Week 10 Report

Thomas Deatherage
NFHM/BioCosmos
25 October 2024

Time Slot

1) What progress did you make in the last week?

- Weekly catchup with Roman.
- Thorough deep dive into the UNICOM paper https://arxiv.org/pdf/2304.05884
    - Discovered a few inconsistencies/ambiguities in our understanding of the paper during catch up with Roman.
- Wrote the captioning script(s) to auto-generate captions for eventual training dataset. Currently works on the VLM4Bio dataset.

2) What are you planning on working on next?

- Resolve misunderstandings in the UNICOM paper
- Get working on building the training step

3) Is anything blocking you from getting work done?

- Nope.

Abstracts & Summaries

Lit review replaced by journal club:
https://humanaugmente-e7j6563.slack.com/archives/C07R7SK1VQU/p1729801794441169

Scripts and Code Blocks

I'll reproduce the most relevant code below, with tedious parts omitted. The basic idea is we pull data from a local store of VLM4Bio, preprocess the images per InternVL's requirements, grab scientific names, and then feed this into InternVL2. InternVL2 then generates an (ideally) high-quality one sentence caption that includes the species name explicitly.

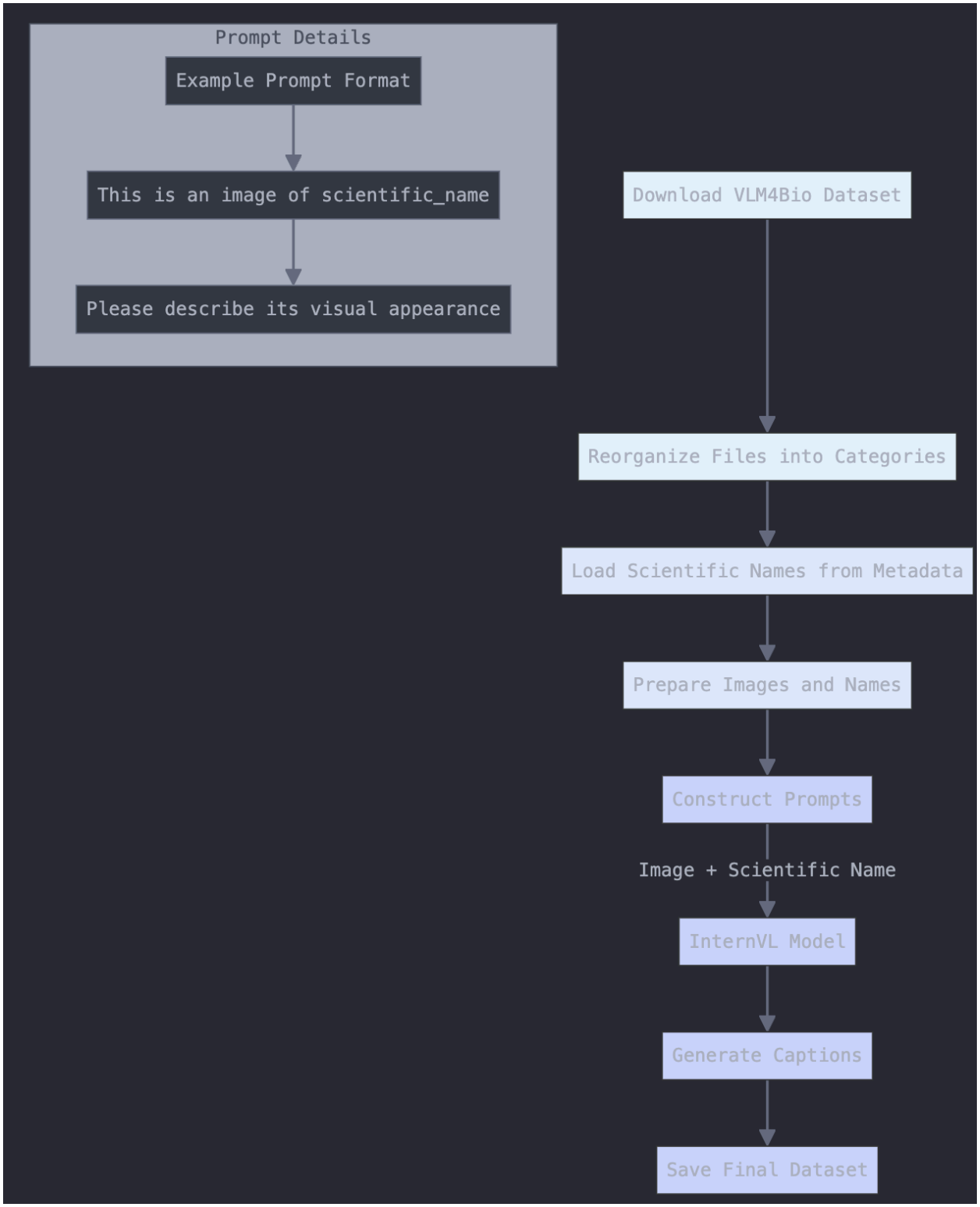
```
class CaptionGenerator:
    def __init__(self, model_name="OpenGVLab/InternVL2-2B", device="cuda"):
        self.device = device
        self.model = AutoModel.from_pretrained(
            model_name,
            torch_dtype=torch.bfloat16,
            low_cpu_mem_usage=True,
            use_flash_attn=True,
            trust_remote_code=True
        ).eval().to(device)
        self.tokenizer = AutoTokenizer.from_pretrained(model_name,
trust_remote_code=True)

        self.transform = self.build_transform(input_size=448)

    def build_transform(self, input_size):
        mean = (0.485, 0.456, 0.406)
        std = (0.229, 0.224, 0.225)
        return T.Compose([
            T.Lambda(lambda img: img.convert('RGB') if img.mode != 'RGB'
else img),
            T.Resize((input_size, input_size),
interpolation=InterpolationMode.BICUBIC),
            T.ToTensor(),
            T.Normalize(mean=mean, std=std)
        ])

    def find_closest_aspect_ratio(self, aspect_ratio, target_ratios, width,
height, image_size):
        best_ratio_diff = float('inf')
        best_ratio = (1, 1)
        area = width * height
        for ratio in target_ratios:
            target_aspect_ratio = ratio[0] / ratio[1]
            ratio_diff = abs(aspect_ratio - target_aspect_ratio)
            if ratio_diff < best_ratio_diff:
                best_ratio_diff = ratio_diff
                best_ratio = ratio
            elif ratio_diff == best_ratio_diff and area > 0.5 * image_size
* image_size * ratio[0] * ratio[1]:
                best_ratio = ratio
        return best_ratio
```

```
    def dynamic_preprocess(self, image, min_num=1, max_num=12,
image_size=448, use_thumbnail=True):
        orig_width, orig_height = image.size
        aspect_ratio = orig_width / orig_height

        # A bunch of stuff to make the image square in accord with the
model's input requirements per InternVL2's documentation
        # . . . . .
        return processed_images

    def process_image(self, image_path):
        image = Image.open(image_path).convert('RGB')
        processed_images = self.dynamic_preprocess(image)
        pixel_values = torch.stack([self.transform(img) for img in
processed_images])
        return pixel_values.to(torch.bfloat16).to(self.device)

    def generate_caption(self, image_path, scientific_name=None):
        pixel_values = self.process_image(image_path)

        if scientific_name:
            prompt = f"<image>\nThis is an image of {scientific_name}.
Please provide a detailed one-sentence caption describing the visual
appearance of this specimen."
        else:
            prompt = "<image>\nPlease provide a detailed one-sentence
caption describing what you see in this image."

        generation_config = dict(max_new_tokens=100, do_sample=False)
        response = self.model.chat(
            self.tokenizer,
            pixel_values,
            prompt,
            generation_config
        )
        return response

def load_metadata(category_dir):
    """Load and validate metadata from the CSV file."""
    metadata_file = os.path.join(category_dir, 'metadata',
'metadata_10k.csv')
    try:
        df = pd.read_csv(metadata_file)
```

```
        return {row['fileNameAsDelivered']: row['scientificName']
                for _, row in df.iterrows() if
pd.notna(row['fileNameAsDelivered'])}
    except Exception as e:
        logging.error(f"Error loading metadata from {metadata_file}:
{str(e)}")
        return {}

def process_vlm4bio_dataset(data_dir, output_path, subset=['Bird', 'Fish',
'Butterfly']):
    generator = CaptionGenerator()
    results = []
    error_log = []

    for category in subset:
        category_dir = os.path.join(data_dir, 'datasets', category)
        images_dir = os.path.join(category_dir, 'images')

        if not os.path.exists(images_dir):
            logging.error(f"Images directory not found: {images_dir}")
            continue

        # Load metadata
        metadata_dict = load_metadata(category_dir)
        if not metadata_dict:
            logging.warning(f"No metadata found for category: {category}")

        # Get list of actual images
        actual_images = set(f for f in os.listdir(images_dir)
                            if f.lower().endswith(('.jpg', '.jpeg', '.png')))

        referenced_images = set(metadata_dict.keys())
        missing_images = referenced_images - actual_images
        unlisted_images = actual_images - referenced_images

        # Process each existing image
        for img_name in tqdm(actual_images, desc=f"Processing {category}"):
            image_path = os.path.join(images_dir, img_name)
            scientific_name = metadata_dict.get(img_name)

            try:
                caption = generator.generate_caption(image_path,
scientific_name)
```

```
                    results.append({
                        'category': category,
                        'image_name': img_name,
                        'scientific_name': scientific_name,
                        'caption': caption,
                        'image_path': image_path,
                        'has_metadata': img_name in metadata_dict
                    })

                except Exception as e:
                    # Log the error as you might expect
                if len(results) % 100 == 0:
                    save_results(results, error_log, output_path)

    save_results(results, error_log, output_path)
    logging.info(f"Processing completed. Results saved to {output_path}")

def save_results(results, error_log, output_path):
    """Save results and error log to files."""
    # Save main results
    df = pd.DataFrame(results)
    df.to_csv(output_path, index=False)

    # Save error log
    error_df = pd.DataFrame(error_log)
    error_path = output_path.replace('.csv', '_errors.csv')
    error_df.to_csv(error_path, index=False)

if __name__ == "__main__":
    # This was set up with some previous scripts.  Don't worry about the
details.
    data_dir = "data/VLM4Bio"
    output_path = "vlm4bio_captions.csv"

    process_vlm4bio_dataset(data_dir, output_path)
```

Flow Charts/Diagrams



Documentation

No documentation to add really just yet

Results Visualization + Proof of Work

Here are some example outputs of captions and their respective images. From a small, informal sampling, the results look promising.

This image depicts a Vireo flavifrons, a bright yellow bird perched on a branch against a clear blue sky.

The image depicts a small bird, Passerculus sandwichensis, perched on a metal pipe against a clear blue sky

This image depicts a bird, specifically a male of the species Contopus sordidulus, perched on a branch with a blurred blue

That last one is interesting, as the metadata made no indication as to the sex of the individual. InternVL included that on its own volition. This may be something that needs to be suppressed with a prompt-tweak, as I don't really trust the model to accurately predict that for the sundry species we're looking at.

Next Week's Proposal

- Resolve lack of clarity in UNCIOM paper
- Start training Clip.