Week 15 Report - NFHM

Romouald Dombrovski

November 29, 2024

1 Time Log

What progress did you make in the last week?

- Reviewed SLURM scripting
- Wrote script to download VLM4Bio dataset onto Hypergator
- Rewrote Bioclip zero-shot script to work with visual transformer models
- Ran zero-shot topk analysis on ViT-H-14 generated embeddings
- Met up with Thomas to plan next steps

2 Abstract

Caron M., et al. Emerging properties in self-supervised vision transformers. In Proceedings of the IEEE International Conference on Computer Vision, pages 9650–9660, 2021.

Abstract In this paper, we question if self-supervised learning provides new properties to Vision Transformer (ViT) that stand out compared to convolutional networks (convnets). Beyond the fact that adapting self-supervised methods to this architecture works particularly well, we make the following observations: first, self-supervised ViT features contain explicit information about the semantic segmentation of an image, which does not emerge as clearly with supervised ViTs, nor with convnets. Second, these features are also excellent k-NN classifiers, reaching 78.3% top-1 on ImageNet with a small ViT. Our study also underlines the importance of momentum encoder, multi-crop training, and the use of small patches with ViTs. We implement our findings into a simple self-supervised method, called DINO, which we interpret as a form of self-distillation with no labels. We show the synergy between DINO and ViTs by achieving 80.1% top-1 on ImageNet in linear evaluation with ViT-Base.

Summary (GPT-4o) The paper "Emerging Properties in Self-Supervised Vision Transformers" explores the capabilities of Vision Transformers (ViTs) trained without supervision, revealing properties not found in supervised ViTs or convolutional networks (convnets). It highlights that self-supervised ViTs learn semantic segmentation directly in their self-attention maps, enabling unsupervised object boundary discovery. The authors introduce DINO, a self-supervised method that combines a momentum encoder and multi-crop training, achieving impressive results such as 80.1% top-1 accuracy on ImageNet linear evaluation with ViT-Base. The framework demonstrates outstanding performance with nearest-neighbor classifiers, robustness to feature extraction, and transferability to downstream tasks. This research suggests self-supervised learning as a pivotal step toward developing BERT-like models for vision tasks, with promising applications in segmentation and retrieval without the need for labeled data.

3 Scripts and Code Blocks

This week the goal was to focus on 2 fronts: first was to make progress on getting the Tree-of-Life dataset onto SLURM, and second was to work on the evaluation script we will used for our trained model. All file changes can be seen in this pull request: https://github.com/BioCosmos-AI/BioCosmos/pull/11/files

For the first goal, I created a SLURM script to download the VLM4Bio dataset. I did this as a test as this dataset was far smaller than the Tree of Life dataset.

```
#!/bin/bash
#SBATCH --job-name=download_vlm4bio
#SBATCH --output=download_vlm4bio.log
#SBATCH --error=download_vlm4bio.err
#SBATCH --time=02:00:00
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --mem=8G
#SBATCH --partition=hpg-dev

# `scontrol show partition` to show types of partition on hypergator

# load module
module load git

# get VLM4Bio dataset
mkdir /blue/arthur.porto-biocosmos/rdombrovski3.gatech/VLM4Bio
git clone https://huggingface.co/datasets/imageomics/VLM4Bio
    /blue/arthur.porto-biocosmos/rdombrovski3.gatech/VLM4Bio/

if [ $? -ne 0 ]; then
    echo "Git clone failed. Exiting."
    exit 1
fi

# check if script exists, then run scripts
first_script_path =
    "/blue/arthur.porto-biocosmos/rdombrovski3.gatech/VLM4Bio/download_bird_images.sh"
if [ -f "$first_script_path" ]; then
    # Run the shell scripts to download + process images as per documentation on
        VLM4Bio page
    bash
        /blue/arthur.porto-biocosmos/rdombrovski3.gatech/VLM4Bio/download_bird_images.sh
    bash
        /blue/arthur.porto-biocosmos/rdombrovski3.gatech/VLM4Bio/process_fish_butterfly_images.
else
```

```
    echo "script $first_script_path does not exist"
    exit 1
fi
```

Having

```
import torch
import torch.nn.functional as F
from sklearn.model_selection import train_test_split


# TODO: these are not correct right now
from ..vlm4bio.cluster_utils import load_df


def zero_shot_accuracy(test_loader, class_prototypes, topk=(1, 3, 5)):
    """
    Method to compute accuracy for top k when comparing test embeddings to class
       prototypes
    Parameters
    ----------
    test_loader : torch.utils.data.DataLoader
       DataLoader for test dataset.
    class_prototypes : Dict[str: torch.tensor]
       Class prototypes of the .
    topk : tuple, optional
       Tuple detailing which topk analysis is to be conducted. The default is
          (1, 3, 5).
    Returns
    -------
    accuracies : Dict[int: float]
       Accuracies for ks in topk of test embeddings to class prototype
          embeddings.
    """

    # assume the prototype embeddings exist in a stack, with the index of the
       prototype being the same as its class idx
    prototype_embeddings =
       torch.stack(list(class_prototypes.values())).squeeze(1) # Shape:
       [num_classes, embedding_dim]

    correct = {k: 0 for k in topk}
    total = 0

    with torch.no_grad():
       for embeddings, labels in test_loader:
          # Compute similarity between test embeddings and class prototypes

          similarities = F.cosine_similarity(embeddings.unsqueeze(1),
              prototype_embeddings.unsqueeze(0), dim=-1) # Shape [batch_size,
              num_classes]

          # Get top-k predictions
          _, predictions = similarities.topk(max(topk), dim=-1) # Shape:
              [batch_size, max(topk)]

          # set predictions for each top k (1, 3, 5)
          for k in topk:
              correct[k] += (predictions[:, :k] ==
                  labels.unsqueeze(1)).any(dim=1).sum().item()

          total += labels.size(0)

    # Compute accuracy for each k
    accuracies = {k: correct[k] / total for k in topk}
    return accuracies


def get_test_train_split(df_cleaned, min_count_per_class=10, test_size=0.1):
    """
    Modified get_test_train_split function. Always stratified, get a minimum
       count of each class.
    Parameters
    ----------
    df_cleaned : pandas Dataframe
       DESCRIPTION.
    min_count_per_class: int
       minimum number of instances (images) for each class. The default is 10.
    test_size : float, optional
       ratio size for the test split. The default is 0.1.
    Returns
    -------
    train_df : pandas Dataframe
       the training split.
    test_df : pandas Dataframe
       the test split.
    """

    species_counts = df_cleaned['scientific_name'].value_counts()
    valid_species = species_counts[species_counts > min_count_per_class].index
    df_filtered = df_cleaned[df_cleaned['scientific_name'].isin(valid_species)]
    calc_min = len(valid_species)/len(df_filtered)
    min_split = max(test_size, calc_min)

    train_df, test_df = train_test_split(df_filtered, test_size=min_split,
        stratify=df_filtered['scientific_name'])
    return train_df, test_df

def get_class_prototypes(train_split):
    """
    Given the train split dataframe, create the class prototypes by getting mean
        embeddings
    Parameters
    ----------
    train_split : pandas Dataframe
        The train split for the data, used to create the class prototypes.
    Returns
    -------
    class_prototypes : Dict[str: torch.tensor]
        DESCRIPTION.
    """
    train_split["embedding_tensor"] =
        train_split["image_embeddings"].apply(torch.tensor)
    class_prototypes = (
        train_split.groupby("scientific_name")["embedding_tensor"]
        .apply(lambda x: torch.stack(list(x.squeeze(0))).mean(dim=0))
        .to_dict()
    )
    return class_prototypes


class EmbeddingsDataset(torch.utils.data.Dataset):
    """
    Mostly a redo of previous datasets. This is assuming the dataframe already
        contains the image embeddings
    """
    def __init__(self, df, cls_to_idx, transform=None):
        self.df = df
        self.transform = transform
        self.cls_to_idx = cls_to_idx

    def __len__(self):
        return len(self.df)

    def __getitem__(self, idx):
        # Load image and apply transformations
        row = self.df.iloc[idx]
        image_embeddings = row['image_embeddings']
        cls = row['scientific_name']
        # image embeddings are size [1, embed_dim] --> squeeze first dim
        return image_embeddings.squeeze(0), self.cls_to_idx[cls]


if __name__ == "__main__":
    # get existing df with image embeddings
    embeds_df = load_df('ViT-H-14-embeddings')

    # split data to train and test
    train_split, test_split = get_test_train_split(embeds_df)

    # get prototypes from training splits
    class_prototypes = get_class_prototypes(train_split)

    prototype_labels = list(class_prototypes.keys())
    prototype_to_idx = {cls: i for i, cls in enumerate(prototype_labels)}

    dataset = EmbeddingsDataset(test_split, prototype_to_idx)

    test_loader = torch.utils.data.DataLoader(dataset, batch_size=32,
        shuffle=False)

    topk_accuracies = zero_shot_accuracy(test_loader, class_prototypes,
        prototype_to_idx, topk=(1, 3, 5))
    print(f"Top-k Accuracies: {topk_accuracies}")
```

I used the previous embeddings generated by the OpenAI CLIP ViT-H-14 model (which I stored in a .mat file) to test this script. I split the dataset into a 90:10 train:test split, and took the average of the image embeddings per class (scientific_name) to create class prototypes for each individual species. I calculated the cosine similarity between the test set and the class prototypes and obtained the topk accuracies for k=1, 3, 5. The results I got for this model were: 1: 0.6284135240572172, 3: 0.8403771131339401, 5: 0.9044213263979194

4 Next Week Proposal

- Get Tree-of-Life 10M dataset downloaded onto blue storage on Hypergator
- Run topk visual model zero shot on other model types
- Create presentation + present for end of semester HAAG event
- Fill out milestone report for NFHM project for Vy