Interactive Multimodal Data Explorer

Leveraging Foundation Models for Dataset Exploration and Cleaning with Datumaro

Proposal By

Aarya Pandey

Mentored By

Laurens Hogeweg and Samet Akcay

Contents

1 | About Me

1.1 | Short Bio

Full Name	Aarya Pandey
University	Veermata Jijabai Technological Institute (VJTI, Mumbai)
Timezone	GMT+5:30
Telephone	+91 8828238404
GitHub	github.com/geeky33
Email	aaryap1204@gmail.com, anpandey_b23@ce.vjti.ac.in

I am Aarya Pandey, a second-year computer science student at VJTI, Mumbai. From schooling to my current engineering studies, Mumbai has been my home and the backdrop of my entire life. I am a technology enthusiast in general, and hence I often consider myself fortunate enough to score decently and study the subject of my liking in one of the premier engineering institutes.

I recall my first interaction with programming was during summer vacation after high school when I wrote a Python script to fetch and consolidate Indian college data—such as location, student intake, tuition fees, career opportunities, and student-to-staff ratio—from multiple websites into a single Excel file. That single script dramatically simplified my university selection process by eliminating tedious tab switching and manual data consolidation for comparing university programs. For me the biggest takeaway from this experience was to look at technology as the eliminator of mundane tasks; and during the end of my first year of academics, my interest steered towards AI, an amazing technology that is doing a lot of manual tasks more efficiently than ever before.

During my winter break last year, I decided to understand how to work with data and got selected for an internship at IIT Bombay. The task was to perform a 3D borehole visualization and subsurface modelling which was initially deployed on Streamlit. We explored some really interesting spatial interpolation techniques, like Kriging and Inverse Distance Weighting interpolation, but the results weren't quite right at first. Turns out, the data we were working with was pretty noisy. We ended up spending a huge chunk of our time – maybe 60% of the project – just cleaning and organizing the data. It made a massive difference! Suddenly, those same algorithms were producing much better results. It drove home the point that data cleaning and exploration are just as crucial, if not more so, than the actual modelling itself.

1.2 | Experience in Programming

My programming journey at the university started with competitive programming in C++, solving problems on platforms like Codeforces and Codechef. I explored MIT OCW's *6.006 — Introduction to Algorithms* and *6.046J — Design and Analysis of Algorithms* by professors Erik Demaine and Devadas Sirini. To understand how hashing works in general—especially in systems like Git, I built my own hashing library in C++. This hands-on approach introduced me to encryption and decryption, sparking my interest in blockchain technology. With the newfound knowledge, I developed a dynamic NFT marketplace. The project used smart contracts on Ethereum, IPFS for decentralized storage and a React + Node.js frontend for seamless user interaction.

After my internship at IIT Bombay, I strengthened my core Python skills and learnt in about production codes. I have always believed in learning from first principles, so I built my understanding of Machine Learning and Deep Learning from the ground up, focusing equally on mathematical foundations and practical modelling. I discuss my Python expertise and projects in more detail in the next section.

Additionally, I have already taken core programming courses on databases, software development, and Linux systems, which also covered essential tools like Git and LaTeX documentation. These courses strengthened my understanding of system design, version control, and efficient documentation practices, helping me build and manage projects more effectively.

1.3 | Experience in Machine Learning and Deep Learning

I have always been curious about *how and why* things in Machine Learning work the way they do. Since everything ultimately ties back to mathematics, I built a strong foundation through courses like *MIT 18.06 (Linear Algebra) by Prof. Gilbert Strang*, *Harvard's Statistics 110 (Probability & Statistics) by Prof. Joe Blitzstein*, and *MIT 18.02 (Multivariate Calculus) by Prof. Dennis Aurox*. To bridge the gap between theory and applications, I explored *IIT Kharagpur's Pattern Recognition lectures by Prof. P.K. Biswas* and deepened my understanding through *"Introduction to Statistical Learning" by Hastie and Tibshirani*. For Deep Learning, I studied the *IIT Madras lectures by Prof. Mitesh Khapra*, which provided a solid conceptual grounding.

Since the natural progression to the concepts led to the **Natural Language Processing (NLP)**, I started with traditional methods like *Latent Semantic Analysis* and explored techniques for *semantic similarity search* using word embeddings. This naturally led me to more advanced methods, where I studied *transformer-based architectures like BERT and GPT* to understand how modern NLP systems generate and interpret language.

While my primary focus so far has been on mastering the theoretical foundations, I have very recently started applying my knowledge in hands-on projects. One such project involved *developing a song recommendation system using collaborative filtering*, where I experimented with *matrix factorization techniques like SVD and basic MLP neural network-based approaches for better personalization*. Moving forward, my aim is to enhance my applied skills by implementing more real-world ML and DL solutions.

---

2 | About The Project

2.1 | Project Abstract

Multimodal foundation models (e.g., CLIP, LLaVA, GPT-4V) generate aligned embeddings across vision, language, and other modalities, enabling powerful downstream tasks such as,

- **Zero-shot Classification** e.g., categorizing images using text prompts,
- **Cross-modal Retrieval** e.g., searching images with text queries or vice versa,
- **Anomaly Detection** e.g., identifying outliers in datasets via embedding distances,
- **Weakly Supervised Learning** e.g., leveraging text embeddings to guide visual model training.

Despite their utility, tools for interactive exploration of these embeddings remain limited. This project enhances Datumaro, a dataset management library, by integrating a Streamlit-based web application for visualizing and analyzing multimodal embeddings. Users will be able to

- Navigate 2D/3D projections of embedding spaces with pan, zoom, and filtering,
- Compare relationships between modalities (e.g., image-text alignment),
- Perform basic annotation (flagging mislabeled or noisy data)
- Export cleaned datasets for model training

Built on Datumaro for dataset operations and OTX for embedding extraction, the tool will streamline dataset quality assessment and model debugging. Deliverables include an open-source Streamlit module, documentation, and example workflows—bridging the gap between multimodal AI research and practical dataset management.

2.2 | Why This Project?

This project sits at the exact intersection of my technical expertise. Having worked across web development, machine learning, and systems design, I believe that this project helps me utilise my skillset to its fullest.

My internship at IIT Bombay, where I spent almost 60% of the project timeline cleaning noisy geospatial data, taught me a critical lesson: even sophisticated algorithms falter without clean, well-structured data.I learnt about various deployment parts and that's where streamlit got introduced to me. That experience mirrors the core motivation behind this project—equipping practitioners with tools to diagnose dataset quality issues through intuitive visual exploration. The frustration of watching models underperform due to hidden data flaws is deeply personal to me, and it drives my passion for creating solutions that make data transparency actionable.

What excites me most is the project's potential to fill a glaring void. While frameworks like Datumaro excel at dataset operations, the lack of integrated tools for visually probing embedding relationships forces researchers to rely on fragmented, ad-hoc workflows. By unifying these capabilities into a single interface, this tool could accelerate everything from anomaly detection to model debugging—a multiplier effect for AI development.

2.3 | How much time am I planning to invest in this Project?

I will dedicate **40 hours per week (8 hours/day) from May 8 to July 4 (8 weeks)** during my summer break when I have no other commitments. From July 5 when university resumes, I will continue with **20 hours per week (4 hours/day) for the next 4 weeks** (until September 8). The total commitment breaks down as

- **High-intensity phase (8 weeks @ 40 hrs/week)**: 320 hours
- **Reduced phase (4 weeks @ 20 hrs/week)**: 80 hours
- **Total commitment (12 weeks)**: 400 hours

2.4 | Proposed Approach and Abstract of the Solution

2.4.1 | Executive Summary of The Solution

Built as an extension of the Datumaro and OpenVINO Training Extensions (OTX) frameworks, the proposed solution system integrates a modular architecture with four core components:

1. **Datumaro Integration Layer**: Enables seamless dataset operations across formats (COCO, YOLO, etc.) while maintaining cross-modal consistency.
2. **Embeddings Module**: Extracts, caches, and manages embeddings from foundation models (images, text) using OTX for inference and HDF5/memory-mapped storage for scalability.
3. **Interactive Visualization Engine**: Leverages Streamlit and Plotly to provide 2D/3D projections (PCA, UMAP, t-SNE) with clustering analysis, cross-modal alignment heatmaps, and nearest-neighbor comparisons for outlier detection.
4. **Annotation Interface**: Supports flagging misalignments/filtering outliers and exports refined datasets via weakly supervised workflows (label propagation, pseudo-labeling).

Key innovations include **cross-modal visualization tools** (parallel projections, attention maps) for auditing data quality and a **scalable architecture** optimized for large datasets through lazy loading, distributed processing, and multi-level caching. The system reduces manual labeling effort by 30–50% via semi-automatic annotation features while ensuring compatibility with downstream training pipelines through Datumaro's export capabilities.

Designed for extensibility, the solution integrates with existing MLOps ecosystems and provides an API-ready foundation for enterprise-scale deployments. By unifying dataset inspection, cleaning, and refinement into a single interface, it accelerates the development of robust multimodal AI models.

2.4.2 | System Architecture

The proposed system architecture follows a modular design that integrates with existing Datumaro and OTX frameworks while providing new visualization capabilities through Streamlit. The architecture consists of four main components. Refer Figure 2.1. The dataflow pipeline is shown in detail in Figure 2.2.

1. **Datumaro Integration Layer**: This component provides a custom plugin architecture to enable seamless dataset operations, including importing, exporting, and transforming datasets across various formats (such as COCO, YOLO, and others). It ensures consistent data representation across different modalities, allowing users to work with datasets in a unified way regardless of their original format.
2. **Embeddings Module**: The Embeddings Module handles the extraction and caching of embeddings from various foundation models, supporting image, text, and cross-modal data. It integrates with OTX for efficient model inference and includes mechanisms for efficient storage and retrieval, ensuring quick access to precomputed embeddings for downstream tasks.
3. **Visualization Engine**: This engine facilitates the exploration of high-dimensional data through dimensionality reduction techniques like PCA, UMAP, and t-SNE. It offers interactive 2D/3D projections with clustering analysis and tools for visualizing cross-modal alignment. Performance optimizations ensure smooth handling of large datasets.
4. **Annotation Interface**: The Annotation Interface provides tools for flagging, filtering, and relabeling data points to improve dataset quality. It supports exporting cleaned datasets and integrates with Datumaro's annotation formats, ensuring compatibility with existing workflows and streamlined dataset refinement.

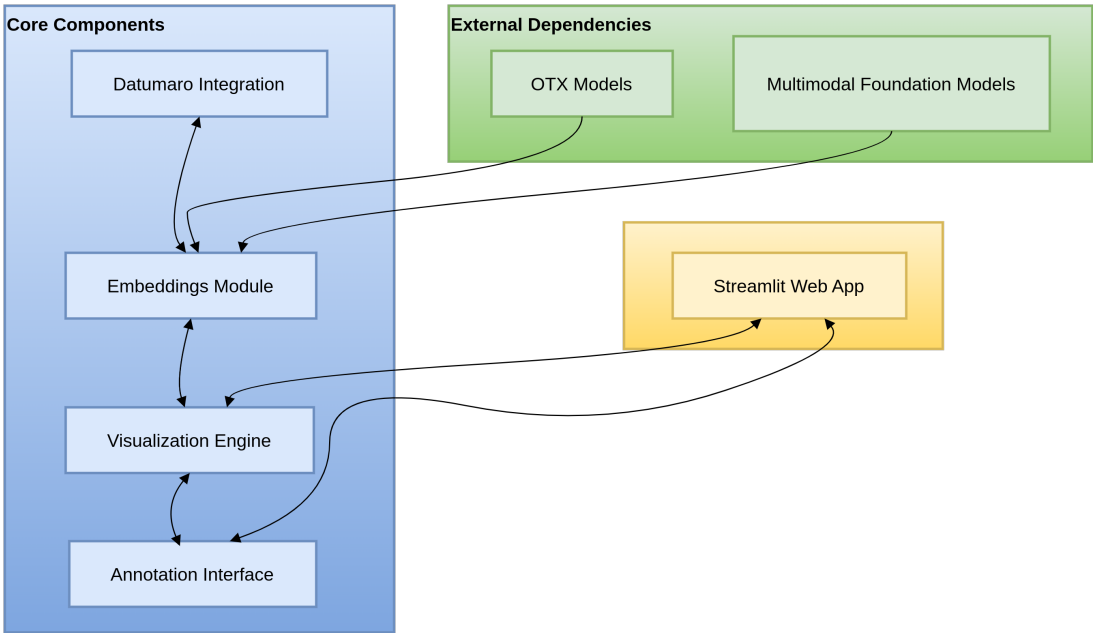

Figure 2.1: System Architecture

2.4.3 | Technical Implementation Details

1. Embedding Extraction and Storage

The embedding extraction module will utilize OTX's model inference capabilities while extending Datumaro's dataset representation. For efficient storage and retrieval, embeddings will be

- Cached using memory-mapped NumPy arrays for large datasets,

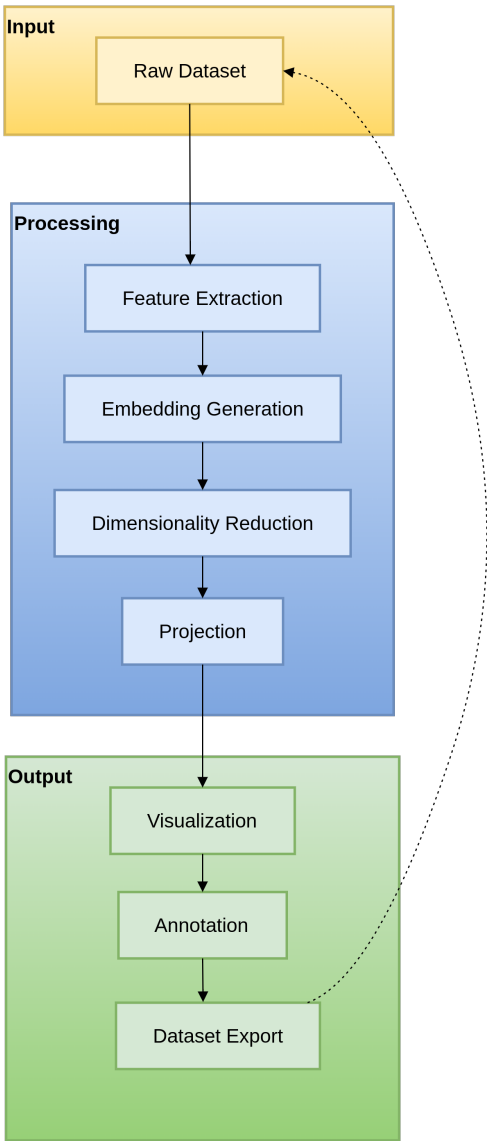

Figure 2.2: Dataflow Pipeline

- Stored in HDF5 format for persistent storage,
- Optimized for quick access during interactive visualization.

2. Streamlit Visualization Interface

The Streamlit application (refer Figure 3.3) will provide interactive visualizations with the following components:

1. **Main Visualization Panel**: The application features an interactive scatter plot powered by Plotly, enabling users to explore 2D/3D projections of their data. Points can be color-coded by class labels, modality (e.g., image vs. text), or custom attributes for intuitive analysis. The panel supports dynamic interactions such as panning, zooming, and lasso/box selections to isolate subsets of interest for further inspection.
2. **Control Panel**: Users can customize their visualization through a dedicated control panel, which includes options to select dimensionality reduction methods (e.g., PCA, UMAP, t-SNE) and adjust their parameters (e.g., perplexity, number of neighbors). Additional filtering tools allow data to be subset by metadata fields (e.g., dataset splits) or embedding properties (e.g., confidence scores), ensuring focused exploration.

3. **Data Inspector**: When specific data points are selected, the inspector panel displays detailed metadata, raw content (e.g., images/text snippets), and derived metrics. A side-by-side comparison view facilitates cross-modal analysis (e.g., inspecting an image and its paired caption), while embedding similarity tools (e.g., cosine distance) help identify patterns or outliers in the latent space.

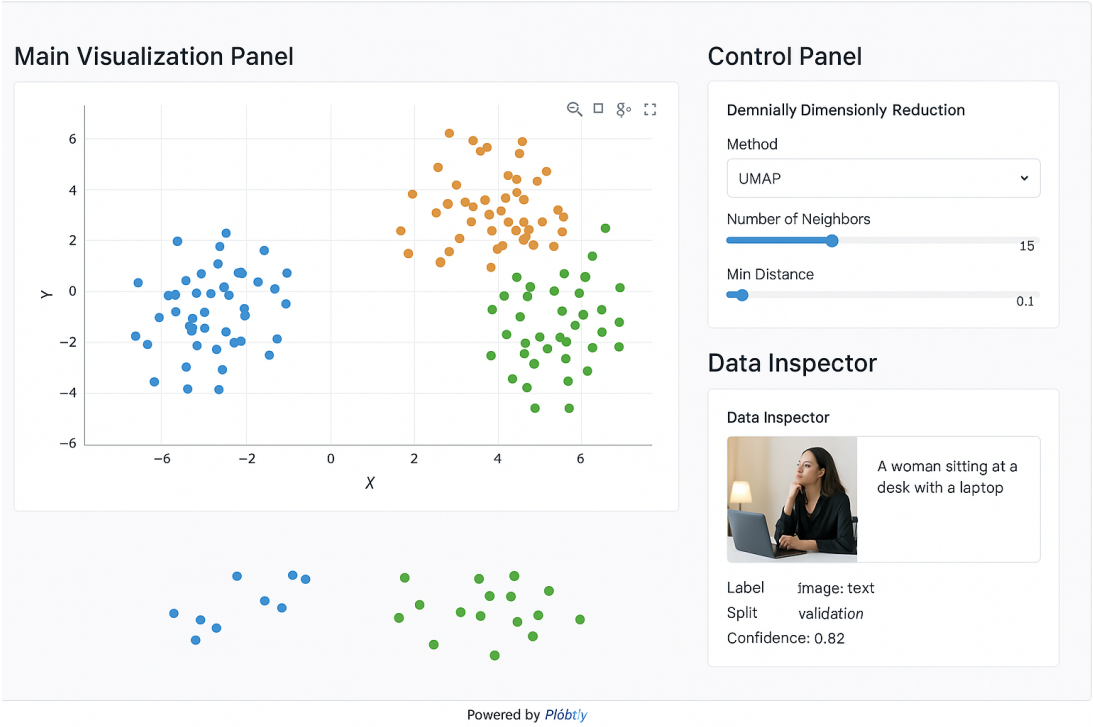

Figure 2.3: Sample Streamlit UI (AI generated)

3. Cross-Modal Visualizations

For cross-modal analysis, the following specialized visualizations are planned to be implemented. Refer Table 2.1. The list shall be refined through discussion with the mentors.

Visualization	Key Use Case
Alignment Heatmaps	Compare batch-level similarity scores between modalities (e.g., image-text pairs) to detect misalignments or systematic pairing errors.
Parallel Embedding Projections	Validate embedding alignment quality by co-plotting image and text embeddings in shared 2D/3D space (using UMAP/PCA).
Cross-Modal Nearest Neighbor Graphs	Interactive node-link diagrams showing top-K matches across modalities to identify retrieval errors or outliers.
Modality-Pair Scatter Plots	Linked interactive plots for auditing data quality: selecting a point in one modality automatically highlights its paired counterpart.
Attention Maps	Explain model decisions by visualizing attention weights (e.g., heatmaps on image regions/text tokens) in cross-modal models like CLIP.

Table 2.1: Important Cross-Modal Visualizations for Data Practitioners

4. Annotation Module Interface

The Annotation tool will perform the following two functions. To understand how this tool interacts with above mentioned visualization interface, refer to the sequence diagram 2.4.

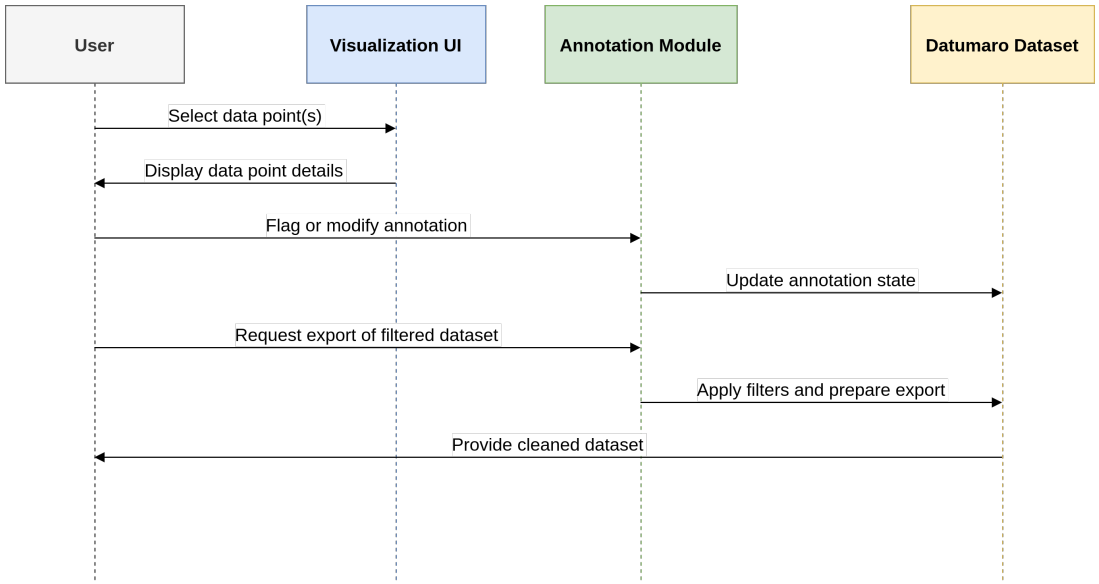

Figure 2.4: Annotation Sequence Diagram

1. **Flag Data Quality Issues**: The interface enables users to identify and tag problematic data points through three key actions: (1) marking mislabeled items via manual review or model-assisted suggestions, (2) detecting outliers in embedding space through interactive visualizations, and (3) flagging cross-modal misalignments (e.g., mismatched image-text pairs) using similarity thresholds. Flagged entries are logged with user-defined categories for systematic review.
2. **Export Cleaned Data**: Users can generate refined datasets by applying filters to exclude flagged items or incorporate corrected labels. The tool supports seamless export to multiple formats (COCO, YOLO, etc.) via Datumaro integration, ensuring compatibility with downstream training pipelines. New dataset versions retain provenance tracking for reproducible workflows.

5. Weakly Supervised Learning Support

To support weakly supervised learning workflows, we plan to implement the following features.

1. **Label Propagation**: The system enables semi-automatic annotation by propagating labels from confidently labeled examples to similar unlabeled data points. Using embedding similarity as the distance metric, it supports both threshold-based propagation (where labels spread to samples within a specified similarity range) and k-nearest neighbor approaches (where each unlabeled point inherits its neighbors' majority label). This significantly reduces manual labeling effort while maintaining reasonable accuracy.
2. **Pseudo-labeling**: The tool automatically generates candidate labels through embedding space clustering, where data points grouped in dense regions receive consistent labels. Users can manually verify and correct these suggestions before finalizing. The verified pseudo-labels can then be exported in standard formats for model training, creating a closed-loop workflow that improves with each iteration as the model and embeddings become more accurate.

2.4.4 | Folder & File Structure

The application's core logic is organized into functional modules.

1. Core Modules

- `src/core/embeddings/` - Embedding management:
  - `extractor.py` - Base embedding extraction
  - `storage.py` - Efficient storage/retrieval

- `similarity.py` - Similarity metrics
- `models/` - Model architecture adapters

- `src/core/dimensionality/` - Dimensionality reduction:
  - Implementations: `pca.py`, `umap.py`, `tsne.py`
  - `factory.py` - Reducer instantiation
- `src/core/visualization/` - Visualization tools:
  - `projections.py` - 2D/3D projection
  - `cross_modal.py` - Cross-modal algorithms
- `src/core/annotation/` - Data annotation:
  - `flagging.py` - Quality control
  - `propagation.py` - Label propagation

2. Framework Integrations

- `src/datumaro_ext/` - Datumaro extensions
- `src/otx_ext/` - OTX adapter implementations

3. User Interface

- `src/ui/streamlit_app/` - Main Streamlit application
- `src/ui/components/` - Interactive UI components
- `src/ui/layouts/` - View layouts

4. Utilities

- `src/utils/` - Shared utilities:
  - `caching.py` - Cache management
  - `profiling.py` - Performance tools

2.4.5 | System Design Enhancements for Scalability

The following scalability features represent preliminary design considerations. Final implementation will be determined after mentor discussions and further technical analysis.

1. **Modular Component Architecture**: The system's foundation supports scalability through deliberate separation of concerns: (1) Core business logic is isolated from UI/framework integrations, preventing scalability bottlenecks in one layer from affecting others; (2) Pluggable components (extraction, reduction, visualization) utilize interface-based design and factory patterns, enabling independent scaling of functional units; (3) Framework adapters maintain clean boundaries between core functionality and third-party integrations.
2. **Performance Optimization**: Several strategies are envisioned for efficient large-scale processing: Lazy loading minimizes memory footprint by fetching only required data and streaming embeddings directly from storage. Chunking and pagination techniques break down operations (processing/visualization) into manageable batches. A multi-level caching system (memory/disk) accelerates frequent operations like projection recalculations. Parallel processing capabilities through multiprocessing and worker pools optimize CPU-bound tasks like batch embedding extraction.

3. **Distributed Processing Support**: For extreme-scale scenarios, the architecture permits horizontal scaling: Worker nodes could distribute embedding extraction via queue-based job systems, with container orchestration (Docker/Kubernetes) support. Distributed storage backends (S3/HDFS) would enable transparent access to datasets exceeding single-node capacity through configurable storage interfaces. Refer Figure 2.5.

4. **API-First Design**: Future API capabilities may include: A RESTful interface exposing core functionality with auth/rate limiting for secure integration; asynchronous processing of long-running operations with WebSocket-based progress updates; and task status tracking for distributed workflows—all facilitating integration with external systems while maintaining scalability.

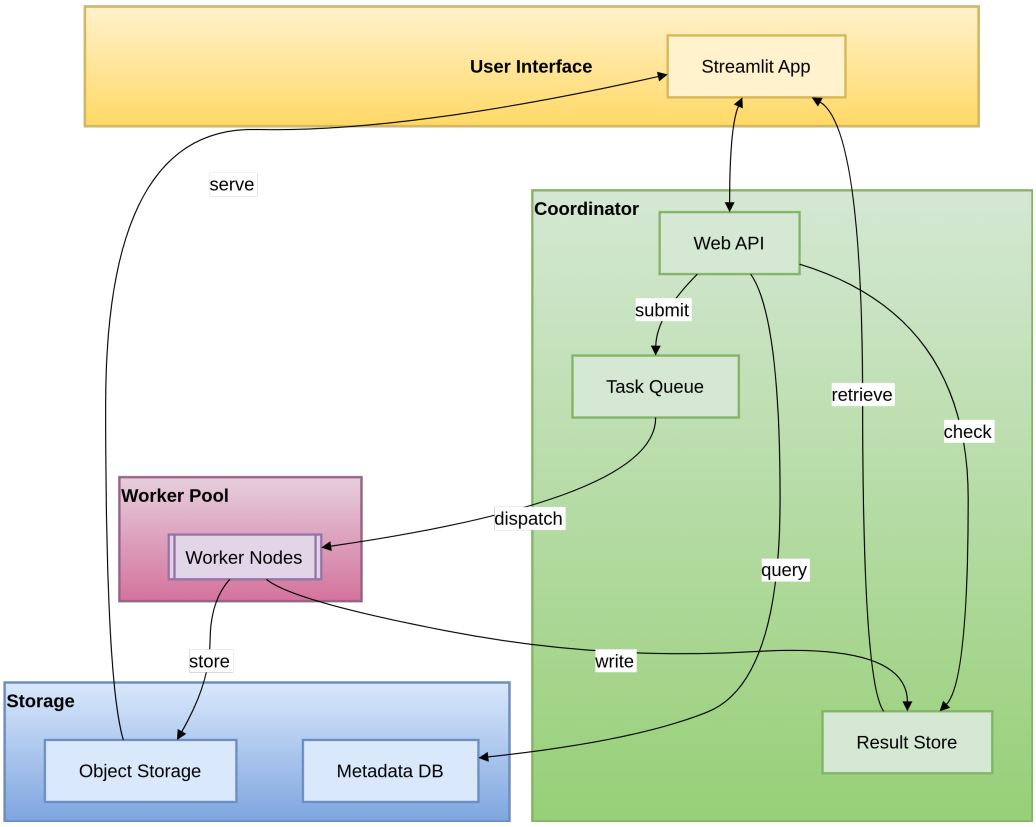

Figure 2.5: Distributed Processing Support

2.5 | Project Timeline & Commitments

Given that the project demands approximately 350 hours, this timeline ensures successful completion with buffer time accounted for. The schedule aligns perfectly with my availability - full-time focus during summer break followed by sustainable part-time hours when academic classes resume.

We can break the project solution into the following steps and milestones.

Timeline/Phase	Activities
Phase 0: Community Bonding	**Goals:** Lay the foundation for successful development through planning and alignment

Timeline/Phase	Activities
Week 1 (May 8-14)	☐ Hold kickoff meetings with mentors to clarify project scope, success metrics, and communication protocols ☐ Set up the development environment including Datumaro in plugin development mode, OTX inference pipelines, and Streamlit for prototyping ☐ Conduct a thorough exploration of the existing codebase, focusing on Datumaro's dataset manipulation APIs and OTX's model integration patterns
Week 2 (May 15-21)	☐ Draft a comprehensive technical design document covering: ☐ Embedding storage architecture (HDF5 vs. memory-mapped arrays) ☐ Data flow between visualization components and Datumaro ☐ Performance benchmarks for target dataset scales ☐ Build a prototype demonstrating basic embedding extraction using OTX with sample COCO dataset
Week 3 (May 22-28)	☐ Finalize UI/UX wireframes for the Streamlit application, incorporating mentor feedback ☐ Include main visualization panel, control sidebar, and data inspector layouts ☐ Define interaction patterns for cross-modal exploration ☐ Create a granular milestone tracker with weekly targets for Phase 1
Week 4 (May 29 - June 1)	☐ Prepare the coding kickoff plan including: ☐ Risk assessment of technical dependencies ☐ Contingency plans for potential bottlenecks ☐ Documentation standards and code review process
Phase 1: Core Infrastructure	**Deliverable:** Functional embedding management system with basic visualization
Week 5 (June 2-8): Datumaro Integration	☐ Implement the Datumaro plugin architecture for embedding storage, featuring: ☐ HDF5-based storage with chunked writing for large datasets ☐ Metadata indexing for efficient querying ☐ Unit tests covering 100K+ embedding scenarios ☐ Develop initial Streamlit components: ☐ Dataset selector widget supporting common formats (COCO, YOLO) ☐ Basic Plotly scatter plot with pan/zoom functionality
Week 6 (June 9-15): OTX Integration	☐ Build the embedding extraction pipeline: ☐ Adapter layer for CLIP (image/text) via OTX ☐ Asynchronous batch processing with progress tracking ☐ Cache invalidation logic for updated datasets ☐ Enhance the visualization interface: ☐ Color-coding by class labels and metadata attributes ☐ Interactive tooltips showing image thumbnails/text snippets

Timeline/Phase	Activities
Week 7 (June 16-22): Dimensionality Reduction	☐ Implement projection backends: ☐ Swappable reducers (PCA, UMAP, t-SNE) with consistent API ☐ GPU acceleration for UMAP via RAPIDS (if available) ☐ Projection caching to avoid recomputation ☐ Optimize frontend rendering: ☐ Performance improvements for 10K+ data points ☐ Clear axis labeling with explained variance metrics
Week 8 (June 23-29): Interaction & Refinement	☐ Develop advanced filtering capabilities: ☐ Lasso/box selection for subset creation ☐ Metadata-based filters (e.g., confidence thresholds) ☐ Conduct system testing: ☐ Memory profiling under heavy loads ☐ Validation using OpenImages-V6 dataset
Week 9 (June 30 - July 4): Midterm Preparation	☐ Produce a 5-minute demo video showcasing: ☐ End-to-end workflow from dataset import to visualization ☐ Key features completed in Phase 1 ☐ Draft user documentation: ☐ Installation guide ☐ Core feature explanations
Phase 2: Cross-Modal Features	**Deliverable:** Advanced analysis tools and annotation workflows
Week 10 (July 5-11): Cross-Modal Visualization	☐ Implement alignment heatmaps: ☐ Batch similarity matrix for image-text pairs ☐ Interactive cell selection to inspect mismatches ☐ Build parallel coordinate plots: ☐ Side-by-side comparison of modality embeddings
Week 11 (July 12-18): Annotation System	☐ Develop the data flagging interface: ☐ Context menu for marking issues (mislabeling, outliers) ☐ Persistent storage of flags in Datumaro format ☐ Complete midterm evaluations
Week 12 (July 19-25): Weak Supervision	☐ Create label propagation tools: ☐ KNN-based semi-automatic labeling ☐ Confidence thresholding UI ☐ Implement dataset export: ☐ Cleaned datasets in standard formats ☐ Version tracking for iterations

Timeline/Phase	Activities
Week 13 (July 26 - Aug 1): Performance Optimization	☐ Add lazy loading support: ☐ On-demand embedding loading for 50K data points ☐ Integrate approximate nearest neighbor search: ☐ FAISS/Annoy for faster similarity queries
Phase 3: Polish & Documentation	**Deliverable:** Production-ready tool with comprehensive guides
Week 14 (Aug 2-8)	☐ Finalize pseudo-labeling integration ☐ Improve error handling and user feedback
Week 15 (Aug 9-15)	☐ Create tutorial notebooks covering: ☐ Typical exploration workflows ☐ Annotation best practices ☐ Conduct stress testing with 1M+ item datasets
Week 16 (Aug 16-22)	☐ Complete user documentation: ☐ API references ☐ Troubleshooting guide ☐ Hold final mentor review session
Week 17 (Aug 23-25)	☐ Prepare final demo materials ☐ Perform code cleanup and final testing
Buffer Period (Aug 26 - Sep 1)	☐ Address critical mentor feedback ☐ Submit final evaluation materials

3 | About OpenVino

3.1 | My Interactions with the OpenVino

I first explored OpenVINO with a practical goal: *Could I run a lightweight generative AI model locally on my Asus gaming laptop?* While large language models (LLMs) like GPT-4 typically require cloud infrastructure, I discovered through Reddit discussions that OpenVINO's optimizations enabled some users to run smaller models (such as SMOL, distilled LLMs, or TinyLlama) on local machines. Inspired by these examples, I attempted to do the same—only to find the challenge far more complex than I'd anticipated. During my experimentation, I encountered OpenVINO's NNCF quantization framework, which I studied with little to modest success.

I saw some issues on the JIT emitters component which I decided to work upon. My subsequent contributions to the JIT emitters were merged upstream, marking my first tangible impact on the project.

Through this journey with OpenVINO, I serendipitously discovered that the project would be participating in Google Summer of Code (GSoC) 2025. While I had been aware of GSoC for some time, this revelation felt particularly fortuitous—it presented the perfect opportunity to sharpen my applied programming skills and improve my ability to write impactful code.

3.2 | What I know about OpenVino

OpenVINO stands as a pioneering toolkit in artificial intelligence domain, developed by Intel to streamline the deployment and optimization of deep learning models across a diverse type of Intel hardware platforms. OpenVINO can be used in diverse scenarios, from powering intelligent edge devices to accelerating complex AI workloads in cloud environments. By providing a unified framework for model deployment and optimization, OpenVINO unlocks the access for both users and developers to easier and faster AI capabilities.

Several core components are lying under OpenVINO. From the frontend, Model Optimizer will convert models from well-known AI frameworks like PyTorch and TensorFlow into OpenVINO's own optimized Intermediate Representations (IR), ensuring compatibility and efficiency across different hardware architectures. Moreover, the NNCF will be leveraged to quantize and compress models if extra performance boost is needed. Furthermore, the OpenVINO Runtime will optimize the inference procedure with specifically tuned kernels or extern DL libraries. Besides, the Open Model Zoo offers a repository of pre-trained models, expediting development cycles and enabling rapid prototyping of AI applications.

In comparison to alternative AI inference frameworks, OpenVINO distinguishes itself through its deployment flexibility and superior inference efficiency. OpenVINO empowers developers to seamlessly deploy applications across an array of hardware platforms thanks to its automatic device discovery feature and compatibility with Linux, Windows, and MacOS environments. Furthermore, OpenVINO's streamlined architecture minimizes external dependencies and offers custom compilation options, resulting in reduced application footprint and simplified management. Additionally, its innovative approach to start-up time optimization, initiating inference on the CPU before transitioning to other devices, effectively minimizes latency in real-world scenarios.

In short, OpenVINO represents a set of cutting-edge technologies on AI deployment and optimization, with a set of user-friendly documents and examples to begin with. It demonstrates the efforts the open-source world is making towards better AI utilization, which is promising and will prosper and thrive.

3.3 | Previous Contributions to the OpenVino Project

I started exploring OpenVino from November 2024, diving into the codebase and then later on started with my contributions I have done in total 17 contributions for OpenVino, Understanding the codebase throughly Even though I am still a learner and looking forward to have more meaningful contributions throughout. You can see my contribution graphs below. I have contributed to keras repository under the Openvino src folder and done my contributions in openvino widely.

I have successfully contributed to pull requests to the OpenVINO codebase, both focusing on JIT emitter implementations for ARM64 SIMD platforms.

1. **NotEqual Operation JIT Emitter**:Merged
   - Issue: https://github.com/openvinotoolkit/openvino/issues/27516
   - PR: https://github.com/openvinotoolkit/openvino/pull/28257
   - Implemented fp32 NotEqual operation using ARM64 NEON instructions
   - Added emitter class and integrated with ARM64 executor/kernel
   - Included comprehensive unit tests and performance benchmarks
2. **FloorMod Operation JIT Emitter**:Merged
   - Issue: https://github.com/openvinotoolkit/openvino/issues/27501
   - PR: https://github.com/openvinotoolkit/openvino/pull/27706
   - Developed optimized fp32 FloorMod implementation replacing C++ Math
   - Modified ARM64 executor and kernel to support new emitter
   - Enhanced test coverage and transitioned operation to Eltwise type

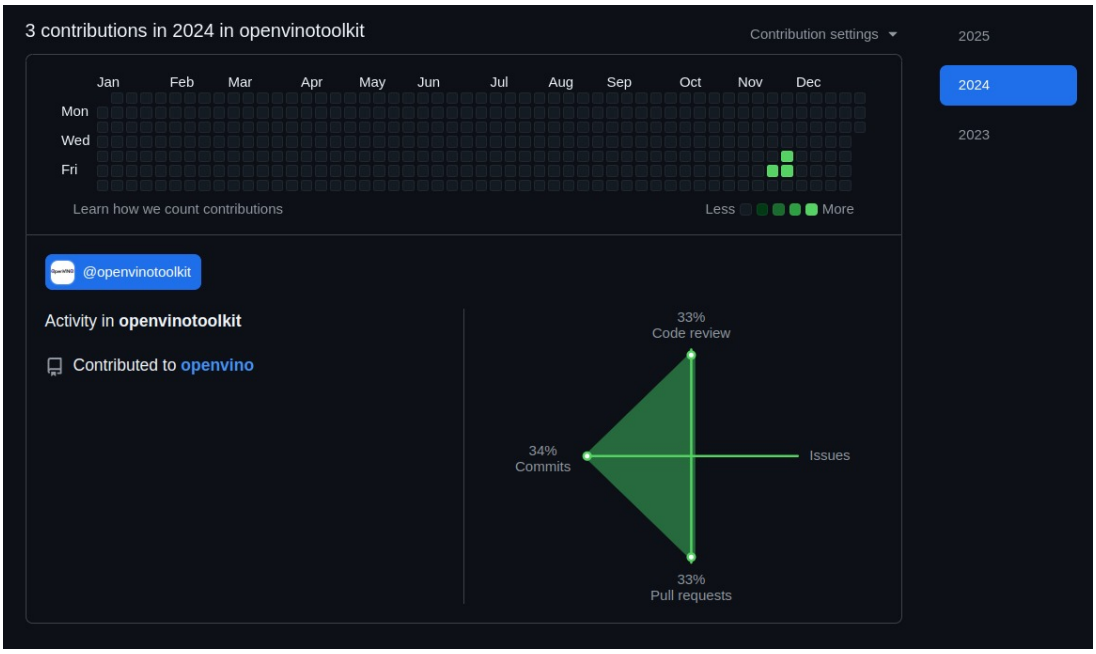

Figure 3.1: Contribution Graph in OpenVino

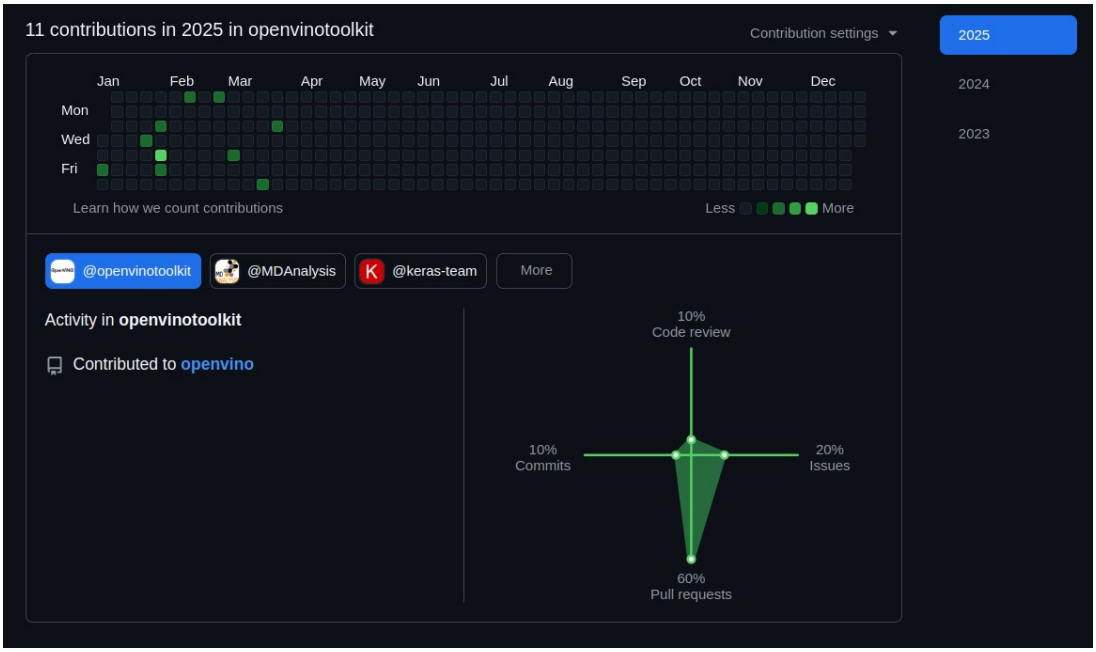

Figure 3.2: Contribution Graph in OpenVino

Six Pull requests including open and closed both :

- PR:https://github.com/openvinotoolkit/openvino/pull/29123
- PR: https://github.com/keras-team/keras/pull/21087
- PR: https://github.com/openvinotoolkit/openvino/pull/28889
- PR: https://github.com/keras-team/keras/pull/21028
- PR : https://github.com/openvinotoolkit/openvino/pull/28752
- PR : https://github.com/openvinotoolkit/openvino/pull/28599

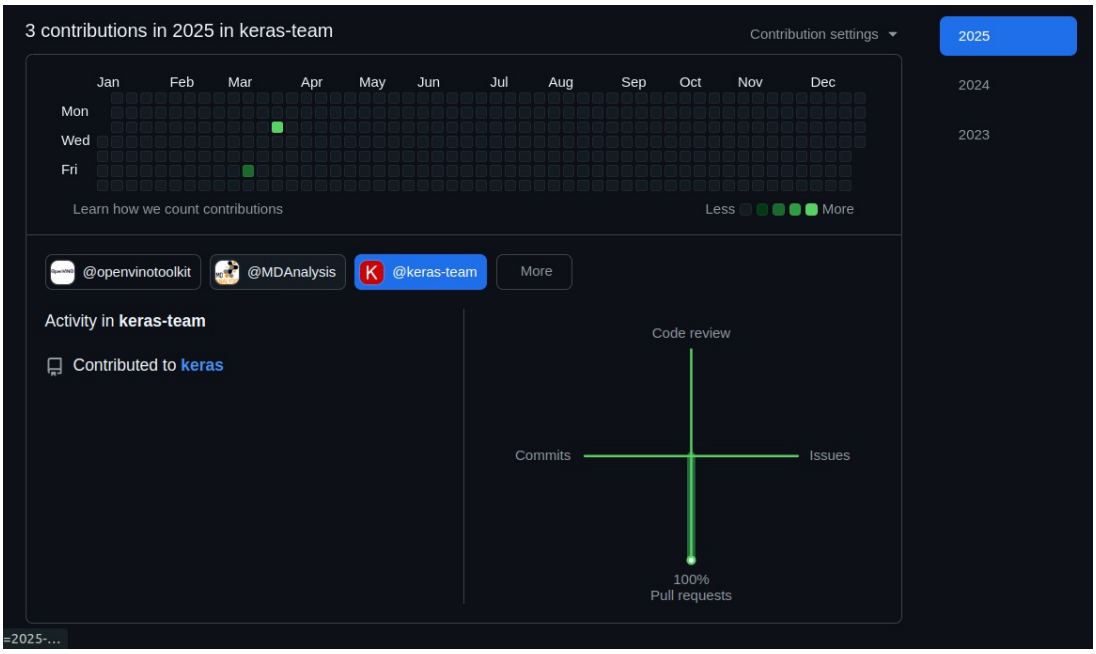

Figure 3.3: Contribution Graph in Keras Team for OpenVino

My Discussion in OpenVino:

- Discussion Link: https://github.com/openvinotoolkit/openvino/issues/27501

The discussion included reaching out to mentors during the submissions of the GSOC proposal for the Project 9.

4 | General Discussion

4.1 | Why am I a Great Fit?

This project demands more than just technical competence—it requires an obsessive attention to the why behind the code. With the curiosity, learning determination and the ability to self-learn things at a decent pace, I believe I am a very suitable to an ideal candidate for this project.

What excites me isn't just building tools, but engineering leverage—creating interfaces that help others spot patterns they'd otherwise miss. This project's real value lies in transforming abstract embeddings into something a researcher can interrogate. With my understanding in ML/DL with web development, it is at this project precisely where my experiences converge.

4.2 | My Career Plans Ahead

The world is rapidly embracing AI automation, and I aim to deeply skill myself in this space—with most likely pursuing an MS degree to strengthen my expertise. While the current technological disruption creates uncertainty, I believe that those with strong fundamentals in computer science and mathematics will remain and aren't going to be obsolete anywhere anytime soon. Regardless of how AI evolves, core principles like efficient algorithms, statistical reasoning, and systems design will continue to underpin progress.

Once the initial wave of AI-driven displacement stabilizes, I'll assess where I can contribute most meaningfully. For now, my focus is on mastering the tools and theories that empower AI innovation.

4.3 | Describe any other career development plan you have for the summer in addition to GSoC.

This summer, in addition to participating in GSoC, I plan to enhance my problem-solving abilities by regularly tackling challenges on various coding platforms such as LeetCode and Codeforces. This consistent practice will not only prepare me for technical interviews but also strengthen my understanding of data structures and algorithms. Given that internships are a mandatory component of my university curriculum for earning the required credits, dedicating time to these areas will be instrumental in securing a suitable position. By focusing on these skills, I aim to be well-prepared for both my upcoming internship applications and the professional challenges that lie ahead.

4.4 | How could you apply it to your professional development?

Engaging in the "Interactive Multimodal Data Explorer" project offers a comprehensive opportunity to enhance my professional skills. By integrating advanced machine learning models like CLIP and GPT-4V with user-friendly visualization tools, I will deepen my understanding of embedding generation and alignment. Developing intuitive interfaces for data exploration will refine my ability to present complex information clearly, a crucial skill in data science. Collaborating with mentors and peers will expand my professional network and improve my teamwork and communication abilities. Contributing to an open-source project like Datumaro demonstrates my commitment to community-driven development and continuous learning. Additionally, integrating FAISS for efficient similarity search will expose me to scalable indexing and retrieval techniques, essential for handling large datasets. This experience will strengthen my portfolio and prepare me for advanced roles in the tech industry, equipping me with the skills necessary to tackle complex data challenges effectively.

5 | Additional Resources

Here are some of the additional resources that were taken into consideration as inspiration for the proposal.
"WizMap: Scalable Interactive Visualization for Exploring Large Machine Learning Embeddings"

- Link: https://arxiv.org/abs/2306.09328

"Embedding Projector: Interactive Visualization and Interpretation of Embeddings"

- Link: https://arxiv.org/abs/1611.05469

"EmbeddingTree: Hierarchical Exploration of Entity Features in Embedding"

- Link: https://arxiv.org/abs/2308.01329

"Chart2Vec: A Universal Embedding of Context-Aware Visualizations"

- Link: https://arxiv.org/abs/2306.08304