Advancing Plankton Monitoring

AUTOMATIC IMAGE ANALYSIS WITH PLANKTOSCOPE IN THE PLANDYO PROJECT

Jaronchai Dilokkalayakul
Information Biology Laboratory,
Tohoku University **東北大学**

AGENDA

1. *Introduction and Objectives*
2. *End-to-End Overview*
3. *Samples and Planktoscope*
4. *Auto-Imaging Pipeline*
5. ***Classification Methodologies***
   with a focus on Gen-AI and Multimodal LLM
6. *Results and Next Steps*

With an Appendix after page 30

INTRODUCTION ABOUT ME

JARONCHAI DILOKKALAYAKUL

***Laboratory Lunch Party**, December 3, 2024*

- ***Graduate School of Information Science***, Information Biology Laboratory, Tohoku University, Japan
- *Data Engineering,* Bachelor of Engineering, International Program, Thai-Nichi Institute of Technology, Thailand

- **IT System Engineer, *Info-Bio Lab***
  November 2024 – Present
- **Data Engineer, *IBM***
  May 2024 - October 2024
- **Data Engineer Intern, *IBM***
  June 2023 - November 2023
- Junior AI Researcher, *TNI*
  November 2022 – April 2023
- Computer Laboratory Assistant, *TNI*
  March 2022 – April 2023
- Programming Teaching Assistant, *TNI*
  November 2021 – March 2022

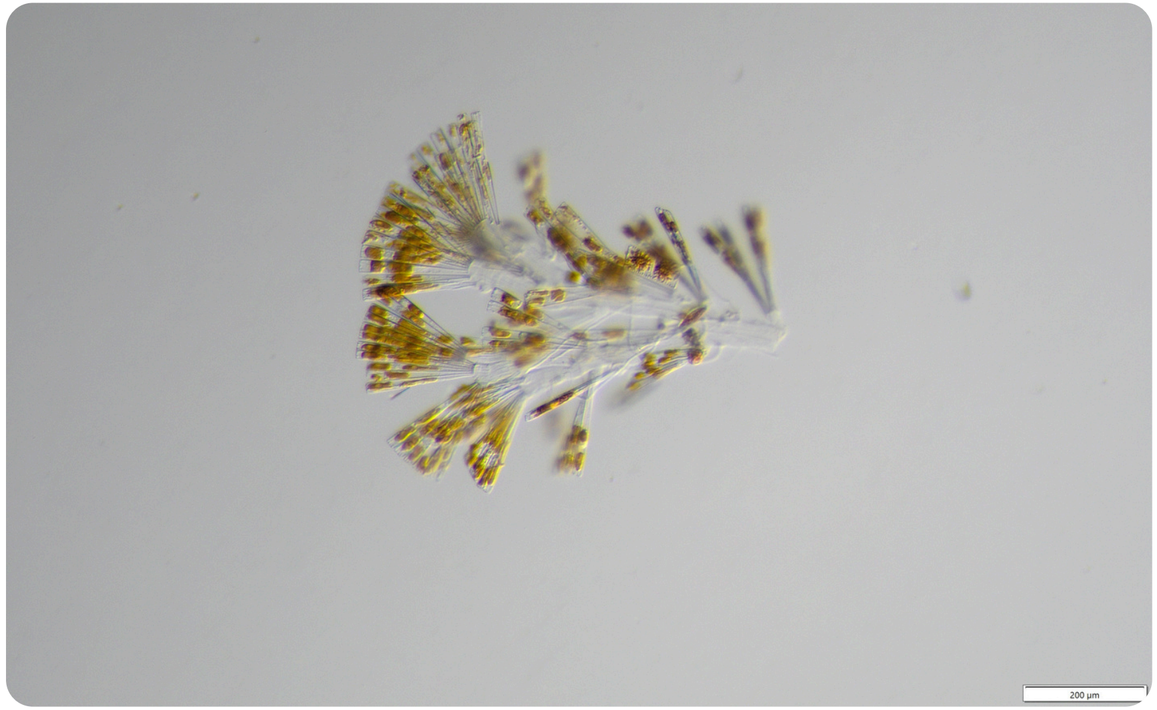
INTRODUCTION TO THIS PROJECT

Objectives and Goals

Develop an **AI-powered analysis pipeline** to identify taxonomy and analyze **plankton trends and their influence on the marine ecosystem** along with LLMs technologies.

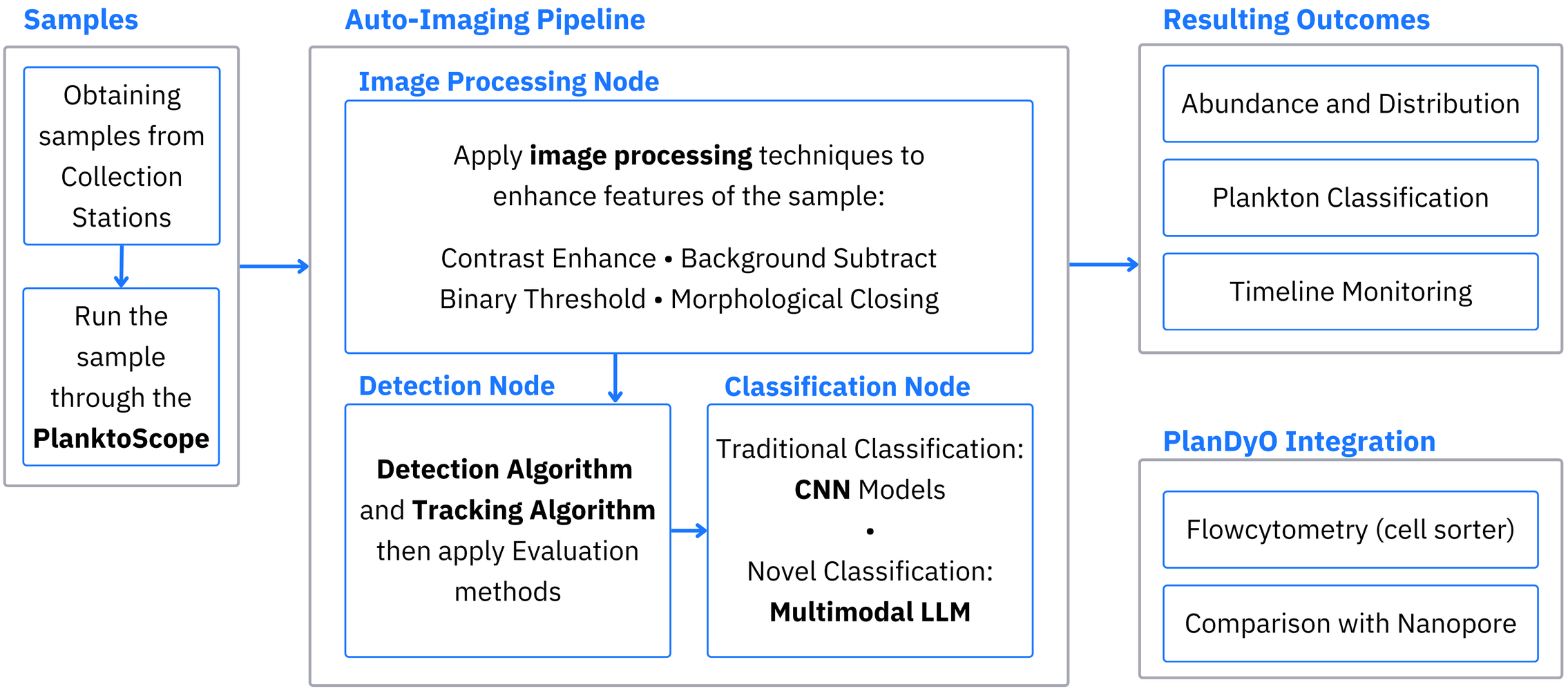
Integration to PlanDyO Project

Improve marine biodiversity studies and support monitoring by automating plankton analysis - Finding the dynamic of an area, and the function of individual species

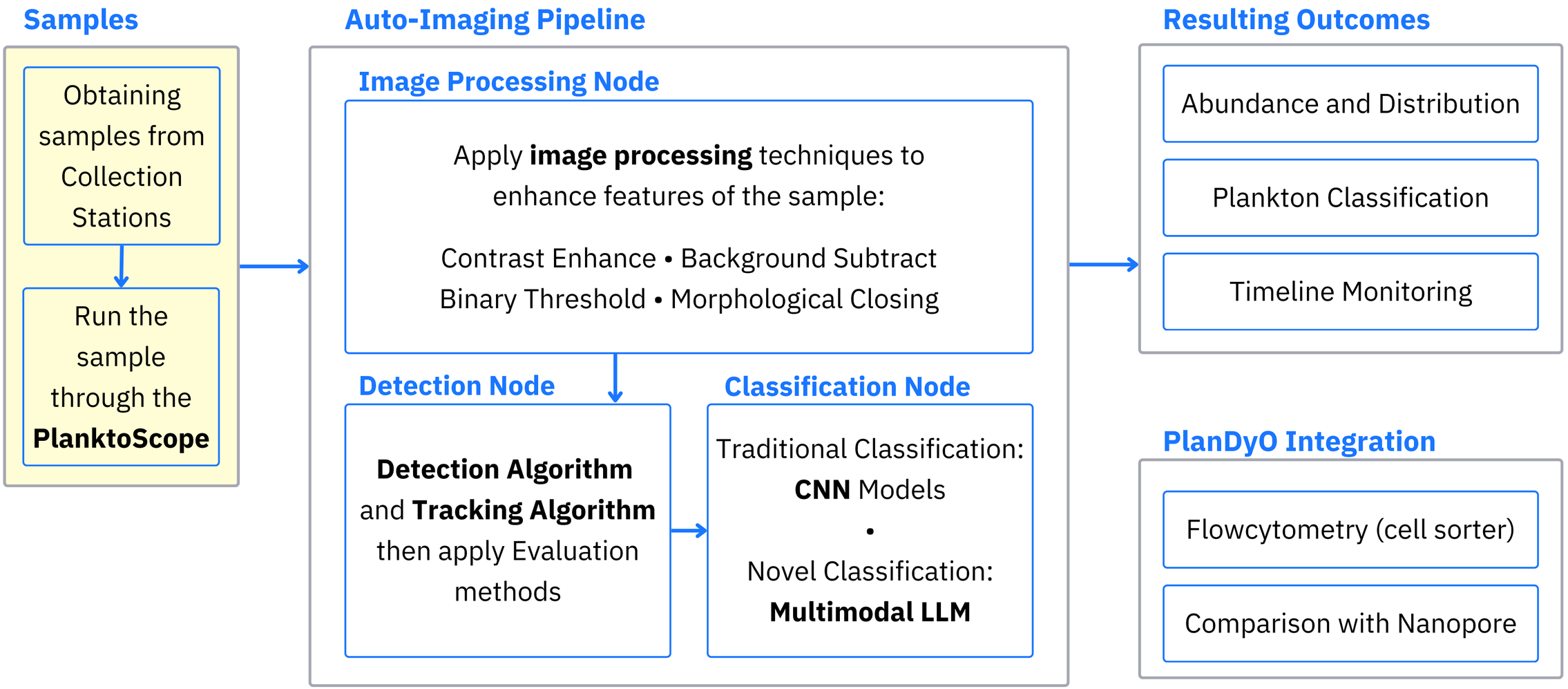


Licmophora Diatom *Image via Olympus Microscope, from Mutsu Bay, Aomori, courtesy of Akane-san*

END-TO-END OVERVIEW OF THE PROJECT



COLLECTING SAMPLES AND PLANKTOSCOPE

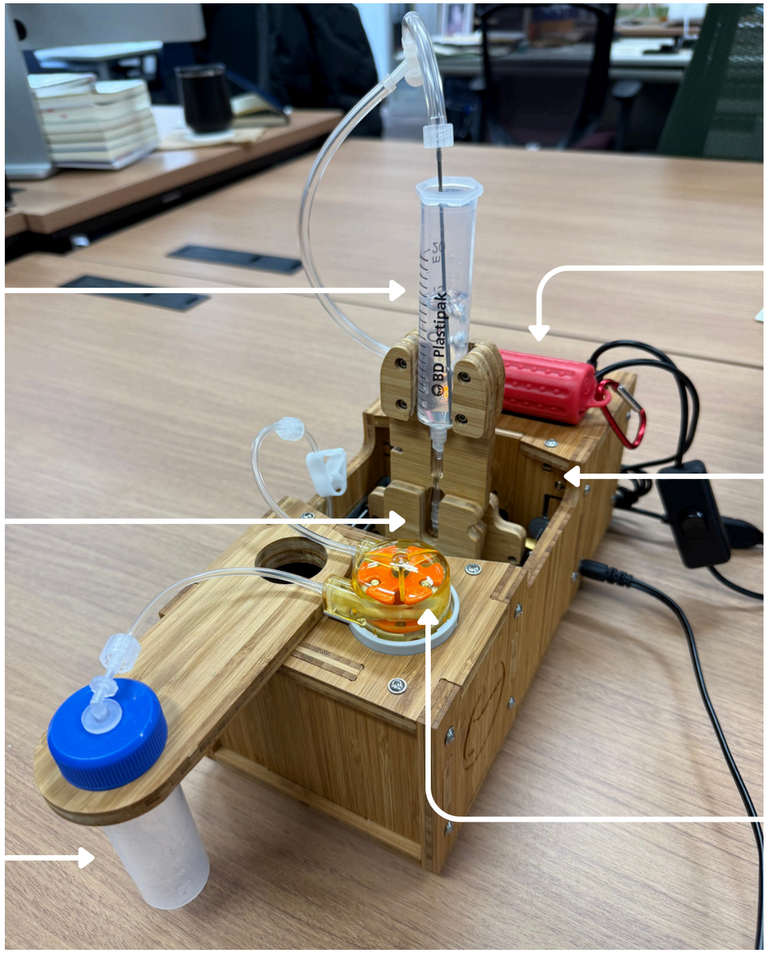


ANATOMY AND USAGE OF A PLANKTOSCOPE

Sample

Flow-Cell

Trash

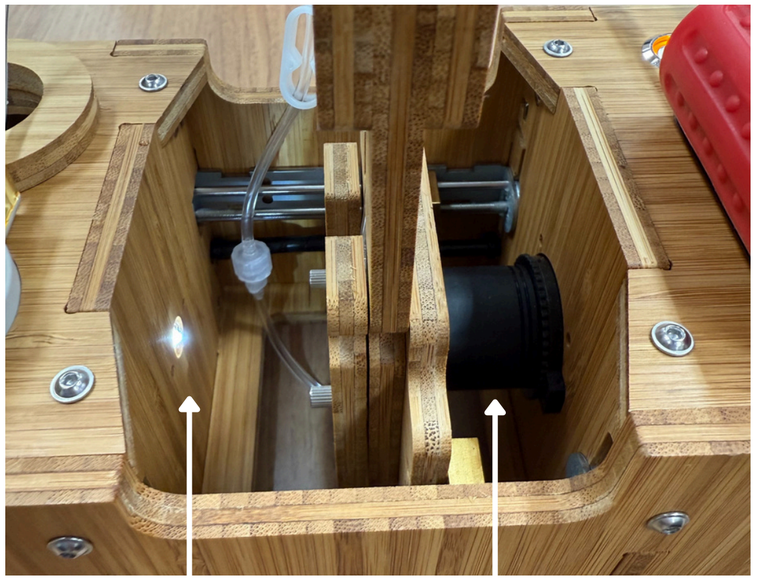


Air Pump

Raspberry-Pi
(Inside)

Peristaltic
Pump

3 milliliters per 1 Sample



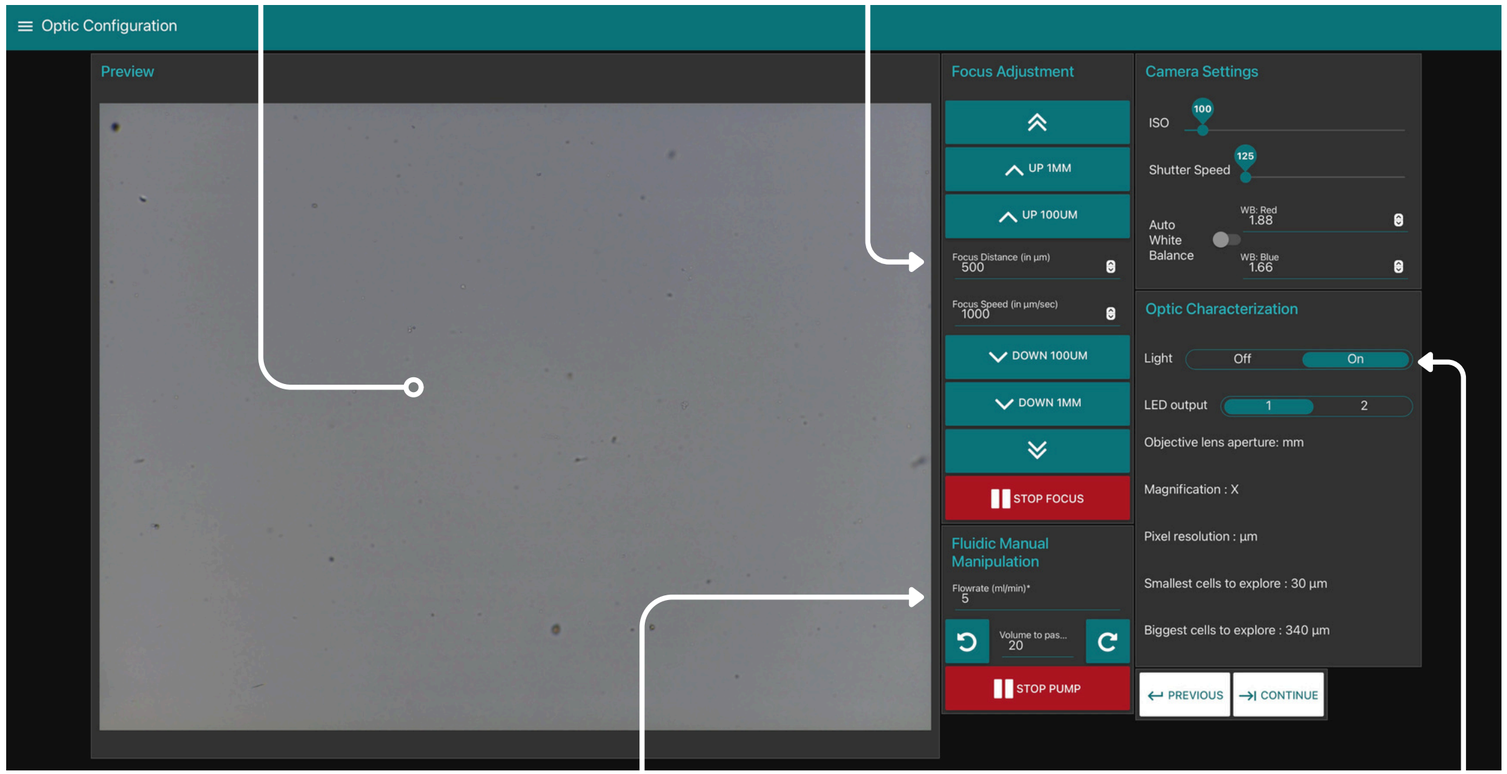
LED

Camera

PLANKTOSCOPE OPERATION

Capture Screen

Optic Configuration



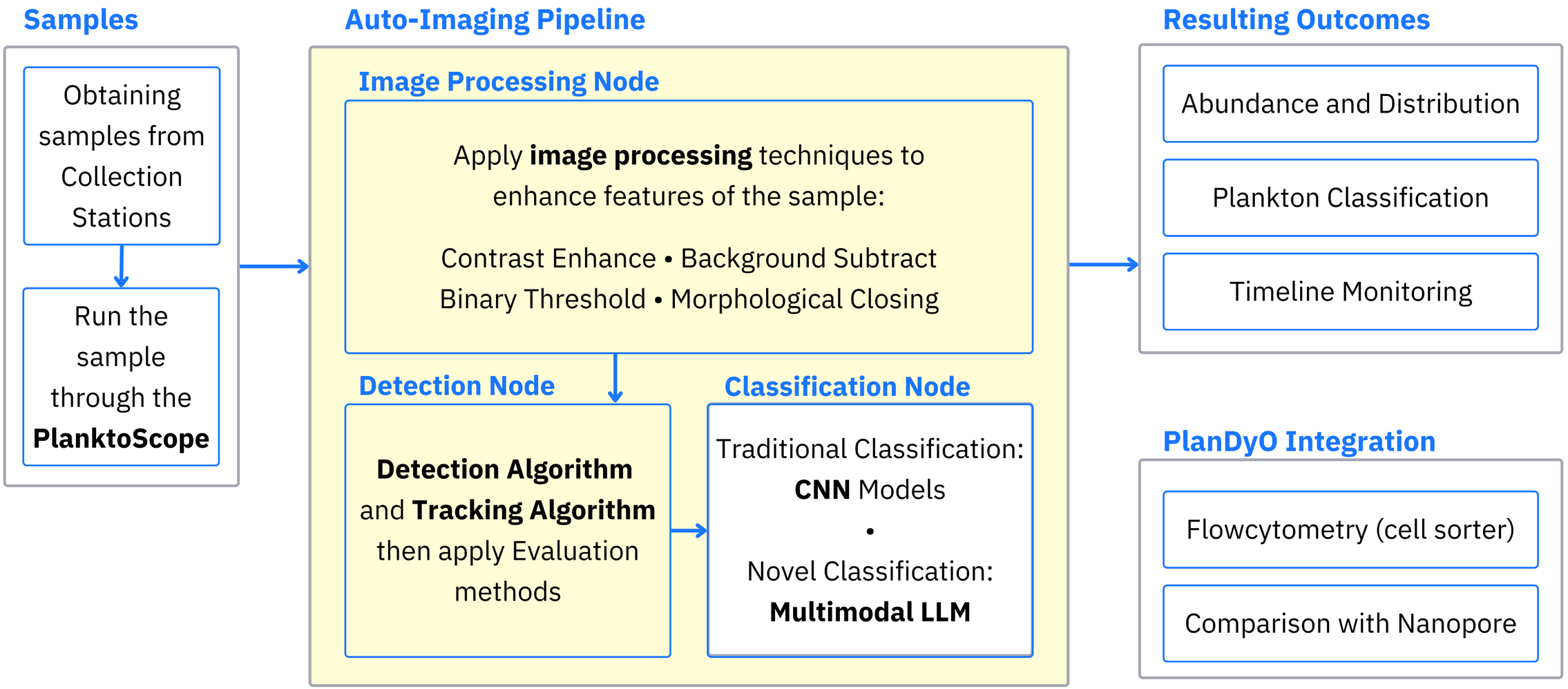
Manual Fluid Manipulation

LED Output

Flow rate: 0.05ml/min

Add raw data

AUTO-IMAGING AND MONITORING PIPELINE



PLANKTOSCOPE SEGMENTATION VS VIDEO + LLM

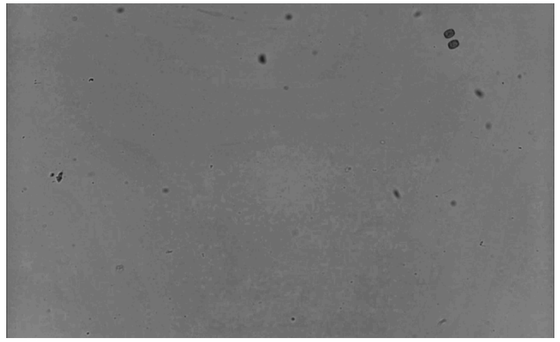
Feature	PlanktoScope (Image-Based)	New Method (Video + LLM)
Input Data	Static images	Continuous video frames
Speed & Efficiency	Limited by single-image processing	Faster due to frame-to-frame consistency
Output Format	Static segmentation	Context-aware, richer taxonomic outputs
Motion Analysis	✕ Not possible	✅ Tracks movement trajectories
Environmental Factors	✕ Not considered	✅ Includes temperature, pH, angle
Behavioral Insights	✕ Limited to morphology	✅ Behavior changes based on factors

AUTO-IMAGING PIPELINE OVERVIEW

Image Processing Node

1 Contrast Enhance

Contrast Limited Adaptive Histogram Equalization (CLAHE)



2 Background subtract

Absolute difference and Guassian blurred background

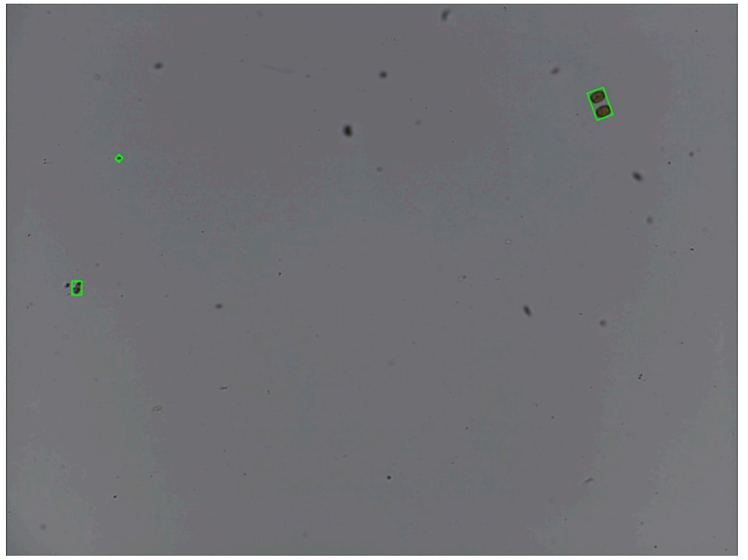
3 Binary threshold

Converts to binary, apply adaptive bg/fg threshold

4 Closed Binary

Uses **morphological closing** (dilation & erosion)

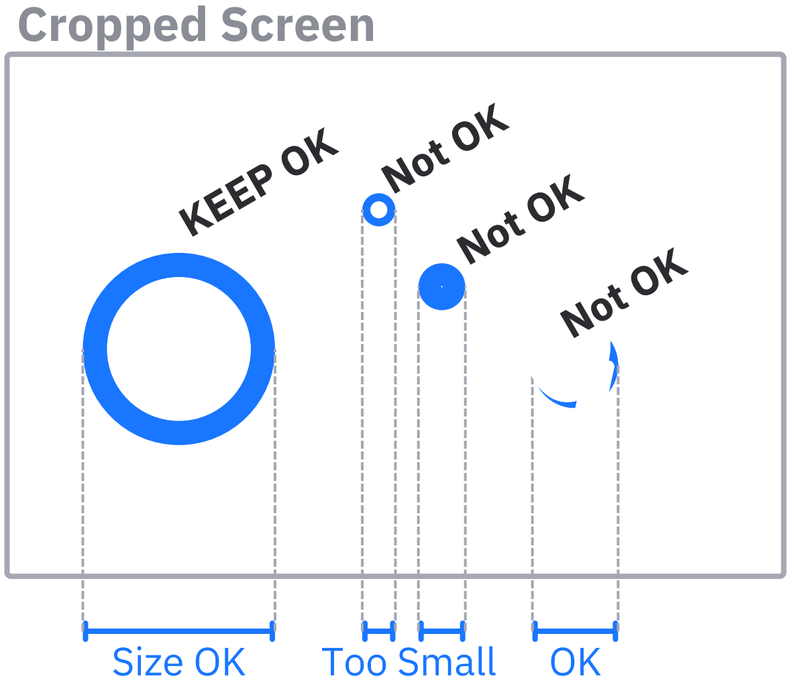
Detection Node



Classification Node

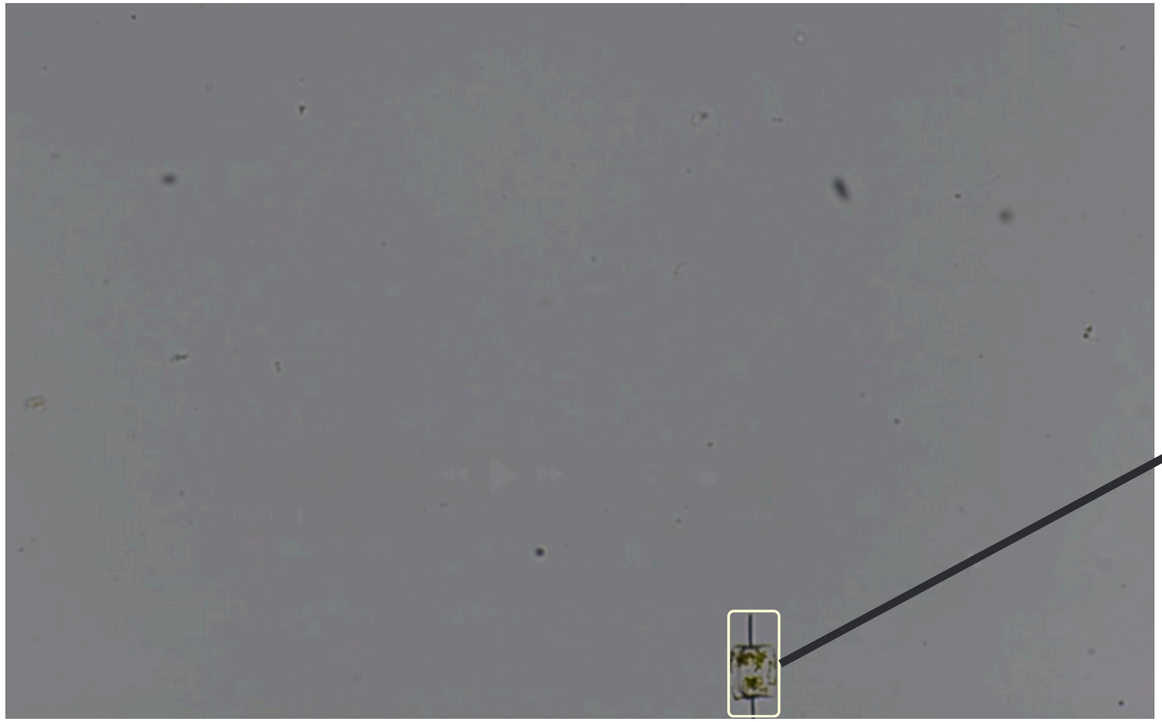
Next Section ..

PLANKTON DETECTION ALGORITHM



Detection → Segmentation → Tracking

IMAGE SEGMENTATION AND PADDING

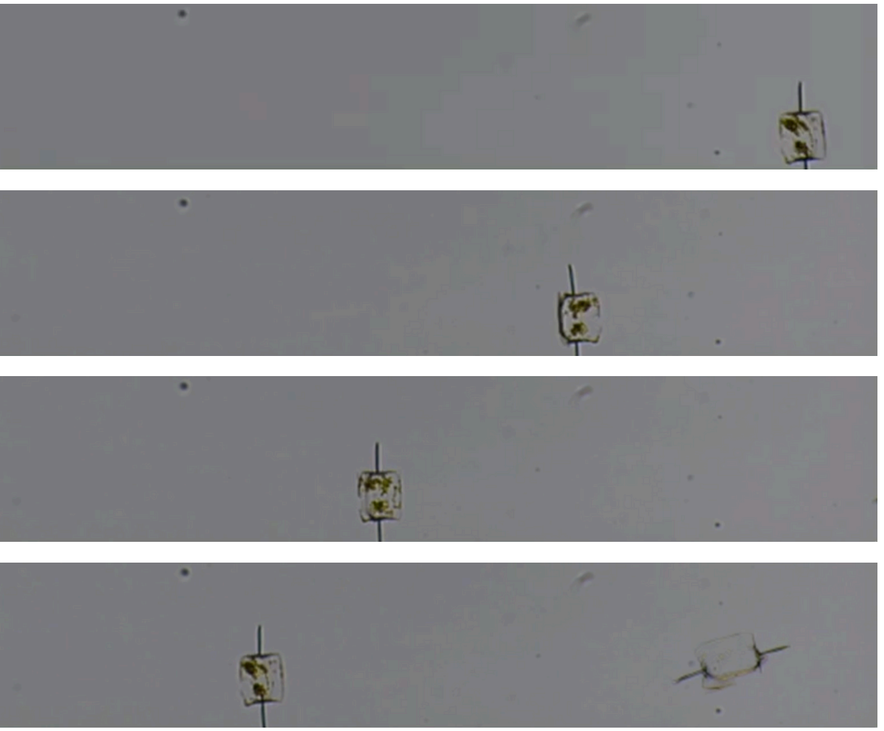


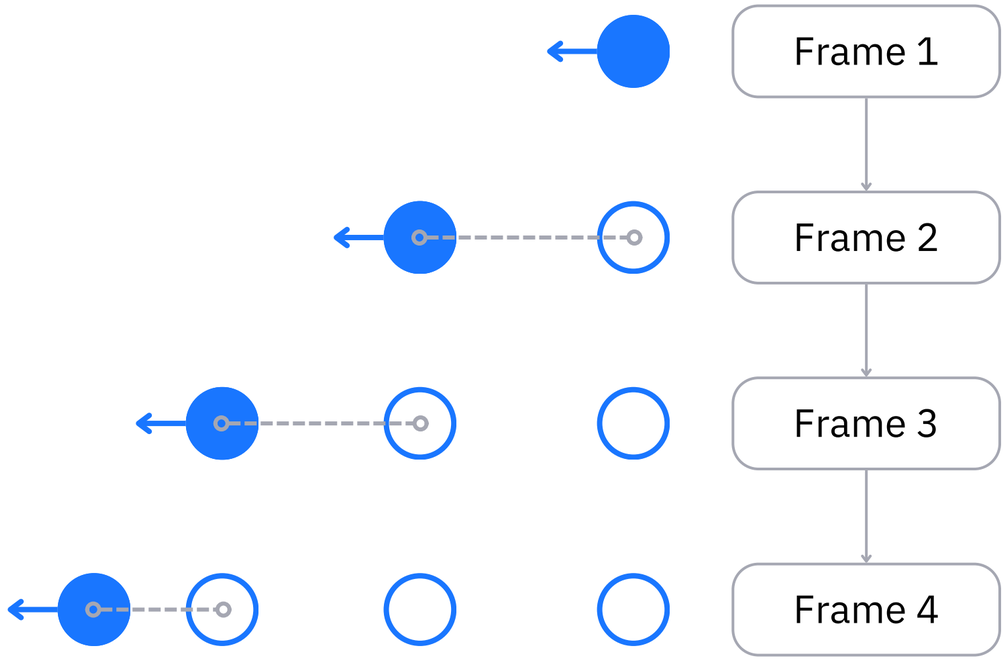
Segmented Image → Saved Image

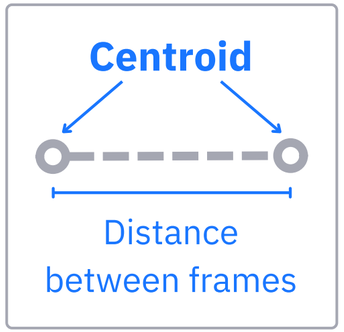
Padding for each plankton

Detection → Segmentation → Tracking

PLANKTON TRACKING ALGORITHM

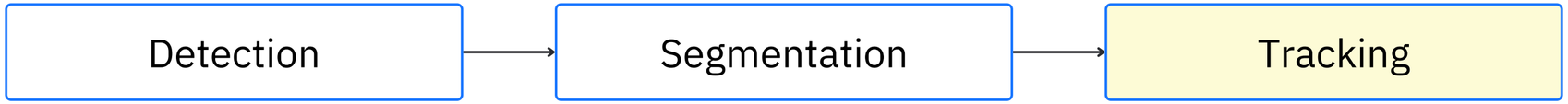




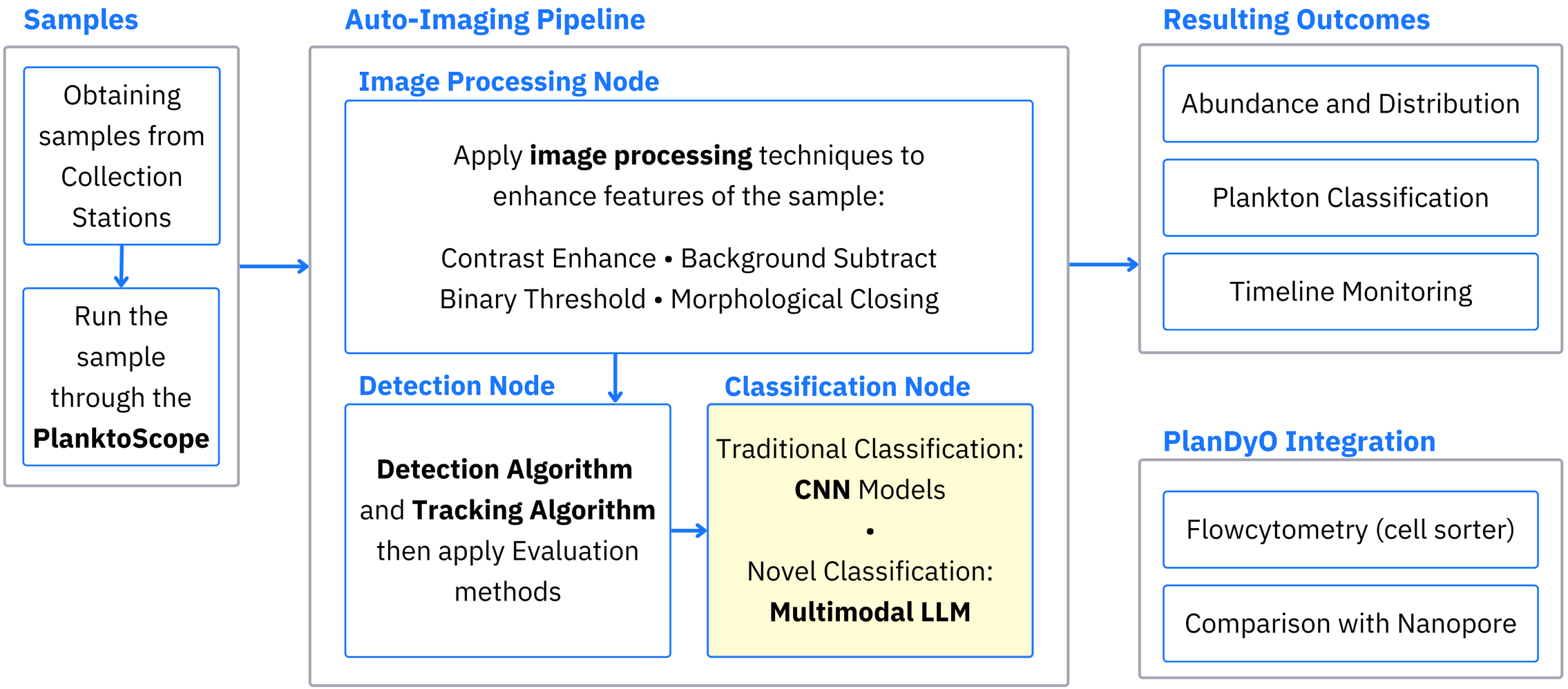


Setting distance threshold to consider one object as the same object

Remark: Plankton speed is a byproduct of this step



CLASSIFICATION METHODOLOGIES



CONVOLUTIONAL NEURAL NETWORK

2006 labeled IFCB images

No Thumbnail Available

Date

2006

Authors

Sosik, Heidi M.
Peacock, Emily E.
Brownlee, Emily F.

Linked Authors

Sosik, Heidi M.
Peacock, Emily E.
Brownlee, Emily F.

Files

2006.zip (700.08 MB)

Citable URI

https://hdl.handle.net/1912/7342

Description

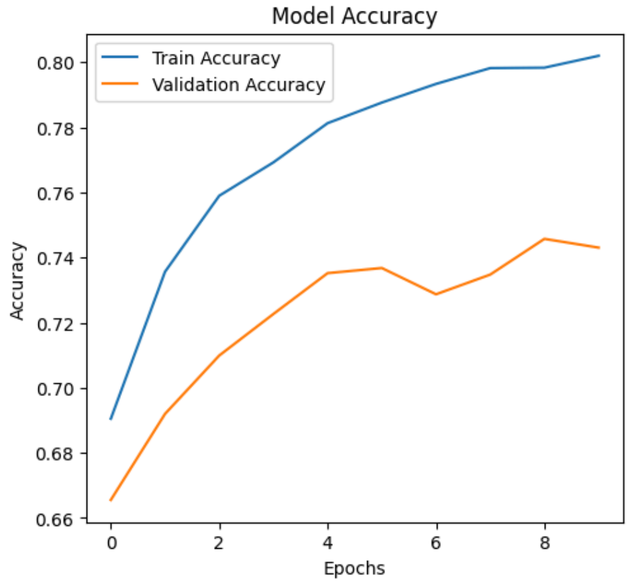
This zipped content contains Annotated Plankton Images from one year and is part of the WHOI-Plankton Collection that spans multiple years. Click on the WHOI-Plankton link below to view all items (other years) in this collection.

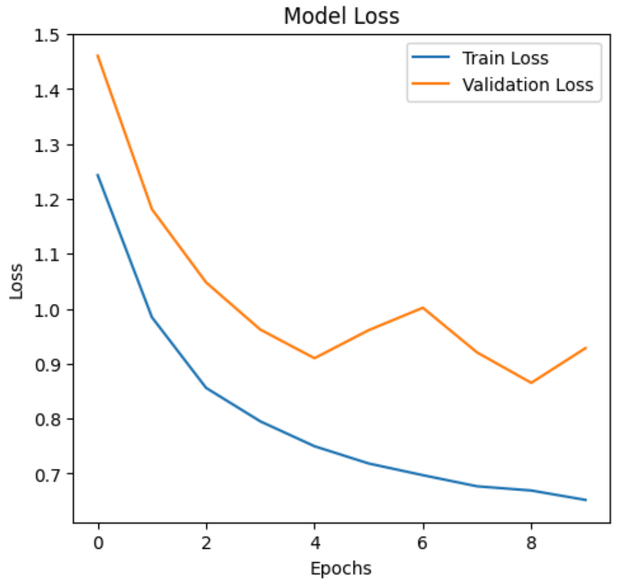
Collections

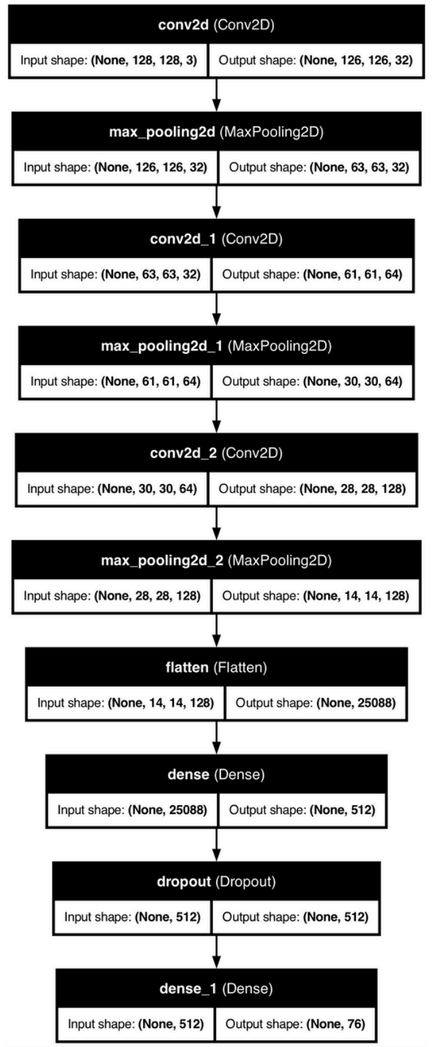
WHOI-Plankton

Full item page

Remark: There are observable skewness in the downloaded dataset



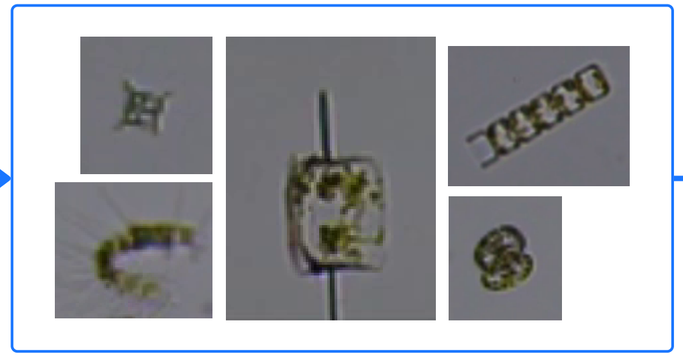




Generative AI and Large Language Model

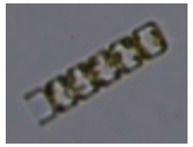
MULTIMODAL LLM IMPLEMENTATION

Detection Node Output



Output individual “Plankton” or “Non-Plankton” images

Gemini via VertexAI

Inquiry Prompt + 

LLM Output

JSON format plankton metadata

Goal of LLM Method

Plankton Classification and Metadata Generation

GUI (Graphic User Interface)

User-friendly, visual interaction (buttons, text fields)

API (Application Programming Interface)

Integrating into apps, automation, or backend

Testing to Implementation

JSON Format is

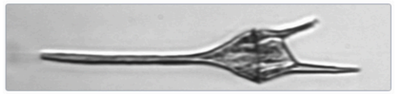
Structured, **machine-readable**, parsable, supports nested data

Integrates with APIs, databases, and applications

MODEL DIRECT COMPARISON

Gemini 2.0 Flash

System Instructions



You are an expert marine biologist with extensive knowledge of plankton taxonomy. You specialize in identifying and classifying various plankton species based on provided image data or textual descriptions.

```
You are classifying an organism to determine whether it is plankton. If it is, provide its taxonomic details (family, genus, species). If
it is not plankton, return `"none"`. The output must strictly adhere to JSON format for seamless integration into a structured
database.

            **Steps:**
            1. Analyze the given data (image, description, or other input) to determine if it belongs to the plankton category.
            2. If it is plankton, classify it as accurately as possible and provide the taxonomic details.
            3. If it is not plankton, return `"none"` in the response.
            4. Ensure the response strictly follows the JSON format.

            **Output Format:**
            ```json
            {
              "family": "Calanidae",
              "genus": "Calanus",
              "species": "Calanus finmarchicus"
            }
            ```
            If it is **not plankton**, return:
            ```json
            {
              "classification": "none"
            }
            ```

            Classify the given input according to the structured taxonomy above. If the organism is not plankton, return `"none"`. Do not
include any additional text or explanation—only the JSON output.
```

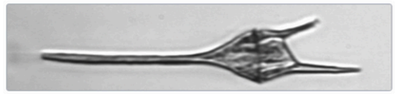
Use code with caution.

```
{
  "family": "Ceratiaceae",
  "genus": "Ceratium",
  "species": "Ceratium furca"
}
```

Use code with caution. Json

Gemini 2.0 Flash Thinking Experimental 01-21

System Instructions



You are an expert marine biologist with extensive knowledge of plankton taxonomy. You specialize in identifying and classifying various plankton species based on provided image data or textual descriptions.

You are classifying an organism to determine whether it is plankton. If it is, provide its taxonomic details (family, genus, species). If it is not plankton, return `"none"`. The output must strictly adhere to JSON format for seamless integration into a structured database.

Steps:
1. Analyze the given data (image, description, or other input) to determine if it belongs to the plankton category.
2. If it is plankton, classify it as accurately as possible and provide the taxonomic details.
3. If it is not plankton, return `"none"` in the response.
4. Ensure the response strictly follows the JSON format.

Output Format:
```json
{
  "family": "Calanidae",
  "genus": "Calanus",
  "species": "Calanus finmarchicus"
}
```
If it is **not plankton**, return:
```json
{
  "classification": "none"
}
```

Classify the given input according to the structured taxonomy above. If the organism is not plankton, return `"none"`. Do not include any additional text or explanation—only the JSON output.

Use code with caution.

Thoughts
*The thoughts produced by the model are experimental

Expand to view model thoughts

```
{
  "family": "Ceratiaceae",
  "genus": "Ceratium",
  "species": "Ceratium tripos"
}
```

Use code with caution. Json

Type something

Run ⌘↵

Adjustable Variables

Temperature

0

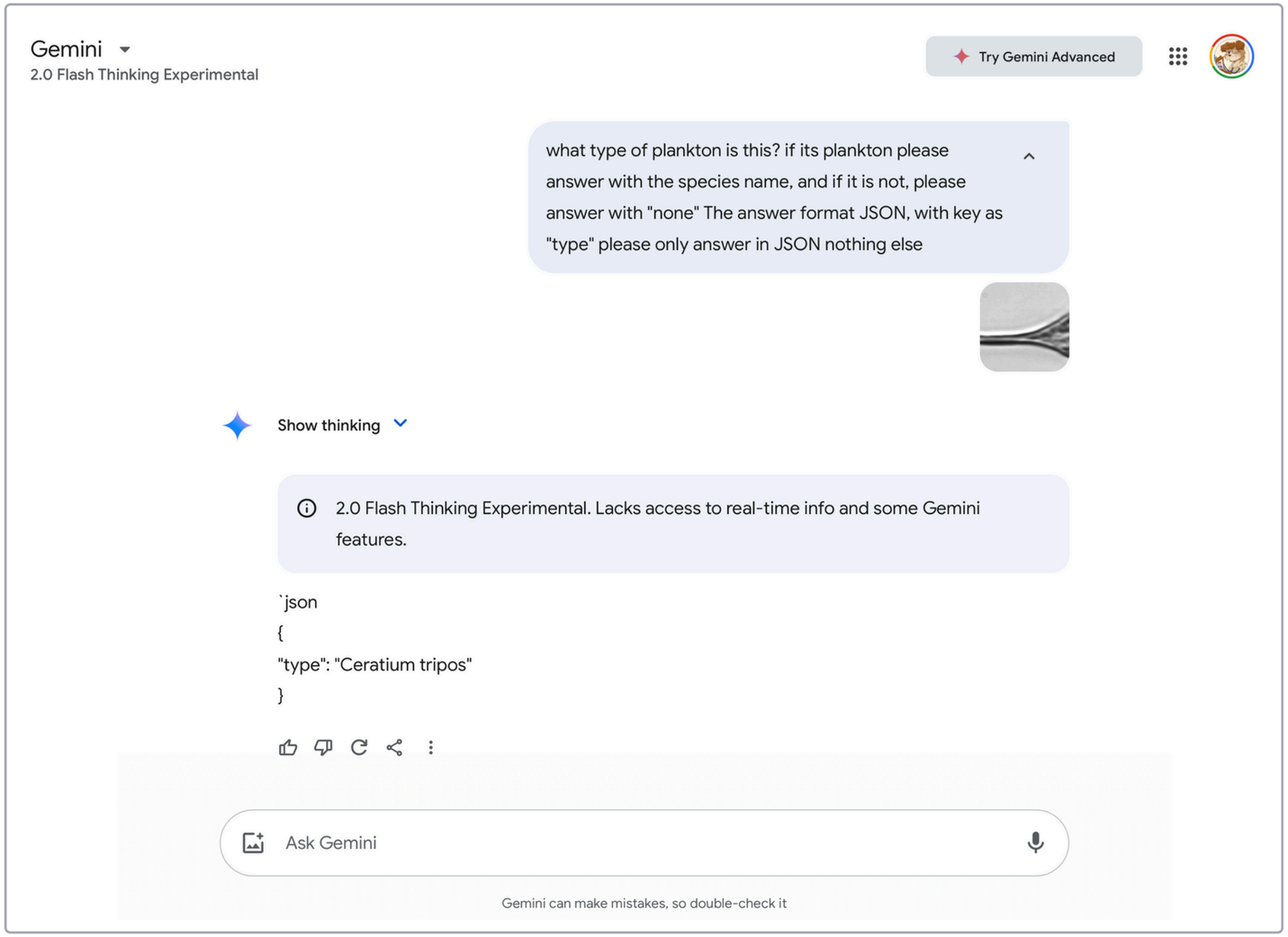
Allowed creativity in the response

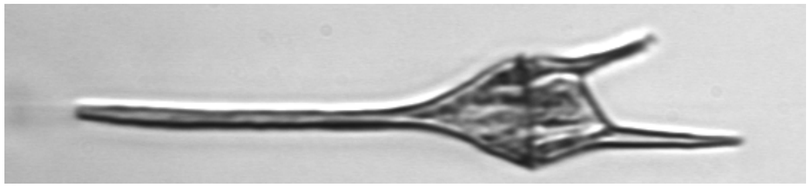
Top P

0.95

Probability threshold for top-p sampling

MULTIMODAL LLM IMPLEMENTATION VIA GUI





ceratium.png

Prompt Drafting - v 0.1

What type of plankton is this? if its plankton please answer with the species name, and if it is not, please answer with "none" The answer format JSON, with key as "type" please only answer in JSON nothing else size of this sample is {size}

Remark: Outputting JSON from GenAi for it to be able to integrate with other parts of the system

MULTIMODAL LLM IMPLEMENTATION VIA API

```python
image = PIL.Image.open('ceratium.png')

response = client.models.generate_content(
    model="gemini-2.0-flash",
    contents=["what type of plankton is this? if its plankton please answer with the species name, and if it is not, please answer with 'none' The answer format JSON, with key as 'type' please only answer in JSON nothing else", image])

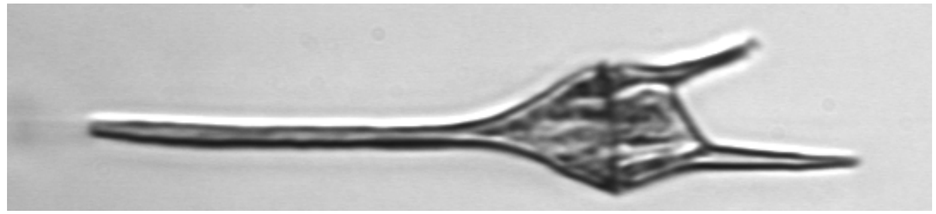
print(response.text)
```

[5] ✓ 2.2s Python

Image Declaration

Prompt

Image Attachement

```
```json
{
  "type": "Ceratium"
}
```
```

Input



ceratium.png

Output

```
```json
{
  "type": "Ceratium"
}
```
```

FINE TUNING OPTIONS FOR THIS USE CASE

Prompt Engineering

Structures LLM queries to utilize retrieved metadata and ensures responses are formatted correctly

See this slide >

Retrieval Augmented Generation

Retrieves similar plankton images and metadata from a vector database using image embeddings

See next slide ...

PERSONA	You are an expert marine biologist with extensive knowledge of plankton taxonomy. You specialize in identifying and classifying various plankton species based on provided image data or textual descriptions.
CONTEXT	You are classifying an organism to determine whether it is plankton. If it is, provide its taxonomic details (family, genus, species). If it is not plankton, return `"none"`. The output must strictly adhere to JSON format for seamless integration into a structured database.
STEPS	**Steps:** 1. Analyze the given image to determine if it belongs to the plankton category. 2. If it is plankton, classify it as accurately as possible and provide the taxonomic details. 3. If it is not plankton, return `"none"` in the response. 4. Ensure the response strictly follows the JSON format.

OUTPUT	**Output Format:**
	{
	"family": "",
	"genus": "",
	"species": ""
	}
	If it is **not plankton**, return:
	{
	"species": "none"
	}

CUE	Classify the given input according to the structured taxonomy above. If the organism is not plankton, return `"none"`. Do not include any additional text or explanation—only the JSON output.

FINE TUNE BY PROMPT ENGINEERING

You are an expert marine biologist with extensive knowledge of plankton taxonomy. You specialize in identifying and classifying various plankton species...

Show thinking

2.0 Flash Thinking Experimental. Lacks access to real-time info and some Gemini features.

JSON

```
{
  "family": "Ceratiaceae",
  "genus": "Ceratium",
  "species": "Ceratium furca"
}
```

```
image = PIL.Image.open('ceratium.png')

response = client.models.generate_content(
    model="gemini-2.0-flash-thinking-exp-01-21",
    contents=["""You are an expert marine biologist with extensive knowledge of plankton taxonomy. You specialize in identifying and classifying

        You are classifying an organism to determine whether it is plankton. If it is, provide its taxonomic details (family, genus, spec

        **Steps:**
        1. Analyze the given data (image, description, or other input) to determine if it belongs to the plankton category.
        2. If it is plankton, classify it as accurately as possible and provide the taxonomic details.
        3. If it is not plankton, return `"none"` in the response.
        4. Ensure the response strictly follows the JSON format.

        **Output Format:**
        ```json
        {
          "family": "Calanidae",
          "genus": "Calanus",
          "species": "Calanus finmarchicus"
        }
        ```
        If it is **not plankton**, return:
        ```json
        {
          "classification": "none"
        }
        ```

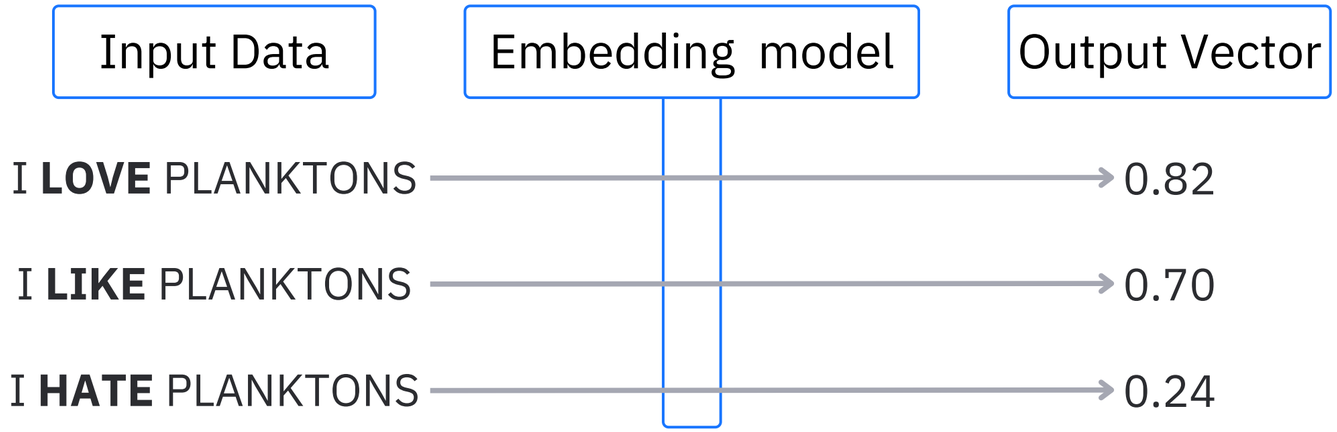
        Classify the given input according to the structured taxonomy above. If the organism is not plankton, return `"none"`. Do not inc
        """,
      image])

print(response.text)
```

[21] ✓ 3.9s

```
```json
{
  "family": "Ceratiaceae",
  "genus": "Ceratium",
  "species": "Ceratium furca"
}
```
```

PRIMER FOR VECTOR EMBEDDING IN RAG

Transforming Data (text) into Number

Output Vector represents semantic embedding

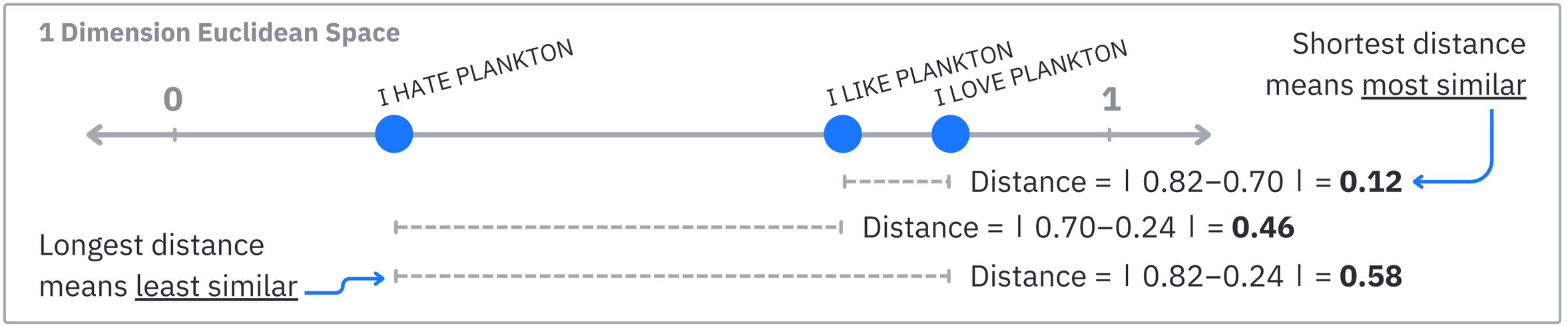


Euclidean Distance

$$d(A,B) = \sqrt{(v_1 - v_2)^2}$$

$$d(A,B) = |v_1 - v_2|$$

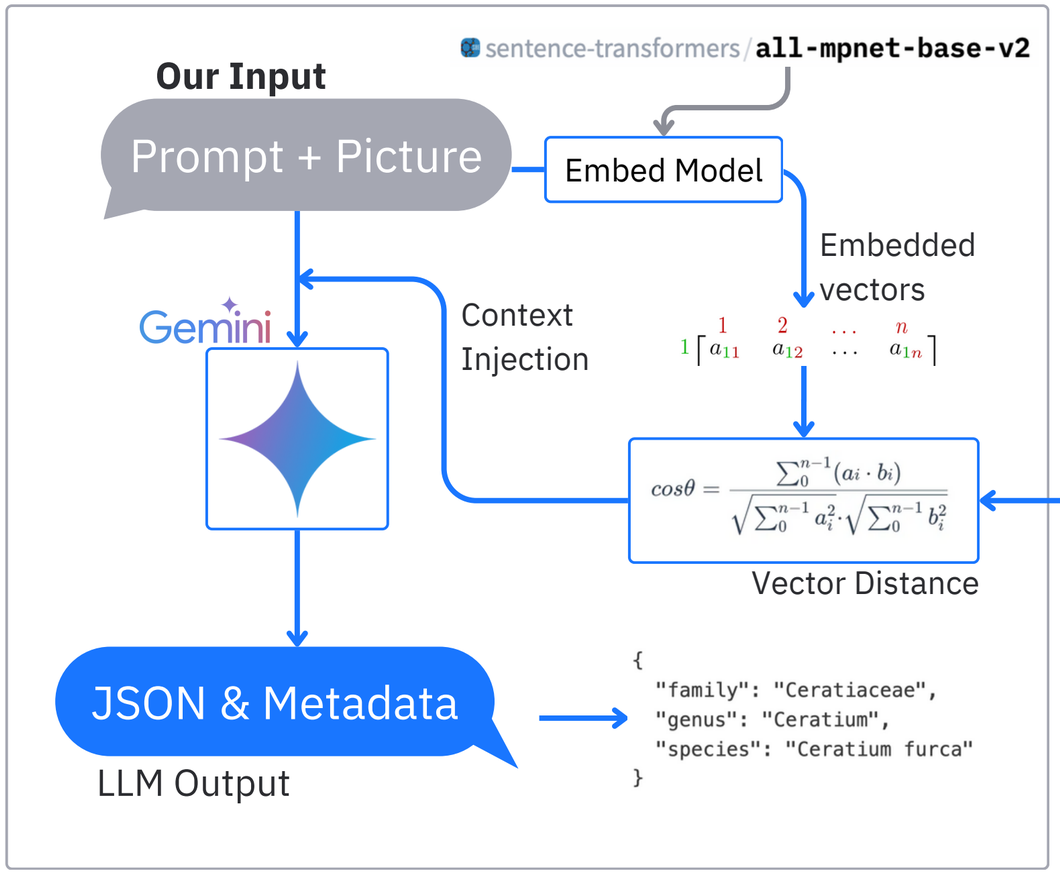
Distance represents semantic similarity



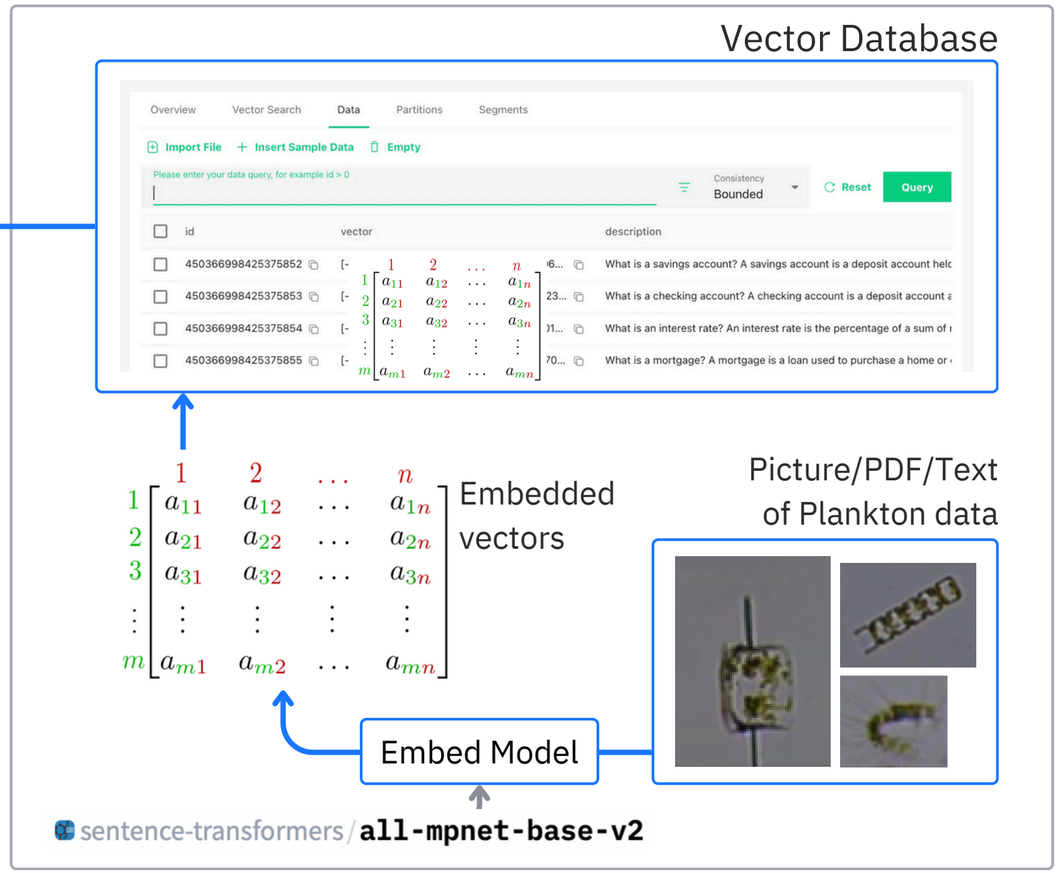
Output vectors in this slide are for example only

FINE TUNE BY RAG IMPLEMENTATION

Classification Inquiry Node



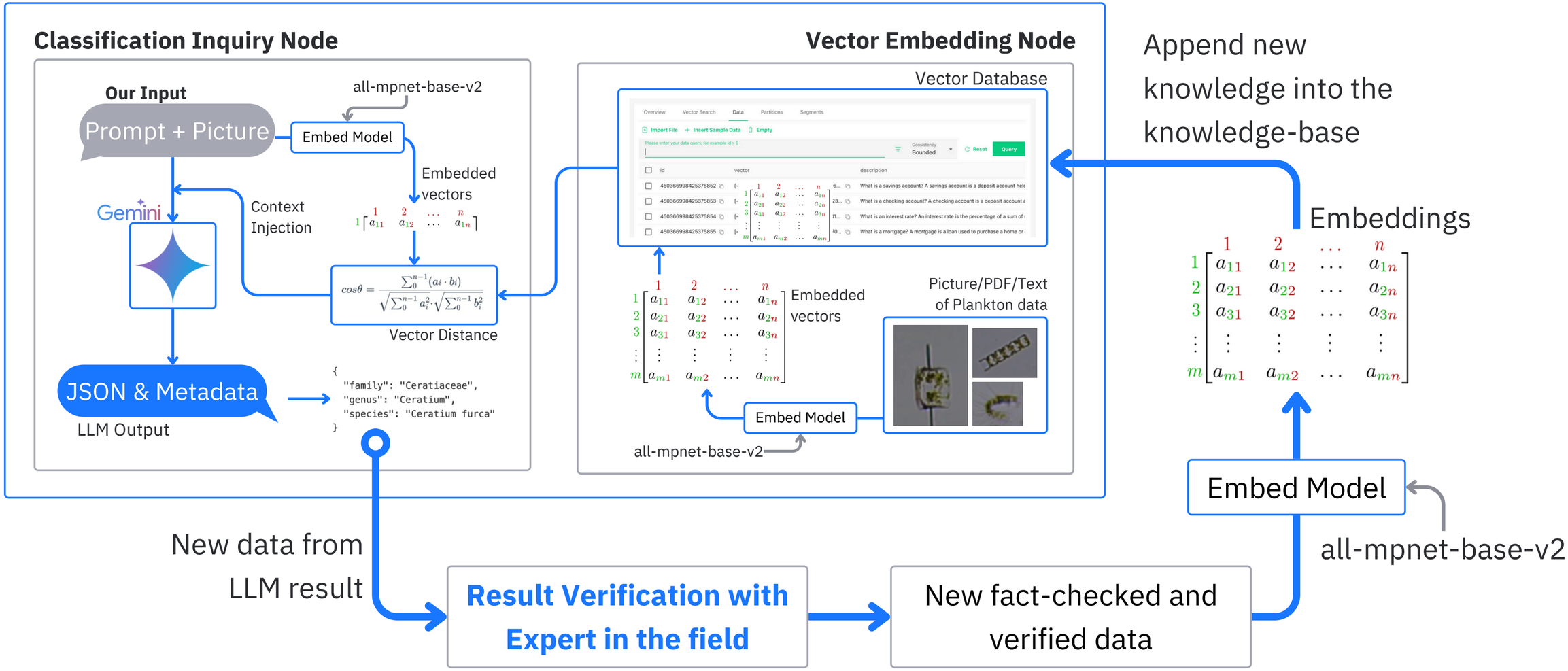
Vector Embedding Node



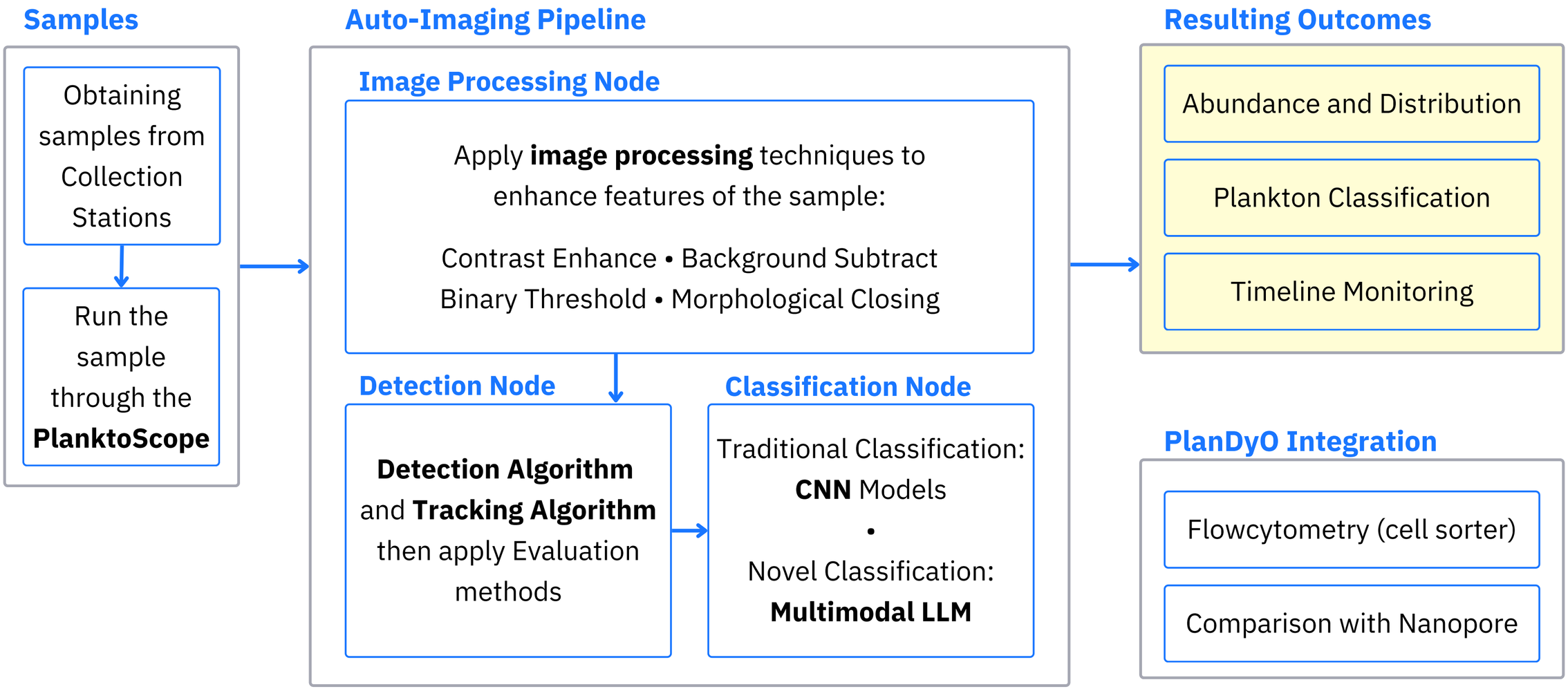
Remark: There are 2 ways of going with this - Local or Vertex AI's Feature Store

RAG FEEDBACK LOOP IMPLEMENTATION

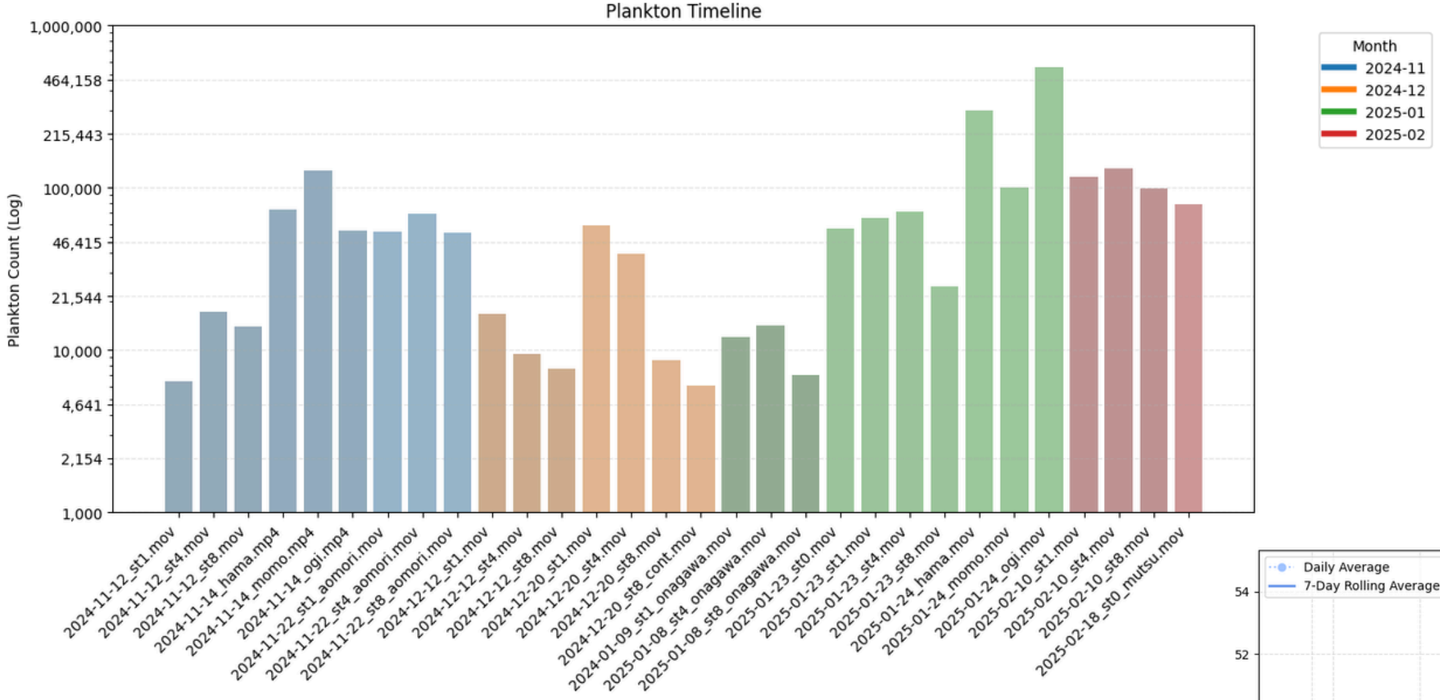
Existing RAG implemented LLM

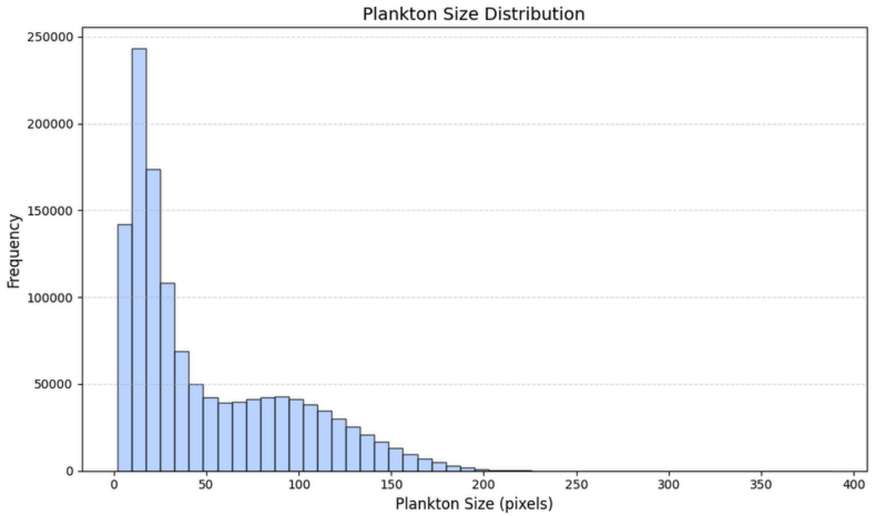


RESULTS AND NEXT STEPS

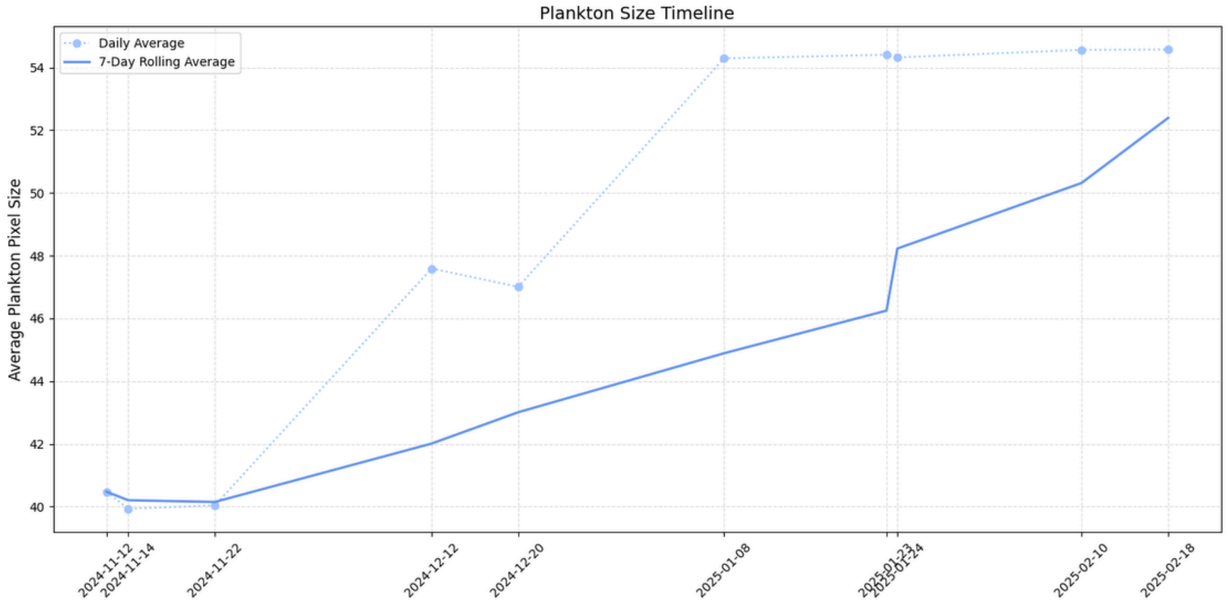


ABUNDANCE AND SIZE DISTRIBUTION





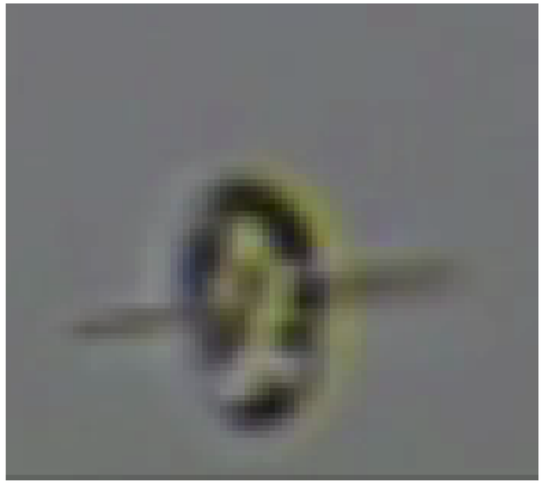
We can observe that there is a higher abundance of planktons after the colder part of winter



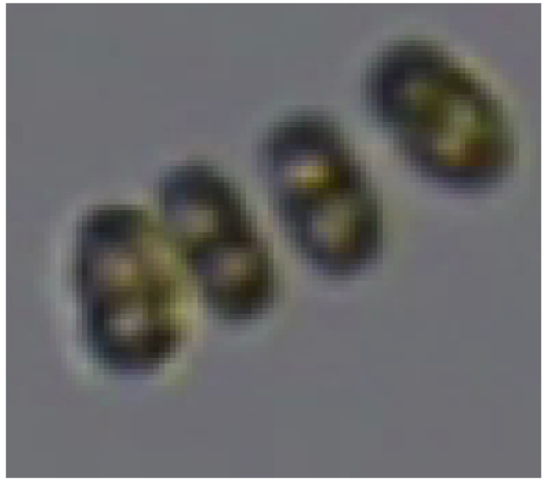
PLANKTON CLASSIFICATION SNIPPETS

{
"family": "Coscinodiscaceae",
"genus": "Coscinodiscus",
"species": "Coscinodiscus sp."
}

{
"family": "Diatomaceae",
"genus": "Thalassiosira",
"species": "Thalassiosira weissflogii"
}



{
"family": "Melosiraceae",
"genus": "Melosira",
"species": "Melosira moniliformis"
}



{
"family": "Melosiraceae",
"genus": "Melosira",
"species": "Melosira varians"
}

{
"family": "Phormidiaceae",
"genus": "Spirulina",
"species": "Spirulina platensis"
}

{
"classification": "none"
}

{
"family": "Chaetocerotaceae",
"genus": "Chaetoceros",
"species": "Chaetoceros debilis"
}

The above samples are correctly predicted

There are some that are still incorrect

Have an awesome day!

Jaronchai Dilokkalayakul
Information Biology Laboratory,
Tohoku University **東北大学**

Next step for this project

- **New knowledge without previous data + Feedback loop**
- **“Agentic AI” -** streamline manual human processes
- Locally hosted LLM and embedding models

Potential Applications in Related Fields

- Marine Biodiversity Monitoring, use LLM + RAG to automate the microorganisms classification process in ocean samples
- Automate functional annotation of in marine microbiomes.
- Identify plankton species impacting fish farms, optimizing feeding and disease prevention.
- **Let’s discuss!**

APPENDIX

Appendix

INTRODUCTION TO TERMINOLOGIES

Planktoscope:
Hardware and software for quantitative imaging of plankton samples

Image Processing:
Analyzing, transforming, and optimizing images by modifying pixel values, patterns, and structures to extract meaningful information.

Multimodal LLM (Large Language Model):
A Multimodal LLM processes and **understands multiple data types** (text, images, audio, etc.) to generate context-aware responses of desired format.

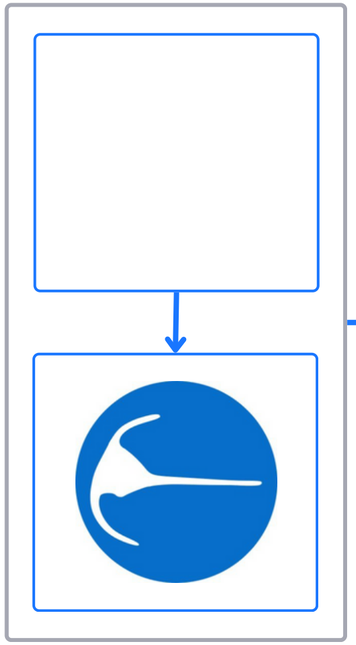
Vector Embedding:
Vector embedding **converts data (text, images) into numerical vectors**, allow similarity searches in high-dimensional space for data retrieval

Semantic Similarity
Measures how similar meanings of words, phrases, or texts are based on context, often using embeddings or language models to quantify closeness.

Appendix

TECH STACK

Samples



Auto-Imaging Pipeline

Image Processing Node

OpenCV

matplotlib

NumPy

Detection Node

OpenCV

Classification Node

Gemini

Tools & Platform

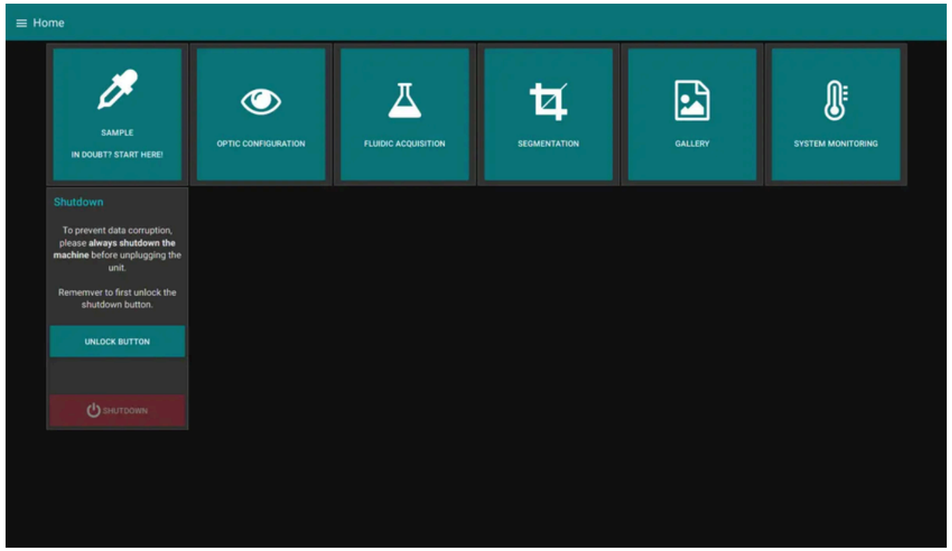
Milvus

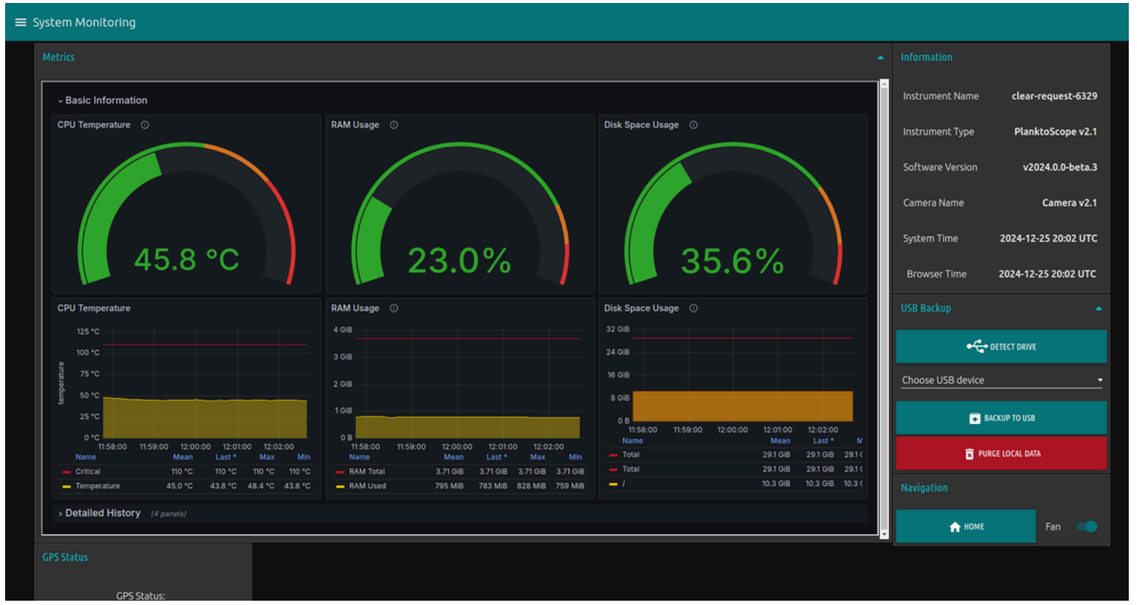
Understanding Google Cloud Stack

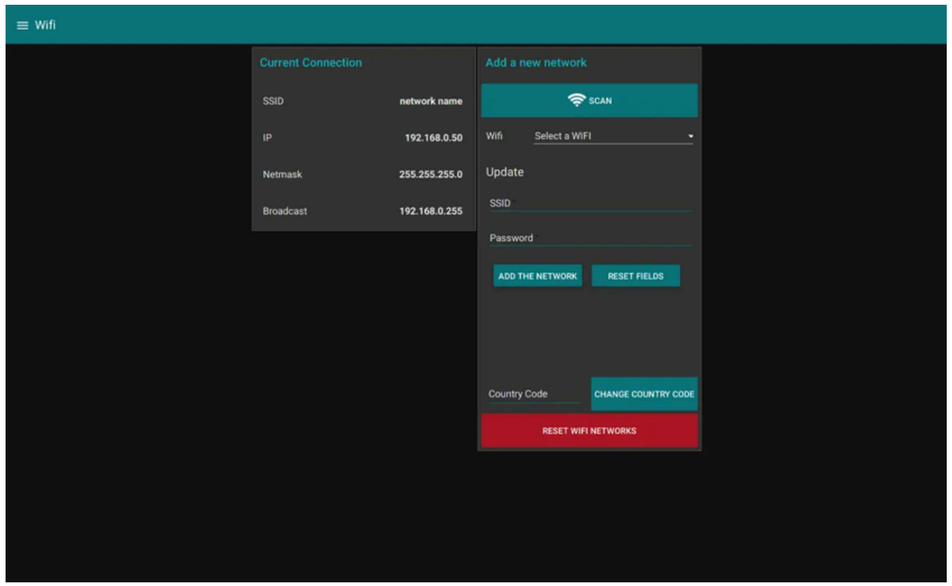
Gemini

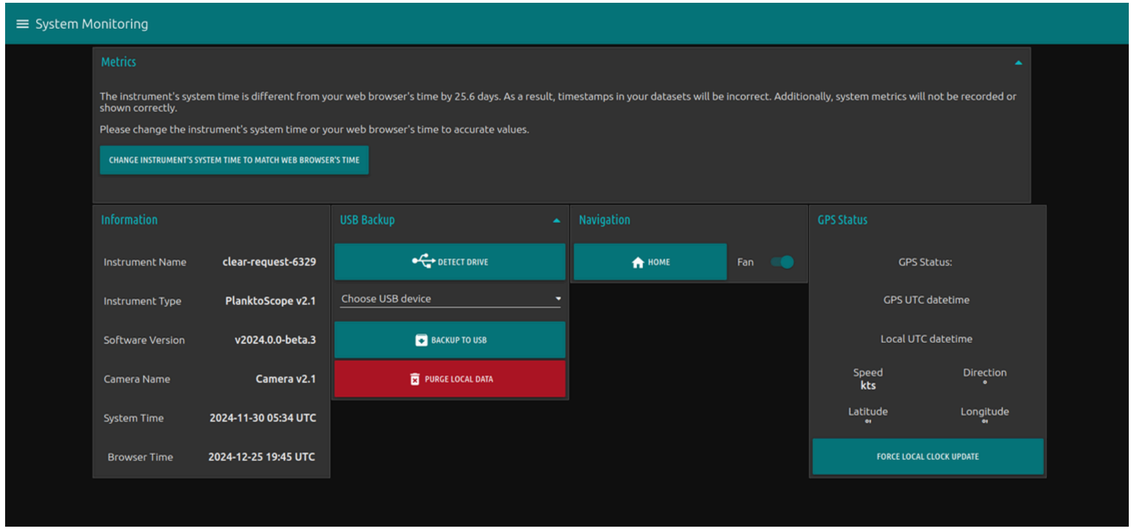
Appendix

PLANKTOSCOPE GRAPHICAL USER INTERFACE







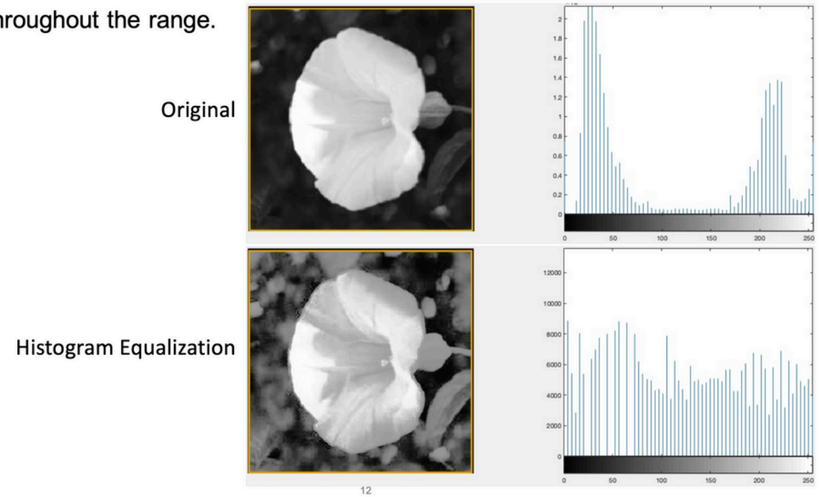


Appendix

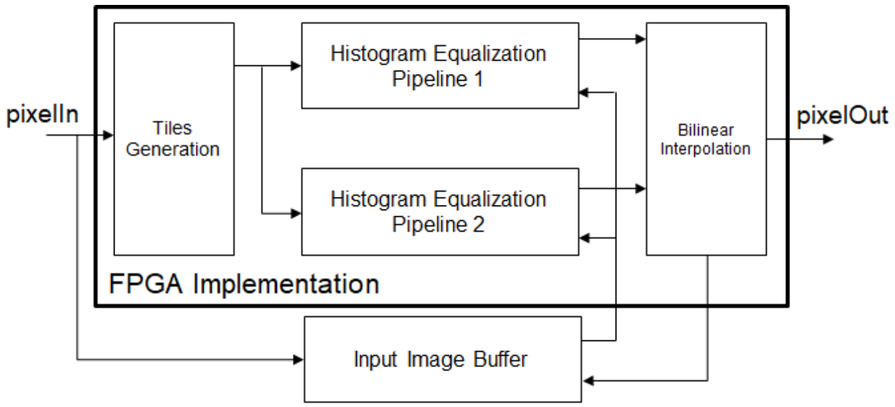
HISTOGRAM EQUALIZATION AND CLAHE

Histogram Equalization

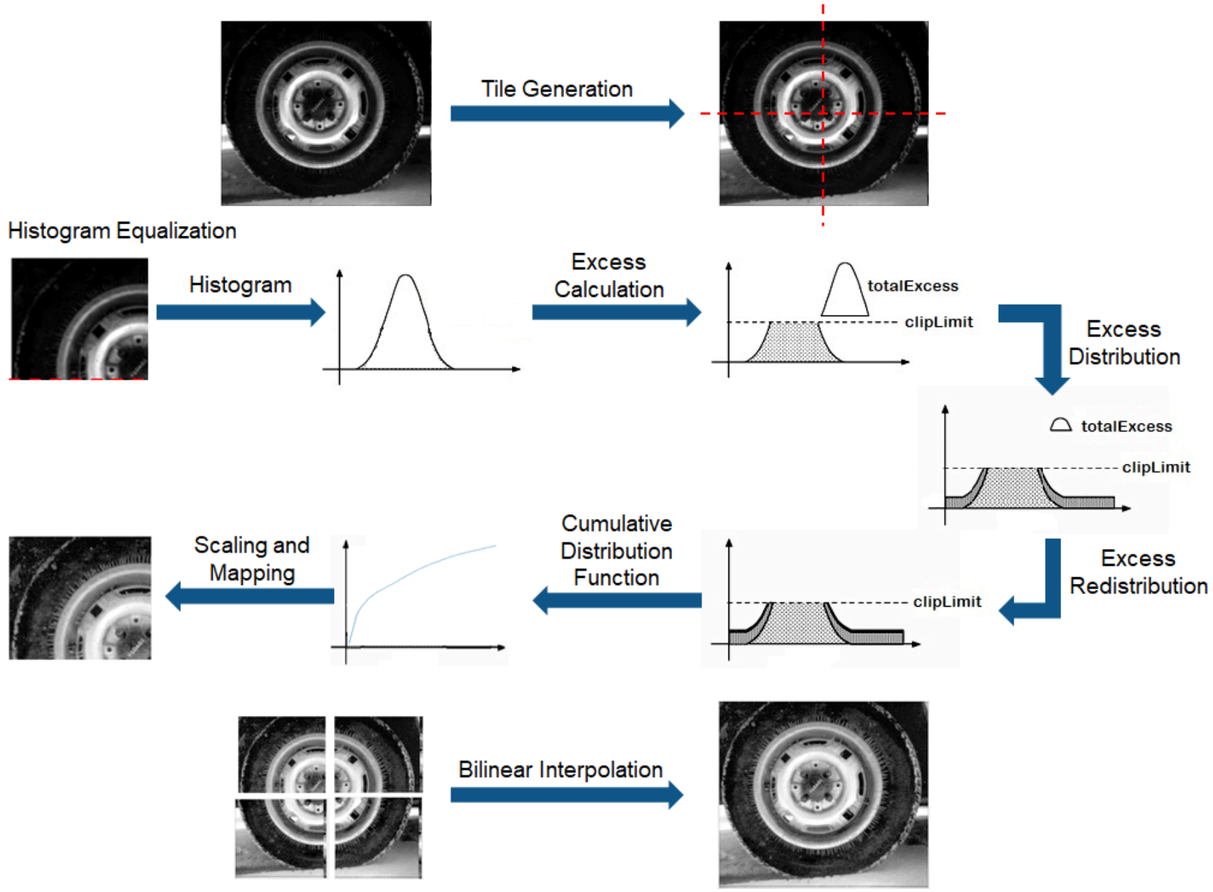
- A histogram processing method to adjust the contrast of the image to have an evenly distributed intensity throughout the range.



Gonzalez, R. C. (2018). Digital image processing (4th ed.). Pearson.



Contrast Limited Adaptive Histogram Equalization



MathWorks. (n.d.). Contrast Limited Adaptive Histogram Equalization. Retrieved February 18, 2025, from https://www.mathworks.com/help/visionhdl/ug/contrast-adaptive-histogram-equalization.html

Appendix

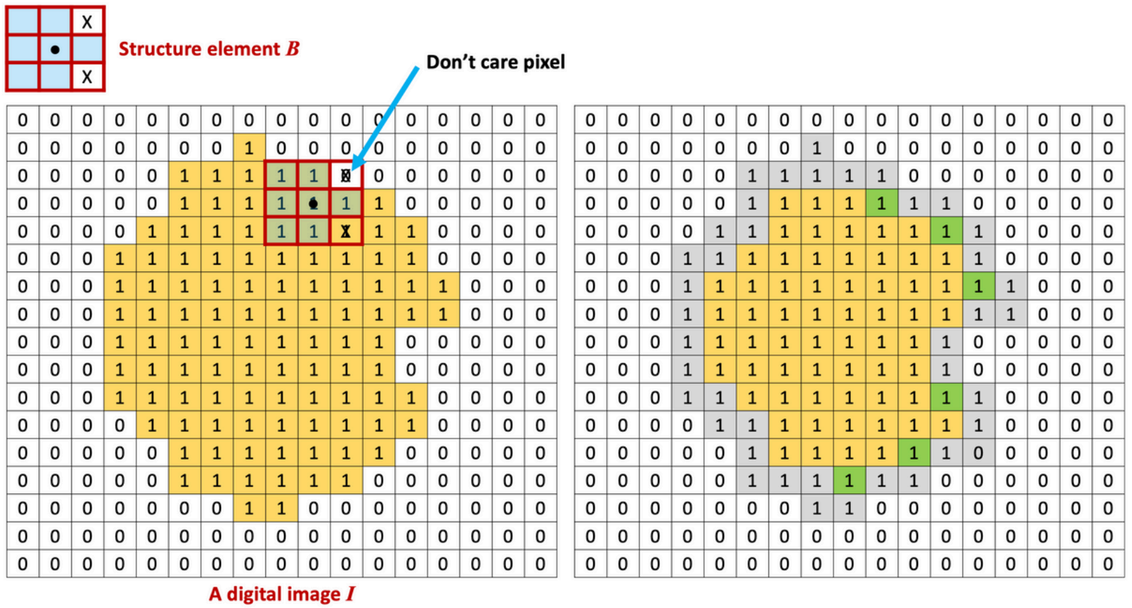
MORPHOLOGICAL OPERATORS ON BINARY IMAGE

Erosion

- An **operator** in the area of mathematical morphology.

$$A \ominus B = \{z|(B)_z \subseteq A\}$$

- The set of all points z such that B is contained in A



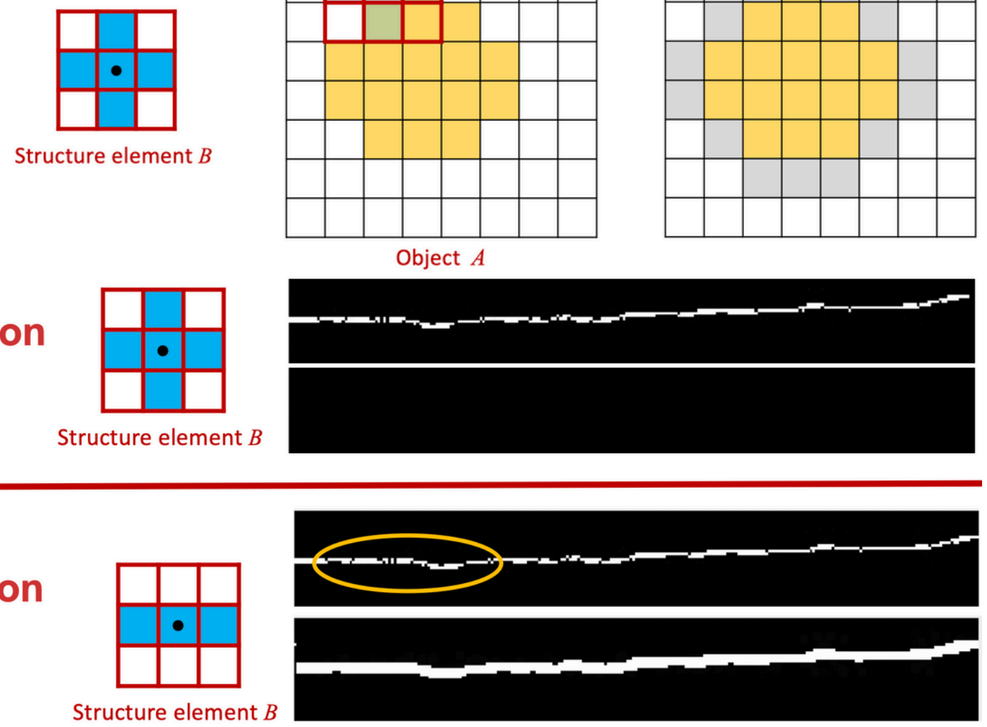
Gonzalez, R. C. (2018). Digital image processing (4th ed.). Pearson.

Dilation

- An **operator** in the area of mathematical morphology.

$$A \oplus B = \{z|(B)_z \cap A \neq \emptyset\}$$

- The set of all points z such that B is overlapped at least one element of A

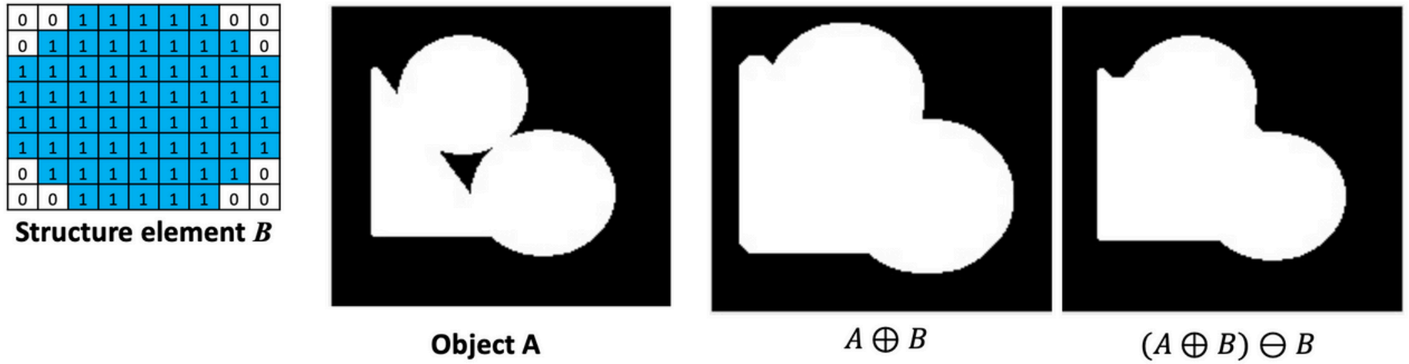


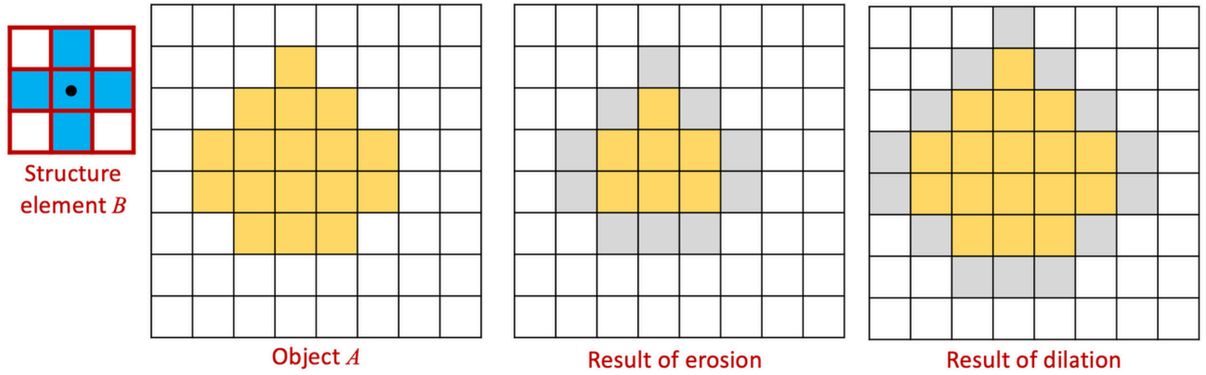
Appendix

ABOUT MORPHOLOGICAL CLOSING

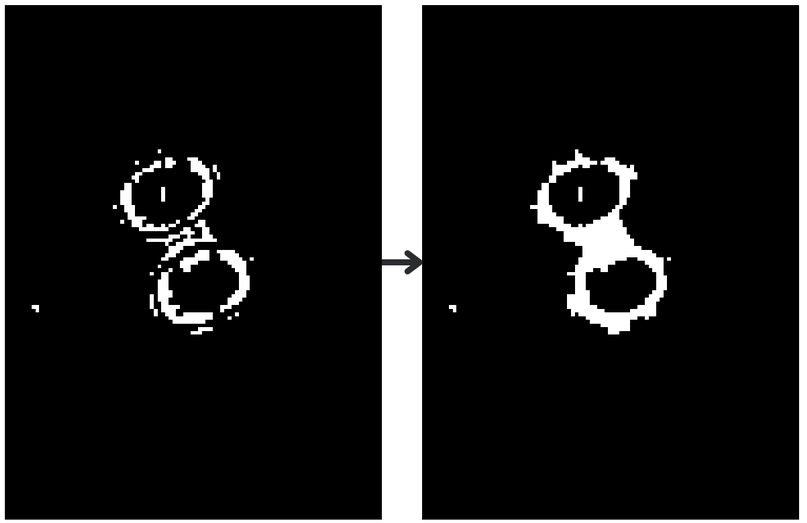
$$A \bullet B = (A \oplus B) \ominus B$$

Dilation — Erosion





Actual Image from the Pipeline



Result of Binary Threshold → Apply Closing Method

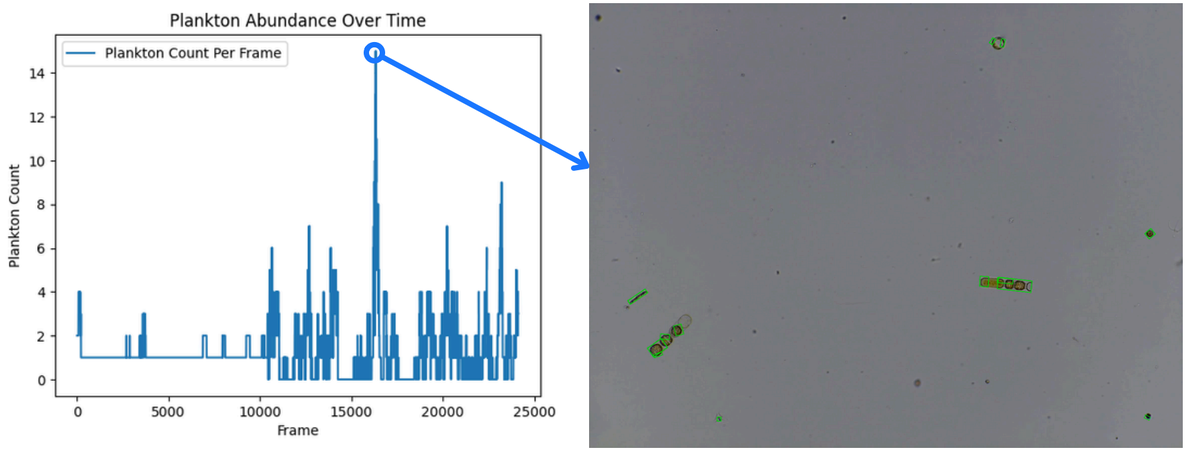
Figures: *Gonzalez, R. C. (2018). Digital image processing (4th ed.). Pearson.*

Appendix

DETECTION & TRACKING ALGORITHM EVALUATION

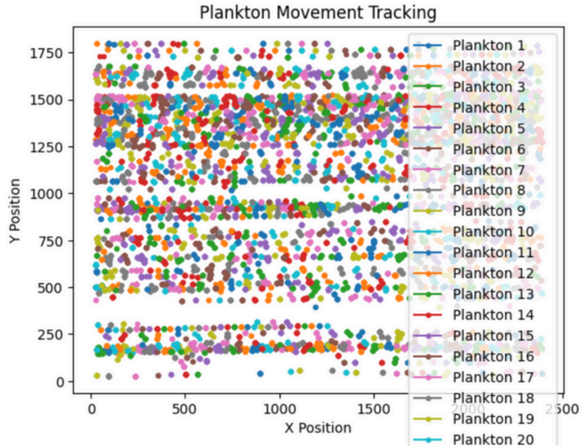
Detection Evaluation

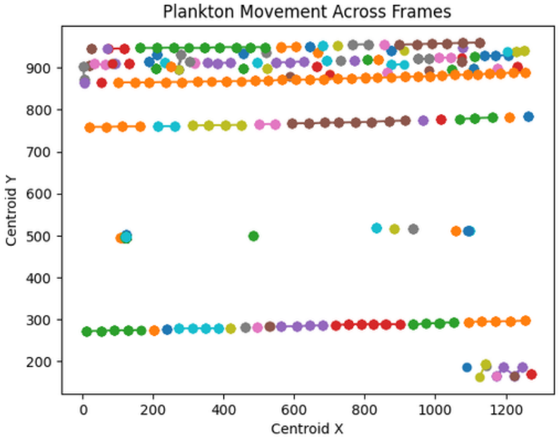
Detection Goal: Must be Closest to real value as possible



Tracking Evaluation

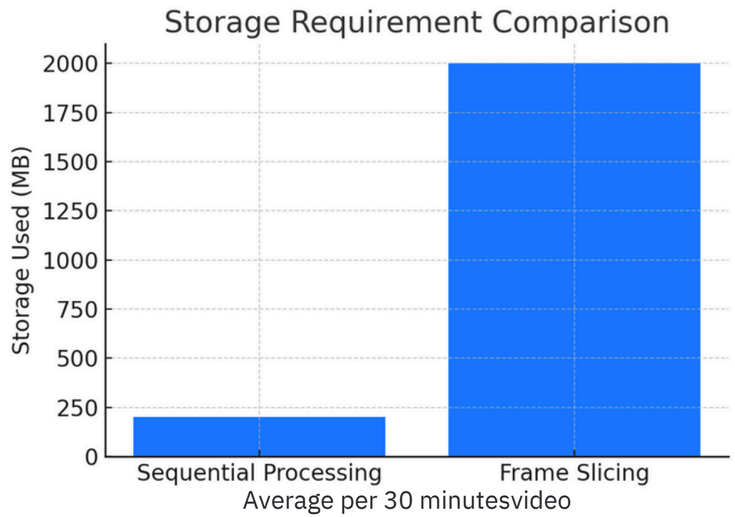
Tracking Goal: Same plankton must have same ID throughout





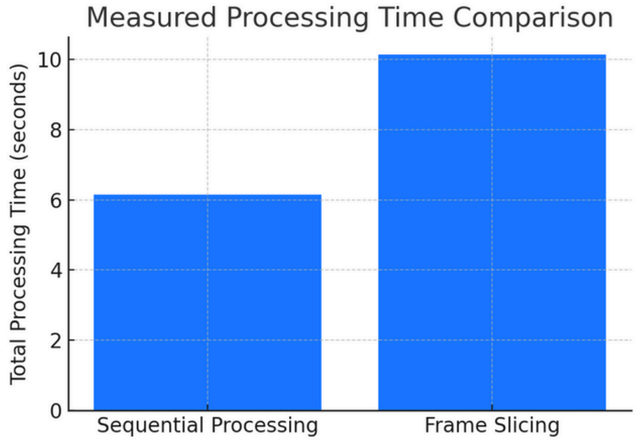
Appendix

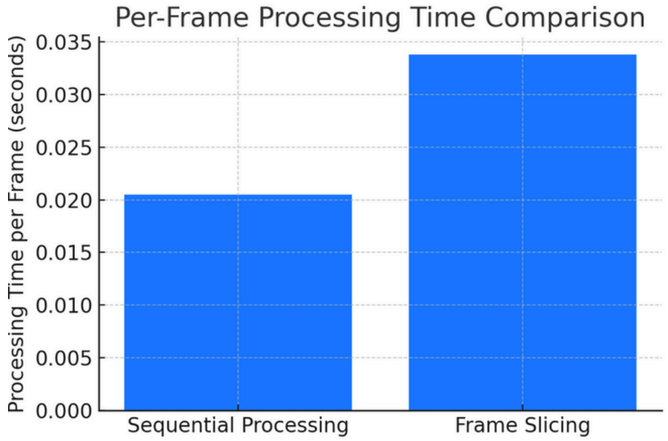
OPTIMIZATION FOR VIDEO PROCESSING



Uses a ***deque*** (trackers = deque(maxlen=150)) to store recent detections

double-ended queue is an abstract data type that generalizes a queue, for which elements can be added to or removed from either the front (head) or back (tail)





Sequential Processing:

- **Minimizing memory overhead**
- Tracks across frames, preventing redundancy
- Real-time processing and large dataset OK!

Frame Slicing Processing:

- Extracts all frames first, storage consuming
- Independent processing
- Increases memory and I/O usage

Appendix

CLASSIFICATION STRATEGIES COMPARISON

Convolutional Neural Network

Great for Image Recognition – Learns patterns from images to classify plankton.

Fast & Efficient – Works well once trained, making quick predictions.

Needs Lots of Labeled Images – Requires a big dataset to learn accurately.

Struggles with Context – Only focuses on visual patterns, not scientific descriptions.

Large Language Model

Understands Context – Can classify plankton using descriptions and external knowledge.

Flexible – Can work with both images and text for classification.

Learn from Descriptions – Can classify even with limited images using existing knowledge.

Computationally Heavy – Requires more power and can be slower than CNNs.

Appendix

2006 IFCB IMAGES DATASET AND PREPROCESSING

2006 labeled IFCB images

No Thumbnail Available

Date

2006

Authors

Sosik, Heidi M.
Peacock, Emily E.
Brownlee, Emily F.

Linked Authors

Sosik, Heidi M.
Peacock, Emily E.
Brownlee, Emily F.

Files

2006.zip (700.08 MB)

Citable URI

https://hdl.handle.net/1912/7342

Description

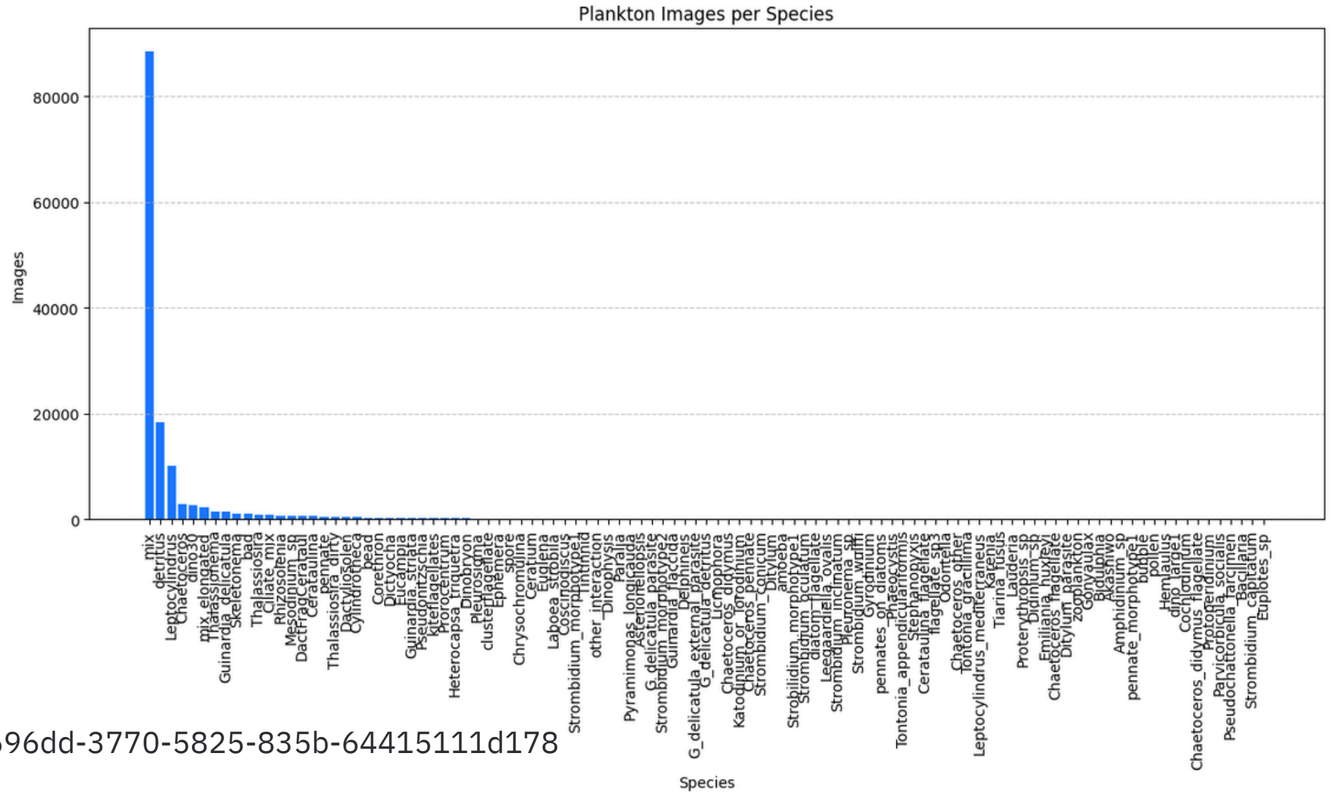
This zipped content contains Annotated Plankton Images from one year and is part of the WHOI-Plankton Collection that spans multiple years. Click on the WHOI-Plankton link below to view all items (other years) in this collection.

Collections

WHOI-Plankton

Full item page

```
Total species: 103
Total images: 140342
Most common species: mix (88377 images)
Least common species: pennate_morphotype1 (0 images)
Species with zero images: 15
```



https://darchive.mblwhoilibrary.org/entities/publication/e01696dd-3770-5825-835b-64415111d178

Appendix

MODEL PROVIDER SELECTION

Vertex AI's Gemini

Token-based pricing | Modality-based pricing

Model	Type	Price	Price with Batch API
Gemini 2.0 Flash	1M Input tokens	$0.15	$0.075
	1M Input audio tokens	$1.00	$0.50
	1M Output text tokens	$0.60	$0.30
Gemini 2.0 Flash Lite	1M Input tokens	$0.075	$0.0375
	1M Input audio tokens	$0.075	$0.0375
	1M Output text tokens	$0.30	$0.15

	Free Tier	Paid Tier, per 1M tokens in USD
Input price	Free of charge	$0.10 (text / image / video) $0.70 (audio)
Output price	Free of charge	$0.40
Context caching price	Free of charge	$0.025 / 1,000,000 tokens (text/image/video) $0.175 / 1,000,000 tokens (audio) Available February 24, 2025
Context caching (storage)	Free of charge, up to 1,000,000 tokens of storage per hour Available February 24, 2025	$1.00 / 1,000,000 tokens per hour Available February 24, 2025
Tuning price	Not available	Not available
Grounding with Google Search	Free of charge, up to 500 RPD	1,500 RPD (free), then $35 / 1,000 requests
Used to improve our products	Yes	No

OpenAI's GPT

Text tokens

Price per 1M tokens · Batch API price

Model	Input	Cached input	Output
gpt-4o ↳ gpt-4o-2024-08-06	$2.50	$1.25	$10.00
gpt-4o-audio-preview ↳ gpt-4o-audio-preview-2024-12-17	$2.50	-	$10.00
gpt-4o-realtime-preview ↳ gpt-4o-realtime-preview-2024-12-17	$5.00	$2.50	$20.00
gpt-4o-mini ↳ gpt-4o-mini-2024-07-18	$0.15	$0.075	$0.60
gpt-4o-mini-audio-preview ↳ gpt-4o-mini-audio-preview-2024-12-17	$0.15	-	$0.60
gpt-4o-mini-realtime-preview ↳ gpt-4o-mini-realtime-preview-2024-12-17	$0.60	$0.30	$2.40
o1 ↳ o1-2024-12-17	$15.00	$7.50	$60.00
o3-mini ↳ o3-mini-2025-01-31	$1.10	$0.55	$4.40
o1-mini ↳ o1-mini-2024-09-12	$1.10	$0.55	$4.40

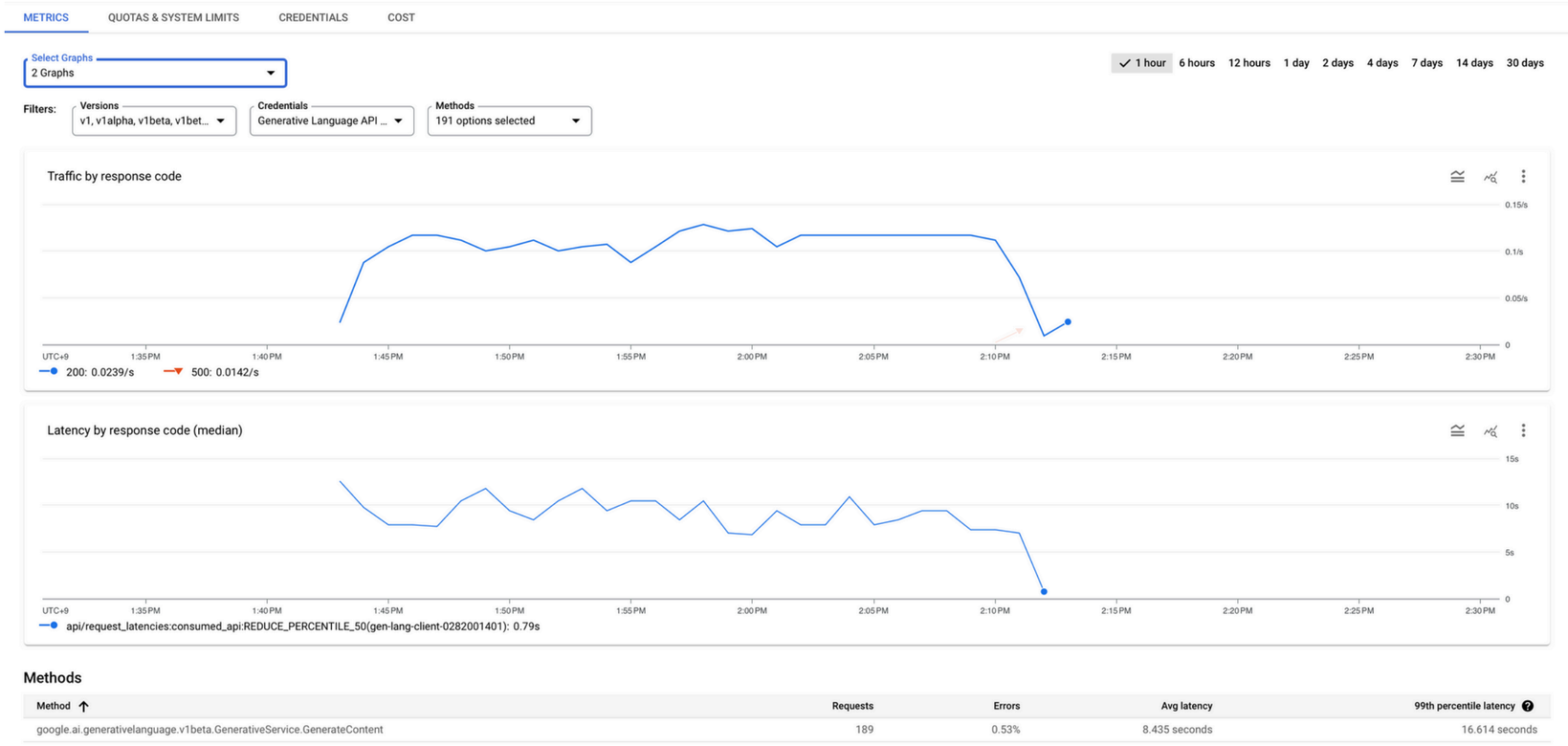
https://platform.openai.com/docs/pricing
https://ai.google.dev/gemini-api/docs/pricing

Appendix

MODEL SELECTION AND USAGE QUOTA LIMIT

Model Selection

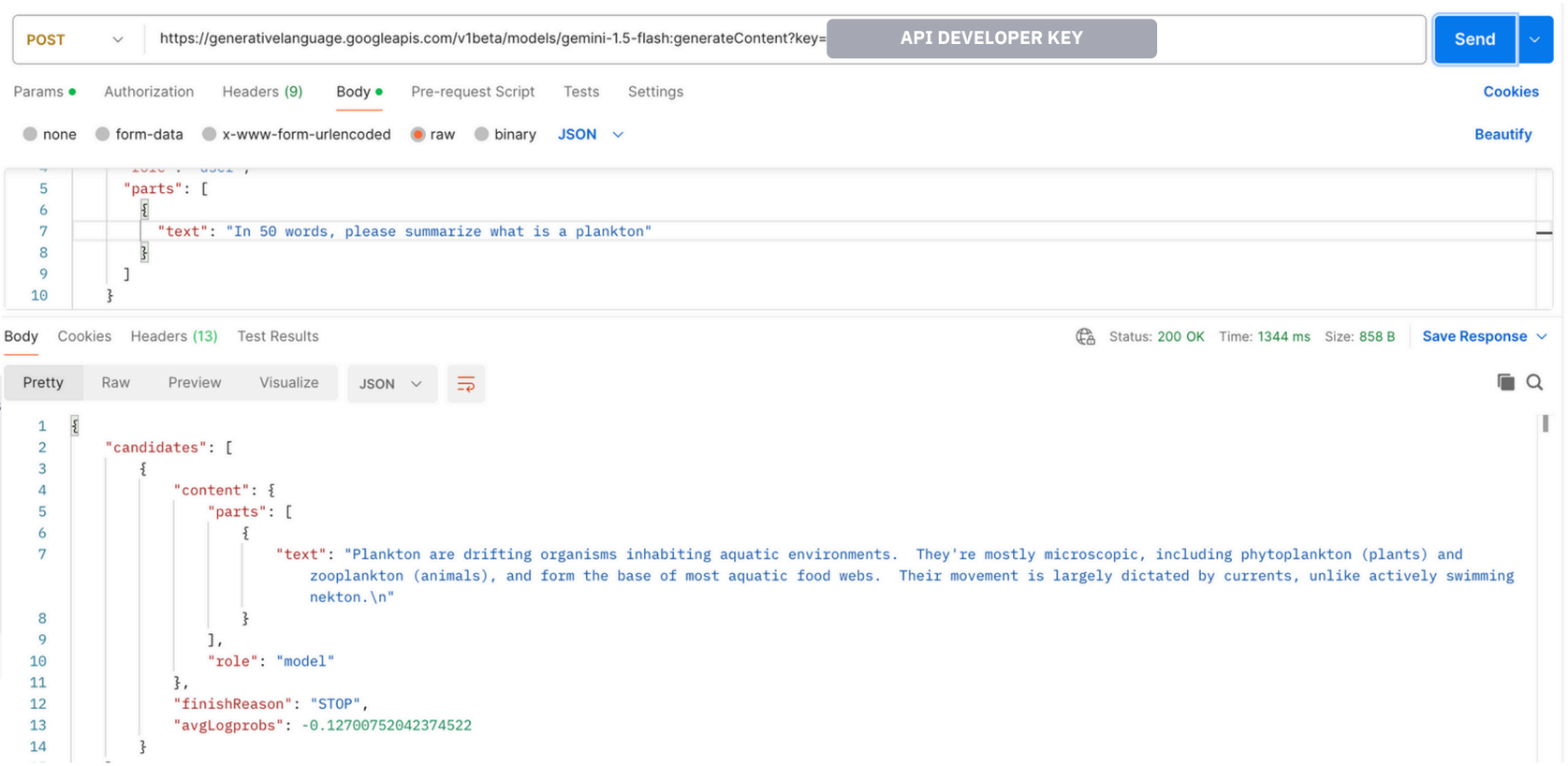
Model	Inputs	Outputs	Use case
Gemini 2.0 Flash **gemini-2.0-flash-001**	Text, Code, Images, Audio, Video, Video with Audio, PDF	Text, Audio (private preview), Images (private preview)	Workhorse model for all daily tasks. Strong overall performance and supports real-time streaming Live API.
Gemini 2.0 Pro **gemini-2.0-pro-exp-02-05**	Text, Images, Video, Audio, PDF	Text	Strongest model quality, especially for code & world knowledge; 2M long context.
Gemini 2.0 Flash-Lite **gemini-2.0-flash-lite-preview-02-05**	Text, Images, Video, Audio, PDF	Text	Our cost effective offering to support high throughput.
Gemini 2.0 Flash Thinking **gemini-2.0-flash-thinking-exp-01-21**	Text, Images	Text	Provides stronger reasoning capabilities and includes the thinking process in responses.



Name	Type	Dimensions (e.g. location)	Value	Current usage percentage	Current usage	Adjustable
Request limit per model per day for a project in the free tier	Quota	model : gemini-2.0-flash-exp	1,500	27.07%	406	Yes
Request limit per model per day for a project in the free tier	Quota	model : gemini-1.5-flash	1,500	0.13%	2	Yes

Appendix

VERTEX AI API AND TESTING VIA POSTMAN



API keys

Quickly test the Gemini API

API quickstart guide

```
curl "https://generativelanguage.googleapis.com/v1beta/models/gemini-1.5-flas
key=GEMINI_API_KEY" \
-H 'Content-Type: application/json' \
-X POST \
-d '{
  "contents": [{
    "parts":[{"text": "Explain how AI works"}]
   }]
  }'
```

Use code with caution.

Create API key

Your API keys are listed below. You can also view and manage your project and API keys in Google Cloud.

Project number	Project name	API key	Created	Plan
...3450	Gemini API	...7uxI	Feb 7, 2025	Free of charge Set up Billing View usage data

Remember to use API keys securely. Don't share or embed them in public code. Use of Gemini API from a billing-enabled project is subject to pay-as-you-go pricing.

Appendix

VERTEX AI FEATURE STORE AS RAG ENGINE

Using BigQuery Table as a mapping unit

Generate content using Vertex AI Gemini API

Call the Vertex AI `GenerateContent` API to use Gemini models to generate content, and specify `RAG_CORPUS_RESOURCE` in the request to retrieve data from the `FeatureOnlineStore` index.

REST | Vertex AI SDK for Python

To learn how to install or update the Vertex AI SDK for Python, see Install the Vertex AI SDK for Python. For more information, see the Vertex AI SDK for Python API reference documentation.

```python
from vertexai.preview import rag
from vertexai.preview.generative_models import GenerativeModel, Tool
import vertexai

# TODO(developer): Update and un-comment below lines
# PROJECT_ID = "your-project-id"
# corpus_name = "projects/{PROJECT_ID}/locations/us-central1/ragCorpora/{rag_corpus_id}"

# Initialize Vertex AI API once per session
vertexai.init(project=PROJECT_ID, location="us-central1")

rag_retrieval_tool = Tool.from_retrieval(
    retrieval=rag.Retrieval(
        source=rag.VertexRagStore(
            rag_resources=[
                rag.RagResource(
                    rag_corpus=corpus_name,
                    # Optional: supply IDs from `rag.list_files()`.
                    # rag_file_ids=["rag-file-1", "rag-file-2", ...],
                )
            ],
            similarity_top_k=3,  # Optional
            vector_distance_threshold=0.5,  # Optional
        ),
    )
)

rag_model = GenerativeModel(
    model_name="gemini-1.5-flash-001", tools=[rag_retrieval_tool]
)
response = rag_model.generate_content("Why is the sky blue?")
print(response.text)
# Example response:
#   The sky appears blue due to a phenomenon called Rayleigh scattering.
#   Sunlight, which contains all colors of the rainbow, is scattered
#   by the tiny particles in the Earth's atmosphere....
#   ...
```

REST | Vertex AI SDK for Python

To learn how to install or update the Vertex AI SDK for Python, see Install the Vertex AI SDK for Python. For more information, see the Vertex AI SDK for Python API reference documentation.

```python
from vertexai.preview import rag
import vertexai

# TODO(developer): Update and un-comment below lines
# PROJECT_ID = "your-project-id"
# feature_view_name = "projects/{PROJECT_ID}/locations/{LOCATION}/featureOnlineStores/{FEATURE_ONLI
# display_name = "test_corpus"
# description = "Corpus Description"

# Initialize Vertex AI API once per session
vertexai.init(project=PROJECT_ID, location="us-central1")

# Configure embedding model (Optional)
embedding_model_config = rag.EmbeddingModelConfig(
    publisher_model="publishers/google/models/text-embedding-004"
)

# Configure Vector DB
vector_db = rag.VertexFeatureStore(resource_name=feature_view_name)

corpus = rag.create_corpus(
    display_name=display_name,
    description=description,
    embedding_model_config=embedding_model_config,
    vector_db=vector_db,
)
print(corpus)
# Example response:
# RagCorpus(name='projects/1234567890/locations/us-central1/ragCorpora/1234567890',
# display_name='test_corpus', description='Corpus Description', embedding_model_config=...
# ...
```

The RAG corpus is created and automatically associated with the Feature Store instance.

RAG APIs use the rag_corpus_id to handle the data upload to the Feature Store instance and to retrieve contexts from the rag_corpus_id.

After the synchronization process completes, you can retrieve relevant contexts from the FeatureOnlineStore index through the RetrieveContexts API.

Appendix

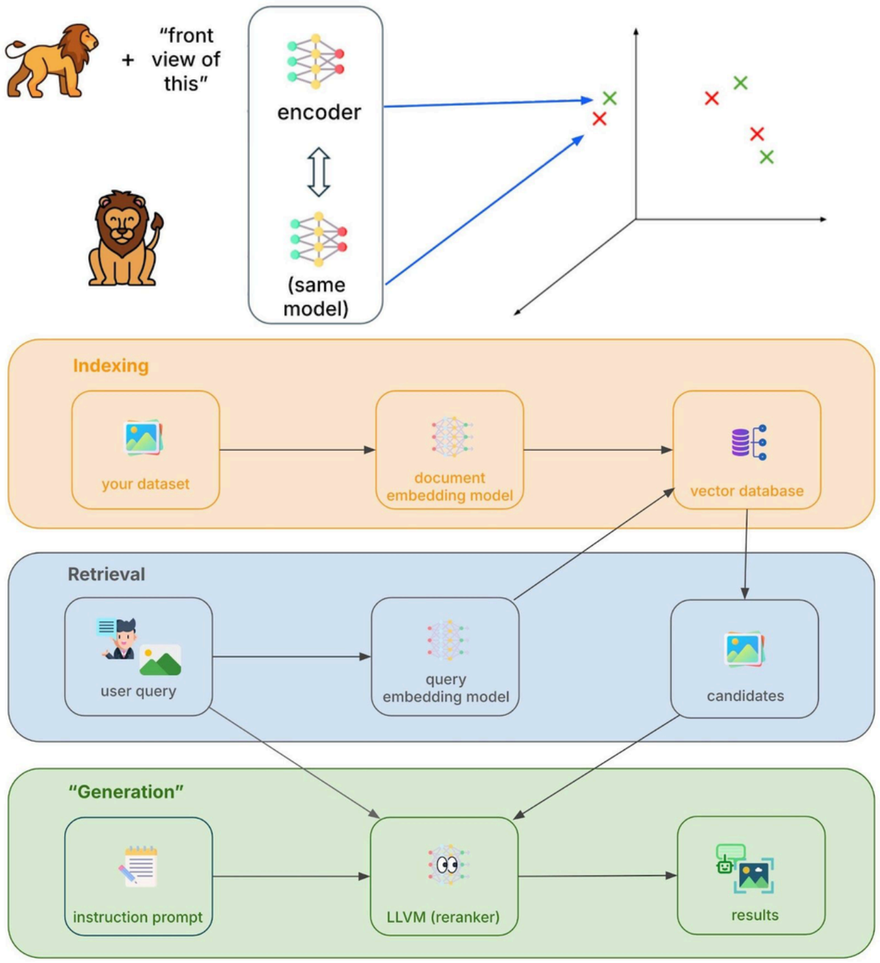
GENERATIVE AI EVALUATION

Metric	Importance	Evaluation Method
Accuracy	Ensures correct species identification	Comparison with labeled & **Human Evaluation**
Confidence Scores	Avoids unreliable classifications	Model's probability scores
Generalization	Tests performance on unseen species	Evaluate on unseen plankton images
Speed & Efficiency	Ensures practical use in research	Measure processing time per image
Bias & Hallucination	Prevents incorrect classifications	Cross-check AI output with expert labels

Human evaluation is important for accuracy, misclassifications + refine the model by feedback loop with expert knowledge

Appendix

MULTIMODAL SEMANTIC SEARCH IMAGES + TEXT



- Self-Supervised Learning: Uses web image pairs and foundation models to generate training data with 36.7M triplets.
- **Open-Ended Retrieval: Supports complex search intents beyond visual similarity**.
- Diverse Intent Handling: Interprets various search instructions in large-scale tests.

https://arxiv.org/abs/2403.19651

TRIPLETS consisting of three components:

- Query Image – The starting image for the retrieval task.
- Instruction – A description specifying how the retrieved image should relate to the query image.
- Target Image – The image that best matches the query image based on the given instruction.

Appendix

FREQUENTLY USED SEMANTIC VECTOR DISTANCE

Euclidean Distance

$$d(a,b) = d(b,a) = \sqrt{\sum_{i=0}^{n-1}(b_i - a_i)^2}$$

Measures the length of a segment that connects 2 points. It's the most commonly used distance metric and is very useful when the data are continuous.

Euclidean Distance

$$cos\theta = \frac{\sum_0^{n-1}(a_i \cdot b_i)}{\sqrt{\sum_0^{n-1} a_i^2} \cdot \sqrt{\sum_0^{n-1} b_i^2}}$$

Inner Product

$$p(A,B) = A \cdot B = \sum_{i=0}^{n-1} a_i \cdot b_i$$

IP is more useful if you need to compare non-normalized data or when you care about magnitude and angle.

If you use IP to calculate similarities between embeddings, you must normalize your embeddings. After normalization, the inner product equals cosine similarity.

Uses the cosine of the angle between two sets of vectors to measure how similar they are. The cosine similarity is always in the interval [-1, 1]. The larger the cosine, the smaller the angle between the two vectors, indicating that these two vectors are more similar to each other. By subtracting their cosine similarity from 1, you can get the cosine distance between two vectors.

Appendix

FINE TUNING GEN-AI BY LAYER OF INTERACTION

User Interface Layer

Skills Required:
Natural Language Understanding

- Model Configuration Layer
- Application Logic Layer
- Model Architecture Layer
- Pre-training Layer
- Evaluation & Monitoring Layer

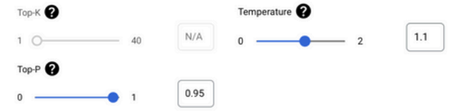
Prompt Engineering
Crafting specific prompts to guide the AI's response. This includes using structured templates, keywords, and context-setting within the prompt.

Example: To get a detailed story, instead of asking "Tell me a story," you might ask "Tell me a detailed and thrilling adventure story set in a medieval fantasy world."

Model Configuration Layer

Skills Required:
Understanding LLM parameters

- Application Logic Layer
- Model Architecture Layer
- Pre-training Layer
- Evaluation & Monitoring Layer



Parameter Tuning
Adjusting hyperparameters like temperature, top-k sampling, or top-p (nucleus) sampling to control the **diversity** and **creativity** of the generated content.

Top-k: Narrows choices to top-k tokens, then samples based on probability
Top-p : Chooses from tokens whose combined probability reaches a threshold p
The most important parameter is the temperature, it **affect the hallucination level**.

Application Logic Layer

Skills Required:
Programming (with/without LLM knowledge), Application Development – **Exception handling**.

- Model Architecture Layer
- Pre-training Layer
- Evaluation & Monitoring Layer

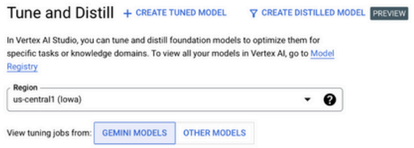
Rule-based Application Level Adjustments
Applying rules or heuristics to modify AI outputs based on the application's needs.
Example: Automatically append "Please provide more details." to any user query detected as too vague.

Post-processing Filters
Implementing filters to refine or alter the AI's output after generation, such as spell-checking, grammar correction, or content moderation

Model Architecture Layer

Skills Required:
Able to utilize tuning and distillation tools

- Pre-training Layer
- Evaluation & Monitoring Layer



Transfer Learning
Leveraging pre-trained models and fine-tuning them on specific tasks or datasets. This includes methods like using a pre-trained Bison or GPT model and fine-tuning it on a new dataset.

Distillation
Creating smaller, efficient models (student models) that approximate the performance of larger, pre-trained models (teacher models) through a distillation process.

Pre-training Layer

Skills Required:
Able to train LLM models

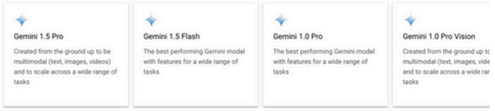
- Evaluation & Monitoring Layer

Domain-specific Pre-training
Pre-training the model on a large corpus of domain-specific data before fine-tuning it on task-specific data. For example, pre-training on medical literature for a healthcare application.

Example: Pre-training a language model on a corpus of biomedical research papers to enhance its ability to understand and generate medical literature.

Evaluation & Monitoring Layer

Skills Required:
Understand domain specific knowledge, to evaluate model performance



Continuous Evaluation
Implementing a feedback loop for continuous evaluation of the model's performance using metrics like accuracy, and user satisfaction.

A/B Testing
Conducting A/B tests to compare different versions of the model or configurations to determine the most effective approach.

Presentation slides are reused from my *GenAI Layers of Interaction* slides

End of Presentation!

Jaronchai Dilokkalayakul
Information Biology Laboratory,
Tohoku University 東北大学