Week 4 Report

Romouald Dombrovski

September 13, 2024

1. Abstract

Xiao, B. et al. (2023) “Florence-2: Advancing a Unified Representation for a Variety of Vision Tasks”. *arXiv:2311.06242v1*. http://doi.org/10.48550/arXiv.2311.06242

Abstract: We introduce Florence-2, a novel vision foundation model with a unified, prompt-based representation for a variety of computer vision and vision-language tasks. While existing large vision models excel in transfer learning, they struggle to perform a diversity of tasks with simple instructions, a capability that implies handling the complexity of various spatial hierarchy and semantic granularity. Florence-2 was designed to take text-prompt as task instructions and generate desirable results in text forms, whether it be captioning, object detection, grounding or segmentation. This multi-task learning setup demands largescale, high-quality annotated data. To this end, we codeveloped FLD-5B that consists of 5.4 billion comprehensive visual annotations on 126 million images, using an iterative strategy of automated image annotation and model refinement. We adopted a sequence-to-sequence structure to train Florence-2 to perform versatile and comprehensive vision tasks. Extensive evaluations on numerous tasks demonstrated Florence-2 to be a strong vision foundation model contender with unprecedented zero-shot and fine-tuning capabilities.

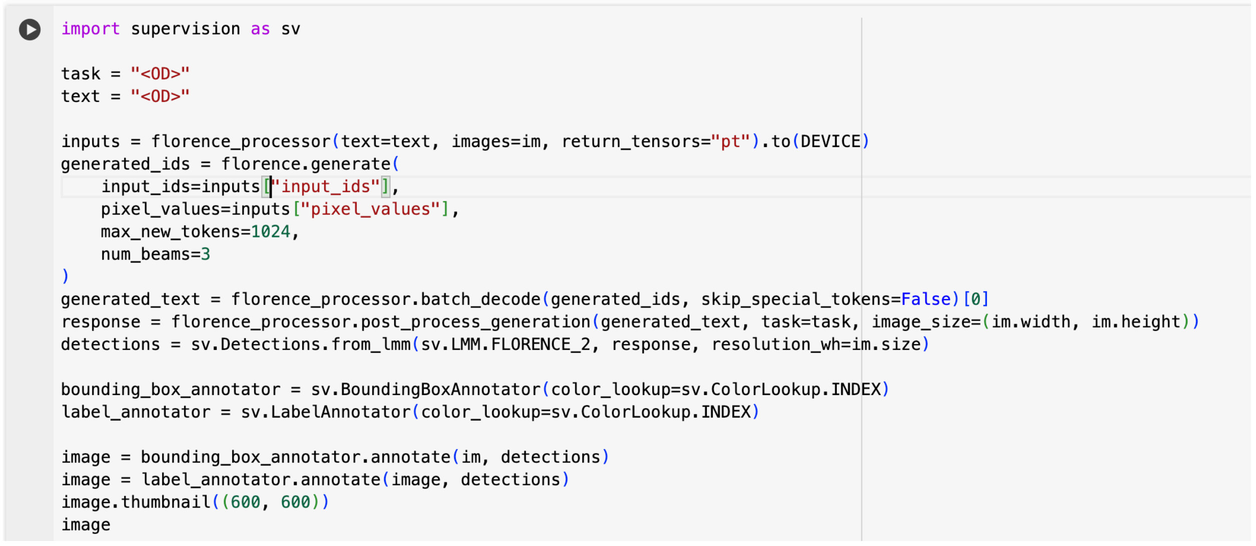
Summary (GPT-4o)**:** The *Florence-2* paper presents a unified vision foundation model capable of performing various computer vision and vision-language tasks within a single architecture. Building upon a sequence-to-sequence framework, the model uses both visual token embeddings from images and text-based prompts to execute tasks such as object detection, segmentation, visual grounding, and image captioning. It integrates a transformer-based multi-modal encoder-decoder, enabling the model to process visual and textual data simultaneously. Florence-2 is trained on a massive dataset called FLD-5B, comprising 126 million images and over 5.4 billion annotated visual-text pairs, which enhances its performance in both zero-shot and fine-tuning settings. By removing the need for task-specific heads and adopting a prompt-based, task-agnostic approach, the model offers a versatile

solution for multitask learning, achieving top-tier results across a broad spectrum of benchmarks.

Relevance:

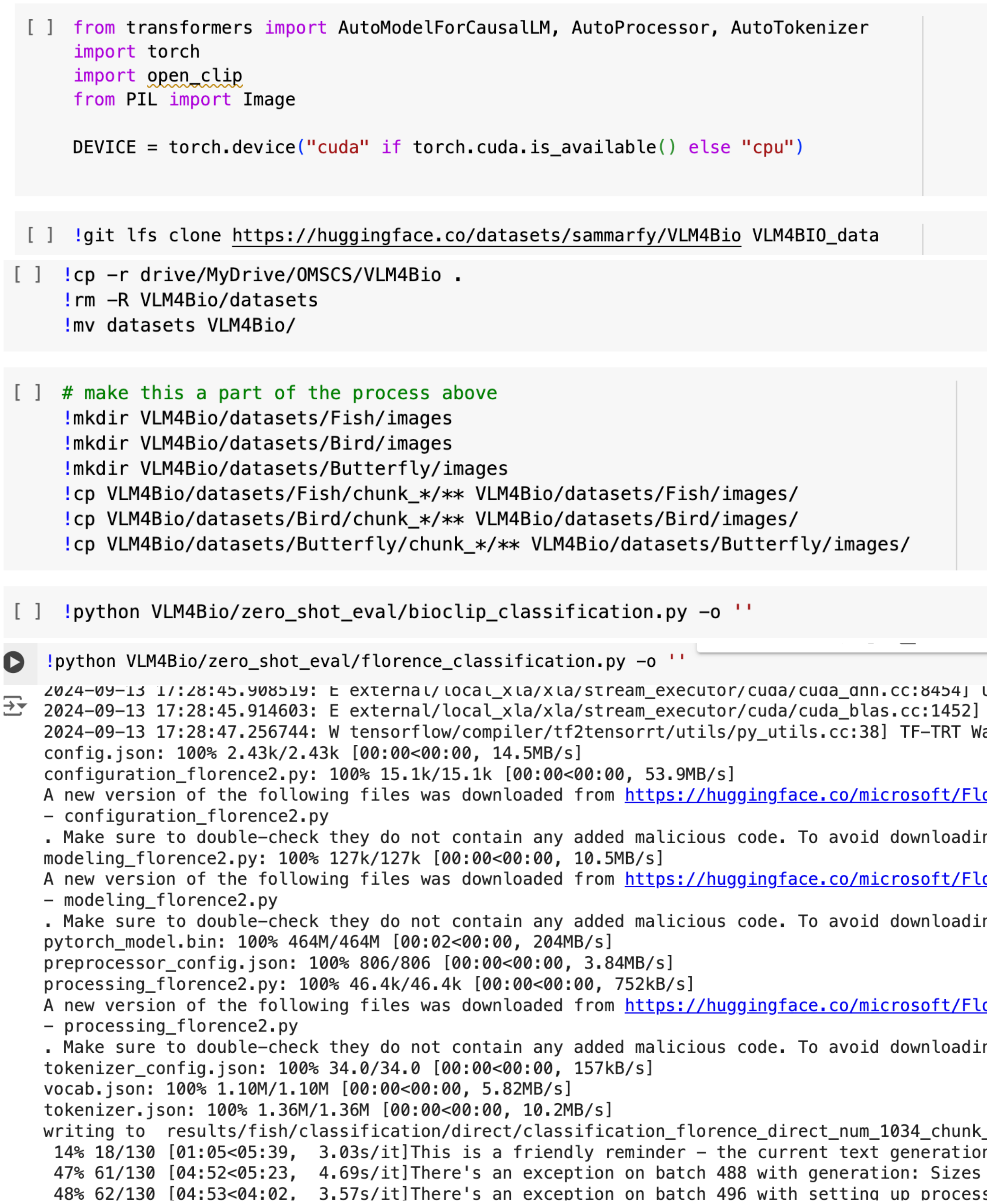
2. Scripts and Code Blocks

This week's efforts were focused on the Machine Learning aspect of the application. I began by learning more about the Florence-2 model (https://blog.roboflow.com/florence-2/), following along with the Roboflow tutorial to understand the limitations of the model.



```
import supervision as sv

task = "<OD>"
text = "<OD>"

inputs = florence_processor(text=text, images=im, return_tensors="pt").to(DEVICE)
generated_ids = florence.generate(
    input_ids=inputs["input_ids"],
    pixel_values=inputs["pixel_values"],
    max_new_tokens=1024,
    num_beams=3
)
generated_text = florence_processor.batch_decode(generated_ids, skip_special_tokens=False)[0]
response = florence_processor.post_process_generation(generated_text, task=task, image_size=(im.width, im.height))
detections = sv.Detections.from_lmm(sv.LMM.FLORENCE_2, response, resolution_wh=im.size)

bounding_box_annotator = sv.BoundingBoxAnnotator(color_lookup=sv.ColorLookup.INDEX)
label_annotator = sv.LabelAnnotator(color_lookup=sv.ColorLookup.INDEX)

image = bounding_box_annotator.annotate(im, detections)
image = label_annotator.annotate(image, detections)
image.thumbnail((600, 600))
image
```

Using this information, I decided to test an untrained classification of three powerful LVMs to the VLM4Bio dataset (https://huggingface.co/datasets/sammarfy/VLM4Bio), which was researched last week. The three models included Openclip (which our application is currently using), Bioclip (which was trained on a different species dataset), and Florence-2. I did the work here using on a Google Colab instance.



```
[ ] from transformers import AutoModelForCausalLM, AutoProcessor, AutoTokenizer
    import torch
    import open_clip
    from PIL import Image

    DEVICE = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

```
[ ] !git lfs clone https://huggingface.co/datasets/sammarfy/VLM4Bio VLM4BIO_data
```

```
[ ] !cp -r drive/MyDrive/OMSCS/VLM4Bio .
    !rm -R VLM4Bio/datasets
    !mv datasets VLM4Bio/
```

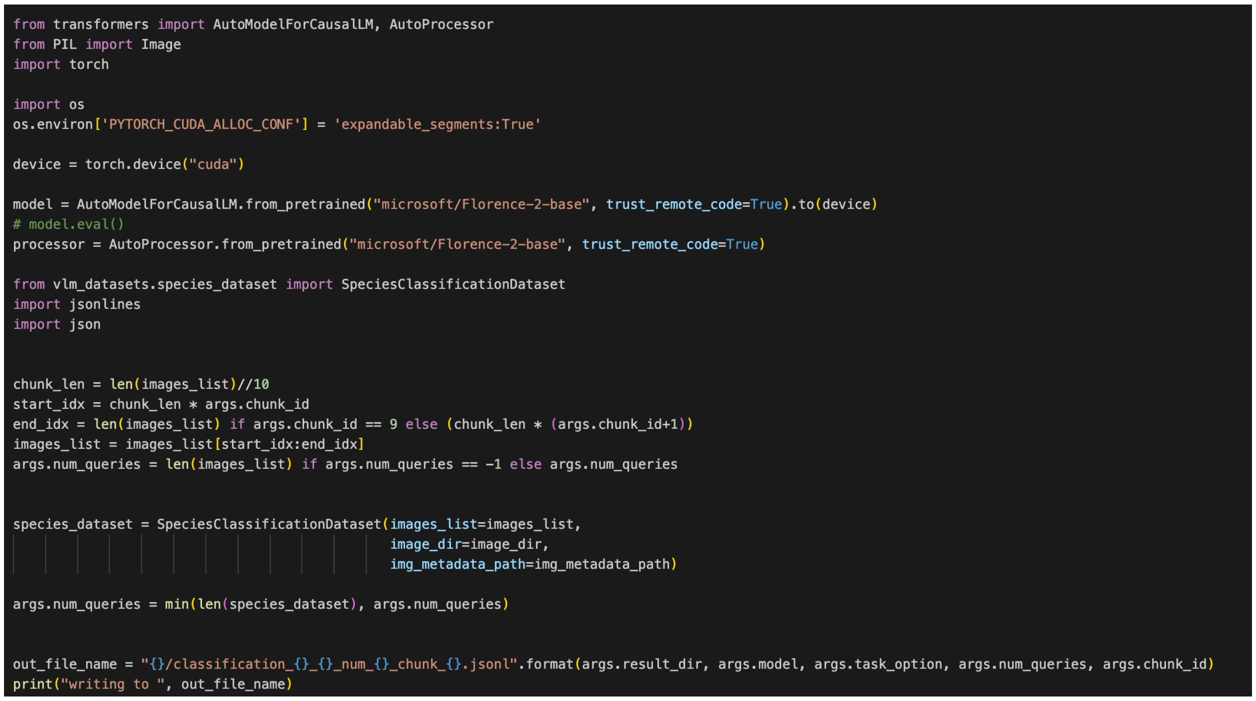
```
[ ] # make this a part of the process above
    !mkdir VLM4Bio/datasets/Fish/images
    !mkdir VLM4Bio/datasets/Bird/images
    !mkdir VLM4Bio/datasets/Butterfly/images
    !cp VLM4Bio/datasets/Fish/chunk_*/** VLM4Bio/datasets/Fish/images/
    !cp VLM4Bio/datasets/Bird/chunk_*/** VLM4Bio/datasets/Bird/images/
    !cp VLM4Bio/datasets/Butterfly/chunk_*/** VLM4Bio/datasets/Butterfly/images/
```

```
[ ] !python VLM4Bio/zero_shot_eval/bioclip_classification.py -o ''
```

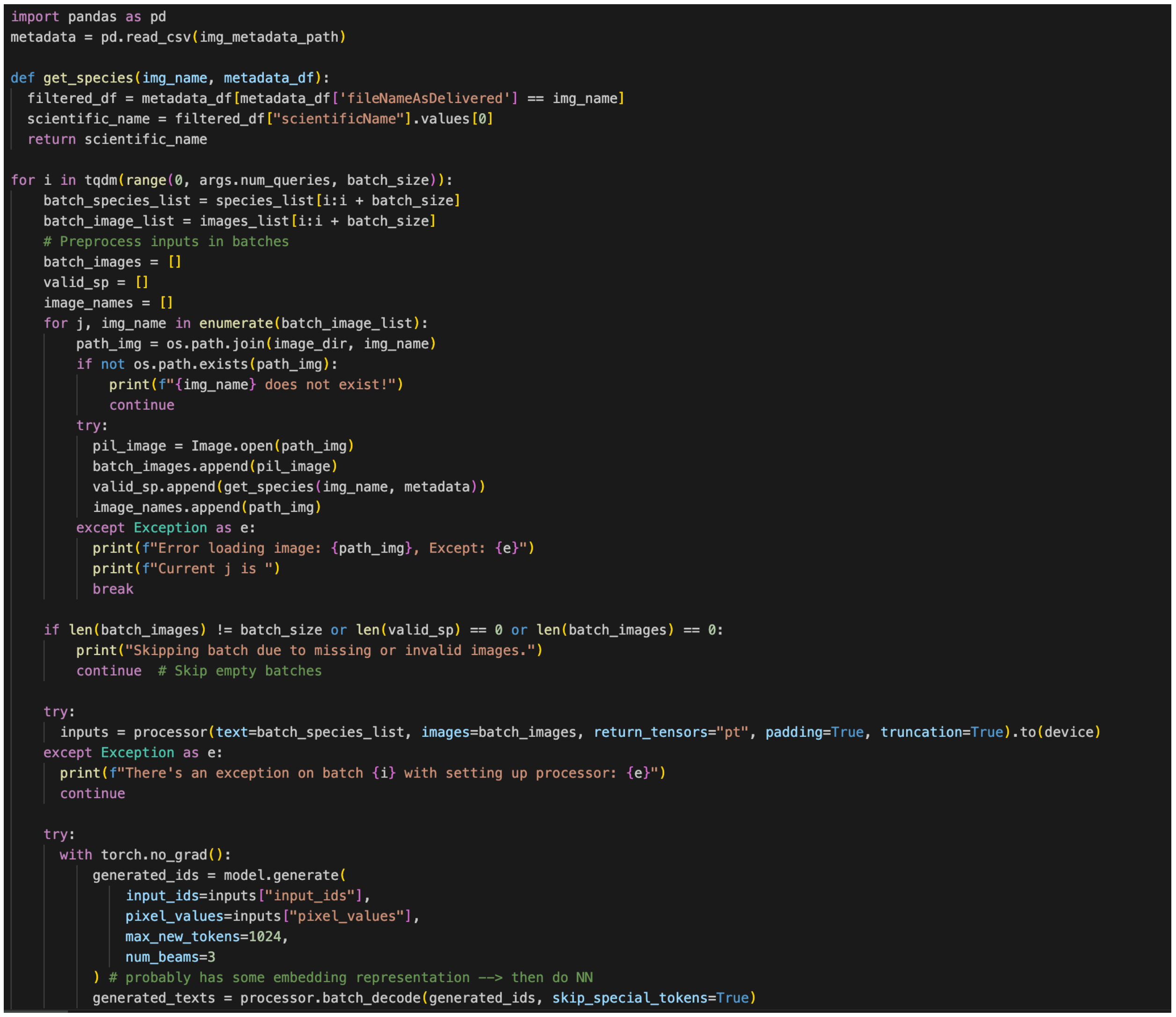
```
!python VLM4Bio/zero_shot_eval/florence_classification.py -o ''
```

```
2024-09-13 17:28:45.908519: E external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:8454] U
2024-09-13 17:28:45.914603: E external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1452]
2024-09-13 17:28:47.256744: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Wa
config.json: 100% 2.43k/2.43k [00:00<00:00, 14.5MB/s]
configuration_florence2.py: 100% 15.1k/15.1k [00:00<00:00, 53.9MB/s]
A new version of the following files was downloaded from https://huggingface.co/microsoft/Flo
- configuration_florence2.py
. Make sure to double-check they do not contain any added malicious code. To avoid downloadin
modeling_florence2.py: 100% 127k/127k [00:00<00:00, 10.5MB/s]
A new version of the following files was downloaded from https://huggingface.co/microsoft/Flo
- modeling_florence2.py
. Make sure to double-check they do not contain any added malicious code. To avoid downloadin
pytorch_model.bin: 100% 464M/464M [00:02<00:00, 204MB/s]
preprocessor_config.json: 100% 806/806 [00:00<00:00, 3.84MB/s]
processing_florence2.py: 100% 46.4k/46.4k [00:00<00:00, 752kB/s]
A new version of the following files was downloaded from https://huggingface.co/microsoft/Flo
- processing_florence2.py
. Make sure to double-check they do not contain any added malicious code. To avoid downloadin
tokenizer_config.json: 100% 34.0/34.0 [00:00<00:00, 157kB/s]
vocab.json: 100% 1.10M/1.10M [00:00<00:00, 5.82MB/s]
tokenizer.json: 100% 1.36M/1.36M [00:00<00:00, 10.2MB/s]
writing to  results/fish/classification/direct/classification_florence_direct_num_1034_chunk_
 14% 18/130 [01:05<05:39,  3.03s/it]This is a friendly reminder - the current text generatior
 47% 61/130 [04:52<05:23,  4.69s/it]There's an exception on batch 488 with generation: Sizes
 48% 62/130 [04:53<04:02,  3.57s/it]There's an exception on batch 496 with setting up process
```

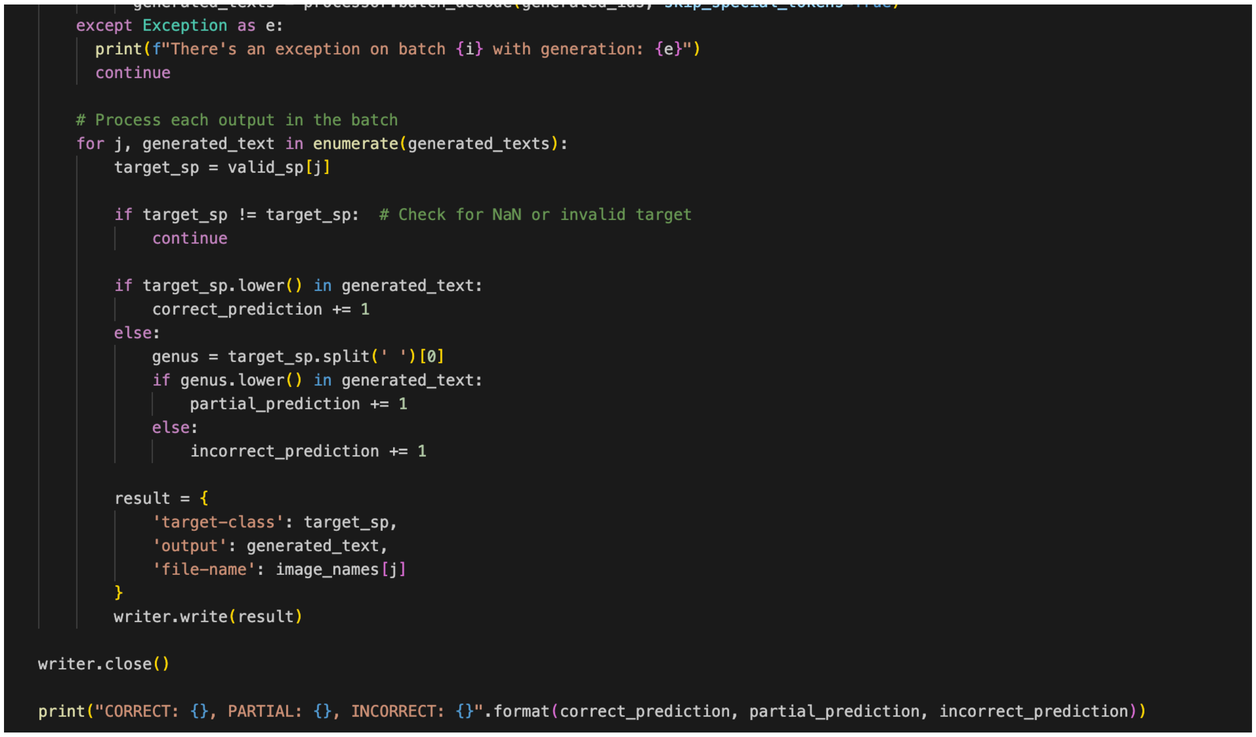
Example of Florence_classification script code:



```python
from transformers import AutoModelForCausalLM, AutoProcessor
from PIL import Image
import torch

import os
os.environ['PYTORCH_CUDA_ALLOC_CONF'] = 'expandable_segments:True'

device = torch.device("cuda")

model = AutoModelForCausalLM.from_pretrained("microsoft/Florence-2-base", trust_remote_code=True).to(device)
# model.eval()
processor = AutoProcessor.from_pretrained("microsoft/Florence-2-base", trust_remote_code=True)

from vlm_datasets.species_dataset import SpeciesClassificationDataset
import jsonlines
import json


chunk_len = len(images_list)//10
start_idx = chunk_len * args.chunk_id
end_idx = len(images_list) if args.chunk_id == 9 else (chunk_len * (args.chunk_id+1))
images_list = images_list[start_idx:end_idx]
args.num_queries = len(images_list) if args.num_queries == -1 else args.num_queries


species_dataset = SpeciesClassificationDataset(images_list=images_list,
                                               image_dir=image_dir,
                                               img_metadata_path=img_metadata_path)

args.num_queries = min(len(species_dataset), args.num_queries)


out_file_name = "{}/classification_{}_{}_num_{}_chunk_{}.jsonl".format(args.result_dir, args.model, args.task_option, args.num_queries, args.chunk_id)
print("writing to ", out_file_name)
```



```python
import pandas as pd
metadata = pd.read_csv(img_metadata_path)

def get_species(img_name, metadata_df):
  filtered_df = metadata_df[metadata_df['fileNameAsDelivered'] == img_name]
  scientific_name = filtered_df["scientificName"].values[0]
  return scientific_name

for i in tqdm(range(0, args.num_queries, batch_size)):
    batch_species_list = species_list[i:i + batch_size]
    batch_image_list = images_list[i:i + batch_size]
    # Preprocess inputs in batches
    batch_images = []
    valid_sp = []
    image_names = []
    for j, img_name in enumerate(batch_image_list):
        path_img = os.path.join(image_dir, img_name)
        if not os.path.exists(path_img):
            print(f"{img_name} does not exist!")
            continue
        try:
          pil_image = Image.open(path_img)
          batch_images.append(pil_image)
          valid_sp.append(get_species(img_name, metadata))
          image_names.append(path_img)
        except Exception as e:
          print(f"Error loading image: {path_img}, Except: {e}")
          print(f"Current j is ")
          break

    if len(batch_images) != batch_size or len(valid_sp) == 0 or len(batch_images) == 0:
        print("Skipping batch due to missing or invalid images.")
        continue  # Skip empty batches

    try:
      inputs = processor(text=batch_species_list, images=batch_images, return_tensors="pt", padding=True, truncation=True).to(device)
    except Exception as e:
      print(f"There's an exception on batch {i} with setting up processor: {e}")
      continue

    try:
      with torch.no_grad():
          generated_ids = model.generate(
              input_ids=inputs["input_ids"],
              pixel_values=inputs["pixel_values"],
              max_new_tokens=1024,
              num_beams=3
          ) # probably has some embedding representation --> then do NN
          generated_texts = processor.batch_decode(generated_ids, skip_special_tokens=True)
```



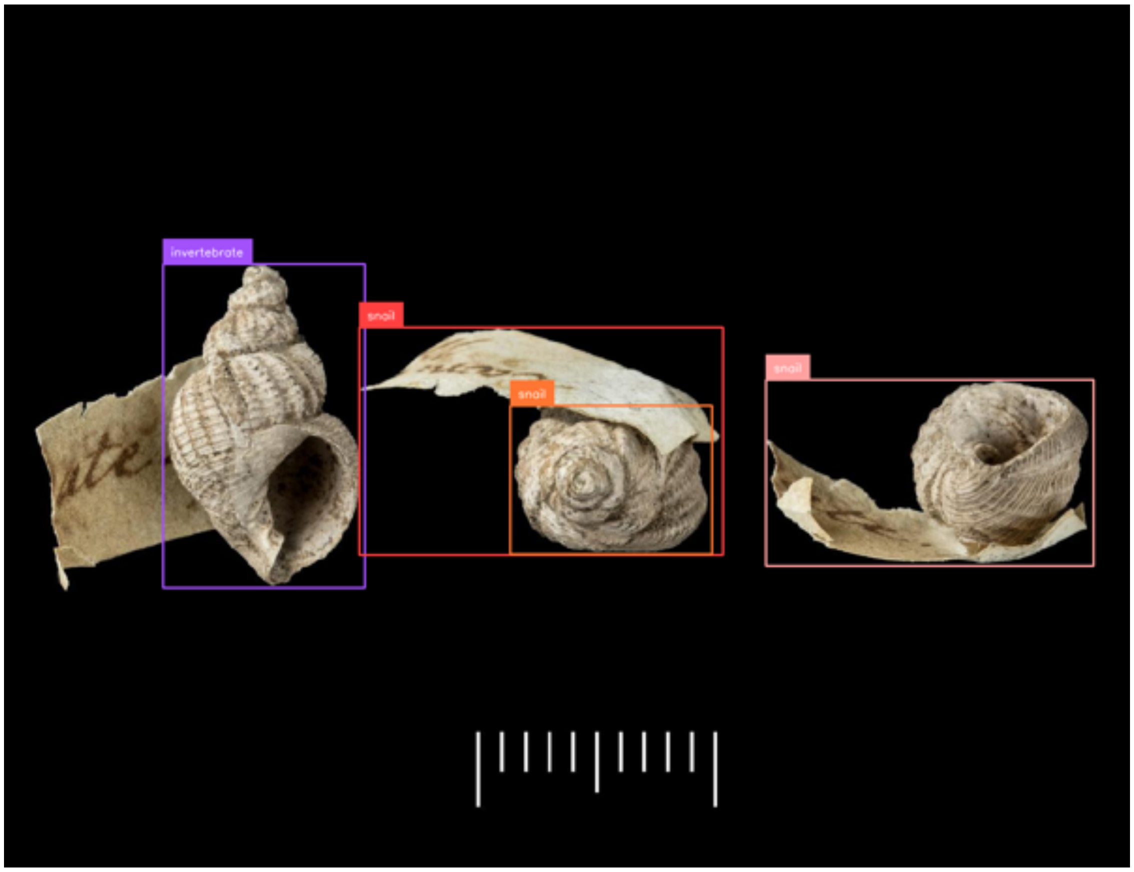
```
          generated_texts = processor.batch_decode(generated_ids, skip_special_tokens=True)
    except Exception as e:
      print(f"There's an exception on batch {i} with generation: {e}")
      continue

    # Process each output in the batch
    for j, generated_text in enumerate(generated_texts):
        target_sp = valid_sp[j]

        if target_sp != target_sp:  # Check for NaN or invalid target
            continue

        if target_sp.lower() in generated_text:
            correct_prediction += 1
        else:
            genus = target_sp.split(' ')[0]
            if genus.lower() in generated_text:
                partial_prediction += 1
            else:
                incorrect_prediction += 1

        result = {
            'target-class': target_sp,
            'output': generated_text,
            'file-name': image_names[j]
        }
        writer.write(result)

writer.close()

print("CORRECT: {}, PARTIAL: {}, INCORRECT: {}".format(correct_prediction, partial_prediction, incorrect_prediction))
```

(also did the similar for bioclip + openclip)

3. Documentation

No documentation has been added, the scripts have so far only been used to test the models, these will be uploaded to the Biocosmos repo.

4. Proof of work



Toggle output scrolling

	model	organism	full_matches	partial_matches	none_matches
0	openclip	fish	5	17	1011
1	openclip	bird	12	95	1002
2	openclip	butterfly	1	601	399
3	bioclip	fish	15	83	935
4	bioclip	bird	661	16	432
5	bioclip	butterfly	26	613	362
6	florence	fish	8	22	457
7	florence	bird	0	0	184
8	florence	butterfly	0	0	56

5. Next Week Proposal

Met up with the team, Dr. Porto and Moritz to discuss the previous week's updates on ML tasks, and discussed future work items. To be done in coming week:

- Study Florence-2 paper in depth
- Attempt to run scripts on PACE
- Continue investigation in Florence-2 species classification scripts with VLM4Bio dataset
- Figure out the way embeddings are generated by the model (generated_ids)
- check difference on classification results by training on some of the data