

# Jax support in DocArray v2

---

- ★ Project idea for GSOC 2023 with JinaAI
- ★ [Project in ideas list](#)

## About me

- ❖ **Name:** Sriniketh Jayasendil
- ❖ **Email:** [srinikethcr7@gmail.com](mailto:srinikethcr7@gmail.com)
- ❖ **GitHub:** [srini047](#)
- ❖ **University:** PSG Institute of Technology and Applied Research
- ❖ **Major:** Electronics and Communication Engineering
- ❖ **Expected graduation date:** May 2024
- ❖ **Time zone:** Asia/Calcutta, IST (UTC +5:30)
- ❖ **Languages:** *English, Hindi* (Full proficiency), *Tamil, Telugu* (Professional working proficiency), *German* (Elementary proficiency)
- ❖ **Portfolio:** <https://sriniketh.design>

## Previous Work

- <https://github.com/srini047/jina-gsoc-prework>
- <https://github.com/docarray/docarray/issues/1264>
- <https://github.com/docarray/docarray/pull/1319>

## Relevant Skills

- ☒ Python
- ☒ Jax framework
- ☒ Numpy and TensorFlow
- ☒ Object-oriented programming (OOPs)
- ☒ Git & GitHub (version control)
- ☒ Technical writing (for documentation purposes)

## Project Information: Jax support in Docarray v2

### 1. Project abstract:

Currently the docarray v2 project supports computation support from TensorFlow, PyTorch, and NumPy majorly. But with an increase in the latest advancements in the field of computation, we need to stay up to date with the tools and technologies to make our workflow smoother and more efficient. With that said one of the recent additions and emerging competitors to TensorFlow is Jax (Autograd and accelerated linear algebra). The main reason to add its support to docarray are many, but a few to name are listed below:

- Extensive computation requires less processing time, thanks to transformations with batch processing, complex differentiation, and parallelization
  - NumPy style API so it is easier for developers to migrate with ease
  - Runs on multiple backends and not limited to CPU, GPU, and TPU
  - Wise use of **@jit** (just-in-time) command to compile and run the tasks faster than primitive methods
- 

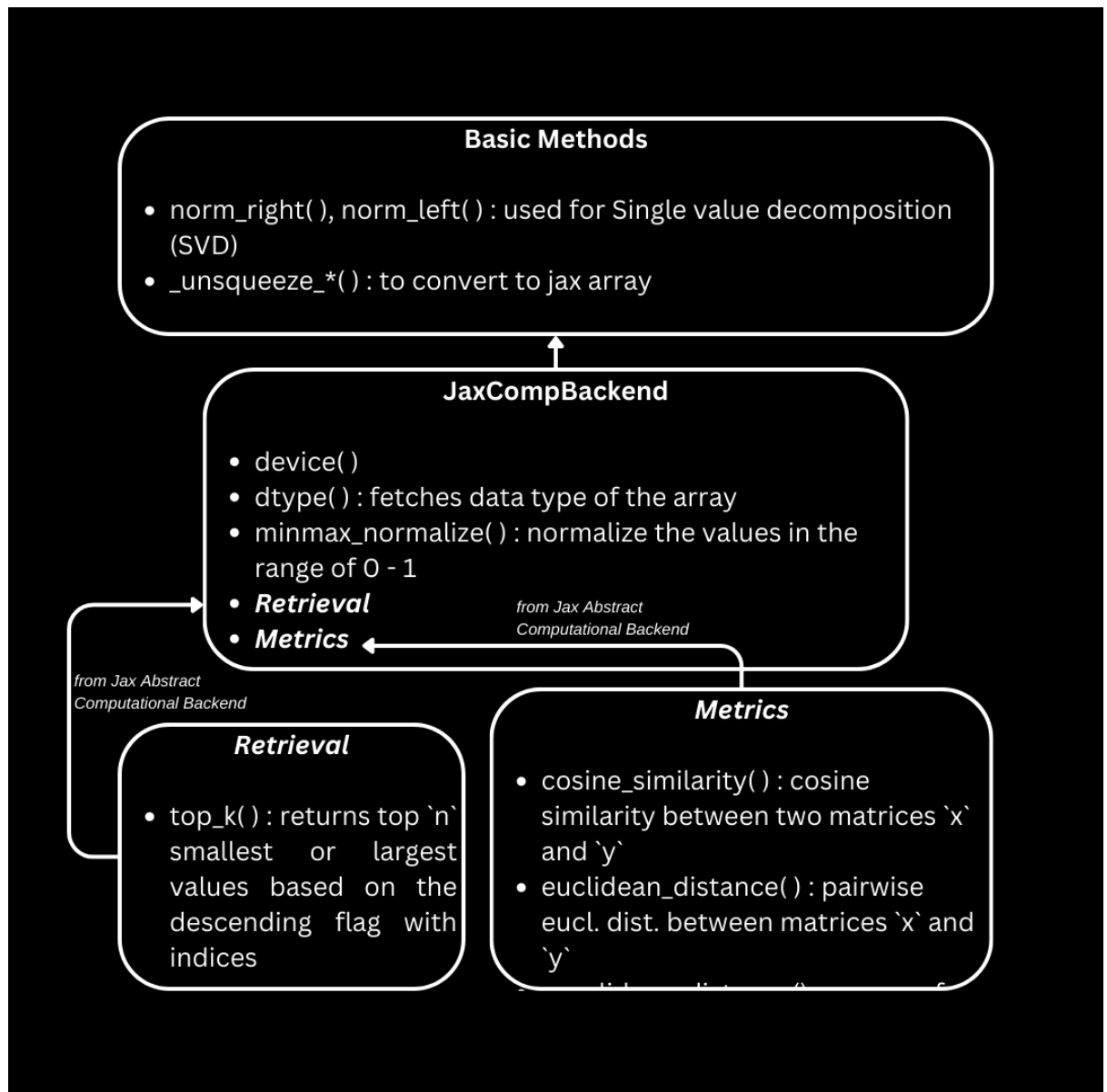
### 2. Detailed description:

- To get familiarized with the project and the Jax framework I worked on the ``jax_backend.py`` that contains the code for adding a computational backend for docarray.

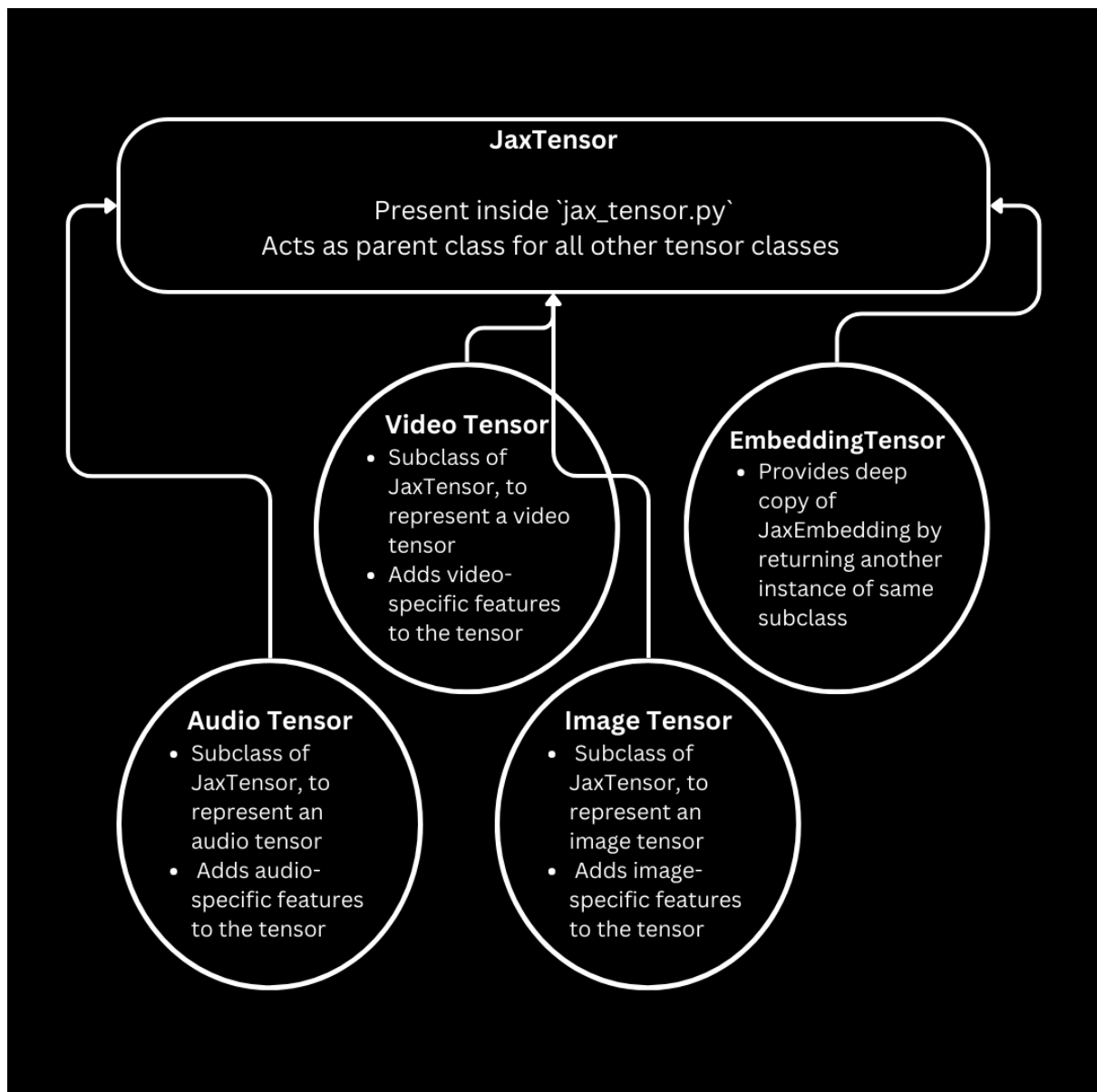
[https://github.com/srini047/jina-gsoc-prework/blob/master/jax\\_backend.py](https://github.com/srini047/jina-gsoc-prework/blob/master/jax_backend.py)

#### Few of my learnings:

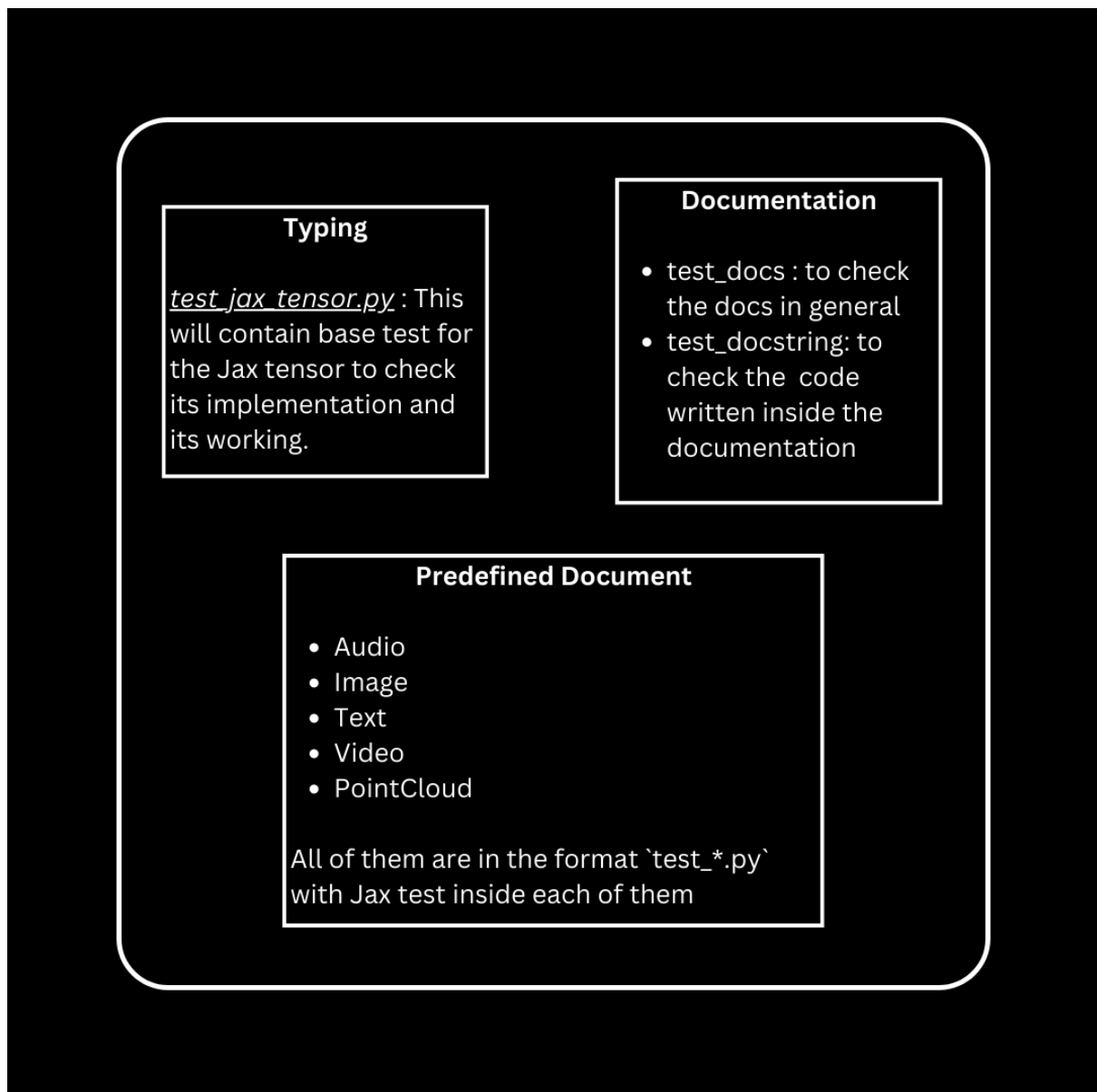
- Syntax of Jax similar to NumPy but numerical computations are way faster (based on testing a few pieces of code)
- Add more functional support like ``reshape()``, ``transpose()``, etc that is not available in tensorflow and numpy
- Expect some friction while building the Jax tensor class\*\* as there is no explicit tensor class available in Jax hence need some research in this area to overcome blockers if any.



- While working on the tensors (image, audio, video) we can make use of `jax.lax` primitive methods that provide a bottom-level view and give us the steering control in defining them. This makes it very much accessible and faster for computation as it is backed by [XLA](#). Protobuf being faster to compute with the addition of Jax will make the computation way faster.



- To make DocumentArrayStack compatible with Jax, we need to ensure that all operations performed on the stack use Jax primitives and that any custom code is Jax-compatible. This is straightforward, as long as care is taken to use correct Jax primitives and datatypes throughout the code.
- After the successful integration, we need to test them to validate the work that requires unit tests at the modular level (computational backend, document array stack, tensor class) and integration test of the entire workflow.



- Add the relevant documentation work to be reflected in the [docs page](#) of docarray. They will be updated [here](#) besides the Pytorch implementation. Also a quick implementation (demo) of the use-case using popular datasets utilizing Jax integration to summarize the entire project.

***(Image represents demo and docs page that would look similar in Jax terms)***

```
[1] import jax
import jax.numpy as jnp
from jax import random

key = random.PRNGKey(0)

WARNING:jax._src.lib.xla_bridge:No GPU/TPU found, falling back to CPU. (Set TF_CPP_MIN_LOG_LEVEL=0 for verbosity info)

[2] key, subkey = random.split(key)
x = random.normal(key, (5000, 5000))

print(x.shape)
print(x.dtype)

(5000, 5000)
float32

[3] y = jnp.dot(x, x)
print(y[0, 0])

63.831306

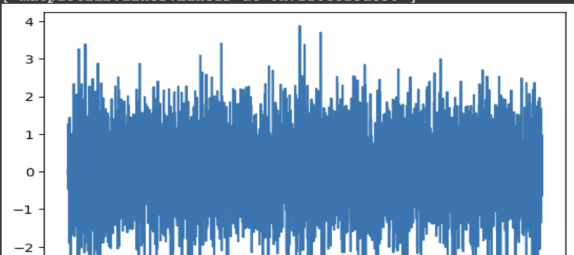
[4] x

DeviceArray([[ -0.03182802,  0.05624727, -0.03447726, ..., -0.6506611 ,
  0.1881411, -0.4943332 ],
 [-0.46550497,  1.034583 , -1.0246143 , ...,  0.7965949 ,
  1.5315311, -0.1446576 ],
 [ 0.4603889 ,  1.2810655 ,  0.1406628 , ..., -0.03909646,
  1.2251555 ,  1.9469196 ],
 ...,
 [-0.1792087 ,  0.12796777, -1.2290448 , ...,  0.07305165,
 -0.7417593 ,  0.84448427],
 [ 0.06088204, -0.359062 , -0.0271101 , ..., -0.78110397,
  0.5980011 ,  0.30552095],
 [-1.2617211 ,  1.0971762 , -0.18960644, ..., -0.7506782 ,
  0.08892686, -1.4386775 ]], dtype=float32)

import matplotlib.pyplot as plt

plt.plot(x[0])

[<matplotlib.lines.Line2D at 0x7ff6ec23d850>]
```



## Search Results

Search finished, found 27 page(s) matching the search query.

### [PyTorch/Deep Learning Frameworks](#)

PyTorch/Deep Learning Frameworks DocArray can be easily integrated into and PaddlePaddle frameworks. The .embedding and .tensor attributes in DocArray can be used to access PyTorch sparse/dense tensors.

### [Image](#)

...; and finally you can chain all these preprocessing steps together in one line. When feeding data into a PyTorch-based ResNet Executor, the image needs to be preprocessed. The image axis should be at first, not at the last....

### [Access Attributes](#)

...> (10,) <class 'numpy.ndarray'> (10,) <class 'numpy.ndarray'> (10,) [1] (1,2) NumPy/TensorFlow/PyTorch/SciPy/PaddlePaddle sparse and dense array.

### [Access Attributes](#)

...Python (nested) list/tuple of numbers, Numpy ndarray, SciPy sparse matrix, dense & sparse tensor, PyTorch dense & sparse tensor, PaddlePaddle dense tensor, image/video/audio doc.blobs Binary string Contain integers...

### [docarray.array.annlite module](#)

...rectly use the embed\_and\_evaluate() function. Parameters: embed\_model - embedding model written in Keras/Pytorch/Paddle device (str) - The computation device, can be either cpu or cuda. batch\_size (int) - Number of samples to process in one batch.

### [docarray.array.chunk module](#)

...rectly use the embed\_and\_evaluate() function. Parameters: embed\_model - embedding model written in Keras/Pytorch/Paddle device (str) - The computation device, can be either cpu or cuda. batch\_size (int) - Number of samples to process in one batch.

### 3. Weekly timeline:

| <u>Timespan</u>                               | <u>Plan of action</u>   |
|---|---|
| <b>Before May 4</b>                           | <ul style="list-style-type: none"> <li>→ Familiarize myself with codebase</li> <li>→ Build app(s) featuring Jina and docarray to get more familiarized with syntax and working nuances</li> <li>→ Understand and build projects using Jax</li> <li>→ Contributing to docarray and jina</li> </ul>   |
| <b>Community bonding<br/>(May 4 - May 28)</b> | <ul style="list-style-type: none"> <li>→ Discuss with the mentors regarding the expectations and set a proper plan and priorities</li> <li>→ Finishing backlog (if any) of the projects and PRs initiated earlier</li> <li>→ Setting up the dev environment as per the requirements</li> </ul>  |
| <b>May 29 - June 4</b>                        | <ul style="list-style-type: none"> <li>→ Discuss with the mentors regarding progress and further plans</li> <li>→ Work on the Jax computational backend on the initial methods namely the <b><i>`norm`</i></b> and <b><i>`squeezing`</i></b> on scalar and single axis</li> <li>→ Work on the comp. backend base class and work on methods like <b><i>`to_numpy`</i></b>, <b><i>`to_device`</i></b>, <b><i>`dtype`</i></b>, and <b><i>`minmax_normalize`</i></b></li> </ul> |
| <b>June 5 - June 11</b>                       | <ul style="list-style-type: none"> <li>→ Discuss with the mentors regarding progress and further plans</li> <li>→ Build the <b><i>`Retrieval`</i></b> class for ensuring appropriate ranking and functionalities</li> <li>→ Build the <b><i>`Metrics`</i></b> class for dealing with distances and similarities that have <b><i>`cosine_similarity`</i></b> and <b><i>`euclidian_distance`</i></b></li> </ul>   |
| <b>June 12 - June 18</b>                      | <ul style="list-style-type: none"> <li>→ Discuss with the mentors regarding progress and further plans</li> <li>→ Start the construction of Jax tensor** class initially by registering the prototype (friction expected)</li> <li>→ Working on the class methods namely <b><i>`get_set_item`</i></b>, <b><i>`iter`</i></b>, and <b><i>`validator`</i></b></li> </ul>   |
| <b>June 19 - June 25</b>                      | <ul style="list-style-type: none"> <li>→ Discuss with the mentors regarding progress and further plans</li> <li>→ Construct the methods like <b><i>`modify_schema`</i></b>, <b><i>`to_json`</i></b>, and <b><i>`unwrap`</i></b> to deal with JSON part</li> <li>→ Construct methods namely <b><i>`to_protobuf`</i></b>, <b><i>`from_protobuf`</i></b> and <b><i>`get_comp_backend`</i></b> to</li> </ul>  |

| <u>Timespan</u>  | <u>Plan of action</u>   |
|--|---|
|  | streamline with protobuf  |
| <b>June 26 - July 2</b>                                  | <ul style="list-style-type: none"> <li>→ Discuss with the mentors regarding progress and further plans</li> <li>→ <b>Check the working</b> of jax comp. backend and jax tensor class (not related to unit or integration testing)</li> <li>→ Start <b>researching</b> about DocArrayStack and how to integrate Jax inside it</li> <li>→ Start documenting the progress until current stage</li> </ul> |
| <b>Mid-term evaluation preparation (July 3 - July 9)</b> | <ul style="list-style-type: none"> <li>→ Discuss with the mentors regarding progress and further plans</li> <li>→ Restructure the codebase (if req.) for evaluation</li> <li>→ Complete <b>document</b> the progress until now</li> <li>→ Surplus time to achieve the plans that were not attained</li> </ul>   |
| <b>Mid-term evaluation (July 10 - July 16)</b>           | <ul style="list-style-type: none"> <li>→ Discuss with the mentors regarding progress and further plans</li> <li>→ Finalize the mid-term evaluation submission</li> <li>→ Also, start working on <b>`DocArrayStacked`</b> class methods starting off with <b>`init`</b>, <b>`from_cols_storage`</b>, and <b>`validate`</b></li> </ul>  |
| <b>July 17 - July 23</b>                                 | <ul style="list-style-type: none"> <li>→ Discuss with the mentors regarding progress and further plans</li> <li>→ Work on class methods namely <b>`get_item`</b>, <b>`set_item`</b>, <b>`update_item`</b> and <b>`delete_item`</b></li> </ul>   |
| <b>July 24 - July 30</b>                                 | <ul style="list-style-type: none"> <li>→ Discuss with the mentors regarding progress and further plans</li> <li>→ Testing the code by writing unit tests for every function inside of <b>JaxCompBackend</b></li> <li>→ Unit test for <b>JaxTensor</b></li> <li>→ Unit testing the Jax part of <b>DocArrayStack</b></li> </ul>   |
| <b>July 31 - August 6</b>                                | <ul style="list-style-type: none"> <li>→ Discuss with the mentors regarding progress and further plans</li> <li>→ Writing integration test on <b>entire workflow</b></li> <li>→ Also, write integration test on predefined dataset on training the NN on <a href="#">toys dataset</a> based upon DocArray and Jax</li> </ul>  |
| <b>August 7 - August 13</b>                              | <ul style="list-style-type: none"> <li>→ Discuss with the mentors regarding progress and further plans</li> <li>→ Work on the <b>documentation</b> side that needs to be</li> </ul>   |



| <u>Timespan</u>  | <u>Plan of action</u>   |
|--|---|
|  | <p>added on these <a href="#">pages</a> besides numpy, tensorflow and pytorch</p> <p>→ Also <b>research</b> on project ideas and gather datasets to be used while building the project</p>  |
| <b>Final evaluation preparation<br/>(August 14 - August 20)</b>      | <p>→ Discuss with the mentors regarding progress and further plans</p> <p>→ Start working on the <b>project</b> utilizing Jax on popular datasets (MNIST, Iris) to start off with</p> <p>→ Then move to other <b>large datasets</b> (&gt;50k) from Kaggle and work on a comprehensive project</p> <p>→ Document the work done</p>           |
| <b>Final evaluation &amp; submission<br/>(August 21 - August 28)</b> | <p>→ Discuss with the mentors regarding progress and further plans</p> <p>→ Complete the <b>project</b> (implementation, demo, blog)</p> <p>→ Complete the <b>entire documentation</b> of the work</p> <p>→ Surplus time to finish of any pending tasks that might be in the backlog</p> <p>→ <b>Wrap-up</b> the entire work and submit</p> |
| <b>After GSOC</b>  | <p>→ Stay in touch with the mentors as well as maintaining and contributing to Jax support for DocArray v2 and Jina community in general</p>  |

## Major Commitment

- I may have my university end-sem exams scheduled but I am not sure about the dates as I get to know only 2 weeks prior. It's expected to be at the end of the May or starting of June

## References

- <https://jax.readthedocs.io/>
- <https://github.com/google/jax>
- <https://youtu.be/uySOXq-II0>
- <https://github.com/docarray/docarray/tree/feat-rewrite-v2/>

**\*\*** - There is no explicit tensor class available in the Jax framework currently. The `tf.tensor` or `torch.Tensor` equivalent in jax is `jnp.ndarray` which is a NumPy-like array class that supports automatic differentiation and can be executed on both CPU and GPU devices. But this means we need to ensure that it supports docarray and doesn't break the current workflow. In case this doesn't work out an alternative solution would be to use [DLPack tensors](#) supported in Jax. This can be achieved by easily converting to `jnp.ndarray` objects using the `jax.interop.dlpack.from_dlpack()` function, and `jnp.ndarray` objects can be exported as DLPack (Deep Learning Pack) tensors using the `jax.interop.dlpack.to_dlpack()` function. But we have to take into account the computation speed in converting from arrays to tensors and vice versa. If we use the above logic then we may need to be careful with the terminologies i.e. use *Jax Array* instead of *Jax Tensor*. In case, this doesn't work then try to implement it the same as `ndarray` i.e. tensor in terms of docarray.