Later Image Classification Model Report

Github Repo: https://github.com/jingyig16/later-northeastern-model

Tilak Patel: patel.til@northeastern.edu
Aryavir Goel: goel.ary@northeastern.edu
Jingy Gong: gong.jing@northeastern.edu
Jiayu Zhong: zhong.jiay@northeastern.edu

Executive Summary

The objective of this report is to examine and analyze an image classification model that clusters and labels images using multiple open-source pre-trained models, that can be scalable for large sets of images as well as images provided from a singular user interface. The model aims to cluster images based on similar semantic features in addition to labeling each cluster, thereby providing a category for a set of images that have similarities. The model also aims to dynamically cluster images depending on how large the dataset is as well as dynamically label each cluster. This means that when a new image that is drastically different enters the model, it will then create a separate cluster and label for that image.

This model does not solve a problem for the client, but rather it expands the software offered to the client's customers by providing their users with a package that allows them to organize their images into meaningful labels/categories. The objective of the client is to provide diverse packages of organizational methods for social media activity to its customers and by integrating this model into its software, the client can expand its offering and thereby drive greater growth and profitability for the future of its company by improving or rather expanding its user functionality.

Introduction

The model's main purpose and objective is to be a cost-effective functional proof of concept for the client. The model does not have to be fully complete or accurate, but rather it should show promising results so that it can be handed over to the client for further development, similar to a turnkey project. The model's input is a set of images, and the output is specific labeled clusters for images that share similar semantic features.

The client's current industry is software development, which is a fast-growing and highly competitive market. With the rapid development of AI and machine learning, the software development industry is estimated to grow from \$203.35B to \$1450.87B by the end of 2031. For companies to differentiate themselves in the market, they will have to provide creative and innovative technological solutions that solve modern-day problems or make some processes more efficient. That's what this image classification model hopes to achieve for the client, by expanding their existing package to provide users with a more diverse and user-friendly platform for social media management.

The model is not fully perfect however and does have limitations that need to be understood. The model does not label images individually but rather dynamically creates clusters and labels based on distinct images. The limitation concerning individual labeling is due to the pre-trained models used which were all cost-effective open-source models, that aligns with the client's objective of utilizing a cost-effective model.

Model Overview

The model itself utilizes 4 different open-source pre-trained models. The first and most important model is CLIP. The CLIP model is the foundation for image classification and is a neural network developed by OpenAI, in the context of our model, it embeds the unsupervised images and creates feature vectors for all the images. CLIP takes the image as an input and creates a vector based on the semantic features of the image, allowing for a very specific image embedding that will give us the foundation for the clustering process. In addition, CLIP has its own preprocessing mechanism before taking in each image from a specific dataset. With these steps, we can capture specific features within an image which can then be used for clustering and labeling later.

The second pre-trained model used is BLIP, which is an open-source visual language model. This model is essential for labeling and the architecture used for BLIP is directly related to the model's **text encoder.** The text encoder architecture of BLIP is a transformer based that generates text and works well with CLIP by using CLIPS shared embedding space or the embeddings generated from CLIP to create a label. Combining this with another language model can allow for a specific and accurate categorization of images that share similar features.

The third essential resource used is an open-source library called FAISS. FAISS' main feature is vector indexing. This means that it can store data as vectors, which are developed through CLIP, and can efficiently index and search these high-dimensional vectors, making it possible to match similar vectors with one another based on cosine similarity. This allows the main model to efficiently search and store vectors that are like one another, allowing for a scalable approach to clustering and categorization.

Finally, the dataset used for this model was raw images without labels with URL access. This made it incredibly more difficult to build the model, but after careful consideration and analysis, the model carries out the steps required to generate meaningful labels through the workings of several pre-trained open-source models that generate feature vectors and from those vectors generate labels or meaningful categories. Each model mentioned above has its own preprocessing mechanism

Implementation

After gaining access to the dataset, we worked on researching open-source cost-efficient models that would allow us to work with unsupervised images. Developing the feature vectors or embeddings was the primary step in the implementation process and that's where CLIP comes in. CLIP has its own preprocessing mechanism, so that was the first step in the implementation process. After preprocessing the images for CLIP, we were able to generate embeddings or feature vectors for an image dataset of around 300 images.

Next, after these embeddings were generated, the next step in the implementation process was to search and store these embeddings. That's where FAISS' vector indexing feature comes into play. With FAISS, we can search and store these embeddings. Why FAISS in this instance? The client placed a major emphasis on an efficient, cost-effective model while being scalable together. FAISS not only allows for an efficient search and store algorithm, but it also can be scalable to large datasets, and small ones all the way down to a single user. This is essential for the client when integrating and scaling it into their software.

After the image embeddings have been stored using FAISS, the clustering process begins. The model used for clustering is K-means, which is a popular and efficient clustering model that works well with unsupervised datasets, or in this case, image embeddings. Once the image embeddings have been generated, they serve as the input into the K-means algorithm. Each embedding is treated as a point in high-dimensional space. K-means begins by initializing k centroids randomly in the embeddings space which represent the centers of the clusters. In addition, the model uses a hierarchical clustering process that selects the k values, and K means clusters the images or essentially fine tunes the clusters based on the k centroid. For the model, it has been coded in a way to dynamically cluster the images based on how large the dataset is. After this process, the points or embeddings are assigned to a centroid based on a distance metric, and once all the embeddings have been assigned, the centroids stabilize, establishing clusters.

The next step in the implementation process was concerned with somehow generating labels from the model. Ultimately, the client needs an efficient way to categorize user images, hence the model has solved the first problem of grouping images with similar features together. If the model can provide labels for the groups of similar images, it will considerably help the client in scaling and integrating the model with their software in the future, or at the least, serve as a proof of concept that this type of model can be functional. Hence, BLIP combined with an LLM text model, Mistral-7B, has been used in the model to derive labels for each of the clusters. BLIP takes each image, or cluster centroid image and generates a caption for it. After this process, these captions are essentially handed over to an LLM which can be changed according to considerations for larger datasets, and the LLM such as GPT (more expensive, but very specific) generates detailed or specific descriptions for each cluster, completing the model output.

Preliminary Results

The model was tested on a dataset of 300 unsupervised photos to assess its ability to cluster and classify images based on semantic similarities. The pipeline featured essential phases such as CLIP preprocessing, embedding generation, FAISS indexing, K-means clustering and hierarchical clustering, and cluster labeling using BLIP coupled with a big language model.

A. Clustering Process and Output

CLIP embeddings were indexed by FAISS and then grouped using the K-means algorithm. This technique dynamically clustered photos based on their semantic properties. The number of clusters was determined dynamically using the dataset's unpredictability and complexity.

Following the clustering process, labels for each cluster were created with BLIP and a big language model. Each cluster was given a label that summarized the common semantic qualities of the photos in that group. For example, a cluster with photographs of household animals was tagged "Cats."

B. Storage and Organization

The clustered photos were grouped into folders based on their labels for a clear and methodical output structure. This folder-based organization improves user accessibility and the model's practical usefulness. This aligns directly with the clients' need for an efficient and specific approach to categorizing user images uploaded to the company's database.

Limitation & Mitigation

A key limitation of the model was the need to use AWS to scale the model for larger datasets. While the basic code was written in Python, switching to AWS required additional configuration and modification. However, with limited experience of the team in cloud computing services, unfortunately, no results were produced. Also, the code could not be directly copied and executed on AWS, necessitating changes and learning that slowed deployment. This limitation highlights the significance of AWS knowledge for future scaling.

The team tested on two AWS instances: r6i.xlarge and g4dn.xlarge. R6i.xlarge had no GPU power and was very slow to produce image descriptions when running Mistral-7B on the instance, taking more than 5 minutes to produce one description for an image. G4dn.xlarge, though powered by NVIDIA T4 GPUs, experienced persistent memory limitations (“CUDA out of memory” error) running CLIP, BLIP, as well as Mistral-7B LLM. The model needs a high-powered GPU instance or cloud computing power in order to run the transformer models and LLM with larger datasets. The current model was able to run on a local laptop with limited resources, but a higher computing power would benefit the efficiency and overall performance of the model.

Another problem was the availability of several non-functional image URLs in the original dataset, which slowed the project's progress in its early phases. However, this problem was quickly fixed by forwarding it to the sponsor, who immediately resolved it by giving updated and functional URLs.

Future Improvement

Resolving the AWS integration issues is an essential step in the project. Addressing technical issues by broadening the team's experience with AWS setup and configurations would allow the model to scale efficiently for larger datasets and facilitate smoother deployment processes. Furthermore, using open-source models as benchmarks can give a consistent standard for evaluating the model's performance, allowing you to identify areas for future optimization.

Exploring different ways will also help to improve the model. Using advanced models like Vision Transformer (ViT) can improve feature extraction and clustering accuracy. Another potential improvement is to convert images into written descriptions, cluster these descriptions, and compare the results to existing clustering methods. This strategy may provide a more cost-effective and scalable solution.

Finally, doing a thorough comparison of alternative models and methodologies, together with a cost analysis, will aid in determining the most effective and efficient strategy. This comparison will verify that the chosen solution meets the client's objectives while maintaining a balance of performance and cost-effectiveness.

Final Results

Clustered Images Output

The final output of the model are sets of clusters or folders which have assigned labels to them that describe what the images are in the folder. The folders of the output are shown in **figure 5, figure 6 and figure 7.** The corresponding images that are assigned to a specific cluster are shown in **figure 8 and figure 9.** Note that there are 1 or 2 images that may be out of place, but with a little more finetuning, the output of the model can produce clusters and corresponding images with a very high accuracy.

When navigating the link above, you may download the zip and you will get the same sets of folders as shown in the **appendices**.This is a direct output for a 300 image dataset generated at random from the whole dataset, and for a single user, the model can be coded to take images from certain User IDs and generate sets of clusters with labels for images from that single user.

Appendices

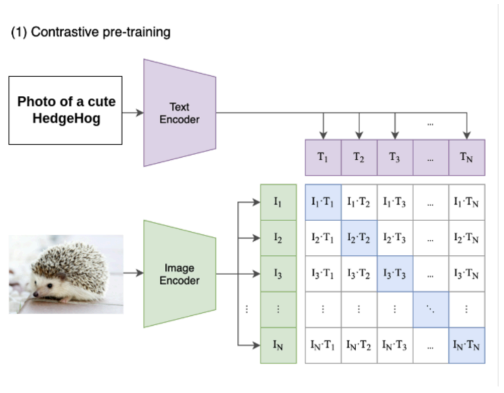


Figure 1, CLIP Embedding

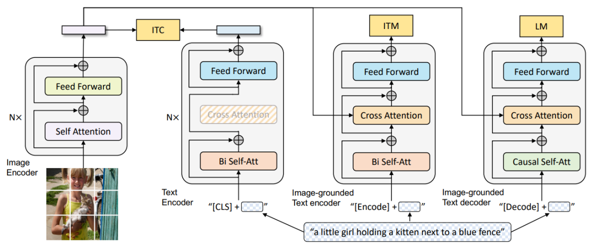


Figure 2, BLIP Text Encoder

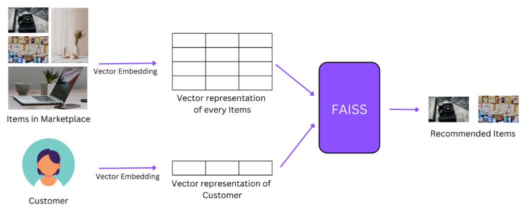


Figure 3, FAISS Index Search

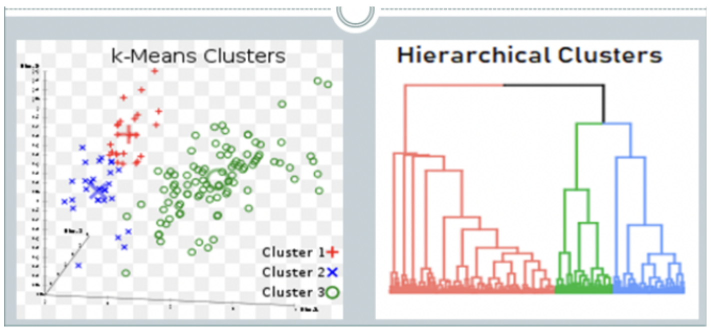


Figure 4, K-means basic clustering and Hierarchical Clustering

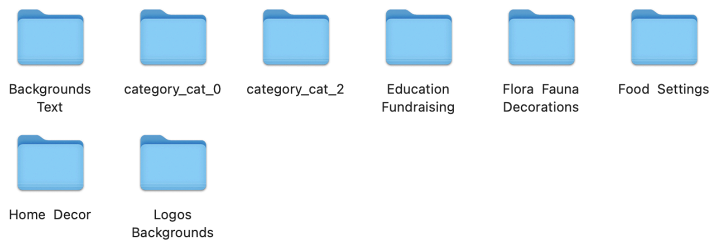


Figure 5, Cluster Labels

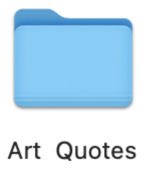
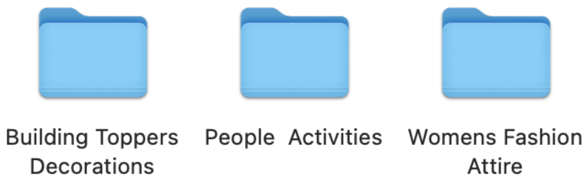


Figure 6, category_cat_0 cluster label


Building Toppers Decorations

People Activities

Womens Fashion Attire

Figure 7, category_cat_2 cluster label

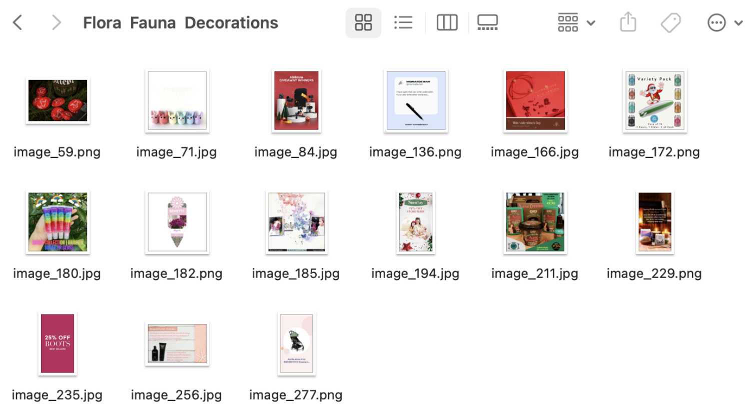


Figure 8, Corresponding images to specific cluster

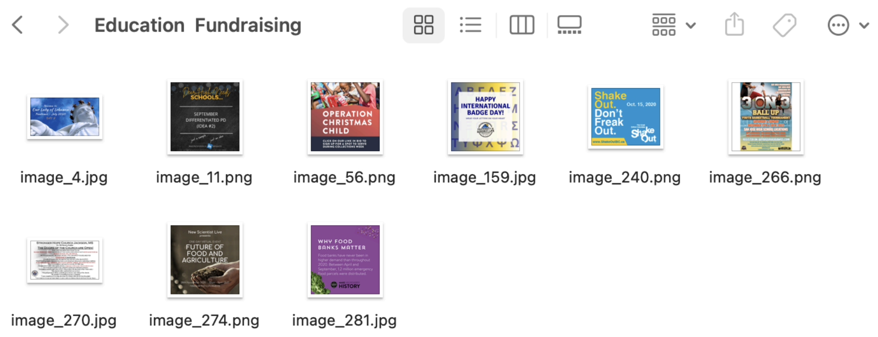


Figure 9, Corresponding images to specific cluster