GALAXIDNET: CONTENT-BASED GALAXY IMAGE RETRIEVAL

Ryan Friberg

Columbia University
Department of Computer Science

ABSTRACT

The study of galactic structure is a vital tool used to better understand the evolution of the universe, cosmology, and even astrophysical theories such as the existence of dark matter. This paper outlines a computer vision pipeline powered by deep learning to extract and find galaxies that share similar visual features from a given database, a practice known as content-based image retrieval. Grouping galaxies in this manner allows for further targeted observation and study to learn more about the cause of the shared features. Using the Galaxy10 DECaLS dataset, this pipeline trains a custom, self-supervised vision transformer model from scratch via contrastive loss. With a trained model, a search dataset is constructed with each image's extracted features to remove repeated inference runs across searches with the same model. With the setup completed, this search dataset can be queried to extract the top-k most similar images to any given query image. In the end, the system showed promising results both numerically and visually.

1. INTRODUCTION

Galaxies exist across the cosmos in many different shapes and colors depending on characteristics such as their age or composition. The study of this diversity is able to reveal critical information about the universe and the conditions necessary to create such galactic structures. Today, analysis of galactic anatomy is one of the most salient methods of gathering evidence for or against various astrophysical and cosmological theories. Examples include the existence of dark matter, the evolution and expansion of the observable universe, and even the Big Bang itself.

In the field of astrophysics, there are established classes of galaxies such as elliptical, spiral, or barred (among others) but even within these categories, galaxies can have diverse visual features [1]. For example, perhaps a spiral galaxy has one arm that stretches further than another, or a galaxy has an unusually high number of satellite galaxies. These types of characteristics are usually not included in standard galactic classification labels as there would simply be too many classes. If an astrophysicist were to find a galaxy with a unique structural or visual feature, being able to find more examples of galaxies with said feature could provide justification for future targeted study of those astronomical bodies. Extracting the shared conditions that gave rise to these features could definitively aid in numerous astrophysical research efforts.

In this paper, I present an application that leverages deep learning to extract the visual features of galactic images and return the set of the k most similar images, a practice known as content-based image retrieval (CBIR). CBIR, or image search, is a unique problem within computer vision as, unlike tasks such as classification, it is loosely defined. Even for a human, it is not always easy to define metrics relating how similar images are that work in every possible case. Additionally, CBIR is shackled by two major technical issues. The first is the potential for a semantic gap: a disparity between the low-level features such as pixel intensity values (seen by the model) and the high-level features such as galactic structure (seen by a human). The second potential difficulty of CBIR is the variability in the images as a model may not be able to learn tensor representations that are robust to an extremely high number and range in visual features. An example of this is having images of the same object from many different angles, in may different lightings, or present in many different background contexts. While it is difficult to fully address the first issue, galaxy images are uniquely suited for CBIR regarding this second issue. Images of galaxies all already look visually similar (in a rough manner of speaking, they are all images of bright blobs on dark backgrounds) and there is only one perspective (Earth's), hopefully allowing the model to learn more granular details between images. To further help in this granular learning, I chose to train a model from scratch as opposed to fine-tuning a model pre-trained on a large and diverse dataset such as ImageNet.

With this motivation and problem in mind, I chose to create a pipeline with a custom transformer-based model, a custom similarity scoring metric, and two custom PyTorch datasets to achieve CBIR for galactic images. With a trained model, the application can be given a query image, generate its feature representation, and run a search operation on a given database. The system is designed to be fully configurable for training and search, allowing for custom training and search frameworks depending on available compute resources and the number of images desired.

Fig. 1. A visualization of the diversity of galactic shape, size, and structure.

2. RELATED WORK

2.1. Astrophysics

The body of relevant astrophysical work has mainly served as the motivation for the creation of this pipeline. There is a whole subfield of astrophysics specifically dedicated to studying galactic structural information and its implications. Conselice (2014 [2]) provides an in-depth overview of the current understanding of a galaxy's lifetime evolution. Due to the immense timelines that the structural changes occur over, studying a galaxy's age in relationship to its structure is able to yield insights about the evolution of the universe as a whole and inform cosmological models such as the age of the universe. Further work has gone into simply documenting the sheer diversity of galaxies. Buta (2011 [1]) provides an outline of many of the agreed-upon categories and how different galaxies differ according to their class. With these two frameworks in place, it is clear to see the significance of the galactic structural study.

2.2. Computer Vision

This paper's technical components leans on approaches and techniques from previous works within the realm of computer vision. First and foremost, the pipeline uses an adaptation of the transformer model architecture from the landmark paper from Dosovitskiy et al. (2021 [3]). On top of this, the survey by Dubey (2022 [4]) influenced the overall structure of the pipeline I ended up creating.

2.3. Contributions

My contributions in this paper are a fully customizable, model and data-agnostic pipeline, and self-supervised pipeline for CBIR, a specific implementation of this agnostic system for galactic image CBIR, and a test-time performance metric to help analyse how well the model is performing beyond examining the loss.

3. METHODOLOGY

3.1. Task Definition

The task of CBIR can be decomposed into two main parts: feature extraction and the actual image search (visualized in Figure 2). To extract features, one could use algorithmic approaches such as the scale-invariant feature transform (SIFT) algorithm [5] but the achievements of deep learning techniques has made training a neural network model the most favorable approach. The general idea is for this model to learn a latent-space encoding of images such that, if training is working, similar images will have "similar" corresponding encoded tensors. This introduces two design choices: the model architecture and the similarity scoring function. Given their proven success and well-documented architecture [3], I chose to build a custom vision transformer (ViT) model. For the similarity scoring, in order to incorporate multiple types of similarities of various dimensionalities, I created a custom similarity score that is a weighted sum of the cosine similarity, euclidean distance, and the L3 Minkowski distance.

With the extracted features, the second component of CBIR boils down to simply using the scoring function for a database search. For any query image, the system runs inference to get a set of features, compares these features with the features of all the images in a dataset, and maintains a list of the most similar entries. To speed up search time, I elected to pre-compute the feature tensors of the database to avoid repeatedly running inference on the database images.

3.2. Data

As in most machine learning systems, the success in a given task is more dependent on the quality of the data than the model architecture. A dataset needs to maximize the important visual feature representation while minimizing the amount of irrelevant visual information. Luckily, astrophysical data is publicly available in extremely vast quantities. Many types and sources of data were considered but in this instance, I elected to use the pre-compiled Galaxy10 DECaLS dataset [6] (examples of which can be seen in Figure 1). Comprised of just under 18,000 high resolution and similarly-framed images of galaxies, this dataset was pooled from the Dark Energy Camera Legacy Survey (DECaLS).

As the name implies, this dataset groups its images into 10 classes based on the galaxy (spiral, elliptical, etc.) as well as the galactic inclination (face or edge-on), however, these labels were not considered in the training process to allow for visual features to be shared across categories.

The pipeline presented in this paper is data-agnostic, meaning there are no specific processing or computations related to the Galaxy10 dataset in the model, its training, the scoring function, or any other part of the code. While adding such computations (such as leveraging the Galaxy10 classes during training), could improve performance, it reduces the applicability of the system to other CBIR tasks. This pipeline is structured in such a way that *any* dataset can be given to the model and a viable CBIR model can be created.

In the future, the idealized form of this specific pipeline could query a massive dataset such as the Sloan Digital Sky Survey (SDSS) directly with right ascension (ra) and declination (dec) values to any given celestial body. The vast search database would be comprised of the documented bodies in the survey. As the SDSS comprises 20 years of data, doing so would also allow astrophysicists to query across the time dimension. Though 20 years is extremely insignificant in the timeline of the universe, short-term, pointed events such as supernovae could be documented and added to research studies in this way.

3.3. Model

Though the type of model is often less influential towards the success of pipelines on novel tasks, it is still important to select an architecture that achieves the highest performance ceiling. A ViT was chosen as this architecture has proven to be an extremely powerful feature extractor across many modalities of data. In this study, I created a custom transfomer architecture that uses a linear encoder with cosine position embeddings as well as a multi-head attention encoder composed of residual layer-normalized blocks. The ending configuration used image patches of size 16x16, a hidden size of 1024, 3 residual blocks and 4 attention heads per block to create an output vector of 2048 per patch in an input image. In all, the model has 18,060,288 trainable parameters. This configuration was settled on due to the its size-to-training-time ratio given the size of the dataset and speeds of the compute resources allocated via the google cloud platform (GCP). One design choice that became relevant when building this model was how to handle an image's channels. The three main choices are as follows: assume all incoming images are gray-scaled, interpret each channel of an image as its own image (for RGB images, this effectively triples the batch size), or to create patches out of the image with all the channels included per patch. Because color information is relevant to galactic visual features, forcing grayscale was not an ideal directive. Between the latter two options, I chose to add the channels to the patches so that images essentially have each patch encoded for both position and color.

Lastly, the decision to create a custom configuration over fine-tuning a pre-trained model came from the hypothesis that CBIR models trained on enormous and visually diverse datasets such as ImageNet may not be able to extract granular galaxy features (i.e. all galaxy images may have highly similar feature representations since the model was trained on images of many different categories). However, investigating fine-tuning and experimenting with even larger models would be an interesting future study. Just as with the data, this system was designed to be model-agnostic, allowing for ease of such analysis.

3.4. Training Loop & Loss Function

There is no singular correct method for training CBIR systems. Depending on the data available, some approaches may leverage gold standard labels and some may be constructed in a fully self-supervised fashion. To make the most widely-applicable model possible, I elected to design a fully self-supervised training loop. During training, for each image in a batch, I generated a positive or negative pair (with a training parameter option to randomly generate one or the other to save on compute time). Generating positive pairs could have simply been sampling from the same class but this yields two issues. First, this would introduce a Galaxy10-specific constraint meaning if another dataset were to be used (which may not even have labels) then this training loop would fail. The second is, as described before, I cannot necessarily guarantee that all the images within the same class can be regarded as visually similar.

To generate these pairs, I instead rely on image transformations by defining both a set of positive transformations and a set of negative transformations. For the positive, these transformations guarantee that the structure overall visual features will remain the in-tact (x/y-flip, rotations, adding minor Gaussian noise, etc.) but that the low-level features given to the model are highly different, addressing the semantic gap. In contrast, the negative transformations can yield more substantial visual alterations (perspective warp, Gaussian blur, etc.) which have the potential to destroy the original visual features. For each positive image in the batch, a random subset of the positive transformations are applied the image itself. For each negative image, a random subset of the negative transformations is applied to a different image from anywhere in the dataset including the same class. The number of transformations to apply was implemented as a configurable training hyper-parameter. Examples can be seen in the Appendix of this paper in Figures 4 & 5.

The model then generates the feature representations for each image in the set of pairs. At this stage, I now have a set of outputs and a label associated between each pair (as opposed to each image itself). I am then able to utilise the average contrastive loss formula:

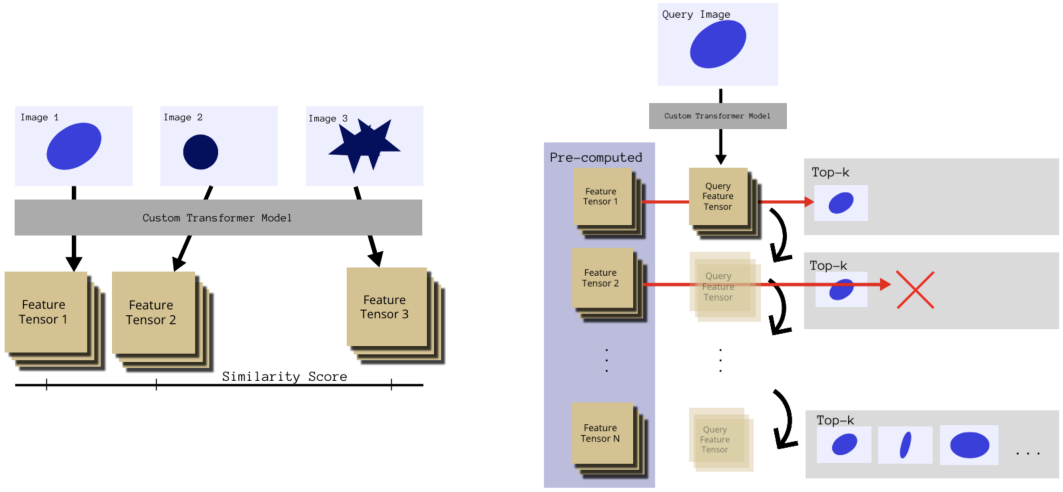

Fig. 2. A visualization of the two components of the pipeline. Feature extraction (left), and search (right).

$$L = \frac{1}{N}\Sigma_{i=0}^{N}\left[((1-y_i)^2 \cdot s_i^2) + (y_i \cdot (\max(0, m - s_i))^2)\right]$$

Where N is the number of pairs, y_i is the pair-wise label, s_i is the similarity score of the pair's outputs, and m is a margin (set to 1.0, as that is the maximum value of my custom similarity function).

Exploiting the randomly selected transformations vastly artificially boosts the training dataset as there are many more possible pairings for any image. However, due to the heavy use of randomness during training, model performance and loss throughout training may not always follow standard loss/accuracy curves but instead jump around non-deterministically.

3.5. Testing

One of the biggest issues of developing a self-supervised CBIR model is the ability to monitor its performance during training. A common method of evaluation for CBIR is to divide the number of relevant results by the total number of extracted images. A goal of this project was to define a novel performance metric that could be computed algorithmically without human intervention. While simply monitoring the loss can reveal information about how well the model is doing, I propose a minisearch loose accuracy method. During testing, for each given batch of images, I created a single positive pair via the same positive transformations from training on image i. I then created (batch_size - 1) negative pairs of image i with every other image in the batch. This setup mimics a top-1 search task where each pair consists of the same query image, and a database option. While computing the overall loss, the testing function performs this mini-search and increments the accuracy if the most similar image is selected.

This approach is far from perfect and very data dependent as there may be images in the batch that are truly more similar to the query image than the query image is to its transformed self. The randomness of the positive-labeled image pair makes this metric difficult to take at face value compared to accuracy in a task such as image classification. Despite this, in the event of visually diverse batches, this formulation does provide reasonably robust information about the model's performance and its feature representations' invariance to simple transformations.

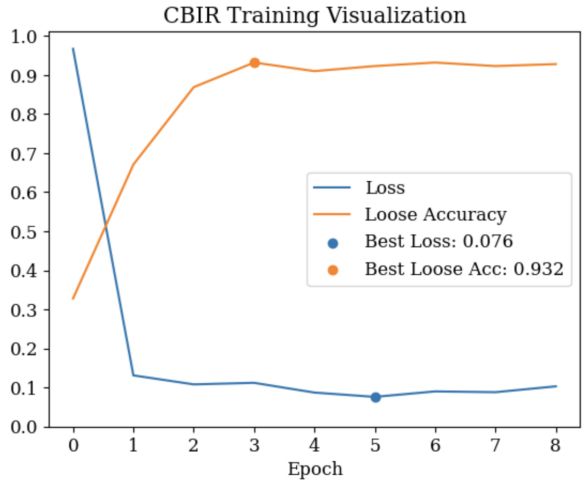

Fig. 3. Mini-search loose accuracy and loss during training.

3.6. Search

With a trained model, the system is able to take a query image, determine its features, and compare it with all the features of the images in the dataset (as visualized in Figure **??**). Depending on the model and dataset sizes, searching can be extremely computationally expensive. As such, before any

search happens, the pipeline constructs a dataset comprised of all of the outputs from a trained model to save on search times. The search functionality keeps track of a user-specified number of the most similar images according to a custom similarity score. The score is defined as:

$$\begin{aligned}\text{sim}(T_1, T_2) = \; & w_1 \text{cos\_sim}(T_1, T_2) \\ & + w_2 \frac{1}{(1 + \text{L2\_dist}(T_1, T_2))} \\ & + w_3 \frac{1}{(1 + \text{L3\_dist}(T_1, T_2))}\end{aligned}$$

This weighted sum converts the distance values into similarity scores and allows for higher-order distance information to be recognized. For this study, all three subcomponent weights were simply set to $1/3$ but future work could benefit from exploring tuning these similarity scoring function weights (or even learning them).

4. RESULTS

The results of this pipeline can be interpreted in two fashions. The first is by purely numerically, relying on the model's loss and the relevancy of the mini-search loose accuracy. As seen in Figure 3, with the right configuration, the model was able to follow traditional loss and metric curves, achieving a minimum validation loss value of 0.076 and a maximum loose accuracy of roughly 93%. However, the more important results are the quality of the image extractions. By randomly sampling from the dataset and manually examining the results, plenty of examples can be found of cases of good results (Figures 6 & 7) (all of the resulting images share at least some visual features with the query image), mixed results (Figures 8 & 9) (only some images seem to share features), and poor results (Figure 10) (none of the images seem to share features).

Unfortunately, to truly evaluate this system, a trained cosmolgist's input on how relevant the images are may be necessary.

5. DISCUSSION & FUTURE WORK

Without full insight into the performance (i.e., seeing trustworthy metrics across a large set of query images), it is difficult to infer why the model is making such decisions. Though the model achieved some success both numerically and visually, the process of designing idealized and general-purposed image transformations and similarity scoring remains a difficult task. I settled on the ones ultimately presented in this pipeline by means of trial and error but much deeper study into the most effective study may be necessary, especially if this pipeline is used on a vastly different dataset.

One thing I personally noticed was that more often than not, the top-k search results were in the "mixed" category. The task-irrelevant features (such as the surrounding stars) seem to sway the model's decision making. This suggests the model is not giving high enough attention to the actual galaxy itself in these circumstances. While I have mentioned various areas for future work throughout this paper, I think the most immediately useful future study would be addressing this erroneous attention. Possible avenues to consider could be adding an object-recognition component to extract the section of the image that solely contains the galaxy. Another generally useful addition would be creating visualization techniques of overlaying the various attention-heads' attention weights on the image to monitor the features that the model regards as important.

6. CONCLUSION

In this paper, I present a methodology for a data and model-agnostic pipeline to conduct content-based image retrieval in a self-supervised fashion with a novel loose scoring metric to monitor model performance during training. Given my personal experience with the subject matter, I chose to apply this system to astrophysical data in the form of images of galaxies. I proposed a full label-less and automated training and testing system that could theoretically be applied to any image dataset. In the end, the pipeline performed reasonably well in both numerical metrics and visual evaluations. Areas for future study could focus on extracting the most relevant subsection of images through techniques such as object detection.

7. GITHUB REPOSITORY

Please find any and all associated code in this Github repository.

8. REFERENCES

[1] Ronald J. Buta, "Galaxy types: Stage, family, and variety," 2011.

[2] Christopher J Conselice, "The evolution of galaxy structure over cosmic time," *Annual Review of Astronomy and Astrophysics*, vol. 52, pp. 291–337, 2014.

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021.

[4] Shiv Ram Dubey, "A decade survey of content based image retrieval using deep learning," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 5, pp. 2687–2704, May 2022.

[5] Sahil Jain, Kiranmai Pulaparthi, and Chetan Fulara, “Content based image retrieval,” 2015.

[6] Henry Leung, “Galaxy10 decals dataset,” 2017-2023.

9. APPENDIX

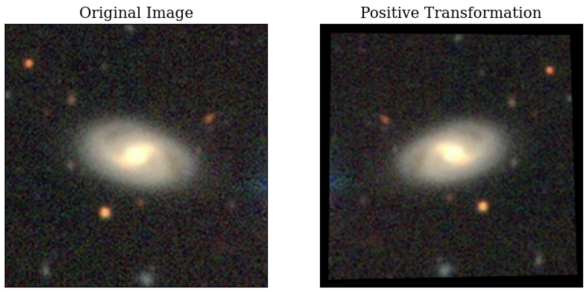

Fig. 4. Example of positive training pair

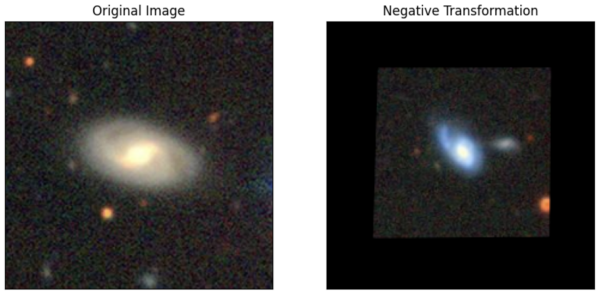

Fig. 5. Example of negative image pair

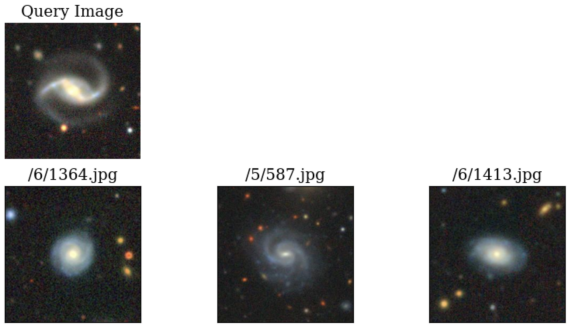

Fig. 6. Example of good search results

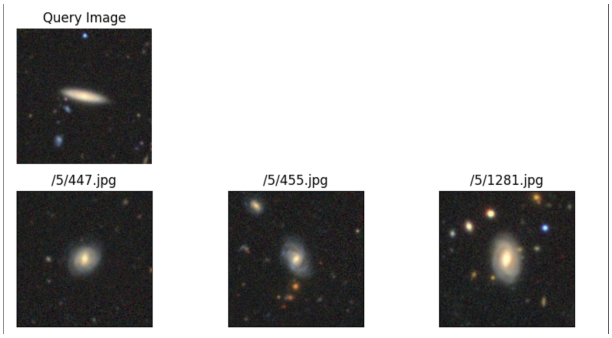

Fig. 7. Second example of good search results

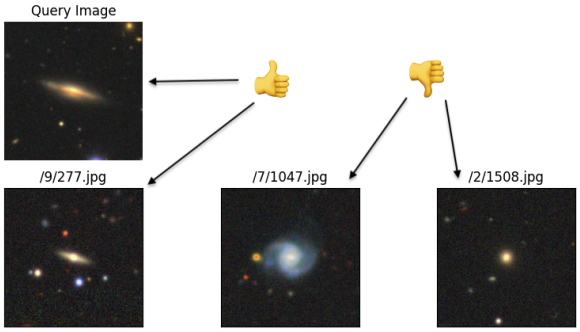

Fig. 8. Example of mixed search results

Fig. 9. Second example of mixed search results

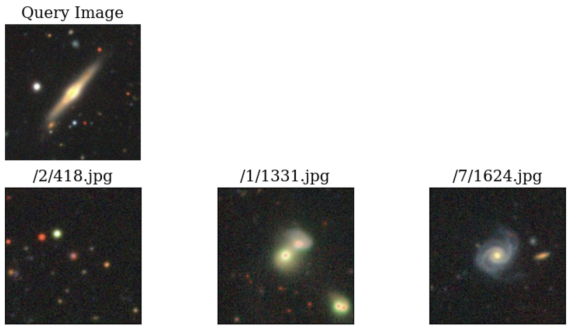

Fig. 10. Example of poor search results