Week 2 Report

Thomas Deatherage
NFHM/BioCosmos
30 August 2024

Time Slot

1) What progress did you make in the last week?
    - Met with the new collaborators at the University of Florida at the ACIS lab.
        - Discussed team organization and goals
    - Merged last week's code changes https://github.com/Human-Augment-Analytics/NFHM/pull/28
    - GaTech team meeting (Vy, Roman, and myself)
    - More "productizing" the codebase (more on this below in the report): https://github.com/Human-Augment-Analytics/NFHM/pull/31
2) What are you planning on working on next?
    - Most likely meeting with ingestor task team to talk about deployment
    - Possibly meet with ML task team.
    - Code contributions as appropriate based on above meetings
3) Is anything blocking you from getting work done?
    - Nope.

Abstracts & Summaries

1) VLM4Bio: A Benchmark Dataset to Evaluate Pretrained Vision-Language Models for Trait Discovery from Biological Images

Conference/Journal: Not stated
Abstract: Images are increasingly becoming the currency for documenting biodiversity on the planet, providing novel opportunities for accelerating scientific discoveries in the field of organismal biology, especially with the advent of large vision-language models (VLMs). We ask if pre-trained VLMs can aid scientists in answering a range of biologically relevant questions without any additional fine-tuning. In this paper, we evaluate the effectiveness of 12

state-of-the-art (SOTA) VLMs in the field of organismal biology using a novel dataset, VLM4Bio, consisting of 469K question-answer pairs involving 30K images from three groups of organisms: fishes, birds, and butterflies, covering five biologically relevant tasks. We also explore the effects of applying prompting techniques and tests for reasoning hallucination on the performance of VLMs, shedding new light on the capabilities of current SOTA VLMs in answering biologically relevant questions using images.
Summary: This paper explores whether advanced computer models can help scientists answer questions about organisms using images without needing extra training. They tested 12 models on a dataset of images of fishes, birds, and butterflies to see how well they could perform tasks related to biology, like identifying species or behaviors.
Link: https://arxiv.org/abs/2408.16176
Code: https://github.com/sammarfy/VLM4Bio

Scripts and Code Blocks

Firstly, I merged last weeks code contributions into the `main` branch this week: https://github.com/Human-Augment-Analytics/NFHM/pull/28.

Additionally, I've opened a new PR with a number of significant improvements in preparation for deploying our code:

1) Properly instrument env vars into the backend API container via Docker.
2) Use a connection pool to for Postgres. Connection pools are generally the correct way for a production app to connect to its database.
3) Model improvements: Actually take advantage of FastAPI's API modeling tools. This improves both type-enforcement in the application code and swagger's auto-generated documentation/client.
4) Slimmer docker image.
5) Where available, use Nvidia GPUs in both the ingestor and the backend API to generate embeddings. I was not able to manually test this as I don't have access to a Nvidia machine.
6) Improved type checking/hinting where possible. (Note this is not always trivial, especially when external packages have little-to-no type help).
7) Per Roman's work, also added a bin/ script for running the idigbio ingestor.
8) Updated documentation to reflect changes from #7

These changes can be found here: https://github.com/Human-Augment-Analytics/NFHM/pull/31

There's too much code to repeat here in its entirety, so I'll focus on the important highlights:

Here I'm using a connection pool for Postgres.

```
15  + # Configure logging
16  + logging.basicConfig(level=logging.INFO)
17  + logger = logging.getLogger(__name__)
18  +
19  +
20  + settings = Settings()
21  +
22  + def get_db_engine() -> AsyncEngine:
23  +     return create_async_engine(
24  +         settings.database_url,
25  +         echo=True,
26  +         poolclass=AsyncAdaptedQueuePool,
27  +         pool_size=settings.pool_size,
28  +         max_overflow=settings.max_overflow
29  +     )
```

Use a GPU (Apple or Cuda) if available. This is done configured for the OpenClip library that we're using to generate vectors in both the backend API and the ingestor.

```
+     logger.info("Starting up application...")
+     try:
+         # MPS is for apple silicon GPU, cuda is ofc nvidia.
+         device = torch.device("mps:0" if torch.backends.mps.is_available() else
  "cuda" if torch.cuda.is_available() else "cpu")
+         logger.info(f"Using device: {device}")
+
```

Simple script to run the idigbio ingestor. Credit to Roman for writing scripts for the other ingestor jobs.

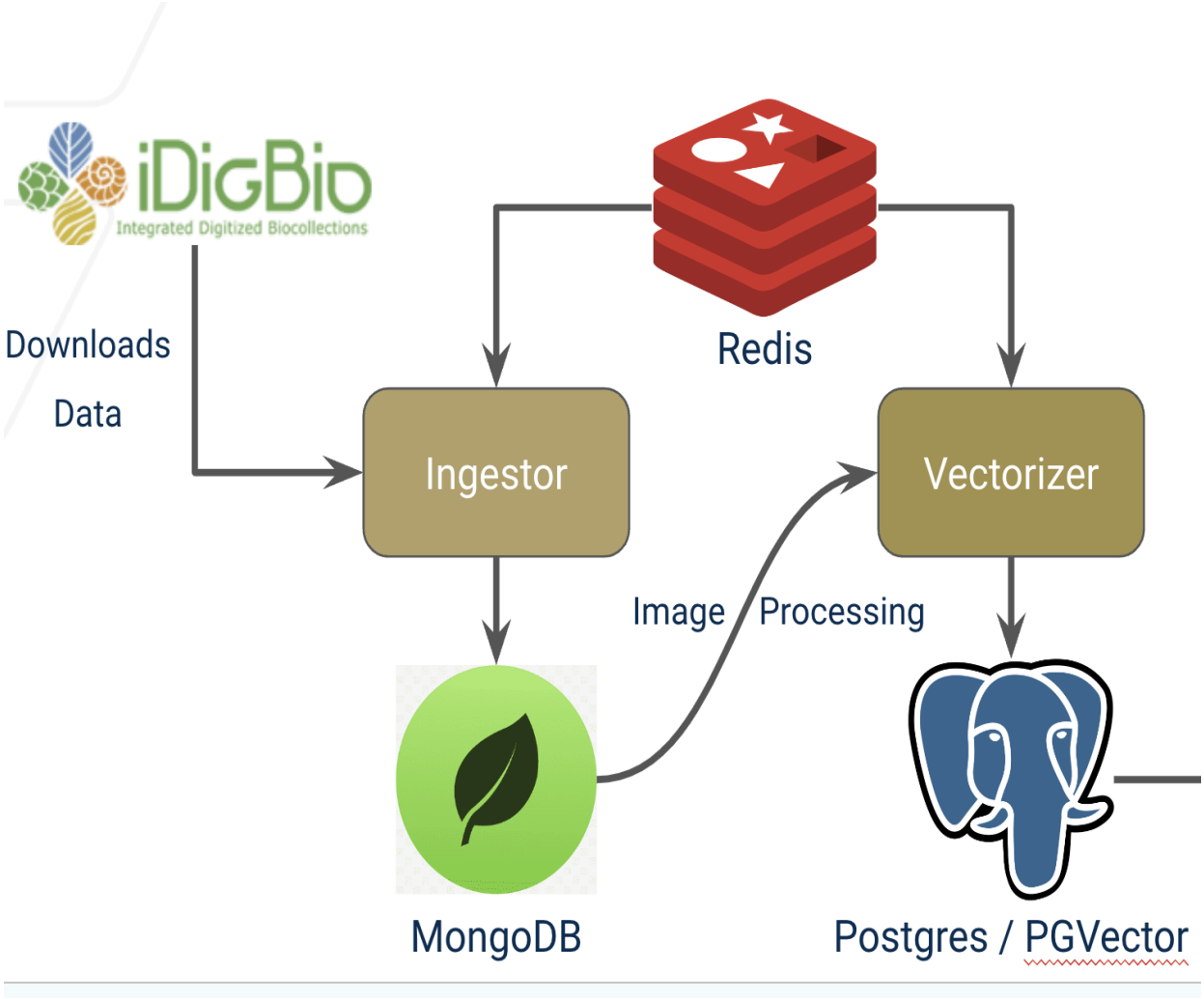
```
+ #!/usr/bin/env sh
+
+ . "$(conda info --base)/etc/profile.d/conda.sh"
+
+ conda init
+ echo "Activating conda environment: Ingestor Worker"
+ echo "IDGIBIO Search Dump to MongoDB"
+
+
+ conda activate ingestor_worker
+
+  export SOURCE_QUEUE="idigbio"
+  export INPUT="inputs.idigbio_search"
+  export QUEUE="ingest_queue.RedisQueue"
+  export OUTPUT="outputs.dump_to_mongo"
+
+  python ingestor/ingestor.py
```

This configuration changes feed host machine env vars to the container (and therefore the backend API app), falling back to local-specific defaults:

```
+       - default
+     environment:
+       - MODEL_NAME=${MODEL_NAME:-ViT-B-32}
+       - MODEL_PRETRAINED=${MODEL_PRETRAINED:-laion2b_s34b_b79k}
+       - DATABASE_URL=${DATABASE_URL:-
  postgresql+asyncpg://postgres:postgres@postgres/nfhm}
+       - POOL_SIZE=${POOL_SIZE:-5}
+       - MAX_OVERFLOW=${MAX_OVERFLOW:-10}
+     # Uncomment below in order to utilize nvidia GPU with the backend api
+     # deploy:
+     #   resources:
+     #     reservations:
+     #       devices:
+     #         - driver: nvidia
+     #           count: 1
+     #           capabilities: [gpu]
+
--
```

Flow Charts/Diagrams

I included this image in my previous report. I'll repeat it here for context.



Documentation

I updated the readme to reflect the newly available scripts for running the various ingestor jobs. Notably, these changes greatly simplify things.

Seeding Mongo with a sample of iDigBio data:

1. ~~Activate the ingestor_worker conda environment:~~ ~~`conda activate ingestor_worker`~~
2. ~~Start by spinning up the iDigBio worker.~~
   - ~~The worker pulls in environment variables to determine which queue to pull from and which worker functions to call. Consequently, you can either set those variables in~~ ~~`.devcontainer/devcontainer.json`~~ ~~-- which will require a rebuild and restart of the dev container -- or you can set them in via the command line. We'll do the latter:~~
     - ~~(from within the dev container):~~
       - ~~Open new tab (or reload terminal) to make sure conda can init:~~
         - ~~`conda activate ingestor_worker`~~
       - ~~Set env vars, e.g.,:~~
         - ~~`export SOURCE_QUEUE="idigbio" // Indicates which queue to read from`~~
           ~~`export INPUT="inputs.idigbio_search" // Indicates which input function to run for the job. I`~~
           ~~`export QUEUE="ingest_queue.RedisQueue" // Indicates which queueing backend to use. Currently`~~
           ~~`export OUTPUT="outputs.dump_to_mongo" // Indicates the output function to run. Output functi`~~
       - ~~Run the job~~
         - ~~`python ingestor/ingestor.py`~~
1. From within a dev container: `$ bin/ingest_idigbio`

The basic process of seeding Mongo with raw GBIF data is essentially the same as with iDigBio. However, you'll need make sure you have the GBIF worker up-and-running in your dev container with the correct environment inputs:

- ~~`conda activate ingestor_worker`~~

  ~~`export SOURCE_QUEUE="gbif"`~~
  ~~`export INPUT="inputs.gbif_search"`~~
  ~~`export QUEUE="ingest_queue.RedisQueue"`~~
  ~~`export OUTPUT="outputs.dump_to_mongo"`~~

  ~~`python ingestor/ingestor.py // Run the job`~~
- `$ bin/ingest_gbif`
- From the workbench of Redis Insight, pass a simple search string to the `gbif` queue:

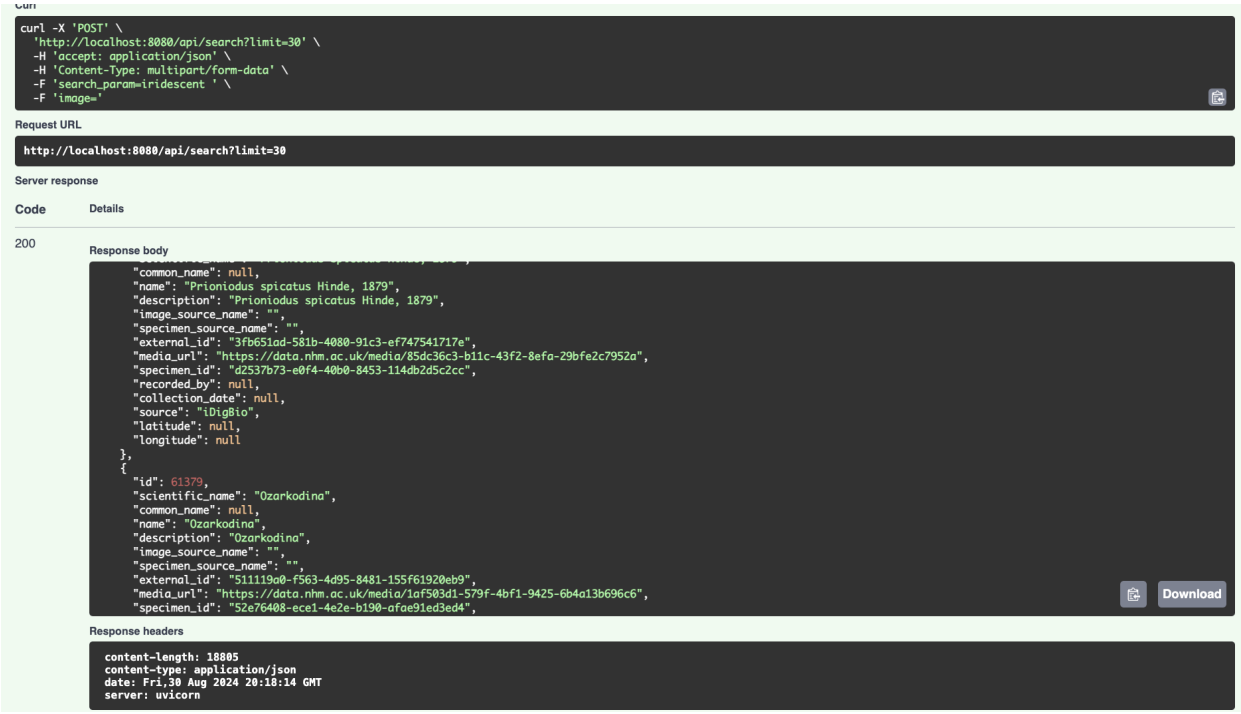
Results Visualization

No results to visualize this week.

Proof of Work

The backend API and ingestor workers work as expected. Here's screencaps showing as much.

The API returns results as usual:



And the ingestor workers run as usual:

```
(ingestor_worker) vscode ➜ /workspaces/NFHM (refactor/more-pre-deploy-code-improvements) $ bin/ingest_idigbio
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/python3.11/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
IDGIBIO Search Dump to MongoDB
/opt/conda/lib/python3.11/site-packages/transformers/utils/generic.py:441: FutureWarning: `torch.utils._pytree._register_pytree_node` is deprecated. P
lease use `torch.utils._pytree.register_pytree_node` instead.
  _torch_pytree._register_pytree_node(
/opt/conda/lib/python3.11/site-packages/transformers/utils/generic.py:309: FutureWarning: `torch.utils._pytree._register_pytree_node` is deprecated. P
lease use `torch.utils._pytree.register_pytree_node` instead.
  _torch_pytree._register_pytree_node(
2024-08-30 20:17:02,698 - vector_embedder - INFO - Using device: cpu
2024-08-30 20:17:02,701 - asyncio - DEBUG - Using selector: EpollSelector
2024-08-30 20:17:02,701 - __main__ - INFO - {"source_queue":"idigbio","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":12,"
postgres":{"host":"postgres","port":5432,"database":"nfhm","table":"search_records","user":"postgres","password":"postgres"},"queue":"ingest_queue.red
is_queue.RedisQueue","input":"inputs.idigbio.idigbio_search","output":"outputs.mongo_output.dump_to_mongo"}
2024-08-30 20:17:02,704 - pymongo.serverSelection - INFO - {"message": "Waiting for suitable server to become available", "selector": "Primary()", "op
eration": "listDatabases", "topologyDescription": "<TopologyDescription id: 66d228be5ace33a052377e6c, topology_type: Unknown, servers: [<ServerDescrip
tion ('mongo', 27017) server_type: Unknown, rtt: None>]>", "clientId": {"$oid": "66d228be5ace33a052377e6c"}, "remainingTimeMS": 29}
2024-08-30 20:17:12,747 - worker.redis - INFO - No message received in 10
2024-08-30 20:17:12,748 - worker.redis - INFO - No message received in 10
2024-08-30 20:17:12,748 - worker.redis - INFO - No message received in 10
```

Next Week's Proposal

- Meeting with the ingestor task team to discuss deployment
    - Take action accordingly
- Meet with Roman and Vy (and optionally Bree and Dr Porto) as usual on Wednesday.

- I think we'll want to discuss what to do with these recurring Wednesday meetings as our team structure has changed.