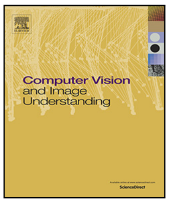
Computer Vision and Image Understanding 187 (2019) 102790

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu



Visual search over billions of aerial and satellite images

Ryan Keisler [*], Samuel W. Skillman, Sunny Gonnabathula, Justin Poehnelt, Xander Rudelis, Michael S. Warren

Descartes Labs, 1613 Paseo De Peralta, Santa Fe, NM, 87501, USA

ARTICLE INFO

Communicated by: Nikos Paragios

Keywords:
Visual search
Remote sensing
Machine learning

ABSTRACT

We present a system for performing visual search over billions of aerial and satellite images. The purpose of visual search is to find images that are visually similar to a query image. We define visual similarity using 512 abstract visual features generated by a convolutional neural network that has been trained on aerial and satellite imagery. The features are converted to binary values to reduce data and compute requirements. We employ a hash-based search using `Bigtable`, a scalable database service from Google Cloud. Searching the continental United States at 1-meter pixel resolution, corresponding to approximately 2 billion images, takes approximately 0.1 s. This system enables real-time visual search over the surface of the earth, and an interactive demo is available at https://search.descarteslabs.com.

1. Introduction

Visual search, the ability to find images that are visually similar to a query image, is a form of pattern recognition applied to the problem of image retrieval. An image retrieval system allows the user to find images that have some logical connection to a set of query parameters such as keywords, captions, or the in the case of content-based image retrieval (CBIR), the visual content of the image itself (Gudivada and Raghavan, 1995; Rui et al., 1999; Smeulders et al., 2000; Liu et al., 2007; Datta et al., 2008). CBIR systems use computer vision techniques (e.g. the SIFT descriptor, Lowe, 2004) to quantify texture, color, and shapes present in an image and thereby define image similarity. Convolutional neural networks (LeCun et al., 1995) trained on large image datasets (Deng et al., 2009) provide another means to extract rich feature representations of photographic imagery, and can be used in CBIR systems for visual search (Sharif Razavian et al., 2014; Babenko et al., 2014; Yue-Hei Ng et al., 2015; Tolias et al., 2015; Zheng et al., 2018).

Visual search is an increasingly common offering from large software companies. Some of these systems are focused on consumer products, e.g. Pinterest Lens (Jing et al., 2015; Zhai et al., 2017), while other search systems from Google, Microsoft, Baidu, or Yandex operate on photographic imagery more generally.

Aerial and satellite imagery provides another large dataset on which to build a visual search system, although there is much less prior work relative to visual search over photographic imagery. A significant milestone came in May 2016, when a team from Carnegie Mellon University released Terrapattern,[1] an interface for visual search over seven metropolitan areas. The work presented in this paper was largely inspired by Terrapattern. We sought to scale a visual search system beyond metropolitan areas to an entire country or the whole world.

Going beyond searching a single imagery layer, it is interesting to consider what visual search over all publicly available aerial and satellite imagery would look like. As organizations like Descartes Labs,[2] Google Earth Engine,[3] and Sentinel Hub[4] continue to build systems to ingest this flood of remote sensing data, what information do they store and in what format?

To begin with, there is the image data itself, which is currently at the petabyte scale and will cross into tens of petabytes in the coming decade. Beyond that, metadata such as the geographical extent of the image, the time at which the image was acquired, and the sensor used to acquire the image are clearly valuable; one can construct databases to efficiently index and search over these metadata. One can also store metadata related to the image data itself: basic summary statistics or the fraction of pixels covered in clouds. What is lacking is metadata encoding a deeper visual or semantic understanding of the image content. Ideally, this "visual metadata" would be significantly smaller than the original image data, say at least 100X smaller, and, like the spatial and temporal metadata described above, would encode some meaningful notion of "distance" between images. The feature vectors presented in this paper provide a step in that direction.

[*] Corresponding author.
E-mail address: ryan@descarteslabs.com (R. Keisler).

[1] http://www.terrapattern.com.
[2] https://descarteslabs.com.
[3] https://earthengine.google.com.
[4] https://www.sentinel-hub.com/.

https://doi.org/10.1016/j.cviu.2019.07.010
Received 3 September 2018; Received in revised form 7 April 2019; Accepted 26 July 2019
Available online 9 August 2019

How will such visual metadata be used for real applications? Possibilities include:

- interactive exploration of the earth's surface: a user wants to quickly search for visually similar objects, and that is the end goal.
- ground truth: a user could efficiently gather ground truth to be used to train a downstream computer vision model.
- pre-processing filter: a user needs to run a computer vision model over a large region but, in order to reduce computational expense, would like to limit the processing to only those images that might contain the object of interest.

This paper is organized as follows. In Section 2 we describe the two image datasets used in this work. In Section 3 we describe how we modified a convolutional neural network to extract useful binary features from this imagery, and in Section 4 we detail the two search methods used to search over these features. Finally, we show example search results in Section 5 and conclude with a summary of future research directions in Section 6.

2. Imagery

We used two imagery sources in the work presented here:

- **Aerial over USA** — We use aerial imagery from the National Agriculture Imagery Program.[5] (NAIP) and the Texas Orthoimagery Program[6] These datasets provide coverage of the lower 48 states of the United States.
- **Landsat 8 over Earth** — We created a custom, pan-sharpened, 15-meter global composite (Warren et al., 2016) using data from Landsat 8, one of the workhorses of NASA's satellite-based earth observation program.

We built a visual search system for each of these datasets, but the systems differ in how the features were generated, as described in Section 3.

We maintain the projection of both datasets in the 60 Universal Transverse Mercator (UTM) projections. These conformal projections preserve angles and have weak area distortion within a UTM zone. Both of these properties are desirable for visual search, as we would only be making the search problem more difficult if we used a projection that introduced strong distortion of angles or areas.

We chunked this imagery into square image tiles, 128 pixels on a side. Adjacent tiles overlap by 64 pixels in the vertical and horizontal directions. While this overlapping tiling scheme requires two times more data than a non-overlapping scheme, it gives any object the chance to be roughly centered in at least one image tile.

We used the Descartes Labs Platform,[7] to access and process this imagery. This platform allows the user to search for the existence of imagery, quickly load that imagery into memory in the form of a python `ndarray` and parallelize computations over tens of thousands of CPUs.

3. Feature vectors

We start with a convolutional neural network with a 50-layer ResNet architecture (He et al., 2016), pre-trained on ImageNet.[8] This network is conveniently provided pre-trained in the Keras[9] deep learning package, which we used with a Tensorflow back-end.[10]

In our initial experiments we used the "out of the box" features generated in the last few layers of the ImageNet-trained network. These layers give surprisingly good similarity search results for satellite imagery, despite being trained on photographic imagery of animals, plants, vehicles, etc. But in the end we modified the network in two ways, as described below.

An overview of the feature-generation process is shown in Fig. 1.

3.1. Binary features

We decided that we ultimately wanted to search over binary features, due to their smaller data footprint. To that end, we encouraged the network to make floating-point features very close to 0.0 or 1.0 at the layer of interest by injecting noise with an amplitude comparable to the width of the layer's activation function (Salakhutdinov and Hinton, 2009). The noise is injected during training but not during inference. The network learns to make almost-binary features at this layer; otherwise the noise destroys the information that the layer is trying to pass on. Finally, we binarize the floating-point features by thresholding at 0.5.

3.2. Customizing for aerial and satellite imagery

We customized this network to work with each source of aerial and satellite imagery. For NAIP, we followed the strategy used by Terrapattern and fine-tuned the network to classify images into 130 object classes from OpenStreetMap (OSM), such as parking lots or golf courses (Zheng et al., 2016). Prior to fine-tuning, we added two fully-connected layers to the end of the network, and extracted 512 binary features from one of them (Vo and Hays, 2019). We emphasize that this supervised learning step is not used to produce a network that is good at identifying particular object classes (although it does ultimately do better on these classes), but rather to create a network that produces features that are useful for generic visual search on aerial and satellite imagery.

For Landsat 8, the OSM classes were less useful due to the much coarser spatial resolution, so we took an unsupervised approach instead. Specifically we used an autoencoder to compress the 2048 features from the penultimate layer of ResNet50 into 512 binary features using a 2048-1024-512-1024-2048 autoencoder architecture. The autoencoder was trained by passing Landsat 8 imagery through the original ResNet50 network.

We also note that we do not directly optimize on the binary features used for look-up, unlike many feature hashing approaches (Norouzi et al., 2012). Instead, we rely on the fact that images of similar classes do map to similar outputs of late layers in the neural net.

At the end of this process, we have mapped the original $128 \times 128 \times 3$ 8-bit image (or 16-bit image in the case of Landsat 8) to 512 bits. This corresponds to a 768X (1536X) reduction in data size. These binary features form a compact representation of the visual information present in each image.

We pre-compute the feature vectors for all of the tiles in each dataset: approximately 2 billion tiles for NAIP and 200 million tiles for Landsat 8. We distributed this computation across tens of thousands of CPUs in the Google Cloud Platform.

We store the binary features vectors in `Redis`, an in-memory data store. The data is stored in key:value pairs. The keys are strings that uniquely identify each image tile, and the values are the corresponding 512-bit feature vectors, each stored as 64 bytes.

4. Search

Now that we have feature vectors, how do we search for similar vectors?

We first define a distance between vectors. We use the Hamming distance: the number of bits that differ between the two binary vectors. A small distance is associated with visual similarity.

Next we need to find the k-nearest vectors to a query vector. We use two methods: an exact, brute-force search, and an approximate, hash-based search, as described below.

[5] https://lta.cr.usgs.gov/NAIP.
[6] https://tnris.org/2015-statewide-orthoimagery-project/.
[7] http://docs.descarteslabs.com.
[8] http://www.image-net.org.
[9] https://keras.io.
[10] https://www.tensorflow.org/.

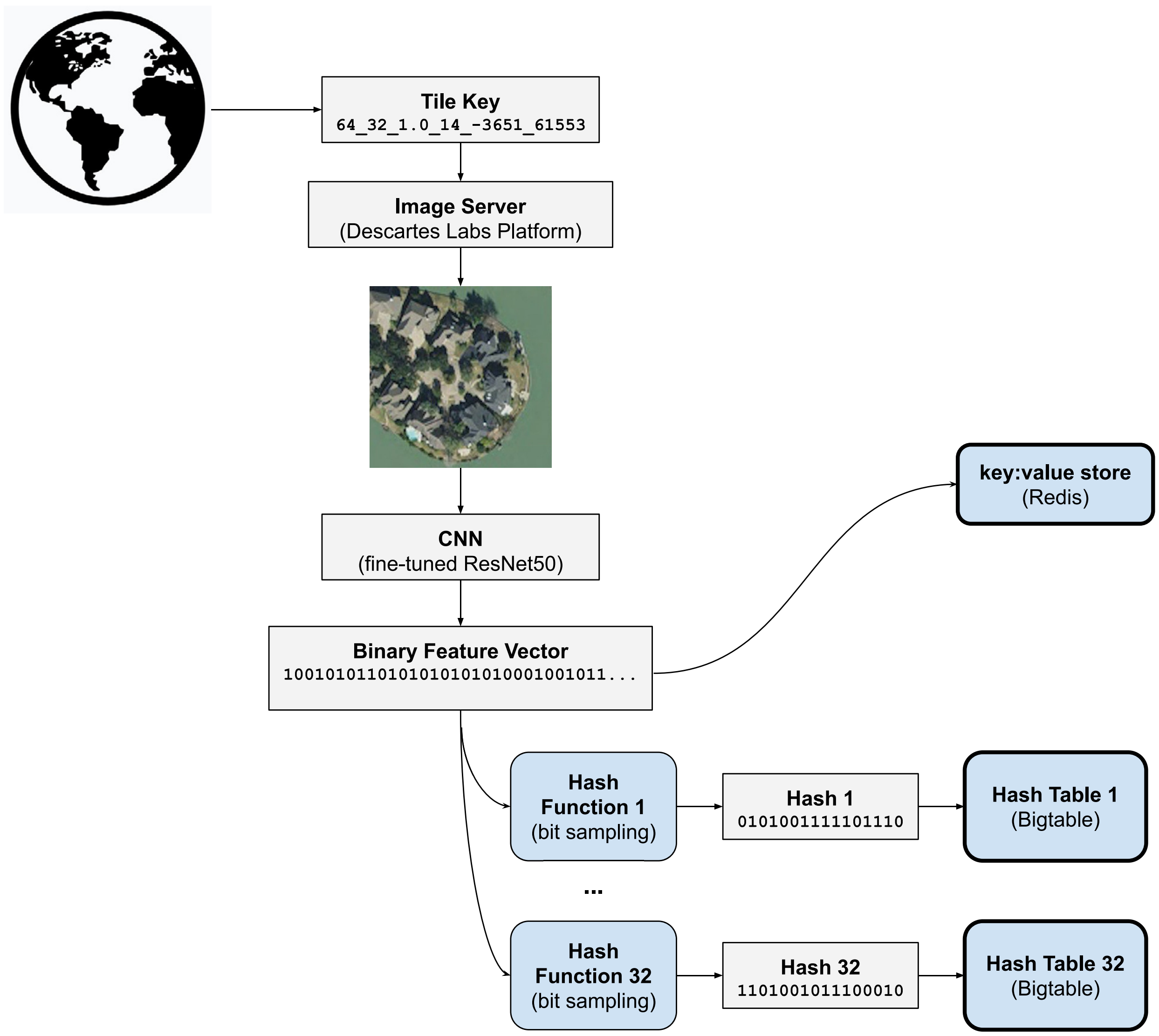


Fig. 1. The feature-generation architecture. The earth is tiled into a set of tiles, and aerial or satellite imagery is retrieved for each tile using the Descartes Labs Platform. Each image is passed through a convolutional neural network (CNN) that has been fine-tuned for aerial or satellite imagery. The resulting feature vector is binarized and stored in a simple key:value store, as well as a set of 32 hash tables used for approximate nearest neighbor search.

4.1. Direct search

By taking advantage of low-level instructions for comparing bits (`XOR`) and counting bits (`__builtin_popcountll`), we can perform a direct, exact, brute-force search over the 200 million Landsat 8 images. Search for similar vectors is a nearest-neighbors search in the 512-bit space. This works as follows.

- A query image tile is selected.
- We retrieve the binary feature vector for this image tile using `Redis`.
- We compute the Hamming distance between the query vector and all vectors using a pre-computed binary file that contains all binary feature vectors. This sub-program is written in C, using the GCC compiler with `-Ofast` optimization.
- We keep the k-nearest feature vectors and determine their corresponding images tiles using a second index file.

This direct, brute-force method searches across the 200 million Landsat 8 images in approximately 2 s using a single thread. While an individual query is single-threaded, we run a server (16 virtual CPUs, Haswell architecture, 32 GB of RAM) which allows for up to 16 concurrent queries that share the feature vector data in memory.

4.2. Hash-based search

The NAIP dataset is approximately ten times larger than the Landsat 8 dataset, with approximately 2 billion images. While the direct search method works, it is too slow for interactive use. Instead we use an approximate, hash-based search method. The general idea is to use hash functions to quickly return a list of candidate neighbors, and then run a brute-force search on that relatively small list of candidates.

More specifically, we use bit sampling, a simple form of locality-sensitive hashing. We use a family of 32 hash functions, each of which simply selects a subset of 16 bits from the full, 512-bit feature vector. Within each hash table, the keys are specific realizations of those 16 bits, e.g. 0101001111101110, and the values are lists of strings (image tile ids) corresponding to tiles that contain those specific realizations of those 16 bits. We store the hash tables in `Bigtable`, a scalable NoSQL database service from Google Cloud. An overview of the hash-based search system is shown in Fig. 2 and works as follows.

- A query image tile is selected.
- We retrieve the binary feature vector for this image tile using `Redis`.
- We apply each of the 32 hash functions to this feature vector, each of which returns a set of candidate neighbors.

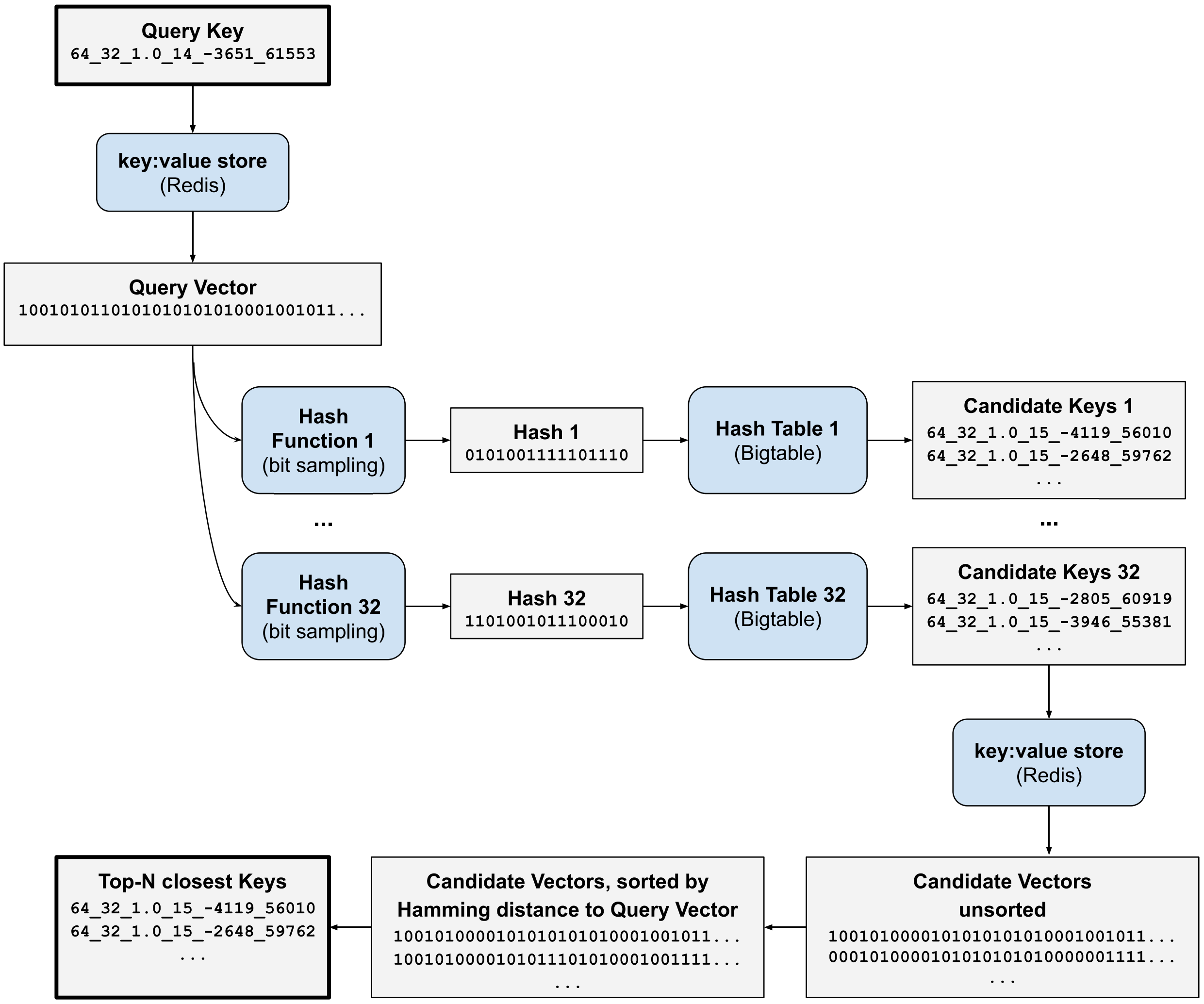


Fig. 2. The hash-based search architecture. A query image is selected by the user, and the binary feature vector for this image is retrieved from a key:value store. After hashing this feature vector through a set of 32 hash tables, we retrieve a set of candidate neighbors. Finally, these candidates are sorted by distance to the original query vector, and the top-N image keys are returned.

- We compute the unique set of candidate neighbors and fetch their corresponding feature vectors using `Bigtable`.
- We perform a brute-force distance calculation between the query vector and the candidate vectors.
- We keep the k-nearest feature vectors and their corresponding images tile ids.

The search takes approximately 0.1 s per query over approximately 2 billion images.

5. Results

5.1. Search results

Search results from NAIP are shown in Figs. 3 and 4. These search queries span a wide range of object types and landscapes, from industrial infrastructure to subtle natural features.

We expect to have relatively good search results for many of the queries shown here (e.g. wind turbines) because the query object was one of the 130 OSM classes used to fine-tune the NAIP network. However, the system also provides good search results over generic images that were not part of the supervised learning phase, such as the sinuous waterways or the trees changing color.

Search results from Landsat 8 are shown in Fig. 5. The Landsat 8 features were generated without any supervised learning or reference to OSM classes, so the search results demonstrate generic visual search.

We show low-quality search results in Fig. 6. While these results do contain images that are visually similar to the query image, they are failures in the sense that they do not contain the query object class. This highlights the fact that there is a balance to strike between providing good search results for common object classes and providing good search results for generic, class-agnostic search.

Finally, we have made a quantitative assessment of search quality for 20 query types. For each query we have manually recorded the precision (fraction of true positives) out of the top-30 search results. We have not attempted to assess recall, given the lack of unified, comprehensive ground truth for these object types.

Our top-30 precision results are as follows. For the 10 object types that were present during the network fine-tuning stage, the average top-30 precision is 86%. The individual results for this category are: airplanes (36%), baseball diamonds (100%), football fields (50%), golf courses (100%), highway overpasses (70%), marinas (100%), parking lots (100%), railyards (100%), storage tanks (100%), and taxiways (100%).

For the 10 object types that were not present during the fine-tuning stage, the average top-30 precision is 58%. The individual results for this category are: barges (10%), CAFOs (97%), center pivot

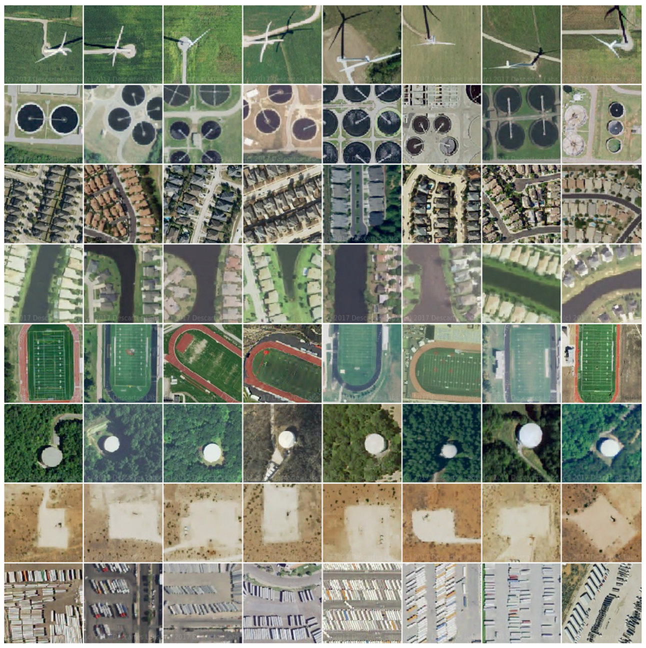


Fig. 3. Example search results of man-made features from NAIP. The leftmost image of each row is the query image, and the rest of the row shows the top seven results. These queries show wind turbines, water treatment plants, suburban homes, waterfront homes, stadiums, isolated storage tanks, oil and gas sites, and containers.

irrigation (97%), cul-de-sacs (33%), dry docks (3%), feed lots (50%), roller coasters (3%), row housing (90%), suburban homes (100%), and shipping containers (93%). As one would expect, the search results are generally of higher quality for object types that were present during the network fine-tuning stage (86% vs 58%), although the precision of the new object types (58%) is still high enough to be useful for some applications, e.g. efficiently gathering ground truth to be used to train a downstream computer vision model.

5.2. User interface

We built a browser-based user interface on top of the search back-end described above. This interface was released publicly in March 2017 at https://search.descarteslabs.com. We encourage the reader to try it out, but the basic structure is: a user clicks anywhere on the map, the top 1000 search results are displayed in thumbnail image form, and their geographical locations are shown on a map.

A screenshot of the user interface is shown in Fig. 7. This particular search query highlights the benefit of showing the geographical distribution of the search results. The user clicked on a single example of piñon-juniper woodlands, a landscape common to the southwestern United States, and the full geographical extent of piñon-juniper woodlands is visible immediately.

6. Conclusion and future directions

We have presented a system for visual search over billions of aerial and satellite images. Binary features encoding visual information are extracted with a convolutional neural network that has been trained on aerial and satellite imagery. Using a hash-based search method we are able to search over approximately 2 billion images in 0.1 s. An interactive demo is available at https://search.descarteslabs.com.

We see three clear future directions for extending the work presented here.

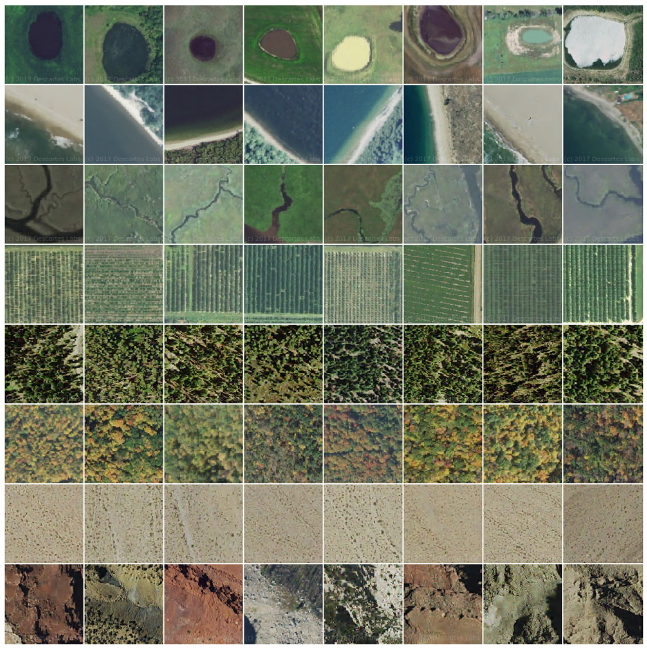


Fig. 4. Example search results of natural features from NAIP. The leftmost image of each row is the query image, and the rest of the row shows the top seven results. These queries show ponds, beaches, sinuous rivers, orchards, forest, trees changing color, desert, and rocky outcroppings.

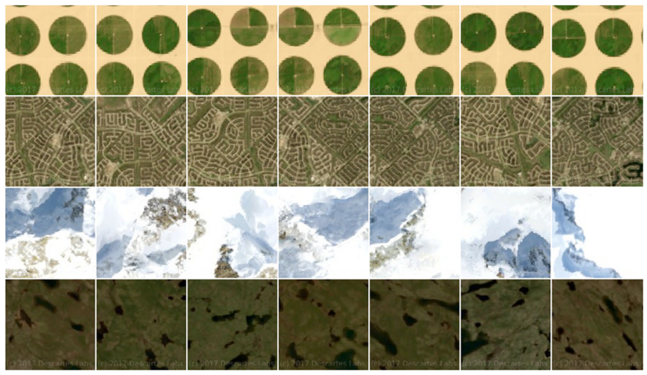


Fig. 5. Example search results from Landsat 8. The leftmost image of each row is the query image, and the rest of the row shows the top seven results. These queries show center-pivot irrigation, suburbs, snow-capped mountains, and arctic ponds.

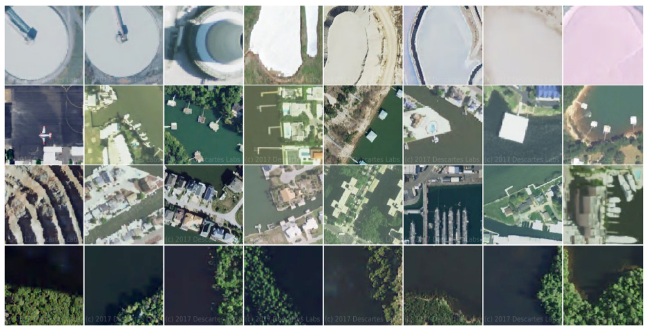


Fig. 6. Low-quality search results for a water treatment plant, a small airport, a mine, and a forest in shadow. While these results do contain images that are visually similar to the query image, they are failures in the sense that they do not contain the query object class.

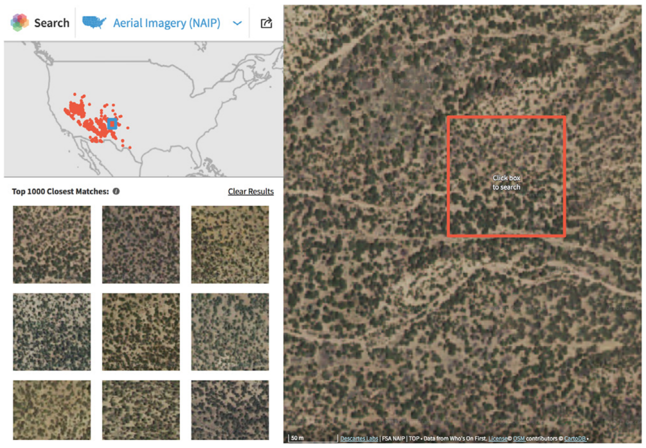


Fig. 7. The user interface shown here is publicly available at https://search.descarteslabs.com. The query image and its local context are shown in the **right** panel. The geographical distribution and thumbnail images of the search results are shown in the **top left** and **bottom left** panels, respectively. In this particular search, the geographical extent of piñon-juniper woodlands, common to the southwestern United States, is immediately visible.

- **Multi-scale search** – The current system searches at one spatial scale, namely square images that are 128 pixels across. Instead we would like to enable search over a handful of spatial scales, both smaller and larger than 128 pixels across.
- **Geospatial filtering** – The current system searches across all images that have been indexed, regardless of their geographical location. In the future we would like to enable geospatial filtering, e.g. only return results from Japan.
- **Temporal filtering** – The current system searches across a single image layer. Although the images within this layer were acquired at different times, that temporal information has not been used in the search. In the future we would like to enable temporal filtering, e.g. only return results that were acquired in May 2018.

With these changes, it should be possible, for example, to process and search over all Landsat 8 scenes, rather than searching over a single Landsat 8 composite image. And while the work presented here used RGB imagery, the underlying technique should work with non-visual bands (infrared, SWIR, SAR, *etc.*) or a different number of bands. Such a system should provide a more flexible way to compactly encode visual information for large collections of aerial and satellite imagery.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.cviu.2019.07.010.

References

Babenko, A., Slesarev, A., Chigorin, A., Lempitsky, V., 2014. Neural codes for image retrieval. In: European Conference on Computer Vision. Springer, pp. 584–599.

Datta, R., Joshi, D., Li, J., Wang, J.Z., 2008. Image retrieval: Ideas, influences, and trends of the new age. ACM Comput. Surv. 40 (2), 5.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L., 2009. Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, pp. 248–255.

Gudivada, V.N., Raghavan, V.V., 1995. Content based image retrieval systems. Computer 28 (9), 18–22.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778.

Jing, Y., Liu, D., Kislyuk, D., Zhai, A., Xu, J., Donahue, J., Tavel, S., 2015. Visual search at pinterest. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, pp. 1889–1898.

LeCun, Y., Bengio, Y., et al., 1995. Convolutional networks for images, speech, and time series. In: The handbook of brain theory and neural networks, Vol. 3361. (10), p. 1995.

Liu, Y., Zhang, D., Lu, G., Ma, W.-Y., 2007. A survey of content-based image retrieval with high-level semantics. Pattern Recognit. 40 (1), 262–282.

Lowe, D.G., 2004. Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. 60 (2), 91–110.

Norouzi, M., Fleet, D.J., Salakhutdinov, R.R., 2012. Hamming distance metric learning. In: Advances in Neural Information Processing Systems. pp. 1061–1069.

Rui, Y., Huang, T.S., Chang, S.-F., 1999. Image retrieval: Current techniques, promising directions, and open issues. J. Vis. Commun. Image Represent. 10 (1), 39–62.

Salakhutdinov, R., Hinton, G., 2009. Semantic hashing. Internat. J. Approx. Reason. 50 (7), 969–978.

Sharif Razavian, A., Azizpour, H., Sullivan, J., Carlsson, S., 2014. CNN features off-the-shelf: an astounding baseline for recognition, In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 806–813.

Smeulders, A.W., Worring, M., Santini, S., Gupta, A., Jain, R., 2000. Content-based image retrieval at the end of the early years. IEEE Trans. Pattern Anal. Mach. Intell. (12), 1349–1380.

Tolias, G., Sicre, R., Jégou, H., 2015. Particular object retrieval with integral max-pooling of CNN activations. arXiv preprint arXiv:1511.05879.

Vo, N., Hays, J., 2019. Generalization in metric learning: Should the embedding layer be embedding layer?. In: 2019 IEEE Winter Conference on Applications of Computer Vision WACV, IEEE, pp. 589–598.

Warren, M.S., Skillman, S.W., Chartrand, R., Kelton, T., Keisler, R., Raleigh, D., Turk, M., 2016. Data-intensive supercomputing in the cloud: Global analytics for satellite imagery. In: Data-Intensive Computing in the Clouds (DataCloud), 2016 Seventh International Workshop on. IEEE, pp. 24–31.

Yue-Hei Ng, J., Yang, F., Davis, L.S., 2015. Exploiting local features from deep networks for image retrieval, In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 53–61.

Zhai, A., Kislyuk, D., Jing, Y., Feng, M., Tzeng, E., Donahue, J., Du, Y.L., Darrell, T., 2017. Visual discovery at pinterest. In: Proceedings of the 26th International Conference on World Wide Web Companion. International World Wide Web Conferences Steering Committee, pp. 515–524.

Zheng, L., Yang, Y., Tian, Q., 2018. SIFT meets CNN: A decade survey of instance retrieval. IEEE Trans. Pattern Anal. Mach. Intell. 40 (5), 1224–1244.

Zheng, L., Zhao, Y., Wang, S., Wang, J., Tian, Q., 2016. Good practice in CNN feature transfer. arXiv preprint arXiv:1604.00133.