Automatic Semantic Image Tagging at Scale Using CLIP: Method, Evaluation, and Comparison with Modern Approaches

Yurij Mikhalevich
QA Wolf
Melbourne, Australia
email: yurij@mikhalevi.ch

***Abstract* - This paper presents an extended study of `rtag`, a practical, scalable, command-line tool for automatic semantic image tagging that leverages the Contrastive Language-Image Pre-Training model. Extending the original conference paper, this work contributes a survey of modern vision-language models and image tagging approaches, an in-depth analysis of prompt engineering strategies for zero-shot tagging, and a systematic comparison with existing tagging models and photo management applications. Evaluating three prompt templates on the ObjectNet dataset, the best template improves top-1 accuracy by 4.36 percentage points (from 25.37% to 29.73%) and top-5 accuracy by 5.94 percentage points (from 48.09% to 54.03%), substantially exceeding the improvement previously reported on ImageNet. We further discuss similarity threshold selection and its effect on tagging precision and recall, as well as standard image metadata formats and their role in integrating automated tagging into photo management workflows. The tool caches image features in an embedded database, enabling efficient re-tagging of previously indexed collections; the scalability analysis confirms approximately linear scaling of both indexing and tagging with the image count, with the cache providing a 25.3% speedup for previously indexed images. These results show that flexible, zero-shot image tagging with arbitrary user-defined vocabularies is practical at scale on consumer hardware, benefiting sectors that handle large volumes of visual content such as media and entertainment, digital archiving, healthcare, and personal photo management.**

Keywords* - *image tagging; zero-shot classification; CLIP; vision-language models; image metadata.

I. Introduction

This paper is an extended version of the conference paper [1], which introduced `rtag`, a command-line tool for automatic semantic image tagging based on CLIP. It substantially extends that work with a comprehensive survey of vision-language models, an analysis of prompt engineering and threshold selection, and a systematic comparison with existing tagging tools.

The unveiling of the Contrastive Language-Image Pre-Training (CLIP) model by OpenAI in 2021 has captured widespread interest across the domains of Natural Language Processing (NLP) and Computer Vision (CV). This innovative model is capable of understanding advanced image representations by analyzing a vast collection of 400 million image and text pairs sourced from the internet. CLIP uniquely identifies the most applicable text snippet for an image through natural language guidance without being directly trained for such tasks. Its zero-shot learning capabilities mirror those found in GPT-2 and GPT-3 [2]. The authors of CLIP have showcased its ability to match the performance of the original ResNet50 on the ImageNet dataset in a zero-shot setting without using any of the 1.28 million labeled examples. This breakthrough addresses some of the most daunting challenges in the field of computer vision.

Since CLIP's release, the vision-language model landscape has evolved rapidly. Models such as ALIGN [3], BLIP [4], SigLIP [5], and OpenCLIP [6] have pushed the boundaries of contrastive learning, while specialized image tagging models like Tag2Text [7] and RAM [8] have advanced image tagging from fixed-vocabulary approaches toward open-set recognition. This rapidly evolving landscape motivates a comprehensive review and comparison of approaches for automatic image tagging.

Given its capabilities, CLIP emerges as an ideal foundation for developing a semantic image tagging tool capable of operating in a zero-shot manner with any user-provided tags or images. This article explores this particular application of CLIP.

The rapid expansion of data, particularly image data, underscores the timeliness and significance of this research. Industry estimates suggest that billions of images are shared online daily across social media, messaging platforms, and cloud storage services. The surge in digital device usage and internet accessibility has led to an unprecedented increase in image production and sharing online, complicating the task of manually locating specific images. In a previous study, we introduced a method that employs natural language queries for image searches using the CLIP model, resulting in the development of a tool named `rclip` [9]. While `rclip` has broad applications, there are a lot of other use cases where it is important to be able to tag images to enable their manual cataloging and search using third-party software. This makes an efficient image tagging solution increasingly necessary and relevant.

The key contributions of this extended work beyond the conference paper are as follows:

- A comprehensive survey of modern vision-language models and image tagging approaches, including CLIP, ALIGN, BLIP, SigLIP, RAM, and Tag2Text.
- A detailed analysis of prompt engineering strategies for CLIP-based tagging and their effect on zero-shot classification accuracy.
- An in-depth discussion of similarity threshold selection and its impact on tagging quality, including the trade-off between precision and recall.

- A systematic comparison with existing image tagging tools and services, highlighting `rtag`'s unique positioning in the ecosystem.
- An extended discussion of image metadata standards (IPTC, XMP, EXIF) and integration with photo management workflows.
- An architecture diagram and enriched qualitative analysis of tagging results.

The remainder of this paper is organized as follows. Section II reviews related works in deep learning, vision-language models, image tagging, and prompt engineering. Section III describes the proposed method for image tagging. Section IV presents the implementation details, including a system architecture diagram and metadata standards. Section V discusses the performance benchmarks. Section VI analyzes the tagging quality results. Section VII provides a systematic comparison with existing image tagging solutions. Section VIII discusses practical applications and limitations. Finally, Section IX concludes the paper and outlines future work.

II. Related Work

The advancement of NLP and CV has been significantly influenced by the development of models capable of understanding and generating insights from both text and images. The CLIP model by OpenAI marks a pivotal moment in this journey, leveraging a novel approach to learn visual concepts from natural language descriptions. This section reviews the literature that contextualizes CLIP within the broader field of image tagging and multimodal learning.

A. Foundations of Deep Learning for Image Understanding

One of the earliest attempts to bridge the gap between vision and language is the work by Socher et al. [10], who introduced a model for generating descriptions of image contents, paving the way for subsequent research in multimodal learning. Their model was among the first to use a neural network to combine visual and textual data, setting a foundation for future exploration in the field.

The concept of using neural networks for image classification was revolutionized by Krizhevsky et al. [11], with the introduction of AlexNet, which demonstrated the potential of deep learning in computer vision tasks. AlexNet achieved a top-5 error rate of 15.3% on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), reducing the previous best error by over 10 percentage points and establishing deep convolutional neural networks (CNNs) as the dominant paradigm for image classification.

Following AlexNet, two architectures significantly advanced the field. VGGNet [12] demonstrated that deeper networks with small (3×3) convolutional filters could achieve strong performance, reaching 7.3% top-5 error on ILSVRC 2014. GoogLeNet (Inception) [13] introduced the inception module, which applies multiple filter sizes in parallel, achieving 6.7% top-5 error while maintaining computational efficiency.

The development of the ResNet model by He et al. [14] represented another major step forward, introducing the concept of residual learning to enable the training of networks with over 100 layers. ResNet won ILSVRC 2015 with a 3.57% top-5 error rate, surpassing human-level performance. The advancements in image classification models like ResNet laid the groundwork for more sophisticated image understanding and tagging capabilities.

In parallel to improvements in image classification, research in NLP was making strides with the development of models like BERT by Devlin et al. [15], which introduced a new method for pre-training language representations using bidirectional transformers. BERT's success in understanding context and nuance in text provided inspiration for exploring similar pre-training approaches in multimodal models.

The intersection of NLP and CV began to take a more defined shape with the introduction of models like ViLBERT by Lu et al. [16], which employed separate pathways for processing visual and textual information before integrating them through co-attentional transformer layers, demonstrating improved performance on tasks requiring both visual and textual understanding.

The Transformer architecture [17], originally developed for machine translation, has proven to be a versatile backbone for multimodal learning. Its self-attention mechanism enables models to focus on relevant parts of the input when generating tags or descriptions, facilitating more nuanced interpretation of visual and textual data.

B. Vision-Language Models and Contrastive Learning

The development of CLIP by Radford et al. [2] stands as a significant milestone in vision-language modeling. CLIP employs a dual-encoder architecture: a Vision Transformer (ViT) [18] encodes images, while a text transformer encodes natural language descriptions. The model is trained using contrastive learning on 400 million image-text pairs collected from the internet (the WebImageText dataset). During training, CLIP learns to maximize the cosine similarity between matching image-text pairs while minimizing it for non-matching pairs. This approach enables remarkable zero-shot transfer: at inference time, the model can classify images into arbitrary categories by computing similarity between image features and text features derived from category descriptions.

CLIP's ViT backbone comes in several variants with different capacity and performance trade-offs. ViT-B/32 achieves 63.2% top-1 accuracy on ImageNet in a zero-shot setting, while ViT-L/14 reaches 75.3%. The choice of backbone directly affects both accuracy and computational cost.

ALIGN [3] scaled contrastive learning to 1.8 billion noisy image-text pairs collected from the web without careful curation. Despite the noise in training data, ALIGN achieved competitive zero-shot performance, demonstrating that scale alone can compensate for imperfect data quality, even with only minimal frequency-based filtering and without careful curation.

BLIP [4] introduced a bootstrapping approach to language-image pre-training, combining a multimodal encoder-decoder architecture with a captioning model that generates synthetic

captions and a filter that removes noisy ones. This self-refinement of training data yields improved performance on both understanding and generation tasks.

SigLIP [5] proposed replacing CLIP's softmax-based contrastive loss with a sigmoid loss function that operates on individual image-text pairs rather than requiring the full batch for normalization. This modification simplifies the training procedure, enables more efficient distributed training, and achieves competitive or superior performance compared to the original CLIP formulation. SigLIP 2 [19] extends this approach with multilingual support, improved zero-shot classification, and dense feature extraction capabilities.

OpenCLIP [6] provides an open-source reproduction of CLIP's training procedure, trained on publicly available datasets such as LAION-5B [20]. OpenCLIP established reproducible scaling laws for contrastive language-image learning, showing that both model size and dataset size contribute to improved zero-shot performance. The availability of open-source training code and public checkpoints has made OpenCLIP a foundation for many downstream applications. OpenCLIP's ViT-H/14 variant trained on LAION-2B achieves approximately 78.0% zero-shot top-1 accuracy on ImageNet, surpassing the original OpenAI CLIP models.

C. Image Tagging Models and Approaches

In the domain of image tagging, prior works have explored various approaches for automating the process. Traditional multi-label classification approaches use supervised learning on fixed label sets. Wang et al. [21] introduced a CNN-RNN framework that enhances the accuracy of tagging by leveraging the correlations between labels. Their method predicts multiple relevant tags for a single image by modeling dependencies among tags, producing more accurate and contextually relevant annotations. However, such supervised approaches are limited to the categories seen during training.

The Recognize Anything Model (RAM) [8] represents a significant advancement in image tagging. RAM uses a large-scale annotation pipeline to automatically generate training data from image-text pairs, supporting over 6,400 tag categories. RAM employs a vision transformer backbone with a tag recognition decoder and achieves strong performance, outperforming CLIP and BLIP on standard tagging benchmarks. While RAM is primarily evaluated on its predefined tag set, its use of a CLIP-based text encoder for label queries gives it open-set capability, enabling recognition of tags not seen during training. However, RAM requires GPU resources for inference, with model sizes of approximately 5.6 GB.

RAM++ [22] builds on RAM's open-set foundation by incorporating multi-grained text supervision to significantly improve recognition of diverse and rare open-set categories. This is achieved through a retrieval-augmented approach that combines RAM's semantic tag embeddings with enhanced text representations. While more flexible than the original RAM, RAM++ still relies on a specialized architecture and substantial computational resources.

Tag2Text [7] combines image tagging with image captioning in a unified framework. The model supports 3,429 tag categories and can simultaneously generate descriptive captions and structured tags. Tag2Text demonstrates that the tagging and captioning tasks are complementary: tags provide grounding for more accurate captions, while the captioning objective improves tag recognition.

Weyand et al. [23] introduced PlaNet, a method for geolocating images using a deep neural network. PlaNet demonstrates the potential of using image content to infer metadata beyond simple object recognition, assigning geographic tags based on visual features.

In contrast to these approaches, `rtag` occupies a distinct niche. While RAM offers open-set tagging capability and Tag2Text operates with a fixed vocabulary, both use specialized architectures. In contrast, `rtag` leverages CLIP's general-purpose zero-shot capability to support completely arbitrary, user-defined tags without any specialized training. This flexibility comes at the cost of lower accuracy on fixed-vocabulary benchmarks, but enables use cases that specialized models cannot address, such as tagging with domain-specific or personal categories.

D. Prompt Engineering for Vision-Language Models

The effectiveness of CLIP-based zero-shot classification is sensitive to the textual prompts used to represent categories. Radford et al. [2] demonstrated that using a simple template like "a photo of a {label}" provides a 1.3 percentage point improvement on ImageNet over using bare class names. Furthermore, using an ensemble of 80 prompt templates (e.g., "a photo of a {label}", "a good photo of a {label}", "a rendering of a {label}") yields an additional 3.5 percentage point improvement over the single template, for a total gain of approximately 4.8 percentage points over bare class names. Even simple template modifications have measurable effects: template variations such as "a photo of a {class}" vs. "a photo of {class}" can produce approximately 5 percentage points difference on fine-grained datasets like Caltech-101.

This sensitivity has motivated research on automatic prompt optimization. CoOp [24] replaces hand-crafted prompt templates with learnable continuous vectors that are optimized on a few labeled examples, achieving significant improvements over hand-crafted prompts on several benchmarks. CoCoOp [25] extends this approach with conditional context optimization, generating input-dependent prompts that generalize better to unseen classes.

An alternative direction uses large language models (LLMs) to generate descriptive prompts automatically. Menon and Vondrick [26] proposed using GPT-3 to generate category descriptions (e.g., for "cat": "has whiskers", "has pointed ears", "is a small domesticated carnivorous mammal"), achieving substantial improvements by leveraging the LLM's world knowledge to create more discriminative text representations. This approach can yield up to 5 percentage points improvement on challenging benchmarks without requiring any labeled images.

Prompt engineering is directly relevant to `rtag`'s design. As we show in Section VI, evaluating three prompt templates on ObjectNet reveals a 4.36 percentage point improvement in top-1 accuracy with the "photo of a [tag]" template, substantially exceeding the 1.3 percentage point gain reported by Radford et al. on ImageNet. This suggests that prompt optimization has an outsized effect on real-world photograph datasets and that further optimization could yield additional gains.

III. Method

This section formalizes the approach behind `rtag`, covering the image-tag similarity computation, the threshold-based tag-selection rule, the role of prompt templates, and the optimizations that make the method scalable.

A. CLIP-Based Similarity Computation

CLIP's dual-encoder architecture produces aligned embeddings for images and text in a shared n-dimensional vector space (where n depends on the CLIP model variant; $n = 512$ for ViT-B/32). Given an image i and a text label (tag) t, the CLIP image encoder f_{img} and text encoder f_{text} produce L2-normalized feature vectors $\mathbf{v}_i = f_{\text{img}}(i)$ and $\mathbf{v}_t = f_{\text{text}}(t)$, respectively.

The similarity between an image and a tag is computed via the dot product (Equation (1)) of their normalized feature vectors:

$$\mathbf{v}_i \cdot \mathbf{v}_t = \sum_{k=1}^{n} v_{i,k} \cdot v_{t,k} \tag{1}$$

Since CLIP produces L2-normalized vectors ($\|\mathbf{v}_i\| = \|\mathbf{v}_t\| = 1$), the dot product is equivalent to cosine similarity (Equation (2)):

$$\cos(\theta) = \frac{\mathbf{v}_i \cdot \mathbf{v}_t}{\|\mathbf{v}_i\| \cdot \|\mathbf{v}_t\|} = \mathbf{v}_i \cdot \mathbf{v}_t \tag{2}$$

For a collection of m images and p tags, the full similarity matrix can be computed efficiently as a single matrix multiplication:

$$\mathbf{S} = \mathbf{I} \cdot \mathbf{T}^{\top} \tag{3}$$

where $\mathbf{I} \in \mathbb{R}^{m \times n}$ is the matrix of image feature vectors and $\mathbf{T} \in \mathbb{R}^{p \times n}$ is the matrix of tag feature vectors. The resulting matrix $\mathbf{S} \in \mathbb{R}^{m \times p}$ contains the cosine similarity score between every image-tag pair.

B. Tag Selection and Threshold Semantics

Given the similarity matrix $\mathbf{S}$, a tag t_j is assigned to image i if and only if their similarity exceeds a configurable threshold τ:

$$\text{assign}(i, t_j) = \begin{cases} 1 & \text{if } S_{ij} > \tau \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The threshold τ controls the trade-off between precision and recall. A higher threshold produces fewer tags per image with higher confidence (higher precision, lower recall), while a lower threshold produces more tags, including less confident ones (lower precision, higher recall).

The default threshold of $\tau = 0.25$ was chosen empirically. In practice, raw CLIP cosine similarity scores for relevant image-tag pairs typically fall in the range of 0.15–0.40, with highly relevant matches reaching 0.30–0.35. A threshold of 0.25 provides a reasonable balance: it captures tags with meaningful semantic relevance while filtering out noise. Unlike classification tasks where the model must select a single best class, image tagging is inherently a multi-label problem where multiple tags can apply simultaneously, making the threshold a critical parameter.

C. Prompt Template Selection

In general, a tag text t can be wrapped in a prompt template T before encoding:

$$\mathbf{v}_t = f_{\text{text}}(T(t)) \tag{5}$$

Template choice matters because CLIP was trained on natural language captions, not isolated labels. A bare label like "cat" lacks the syntactic context of a natural sentence, potentially leading to suboptimal embeddings. Templates such as "photo of a [tag]" bridge this gap by framing the label as a caption-like phrase. However, `rtag` passes tags directly to CLIP's text encoder without wrapping them in a template. This design choice preserves maximum flexibility: tags are not restricted to object labels but can be arbitrary descriptive text such as "warm lighting," "taken outdoors," or "birthday party." The benchmark in Section VI evaluates the effect of different prompt templates on classification accuracy, showing that templates like "photo of a [tag]" substantially improve results when tags are simple object labels.

Template ensembling offers a way to further improve embeddings. Given N templates $T_1, \ldots, T_N$, the ensembled tag embedding is:

$$\mathbf{v}_t^{\text{ens}} = \text{normalize}\left(\frac{1}{N}\sum_{k=1}^{N} f_{\text{text}}(T_k(t))\right) \tag{6}$$

Radford et al. [2] report a 3.5 percentage point improvement on ImageNet using an 80-template ensemble over a single prompt template. User-configurable template strings and ensembling are promising directions for future work (Section IX).

D. Scalability Optimizations

To allow tagging any kind of image with any set of tags, the approach suggests allowing the input of a custom list of tags. While this approach works reasonably fast on a few images and tags, it may not be scalable when dealing with a large number of images or tags, particularly without access to GPUs. To address this scalability issue, we propose two optimizations:

- **Pre-compute tag feature vectors**: Tag embeddings are computed once and reused across all images. Since the

number of tags p is typically much smaller than the number of images m, the text encoder inference cost $O(p \cdot c_{\text{text}})$ is negligible compared to image encoding.
- **Utilize `rclip`'s cache layer**: The `rclip` [27] cache stores computed image feature vectors in an SQLite [28] database after initial processing. By reusing these cached vectors, `rtag` avoids the expensive image encoding step entirely for previously indexed images.

Without caching, the total computational cost is $O(m \cdot c_{\text{img}} + p \cdot c_{\text{text}} + m \cdot p \cdot d)$, where c_{img} is the cost of encoding one image, c_{text} is the cost of encoding one text label, and d is the embedding dimension. With caching, the cost reduces to $O(p \cdot c_{\text{text}} + m \cdot p \cdot d)$, since $c_{\text{img}} \gg p \cdot d$ (CLIP image inference dominates). The similarity computation $O(m \cdot p \cdot d)$ is a simple matrix multiplication and is fast even for large datasets.

The `rclip` cache also handles incremental updates: when new images are added to a directory, only the new images require CLIP inference. This is particularly useful when the user needs to retag images with a different set of tags or when the images are already indexed by `rclip`.

Once the tags are selected, they are written to the image metadata, allowing other software to utilize the tags. For example, the user can search for images by tag using the operating system's file manager or a third-party image management application.

IV. Implementation

This section presents how the method is realized in practice, describing the system architecture, the core processing pipeline, the available operating modes, and the image metadata standards used to store the resulting tags.

A. System Architecture

The method described in Section III is implemented in the Python utility called `rtag` [29]. `rtag` provides an easy-to-use Command-Line Interface (CLI) that allows users to tag images within any directory on a computer where `rtag` is installed. Tagging images within nested directories of a complex directory subtree is also supported. To use it, the user opens the terminal, navigates to the target directory, types `rtag`, and hits "Enter."

Figure 1 shows the overall system pipeline. The architecture consists of two input paths—tag processing and image processing—that converge at the similarity computation stage. Tags are encoded once by CLIP's text encoder, while image feature vectors are retrieved from the `rclip` SQLite cache (or computed on the fly for unindexed images). The dot-product similarity between all image-tag pairs is computed as a matrix multiplication. Tags exceeding the threshold are passed to the IPTC metadata writer, which embeds them into the image files.

The solution uses the `rclip` library [27] to compute the feature vectors. `rclip` uses the `ViT-B/32` version of OpenAI's CLIP model [2]. The tool writes the computed tags to the image IPTC metadata [30] using the IPTCInfo3 Python library [31].

B. Core Processing Pipeline

A simplified version of the image processing loop is shown in Figure 2. The pipeline operates as follows: (1) tag feature vectors are computed once by passing all tags through CLIP's text encoder; (2) image feature vectors are retrieved from the `rclip` SQLite database; (3) for each image, the dot product between its feature vector and all tag vectors is computed using a single matrix multiplication (`image_features @ tag_features.T`); (4) tags with similarity above the threshold are selected; and (5) selected tags are written to the image's IPTC metadata.

Images are processed sequentially in the current implementation. Each image's feature vector is a compact 512-dimensional float32 array (2 KB), making the per-image memory footprint negligible. The dominant memory cost is the tag feature matrix, which for the default ImageNet-1k label set (1,000 tags) is approximately 2 MB.

C. Operating Modes and Configuration

Figure 3 shows how tags are loaded. By default, `rtag` uses the list of the ImageNet-1k [32] labels as tags. The ImageNet-1k label set contains 1,000 categories spanning a broad range of everyday objects, animals, vehicles, and natural scenes, making it a reasonable default for general-purpose tagging. The user can provide their own list of tags by specifying the `--tags-filepath` argument, e.g., `rtag --tags-filepath ./tags.txt`, where the file contains a newline-separated list of tags.

`rtag` supports two modes of operation:
- **Append mode** (`--mode append`): Computed tags are merged with existing tags in the image metadata. Duplicate tags are automatically removed. This mode is useful for incremental tagging, e.g., first tagging with general categories and then adding domain-specific tags.
- **Overwrite mode** (`--mode overwrite`): Existing tags are replaced entirely with the computed tags. This mode is appropriate when re-tagging with a completely new tag set.

The `--threshold` argument allows the user to control the number of tags written per image. A higher threshold produces fewer, more confident tags; a lower threshold produces more tags with greater recall. The default value of 0.25 produces optimal results in practice.

The `--dry-run` argument shows what tags would be written without actually modifying any files. This is useful for experimenting with different tag sets and threshold values before committing to changes.

The `rtag` source code is published on GitHub under the MIT license [29].

D. Image Metadata Standards

`rtag` writes tags using the IPTC IIM (Information Interchange Model) standard [30] via the IPTCInfo3 library [31]. The IPTC Photo Metadata Standard defines a "keywords" field specifically designed for descriptive tags, making it the natural choice for image tagging applications.

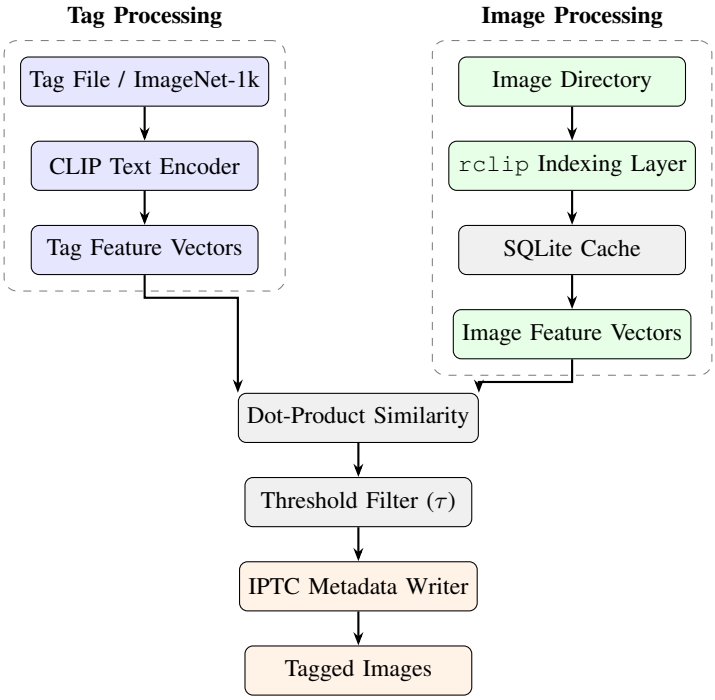


Figure 1. The `rtag` system pipeline. The *tag processing* path (left) encodes user-provided tags (or the default ImageNet-1k label set) using CLIP's text encoder. The *image processing* path (right) retrieves image feature vectors from the `rclip` SQLite cache, computing them on the fly for unindexed images. The similarity computation stage computes the dot product between all tag and image vectors, applies the threshold filter, and writes matching tags to IPTC metadata.

Several metadata standards exist for embedding information in image files:

- **EXIF** (Exchangeable Image File Format): Primarily stores camera settings and capture parameters (exposure, focal length, GPS coordinates). EXIF has limited support for keywords and is not well-suited for tagging workflows.
- **IPTC IIM**: A well-established standard for editorial metadata, including the "keywords" field used by `rtag`. IPTC is broadly supported by photo management software, including Adobe Lightroom, digiKam, macOS Finder, and Windows Explorer.
- **XMP** (Extensible Metadata Platform) [33]: Adobe's more general metadata framework, based on RDF/XML. XMP can represent all IPTC fields and more. XMP metadata can be embedded in the image file or stored in a separate sidecar file (typically with a `.xmp` extension).

IPTC was chosen for `rtag` because of its broad compatibility with existing photo management software and operating system file managers. A key limitation of the current approach is that IPTC metadata is embedded directly in the image file, which modifies the original. For workflows where non-destructive editing is important—particularly with RAW image formats—XMP sidecar files would be preferable, as they store metadata alongside the image without altering it. Support for XMP sidecar output is planned for a future release (Section IX).

V. Performance Analysis

This section describes the benchmarking methodology and reports the resulting processing times, analyzing how `rtag` scales with the image count and how much the feature cache contributes.

A. Benchmarking Methodology

Performance was benchmarked on the ObjectNet [34] dataset, which contains 50,273 images of 313 object categories. ObjectNet was chosen because its images feature controlled variations in background, rotation, and viewpoint, making it more challenging and representative of real-world conditions than standard benchmarks where objects are typically centered and well-lit.

All benchmarks were run on an Apple M1 Max CPU (10-core, 8 performance + 2 efficiency cores, 64 GB unified memory). Before running the benchmarks, the dataset was resized to 224 pixels by smaller dimension and converted to JPG using ImageMagick [35]. The 224-pixel dimension matches CLIP's expected input resolution. Processing times are wall-clock measurements from a single run.

B. Results and Scalability Analysis

Table I shows `rtag`'s performance when tagging previously indexed and unindexed images.

The theoretical scaling factor between the full dataset and the subset is $50{,}273/965 = 52.1\times$. The observed scaling factors are $381.25\text{s}/12.04\text{s} = 31.66\times$ (unindexed) and $284.79\text{s}/9.14\text{s} = 31.15\times$ (indexed). The sub-linear observed

```
tag_features = (
  model.compute_text_features(tags)
)

for image in tqdm(
  rclipDB.get_image_vectors_by_dir_path(
      current_directory),
  unit='images',
):
  image_path = image['filepath']
  image_features = np.frombuffer(
    image['vector'],
    np.float32,
  )

  similarities = image_features @
      tag_features.T

  new_tags = []
  for tag, similarity in zip(tags,
      similarities):
    if similarity > args.threshold:
      new_tags.append(tag)

  if not new_tags:
    continue

  image_metadata = IPTCInfo(
    image_path,
    force=True,
  )

  if args.mode == 'append':
    existing_tags = image_metadata['
        keywords']
    image_metadata['keywords'] = [*set(
        existing_tags + new_tags)]
  elif args.mode == 'overwrite':
    image_metadata['keywords'] = new_tags
  else:
    raise ValueError(f'Invalid mode: {
        args.mode}')

  image_metadata.save()
```

Figure 2. Image processing loop.

```
def load_tags_from_file(path: str):
  with open(path, 'r') as f:
    return f.read().splitlines()

tags_filepath = args.tags_filepath or
    get_imagenet_tags_filepath()
tags = load_tags_from_file(tags_filepath)
```

Figure 3. Tags loading.

scaling (31× vs. the theoretical 52×) is expected: fixed overhead costs such as model loading, SQLite database initialization, and Python interpreter startup are amortized over more images in the larger dataset, improving the effective per-image throughput.

The `rclip` cache provides a substantial speedup. For the full dataset, indexed processing (4m44.790s) is 25.3% faster than unindexed processing (6m21.250s). For the subset, the speedup is 24.1% (9.14s vs. 12.04s). This confirms that the caching mechanism provides a consistent benefit regardless of dataset size.

The effective throughput for the full dataset is approximately 177 images per second for indexed images ($50{,}273/284.79$) and 132 images per second for unindexed images ($50{,}273/381.25$). At these rates, a personal photo library of 50,000 images can be fully tagged in under 5 minutes when previously indexed by `rclip`.

The real-life `rtag` application possibilities go far beyond results demonstrated in these benchmarks because `rtag` can be used with any tags and images and not just the ones present in the dataset.

VI. Tagging Quality Analysis

This section explains the evaluation methodology, presents the quantitative results, examines the effect of prompt engineering, and offers a qualitative analysis of representative tagged images.

A. Evaluation Methodology

Tagging quality was evaluated using top-k accuracy on the ObjectNet [34] dataset. Top-k accuracy measures the fraction of images for which the correct label appears among the k highest-scoring tags. This metric is appropriate for evaluating zero-shot tagging systems because the model must rank the

TABLE I. Tagging performance on Apple M1 Max CPU.

Dataset	# of images	Indexed	Processing time
`ObjectNet` dataset	50,273	No	6m21.250s
`ObjectNet` dataset	50,273	Yes	4m44.790s
`ObjectNet` subset	965	No	0m12.040s
`ObjectNet` subset	965	Yes	0m09.140s

correct tag among the entire candidate vocabulary (313 ObjectNet category labels in our evaluation), rather than selecting from a small set of choices.

ObjectNet's ground truth provides a single correct label per image, drawn from 313 categories, of which 113 overlap with the ImageNet-1k label set. The evaluation assigns each image the tags with the highest similarity scores and checks whether the ground truth label appears in the top k. This simulates a realistic use case where the user provides a broad set of candidate tags and relies on the system to identify the most relevant ones.

ObjectNet was specifically designed with controlled biases: objects appear against random backgrounds, in unusual rotations, and from atypical viewpoints. This makes it substantially more challenging than standard benchmarks. Published results show that models typically experience a 40–45% performance drop on ObjectNet compared to their ImageNet accuracy, making it a stringent test of real-world generalization.

Alternative evaluation metrics include mean average precision (mAP) and per-category F1 score, which are more suited to multi-label evaluation. However, ObjectNet provides single-label ground truth, making top-k accuracy the natural choice for this dataset.

B. Results Analysis

Table II presents the tagging quality results.

The best template, `photo of a [tag]`, achieves 29.73% top-1 accuracy and should be interpreted in context. CLIP ViT-B/32 achieves approximately 44.3% top-1 accuracy on ObjectNet in standard zero-shot classification [6], where the model selects from ObjectNet's 113 overlapping categories [34] and is evaluated only on images from those categories. `rtag`'s evaluation uses all 313 ObjectNet category labels as the candidate tag set and evaluates on all 50,273 images from all 313 categories, making the task harder in two ways: the model must rank the correct tag among approximately 2.8 times more candidates, and it must classify images from the 200 additional categories that have no overlap with ImageNet.

The top-5 accuracy of 54.03% means that the correct tag appears among the five highest-scoring tags for over half of all images. This is practically useful: even when the model does not rank the correct tag first, it often ranks it highly, enabling effective tag-based browsing and search.

The gap between top-1 and top-5 accuracy (approximately 24 percentage points) indicates that CLIP frequently assigns high similarity scores to semantically related tags. This is a characteristic of contrastive text encoders: training on image–caption pairs causes embeddings for synonyms and taxonomically related terms (e.g., "speaker" and "loudspeaker") to be mathematically adjacent in the shared embedding space. For example, a "speaker" image may receive high scores for both "speaker" and "loudspeaker," and an image of a camera may score highly for both "camera" and "projector." In a tagging context, this is often desirable: semantically related tags improve discoverability.

TABLE II. `rtag` tagging quality on ObjectNet.

Template	Top-1 accuracy	Top-5 accuracy
`[tag]`	25.37%	48.09%
`photo of [tag]`	26.94%	50.20%
`photo of a [tag]`	29.73%	54.03%

C. Prompt Engineering Effect

Prompt template choice has a substantial effect on tagging accuracy. The three templates evaluated show a progressive improvement: the bare `[tag]` template achieves 25.37% top-1 accuracy, `photo of [tag]` improves this by 1.57 percentage points to 26.94%, and `photo of a [tag]` adds another 2.79 percentage points, reaching 29.73%. The total improvement from bare tags to the best template is 4.36 percentage points in top-1 accuracy (25.37% to 29.73%) and 5.94 percentage points in top-5 accuracy (48.09% to 54.03%). Notably, the addition of the article "a" alone accounts for more than half of the total gain.

Radford et al. [2] reported a 1.3 percentage point improvement on ImageNet with the "a photo of a {label}" template. `rtag`'s observed improvement on ObjectNet is more than three times larger (4.36pp vs. 1.3pp). This amplified effect may reflect the nature of the dataset: ObjectNet consists entirely of real-world photographs, making the "photo of" framing especially appropriate—CLIP was trained on internet image-caption pairs where captions frequently describe photos.

The progressive improvement across all three templates suggests that prompt framing has an outsized effect on datasets of real-world photographs. While template sensitivity is well documented on standard classification benchmarks (e.g., more than 5% in accuracy on Caltech-101 [24]), the improvement on ObjectNet is notable for its magnitude on a highly diverse, real-world dataset with 313 categories.

CLIP's 80-template ensemble yields a 3.5 percentage point improvement on ImageNet over a single prompt template. Notably, `rtag` already exceeds this ensemble gain with a single well-chosen template on ObjectNet (4.36pp), suggesting that template selection may be more impactful than ensembling for datasets well-matched to a specific prompt. Implementing template ensembling in `rtag` could provide further gains at the cost of increased text encoding time (proportional to the number of templates).

Recent work on LLM-generated prompts [26] has shown that using large language models to create category descriptions can yield substantial improvements on challenging benchmarks. Learnable prompt optimization approaches such as CoOp [24] and CoCoOp [25] offer another avenue, though they require labeled examples for the target domain. These directions represent significant potential for improving `rtag`'s tagging quality in future iterations.

D. Qualitative Analysis

Figures 4, 5, and 6 (at the end of the paper) show example images from the ObjectNet dataset tagged by `rtag` using the default ImageNet-1k tag set and a threshold of 0.25.

The speaker image (Figure 4) is assigned a broad set of tags. True positives include the combined label "loudspeaker, speaker, speaker unit," a direct match for the object. Related but less precise tags such as "radio" and "cassette player" reflect CLIP's semantic associations: speakers are commonly found alongside audio playback equipment. These semantically related tags, while not exact matches for the object, can be useful for browsing—a user searching for audio equipment would find this image through multiple related queries.

The soap image (Figure 5) receives a wide variety of tags. Category confusion between "soap dispenser," "bathtub," and "dishwasher" illustrates CLIP's tendency to associate objects with the contexts in which they commonly appear. Soap is frequently depicted near bathtubs and cleaning contexts, leading CLIP to assign high similarity to these related concepts. This contextual association is a feature of CLIP's training on captioned internet images, where co-occurring objects are frequently mentioned together.

The camera image (Figure 6) receives a cluster of tags centered on optical devices. The presence of "binoculars" and "opera glasses" alongside "reflex camera" and "Polaroid camera" shows CLIP's semantic clustering of optical devices. These objects share visual features (lenses, eyepieces, compact form factors) that lead to similar feature vectors in CLIP's embedding space.

A general observation across these examples is that CLIP tends to assign semantically related tags even when they are not exact matches. In a tagging context, this behavior is often beneficial: users browsing tagged images by related concepts will find relevant results even when the exact tag is not present. The threshold parameter allows users to control the specificity of tags—raising the threshold would retain only the highest-confidence tags, reducing both true and false positives. Despite occasional false positives, the added tags allow users to surface the images they are looking for, the primary goal in a browsing and retrieval context.

VII. Comparison with Existing Image Tagging Solutions

To contextualize `rtag`'s capabilities, we compare it with specialized image tagging models and photo management software that offer automatic tagging features.

A. Specialized Image Tagging Models

RAM (Recognize Anything Model) [8] supports over 6,400 tag categories and uses an automatic annotation pipeline trained on large-scale image-text pairs. RAM employs a Swin Transformer backbone with a specialized tag recognition head, achieving state-of-the-art performance on standard tagging benchmarks. While RAM's open-set architecture allows it to recognize categories beyond its predefined tags, it requires GPU resources for efficient inference and has a model size of approximately 5.6 GB.

RAM++ [22] improves upon RAM's open-set capability by incorporating multi-grained text supervision alongside RAM's semantic tag embeddings. This significantly enhances recognition of diverse and rare categories, though the approach still relies on RAM's specialized architecture.

Tag2Text [7] integrates tagging (3,429 categories) with image captioning. While more versatile than pure tagging models, Tag2Text requires similar computational resources and operates with a predefined category set for its tagging component.

`rtag` takes a fundamentally different approach. Rather than training a specialized tagging model, it leverages CLIP's general-purpose zero-shot capability. This means any text string can serve as a tag—the user is not limited to a predefined vocabulary. While this flexibility comes at the cost of lower accuracy compared to specialized models on their benchmark categories, it enables use cases that fixed-vocabulary models cannot support, such as tagging with highly domain-specific or personal categories.

B. Photo Management Software with Automatic Tagging

Several photo management applications offer built-in automatic tagging:

Google Photos uses proprietary cloud-based ML models trained on billions of photos, offering excellent accuracy for common objects, scenes, and faces. However, it requires uploading all photos to Google's cloud (raising privacy concerns), does not support custom tags, and the ML-generated searchable tags cannot be exported as standard IPTC or XMP metadata (though some metadata such as descriptions and locations can be exported via Google Takeout as JSON sidecar files).

Apple Photos performs on-device ML-based classification for objects, scenes, and faces. While privacy-preserving, it is locked to the Apple ecosystem and provides no user-facing CLI. The macOS version supports custom user-defined keywords via its Keyword Manager, which can be embedded as standard IPTC metadata upon export; the iOS version is more limited, offering only captions. Apple also provides PhotoKit, a developer API for interacting with the Photos library on iOS and macOS, though there is no command-line interface for end users.

digiKam [36] is an open-source photo management application with deep learning-based face recognition and image classification. It reads and writes IPTC and XMP metadata, making it compatible with `rtag`'s output. However, digiKam is GUI-only and has limited zero-shot tagging capability.

Immich [37] is an open-source, self-hosted alternative to Google Photos with ML-based tagging (using CLIP models) and face recognition. It requires server deployment and does not operate as a standalone command-line tagging tool.

PhotoPrism [38] is an open-source photo management application using TensorFlow-based classification. It offers basic automatic tagging but with limited accuracy compared to CLIP-based approaches and requires server deployment.

C. Feature Comparison

Table III summarizes the key differences between `rtag` and existing solutions across several dimensions.

TABLE III. Comparison of image tagging solutions.

Feature	rtag	Google Photos	Apple Photos	digiKam	Immich	PhotoPrism
Interface	CLI	Web/Mobile	Desktop/Mobile/Web	GUI	Web/Mobile	Web
Custom user tags	Yes	No	Yes (manual)	Yes (manual)	Yes (manual)	Yes (manual)
Arbitrary tag vocabulary	Yes	Proprietary	Manual	Manual	Manual	Fixed
Metadata standard	IPTC	Proprietary/JSON	IPTC (export)	IPTC/XMP	XMP/IPTC	YAML/XMP
Offline operation	Yes	No	Yes	Yes	Yes	Yes
Open source	Yes	No	No	Yes	Yes	Yes
GPU required	No	N/A	No	Optional	Optional	Optional

D. Discussion

`rtag` fills a unique niche in the image tagging ecosystem. It is the only tool that combines zero-shot tagging with arbitrary user-defined tags, CLI-first design, standard metadata output (IPTC), and offline CPU-only operation. This combination is particularly valuable for users who need scriptable, automatable tagging workflows that integrate with existing photo management software.

The fundamental trade-off is accuracy vs. flexibility. Specialized tagging models achieve higher accuracy on common categories because they are trained specifically for the tagging task. `rtag` sacrifices some accuracy by using CLIP's general-purpose embeddings, but gains the ability to tag with a virtually unlimited vocabulary—including highly specific niche subjects and arbitrary label combinations that would never appear in a standard closed-vocabulary dataset.

`rtag` is also complementary to photo management software. Users can run `rtag` to pre-tag images with IPTC metadata before importing them into applications like digiKam or Lightroom, which can read and display the tags. This workflow combines `rtag`'s zero-shot flexibility with the rich browsing and editing features of dedicated photo management tools.

VIII. Discussion

This section discusses the practical applications enabled by `rtag` and the limitations that constrain its current tagging quality.

A. Practical Applications

`rtag`'s combination of zero-shot tagging, arbitrary tag vocabularies, and scriptable CLI interface makes it applicable across several domains:

Media and entertainment: Bulk tagging of stock photo libraries with content descriptors, scene types, and mood keywords. Studios and agencies can define custom tag vocabularies tailored to their cataloging needs.

Digital archiving: Libraries, museums, and historical archives can tag digitized photo collections with subject matter, time period, and content descriptors. The ability to define custom tags is critical for specialized archival vocabularies.

Security and surveillance: Tagging extracted stills from CCTV or body camera footage with object and scene categories to enable keyword-based retrieval, useful for reviewing specific incidents.

Healthcare: Categorizing clinical photographs (dermatological images, wound documentation, surgical site photos) with domain-specific tags using a custom tag file. This can assist in organizing and retrieving images for clinical review.

Personal photo management: Tagging personal photo libraries with content categories to enable search via the operating system's file manager. Since `rtag` writes standard IPTC metadata, tags are visible in macOS Finder, Windows Explorer, and most photo management applications.

E-commerce: Product image tagging for catalog management, enabling attribute-based search (color, material, style) using custom tag vocabularies defined by the catalog taxonomy.

B. Limitations

Several limitations should be considered when evaluating `rtag`:

Model capacity: `rtag` currently uses CLIP ViT-B/32, the smallest backbone variant. Larger models (ViT-L/14, ViT-H/14) would improve tagging quality but at the cost of increased inference time and memory usage.

File modification: IPTC metadata is embedded directly in the image file, which modifies the original. No non-destructive option (XMP sidecar files) is currently available.

Single-label evaluation: ObjectNet provides single-label ground truth, limiting evaluation to top-k accuracy. Multi-label evaluation with metrics such as mAP or F1 score would provide a more complete picture of tagging quality but requires a multi-label benchmark dataset.

No GPU utilization: The current implementation runs entirely on CPU. While this ensures broad compatibility, GPU acceleration would significantly improve throughput for large-scale tagging tasks.

English-only: CLIP ViT-B/32 is trained primarily on English text, limiting tag support to English labels. Multilingual CLIP variants such as SigLIP 2 [19] could enable multi-language tagging.

Input resolution: CLIP's 224×224 pixel input resolution may lose fine details that are important for distinguishing visually similar objects.

Threshold sensitivity: The single global threshold τ applies uniformly to all tags. Per-category threshold optimization could improve results but is not currently implemented.

IX. Conclusion and Future Work

This paper presented an extended study of `rtag`, a command-line tool for automatic semantic image tagging based on OpenAI's CLIP model. Building on the original

conference publication, we provided a comprehensive survey of modern vision-language models and image tagging approaches, situating `rtag` within the broader landscape of contrastive learning, specialized tagging models, and prompt engineering research.

The core method computes dot-product similarity between CLIP-encoded image features and tag text features, assigning tags that exceed a configurable threshold. By leveraging the `rclip` caching infrastructure, the tool achieves efficient re-tagging with a 25.3% speedup for previously indexed images. The scalability analysis confirmed approximately linear scaling at larger dataset sizes, as fixed overhead costs (model loading, database initialization) are amortized over more images, with throughput of approximately 177 images per second on an Apple M1 Max CPU for indexed images.

Our analysis of prompt engineering effects across three templates showed a progressive improvement, with the "photo of a [tag]" template improving top-1 accuracy by 4.36 percentage points and top-5 accuracy by 5.94 percentage points on the ObjectNet dataset—more than three times the 1.3 percentage point gain reported by Radford et al. on ImageNet. This substantial improvement demonstrates that prompt template selection is a high-impact optimization for real-world photograph datasets, with further gains possible through template ensembling and learnable prompt tuning.

The systematic comparison with existing solutions revealed that `rtag` occupies a unique niche: it is the only tool combining zero-shot tagging with arbitrary user-defined tags, CLI-first design, standard IPTC metadata output, and offline CPU-only operation. While specialized models like RAM achieve higher accuracy on common tag categories, `rtag`'s flexibility and simplicity enable practical use cases without requiring specialized architectures or GPU resources.

The discussion of metadata standards highlighted the importance of using established formats like IPTC for interoperability with photo management software.

Building on the conference paper's [1] original future work items, we identify the following directions for strengthening `rtag` as a versatile tool for automated image organization.

From the conference paper:

- XMP sidecar file support for non-destructive tagging.
- Parallel image processing for improved throughput.
- Preventing re-indexing on file rename.
- Evaluation on additional datasets.

New directions identified in this extended study:

- Support for larger CLIP backbones (ViT-L/14, ViT-H/14) via migration to the `open_clip` library [6].
- GPU acceleration via MPS (Apple Silicon) and CUDA backends.
- User-configurable prompt template strings (e.g., `--template "photo of a [tag]"`) to let users opt into template wrapping when their tags are simple object labels, while preserving the default bare-tag behavior for maximum flexibility.
- Prompt template ensembling as default behavior, averaging embeddings across multiple templates for improved accuracy.
- Tuning of image preprocessing and cropping together with per-class similarity thresholds to reduce incorrect or missed labels.
- Learnable prompt tuning using CoOp [24] or CoCoOp [25] for domain-specific tagging applications.
- Integration with specialized tagging models (RAM, Tag2Text) for hybrid pipelines that combine zero-shot flexibility with specialized accuracy.
- Hierarchical tagging using WordNet or other ontologies to organize tags into taxonomic structures.
- Utilizing existing image metadata (GPS coordinates for location-based tagging, EXIF date for temporal tags) to enrich the tagging process.
- Multi-language tag support via multilingual CLIP models such as SigLIP 2 [19].
- Evaluation on multi-label benchmarks with metrics such as mAP and F1 score for a more comprehensive quality assessment.

`rtag` is open-source and available on GitHub [29].

References

[1] Y. Mikhalevich, "Automatic semantic image tagging at scale: AI-powered command-line tool based on CLIP," in *Proceedings of CONTENT 2024, The Sixteenth International Conference on Creative Content Technologies*, IARIA, 2024, pp. 7–13, ISBN: 978-1-68558-159-6.

[2] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, et al., "Learning transferable visual models from natural language supervision," in *International Conference on Machine Learning*, PMLR, 2021, pp. 8748–8763.

[3] C. Jia, Y. Yang, Y. Xia, Y.-T. Chen, Z. Parekh, H. Pham, et al., "Scaling up visual and vision-language representation learning with noisy text supervision," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 139, PMLR, 2021, pp. 4904–4916. [retrieved: Mar. 2026]. [Online]. Available: https://proceedings.mlr.press/v139/jia21b.html

[4] J. Li, D. Li, C. Xiong, and S. Hoi, "BLIP: Bootstrapping language-image pre-training for unified vision-language understanding and generation," in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 162, PMLR, 2022, pp. 12 888–12 900. [retrieved: Mar. 2026]. [Online]. Available: https://proceedings.mlr.press/v162/li22n.html

[5] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer, *Sigmoid loss for language image pre-training*, 2023. arXiv: 2303.15343 [cs.CV]. [retrieved: Mar. 2026]. [Online]. Available: https://arxiv.org/abs/2303.15343

[6] M. Cherti, R. Beaumont, R. Wightman, M. Wortsman, G. Ilharco, C. Gordon, et al., "Reproducible scaling laws for contrastive language-image learning," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2023, pp. 2818–2829. DOI: 10.1109/cvpr52729.2023.00276 [retrieved: Mar. 2026]. [Online]. Available: http://dx.doi.org/10.1109/CVPR52729.2023.00276

[7] X. Huang, Y. Zhang, J. Ma, W. Tian, R. Feng, Y. Zhang, et al., *Tag2text: Guiding vision-language model via image tagging*, 2023.

[8] Y. Zhang, X. Huang, J. Ma, Z. Li, Z. Luo, Y. Xie, et al., *Recognize anything: A strong image tagging model*, 2023.

[9] Y. Mikhalevich, "Using text queries to look up unlabeled images: A command-line search tool based on clip," in *Proceedings of CONTENT 2023, The Fifteenth International Conference on Creative Content Technologies*, IARIA, 2023, pp. 7–11.

[10] R. Socher, A. Karpathy, Q. V. Le, C. D. Manning, and A. Y. Ng, "Grounded compositional semantics for finding and describing images with sentences," *Transactions of the Association for Computational Linguistics*, vol. 2, pp. 207–218, 2014.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[13] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, et al., "Going deeper with convolutions," 2014. arXiv: 1409.4842 [cs.CV]. [retrieved: Mar. 2026]. [Online]. Available: https://arxiv.org/abs/1409.4842

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[16] J. Lu, D. Batra, D. Parikh, and S. Lee, "Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2019.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, et al., "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 6000–6010.

[18] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, et al., "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021.

[19] M. Tschannen, A. Gritsenko, X. Wang, M. F. Naeem, I. Alabdulmohsin, N. Parthasarathy, et al., *Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features*, 2025. arXiv: 2502.14786 [cs.CV]. [retrieved: Mar. 2026]. [Online]. Available: https://arxiv.org/abs/2502.14786

[20] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, et al., "Laion-5b: An open large-scale dataset for training next generation image-text models," in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, ser. NIPS '22, New Orleans, LA, USA: Curran Associates Inc., 2022, ISBN: 9781713871088.

[21] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang, and W. Xu, "Cnn-rnn: A unified framework for multi-label image classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2285–2294.

[22] X. Huang, Y.-J. Huang, Y. Zhang, W. Tian, R. Feng, Y. Zhang, et al., *Open-set image tagging with multi-grained text supervision*, 2023. arXiv: 2310.15200 [cs.CV]. [retrieved: Mar. 2026]. [Online]. Available: https://arxiv.org/abs/2310.15200

[23] T. Weyand, I. Kostrikov, and J. Philbin, "Planet - photo geolocation with convolutional neural networks," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII 14*, Springer, 2016, pp. 37–55.

[24] K. Zhou, J. Yang, C. C. Loy, and Z. Liu, "Learning to prompt for vision-language models," *International Journal of Computer Vision*, vol. 130, no. 9, pp. 2337–2348, 2022.

[25] K. Zhou, J. Yang, C. C. Loy, and Z. Liu, "Conditional prompt learning for vision-language models," 2022. arXiv: 2203.05557 [cs.CV]. [retrieved: Mar. 2026]. [Online]. Available: https://arxiv.org/abs/2203.05557

[26] S. Menon and C. Vondrick, "Visual classification via description from large language models," 2022. arXiv: 2210.07183 [cs.CV]. [retrieved: Mar. 2026]. [Online]. Available: https://arxiv.org/abs/2210.07183

[27] Y. Mikhalevich, *rclip*, version 1.7.26, Feb. 2024. [retrieved: Mar. 2026]. [Online]. Available: https://github.com/yurijmikhalevich/rclip

[28] R. D. Hipp, *SQLite*, version 3.31.1, 2020. [retrieved: Mar. 2026]. [Online]. Available: https://www.sqlite.org/index.html

[29] Y. Mikhalevich, *rtag*, version 0.1.0, Feb. 2024. [retrieved: Mar. 2026]. [Online]. Available: https://github.com/yurijmikhalevich/rtag

[30] *Iptc photo metadata standard*, Jan. 2023. [retrieved: Mar. 2026]. [Online]. Available: https://www.iptc.org/standards/photo-metadata/iptc-standard/

[31] T. Gulacsi and J. Campbell, *IPTCInfo3*, version 2.1.4, 2019. [retrieved: Mar. 2026]. [Online]. Available: https://github.com/james-see/iptcinfo3

[32] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, et al., "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[33] Adobe Systems Incorporated, "Xmp specification part 1: Data model, serialization, and core properties," Tech. Rep., 2012, Version 2012. [retrieved: Mar. 2026]. [Online]. Available: https://www.adobe.com/devnet/xmp.html

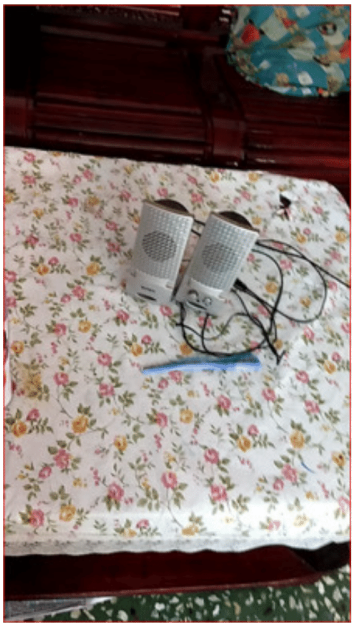
[34] A. Barbu, D. Mayo, J. Alverio, W. Luo, C. Wang, D. Gutfreund, et al., "Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models," *Advances in Neural Information Processing Systems*, vol. 32, pp. 9448–9458, 2019.

[35] ImageMagick Studio LLC, *Imagemagick*, version 7.0.10, Jan. 4, 2023. [retrieved: Mar. 2026]. [Online]. Available: https://imagemagick.org

[36] digiKam Team, *Digikam: Professional photo management*, 2026. [retrieved: Mar. 2026]. [Online]. Available: https://www.digikam.org/

[37] A. Tran and contributors, *Immich: High-performance self-hosted photo and video management*, 2026. [retrieved: Mar. 2026]. [Online]. Available: https://immich.app/

[38] M. Neubert and contributors, *Photoprism: AI-powered photos app for the decentralized web*, 2026. [retrieved: Mar. 2026]. [Online]. Available: https://www.photoprism.app/



```
Profile-iptc: 334 bytes
  unknown[2,0]:
  Keyword[2,25]: cassette
  Keyword[2,25]: cassette player
  Keyword[2,25]: dial telephone, dial phone
  Keyword[2,25]: electric fan, blower
  Keyword[2,25]: handkerchief, hankie, hanky, hankey
  Keyword[2,25]: iPod
  Keyword[2,25]: loudspeaker, speaker, speaker unit, loudspeaker system, speaker system
  Keyword[2,25]: modem
  Keyword[2,25]: mosquito net
  Keyword[2,25]: notebook, notebook computer
  Keyword[2,25]: radio, wireless
  Keyword[2,25]: sewing machine
  Keyword[2,25]: tape player
```

Figure 4. An ObjectNet [34] image of speakers tagged by `rtag` using the default ImageNet-1k label set (threshold $\tau = 0.25$). The tool correctly assigns the combined label "loudspeaker, speaker, speaker unit" as a top tag. Semantically related tags such as "radio" and "cassette player" reflect CLIP's learned associations between audio equipment concepts.

```
Profile-iptc: 280 bytes
  unknown[2,0]:
  Keyword[2,25]: bathtub, bathing tub, bath, tub
  Keyword[2,25]: dishwasher, dish washer, dishwashing machine
  Keyword[2,25]: maraca
  Keyword[2,25]: plunger, plumber's helper",
  Keyword[2,25]: screwdriver
  Keyword[2,25]: soap dispenser
  Keyword[2,25]: toilet seat
  Keyword[2,25]: tub, vat
  Keyword[2,25]: washbasin, handbasin, washbowl, lavabo, wash-hand basin
  Keyword[2,25]: butternut squash
```

Figure 5. An ObjectNet [34] image of soap tagged by `rtag` using the default ImageNet-1k label set (threshold $\tau = 0.25$). The presence of tags like "bathtub" and "dishwasher" alongside "soap dispenser" illustrates CLIP's contextual associations—soap frequently co-occurs with bathroom and cleaning contexts in the training data.

```
Profile-iptc: 270 bytes
  unknown[2,0]:
  Keyword[2,25]: binoculars, field glasses, opera glasses
  Keyword[2,25]: bolo tie, bolo, bola tie, bola
  Keyword[2,25]: espresso maker
  Keyword[2,25]: holster
  Keyword[2,25]: lens cap, lens cover
  Keyword[2,25]: loupe, jeweler's loupe",
  Keyword[2,25]: Polaroid camera, Polaroid Land camera
  Keyword[2,25]: reflex camera
  Keyword[2,25]: slot, one-armed bandit
  Keyword[2,25]: tripod
```

Figure 6. An ObjectNet [34] image of a camera tagged by `rtag` using the default ImageNet-1k label set (threshold $\tau = 0.25$). Tags such as "binoculars" and "opera glasses" appear alongside "reflex camera" and "Polaroid camera," showing CLIP's semantic clustering of optical devices that share visual features such as lenses and compact form factors.