Journal, Vol. XXI, No. 1, 1-5, 2018
Additional note

Reverse Image Search for Scientific Data within and beyond the Visible Spectrum

Flavio H. D. Araujo[1,2,3,4,a], Romuere R. V. Silva[1,2,3,4,a], Fatima N. S. Medeiros[3],Dilworth D. Parkinson[2], Alexander Hexemer[2], Claudia M. Carneiro[5], Daniela M. Ushizima[1,2,*]

Abstract
The explosion in the rate, quality and diversity of image acquisition instruments has propelled the development of expert systems to organize and query image collections more efficiently. Recommendation systems that handle scientific images are rare, particularly if records lack metadata. This paper introduces new strategies to enable fast searches and image ranking from large pictorial datasets with or without labels. The main contribution is the development of `pyCBIR`, a deep neural network software to search scientific images by content. This tool exploits convolutional layers with locality sensitivity hashing for querying images across domains through a user-friendly interface. Our results report image searches over databases ranging from thousands to millions of samples. We test `pyCBIR` search capabilities using three convNets against four scientific datasets, including samples from cell microscopy, microtomography, atomic diffraction patterns, and materials photographs to demonstrate 95% accurate recommendations in most cases. Furthermore, all scientific data collections are released.

Keywords
Reverse image search — Content-based image retrieval — Scientific image recommendation — Convolutional neural network

[1] *University of California, Berkeley, CA, USA*
[2] *Lawrence Berkeley National Laboratory, Berkeley, CA, USA*
[3] *Federal University of Ceará, Fortaleza, CE, Brazil*
[4] *Federal University of Piauí, Picos, PI, Brazil*
[5] *Federal University of Ouro Preto, Ouro Preto, MG, Brazil*
[*] **Corresponding author**: dushizima@lbl.gov
[a] **Contributed equally to this work.**

Contents

1. Introduction

With the increased availability of large data repositories, a substantial amount of time is spent searching for pictures, which is seldom an efficient procedure. Recent reports [1, 2] point out that the growth in data size, rates and variety is significant; they also suggest that scientific data will grow twice as quickly as any other sector, yet less than 3% of that data will be tagged in a meaningful way. Several imaging facilities will soon be generating 1 to 50 petabytes of data per year, which poses several challenges: (a) inadequate or insufficient meta-data describing experimental records; (b) the impracticality of manual curation of massive datasets; and (c) the lack of tools adapted to the new data acquisition modes.

Photo organizers that rely on curated data have improved to include operations such as sorting and categorization by

dates or media types, metadata and other user annotations. However, manual insertion of metadata is seldom achievable at scale, and even impossible in some scenarios, such as with high-throughput imaging instruments. There is a critical need to create automated and accurate methods to organize, query and retrieve unlabeled images, since "everyone searches all the time" [3]. Building upon recent work on deep learning [4, 5, 6] and the ability to create annotations for unlabeled datasets from curated ones [7], new expert systems promise to change the human experience from hardly relevant retrievals to broadly useful (over 85% accuracy) results. As an example, Google Photos has provided automated and custom-labeling features since 2015, so that users can quickly organize large collections [8], with high retrieval accuracies for face detection. One of the main challenges in image recognition is to perform tasks that are easy for humans to do intuitively, but hard to describe formally [4]. For example, domain scientists [9], who are visually trained to identify complex patterns from their experimental data, but many times are unable to describe mathematically the primitives that construct the motif.

By using methods to identify patterns from pictures, recommendation systems, also known as reverse image search tools, represent an excellent opportunity for data reduction, in which the imaging acquisition, data collection and storage strategies are integrated and tailored toward a desired pattern. Such systems could support scientists in adjusting experimental parameters fast enough for optimal data collection, combating a major problem at imaging facilities where overwhelming amounts of irrelevant data are collected daily.

The main contributions of this work are as follows: (a) Development of a new recommendation system for visual image search with an inferential engine that learns compact signatures for recovering images within massive datasets; it uses approximate ranking, and includes 10 schemes to measure distance based on different sets of features from labeled and/or unlabeled images; (b) Deployment of `pyCBIR` as an interactive system to enable a generic user to search image collections from diverse domains through an intuitive graphical user interface in Qt© as illustrated in Figure 6; (c) Evaluation of three Convolutional Neural Networks (CNN) implementations available within `pyCBIR`, describing scientific problems that rely upon deep and complex networks, but also shallow architectures. Because `pyCBIR` allows generalization of processing pipelines, it is quickly extensible to different training datasets; here we test data collections from four science domains, and report on accuracy and time consumption given different architectural choices; (d) Establishment of reproducible work, containing both codes for benchmarks and tests using image repositories publicly available, and software based on open-source tools.[1].

First, we discuss previous work on CBIR in the context of the proposed expert system `pyCBIR` in Sec. 2, including capabilities and benefits of `pyCBIR` to a generic user. Sec.3 explains the four data collections of images across domains. Sec.4 presents detailed information on computational methods tested in Sec.5, which focuses on results of applying different learning strategies to the different datasets. Finally, Sec. 6 evaluates the impact of `pyCBIR`, including perspectives on future requirements.

[1] Source codes/data to be published upon paper acceptance at camera.lbl.gov

2. Related work

The term content-based image retrieval (CBIR) was introduced in 1992 by Kato [10, 11], and it has been associated with systems that provide image matching and retrieval for queries performed by visual example. A quarter of century later, most image retrieval systems available for scientific image search still rely on keyword-based image retrieval [12], although most of the image collections generated by humans lack proper annotations [13]. Efforts to optimize image search employing CBIR systems [14, 15, 16] exploit computer vision and machine learning algorithms to represent images in terms of more compact primitives. Given an image as an input query, instead of keywords or metadata, such an approach allows matching samples by similarity.

Since 2003, the Bag-of-Words (BoW) model has been predominantly viewed as the state-of-the-art in CBIR, augmented by 13 years of Scale Invariant Feature Transform (SIFT) methods [17]. In addition, CBIRs combining textural descriptors, such as Gabor and Fourier [18, 19], GLCM [19], continue to be broadly used to organize natural images, and made use of similarity search based on the Euclidean distance. Tsochatzidis et al [20] extend previous ideas and include exploration of Histogram of Oriented Gradients (HOG) descriptors combined with the Earth Mover's Distance to recover mammograms based on similarity. Moreover, image retrieval methods have advanced in two main directions: SIFT-based and CNN-based, with promising improved accuracy when combining CNN and SIFT features [17]. To the best of our knowledge, there are no software tools that allow combining both strategies for CBIR tasks. Several free engines for CBIR are thriving at natural and biomedical image organization, and e-commerce tasks [21, 22], but underlying codes for end-to-end workflow remain closed, and are seldom generalizable to other scientific image collections.

2.1 CNN and visual recognition

These are significant strides toward automating image catalogs, and motivates our efforts to construct convolutional neural networks (CNN) using Google TensorFlow to organize scientific data. TensorFlow [23] is an open-source software library for Machine Intelligence that presents advantages regarding flexibility, portability, performance, and compatibility to GPU. In order to deliver high-performance C++ code, TensorFlow uses the Eigen linear algebra library in addition to CUDA numerical libraries, such as cuDNN to accelerate core computations and scale to large datasets.

A typical CNN pipeline is shown in Figure 1, consisting of three main "neural" layers: convolutional layers, pooling

layers, and fully connected layers. This algorithm requires two stages for training the network: (a) a forward stage, which represents the input image in each layer and outputs a prediction used to compute the loss cost based on the curated data (labeled samples), and (b) a backward stage, which computes the gradients of layer parameters to drive the cost function to very low values [24, 4].

By exploring CNN algorithms that automatically extract features at multiple levels of abstraction from large datasets, CBIR systems can benefit from complex non-linear functions that map unprocessed input data to the results, bypassing human-designed characterization reliant on domain knowledge. Wan et al. [25] investigated different deep learning frameworks for CBIR when applied to natural images, such as the ILSVRC2012 (ImageNet Large Scale Visual Recognition Challenge) dataset. That paper reported mean average precision of 0.4711 using a massive image collection with 10,000,000 hand-labeled images depicting 10,000+ object categories as training.

2.2 Expert systems for material recognition

Apart from natural scenes, recent work on recognizing material categories from images [26, 27, 28, 29] includes experiments using the Flickr Material Dataset (FMD) and/or the Materials in Context Database (MINC). Sharan et al. [28] explored low and mid-level features, such as color, SIFT, HOG, combined with an augmented Latent Dirichlet Allocation model under a Bayesian generative perspective, achieving 44.6% accurate recognition rate on FMD. Using a CNN-based feature extraction mechanism, Bell et al. [26] designed materials recognition frameworks employing two Caffe-based [30] architectures: AlexNet and GoogLenet trained on materials patches from the MINC database, achieving accuracy of 79.1% and 83.3%, respectively. Additionally, attempts to train CNNs with public datasets such as FMD were less favorable since FMD alone contained a small amount of samples. Moreover, those authors noticed that the AlexNet trained on the MINC database outperformed FMD classification, showing 66.5% accuracy.

In [29], different authors continued investigating material categories, now tuning a Caffe VGG-D pre-trained using two datasets: MINC and ILSVRC2012 [31] for image classification. They resort to a customized feature selection and integration method to concatenate the values of the 7^{th} layer for both networks. Finally, the integrated features were input to a support vector machine (SVM) with radial basis function kernels for training and testing: they improved accuracy to 82.3% for the FMD images.

Most CBIR or materials classifiers rely exclusively on customized features and/or large amounts of labeled data, which is a limiting factor to the usability of software tools. Another limitation is the availability of these tools as well as datasets used to create and test them. Our paper describes the design of `pyCBIR`, with CNNs trained both on specific datasets as well as the ImageNet [32] following an Inception-ResNet-v2 model, which originally created probability maps for 1,000 classes. Another key idea is to focus on scientific data exploration, including generalizable CNNs models for CBIR tasks in different domain areas.

Figure 2 introduces our recommendation tool, `pyCBIR`, which provides a ranking system to enable humans to quickly interact with images, offering mechanisms to incorporate labeled data when available. We aim to provide a single workflow that allows to use different feature vectors, including those from CNNs, to construct image signatures. `pyCBIR` delivers the classification accuracy supported by Lenet [33] and Inception-ResNet-v2 [34] architectures, exploiting optimized routines from TensorFlow [23]. `pyCBIR` uses python to model the dataflow graph, which is responsible for coordinating the execution of operations that transform inputs into ranked image samples, whose labels can be used in the calculation of uncertainty values.

The next section describes experiments with `pyCBIR` on materials databases such as FMD, as well as novel databases of scientific images, and different CNN architectures with increasing levels of complexity.

3. Description of Scientific Databases

We have explored deep learning and considered samples from diverse imaging systems, varying in terms of their electromagnetic wave interaction with the samples and space scale. These sample collections require nontrivial mathematical descriptions to build indices to recover relevant results, taking into account image composition and structure. We summarize the four databases under investigation in Table 1, describing the main characteristics shown by fibers, films, cells and other materials.

3.1 Fiber profiles

The fiber database consists of volume cross-sections, based on hard X-ray microtomography (microCT) from ceramic matrix composites (CMC). This imaging technique allows for inspection of the structural properties and quality control [35, 36].

Figure 1 illustrates a CMC sample cross-section, with 1mm in diameter and 55mm length, reinforced with hundreds of ceramic fibers of approximately 10μm diameter. Each fiber is coated with a boron nitride layer, which has a lower X-ray absorption coefficient, therefore fiber cross-sections appear as dark rings. Frequently, the 3D images are examined manually, slice by slice (2D), in order to identify defects. As an alternative, there is increasing interest in automation by designing "inspecting bots" that detect mechanical deformations and sort the experimental instances (3D image stacks) according to the structural damage.

A major demand is the ability to perform pattern ranking which can steer data management needed by beamline scientists. The first step in this direction is illustrated in Figure 1, where cross-section are scrutinized for damaged fiber struc-

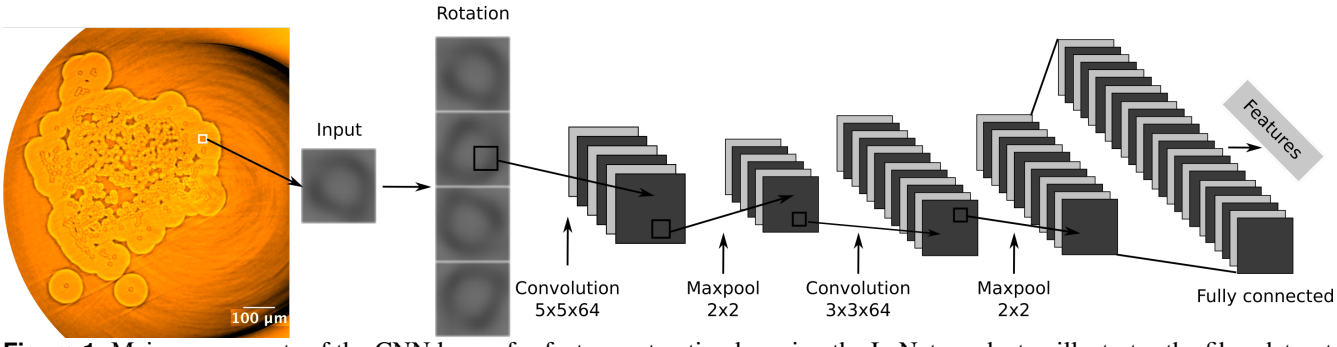


Figure 1. Main components of the CNN layers for feature extraction by using the LeNet: each step illustrates the fiber dataset transformation, although the LeNet architecture applies to the other datasets.

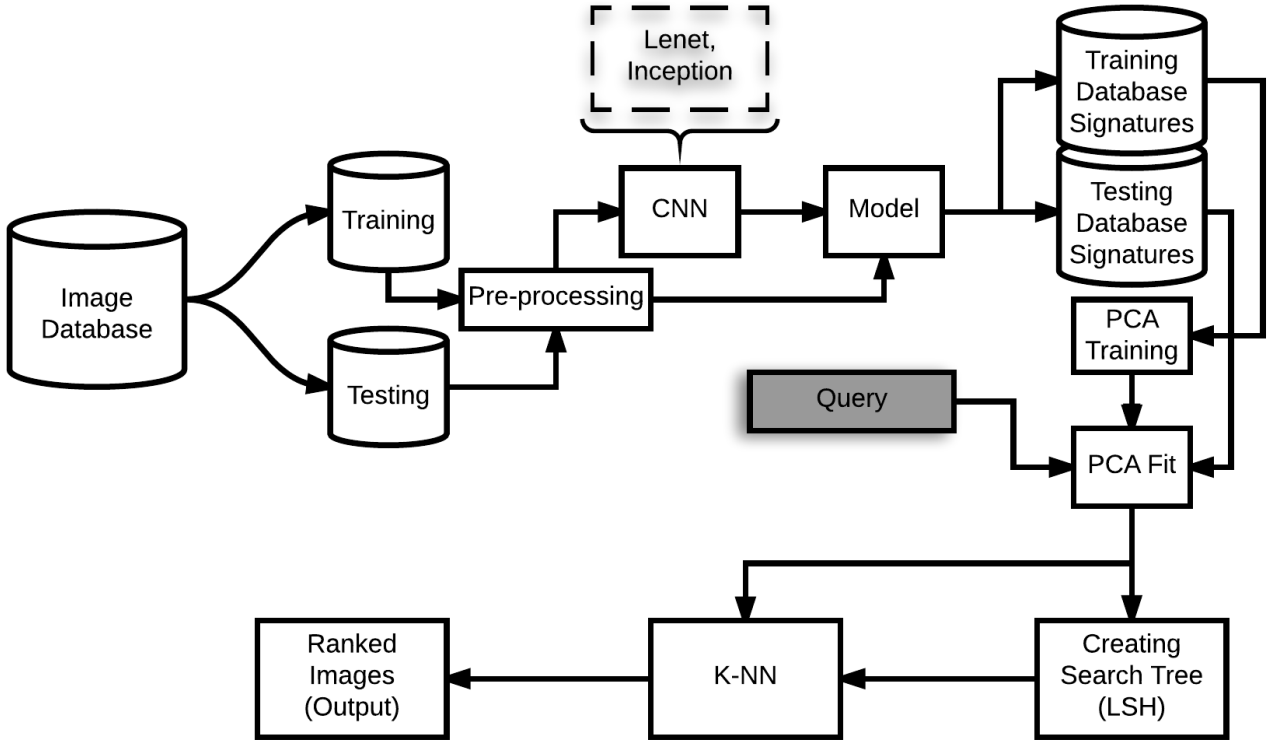


Figure 2. Diagram showing how data moves through `pyCBIR` modules during reverse image search: gray box emphasizes the starting point given a trained state.

tures, commonly used as a guiding pattern to detect sample deformation [7, 36].

Our paper reports results on labeled samples that went through a triage including automated segmentation methods based on interactive computer vision [37, 38, 39] and visual inspection by domain scientists. To the best of our knowledge, all the images contain accurate labels, which are used to determine the success rate of the CNNs, with half of the samples with fibers and the other half containing areas with no fibers. Figure 1 shows how to obtain these fiber profile images.

3.2 GISAXS

Grazing Incidence Small Angle X-ray Scattering (GISAXS) is a method for characterizing the nanostructural features of materials, especially at surfaces and interfaces, which would otherwise be impossible using standard transmission-based scattering techniques. As a surface-sensitive tool for simultaneously probing the electron density of the sample, this imaging modality supports measurements of the size, shape, and spatial organization of nanoscale objects located at the top of surfaces or embedded in mono- or multi-layered thin film materials.

Individual GISAXS images, as illustrated in Figure 3, serve as static snapshots of nanoscale structure, while succes-

Table 1. Scientific data under investigation: experimental specifications, number of samples and respective image size.

Specimen	Modality	#Samples and Size	Target data analysis
Ceramic composite	X-ray microCT	1,013,550 16×16	Detection of fiber profiles from 3D cross-sections. Sec. 3.1. Fig. 1.
Thin films	GISAXS	4,024,789 100×100	Classification of simulated scattering patterns into space groups. Sec. 3.2. Fig. 3.
Pap smears	Light microscopy	3,393 100×100	Inspection of cervical cell morphology for cancer detection. Sec. 3.3. Fig. 4.
Materials patterns	Photography public DB	1,000 512×384	Inspection of molecular structure with 2D orientation classification. Sec. 3.4. Fig. 5.

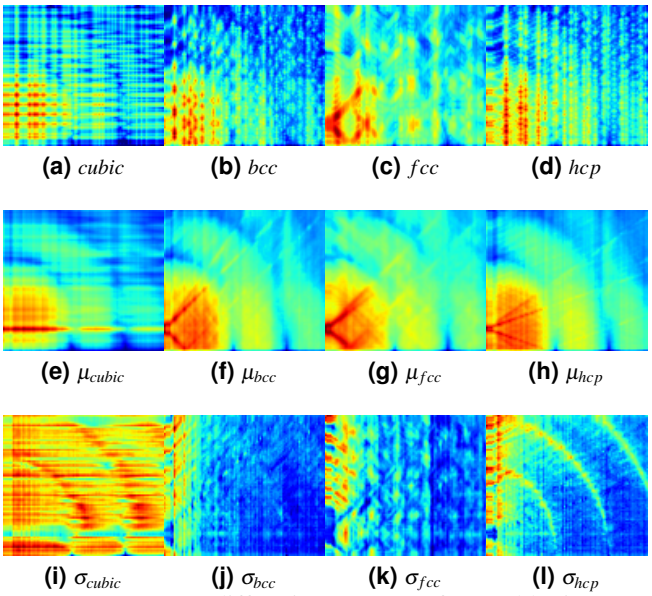


Figure 3. GISAXS diffraction patterns of crystal lattices: (a-d) a sample of each structure class, (e-h) images representing the samples average (μ) and (k-o) the standard deviation (σ) of a subset of 1,000 random selected samples from each class.

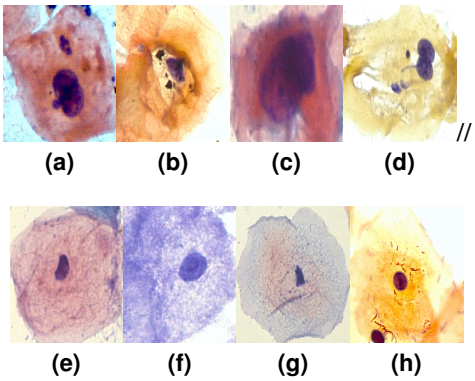


Figure 4. Examples of cell images from CRIC database: abnormal cells (a-d) and normal cells (e-h).

sive images provide a means to monitor and probe dynamical processes. Although microscopy techniques provide valuable local information on the structure, GISAXS is the only method to provide statistical information at the nanometer level [40]. A major bottleneck preventing GISAXS from reaching its full potential has been the availability of curated data, analysis methods and modeling resources for interpreting the experimental data [41].

In order to advance GISAXS diffraction image understanding and usability, we have used GISAXS simulation codes to generate more complete catalogs of potential experimental outcomes. Our paper takes data from HipGISAXS, which is a massively parallel simulator, developed using C++, augmented with MPI, NVIDIA CUDA, OpenMP, and parallel-HDF5 libraries, to take advantage of large-scale clusters of multi/many-cores and graphics processors. HipGISAXS currently supports Linux, Mac and Windows based systems. It is able to harness computational power from any general-purpose CPUs including state-of-the-art multicores, as well as NVIDIA GPUs and Intel processors, delivering experimental simulations at high resolutions [40, 42]. Using the HipGISAXS code, beamline scientists can create sample image collections of scattering patterns corresponding to four different crystal unit cell structures or lattices, such as Cubic: Simple cubic (8 on the corners of a cube), BCC: Body Centered Cubic (8 on corners, one in the center of cube), FCC: Face Centered Cubic (8 on corners, one in the center of each face of cube), and HCP: Hexagonal Close Packed (non-cubic, but one of the most commonly occurring lattices).

3.3 Cervical Cells

According to the National Cancer Institute (NCI) [43], Pap tests are an essential mechanism to detect abnormal cervical cells [44, 45] as part of regular screenings, which can reduce cervical cancer rates and mortality by 80 percent. However, cervical cancer continues to be the fourth leading cause of cancer deaths in Brazil [46, 47], where most of the female population depends on visually-screened cervical cytology

from routine conventional Pap smears. Although more than 80% of exams in the U.S. use liquid-based Pap test, this protocol is more than 50% more expensive than conventional Pap smears, and remains unavailable for most of the world's population.

Sparking more interest in supporting cell analysis using conventional Pap smears, pathologists have harvested an anonymized image databases from the Brazilian Public Health System [48], containing samples from a heterogeneous population across age, race, and socio-economical status. A large portion of these images is available through the Cell Recognition for Inspection of Cervix (CRIC) database [49] that catalogs numerous cases of cervical cells, classified according to the Bethesda System as atypical squamous cells of high risk and undetermined significance (#ASCH=470 and #ASCUS=116), normal (#Normal=343), low-grade and high-grade squamous intra-epithelial lesions (#LSIL=115 and #HSIL=1,018), and invasive carcinoma (#IC=60). This paper uses a subset of the CRIC database[2], previously classified by at least three cytopathologists, comprising 169 digitized Pap smear glass slides, which results in 3,393 cervical cells with normal or abnormal morphology, including overlapping cells.

Figure 4 displays image samples of the CRIC collection digitized from conventional Pap smears, containing special characteristics such as coming from a broad racial diversity, which is a trace of the Brazilian population.

3.4 Public images: the Flicker Material Database

The Flicker Material Database (FMD) [28] was designed to facilitate progress in material recognition, and it contains real world snapshots of ten common material categories: fabric, foliage, glass, leather, metal, paper, plastic, stone, water, and wood, as illustrated in Figure 5. According to Sharan et al. [28], each image in this database (100 images per category) was selected manually from Flickr.com to ensure a variety of illumination conditions, compositions, colors, texture surface shapes, material sub-types, and object associations.

The intentional diversity of FMD reduces the chances that simple or low-level information descriptors, e.g., color or first order intensity features, are enough to distinguish material categories. Strategies to construct middle-level features have enabled accuracy improvements in materials recognition problems [50], especially when including larger materials databases, such as MINC [26]. This previous research on FMD description and learning schemes points out limitations of using FMD alone as the training data source. We address some of these gaps, using two model training approaches as discussed in the next section.

4. Methodology

Data-driven algorithms that learn from accumulated experience, such as those in `pyCBIR`, can support ranking image

[2]Original images will be posted at http://cricdatabase.com.br/ upon paper acceptance.

(a) Fabric **(b)** Foliage **(c)** Glass **(d)** Leather **(e)** Metal

(f) Paper **(g)** Plastic **(h)** Stone **(i)** Water **(j)** Wood

Figure 5. Examples of the Flicker Material Database for of its each classes.

sets in face of (a) the difficulties to obtain specific knowledge needed for modeling, (b) limitations in learning the intricacies of every science domain, and (c) restricted generalizations of hard-coded knowledge rules.

This section describes how `pyCBIR` uses CNN in order to provide data reduction by automatically learning compact signatures that represent each image. An essential step required to organize the database is to construct models for different science problems in conjunction with algorithms for enhanced search experience. The next subsections explain how we use CNN to obtain image characteristics and rank images by similarity. Although we omit results using classic feature extraction methods in `pyCBIR`, they are also available and include Gray-Level Co-Occurrence Matrix (GLCM), Histogram of Oriented Gradients (HOG), Histogram features, Local Binary Pattern (LBP) and Daisy [51].

Figure 6 shows the `pyCBIR` graphical interface and a result using the cervical cells database. The first column in the visual result corresponds to the query images, and the other columns are the ranked outputs for each query. Given a fully labeled data collection, there will be green boxes around images correctly classified during similarity retrieval, and red ones around the incorrect classified pictures.

4.1 Neural network, CNN and topology

Among the several NN design options, this paper explores two different CNN architectures: (a) the Lenet [33], a neuronal arrangement that switches from fully connected to sparsely connected neurons allowing feedback in real-time for most applications, particularly when considering graphic card units for computation. Due to its simplicity, training often performs well with a smaller number of examples in comparison with deeper NN, however it is mostly inaccurate when dealing with complex recognition problems; and (b) Inception-ResNet-v2 [34], a deeper and wider architecture formed by multiple sub-networks, in which hierarchical layers promote many levels of non-linearity needed for more elaborated pattern classification. This model requires roughly twice as much memory and computation as the previous version (Inception v3 [52]), but it has demonstrated to be more accurate than previous state-of-the-art models, particularly when considering searches such as the Top-1 and Top-5 recommendations using

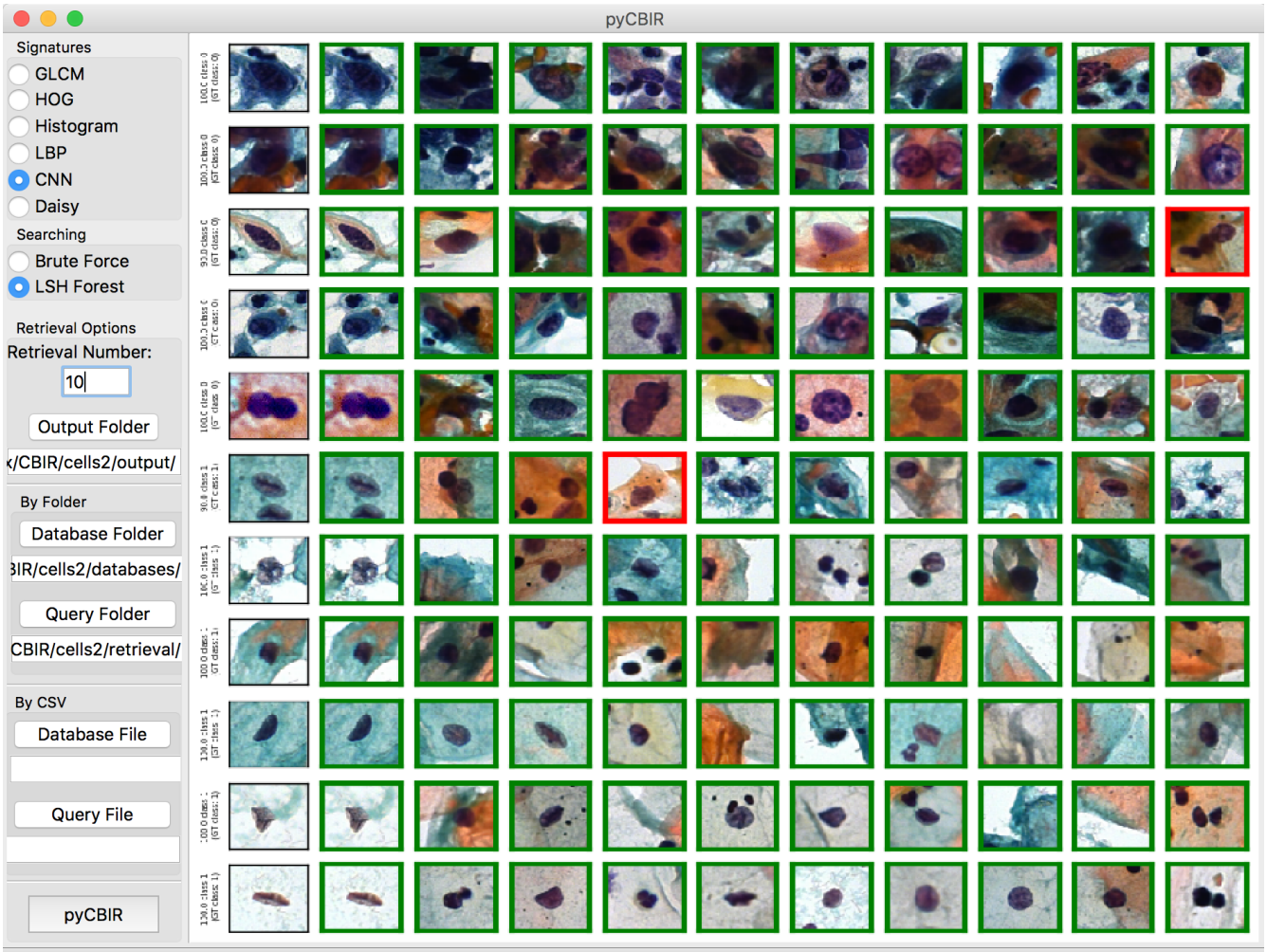


Figure 6. `pyCBIR` interface: retrieval options (left) with feature extraction, searching method, retrieval number, and data paths, and retrieval results (right) with query images (first column) and top matches; green border indicates match, and red, misclassified images.

the ILSVRC2012 benchmark database.

4.2 Query with signatures from CNN layer

Typically, CNN forms a hierarchical feature extractor that maps the input image into increasingly refined features, which serve as input to a fully connected layer that solves the classification. Figure 1 illustrates the alternating convolutional and pooling layers, which transform and reduce the input data before it reaches deeper stages as the fully connected layer. Notice in Figure 2 that we bypass the classification layer to use the features as signatures to drive the retrieval process, so `pyCBIR` can search and match the current image-query to the most similar samples in the database using "machine-designed" features.

With the purpose of suggesting a category based on a predefined taxonomy, we denote Y classes to an image database X, reduced into signatures, consisting of n images x_i, for $1 \leq i \leq n$. Searches will occur in a h-dimensional space, obtained through x_i transformation throughout the CNN. Alternatively, principal component analysis (PCA) can be used in order to reduce the signature dimensionality to improve the computational cost, while preserving the retrieval success rate.

The `pyCBIR` retrieval module will recognize similar samples through a similarity function S, so that $S(x_i, x_q)$ returns relevant items as well as the respective uncertainty value. In other words, the engine returns the top k most similar images, their respective y_j for a query-image $x_q \in R^{h \times n}$, and $S(x_i, x_q)$, defined as follows:

$$S(x_i, x_q) = \frac{x_i \cdot x_q}{(||x_i|| * ||x_q||)}. \quad (1)$$

where $\cdot$ is the dot product.

We report our results using the cosine similarity metric, but other distance metrics are available through the `pyCBIR` graphical user interface, including Euclidean distance, Infinity distance, Pearson correlation, Chi-square dissimilarity, Kullback-Leibler divergence, Jeffrey divergence, Kolmogorov-Smirnov divergence, Cramer divergence, and Earth mover's

distance [53].

4.3 Indexing and searching methods

Quick feedback when searching images by similarity is essential: a linear search through a database with millions of items may lead to unacceptable waiting times. Therefore, after the computation of the image signatures, we map them to a lower dimensional space using PCA, and use the most significant components as input to an indexing algorithm.

We propose an indexing routine that employs the Locality Sensitive Hashing Forest (LSH) [54], whose polynomial cost and sub-linear query time considerably speed up the information retrieval. This algorithm delivers efficient approximate nearest-neighbor queries by improving the original locality sensitivity filtering (LSF) [55] scheme, which otherwise would require tuning parameters, such as the number of features and the distance radius, as a function of the data domain. Moreover, it enables fast signature insertion and deletion while ensuring minimal use of storage.

By using LSH, `pyCBIR` selects a small set of potential images to be compared against the image-query; this happens because similar images, according to some metric, e.g., cosine similarity, are more likely to hash to the same bucket. We used random projection as the hash function to approximate the cosine distance between vectors, so that our system can efficiently translate CNN-calculated signatures into 32-bit fixed-length hash values. In addition, we store pre-computed versions of the LSH in disk for future analogous search requests using the same trained model.

Algorithm 1 shows the steps of the LSH-based indexing scheme before returning the result R of ranked outputs, where $|.|$ is the length of a set, **s** is an image signature within the set S, **s'** is the transformed (hashed) signature. The LSH-based indexing function retrieves the approximate nearest neighboring items from the hash table, and its output are the k most similar images to the query-image set Q.

Algorithm 1: LSH-based indexing.

Input :
- **S** signature database.
- **Q** set of signatures from query-images.
- ***k*** number of image matches.

Output :
- **R** ranked output (top-k matches).

```
begin
    if ∃ LSH previously computed then
        Read the LSH from file;
    else
        for s ∈ S do
            LSH.add(Random_projection_hash(s));
        end
    end
    Create R with |Q| lines and k columns;
    for q ∈ Q do
        R_{q,k} = LSH.similarity(q,k);
    end
end
```

4.4 Database augmentation

CNN-based recognition systems often require a large number of examples in order to fine-tune models during the training stage and deliver accurate classification results. A few strategies are commonly devised to deal with relatively small datasets, for example, modifications of the original observations to generate new ones, following expected distortions given certain degrees of freedom. In this context, data augmentation consists of applying mathematical transformations to typical samples in order to generate new images that are slightly different, but relatively similar so that they will belong to the same class. The most common image transformations are scaling, translations, rotations, noise addition and blurring. Image augmentation can be included as an additional processing step as part of the CNN training, and here we use the cell and FMD databases since they contain a limited number of samples (less than 2,000 per class). Therefore we performed 12 translations (three values in each direction: cells 7, 14 and 20 pixels; FMD 8, 16 and 24 pixels) and 3 rotations (every 90^o) to each image, augmenting the dataset by a factor of 51.

4.5 Evaluation metrics

We used the Mean Average Precision (MAP) [56] metric to evaluate the quality of the retrieved images. To compute MAP, the Average Precision score $AP(Q)$ is defined for each image Q in the rank as:

$$AP(Q) = \frac{\sum_{n=1}^{M}(P(n) * f(n))}{N}, \tag{2}$$

where $P(n)$ is the precision at cut-off n in the rank, $f(n)$ is equal to 1 if the image at rank n belongs to the same class of the query, and 0 otherwise. M is the number of images in the rank and N is the number of images of the same class given by the query. The MAP score is obtained by averaging the AP score over all images in the rank. The higher the MAP score is, the better the performance.

We also computed the classification accuracy rate for each class in the databases. This accuracy was calculated by using the k-nearest neighbor [57] for different values of k.

5. Experimental results

This section describes `pyCBIR`'s inputs and outputs, how to use different datasets as well as how to compare the output from different networks. The results refer to the databases described in Section 3 and the algorithms discussed in Section 4.

5.1 CNN training

When deriving PCA-based signatures from LeNet or Inception-ResNet-v2 outputs, both schemes require only two parameters:

the initial learning rate and the decay factor. We set the initial learning rate as 0.1 and the decay factor of 0.04 in the LeNet. The fine-tuning operation requires an initial learning rate smaller than the one used to train the CNN with random initialization. Here, we report experiments setting this parameter to 0.008 and the decay factor as 0.0004. Experiments showed that slightly different values affect the training time, but may lead to similar classification accuracy results.

Table 2. Number of epochs and processing time to train both LeNet and Inception-ResNet-v2 neural networks using different image databases.

Database	Epochs	Processing Time
LeNet		
Fibers	5	9min
GISAXS	5	45min
Cells	6	70min
FMD	20	42min
Inception-ResNet-v2		
Fibers	21	336min
GISAXS	13	273min
Cells	17	289min
FMD	51	153min

The number of epochs varies with respect to the databases due to the image inter-class variation, the number of classes, and the image size, while the number of epochs was set according to the loss function. When the loss remains constant between epochs and it is lower than 0.1, the CNN training automatically stops. Table 2 shows the number of epochs and processing time to train the LeNet and the longer fine tuning step needed by the Inception-ResNet-v2.

5.2 Performance evaluation

In order to evaluate the fibers and GISAXs databases, we used 10% of the images from both databases for LeNet training, Inception-ResNet-v2 fine-tuning and auto-value estimation of the PCA. We used the other 90% of both databases to calculate the MAP and k-accuracy as in Tables 3-4. For the Cells (Table 5) and FMD (Table 6) databases, we used 50% of the image sets for the LeNet training, Inception-ResNet-v2 fine-tuning and PCA transformation, and the other half to calculate the MAP and k-accuracy measures. Such different splits were necessary given the sizes of these databases; for example, the cells and FMD databases have a few thousand samples, which is a limited amount of data for the CNN training. Furthermore, we augmented both training subsets using affine image transformations as a preprocessing step. To illustrate how this step impacts our pipeline, we performed experiments with and without data augmentation for the cell database. The MAP value without augmentation yielded 0.85 as opposed to 0.94 when considering the augmentation step (preprocessing). The mean average also increased, which indicates that the CNN converged to a model that best classifies the augmented set.

Figure 7 shows the MAP results for all databases in comparison with the number of PCA components, when considering the LeNet and Inception-ResNet-v2 pre-trained and after fine-tuning. The curves show that it is possible to reduce the number of features in the retrieval process by using 16 or less components. We notice that this reduction improves the indexing and searching procedures when applied to the four different image sets.

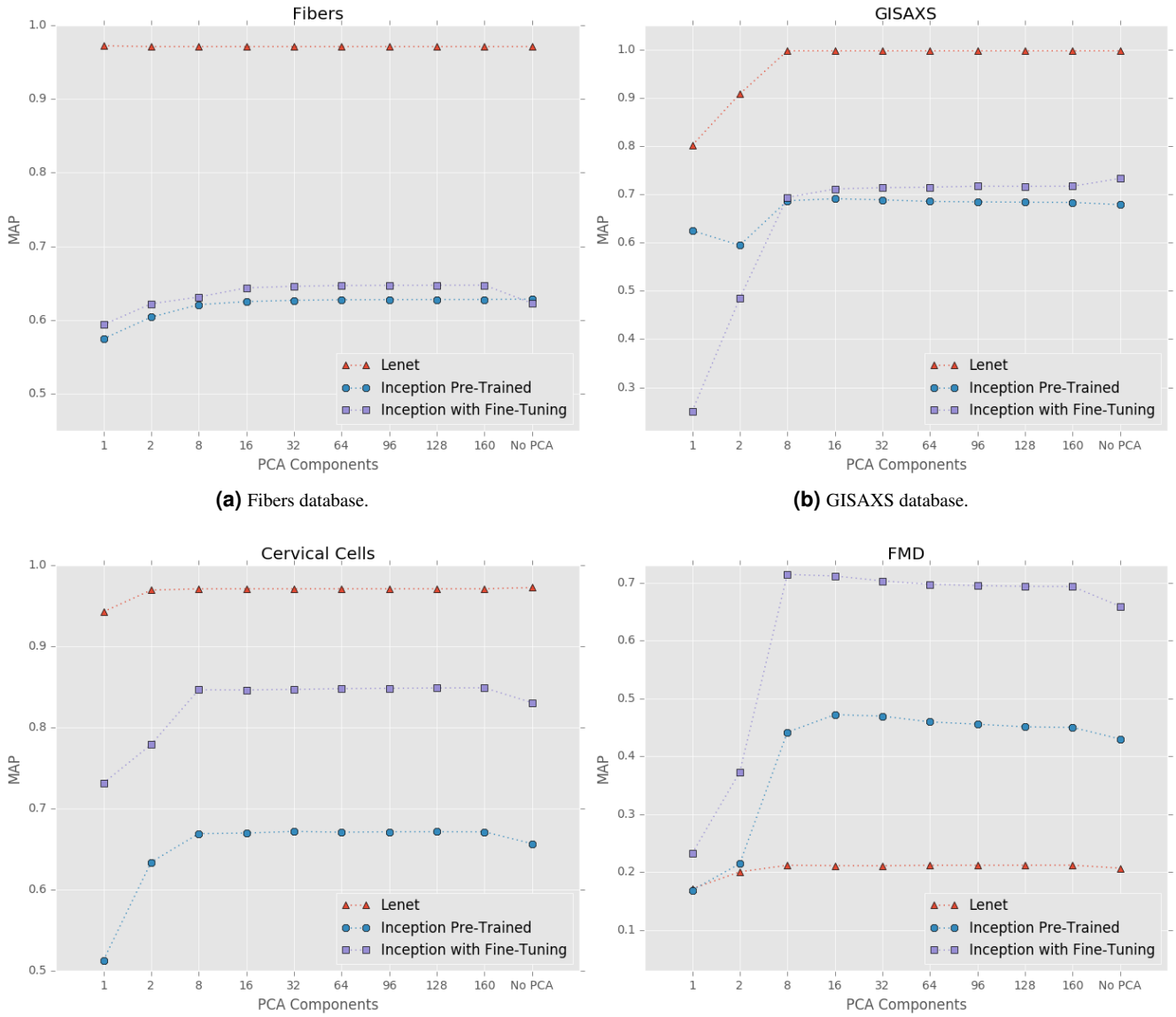
Table 3. Accuracy rate of the LeNet and the Inception-ResNet-v2 for the Fibers database using different k values, where Ω is the number of images of the class.

	LeNet				
k	1	5	10	20	Ω
No-Fibers	0.973	0.977	0.979	0.978	0.963
Fibers	0.975	0.988	0.991	0.990	0.996
	Inception-ResNet-v				
No-Fibers	0.781	0.777	0.797	0.746	0.667
Fibers	0.925	0.978	0.980	0.983	0.975
	Inception-ResNet-v2 pre trained				
No-Fibers	0.727	0.743	0.736	0.743	0.167
Fibers	0.825	0.890	0.927	0.933	1.000

Surprisingly, the LeNet achieved better results than Inception-ResNet-v2 regarding the fibers, GISAXs and cell databases. One of the reasons this CNN outperformed Inception-ResNet-v2 for these databases was due to the size of the images. The Inception-ResNet-v2 expects an input image of 299x299, therefore the current pipeline resizes the images from 100x100 (Cells and GISAXS) and 16x16 (Fibers) to such representation, an operation that might distort important aspects of the data. We also tested zero-padding operations instead of resizing, but the results accuracy remained the same. The lower MAP and k-accuracy values obtained by using the Inception-ResNet-v2 pre-trained are most likely due to the learned model, which depended on a database with a wide variety of objects (ILSVRC2012), which poorly correlates to our image databases.

Regarding the FMD images, the Inception-ResNet-v2 outperformed the LeNet, which might be influenced by the similarity between FMD and the ILSVRC2012 database. Recall that the FMD samples contain complex patterns, resembling ILSVRC2012 data points. Fine-tuning the Inception-ResNet-v2 further improves the results in relation to the pre-trained due to tuned layers customized to extract features for the FMD data.

We also computed the classification k-accuracy for each class of all databases, where $k \in [1, 5, 10, 20, \Omega]$, and Ω is the total number of images in a particular class. Tables 3, 4, 5 and 6 confirm the MAP results presented in Figure 7. Based on these results, when there are more than a couple of thousand images and a few classes to train for, we observed that the LeNet outperforms the deeper network for our databases. In contrast, when there are millions of images to fine-tune the network, the Inception-ResNet-v2 performed better for a single database, FMD, characterized by several classes and larger images. When no classes/labels are available, the pre-trained Inception-ResNet-v2 is a promising starting point for tasks such as image sorting.



(a) Fibers database.

(b) GISAXS database.

(c) Cells database.

(d) FMD.

Figure 7. MAP values in relation to the number of PCA components for the LeNet and Inception-ResNet-v2.

5.3 Visual Results

`pyCBIR` also provides a graphical user interface (GUI) (Figure 6). The main advantages of the `pyCBIR` GUI are: (a) The visual output shows both correct and miss-classified results when ground-truth is available; (b) `pyCBIR` allows to choose other feature extraction methods, besides CNN, that do not require training or labeled samples; (c) `pyCBIR` contains ten different similarity metrics for evaluation of the searching results; and (d) One can load the database from a comma-separated values (CSV) file or simply querying the file system. As result, `pyCBIR` shows the k-ranked outputs ($k = 10$ in Figure 6), the first column represents the query images. Each output has a bound box that corresponds to the correct retrieved images (green box), and miss-classified images (red box). For each execution `pyCBIR` saves the result as a portable network graphics (PNG) file, as Figures 8, 9, 10 illustrate the use of `pyCBIR` for Fibers, GISAXS and FMD databases and their corresponding ranked outputs.

5.4 Time during image-query-based retrieval

In addition to accuracy, the computational cost to search images given a query is another measure of the value of recommendation systems. Table 7 shows the computational time to retrieve k images (equal to the database size) given a query. We used the value k because it is the worst case of image searching (sort all images of the database given a query). We also computed the time for the 1^{st} (before LSH creation) and 2^{nd} execution (after LSH creation).

Although the LSH-based module computes 32-bit hashes,

Table 4. Accuracy rate of the LeNet and the Inception-ResNet-v2 for the GISAXS database using different k values, where Ω is the number of images of the class.

	LeNet				
k	1	5	10	20	Ω
bcc	1.000	1.000	1.000	1.000	1.000
fcc	1.000	1.000	1.000	1.000	0.999
cubic	1.000	1.000	1.000	1.000	1.000
hpc	1.000	1.000	1.000	1.000	1.000
	Inception-ResNet-v2				
k	1	5	10	20	Ω
bcc	1.000	0.996	0.990	0.980	0.974
fcc	0.998	1.000	1.000	1.000	0.998
cubic	1.000	0.998	1.000	0.998	0.972
hpc	1.000	1.000	1.000	1.000	0.996
	Inception-ResNet-v2 pre trained				
bcc	0.980	0.975	0.972	0.959	0.599
fcc	1.000	1.000	1.000	1.000	1.000
cubic	0.993	0.995	0.995	0.991	0.993
hpc	0.995	0.996	0.997	0.995	0.698

Table 5. Accuracy rate of the LeNet and the Inception-ResNet-v2 for the Cells database using different k values. Ω is the number of images of the class.

	LeNet				
k	1	5	10	20	Ω
Normal	0.962	0.970	0.975	0.977	0.969
Abnormal	0.969	0.979	0.983	0.981	0.984
	Inception-ResNet-v2				
k	1	5	10	20	Ω
Normal	0.938	0.932	0.935	0.924	0.810
Abnormal	0.971	0.985	0.991	0.986	0.984
	Inception-ResNet-v2 pre trained				
Normal	0.846	0.861	0.857	0.851	0.728
Abnormal	0.893	0.929	0.947	0.952	0.957

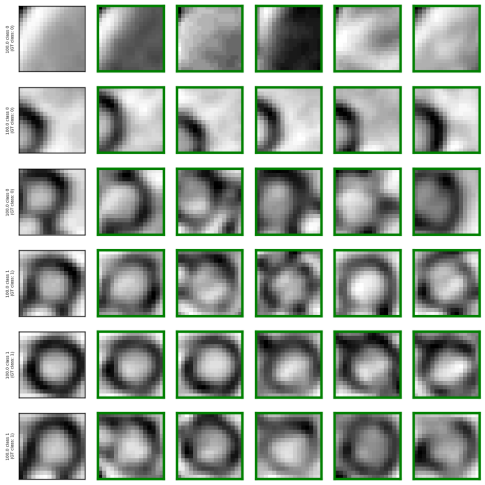


Figure 8. Results using the LeNet and fibers database for 6 query images chosen randomly and their corresponding top-5 ranked outputs.

the computational cost dramatically increases with the dimension of the input vector as well, therefore we use only the most significant principal components. Our motivations to keep

Table 6. Accuracy rate of the LeNet and the Inception-ResNet-v2 for the FMD using different k values, where Ω is the number of images of the class.

	LeNet				
k	1	5	10	20	Ω
Fabric	0.140	0.200	0.200	0.280	0.320
Foliage	0.240	0.360	0.440	0.440	0.360
Glass	0.120	0.200	0.260	0.140	0.040
Leather	0.140	0.300	0.200	0.080	0.100
Metal	0.360	0.540	0.540	0.480	0.520
Paper	0.200	0.280	0.240	0.340	0.260
Plastic	0.140	0.160	0.220	0.300	0.280
Stone	0.200	0.340	0.320	0.280	0.340
Water	0.220	0.340	0.320	0.280	0.280
Wood	0.500	0.700	0.760	0.700	0.700
	Inception-ResNet-v2				
Fabric	0.80	0.88	0.88	0.90	0.88
Foliage	0.96	0.90	0.90	0.92	0.92
Glass	0.82	0.86	0.82	0.78	0.80
Leather	0.90	0.86	0.82	0.82	0.82
Metal	0.70	0.76	0.70	0.70	0.66
Paper	0.86	0.92	0.92	0.94	0.92
Plastic	0.66	0.60	0.58	0.54	0.50
Stone	0.78	0.80	0.80	0.76	0.74
Water	0.94	0.94	0.94	0.94	0.94
Wood	0.82	0.86	0.90	0.94	0.92
	Inception-ResNet-v2 pre trained				
Fabric	0.680	0.800	0.820	0.760	0.740
Foliage	0.760	0.760	0.740	0.800	0.700
Glass	0.840	0.900	0.900	0.860	0.900
Leather	0.520	0.680	0.620	0.600	0.460
Metal	0.920	0.900	0.860	0.860	0.820
Paper	0.520	0.620	0.600	0.700	0.740
Plastic	0.620	0.780	0.840	0.860	0.820
Stone	0.660	0.740	0.600	0.560	0.620
Water	0.780	0.800	0.800	0.700	0.560
Wood	0.860	0.880	0.880	0.840	0.820

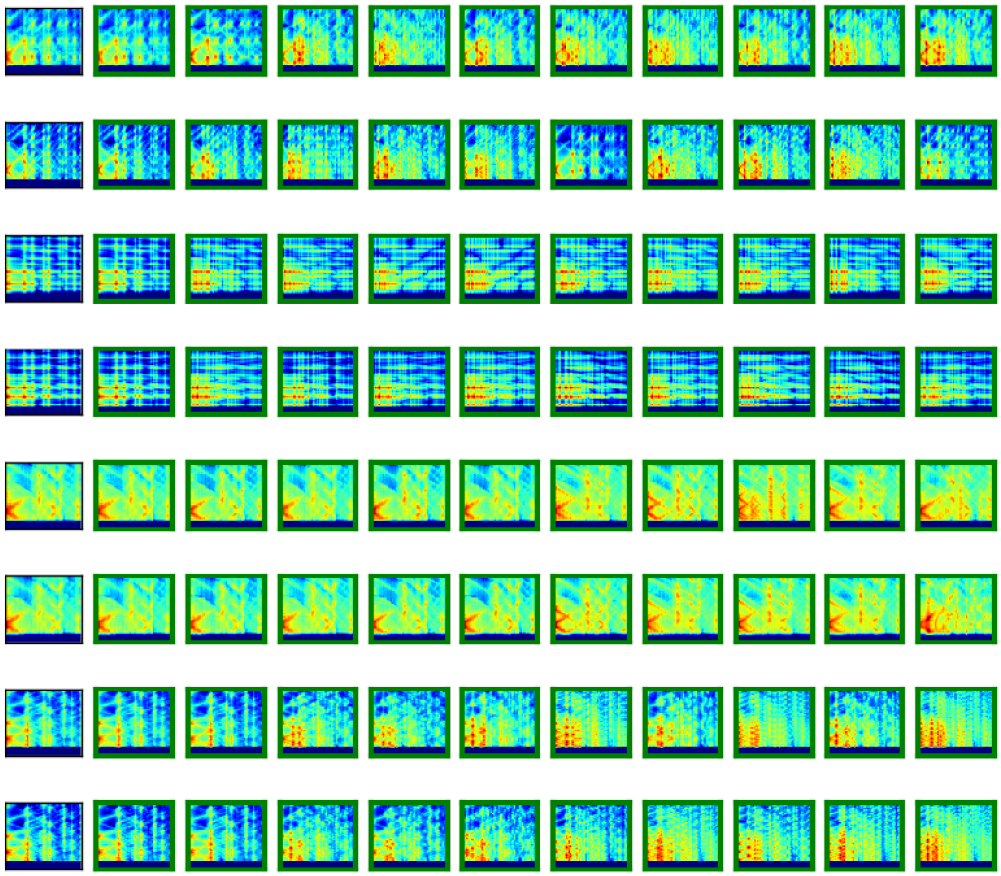


Figure 9. Results using the LeNet and GISAXS database for 8 query images chosen randomly and their corresponding top-10 ranked outputs.

only the 16 most significant PCA components are: (a) this subset of components explains more than 98% of the data for each scientific domain under investigation; (b) our results showed that the Mean Average Precision remains constant around 16 components, as illustrated in Figure 7; (c) this representation is more than 3 times faster than using the raw signatures with the Inception-ResNet-v2, as Table 7 illustrates.

All computational experiments involving `pyCBIR` ran on a Deep Learning Machine (DevBox) with six cores of Intel Xeon E5-2643 @ 3.40 GHz, four graphics processors GeForce GTX Titan-X and 251GB memory. However, we have been able to run `pyCBIR` in standard laptops as well, but at a much higher computing time and restricted to smaller subsets. Software dependencies include Ubuntu 14.04, CUDA 7.5, cuDNN v4, Google Tensor Flow v.0.11.0 and python 3.5.2 through Anaconda 4.1.1. Also, `pyCBIR` relies on an assortment of packages within the Python ecosystem, such as: numpy, scipy, scikit-learn, scikit-image, PyQt5 and matplotlib.

6. Conclusions and future works

Material scientists and cytopathologists rely on visual exploration of image microstructures, but manual data curation of experimental data continues to be scarce and time-consuming. This paper described how our recommendation system leverages limited curated datasets to provide functions to automatically organize large catalogs of scientific images, and classify unseen cases using deep learning, offering new ways for searching image-centric data. We also designed image augmentation routines that allow increasing number of samples following typical image transformations. Additionally, the algorithms behind our software tool `pyCBIR` expose key parameters to control performance, such as the number of CNN layers, CNN epochs, and image sizes.

Our results showed the importance of feature reduction in the searching process, for example by using LSH, and indicates a promising research direction for the system improvement. Because `pyCBIR` is capable of easily performing SIFT-based and CNN-based image retrieval, we will explore combining CNN and SIFT features to improve accuracy.

Being able to detect materials properties [58] in real time will add an entirely new level of experimental capability, including triage, quality assurance and prioritization. Tying this capability to the control systems at imaging instruments, such as at synchrotron beamlines, promises to enable scientists to automatically steer the machine in response to specific structures present in the sample with minimum human interference.

One limitation of the `pyCBIR` is the image resizing step to normalize data collections, which can affect the accuracy

Figure 10. Results using the Inception-ResNet-v2 and fmd database for 10 query images chosen randomly and their corresponding top-5 ranked outputs.

Table 7. Time in seconds to retrieve all database images given a query by using the LeNet and the Inception-ResNet-v2. In the PCA results we used 16 components.

		Fibers	GISAXS	Cells	FMD
LeNet	1^{st}	12.08	61.42*	0.020	0.015
	2^{nd}	10.35	61.75*	0.017	0.008
Inception-ResNet-v2	1^{st}	19.41*	75.11*	0.130	0.028
	2^{nd}	19.63*	75.17*	0.032	0.009
Using PCA	1^{st}	10.27	22.29	0.017	0.009
	2^{nd}	9.28	19.75	0.014	0.005

*use brute force search; 1^{st} means the first iteration and 2^{nd} means second iteration and beyond.

of the retrieval task since this operation will insert often undesirable distortions. For example, the LeNet assumes samples of 32×32 images, otherwise the computational cost of this network becomes undesirable for large images; the limited (only two layers) feature reduction. Regarding the Inception-ResNet, the system will expect images of size 299×299. Another drawback of the `pyCBIR` is the restriction of the LSH Forest to collections smaller than 2 million samples, in which cases the search process must be the brute force. This restriction occurs because the LSH uses hashes of 32-bits, and the increase in the number of examples also increases the number of collisions. It is worth mentioning that this limitation affects only the time of the retrieval process.

Future work will include more scalable hashing mechanisms that circumvent the maximum size of the search tree available in `scikit-learn`. Another challenge will be to expand the automated data curation capability to also extrapolate metadata to unseen samples using visual attributes combined with natural language processing.

Acknowledgments

This work was supported by the Office of Science, of the U.S. Department of Energy (DOE) under Contract No. DE-AC02-05CH11231, the Moore-Sloan Foundation, CNPq (304673/2011-0, 472565/2011-7, 401120/2013-9, 401442/2014-4,444784/2014-4, 306600/2016-1) and Fapemig (APQ-00802-11). This work

is partially supported by the DOE Advanced Scientific Computing Research (ASCR) Early Career Research Project Image across Domains, Experiments, Algorithms and Learning (IDEAL), as well as the Center for Applied Mathematics for Energy Research Applications (CAMERA), which is a partnership between Basic Energy Sciences (BES) and ASCR within DOE. We are specially grateful to James Sethian and Saul Perlmutter for incentivizing exploration of multidisciplinary science questions and deployment of open-source code. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DOE or the University of California.

References

[1] GE Digital. Predix: the industrial internet platform. https://www.ge.com/digital/sites/default/files/predix-platform-brief-ge-digital.pdf, 2016. Accessed on Dec 13, 2016.

[2] Dave Evans. The internet of things: how the next evolution of the internet is changing everything. http://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf, 2016. Accessed on Dec 13, 2016.

[3] Miguel P. Eckstein. Visual search: A retrospective. *Journal of Vision*, 11(5):14, 2011.

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[5] Oscar Araque, Ignacio Corcuera-Platas, J. Fernando Sánchez-Rada, and Carlos A. Iglesias. Enhancing deep learning sentiment analysis with ensemble techniques in social applications. *Expert Systems with Applications*, 77:236 – 246, 2017.

[6] Filipe Marcel Fernandes Gonçalves, Ivan Rizzo Guilherme, and Daniel Carlos Guimarães Pedronette. Semantic guided interactive image retrieval for plant identification. *Expert Systems with Applications*, 91:12 – 26, 2018.

[7] Daniela M. Ushizima, Hrishikesh A. Bale, E. Wes Bethel, Peter Ercius, Brett A. Helms, Harinarayan Krishnan, Lea T. Grinberg, Maciej Haranczyk, Alastair A. Macdowell, Katarzyna Odziomek, Dilworth Y. Parkinson, Talita Perciano, Robert O. Ritchie, and Chao Yang. Ideal: Images across domains, experiments, algorithms and learning. *The Journal of The Minerals, Metals & Materials Society*, pages 1–10, Sep 2016.

[8] JR Raphael. How google photos new custom-labeling feature can help clean up your collection. https://www.computerworld.com/article/2988232/android/google-photos-custom-labeling.html, 2015. Accessed on Oct 15, 2017.

[9] J. Donatelli, M. Haranczyk, A. Hexemer, H. Krishnan, X. Li, L. Lin, F. Maia, S. Marchesini, D. Parkinson, T. Perciano, D. Shapiro, D. Ushizima, C. Yang, and J.A. Sethian. Camera: The center for advanced mathematics for energy research applications. *Synchrotron Radiation News*, 28(2):4–9, 2015.

[10] Toshikazu Kato. Database architecture for content-based image retrieval. In *Proc. of SPIE Image Storage and Retrieval Systems*, volume 1662, pages 112–123, San Jose, CA, USA, 1992.

[11] Kyoji Hirata and Toshikazu Kato. Query by visual example - content based image retrieval. In *Proceedings of the 3rd International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '92, pages 56–71, London, UK, 1992. Springer-Verlag.

[12] Egon L. van den Broek, Eva M. van Rikxoort, and Theo E. Schouten. *Human-Centered Object-Based Image Retrieval*, pages 492–501. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[13] Wes Bethel, Martin Greenwald, and Lucy Nowell. Management, visualization, and analysis of experimental and observational data (EOD) - the convergence of data and computing. In *DOE ASCR Workshop*, pages 2–30. DOE, 2015.

[14] Wei Yu, Kuiyuan Yang, Hongxun Yao, Xiaoshuai Sun, and Pengfei Xu. Exploiting the complementary strengths of multi-layer cnn features for image retrieval. *Neurocomputing*, 237:235 – 241, 2017.

[15] Maria Tzelepi and Anastasios Tefas. Deep convolutional learning for content based image retrieval. *Neurocomputing*, 275:2467 – 2478, 2018.

[16] Amin Khatami, Morteza Babaie, H.R. Tizhoosh, Abbas Khosravi, Thanh Nguyen, and Saeid Nahavandi. A sequential search-space shrinking using cnn transfer learning and a radon projection pool for medical image retrieval. *Expert Systems with Applications*, 100:224 – 233, 2018.

[17] Liang Zheng, Yi Yang, and Qi Tian. SIFT meets CNN: A decade survey of instance retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[18] Nishant Shrivastava and Vipin Tyagi. An efficient technique for retrieval of color images in large databases. *Computers and Electrical Engineering*, 46:314 – 327, 2015.

[19] M.A. Hannan, M. Arebey, R.A. Begum, Hassan Basri, and Md. Abdulla Al Mamun. Content-based image retrieval system for solid waste bin level detection and performance evaluation. *Waste Management*, 50:10 – 19, 2016.

[20] Lazaros Tsochatzidis, Konstantinos Zagoris, Nikolaos Arikidis, Anna Karahaliou, Lena Costaridou, and Ioannis Pratikakis. Computer-aided diagnosis of mammographic masses based on a supervised content-based image retrieval approach. *Pattern Recognition*, 71:106 – 117, 2017.

[21] Qian Yu, Feng Liu, Yi-Zhe SonG, Tao Xiang, Timothy Hospedales, and Chen Change Loy. Sketch me that shoe. In *Computer Vision and Pattern Recognition*, pages 799–807, 2016.

[22] Pakizar Shamoi, Atsushi Inoue, and Hiroharu Kawanaka. Deep color semantics for e-commerce content-based image retrieval. In *Conf. Fuzzy Logic in Artificial Intelligence*, pages 14–20, 2015.

[23] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[24] Yanming Guo, Yu Liu, Ard Oerlemans, Songyang Lao, Song Wu, and Michael S. Lew. Deep learning for visual understanding: A review. *Neurocomputing*, 187:27–48, 2016.

[25] Ji Wan, Dayong Wang, Steven Chu-Hong Hoi, Pengcheng Wu, Jianke Zhu, Yongdong Zhang, and Jintao Li. Deep learning for content-based image retrieval: A comprehensive study. In *Proceedings of the ACM International Conference on Multimedia, MM '14, Orlando, FL, USA*, pages 157–166, 2014.

[26] Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. Material recognition in the wild with the materials in context database. *CoRR*, abs/1412.0623, 2014.

[27] Ce Liu, Lavanya Sharan, Edward H. Adelson, and Ruth Rosenholtz. Exploring features in a bayesian framework for material recognition. In *CVPR*, pages 239–246. IEEE Computer Society, 2010.

[28] Lavanya Sharan, Ruth Rosenholtz, and Edward H. Adelson. Accuracy and speed of material categorization in real-world images. *Journal of Vision*, 14(9):12, 2014.

[29] Yan Zhang, Mete Ozay, Xing Liu, and Takayuki Okatani. Integrating deep features for material recognition. *Computing Research Repository*, abs/1511.06522, 2015.

[30] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[31] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[32] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[33] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[34] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *Computing Research Repository*, abs/1602.07261, 2016.

[35] H A Bale, A Haboub, et al. Real-time quantitative imaging of failure events in materials under load at temperatures above 1,600C. *Nat Mater*, 12:40–46, 2012.

[36] D.M. Ushizima, T. Perciano, H. Krishnan, B. Loring, H. Bale, D. Parkinson, and J. Sethian. Structure recognition from high resolution images of ceramic composites. *IEEE International Conference on Big Data*, October 2014.

[37] Maryana Alegro, Edson Amaro-Jr, Burlen Loring, Helmut Heinsen, Eduardo Alho, Lilla Zollei, Daniela Ushizima, and Lea T Grinberg. Multimodal whole brain registration: Mri and high resolution histology. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 194–202, 2016.

[38] KATARZYNA ODZIOMEK, DANIELA USHIZIMA, PRZEMYSLAW OBERBEK, KRZYSZTOF JAN KURZYDŁOWSKI, TOMASZ PUZYN, and MACIEJ HARANCZYK. Scanning electron microscopy image representativeness: morphological data on nanoparticles. *Journal of Microscopy*, 265(1):34–50, 2016.

[39] Maryana Alegro, Panos Theofilas, Austin Nguy, Patricia Castruita, William Seeley, Hans Heinsen, Daniela Ushizima, and Lea Grinberg. Automating cell detection and classification in human brain fluorescent microscopy images using dictionary learning and sparse coding. *Journal of Neuroscience Methods*, 282:20–33, 2017.

[40] Alexander Hexemer and Peter Müller-Buschbaum. Advanced grazing-incidence techniques for modern soft-matter materials analysis. *IUCrJ*, 2(1):106–125, 2015.

[41] S. Chourou, A. Sarje, X.S. Li, E. Chan, and A. Hexemer. Hipgisaxs: A high performance computing code for simulating grazing incidence x-ray scattering data. *Journal of Applied Crystallography*, (6):1781–1795, 2013.

[42] Alex Hexemer. HipGISAXS: A massively-parallel high-performance x-ray scattering data analysis code. "`http://www.camera.lbl.gov/gisaxs`", 2016. Accessed on Dec 13, 2016.

[43] Nci. National Cancer Institute. `http://www.cancer.gov/types/cervical/hp/cervical-screening-pdq` visited on 2017-11-17, 2017.

[44] Ilana Lowy. *A Woman's Disease: The history of cervical cancer*. Oxford, 2011.

[45] Z. Lu, G. Carneiro, A. P. Bradley, D. Ushizima, M. S. Nosrati, A. G. C. Bianchi, C. M. Carneiro, and G. Hamarneh. Evaluation of three algorithms for the segmentation of overlapping cervical cells. *IEEE Journal of Biomedical and Health Informatics*, 21(2):441–450, March 2017.

[46] Inca. Instituto Nacional de Cancer. `http://www2.inca.gov.br/wps/wcm/connect/tiposdecancer/site/home/colo_utero` visited on 2017-11-17, 2016.

[47] Paulo H. C. Oliveira, Gladston Moreira, Daniela Mayumi Ushizima, Claudia M. Carneiro, Fátima N. S. de Medeiros, Flávio Henrique Duarte de Araújo, Romuere R. V. e Silva, and Andrea G. C. Bianchi. A multi-objective approach for calibration and detection of cervical cells nuclei. In *2017 IEEE Congress on Evolutionary Computation, CEC 2017, Donostia, San Sebastián, Spain, June 5-8, 2017*, pages 2321–2327, 2017.

[48] Ministry of Health. Brazilian Unified Health System. `http://portalsaude.saude.gov.br/index.php/cidadao/principal/english`, 2016. Accessed on Dec 13, 2016.

[49] CRIC. Cell Recognition for Inspection of Cervix. `https://cricdatabase.com.br`, 2018.

[50] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 3606–3613, Washington, DC, USA, 2014. IEEE Computer Society.

[51] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors. scikit-image: image processing in Python. *PeerJ*, 2:e453, 6 2014.

[52] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *Computer Vision and Pattern Recognition*, 2015.

[53] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python. `http://www.scipy.org/`, 2001. Accessed on Dec 13, 2016.

[54] Mayank Bawa, Tyson Condie, and Prasanna Ganesan. LSH forest: Self-tuning indexes for similarity search. In *Fourteenth International World Wide Web Conference (WWW 2005)*, 2005.

[55] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 604–613, New York, NY, USA, 1998. ACM.

[56] Bin Wang, Douglas Brown, Yongsheng Gao, and John La Salle. March: Multiscale-arch-height description for mobile retrieval of leaf images. *Information Sciences*, 302:132 – 148, 2015.

[57] N. S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[58] Mohamad Hussein Hobballah, Amadou Ndiaye, Franck Michaud, and Mark Irle. Formulating preliminary design optimization problems using expert knowledge: Application to wood-based insulating materials. *Expert Systems with Applications*, 92:95 – 105, 2018.