Week 2 Report

Romouald Dombrovski

August 23, 2024

1. Abstract

Stevens, S. et al. (2024) “BIOCLIP: A Vision Foundation Model for the Tree of Life”. *arXiv preprint arXiv:2311.18803v3*. https://doi.org/10.48550/arXiv.2311.18803

Abstract: Images of the natural world, collected by a variety of cameras, from drones to individual phones, are increasingly abundant sources of biological information. There is an explosion of computational methods and tools, particularly computer vision, for extracting biologically relevant information from images for science and conservation. Yet most of these are bespoke approaches designed for a specific task and are not easily adaptable or extendable to new questions, contexts, and datasets. A vision model for general organismal biology questions on images is of timely need. To approach this, we curate and release TREEOFLIFE-10M, the largest and most diverse ML-ready dataset of biology images. We then develop BIOCLIP, a foundation model for the tree of life, leveraging the unique properties of biology captured by TREEOFLIFE-10M, namely the abundance and variety of images of plants, animals, and fungi, together with the availability of rich structured biological knowledge. We rigorously benchmark our approach on diverse fine-grained biology classification tasks and find that BIOCLIP consistently and substantially outperforms existing baselines (by 16% to 17% absolute). Intrinsic evaluation reveals that BIOCLIP has learned a hierarchical representation conforming to the tree of life, shedding light on its strong generalizability.

Summary (GPT-4o)**:** The paper "BIOCLIP: A Vision Foundation Model for the Tree of Life" introduces BIOCLIP, a novel vision model designed for biological image classification, leveraging a new dataset called TREEOFLIFE-10M, which is the largest and most diverse collection of biology images to date, containing over 10 million images spanning 454,000 taxa. BIOCLIP uses a contrastive learning approach, similar to CLIP, but uniquely incorporates the hierarchical taxonomy of life to enhance generalization, even to species not seen during training. The model significantly outperforms existing baselines in zero-shot and few-shot classification tasks by 16-17%, demonstrating its ability to effectively generalize across the tree of

life. The authors highlight BIOCLIP's potential to lower the barriers for applying AI in biological research, enabling more accurate and efficient analysis of biological images.

Relevance: This paper introduces a new model, BIOCLIP, based off the CLIP model which is currently being used in our application. As we explore novel ways to improve our process, we are discussing a comparison between existing open source models, and especially highly trained ones like the one mentioned in the paper should be in high consideration.

2. Scripts and Code Blocks

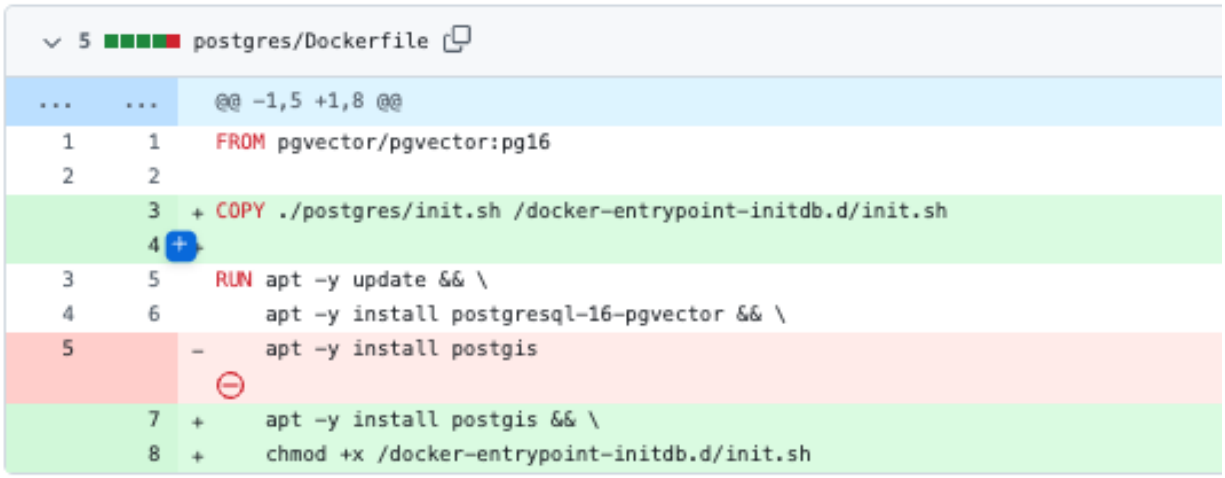
This week's efforts consisted of some improvement, and refactoring of the existing code base. This served three purposes: to allow faster on-boarding for future collaborators, to improve code quality and cleanliness, and to familiarize myself with the different working parts of the application.
My code changes can be found on the following pull requests:
https://github.com/Human-Augment-Analytics/NFHM/pull/29
https://github.com/Human-Augment-Analytics/NFHM/pull/30

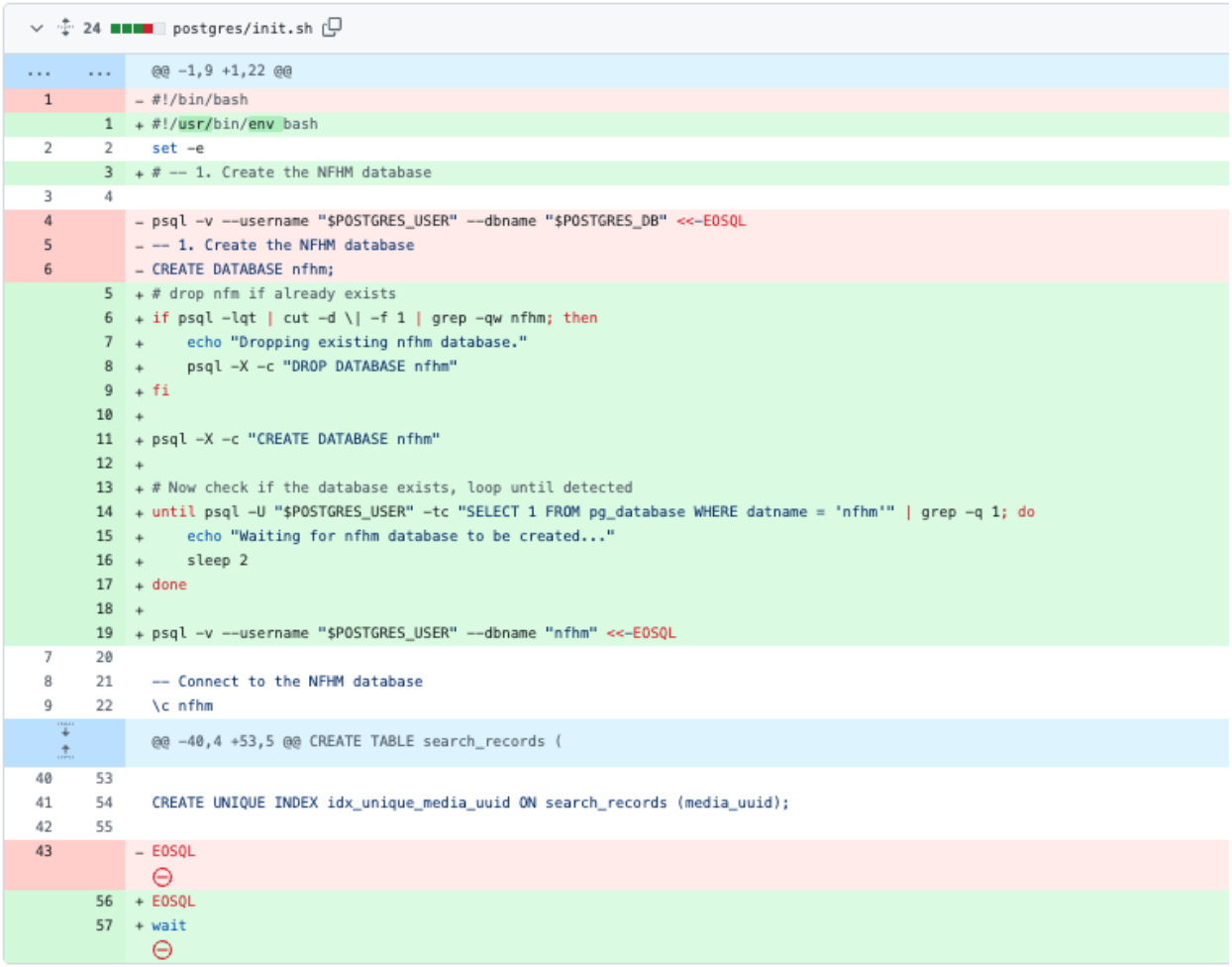
Dockerfile updates:
When going through the docker-compose process, I noticed one of the containers was not being built properly. I initially solved it by removing some extraneous bash code on the build process:

10 .devcontainer/compose.yml

```
@@ -59,11 +59,11 @@ services:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres
      POSTGRES_DB: nfhm
-    command: |
-      bash -c "
-        su - postgres -c '/usr/local/bin/docker-entrypoint.sh'
-        su - postgres -c '/usr/lib/postgresql/16/bin/postgres -D /var/lib/postgresql/data/'
-      "
+    # command: |
+    #   bash -c "
+    #     su - postgres -c '/usr/local/bin/docker-entrypoint.sh'
+    #     su - postgres -c '/usr/lib/postgresql/16/bin/postgres -D /var/lib/postgresql/data/'
+    #   "
     ports:
       - 5432:5432
```

This allowed the postgres container to build successfully (by going through the image's docker-entrypoint script), but our process of creating the database and relevant tables was not being accomplished. Through debugging, I was able to eventually update the Dockerfile and the initializing sql query which was being run on startup of the container:





```
24 ■■■■ postgres/init.sh

@@ -1,9 +1,22 @@
- #!/bin/bash
+ #!/usr/bin/env bash
  set -e
+ # -- 1. Create the NFHM database
- psql -v --username "$POSTGRES_USER" --dbname "$POSTGRES_DB" <<-EOSQL
- -- 1. Create the NFHM database
- CREATE DATABASE nfhm;
+ # drop nfm if already exists
+ if psql -lqt | cut -d \| -f 1 | grep -qw nfhm; then
+     echo "Dropping existing nfhm database."
+     psql -X -c "DROP DATABASE nfhm"
+ fi
+
+ psql -X -c "CREATE DATABASE nfhm"
+
+ # Now check if the database exists, loop until detected
+ until psql -U "$POSTGRES_USER" -tc "SELECT 1 FROM pg_database WHERE datname = 'nfhm'" | grep -q 1; do
+     echo "Waiting for nfhm database to be created..."
+     sleep 2
+ done
+
+ psql -v --username "$POSTGRES_USER" --dbname "nfhm" <<-EOSQL

  -- Connect to the NFHM database
  \c nfhm

@@ -40,4 +53,5 @@ CREATE TABLE search_records (

  CREATE UNIQUE INDEX idx_unique_media_uuid ON search_records (media_uuid);

- EOSQL
+ EOSQL
+ wait
```

Ingestor_scripts

The next upgrade was a smaller one. I noted that our ingestor jobs were being run using several cmdline commands in tandem.
e.g. To run the embedding worker:

```
conda activate ingestor_worker

export SOURCE_QUEUE="embedder"
export INPUT="inputs.vector_embedder"
export QUEUE="ingest_queue.RedisQueue"
export OUTPUT="outputs.index_to_postgres"

python ingestor/ingestor.py
```

To speed up this process, I simply put these commands into a bash script which could be run from the bin/ directory of the project:

18 bin/ingest_embed

```
@@ -0,0 +1,18 @@
+ #!/usr/bin/env sh
+
+ # This script is run as a postCreateCommand from the dev container
+
+ . "$(conda info --base)/etc/profile.d/conda.sh"
+
+ conda init
+ echo "Activating conda environment: Ingestor Worker"
+ echo "Generating Embeddings"
+
+ conda activate ingestor_worker
+
+ export SOURCE_QUEUE="embedder"
+ export INPUT="inputs.vector_embedder"
+ export QUEUE="ingest_queue.RedisQueue"
+ export OUTPUT="outputs.index_to_postgres"
+
+ python ingestor/ingestor.py
```

Vue component upgrade

I noticed that our app was built as one Vue component, and decided that it would be beneficial to refactor the code into multiple logical components. The changes are perhaps too much to put into this document, but are covered in the following commit:
https://github.com/Human-Augment-Analytics/NFHM/pull/30/commits/4d928ede102327389145d2614ee7c0d1d8875e8a

3. Documentation

With the script changes, the Readme should also change to reflect the usage of the ingestor_workers.

4. Proof of work

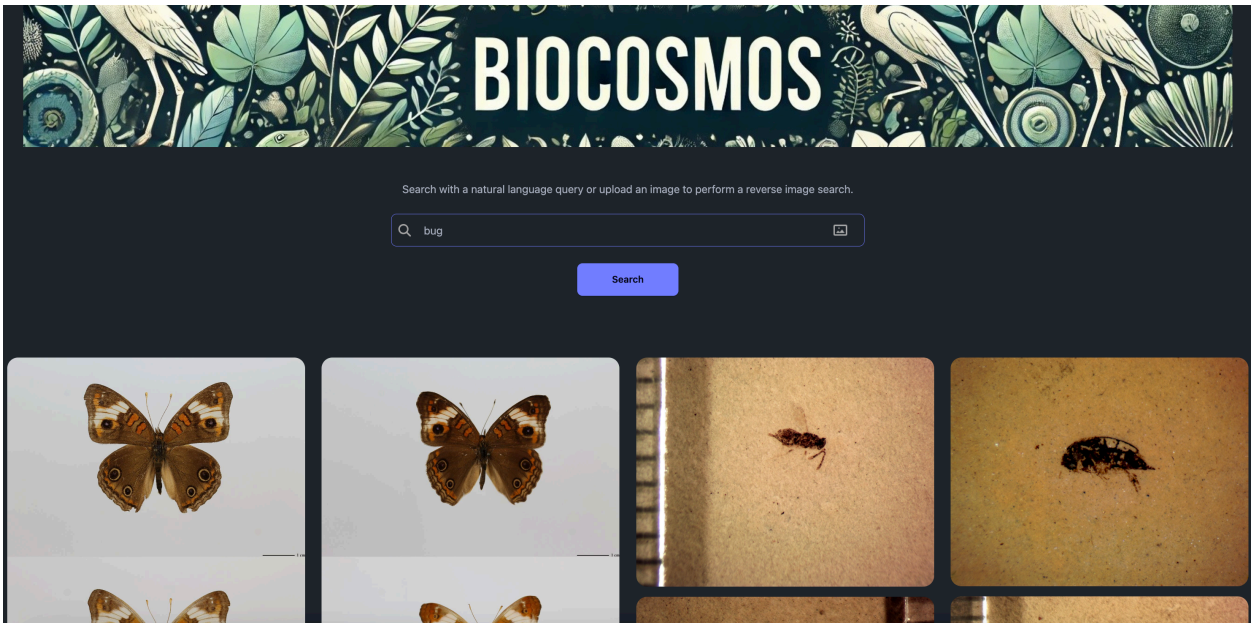
Proof that the postgres container is working:

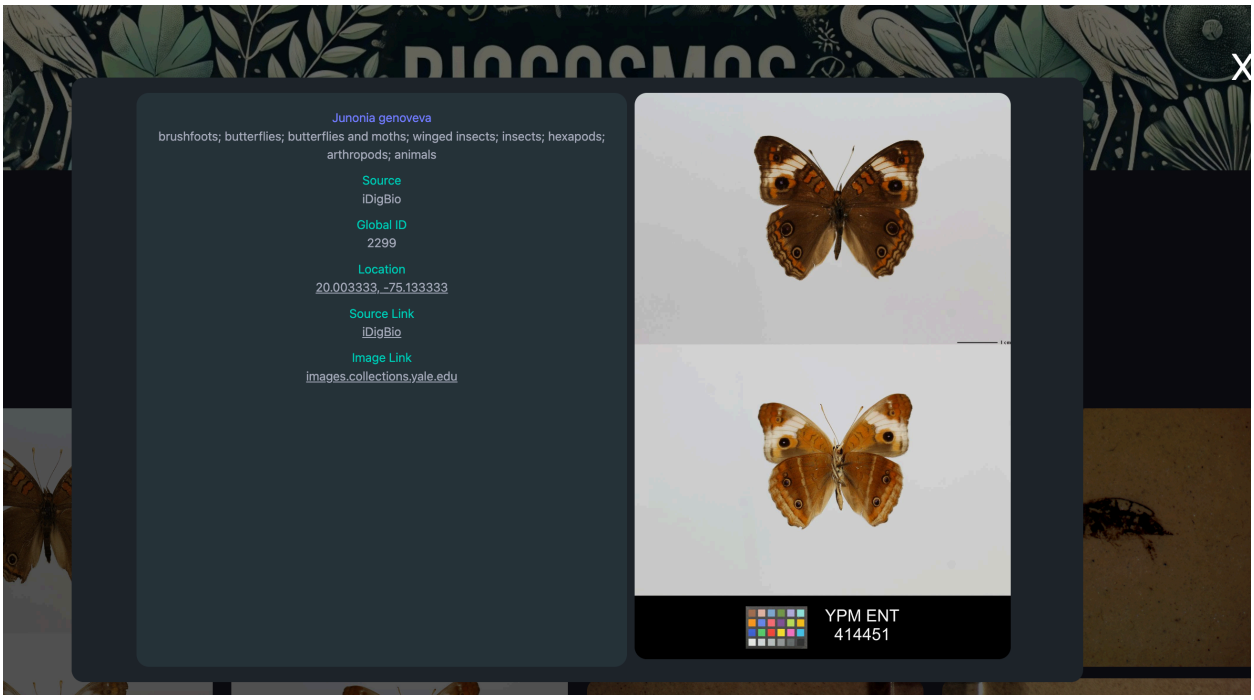
nfhm_devcontainer			Running (7/7)		6.32%	14 hours ago	
postgres-1 23c05df8089a	nfhm_devcontainer-postgres		Running	5432:5432	0%	16 hours ago	
mongo-express-1 4d3020246aea	mongo-express		Running	8081:8081	0.01%	16 hours ago	
jupyter-1 443b45ef525e	jupyter/datascience-notebook:latest		Running	8888:8888	0%	16 hours ago	
mongo-1 43019d4b05e1	mongo:latest		Running	27018:27017	0.96%	16 hours ago	

Proof that the scripts do as required:

```
(base) vscode → /workspaces/NFHM (vue-class-refactor) $ bin/ingest_embed
no change     /opt/conda/condabin/conda
no change     /opt/conda/bin/conda
no change     /opt/conda/bin/conda-env
no change     /opt/conda/bin/activate
no change     /opt/conda/bin/deactivate
no change     /opt/conda/etc/profile.d/conda.sh
no change     /opt/conda/etc/fish/conf.d/conda.fish
no change     /opt/conda/shell/condabin/Conda.psm1
no change     /opt/conda/shell/condabin/conda-hook.ps1
no change     /opt/conda/lib/python3.12/site-packages/xontrib/conda.xsh
no change     /opt/conda/etc/profile.d/conda.csh
no change     /home/vscode/.bashrc
No action taken.
Activating conda environment: Ingestor Worker
Generating Embeddings
2024-08-30 19:55:28,295 - asyncio - DEBUG - Using selector: EpollSelector
2024-08-30 19:55:28,295 - __main__ - INFO - {"source_queue":"embedder","redis":{"host":"redis","port":6379,"database":0,"username":null,"password":null},"mongo":{"host":"mongo","port":27017,"
database":"NHFM","username":"root","password":"example","input_collection":"idigbio"},"number_of_workers":1,"postgres":{"host":"postgres","port":5432,"database":"nfhm","table":"search_records
","user":"postgres","password":"postgres"},"queue":"ingest_queue.redis_queue.RedisQueue","input":"inputs.vector_embeddings.vector_embedder","output":"outputs.postgres_output.index_to_postgres
```

Proof that the application runs after refactoring components:





5. Next Week Proposal

This week we met up with our future collaborators from the ACIS lab at UF. Many of the tasks that we'll be working on will come up as we divide ourselves into sub-teams. Here are some tasks that I foresee working on in the coming week:

- help set up our application on the ACIS infrastructure
- help set up Vite build tool for faster front end building
- help update components from js modules to vue modules
- explore alternative models to be used in our pipeline (e.g. Florence, BioClip, etc.)