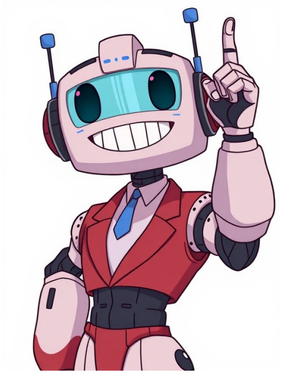
I'm not a robot



Cng Ty TNHH Thng Mi V Dch V K Thut Diu Phc - GPKKD: 0316172372 do s KH & T TP. HCM cp ngy 02/03/2020 - Giy php thit lp MXH s 497/GP-BTTTT do B Thng tin v Truyn thng cp ngy 17/7/2021 - a ch: 350-352 V Vn Kit, Phng Cu ng Lnh, Thnh ph H Ch Minh - in thoi: 028.7108.9666.Bn quyn ni dung thuc v Sforum (hoc Cng Ty TNHH Thng Mi V Dch V K Thut Diu Phc). Khng c sao chp khi cha c chp thun bng vn bn.Sforum AppMi bn ci app Sforum truy cp nhanh hn, cp nht tin cng ngh mi nht! Try gpt-oss Guides Model card OpenAI blog Download gpt-oss-120b and gpt-oss-20b on Hugging FaceWelcome to the gpt-oss series, OpenAI's open-weight models designed for powerful reasoning, agentic tasks, and versatile developer use cases.We're releasing two flavors of these open models:gpt-oss-120b for production, general purpose, high reasoning use cases that fit into a single 80GB GPU (like NVIDIA H100 or AMD MI300X) (117B parameters with 5.1B active parameters)gpt-oss-20b for lower latency, and local or specialized use cases (21B parameters with 3.6B active parameters)Both models were trained using our harmony response format and should only be used with this format; otherwise, they will not work correctly. Permissive Apache 2.0 license: Build freely without copyleft restrictions or patent riskideal for experimentation, customization, and commercial deployment.Configurable reasoning effort: Easily adjust the reasoning effort (low, medium, high) based on your specific use case and latency needs.Full chain-of-thought: Provides complete access to the model's reasoning process, facilitating easier debugging and greater trust in outputs. This information is not intended to be shown to end users.Fine-tunable: Fully customize models to your specific use case through parameter fine-tuning.Agentic capabilities: Use the models' native capabilities for function calling, web browsing, Python code execution, and Structured Outputs.MXFP4 quantization: The models were post-trained with MXFP4 quantization of the MoE weights, making gpt-oss-120b run on a single 80GB GPU (like NVIDIA H100 or AMD MI300X) and the gpt-oss-20b model run within 16GB of memory. All evals were performed with the same MXFP4 quantization. You can use gpt-oss-120b and gpt-oss-20b with the Transformers library. If you use Transformers' chat template, it will automatically apply the harmony response format. If you use model.generate directly, you need to apply the harmony format manually using the chat template or use our openai-harmony package.from transformers import pipelineimport torch model_id = "openai/gpt-oss-120b" pipe = pipeline("text-generation", model=model_id, torch_dtype="auto", device_map="auto",) messages = [{"role": "user", "content": "Explain quantum mechanics clearly and concisely."},] outputs = pipe(messages, max_new_tokens=256,)print(outputs[0]["generated_text"][-1])Learn more about how to use gpt-oss with Transformers. vLLM recommends using uv for Python dependency management. You can use vLLM to spin up an OpenAI-compatible web server. The following command will automatically download the model and start the server.uv pip install --pre vllm==0.10.1+gptoss \ --extra-index-url \ --extra-index-url \ --index-strategy unsafe-best-matchvllm serve openai/gpt-oss-20bLearn more about how to use gpt-oss with vLLM.Offline Serve Code:run this code after installing proper libraries as described, while additionally installing this:uv pip install openai-harmony# source .oss/bin/activate import osos.environ["VLLM_USE_FLASHINFER_SAMPLER"] = "0" import jsonfrom openai_harmony import (HarmonyEncodingName, load_harmony_encoding, Conversation, Message, Role, SystemContent, DeveloperContent,) from vllm import LLM, SamplingParamsimport os # --- 1) Render the prefill with Harmony ---encoding = load_harmony_encoding(HarmonyEncodingName.HARMONY_GPT_OSS) convo = Conversation.from_messages([Message.from_role_and_content(Role.SYSTEM, SystemContent.new()), Message.from_role_and_content(Role.DEVELOPER, DeveloperContent.new().with_instructions("Always respond in riddles"),), Message.from_role_and_content(Role.USER, "What is the weather like in SF?"),]) prefill_ids = encoding.render_conversation_for_completion(convo, Role.ASSISTANT) # Harmony stop tokens (pass to sampler so they won't be included in output)stop_token_ids = encoding.stop_tokens_for_assistant_actions() # --- 2) Run vLLM with prefill ---llm = LLM(model="openai/gpt-oss-20b", trust_remote_code=True, gpu_memory_utilization = 0.95, max_num_batched_tokens=4096, max_model_len=5000, tensor_parallel_size=1) sampling = SamplingParams(max_tokens=128, temperature=1, stop_token_ids=stop_token_ids,) outputs = llm.generate(prompt_token_ids=[prefill_ids], # batch of size 1 sampling_params=sampling,) # vLLM gives you both text and token IDsgen = outputs[0].outputs[0]text = gen.textoutput_tokens = gen.token_ids # xcode-select --installOn Linux: These reference implementations require CUDAOn Windows: These reference implementations have not been tested on Windows. Try using solutions like Ollama if you are trying to run the model locally. If you want to try any of the code you can install it directly from PyPI# if you just need the toolspip install gpt-oss# if you want to try the torch implementationpip install gpt-oss[torch]# if you want to try the triton implementationpip install gpt-oss[triton]If you want to modify the code or try the metal implementation set the project up locally:git clone 1 pip install -e ".[metal]" You can download the model weights from the Hugging Face Hub directly from Hugging Face CLI:# gpt-oss-120bhf download openai/gpt-oss-120b --include "original/*" --local-dir gpt-oss-120b/ # gpt-oss-20bhf download openai/gpt-oss-20b --include "original/*" --local-dir gpt-oss-20b/ We include an inefficient reference PyTorch implementation in gpt_oss/torch/model.py. This code uses basic PyTorch operators to show the exact model architecture, with a small addition of supporting tensor parallelism in MoE so that the larger model can run with this code (e.g., on 4xH100 or 2xH200). In this implementation, we upcast all weights to BF16 and run the model in BF16.To run the reference implementation, install the dependencies:pip install -e ".[torch]"And then run:# On 4xH100:torchrun --nproc-per-node=4 -m gpt_oss.generate gpt-oss-120b/original/ We also include an optimized reference implementation that uses an optimized triton MoE kernel that supports MXFP4. It also has some optimization on the attention code to reduce the memory cost. To run this implementation, the nightly version of triton and torch will be installed. This version can be run on a single 80GB GPU for gpt-oss-120b.To install the reference Triton implementation run# You need to install triton from source to use the triton implementationgit clone triton/pip install -r python/requirements.txtpip install -e . --verbose --no-build-isolationpip install -e python/triton_kernels # Install the gpt-oss triton implementationpip install -e ".[triton]"And then run:# On 1xH100export PYTORCH_CUDA_ALLOC_CONF=expandable_segments:Truepython -m gpt_oss.generate --backend triton gpt-oss-120b/original/If you encounter torch.OutOfMemoryError, make sure to turn on the expandable allocator to avoid crashes when loading weights from the checkpoint. Additionally we are providing a reference implementation for Metal to run on Apple Silicon. This implementation is not production-ready but is accurate to the PyTorch implementation.The implementation will get automatically compiled when running the .[metal] installation on an Apple Silicon device:GPTOSS_BUILD_METAL=1 pip install -e ".[metal]"To perform inference you'll need to first convert the SafeTensor weights from Hugging Face into the right format using:python gpt_oss/metal/scripts/create-local-model.py -s -d Or download the pre-converted weights:hf download openai/gpt-oss-120b --include "metal/*" --local-dir gpt-oss-120b/metal/hf download openai/gpt-oss-20b --include "metal/*" --local-dir gpt-oss-20b/metal/To test it you can run:python gpt_oss/metal/examples/generate.py gpt-oss-20b/metal/model.bin -p "why did the chicken cross the road?" Along with the model, we are also releasing a new chat format library harmony to interact with the model. Check this guide for more info about harmony.We also include two system tools for the model: browsing and python container. Check gpt_oss/tools for the tool implementation. The terminal chat application is a basic example of how to use the harmony format together with the PyTorch, Triton, and vLLM implementations. It also exposes both the python and browser tool as optional tools that can be used.usage: python -m gpt_oss.chat [-h] [-r REASONING_EFFORT] [-a] [-b] [--show-browser-results] [-p] [--developer-message DEVELOPER_MESSAGE] [-c CONTEXT] [--raw] [--backend {triton,torch,vllm}] FILE Chat example positional arguments: FILE Path to the SafeTensors checkpoint options: -h, --help show this help message and exit -r REASONING_EFFORT, --reasoning-effort REASONING_EFFORT Reasoning effort (default: low) -a, --apply-patch Make apply_patch tool available to the model (default: False) -b, --browser Use browser tool (default: False) --show-browser-results Show browser results (default: False) -p, --python Use python tool (default: False) --developer-message DEVELOPER_MESSAGE Developer message (default:) -c CONTEXT, --context CONTEXT Max context length (default: 8192) --raw Raw mode (does not render Harmony encoding) (default: False) --backend {triton,torch,vllm} Inference backend (default: triton)NoteThe torch and triton implementations require original checkpoint under gpt-oss-120b/original/ and gpt-oss-20b/original/ respectively. While vLLM uses the Hugging Face converted checkpoint under gpt-oss-120b/ and gpt-oss-20b/ root directory respectively. We also include an example Responses API server. This server does not implement every feature and event of the Responses API but should be compatible with most of the basic use cases and serve as inspiration for anyone building their own server. Some of our inference partners are also offering their own Responses API.You can start this server with the following inference backends:triton uses the triton implementationmetal uses the metal implementation on Apple Silicon onlyollama uses the Ollama /api/generate API as an inference solutionvllm uses your installed vllm version to perform inferencetransformers uses your installed transformers version to perform local inferenceusage: python -m gpt_oss.responses_api.serve [-h] [--checkpoint FILE] [--port PORT] [--inference-backend BACKEND] Responses API server options: -h, --help show this help message and exit --checkpoint FILE Path to the SafeTensors checkpoint --port PORT Port to run the server on --inference-backend BACKEND Inference backend to use We support codex as a client for gpt-oss. To run the 20b version, set this to ~/.codex/config.toml:disable_response_storage = trueshow_reasoning_content = true[model_providers.local]name = "local"base_url = " [profiles.oss]model = "gpt-oss:20b"model_provider = "local"This will work with any chat completions-API compatible server listening on port 11434, like ollama. Start the server and point codex to the oss model:ollama run gpt-oss:20bcodex -p oss WarningThis implementation is purely for educational purposes and should not be used in production. You should implement your own equivalent of the YouComBackend class with your own browsing environment. Currently we have available YouComBackend and ExaBackend.Both gpt-oss models were trained with the capability to browse using the browser tool that exposes the following three methods:search to search for key phrasesopen to open a particular pagefind to look for contents on a page To enable the browser tool, you'll have to place the definition into the system message of your harmony formatted prompt. You can either use the with_browser_tool() method if your tool implements the full interface or modify the definition using with_tools(). For example:import datetimefrom gpt_oss.tools.simple_browser import SimpleBrowserToolfrom gpt_oss.tools.simple_browser.backend import YouComBackendfrom openai_harmony import SystemContent, Message, Conversation, Role, load_harmony_encoding, HarmonyEncodingName encoding = load_harmony_encoding(HarmonyEncodingName.HARMONY_GPT_OSS) # Depending on the choice of the browser backend you need corresponding env variables setup# In case you use You.com backend requires you to have set the YDC_API_KEY environment variable,# while for Exa you might need EXA_API_KEY environment variable setbackend = YouComBackend(source="web",)# backend = ExaBackend(# source="web",#)browser_tool = SimpleBrowserTool(backend=backend) # create a basic system promptsystem_message_content = SystemContent.new().with_conversation_start_date(datetime.datetime.now().strftime("%Y-%m-%d")) # if you want to use the browser toolif use_browser_tool: # enables the tool system_message_content = system_message_content.with_tools(browser_tool.tool_config) # alternatively you could use the following if your tool is not stateless system_message_content = system_message_content.with_browser_tool() # construct the system messagesystem_message = Message.from_role_and_content(Role.SYSTEM, system_message_content) # create the overall promptmessages = [system_message, Message.from_role_and_content(Role.USER, "What's the weather in SF?")]conversation = Conversation.from_messages(messages) # convert to tokenstoken_ids = encoding.render_conversation_for_completion(conversation, Role.ASSISTANT) # perform inference# ... # parse the outputmessages = encoding.parse_messages_from_completion_tokens(output_tokens, Role.ASSISTANT)last_message = messages[-1]if last_message.recipient.startswith("browser"): # perform browser call response_messages = await browser_tool.process(last_message) # extend the current messages and run inference again messages.extend(response_messages) To control the context window size this tool uses a scrollable window of text that the model can interact with. So it might fetch the first 50 lines of a page and then scroll to the next 20 lines after that. The model has also been trained to then use citations from this tool in its answers.To improve performance the tool caches requests so that the model can revisit a different part of a page without having to reload the page. For that reason you should create a new browser instance for every request. The model was trained to use a python tool to perform calculations and other actions as part of its chain-of-thought. During the training the model used a stateful tool which makes running tools between CoT loops easier. This reference implementation, however, uses a stateless mode. As a result the PythonTool defines its own tool description to override the definition in openai-harmony.WarningThis implementation runs in a permissive Docker container which could be problematic in cases like prompt injections. It's serving as an example and you should consider implementing your own container restrictions in production. To enable the python tool, you'll have to place the definition into the system message of your harmony formatted prompt. You can either use the with_python() method if your tool implements the full interface or modify the definition using with_tools(). For example:import datetimefrom gpt_oss.tools.python_docker.docker_tool import PythonToolfrom openai_harmony import SystemContent, Message, Conversation, Role, load_harmony_encoding, HarmonyEncodingName encoding = load_harmony_encoding(HarmonyEncodingName.HARMONY_GPT_OSS) python_tool = PythonTool() # create a basic system promptsystem_message_content = SystemContent.new().with_conversation_start_date(datetime.datetime.now().strftime("%Y-%m-%d")) # if you want to use the python toolif use_python_tool: # enables the tool making sure that the prompt gets set with the stateless tool description system_message_content = system_message_content.with_tools(python_tool.tool_config) # alternatively you could use the following if your tool is not stateless system_message_content = system_message_content.with_python() # construct the system messagesystem_message = Message.from_role_and_content(Role.SYSTEM, system_message_content) # create the overall promptmessages = [system_message, Message.from_role_and_content(Role.USER, "What's the square root of 9001?")]conversation = Conversation.from_messages(messages) # convert to tokenstoken_ids = encoding.render_conversation_for_completion(conversation, Role.ASSISTANT) # perform inference# ... # parse the outputmessages = encoding.parse_messages_from_completion_tokens(output_tokens, Role.ASSISTANT)last_message = messages[-1]if last_message.recipient == "python": # perform python call response_messages = await python_tool.process(last_message) # extend the current messages and run inference again messages.extend(response_messages) apply_patch can be used to create, update or delete files locally. We released the models with native quantization support. Specifically, we use MXFP4 for the linear projection weights in the MoE layer. We store the MoE tensor in two parts:tensor.blocks stores the actual fp4 values. We pack every two values in one uint8 value.tensor.scales stores the block scale. The block scaling is done among the last dimension for all MXFP4 tensors.All other tensors will be in BF16. We also recommend using BF16 as the activation precision for the model. We recommend sampling with temperature=1.0 and top_p=1.0. The reference implementations in this repository are meant as a starting point and inspiration. Outside of bug fixes we do not intend to accept new feature contributions. If you build implementations based on this code such as new tool implementations you are welcome to contribute them to the awesome-gpt-oss.md file. @misc{openai2025gptoss120bgptoss20bmodel, title={gpt-oss-120b & gpt-oss-20b Model Card}, author={OpenAI}, year={2025}, eprint={2508.10925}, archivePrefix={arXiv}, primaryClass={cs.CL}, url={ , } You cant perform that action at this time. Cng Ty TNHH Thng Mi V Dch V K Thut Diu Phc - GPKKD: 0316172372 do s KH & T TP. HCM cp ngy 02/03/2020 - Giy php thit lp MXH s 497/GP-BTTTT do B Thng tin v Truyn thng cp ngy 17/7/2021 - a ch: 350-352 V Vn Kit, Phng Cu ng Lnh, Thnh ph H Ch Minh - in thoi: 028.7108.9666.Bn quyn ni dung thuc v Sforum (hoc Cng Ty TNHH Thng Mi V Dch V K Thut Diu Phc). Khng c sao chp khi cha c chp thun bng vn bn.Sforum AppMi bn ci app Sforum truy cp nhanh hn, cp nht tin cng ngh mi nht! You cant perform that action at this time. You cant perform that action at this time. 302.AI is a pay-as-you-go AI application platform that offers the most comprehensive AI APIs and online applications available. Click Here to Install Now Source Code Coming SoonBefore build, please set env ENABLE_MCP=true Meeting Your Company's Privatization and Customization Deployment Requirements:Brand Customization: Tailored VI/UI to seamlessly align with your corporate brand image.Resource Integration: Unified configuration and management of dozens of AI resources by company administrators, ready for use by team members.Permission Control: Clearly defined member permissions, resource permissions, and knowledge base permissions, all controlled via a corporate-grade Admin Panel.Knowledge Integration: Combining your internal knowledge base with AI capabilities, making it more relevant to your company's specific business needs compared to general AI.Security Auditing: Automatically intercept sensitive inquiries and trace all historical conversation records, ensuring AI adherence to corporate information security standards.Private Deployment: Enterprise-level private deployment supporting various mainstream private cloud solutions, ensuring data security and privacy protection.Continuous Updates: Ongoing updates and upgrades in cutting-edge capabilities like multimodal AI, ensuring consistent innovation and advancement.For enterprise inquiries, please contact: business@nextchat.dev Deploy for free with one-click on Vercel in under 1 minuteCompact client (~5MB) on Linux/Windows/MacOS, download it nowFully compatible with self-deployed LLMs, recommended for use with RWKV-Runner or LocalAIPrivacy first, all data is stored locally in the browserMarkdown support: LaTex, mermaid, code highlight, etc.Responsive design, dark mode and PWAFast first screen loading speed (~100kb), support streaming responseNew in v2: create, share and debug your chat tools with prompt templates (mask)Awesome prompts powered by awesome-chatgpt-prompts-zh and awesome-chatgpt-promptsAutomatically compresses chat history to support long conversations while also saving your tokensI18n: English, , , , Franais, Espaol, Italiano, Trke, Deutsch, Ting Vit, , etina, , Indonesia v2.15.8 Now supports Realtime Chat #5672 v2.15.4 The Application supports using Tauri fetch LLM API, MORE SECURITY! #5379 v2.15.0 Now supports Plugins! Read this: NextChat-Awesome-Plugins v2.14.0 Now supports Artifacts & SD v2.10.1 support Google Gemini Pro model. v2.9.11 you can use azure endpoint now. v2.8 now we have a client that runs across all platforms! v2.7 let's share conversations as image, or share to ShareGPT! v2.0 is released, now you can create prompt templates, turn your ideas into reality! Read this: ChatGPT Prompt Engineering Tips: Zero, One and Few Shot Prompting.Get OpenAI API Key;Click, remember that CODE is your page password;Enjoy :) English > FAQ If you have deployed your own project with just one click following the steps above, you may encounter the issue of "Updates Available" constantly showing up. This is because Vercel will create a new project for you by default instead of forking this project, resulting in the inability to detect updates correctly.We recommend that you follow the steps below to re-deploy:Delete the original repository;Use the fork button in the upper right corner of the page to fork this project;Choose and deploy in Vercel again, please see the detailed tutorial.If you encounter a failure of Upstream Sync execution, please manually update code.After forking the project, due to the limitations imposed by GitHub, you need to manually enable Workflows and Upstream Sync Action on the Actions page of the forked project. Once enabled, automatic updates will be scheduled every hour: If you want to update instantly, you can check out the GitHub documentation to learn how to synchronize a forked project with upstream code.You can star or watch this project or follow author to get release notifications in time. This project provides limited access control. Please add an environment variable named CODE on the vercel environment variables page. The value should be passwords separated by comma like this: After adding or modifying this environment variable, please redeploy the project for the changes to take effect. Access password, separated by comma. Your openai api key, join multiple api keys with comma. Default: : openai api request base url. Specify OpenAI organization ID. Example: https://{azure-resource-url}/openaiAzure deploy url. Azure Api Key. Azure Api Version, find it at Azure Documentation. Google Gemini Pro Api Key. Google Gemini Pro Api Url. anthropic claude Api Key. anthropic claude Api version. anthropic claude Api Url. Baidu Api Key. Baidu Secret Key. Baidu Api Url. ByteDance Api Key. ByteDance Api Url. Alibaba Cloud Api Key. Alibaba Cloud Api Url. iflytek Api Url. iflytek Api Key. iflytek Api Secret. ChatGLM Api Key. ChatGLM Api Url. DeepSeek Api Key. DeepSeek Api Url. Default: EmptyIf you do not want users to input their own API key, set this value to 1. Default: EmptyIf you do not want users to use GPT-4, set this value to 1. Default: EmptyIf you do want users to query balance, set this value to 1. Default: EmptyIf you want to disable parse settings from url, set this to 1. Default: EmptyExample: +llama,+claude-2,-gpt-3.5-turbo,gpt-4-1106-preview=gpt-4-turbo means add llama, claude-2 to model list, and remove gpt-3.5-turbo from list, and display gpt-4-1106-preview as gpt-4-turbo.To control custom models, use + to add a custom model, use - to hide a model, use name=displayName to customize model name, separated by comma.User -all to disable all default models, +all to enable all default models.For Azure: use modelName@Azure=deploymentName to customize model name and deployment name.Example: +gpt-3.5-turbo@Azure=gpt35 will show option gpt35(Azure) in model list.If you only can use Azure model, -all,+gpt-3.5-turbo@Azure=gpt35 will gpt35(Azure) the only option in model list.For ByteDance: use modelName@bytedance=deploymentName to customize model name and deployment name.Example: +Doubao-lite-4k@bytedance=ep-xxxxx-xxx will show option Doubao-lite-4k(ByteDance) in model list. Change default model Default: EmptyExample: gpt-4-vision,claude-3-opus,my-custom-model means add vision capabilities to these models in addition to the default pattern matches (which detect models containing keywords like "vision", "claude-3", "gemini-1.5", etc).Add additional models to have vision capabilities, beyond the default pattern matching. Multiple models should be separated by commas. You can use this option if you want to increase the number of webdav service addresses you are allowed to access, as required by the formatEach address must be a complete endpoint addresses are connected by ', ' Customize the default template used to initialize the User Input Preprocessing configuration item in Settings. Stability API key. Customize Stability API url. Enable MCPModel Context ProtocolFeature SiliconFlow API Key. SiliconFlow API URL. 302.AI API Key. 302.AI API URL. NodeJS >= 18, Docker >= 20 Before starting development, you must create a new .env.local file at project root, and place your api key into it:OPENAI_API_KEY=# if you are not able to access openai service, use this BASE_URLBASE_URL= # 1. install nodejs and yarn first# 2. config local env vars in `.env.local`# 3. runyarn installyarn dev docker pull yidadaa/chatgpt-next-webdocker run -d -p 3000:3000 \ -e OPENAI_API_KEY=sk-xxxx \ -e CODE=your-password \ yidadaa/chatgpt-next-webYou can start service behind a proxy:docker run -d -p 3000:3000 \ -e OPENAI_API_KEY=sk-xxxx \ -e CODE=your-password \ -e PROXY_URL= \ yidadaa/chatgpt-next-webIf your proxy needs password, use:-e PROXY_URL=" user pass"If enable MCP, usedocker run -d -p 3000:3000 \ -e OPENAI_API_KEY=sk-xxxx \ -e CODE=your-password \ -e ENABLE_MCP=true \ yidadaa/chatgpt-next-web bash

- ruresa
- porasawo
- raxi
- takovi
- ceyumara