Week 8 Report - NFHM

Romouald Dombrovski

October 11, 2024

1 Time Log

What progress did you make in the last week?

- Added InternVL model eval metrics
- Ran model eval with all models
- Set up Ollama Kotaemon locally to test LLM usage
- Updated frontend to use Vite build manager
- Fixed Readme for Biocosmos, wrote one for frontend
- Refactored frontend components to Composition API
- Did preliminary research on UNICOM

2 Abstract

Li, X., Xie, L., Zhang, J., Guo, J., Yao, L. (2024). UNICOM: Universal and Compact Representation Learning for Image Retrieval. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2024.

Abstract Modern image retrieval methods typically rely on fine-tuning pre-trained encoders to extract image-level descriptors. However, the most widely used models are pre-trained on ImageNet-1K with limited classes. The pre-trained feature representation is therefore not universal enough to generalize well to the diverse open-world classes. In this paper, we first cluster the large-scale LAION 400M dataset into one million pseudo classes based on the joint textual and visual features extracted by the CLIP model. Due to the confusion of

label granularity, the automatically clustered dataset inevitably contains heavy inter-class conflict. To alleviate such conflict, we randomly select partial inter-class prototypes to construct the margin-based softmax loss. To further enhance the low-dimensional feature representation, we randomly select partial feature dimensions when calculating the similarities between embeddings and class-wise prototypes. The dual random partial selections are with respect to the class dimension and the feature dimension of the prototype matrix, making the classification conflict-robust and the feature embedding compact. Our method significantly outperforms state-of-the-art unsupervised and supervised image retrieval approaches on multiple benchmarks. The code and pre-trained models are released to facilitate future research https://github.com/deepglint/unicom.

Summary (GPT-4o) The paper "UNICOM: Universal and Compact Representation Learning for Image Retrieval" presents a new framework designed to address the challenges of large-scale image retrieval by focusing on both universal applicability and compactness of image representations. The authors introduce a novel contrastive learning approach combined with a distillation process that helps to generate highly compressed embeddings without sacrificing retrieval accuracy. UNICOM's compact representations are designed to work across diverse image datasets, ensuring strong performance without the need for task-specific fine-tuning or adjustments. The paper demonstrates that, despite its smaller embedding size, UNICOM achieves competitive or even superior results on standard image retrieval benchmarks compared to other state-of-the-art methods. This balance between retrieval accuracy and efficiency makes UNICOM well-suited for real-world scenarios where computational and storage resources are limited, such as mobile or cloud-based applications. Overall, UNICOM offers a scalable and practical solution for universal image retrieval tasks, with a focus on reducing redundancy while maintaining retrieval precision.

3 Scripts and Code Blocks

The Model eval script, including all models can be found within the Zero Shot script here:
`https://github.com/BioCosmos-AI/BioCosmos/blob/main/jupyter-workpad/vlm4_bio/Zero_Shot_VLM4Bio.ipynb`
Here are the additions of InternVL:

```
elif model_name == 'internvl':
  model = AutoModel.from_pretrained(
      'OpenGVLab/InternViT-300M-448px',
      torch_dtype=torch.bfloat16,
      low_cpu_mem_usage=True,
      trust_remote_code=True).cuda().eval()

  processing = CLIPImageProcessor.from_pretrained('OpenGVLab/InternViT-300M-448px')
```

```python
def get_image_embeddings_intern(image_filename, model, preprocess):
    image_folder = "downloaded_images"

    try:
        image_path = os.path.join(image_folder, image_filename)
        image = Image.open(image_path).convert('RGB')
        pixel_values = preprocess(images=image, return_tensors='pt').pixel_values

        with torch.no_grad():
            pixel_values = pixel_values.to(torch.bfloat16).cuda()
            image_embeddings = model(pixel_values).pooler_output
        return image_embeddings.to(torch.float32).cpu().numpy()
    except Exception as e:
        print(f"Error processing image {image_path}: {e}")
        return None  # Return None for failed image processing
```

The Vue refactor pull request can be found here:
`https://github.com/Human-Augment-Analytics/NFHM/pull/38`

Vite Config:

13 ■■■■■ frontend/vite.config.js

```
...  ...   @@ -0,0 +1,13 @@
          1  + import { defineConfig } from 'vite'
          2  + import vue from '@vitejs/plugin-vue'
          3  +
          4  + // https://vitejs.dev/config/
          5  + export default defineConfig({
          6  +   plugins: [vue()],
          7  +   server: {
          8  +     port: 3000,
          9  +   },
         10  +   build: {
         11  +     outDir: 'dist',
         12  +   },
         13  + });
```

Here are some examples of the rewritten components:

Search Component:

123 ■■■■ frontend/src/components/SearchComponent.vue

```
@@ -0,0 +1,123 @@
+ <template>
+   <div class="heading">
+     <div class="inp-button">
+       <p>
+         Search with a natural language query or upload an image to perform a reverse image search.
+       </p>
+       <div class="inputs">
+         <label id="magglass">
+           <input
+             type="text"
+             placeholder="Search"
+             name="searchQuery"
+             id="searchQuery"
+             class="search-query input input-bordered input-primary"
+             v-model="inputQuery"
+           />
+         </label>
+
+         <label for="fileUpload" class="custom-file-upload">
+           <img
+
  src="
  GywssGyByY5yXEBvg+OuhEbvUZFABcJKW9ASSlwLk3QBM5ARhjFoCJ43jeGQCsq3kYhjNJ16JW0rETANgAb0n7Ig+CYCLpVJv5bStAFEVL4Fkx2kk61JdK0gtYNQIkSTIFbg02917+x39MUVtA78fOdwTJrOd61GD6AJGcB5JWDDmgAAAAAE
+           />
+         </label>
+         <input
+           type="file"
+           class="hidden"
+           name="fileUpload"
+           id="fileUpload"
+           @change="uploadFileChanged"
+           ref="fileUploadElement"
+           accept="image/*"
+         />
+       </div>
+       <div class="buttons">
+         <button class="btn btn-primary" @click="submitQuery">
+           Search
+         </button>
+       </div>
+     </div>
+   </div>
+ </template>
```

123 ■■■■■ frontend/src/components/SearchComponent.vue

```
+ <script>
+ import { ref } from 'vue';
+ import axios from 'axios';
+ import Swal from 'sweetalert2';
+
+ export default {
+   name: 'SearchComponent',
+   setup(props, { emit }) {
+     // Define reactive state using refs
+     const inputQuery = ref('');
+     const fileQuery = ref('');
+     const fileUploadElement = ref(null);
+
+     // Methods
+     const submitQuery = () => {
+       const bodyFormData = new FormData();
+       bodyFormData.append('search_param', inputQuery.value);
+       queryAPI(bodyFormData);
+     };
+
+     const uploadFileChanged = (event) => {
+       const files = event.target.files;
+       if (files.length > 0) {
+         const bodyFormData = new FormData();
+         bodyFormData.append('image', files[0]);
+         queryAPI(bodyFormData);
+       }
+     };
+
+     const queryAPI = (formData) => {
+       const endpoint = window.location.origin + "/api/search";
+       console.log('calling endpoint: ' + endpoint)
+       axios({
+         method: 'post',
+         url: endpoint,
+         data: formData,
+         headers: { 'Content-Type': 'multipart/form-data' },
+       })
+         .then((response) => {
+           if (response.data?.records) {
+             emit('search-result', response.data);
+           } else {
+             emit('search-result', []);
+           }
+         })
+         .catch((error) => {
```

```
+
+     const searchQueryChanged = (event) => {
+       inputQuery.value = event.target.value;
+       if (fileQuery.value !== '') {
+         fileQuery.value = '';
+         if (fileUploadElement.value) {
+           fileUploadElement.value.value = null;
+         }
+       }
+     };
+
+     return {
+       inputQuery,
+       fileQuery,
+       fileUploadElement,
+       submitQuery,
+       uploadFileChanged,
+       queryAPI,
+       searchQueryChanged,
+     };
+   },
+ };
+ </script>
+
```

Item Dialog:

80 frontend/src/components/ItemDialog.vue

```
@@ -0,0 +1,80 @@
+ <template>
+   <h2 class="text-primary">{{ clickedItem.name }}</h2>
+   <div class="view-image">
+     <span @click="closeFocus">X</span>
+     <div class="image-content">
+       <div class="text-content">
+         <div class="details">
+           <h2 class="text-primary">{{ clickedItem.name }}</h2>
+           <p>{{ clickedItem.description }}</p>
+         </div>
+         <div class="image-attributes">
+           <div class="image-attribute">
+             <div class="key">Source</div>
+             <div class="value">{{ clickedItem.source }}</div>
+           </div>
+           <div class="image-attribute">
+             <div class="key">Global ID</div>
+             <div class="value">{{ clickedItem.id }}</div>
+           </div>
+           <div class="image-attribute">
+             <div class="key">Location</div>
+             <div class="value link">
+               <a :href="clickedItem.map_url" target="_blank">{{ clickedItem.latitude }}, {{ clickedItem.longitude }}</a>
+               </div>
+             </div>
+           <div class="image-attribute">
+             <div class="key">Source Link</div>
+             <div class="value link">
+               <a :href="clickedItem.source_link" target="_blank">{{ clickedItem.source }}</a>
+             </div>
+           </div>
+           <div class="image-attribute">
+             <div class="key">Image Link</div>
+             <div class="value link">
+               <a :href="clickedItem.media_url" target="_blank">{{ clickedItem.image_source_name }}</a>
+             </div>
+           </div>
+         </div>
+       </div>
+       <div class="image-src">
+         <img alt="" :src="clickedItem.media_url" @error="handleImageError" />
+       </div>
+     </div>
+   </div>
+ </template>
```

```
+
+ <script>
+ import { defineComponent, toRefs } from 'vue';
+
+ export default defineComponent({
+   name: 'ItemDialog',
+   props: {
+     clickedItem: {
+       type: Object,
+       required: true
+     }
+   },
+   setup(props, { emit }) {
+     const { clickedItem } = toRefs(props);
+
+     const closeFocus = () => {
+       emit('close-focus');
+     };
+
+     const handleImageError = (event) => {
+       event.target.src = 'static/unavailable-image.jpg';
+     };
+
+     return {
+       clickedItem,
+       closeFocus,
+       handleImageError
+     };
+   }
+ });
+ </script>
```

Results Display:

78 ■■■■■ frontend/src/components/ResultsDisplay.vue

```
@@ -0,0 +1,78 @@
+ <template>
+   <div class="box" v-if="display">
+     <div class="img-column" v-for="(column, index) in resultColumns" :key="index">
+       <template v-for="result in column" :key="result.id">
+         <img
+           :src="result.image_source_name !== 'NA' ? result.media_url : 'static/unavailable-image.jpg'"
+           @error="handleImageError"
+           class="img-fluid"
+           :id="result.id"
+           @click="displayImage(result)"
+         />
+       </template>
+     </div>
+   </div>
+ </template>
+
```

78 frontend/src/components/ResultsDisplay.vue

```
+ <script>
+ import { defineComponent, ref, watch } from 'vue';
+ import Result from '../models/Result.js';
+
+ export default defineComponent({
+   name: 'ResultsDisplay',
+   props: {
+     apiResult: {
+       type: Object,
+       required: true
+     }
+   },
+   setup(props, { emit }) {
+     // Reactive state
+     const resultColumns = ref([[], [], [], []]);
+     const display = ref(false);
+
+     // Watcher for `apiResult` prop
+     watch(
+       () => props.apiResult,
+       async (newValue) => {
+         resultColumns.value = [[], [], [], []]; // Reset columns
+         if (newValue.length !== 0) {
+           await addImages(newValue);
+         }
+       }
+     );
+
+     // Methods
+     const displayImage = (value) => {
+       emit('selected-result', value);
+     };
+
+     const addImages = async (apiResult) => {
+       resultColumns.value = [[], [], [], []];
+       let columnNumber = 0;
+       for (let index in apiResult.records) {
+         columnNumber = index % 4;
+         const item = new Result(apiResult.records[index]);
+         resultColumns.value[columnNumber].push(item);
+       }
+       display.value = true;
+     };
+
+     const handleImageError = (event) => {
+       event.target.src = 'static/unavailable-image.jpg';
+     };
+
+     return {
+       resultColumns,
+       display,
+       displayImage,
+       addImages,
```

App.vue:

86 ■■■■■ frontend/src/App.vue

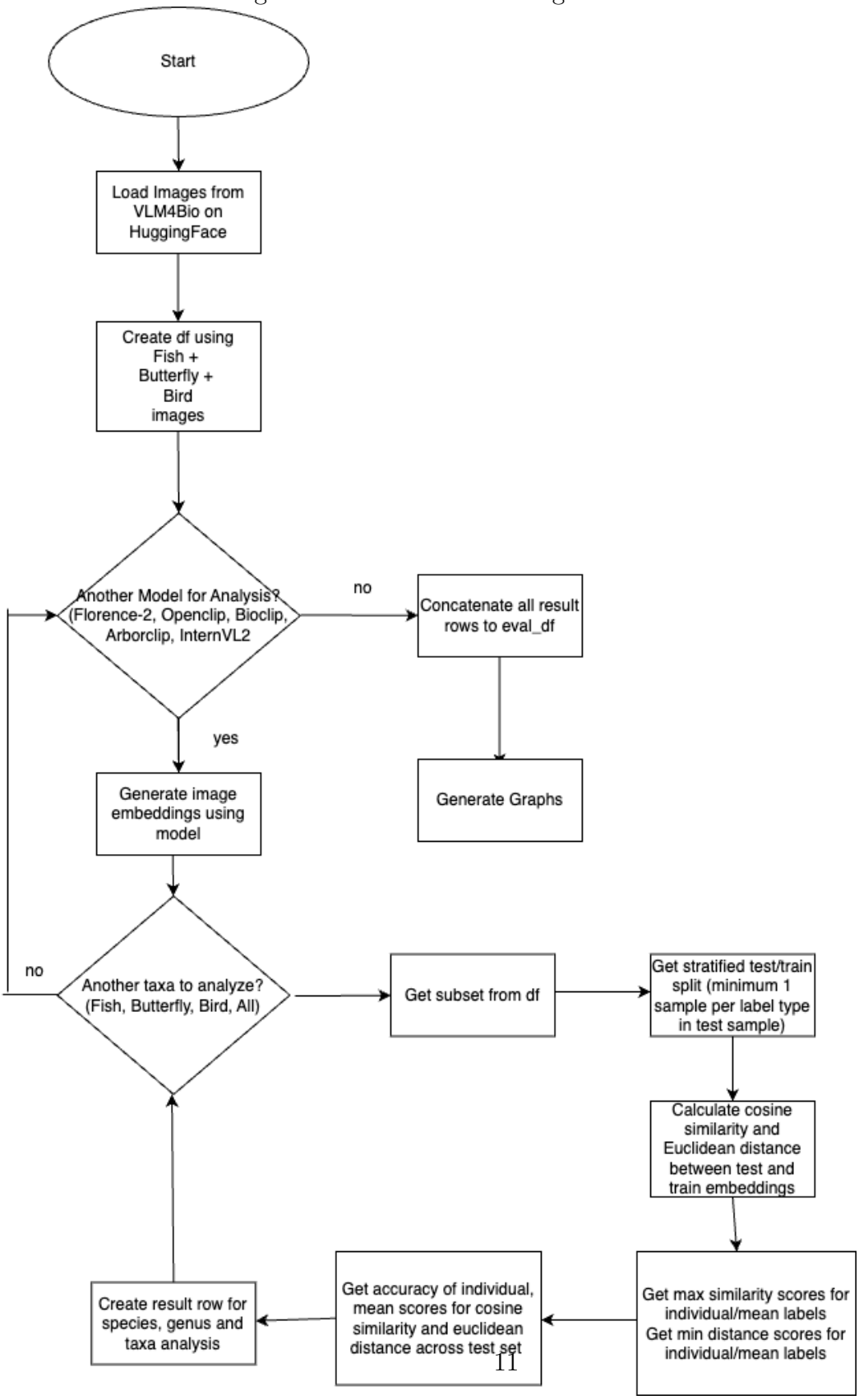
```
...  ...   @@ -0,0 +1,86 @@
      1  + <template>
      2  +   <div>
      3  +     <div class="bannerimage"></div>
      4  +     <div class="container">
      5  +       <search-component @search-result="onSearchResult" />
      6  +       <results-display
      7  +         :api-result="apiSearchResults"
      8  +         @selected-result="onSelectResult"
      9  +       />
     10  +       <item-dialog
     11  +         v-if="focusImage"
     12  +         :clicked-item="selected"
     13  +         @close-focus="onCloseFocus"
     14  +       />
     15  +     </div>
     16  +   </div>
     17  + </template>
     18  +
```

86 ■■■■■ frontend/src/App.vue

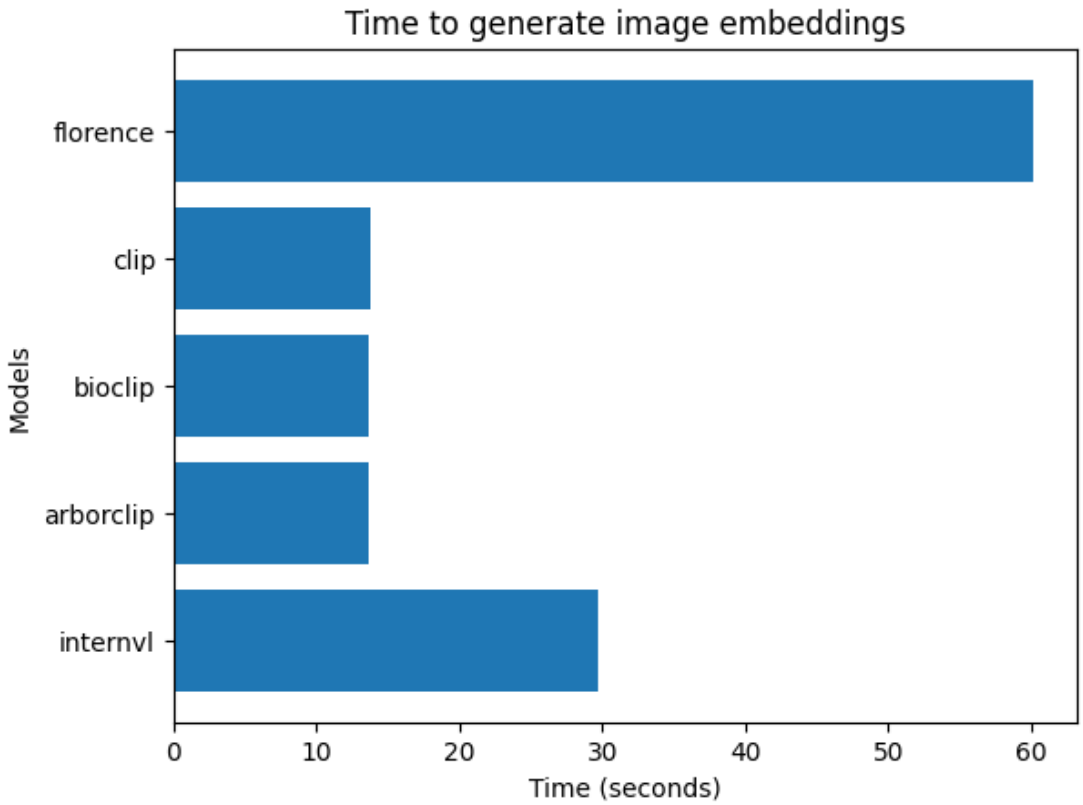
```
+ <script>
+ import { defineComponent, ref, onMounted, onUnmounted } from 'vue';
+ import ItemDialog from './components/ItemDialog.vue';
+ import ResultsDisplay from './components/ResultsDisplay.vue';
+ import SearchComponent from './components/SearchComponent.vue';
+
+ export default defineComponent({
+   name: 'App',
+   components: {
+     ItemDialog,
+     ResultsDisplay,
+     SearchComponent,
+   },
+   setup() {
+     // Define reactive state using ref()
+     const apiSearchResults = ref([]);  // API search results
+     const focusImage = ref(false);     // Show or hide ItemDialog
+     const selected = ref({});          // Selected item for ItemDialog
+
+     // Methods
+     const onSearchResult = (results) => {
+       apiSearchResults.value = results;
+     };
+
+     const onCloseFocus = () => {
+       focusImage.value = false;
+     };
+
+     const onSelectResult = (result) => {
+       focusImage.value = true;
+       selected.value = result;
+     };
+
+     const closeFocus = () => {
+       focusImage.value = false;
+     };
+
+     const keyDownHandler = (event) => {
+       if (event.key === 'Escape' && focusImage.value) {
+         focusImage.value = false;
+       }
+     };
+
+     // Lifecycle hooks for key event listener
+     onMounted(() => {
+       window.addEventListener('keydown', keyDownHandler);
+     });
+
+     onUnmounted(() => {
+       window.removeEventListener('keydown', keyDownHandler);
+     });
+
+     return {
```

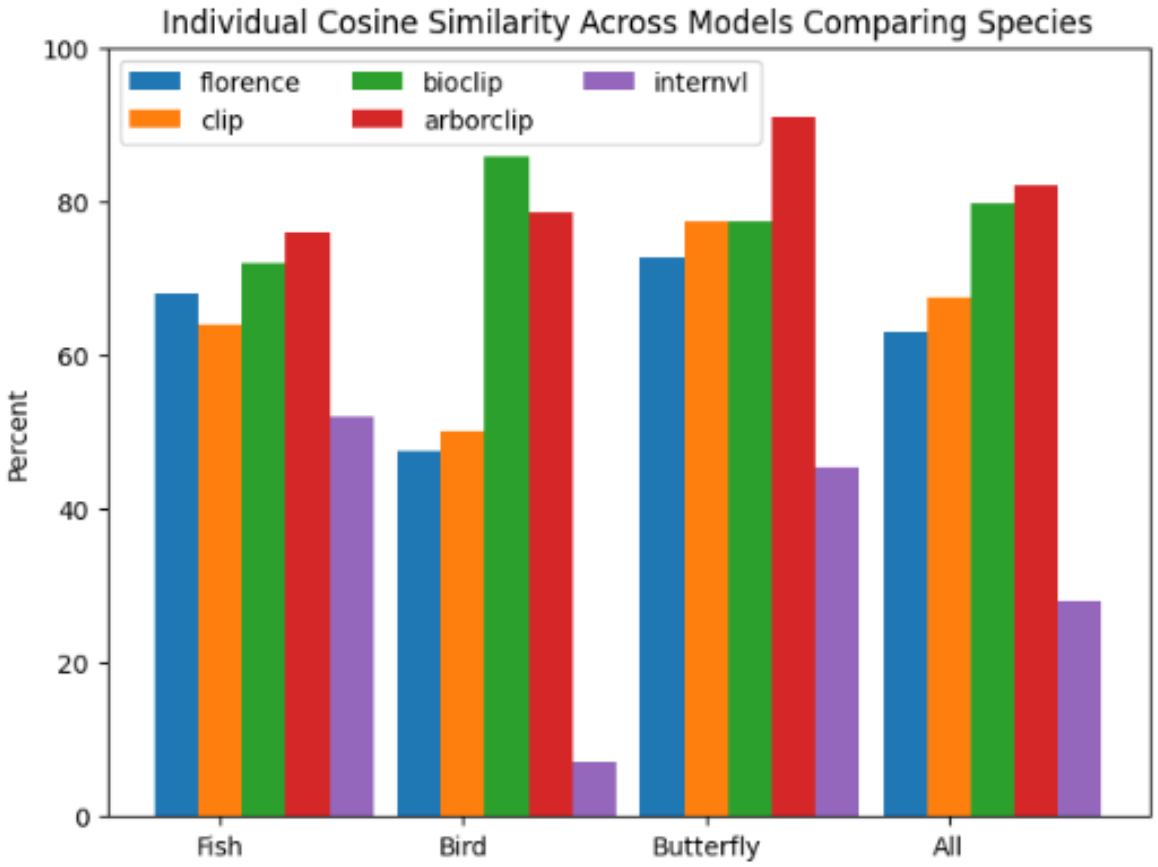
4 Visualization

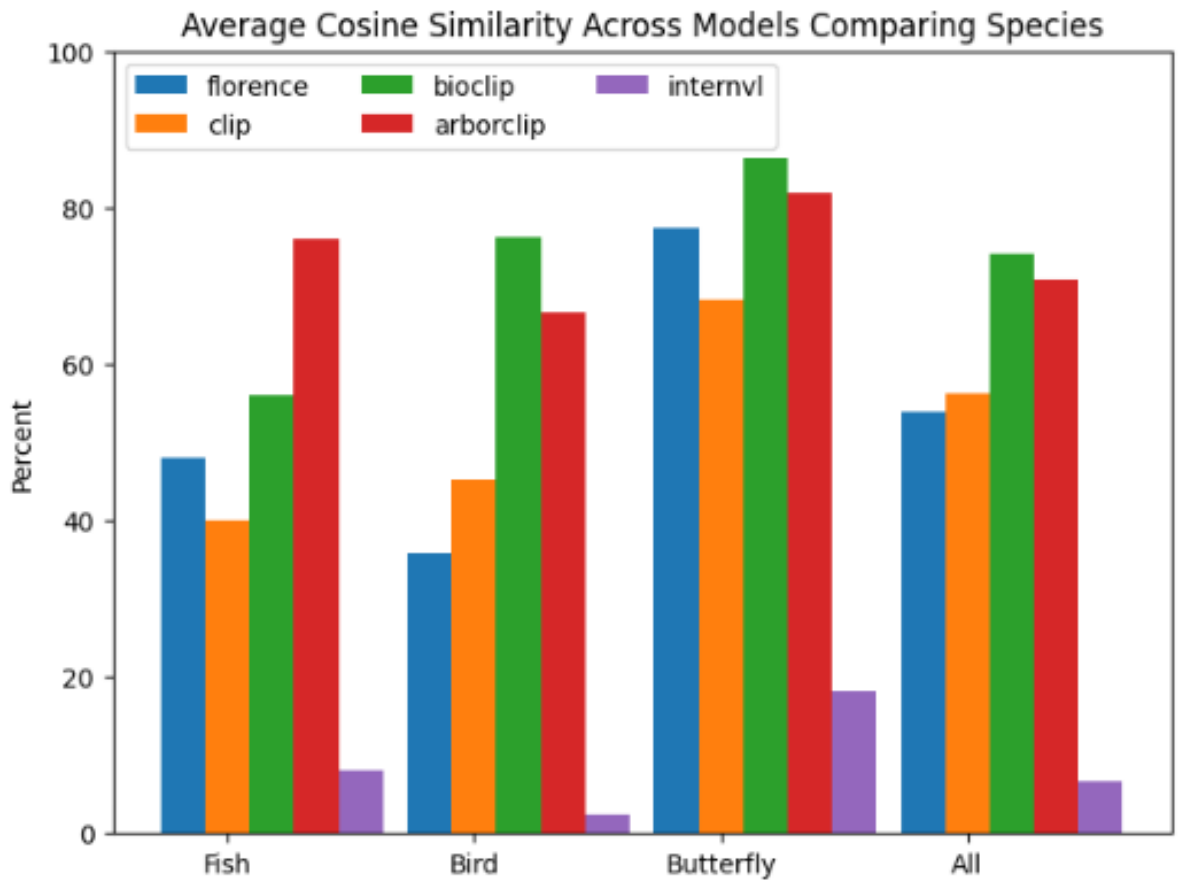
Here is a flow describing the method of conducting the Zero Shot Model Eval:

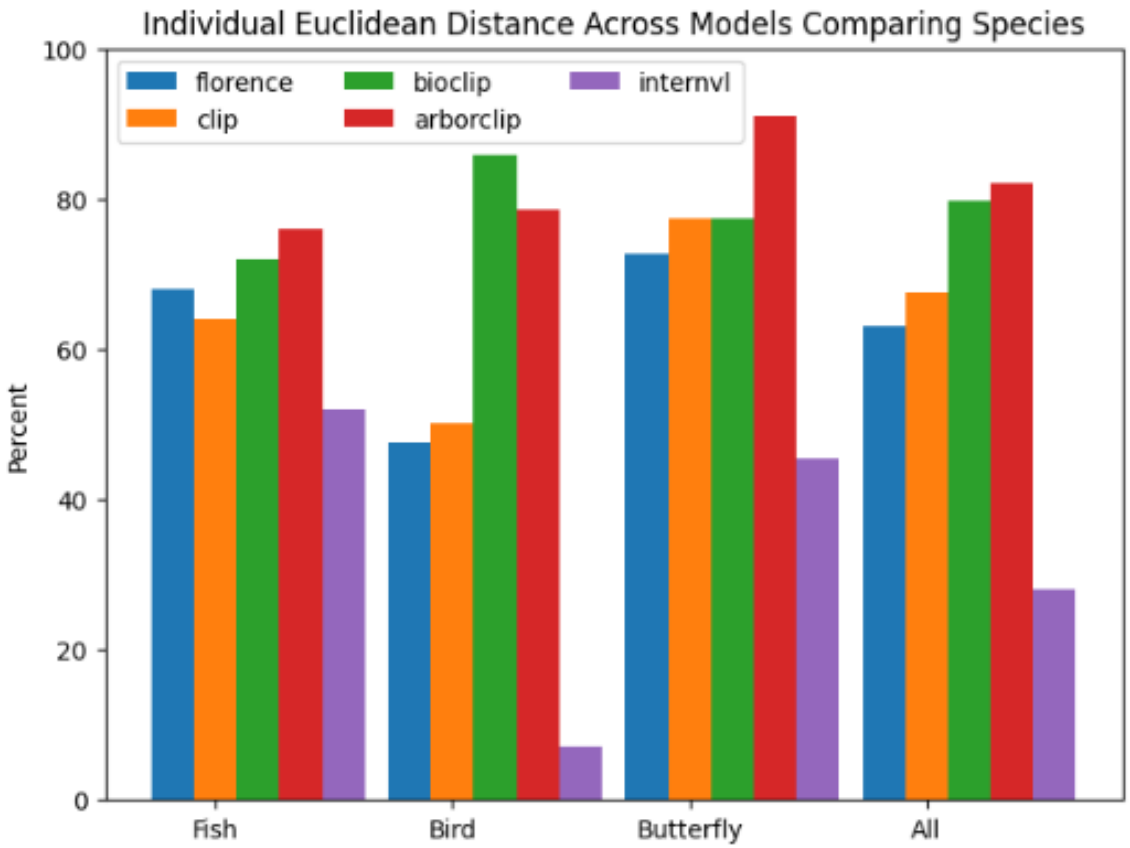


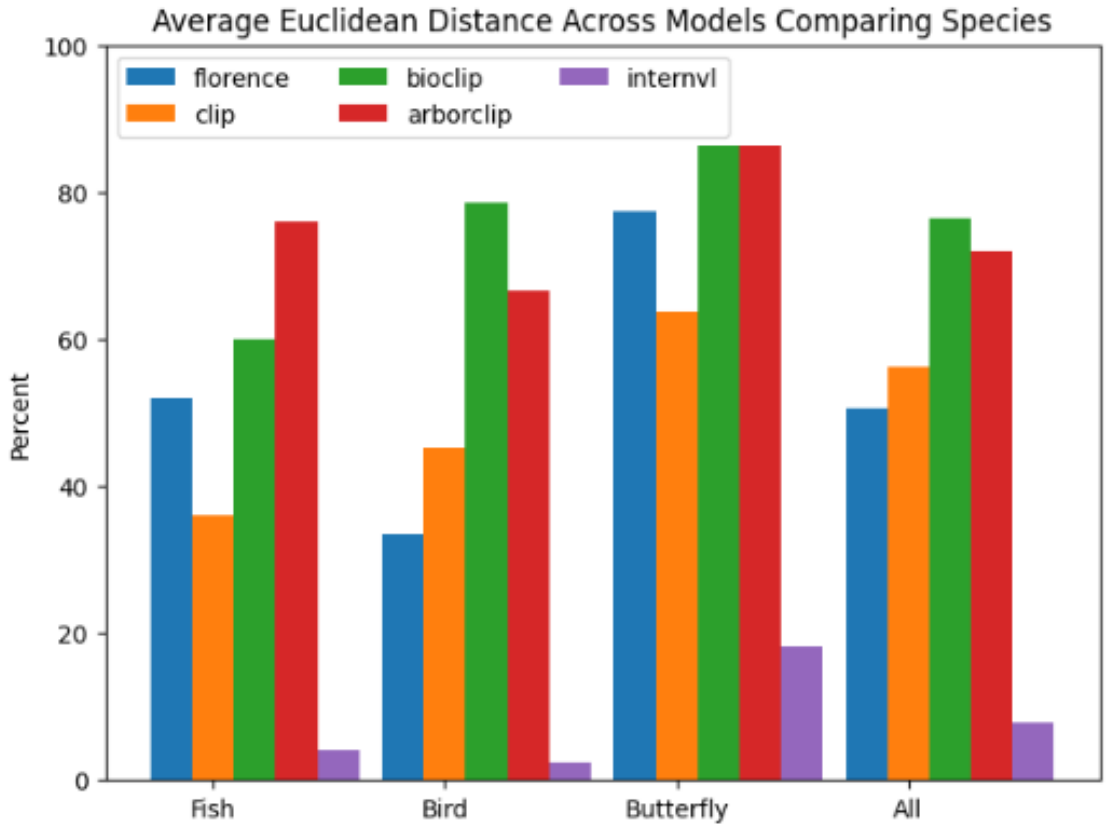
Here are the results of the Zero Shot Model Eval, including InternVL:




Individual Cosine Similarity Across Models Comparing Species
florence
clip
bioclip
arborclip
internvl
Percent
100
80
60
40
20
0
Fish
Bird
Butterfly
All


Average Cosine Similarity Across Models Comparing Species
florence
clip
bioclip
arborclip
internvl
Percent
100
80
60
40
20
0
Fish
Bird
Butterfly
All


Individual Euclidean Distance Across Models Comparing Species
florence
clip
bioclip
arborclip
internvl
Percent
100
80
60
40
20
0
Fish
Bird
Butterfly
All



5 Next Week Proposal

- Write questions relevant to NFHM on writing-dev channel
- Test image retrieval using UNICOM on VLM4Bio dataset
- Test knn using UNICOM on VLM4Bio dataset
- Investigate patch pooling on zero shot script