Week 6 Report - NFHM

Romouald Dombrovski

September 27, 2024

1. Time Log

What progress did you make last week?

- Obtained nearest neighbor (k=1) script for image embedding evaluation
- Upgraded metrics evaluation pipeline for comparison of openclip, bioclip, Florence-2
- Obtained result for stratified comparison using cosine similarity on VLM4Bio dataset (Fish, Bird, Butterfly)
- Researched various training methodologies for Florence-2 model

2. Abstract

Yang, C., et al. 2024. Arboretum: A Large Multimodal Dataset Enabling AI for Biodiversity. *arXiv:2406.17720v1*. http://doi.org/10.48550/arXiv.2406.17720

Abstract: We introduce ARBORETUM, the largest publicly accessible dataset designed to advance AI for biodiversity applications. This dataset, curated from the iNaturalist community science platform and vetted by domain experts to ensure accuracy, includes 134.6 million images, surpassing existing datasets in scale by an order of magnitude. The dataset encompasses image-language paired data for a diverse set of species from birds (Aves), spiders/ticks/mites (Arachnida), insects (Insecta), plants (Plantae), fungus/mushrooms (Fungi), snails (Mollusca), and snakes/lizards (Reptilia), making it a valuable resource for multimodal vision-language AI models for biodiversity assessment and agriculture research. Each image is annotated with scientific names, taxonomic details, and common names, enhancing the robustness of AI model training. We showcase the value of ARBORETUM by releasing a suite of CLIP models trained using a subset of 40 million captioned images. We introduce several new benchmarks for rigorous assessment, report accuracy for zero-shot learning, and evaluations across life stages, rare species, confounding species, and various levels of the taxonomic hierarchy. We anticipate that ARBORETUM will spur the development of AI models that can enable a variety of digital tools ranging from pest control strategies, crop monitoring, and worldwide biodiversity assessment and environmental conservation. These

advancements are critical for ensuring food security, preserving ecosystems, and mitigating the impacts of climate change. ARBORETUM is publicly available, easily accessible, and ready for immediate use. Please see the project website for links to our data, models, and code.

Summary (GPT-4o)**:** The paper introduces ARBORETUM, the largest publicly accessible multimodal dataset designed to advance AI applications in biodiversity and agriculture research. Curated from the iNaturalist community science platform, ARBORETUM comprises 134.6 million images across seven taxonomic classes, surpassing existing datasets in scale and diversity. Each image is paired with comprehensive annotations, including scientific and common names, and is vetted by experts for accuracy. To demonstrate its utility, the authors release a suite of CLIP-based vision-language models trained on a 40 million image subset, setting benchmarks for zero-shot classification and generalization across species, life stages, and unseen classes. ARBORETUM aims to enable the development of AI models that support tasks like species identification, pest control, and environmental conservation, addressing the challenges posed by traditional datasets like ImageNet. The dataset, associated models, and code are publicly available to facilitate research and practical applications in biodiversity monitoring and conservation efforts.

3. Scripts, Code Blocks and Visualization

Files taken in as df, and models used to convert image into embeddings:

	image_filename	scientificName	taxa	image_embeddings
0	JFBM-FISH-0028839.jpg	Cottus bairdii	Fish	[[[0.8877, -0.5737, -0.155, 0.015015, 0.994, 0...
1	UWZM-F-0004319.JPG	Cottus bairdii	Fish	[[[0.3105, -0.7305, 0.5195, -0.2764, 1.129, 0....
2	JFBM-FISH-0046588.jpg	Cottus perplexus	Fish	[[[0.4592, -0.5615, 0.11847, -0.0919, 1.135, 0...
3	JFBM-FISH-0030139.jpg	Cottus bairdii	Fish	[[[0.6714, -0.635, 0.2915, 0.0762, 1.375, 0.51...
4	JFBM-FISH-0027155.jpg	Cottus bairdii	Fish	[[[0.764, -0.605, 5.92e-05, -0.4448, 0.91, 0.7...
...	...	...	...	...
195	Butterfly_imbalanced_train_Temenis_laothoe_303...	Temenis laothoe	Butterfly	[[[0.1575, 0.639, 0.2217, -0.779, 1.239, 0.913...
196	Butterfly_imbalanced_train_Temenis_laothoe_294...	Temenis laothoe	Butterfly	[[[0.3545, 0.635, 0.04785, -0.7617, 1.113, 0.8...
197	Butterfly_imbalanced_train_Temenis_laothoe_359...	Temenis laothoe	Butterfly	[[[0.2137, 0.744, 0.1417, -0.836, 1.591, 1.001...
198	Butterfly_imbalanced_train_Temenis_laothoe_364...	Temenis laothoe	Butterfly	[[[0.1431, 0.9116, 0.1078, -0.7437, 1.207, 0.9...
199	Butterfly_imbalanced_train_Temenis_laothoe_297...	Temenis laothoe	Butterfly	[[[0.3555, 0.595, -0.003891, -0.8804, 1.303, 0...

600 rows × 4 columns

Split test/train for eval, stratify for rare images:

```python
from sklearn.model_selection import train_test_split

def get_test_train_split(df_cleaned, test_size=0.1, stratify_by_scientific_name=False):
  if stratify_by_scientific_name:
    # if we want to get at least one from each we can filter out all options with only 1 image
    species_counts = df_cleaned['scientificName'].value_counts()
    valid_species = species_counts[species_counts > 1].index
    df_filtered = df_cleaned[df_cleaned['scientificName'].isin(valid_species)]
    # with stratification, we need to specify the test_size because we need to hit a minimum
    min_split = len(valid_species)/len(df_filtered)
    print(f'The min split for stratification is: {min_split}')

    train_df, test_df = train_test_split(df_filtered, test_size=min_split, stratify=df_filtered['scientificName'])
  else:
    train_df, test_df = train_test_split(df_cleaned, test_size=test_size)
  return train_df, test_df
```

Cosine Similarity comparison

```python
# Function to calculate cosine similarity between two 3D embeddings
def calculate_similarity(embedding1, embedding2, model):
    # Average across the token dimension (axis=1) to get a single vector
    if model == 'florence':
      embedding1_avg = np.mean(embedding1, axis=1)
      embedding2_avg = np.mean(embedding2, axis=1)
    else:
      embedding1_avg, embedding2_avg = embedding1, embedding2

    # Calculate cosine similarity between the averaged embeddings
    similarity = cosine_similarity(embedding1_avg, embedding2_avg)
    return similarity[0][0]
```

Loop for comparison:

```python
models = ['florence', 'clip', 'bioclip']
analysis_taxa = ['Fish', 'Bird', 'Butterfly', 'All']
eval_df = pd.DataFrame(columns=['model', 'taxa', 'column', 'accuracy_individual', 'accuracy_summed'])
rows = []

full_species_df['genus'] = full_species_df['scientificName'].apply(extract_genus)

# should run on L4 --> 3 models and a lot of RAM usage
# cleaned_dfs.append(full_species_df)
for model_name in models:
  print(f"WORKING ON MODEL: {model_name}")
  model, processing = get_model(model_name)

  get_image_embeddings = get_image_embeddings_florence if model_name == 'florence' else get_image_embeddings_clip
  full_species_df['image_embeddings'] = full_species_df['image_filename'].apply(lambda x: get_image_embeddings(x, model, processing))

  # we're gonna compare species by species within each taxa, and against each
  for taxa in analysis_taxa: # cleaned_dfs can have full species appended to it too
    print(f"WORKING ON TAXA: {taxa}")

    df = full_species_df if taxa == 'All' else full_species_df[full_species_df['taxa'] == taxa]
    train_df, test_df = get_test_train_split(df, test_size=0.2, stratify_by_scientific_name=True)

    test_df, accuracy_individual, accuracy_summed = apply_best_match(test_df, train_df, 'scientificName', model_name)
    rows.append({'model': model_name, 'taxa': taxa, 'column': 'species', 'accuracy_individual': f"{accuracy_individual * 100:.2f}%", 'accuracy_summed': f"{accuracy_summed * 100:.2f}%"})
    test_df, accuracy_individual, accuracy_summed = apply_best_match(test_df, train_df, 'genus', model_name)
    rows.append({'model': model_name, 'taxa': taxa, 'column': 'genus', 'accuracy_individual': f"{accuracy_individual * 100:.2f}%", 'accuracy_summed': f"{accuracy_summed * 100:.2f}%"})
    if taxa == 'All': # only relevant when comparing againt other taxa
      test_df, accuracy_individual, accuracy_summed = apply_best_match(test_df, train_df, 'taxa', model_name)
      rows.append({'model': model_name, 'taxa': taxa, 'column': 'taxa', 'accuracy_individual': f"{accuracy_individual * 100:.2f}%", 'accuracy_summed': f"{accuracy_summed * 100:.2f}%"})

eval_df = pd.concat([eval_df, pd.DataFrame(rows)], ignore_index=True)
eval_df
```

Result:

	model	taxa	column	accuracy_individual	accuracy_summed
0	florence	Fish	species	60.00%	24.00%
1	florence	Fish	genus	84.00%	36.00%
2	florence	Bird	species	47.62%	4.76%
3	florence	Bird	genus	88.10%	38.10%
4	florence	Butterfly	species	86.36%	18.18%
5	florence	Butterfly	genus	95.45%	40.91%
6	florence	All	species	55.06%	13.48%
7	florence	All	genus	88.76%	44.94%
8	florence	All	taxa	100.00%	100.00%
9	clip	Fish	species	56.00%	24.00%
10	clip	Fish	genus	84.00%	36.00%
11	clip	Bird	species	38.10%	9.52%
12	clip	Bird	genus	76.19%	71.43%
13	clip	Butterfly	species	72.73%	18.18%
14	clip	Butterfly	genus	100.00%	72.73%
15	clip	All	species	59.55%	13.48%
16	clip	All	genus	85.39%	66.29%
17	clip	All	taxa	100.00%	100.00%
18	bioclip	Fish	species	64.00%	36.00%
19	bioclip	Fish	genus	92.00%	76.00%
20	bioclip	Bird	species	83.33%	35.71%
21	bioclip	Bird	genus	97.62%	90.48%
22	bioclip	Butterfly	species	86.36%	50.00%
23	bioclip	Butterfly	genus	100.00%	95.45%
24	bioclip	All	species	80.90%	32.58%
25	bioclip	All	genus	95.51%	79.78%
26	bioclip	All	taxa	100.00%	100.00%

4. Next Week Proposal

- Add Arboretum model to evaluation pipeline.
- Add other embedding nearest neighbor comparison algorithms (e.g. Euclidean distance).
- Add a k=5 comparison to the evaluation pipeline.
- Run all images dataset (~10,000/taxa) evaluation.
- Create a rundown of all Florence-2 tasks and examples of relevant use cases for Biocosmos.
- Draft of roadmap for project.