Rapid Visual Search Over Aerial Images Using CNN Fingerprinting

Noa Schwartz
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA
noaleetz@mit.edu

Sean Condon
Department of Physics
Massachusetts Institute of Technology
Cambridge, MA
scondon@mit.edu

***Abstract*—The availability of satellite and aerial imagery has increased massively in recent years, and the analysis of this type of data is invaluable for many modern tasks: optimizing crop yield, predicting weather, monitoring infrastructure, etc. As the total amount of aerial imagery data exceeds the petabyte scale, it is necessary to have efficient methods to analyze and compress this type of data. This paper presents one such method: using convolutional neural networks to compress large datasets of aerial images into low-dimensional fingerprints, which enables rapid similarity search between a query image and the rest of the dataset. The image data is first cut into small chunks of 256 x 256 pixels, then each is fed through a pretrained CNN which produces a fingerprint of 512 information-dense features, resulting in a data size reduction of 384x. Euclidean distance in the 512-dimensional fingerprint feature space is interpreted as a proxy for image similarity, enabling a user to reliably find all images similar to a query image with speeds orders of magnitude faster than searching through the original, uncompressed data.**

I. Introduction

Content-Based Image Retrieval (CBIR) describes the process of retrieving images from a significantly large database of candidate images. The term "Content-Based" specifies that the contents of the queried image, where contents refers to shape, texture, color, and spatial layout, is used to search and present similar digital images. The similarity between the query image and the database images is determined by ranking the database images in order of a computed similarity score.

In CBIR, such low-level features are extracted from the query and are used to compare the visual similarity. One of the central features of CBIR is feature extraction- for each image query, a new, unique set of features is used to compare images (features are learned automatically from the data) [9]. This is particularly valuable in databases of digital images, where we need to transform the data to a lower-dimensional space while retaining meaningful properties from the original data. One possible way to address feature extraction is through the use of a CNN to compress the original images into a more space-efficient form of data, as we will discuss over the course of this paper. The new form of data is what is used to perform similarity comparisons.

Using learning-based rather than hand-engineered feature generation is a shift specific to the 2011-2020 time frame, due to the emergence of deep learning. Beginning in 2011, Krizhevsky and Hinton used a deep autoencoder to use short binary codes to represent the database images. A year later, Kang et al. introduced deep multi-view hashing to capture multiple views of data by using view-specific and shared hidden nodes to model the layers. In 2014, Babenko et al. formalized the use of activation functions on the top layers of CNNs to serve as the neural codes for the image retrieval application. The neural codes were further compressed with PCA. In 2015, Lai et al. used a combination of convolution layers and hash bits, where the layers formed the intermediate image features serving as the inputs to the hashing algorithm [6]. Throughout the last decade, it is clear that the deep learning models applied to image retrieval have evolved tremendously. Researchers have explored a variety of network types, including CNNs, Autoencoders, GANs, in the context of both supervised and unsupervised models [8]. Throughout the course of this paper, we will discuss the implementation of a CNN to transform tiled images into arrays of size 512. As research progresses, the goal will continue to be to master the tradeof between

Fig. 1: A high-resolution aerial image from our dataset. This specific image is 12000 x 14000 pixels with a resolution of about 10 cm per pixel, so it will be processed into 2500 tiles of 256 x 256 pixels each. Each tile will then be compressed into a information-dense fingerprint for similarity search.

reduction in representation size and the meaningfulness of the extracted data.

II. Motivation

CBIR has developed into a search problem of interest because search via querying metadata, such as tags, keywords, or descriptions have a dependence on manual human annotation. This dependence is limiting to image retrieval applications because (1) accuracy of the model is bounded by the quality of human annotation and (2) it is not scalable for large databases. The explosion in database size can be attributed largely to recent technological development (improved smartphone camera quality, the ease of sharing images and videos on the Internet) resulting in widespread use of shared and stored multimedia data.

Some images databases are created through automatic image generation, such as surveillance cameras- such a use case is also not scalable using human annotation. Another difficulty that metadata based image retrieval presents is the added manual curation needed to prevent annotated data from being miscategorized when categories include overlapping subclasses. For example, "mint" can be in a superclass "plant" or "ice cream", requiring more advanced semantic understanding by a labeler to use the remaining contents in the image in order to perform a correct annotation. As systems with large digital image databases become more common in a wide variety of domains, including scientific, medical, advertising/marketing, and consumer products the need for efficiently storing images so that they can be identified has become top of mind [2].

III. Approach

A. Aerial Image Data Set

The dataset we used as our digital image database (obtained from the DroneDeploy Aerial Segmentation Benchmark Challenge) contains aerial orthomosaics captured by drones belonging to 6 different classes: Ground, Water, Vegetation, Cars, Clutter, and Building [5]. The image resolution is approximately 10cm per pixel, providing a great level of detail with which to perform image extraction. The images are stored as RGB TIFFS. In total, the original dataset, prior to any preprocessing required for feature extraction is 10GB.

B. Tiling

The .TIF images described in section III-A are processed into tiles of 256 x 256 pixels. This size was chosen so that objects in the images (such as cars, machinery, trees, etc.) are clearly visible, but each tile does not typically contain many different objects.

The tiling is accomplished by sliding a 256 x 256 window over each .TIF image with a stride of 128 pixels. Using a stride length of half the window size ensures that most small motifs will be well-centered in at least one tile. Image dimensions are cropped to be evenly divisible by the window length so that no padding is needed. Additionally, because most of the .TIF images have non-rectangular borders, some tiles will contain majority black pixels. For this reason, tiles that have $> 50\%$ black pixels are discarded.

Fig. 2: 25 tiles randomly selected from the 45,000 total tiles. Each is 256 x 256 RGB pixels and is created using the workflow described in section III-B

.

The resulting 256 x 256 RGB images are saved as 3-dimensional matrices by the NumPy computational package for Python so that they can be easily read into the neural network for fingerprinting. After discarding the majority black tiles, the total size of the data at the current step is 9 GB. However, the 8-bit integer pixel values will need to be converted to 32-bit floating decimal point values for the fingerprinting step (described more in section III-C1), so the actual size at this step is closer to 70 GB.

Figure 2 shows a batch of these tiles.

C. CNN-Generated Fingerprints

During fingerprinting, the tiles that were created in section III-B are fed through a convolutional neural network to extract a low-dimensional feature vector for each. The tiles are first pre-processed so that pixel value distributions are optimal for the CNN, and then feature vectors are inferred using a pretrained neural network.

1) Pre-Processing: In order for optimal inference by the pretained neural network, pixel values for tiles must have certain distributions. First, tile dimensions are arranged in the format C x H x W, where C is the channel (3 for our RGB images), and H and W for image height and width (both 256). Pixel values in these dimensions are standardized to the means and standard deviations in table I.

Layer Name	Output Size	ResNet-18
conv1	112 × 112 × 64	7 × 7, 64, stride 2
conv2_x	56 × 56 × 64	3 × 3 max pool, stride 2 $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$
conv3_x	28 × 28 × 128	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$
conv4_x	14 × 14 × 256	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$
conv5_x	7 × 7 × 512	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$
average pool	1 × 1 × 512	7 × 7 average pool
fully connected	1000	512 × 1000 fully connections
softmax	1000	

Fig. 3: Resnet-18 convolutional neural network architecture. The pretained weights of this network optimized for ImageNet classification are used to generate fingerprints, which come from the penultimate 1 x 1 x 512 dimensional layer average_pool. Figure courtesy of [10].

Dimension	Mean	Standard Deviation
C	0.485	0.229
H	0.456	0.224
W	0.406	0.225

TABLE I: Standardizations for 3-dimensional image tiles (C = channel, H = height, W = width)

PyTorch's torchvision library comes with a transform functionality for this purpose. The normalize transform is used.

2) Model: Fingerprints are generated using the ResNet-18 architecture, a deep convolutional neural network [7]. The structure of this network is shown in figure 3. In this project, we use the weights of ResNet-18 optimized for image classification on the ImageNet dataset, with 1000 different classes of images such as tiger shark, coffee cup, rugby ball, etc [4]. It is worth noting that the weights of the network are not optimized for classification of satellite and aerial images, yet they still perform well on the aerial image data.

The final network of the ResNet-18 network is a 1000 neuron fully-connected layer with softmax activation, meant to classify the input into one of the 1000 possible classes of ImageNet. Because we want to generalize the network to perform well on our aerial data, we remove this final layer and use the second to last layer, 1 x 512 dimension average pooling layer, to generate the fingerprints for each input tile. The rationale of this decision is two-fold: the 512 dimensions of the penultimate layer offer more compression than the 1000 dimensions of the final layer, and the penultimate layer theoretically contains the most high-level information of the input image without being specialized for a class of image (like tiger shark or coffee cup) that is not relevant to aerial imagery.

Each of the 45,000 tiles in our aerial imagery dataset is fed through the pretrained ResNet-18 model, and a 512-dimensional fingerprint is taken from the activation of the penultimate layer. These are stored in a hash-table, where the fingerprint of a tile is the value, and a unique identifier (UID) for the original image is the key [10].

D. Nearest K Fingerprints

In the previous section, we demonstrated how our original aerial dataset is transformed into a hash-map mapping tile UID to the corresponding 512-dimensional fingerprint. Now, we move to discuss how similar vectors are retrieved.

1) Distance: Given two fingerprints, we would like to calculate the distance between them. This distance is inversely proportional to the similarity score between the two vectors. There are plenty of ongoing research discussions which evaluate the effectiveness of several similarity metrics in the context of content-based image retrieval. Some also explore how specific use cases of CBIR might require different similarity metrics. For example, Cho et al. compared four common similarity measures, including linear discriminant analysis (LDA), Bayesian neural network (BNN), cosine similarity measure (Cos), and Euclidean distance (ED), to determine which performs better when characterizing breast masses on ultrasound images [3].

Euclidean distance (ED) is most commonly used for measuring the distance between two feature vectors, and is what we will use to determine whether two fingerprints are similar. It performs particularly well in cases where the feature vector elements are of equal importance.

ED is calculated by taking the square root of the sum of squares of the differences between the vector components. Given two tile fingerprints $a = (x_1, x_2, ..., x_{512})$ and $b = (y_1, y_2, ..., y_{512})$, the ED is as follows [1]:

$$ED(a, b) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

.

2) Search: Now, we need a mechanism to find the k-nearest fingerprints. To do this, we perform the distance calculation between the query tile's fingerprint and the fingerprint of all other tiles in the hash-map. This brute-force calculation will yield a list sorted by Euclidean Distance of the candidate tiles relative to the query tile, and the nearest-k (those with the smallest ED) will be returned to the user.

E. Interface

The search flow will look as follows:

- A query tile image and k-value (the number of similar tiles to retrieve) are selected
- A fingerprint is generated for the queried tile
- A brute-force distance calculation is done between this fingerprint and the fingerprints of the database images
- The nearest-k tiles are presented (Fig. 4)

Fig. 4: Query Result for a Single Queried Tile with k=10

IV. Results

1) Data compression: Recall from section III-B that the original tiles are 256 x 256 x 3 = 200,000 values (originally 8-bit integers converted into 32-bit floats), and the fingerprints for each tile are 512 values (also 32-bit floats). This gives a theoretical maximum data compression rate of:

$$\eta = \frac{256 * 256 * 3}{512} = 384$$

when comparing the fingerprints to the tiles after both are converted to 32-bit floats. And a maximum compression rate of:

$$\eta_0 = \frac{256 * 256 * 3}{512} * \frac{8}{32} = 96$$

when comparing the original 8-bit integer tiles to the 32-bit float fingerprint vectors.

This means that any computational algorithm to compare visually similarity can work 384x faster when operating in the 512-dimensional fingerprint space versus the 200,000-dimensional tile space. In actuality, these fingerprints have to be stored in a hash table that keeps unique identifiers (UIDs) for the original tiles as keys mapped to the 512-value fingerprints as values. So the actual data compression rate depends greatly on the size of the UIDs.

We used a standard Python dictionary for this hash table, which had a total size of 400 MB. Compared to the original 8-bit integer tile dataset (of total size 9.14 GB), we achieve a data compression of 23x; compared to the pre-processed 32-bit float tile dataset (of total size 70 GB), we achieve a data compression of 175x.

2) Search: Assuming a model of computation where arithmetic calculations, including the Euclidean Distance between two fingerprints, can be done in $O(1)$ time, the time to calculate the distance between the query fingerprint and each candidate fingerprint is constant. Given N_{tiles}, to do this for the entire hash-map of candidate fingerprints, we require $O(N_{tiles})$ time. By maintaining a list of the smallest-k distances seen and the UID of the respective candidate, as we walk through the hash-map containing the candidate fingerprints, a comparison can be performed in constant time after each new distance calculation to see if the new distance is in the smallest-k. In total, maintaining this list requires $k * N_{tiles}$, which can be bounded by $O(N_{tiles})$. The time complexity to perform search for the nearest-k tiles is linear in the number of tiles.

Table II summarizes the actual results of the search time required to retrieve the UIDs of the nearest-k tiles to the query for several values of k.

TABLE II: Results for Search Time

k	Mean (sec)	Standard Deviation (sec)
10	3.24	0.20
5	3.28	0.15
2	3.05	0.17

As the table presents, as k increases there is no significant change in the time required to complete search.

3) Visual Similarity: The high rates of data compression and search speeds are only impressive if the algorithm is able to return images that look visually similar to a query image. We noticed that the fingerprints generated by ResNet-18 contain a remarkable amount of visual information from the tiles, and a high degree of visual similarity is obtained between any query image and the discovered k most similar images.

Please see Fig. 5 for some example queries for k = 10. Note that these examples come from randomly selected queries out of the 45,000 possible tiles. About 15 queries were made, and 7 were selected that had large diversity in color and motif, but all queries returned 10 visually similar images.

V. Conclusion

In this paper, we demonstrated that using pretrained convolutional neural networks to generate information-dense fingerprints for a large dataset of aerial images is a great way to speed up computation when doing a visual similarity search. This procedure produced a data compression rate of 175x, allowing a user on a personal computer to search for visually similar images to a query in a 70 GB dataset in a matter of a few seconds. And this computational speed and data compression does not make obvious compromises on the quality of image similarity.

Image similarity search is important for a variety of aerial image-based tasks, such as optimizing crop yield, managing infrastructure via satellite, predicting weather, etc. And as the total amount of satellite and aerial imagery grows in the coming years, it will become even more necessary to have ways to meaningfully compress this data - such as the method presented in this paper - so that it can be easily used for these tasks.

References

[1] Mutasem K. Alsmadi. "An efficient similarity measure for content based image retrieval using memetic algorithm". In: *Egyptian Journal of Basic and Applied Sciences* 4.2 (June 2017), pp. 112–122. ISSN: 2314-808X. URL: https://www.sciencedirect.com/science/article/pii/S2314808X16300628.

[2] A. Berman and L. Shapiro. "A Flexible Image Database System for Content-Based Retrieval". In: *Comput. Vis. Image Underst.* 75 (1999), pp. 175–195.

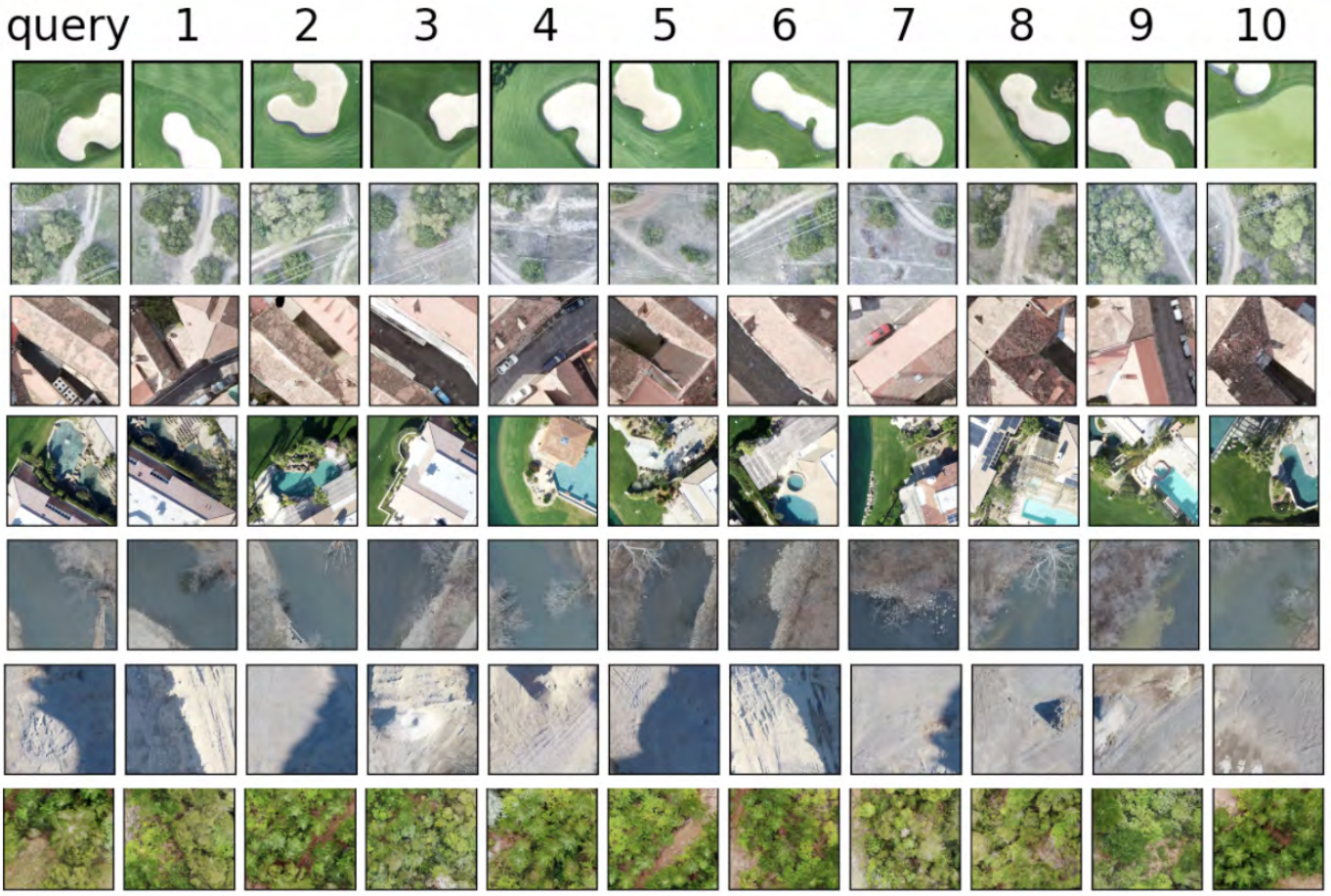

Fig. 5: Example of a query result where k = 10, returning the ten most similar images to the query images shown in the left most column. The visual similarity search was run on a few random queries chosen from all 45000 tiles, and some of these queries are shown in this figure, chosen to present a variety of colors and motifs.

[3] Hyun-Chong Cho et al. "Similarity evaluation in a content-based image retrieval (CBIR) CADx system for characterization of breast masses on ultrasound images". eng. In: *Medical physics* 38.4 (Apr. 2011). PMC3069991[pmcid], pp. 1820–1831. ISSN: 0094-2405. DOI: 10.1118/1.3560877. URL: https://doi.org/10.1118/1.3560877.

[4] Jia Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[5] *DroneDeploy Machine Learning Segmentation Benchmark*. URL: https://github.com/dronedeploy/dd-ml-segmentation-benchmark.

[6] S. Dubey. "A Decade Survey of Content Based Image Retrieval using Deep Learning". In: *ArXiv* abs/2012.00641 (2020).

[7] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[8] Ryan Keisler et al. "Visual search over billions of aerial and satellite images". In: *Computer Vision and Image Understanding* 187 (2019), p. 102790. ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2019.07.010. URL: https://www.sciencedirect.com/science/article/pii/S1077314219301067.

[9] Afshan Latif et al. "Content-Based Image Retrieval and Feature Extraction: A Comprehensive Review". In: *Mathematical Problems in Engineering* 2019 (Aug. 2019), p. 9658350. ISSN: 1024-123X. DOI: 10.1155/2019/9658350. URL: https://doi.org/10.1155/2019/9658350.

[10] Paolo Napoletano, Flavio Piccoli, and Raimondo Schettini. "Anomaly Detection in Nanofibrous Materials by CNN-Based Self-Similarity". eng. In: *Sensors (Basel, Switzerland)* 18.1 (Jan. 2018). s18010209[PII], p. 209. ISSN: 1424-8220. DOI: 10.3390/s18010209. URL: https://doi.org/10.3390/s18010209.