Large Language Models

AI 109 Spring 2026

Richard Kelley

The Supervised Learning Workflow

- Start with a **task**
- Collect **examples** of the input to the task.
  - These *are not* labeled yet.
- Use humans to **label** the data.
  - This is often **outsourced** to third parties.
- Use the labeled data set and a training algorithm to **train** a model.
- **Deploy** the model to customers.
- Profit.

Self-Supervised Learning

- We have seen that LLMs try to solve task of **next-token prediction**.
  - “In the beginning was the Word”
  - [(“In”, “the”), (“the”, “beginning”), (“beginning”, “was”), (“was”, “the”), (“the”, “Word”)]
- Whenever the labels are built automatically from a data set, that’s called **self-supervised learning**.
- We turn a hard problem (understanding human language) into an easy one (predict the next word).
- To understand modern AI, we have to understand supervised learning.

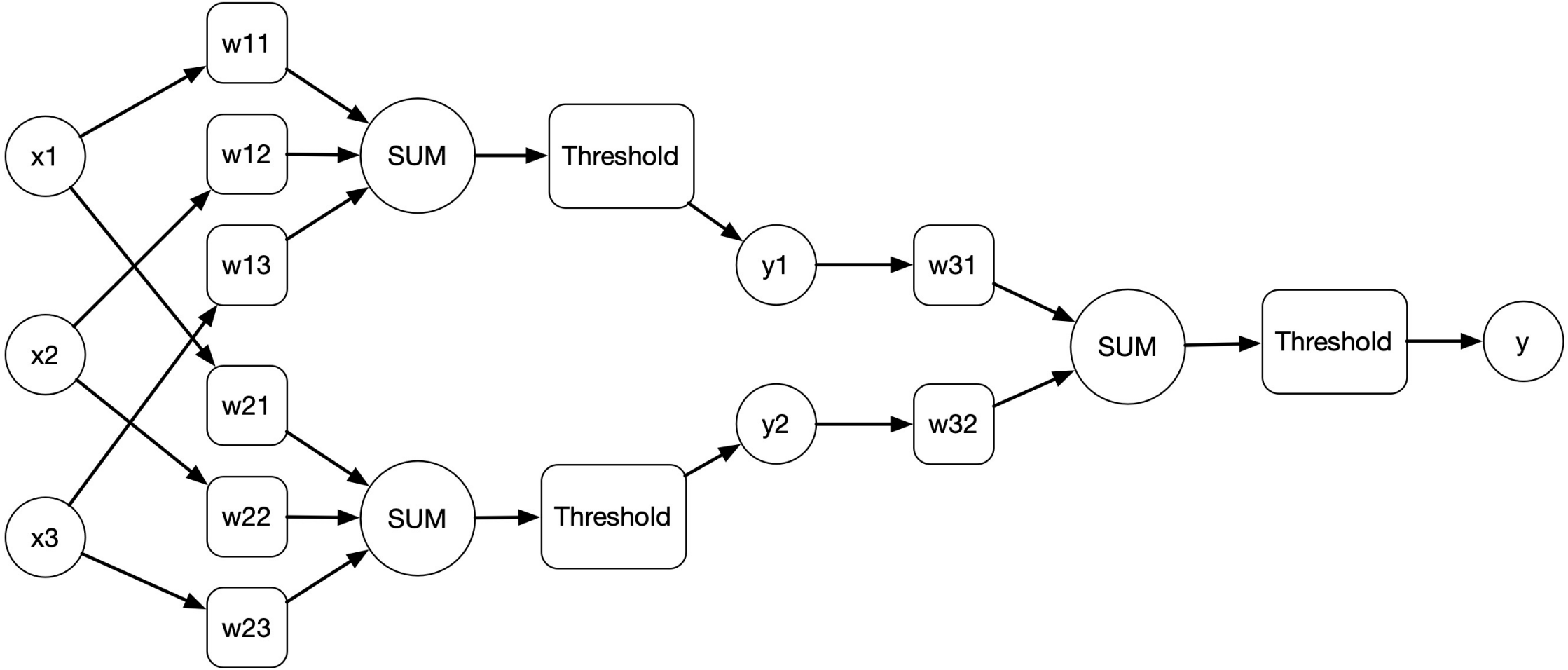
Self-Supervised Learning

- **Self-supervised learning** starts with unlabeled data, so it resembles **unsupervised learning**.
- But it creates prediction targets from the data itself, so it also resembles **supervised learning**.
- It is a bridge between the two:
  - no human-provided labels are needed
  - the system still learns by solving a prediction problem
- This idea is central to many modern AI systems, including large language models.

Neural Networks

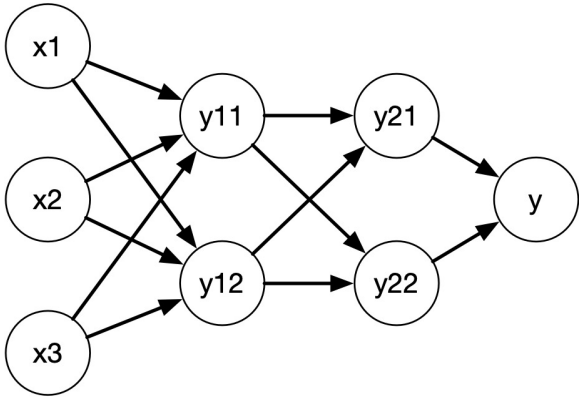
- A modern **neural network** is built by stacking many **linear layers** and **nonlinear activation functions**.
- The linear layers transform the representation from one stage to the next.
- The nonlinearities prevent the whole model from collapsing into one linear transformation.
- Modern AI models often use substantial depth, meaning many such layers are stacked together.
- The linear layers contain adjustable **parameters** such as weights and biases.
- Those parameters are the quantities that are **learned** from experience during training.

Neural Networks

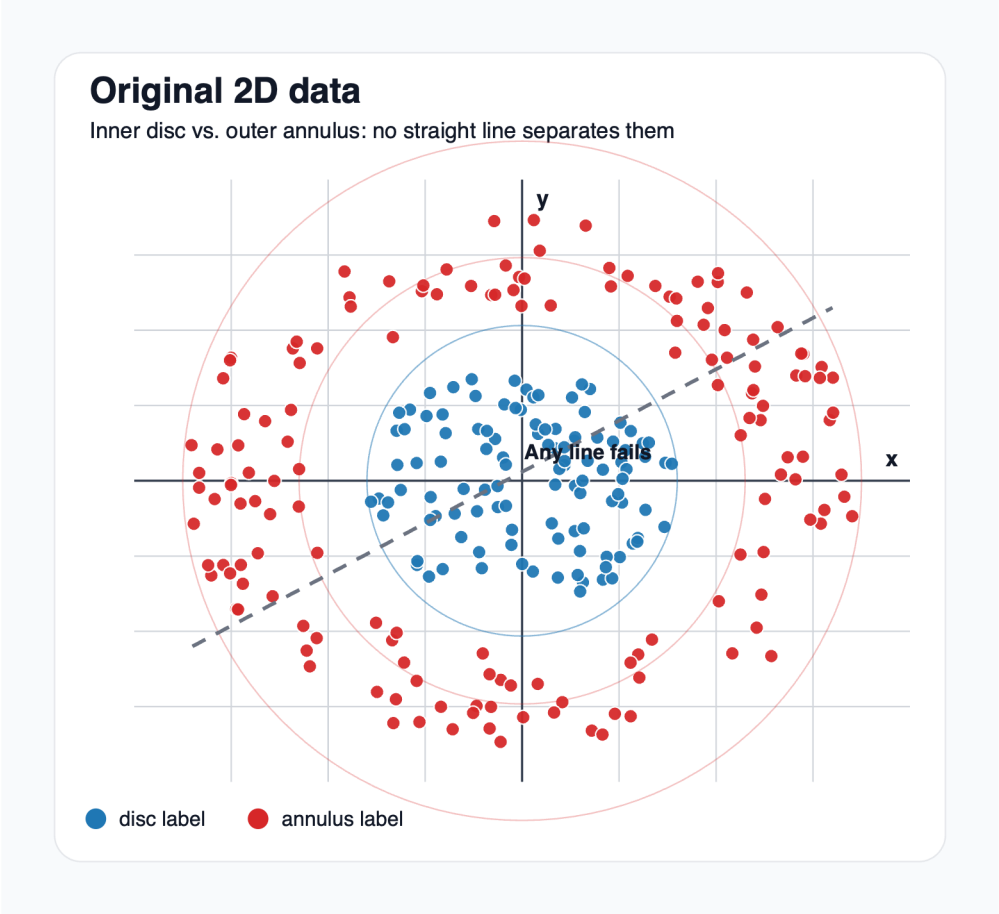


Deep Learning

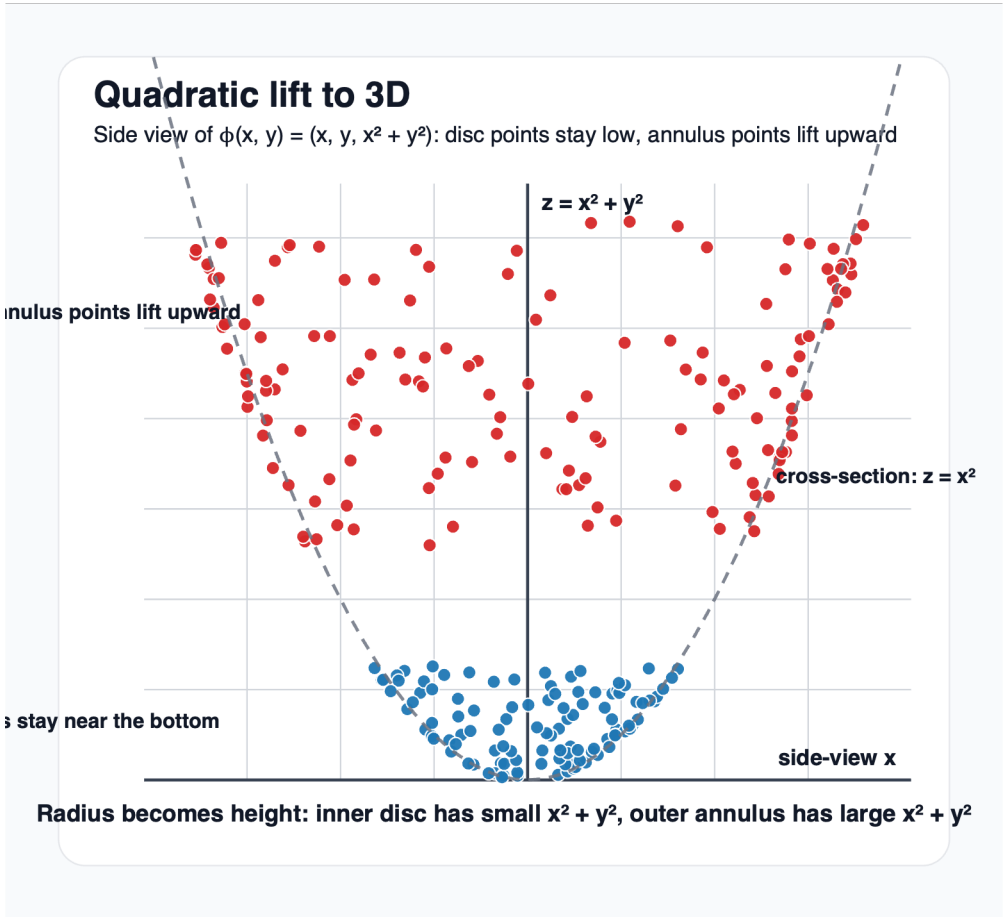
- **Deep learning** refers to neural networks with many layers.
- The word **deep** refers to the depth of the model, meaning how many transformations are stacked on top of one another.
- A deeper network can build more complex representations by composing many simpler steps.
- The hidden layers can be thought of as **intermediate representations** that are more useful for the ultimate goal.
  - So deep learning is a process of finding better representations of data for solving problems.



Learning Representations



Learning Representations



Neural Networks as Building Blocks of LLMs

- You've almost seen everything you need to understand how AI works in systems like ChatGPT and Claude.
- There are a few more new ideas that we'll cover this week.
- Our goal is to understand how neural networks are the building blocks that make up modern LLMs.
  - Amazingly, there are only a few really new ideas in the AI that works – a lot of the success we've seen since 2020 has been due to **scale**.

Training vs Inference

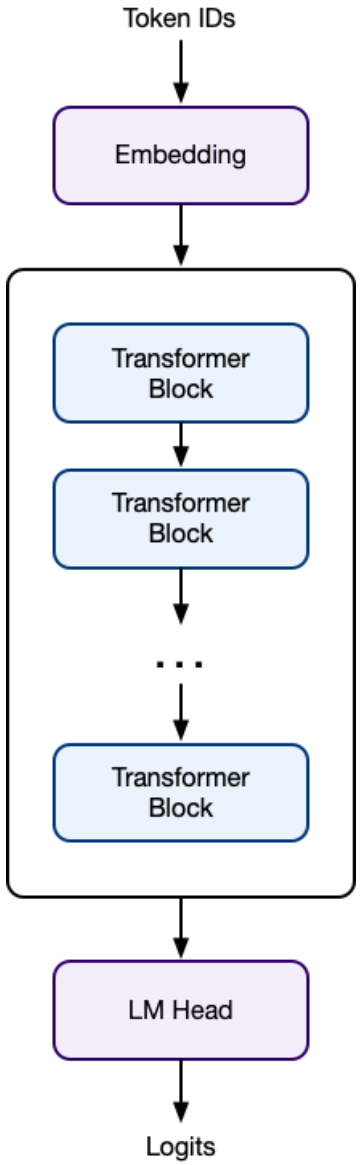
- **Training** updates model parameters by processing many examples and computing updates that are added to the parameters.
  - These updates are called **gradients**.
  - Training an LLM is exactly like training any other model for supervised learning.
- **Inference** keeps the parameters fixed and computes predictions for one request or one batch of requests.
- We'll talk about both today.

Neural Network Architectures

- In deep learning, you have to make choices about how you combine the linear layers and activation functions.
  - How big do I make my linear layers?
  - What kind of activation should I use?
  - How many parameters should I try to have?
- The precise choices you make lead to different neural network **architectures**.
- Successful architectures are often given names.
  - Not just for scientific clarity, but also for marketing purposes.

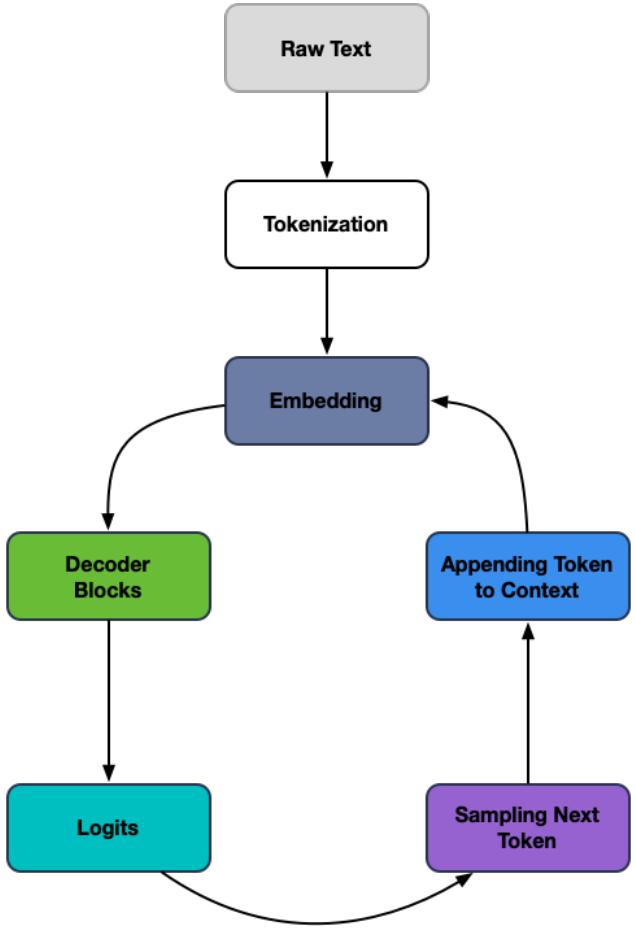
Transformers

- The **transformer** is the dominant architecture for most large language models used for text generation.
  - First developed in 2017.
  - Originally designed for translating between natural languages.
- The model reads the existing token sequence and predicts the next token, then repeats that operation **autoregressively**.
- **autoregressive**: predicts future values based on combinations of past values.



Pipeline Overview

- A request begins as raw text, which is converted into **tokens** before the model sees it.
- Those tokens pass through embeddings and repeated transformer blocks to produce scores over the vocabulary.
- A decoding rule selects the next token, appends it to the context, and repeats the process.
- The loop ends when a stopping condition is met, such as an end token or a length limit.

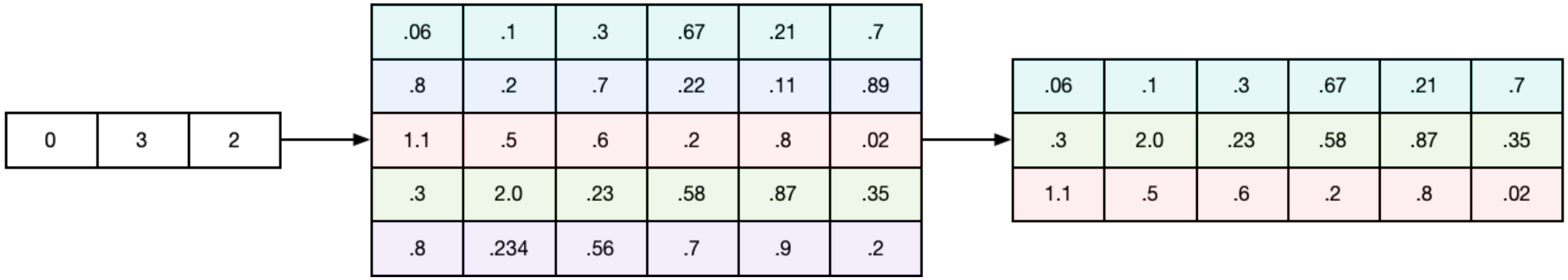


Raw Text to Tokens

- Models do not read characters or words directly; they consume discrete integer identifiers called tokens.
- A **tokenization** scheme breaks text into pieces that are practical for the model's vocabulary and training data.
  - Common tokens can correspond to whole words, subwords, punctuation marks, or whitespace patterns.
  - The string `unbelievable!` might be represented as one token in one tokenizer and several pieces in another.
  - When you type "hello, world" to ChatGPT, it actually sees something like [45, 7, 12, 51]

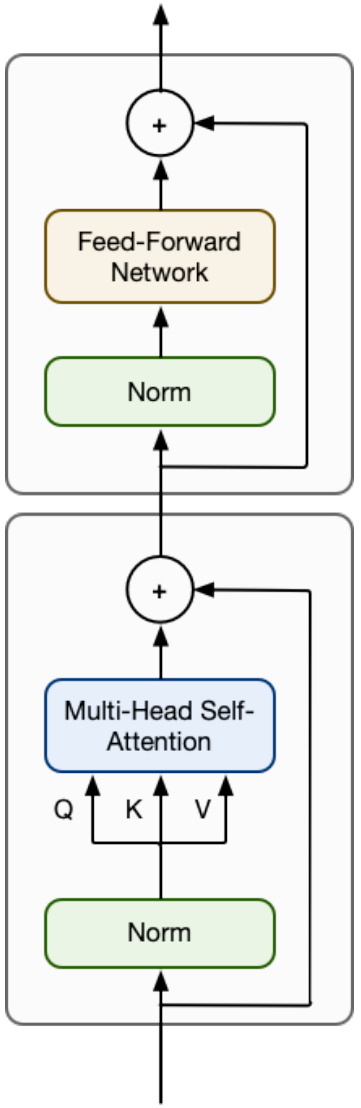
Embeddings for Transformers

- Each token id is mapped to a learned vector called an **embedding**, which gives the model a numerical representation of the token.
- The result is a sequence of vectors, one per token position, ready to enter the first transformer block.



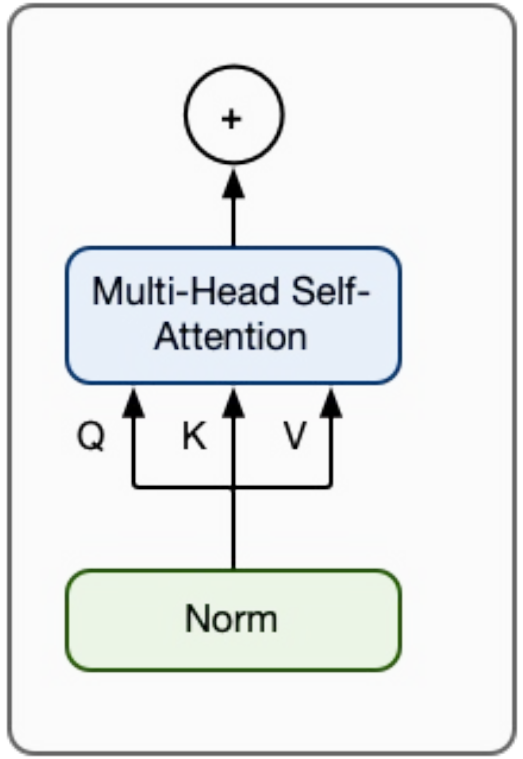
Repeated Blocks

- A transformer is built from a stack of repeated blocks rather than one giant monolithic computation.
- Each block applies the same high-level pattern:
  - attention-style mixing across positions,
  - followed by a feed-forward transformation.
    - **Feed-forward network** is another name for multi-layer perceptron.



Attention Purpose

- After embedding, the first major step inside a transformer block is the attention module.
- **Attention** lets each token position combine information from other positions in the existing context.
- This is the mechanism that allows later tokens to depend on earlier instructions, names, variables, or examples.
- Without attention, the model would struggle to understand **sequences**.

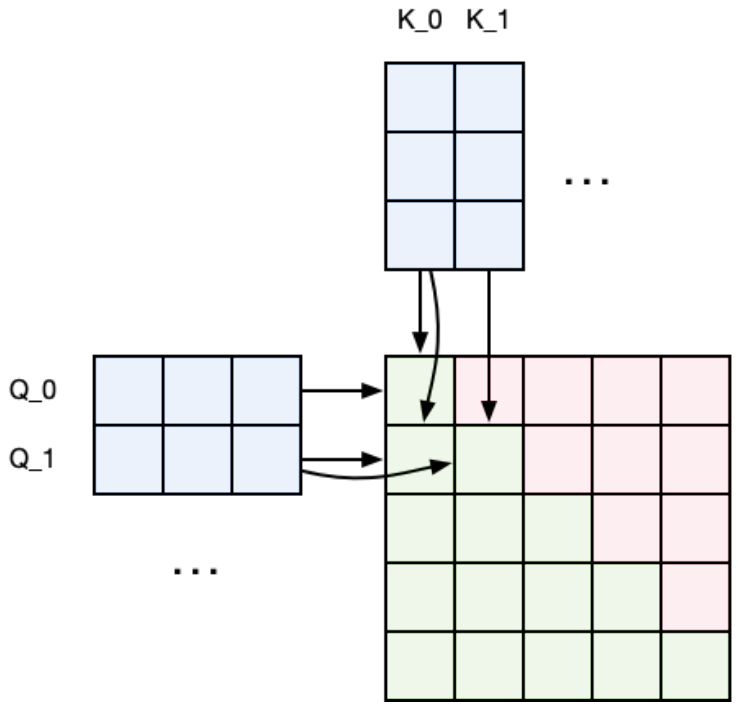


Attention Inputs

- At a high level, each position produces **query**, **key**, and **value** vectors from its current representation.
  - Query vectors ask which prior positions matter.
  - Key vectors describe what each position offers.
  - Value vectors carry content to be aggregated.
- The resulting weighted combination becomes part of the new representation for that position.
- This is the key mechanism that makes LLMs work!

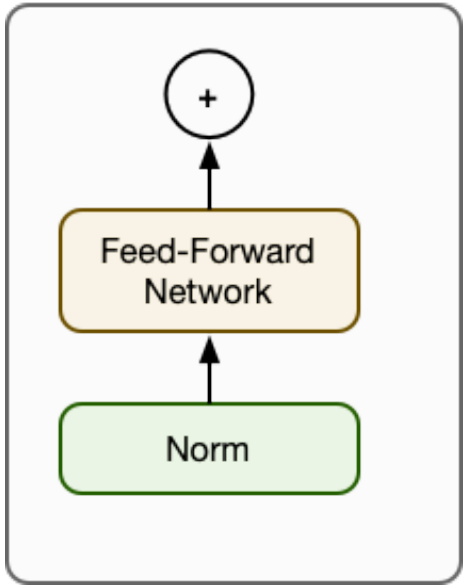
Masked Attention

- In a decoder-only model, generation uses **causal** or **masked self-attention**.
- Each of the squares is a **dot product.**
- A token at position `i` may depend on positions `1` through `i`, but not on future positions that have not been generated yet.
- This masking preserves the next-token prediction structure required for autoregressive generation.



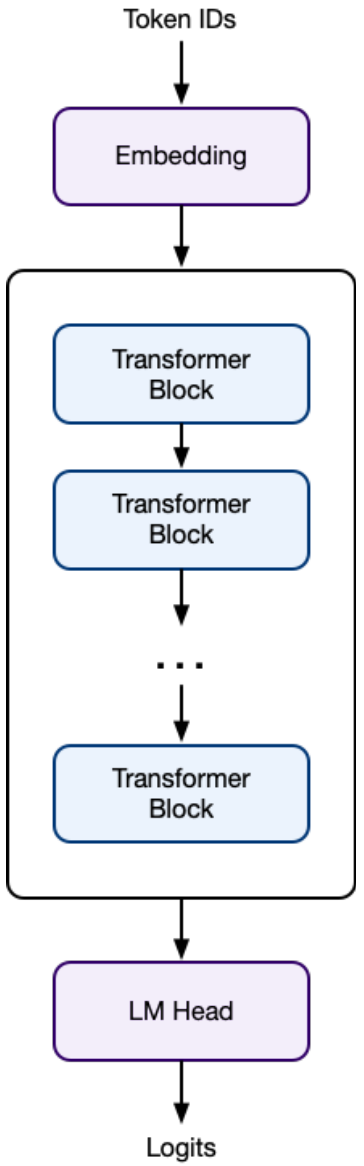
Feedforward Layers

- A **feedforward network** acts independently at each position after attention has mixed information across positions.
  - Its job is to transform the position representation through learned nonlinear mappings, not to communicate across time steps.
  - Attention mixes information across positions; feedforward layers enrich the representation at each position.



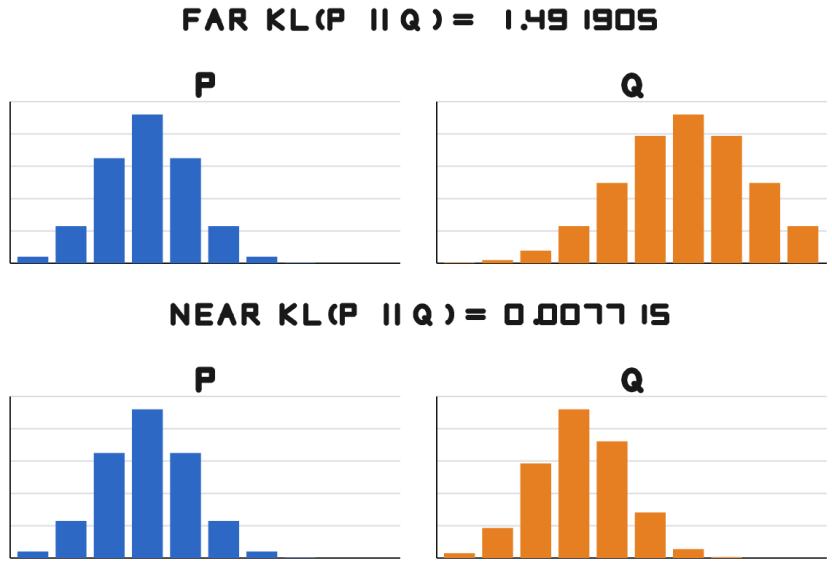
Logits and Next-Token Prediction

- After the final block, the model projects the last-position representation into one score per vocabulary item.
  - These raw scores are called **logits**.
- A bigger logit means the model currently considers that token more compatible with the observed context.
- The model does not generate an entire sentence in one shot; it generates one next-token distribution, selects a token, appends it, and then solves a slightly larger next-token problem.



Training an LLM

- The task is next-token prediction.
- The model produces a probability for every possible next token in the sequence it sees.
  - This looks like a **histogram**.
- The training process compares the histogram of probabilities guessed by the model vs the actual histogram of next-token probability to calculate a loss.
  - This loss signal is used to update the model parameters.
  - The technical term for the difference between the two histograms is the **Kullback-Leibler Divergence** (KL divergence for short).



Decoding

- **Decoding** is the process that begins once generation starts and the model emits tokens one at a time.
- Each decode step depends on the token chosen at the previous step, so the process is *inherently sequential*.

Greedy Decoding

- **Greedy decoding** chooses the highest-scoring token at each step.
- It is simple and deterministic once the logits are fixed.
- Greedy decoding can be fast to describe algorithmically, but it may produce repetitive or low-diversity outputs.

Sampling Choices

- **Sampling-based decoding** draws from a controlled distribution instead of always taking the top-scoring token.
- Different strategies can change which candidate tokens remain plausible at each step.
  - The same prompt can produce different outputs when **stochastic** decoding is used.

Stopping Rules

- Generation stops when the model emits a designated end token, reaches a maximum length, or hits an application-specific stopping rule.
- The runtime cost of one request therefore depends on both the input length and the generated output length.
- End-to-end analysis must account for both quantities.

Questions for Next Time

- What kind of data do we use to training a transformer-based LLM?
  - Where does it come from?
  - How much do we really need?
- What if we want our LLM to be good at a particular task?
- If we train our LLM on the internet, how can we make sure it doesn’t sound like Reddit or 4chan?