Automatic Semantic Image Tagging at Scale: AI-Powered Command-Line Tool Based on CLIP

Yurij Mikhalevich
Lightning AI
Dubai, United Arab Emirates
email: yurij@mikhalevi.ch

***Abstract*—This study introduces an efficient and scalable solution for image tagging through the utilization of OpenAI's Contrastive Language-Image Pre-Training (CLIP) model. It features a user-friendly Command-Line Interface (CLI) and leverages a caching mechanism for image features introduced by the rclip tool, which is powered by the SQLite 3 Relational DataBase Management System (RDBMS). This enables effective tagging of image catalogs already indexed by rclip. The performance of this method was tested on the ObjectNet dataset, achieving 27.22% accuracy for the top-1 prediction and 50.42% for the top-5 predictions. A scalability analysis revealed that both the indexing and tagging processes increase linearly with the image count. For instance, processing 50,273 unindexed images on an Apple M1 Max CPU was 31.66 times longer than tagging 965 unindexed images, and handling 50,273 indexed images was 31.15 times longer than 965 indexed images. This strategy is particularly beneficial for sectors that handle large amounts of visual content and require external processing tools, such as the media and entertainment, security, and healthcare industries.**

Keywords-image tagging; image indexing; photo management; computer vision

I. Introduction

The unveiling of the 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 [1]. 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.

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. 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` [2]. While `rclip` has broad applications, there are a lot of other use cases where it's important to be able to tag images to enable their manual cataloging and search using 3rd-party software. This makes an efficient image tagging solution increasingly necessary and relevant.

In Section 2, the paper explores related works in the field of image tagging. Section 3 describes the proposed method for image tagging, which is built on top of `rclip` powered by CLIP. Section 4 presents the implementation details of the proposed method. Section 5 discusses the performance of the proposed method. Section 6 presents the tagging quality measurement results of the proposed method. Section 6 discusses the implementation and future plans. Finally, Section 8 concludes the paper and discusses future work.

II. Related Works

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.

One of the earliest attempts to bridge the gap between vision and language is the work by Socher et al. [3], 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. [4], with the introduction of AlexNet, which demonstrated the potential of deep learning in computer vision tasks. This work was instrumental in showing how deep learning could be applied to achieve significant improvements in image classification accuracy.

Following this, the development of the ResNet model by He et al. [5] represented another major step forward, introducing the concept of residual learning to enable the training of much deeper networks. 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. [6], which introduced a new method for pre-training language representations. 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. [7], which employed separate pathways for processing visual and textual information before integrating them, demonstrating improved performance on tasks requiring both visual and textual understanding.

The development of CLIP by Radford et al. [1] stands as a significant milestone, combining the learnings from both fields to create a model that learns visual concepts from natural language descriptions. This model's ability to perform zero-shot image classification and tagging by leveraging a vast dataset of image-text pairs collected from the internet has opened new avenues for research and application.

In the domain of image tagging specifically, prior works have explored various approaches for automating the process. Weyand et al. [8] introduced a method for geolocating images using a deep neural network, demonstrating the potential of using image content to infer metadata. This work highlighted the capability of deep learning models to extract and infer contextual information from images beyond simple object recognition.

The advancement of deep learning in image tagging has been notably propelled by research focusing on the intricacies of multi-label classification. One important study by Wang et al. [9] introduced a CNN-RNN framework that enhances the accuracy of tagging by leveraging the correlations between labels in a multi-label setting. Their method demonstrates how models can be effectively trained to predict multiple relevant tags for a single image, considering the dependencies among these tags to improve the precision of the tagging process. This approach underlines the complexity of image tagging as a multifaceted problem, benefiting from models that capture and utilize the nuanced interrelations among tags to produce more accurate and contextually relevant annotations. The progress highlighted by Wang et al. underscores the transformative potential of deep learning techniques in the domain of semantic image tagging, offering insights into the development of more sophisticated and efficient tagging solutions.

Subsequently, the integration of attention mechanisms, as seen in models like the Transformer [10], has been adapted to multimodal learning, enhancing the ability of models to focus on relevant parts of the input when generating tags or descriptions. This approach has notably improved the precision of image tagging, facilitating more nuanced interpretation of visual and textual data. Among such models, ViLBERT [7] and CLIP [1] exemplify the advancements in multimodal learning. ViLBERT utilizes a co-attentional layer to process visual and textual inputs in parallel, enabling improved performance on tasks like visual question answering and visual commonsense reasoning. Similarly, CLIP leverages transformer architectures to learn visual concepts from natural language supervision, achieving a profound understanding of images through textual descriptions. These models underscore the significant role of transformers in bridging the semantic gap between different types of data, thereby enhancing AI's capability in multimodal understanding and interaction.

The exploration of CLIP [1] and its application to image tagging, as discussed in this paper, builds upon these foundational works. By leveraging CLIP's zero-shot learning capabilities, we propose a novel method for semantic image tagging that can adapt to a wide range of tags and images, addressing the challenges posed by the ever-increasing volume of digital images.

III. Method

CLIP's text transformer allows us to convert a text label (tag) into a n-dimensional vector (where n differs depending on the CLIP model used). With CLIP's image transformer, it is possible to convert an image to a n-dimensional vector. Then, we can calculate the dot product (Equation 1) of the normalized image vector and each of the normalized tag vectors. After this, we pick tags with the dot product higher than the configured threshold; this gets us the tags that match the image the most.

$$\boldsymbol{a} \cdot \boldsymbol{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n \tag{1}$$

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. This is particularly challenging if there is no access to GPUs to run the CLIP model. In order to address this scalability issue, this article proposes two optimizations:

- pre-compute tag feature vectors and reuse them when processing all of the images;
- utilize the `rclip`'s [11] cache layer to avoid computing feature vectors for images already processed by `rclip` and allow us to retag the images quickly.

The cache layer implemented by `rclip` stores the computed image vectors on disk after the initial processing of images so that they can be accessed later. We propose to reuse this logic in the image tagging process. This approach reduces the computational overhead associated with image tagging and improves the response time for users. This is particularly useful when the user needs to retag the images with a different set of tags or when the user already has their images indexed by `rclip`.

`rclip` also handles adding new images to the cache to avoid recomputing the whole cache when the image catalog is updated.

Once we obtain the tags, we store them in the image metadata. This allows other software to utilize the tags. For example, the user can use the tags to search for images using the operating system's file manager or a third-party image management software.

IV. Implementation

The method described above is implemented in the Python utility called `rtag` [12]. `rtag` provides an easy-to-use CLI interface 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 should open the terminal, navigate to the directory containing the files that they want to tag, type `rtag`, and hit "Enter."

The solution uses the `rclip` library [11] to compute the feature vectors. `rclip` uses `ViT-B/32` version of the OpenAI's CLIP model [1]. The tool writes the computed tags to the image IPTC metadata [13] using the IPTCInfo3 Python library [14]. A simplified version of the image processing loop source code is shown in Figure 1.

By default, `rtag` uses the list of the `ImageNet-1k` [15] labels as tags. The user can provide their own list of tags by specifying the `--tags-filepath` argument when running `rtag`. For example, `rtag --tags-filepath ./path/to/tags.txt`. `./path/to/tags.txt` should contain a newline-separated list of tags. Figure 2 shows how the tags are loaded from the file.

`rtag` supports two modes of operation: `append` and `overwrite`. In the `append` mode, the tool appends the computed tags to the existing tags in the image metadata. In the `overwrite` mode, the tool overwrites the existing tags with the computed tags. The user can specify the mode by providing the `--mode` argument when running `rtag`. For example, `rtag --mode append`. Figure 1 shows how setting the mode changes the tool behavior.

`rtag` also supports the `--threshold` argument, which allows the user to specify the similarity threshold. The user can use this argument to control the number of tags that are written to the image metadata. The default value is 0.25, which produces optimal results.

Another useful `rtag` feature is the ability to do a dry run. The user can use the `--dry-run` argument to see what tags would be written to the image metadata without actually writing them. This is particularly useful when the user wants to see how the tool behaves with a specific set of tags and images and wants to adjust the similarity threshold.

The `rtag` source code is published on GitHub under the MIT license [12].

V. Performance

`rtag` was benchmarked on the `ObjectNet` 50,273 images dataset [16], on the `Apple M1 Max` CPU. Table I shows how `rtag` performs when tagging previously indexed and unindexed images. As you can see from the table, running `rtag` on 50,273 unindexed images took 6m21.250s, while tagging 965 unindexed images took 0m12.040s. A similar linear scaling relationship is preserved when running `rtag` on indexed images. Tagging 50,273 indexed images took 4m44.790s while tagging 965 indexed images took 0m09.140s.

```
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 1. 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 2. Tags loading

It should be noted that before running the benchmarks, the dataset was resized to 224px by smaller dimension and converted to JPG using ImageMagick [17].

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

As Table II shows, `rtag` achieves 27.22% top-1 accuracy and 50.42% top-5 accuracy rate on the `ObjectNet` [16] 50,273 images dataset. The `ObjectNet` dataset was chosen for the quality measurement because it is a diverse and challenging dataset that contains images of objects in their natural environments.

The table also shows that `rtag` shows better results when we compute label feature vectors from a `photo of [tag]` string instead of using the `[tag]` as is. Further research is needed to understand whether it is a good idea to default to using `photo of [tag]` in `rtag`.

The benchmark source code is available in the project repository on GitHub [12].

To get a better understanding of `rtag`'s performance, see Figure 3, Figure 4, and Figure 5 showing a few images from the `ObjectNet` dataset tagged by `rtag`.

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

TABLE II
`rtag` tagging quality

Model	Top-1 accuracy	Top-5 accuracy
`ObjectNet`	25.28%	48.05%
`ObjectNet` photo of [tag]	27.22%	50.42%

VII. Discussion

ObjectNet is a large, real-world, challenging test set for object recognition with control where object backgrounds, rotations, and imaging viewpoints are random. Despite this making the test set perfect for benchmarking `rtag`, it would be interesting to see how `rtag` performs on other datasets.

The development of the command-line tool, `rtag`, which is based on `rclip` and employs OpenAI's CLIP model, has resulted in an efficient and user-friendly utility for image tagging. However, there are still possibilities for further improvements.

Future plans include:

- to improve the tool's tagging performance by processing multiple images in parallel;
- to add support for writing tags into the XMP sidecar files and leaving the image files intact;
- to improve the tool's performance by preventing it from re-indexing files when they are renamed;
- to assess the tool's performance on other datasets;
- to enrich `rtag`'s tagging capabilities by utilizing metadata, which already exists within the images, e.g., existing GPS coordinates can be used to tag images with human-readable location-based tags;
- to do an in-depth comparison of `rtag` with other existing image tagging tools.

Even with its current performance and capabilities, `rtag` is an incredibly valuable tool.

VIII. Conclusion

In summary, this paper introduces a practical and scalable method of tagging images of any kind with any set of labels. The method is based on the CLIP model. The approach has demonstrated impressive results on the ObjectNet dataset, indicating its potential applicability to a wide range of industries reliant on visual data and using image processing tools requiring images to be tagged.

References

[1] A. Radford *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*, PMLR, 2021, pp. 8748–8763.

[2] 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.

[3] 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.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[5] 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.

[6] 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.

[7] J. Lu, D. Batra, D. Parikh, and S. Lee, “Vilbert: Pre-training 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, pp. 13–23.

[8] 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.

[9] J. Wang *et al.*, “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.

[10] A. Vaswani *et al.*, “Attention is all you need,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 6000–6010.

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

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

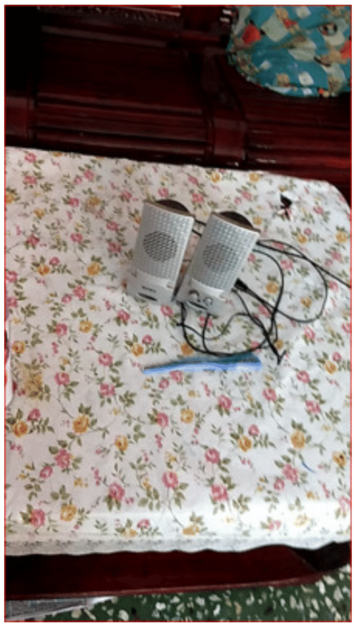
[13] *Iptc photo metadata standard*, Jan. 2023. [Online]. Available: https://www.iptc.org/standards/photo-metadata/iptc-standard/ [retrieved: Feb. 2024].

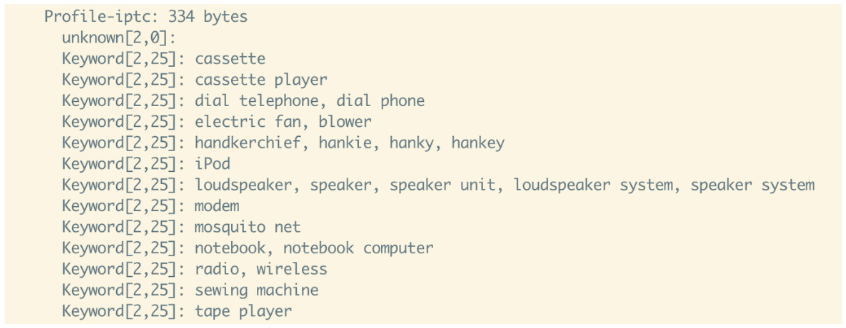
[14] T. Gulacsi and J. Campbell, *IPTCInfo3*, version 2.1.4, 2019. [Online]. Available: https://github.com/james-see/iptcinfo3 [retrieved: Jan. 2024].

[15] O. Russakovsky *et al.*, “ImageNet Large Scale Visual Recognition Challenge,” *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[16] A. Barbu *et al.*, “Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models,” vol. 32, 2019, pp. 9448–9458.

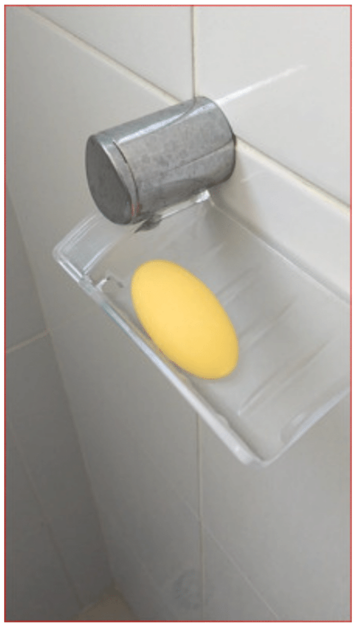
[17] ImageMagick Studio LLC, *Imagemagick*, version 7.0.10, Jan. 4, 2023. [Online]. Available: https://imagemagick.org [retrieved: Feb. 2024].

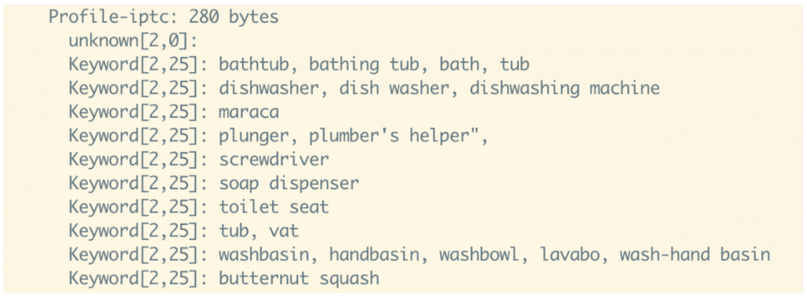




```
  Profile-iptc: 334 bytes
    unknown[2,0]:
    Keyword[2,25]: cassette
    Keyword[2,25]: cassette player
    Keyword[2,25]: dial telephone, dial phone
    Keyword[2,25]: electric fan, blower
    Keyword[2,25]: handkerchief, hankie, hanky, hankey
    Keyword[2,25]: iPod
    Keyword[2,25]: loudspeaker, 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
```

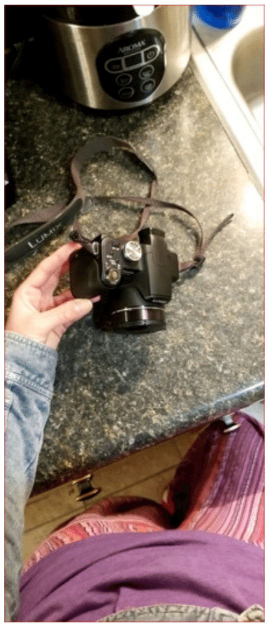
Figure 3. An image of speakers tagged by rtag

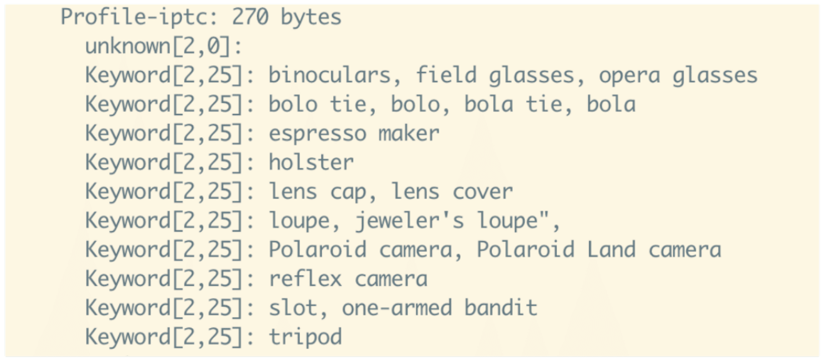




```
  Profile-iptc: 280 bytes
    unknown[2,0]:
    Keyword[2,25]: bathtub, bathing tub, bath, tub
    Keyword[2,25]: dishwasher, dish washer, dishwashing machine
    Keyword[2,25]: maraca
    Keyword[2,25]: plunger, plumber's helper",
    Keyword[2,25]: screwdriver
    Keyword[2,25]: soap dispenser
    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 4. An image of a soap tagged by rtag





```
    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 5. An image of a camera tagged by rtag