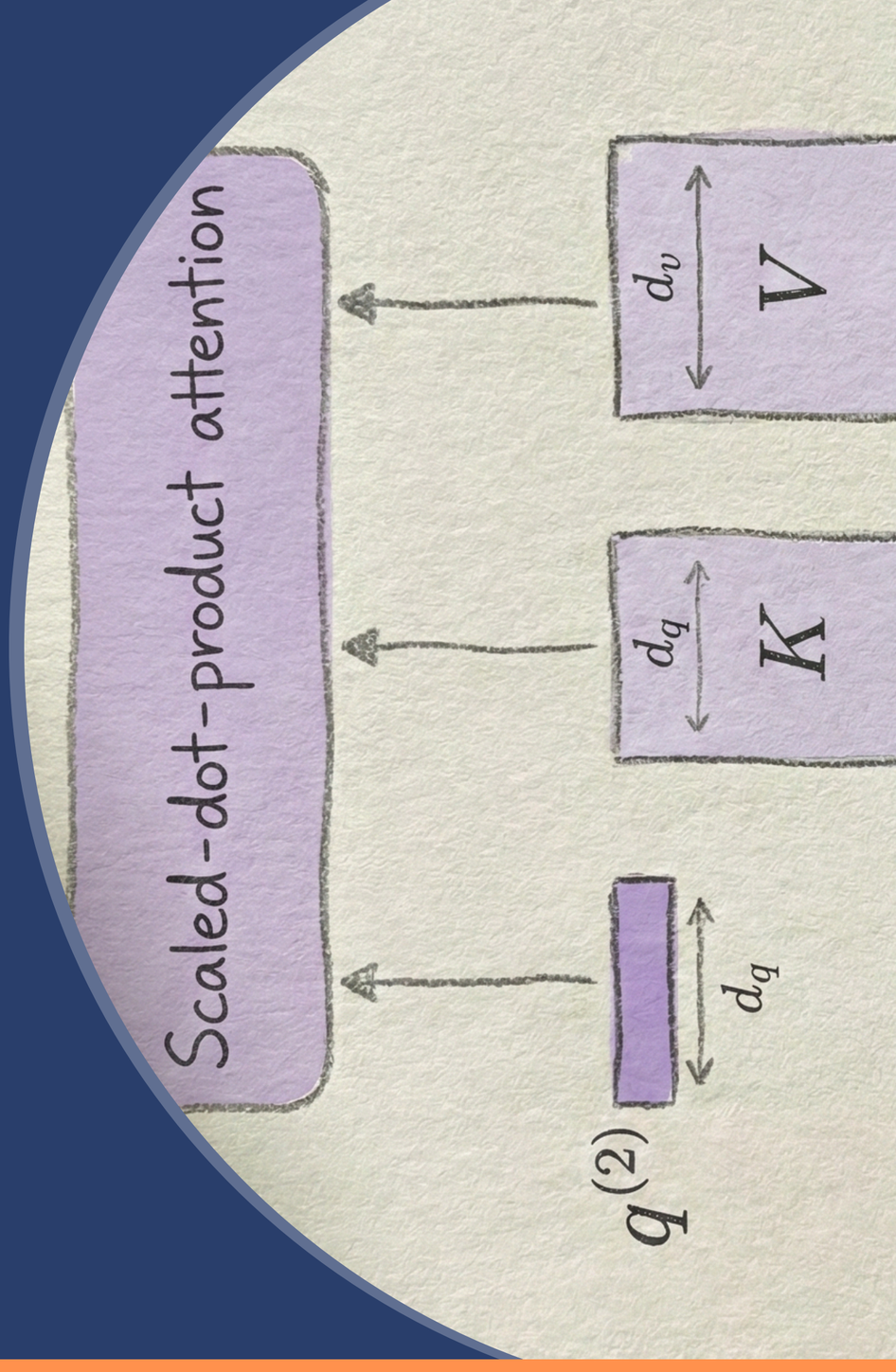
Free Webinar

Deep Dive into Legendary Self-Attention Mechanism

Unlocking the Secret Sauce of Large Language Models

Lhuqita Fazry
Founder Rumah Coding
Lhuqita Fazry



Lhuqita Fazry, S.Si., M.Kom.
(Founder Rumah Coding)

Pendidikan:

- S1: Math, Universitas Indonesia
- S2: Ilmu Komputer, Universitas Indonesia

Pengalaman Riset:

- Intelligent Robots and Systems (IRoS) Lab, Faculty of Computer Science, Universitas Indonesia
- Human-AI Interaction Laboratory, Japan Advance Institute of Technology (JAIST)

Lhuqita Fazry

Rumah Coding

HifiDiff: Two Stream Diffusion Models for High Fidelity Speech Generation of Unseen Languages 28th International Conference of Oriental COCOSDA

Lhuqita Fazry, Kurniawati Azizah, Dipta Tanaya, Ayu Purwarianti, Dessi Lestari and Sakriani Sakti

Hybrid Wavelet-Attention Model for Detecting Changes in High-Resolution Remote Sensing Images IEEE Access

Lhuqita Fazry, Mgs M Luthfi Ramadhan, Alif Wicaksana Ramadhan, Muhammad Febrian Rachmadi, Aprinaldi Jasa Mantau, Lukito Edi Nugroho, Chi-Hung Chi, Wisnu Jatmiko

Change Detection of High-resolution Remote Sensing Images Through Adaptive Focal Modulation on Hierarchical Feature Maps IEEE Access

Lhuqita Fazry, Mgs M Luthfi Ramadhan, Wisnu Jatmiko

Hierarchical Vision Transformers for Cardiac Ejection Fraction Estimation
7th International Workshop on Big Data and Information Security (IWBIS)

Lhuqita Fazry, Asep Haryono, Nuzulul Khairu Nissa, Sunarno, Naufal Muhammad Hirzi, Muhammad Febrian Rachmadi, Wisnu Jatmiko

A Split-then-Join Approach to Abstractive Summarization for Very Long Documents in a Low Resource Setting
9th International Conference of Computer and Informatics Engineering

Lhuqita Fazry, Evi Yulianti

Improving Remote Sensing Change Detection via Locality Induction on Feed-forward Vision Transformer Journal of Computer Science and Information

Lhuqita Fazry, Mgs M Luthfi Ramadhan, Wisnu Jatmiko

Unsupervised Raindrop Removal from a Single Image using Conditional Diffusion Models arXiv preprint arXiv:2505.08190

Lhuqita Fazry, Valentino Vito, Laksmita Rahadianti, Aniati Murni Arymurthy

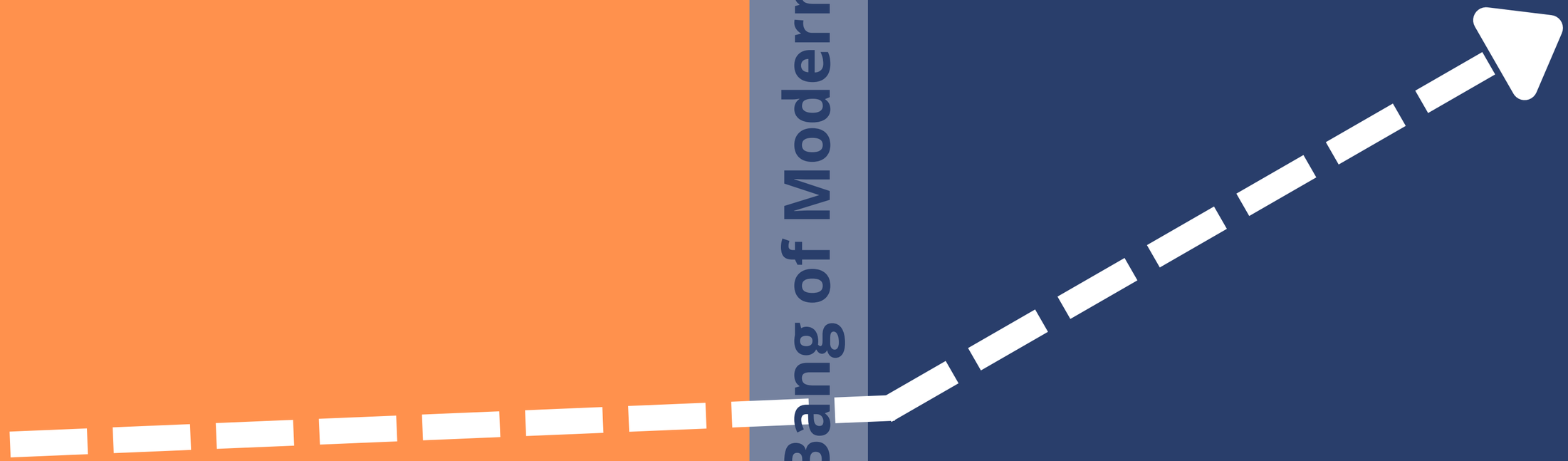
Contents

Paper that Changed the World

Era RNN/LSTM

Era GPT/LLM

Big Bang of Modern AI



Sequential, Long-term dependency issue

2017

"Legendary"
(foundation of modern generative AI)

*) Scalability impact, in terms of parallel computation power and model size (parameter count)

The Paper That Changed Everything

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

- Paper by Google Brain
- Remove Recurrence (RNN), **use only** attention.
- **200k+** citations

NIPS papers

https://papers.neurips.cc › paper › 7181-attention... PDF

Attention is All you Need

by A Vaswani · Cited by 207552 — We propose **a** new simple

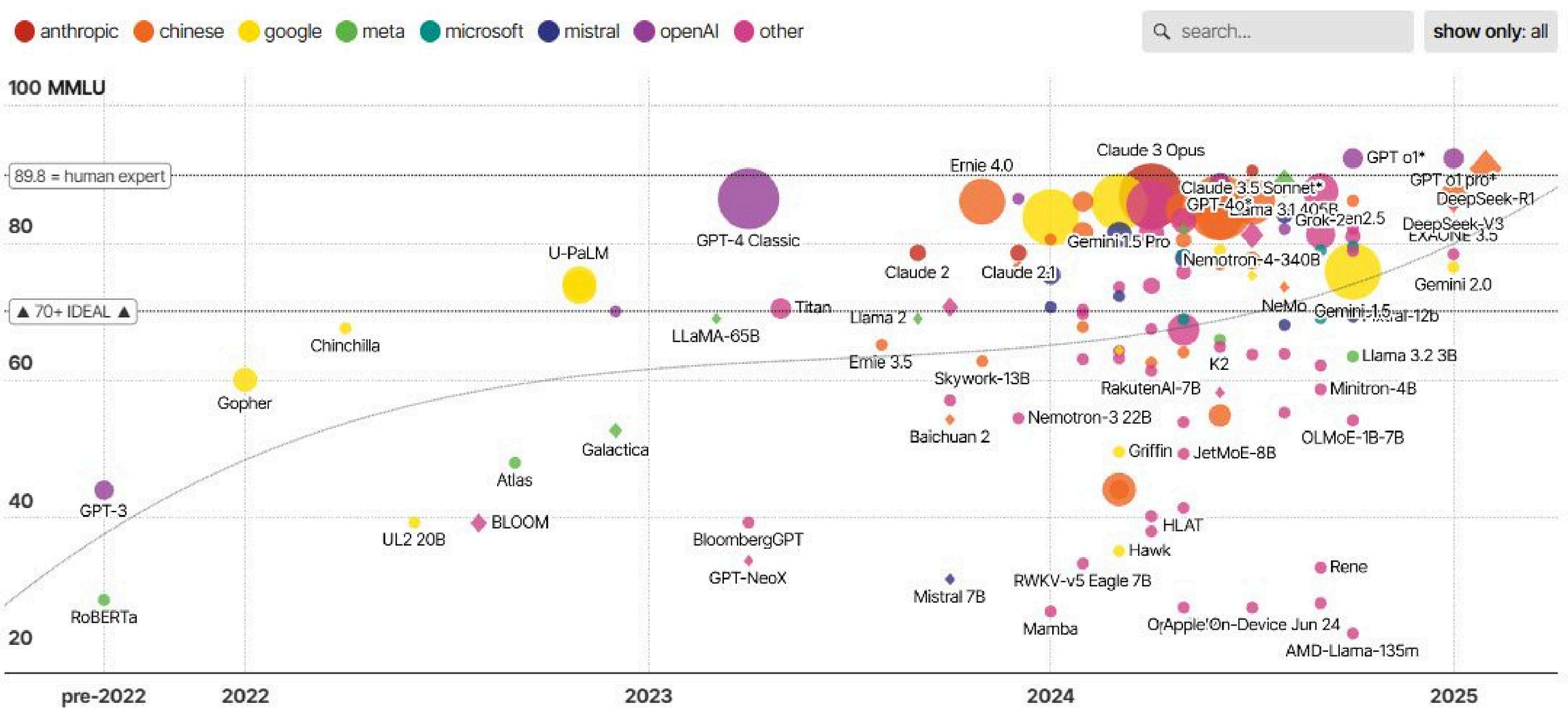
based solely on **attention** mechanisms, dispensing with recur

11 pages

Major Large Language Models (LLMs)

ranked by capabilities, sized by billion parameters used for training

CLICK LEGEND ITEMS TO FILTER



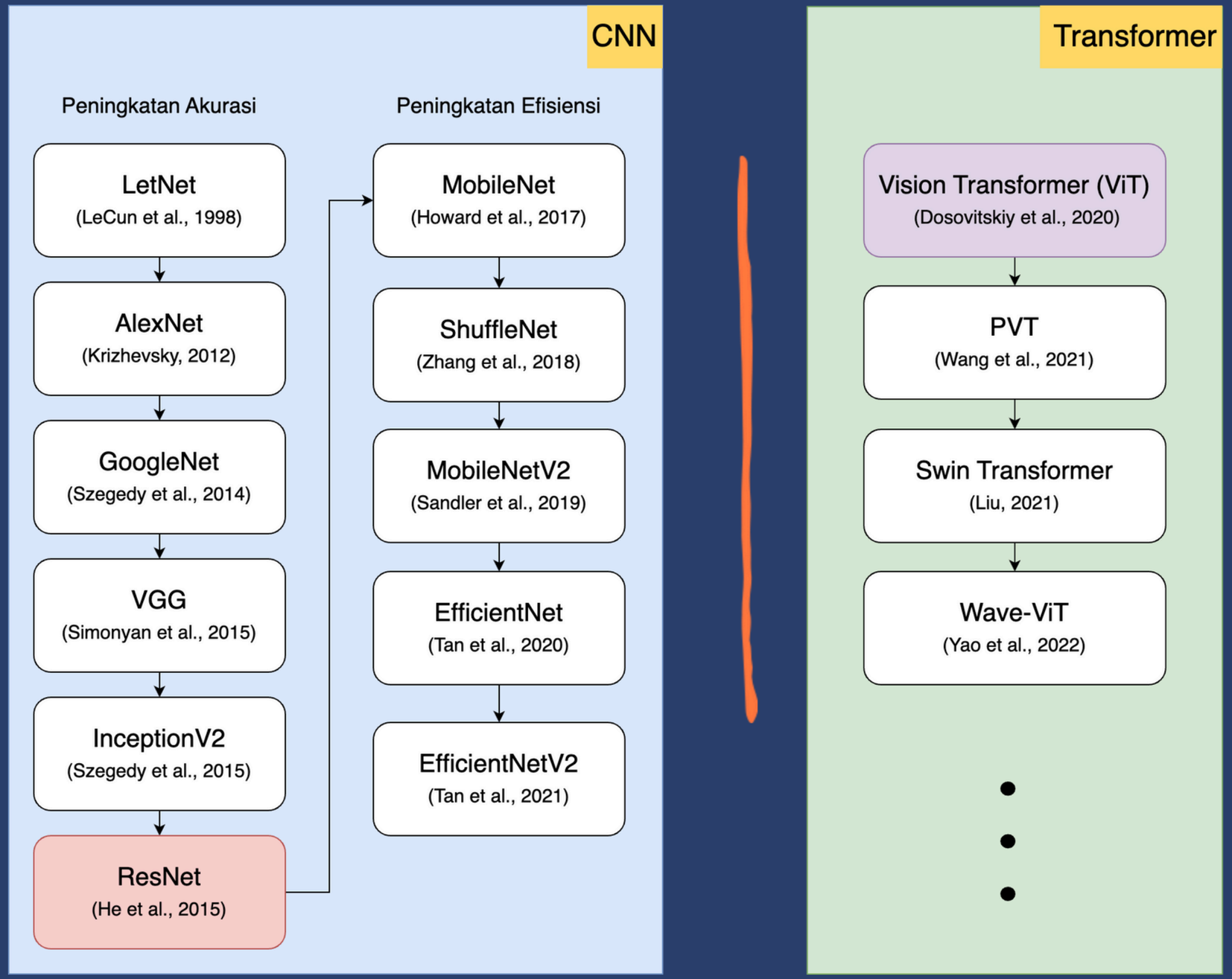
David McCandless, Tom Evans, Paul Barton

Informationisbeautiful // Jan 2024

MMLU = benchmark for measuring LLM capabilities

* = parameters undisclosed // source: LifeArchitect // data

https://informationisbeautiful.net/visualizations/the-rise-of-generative-ai-large-language-models-llms-like-chatgpt/



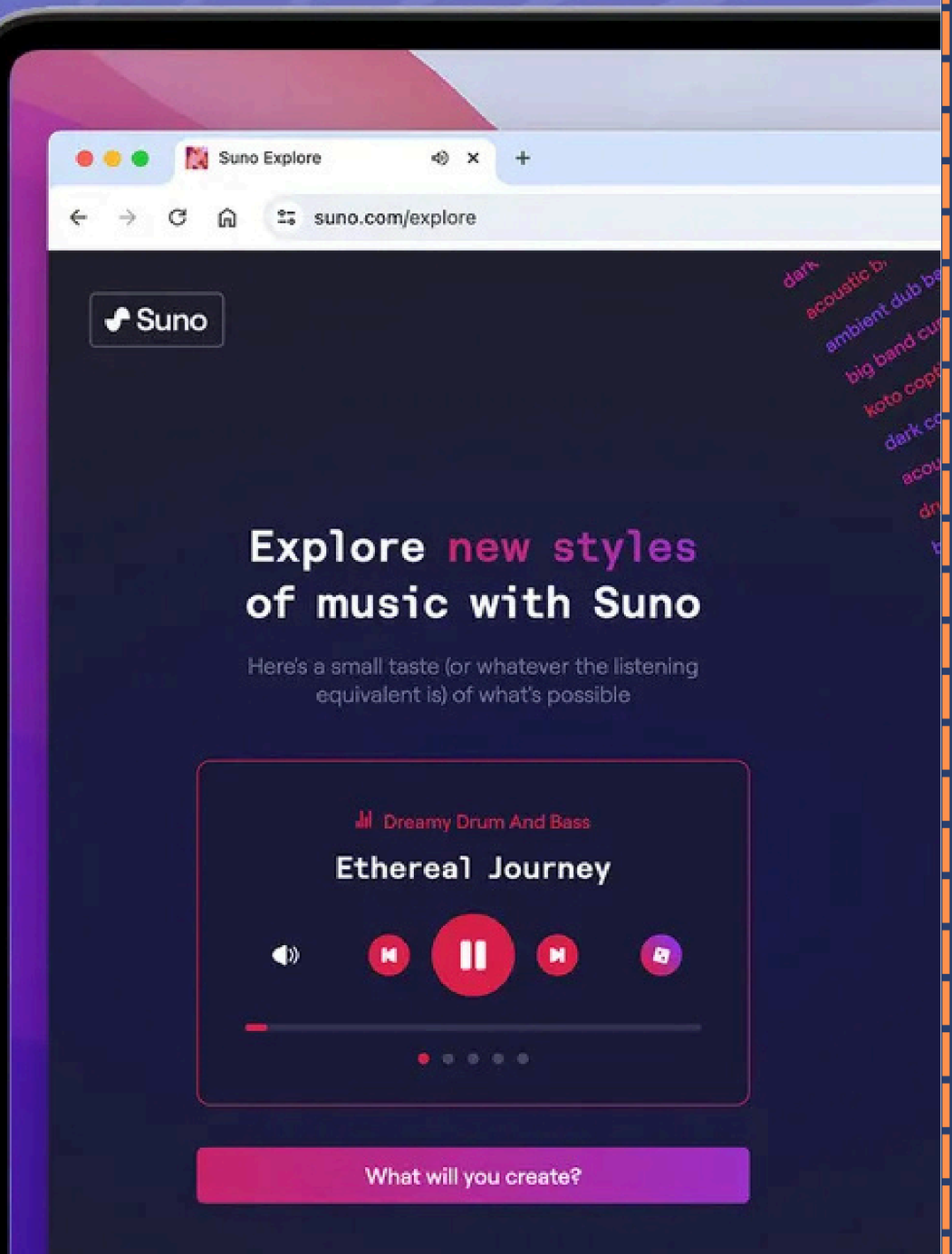
Computer Vision

AlphaFold 3

Biology & Healthcare (Protein Folding & Genomics)

One of AI's greatest scientific achievements.

Self-attention is used to predict 3D protein structures from 1D amino acid sequences.

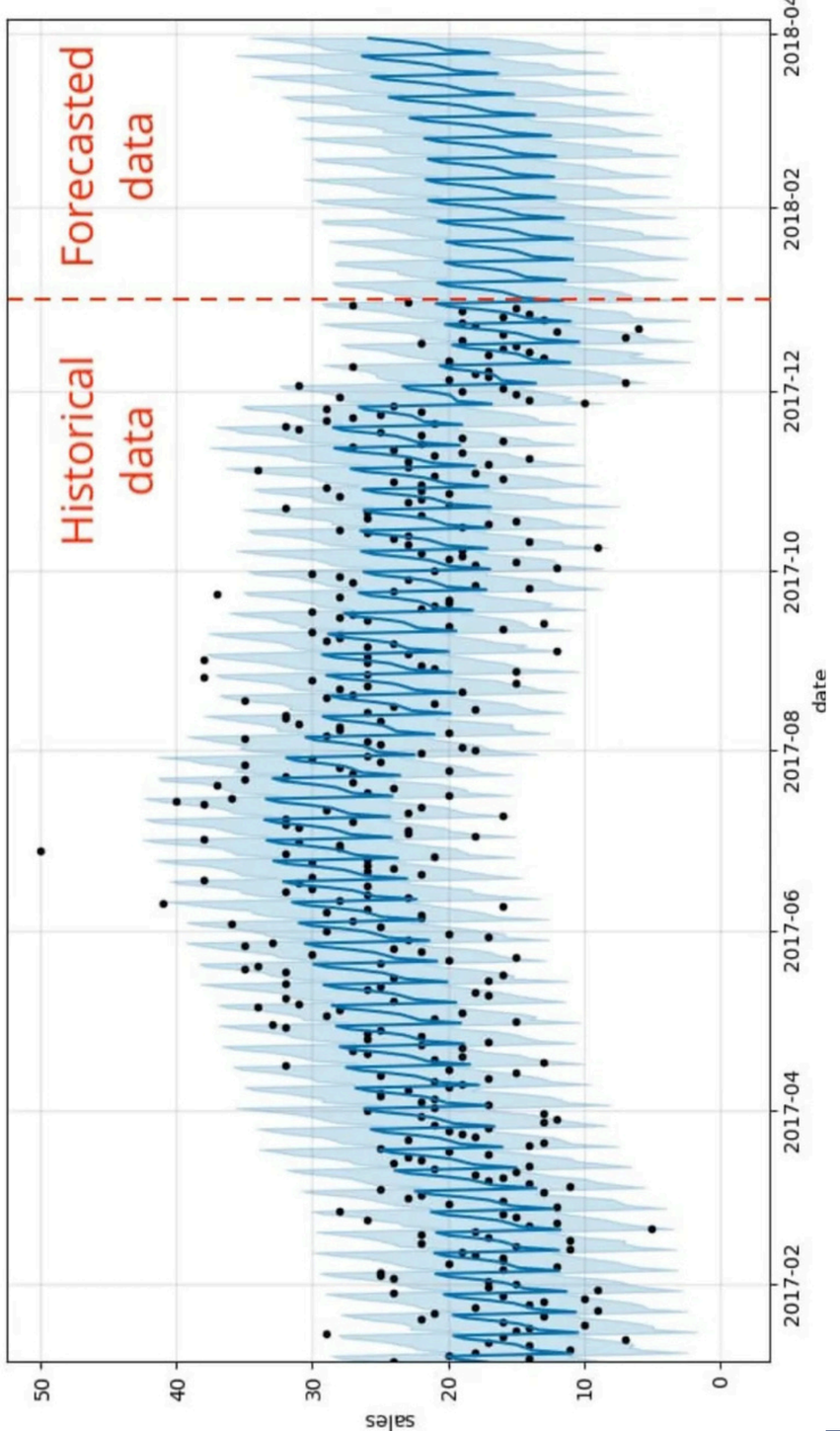


Audio & Speech Processing

Speech Recognition (ASR): Models like OpenAI's Whisper use Transformers to transcribe **speech into text** with unprecedented accuracy.

Music Generation: Models like Suno AI or MusicLM (Google) use self-attention to generate coherent music.

https://www.techradar.com/computing/artificial-intelligence/what-is-suno-ai



Time Series Analysis (Forecasting)

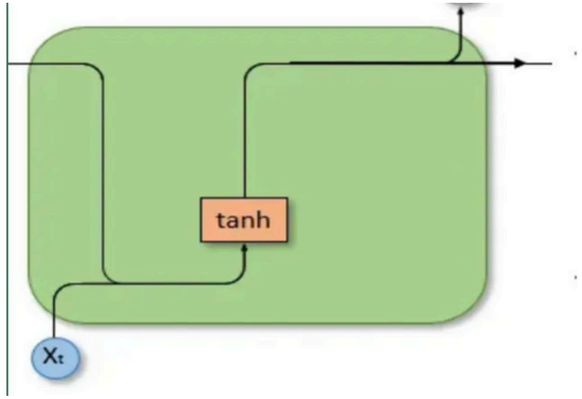
Finance & Weather: Used for predicting stock prices, electricity consumption, or weather patterns. Transformers (such as Temporal Fusion Transformers) excel at capturing long-range seasonal patterns and complex dependencies that traditional methods like ARIMA or LSTM often miss.

https://medium.com/analysts-corner/what-is-time-series-forecasting-in-context-of-business-analytics-c143b1885d3a

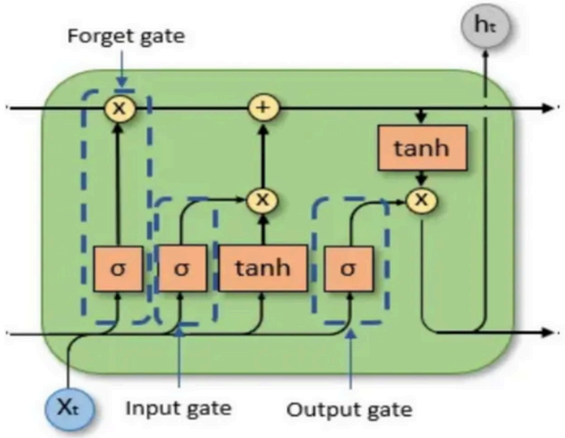
The Era of Recurrence & The Sequential Trap

RNN Family

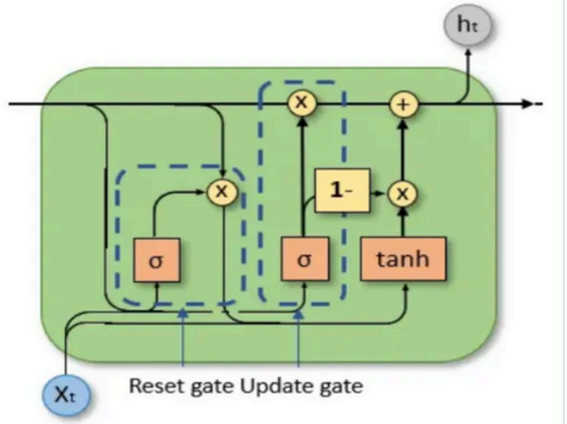
RNN



LSTM



GRU



Recurrent Neural Network (RNN)

- A neural network with a "Memory Loop".
- It processes the current word (x_t) and the memory from the previous step (h_{t-1}).
- Vanishing Gradient Problem.

Long Short-Term Memory (LSTM)

- Designed explicitly to solve the Vanishing Gradient problem using gating mechanism.
- **Forget Gate**: Decides what information to throw away.
- **Input Gate**: Decides what new information to store.
- **Output Gate**: Decides what to output based on the memory.

Gated Recurrent Unit (GRU)

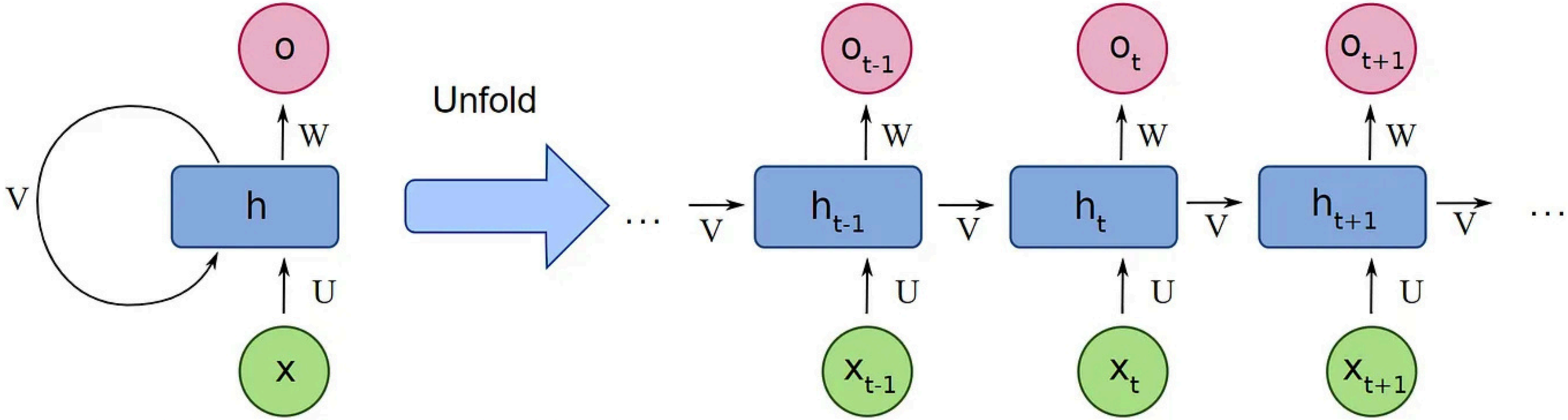
- A simplified version of LSTM.
- Fewer parameters to train = Faster Training.
- Less memory consumption.
- Often achieves similar performance to LSTM on smaller datasets.

https://python.plainenglish.io/introducing-gru-rnn-and-lstm-a-beginners-guide-to-understanding-these-revolutionary-deep-35b509a34a5a

The Problem #1: Sequential Computation

- Why Recurrence? Because language is a stream, not a snapshot
- Great for short sequences, but flawed for scale.
- The current step (t) must wait for the previous step (t-1) to complete.
- Impossible to parallelize, so the training is incredibly slow and inefficient.

$$h_t = f(h_{t-1}, x_t)$$



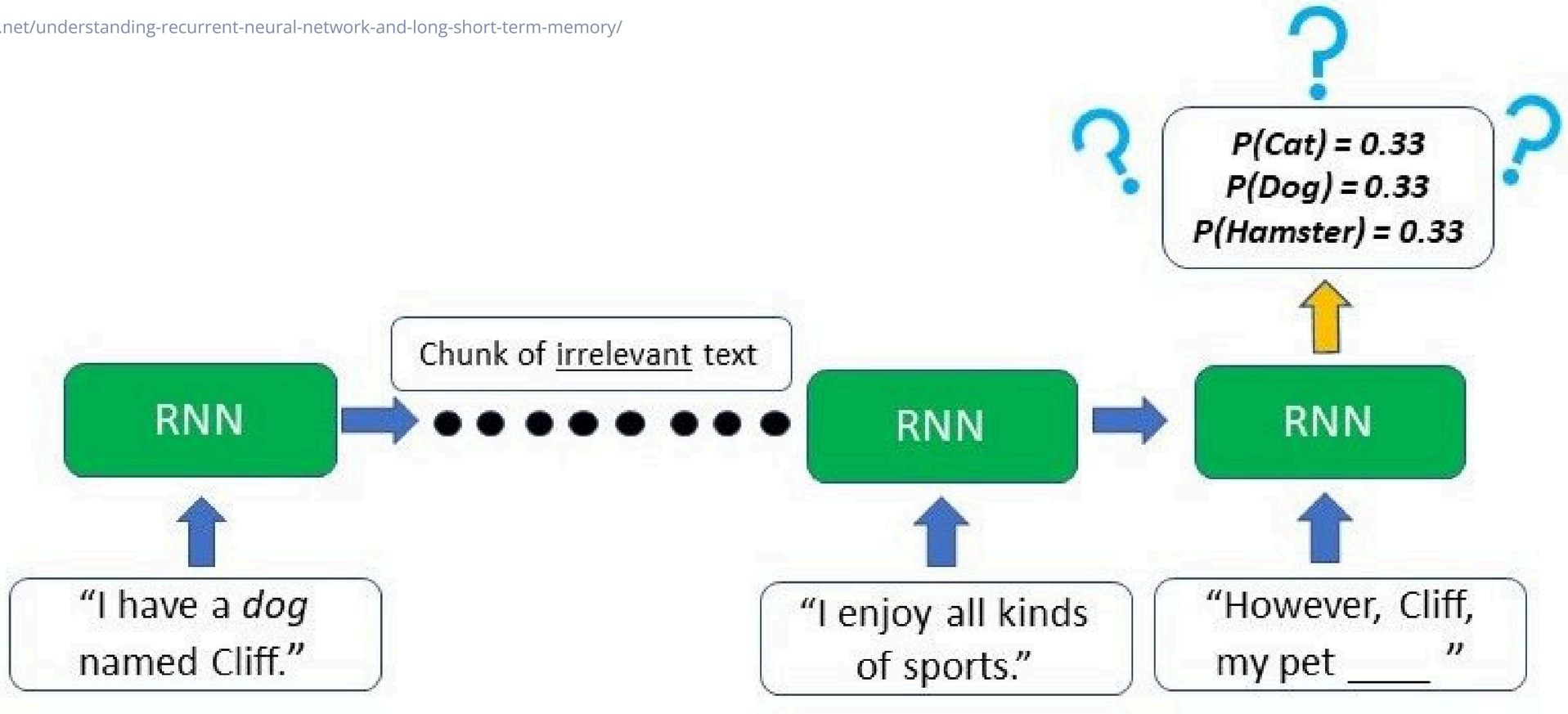
https://medium.com/deeplearningbrasilia/deep-learning-recurrent-neural-networks-f9482a24d010

If you look at a photograph (CNN), you see the whole image at once. But if you read a sentence (RNN), you read word by word. To understand the end of the sentence, you must remember the beginning. Recurrence is simply the mathematical way of simulating that memory.

The Problem #2: Long-term Dependency

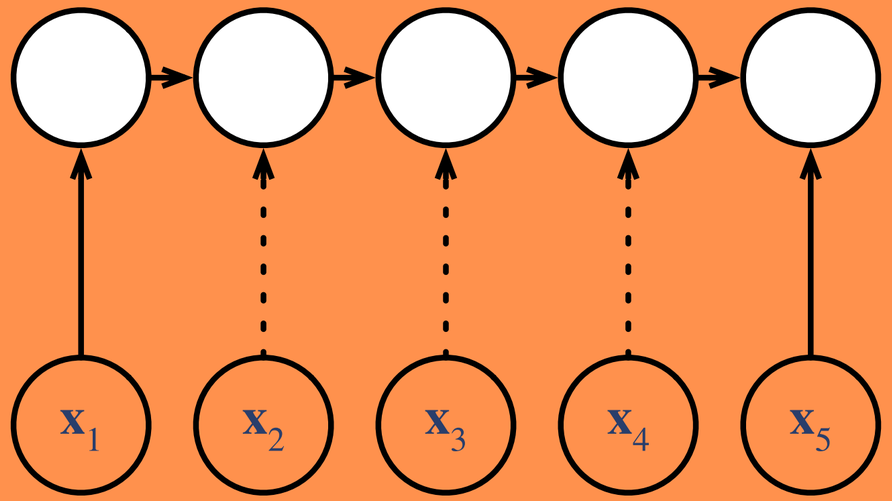
- Context from the first word must survive through hundreds of time-steps.
- The longer the sentence, the more the information fades (Vanishing Gradient).
- Consequence: Models fail to understand context in long documents.

https://irendering.net/understanding-recurrent-neural-network-and-long-short-term-memory/



RNN

- Time complexity **O(N)**
- Linear path
- Bottleneck.

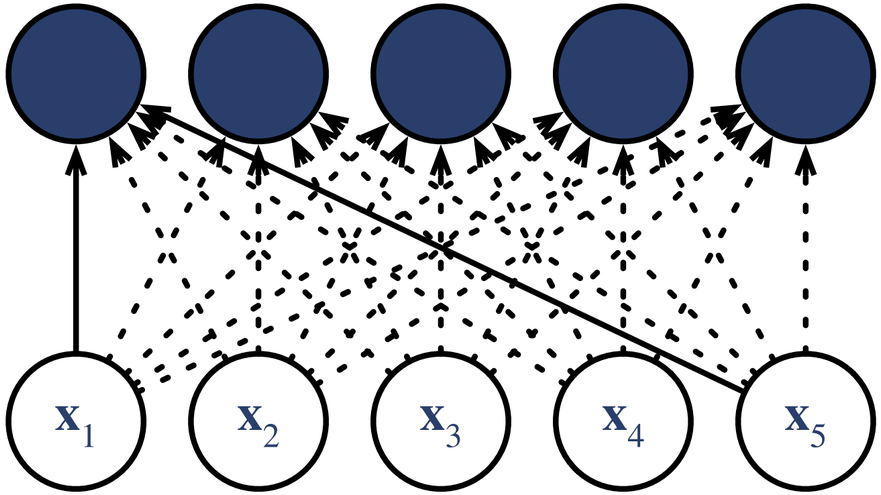


If the data has 1 million words, we wait for 1 million steps

https://developmentseed.org/tensorflow-eo-training-2/docs/Lesson7b_comparing_RNN_transformer_architectures.html

Transformer

- Time complexity **O(1)**
- All-to-All connection
- Massive Speedup



Transformer doesn’t wait

The Magic of Q, K, V

The secret to **Self-Attention** lies in three vectors: Query, Key, and Value

Query (Q), Key (K), and Value (V)

Attention is essentially a **Retrieval System**. The analogy:

- **Query (Q)**: A person holding a sticky note with a topic written on it (e.g., "Nature").
- **Key (K)**: Labels on the spine of the books/folders.
- **Value (V):** The actual content inside the book/folder.

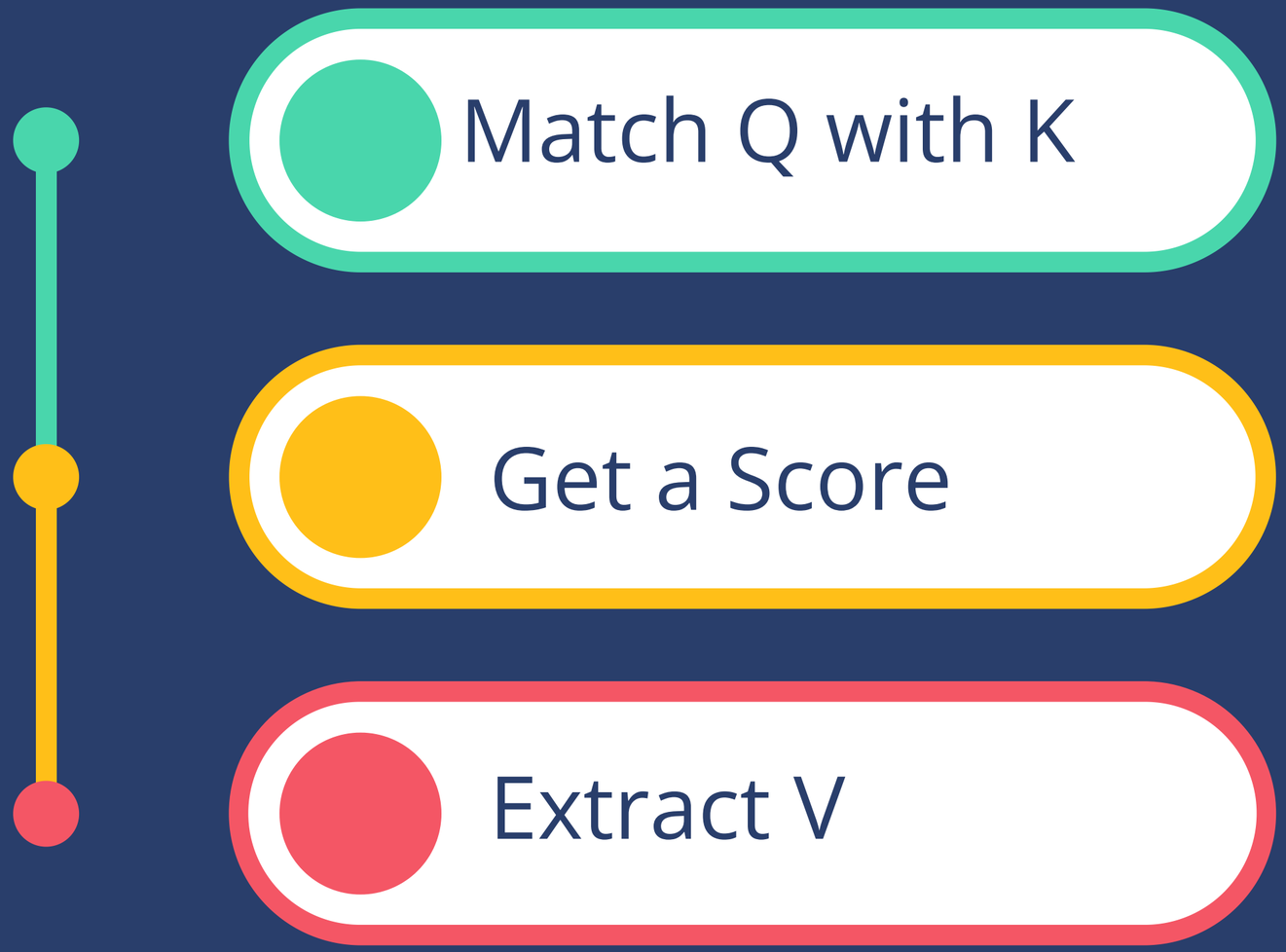
$$Attention(q, k, v) = \overbrace{softmax\left(\frac{qk^T}{\sqrt{d_k}}\right)}^{\text{Attention weights}} v$$

from (q) to (k)

vector dimensionality of K, V

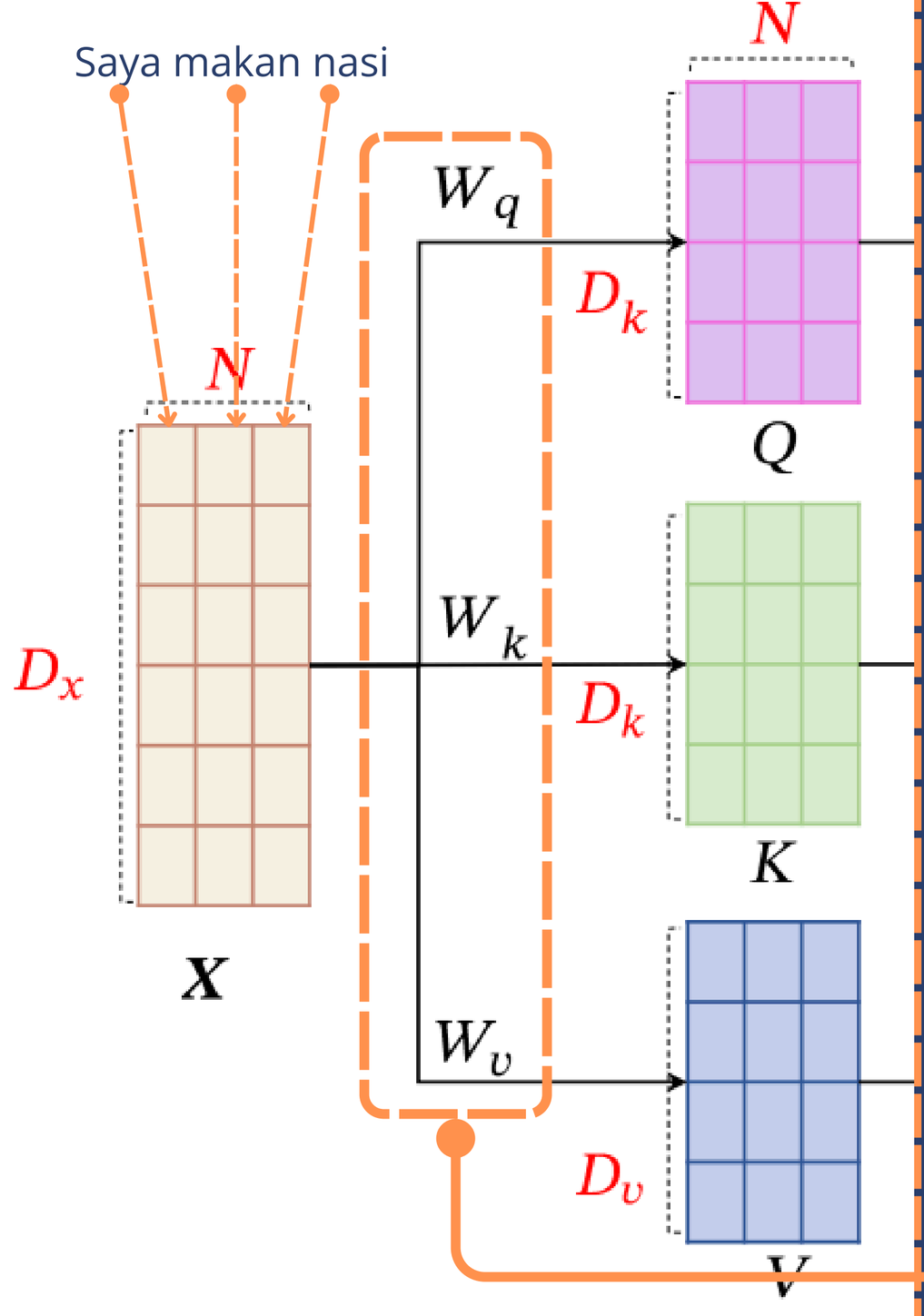
https://vasudev.bearblog.dev/vision-transformer-1/

The Database Analogy



For every word (token), the model creates 3 vectors:

- **Query (Q):** What am I looking for?
  The current token asking for context.
- **Key (K):** What do I offer?
  Other tokens advertising their content.
- **Value (V):** What is my actual information?
  The content to be extracted.



The Mechanics - Step 1: Linear Transformation

- The raw Input Embedding (**X**) is multiplied by three separate weight matrices ($\mathbf{W_Q}$, $\mathbf{W_K}$, $\mathbf{W_V}$)
- The model **learns** how to construct the best Queries, Keys, and Values during training.
- Why separate them?
  A word needs to behave differently when it is **asking for context** vs. when it is **giving context**.

https://www.cnblogs.com/hbuwyg/p/16978264.html

Imagine a word **“Bank”** in a sentence

Asking Context

The word “Bank” is asking for context

- “Bank” wants to know who he is. So he is looking for clues.
- His query (Q): "I need help understanding nature/geography."
- “Bank” is broadcasting a request.

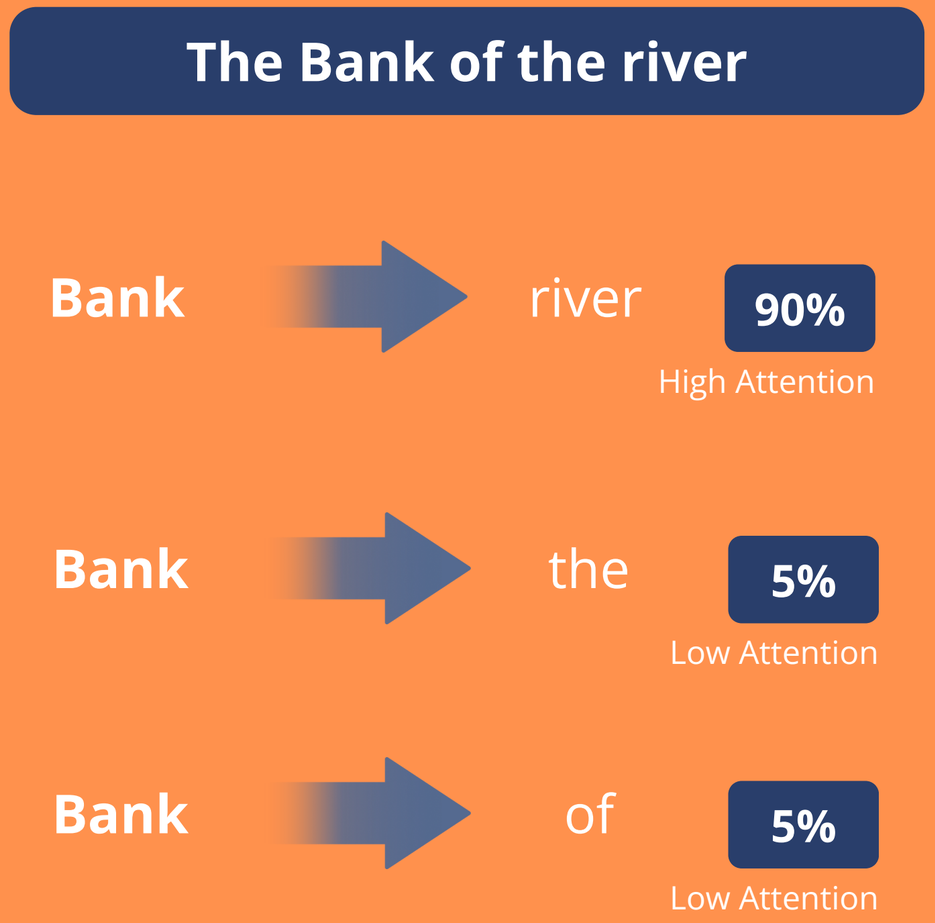
“Bank” as Query (Q)

Giving Context

The work “Bank” is giving context

- Now imagine someone else, the word "Money", is looking for context.
- "Money" sends out a Query: "I need financial context."
- "Bank" need to answer him.
- The Key (K): "I am a financial institution."
- The Value (V): [Here is the financial concept of a bank].

“Bank” as Key (K) / Value (V)

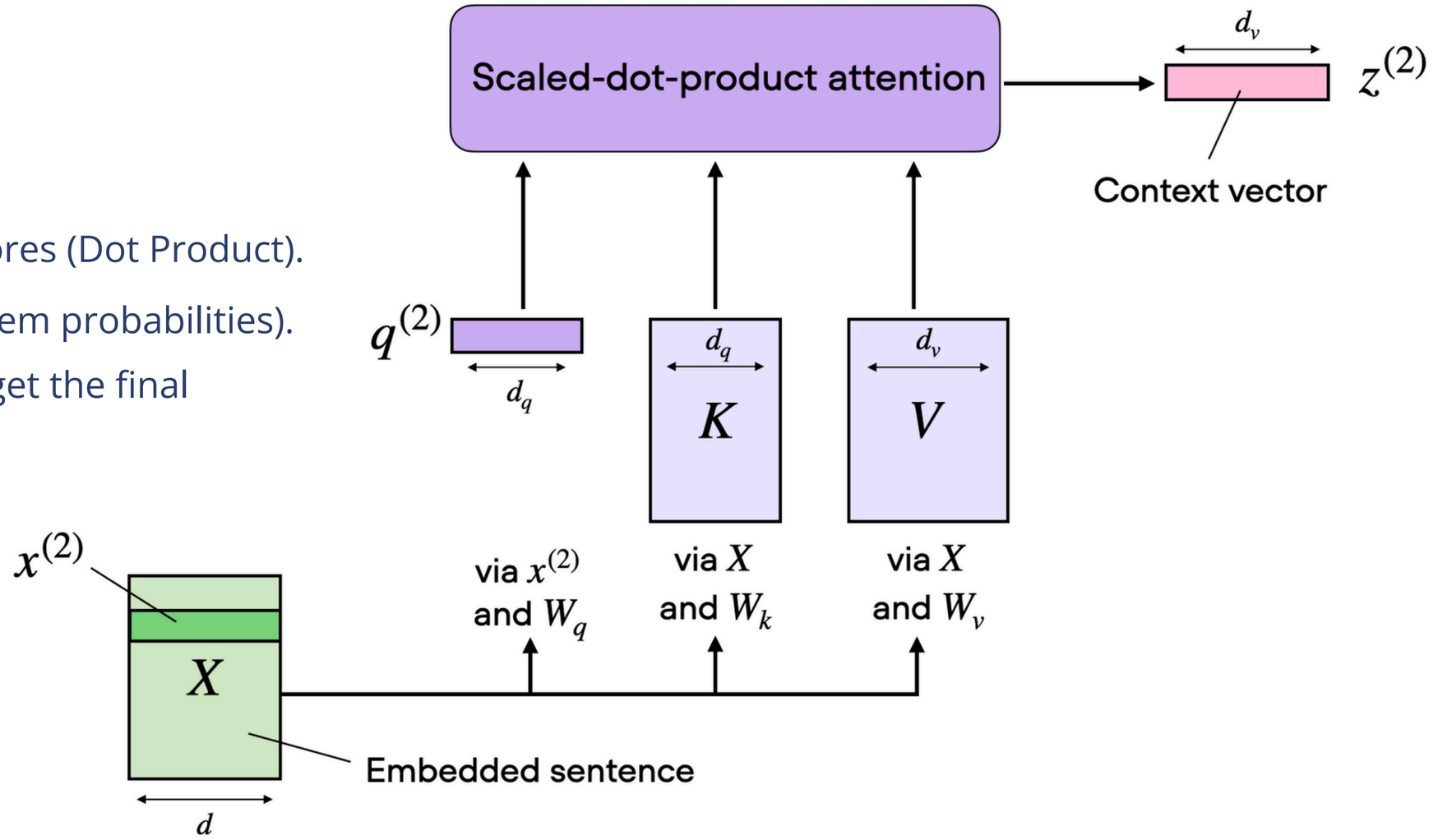


The Mechanics - Step 2: The "Bank" Example

- Words with multiple meanings (Polysemy)
- The Matching Process (Dot Product):
  - Q ("Bank") looks for context related to nature/water.
  - K ("River") shouts "I am nature! I am water!"
  - Result: High compatibility score.
- The model pays high attention to "river" and ignores "The" or "of".

Full Pipeline

- Create **Q**, **K**, **V** vectors.
- Match **Q** and **K** to calculate scores (Dot Product).
- Scale and Softmax (to make them probabilities).
- Combine the scores with V to get the final representation.



The Result: **A context-aware vector.** "Bank" is no longer just a word; it's now "Bank-influenced-by-River".

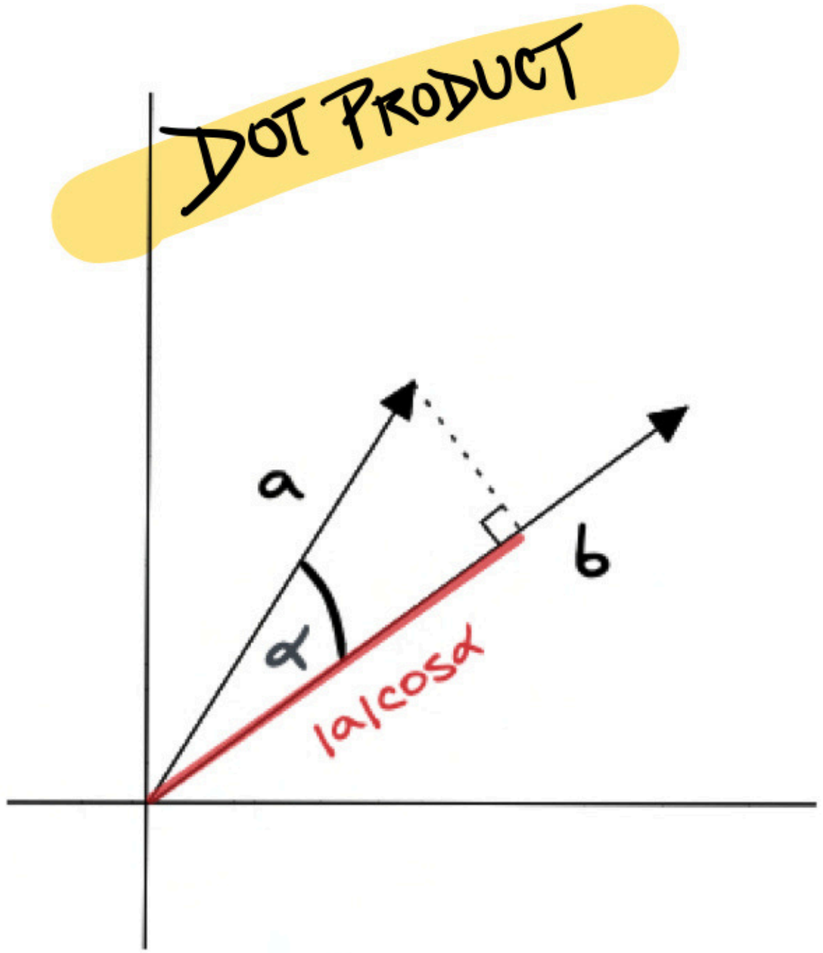
$$Attention(q, k, v) = \overbrace{softmax\left(\frac{qk^T}{\sqrt{d_k}}\right)v}^{\text{Attention weights}}$$

from (q) to (k) vector dimensionality of K, V (d_k)

https://sebastianraschka.com/blog/2023/self-attention-from-scratch.html

A Closer Look into the Math

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

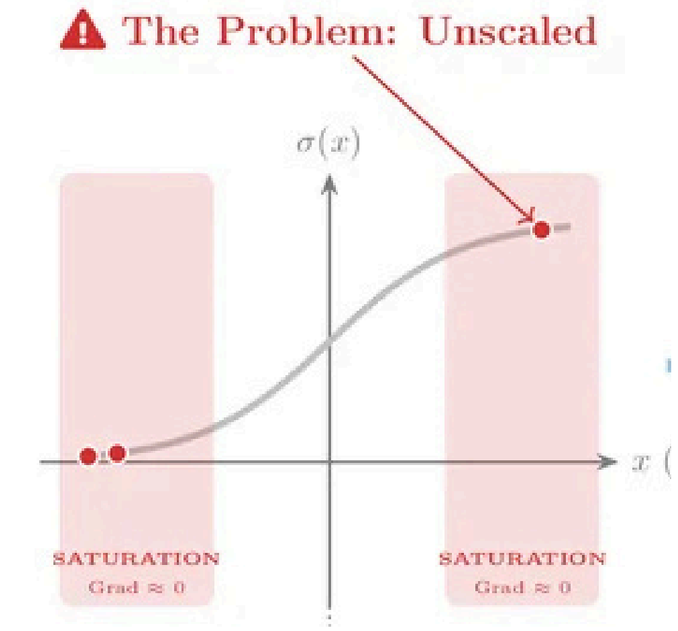


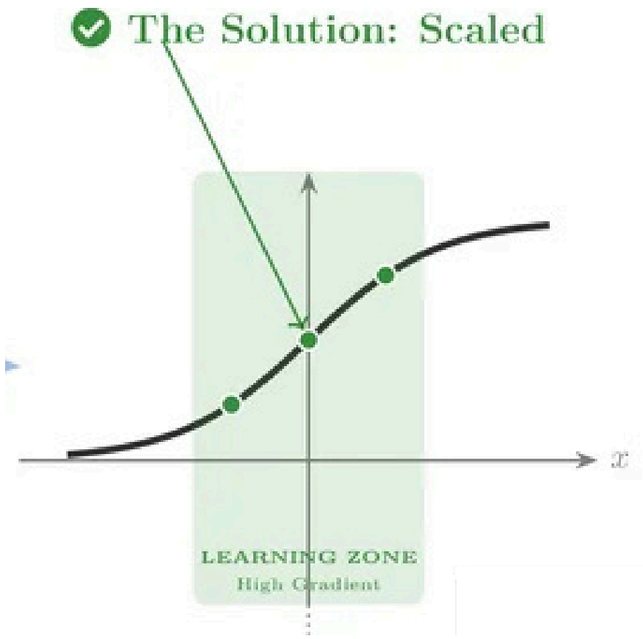
$$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}||\mathbf{b}| cos\alpha$$

https://intuitiveshorts.substack.com/p/short-12-mips-vs-mcss-maximum-inner

Step 1 - The Dot Product (QK^T)

- **What is it?** Matrix multiplication of Queries and Transposed Keys.
- Why Dot Product?
  - It calculates Similarity.
  - If vector **Q** aligns with vector **K**, the result is a large positive number.
  - If they are unrelated, the result is near zero or negative.
- **The Output**: A raw score matrix (Logits) showing how much each word "likes" every other word.

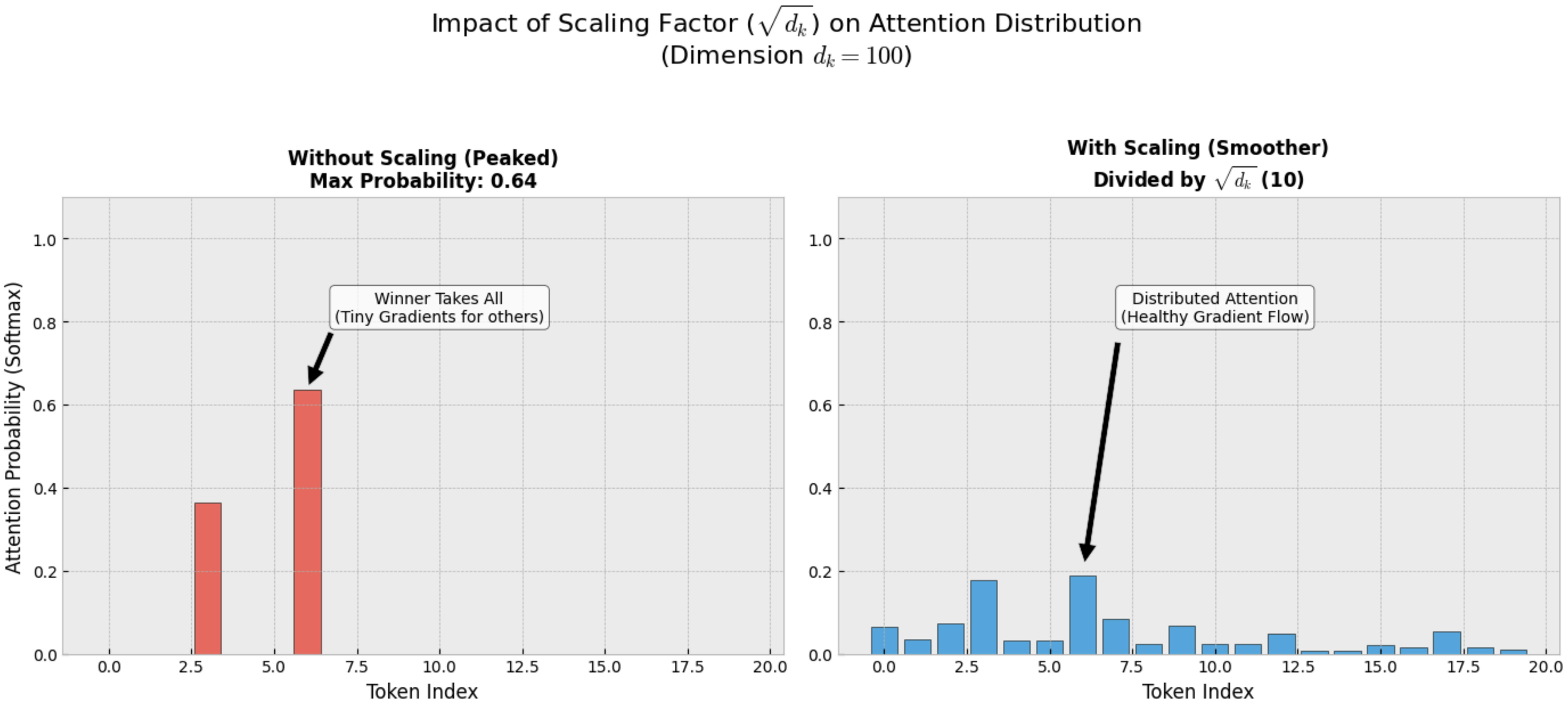




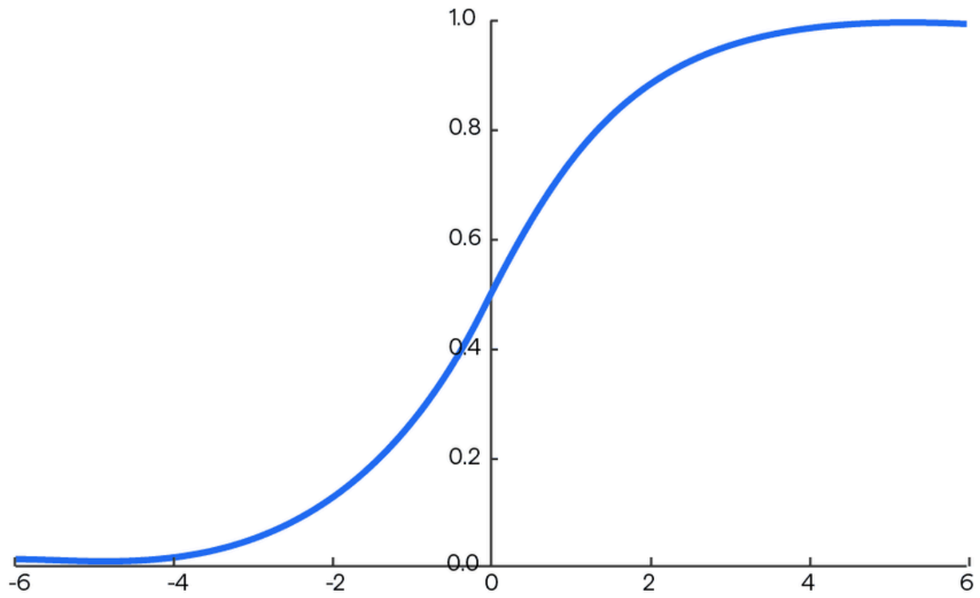
Step 2 - The Scaling Factor $\sqrt{d_k}$

- As the dimension of vectors ($\mathbf{d_k}$) grows, the dot product values can get **explodingly huge**.
- Large values push Softmax into regions where gradients are tiny (**Vanishing Gradients**).
- Divide by $\sqrt{d_k}$ (square root of the dimension) can keeps the scores **stable** and ensures the model learns efficiently.

https://www.reddit.com/r/learnmachinelearning/comments/1p8xeye/transformer_model_in_nlp_part_6/


Impact of Scaling Factor ($\sqrt{d_k}$) on Attention Distribution
(Dimension $d_k = 100$)
Without Scaling (Peaked)
Max Probability: 0.64
Winner Takes All
(Tiny Gradients for others)
Attention Probability (Softmax)
Token Index
With Scaling (Smoother)
Divided by $\sqrt{d_k}$ (10)
Distributed Attention
(Healthy Gradient Flow)
Token Index

$$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

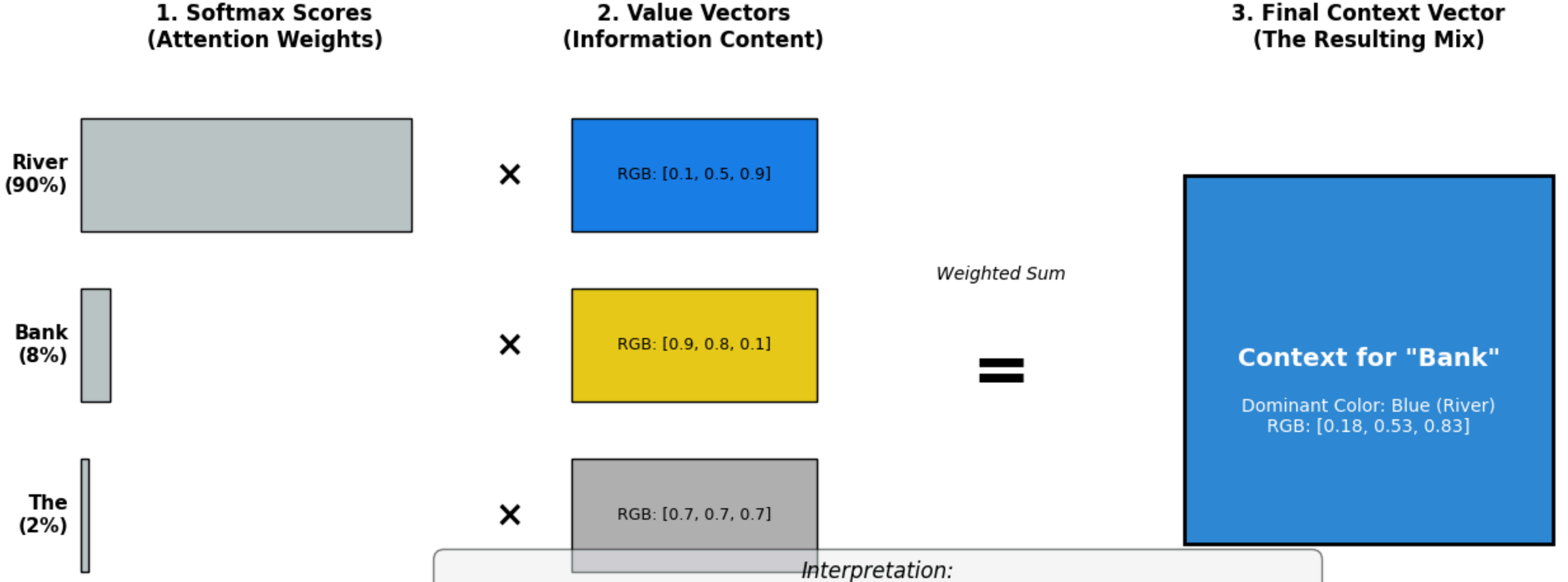


https://botpenguin.com/glossary/softmax-function

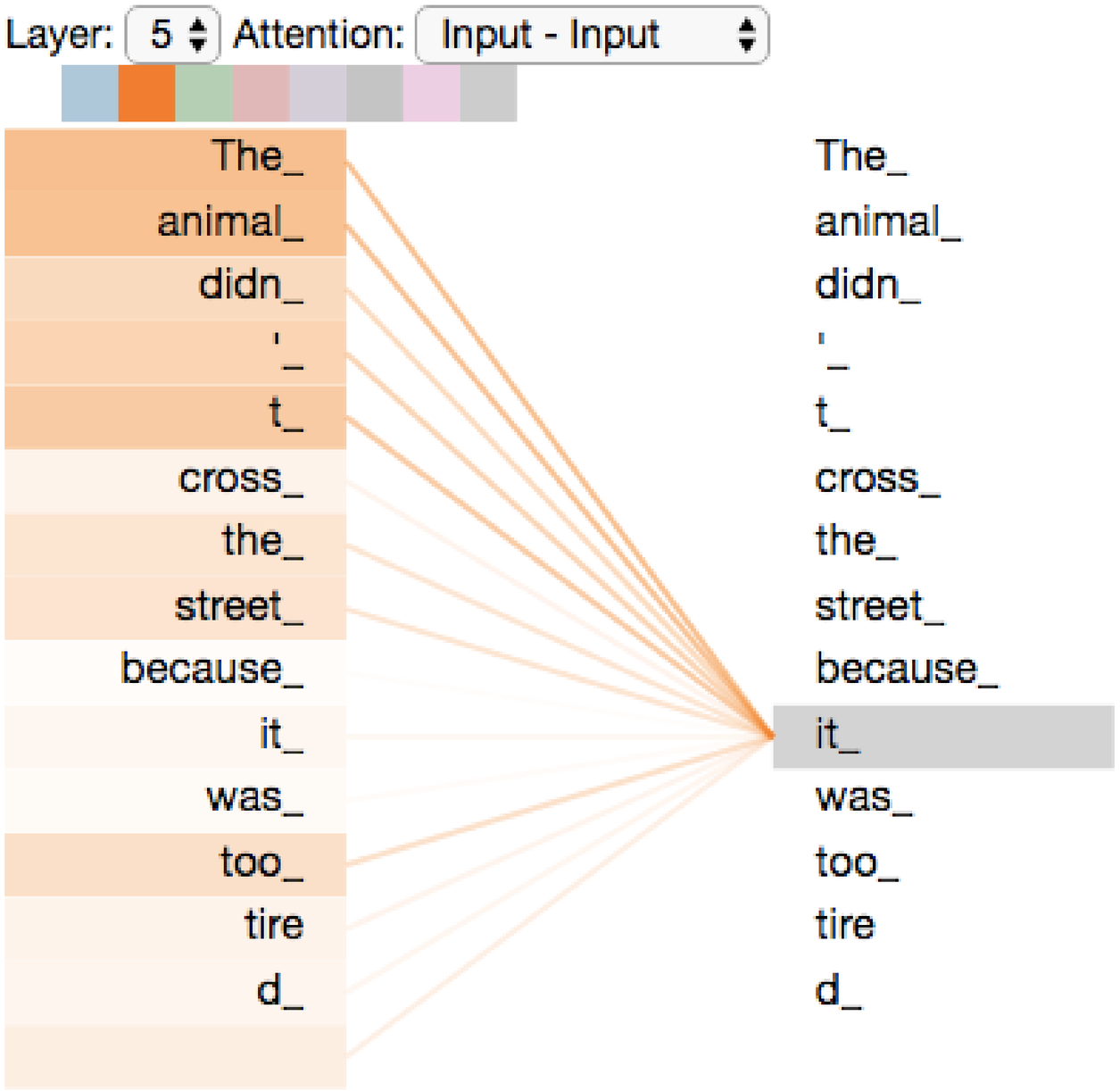
Step 3 - Softmax & The Value Mix

- **Softmax**: Converts raw scores into Probabilities (0.0 to 1.0). Ensures the total attention adds up to 100%.
- Multiply by **V** means we take a Weighted Average of all Values.
- **Final Output:** A new vector that is a blend of all relevant context.

Visualizing Attention: Mixing Value Vectors



Interpretation:
The word 'Bank' pays 90% attention to 'River'.
Therefore, the final meaning (Vector) is 90% composed of 'River's' content (Blue), mixed with a tiny bit of its own original content (Yellow).

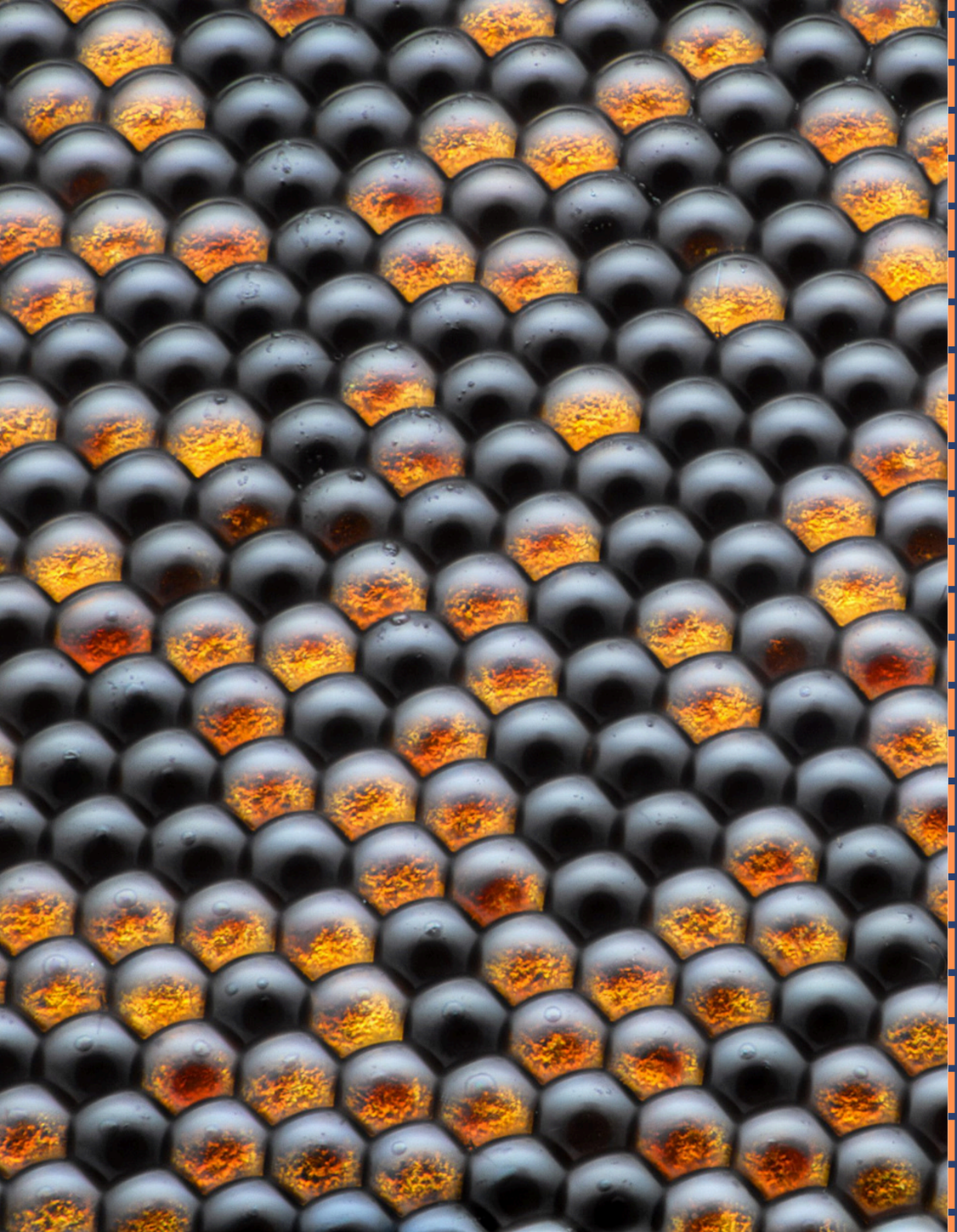


Attention Visualization

When the model is processing the word “it”, self-attention allows it to associate “it” with “animal”.

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

Refinement Step

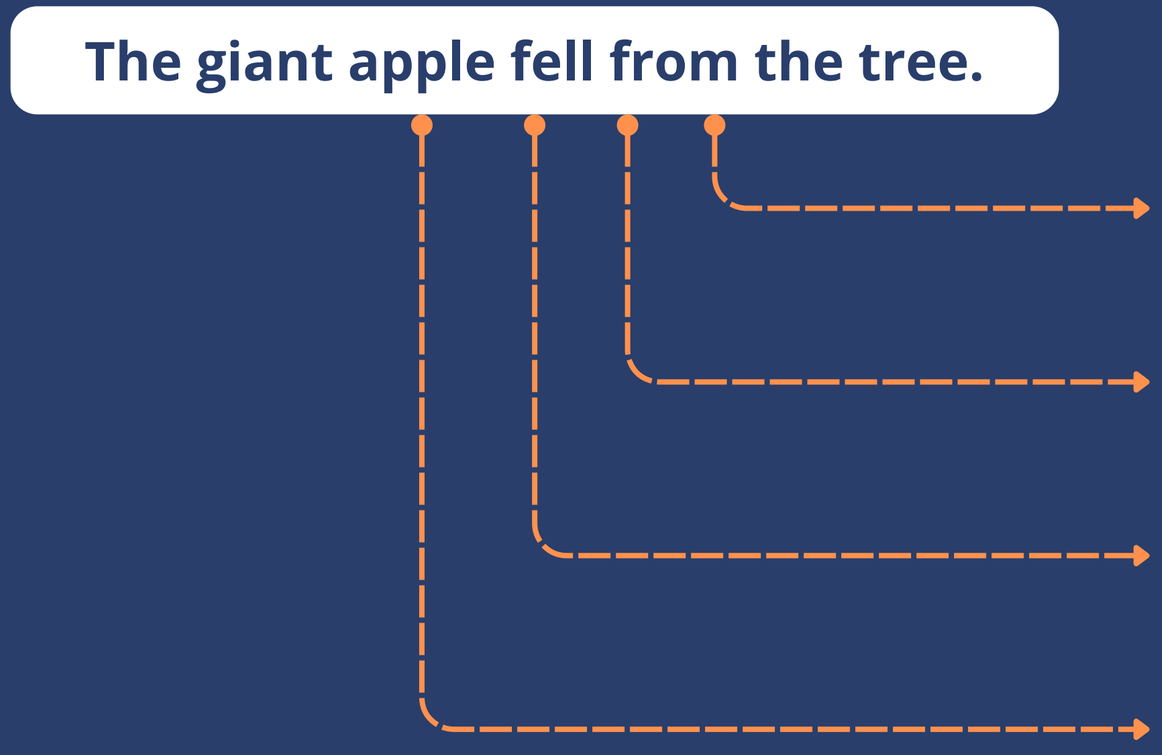


Multi-Head Attention, One Is Not Enough

- Single-Head Attention can only focus on **one** aspect of the relationship at a time.
- To solve this problem, we run the Attention mechanism multiple times in parallel.
- Standard Numbers:
  - Transformer (Original): 8 Heads.
  - GPT-3: 96 Heads.
  - GPT-4: Much more.

https://www.nikonsmallworld.com/galleries/2019-photomicrography-competition/housefly-compound-eye-pattern

Different Head, Different Perspectives

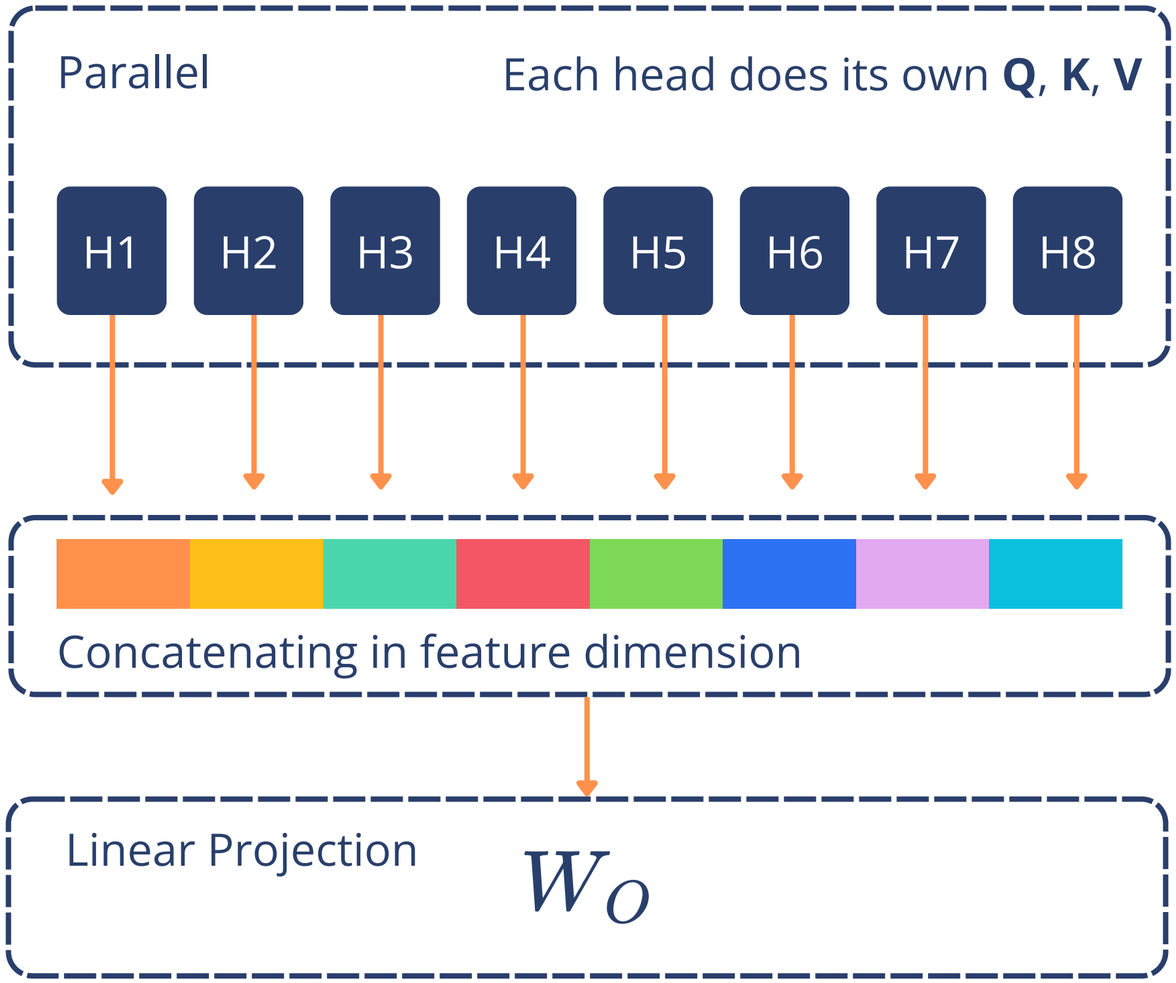


Head 1 (Grammar Focus):
"The" -> "apple" (Article to Noun).

Head 2 (Descriptive Focus):
Connects "giant" -> "apple" (Adjective to Noun).

Head 3 (Action Focus):
Connects "fell" → "apple" (Verb to Subject).

Head 4 (Preposition Focus):
Connects "from" -> "tree".

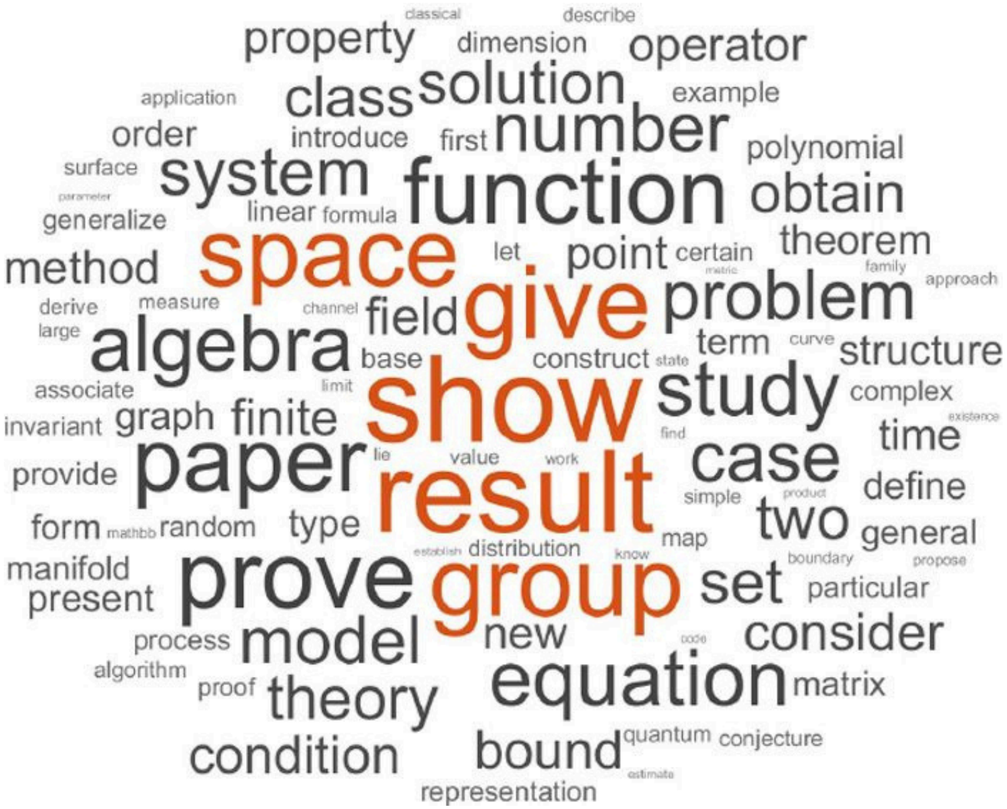


Concatenation The Heads

- Divide the embedding dimension (d_{model}) by the number of heads (h).
- Each head does its own **Q**, **K**, **V** math.
- Glue the outputs side-by-side (contatenation).
- Multiply by a final weight matrix ($\mathbf{W_o}$) to blend the insights together.

Man bites Dog = **Dog bites Man**

(in Pure Transformer, these sentences are same)



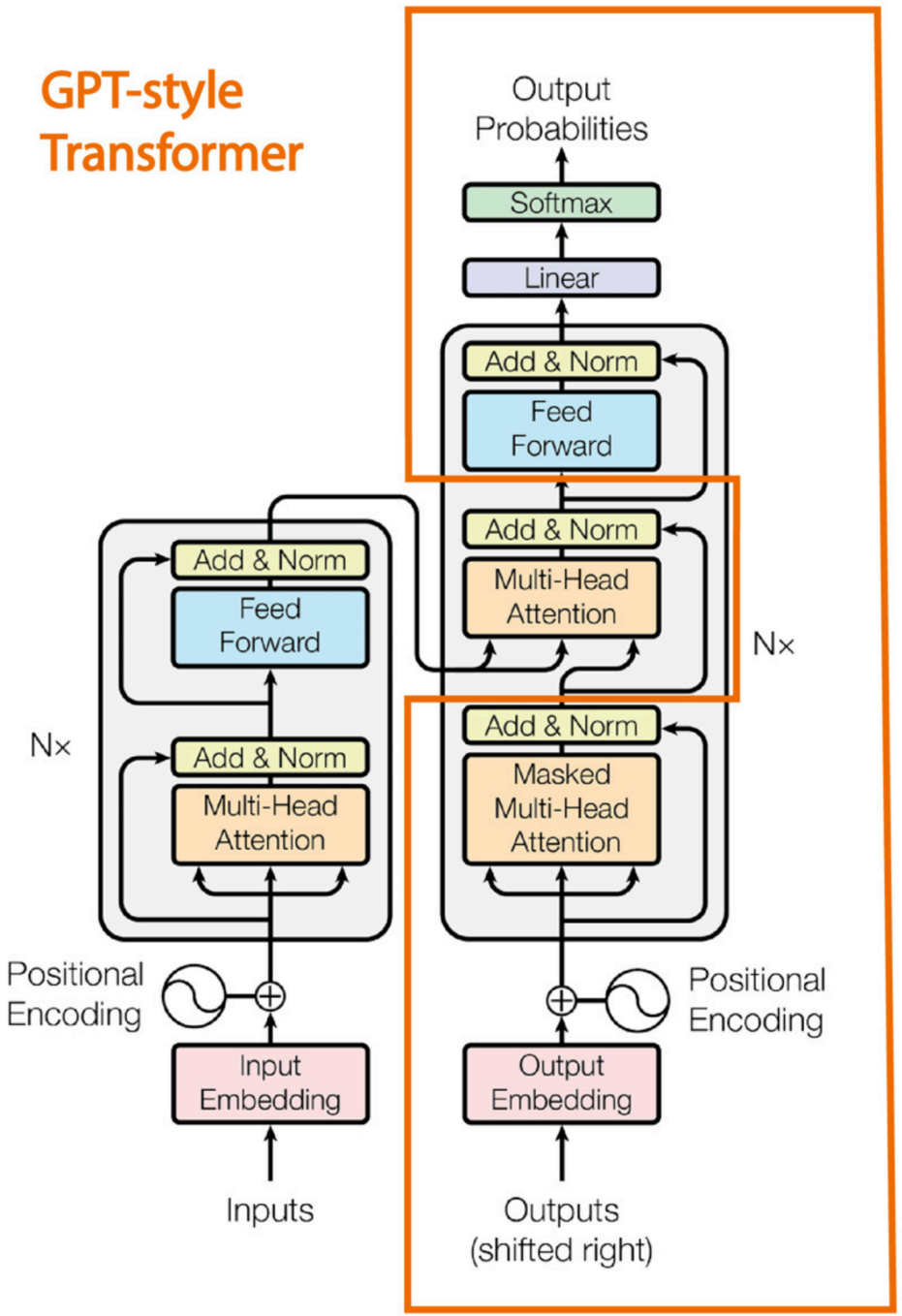
Positional Encoding, Fixing the "Bag of Words"

- Self-Attention is **Permutation Invariant**.
  - It treats inputs as a "**set**", not a "sequence".
  - It doesn't inherently know that "Man" comes before "Bites".

- **The Fix:** Positional Encoding.
  - We **inject** information about the order of words into the embeddings before they enter the Transformer.
  - Uses Sine and Cosine waves (Original paper) or Learned Embeddings.

Input = Embedding Vector (Content) + Position Vector (Order)

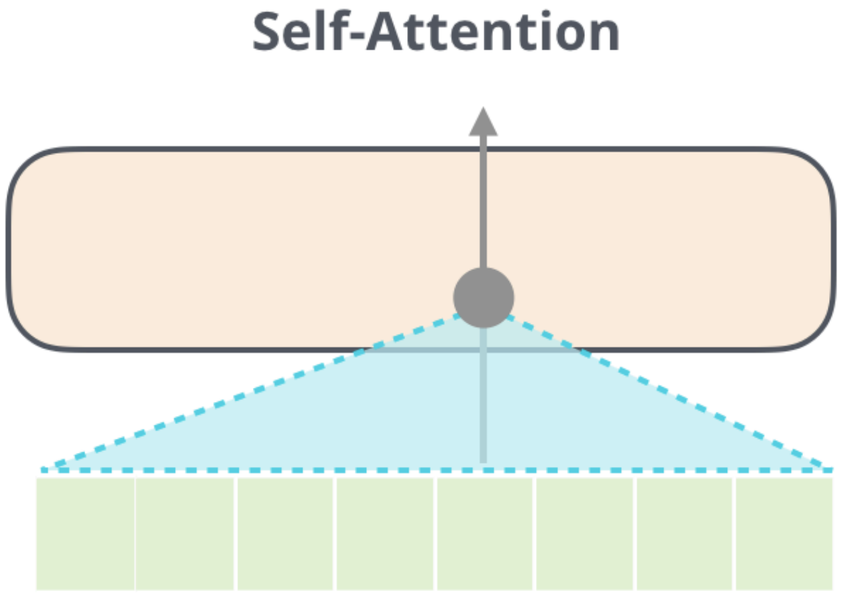
The Text Generation Flow

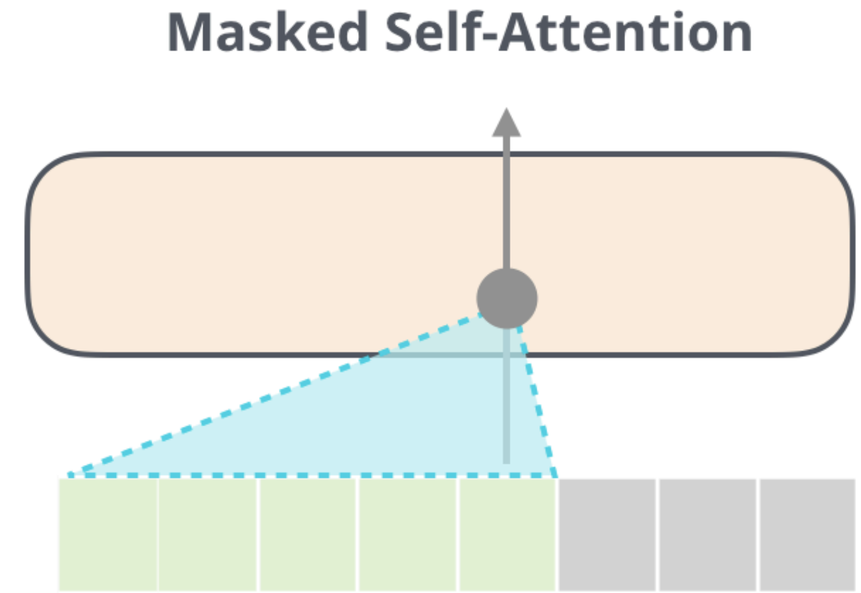


Encoder vs. Decoder

- For Text Generation: We use the Decoder-Only architecture (e.g., GPT-3, Llama, Claude).
- Encoders see the whole sentence at once
- Decoders generate one word at a time, moving from Left to Right.

https://towardsdatascience.com/the-transformer-architecture-of-gpt-models-b8695b48728b/

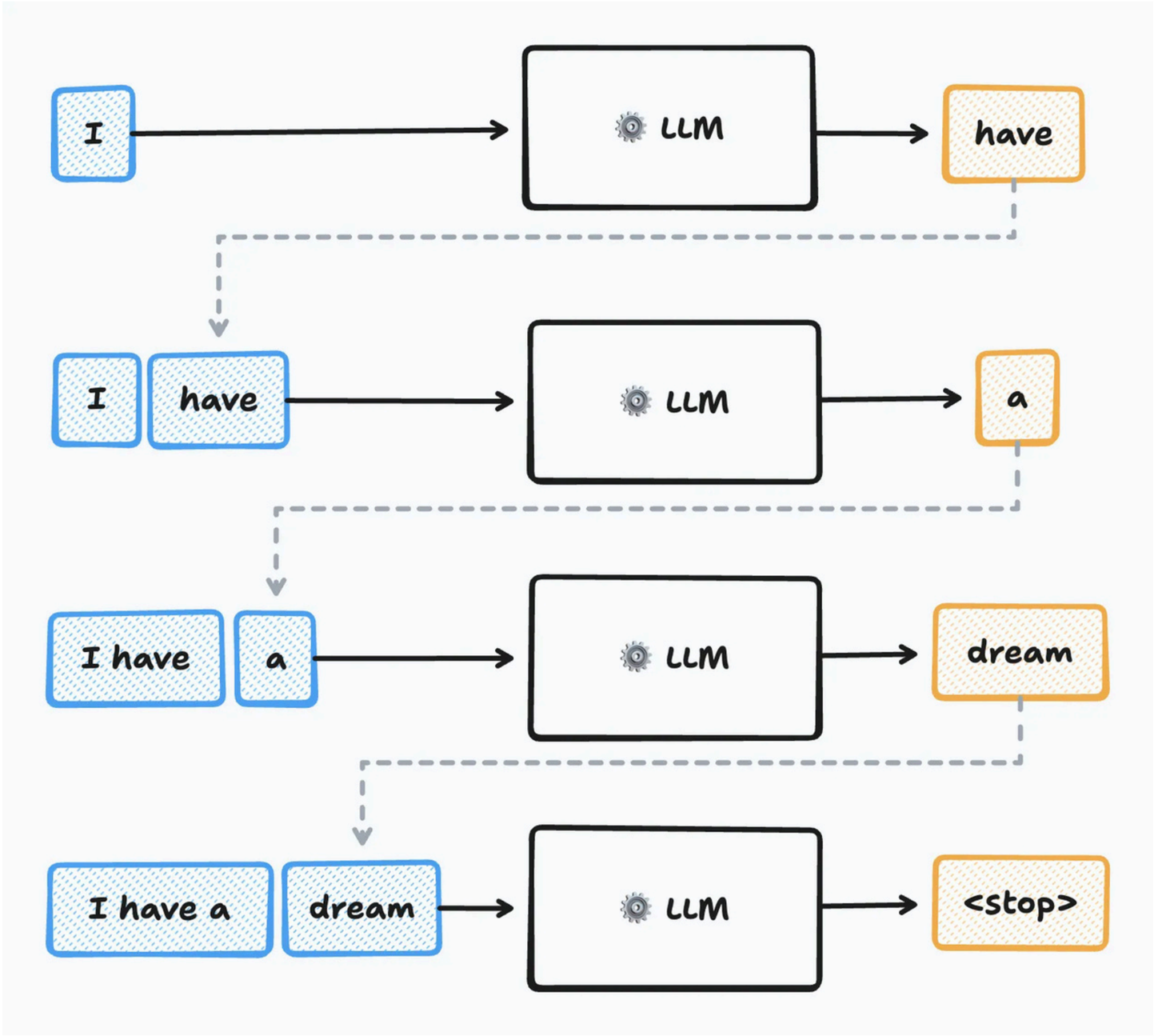




Masked Self-Attention

- If the model sees the future word during training, it won't learn anything.
- We artificially set the attention scores for future tokens to Negative Infinity.
- When passed through Softmax, these become Zero.
- The model is forced to predict the next word based only on past words.

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



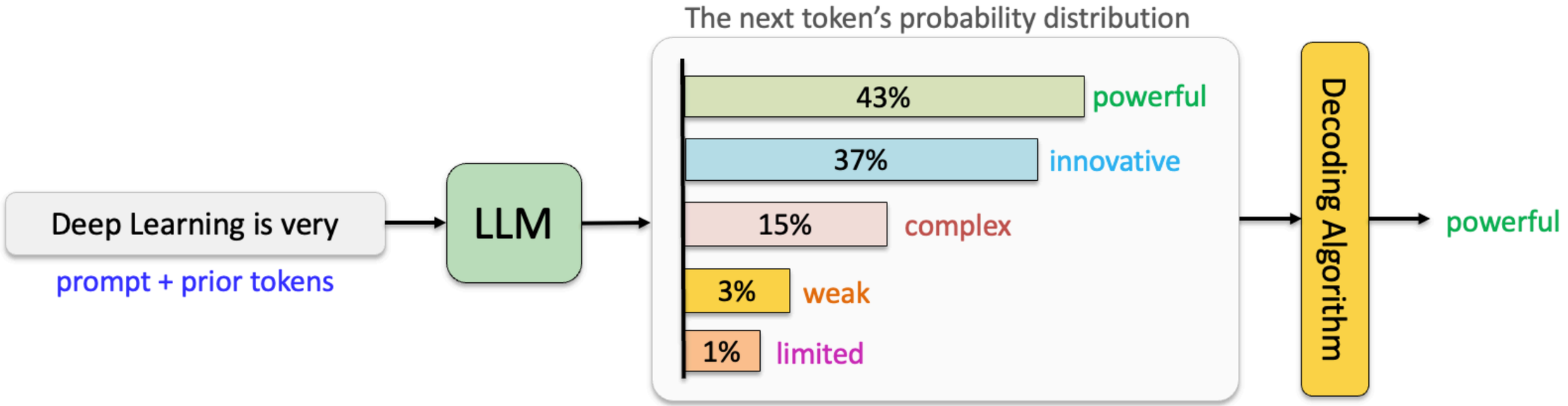
The Autoregressive Loop

- Autoregressive Generation.
- **Input**: Feed current context tokens.
- **Self-Attention**: The last token looks back at all previous tokens to gather context.
- **Prediction**: Model outputs a probability distribution for the next word.
- **Loop**: The predicted word is appended to the input, and the process repeats.

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

From Probabilities to Text (Decoding)

- The model doesn't output text; it outputs Logits (Raw scores for every word in the dictionary).
- This logits are then converted into probabilities by softmax
- Common strategy is greedy. Always pick the highest %.



https://aiml.com/what-are-some-of-the-approaches-for-decoding-the-next-word-in-llms/

https://medium.com/deeplearningbrasilia/deep-learning-recurrent-neural-networks-f9482a246010

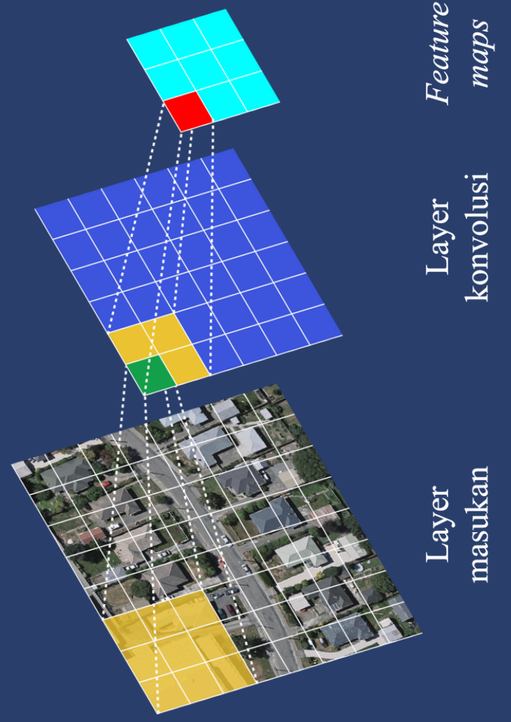
Link to Self-Attention Demo
(Google Colab)

s.id/PAHIT

Talk is cheap, show me the code

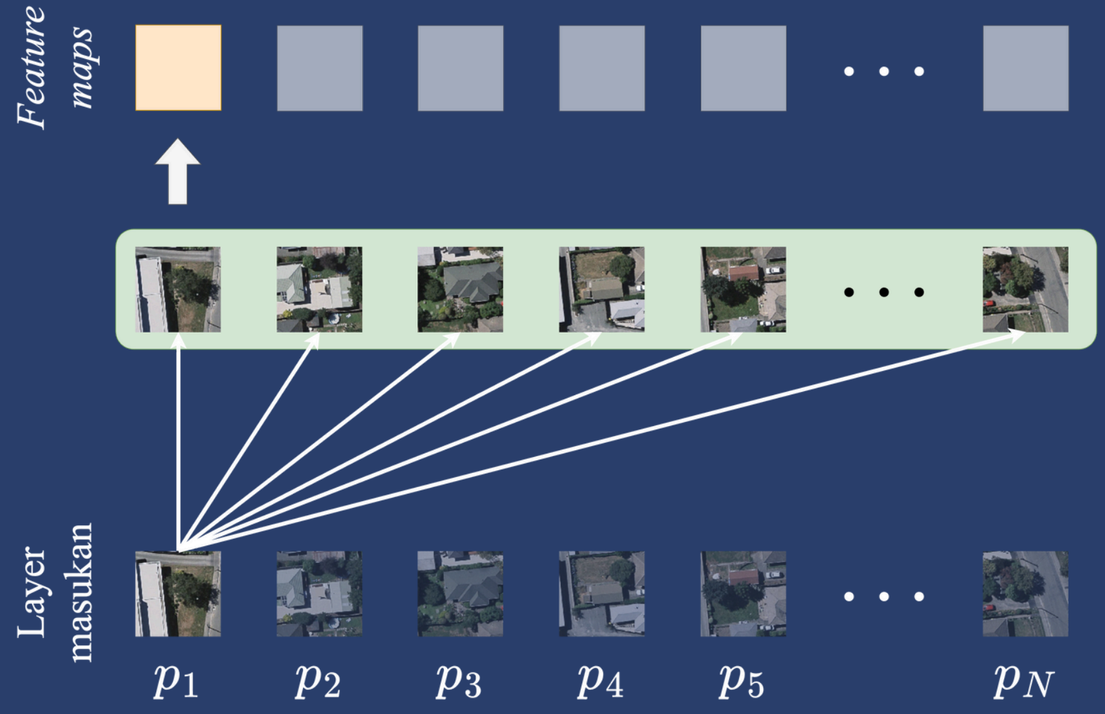
CNN

$$r_i = r_{i-1} + (k-1)\prod_{k=1}^{i-1} s_k$$



Vision Transformer (ViT

$$r_i = N$$



Terima Kasih

Lhuqita Fazry