Week 5 Report - NFHM

Romouald Dombrovski

September 20, 2024

1. Time Log

What progress did you make last week?

- Met up with Team and Dr. Porto to discuss splitting of responsibilities and re-focus on analyzing Florence-2
- Re-wrote the VLM4Bio scripts for classification, and uploaded to NFHM repo
- Reviewed and tested Thomas' updates for app startup including db versioning
- Fixed index out of range bugs for Florence-2 classification task
- Added Cosine similarity to Florence-2 classification (results not great)

2. Abstract

Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. 2022. BEiT: BERT pre-training of image transformers. *International Conference on Learning Representations*

Abstract: We introduce a self-supervised vision representation model BEIT, which stands for Bidirectional Encoder representation from Image Transformers. Following BERT [DCLT19] developed in the natural language processing area, we propose a masked image modeling task to pretrain vision Transformers. Specifically, each image has two views in our pre-training, i.e., image patches (such as 16×16 pixels), and visual tokens (i.e., discrete tokens). We first "tokenize" the original image into visual tokens. Then we randomly mask some image patches and fed them into the backbone Transformer. The pre-training objective is to recover the original visual tokens based on the corrupted image patches. After pre-training BEIT, we directly fine-tune the model parameters on downstream tasks by appending task layers upon the pretrained encoder. Experimental results on image classification and semantic segmentation show that our model achieves competitive results with previous pre-training methods.

Summary (GPT-4o)**:** The paper presents BEIT (Bidirectional Encoder representation from Image Transformers), a self-supervised pre-training method for vision Transformers inspired by BERT's success in NLP. BEIT tackles the data-hungry nature of vision Transformers by introducing masked image modeling (MIM), where images are tokenized into discrete visual tokens using a discrete variational

autoencoder (dVAE). During pre-training, a portion of image patches is masked, and the model learns to predict the corresponding visual tokens instead of raw pixels, allowing it to focus on higher-level semantic information rather than low-level pixel details. This approach enables BEIT to effectively pre-train vision Transformers, which are later fine-tuned for tasks like image classification and semantic segmentation. The model outperforms both from-scratch training and previous self-supervised models, showing faster convergence and better stability during fine-tuning. BEIT also learns to identify semantic regions in images without human annotations, offering a scalable and efficient solution for pre-training vision models. The paper suggests that BEIT can further improve by scaling up model and data sizes and potentially integrating multimodal pre-training with shared architectures for text and images.

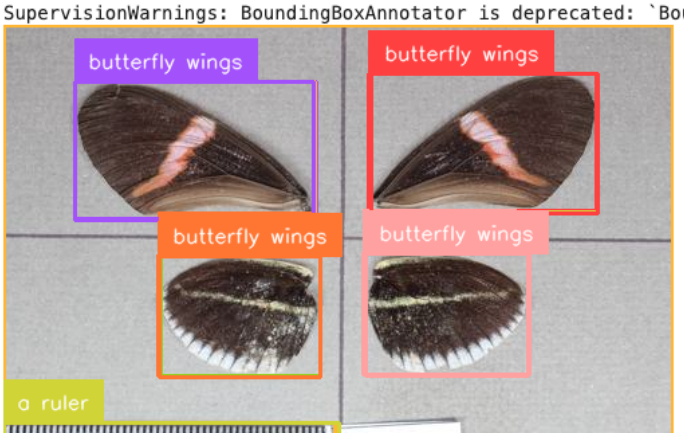
3. Scripts and Code Blocks

Work can be viewable in the following pull request: https://github.com/Human-Augment-Analytics/NFHM/pull/34

Some of the testing done for image captioning

```python
# @title Example caption to phrase grounding inference

task = "<CAPTION_TO_PHRASE_GROUNDING>"
text = "<CAPTION_TO_PHRASE_GROUNDING> butterfly wings on a table with a ruler at the bottom."

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=(image.width
detections = sv.Detections.from_lmm(sv.LMM.FLORENCE_2, response, resolution_wh=image.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(image, detections)
image = label_annotator.annotate(image, detections)
image.thumbnail((600, 600))
image
```

```
SupervisionWarnings: BoundingBoxAnnotator is deprecated: `BoundingBoxAnnotator` is deprecated and has be
```



Updates to Florence-2 classification code:

```python
for i in tqdm(range(0, args.num_queries)):
    image_name = images_list[i]
    path_img = os.path.join(image_dir, image_name)
    try:
      pil_image = Image.open(path_img)
    except Exception as e:
      print(f"Error loading image: {path_img}, Except: {e}")
      break

    try:
        inputs = processor(text='<OD>', images=pil_image, 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
    tok = processor.tokenizer(species_list, padding=True, return_tensors="pt")
    text = tok["input_ids"].to(device)

    with torch.no_grad(), torch.cuda.amp.autocast():
      image_features = model._encode_image(inputs["pixel_values"])
      text_features = model.get_input_embeddings()(text)
      image_features = image_features.mean(dim=1)
      text_features = text_features.mean(dim=1)
      image_features /= image_features.norm(dim=-1, keepdim=True)
      text_features /= text_features.norm(dim=-1, keepdim=True)
      text_probs = (100.0 * image_features @ text_features.T).softmax(dim=-1)

    ranks = np.argsort(text_probs[0].detach().cpu().numpy())[::-1]

    result = dict()

    top1_idx = ranks[:1]
    pred_sp = species_list[top1_idx[0].item()]

    top5_idx = ranks[:5]
    top5_sp = [species_list[idx] for idx in top5_idx]
    top5_score = [str(text_probs[0, idx].item()) for idx in top5_idx]
    target_sp = get_species(image_name, img_metadata)

    result['target-class'] = target_sp
    result['output'] = pred_sp
    result['top5'] = ','.join(top5_sp)
    result['top5_score'] = ','.join(top5_score)

    if target_sp == pred_sp:
      correct_prediction += 1
    else:
      genus = target_sp.split(' ')[0]
      if genus in pred_sp:
```

Made the scripts run for the entire dataset, for each species (~10000 images).
Example with Butterfly:

```
[ ] !python VLM4Bio/vlm4bio_classification.py -o '' -m 'bioclip' -d 'butterfly' -c '-1'
```

```
2024-09-20 17:35:36.391760: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:485] Unable to
2024-09-20 17:35:36.412894: E external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:8454] Unable t
2024-09-20 17:35:36.419278: E external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1452] Unable
2024-09-20 17:35:37.597216: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning:
Arguments Provided:  Namespace(model='bioclip', task_option='direct', result_dir='./results/butterfl
writing to  ./results/butterfly/classification/direct/classification_bioclip_direct_num_10013_chunk_
open_clip_pytorch_model.bin: 100% 599M/599M [00:56<00:00, 10.6MB/s]
open_clip_config.json: 100% 469/469 [00:00<00:00, 3.53MB/s]
100% 10013/10013 [05:10<00:00, 32.27it/s]
MODEL: bioclip..... CORRECT: 1400, PARTIAL: 3939, INCORRECT: 4674
```

```
!python VLM4Bio/vlm4bio_classification.py -o '' -m 'openclip' -d 'butterfly' -c '-1'

2024-09-20 17:43:08.408205: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:485] Unab
2024-09-20 17:43:08.429847: E external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:8454] Una
2024-09-20 17:43:08.436394: E external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1452] Un
2024-09-20 17:43:09.592087: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warn
Arguments Provided:  Namespace(model='openclip', task_option='direct', result_dir='./results/bu
writing to  ./results/butterfly/classification/direct/classification_openclip_direct_num_10013_
open_clip_pytorch_model.bin: 100% 605M/605M [00:01<00:00, 370MB/s]
100% 10013/10013 [05:06<00:00, 32.62it/s]
MODEL: openclip..... CORRECT: 46, PARTIAL: 4261, INCORRECT: 5706
```

```
!python VLM4Bio/vlm4bio_classification.py -o '' -m 'florence-2' -d 'butterfly' -c '-1'

2024-09-20 20:20:52.658185: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on
2024-09-20 20:20:52.675403: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:485] Unab
2024-09-20 20:20:52.697109: E external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:8454] Una
2024-09-20 20:20:52.703622: E external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1452] Un
2024-09-20 20:20:52.719605: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlo
To enable the following instructions: AVX2 AVX512F AVX512_VNNI FMA, in other operations, rebuil
2024-09-20 20:20:53.899998: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warn
Arguments Provided:  Namespace(model='florence-2', task_option='direct', result_dir='./results/
writing to  ./results/butterfly/classification/direct/classification_florence-2_direct_num_1001
config.json: 100% 2.43k/2.43k [00:00<00:00, 15.1MB/s]
configuration_florence2.py: 100% 15.1k/15.1k [00:00<00:00, 66.3MB/s]
A new version of the following files was downloaded from https://huggingface.co/microsoft/Flore
- configuration_florence2.py
. Make sure to double-check they do not contain any added malicious code. To avoid downloading
modeling_florence2.py: 100% 127k/127k [00:00<00:00, 27.6MB/s]
A new version of the following files was downloaded from https://huggingface.co/microsoft/Flore
- modeling_florence2.py
. Make sure to double-check they do not contain any added malicious code. To avoid downloading
pytorch_model.bin: 100% 464M/464M [00:01<00:00, 464MB/s]
preprocessor_config.json: 100% 806/806 [00:00<00:00, 5.84MB/s]
processing_florence2.py: 100% 46.4k/46.4k [00:00<00:00, 37.3MB/s]
A new version of the following files was downloaded from https://huggingface.co/microsoft/Flore
- processing_florence2.py
. Make sure to double-check they do not contain any added malicious code. To avoid downloading
tokenizer_config.json: 100% 34.0/34.0 [00:00<00:00, 254kB/s]
vocab.json: 100% 1.10M/1.10M [00:00<00:00, 2.23MB/s]
tokenizer.json: 100% 1.36M/1.36M [00:00<00:00, 59.7MB/s]
100% 10013/10013 [10:48<00:00, 15.44it/s]
MODEL: florence-2..... CORRECT: 25, PARTIAL: 61, INCORRECT: 9927
```

From this we can see that Florence-2 is doing no better than noise probability in determining the correct species from the image. As per Dr. Porto's suggestion, next week will be focused on working with NN approach on image embeddings.

4. Documentation

Documentation can be viewed on the jupyter notebook found on NFHM repo https://github.com/Human-Augment-Analytics/NFHM/blob/main/jupyter-workpad/vlm4_bio/vlm4bio_classification.ipynb

5. Next Week Proposal

- Investigate using generated_ids with caption text on as cosine sim.
- Change classification strategy from cosine similarity of text-image embedding to NN of image-embeddings
- Create model-eval statistics with old method (cosine sim.) vs new method (NN)