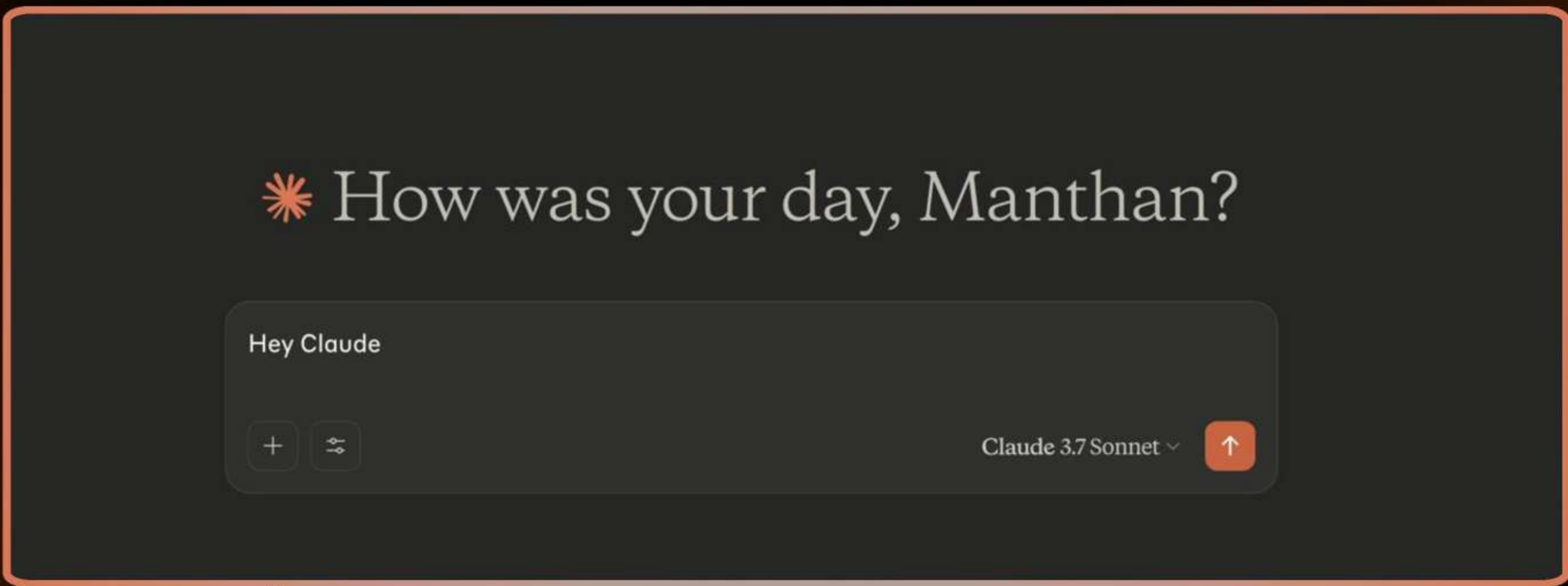
How Claude Thinks?

User Flow

Lets start from the basics.

Given an input to Claude, say "Hello Claude", let's find out what really happens next.



User input Token : "**Hello Claude**"

Text Preprocessing
(Normalization, Cleaning)

Processed Text is passed to Step 2

1. Process Text

Applied pure RL (similar to R1-Zero) to enhance reasoning skills.

2. Convert to tokens

The preprocessed text is broken down into meaningful units (tokens) based on Claude's vocabulary.

3. Create vectors

Each token is transformed into a high-dimensional vector (embedding) that captures its semantic meaning.

4. Add position info

Positional encoding is added to each token vector so the model knows where each token appears in the sequence.

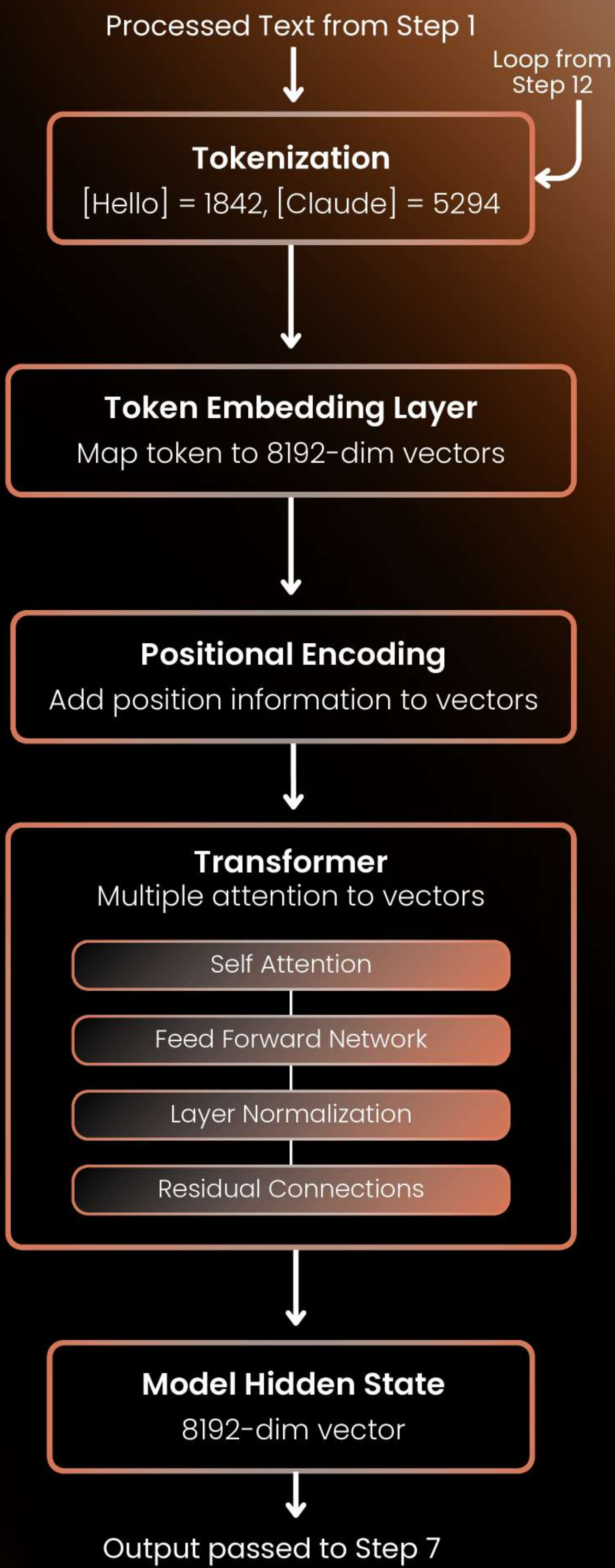
5. Transformer

The encoded tokens enter the transformer architecture as the initial input.

Goes through about 24+ layers within the transformer, before the final processing

6. Extract Features

After processing through multiple transformer layers with self-attention and feed-forward networks, the model produces a rich contextual representation.



7. Calculate vocabulary scores

The model's hidden state is projected to vocabulary-sized logits representing raw scores for each potential next token.

8. Convert to probabilities

The logits are passed through a softmax function to convert them into a proper probability distribution.

9. Token probabilities

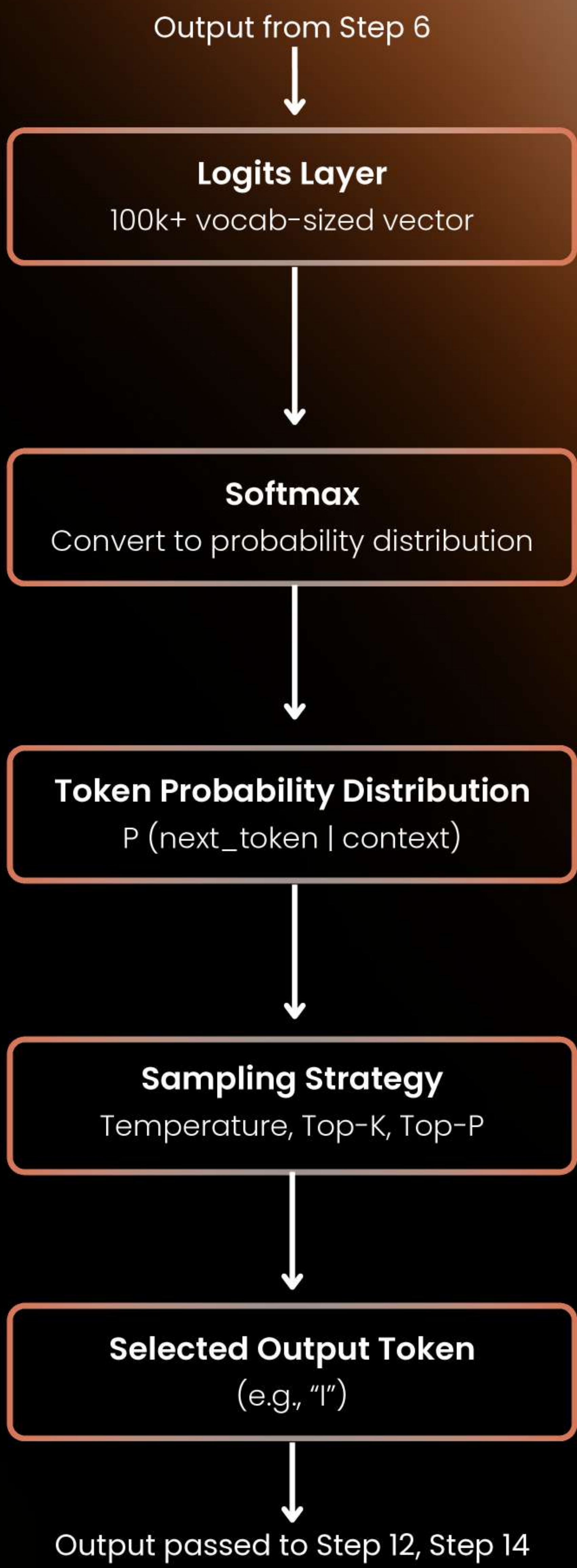
A probability is assigned to each possible next token in the vocabulary, conditioned on the context.

10. Apply Sampling

A sampling strategy (like temperature, top-k, or nucleus sampling) is applied to introduce controlled randomness.

11. Select token

Based on the sampling strategy, a specific token is selected as the next output token.



Output from Step 11

Accumulate Response Token
(t, 'am', 'Claude',...)

Updated Context Window
(T', 'am', 'Claude',...)

13. Accumulate

Generated tokens are accumulated to form the growing response text.

12. Add to context

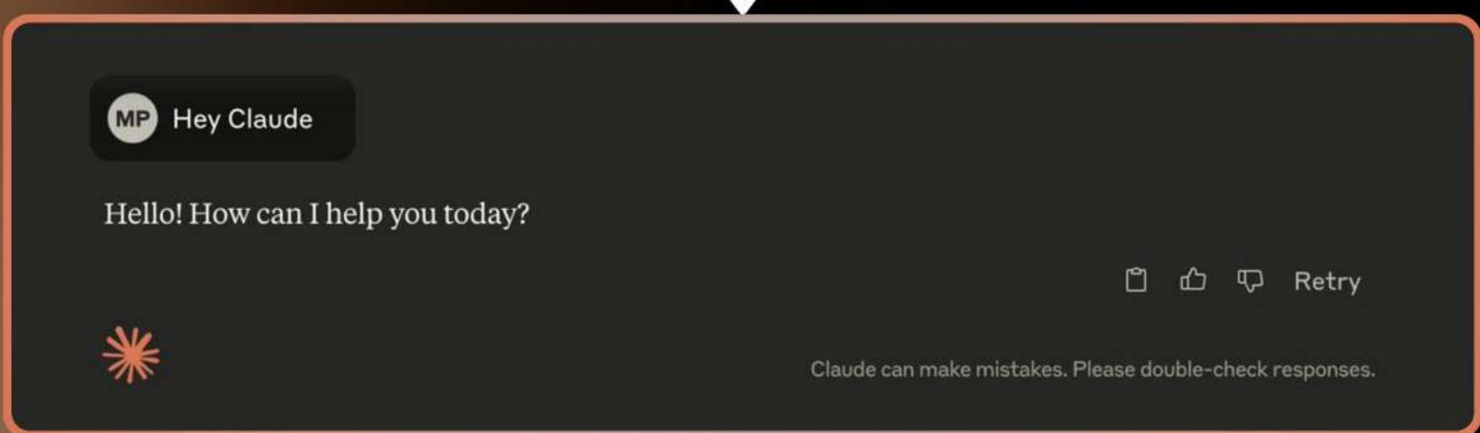
The newly generated token is appended to the existing context window for subsequent token generation.

Final Response Text
'I am Claude, how can I help you today?'

Loop for next token
The process returns to Step 2. Tokenization to generate the next token, creating an auto-regressive generation loop.

14. Complete response

The final complete response is assembled from all generated tokens.



Complete Architecture Overview

Software Engineering

Claude 3.7 Sonnet achieves state-of-the-art performance on SWE-bench Verified, which evaluates AI models' ability to solve real-world software issues.

	Claude 3.7 Sonnet *64K extended thinking*	Claude 3.7 Sonnet *No extended thinking*	Claude 3.5 Sonnet (new)	OpenAI o1[1]	OpenAI o3-mini[1] *High*	DeepSeek R1 *32K extended thinking*	Grok 3 Beta *Extended thinking*
Graduate-level reasoning *GPQA Diamond*[3]	78.2% / 84.8%	68.0%	65.0%	75.7% / 78.0%	79.7%	71.5%	80.2% / 84.6%
Agentic coding *SWE-bench Verified*[2]	—	62.3% / 70.3%	49.0%	48.9%	49.3%	49.2%	—
Agentic tool use *TAU-bench*	—	Retail 81.2%	Retail 71.5%	Retail 73.5%	—	—	—
	—	Airline 58.4%	Airline 48.8%	Airline 54.2%	—	—	—
Multilingual Q&A *MMMLU*	86.1%	83.2%	82.1%	87.7%	79.5%	—	—
Visual reasoning *MMMU (validation)*	75%	71.8%	70.4%	78.2 %	—	—	76.0% / 78.0%
Instruction-following *IFEval*	93.2%	90.8%	90.2%	—	—	83.3%	—
Math problem-solving *MATH 500*	96.2%	82.2%	78.0%	96.4%	97.9%	97.3%	—
High school math competition *AIME 2024*[3]	61.3% / 80.0%	23.3%	16.0%	79.2% / 83.3%	87.3%	79.8%	83.9% / 93.3%

Methodology: We report pass@1 averaged over several trials for most evals to reduce variance, up to an average over 16 trials for AIME and SWE-bench Verified. For our models and several others we additionally report results that benefit from "parallel test time compute" via sampling of multiple chain-of-thought sequences.

1. **OpenAI:** o1 results on TAU-bench Retail and TAU-bench Airline originally reported by OpenAI but later deleted with no alternative sources available. Note: TAU-bench results may not be comparable.
2. **SWE-bench Verified:** Claude 3.7 Sonnet scores 62.3% out of 500 problems using pass@1 with bash/editor tools plus a "thinking tool" for single-attempt patches—no additional test-time compute used. The 70.3% score uses internal scoring and custom scaffold on a reduced subset of problems. OpenAI results from o3-mini system card cover a different subset of problems with a custom scaffold. DeepSeek R1 results use the 'Agentless' framework.
3. **GPQA/AIME:** Claude 3.7 Sonnet's GPQA and AIME 2024 high scores use internal scoring with parallel test time compute, while o1 and Grok 3's high results use majority voting with N=64 samples.

Follow for more:

LEAD GEN MAN

www.leadgenman.com