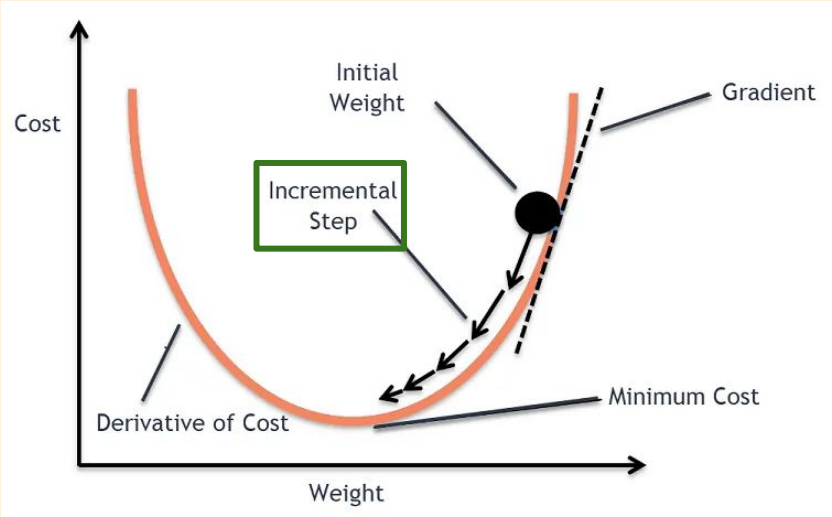
Gradient Descent

2. It computes the gradient (slope) of the cost function, which indicates the direction of the steepest increase.



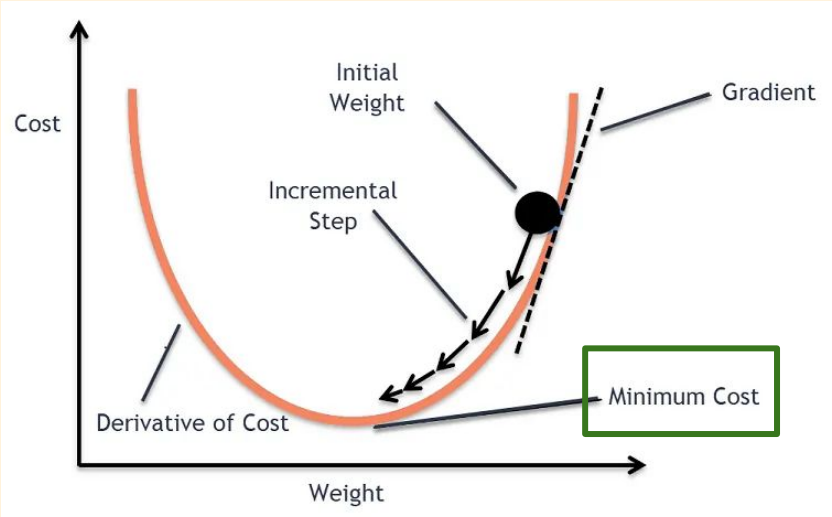
Gradient Descent

3. It then updates the parameters by taking a small step in the opposite direction of the gradient (the negative gradient), which is the direction of steepest decrease.



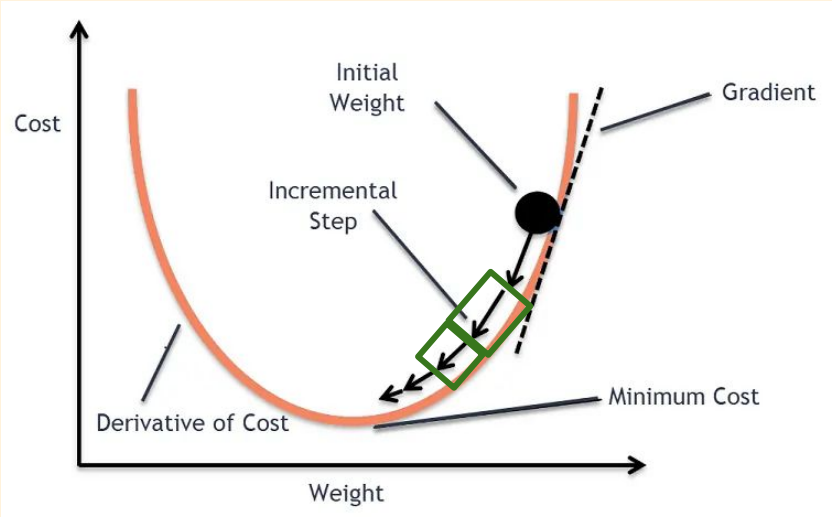
Gradient Descent

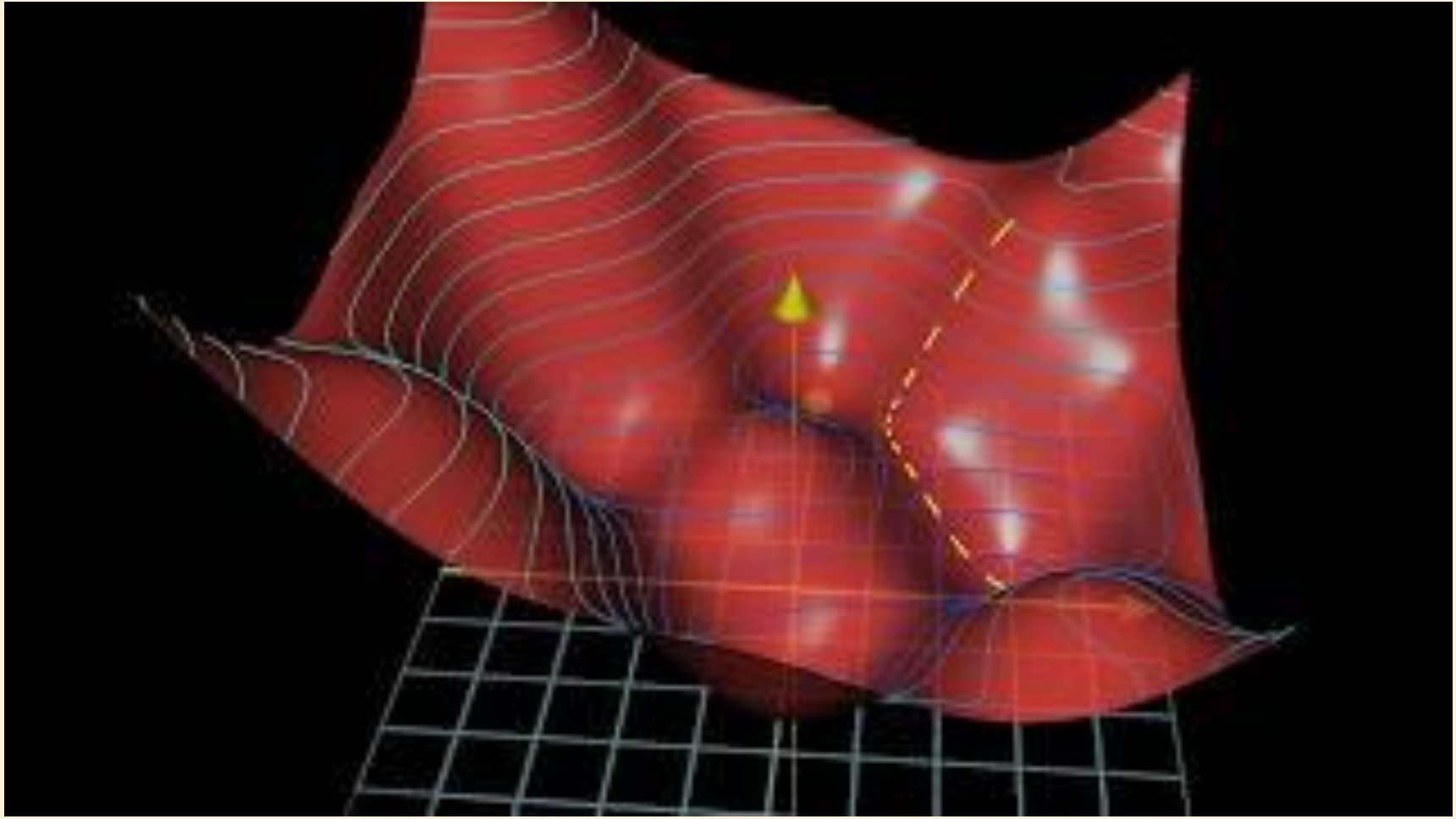
4. This process repeats, with each step getting smaller as it approaches the minimum, until the model converges (the loss stops changing significantly).



Learning Rate

A parameter that controls how much the weights are adjusted with respect to the gradient of the loss function during training. It determines **the step size taken towards the minimum** of a loss function





Discussion (10 min)

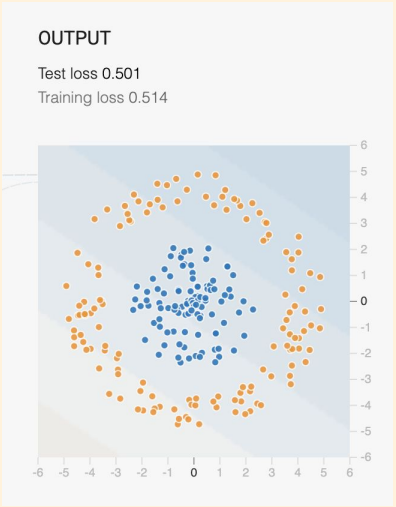
1. Explain how gradient descent works, intuitively

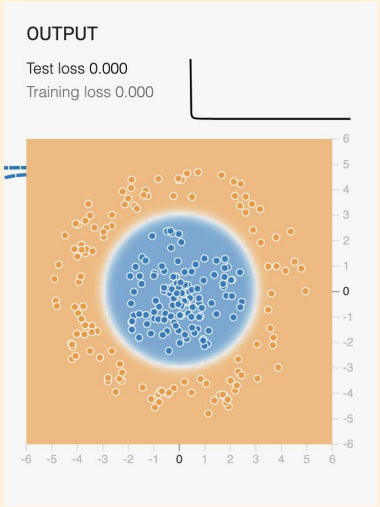
2. **[Use AI]** Look up what “local minima” are. How can you modify training to escape a local minimum?

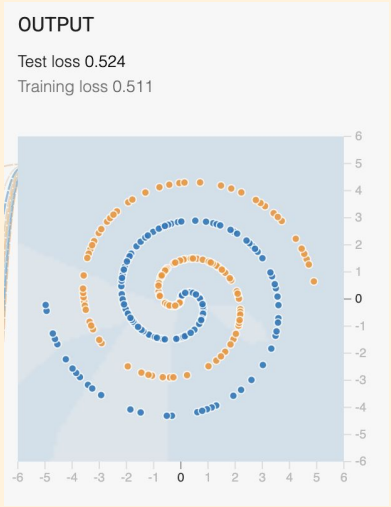
Activity (15 min)

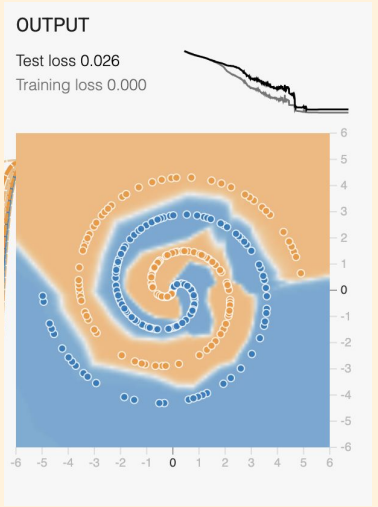
Go to playground.tensorflow.org

Adjust features, layers, neurons, activations, and learning rate.









Discussion (10 min)

1. **[Use AI]** If training loss goes down but test loss rises, what is happening? What are 2–3 actions you could take, and what outcome would you expect from each?

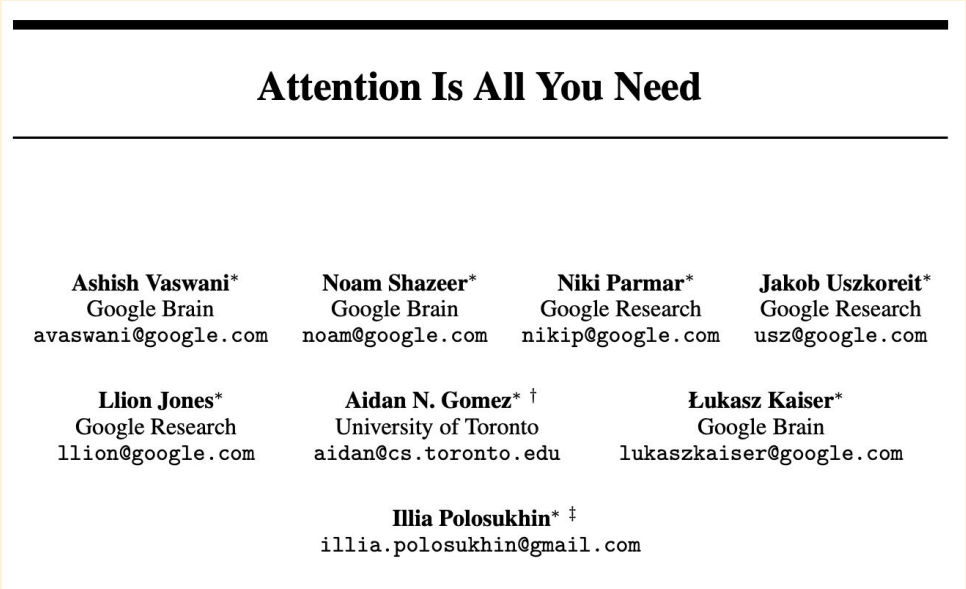
2. **[Use AI]** If both training and test loss don’t go down, what is happening? What are 2–3 actions you could take, and what outcome would you expect from each?

Today’s Lecture

1. Neural Networks
2. Training
3. **The Transformer**

The transformer

A specific kind of network architecture. It is still a feedforward neural network, but based on attention.



Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Components of the Transformer

1. **Token embedding converts text into continuous vectors**
2. Positional encoding introduces sequence order
3. Self-attention layers compute token-to-token relationships
4. Feed-forward layers transform representations
5. Stacked blocks enable deep hierarchical features

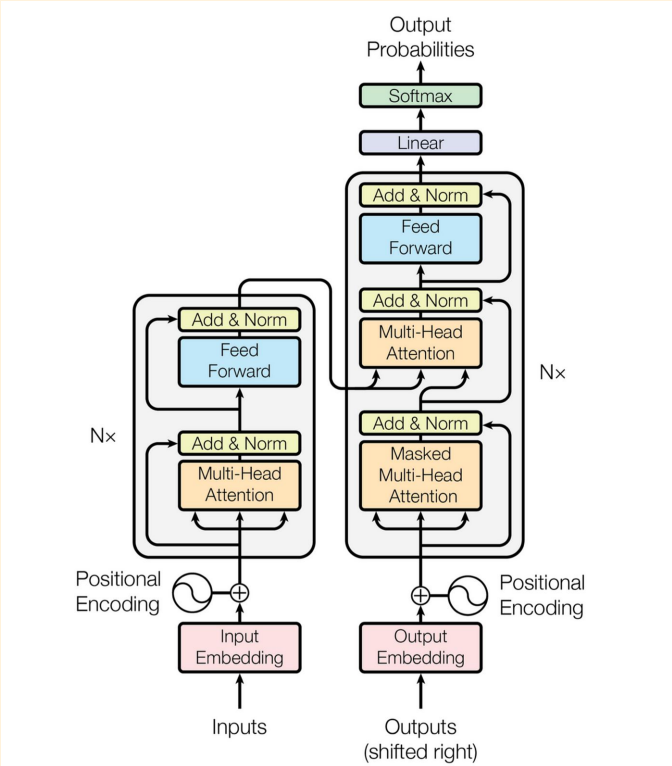
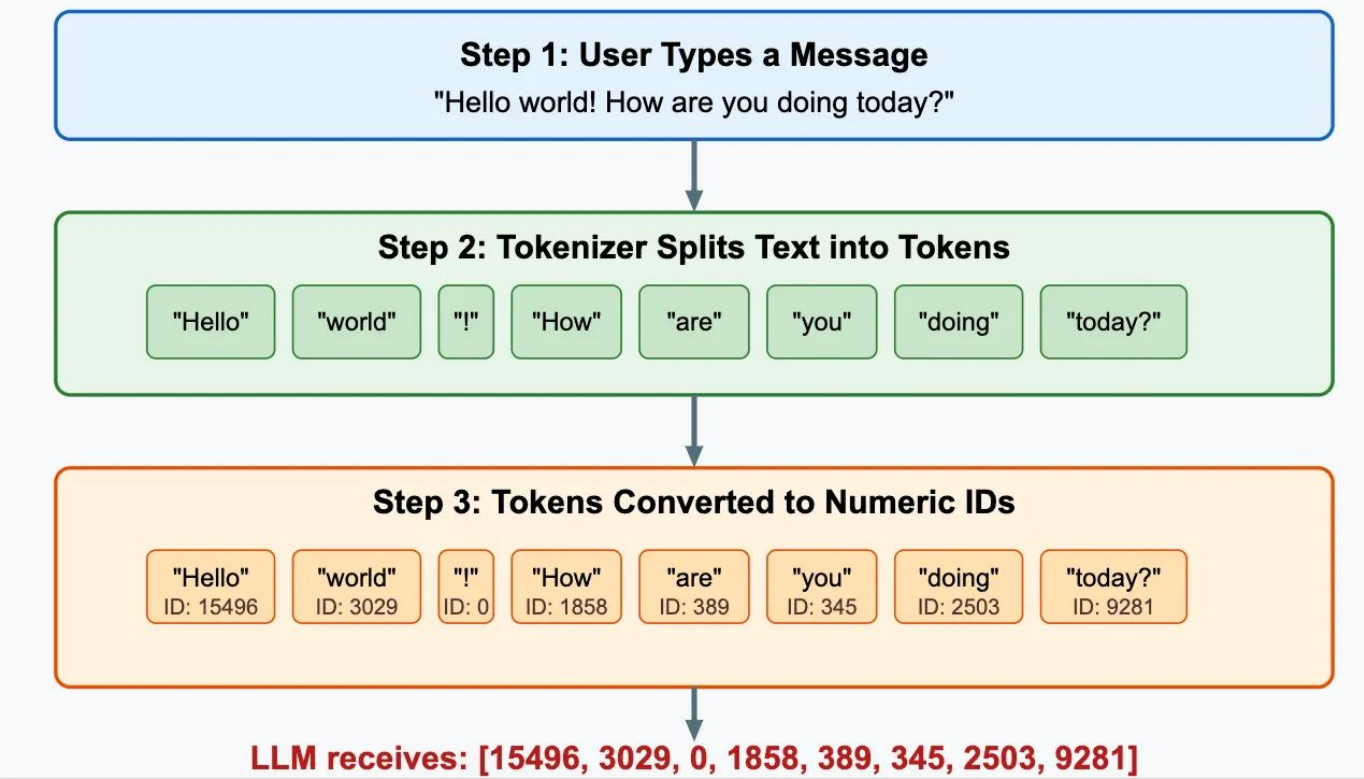


Figure 1: The Transformer - model architecture.

Raw Text to Tokens



Raw Text to Tokens

How is text split into tokens? Via a *tokenization algorithm*

- “unbelievable” → “un | believ | able”
- “GPT-5?” → “GPT”, “-”, “5”, “?”
- Emojis → each emoji is usually a token
- URLs → often split into many tokens

Why a good tokenization algorithm matters for efficiency.

- Tokens too short: text is broken up into too many pieces
- Tokens too long: the model can’t generalize to unseen words

Text to Tokens: Exercise

Exercise: How does each tokenization scheme handle the following words
Training words: “burning”, “satisfactory”, “information”

Word-Level	Subword	Character-level
running satisfactory information	burn #ing satis #fac# #tory inform #a# #tion	b u r n i g s … etc.

1. Which vocabulary produces the fewest tokens overall for the training words?

2. Which vocabulary is the most flexible for unseen words?

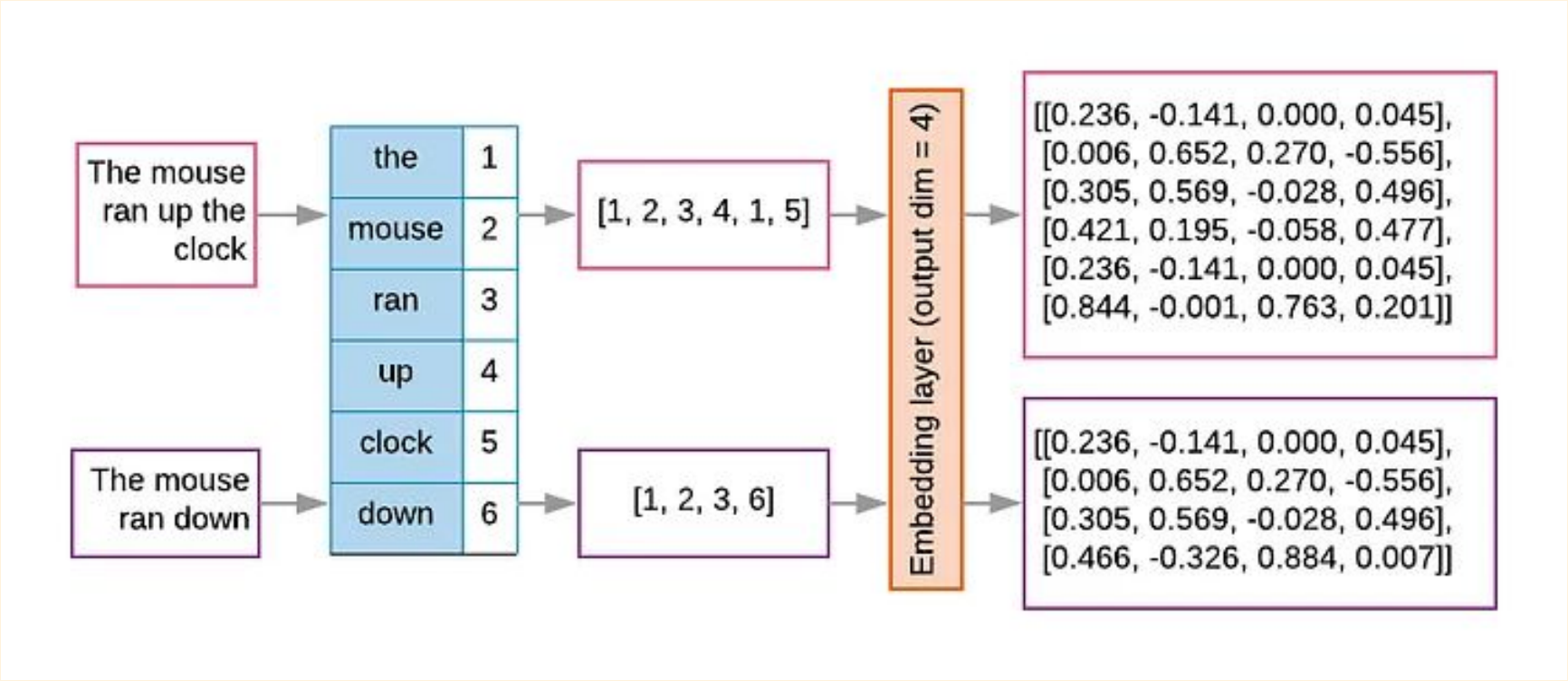
Text to Tokens: Exercise

Exercise: How does each tokenization scheme handle the following words
Unseen words: “satisfaction”, “informing”

Word-Level	Subword	Character-level
running satisfactory information	burn #ing satis #fac# #tory inform #a# #tion	b u r n i g s … etc.

1. Which vocabulary produces the fewest tokens overall for the unseen words?
2. Which vocabulary is the least efficient for unseen words?

Embeddings


The mouse ran up the clock
The mouse ran down
the 1
mouse 2
ran 3
up 4
clock 5
down 6
[1, 2, 3, 4, 1, 5]
[1, 2, 3, 6]
Embedding layer (output dim = 4)
[[0.236, -0.141, 0.000, 0.045],
[0.006, 0.652, 0.270, -0.556],
[0.305, 0.569, -0.028, 0.496],
[0.421, 0.195, -0.058, 0.477],
[0.236, -0.141, 0.000, 0.045],
[0.844, -0.001, 0.763, 0.201]]
[[0.236, -0.141, 0.000, 0.045],
[0.006, 0.652, 0.270, -0.556],
[0.305, 0.569, -0.028, 0.496],
[0.466, -0.326, 0.884, 0.007]]

Embeddings

What do embeddings capture?

- Semantic similarity
- Syntactic roles
- Frequency-based structure

Why are embeddings useful?

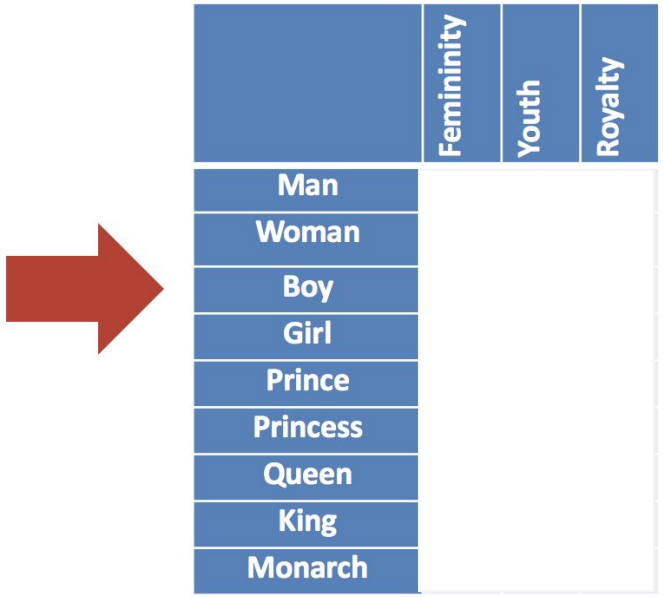
- Numerical representation of language
- Words used similarly are close in vector space

Embeddings are learned from next-token prediction during LLM training

Embeddings: Exercise

Try to build a lower dimensional embedding

Vocabulary:
Man, woman, boy, girl, prince, princess, queen, king, monarch



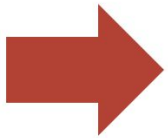
Each word gets a 1x3 vector

Similar words… similar vectors

Embeddings: Exercise

Try to build a lower dimensional embedding

Vocabulary:
Man, woman, boy, girl, prince, princess, queen, king, monarch



	Femininity	Youth	Royalty
Man	0	0	0
Woman	1	0	0
Boy			
Girl			
Prince			
Princess			
Queen			
King			
Monarch			

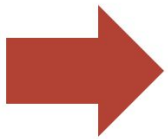
Each word gets a 1x3 vector

Similar words… similar vectors

Embeddings: Exercise

Try to build a lower dimensional embedding

Vocabulary:
Man, woman, boy, girl, prince, princess, queen, king, monarch



	Femininity	Youth	Royalty
Man	0	0	0
Woman	1	0	0
Boy	0	1	0
Girl	1	1	0
Prince	0	1	1
Princess	1	1	1
Queen	1	0	1
King	0	0	1
Monarch	0.5	0.5	1

Each word gets a 1x3 vector

Similar words… similar vectors

Step 2: Embeddings

We can’t simply use “femininity”, “youth”, or “royalty” to describe all words in the English language

More on embeddings:

- Each dimension doesn’t measure a clear construct; it’s quite complex
- Common dimensionality is 300-700
- But, research shows that enough is captured to do “word math” with embeddings
  - *“king – man + woman ≈ queen”*
  - *“doctor – hospital + school ≈ teacher”*

Components of the Transformer

1. Token embedding converts text into continuous vectors
2. **Positional encoding introduces sequence order**
3. Self-attention layers compute token-to-token relationships
4. Feed-forward layers transform representations
5. Stacked blocks enable deep hierarchical features

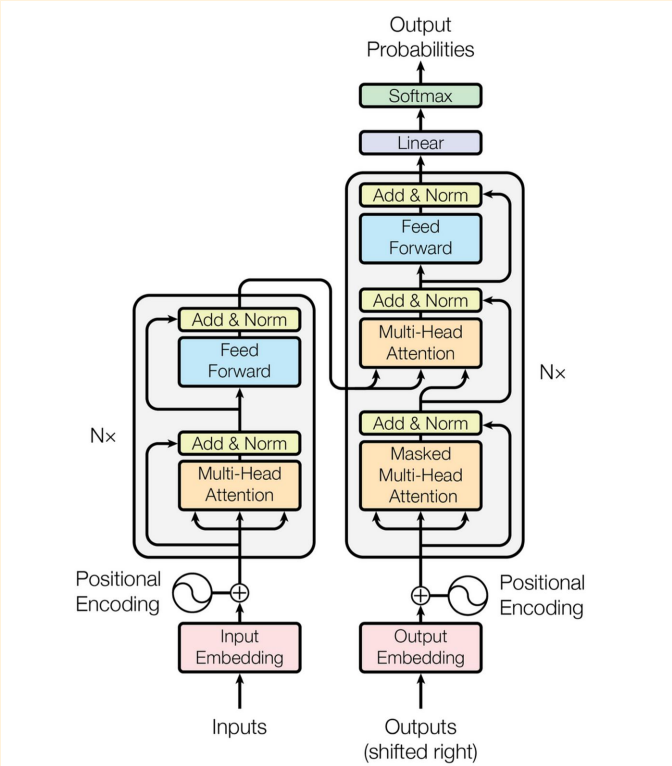
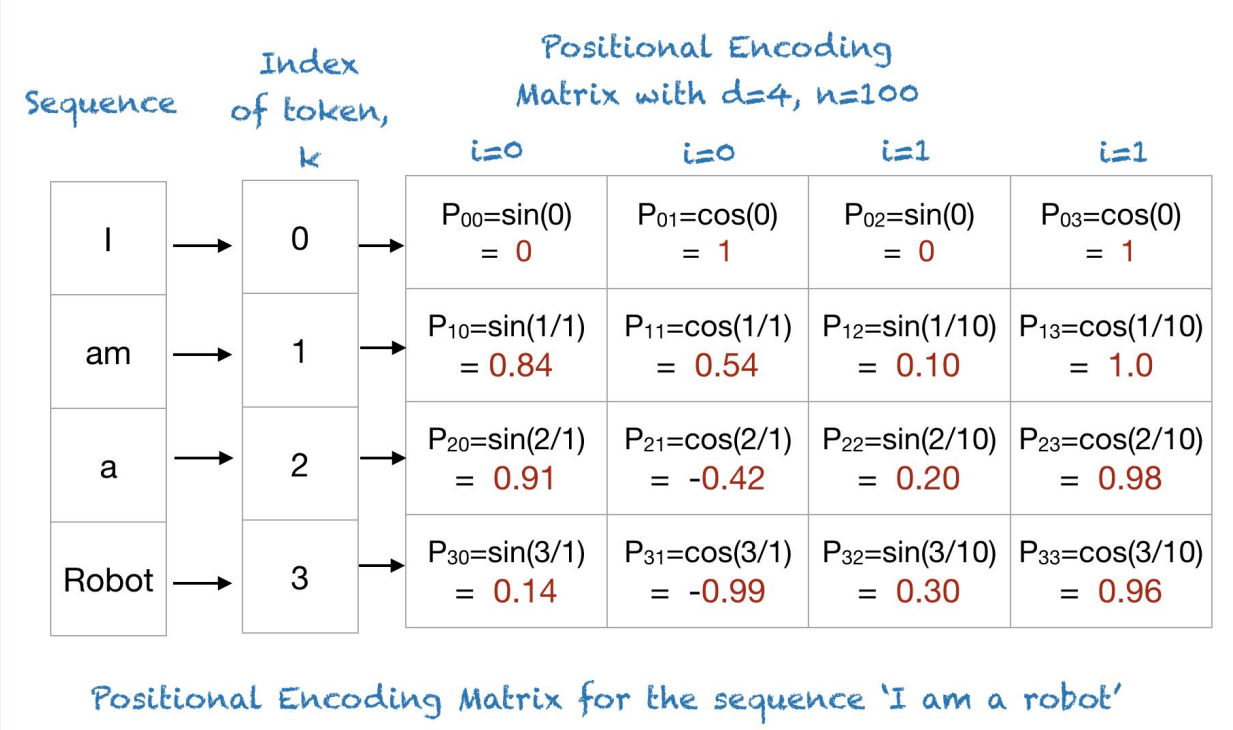


Figure 1: The Transformer - model architecture.

Positional Encodings

Transformers process tokens in parallel

Positional encodings allow transformers to know the order of tokens



Components of the Transformer

1. Token embedding converts text into continuous vectors
2. Positional encoding introduces sequence order
3. **Self-attention layers compute token-to-token relationships**
4. Feed-forward layers transform representations
5. Stacked blocks enable deep hierarchical features

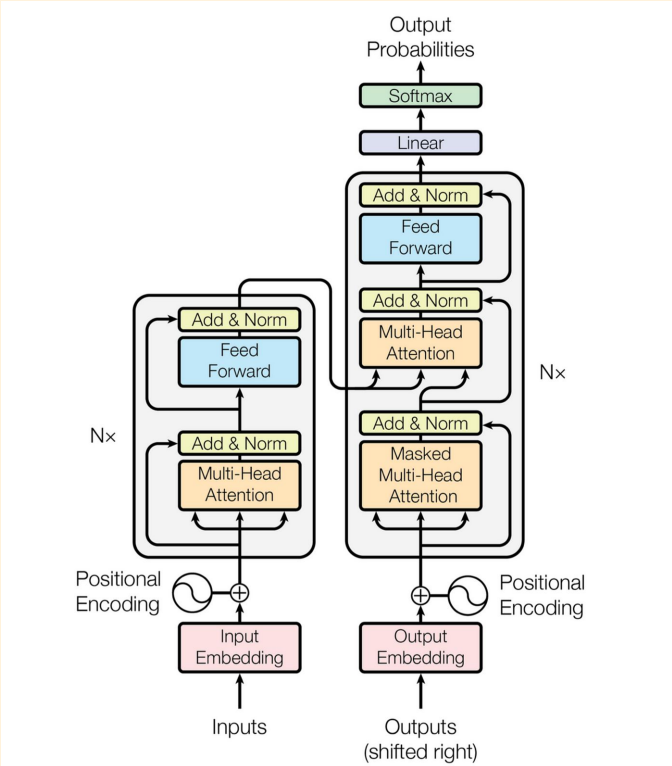


Figure 1: The Transformer - model architecture.

Problem with static word embeddings

They are static! The embedding for a word doesn't reflect how its meaning changes in context.

```
The chicken didn't cross the road because it was too tired
```

What is the meaning represented in the static embedding for "it"?

Contextual Embeddings

- Intuition: a representation of meaning of a word should be different in different contexts!

- Contextual Embedding: each word has a different vector that expresses different meanings depending on the surrounding words

- How to compute contextual embeddings?
  - Attention!!

Contextual Embeddings

```
The chicken didn't cross the road because it
```

What should be the properties of "it"?

```
The chicken didn't cross the road because it was too tired
The chicken didn't cross the road because it was too wide
```

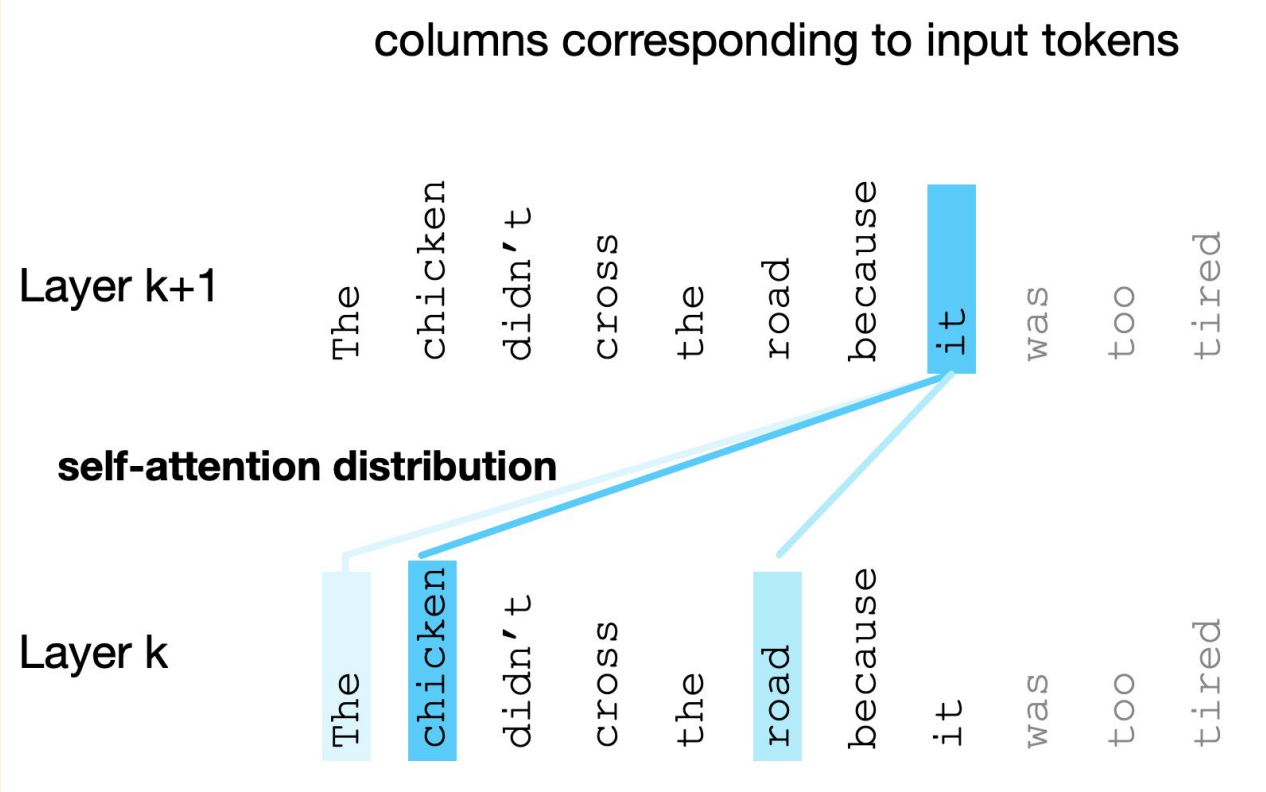
At this point in the sentence, it's probably referring to either the chicken or the street

Intuition of attention

Build up the contextual embedding from a word by selectively integrating information from all the neighboring words

We say that a word "attends to" some neighboring words more than others

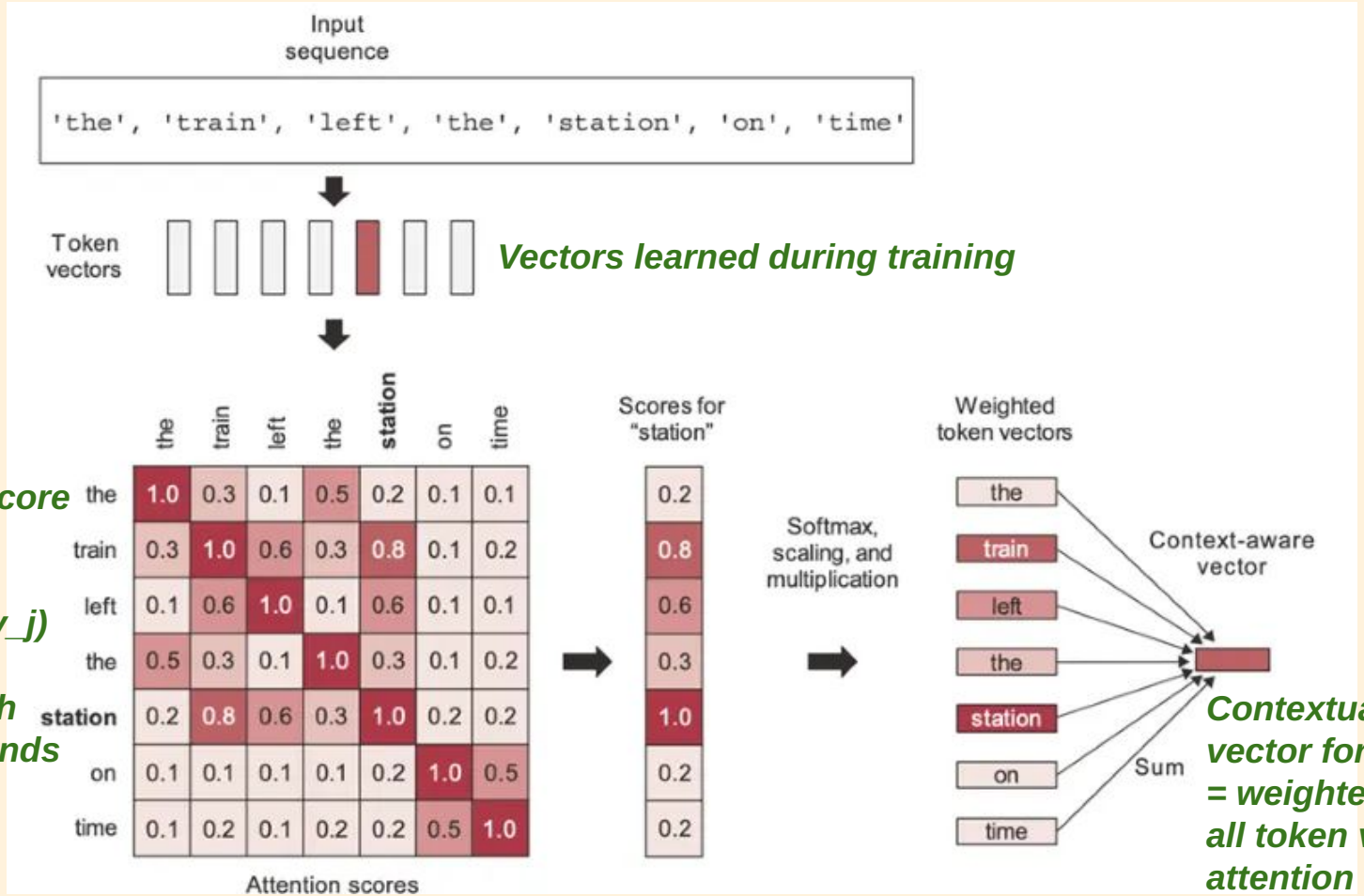
Intuition of attention



So, what is attention?

A mechanism for helping compute the **contextual embedding for a token** by selectively attending to and integrating information from surrounding tokens (at the previous layer).

In practice: computed using a weighted sum of vectors.



Attention score A(i,j)

= dot(v_i, v_j)

= how much token i attends to token j

Contextualized vector for "station" = weighted sum of all token vectors by attention to "station"

Core of the Transformer is Self Attention as in the original paper: "**Attention** is all you need".

arxiv.org/pdf/1706.03762.pdf

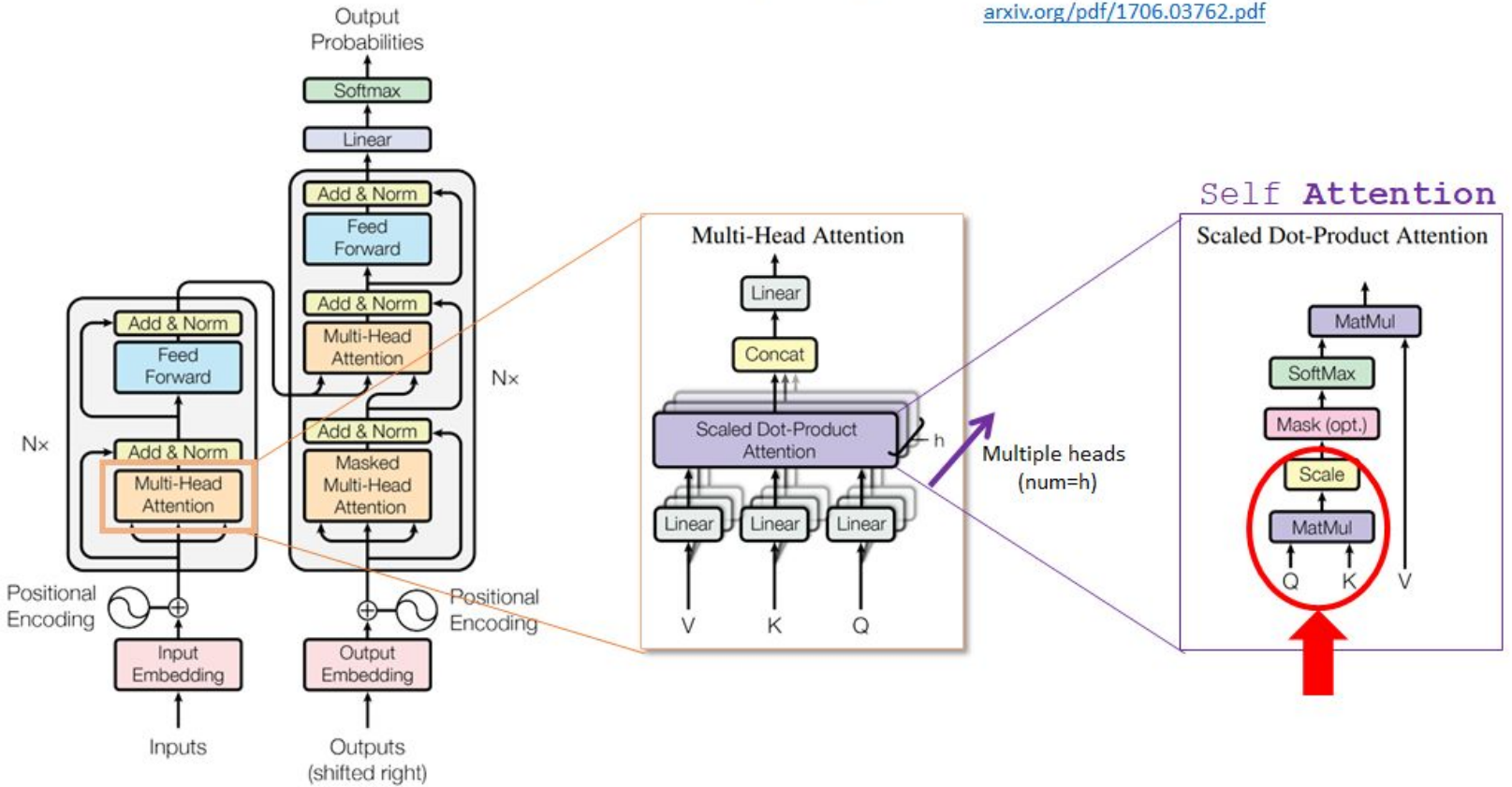


Figure 1: The Transformer - model architecture.

Components of the Transformer

1. Token embedding converts text into continuous vectors
2. Positional encoding introduces sequence order
3. Self-attention layers compute token-to-token relationships
4. **Feed-forward layers transform representations**
5. Stacked blocks enable deep hierarchical features

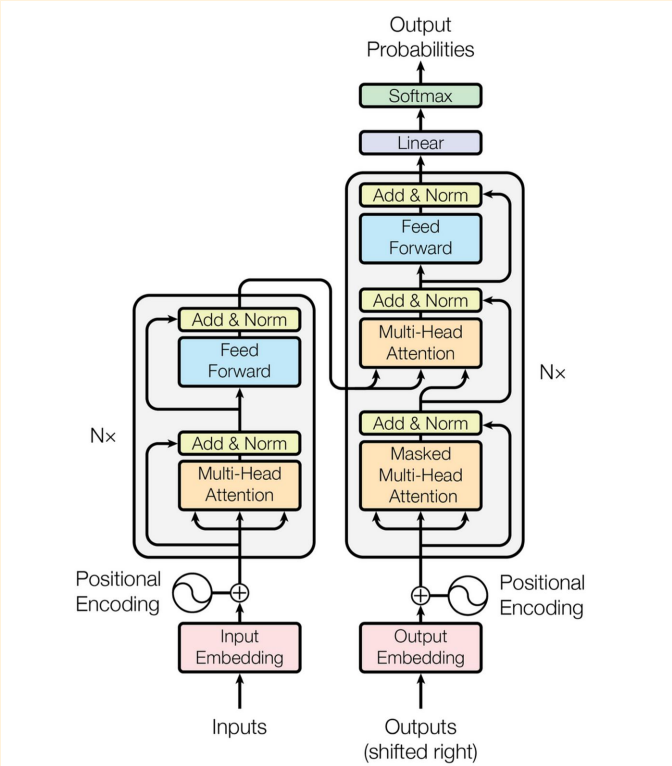


Figure 1: The Transformer - model architecture.

Components of the Transformer

1. Token embedding converts text into continuous vectors
2. Positional encoding introduces sequence order
3. Self-attention layers compute token-to-token relationships
4. Feed-forward layers transform representations
5. **Stacked blocks enable deep hierarchical features**

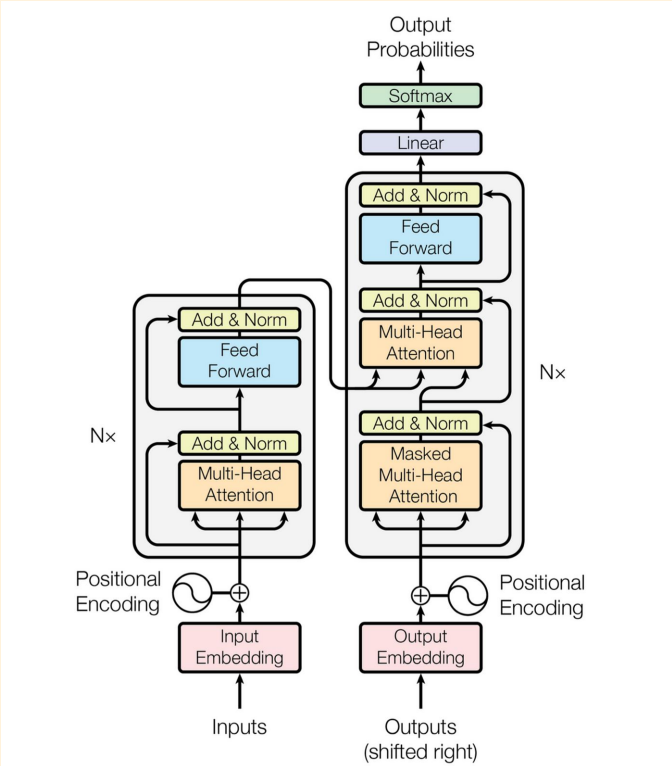


Figure 1: The Transformer - model architecture.

Discussion (10 min)

1. Explain the 5 components of a transformer

2. **[Use AI]** A transformer's *context window* is the number of tokens it can take as input. What happens to the number of attention scores as the context window increases?