Demonstration of MaskSearch: Efficiently Querying Image Masks for Machine Learning Workflows

Lindsey Linxi Wei*
University of Washington
linxiwei@cs.washington.edu

Chung Yik Edward Yeung*
University of Washington
chungy04@cs.washington.edu

Hongjian Yu*
University of Washington
hjyu@cs.washington.edu

Jingchuan Zhou*
University of Washington
jzhou27@cs.washington.edu

Dong He
University of Washington
donghe@cs.washington.edu

Magdalena Balazinska
University of Washington
magda@cs.washington.edu

ABSTRACT

We demonstrate MaskSearch, a system designed to accelerate queries over databases of image masks generated by machine learning models. MaskSearch formalizes and accelerates a new category of queries for retrieving images and their corresponding masks based on mask properties, which support various applications, from identifying spurious correlations learned by models to exploring discrepancies between model saliency and human attention. This demonstration makes the following contributions: (1) the introduction of MaskSearch's graphical user interface (GUI), which enables interactive exploration of image databases through mask properties, (2) hands-on opportunities for users to explore MaskSearch's capabilities and constraints within machine learning workflows, and (3) an opportunity for conference attendees to understand how MaskSearch accelerates queries over image masks.

PVLDB Reference Format:
Lindsey Linxi Wei, Chung Yik Edward Yeung, Hongjian Yu, Jingchuan Zhou, Dong He, Magdalena Balazinska. Demonstration of MaskSearch: Efficiently Querying Image Masks for Machine Learning Workflows. PVLDB, 17(12): 4297 - 4300, 2024.
doi:10.14778/3685800.3685859

1 INTRODUCTION

Masks highlight or isolate certain parts of an image based on desired properties for further processing or analysis. Machine learning tasks over image databases often involve generating masks, such as image segmentation masks [13] and model saliency maps [16]. These masks are crucial for various applications, from model explanation [6, 16] to traffic analysis [1]. For example, practitioners developing image classification models can generate saliency maps to examine which pixels contribute the most to the predictions.

Consider a scenario further discussed in §4, Alice, a data engineer, uses the iWildCam dataset [4] for developing a wild animal image classification model. Facing validation accuracy issues, she computes saliency maps [16] and object bounding boxes (e.g., generated by YOLO [13]) for the misclassified images, an example of which is shown in Figure 1. The red pixels in the figure on the right of Figure 1 indicate higher importance for the model's prediction, and the blue pixels indicate lower importance. She finds that the model focuses on the background pixels, notably outside the ground-truth object bounding boxes, rather than the animals, leading to misclassifications when background conditions change. To correct the model's focus, Alice wishes to augment the dataset and retrain the model to ensure that it relies on relevant features to make predictions. She first retrieves a group of images where the model focuses on the areas outside the object bounding boxes. She then augments the dataset by randomizing the pixels outside object bounding boxes in these images while leaving the original labels unchanged and retrains the model with the augmented dataset. Such an approach is known to help improve model performance [18].

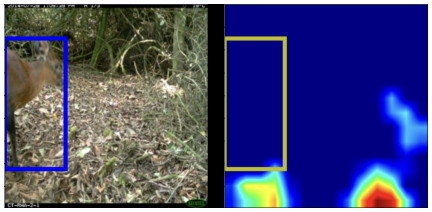

Figure 1: An example misclassified image [4] and its saliency map with the object bounding boxes (blue and yellow boxes). Salient (red) pixels are in the background, which shows that the model relies on irrelevant pixels to classify the image.

As the scenario shows, the ability to retrieve images and masks based on the properties of the latter is valuable to machine learning workflows. However, the efficient execution of these queries suffers from insufficient systems support [8].

We recently developed MaskSearch [10], a system that addresses this challenge by accelerating queries over databases of image masks. MaskSearch's contributions include formalizing a class of image and mask retrieval queries with support for aggregations and top-k computations, introducing a novel indexing technique over masks and an efficient execution framework, and implementing a prototype that significantly outperforms existing solutions in query execution efficiency for both individual and multi-query workloads that simulate machine learning workflows.

In this demonstration, we introduce a graphical user interface (GUI) for MaskSearch (§3), which enables users to execute queries without writing SQL and conveniently displays images, masks, and

*Equal contribution.

Proceedings of the VLDB Endowment, Vol. 17, No. 12 ISSN 2150-8097.
doi:10.14778/3685800.3685859

bounding boxes. We also illustrate MaskSearch's utility across multiple scenarios (§4) in addition to the aforementioned scenario:

- *Scenario 2* demonstrates how MaskSearch can assist in identifying adversarial attacks. We show the ability of MaskSearch to retrieve maliciously attacked images in a dataset by calculating the dispersion of model saliency, relieving the effort required for finding attacks unrecognizable to human eyes.
- *Scenario 3* demonstrates how MaskSearch helps in investigating discrepancies between model saliency and human attention.

Overall, this demonstration will enable conference attendees to experiment with MaskSearch hands-on and appreciate the flexibility and performance of the system.

2 SYSTEM OVERVIEW

In this section, we summarize the MaskSearch system [10].

Data Model. An image mask is a 2D array of pixel values represented by floating-point numbers within the $[0, 1)$ range. MaskSearch supports queries over a database of masks by exposing those masks through a conceptually relational view with one attribute holding the mask data and the other attributes capturing the mask metadata.

```
MasksDatabaseView (
  mask_id INTEGER PRIMARY KEY,
  image_id INTEGER,  // Image from which the mask was derived
  model_id INTEGER,  // Model that generated the mask
  mask_type INTEGER, // Type of mask (e.g., saliency map)
  mask REAL[][]);
```

Region of Interest (ROI). An ROI is defined by a bounding box that specifies the area of interest within a mask. It is not included in `MasksDatabaseView` since it is query-dependant and may be computed on the fly (e.g., object detector applied to the image).

CP Function. CP stands for "Count Pixels". $\texttt{CP}(mask, roi, (lv, uv))$ counts the number of pixels within the ROI in the mask whose values fall within the specified value range $[lv, uv)$. Users can use multiple `CP` functions and apply arithmetic operations in queries.

MaskSearch supports various query types, including filter queries, top-k queries, and aggregation queries, as detailed below.

Filter Query. This query type retrieves masks based on filter conditions on $\texttt{CP}(mask, roi, (lv, uv))$. The filter condition is defined by a threshold T and an inequality symbol.

```
SELECT mask_id FROM MasksDatabaseView
WHERE CP(mask, roi, (lv, uv)) < T;
```

Top-K Query. This query type retrieves the top-k masks ranked by $\texttt{CP}(mask, roi, (lv, uv))$. The ranking order can be ascending (`ASC`) or descending (`DESC`).

```
SELECT mask_id FROM MasksDatabaseView
ORDER BY CP(mask, roi, (lv, uv)) DESC LIMIT K;
```

Aggregation Query. MaskSearch supports both scalar aggregation and mask aggregation. For scalar aggregation, the user can aggregate the outputs of `CP` functions through the `SCALAR_AGG` function. MaskSearch supports aggregation functions like `SUM`, `AVG`, `MIN`, and `MAX`. Mask aggregation facilitates the combination or comparison of information across multiple masks (of the same image), treating aggregated masks as new queryable entities. The user needs to define a function `MASK_AGG` that takes in a list of masks and returns an aggregated mask: `MASK_AGG`$\rightarrow$ `REAL[][]`, where `MASK_AGG` can be any function $f(m_1, m_2, ..., m_n)$, where m_i represents a mask. For example, $\texttt{intersect}(m_1 > 0.8, ..., m_n > 0.8)$ outputs the intersection of the masks $m_1, ..., m_n$ thresholded by 0.8 (pixels > 0.8 becomes 1; otherwise 0).

```
SELECT image_id FROM MasksDatabaseView
WHERE mask_type IN (1, 2, ..., n)
GROUP BY image_id ORDER BY CP(MASK_AGG(mask), roi, (lv, uv));
```

To efficiently support these queries, MaskSearch introduces two key components: the Cumulative Histogram Index (CHI) and a filter-verification query execution framework. CHI is a novel indexing technique that stores pixel counts for different key combinations of spatial locations and pixel values, which enables the efficient derivation of upper and lower bounds for pixel counts of arbitrary ROIs and pixel value ranges specified by the user at query time. The filter-verification framework leverages CHI to compute bounds to determine which masks can be added directly to the result set or pruned without loading them from disk to memory and which require further verification by loading them from disk and applying the predicate. This approach significantly reduces disk I/O which is the bottleneck for query execution. The details of MaskSearch and the performance comparison with existing solutions can be found in [10]. In this demonstration, attendees will be able to explore how MaskSearch executes queries and experience its ease-of-use and query performance improvements.

Limitations. MaskSearch needs to build CHI for the aggregated masks for mask aggregation queries. With the incremental indexing technique [10], the start-up overhead gets amortized quickly. The demo currently supports only a single ROI per image/mask; however, this is not a limitation of MaskSearch.

3 MASKSEARCH INTERFACE

This section describes MaskSearch 's interface (Figure 2).

MaskSearch allows users to load and specify their own models, datasets, and masks. In this demonstration, the GUI loads the datasets, models, and masks for the corresponding scenarios. This process is followed by the automatic calculation and display of the model's accuracy and a confusion matrix where each clickable cell represents the images whose ground truth label and predicted label are the corresponding row and column of the cell, respectively. For example, cell (146, 17) represents images of class 146 that were classified as class 17. The GUI shows the top-100 cells in terms of the number of misclassifications. As illustrated in Step 1 in Figure 2, this functionality allows for detailed visualization of the images from the selected cell (146, 17) with their corresponding masks. Due to space constraints, the initial data loading, confusion matrix, and the illustrative figure for CHI are not presented in Figure 2.

Input Section. The Input Section is demonstrated on the left of Steps 2 and 3 in Figure 2. It simplifies the creation and manipulation of search queries by providing a form that guides users through specifying their query, including defining an optional ROI (full mask by default), upper and lower bounds of the pixel value range, and choosing between different queries such as Top-K Query, Filter Query, and Aggregation Query. The ROIs are provided by the user, such as object bounding boxes generated by an off-the-shelf model. Based on the aforementioned user-specified parameters, the GUI generates an SQL query shown in the "Query Command" window, which allows the users to inspect the formalized query and, if necessary, directly modify the SQL query for their search. Clicking "Execution Detail" (needs to happen after clicking "Start Query")

Figure 2: An example workflow of using MaskSearch's GUI in Scenario 1. In Step 1, 146 -> 17 means that the image with a ground truth label 146: Meleagris Ocellata was misclassified as class 17: Panthera Onca. The images are from WILDS [4].

triggers the GUI to show the number of masks loaded from disk during query execution vs. the number of total masks.

Query Result Section. The Query Result Section, presented on the right of Steps 2 and 3 in Figure 2, displays the query results as a combination of images and their corresponding masks, dependent on the specific scenario. For example, in Step 2 of Figure 2, the returned images are overlaid with their saliency maps and the object bounding boxes. The GUI also offers users the ability to click and zoom in on the query results in a popup window.

Dataset Augmentation. To extend MaskSearch for machine learning workflows, this demonstration also incorporates a dataset augmentation feature, which is further described in §4.

4 DEMONSTRATION SCENARIOS

Our demonstration will walk through a series of scenarios that show MaskSearch's utility in real-world machine learning workflows:

Scenario 1: Debugging Image Classification Models [18], illustrated in Figure 2. Recall the scenario mentioned in §1. Alice noticed that the model learned to rely on the presence of confounding factors in the background to classify the animals, as shown in Step 1 in Figure 2. To mitigate the model's reliance on confounding factors, Alice can first use a Top-K query to retrieve the images with the least number of high-value pixels in the ROI (object bounding box generated by YOLO [13]) normalized by the area of the ROI, as shown in Step 2 in Figure 2. Another option is to use a Filter query to retrieve all the images for which the normalized number of high-value pixels in the ROI is below a threshold. She can then augment her training set by randomizing the pixels outside the ROI in the retrieved images with the original labels, as shown in Step 3 in Figure 2, and retrain her model on the augmented training set, which guides the model to classify the animals without relying on the randomized background pixels.

In this scenario, we demonstrate MaskSearch 's ability to execute Top-K and Filter queries efficiently. On an AWS EC2 p3.2xlarge instance which has an Intel Xeon E5-2686 v4 processor with 8 vCPUs and 61 GiB of memory, and EBS gp3 volumes provisioned with 125 MiB/s throughput for disk storage, without MaskSearch, the median execution times of 5 Filter queries and 5 Top-K queries (OS page cache cleared before each run) on 22,275 images (with their model saliency masks) from the *iWildCam* dataset [4] are both around 100 seconds (wall clock time). In contrast, it takes MaskSearch less than a second to execute the same queries (OS page cache cleared before each run), which is a 100× speedup.

The conference attendees will interact with MaskSearch using the interface. They will be able to explore misclassified images and execute Top-K and Filter queries (we will pre-populate the fields and the attendees will be able to change the values). After clicking "Start Query", returned images overlaid by their corresponding saliency maps will be displayed. Attendees will also be able to click the "Augment" button to augment those images, and the result will be shown on the interface. Finally, we will provide an additional tab showing the details of the CHI and how different image masks were effectively filtered during query execution.

Scenario 2: Identifying Adversarial Attacks [20]. Claudia is an ML engineer who develops and maintains an image classification model that performs with high accuracy in production. During a routine check, she discovers that there is a significant drop in the prediction accuracy. Claudia examines the misclassified images manually and they look normal. However, after computing the model saliency maps for those images, she notices that the model's attention is diffused across irrelevant regions similar to the example shown in Figure 3 (b). Hence, she starts to suspect the misclassification may be due to malicious modifications that mislead the model to focus on irrelevant pixels. She wishes to retrieve the saliency maps that contain the most mid-value pixels, which indicates diffused model attention. With MaskSearch, she specifies the ROI as the full mask and issues a Top-K query. An example query she might use is,

```
SELECT mask_id FROM MasksDatabaseView
ORDER BY CP(mask, full_img, (0.2, 0.6)) DESC LIMIT 25;
```

By examining the returned masks (and their corresponding images), Claudia could better understand whether (and to what extent) the images were maliciously modified and improve the model's resilience to such malicious modifications.

The conference attendees will be able to walk through the scenario with the same interface shown in Figure 2. They will first

(a) before the attack (b) after the attack

Figure 3: Saliency masks before and after a malicious attack on an example image from ImageNet [5]. The object of interest in the image is the fish held by the man.

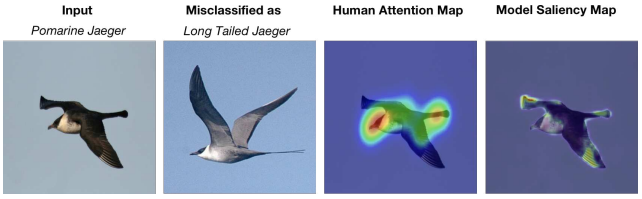

Figure 4: Comparison of human attention maps and model saliency maps on images from CUB-200-2011 [19]. The human attention map shows that humans look at the head and tail of the Pomarine Jaeger to classify it, which are the discriminate traits. The model saliency map shows that the model is focusing on the wings instead. This explains why the model misclassifies Pomarine Jaeger as Long Tailed Jaeger.

see both attacked and unattacked images and their corresponding saliency maps shown side-by-side to explore different patterns, e.g., focused attention vs. diffused attention, between the two categories; Based on the observation of which range the majority of diffused attention pixel values fall within, they can establish custom upper and lower bounds in Top-K query to obtain masks with the most (or least) diffused attention. Attendees will be able to examine the model saliency maps overlaid on the returned images.

Scenario 3: Investigating discrepancies between model saliency and human attention [2]. This scenario demonstrates MaskSearch's ability to perform aggregation queries efficiently. Fine-grained image classification requires identifying local and discriminate regions that correspond to subtle visual traits. Exploiting human attention can rectify models that deviate from critical traits for making correct predictions [14]. An example is illustrated in Figure 4. Imagine a scenario in which a researcher, Bob, wants to investigate whether a fine-grained classification model is looking at the same region as humans to classify images. He first thresholds the saliency maps and human attention maps (pixels > threshold becomes 1; otherwise 0) to reduce noise in the masks. With MaskSearch, he can then efficiently retrieve the images where the attention of the model and human experts has the lowest degree of alignment by aggregating the human attention and model saliency masks (group by `image_id`) and computing the Intersection over Union (IoU). An example query he might use is shown below:

```
SELECT image_id,
    CP(intersect(mask > 0.8), roi, (lv, uv))
        / CP(union(mask > 0.8), roi, (lv, uv)) as iou
FROM MasksDatabaseView WHERE mask_type IN (1, 2)
GROUP BY image_id ORDER BY iou ASC LIMIT 25;
```

In this scenario, the conference attendees will be guided to execute aggregation queries on the given human attention map and model saliency map with MaskSearch. They need to define a value T for thresholding the two masks and either start a Filter query or a Top-K query following the same input procedure described in Scenario 1, except that the ROI is set to the whole image. The query will return a list of images where the human attention map and model saliency map have the lowest IoU. Attendees will see the two masks of those images presented side-by-side on the GUI.

5 RELATED WORK

Although prior work has proposed systems that support queries over image databases [3, 7, 15], these methods are not optimized for MaskSearch's target queries. Array databases [12] specialize in handling multi-dimensional dense arrays but do not support efficient searching from large numbers of arrays. MaskSearch falls into the group of systems that support ML model inspection, explanation, and debugging [9, 11, 17], among which DeepEverest [9] is most relevant to MaskSearch, but it targets a fundamentally different class of queries.

ACKNOWLEDGMENTS

This work was supported in part by NSF grant OAC-1934292 and a gift from Microsoft.

REFERENCES

[1] 2024. Traffic Monitoring. https://datafromsky.com/traffic-monitoring/. Last accessed: Jul. 18, 2024.
[2] Das el al. 2017. Human attention in visual question answering: Do humans and deep networks look at the same regions? *Computer Vision and Image Understanding* (2017).
[3] Beaver et al. 2010. Finding a Needle in Haystack: Facebook's Photo Storage. In *OSDI*, Vol. 10. 1–8.
[4] Beery et al. 2020. The iWildCam 2020 Competition Dataset. *arXiv preprint arXiv:2004.10340* (2020).
[5] Deng et al. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*. 248–255.
[6] DeGrave et al. 2021. AI for radiographic COVID-19 detection selects shortcuts over signal. *Nature Machine Intelligence* 3, 7 (2021), 610–619.
[7] Flickner et al. 1995. Query by image and video content: the QBIC system. *Computer* 28, 9 (1995), 23–32.
[8] Hong et al. 2020. Human factors in model interpretability: Industry practices, challenges, and needs. *PACM HCI* 4, CSCW1 (2020), 1–26.
[9] He et al. 2021. DeepEverest: Accelerating Declarative Top-K Queries for Deep Neural Network Interpretation. *Proc. VLDB Endow.* 15, 1 (2021), 98–111.
[10] He et al. 2023. MaskSearch: Querying Image Masks at Scale. *arXiv preprint arXiv:2305.02375* (2023).
[11] Mehta et al. 2020. Toward Sampling for Deep Learning Model Diagnosis. In *ICDE*. IEEE, 1910–1913.
[12] Papadopoulos et al. 2016. The tiledb array data storage manager. *Proc. VLDB Endow.* 10, 4 (2016), 349–360.
[13] Redmon et al. 2016. You only look once: Unified, real-time object detection. In *CVPR*. 779–788.
[14] Rong et al. 2021. Human attention in fine-grained classification. *arXiv preprint arXiv:2111.01628* (2021).
[15] Remis et al. 2021. Using VDMS to Index and Search 100M Images. *Proc. VLDB Endow.* 14, 12 (2021), 3240–3252.
[16] Selvaraju et al. 2017. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In *ICCV*. 618–626.
[17] Sellam et al. 2019. Deepbase: Deep inspection of neural networks. In *SIGMOD*. 1117–1134.
[18] Teso et al. 2023. Leveraging explanations in interactive machine learning: An overview. *Frontiers in Artificial Intelligence* 6 (2023).
[19] Wah et al. 2011. *Caltech-UCSD Birds-200-2011 Dataset.* Technical Report CNS-TR-2011-001. California Institute of Technology.
[20] Ye et al. 2022. Detection defense against adversarial attacks with saliency map. *International Journal of Intelligent Systems* 37, 12 (2022).