Conditional Image Retrieval

Mark Hamilton  
MIT, Microsoft  
markth@mit.edu

Stephanie Fu  
MIT, Microsoft Garage

William T. Freeman  
MIT, Google

Mindren Lu  
MIT, Microsoft Garage

Figure 1: Conditional image retrieval results on open access artwork from the Metropolitan Museum of Art and Rijksmuseum using culture and media, top row text, as conditioners.

Abstract

This work introduces Conditional Image Retrieval (CIR) systems: IR methods that can efficiently specialize to specific subsets of images on the fly. These systems broaden the class of queries IR systems support, and eliminate the need for expensive re-fitting to specific subsets of data. Specifically, we adapt tree-based K-Nearest Neighbor (KNN) data-structures to the conditional setting by introducing additional inverted-index data-structures. This speeds conditional queries and does not slow queries without conditioning. We present two new datasets for evaluating the performance of CIR systems and evaluate a variety of design choices. As a motivating application, we present an algorithm that can explore shared semantic content between works of art of vastly different media and cultural origin. Finally, we demonstrate that CIR data-structures can identify Generative Adversarial Network (GAN) “blind spots”: areas where GANs fail to properly model the true data distribution.

Preprint. Under review.
Algorithm 1: Querying a CKNN Tree

**input:** A point, \( q \), a condition, \( S \subseteq X \), a tree, \( \text{root} \), and an inverted index, \( I \)

**output:** Closest point, \( p^* \in S \), to \( q \)

```python
def SearchNode(n):
    if \( n \in \text{validNodes} \) then
        if \( n \) is a leaf node then
            \( p \leftarrow \) closest point in \( S \)
            if \( d(p, q) < d(p^*, q) \) then
                \( p^* \leftarrow p \)
            else
                \( \text{potentials} \leftarrow \) children of \( n \) which could hold a closer point
                for \( \text{child} \) in \( \text{potentials} \) do
                    SearchNode(child)
    SearchNode(root); return \( p^* \)
```

1 Introduction

In many Image Retrieval (IR) applications, it is natural to limit the scope of the retrieval to a subset of images. For example, returning similar clothes by a certain brand, or similar artwork from a specific artist. Currently, it is a challenge for IR systems to restrict their attention to sub-collections of images on the fly. More specifically, a core component of many IR systems, KNN data-structures, only support queries over the entire corpus. Currently, restricting retrieved images to a particular class or filter requires filtering the “unconditional” query results, or building a new KNN data-structure for each filter. Filtering unconditional results can be costly, especially if many of the retrieved unconditional images do not fit the user’s filter, or if the valid images are not close to the query image. Furthermore, re-building the KNN data-structure on each subset of the data incurs a steep computational penalty and can result in \( 2^n \) data-structures, where \( n \) is the total number of images. To mitigate these issues, we introduce “conditional” variants of tree-based KNN methods that can efficiently and adaptively prune their structure to fit arbitrary conditional predicates. We evaluate the query-time performance of these methods compared to their unconditional counterparts and introduce two new datasets to quantitatively measure success on CIR tasks.

We apply these conditional KNN data-structures on the combined open-access collections of the Metropolitan Museum of Art [1] and the Rijksmuseum [2] to create an algorithm capable of identifying artistic connections across time, space, culture and media. Additionally, we investigate the structure of these conditional neighbor trees and show that they can reveal areas of poor convergence and diversity (“blind spots”) in image based GANs. We summarize the contributions of this work as follows:

- We contribute a simple modification to existing KNN data-structures to allow users to efficiently filter resulting neighbors using arbitrary logical predicates, enabling efficient CIR
- We introduce two new datasets for evaluating CIR systems and use these datasets to evaluate various featurization strategies
- We use CIR to discover shared structure across genres, styles, artists, and media in the visual arts
- We use CIR data-structures to discover “blind spots” where GANs fail to match the true data

2 Background

IR systems aim to retrieve a list of relevant images that are related to a query image. “Relevance” in IR systems often refers to the “semantics” of the image such as its content, objects, or meaning. Many existing IR systems map images to "feature space" where distance better corresponds to relevance. In feature space, KNN can provide a ranked list of relevant images [58]. Good features and distance metrics aim to align with our intuitive senses of similarity between data [87], show invariance to certain forms of noise [26], and improve performance of other algorithms [36]. There is a considerable
body of work on learning good "features" for images, [11, 16, 21, 23, 25, 46, 76, 63, 36]. In this work we leverage features from intermediate layers of deep supervised models, which perform well in a variety of contexts and are ubiquitous throughout the literature. Nevertheless, our methods could apply to any features found in the literature including those from text, sound, and tabular data.

There are a wide variety of KNN algorithms, each with their own strengths and weaknesses. Typically, these methods are either tree-based, graph-based, or hash-based [6]. Tree-based methods partition target points into hierarchical subsets based on their spatial geometry and include techniques such as the KD Tree [12], RP Tree [13], PCA Tree [7], Ball Tree [68], M-Tree [90], VP Tree [89], some inverted index approaches [8], and tree ensemble approaches [13, 14, 88, 62]. Some tree-based data-structures allow exact search, and have formal guarantees on their performance [13]. Graph-based methods rely on greedily traversing an approximate KNN graph of the data, and have gained popularity due to their superior performance in the approximate NN domain [6, 22, 72, 43] [59]. There are numerous hash-based approaches in the literature and [83] provides a systematic overview. Notably, [45] provides theoretical bounds on probabilistic exact retrieval using angular distance hashing. To our knowledge, neither graph nor hash-based retrieval methods can guarantee finding the nearest neighbor. However, fast approximate search is often sufficient for many applications. In our work we focus on tree-based methods because it is unclear how to create an analogous method for graph-based data-structures. Nevertheless, tree based methods are widely used, especially when exact results are required.

Conditional K-Nearest Neighbors (CKNN) aims to find the $k$ closest points to a query point, $q$, with distance function, $d$. These points must satisfy a given logical predicate (condition), $S$, which we represent as a subset of the full corpus of points, $\mathcal{X}$:

$$
CNN(q, S \subseteq \mathcal{X}) = \arg\min_{t \in S} d(q, t)
$$

When the conditioner, $S$, equals the full space, $\mathcal{X}$, we recover the standard KNN definition. The goal of this work is to show that, for a broad class of KNN data-structures, it is possible to perform "predicate push-down" [34, 49] and move the logical predicate into the KNN data-structure to improve search speed. We stress that this work does not aim to make the fastest KNN algorithm, or to exhaustively implement predicate push-down in all KNN methods. Our aim is to show that conditioning a KNN data-structure on the fly is possible, has small overhead, yields new tools for image analysis, and can improve performance when added to a commonly used implementation [69] of the Ball tree and KD tree data-structures.

### 3 Conditional Nearest Neighbor Data-Structures

We aim to efficiently specialize an existing KNN data-structure trained on a corpora of points $\mathcal{X}$ to a particular subset of points, $S \subseteq \mathcal{X}$. In the context of tree based KNN methods, we can filter or "prune" both irrelevant data in the leaves of the tree and inner nodes without relevant children. More specifically, Figure 2 shows how a node can be pruned if none of the points below it satisfy the conditioning predicate. We can filter nodes efficiently with an inverted index [45], $I$, that maps points, $x \in \mathcal{X}$ to the collection of their dominating nodes, $I(x) = \{ n : x \text{ below node } n \}$. We can form the union of dominating nodes, as the first line of Algorithm 1 above, to compute the subset of nodes that remain after pruning. One can quickly prune nodes during traversal by checking node membership in this set. Furthermore, this set can be cached and shared between queries with the same conditioner. Leaf node points can be pruned through direct evaluation of the predicate. For scenarios where the predicates have additional structure, such as representing class labels, one can define a smaller class-based inverted index, $I_{class}(c)$ which maps a class label, $c$, to the set of dominating nodes. For these predicates, union and intersection operators commute through the class-based inverted index:

$$
I(S_a \cap S_b) = I_{class}(a) \cap I_{class}(b) \quad \text{and} \quad I(S_a \cup S_b) = I_{class}(a) \cup I_{class}(b),
$$

where $S_a$ is the subset of points with label $a$. This principle speeds a broad class of queries and accelerates document retrieval frameworks like Elastic Search [27] and its backbone, Lucene [61]. Our conditional inverted indices can apply to KNN trees independent of how the tree splits points (Ball, Hyperplane, Cluster), the branching factor, and the topology of the tree. They also apply to ensembles of trees and to multi-probe LSH methods by pruning hash buckets.
Table 1: Query and specialization times (ms) of Conditional KNN methods across several benchmarking datasets from [6]. CBT and CKD represent conditional variants of the Ball Tree and KD tree and SBT and SKD are trees specifically built for each query subset, BF represents Brute Force search. Conditional methods rival specialized trees without the 3-25k ms creation overhead.

| Dataset   | CKN Method | SBT | CBT | SKD | CKD | BF |
|-----------|------------|-----|-----|-----|-----|----|
| FMNIST    | 5+3k       | 8   | 7+4k| 20  | 8   |    |
| MNIST     | 5+3k       | 8   | 7+4k| 23  | 7   |    |
| Glove-25  | 6+9k       | 22  | 7+10k| 25  | 29  |    |
| Glove-50  | 24+13k     | 34  | 27+15k| 50  | 42  |    |
| Glove-100 | 47+20k     | 55  | 52+23k| 86  | 69  |    |
| Glove-200 | 52+33k     | 77  | 61+39k| 142 | 125 |    |
| NYT       | 11+9k      | 21  | 13+10k| 38  | 29  |    |
| SIFT      | 30+20k     | 52  | 35+24k| 69  | 66  |    |

Table 2: Performance of CIR (Accuracy @N) on content recovery across style variations for both the ConditionalFont (CF) and ConditionalArt (CA) datasets using a variety of features from pre-trained networks. Results show CIR retrieves the same content image across different styles. For full details on experimental conditions please see Section 7.

| Dataset | Metric | RN50 | RN101 | MN | SN | DN | RNext | dlv3101 | MRCNN | Random |
|---------|--------|------|-------|----|----|----|-------|---------|-------|--------|
| CA      | @1     | .50  | .51   | .35 | .44 | .59 | .46   | .37     | .45   | .0002  |
|         | @10    | .70  | .68   | .71 | .62 | .76 | .65   | .55     | .63   | .002   |
| CF      | @1     | .41  | .37   | .39 | .44 | .43 | .38   | .33     | .44   | .016   |
|         | @10    | .77  | .76   | .76 | .80 | .79 | .76   | .73     | .79   | .16    |

3.1 Implementation

We implement predicate push-down within the existing Ball Tree and KD tree implementations in the popular SciKit-learn framework [69]. These implementations support exact retrieval with several metrics, OpenMP parallelization [17], and Cython acceleration [10]. We also implement several (optional) accelerations such as dense bit-array set operations, and caching node subsets on repeated conditioner queries. Finally, we contribute a Spark based implementation of a Conditional Ball tree to Microsoft ML for Apache Spark [29, 30] for large-scale, elastic, and distributed queries across a wide variety of databases and formats.

3.2 Performance

To understand the performance implications of predicate push-down we quantify the overheads and the speedups on several large datasets and sizes of conditions. For our datastructures and datasets, we found that cost of creating the predicate inverted index is negligible when compared to building the underlying tree. We detail this investigation in Section C of the Appendix. In the scenario when $S = X$, we can directly compare the performance of conditional and unconditional data-structures. Figure 10 and Table 3 in the Appendix show query time distributions for conditional and unconditional variants of the KD tree and Ball Tree on several large-scale benchmarking datasets from the literature [6, 4, 67]. These experiments show that checking the node subset before each branch traversal adds negligible overhead. Thus we conclude that performance is dominated by the underlying KNN data-structure the dataset is applied to.
Figure 4: Representative samples from the ConditionalArt dataset (left) and ConditionalFont dataset (right). CIR systems conditioned on style should retrieve images of the same content.

To understand the performance of our approach we compare the performance of re-using the global KNN data-structure (our approach), with that of a predicate-specific data-structure made on the fly. In general, predicate-specific data-structures will always yield faster queries, but incur a large cost to re-make the data-structure for each query. In Table 1 we show how predicate-specific and pruned data-structures compare on several datasets from the literature. In this experiment, we use class labels for MNIST and FMNIST conditions and K-means cluster labels ($k = 10$) for the other datasets. Times are averaged across all conditions. To understand the effects of predicate size, we benchmark on 488k Resnet50-featurized images ($dim = 2048$) from the combined MET and Rijksmusem open-access collections with a randomly chosen test set ($n = 1000$). We condition on specific artwork media and cultures, as well as pairwise unions of the largest four classes to form subsets that contain more than half of the points. In Figure 3 we compare the performance of conditionally pruned trees, dedicated trees made on each subset, and a brute-force approach. We note that conditional trees approach the performance of dedicated trees for most of the predicates tested, but without the expensive cost of recreating the tree (Creation time not included in Figure 3). Furthermore, we note that the approach dominates brute force search until the subset reaches around 10% of the data for KD trees and 1% of the data for Ball trees. Thus, performance depends both on underlying algorithm and conditioner size. Our approach re-uses the global KNN data-structure without making assumptions on the distribution of predicates. This prioritizes broad applicability and limited code modification over query-time optimization. We hope future investigations can examine the effects of modifying the data-structure to account for a prior over predicates. For implementation, experimentation, environment, and computing details please see Section 7.

4 Limitations

This work does not aim to create the fastest KNN algorithm, but rather presents a broadly applicable conditioning scheme that’s simple, effective, and yields fruitful applications. We note that KNN retrieval chooses particular items significantly more than others, due to effects such as the “hubness problem” and we direct readers to [20] for possible solutions. We identify some lower dimensional failure cases that do not correspond to those mentioned by Dinu et al and explore this further in Section B of the Appendix. Our approach does not modify the KNN data-structure without making assumptions on the distribution of predicates. This prioritizes broad applicability and limited code modification over query-time optimization. We hope future investigations can examine the effects of modifying the data-structure to account for a prior over predicates. For implementation, experimentation, environment, and computing details please see Section 7.

5 Discovering Shared Structure in the Visual Arts

One application of CIR is to reveal cultural connections in the visual arts. More specifically, CIR on the combined Met and Rijksmusem collections finds striking connections between art from different histories and mediums. These matches often highlight cultural exchange and shared inspiration. For example, similarities between the Dutch Double Face Banyan (middle row left) and the Chinese ceramic figurine (middle row 3rd from left) of Figure 1 can be traced to the flow of porcelain and iconography from Chinese to Dutch markets during the 16th-20th centuries [47, 81]. We hope CIR can help art-historians and inspire the public to explore new artistic traditions. To this end, we introduce a website to explore CIR in the visual arts in Section A of the Appendix.

To create this method, we leverage features from the penultimate layer of ResNet50 [32] trained on ImageNet [19] which has been shown to capture many aspects of image semantics such as texture,
color, content, and pose\cite{66}. KNN retrieval on these features also yields quick and effective solutions for image retrieval, semantic in-painting, and image duplication detection\cite{31,53}. In this work we use “content” features from ResNet50. Alternatively one could use “style” based features from methods like AdaIN\cite{35}, or the Gram Matrix of\cite{24} to create a CIR system that retrieves images with similar “styles” and an approximate invariance to content.

5.1 Evaluation by Non-parametric Style Transfer

One of the largest design decisions in a CIR system is the choice of features used for queries. We quantitatively evaluate the impact of different featurization strategies by retrieving similar-content images across different stylizations. More specifically, if the conditioning information represents the image “style” and the features represent the “content”, CIR should find an image with the same content, but constrained to be in the style of the conditioner, such as “glass” or “Egyptian” in Figure\ref{fig:1}. Through this lens, CIR systems can act as “non-parametric” style transfer systems. This differs from the wide variety of style transfer and visual analogy methods in the literature as these approaches generate the image directly as opposed to finding the image within an existing corpora\cite{35,24,42,78,50}.

In this vein, we introduce two datasets with known style and content annotations: the ConditionalFont and ConditionalArt datasets. The ConditionalFont dataset contains 15687 32 × 32 greyscale images of 63 ASCII characters (content) across 249 fonts (style). This dataset serves as a simple human created baseline that echoes past work from the style-transfer literature\cite{78}. The ConditionalArt dataset consists of 10000 color images of varying resolution formed from 5000 content images from the MS COCO\cite{52} dataset and 200 style images from the WikiArt dataset\cite{65}. The contents and styles are transferred using a trained Adaptive Instance Normalization network\cite{35}, which serves as a consistent way to generate images of specific styles. We note that although the “ground truth” is an algorithmic style transfer method,\cite{41} show that these methods align with human intuition. Some representative samples from these two datasets are shown in Figure\ref{fig:4}. With these datasets, one can measure how CIR features, metrics, and query strategies align with the high-level task of matching content across styles. To measure retrieval accuracy, we choose 10000 random query images from the datasets and 10000 random styles to serve as conditions for our CIR system. For each query image we use CIR to retrieve the query image’s KNNs conditioned on its corresponding style. We then check whether any retrieved images have the same content as the original query image. In Table\ref{table:2} we explore how the choice of featurization algorithm affects CIR systems. All methods outperform the random baseline of Table\ref{table:2}, indicating that they are implicitly performing non-parametric content-
Discovering "Blind Spots" in GANs

Efficient high-dimensional KNN search data-structures need to robustly partition the data based on its variance and geometry. For example, recent tree-based KNN data-structures leverage unsupervised learning techniques such as hierarchical clustering [84] and slicing along PCA directions [7]. In this light, we can leverage KNN data-structures as a model of the hierarchical topology of the data and use this to study the “heterogeneity” of conditioning information relative to the full dataset. In Figure 5, we show that conditioners such as American Art in the MET and Rijks collections exhibit significant locality (41% node pruning) in the Conditional Ball Tree when compared to an equivalently sized random condition (12% node pruning). To quantify this, we introduce an simple and interpretable statistic, the Relative Conditioner Density (RCD), to measure the degree of over and under representation of a class \( c \) with corresponding subset \( S_c \subseteq X \), at node \( n \) in the KNN tree:

\[
RCD(n, c) = \frac{|n \cap S_c|}{|n|} \frac{|X|}{|S_c|}
\]

Here, \( |n| \) represents the number of data below node \( n \) in the tree. The RCD measures how much a node’s empirical distribution of labels differs from the global statistics, \( RCD > 1 \) occurs when the node over-represents class \( c \), and \( RCD < 1 \) occurs when the node under-represents the label, \( c \). We apply this analysis to understand how samples from generative models, such as image-based GANs, differ from true data. In particular, one can form a conditional tree containing true data and generated samples, each with their own classes, \( c_t \) and \( c_g \) respectively. In this context, nodes with \( RCD(\cdot, c_g) \ll 1 \) are regions of space where the network under-represents the real dataset. In Figure 5 we form a Conditional Ball Tree for samples from a trained Progressive GAN [44] and the training dataset: CelebA HQ [54]. Coloring the nodes by RCD reveals a considerable amount of statistically significant structural differences between the two distributions. By simply thresholding the RCD (\(< 0.6\)), we find types of images that GANs struggle to reproduce. We show samples from two low-RCD nodes in Figure 6 and also note their location in Figure 5. From a visual inspection of the real samples below this node, one can see that Progressive GAN struggles to generate realistic images of brimmed hats as well as microphones. Though we do not focus this work on thoroughly investigating issues of diversity in GANs, this suggests GANs have difficulty representing data that is not in the majority. This aligns with the findings of [9], without requiring an additional object detection or semantic segmentation ontology.

Experimental Details

We perform all timed comparisons in Table 1 and Table 3 of the Appendix, on an Azure NV24 Virtual Machine running Ubuntu 16.04, Python 3.7, and scikit-learn v0.22.2 [69]. We use Scikit-learn for...
Ball Tree and KD Tree implementations and use numpy v1.18.1 for brute force implementations. We leverage the benchmarking datasets (FMNIST, MNIST, GLOVE, NYT, SIFT) from [6] and use the same distance metrics. We used scikit-learn’s K-means clustering implementation to form reasonable conditioning subsets. To form image features for Table 2 we use trained networks from the torchvision v0.6 repository. In particular, we use ResNet50 (RN50), ResNet101 (RN101), MobileNetV2 (MN), SqueezeNet (SN), DenseNet (DN), ResNeXt (RNext), DeepLabV3 ResNet101 (dlv3101), and Mask R-CNN (MRCNN). Features are taken from the penultimate layer of the backbone, and the matches of Table 2 are computed with respect to cosine distance. We use trained a Progressive GAN from the open-source Tensorflow implementation accompanying [44].

8 Related Work

Image retrieval and nearest neighbor methods have been thoroughly studied in the literature, and there are many directions for future study. There are several survey works on KNN retrieval, but they only mention unconditional varieties [68, 14, 83]. Marchiori [60] has studied the mathematical properties of conditional nearest neighbor retrieval for large margin classification, but works primarily with graph based methods as opposed to trees. They do not apply this to modern deep features and do not aim to improve query speed. There are a wide variety of featurization strategies for IR systems. Gordo et al. [26] learn features optimized for IR. Siamese networks such as FaceNet embed data using tuples of two data and a similarity score and preserving this similarity in the embedding [46, 77]. These methods do not perform conditional retrieval, and features from these methods could be used to improve CIR systems. Conditional Similarity Networks augment tuple embedding approaches with the ability to handle different notions of similarity with different embedding dimensions [80]. This models conditions as similarities, but does not limit the search space of retrieved images in a generic fashion. These features have potential to yield in neighbor trees that, when pruned, have a similar structure and performance to dedicated trees. Sketch-based IR uses line-drawings as query-images, but does not aim to restrict the set of candidate images generically [56]. Style transfer [40] and visual analogies [50, 74] yield results similar to our art exploration tool, but generate the analogous images rather than retrieve them from an existing corpus. [79] splits IR systems into conditional subsystems but does not tackle generic conditioners or provide experimental evaluation. [71] create an IR system conditioned on text input, but do not address the problem of explicitly conditioning the query space. [23] and [51] respectively learn and use a hierarchy of concepts concurrently with IR features, which could be a compelling way to learn useful conditions for an Conditional IR system.

9 Conclusion

We have presented and evaluated new methods for efficient conditional image retrieval. We provided simple and generic ways to modify broad classes of KNN architectures to support arbitrary conditional queries on the fly. These queries can shed light on hidden commonalities between corpora, improve result diversity, and tailor results to better fit a users specifications. We demonstrated that this approach speeds conditional queries and approaches optimal results for relative condition fractions above 5% without adding overhead to the unconditional model. We showed that conditional IR can help reveal shared visual content across seemingly disparate artistic cultures and media. We introduced two new datasets, ConditionalFont and ConditionalArt, to quantitatively evaluate conditional IR and systematically explore the effect of different featurization strategies. Lastly, we explored the topology of CKNN data-structures to identify several “blind spots” in the ProGAN network trained on CelebA HQ.

10 Broader Impact

We have explored Conditional Image Retrieval systems to create a method to uncover the hidden commonalities in art historical corpora. This method can provide a useful research tool for historians and inspire the public to view the MET and Rijks collections in novel ways. By showing how works from a variety of cultures, media, and times share a common structure, we hope to promote respect for the work of other cultures, and dispel myths of cultural superiority.
To reduce the possibility of a hateful or adversarial attack on the system, we limit the scope of our interactive website to images already in the corpora of the MET and Rijksmuseum, as opposed to user uploaded images. Even with this limitation, works of art can contain nudity or offensive depictions of race and gender. An appropriate amount of viewer discretion is advised when exploring this algorithm with children. We also stress that the collection of images represents a particular biased subset of the human culture. Factors such as the bias of curators, patrons and art critics, as well as whether a work survives into the modern era all play a role. These factors can result in systematically fewer works by minorities, “outside artists” [57], and cultures lost to time. Furthermore, we note that the choice of dataset is a significant factor when considering the moral implications of the method, for instance applying CIR to facial recognition or resume retrieval requires a more careful analysis of bias before deploying to production.

In section [6] we demonstrate how CKNN data-structures can pinpoint issues in GANs. We hope this approach will be used by the community to identify and fix methods so that they better reflect the true diversity of the data. We note however that GANs that appear homogeneous with respect to a particular feature space and data-structure are not guaranteed to be fair, as more subtle traits might not have sufficient spatial locality in the feature space.

References

[1] The Metropolitan Museum of Art Open Access CSV, 2019. URL https://github.com/metmuseum/openaccess.

[2] The Rijksmuseum Open Access API, 2019. URL https://data.rijksmuseum.nl/.

[3] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16), pages 265–283, 2016.

[4] L. Amsaleg and H. Jégou. Datasets for approximate nearest neighbor search. 2017. URL: http://corpus-texmex.irisa.fr.

[5] A. Andoni, P. Indyk, T. Laarhoven, I. P. Razenshteyn, and L. Schmidt. Practical and optimal LSH for angular distance. CoRR, abs/1509.02897, 2015. URL http://arxiv.org/abs/1509.02897.

[6] M. Aumüller, E. Bernhardsson, and A. J. Faithfull. Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. CoRR, abs/1807.05614, 2018. URL http://arxiv.org/abs/1807.05614.

[7] Y. Bachrach, Y. Finkelstein, R. Gilad-Bachrach, L. Katzir, N. Koenigstein, N. Nice, and U. Paquet. Speeding up the xbox recommender system using a euclidean transformation for inner-product spaces. In Proceedings of the 8th ACM Conference on Recommender systems, pages 257–264, 2014.

[8] D. Baranchuk, A. Babenko, and Y. Malkov. Revisiting the inverted indices for billion-scale approximate nearest neighbors. CoRR, abs/1802.02422, 2018. URL http://arxiv.org/abs/1802.02422.

[9] D. Bau, J.-Y. Zhu, J. Wulff, W. Peebles, H. Strobelt, B. Zhou, and A. Torralba. Seeing what a gan cannot generate. 2019.

[10] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. S. Seljebotn, and K. Smith. Cython: The best of both worlds. Computing in Science & Engineering, 13(2):31–39, 2011.

[11] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. IEEE transactions on pattern analysis and machine intelligence, 35(8):1798–1828, 2013.

[12] J. L. Bentley. Multidimensional binary search trees used for associative searching. Communications of the ACM, 18(9):509–517, 1975.
[13] E. Bernhardsson. Approximate nearest neighbors oh yeah. https://github.com/spotify/annoy, 2019.

[14] N. Bhatia et al. Survey of nearest neighbor techniques. arXiv preprint arXiv:1007.0085, 2010.

[15] L. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. CoRR, abs/1706.05587, 2017. URL http://arxiv.org/abs/1706.05587.

[16] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In Advances in neural information processing systems, pages 2172–2180, 2016.

[17] L. Dagum and R. Menon. Openmp: an industry standard api for shared-memory programming. IEEE computational science and engineering, 5(1):46–55, 1998.

[18] S. Dasgupta and Y. Freund. Random projection trees and low dimensional manifolds. In Proceedings of the fortieth annual ACM symposium on Theory of computing, pages 537–546, 2008.

[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition, pages 248–255. Ieee, 2009.

[20] G. Dinu, A. Lazaridou, and M. Baroni. Improving zero-shot learning by mitigating the hubness problem. arXiv preprint arXiv:1412.6568, 2014.

[21] A. Fedosejev. React. js essentials. Packt Publishing Ltd, 2015.

[22] C. Fu and D. Cai. Efanna: An extremely fast approximate nearest neighbor search algorithm based on knn graph. arXiv preprint arXiv:1609.07228, 2016.

[23] X. Gao, T. Mu, J. Y. Goulermas, J. Thiyagalingam, and M. Wang. An interpretable deep architecture for similarity learning built upon hierarchical concepts. IEEE Transactions on Image Processing, 29:3911–3926, 2020.

[24] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2414–2423, 2016.

[25] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Advances in neural information processing systems, pages 2672–2680, 2014.

[26] A. Gordo, J. Almazán, J. Revaud, and D. Larlus. Deep image retrieval: Learning global representations for image search. In European conference on computer vision, pages 241–257. Springer, 2016.

[27] C. Gormley and Z. Tong. Elasticsearch: the definitive guide: a distributed real-time search and analytics engine. " O’Reilly Media, Inc.", 2015.

[28] E. Grave, A. Joulin, and Q. Berthet. Unsupervised alignment of embeddings with wasserstein procrustes, 2018.

[29] M. Hamilton, S. Raghuhatathan, A. Annavajhala, D. Kirsanov, E. de Leon, E. Barzilay, I. Matiach, J. Davison, M. Busch, M. Oprescu, et al. Flexible and scalable deep learning with mmlspark. arXiv preprint arXiv:1804.04031, 2018.

[30] M. Hamilton, S. Raghuhatathan, I. Matiach, A. Schonhoff, A. Raman, E. Barzilay, K. Rajendran, D. Banda, C. J. Hong, M. Knoritzer, et al. Mmlspark: Unifying machine learning ecosystems at massive scales. arXiv preprint arXiv:1810.08744, 2018.

[31] J. Hays and A. A. Efros. Scene completion using millions of photographs. ACM Transactions on Graphics (TOG), 26(3):4, 2007.
[32] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In European conference on computer vision, pages 630–645. Springer, 2016.

[33] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask R-CNN. CoRR, abs/1703.06870, 2017. URL http://arxiv.org/abs/1703.06870

[34] J. M. Hellerstein and M. Stonebraker. Predicate migration: Optimizing queries with expensive predicates. In Proceedings of the 1993 ACM SIGMOD international conference on Management of data, pages 267–276, 1993.

[35] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In Proceedings of the IEEE International Conference on Computer Vision, pages 1501–1510, 2017.

[36] M. Huh, P. Agrawal, and A. A. Efros. What makes imagenet good for transfer learning? arXiv preprint arXiv:1608.08614, 2016.

[37] F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer. Densenet: Implementing efficient convnet descriptor pyramids. arXiv preprint arXiv:1404.1869, 2014.

[38] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. SqueezeNet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. arXiv preprint arXiv:1602.07360, 2016.

[39] M. Iwasaki and D. Miyazaki. Optimization of indexing based on k-nearest neighbor graph for proximity search in high-dimensional data. arXiv preprint arXiv:1810.07355, 2018.

[40] Y. Jing, Y. Yang, Z. Feng, J. Ye, and M. Song. Neural style transfer: A review. CoRR, abs/1705.04058, 2017. URL http://arxiv.org/abs/1705.04058

[41] Y. Jing, Y. Yang, Z. Feng, J. Ye, Y. Yu, and M. Song. Neural style transfer: A review. IEEE Transactions on Visualization and Computer Graphics, page 1–1, 2019. ISSN 2160-9306. doi: 10.1109/tvcg.2019.2921336. URL http://dx.doi.org/10.1109/tvcg.2019.2921336

[42] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In European conference on computer vision, pages 694–711. Springer, 2016.

[43] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with gpus. IEEE Transactions on Big Data, 2019.

[44] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2017.

[45] D. E. Knuth. The art of computer programming, volume 3. Pearson Education, 1997.

[46] G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. In ICML deep learning workshop, volume 2, 2015.

[47] C. Le Corbeiller. China Trade Porcelain: Patterns of Exchange: Additions to the Helena Woolworth McCann Collection in the Metropolitan Museum of Art. Metropolitan Museum of Art, 1974.

[48] Y. LeCun and C. Cortes. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/

[49] A. Y. Levy, I. S. Mumick, and Y. Sagiv. Query optimization by predicate move-around. In VLDB, pages 96–107, 1994.

[50] J. Liao, Y. Yao, L. Yuan, G. Hua, and S. B. Kang. Visual attribute transfer through deep image analogy. arXiv preprint arXiv:1705.01088, 2017.

[51] L. Liao, X. He, B. Zhao, C.-W. Ngo, and T.-S. Chua. Interpretable multimodal retrieval for fashion products. In Proceedings of the 26th ACM international conference on Multimedia, pages 1571–1579, 2018.
[52] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In European conference on computer vision, pages 740–755. Springer, 2014.

[53] T. Liu, C. Rosenberg, and H. A. Rowley. Clustering billions of images with large scale nearest neighbor search. In 2007 IEEE Workshop on Applications of Computer Vision (WACV’07), pages 28–28. IEEE, 2007.

[54] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In Proceedings of International Conference on Computer Vision (ICCV), December 2015.

[55] D. G. Lowe. Object recognition from local scale-invariant features. In Proceedings of the seventh IEEE international conference on computer vision, volume 2, pages 1150–1157. Ieee, 1999.

[56] P. Lu, G. Huang, Y. Fu, G. Guo, and H. Lin. Learning large euclidean margin for sketch-based image retrieval. CoRR, abs/1812.04275, 2018. URL http://arxiv.org/abs/1812.04275

[57] J. Maizels. Outsider Art Sourcebook: Art Brut, Folk Art, Outsider Art. Raw Vision, 2009.

[58] C. D. Manning, P. Raghavan, and H. Schütze. Introduction to information retrieval. Cambridge university press, 2008.

[59] S. Marcel and Y. Rodriguez. Torchvision the machine-vision package of torch. In Proceedings of the 18th ACM international conference on Multimedia, pages 1485–1488, 2010.

[60] E. Marchiori. Class conditional nearest neighbor for large margin instance selection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(2):364–370, 2009.

[61] M. McCandless, E. Hatcher, O. Gospodnetić, and O. Gospodnetić. Lucene in action, volume 2. Manning Greenwich, 2010.

[62] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. VISAPP (1), 2(331–340):2, 2009.

[63] B. Nadler, S. Lafon, R. Coifman, and I. G. Kevrekidis. Diffusion maps—a probabilistic interpretation for spectral embedding and clustering algorithms. In Principal manifolds for data visualization and dimension reduction, pages 238–260. Springer, 2008.

[64] D. Newman. Bag of words data set, 2008.

[65] K. Nichol. Painter by numbers, wikiart, 2016.

[66] C. Olah, A. Mordvintsev, and L. Schubert. Feature visualization. Distill, 2017. doi: 10.23915/distill.00007. https://distill.pub/2017/feature-visualization.

[67] A. Oliva and A. Torralba. Building the gist of a scene: The role of global image features in recognition. Progress in brain research, 155:23–36, 2006.

[68] S. M. Omohundro. Five balltree construction algorithms. International Computer Science Institute Berkeley, 1989.

[69] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. the Journal of machine Learning research, 12:2825–2830, 2011.

[70] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pages 1532–1543, 2014.

[71] B. A. Plummer, P. Kordas, M. Hadi Kiapour, S. Zheng, R. Piramuthu, and S. Lazebnik. Conditional image-text embedding networks. In Proceedings of the European Conference on Computer Vision (ECCV), pages 249–264, 2018.
[72] L. Prokhorenkova. Graph-based nearest neighbor search: From practice to theory. *arXiv preprint arXiv:1907.00845*, 2019.

[73] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[74] S. E. Reed, Y. Zhang, Y. Zhang, and H. Lee. Deep visual analogy-making. In *Advances in neural information processing systems*, pages 1252–1260, 2015.

[75] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381, 2018. URL http://arxiv.org/abs/1801.04381

[76] B. M. Sarwar, G. Karypis, J. A. Konstan, J. Riedl, et al. Item-based collaborative filtering recommendation algorithms. *Www*, 1:285–295, 2001.

[77] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[78] J. B. Tenenbaum and W. T. Freeman. Separating style and content with bilinear models. *Neural computation*, 12(6):1247–1283, 2000.

[79] C. Traina, A. J. M. Trains, and J. M. de Figuciredo. Including conditional operators in content-based image retrieval in large sets of medical exams. In *Proceedings. 17th IEEE Symposium on Computer-Based Medical Systems*, pages 85–90, 2004.

[80] A. Veit, S. Belongie, and T. Karaletsos. Conditional similarity networks, 2016.

[81] T. Volker. *Porcelain and the Dutch East India Company: as recorded in the Dagh-Registers of Batavia Castle, those of Hirado and Deshima and other contemporary papers; 1602-1682*, volume 11. Brill Archive, 1954.

[82] S. v. d. Walt, S. C. Colbert, and G. Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.

[83] J. Wang, H. T. Shen, J. Song, and J. Ji. Hashing for similarity search: A survey. *CoRR*, abs/1408.2927, 2014. URL http://arxiv.org/abs/1408.2927

[84] X. Wang. A fast exact k-nearest neighbors algorithm for high dimensional search using k-means clustering and triangle inequality. In *The 2011 International Joint Conference on Neural Networks*, pages 1293–1299. IEEE, 2011.

[85] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[86] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. *CoRR*, abs/1611.05431, 2016. URL http://arxiv.org/abs/1611.05431

[87] D. L. Yamins, H. Hong, C. F. Cadieu, E. A. Solomon, D. Seibert, and J. J. DiCarlo. Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the National Academy of Sciences*, 111(23):8619–8624, 2014.

[88] D. Yan, Y. Wang, J. Wang, H. Wang, and Z. Li. K-nearest neighbors search by random projection forests. *IEEE Transactions on Big Data*, 2019.

[89] P. N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Soda*, volume 93, pages 311–21, 1993.

[90] P. Zezula, P. Savino, G. Amato, and F. Rabitti. Approximate similarity retrieval with m-trees. *The VLDB Journal*, 7(4):275–293, 1998.

[91] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016.
A Website

Figure 7: Example of the conditioning abilities of MosAIc. The query image (top left) is provided along with a medium conditioner restricts retrieved images to sculptures. Selecting a match from the carousel below brings it up beside the query image for comparison.

Figure 8: Using SHAP to explain the similarity between the query image and any requested match. Unimportant pixels have been masked. In the example above, the subject of the work is selected as an important contribution to the similarity.

As an application of CIR for the public, we introduce MosAIc (aka.ms/mosaic), a website that allows users to explore art matches conditioned on culture and medium. MosAIc’s front-end is built on React [21], and its back-end is built on Azure Kubernetes Service and Azure App Services. Figure 7 provides an example of a user query conditioned on the “sculpture” label. This query yields a Dutch sculpture that depicts a female wearing a blue dress and an outer layer across the shoulders, much like the query image. Users can view rows of matched artwork returned for a specific query to see multiple matches for each conditioner. MosAIc also leverages SHAP to explain match rationales as in Figure 8 and Azure Search to provide text-based querying in addition to CIR. This allows users to find and select specific works from the collection to explore with CIR.

B Visualizing Failure Cases

Figure 9 (a) shows how conditioners that do not share a common support can yield low diversity conditional neighbors. Though sharing a common support is certainly helpful, it is not mandatory as shown by Figure 9 (b). Some potential mitigations for these effects could be to fine tune learned embeddings to promote diverse queries, or to re-weight query outputs based on diversity. Additionally, an initial alignment with an optimal transport method could mitigate these effects [28].

Figure 9: A schematic illustration of how conditional KNN can yield to a lack of diversity in particular geometries. (a) shows how low diversity can occur when there is no overlap of supports. Figure (b) shows how support intersection is not necessary for quality alignment

C Evaluating the Overhead of Conditional Retrieval

We find that our CKNN data-structures do not contribute overhead to “unconditional” queries where $S = X$. Figure 10 shows a box-plot of query times for unconditional queries on the GIST 960 embeddings, this demonstrate that checking conditioner subsets do not affect the distribution of query times. Table 3 shows this affect on a broad class of large-scale benchmarking datasets with both
euclidean and angular distance. For both Figure 10 and Table 3, we use training and testing sets from [6].

Figure 10: Performance of CKNN in unconditional case when $S = \mathcal{X}$. Checking the remaining node set does not noticeably affect query time for both KD and Ball Tree (BT) methods. BF represents a Brute Force comparison.

| Dataset   | Size    | Dim | Metric | Method | BF       | BT       | CBT      | KD       | CKD      | Annoy    |
|-----------|---------|-----|--------|--------|----------|----------|----------|----------|----------|----------|
| FMNIST    | 60,000  | 784 | E      | BF     | 122 ± 3  | 62 ± 2   | 62 ± 2   | 67 ± 8   | 68 ± 8   | 14 ± 3   |
| GIST      | 1,000,000 | 960 | A      | BT     | 2511 ± 58| 1375 ± 89| 1330 ± 14| 1618 ± 276| 1599 ± 256| 45 ± 14  |
| Glove     | 1,183,514 | 100 | A      | CBT    | 446 ± 9  | 415 ± 7  | 441 ± 9  | 447 ± 22 | 473 ± 21 | 17 ± 2   |
| Glove     | 1,183,514 | 200 | A      | KD     | 745 ± 9  | 454 ± 11 | 613 ± 11 | 530 ± 11 | 682 ± 10 | 20 ± 3   |
| Glove     | 1,183,514 | 25  | A      | CKD    | 232 ± 8  | 145 ± 17 | 147 ± 17 | 64 ± 38  | 67 ± 38  | 12 ± 1   |
| Glove     | 1,183,514 | 50  | A      | Annoy  | 316 ± 6  | 240 ± 5  | 246 ± 5  | 246 ± 34 | 252 ± 34 | 14 ± 1   |
| MNIST     | 60,000  | 784 | E      | BF     | 121 ± 2  | 63 ± 2   | 63 ± 1   | 68 ± 15  | 68 ± 15  | 18 ± 2   |
| NYT       | 290,000 | 256 | A      | BT     | 215 ± 3  | 115 ± 13 | 120 ± 13 | 108 ± 59 | 112 ± 61 | 20 ± 1   |
| SIFT      | 1,000,000 | 128 | A      | CBT    | 436 ± 4  | 192 ± 37 | 270 ± 51 | 146 ± 59 | 190 ± 77 | 13 ± 1   |

Table 3: Unconditional benchmarking results on reference datasets from [6]. Results show that our approach does not introduce overheads in the unconditional setting and can be used to broaden the family of KNN operations without sacrificing speed. We include a single approximate KNN approach, Annoy, to allow for comparison between our exact KNN baselines, and a widely used approximate KNN method. Metric “A” and “E” represent angular and euclidean distance respectively.