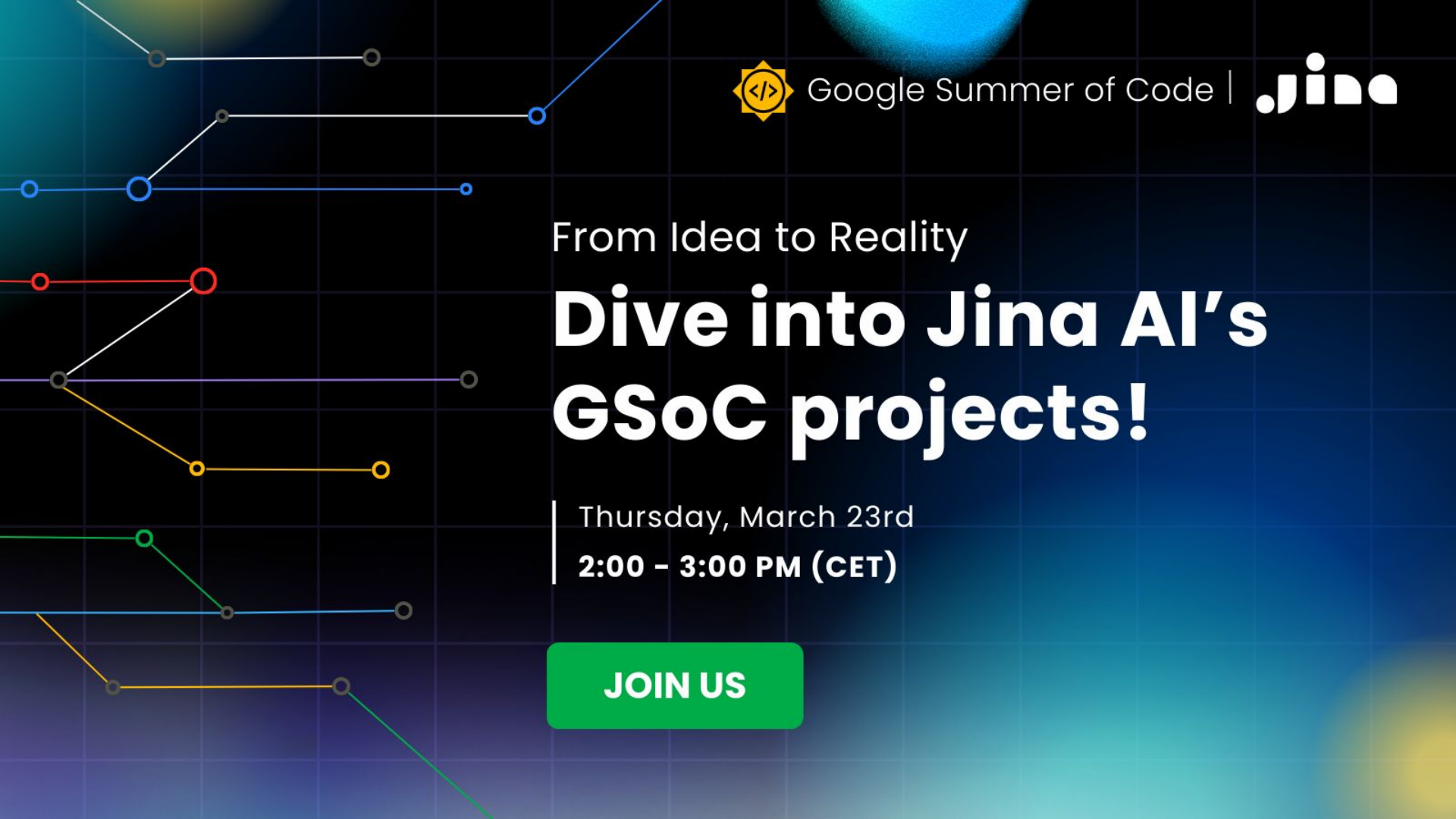

Google Summer of Code |
From Idea to Reality
Dive into Jina AI's GSoC projects!
Thursday, March 23rd
2:00 - 3:00 PM (CET)
JOIN US

AGENDA

1. Welcome to Jina AI x GSoC

2. Mentors introduction

3. Projects walk-through

Q&A Sessions throughout

Welcome to Jina AI x GSoC

Multimodal AI
from the community, for everyone

Empower businesses and developers to create cutting-edge neural search, generative AI, and multimodal services using state-of-the-art LMOps, MLOps and cloud-native technologies.

💻 jina.ai
👩‍💻 get.jina.ai
📖 docs.jina.ai
👫 jina.ai/community

About Jina AI

🚀 Founded in **Feb, 2020**

👫 International Team of **65**

🌎 **4** Offices Globally

💵 Raised **$38M+**

Forbes AI30 DACH 2021

AI 100 2021 CBINSIGHTS

CBINSIGHTS AI 100 2022

🏆 Forbes AI30 DACH 2021

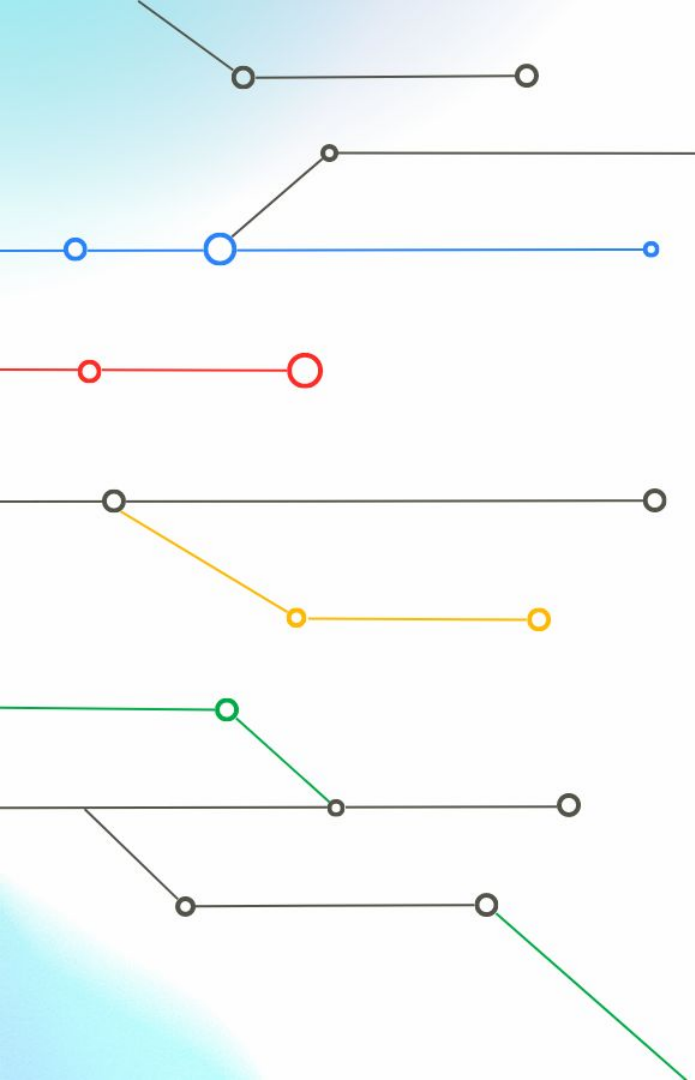
🏆 CBInsights AI 100 2021

🏆 CBInsights AI 100 2022

About Jina AI

Commitment to Open Source

- We have multiple OS projects
- As part of our dedication to OS, we donated DocArray November 2021
- Proud of our talented community!
- We encourage collaboration



Jina AI was selected for

Google Summer of Code 2023

VIEW IDEAS LIST

github.com/jina-ai/gsoc

Mentors Introductions

Mentors

Introduction


Joan Fontanals Martínez - Head of Engineering


Felix Wang - Engineering Manager


Philip Vollet - Head of Developer Experience


Girish Chandrashekar - Senior Software Engineer


Johannes Messner - Senior Software Engineer


Sami Jaghouar - Senior Software Engineer


Alaeddine Abdessalem - Software Engineer

Projects

Title	Skills needed	Mentors	Difficulty	Size
Build Executor (model) UI in Jina	Python	Alaeddine Abdessalem, Philip Vollet	Easy	175 hours
DocArray wrap ANN libraries	Python, ANN Search experience	Johannes Messner, Sami Jaghouar, Philip Vollet	Medium	175 hours
Research about deploying LLM with Jina	Python, Pytorch, CUDA, docker, Kubernetes	Alaeddine Abdessalem, Joan Fontanals Martínez	Medium	350 Hours
Expand ANNLite capabilities with BM25 to build Hybrid Search	Python, C++, Lucene, ANN, Inverted Index	Felix Wang, Joan Fontanals Martínez, Girish Chandrashekar	Hard	350 hours
Make ANNLite the go-to Vector Search library to be scaled by Jina using the StatefulExecutor feature	ANN, C++, Python, Databases	Felix Wang, Joan Fontanals Martínez	Hard	350 Hours
JAX support in DocArray v2	Python, AI/ML, JAX Framework experience	Sami Jaghouar	Hard	175 Hours

Title	Skills needed	Mentors	Difficulty	Size
Build Executor (model) UI in jina	Python	Alaeddine Abdessalem, Philip Vollet	Easy	175 hours
Research about deploying LLM with Jina	Python, Pytorch, CUDA, docker, Kubernetes	Alaeddine Abdessalem, Joan Fontanals Martínez	Medium	350 Hours

Project idea 1

Build Executor (model) UI in Jina

Info	Details
Skills needed	Python
Project size	175 hours
Difficulty level	Easy
Mentors	@Alaeddine Abdessalem @Philip Vollet

🔗 https://github.com/jina-ai/GSoC/issues/16

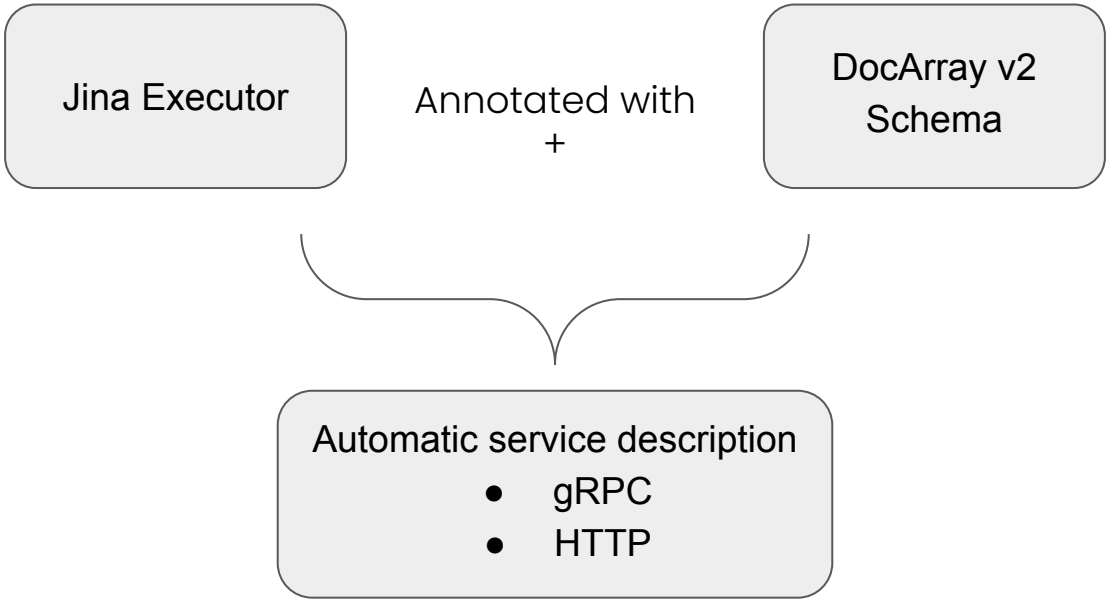
- **Project Brief Description**

Jina Executors are components that perform certain tasks and expose them as services using gRPC. Executors accept DocumentArrays as input and output. However, with DocArray v2 focusing on type annotations and enabling annotation of executor endpoints, it becomes possible for executors to describe their services and input/output in the same way as OpenAPI schemas. This allows us to offer built-in UIs for executors, enabling people to easily use their services with multi-modal data. The goal is to build this feature in Jina using Gradio.

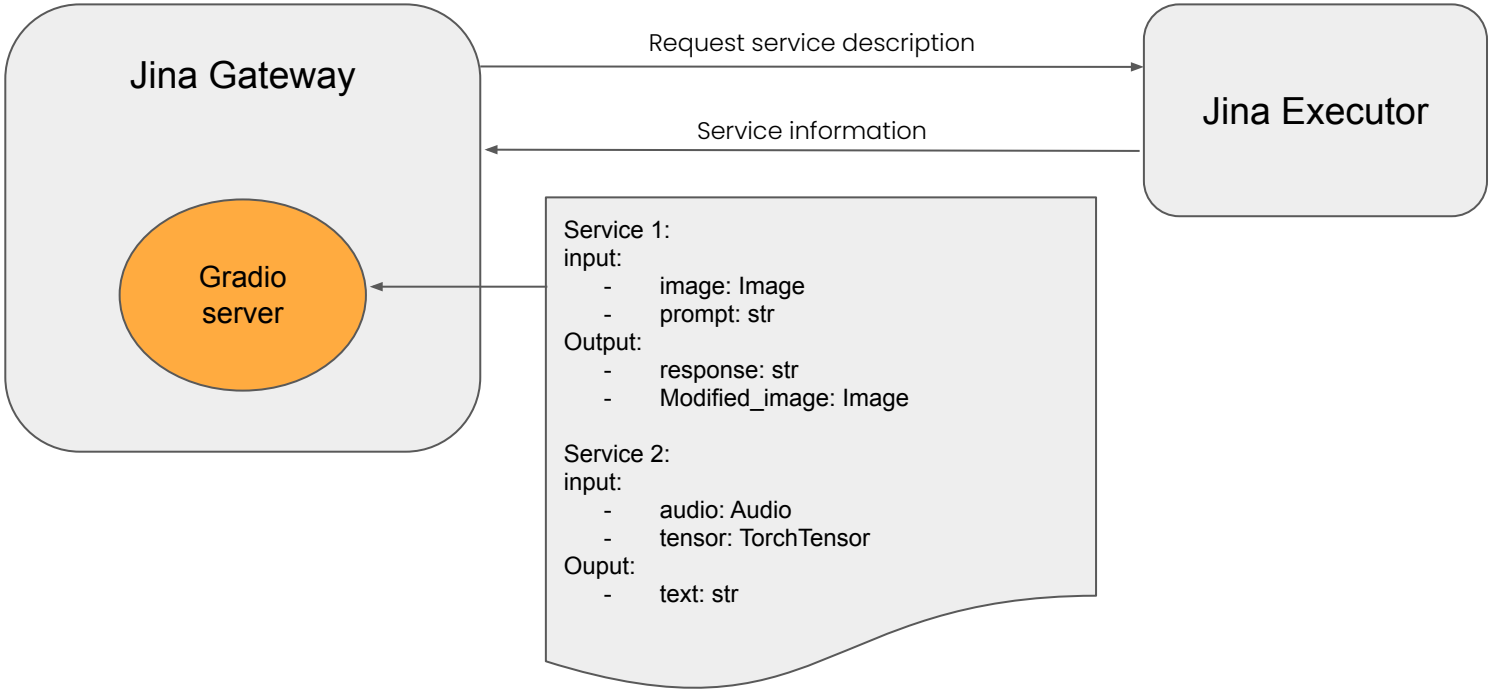
- **Expected outcomes**

Submit one or more Pull Requests (PRs) to the Jina repository that enables providing a built-in Executor UI for Executors. The UI can be built using Gradio and should be able to infer information about the Executor service using type annotations.

Step 1: OpenAPI-like service description



Step 2: Jina Gateway leverages the service description to build a UI



Q&A

Project idea 2

Research about deploying LLM with Jina

Info	Details
Skills needed	Python
Project size	350 hours
Difficulty level	Easy
Mentors	@Alaeddine Abdessalem @Philip Vollet

🔗 https://github.com/jina-ai/GSoC/issues/18

- **Project Brief Description**

Large language models require technologies to enable scalable GPU deployment. This includes:

- Model partitioning/optimizer across GPUs (within/across nodes)
- Weight offloading
- Model compression
- Hardware optimization

Libraries like DeepSpeed, Accelerate, FlexGen enable these techniques for LLMs.
We aim to assess deploying LLMs with Jina and build integrations with these libraries to enable LLM inference using Jina.

- **Expected outcomes**

The project aims to demonstrate the capability of Jina to deploy and scale LLMs and build generative applications in a cost-efficient manner.

Goals:

- Enable some of the following technologies in order to serve a LLM using jina in a local setup
  - Model partitioning/optimizer across GPUs (within/across nodes)
  - Weight offloading
  - Model compression
  - Hardware optimization

P.S: the tech stack/libraries to be used is up to the contributor to choose

- Assess these technologies within a cloud environment (kubernetes, jcloud)
- If necessary, build integration/support for these technologies within jina framework
- Demo/showcase project with the introduced technology

Q&A

Title	Skills needed	Mentors	Difficulty	Size
Expand ANNLite capabilities with BM25 to build Hybrid Search	Python, C++, Lucene, ANN, Inverted Index	Felix Wang, Joan Martínez, Girish Chandrashekar	Hard	350 hours
Make ANNLite the go-to Vector Search library to be scaled by Jina using the StatefulExecutor feature	ANN, C++, Python, Databases	Felix Wang, Joan Martínez	Hard	350 Hours

Project idea 3

Expand ANNLite capabilities with BM25 to build Hybrid Search

Info	Details
Skills needed	Python, C++, Lucene, ANN, Inverted Index
Project size	350 hours
Difficulty level	Hard
Mentors	@Felix Wang @Joan Martínez @Girish Chandrashekar

🔗 https://github.com/jina-ai/GSoC/issues/19

- **Project Brief Description**

In relation to Research about deploying LLM with Jina project, another interesting approach would be to incorporate BM25 and Hybrid Search into ANNLite, which would enable Jina to build scalable Hybrid Search solutions in the cloud with a powerful default solution.

ANNLite is Jina's vector search library using HNSW.

This project is about evaluating and trying to apply Hybrid Search approaches on top of ANNLite.

- **Expected outcomes**

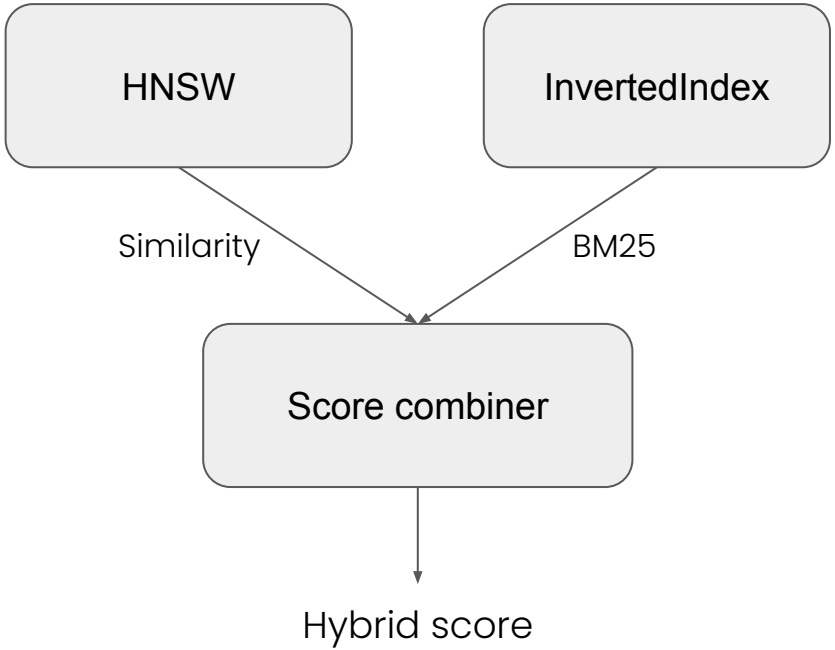
ANNLite is ready to be used as a default library to solve Hybrid Search applications.

ANNLite

ANNLite is a vector search library developed by Jina

- Does pure Vector ANN search with HNSW algorithm
- Library written in C++ wrapped as a Python library
- Hybrid search has proven to provide great value when doing Text Search.
    - Combine the robustness of BM25 with the semantic understanding of Vector embeddings provide by DL models.
    - Combine smartly the scores to provide the best precision/recall

ANNLite


HNSW
InvertedIndex
Similarity
BM25
Score combiner
Hybrid score

ANNLite

What is expected?

- Integration of Inverted Index embedded in ANNlite.
- Expand `ANNLite` search API to enable hybrid search.
- Unit tests and integration tests.
- Evaluation of the performance of the solution in terms of memory requirements/latency.
- Evaluation of the quality improvements on different benchmarks

Q&A

Project idea 4

Make ANNLite the go-to Vector Search library to be scaled by Jina using the StatefulExecutor feature

Info	Details
Skills needed	ANN, C++, Python, Databases
Project size	350 hours
Difficulty level	Hard
Mentors	@Felix Wang @Joan Martínez

🔗 https://github.com/jina-ai/GSoC/issues/20

- **Project Description**

Jina is developing a stateful executor feature that enables Deployments with a state to be replicated and scaled. This opens the door to having a Vector Database in our ecosystem effectively and robustly. Iterating on ANNLite to act as the "Lucene" for Jina would be a great opportunity.

- **Expected outcomes**

Prove and come up with an Executor in our Hub that uses ANNlite or DocArray with ANNLite as a backend to be the default Vector Databases for all our examples for mid-sized data requirements.

Stateful services

Scaling stateful services

- Scaling a Stateful microservice is challenging, while scaling a stateless service is easy using Jina/Jina + Kubernetes.
- Make sure that all replicas reach a consensus on the state of a service in a robust manner is challenging.
- Jina is adding a feature that uses one implementation of RAFT consensus algorithm.
- Among other things, this can be used to make sure local vector indexes can be scaled.
- This could enable to easily enable better throughput/QPS for local vector indexes by using Jina.

RAFT

RAFT algorithm

- Makes sure that a cluster of nodes agree on the state of a Finite State Machine when the same commands are applied in the same order.
- Popular solution for many databases (Hashicorp Consul, etc ...)
- What can be a Finite State Machine? Anything that DETERMINISTICALLY changes an internal state when given the same input command.
- Executors can be thought to meet this requirement. We could build them such given the same input DocumentArray, its internal state is deterministically provided.
- Jina will use this to make sure that replicas of these Executors are synced.

ANNLite

What is expected?

- Build an Executor using the StatefulExecutor feature that uses ANNLite or any other local vector library under the hood.
- Test and prove it works and can be scaled by Jina reliably, report bugs and improvements to Jina framework.
- Explore the limitations of this solution to scale Local vector databases. How many documents can you index? How much QPS can you get?
- If possible, try deploying it to Kubernetes

Q&A

Title	Skills needed	Mentors	Difficulty	Size
DocArray wrap ANN libraries	Python, ANN Search experience	Johannes Messner, Sami Jaghouar, Philip Vollet	Medium	175 Hours
JAX support in DocArray v2	Python, AI/ML, JAX Framework experience	Sami Jaghouar	Hard	175 Hours

Project idea 5

JAX support in DocArray v2

Info	Details
Skills needed	Python, Deep Learning , JAX
Project size	175 hours
Difficulty level	HARD
Mentors	@Sami Jaghouar

🔗 https://github.com/jina-ai/GSoC/issues/21

- **Project Brief Description**

DocArray is a library for representing, sending, and storing multi-modal data, with a focus on applications in ML and Neural Search. It currently supports several deep learning frameworks, including PyTorch and TensorFlow. Jax is becoming increasingly popular for deep learning, so we want to integrate it into DocArray.

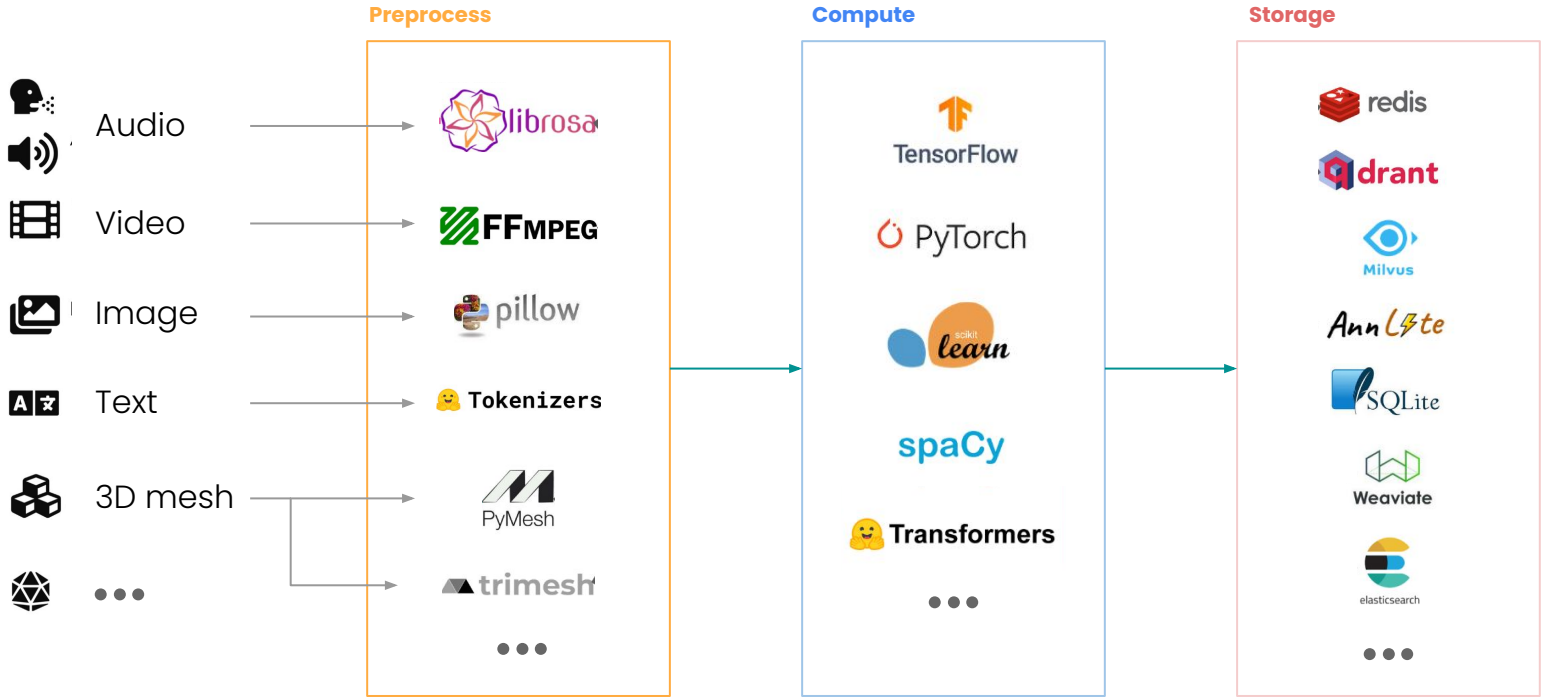
The project we propose is to add Jax as a backend for DocArray, alongside PyTorch and TensorFlow. The first part would involve rewriting and translating all of the computational backend functions of DocArray with the Jax framework. Then, we would battle-test the implementation against a real Jax use case, such as integrating DocArray with Jax support for model training and serving.

- **Expected outcomes**

We aim to provide JAX with the same level of support in DocArray as we do for PyTorch, Numpy, and TensorFlow. The integration should be thoroughly tested and documented.

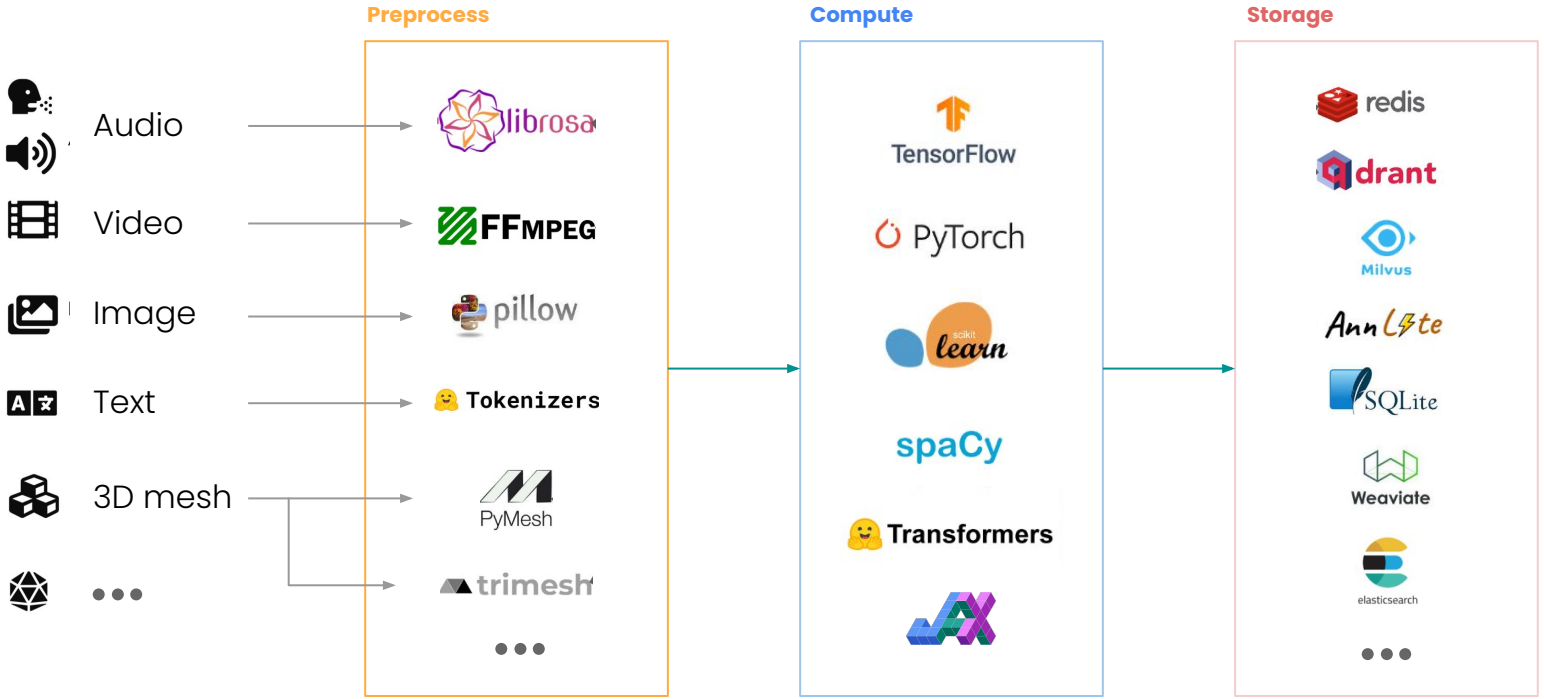
JAX support in DocArray v2

DocArray



JAX support in DocArray v2

DocArray



JAX support in DocArray v2

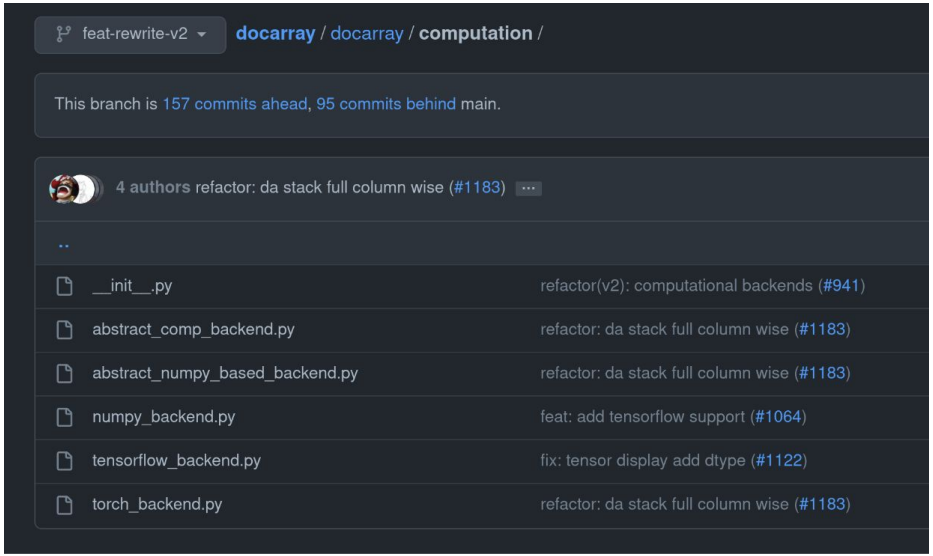
JAX

- Jax is a deep learning framework like : Pytorch, Tensorflow
- Backed by google
- Functional oriented programming
- New kids on the block, great adoption in the research community
- Numpy like API

JAX <-> DocArray

What is expected

1. Add a JAX Computational Backend (common computational interface)



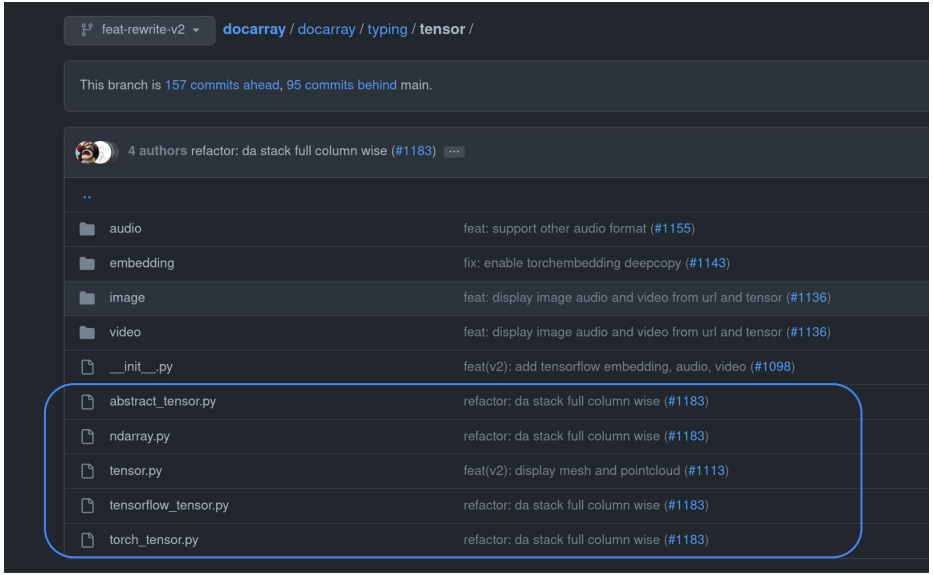
- reshape()
- empty()
- to()
- ...

- Cosine similarity
- ...

JAX <-> DocArray

What is expected

2. Add a JAX Tensor class



- Protobuf
- Inheritance from JAX for compatibility with JAX function

JAX <-> DocArray

What is expected

3. Make DocumentArrayStack compatible

```
from docarray import DocumentArrayStack
from docarray.documents import TextDoc
from docarray.typing import JaxTensor

da = DocumentArrayStack[TextDoc]([TextDoc('hello')], tensor_type = JaxTensor )
```

JAX <-> DocArray

What is expected

4. Test, test ,test

- Unit test
- Integration test
- End to end test
- Documentation and notebook

Sum up

1. Add a JAX Computational Backend
2. Add a JAX Tensor class
3. Make DocumentArrayStack compatible
4. Test, test, test and documentation

Q&A

Project idea 6

DocArray wrap ANN libraries

Info	Details
Skills needed	Python, ANN Search experience
Project size	175 hours
Difficulty level	Medium
Mentors	@Johannes Messner, @Sami Jaghouar, @Philip Vollet

🔗 https://github.com/jina-ai/GSoC/issues/17

- **Project Brief Description**

DocArray v2 will introduce a Document Index abstraction. This lets users store documents and retrieve them via approximate nearest neighbor search, using various backends. There will be multiple indexes with different backends (Elastic, Qdrant, Weaviat, etc.), sharing a common API.
The goal is to integrate different ANN libraries into DocArray, making its scalable vector database accessible to smaller use cases. There is already an HNSWLib-based implementation; indexes based on other libraries (Annoy, Faiss, etc.) could also be added. Vector databases like Milvus or Redis could be explored too."

- **Expected outcomes**

We have a set of DocStores implementations in DocArray that support the most popular ANN libraries, such as FAISS, Annoy, and Hnswlib.

DocArray wrap ANN libraries

Potential other ANN libraries

- FAISS
- Annoy
- PyNNDescent

Q&A

Timeline

March 20, 2023 – April 4, 2023

Contributor Application Period

Potential GSoC contributors can register and submit their proposals to the mentor organizations that interest them.

April 4, 2023 – April 26, 2023

Proposal Review Period

Mentors review and select contributor proposals.

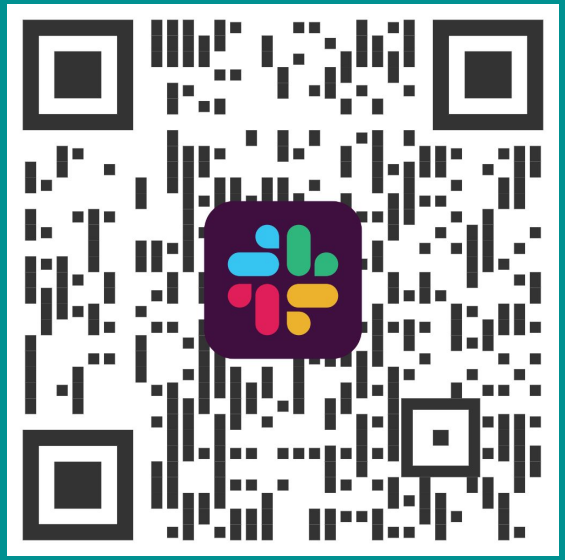
May 4, 2023

Contributor Projects Announced

Accepted GSoC contributors are paired with a mentor and start planning their projects and milestones.

.jina

Join our Slack, engage with our community!



jina-ai.slack.com

Diverse ways to Contribute

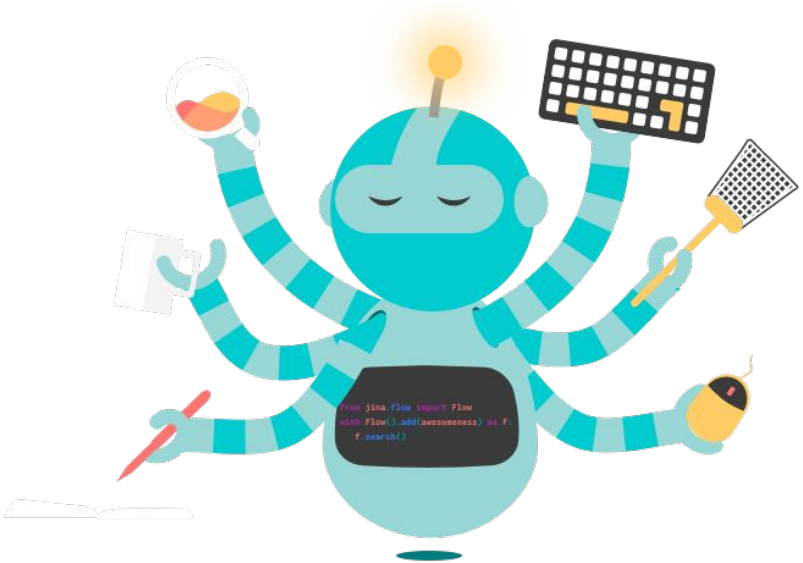
💻 Coder: oss.jina.ai

📝 Writer: docs.jina.ai or jina.ai/news

🎨 Designer: jina.ai/news or twitter.com/jinaAI_

👫 Community builder: jina.ai/community

🧬 There is always a place for you on our team.



Thank you!

jina.ai slack.jina.ai @JinaAI_ github.com/jina-ai jina.ai/careers/