QIK: A System for Large-Scale Image Retrieval on Everyday Scenes With Common Objects

Arun Zachariah
azachariah@mail.missouri.edu
University of Missouri-Columbia
Columbia, Missouri

Mohamed Gharibi
mggvf@mail.umkc.edu
University of Missouri-Kansas City
Kansas City, Missouri

Praveen Rao
praveen.rao@missouri.edu
University of Missouri-Columbia
Columbia, Missouri

ABSTRACT

In this paper, we propose a system for large-scale image retrieval on everyday scenes with common objects by leveraging advances in deep learning and natural language processing (NLP). Unlike recent state-of-the-art approaches that extract image features from a convolutional neural network (CNN), our system exploits the predictions made by deep neural networks for image understanding tasks. Our system aims to capture the relationships between objects in an everyday scene rather than just the individual objects in the scene. It works as follows: For each image in the database, it generates most probable captions and detects objects in the image using state-of-the-art deep learning models. The captions are parsed and represented by tree structures using NLP techniques. These are stored and indexed in a database system. When a user poses a query image, its caption is generated using deep learning and parsed into its corresponding tree structures. Then an optimized tree-pattern query is constructed and executed on the database to retrieve a set of candidate images. Finally, these candidate images are ranked using the tree-edit distance metric computed on the tree structures. A query based on only objects detected in the query image can also be formulated and executed. In this case, the ranking scheme uses the probabilities of the detected objects. We evaluated the performance of our system on the Microsoft COCO dataset containing everyday scenes (with common objects) and observed that our system can outperform state-of-the-art techniques in terms of mean average precision for large-scale image retrieval.

CCS CONCEPTS

• **Information systems** → **Multimedia and multimodal retrieval**; **Image search**.

KEYWORDS

image retrieval; deep learning; NLP; indexing; ranking

ACM Reference Format:
Arun Zachariah, Mohamed Gharibi, and Praveen Rao. 2020. QIK: A System for Large-Scale Image Retrieval on Everyday Scenes With Common Objects. In *2020 International Conference on Multimedia Retrieval (ICMR'20), June 8–11, 2020, Dublin, Ireland.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3372278.3390682

ICMR '20, June 8–11, 2020, Dublin, Ireland

ACM ISBN 978-1-4503-7087-5/20/06…$15.00
https://doi.org/10.1145/3372278.3390682

1 INTRODUCTION

Content-based image retrieval (CBIR) has been a topic of research for many years. In CBIR, given a query image, the goal is to find images in the database that are similar to the query image. Typically, an image is mapped to a feature vector and a similarity metric is defined between feature vectors and used to identify images similar to the query image. Advances in computer vision and deep learning have resulted in numerous research efforts and commercial solutions for CBIR. As image datasets continue to grow in volume on the Web and in domains such as healthcare, insurance, precision agriculture, remote sensing, astronomy, and defense, there continues to be great interest in efficient large-scale image retrieval.

Current image retrieval systems typically have two stages: the filtering stage to identify a set of candidate images and a re-ranking stage, where a small number of similar candidates are re-ranked based on specific criteria. Several approaches used hand-crafted local features [7, 25] and techniques based on bag-of-words method, hamming embedding, large vocabularies, spatial matching, etc., to represent images in high-dimensional space [13, 19–21, 31, 32, 37, 42]. For re-ranking, a few strategies have been proposed based on local features and geometric verification [5, 31, 32].

Advances in CNNs have resulted in new methods for image understanding including image recognition and object detection. Deep CNNs have achieved remarkable accuracy for object recognition on standard benchmarks (*e.g.*, GoogLeNet [39], Inception v3 [40], ResNet [18]). Another interesting success is the development of image captioning models like Show and Tell [45] and Show, Attend and Tell [46] that use a combination of convolutional layers/networks and recurrent neural networks. Thus, it is now more feasible than before to extract useful knowledge from images and understand their context on a large-scale.

Several techniques have recently explored the use of CNNs for large-scale image retrieval. They rely on CNN features for global image representations enabling fast filtering [4, 6, 15, 23, 33, 43, 47]. For re-ranking, local image representations from CNNs have been employed through spatial matching and geometric verification [27–29]. DELF [29, 41] extracts deep local features from CNNs for indexing and ranks based on geometric verification. An instance retrieval technique, which we will hereinafter refer to as FR-CNN [36], leverages both local and global features from a Faster R-CNN [34].

Unlike prior approaches that rely on features constructed from CNNs developed for image understanding tasks, we explore if predictions made by such networks for tasks such as image captioning and object detection can be used in a novel and effective way for large-scale image retrieval. Thus, rather than developing local or

global descriptors for images by directly using CNN-based features, we look ahead in the processing pipeline to employ the actual predictions. Specifically, we investigate how both the filtering and ranking steps during image retrieval can benefit from these predictions. We will refer to the predictions made on an image collectively as the *probabilistic image understanding* (PIU) of the image.

In this paper, we propose a system called QIK (**Q**uerying **I**mages Using Contextual **K**nowledge) for large-scale image retrieval on everyday scenes (with common objects) by synergistically combining PIUs and NLP. The key contributions of our work are as follows:

- QIK is a generic framework that aims to capture the context of everyday scenes and learn the relationships between objects in them for efficient image retrieval. For this purpose, QIK generates the PIU of an image using state-of-the-art image captioning and object detection models. The PIUs are processed during image retrieval.
- QIK employs modern techniques in NLP for linguistic analysis of the PIUs. For instance, image captions are transformed into tree structures widely used in computational linguistics [22]. These tree structures are queried during the filtering step of image retrieval and later used for ranking the candidate images.
- We conducted an evaluation of QIK against state-of-the-art techniques such as DELF [29], DIR [15], CroW [23], and FR-CNN [36] using the Microsoft COCO dataset [24], which is a well-known dataset containing complex everyday scenes with common objects. We computed the mean average precision (mAP) for top-k matches of a query to compare QIK and its competitors. First, we observed that using image captions for filtering and ranking, QIK performed better than when only the detected objects were used for image retrieval in terms of mAP. This asserts that leveraging relationships between objects during image retrieval yields better quality results. Furthermore, QIK using image captions outperformed the aforementioned competitors in terms of mAP.

The rest of the paper is organized as follows: Section 2 discusses recent related work and our motivation; Section 3 presents the design details of QIK; Section 4 discusses the implementation and performance evaluation of QIK and comparison with state-of-the-art techniques; and finally, we conclude in Section 5.

2 RELATED WORK AND OUR MOTIVATION

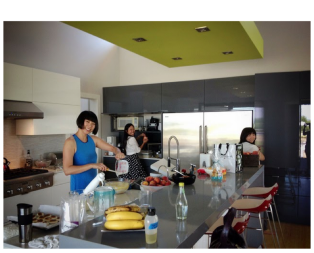
Related Work. We briefly discuss recent techniques that use CNN-based features for image retrieval. FR-CNN [36] uses image-wise and region-wise representations pooled from an object detection CNN such as Faster R-CNN [34]. The image-wise representations serve as global features and are used during the filtering step. The region-wise representations serve as local features and are used for re-ranking. DIR [15] produces a global and compact fixed-length representation of each image by aggregating many region-wise descriptors so that it is robust to scale and transformation. The regions to be pooled are predicted using a region proposal network. DELF [29] is a local feature descriptor that employs an attention-based keypoint selection mechanism to identify semantically useful local features needed for image retrieval. A more recent work by Tiechman *et al.* [41] proposes a novel region aggregation method for image retrieval. It extracts DELF features and object regions from an image and extends the idea of aggregated selective math kernels [42]. The region aggregation method helps in re-balancing the visual information in an image. CroW [23] is an efficient non-parametric weighting and aggregation scheme to transform convolutional image features to a compact global image feature. The output of the convolutional layers are aggregated before the fully connected layers. Gordo *et al.* [16] used human-annotated region level captions during training time to generate global visual representation of images for semantic retrieval. At query time, only the query image is used without any captions. VistaNet [44] combines text with images but for sentiment analysis.

Our Motivation. Prior image retrieval approaches that rely on CNN-based features have been tested on datasets that contain objects such as buildings, scenic views, and landmarks (*e.g.*, the Oxford Dataset [31], the Paris Dataset [32], the INRIA Dataset [19], the Google Landmarks Dataset [2]). Although some of them have been extended with 100k distractor images (*e.g.*, Paris 106k, Oxford 105k), the evaluated queries were still on buildings, landmarks, etc. Images containing everyday scenes with common objects are quite different from these images as they contain objects in the foreground and background. In such an image, certain objects become the main focus of the image when a human observes it. Based on our experiments using the MS COCO dataset, we observed that techniques based on CNN-based features failed to precisely capture the main aspect of an image leading to false positives. Figure 1 shows a set of query images as well as false positives output by the evaluated techniques. For the query image in Figure 1(a), DELF returned an image of a kitchen, however, without people in it (Figure 1(b)). For the query image in Figure 1(c), FR-CNN returned an image with baseball players instead of skateboarders possibly due to the hand and leg movements (Figure 1(d)). For the query image in Figure 1(e), CroW returned an image of a room with daylight but without people (Figure 1(f)). Finally, for the query image in Figure 1(g), DIR returned an image of a space similar to the query image but without a person wearing a tie (Figure 1(h)).

We emphasize that human cognition can capture the key essence of an image and describe it aptly via a caption; it can ignore objects (or regions) in an image that do not really matter to describe the main context of the image. Hence, *we posit that an accurate image captioning system can be very useful to describe images containing everyday scenes with common objects*; image retrieval based on automatically generated captions of images could perform better than techniques relying on CNN-based features. This motivated us to design a new approach called QIK to provide superior image retrieval performance than its competitors for images containing everyday scenes. Our approach moves away from directly building global or local image descriptors using features of CNNs/deep neural networks. Instead, it aims to use the predictions made by these networks for image understanding tasks, *i.e.*, PIUs of images, in a novel way and employ modern NLP techniques for efficient and accurate large-scale image retrieval.

3 PROPOSED DESIGN

In this section, we first present the design of QIK for large-scale image retrieval on everyday scenes with common objects. Our



(a) Query image for DELF [29]

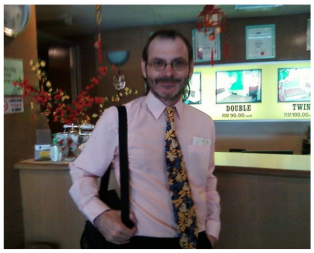
(b) A false positive by DELF

(c) Query image for FR-CNN [36]

(d) A false positive by FR-CNN

(e) Query image for CroW [23]

(f) A false positive by CroW



(g) Query image for DIR [15]

(h) A false positive by DIR

Figure 1: On the left, we show a query image from the MS COCO dataset used for image retrieval. On the right, we show a false positive output by a technique using CNN-based features when retrieving the top-k matches for the query image. We set $k = 8$.

design is motivated by the fact that recent techniques based on CNN-based features [15, 23, 29, 36] may fail to fetch precise matches for everyday scenes. We begin with an overview of QIK followed by the technical details on how QIK indexes an image repository and performs image retrieval using PIUs.

3.1 Overview of QIK

Figure 2 shows the architecture of QIK. Given an image repository, QIK generates PIU for each image using state-of-the-art deep learning models for image captioning and object detection. Pre-trained models can be used. As a result, QIK captures the context of everyday scenes and learns the relationships between objects in them. The generated PIU will contain most probable captions and objects. On each caption, QIK constructs a sentence parse tree and a dependency tree [22], which have been widely studied in NLP and computational linguistics. The entire collection of trees, which are ordered trees, are represented in XML [10]. The XML documents are stored and indexed using an XML database system. The other contents of the PIU (*i.e.*, detected objects) are also indexed.

QIK can use either captions or detected objects for image retrieval. In the first case, given a query image, QIK generates the most probable caption (using the same image captioning model) and the associated parse and dependency trees. Using the parse tree, QIK generates an optimized XPath query [8] containing only essential keywords in the caption while preserving the ordering between these keywords and their relationships. After executing the XPath query on the XML documents, a set of candidate images are retrieved. Finally, the candidate images must be ranked as the database system does not enforce any ordering between the candidate images. In order to rank and return the top-k relevant images, QIK relies on tree edit distance [9], which is the *minimum cost* to transform one tree to another via node insert, delete, and relabel operations. For each candidate image, the tree edit distance between its caption's parse tree (or dependency tree) and the parse tree (or dependency tree) of the query image's caption is computed. Finally, the candidate images are ranked in increasing order of the computed tree edit distance, and the top-k matches are returned to the user. More details will be provided later in this section.

In the second case, given a query image, QIK detects objects in an image (using an object detection model [34, 35]) with probability greater than a user-specified threshold. It then retrieves a set of candidate images that contain all of the detected objects. For ranking, it combines the probabilities of the detected objects in the query image and in a candidate image to compute a score for the image. The candidate images are ranked in decreasing order of the score, and the top-k matches are returned to the user. More details will be provided later in this section.

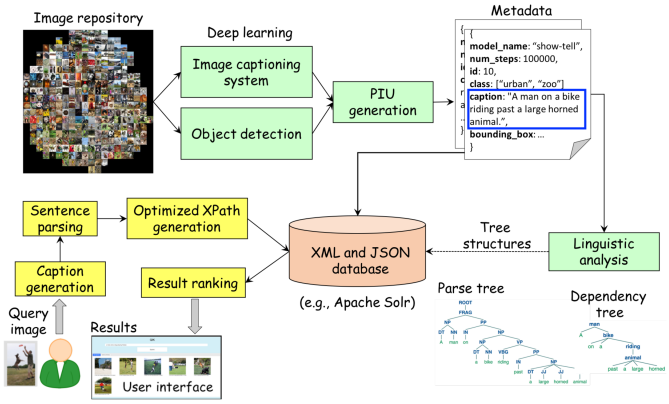


Figure 2: Architecture of QIK

3.2 PIU Generation, Linguistic Analysis and Indexing

QIK goes beyond the raw CNN-features used by other approaches. It generates PIU for each image in the image repository by leveraging the predictions made by deep neural networks developed for image understanding tasks. For generating image captions, each image is provided to a pre-trained image captioning model to predict the most probable captions. These captions denote the context of the everyday scene in the image and also describe the relationship between objects in the scene. In addition, each image is run against an object detection model to identify the most probable objects in the image (with probability greater than a user-defined threshold). The bounding boxes can also be stored as part of the image's PIU. QIK is a generic framework because it is designed to accommodate multiple image understanding models. The PIU generated from different models can be queried together during image retrieval.

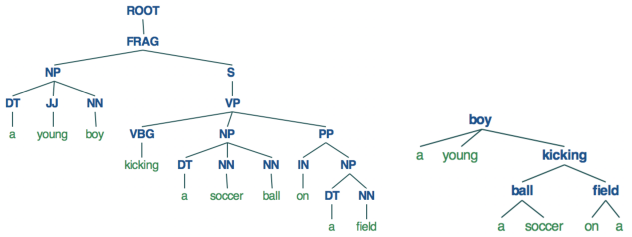
Figure 3: An example image (Source: Flickr30K [48])

Example 3.1. Consider an image shown in Figure 3. The Show and Tell captioning model produces three captions by default. The most probable caption is "a young boy kicking a soccer ball on a field." Suppose an object detection model identifies two objects with probabilities greater than 0.9. Suppose these two objects are "soccer ball" and "person". Together, they constitute the PIU of the image.

QIK employs state-of-the-art NLP techniques to perform linguistic analysis of the image PIUs. Specifically, the image captions are analyzed linguistically. For each caption, QIK constructs its sentence parse tree and dependency tree. A *parse tree* represents the structure of the sentence/phrase based on the grammar rules by identifying tokens denoting noun phrases, nouns, verb phrases, verbs, adjectives, determiners, prepositions, etc., in the sentence/phrase. These tokens are based on parts-of-speech (POS) tagging in computational linguistics [22]. A *dependency tree* on the other hand provides a representation of how words in a sentence/phrase are connected by syntactic dependencies [22].

Example 3.2. Consider the caption "a young boy kicking a soccer ball on a field." Figure 4(a) shows the parse tree of the caption output by the Stanford Parser [38]. Figure 4(b) shows the corresponding dependency tree output by the Stanford Parser [12].

QIK uses both JSON and XML data models to store the PIUs of images. This is because of the nature of contents in a PIU. For instance, the parse and dependency trees are ordered trees; hence, XML is the appropriate choice as it preserves ordering of the tree



(a) Parse tree (b) Dependency tree

Figure 4: Examples of parse and dependency trees

nodes–this is critical when processing a query image during image retrieval. However, for storing detected objects, their bounding boxes, etc., it is beneficial to store in JSON as it consumes less storage space.Both the JSON and XML documents representing the PIUs are stored and indexed to enable fast query processing.

```
<ROOT><FRAG><NP><DT>a</DT><JJ>young</JJ><NN>boy</NN></NP><S>
<VP><VBG>kicking</VBG><NP><DT>a</DT><NN>soccer</NN>
<NN>ball</NN></NP><PP><IN>on</IN><NP><DT>a</DT><NN>field</NN>
</NP></PP></VP></S></FRAG></ROOT>
```

(a) Parse tree representation

```
<boy><a/><young/><kicking><ball><a/><soccer/></ball>
<field><on/><a/></field></kicking></boy>
```

(b) Dependency tree representation

Figure 5: XML representation

Example 3.3. Figure 5(a) shows the XML document for the parse tree in Figure 4(a). The leaf nodes of the parse tree are represented as text nodes in XML. The dependency tree is also represented similarly. However, the leaf nodes are treated as XML elements as shown in Figure 5(b).

Algorithm 1 sketches the key steps involved in generating, analyzing, and indexing the PIUs in QIK. For each image in the repository, the most probable captions are generated and their parse and dependency trees are constructed and stored in an XML database. The objects detected in the image with a probability greater than a user-specified threshold are also indexed.

3.3 Retrieval Using Captions in PIUs

We first show how QIK processes a query image when using image captions and returns the top-k relevant matches to the user. Algorithm 2 sketches the overall steps during image retrieval. The filtering step is denoted by Lines 1-8. The caption of the query image is first predicted using the same pre-trained image captioning model that was used during indexing. The most probable objects with probability greater than a user-defined threshold are also predicted. For the caption, Algorithm 3 is invoked to generate a basic XPath query. After that, an optimized XPath query is generated by invoking Algorithm 4. A query for objects can also be generated. The optimized XPath query is executed to obtain an initial set of qualifying image IDs. The query for objects is also executed to

Algorithm 1 IndexPIU(*img*)

Input: *img* denotes an image in the database

Predict the most probable captions C of *img*
Predict the most probable objects O in *img* with probability greater than a user-defined threshold
for each caption $c \in C$ **do**
    Generate the parse tree p for c
    Generate the dependency tree d for c
    Represent p in XML
    Represent d in XML
    Store and index the XML documents in the database system
for each object $o \in O$ **do**
    Represent o as a JSON record containing the probability of o
    Store and index the JSON record in the database system

Algorithm 2 RetrieveImages(k, img_q)

Input: k denotes the maximum number of matches to return
Input: img_q denotes the query image

Predict the most probable caption C for img_q
Predict the most probable objects O in img_q
$q \leftarrow GenerateBasicXPath(C)$
$q' \leftarrow GenerateOptimizedXPath(q)$
Generate a query q'' for O
Execute q' on the XML database to obtain set of image IDs
Execute q'' on the JSON database to obtain set of image IDs
Compute the intersection of the above two sets to obtain the candidate images
for each candidate image img_c **do**
    Compute tree edit distance between the parse tree (or dependency tree) of the caption of img_c and the parse tree (or dependency tree) of the caption C
Sort (in ascending order) the candidate images based on the computed tree edit distance values
return top-k matches

obtain another set of qualifying image IDs. The intersection of these two sets produces the candidate image IDs that match the criteria specified in the XPath query (based on captions) as well as the objects in the query image.

Next, we provide a brief introduction to XPath [8], a query language for selecting nodes in an XML document. A simple XPath query can be written as $/A_1::N_1[p_1]/\cdots/A_i::N_i[p_i]/\cdots/A_n::N_n[p_n]$, where A_i denotes an XPath axis, N_i denotes an XML element name, and p_i denotes a predicate of that node. A predicate may be empty for a node. Although there are 13 XPath axes [8], we only use 4 of them: (a) `child` to indicate a parent-child relationship, (b) `descendant` to indicate an ancestor-descendant relationship, (c) `following` to indicate that a node follows the other in document order (a.k.a. preorder), and (d) `following-sibling` to indicate that two nodes share a common parent.

Next, we discuss the details of basic XPath generation (Algorithm 3) and optimization (Algorithm 4) performed during image retrieval. In Algorithm 3, a query caption is first parsed into its parse tree, which is then pruned by removing non-essential keywords such as "on", "a", "in", and others (Line 2). We basically ignore prepositions, determiners, conjunctions, etc., in the query image's caption during the filtering step. The pruned tree is traversed in preorder (Lines 3-14) to generate a basic XPath query containing XPath axes such as `child`, `following`, and `following-sibling` to preserve the ordering of essential keywords in the caption.

Algorithm 3 GenerateBasicXPath(C)

Input: C denotes the image caption
Output: An XPath expression

Let D denote the parse tree of C
Prune D by removing subtrees rooted at POS tags such as `DT`, `IN`, and other non-essential keywords
for each node n in preorder traversal of D **do**
    Let `t` denote node label of n
    if n is the root node of D **then**
        $q \leftarrow$ `/child::t`
    else if n is child of the previous node (in preorder) **then**
        Append `/child::t` to q
    else if n is a sibling of the previous node (in preorder) **then**
        Append `/following-sibling::t` to q
    else if n is a leaf node **then**
        Append `[text()="t"]` to q
    else
        Append `/following::t` to q

Example 3.4. Let us suppose the caption predicated for a query image is "a young boy kicking a soccer ball on a field." A parse tree is constructed as shown in Figure 4(a). Algorithm 3 produces the basic XPath query as shown in Figure 6.

```
/child::ROOT/child::FRAG/child::NP/child::JJ[text()="young"]/
following-sibling::NN[text()="boy"]/following::S/
child::VP/child::VBG[text()="kicking"]/
following-sibling::NP/child::NN[text()="soccer"]/
following-sibling::NN[text()="ball"]/following::PP/
child::NP/child::NN[text()="field"]
```

Figure 6: Basic XPath query

From the basic XPath query, an optimized query is generated using Algorithm 4. The key idea is to replace a sequence of XPath axes with a single axis that still specifies the same constraint on the keywords as the original query in order to reduce the length of the XPath query in terms of the number of nodes and axes. The XPath query is processed from left-to-right, one axis-node pair at a time. Each time a node containing a predicate is encountered, an XPath axis and node name are appended to the optimized query (Lines 3-10). The axis type depends on the sequence of axes encountered since the previous node with a predicate. A sequence of `child` axes from the root node of the query to the first node with a predicate is replaced by the `child` axis (Line 9). When only one axis appears (*e.g.*, `child`, `following`, `following-sibling`) since the previous node with a predicate (*e.g.*, an adjective and its noun), it is replaced by that axis (Line 9). A sequence of axes where the leading axis is `following` or `following-sibling` since the previous node with a

predicate is replaced by `following` (Line 4 or Line 6). The first axis of the generated query (`child`) is replaced finally by `descendant`.

Algorithm 4 GenerateOptimizedXPath(q)

Input: q denotes an input XPath expression
Output: Optimized XPath expression

```
1:  q' ← NULL; leadingAxis ← NULL
2:  for each axis x and node n in q (from left-to-right) do
3:    if n has predicate p then
4:      if leadingAxis is following then
5:        Append /following::n[p] to q'
6:      else if leadingAxis is following-sibling then
7:        Append /following::n[p] to q'
8:      else
9:        Append /x::n[p] to q'
10:     leadingAxis ← NULL
11:   else if x is child then
12:     continue
13:   else if x is following::sibling or following then
14:     leadingAxis ← x
15:   else
16:     print("Invalid axis");
17:     return NULL
18: Replace the first axis in q' with descendant
19: return q'
```

Example 3.5. Algorithm 4 transforms the basic XPath query shown in Figure 6 into an optimized XPath query (Figure 7).

```
/descendant::JJ[text()="young"]/following-sibling::NN[text()="boy"]/
following::VBG[text()="kicking"]/following::NN[text()="soccer"]/
following-sibling::NN[text()="ball"]/following::NN[text()="field"]
```

Figure 7: Optimized XPath query

After the candidate images are obtained, they are ranked using the tree edit distance metric. As shown in Algorithm 2, we compute the tree edit distance between the parse (or dependency tree) of the candidate image's caption and the parse tree (or dependency tree) of the query image's caption. We rank the candidate images by increasing order of the computed tree edit distance value and return the top-k matches to the user.

3.4 Retrieval Using Detected Objects in PIUs

In this section, we show how QIK processes a query using detected objects in images and returns the top-k relevant matches to the user. Algorithm 5 sketches the overall steps during image retrieval. The filtering step is denoted by Lines 1-3. First, given a query image, the objects detected in it (using an object detection model) that have probabilities greater than a user-specified threshold are selected. Then a Boolean AND query is constructed on these objects. The JSON database is queried to fetch all the candidate images that contain every object in the query. In the ranking step (Lines 4-7), a score is computed for each candidate image by using the probabilities of the detected objects. Given a candidate image, its score is the sum of the product of the probabilities of the selected objects in the candidate image and the query image. The intuition is that a candidate image containing objects of higher probabilities will yield a higher score. A higher probability object will dominate the score over a lower probability object in the query image. Finally, the images are sorted in descending order by their scores, and the top-k results are returned.

Algorithm 5 RetrieveImagesObj(k, img_q)

Input: k denotes the maximum number of matches to return
Input: img_q denotes the query image

1: Let $O = \{(o_1, p_1), (o_2, p_2), \ldots, (o_n, p_n)\}$ denote the most probable objects and their probabilities in img_q such that p_i is greater than a user-specified threshold
2: Generate a query q to specify a Boolean AND of the objects $o_1, o_2, \ldots, o_n$
3: Execute q on the JSON database to obtain set of image IDs along with the probabilities of the objects in each candidate image
4: **for** each candidate image img_c **do**
5: Let $O_c = \{(o_1, r_1), (o_2, r_2), \ldots, (o_n, r_n)\}$ denote the matched objects and their probabilities in the candidate image
6: score(img_c) = $\sum_{i=1}^{n} p_i \times r_i$
7: Sort (in descending order) the candidate images based on their scores
8: **return** top-k matches

4 PERFORMANCE EVALUATION

In this section, we report the performance evaluation of QIK and compare it with four different image retrieval techniques that utilize global and/or local feature descriptors, namely, DIR [15], DELF [29], CroW [23], FR-CNN [36], and LIRE [26]. LIRE is an open source CBIR system that provides a wide range of options such as color histograms, color and edge directivity descriptor (CEDD), etc., to extract local features and other well-known techniques [7, 25] to extract global feature vectors. These are then indexed for fast image retrieval. We used the original code published by the authors of DIR, DELF, CroW, FR-CNN, and LIRE for comparison with QIK.

4.1 Implementation and Experimental Setup

QIK was primarily written in Java and compiled using Java 1.8. The parse tree and dependency trees for captions were generated using the Stanford Parser package (version 3.9.2) [3]. The entire compiled application was deployed on Apache Tomcat 9.0.20. XML data was stored and indexed using BaseX [1, 17] (version 9.2), a high performance XML engine. QIK used the APTED library [30], a robust, main-memory implementation, for computing tree edit distance required during the ranking step.

QIK used a pre-trained Inception v3 model [40] for initializing the parameters of Show and Tell [45] for generating captions of images. The training was done for 3 million steps on a single NVIDIA GeForce Titan X Pascal 12GB GPU on the MS COCO dataset [24] comprising of 83K training images and 41K validation images. The accuracy of the model was further improved by executing a second round of training for 2 million steps to enable fine tuning of the

Inception v3 model. For object detection, QIK used Faster-RCNN with NASNet-A image featurization [49] trained on MS COCO.

We ran the experiments on CloudLab [14] in the Wisconsin data center. All nodes had two Intel Xeon Silver 4114 10-core CPUs (2.20 GHz) and 192 GB of RAM, and ran Ubuntu 16.04.

4.2 Dataset

For evaluation, we used MS COCO [24] containing 124K images of complex everyday scenes, involving 80 common objects in their natural context. Each image had 5 human-annotated captions. We chose a random subset of 15K images for evaluation as some of the competitors of QIK could not operate on larger number of images.

4.3 Queries

For the 15K dataset, we generated two-, three-, and four-object combinations using the 80 objects specified in MS COCO such as "person + couch", "person + car + cup", etc. The number of two-object, three-object, and four-object combinations were 50, 50, and 40, respectively. Consider only two-object combinations. For each combination c, we did the following: We selected the images containing those two objects based on human labeling from the 15K dataset. Let I denote the selected images. For each image $i \in I$, we identified the true matches for i as a query image using a pre-trained Universal Sentence Encoder [11] model. Essentially, we computed the similarity between the human-annotated captions of i against the human-annotated captions of other images in I and used a similarity threshold τ to determine a true match. That is, if any caption of i was similar to a caption of an image j in I with similarity greater than τ, then j was considered a true match for i. Thus, we completely relied on human judgment by using their annotations for deciding the true matches for an image query. We computed the mAP value for the combination c for different top-k matches (k=2, k=4, k=8, and k=16). We followed the same procedure for three-object and four-object combinations. The total number of image queries in two-object, three-object, and four-object combinations were 4406, 1585, and 561, respectively.

4.4 Results

Next, we present the image retrieval performance of QIK and its competitors in terms of mAP and retrieval time.

4.4.1 QIK*: Captions vs. Detected Objects.* We first compared the image retrieval performance of QIK when using captions versus detected objects in PIUs. Hereinafter, we denote them by $\texttt{QIK}_c$ and $\texttt{QIK}_o$, respectively. Note that $\texttt{QIK}_c$ used Algorithm 2 and $\texttt{QIK}_o$ used Algorithm 5. Our goal was to show that captions provide superior performance as they can capture the relationships between important objects in an image compared to just retrieving images containing certain objects. Table 1, Table 2, and Table 3 show the average of the mAP values for the two-object, three-object, and four-object combinations, respectively. Clearly, $\texttt{QIK}_c$ outperformed $\texttt{QIK}_o$ for two different probability thresholds for object detection, *i.e.*, 0.9 and 0.8. Thus, one can conclude that captions in PIUs indeed capture the object relationships leading to superior image retrieval performance for everyday scenes.

Table 1: $\texttt{QIK}_c$ vs $\texttt{QIK}_o$: two-object combinations (avg. mAP)

	τ=0.6				τ=0.7			
	k=2	k=4	k=8	k=16	k=2	k=4	k=8	k=16
$\texttt{QIK}_c$	**0.94**	**0.96**	**0.96**	**0.94**	**0.81**	**0.85**	**0.84**	**0.81**
$\texttt{QIK}_o^{0.9}$	0.85	0.83	0.82	0.79	0.64	0.62	0.61	0.57
$\texttt{QIK}_o^{0.8}$	0.80	0.79	0.77	0.75	0.60	0.60	0.57	0.56

Table 2: $\texttt{QIK}_c$ vs $\texttt{QIK}_o$: three-object combinations (avg. mAP)

	τ=0.6				τ=0.7			
	k=2	k=4	k=8	k=16	k=2	k=4	k=8	k=16
$\texttt{QIK}_c$	**0.93**	**0.92**	**0.93**	**0.94**	**0.83**	**0.80**	**0.80**	**0.78**
$\texttt{QIK}_o^{0.9}$	0.81	0.78	0.78	0.77	0.58	0.59	0.57	0.52
$\texttt{QIK}_o^{0.8}$	0.52	0.50	0.52	0.50	0.73	0.72	0.70	0.67

Table 3: $\texttt{QIK}_c$ vs $\texttt{QIK}_o$: four-object combinations (avg. mAP)

	τ=0.6				τ=0.7			
	k=2	k=4	k=8	k=16	k=2	k=4	k=8	k=16
$\texttt{QIK}_c$	**0.91**	**0.95**	**0.97**	**0.94**	**0.81**	**0.82**	**0.83**	**0.78**
$\texttt{QIK}_o^{0.9}$	0.80	0.82	0.80	0.76	0.52	0.52	0.53	0.49
$\texttt{QIK}_o^{0.8}$	0.68	0.67	0.66	0.65	0.48	0.47	0.49	0.47

4.4.2 $\texttt{QIK}_c$ *vs. Its Competitors.* Next, we compared $\texttt{QIK}_c$ with its competitors, which used CNN-based features for filtering. For fair evaluation, we used the default parameters in the code of DIR, DELF, CroW, and FR-CNN. For LIRE, we used CEDD to extract the features of images and indexed them using Lucene. In addition, the sentence parse trees were used during the ranking step for tree edit distance computation. (The results obtained by using dependency trees were similar and are not shown in the interest of space.) For the two-object combinations, we computed the mAP value for each combination and report the average of the mAP values. Similarly, we report the average of mAP values for the three-object and four-object combinations. Tables 4, 5, and 6 report these values for two different τ values and different values of k. In all cases, QIK outperformed its competitors by virtue of using captions in PIUs and applying NLP processing. $\texttt{QIK}_c$ was able to capture the the relationships between objects in everyday scenes leading to superior performance. Other approaches relied on local/global descriptors of images constructed from features with or without CNNs. In most cases, CroW was the best approach among the chosen competitors; LIRE was the worst approach. This confirms that techniques using CNN-based features are superior to traditional feature-based indexing of images.

We measured the average time taken by each technique for image retrieval. As reported in Table 7, QIK was competitive in terms of average retrieval time. LIRE ran the fastest but yielded the lowest mAP value. Although CroW's mAP value was the second best, it was the slowest.

4.4.3 Scalability of QIK*.* To test the scalability of QIK, we indexed 124K images in MS COCO. We executed 2720 queries; on an average, QIK took 0.5 seconds for each query. This shows that QIK can support efficient retrieval on large image repositories.

Table 4: Results for two-object combinations (avg. of mAP)

	τ=0.6				τ=0.7			
	k=2	k=4	k=8	k=16	k=2	k=4	k=8	k=16
QIK	**0.94**	**0.96**	**0.96**	**0.94**	**0.81**	**0.85**	**0.84**	**0.81**
CroW	0.86	0.85	0.83	0.82	0.71	0.66	0.64	0.61
FR-CNN	0.82	0.84	0.83	0.79	0.63	0.64	0.63	0.59
DIR	0.80	0.79	0.78	0.76	0.56	0.57	0.54	0.51
DELF	0.52	0.52	0.49	0.49	0.35	0.32	0.29	0.27
LIRE	0.40	0.41	0.40	0.37	0.19	0.19	0.17	0.16

Table 5: Results for three-object combinations (avg. of mAP)

	τ=0.6				τ=0.7			
	k=2	k=4	k=8	k=16	k=2	k=4	k=8	k=16
QIK	**0.93**	**0.92**	**0.93**	**0.94**	**0.83**	**0.80**	**0.80**	**0.78**
CroW	0.83	0.82	0.82	0.81	0.65	0.59	0.59	0.57
FR-CNN	0.76	0.78	0.78	0.76	0.50	0.55	0.56	0.53
DIR	0.78	0.78	0.75	0.68	0.49	0.52	0.51	0.45
DELF	0.50	0.51	0.48	0.47	0.31	0.29	0.27	0.25
LIRE	0.37	0.36	0.35	0.38	0.14	0.15	0.14	0.14

Table 6: Results for four-object combinations (avg. of mAP)

	τ=0.6				τ=0.7			
	k=2	k=4	k=8	k=16	k=2	k=4	k=8	k=16
QIK	**0.91**	**0.95**	**0.97**	**0.94**	**0.81**	**0.82**	**0.83**	**0.78**
CroW	0.86	0.90	0.88	0.86	0.71	0.66	0.65	0.62
FR-CNN	0.90	0.89	0.87	0.84	0.59	0.64	0.63	0.57
DIR	0.84	0.80	0.75	0.72	0.53	0.53	0.52	0.50
DELF	0.64	0.64	0.57	0.49	0.45	0.40	0.33	0.26
LIRE	0.42	0.42	0.45	0.41	0.13	0.17	0.18	0.17

Table 7: Average time taken (in seconds) for image retrieval

	Two-object combination	Three-object combination	Four-object combination
QIK	0.49 s	0.54 s	0.54 s
CroW	8.67 s	8.49 s	8.50 s
FR-CNN	0.73 s	0.73 s	0.74 s
DIR	0.45 s	0.44 s	0.44 s
DELF	0.51 s	0.52 s	0.48 s
LIRE	0.33 s	0.34 s	0.34 s

5 CONCLUSIONS

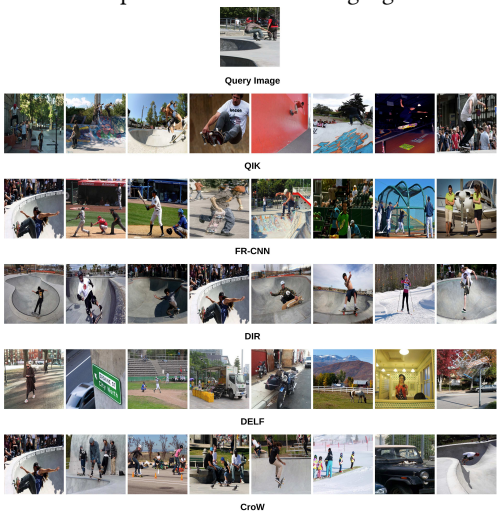
QIK is an efficient system for large-scale image retrieval using PIUs of images. It leverages the predictions of deep neural networks designed for image understanding tasks, thereby capturing relationships between multiple objects in complex scenes. The captions predicted for the images are analyzed linguistically by constructing sentence parse trees and dependency trees. During the filtering step, the parse tree of a query image's caption is transformed to an optimized XPath query and executed to fetch a set of candidate images from the database. Finally, the tree edit distance is used for ranking and returning the top-k results for a query. Through performance evaluation on the MS COCO dataset, we observed that QIK outperformed state-of-the-art techniques for large-scale image retrieval. QIK is available on GitHub at https://github.com/MU-Data-Science/QIK.

Acknowledgments

This work was supported by the National Science Foundation under Grant No. 1747751. Part of this work was done when the first and last authors were at University of Missouri-Kansas City.

6 APPENDIX A

(a) Predicted caption: "a woman feeding a giraffe at a zoo"



(b) Predicted caption: "a man riding a skateboard up the side of a ramp"

Figure 8: Results fetched by QIK and its competitors.

Figure 8 shows two queries and the output of different techniques for $k = 8$. As seen in Figure 8(a), QIK returned only images of people with one or more giraffes. However, other techniques returned at least one false positive for the query as defined in Section 4.3. Similarly, Figure 8(b) shows that QIK returned only images of skateboarders. However, other techniques returned at least one false positive.

REFERENCES

[1] 2019. BaseX: A robust, high-performance XML database engine. Available from http://basex.org/.

[2] 2019. Google Landmarks Dataset v2. https://github.com/cvdfoundation/google-landmark.

[3] 2019. The Stanford Parser: A statistical parser. Available from https://nlp.stanford.edu/software/lex-parser.shtml.

[4] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. 2016. NetVLAD: CNN architecture for weakly supervised place recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[5] Yannis Avrithis and Giorgos Tolias. 2014. Hough Pyramid Matching: Speeded-Up Geometry Re-ranking for Large Scale Image Retrieval. *International Journal of Computer Vision* 107, 1 (01 Mar 2014), 1–19. https://doi.org/10.1007/s11263-013-0659-3

[6] Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky. 2014. Neural Codes for Image Retrieval. In *Computer Vision – ECCV 2014*, David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars (Eds.). Springer International Publishing, Cham, 584–599.

[7] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. 2008. SURF: Speeded-Up Robust Features. *Computer Vision and Image Understanding* 110, 3 (2008), 346 – 359.

[8] Anders Berglund, Scott Boag, Don Chamberlin, Mary F. Fernandez, Michael Kay, Jonathan Robie, and Jerome Simeon. 2002. *XML Path Language (XPath) 2.0 W3C Working Draft 16.* Technical Report WD-xpath20-20020816. World Wide Web Consortium. http://www.w3.org/TR/xpath20/

[9] Philip Bille. 2005. A survey on tree edit distance and related problems. *Theoretical Computer Science* 337, 1 (2005), 217 – 239. https://doi.org/10.1016/j.tcs.2004.12.030

[10] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, and Eve Maler. 2000. *Extensible Markup Language (XML) 1.0 Second Edition W3C Recommendation.* Technical Report REC-xml-20001006. World Wide Web Consortium. http://www.w3.org/TR/2000/REC-xml-20001006

[11] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal Sentence Encoder. *CoRR* abs/1803.11175 (2018). arXiv:1803.11175 http://arxiv.org/abs/1803.11175

[12] Danqi Chen and Christopher Manning. 2014. A Fast and Accurate Dependency Parser using Neural Networks. In *Proc. of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 740–750.

[13] Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and CÃŋdric Bray. 2004. Visual categorization with bags of keypoints. In *In Workshop on Statistical Learning in Computer Vision, ECCV*. 1–22.

[14] Dmitry Duplyakin, Robert Ricci, Aleksander Maricq, Gary Wong, Jonathon Duerig, Eric Eide, Leigh Stoller, Mike Hibler, David Johnson, Kirk Webb, Aditya Akella, Kuangching Wang, Glenn Ricart, Larry Landweber, Chip Elliott, Michael Zink, Emmanuel Cecchet, Snigdhaswin Kar, and Prabodh Mishra. 2019. The Design and Operation of CloudLab. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*. Renton, WA, 1–14.

[15] Albert Gordo, Jon Almazán, Jerome Revaud, and Diane Larlus. 2016. Deep Image Retrieval: Learning Global Representations for Image Search. In *Computer Vision – ECCV 2016*, Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (Eds.). Springer International Publishing, Cham, 241–257.

[16] A. Gordo and D. Larlus. 2017. Beyond Instance-Level Image Retrieval: Leveraging Captions to Learn a Global Visual Representation for Semantic Retrieval. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 5272–5281.

[17] Christian Grün, Sebastian Gath, Alexander Holupirek, and Marc H. Scholl. 2009. XQuery Full Text Implementation in BaseX. In *Proceedings of the 6th International XML Database Symposium on Database and XML Technologies* (Lyon, France) *(XSym '09)*. Springer-Verlag, Berlin, Heidelberg, 114–128. https://doi.org/10.1007/978-3-642-03555-5_10

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. *CoRR* abs/1512.03385 (2015).

[19] Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2008. Hamming Embedding and Weak Geometric Consistency for Large Scale Image Search. In *Proceedings of the 10th European Conference on Computer Vision: Part I* (Marseille, France) *(ECCV '08)*. Springer-Verlag, Berlin, Heidelberg, 304–317. https://doi.org/10.1007/978-3-540-88682-2_24

[20] H. Jegou, M. Douze, C. Schmid, and P. Perez. 2010. Aggregating local descriptors into a compact image representation. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 3304–3311.

[21] H. Jegou, F. Perronnin, M. Douze, J. Sanchez, P. Perez, and C. Schmid. 2012. Aggregating Local Image Descriptors into Compact Codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34, 9 (Sep. 2012), 1704–1716.

[22] Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition* (2nd ed.). Prentice Hall, USA.

[23] Yannis Kalantidis, Clayton Mellina, and Simon Osindero. 2016. Cross-Dimensional Weighting for Aggregated Deep Convolutional Features. In *Computer Vision – ECCV 2016 Workshops*, Gang Hua and Hervé Jégou (Eds.). 685–701.

[24] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollar, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *Computer Vision – ECCV 2014*. Springer International Publishing, Cham, 740–755.

[25] David G. Lowe. 2004. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision* 60 (2004), 91–110.

[26] Mathias Lux and Savvas A. Chatzichristofis. 2008. LIRE: Lucene Image Retrieval: An Extensible Java CBIR Library. In *Proceedings of the 16th ACM International Conference on Multimedia* (Vancouver, British Columbia, Canada) *(MM '08)*. 1085–1088.

[27] Anastasiya Mishchuk, Dmytro Mishkin, Filip Radenovic, and Jiři Matas. 2017. Working Hard to Know Your Neighbor's Margins: Local Descriptor Learning Loss. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, CA) *(NIPS'17)*. 4829–4840.

[28] Dmytro Mishkin, Filip Radenović, and Jiři Matas. 2018. Repeatability Is Not Enough: Learning Affine Regions via Discriminability. In *Computer Vision – ECCV 2018*. Springer International Publishing, Cham, 287–304.

[29] Hyeonwoo Noh, Andre Araujo, Jack Sim, Tobias Weyand, and Bohyung Han. 2017. Large-Scale Image Retrieval with Attentive Deep Local Features. In *Proc. of 2017 IEEE International Conference on Computer Vision (ICCV)*. 1–10.

[30] Mateusz Pawlik and Nikolaus Augsten. 2015. Efficient Computation of the Tree Edit Distance. *ACM Trans. Database Syst.* 40, 1, Article 3 (March 2015), 40 pages.

[31] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. 2007. Object retrieval with large vocabularies and fast spatial matching.. In *Proc. of CVPR 2007*.

[32] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. 2008. Lost in quantization: Improving particular object retrieval in large scale image databases. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*. 1–8. https://doi.org/10.1109/CVPR.2008.4587635

[33] F. Radenovic, G. Tolias, and O. Chum. 2019. Fine-Tuning CNN Image Retrieval with No Human Annotation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41, 7 (July 2019), 1655–1668.

[34] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards Real-time Object Detection with Region Proposal Networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1* (Montreal, Canada) *(NIPS'15)*. 91–99.

[35] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2017. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* 39, 6 (June 2017), 1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031

[36] Amaia Salvador, Xavier Giro-i Nieto, Ferran Marques, and Shin'ichi Satoh. 2016. Faster R-CNN Features for Instance Search. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

[37] Sivic and Zisserman. 2003. Video Google: a text retrieval approach to object matching in videos. In *Proceedings Ninth IEEE International Conference on Computer Vision*. 1470–1477 vol.2.

[38] Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013. Parsing with Compositional Vector Grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, 455–465.

[39] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going Deeper with Convolutions. In *Computer Vision and Pattern Recognition*.

[40] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2015. Rethinking the Inception Architecture for Computer Vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), 2818–2826.

[41] Marvin Teichmann, Andre Araujo, Menglong Zhu, and Jack Sim. 2019. Detect-To-Retrieve: Efficient Regional Aggregation for Image Search. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[42] Giorgos Tolias, Yannis Avrithis, and Herve Jegou. 2016. Image Search with Selective Match Kernels: Aggregation Across Single and Multiple Images. *Int. J. Comput. Vision* 116, 3 (Feb. 2016), 247–261.

[43] Giorgos Tolias, Ronan Sicre, and Herve Jegou. 2016. Particular object retrieval with integral max-pooling of CNN activations. In *International Conference on Learning Representations*. 1–12.

[44] Quoc-Tuan Truong and Hady W. Lauw. 2019. VistaNet: Visual Aspect Attention Network for Multimodal Sentiment Analysis. In *The Thirty-third AAAI Conference*. AAAI Press, Honolulu, Hawaii, 305–312.

[45] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2017. Show and Tell: Lessons Learned from the 2015 MS COCO Image Captioning Challenge. *IEEE Trans. Pattern Anal. Mach. Intell.* 39, 4 (April 2017), 652–663.

[46] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *Proceedings of the*

32nd International Conference on Machine Learning, Vol. 37. Lille, France, 2048–2057.

[47] A. B. Yandex and V. Lempitsky. 2015. Aggregating Local Deep Features for Image Retrieval. In *2015 IEEE International Conference on Computer Vision (ICCV)*. 1269–1277. https://doi.org/10.1109/ICCV.2015.150

[48] Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics* 2 (2014), 67–78. https://doi.org/10.1162/tacl_a_00166

[49] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. 2018. Learning Transferable Architectures for Scalable Image Recognition. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8697–8710.