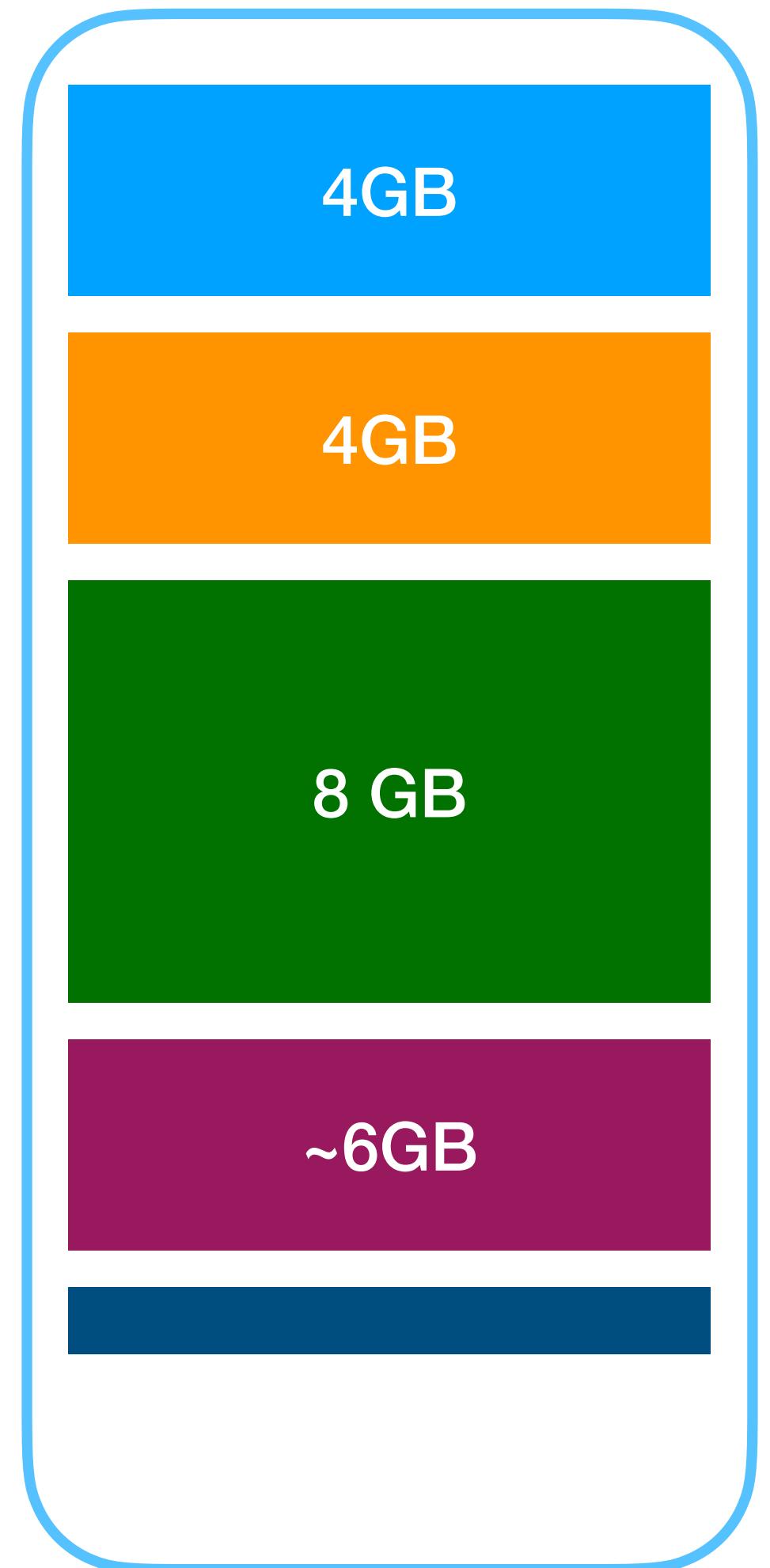


Fine tune a LLM: how much memory do I need?

RTX 4090
24 GB

tuning a 2B model



Original model

2B (FP16) = $2 \times 16 / 8 = 4\text{GB}$

Gradient

Same size with the model

Optimizer

Two tensors: mean + variance

Activation

Model architecture, batch size, sequence length etc, ~1.5x model size

Something else



Understand temperature in LLM

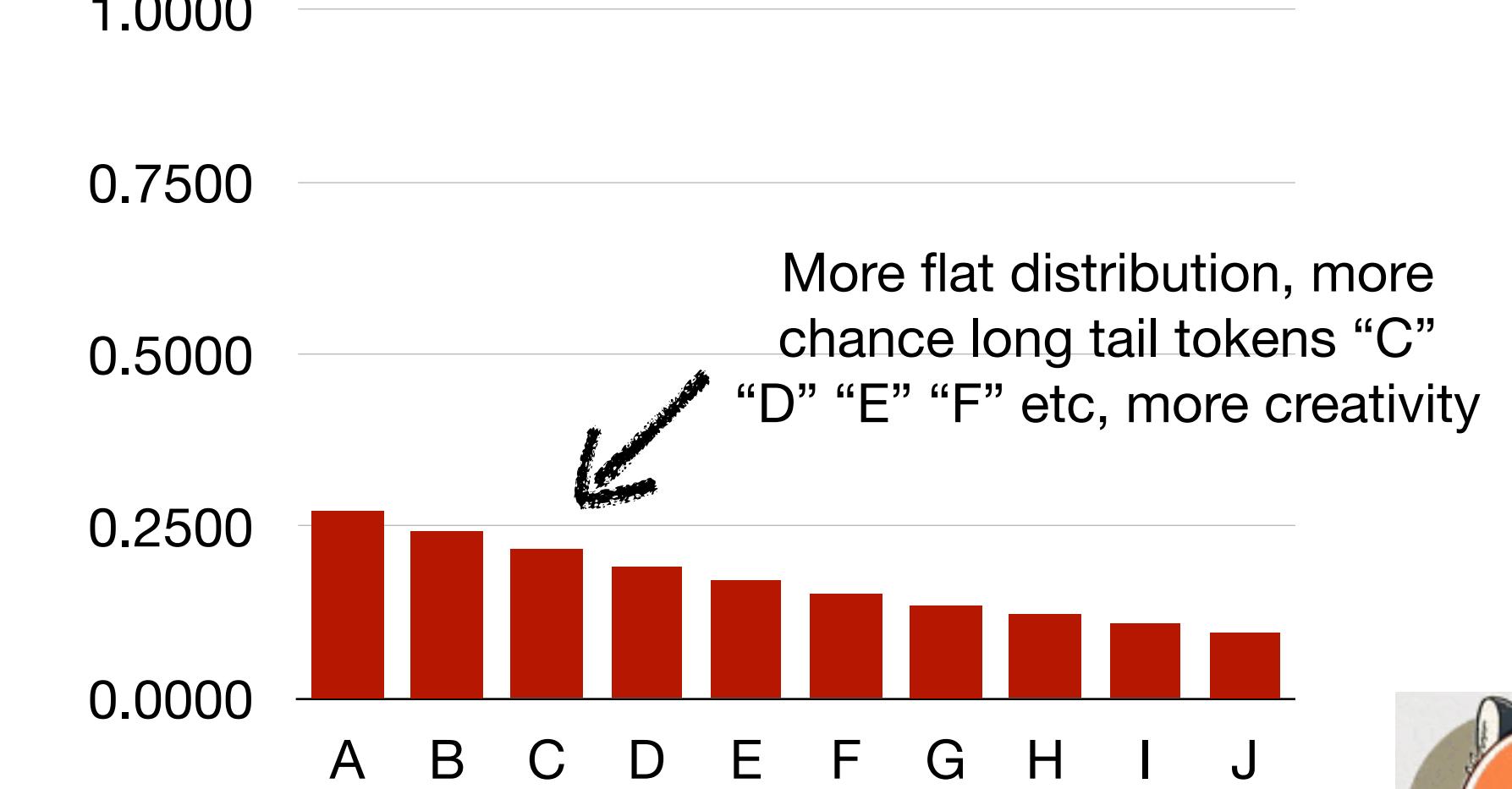
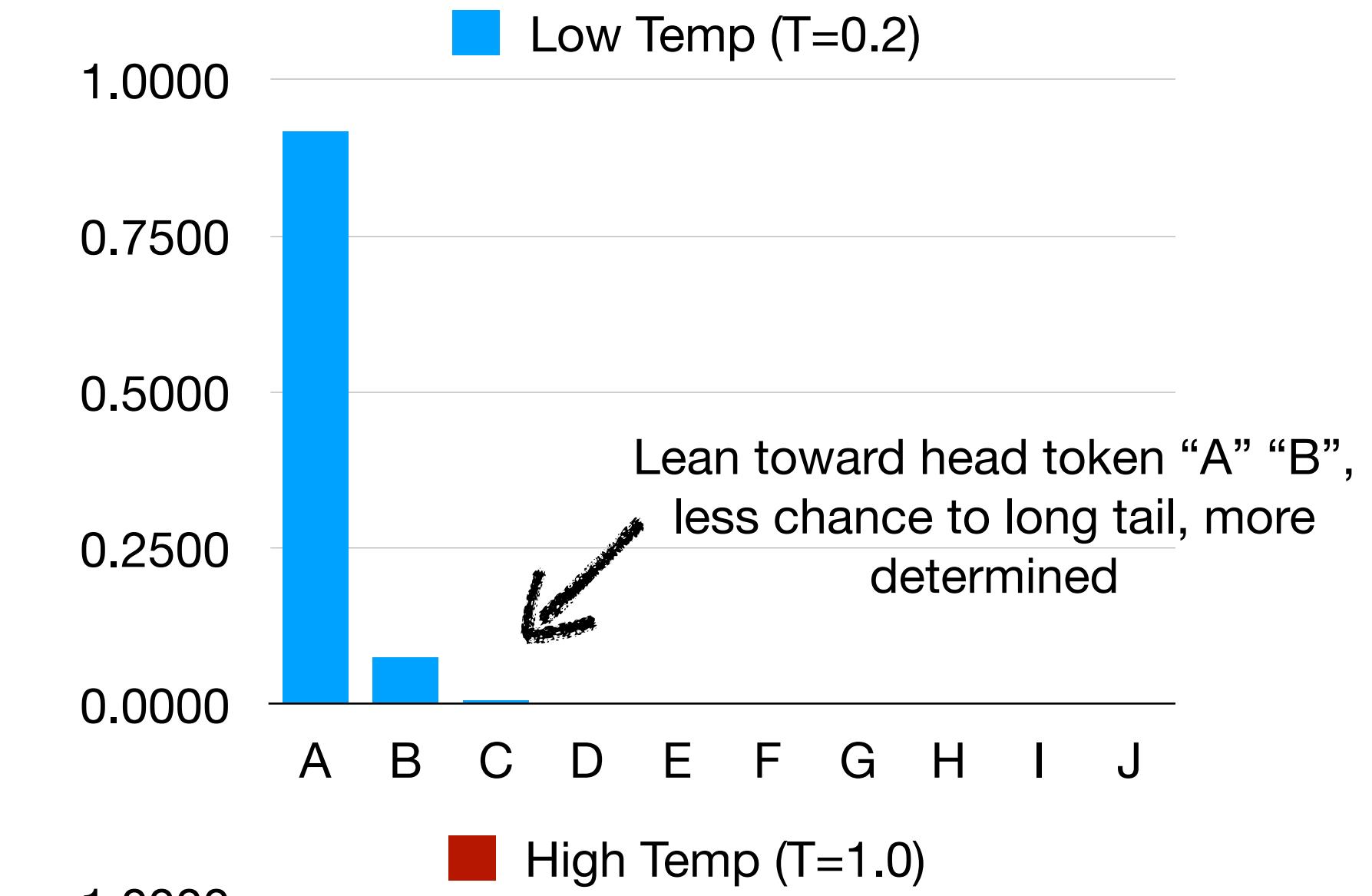
```
response = openai.ChatCompletion.create(  
    model='gpt-4o',  
    temperature=0.7,  
    max_tokens=30,  
    messages=[{  
        'role': 'user',  
        'content': question  
    }],  
)
```

$$P(x_i) = \frac{\exp(z_i/T)}{\sum_{j=1}^N \exp(z_j/T)}$$

Softmax with Temperature
Hinton et al "Distilling the Knowledge in a Neural Network" 2015
<https://arxiv.org/abs/1503.02531>

T = 0.2

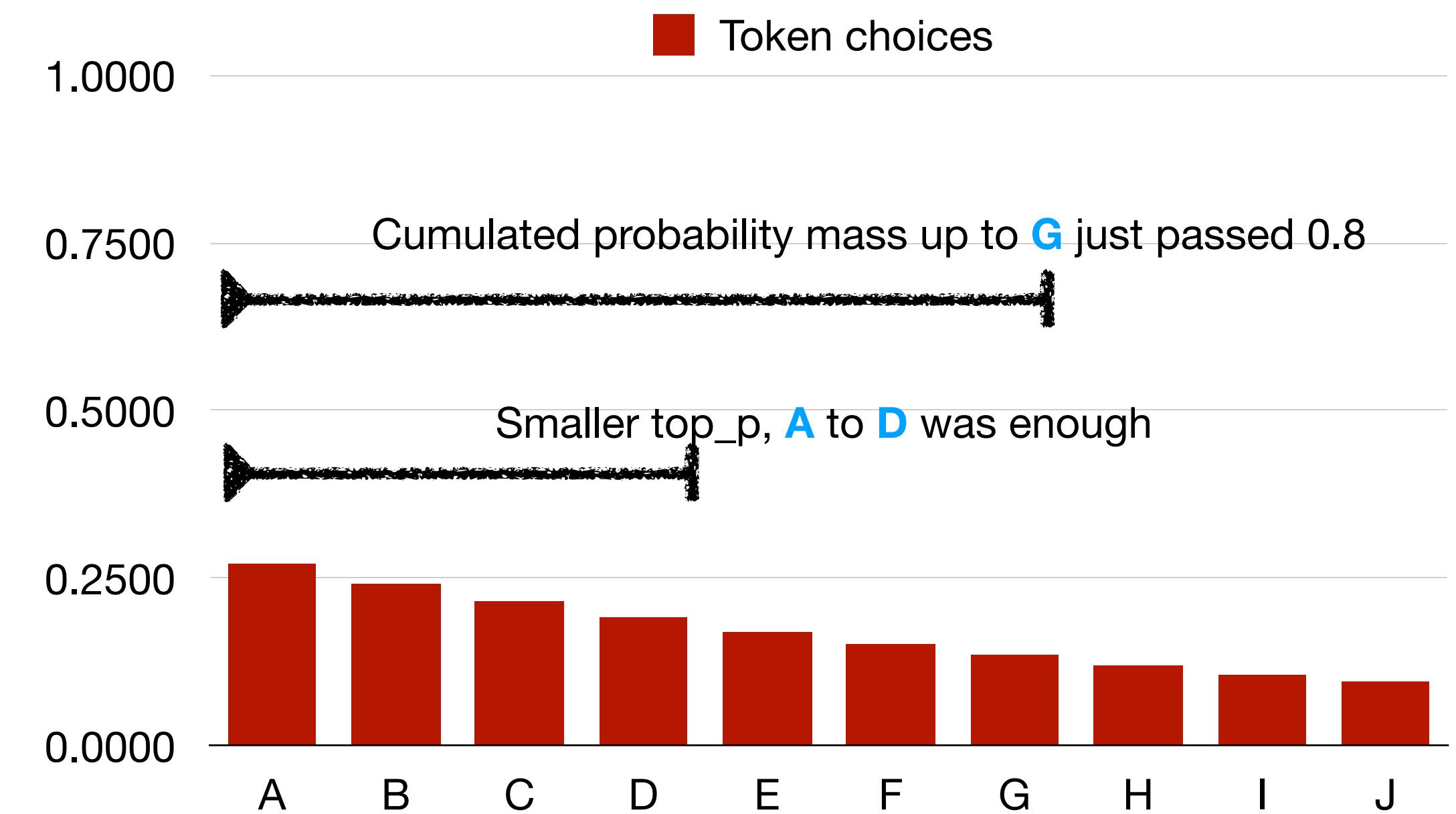
T = 1.0



Understand top_p in LLM

```
response = openai.ChatCompletion.create(  
    model='gpt-4o',  
    max_tokens=30,  
    top_p=0.8,  
    messages=[ {  
        'role': 'user',  
        'content': question  
    } ],  
)
```

$$\sum_{x \in V^{(p)}} P(x|x_{1:i-1}) \geq p$$



Top P or "Nucleus sampling"

Holtzman et al, "The Curious Case of Neural Text Degeneration" 2019

<https://arxiv.org/abs/1904.09751>

More flexible than top K to the shape of the probability distribution, across different contexts.



Understand Boltzmann distribution and neural networks

What computer scientists see

Probability of class **i** given input **x** in a neural network

$$P(\text{class } i|x) = \text{softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

logits (pre-softmax activations)

interpret neural network input as probabilities instead of numbers



What physicists see

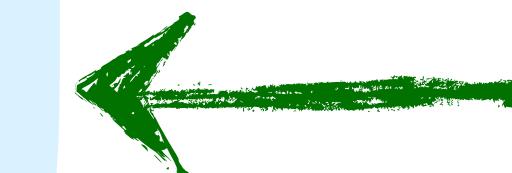
Probability of a certain state

$$P(s) = \frac{1}{Z} \exp\left(-\frac{E(s; \theta)}{T}\right)$$

Partition function (normalization)

Energy of the state

Temperature parameter



Optimization and generalization can make statistical sense.



Understand prompting

Prompting introduces a **task-specific bias** to the model's output distribution by its **activation patterns**, effectively **aligning the target task** with the LLM's pre-trained task manifold.

$$P(y_t | \text{prompt}, x) = \text{softmax}(f(h_L + \beta \nabla_{\text{task}})/\tau)$$
$$h_L = g_L(\dots g_2(g_1([E_{\text{prompt}}; E_x])))$$

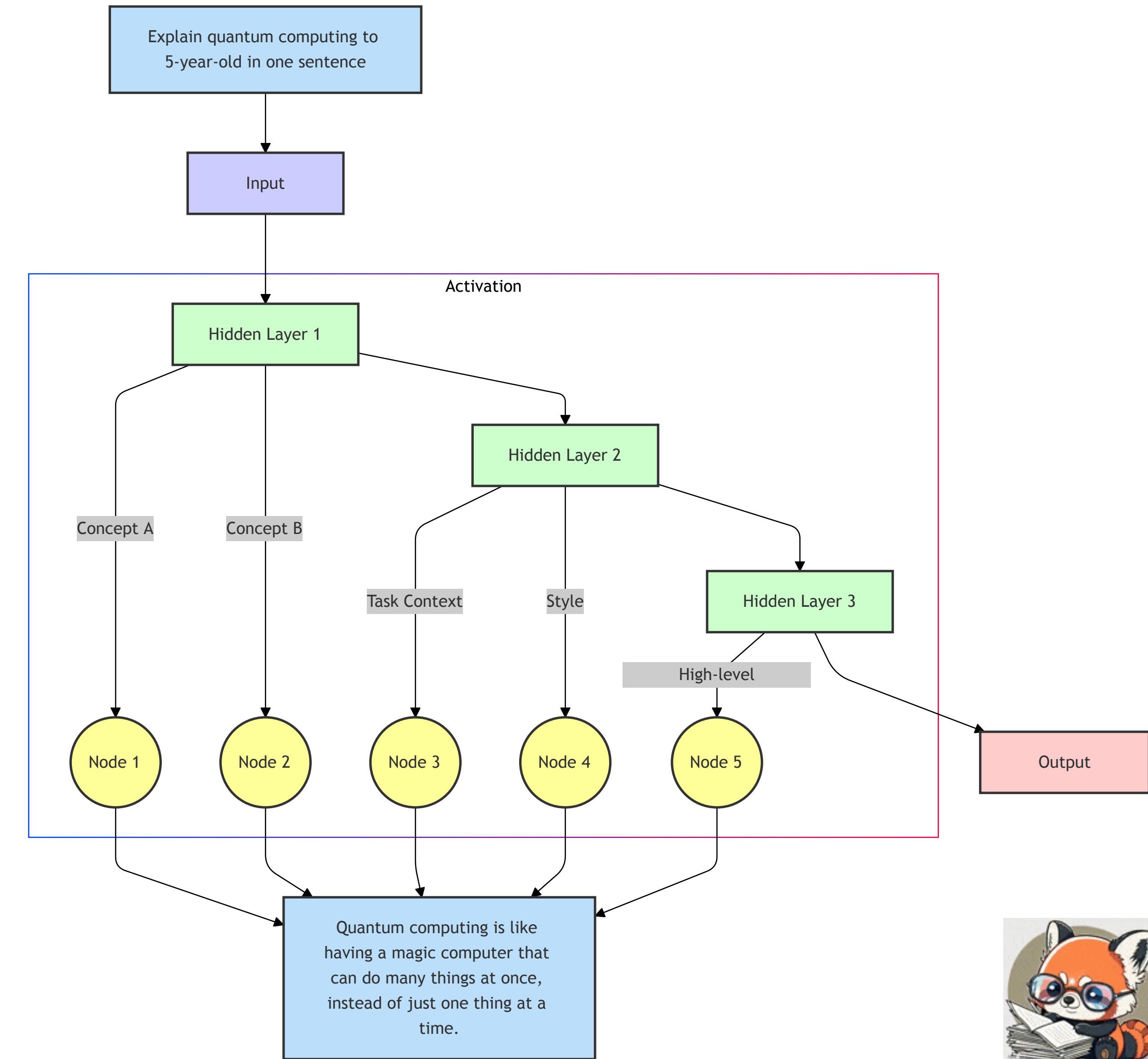
Hidden states

Task gradient

$$\nabla_{\text{task}} = \frac{\partial \mathcal{L}_{\text{task}}}{\partial h_L}$$

Alignment strength

$$\beta = \gamma \cdot \text{sim}(h_L, h_{\text{task}})$$



Understand tool using

LLMs translate language tasks into tool actions by computing $P(\text{tool}|\text{context})$ through **attention-based alignment** between task requirements and tool capabilities.

$$P(\text{action}|\text{state}) = \int P(\text{action}|\text{intention})P(\text{intention}|\text{state})dI$$

Connect state observation to action selection

$$P(\text{tool}|\text{context}) = \text{softmax}\left(\frac{h_{\text{tool}}^{\text{ctx}} W_{\text{out}}}{\tau}\right)$$

Context-aware tool selection

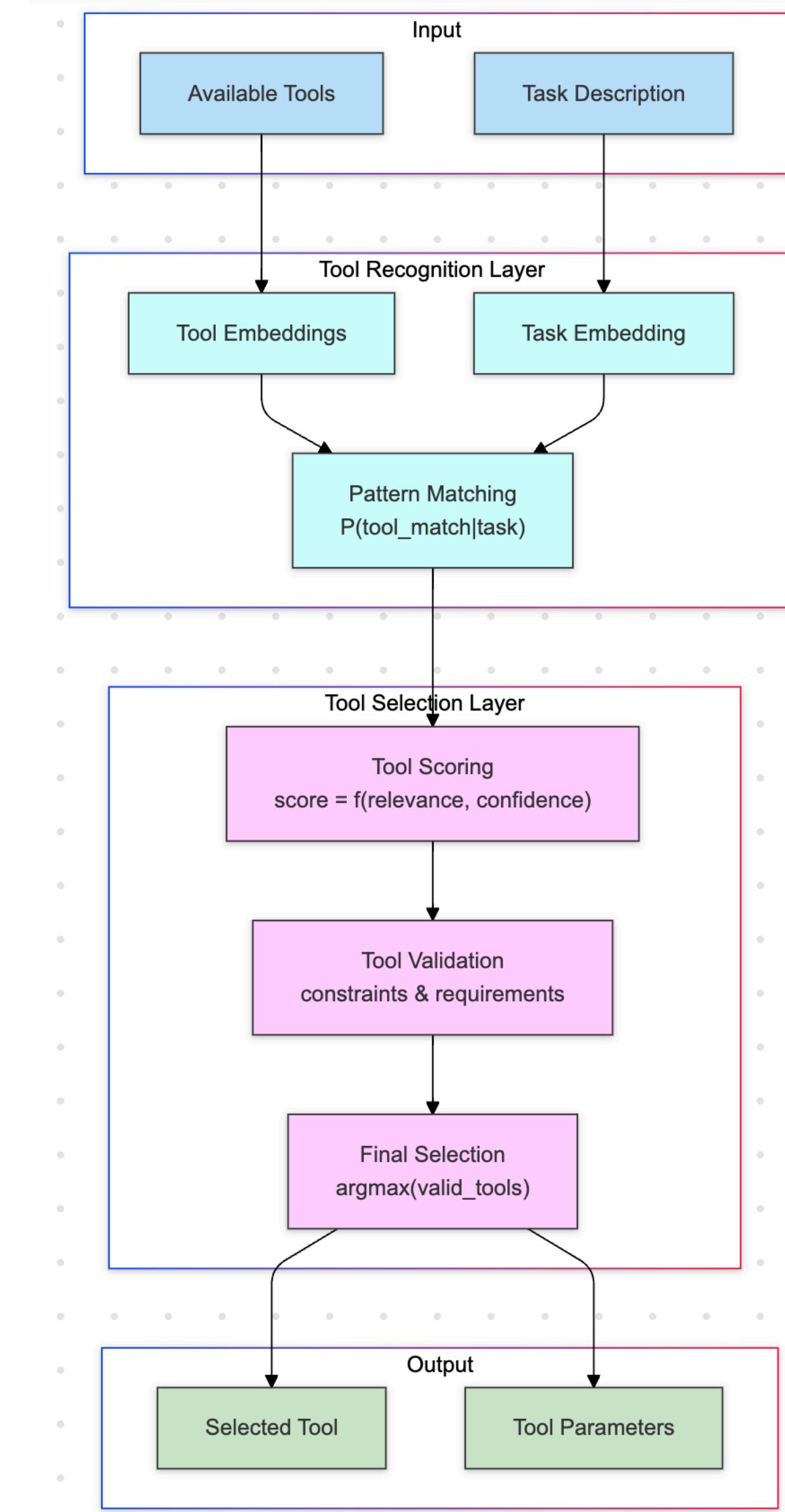
Temperature control

$$\alpha_{\text{tool}} = \text{softmax}\left(\frac{h_{\text{context}} W_Q (h_{\text{tool}} W_K)^T}{\sqrt{d_k}}\right)$$

Attention of tools

$$h_{\text{context}} = \text{LayerNorm}([h_{\text{task}}; h_{\text{tools}}])$$

Hidden states of context



Understand chain-of-thoughts (CoT)

Chain-of-thought (CoT) emerges from attention mechanism building up **a working memory** of key-value pairs from each reasoning step, while **hidden states evolve** by attending to this memory to compute next steps, creating **a computational cycle** where each step can query and build upon previous computations.

Attention-based Memory Access

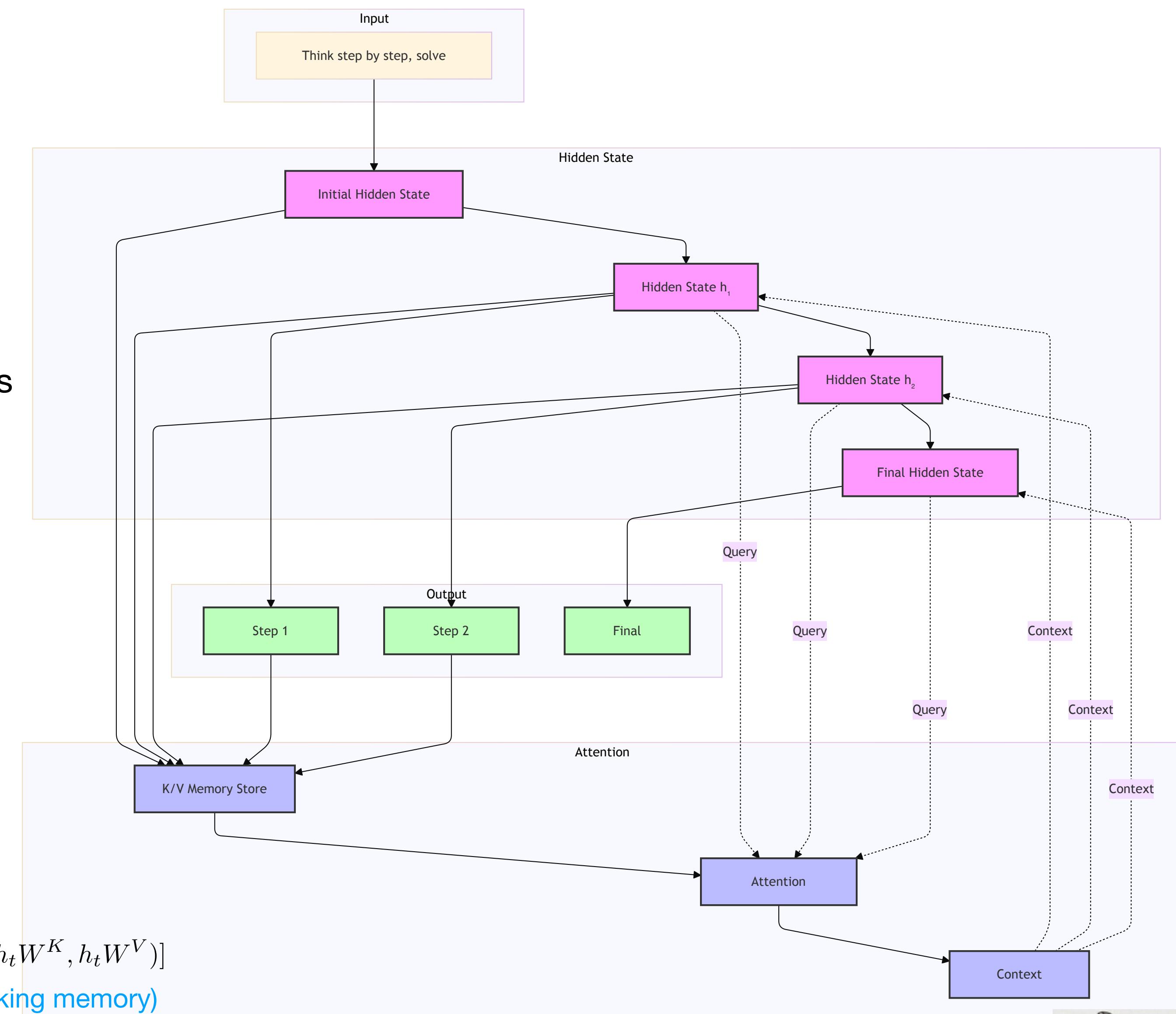
$$c_t = \text{softmax} \left(\frac{h_t W^Q (M_t^K)^T}{\sqrt{d}} \right) M_t^V$$

Hidden State Evolution

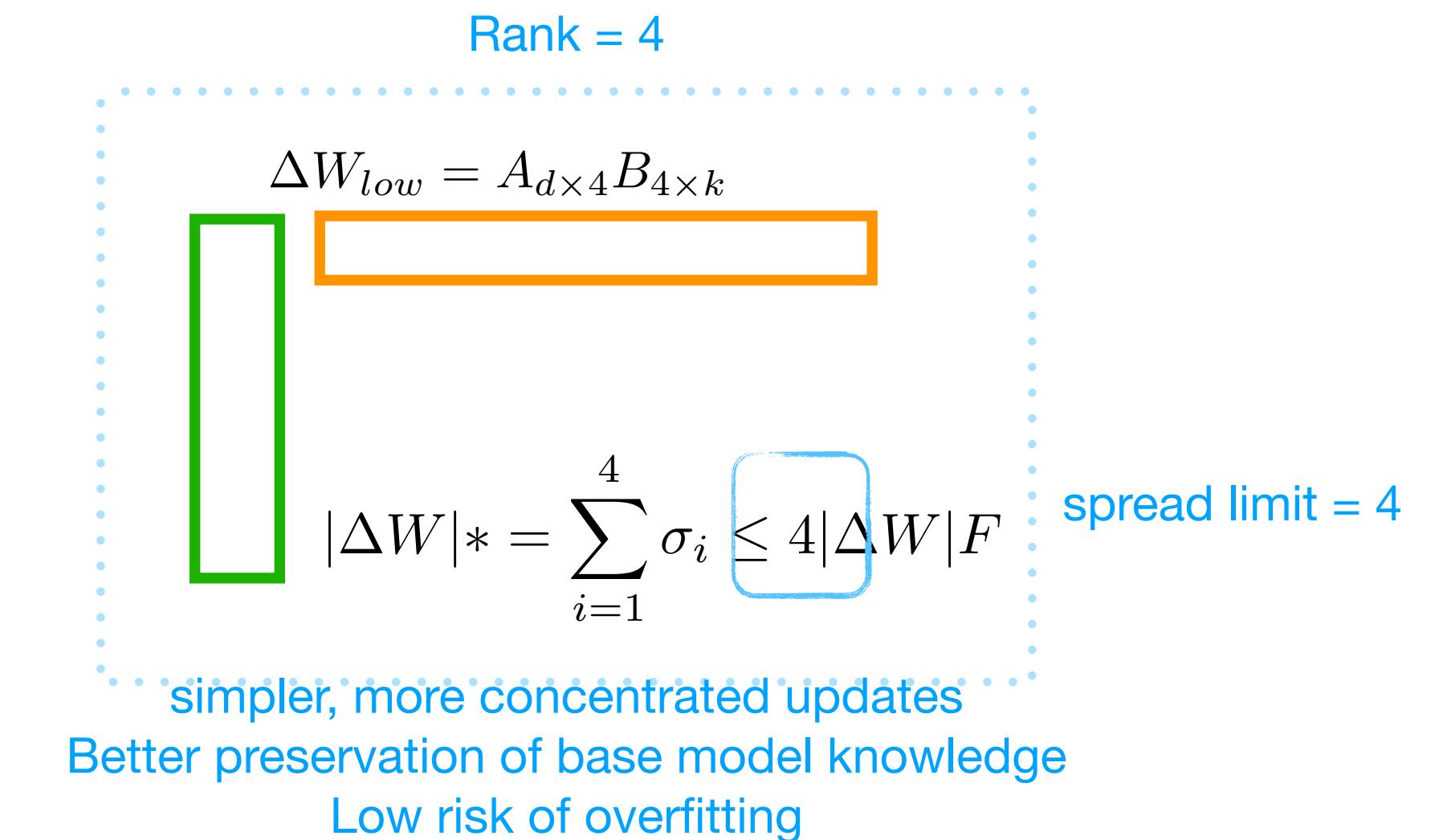
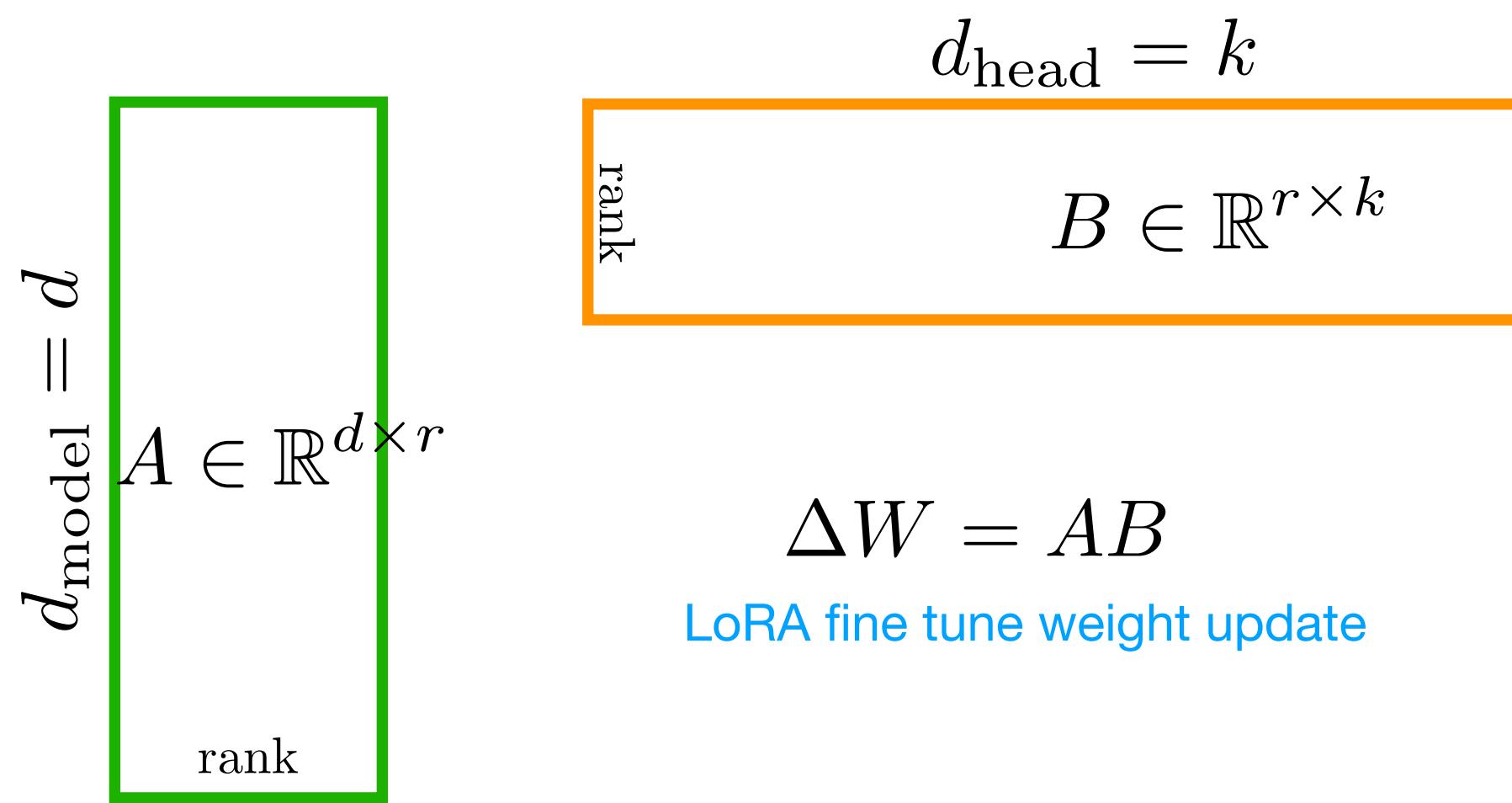
$$h_t = f(\text{attention}(h_{t-1}, [h_1, \dots, h_{t-1}]))$$

$$M_t = [K_t, V_t] = [(h_1 W^K, h_1 W^V), (h_2 W^K, h_2 W^V), \dots, (h_t W^K, h_t W^V)]$$

KV Memory Collection (working memory)



Understand LoRA ranks



Nuclear Norm (sum all singular values) $\min(d, k)$

$$|\Delta W|* = \sum_{i=1}^{\min(d, k)} \sigma_i$$

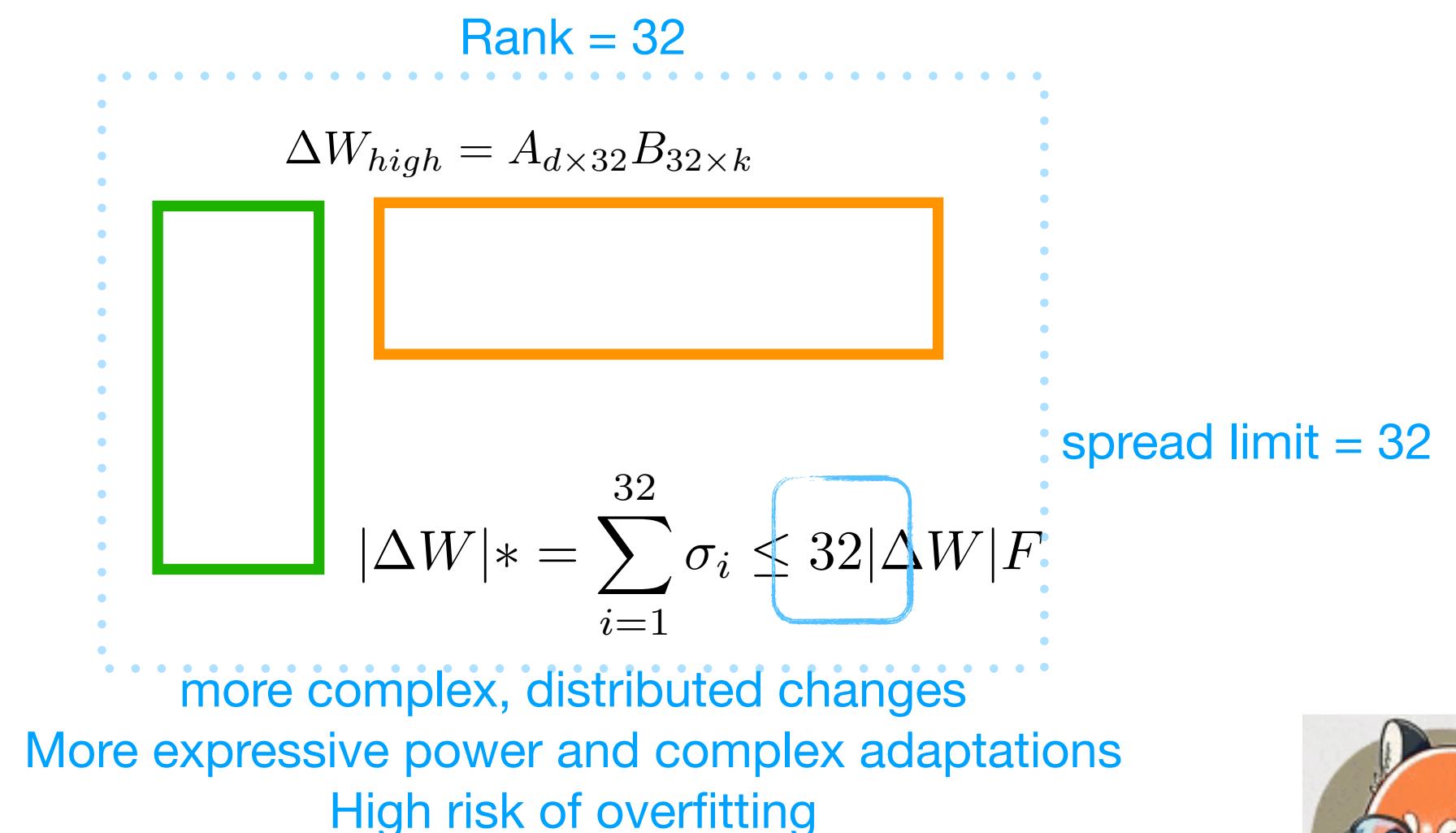
spread limit or “rank” (power of LoRA fine-tune)

Frobenius Norm as sqrt(sum of squared singular values)

$$|\Delta W|F = \sqrt{\sum_{i=1}^{\min(d, k)} \sigma_i^2}$$

$\sum_{i=1}^r \sigma_i \leq \sqrt{r} \sqrt{\sum_{i=1}^r \sigma_i^2}$

Cauchy-Schwarz Interpretation



Understand LLM inference time

Total time

Position embeddings

$$T_{pos} = \mathcal{O}(s \cdot d)$$

+

Self-attention

$$T_{attn} = \mathcal{O}(s \cdot d \cdot h)$$

Number of layers ✖

32 layers in Llama 8B

Feed-forward network

$$T_{ffn} = \mathcal{O}(s \cdot d \cdot 4d)$$

Layer norm

$$T_{ln} = \mathcal{O}(s \cdot d)$$

+

Final layer norm

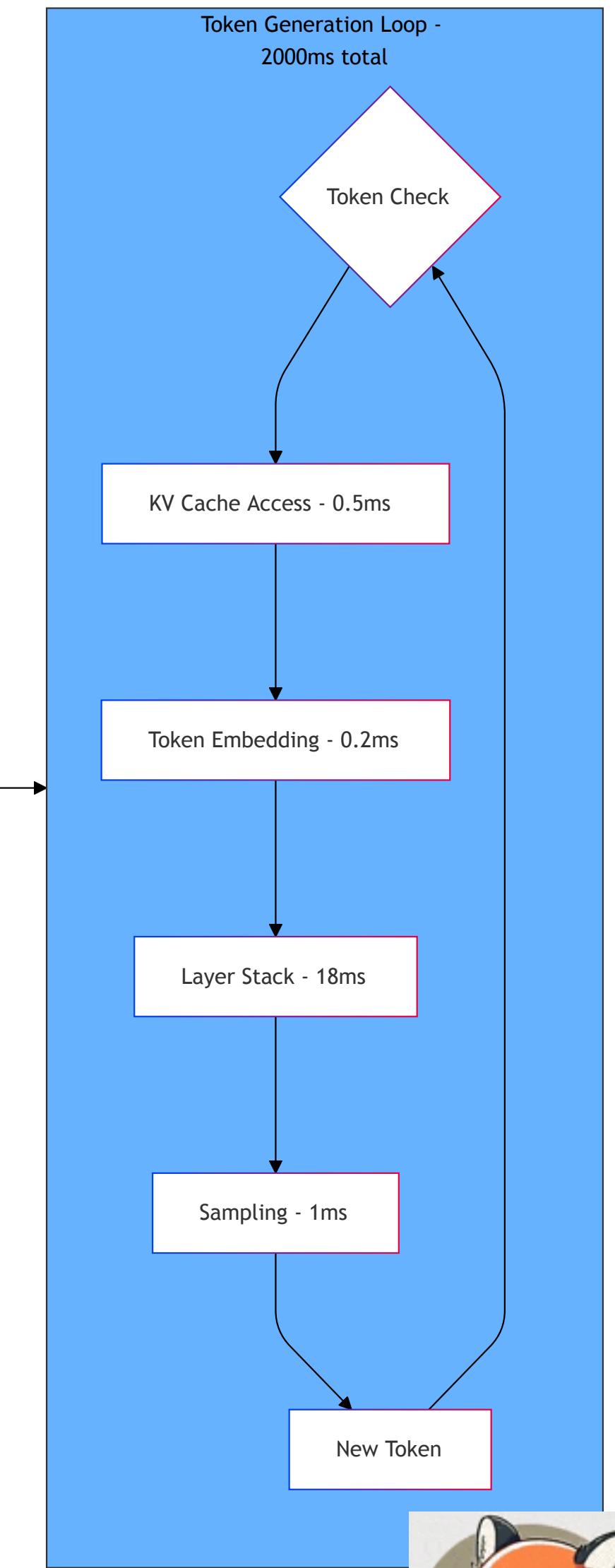
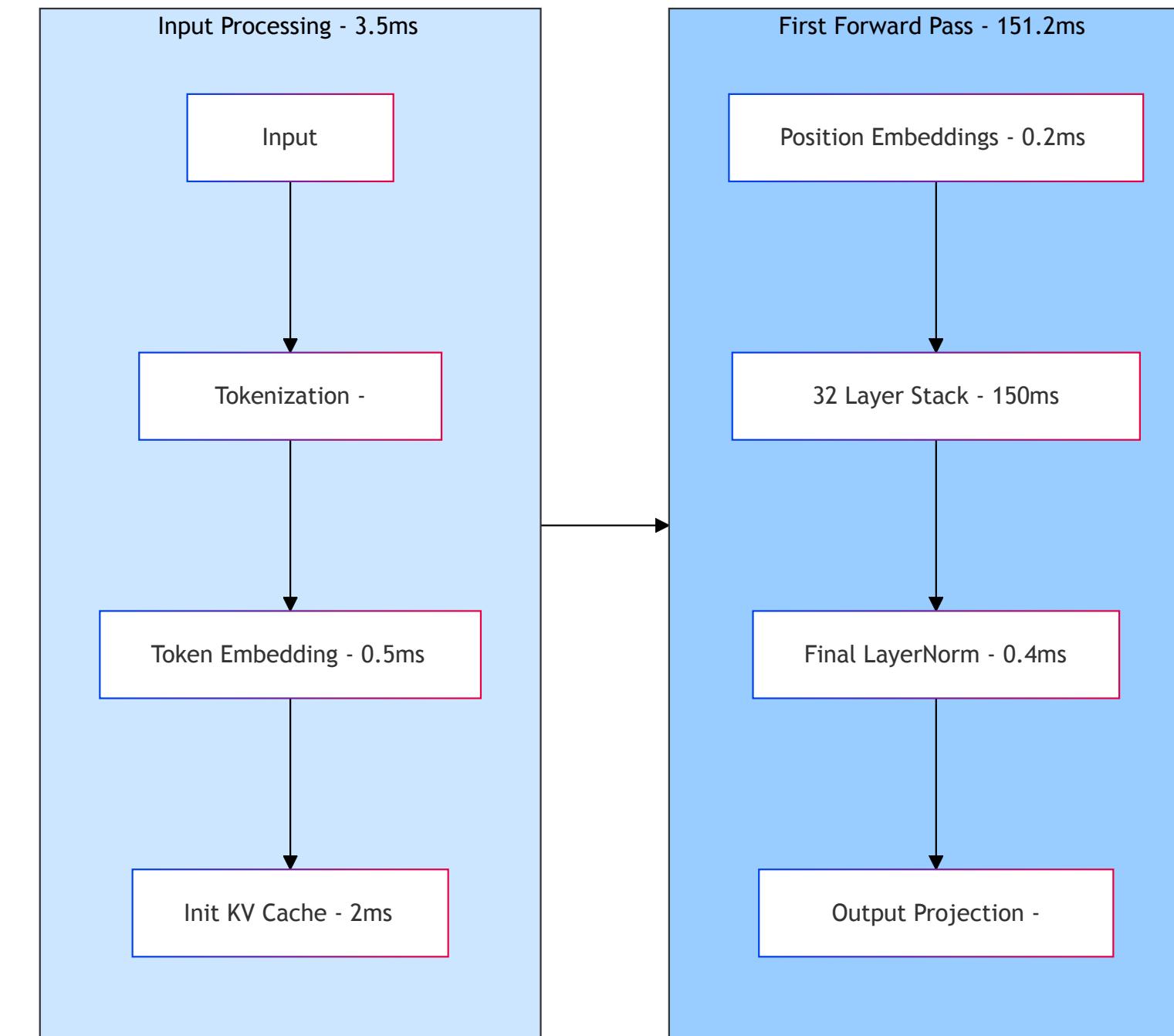
$$T_{ln} = \mathcal{O}(s \cdot d)$$

+

Output projection

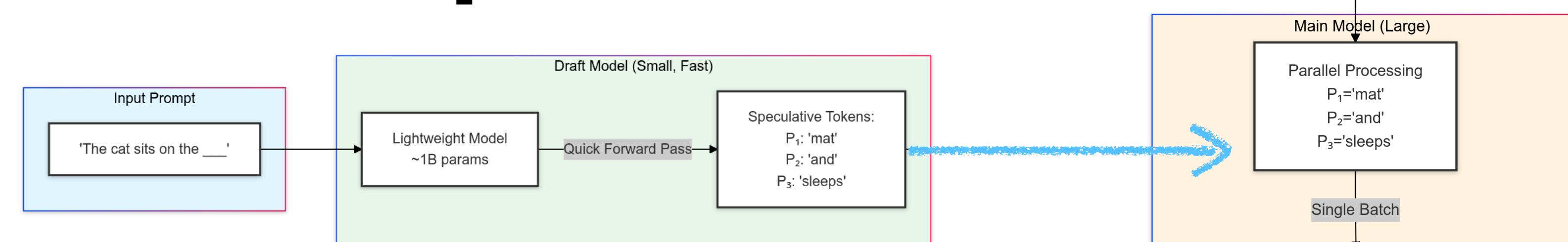
$$T_{proj} = \mathcal{O}(s \cdot d \cdot v)$$

Example: Llama 8B (FP32) on T4 GPU
512 tokens input 100 tokens output

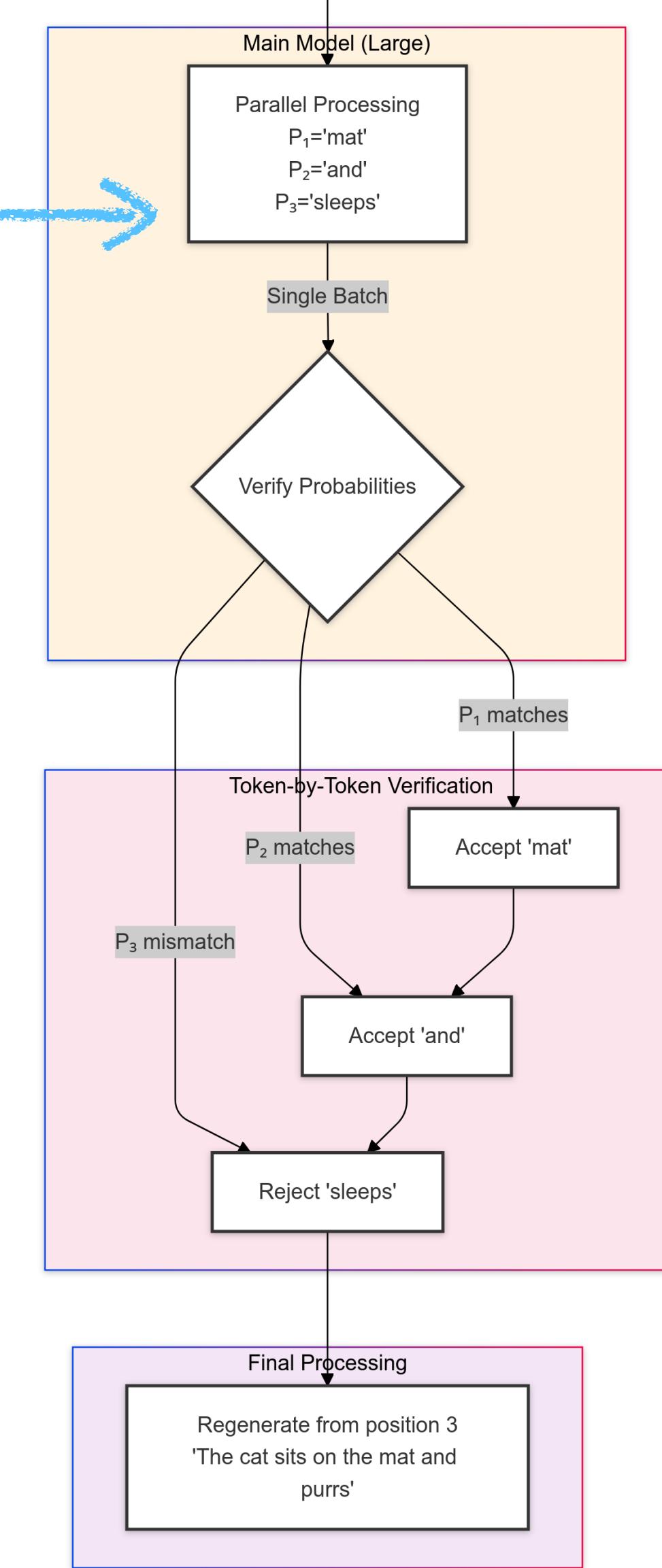
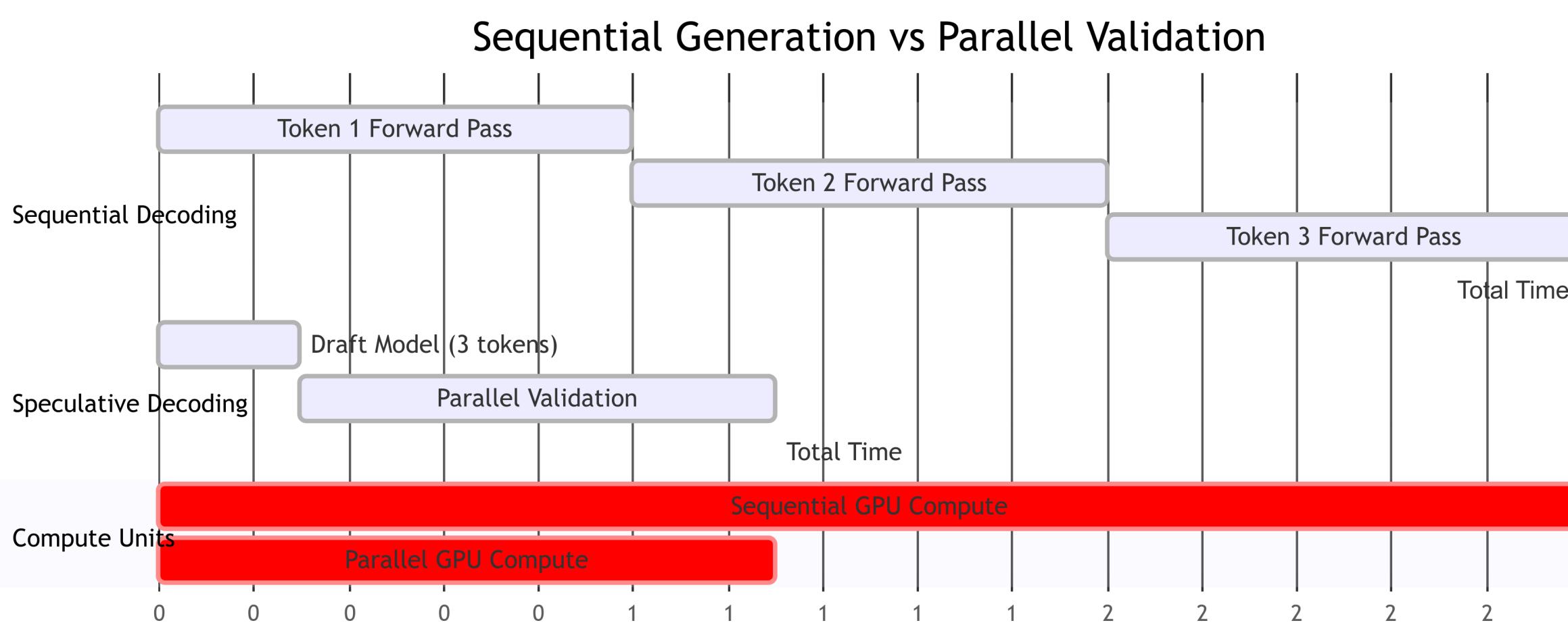


s = sequence length (e.g. 512)
 d = hidden dimension (4096)
 h = number of attention heads (32)
 v = vocabulary size (e.g. 32k)

Understand speculative decoding



Speculative decoding uses a small model to quickly guess multiple next tokens that a large model can verify in parallel, replacing sequential token generation with batch processing when predictions are correct.



Understand RoPE and lost-in-the-middle

RoPE enables **longer context** through its **geometric frequency progression** that preserves **relative positional** information via rotational embeddings

It suffers from **attention score decay** proportional to $\exp(-|m-n|\lambda)$, leading to **degraded performance for middle tokens** as they compete for attention without strong positional anchors like sequence endpoints.

attention(q,k) only needs relative positions

$$\begin{aligned} QK_{m,n}^T &= \sum_d \frac{e^{im\theta_d} x_q \cdot e^{in\theta_d} x_k}{\sqrt{d}} \\ &= \sum_d \frac{|x_q||x_k| \cos((m-n)\theta_d)}{\sqrt{d}} \end{aligned}$$

$\cos((m-n)\theta_d)$ oscillates much for large $|m-n|$ of large open angle

$\theta_d = \omega^{d/D} \theta_0$ high dimension decays faster, so decay score is

$$\text{score}(m, n) \propto \exp\left(-\frac{|m-n|}{\lambda}\right)$$

$$\begin{aligned} Q_{RoPE} &= e^{im\theta} x_q W_q \\ &= [\cos(m\theta)x_q - \sin(m\theta)y_q] W_q \\ K_{RoPE} &= e^{in\theta} x_k W_k \\ &= [\cos(n\theta)x_k - \sin(n\theta)y_k] W_k \end{aligned}$$

