CREATE CLOUD NATIVE AGENTS AND EXTENSIONS FOR LLMS

Vivian Hu
@alabulei

Michael Yuan
@juntao

Current tech stacks for LLM Apps

- LLM inference – Pytorch + Python
- LLM agent/extension – Langchain + Python

The Challenges for LLM apps in Cloud Native

The tech stack is too heavy with complex python dependencies.

The Challenges for LLM apps in Cloud Native

- The AI apps are not portable due to the diversity of the GPU and CPU devices.
- We’re forced to use API server.
- However, API server is not flexible and not performant.
  - multi-models
  - complex RAG

- Even Python is not cross device portable

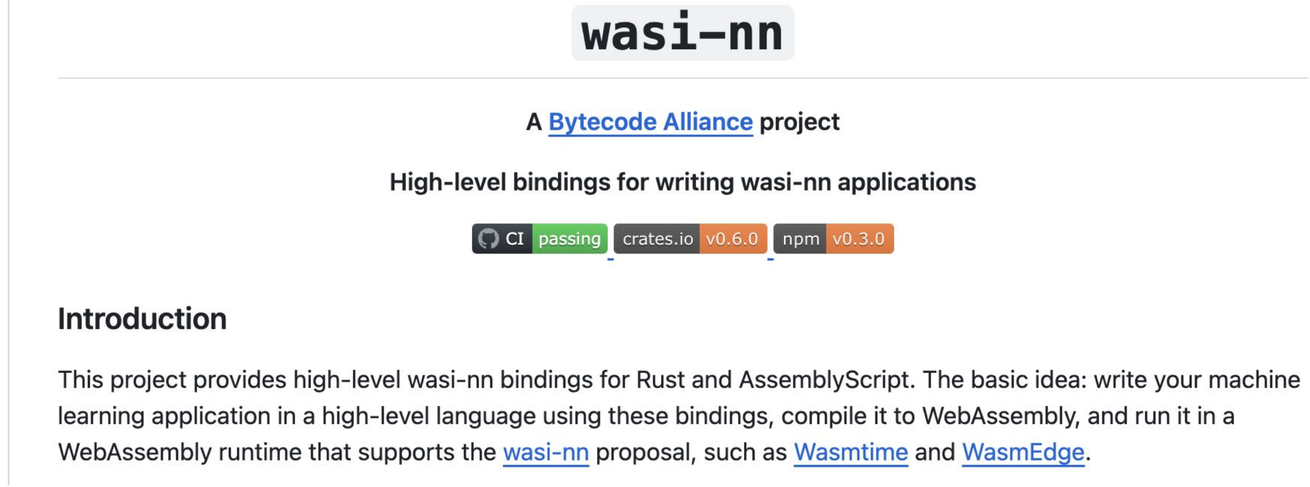
How can we solve the problem?

Why Wasm?

- Portable: the compiled Wasm app can be run across different platforms without recompiling.
- Lightweight: The runtime is only 30 MBs and the inference app is only 4 MBs.

How?

- Wasi-nn proposal
- Support major AI frameworks
  - OpenVINO
  - TensorFlow
  - Pytorch



`wasi-nn`

A Bytecode Alliance project

High-level bindings for writing wasi-nn applications

CI passing | crates.io v0.6.0 | npm v0.3.0

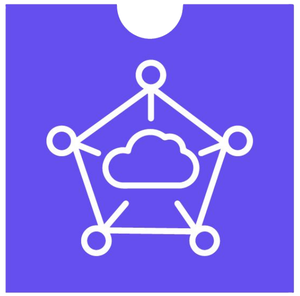
Introduction

This project provides high-level wasi-nn bindings for Rust and AssemblyScript. The basic idea: write your machine learning application in a high-level language using these bindings, compile it to WebAssembly, and run it in a WebAssembly runtime that supports the wasi-nn proposal, such as Wasmtime and WasmEdge.

Extend LLMs to WASI-NN

WasmEdge-Wasi-NN

- Integrate llama.cpp as a new backend
- llama.cpp is the inference of LLaMA model in pure C/C++



WasmEdgeRuntime

What do we get?

- Write once, run everywhere
- Zero python dependency
- Cloud-ready apps
- Native speed

Demo #1
Use Wasm as a cross-platform runtime for LLM inference

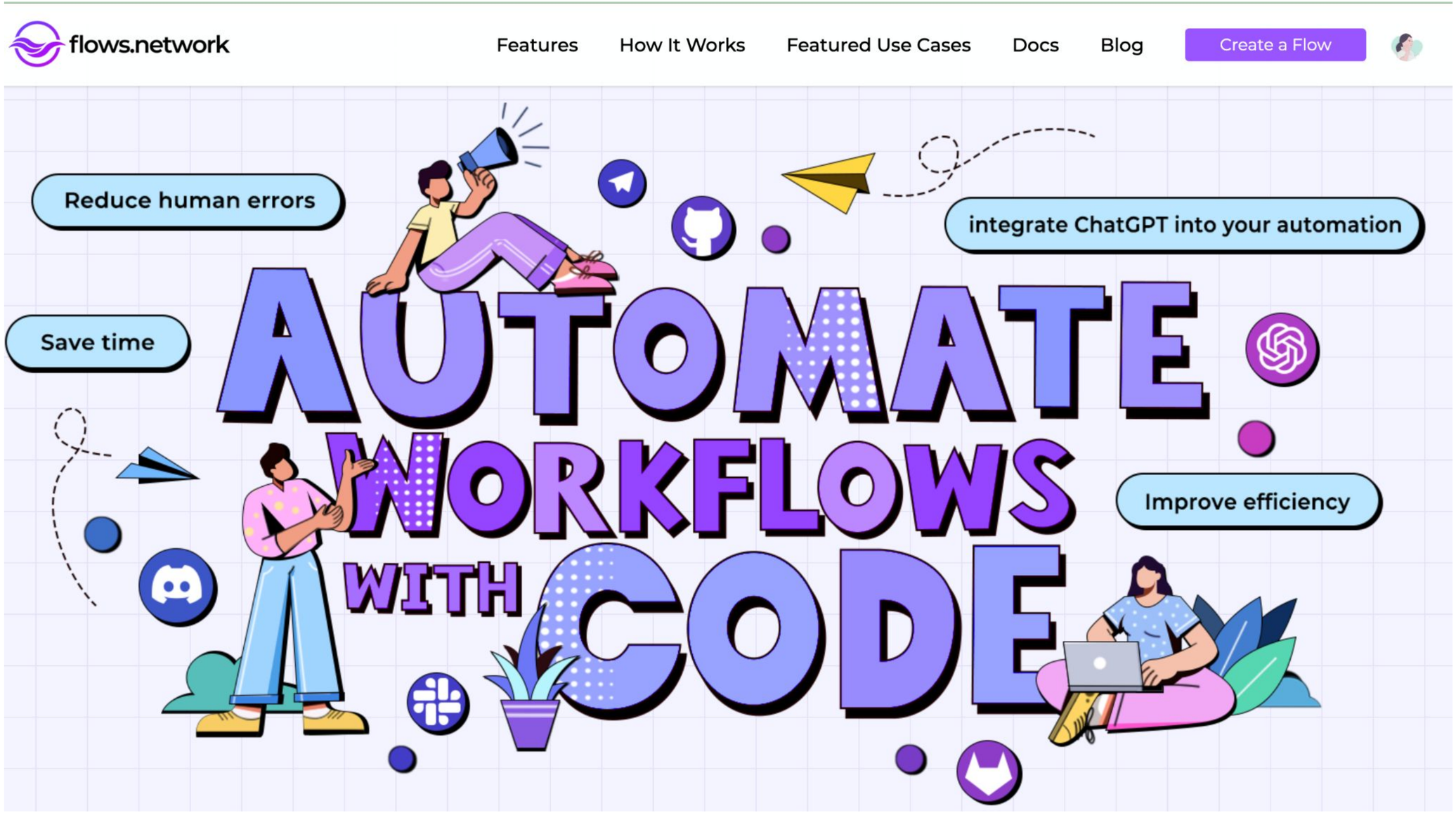
Create an LLM web service on a MacBook, run it on a NVIDIA device.

It's also an OpenAI-compatible API server

- But it's not a simple API server.
- Reuse the ecosystem around OpenAI, like Langchain and flows.network
  - build LLM agent
  - build RAG application
  - build chatbot

Wasm is the best plugin mechanism for LLMs

- Wasm is a great runtime for lightweight serverless function
- It can manipulate LLM input and output like langchain does,
- but without Python

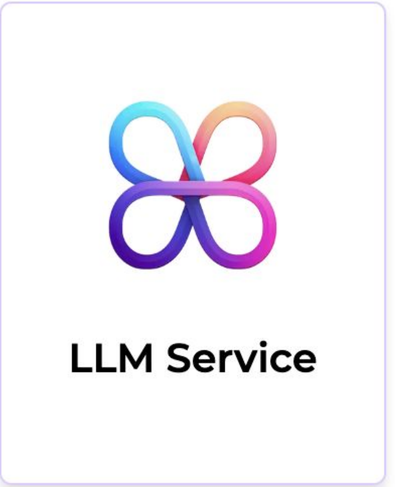

flows.network
Features
How It Works
Featured Use Cases
Docs
Blog
Create a Flow
Reduce human errors
integrate ChatGPT into your automation
Save time
AUTOMATE
WORKFLOWS
WITH
CODE
Improve efficiency

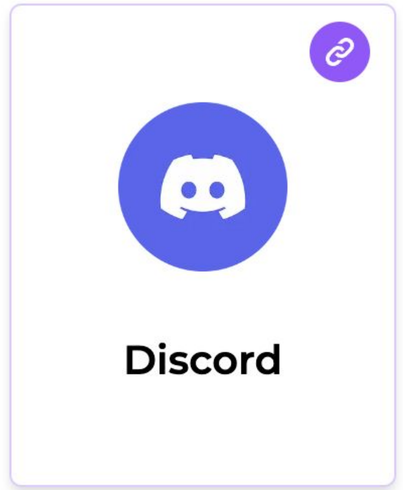
flows.network

- A serverless platform for LLM powered SaaS workflows
- The function is written in Rust
- WasmEdge is the runtime to run the compiled Wasm binary

Create a Discord bot with flows.network

- flows.network provides API integration with Discord and WasmEdge-Wasi-nn runtime
- With WasmEdge, you can use any open source LLMs as backend

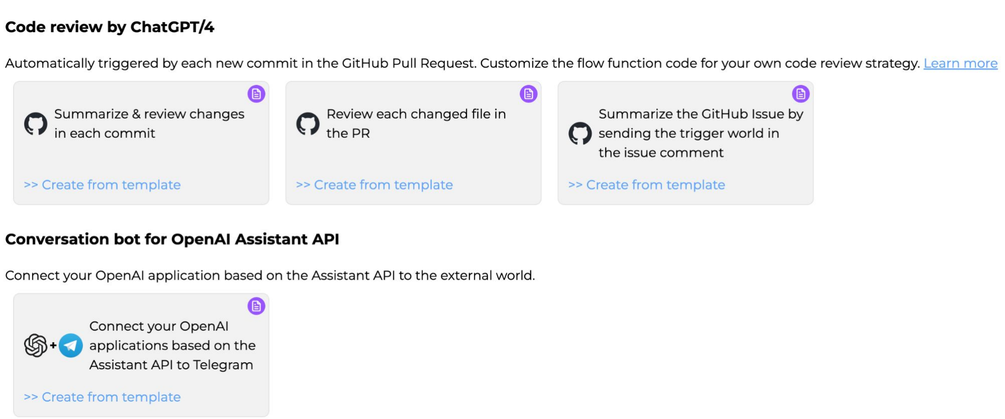


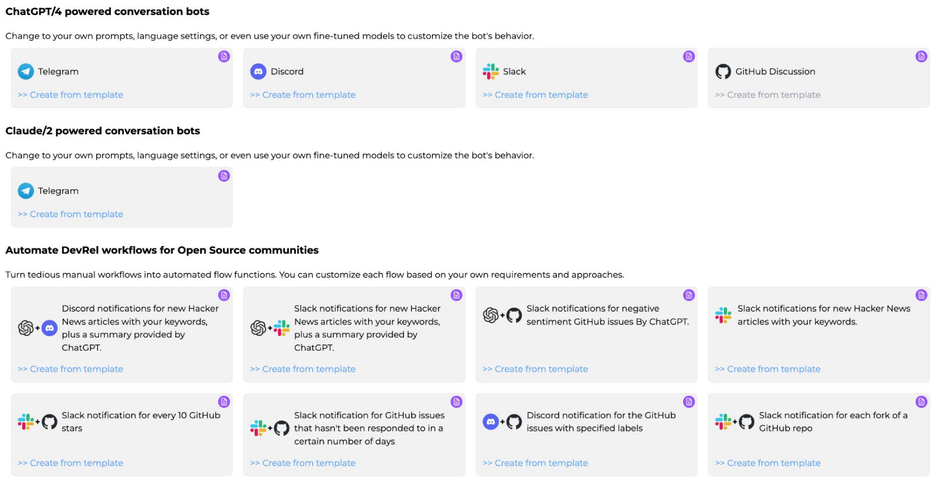


Demo #2
WASM is the runtime for LLM extensions.

LLM agent based on ChatGPT

https://flows.network/start (All the function is written in Rust.)





Roadmap

- Add more LLM-related backends for WASI-NN plugins
  - MLX for Apple chips
  - Intel Extension for Transformers
  - OpenAI’s whisper
  - Burn
- Add support for embedding models
- Container management on GPU

Resources

- WasmEdge: https://github.com/WasmEdge/WasmEdge/
- WASI-NN: https://github.com/bytecodealliance/wasi-nn
- WasmEdge-WASI-NN: https://github.com/second-state/wasmedge-wasi-nn
- LlamaEdge: https://llamaedge.com/
- Source code: https://github.com/LlamaEdge/LlamaEdge
- Flows.network: https://flows.network/start

THANKS!