Satellite Image Similarity

Andrew Ryan
Department of Electrical and Computer Engineering
Virginia Polytechnic Institute and State University
Arlington, VA 22203
andrp11@.edu

Sally Owen
Department of Electrical and Computer Engineering
Virginia Polytechnic Institute and State University
Blacksburg, VA 24060
osally99@vt.edu

Abstract

Given the vast amount of overhead imagery collected of Earth every day from satellites, airplanes, and drones; techniques are required to examine and search for features across this petabyte scale data source. Object detection, image classification, and image segmentation are popular supervised learning tasks with remotely sensed imagery in order to localize objects or features within a scene. This work aims to explore a task where a user has a query image and wishes to find images similar to it. At its core, this is a content based image retrieval task, similar to a Google Reverse Image Search, but applied to an overhead imagery source. This paper explores this effort by building a small scale framework similar to [1], consisting of a neural network to extract an image's latent information and computing the similarity between it and known samples previously seen before.

1 Introduction

There are many ways in which two images could be deemed similar, for example color, brightness, texture, or shapes. However, it is difficult for a human to make a quantitative assessment to identify exactly which of those features, or qualities, about the images that make them similar. This is a core problem in content based image retrieval (CBIR), which is a system that allows the user to make a query to find similar images. To accomplish this, CBIR uses computer vision, and convolutional neural networks specifically, to extract the latent features within the image and the relationships between them. This is a task similar to topic modeling with text data to classify what a particular document is about using the words of sentences within the document. Using the latent features, a vector representation of the image, the distance, or similarity, between two images can be computed to identify how far apart they are from one another. Images closer are deemed to be more similar than images farther away.

This work is laid out as follows: the second section describes the methods undertaken here, section three discusses the results, and section for discusses future work and improvements that can be made.

2 Methods

To conduct these experiments and begin laying the foundations for a small scale CBIR system for satellite imagery, we first identified the dataset to be used and prepared it for use with a convolutional

Preprint. Under review.

neural network (CNN). Once the data was prepared for modeling, we trained a neural network and attempted cursory optimization of the hyperparameters. Each of the images from the training dataset were then fed through the network to extract their embeddings, after which some sample query images, which were held out of training, were also run through the network to extract their embeddings as well. From here, the pairwise distance between the query images and every image from the dataset was calculated. These similarities were indexed and sorted to return the k closest images to the query.

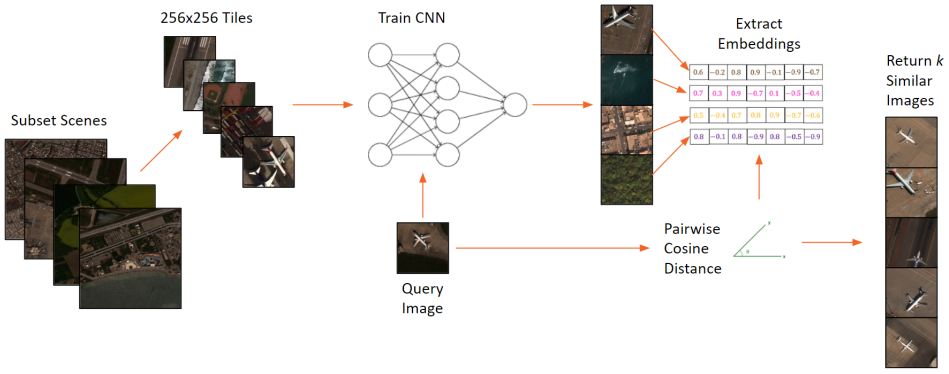

Figure 1: Satellite Image CBIR Diagram

2.1 Dataset

For this work, the xView dataset was selected. xView, an object detection dataset, is a publicly available dataset of satellite images and bounding box labeled objects. xView was released by the National Geospatial-Intelligence Agency (NGA) and the Defense Innovation Unit Experimental (DIUx) and contains high resolution, 30cm, imagery from Maxar Technologies with nearly 1 million objects spread across 60 different classes. xView was selected because the imagery in the dataset is diverse and covers various geographic areas and types of objects.

The images from xView are subsets of full satellite images, which is helpful in reducing the dataset size because full size satellite images can be 10 to 40GB per image. However, the images are still quite large and not easily consumable by a neural network, so additional processing is needed. Because the dataset was not being used for its intended purpose for this work, we took a naive approach to further process the images. Each of the 846 images in the dataset were split into 256 x 256 tiles, which resulted in 8,800 tiles. See Figure 2 for an example of the reduction from full satellite image scene, to the subset provided by xView, to the tiling process we used to further break down the dataset. After the images were tiled, we arbitrarily classified each of the tiles into the following classes: agriculture, airport, boats, clay, clouds, desert, developed, forest, grass, mining, oil, parking, planes, port, rail, roads, sports, water [see: Figure 3]. This was just a simple binning of tiles based on what we thought was the primary focus of the tile.

2.2 Model

For this work, we used a ResNet50[1] network [see: Figure 4] pretrained with the ImageNet dataset, which was also used in [1].The PyTorch framework was used to train the network. Basic training data and hyperparameter tuning was done, specifically we tried various train/test splits, batch sizes, optimizers, learning rates, and epochs. The best hyperparameters identified were a train/test split of 90/10, batch size of 64, Stochastic Gradient Descent, 0.01 learning rate, cross entropy loss, and 30 epochs.

2.3 Similarity Search

Once the network was trained, the training data was run through network to extract the embedding for each image. Then, the query images were run through the network to get their embeddings. Once

[1]https://arxiv.org/pdf/1512.03385.pdf

Figure 2: Example of a full satellite image, scene subset provided by xView, and 256 x 256 tile

Figure 3: Example of each class.

this was done, the similarities between the query images and the training data are calculated. We used the cosine similarity, which is a slight deviation from [1] where they used the Hamming distance. We decided to use the cosine similarity to compute distance between two vectors because the vectors could be far apart in 2-dimensional space using a metric such as Euclidean or Manhattan distance, but be oriented closer in a multi-dimensional space when looking at the angle between them. Once the similarities have been computed, the distances between the images are sorted to return the k-most similar images. See Figure 5 for examples of the similarity queries.

3 Results & Future Work

For an initial attempt at this problem, the results are promising. Queries are returning images that look visually similar though with some noise and the most similar image might not have the highest scores. There is much that can still be done to improve the process to get better results however. Many improvements can be made in the training dataset generation and curation. As mentioned previously, the dataset was intended for object detection so we took the images from the dataset and tiled them to create a classification dataset. We used a naive approach by creating 256 x 256 tiles of each image. This was done irrespective to objects such as planes, boats, etc. so many tiles containing objects may have had parts of the objects missing. Additionally, because we applied this broad tiling scheme there are tiles which could have multiple class within each tile. For example, a tile at an airport could contain planes, buildings, roads, and parking lots, or a tile of a developed area could have buildings, roads, parking lots, green space, etc. Applying this broad tiling scheme to broad classes and then applying only a single class to it can cause the model to become confused. In the future, it would be worth exploring incorporating ancillary data, such as OpenStreetMap, to provide information such as land use, zoning information, feature classes, etc. then use a segmentation approach to classify the images. A combined approach including the images chipped as a true object detection dataset could

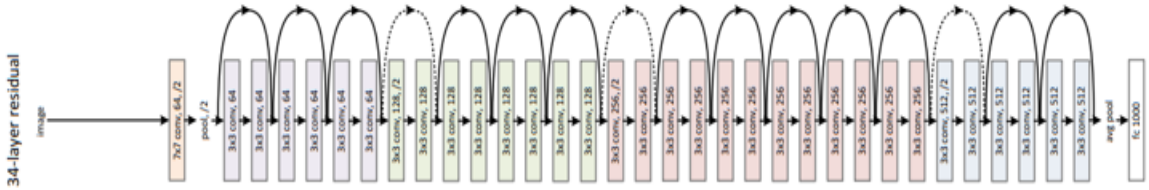

Figure 4: ResNet architecture

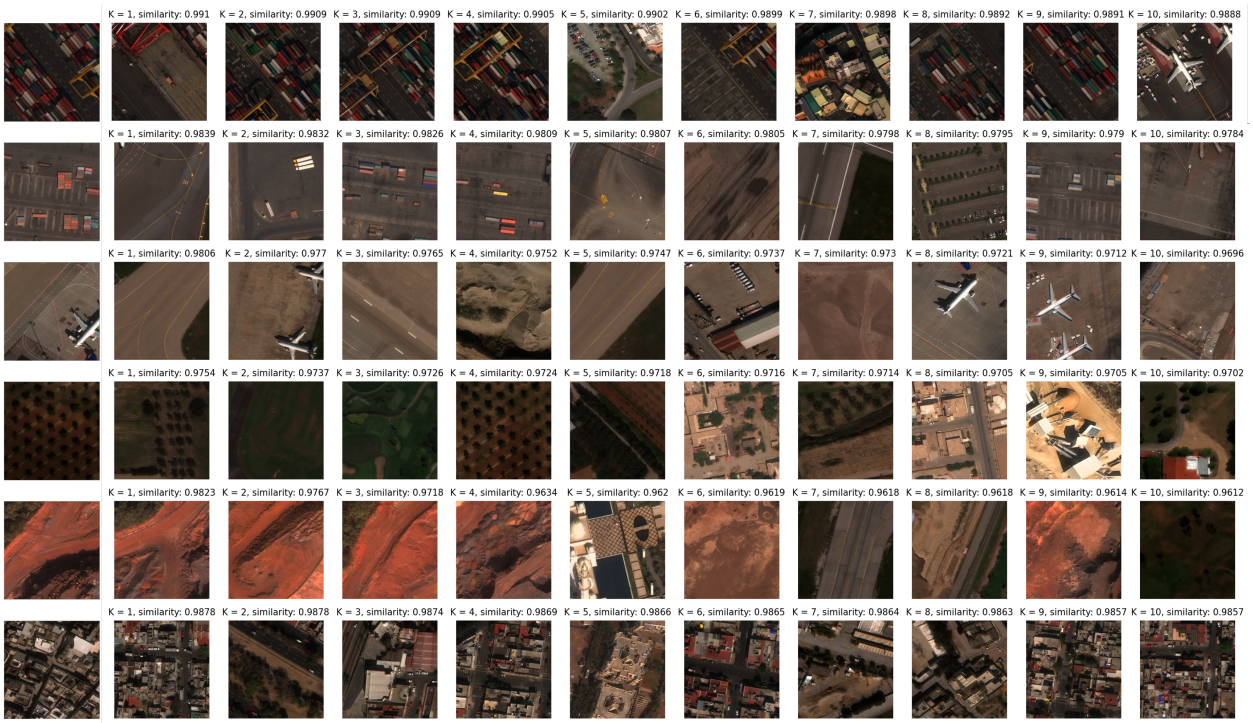

Figure 5: Similarity Results, first column are query images

maximize class refinement for training a general classifier on both landuse and objects. In addition, many of the classes were very unbalanced. Getting more training data, and undertaking the above approach, could allow for better classifier performance and hopefully better similarity scores.

We deviated a bit from [1] when computing the similarities. They used the hamming distance to compute the distance between binary strings whereas we used the cosine similarity metric. Exploring the method using the hamming distance could be worthwhile and may provide better similarity results. Other distance metrics, such as Euclidean and Manhattan, should be applied to benchmark which distance metric would maximize the performance of this work. Additionally, we relied heavily on the classifier scores to understand how the model was performing and whether or not we needed to make adjustments. However, the we had no way to score how well the similarity search was performing. Depending on the query image, random images might get returned with a very high similarity. A metric should be identified or developed to help evaluate the performance of this task.

A core piece of this work was trying to explore using an autoencoder for an unsupervised learning approach. We were able to train an autoencoder but wern't able to extract the image embeddings from the network. This avenue should still be pursued as it could help alleviate possible issues with the network learning the class label of an image versus the true latent information of an image. We also only looked at ResNet50, other networks should be evaluated to identify if there is a more optimal network for this task, e.g. ResNet34, ResNet101, VGG, Inception, custom networks or even vision transformers (ViT).

Finally, we would like to explore developing a user interface around this and returning the exact spatial location (latitude, longitude) of the similar images. This is something that [1] completed and provides a better user interface and more functionality.

4 Individual Contributions

Andrew Coded framework with PyTorch (dataloaders, train, test, model training functions), researched which network to start with how to extract the embeddings and compute the similarities between query images and the dataset, provided text for report and helped with formatting

Sally Identified the dataset, how to tile the dataset, conducted parameter and hyperparameter tuning, assisted with coding additional functionality, identified how to get the code working in Google Colab, trained models, provided text for report and helped with formatting

5 References

[1] Keisler, R., Skillman, S. W., Gonnabathula, S., Poehnelt, J., Rudelis, X., &; Warren, M. S. (2019). Visual search over billions of aerial and satellite images. Computer Vision and Image Understanding, 187, 102790. https://arxiv.org/pdf/2002.02624.pdf