ChatGPT limitations

Tom Rochette <tom.rochette@coreteks.org>

November 9, 2025 — 0e095c8d

0.1 Context

LLMs have limitations.
The goal of this study is to identify those limitations in the context of building an AGI based on ChatGPT.

0.2 Learned in this study

0.3 Things to explore

- Collaboration between ChatGPT agents

1 Overview

1.1 Execution of actions/behaviors

It does not execute actions, but it can generate text explaining what actions to take.
While this is a current limitation that is being addressed by plugins, the next logical step is to have ChatGPT execute actions.
As an example of a machine learning engineer working on a project, while ChatGPT can help with devising a plan, it cannot execute the plan.
If I already have infrastructure and I want help testing different models and hyperparameters, I still need to write the necessary prompts for ChatGPT, verify the code implements my desired intent, copy the generated code, execute it, and then analyze the results.
Furthermore, any execution of machine learning code can take from minutes to hours to execute.
In other words, while it helps me reduce the amount of time spent on certain tasks such as reflecting, generating a plan, writing code, because it does not execute those, I am still left with doing them myself.

See Produce a classification model.

As of 2025-10-30, this is not a limitation anymore.
Most of the frontier models can execute code or actions directly, interacting with the environment.
I have however yet to see LLMs being used to autonomously execute complex tasks that involve interacting with multiple systems, such as consuming issues from GitHub, understanding the requirements, validating the request with stakeholders, implementing the necessary code changes, testing them, and producing a pull request.
The agents fall short mainly in the human interaction department.

1.2 Recognizing proper responses/behaviors

While it generates text responses, it is not always clear whether those responses are appropriate and whether they will lead to the desired outcome.
For the sake of this argument, we could imagine ChatGPT returning as a task to generate a program that will execute a `while true` loop.
This indicates that we need to set a timeout on any execution of code as to prevent it from getting stuck at a

given step.
We may alternatively run steps outside of the main execution loop.
We however will need to find a way to increase the timeout when necessary if execution requires it (e.g., training a machine learning model that takes hours).
Ideally the LLM is able to inspect the code it generated and reason about whether it will terminate or not. Here we're already faced with the halting problem, where it's impossible to determine whether a given program will finish running or continue indefinitely.

1.3 Long term memory

As ChatGPT uses context to generate output, it is limited by the length of this context.
While it is possible to increase the context size, it is not possible to increase it indefinitely (yet).
Encoding a set of results in a format that is unfamiliar to ChatGPT will result in it being unable to make use of this encoded knowledge.
As such, it means that we need to produce answers that remain within the same realm of vocabulary.
Alternatively, newly generated knowledge needs to be kept in a separate system that can be queried when necessary.

1.3.1 Remembering prior conversations

It is common to ask ChatGPT to behave a certain way in order to modulate its output.
While this may work for a few prompts, ChatGPT will soon revert to its default behavior, forgetting to respect whatever constraint you might have imposed on it in prior prompts.

As of 2025-11-08 we're seeing memory become a feature offered by OpenAI and Anthropic.
Anthropic even generates a profile of the user based on prior conversations which they say is regenerated every night.

1.4 Rate limits

While only a technical limitation, if you plan on running many agents in parallel you will need to consider the rate limits imposed.
Ignoring the two first levels (free trial users and pay-as-you-go users for the first 48 hours), you will be able to send 3500 requests per minute and 90000 tokens per minute.
Rounding a little, this would mean you can run around 60 agents in parallel sending 1 request per second.
Using GPT-4 8k is limited to 40k TPM and 200 RPM while GPT-4 32k is limited to 80k TPM and 400 RPM.
At 1 request per second, you can run 3 GPT-4 8k agents or 6 GPT-4 32k agents.

This is all assuming that responses are instantaneous, which is unlikely.
According to GPT for Work, the response time of gpt-3.5-turbo is around 10 seconds while the response time of gpt-4 is around 20 seconds.

As of 2025-11-09, rate limits have changed significantly.
Rates vary per model, with tiers offering RPM going from 500 to 30k and TPM going from 30k to 180M.
Model throughput now varies based on the model and the provider, with providers such as Groq and Cerebras offering significantly faster response times due to specialized hardware.

If you are using LLMs professionally, I would say that rate limiting is not a concern unless you plan on running hundreds or thousands of agents in parallel.

2 Notes

3 See also

- Produce a classification model

4 References

- https://gptforwork.com/tools/openai-api-response-time-tracker