PHOcus: Efficiently Archiving Photos

Susan B. Davidson
University of Pennsylvania
susan@cis.upenn.edu

Shay Gershtein
Tel Aviv University
shayg1@mail.tau.ac.il

Tova Milo
Tel Aviv University
milo@post.tau.ac.il

Slava Novgorodov
Tel Aviv University
slavanov@post.tau.ac.il

May Shoshan
Tel Aviv University
mayshoshan@mail.tau.ac.il

ABSTRACT

Our ability to collect data is rapidly outstripping our ability to effectively store and use it. Organizations are therefore facing tough decisions of what data to archive (or dispose of) to effectively meet their business goals. PHOcus addresses this problem in the context of image data (photos) by proposing which photos to archive to meet an online storage budget. The decision is based on factors such as usage patterns and their relative importance, the quality and size of a photo, the relevance of a photo for a usage pattern, the similarity between different photos, as well as policy requirements of what photos must be retained. We formalize the photo archival problem and give an efficient algorithm with an optimal approximation guarantee. We then demonstrate our system, PHOcus, on an e-commerce application as well as with personal photos on a smartphone, and discuss how many of the inputs to the problem can be automatically obtained.

PVLDB Reference Format:
Susan B. Davidson, Shay Gershtein, Tova Milo, Slava Novgorodov, and May Shoshan. PHOcus: Efficiently Archiving Photos. PVLDB, 15(12): 3630 - 3633, 2022.
doi:10.14778/3554821.3554861

1 INTRODUCTION

Although the Big Data revolution has enabled incredible advances in areas such as medicine, commerce, transportation, and science, we are facing an inflection point [9]: The ability to collect data outstrips our ability to effectively use it and will eventually outstrip our ability to store it [2]. Organizations must therefore determine which data to move to larger, cheaper, and typically slower storage (or dispose of) to meet an online storage budget, based on factors such as *usage* patterns (workflows) and relative importance of the workflows, the *quality* and *size* of data as well as the *similarity* between different data, requirements on the quality of input data to the workflows, as well as *policy* requirements, e.g. GDPR [3] regulations or document retention regulations for banks. Due to the volume of data, these disposal decisions must be automated.

Image data is an important special case of this problem due to the large file sizes as well as its abundance: it is estimated that up to three billion images are shared on the internet daily, with more than 340 million photos being uploaded on Facebook, over 90 million photos being posted on Instagram and more than 500 million GIFs posted on Twitter.

As a concrete example of the need to reduce the amount of online image data, consider an e-commerce application such as XYZ[1]. XYZ has a huge archive of images of products that are displayed throughout a hierarchy of landing pages of product categories. For each page, there is a pre-defined subset of images that are relevant for the product category, out of which a small set are displayed. Each image may be relevant for a large number of different pages, and its value may differ between pages. For example, an image of an iPhone which shows the model number may be very valuable for a page which compares different models of iPhones, but not as valuable for a page which displays smartphones in general. Some of the images may also be required to appear on certain pages due to legal contracts (*policies*). For example, a company may require only approved images to be used on pages that are specific to their products. Finally, the landing pages themselves may vary in importance, reflecting the relative popularity of product categories. To speed up the page display, images that are used on pages are stored in a fast-access cache, which is much smaller than the size of the archive. The problem is to find a set of images that can fit in the cache, meet the content and policy requirements of each of the landing pages, and maximize the value over all pages.

The e-commerce example is just one among many instances of what we call the *Photo Archive Reduction problem* (PAR). At a personal level, you may encounter it as the need to delete photos locally on your smartphone to meet some storage budget, relying on cloud storage for your full set of photos. You may have explicitly organized subsets of the photos in albums, or implicitly organized them by labeling photos with the same tag. Image tagging software may also automatically organize photos by features such as date, location and facial recognition. You may require that some of your photos remain in local storage for fast access, for example, photos of your passport, vaccination record and recent favorite photos of your family. Again, the problem is to automate the (local) deletion (and uploading to the cloud) of photos to maximize the value across these pre-defined subsets of photos and satisfy retention requirements to meet storage constraints.

The PHOcus system, presented in this demo, address this important problem. Our first contribution in this demo is a formalization of PAR, its inputs, and the optimization goal used in PHOcus. Our second contribution is the algorithm underlying our implementation. Since an exact solution for PAR is not computationally feasible,

Proceedings of the VLDB Endowment, Vol. 15, No. 12 ISSN 2150-8097.
doi:10.14778/3554821.3554861

[1]Company name omitted due to privacy considerations.

we use in PHOcus an efficient algorithm with an optimal approximation guarantee. Our third contribution is a demonstration of how to obtain the inputs to PAR. Some of these inputs are straightforward – the input set of photos, the size of each photo, retention requirements, and the overall storage constraint. Other inputs seem difficult to obtain, but in fact there are common practical solutions for obtaining them automatically. We show examples of how to do this in the e-commerce setting in the demo – the pre-defined subsets of photos, the relative importance of pre-defined subsets of photos, or the value of a photo for a pre-defined subset. Our final contribution is a demonstration of the operation of PHOcus in the two scenarios discussed above. In particular, PHOcus proposes a solution which can be easily refined manually if desired.

It should be noted that we have also evaluated the e-commerce application within XYZ on several product categories (separately, as each category is assigned different analysts and has its own space constraint). Initial reports indicate that PHOcus significantly reduces manual work; concretely, analysts reported that it took them less then 10 minutes on medium size datasets compared to hours of manual work invested without PHOcus.

Demonstration Overview. Our demonstration will showcase the two photo archiving use-cases described above. The first use-case demonstrates how PHOcus is being used in an e-commerce company for deriving a space efficient subset of representative fast-cached photos for landing pages. The second use-case demonstrates the management of personal photo albums, which may be of more general interest to the audience. For each use-case, the demonstration consists of two parts: (1) examining the raw data and defining a specific instance of PAR by providing the necessary input; and (2) interactively adjusting the requirements to fine-tune the solution.

2 RELATED WORK

The problem of selecting a subset of photos has been considered in various settings, e.g. image collection for e-commerce, story-telling from personal albums, online image recommendation systems, and while training machine learning algorithms [11–14, 16].

However, this work differs in several important ways from our setting. First, while most work has a similar goal of *summarization* they also directly optimize the diversity of the selected photos, which leads to non-monotone objective functions [14, 16]. In contrast, diversity is a by-product of the similarity function used in our setting, guaranteeing monotonicity of the objective function and thus allowing for approximation guarantees (see Section 3). Second, most work aims to select a specified *number* of photos [12–14, 16], whereas our setting is more general: the cardinality constraint is extended to a knapsack constraint (a bound on the sum of non-uniform sizes of the photos). To allow the user control of which aspects/categories of photos should be preserved and to what extent, our input includes an explicit specification of the relevant photo subsets (along with importance weights), instead of attempting to derive such aspects implicitly based on embedding similarity or clustering [12, 14]. *Caching* algorithms (e.g., [7, 17]) are also related, but in contrast to our work these works either only consider varying size individual items without predefined subsets (as, e.g. in files caching) or do consider item subsets but of fix size items (as e.g., in semantic query caching).

To model the distances/similarities of photos to measure how well a non-selected photo is captured by the retained photos, we adapt existing techniques [14] to our setting, as discussed in Section 4. Finally, we note that there are similar problems in other domains, where the set of items does not consist of photos. For example, [4] studied the optimal selection of products to offer for sale out of the entire catalog. While the input derivation is naturally unrelated, the theoretical approach is a simplified variant of our solution.

3 MODEL AND APPROACH

We now describe the formal model of PAR, discuss the algorithmic approach, and contrast our solution with related work.

Input. The input set of photos is denoted by P, and the set of photos that must be retained by S_0. The number of input photos is denoted by $n = |P|$. The cost of each photo is its size (in terms of the disk space required to store it) and is given by the cost function $C : P \mapsto \mathbb{R}_+$. For a subset of photos $S \subseteq P$, we denote its cost, which is defined as the sum of the individual costs, by $C(S) = \sum_{p \in S} C(p)$. The set of pre-defined subsets of photos is denoted by $Q \subseteq 2^P$.

The function $\mathcal{W} : Q \mapsto \mathbb{R}_+$ assigns a positive weight to each pre-defined subset, to reflect its importance (i.e. how valuable it is to retain the photos in this subset). Given any specific $q \in Q$, the relevance function $\mathcal{R} : Q \times P \mapsto \mathbb{R}_+$ assigns a score to each photo p in q that reflects how relevant it is for this pre-defined subset (the score for photos in $P \setminus q$ is 0). The relevance scores of all photos in a pre-defined subset are normalized (in advance) to sum up to 1: $\forall q \in Q : \sum_{p \in q} \mathcal{R}(q, p) = 1$. A (contextualized) similarity function $\mathcal{SIM} : Q \times P \times P \mapsto [0, 1]$ produces a normalized measure of the similarity of any given pair of photos, w.r.t. a given pre-defined subset. Note that the similarity of any pair of photos differs based on the pre-defined subset, which is referred to as the *context* of the similarity. In particular, if at least one of the two photos is not in the context subset, then the similarity score is defined to be 0. Moreover, the similarity of any photo to itself is 1. However, a similarity score of 1 does not necessarily imply that the pair of photos are identical.

Lastly, the budget B is an upper bound on the cost of the solution, e.g., the capacity of the cache. Hence, the complete input for a PAR instance consists of the tuple $\langle P, S_0, Q, C, \mathcal{W}, \mathcal{R}, \mathcal{SIM}, B \rangle$.

Example 3.1. Consider the e-commerce application introduced earlier, which will form the basis of the demo: P is the image archive. S_0 is the set of images that are required to appear on certain pages due to legal contracts. Q represents the set of landing pages, each of which is defined by a set of relevant images for the page and its title (e.g., “Nike red shirts”, “LG smartphones” or “shoes”). C gives the size of each image. $\mathcal{W}$ represents the relative importance of concrete landing page and is calculated based on the landing page popularity, i.e. the number of visits in the last 90 days, normalized by sum of all visits to all landing pages. $\mathcal{R}$ represents the relevance of each image to the landing page, and is computed based both on the quality of the image (using an internal model [1]) and the relevance score of the product represented in the image (using the product title and search engine retrieval score). Finally, $\mathcal{SIM}$ (similarity) is calculated using cosine similarity between image embeddings. The embeddings are based on the ResNet-50 network [5] and is trained on over 50M product images [1].

Objective. Given an input PAR instance, the goal is to find the "best" subset of photos to retain, i.e a solution $S \subseteq P$ such that $C(S) \leq B, S_0 \subseteq S$, and the score of S is maximized. To describe how a solution is scored, we first need the following definition: given a pre-defined subset $q \in Q$, a photo $p \in q$, and a solution $S \subseteq P$, the *nearest neighbour* of p in S w.r.t. q is the most similar photo in $S \cap q$ to p, denoted by $NN(q,p,S) = \arg\max_{p' \in S \cap q} \mathcal{SIM}(q,p,p')$ (if there are several such photos, we choose one arbitrarily). Observe that if $p \in S$, then $NN(q,p,S) = p$. Note that defining the contribution to the objective over a specific non-selected photo via the similarity to the nearest neighbor implies that solving the problem over each single subset is equivalent to the facility location problem, which is a typical modeling choice. We define the *score* of a solution S w.r.t. a given pre-defined subset $q \in Q$ as

$$\mathcal{G}(q,S) = \sum_{p \in q} \mathcal{R}(q,p) \cdot \mathcal{SIM}(q,p,NN(q,p,S)).$$

Abusing notation, the overall score of a solution S is defined as the weighted sum of the scores w.r.t. pre-defined subsets :

$$\mathcal{G}(S) = \sum_{q \in Q} W(q) \cdot \mathcal{G}(q,S).$$

Thus, the goal is to produce $\arg\max_{S_0 \subseteq S \subseteq P, C(S) \leq B} \mathcal{G}(S)$.

Theoretical Approach. It can be shown that PAR cannot be approximated beyond a $(1 - 1/e)$ factor, unless $P = NP$, via a straightforward reduction from the Maximum Coverage problem. We, nevertheless, solve the problem via an efficient algorithm with a tight worst-case approximation guarantee.

Concretely, since the objective function can be proved to be non-negative, monotone and submodular, we use the approach of [15] that applies to the maximization of such set functions subject to a knapsack constraint, which in our setting is the budget constraint. The proof of these properties can be found in a full version of this demo paper (in preparation).

The algorithm of [15] is an extension of the standard iterative greedy algorithm that also uses enumeration. In the following description, the construction of each examined solution begins with selecting all photos in S_0. For brevity, we describe the algorithm for $S_0 = \emptyset$, however, note that, in general, the final solution is the union of the described solution with S_0 (the approximation guarantees may only improve when $S_0 \neq \emptyset$). The algorithm first examines all solutions of cardinality at most 2 (i.e. 1 or 2 photos), denoting the best such solution by S_1. It also examines all valid selections of size 3 (there are at most $O(n^3)$ such selections), unrelated to S_1, and greedily extends each such selection until reaching the budget bound, denoting the best such solution by S_2. The final output is the best solution out of S_1 and S_2. This algorithm is proven in [15] to achieve a $(1 - 1/e)$ approximation ratio, which is optimal, given the aforementioned hardness bound.

Optimizations. To facilitate efficiency, we leverage both the fact that the above algorithm is embarrassingly parallelizable and the submodularity of the objective which lends itself to an improved implementation of the greedy procedure employed by the algorithm. Concretely, observe that all $O(n^2)$ solutions examined for deriving S_1 and all $O(n^3)$ greedy procedures employed to derive S_2 are completely independent, and thus can be evaluated in parallel. Furthermore, for the greedy procedure we use the accelerated implementation of [10], which employs lazy evaluations that take advantage of the submodularity of the objective to examine fewer possible solutions.

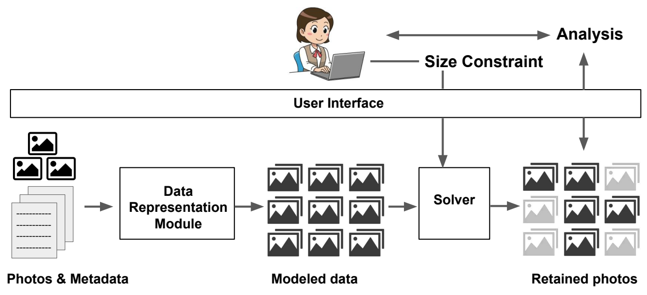

Figure 1: System Architecture

4 SYSTEM ARCHITECTURE

We implemented PHOcus using Python and Flask, and its high-level architecture is shown in Figure 1. The system consists of a user interface and two modules: the *Data Representation Module* and the *Solver*. The user interacts with the system via the UI to specify the input, as described below. Note that the similarity function $\mathcal{SIM}$ is computed automatically at a later stage.

The pre-defined subsets may be specified in one of three ways:

(1) Directly: each photo is tagged with all the subsets that include it. The relevance scores, which are assumed to be uniform by default, may be adjusted via the UI;
(2) Queries: users provide queries such as ("Paris vacation"), and the subsets are computed via the PHOcus search engine. The confidence scores of the engine are then converted into the relevance scores; or
(3) Automatic tagging: Subsets are derived by automatic tagging methods. The weights for subsets derived by all methods may be adjusted using dedicated UI.

The Data Representation module receives this input and, if it contains user-provided queries, uses an internal search engine to compute the result sets and the relevance scores. It then normalizes the relevance scores (as described in Section 3), and derives the contextualized similarities (SIM). This is done using the approach in [14], which computes the distance between two photos based on both quantitative and categorical attributes that are derived via standard methods, including, e.g., reading the EXIF metadata and generating visual words via the SIFT algorithm [8]. An important adaptation to our setting, however, is normalizing the distances differently for each context subset. This is done by dividing all distances by the maximum distance between any two photos in the context. Intuitively, this emphasizes smaller variations across photos for more granular queries. For example, when searching for photos of all "trips", having many photos of the same trip to Paris in 2016 may seem redundant. However, when searching for this specific trip, photos are only redundant if they are very similar in more specific features (e.g., many photos of Eiffel Tower taken in the same day).

The complete input is then passed to the Solver module, which runs the optimization algorithm described in Section 3. The proposed solution is finally displayed to the user, along with various metrics, as discussed next in Section 5.

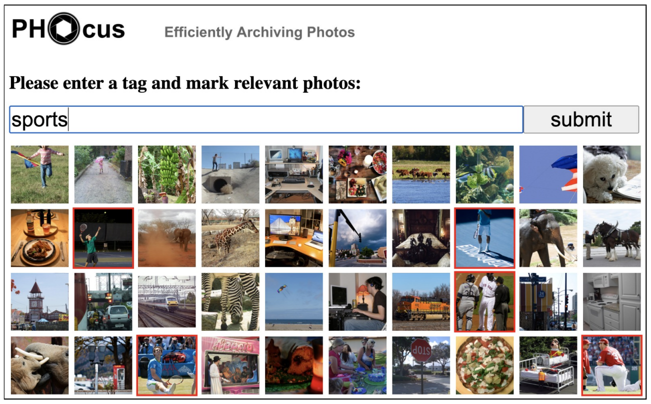
PHOcus
Efficiently Archiving Photos
Please enter a tag and mark relevant photos:
sports
submit

Figure 2: Generating the input

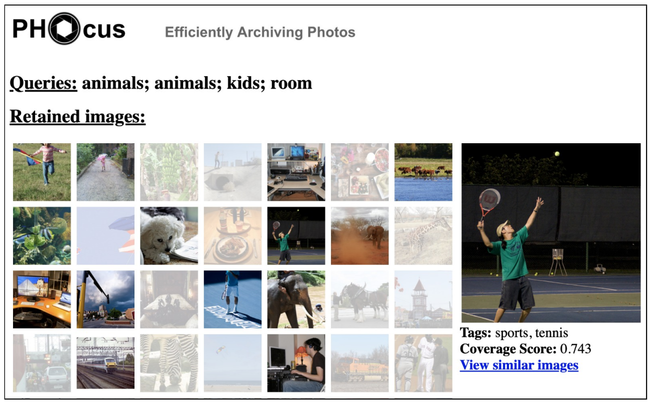

Figure 3: Analysis

5 DEMONSTRATION SCENARIO

To highlight the general applicability of our model and approach, our demonstration will showcase the two photo archiving use-cases described in the introduction. The first use-case demonstrates how PHOcus would be used in an e-commerce company for deriving a space efficient subset of representative fast-cached photos for landing pages, whereas the second use-case demonstrates the management of personal photo albums, which may be of more general interest. PHOcus schematically operates the same for both cases.

The demonstration consists of two parts: (1) examining the raw data and defining a specific instance of PAR by providing the necessary input; and (2) interactively adjusting the requirements to fine-tune the solution.

Defining a problem instance. The audience will play the role of a user who wishes to reduce a photo archive to fit into a specified space constraint; in the e-commerce scenario the user is typically a business analyst. The audience will first choose one of the two use-cases, and the dataset will be presented via the UI. The e-commerce dataset is a real-life XYZ dataset used for this problem, whereas the personal photo dataset is taken from [6].

The user may examine the photos and click on each one to view its size and any existing metadata, and then specify the input using the UI. Most importantly, users may specify the input subsets by adding tags to photos, submitting queries to the search bar, or enabling automatic tag generation (as discussed in Section 4). An example of specifying subsets that fits specific tag is given in Figure 2, where the user first has clicked on several photos (indicated by red boxes), and then entered a tag next to the submit button. For the e-commerce use-case there is already an initial set of photos used by XYZ, which the user may also use. The user will indicate the relative importance weights of the subsets using sliders; the relevance scores may also be viewed and adjusting using a similar UI. Finally, a user may mark specific photos to add to S_0, the set of photos that must be retained, and specify the total space capacity of the final reduced collection.

Interactive adjustments. To continue our demonstration, PHOcus will compute the solution and present to the user the suggested set of photos to retain (the user is also shown the complement set). The user can examine separately the retained photos for each pre-defined subset and sort the photos by various metrics. These metrics include the contribution to the score over that subset or the contribution normalized by the size, and are displayed by clicking on a photo. Moreover, the user can view the granular scores over the subsets, and for each subset the user can inspect the photos that were *not* selected. By clicking on such a photo, PHOcus shows the selected photo that is most similar to it. The user may then modify the solution by adjusting the input (the subsets, their weights, and the budget), requiring that some specific photos are retained/disposed of, and even manually adjusting the relevance scores for the result sets computed for the queries, which can be inspected in a separate view. PHOcus will then recompute the solution and present it to the user along with a compact visualization of the changes, thereby allowing a "what-if"-style analysis, as illustrated in Figure 3.

REFERENCES

[1] Arnon Dagan, Ido Guy, and Slava Novgorodov. An image is worth a thousand terms? analysis of visual e-commerce search. In *SIGIR*, 2021.
[2] Dataage 2025 - the digitization of the world, seagate us. http://www.seagate.com/our-story/data-age-2025.
[3] General Data Protection Regulation (GDPR). https://en.wikipedia.org/wiki/General_Data _Protection_Regulation.
[4] Shay Gershtein, Tova Milo, and Slava Novgorodov. Inventory reduction via maximal coverage in e-commerce. In *EDBT*, pages 522–533, 2020.
[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of CVPR*, pages 770–778, 2016.
[6] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, et al. The open images dataset v4. *IJCV*, 128(7):1956–1981, 2020.
[7] Liying Li, Guodong Zhao, and Rick S Blum. A survey of caching techniques in cellular networks: Research issues and challenges in content placement and delivery strategies. *IEEE Communications Surveys & Tutorials*, 20(3).
[8] David G Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.
[9] Tova Milo. Getting rid of data. *ACM J. Data Inf. Qual.*, 12(1):1:1–1:7, 2020.
[10] Michel Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization techniques*, pages 234–243. Springer, 1978.
[11] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, and Amin Karbasi. Fast constrained submodular maximization: Personalized data summarization. In *International Conference on Machine Learning*, pages 1358–1367. PMLR, 2016.
[12] Ian Simon, Noah Snavely, and Steven M Seitz. Scene summarization for online image collections. In *Proc. of ICCV*, pages 1–8. IEEE, 2007.
[13] Anurag Singh, Lakshay Virmani, and AV Subramanyam. Image corpus representative summarization. In *Proc. of BigMM*, pages 21–29, 2019.
[14] Pinaki Sinha, Sharad Mehrotra, and Ramesh Jain. Summarization of personal photologs using multidimensional content and context. In *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, pages 1–8, 2011.
[15] Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43, 2004.
[16] Sebastian Tschiatschek, Rishabh K Iyer, Haochen Wei, and Jeff A Bilmes. Learning mixtures of submodular functions for image collection summarization. In *Advances in neural information processing systems*, pages 1413–1421, 2014.
[17] Victor Zakhary, Divyakant Agrawal, and Amr El Abbadi. Caching at the web scale: [tutorial]. In *Proc. of WWW*, pages 909–912, 2017.