

MASARYK UNIVERSITY
FACULTY OF INFORMATICS



Automatic Image Annotation for Microstock Sites

MASTER'S THESIS

Bc. Michal Červeňanský

Brno, Spring 2021

MASARYK UNIVERSITY
FACULTY OF INFORMATICS



Automatic Image Annotation for Microstock Sites

MASTER'S THESIS

Bc. Michal Červeňanský

Brno, Spring 2021

This is where a copy of the official signed thesis assignment and a copy of the Statement of an Author is located in the printed version of the document.

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Bc. Michal Červeňanský

Advisor: RNDr. Petra Budíková, Ph.D.

Acknowledgements

I would like to thank my advisor Petra Budíková for her guidance and endless patience. I sincerely appreciate the time she was willing to invest in this thesis.

My gratitude also belongs to Michal Batko for helping me with technical difficulties.

Abstract

This thesis addresses the issue of image annotation for the microstock industry. It attempts to bridge the gap between a real-life problem of image annotation and the state-of-the-art research of object detection and image classification techniques. Building upon the existing MUFIN Annotation Framework, we develop an annotation pipeline optimized for obtaining keyword annotation for microstock usage. Specifically, we improve the accuracy of MUFIN annotation using high-precision seed keywords acquired from image metadata and state-of-the-art machine learning technologies. The proposed solution is evaluated over real-world microstock data.

Keywords

image annotation, image classification, object detection, microstock, convolutional neural networks, MUFIN Annotation Framework, seed keywords

Contents

1	Introduction	1
2	Microstock Industry	3
2.1	<i>Industry Characteristics</i>	3
2.2	<i>Most Popular Microstocks</i>	3
2.2.1	Shutterstock	4
2.2.2	Adobe Stock	4
2.2.3	iStock	4
2.3	<i>IPTC Metadata and Keywording</i>	4
2.3.1	IPTC Metadata	5
2.3.2	Tips on Keywording Photos	6
2.4	<i>Available Solutions for Generating Keywords</i>	8
2.5	<i>Workflow</i>	10
3	MUFIN Annotation Framework	13
3.1	<i>Overview</i>	13
3.2	<i>ConceptRank</i>	15
3.3	<i>Relevance Feedback</i>	16
4	MUFIN for Microstock Use	19
4.1	<i>Dataset of Hand-annotated Images</i>	19
4.2	<i>MUFIN Performance and Optimization</i>	21
4.3	<i>Resulting Keywords and Analysis of Weaknesses</i>	23
5	Enhancing MUFIN Annotation by Seed Keywords	25
5.1	<i>Problem Analysis</i>	25
5.2	<i>Seed Keywords from Manual Annotation</i>	26
5.3	<i>Seed Keywords from Image Caption</i>	26
5.4	<i>Seed Keywords from Classification</i>	27
5.4.1	Datasets	27
5.4.2	Models	29
5.5	<i>Positive Feedback from Object Detection</i>	31
5.5.1	Datasets	32
5.5.2	Models	37
6	Implementation of Annotation Pipeline	41

6.1	<i>IPTC Data</i>	43
6.2	<i>Object Detection and Box Cropping</i>	44
6.3	<i>Image and Box Classification</i>	44
6.4	<i>MUFIN Annotation and Merging Results</i>	46
6.5	<i>Configuration</i>	47
7	Evaluation	49
7.1	<i>Comparison to the Manual Annotation</i>	49
7.2	<i>Result Analysis</i>	50
7.3	<i>Performance Analysis</i>	56
8	Proposed Workflow	59
9	Conclusion	61
	Bibliography	63

1 Introduction

Technological progress allowed for an enormous increase in the volume of multimedia data like images, videos, and sounds. Everybody is able to photograph and make a video using a mobile phone or a more affordable digital camera. This content is often shared on social media and, after a moment, forever forgotten in the never-ending stream of new content. What if this does not have to be the case? Authors of blogs and articles are looking for photos to illustrate their content every day. This is the business model of *photo stock* websites also known as *photo banks* or *microstock*. Even amateur photographer is able to earn money by uploading his photos.

This means we need to be able to annotate our content efficiently. We are doing this using keywords, English words describing the image. Keywording is a fundamental part of the microstock industry. Keywords not only describe images but, most importantly, are used by potential buyers while searching for the desired image. While keywording is the essential part of preparing photos for microstock, it is also the most time-consuming. Therefore there are multiple tools to help photographers work more effectively. However, these simple web base tools provide only recommendations of keywords and cannot be integrated into the workflow, making keywording a fully automated process.

This thesis aims to develop tools especially for the usage of automatic obtaining keywords for microstock. The main objectives of the thesis are:

- Analyse the performance of the MUFIN Annotation Framework in the use case of providing microstock annotation and expose its weaknesses.
- Propose possibilities of improving MUFIN performance and overcome its deficiencies.
- Implement tool in a form that it will be usable by the microstock community. Analyse and present obtained results.
- Propose effective workflow of preparing batches images for microstock while integrating newly developed tool.

The thesis is organized as follows. In Chapter 2 we introduce the reader to the microstock industry, share our experiences with the most popular photo stock sites as well as explain *IPTC* standard. We finish the chapter by looking into already available tools as well as popular workflow. Chapter 3 introduces the MUFIN annotation framework, a crucial part of the developed pipeline. We give a general overview of the used algorithm as well as some specifics like *ConceptRank* algorithm or the possibility of using *relevance feedback* to improve the annotation. We introduce our dataset of 1,000 hand-annotated images, which was used during experiments in Chapter 4 as well as discuss the relevance of results and optimal settings of the MUFIN annotation framework. In Chapter 5 we analyse the problem of annotation in detail and propose a solution in the form of *seed keywords*. We propose multiple possibilities of getting relevant seed keywords and explain them in detail. Chapter 6 describes the implementation of the annotation pipeline, starting with explaining possibilities of configuring the pipeline and describing individual module and annotation steps. We evaluate the relevance of obtained annotation as well as discuss achieved performance in Chapter 7. Finishing with Chapter 8 we propose recommended effective workflow using our specifically developed pipeline.

2 Microstock Industry

Microstock, also known as photo banks, is an attractive possibility of monetizing photography even for amateurs. This chapter will define the characteristics of the microstock industry, introduce the most popular microstock sites, and discuss current popular workflow and helpful tools.

2.1 Industry Characteristics

Microstock photography is a type of stock photography where amateur and hobbyist photographers submit photos for online distribution at a much lower cost than those sourced from typical stock photo vendors. Because microstock photography is low-cost and generally distributed on a royalty-free basis, it is an economical way for small publications or businesses, particularly online businesses, to add visual website elements. [1].

The Microstock has emerged in the last 15 years spreading around the world due to the thrust of technological change and the spread of the Internet. In the 90s, the big stock photography agencies (i.e. Corbis and Getty Images) had developed a business model based on accumulating stock photos of the highest quality, with the help of professional photographers, who were sending developed negatives from all over the world. The technology to capture digital images of good quality has given access to a growing multitude of non-professionals photographers who are able to shoot photographs of more than acceptable quality [2].

2.2 Most Popular Microstocks

We introduce the three most popular microstock sites. In our experience, these are also the sites where we managed to sell the most of images and therefore provided the biggest profit. Starting with Shutterstock in the first place, Adobe stock in the second, and iStock in third. These sites also provide easy to use user interface and helpful features.

2. MICROSTOCK INDUSTRY

2.2.1 Shutterstock

Shutterstock¹ is one of the most established and most well-paying microstock agencies. It was founded back in 2003, and since then, it has gained 200 million images and 10 million videos. Moreover, what is more important, millions of paying customers [3].

2.2.2 Adobe Stock

Adobe Stock² is microstock that was known as Fotolia before Adobe acquired it in 2015. Adobe inherited 100 million images and more than 10 million clips in addition to graphics and 3D content. The most significant selling point for it is that Adobe Stock is available as part of Creative Cloud in Photoshop, Illustrator, InDesign, and other Adobe products [3].

2.2.3 iStock

iStock³ is a subsidiary of a company Getty Images that was founded back in 1995. Initially being a marketing and corporate media materials supplier, Getty adapted to modern demands in 2006 by acquiring iStock. Now it features more than 200 million assets available for subscription, and it was the first to come up with selling royalty-free photos online [3].

2.3 IPTC Metadata and Keywording

As in every business, also in microstock is the goal to maximize profit while minimizing the cost, in our case, the time complexity of managing and developing the microstock portfolio. While uploading photo to microstock sites you have to provide *image caption*, *category* and a list of *keywords*. While caption and keywords are loaded from *IPTC* metadata of the image, the category is not standardized among microstock sites.

1. <https://www.shutterstock.com/cs/>

2. <https://stock.adobe.com/>

3. <https://www.istockphoto.com/>

2.3.1 IPTC Metadata

IPTC metadata⁴ also known as IPTC headers or Information Interchange Model (IIM), is a file structure and set of metadata attributes that can be applied to text, images, and other media types. It was developed in the early 1990s by the International Press Telecommunications Council (IPTC) to expedite the international exchange of news among newspapers and news agencies [4].

Photographers are used to storing image caption and keywords directly in IPTC metadata of *jpeg* images. This enables them to do the process of annotating the images only once, and the microstock sites will prefill the upload form using these metadata.

The image caption is a sentence describing an image with a maximum length of usually 200 characters. A more critical part of annotation is keywords. Potential buyers use keywords to search for the desired image. If our image contains inputted keywords, it should be presented to the potential buyer. The algorithm of this search is not described and might be different for each microstock site. In our experiences, microstock sites accept only direct hits of keywords and do not implement any lemmatization or stemming, described later in Section 5.3. Some microstock sites support other languages like English and are able to translate keywords and caption. However, this translation leaves a lot to be desired, and we strongly recommend always using English. It is best to use as many keywords as possible, with the upper limit usually being 50. In Figure 2.1 we can observe which keywords were used to buy an image.

4. <https://iptc.org/std/photometadata/specification/IPTC-PhotoMetadata>

2. MICROSTOCK INDUSTRY

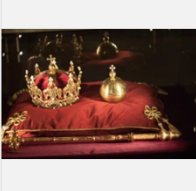

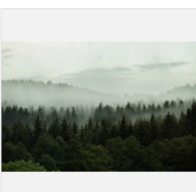
	Image Details	Total Downloads
	<p>ID: 537734683 Luxury crown jewelry on red satin, castle, Slovakia</p> <p>Downloads per Keyword: crown (40.7%), king (18.1%), monarchy (9.6%)</p> <p>Show more +</p>	241
	<p>ID: 1169394439 Goose bumps of young girl getting ready for hike.</p> <p>Downloads per Keyword: goosebump (17.1%), goose (13.4%), bump (13.4%)</p> <p>Show more +</p>	122
	<p>ID: 1132555742 Beautiful foggy forest in the heart of Czech republic</p> <p>Downloads per Keyword: forest (20.8%), pine (20.8%), hill (8.3%)</p> <p>Show more +</p>	73

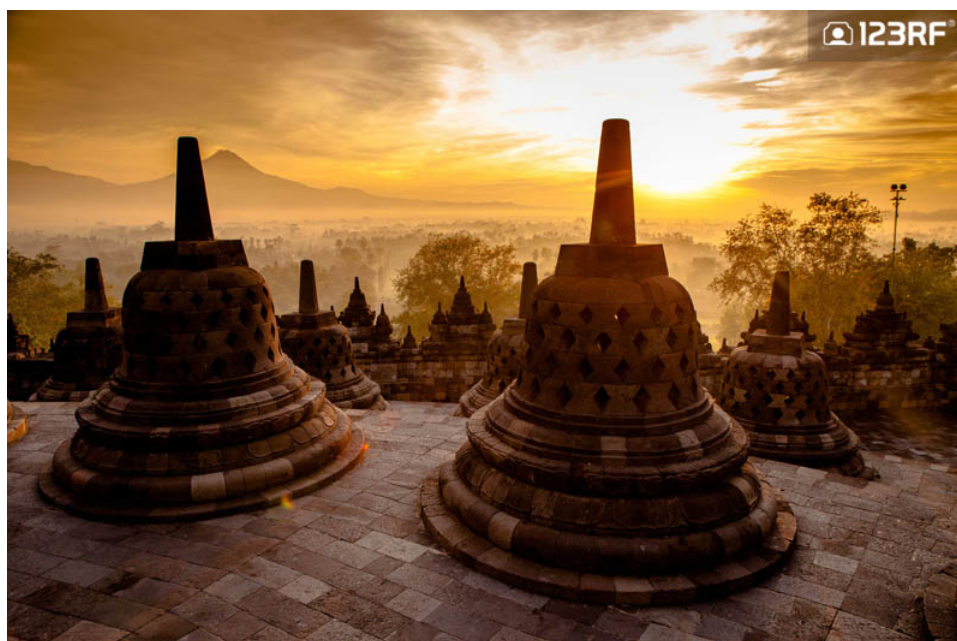
Figure 2.1: Shutterstock top performers feature shows which keyword buyers used to search for desired image.

2.3.2 Tips on Keywording Photos

Proper keywording is an important factor as it will determine whether or not users can easily find the content. Here are the best practices for extracting the right keywords to describe the content. Pay attention to the 4 basic questions – What, Who, When and Where? We will use image in 2.2 as an example. Proposed keywording process is shown in Table 2.1 [5].

Table 2.1: Example of obtaining keywords by answering questions [5].

What do you see? What is the theme?	temple, trees, mountain, stupa, sun, sunrise, sunset, monument, travel, destination
Who is on the image? How does the content make you feel?	nobody, majestic, captivating, awe, scenic
When was the image taken? What is the occasion?	day, evening, summer, sunrise, sunset
Where does the scene take place?	Indonesia, Borobudur, outdoors, landscape



Description	Borobudur temple at sunrise in Yogyakarta, Indonesia
Keywords	ancient, amazing, asian, borobudur, buddhist, candi, candi borobudur, design, destination, dome, faith, holy place, indonesia, java island, landscape, magnificent, monument, morning, place of worship, rock, sanctuary, shrine, sightseeing, statue, stone, structure, stunning, stupa, sun, sunrays, sunrise, sunshine, temple, tourism, tourist attraction, travel, travel destination, worship, yogyakarta

Figure 2.2: Example of photo stock image annotated using description and caption. Image is courtesy of [Phongphon Sutantayawalee](#) [5].

2.4 Available Solutions for Generating Keywords

Photographers are able to use various tools to obtain keywords without having to write them down by hand. These tools are based on the visual similarity of images. This similarity can be defined by the user selecting images similar to his, Figure 2.3, or automatically comparing visual descriptors, Figure 2.3. This will be further discussed in Chapter 3.

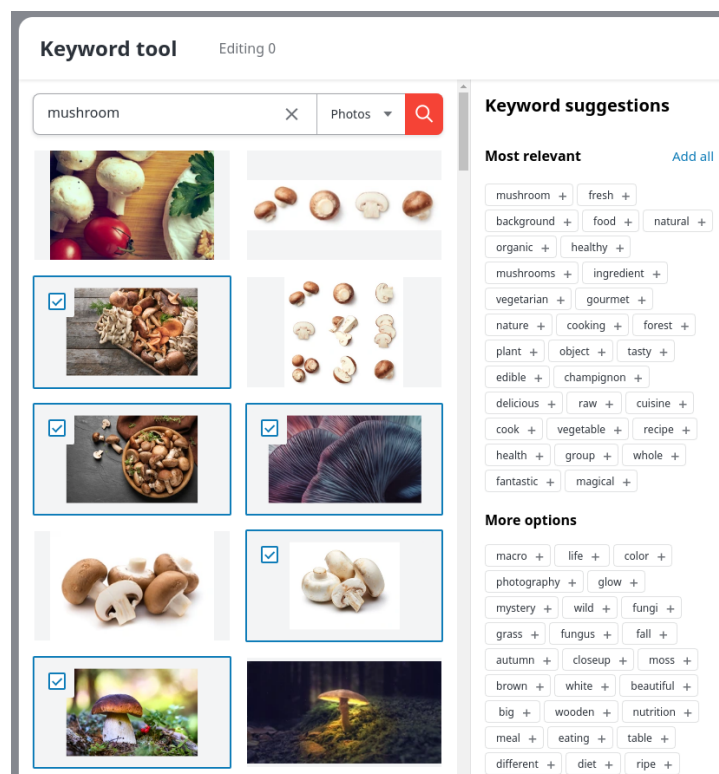
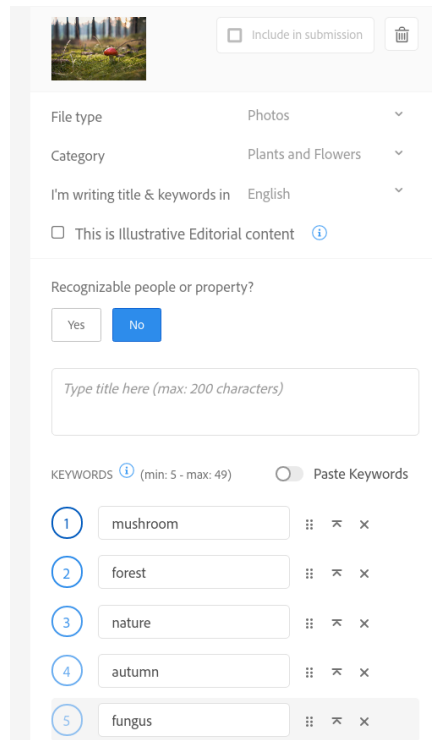


Figure 2.3: Shutterstock Keywording tool suggest user images based on inputted keyword and allows to select images similar to users and then suggest him keywords obtained from selected images.



The screenshot shows the Adobe Stock submission interface for an image. At the top, there is a thumbnail of a forest scene with mushrooms and a red flower. To the right of the thumbnail are two buttons: "Include in submission" and a trash icon. Below the thumbnail, the form fields are as follows:

- File type:** Photos (dropdown menu)
- Category:** Plants and Flowers (dropdown menu)
- I'm writing title & keywords in:** English (dropdown menu)
- This is Illustrative Editorial content (with an information icon)
- Recognizable people or property?** (with "Yes" and "No" buttons, "No" is selected)
- Title:** A text input field with the placeholder "Type title here (max: 200 characters)".
- KEYWORDS:** A section with a "Paste Keywords" toggle (disabled) and a list of 5 auto-filled keywords: "mushroom", "forest", "nature", "autumn", and "fungus". Each keyword is in a numbered box with edit and delete icons.

Figure 2.4: Adobe Stock features a brand new auto filling of image category and up to 25 keywords. Implementation details of this annotation tool were not made public.

There are many tools similar to Shutterstock Keywording tool such as Mykeyworder⁵ and popular Microstock Keyword Tool⁶, these are not connected to any microstock. Tool introduced in Adobe Stock user interface shows similarities in resulting annotation to *MUFIN Annotation Framework*. However, all introduced tools, except the tool by Adobe Stock, require user interaction to select similar images and choose from suggested keywords. Tool by Adobe Stock provides keywords automatically, but it does not provide more than 25 keywords. There is no API available, so if the user wants to use this annotation on other microstocks, he needs to copy keywords into IPTC metadata separately for each image.

5. <https://www.mykeyworder.com/>

6. <https://microstockgroup.com/tools/keyword.php>

2.5 Workflow

Current workflow popular among photographers starts with annotating images directly into IPTC metadata of the image. Users on Windows platform can use:

- Windows explorer directly, Figure 2.5
- Adobe products such as Lightroom, Figure 2.6
- PhotoStock⁷ developed by Czech microstock community
- Exif Pilot⁸ and many others

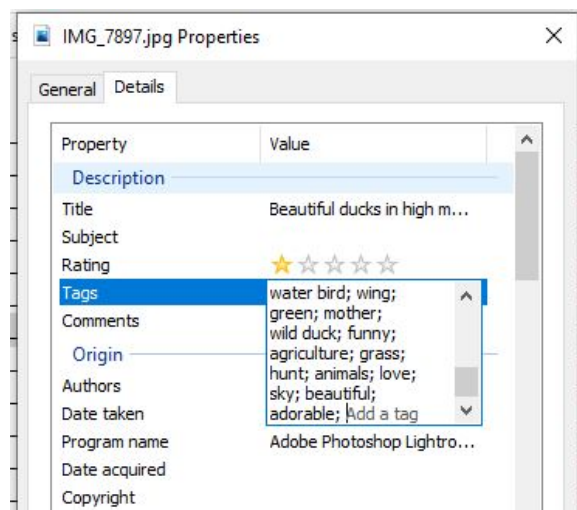


Figure 2.5: IPTC is part of image properties shown by Windows explorer.

7. <http://www.sovpag.com/index.php?language=2&category=24&article=13>

8. <https://www.colorpilot.com/exif.html>

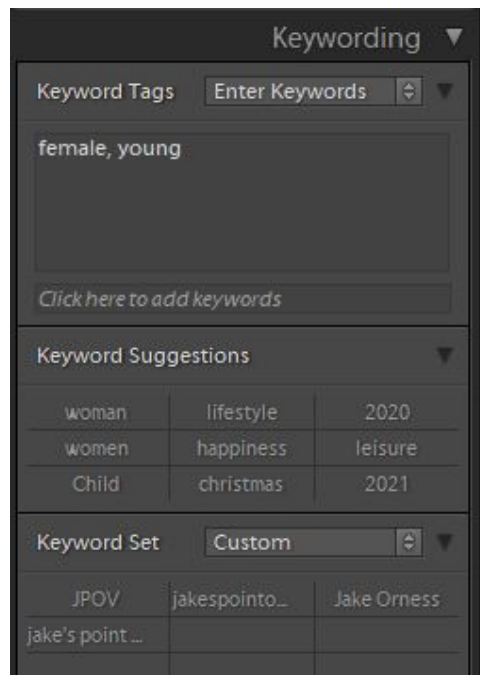


Figure 2.6: Adobe Lightroom is considered as industry standard image editor, it also supports IPTC.

2. MICROSTOCK INDUSTRY

Linux users are very limited in available tools. We recommend XnView MP⁹, Figure 2.7

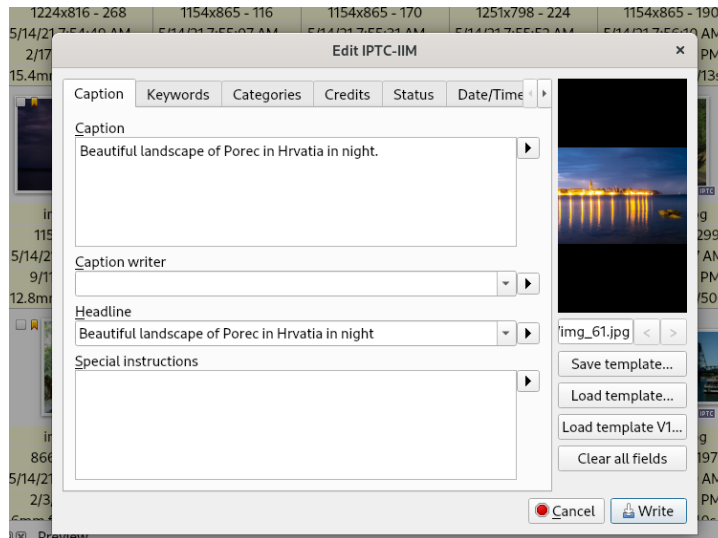


Figure 2.7: XnViewMP is handy and multi-platform image viewer.

The next step is annotating the image by writing a caption and up to 50 keywords. For obtaining keywords, it is possible to use one of the tools described in Section 2.4. With caption and keywords stored in image IPTC metadata, we can either upload the image to the microstock site using *FTP* – File Transfer Protocol, recommended application is *FileZilla*¹⁰ or upload the image directly using the user interface of the microstock website. The last step is completing the upload form and waiting for approval of the image.

9. <https://www.xnview.com/en/xnviewmp/>

10. <https://filezilla-project.org/>

3 MUFIN Annotation Framework

Multimedia information is becoming ubiquitous, and automated annotation tools are desired in many situations. Even though the annotation task details may differ for individual use-cases, the basic structure of the software solutions is usually very similar. Following this observation, in this chapter, we will closely look at the MUFIN Annotation Framework [6, 7], which is the backbone of our image annotation application. We will start with the description of the MUFIN Annotation Framework and then look more closely into the unique ConceptRank algorithm as well as the possibility to improve annotation further using Relevance feedback.

3.1 Overview

Search-based annotation is one of the possibilities of obtaining relevant annotation. This technique attempts to determine the descriptive keywords by analyzing the annotation of similar already annotated multimedia objects, which are detected by content-based retrieval techniques. The main advantage of Search-based annotation is the ability to work with a large volume of even lower quality or noisy data. One of the main challenges of this approach is the extraction of semantically related keywords from the possibly noisy descriptions of similar objects [8]. Vocabulary size is not limited and is defined by the dataset. In contrast, traditional machine learning models provide high accuracy on a limited number of classes. The classification model would excel in the problem of classifying images into 5 classes. However, with the increasing number of classes, training is more difficult, and accuracy is decreasing. This makes this approach not suitable for annotation using keywords.

MUFIN Annotation Framework supports a wide range of annotation tasks by defining a modular and flexible architecture where individual components can be easily combined and reused. Several search-based algorithms and candidate keywords processing components are currently available within the MUFIN Annotation Framework, as well as a pipelining mechanism that passes a central annotation record object between the components [8].

3. MUFIN ANNOTATION FRAMEWORK

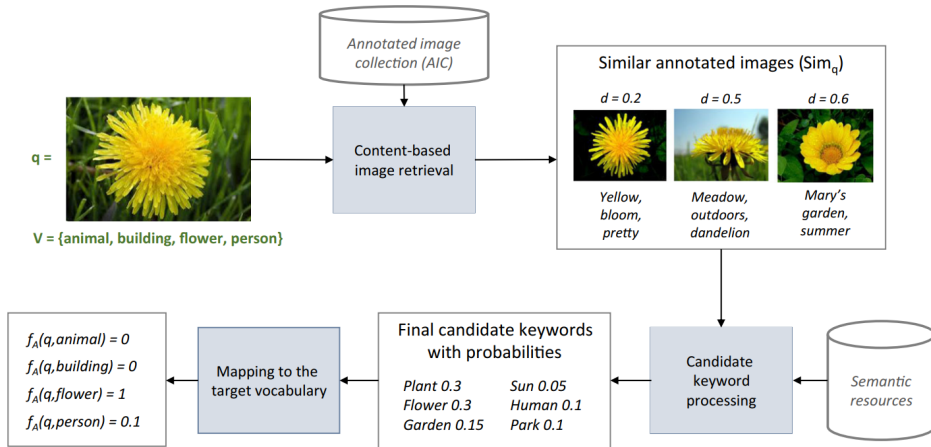


Figure 3.1: General scheme of the search-based annotation [8].

Such a tool can be used by contributors of image-stock sites but also for personal photo tagging. Annotation of any image by either providing its URL or uploading the image file can be done using the MUFIN Image Annotation web application. In the current – prototype – implementation, several testing images are also available [9].

Obtained keywords should serve for general text search, so there is essentially no limitation of the target vocabulary. However, evaluating the effectiveness of the annotation tool is quite tricky with unlimited vocabularies. The basic structure of the MUFIN Image Annotation software reflects the general architecture of any search-based annotation system, as depicted in Figure 3.1 [8].

The MUFIN Annotation Framework utilizes two components while providing search-based annotation. Firstly *content-based image retrieval* retrieves the nearest neighbors of a given image based on visual similarity from *Profiset*, a collection of 20M high-quality images with rich and systematic annotations, which were obtained from Profimedia¹. The image search system enables fast retrieval of similar images from very large collections [10]. The visual similarity is computed using the DeCAF global descriptors defined by the convolutional network weights learned on a set of pre-defined object recognition tasks [11].

1. <http://disa.fi.muni.cz/research-directions/software/profiset/>

The second part is building a *semantic network*, a data structure used for recording all available information about candidate keywords and their relationships. Specifically, a directed multigraph is utilized where weights can be associated with both nodes and edges. Nodes represent candidate semantic concepts. Edges are formed by semantic relationships. The weight of a node represents the current estimate of the semantic concept probability, whereas the weight of edge $U \rightarrow V$ expresses the conditional probability of concept V being relevant given that U is relevant ($P(V|U)$) [8].

3.2 ConceptRank

The ConceptRank component of the MUFIN Annotation Framework encapsulates the semantic analysis. It accepts an annotation record containing a set of candidate keywords and returns an updated record with a new candidate set containing semantic concepts. The component provides generic support for creating the semantic network and the actual algorithm for node probability computation. It accepts an annotation record containing a set of candidate keywords and returns an updated record with a new candidate set containing semantic concepts. The component provides generic support for creating the semantic network and the actual algorithm for node probability computation.

When the semantic network is created, ConceptRank obtains a rich set of candidate semantic concepts linked by relationships. For some of the concepts, there are probability estimates from previous annotation phases, other concepts have zero starting probability. Now, the probabilities of all nodes will be updated, taking into account the initial probabilities and the semantic links which transmit the scores between nodes. Since the network nodes mutually influence each other's probability, a steady-state of this system is required. For this purpose, ConceptRank uses the random walk with restarts (RWR), an algorithm that was successfully used in many similar scenarios, including the famous PageRank [12, 13]. In the basic PageRank, it is assumed that no prior knowledge about the importance of individual pages is available, and thus all nodes enter the computation as equal. However, there are also advanced versions of the algorithm that allow to prioritize some of the nodes. This can be done by biasing the restart

vector u . For instance, in the TrustRank algorithm, the probability of being selected after restart is no longer the same for all nodes. Some trustworthy pages are identified as desirable and more likely to be targeted after the restart [14].

ConceptRank algorithm utilizes the biased random walk with restarts to compute the probability of nodes in the semantic network. Instead of the random user browsing the web, a random association that explores the candidate concept space is modelled [8].

3.3 Relevance Feedback

Relevance feedback is a component of the MUFIN Annotation Framework, which enables the annotation process to run in several iterations. This explicit relevance feedback allows the system to take into account the user's individual needs and preferences. Initially, the input image is annotated automatically by the Annotation Tool. The resulting annotations are provided to the user, who evaluates their relevance. The Annotation process is then restarted, but this time the process also receives the additional information retrieved by the user. Whole process is illustrated on Figure 3.2. This principle is meant to improve the initial Image search phase and therefore rise the relevance level of the processed textual data. The user feedback phase may repeat several times until the user is satisfied with the result [9, 15].

Using the combination of content-based search and text re-ranking, a new result precision of 73.8 % (lower bound) was achieved, which is a 15 % improvement over the initial annotation. This means that at least 4.5 new relevant keywords were identified on average within each 30-keyword annotation. The evaluation was done on a subset of Profiset collection, 160 images as test queries were selected. Out of these, 80 photos were selected from Promedia logs of popular queries. Another 80 were chosen randomly from images sold in a two-year period [15].

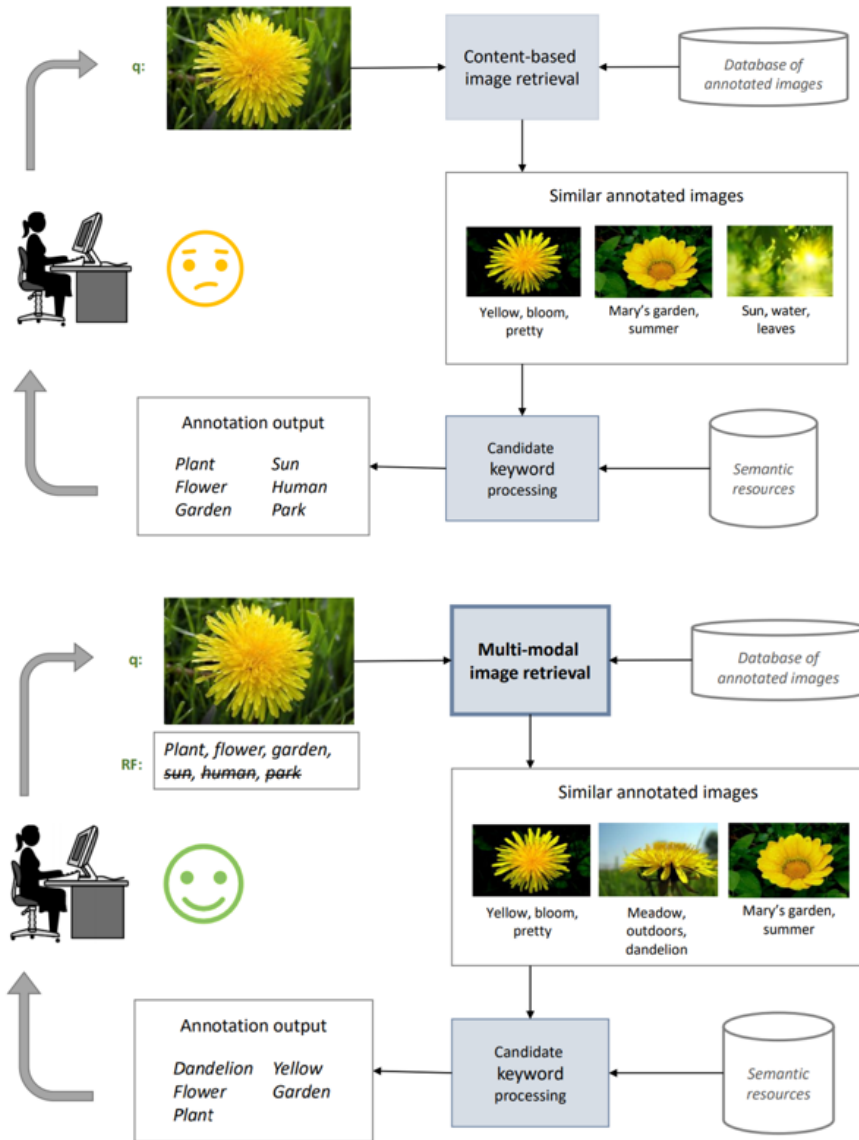


Figure 3.2: Search-based annotation without RF and with RF; the feedback is exploited in the image retrieval phase [15].

4 MUFIN for Microstock Use

We will start this chapter by introducing our dataset of 1,000 hand-annotated images. These images give us baseline annotation, which helps us to compare new results with already existing annotation. Then we are going to analyse the performance of the MUFIN Annotation Framework and try to optimize its parameters. The chapter will end with an in-depth analysis of the strengths and weaknesses of the MUFIN Annotation Framework.

4.1 Dataset of Hand-annotated Images

Thanks to our microstock experiences, we were able to build a dataset of 1,000 hand-annotated images with caption and 50 keywords obtained using the workflow described in Section 2.5. These images are already uploaded to the most popular microstock websites Shutterstock, Adobe Stock, or Getty Images, and many of them generated profit. Based on this we assume, that the hand-annotation using proposed workflow is relevant and a good compromise between time complexity and annotation relevance. This dataset will serve as a baseline for MUFIN evaluation. We assume that the Profiset dataset used in MUFIN was created using a similar workflow, and the used vocabulary should be at least partially similar.

Image in various aspect ratios (4:3, 3:2) in both landscape and portrait orientation. They have been resized to 1 megapixel to make them easier to work with. This resolution is sufficient for our usage due to the fact that the input of both computing visual descriptors in MUFIN, as well as the input of the classification model, is 224x224 pixels. However, *object detection* and *classification* of detected objects discussed later in Chapter 6 might benefit from higher image resolution. Dataset contain various types of images:

- Landscapes, flowers, underwater photos
- Animals
- Objects, food and vehicles
- Human body parts
- Underwater photos

4. MUFIN FOR MICROSTOCK USE

Image annotation is stored directly in IPTC metadata of images. The dataset is available on Google drive¹, hosted directly² or a preview of images squashed to squares with keywords is available in form of static website³ as can be seen on Figure 4.1.

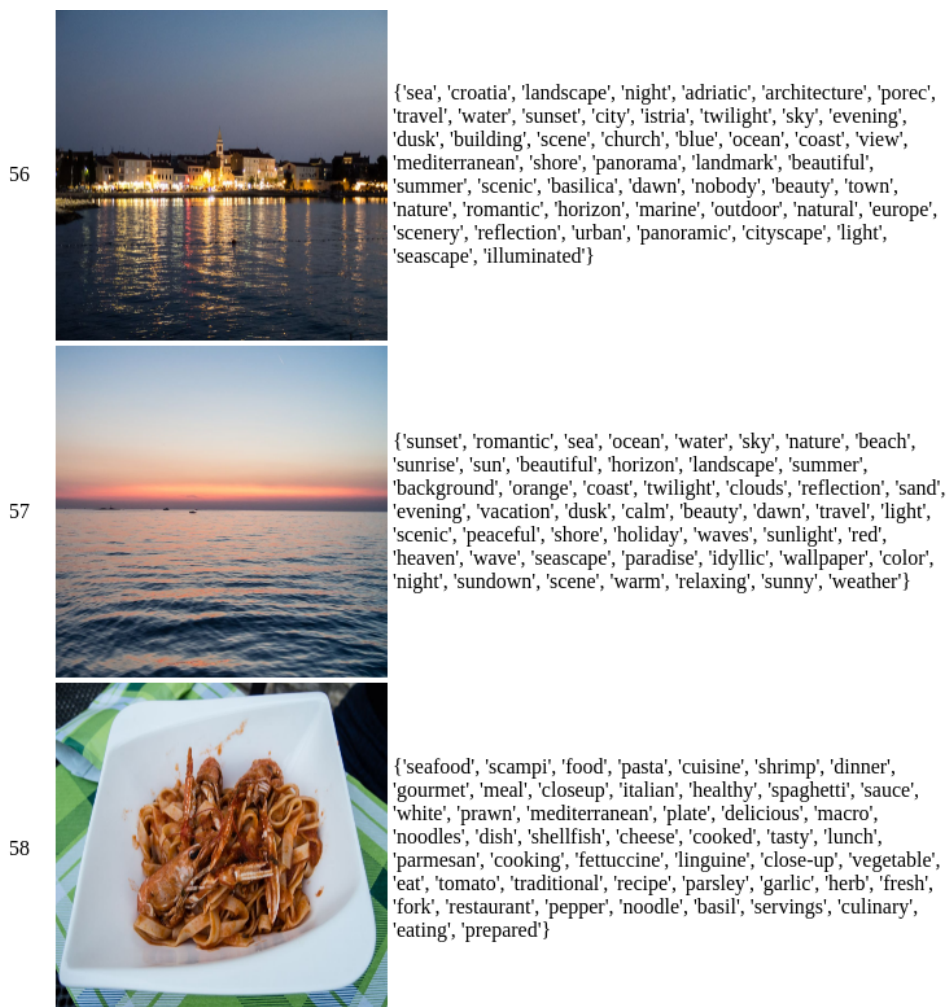


Figure 4.1: Example of images with hand-annotated keywords

1. <https://drive.google.com/drive/u/1/folders/1QspvyBuB-WZIchC8Vp2JBLZLH27HiV2i>
2. http://michal.cervenansky.eu/dt_dataset/img_01.jpg
3. http://michal.cervenansky.eu/dt_dataset/html/my_keywords.html

4.2 MUFIN Performance and Optimization

To evaluate annotation tasks with unlimited vocabularies, we should ideally use a ground truth of all English keywords relevant for a given image. Unfortunately, it is not feasible to collect such a ground truth since there may be literally a thousand words describing each picture [15]. Leveraging the fact that MUFIN uses vocabulary very similar to hand-annotation using the workflow described in Section 2.5 we decided to measure the relevance of annotation by computing the number of overlapping keywords between our hand-annotation and MUFIN results.

MUFIN uses a parameter *similarImages* later discussed in Section 6.5. This parameter determines the number of visually similar images to retrieve from the similarity index to seed the annotation and thus affects the resulting annotation. Our goal was to find the value of the *similarImages* parameter with the biggest average overlap of keywords. We perceived the optimization as a problem of finding local maximum on given domain space of discrete points. We started with a sparse step of 50 as shown in Figure 4.2.

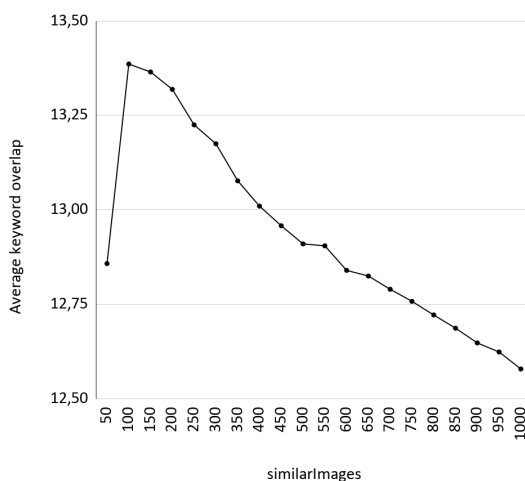


Figure 4.2: Sparse search of *similarImages* parameter with step 50. The biggest average overlap of keywords of 13,386 was found with *similarImages* = 100.

4. MUFIN FOR MICROSTOCK USE

We achieved the best the biggest average overlap of keywords of 13.386 with *similarImages* = 100. That enables us to continue with a more dense search. This time with a step of just 5 starting around *similarImages* = 100. The biggest average overlap of keywords of 13.461 was found with *similarImages* = 115 as can be seen in Figure 4.3. The measure of average keyword overlap gives us only a general idea about MUFIN performance. Due to not perfect quality of hand-annotation and usage of slightly different vocabularies, it is not possible to find the ideal *similarImages* parameter value. However, provided experiments yield a good idea of how to set the *similarImages* to achieve good results.

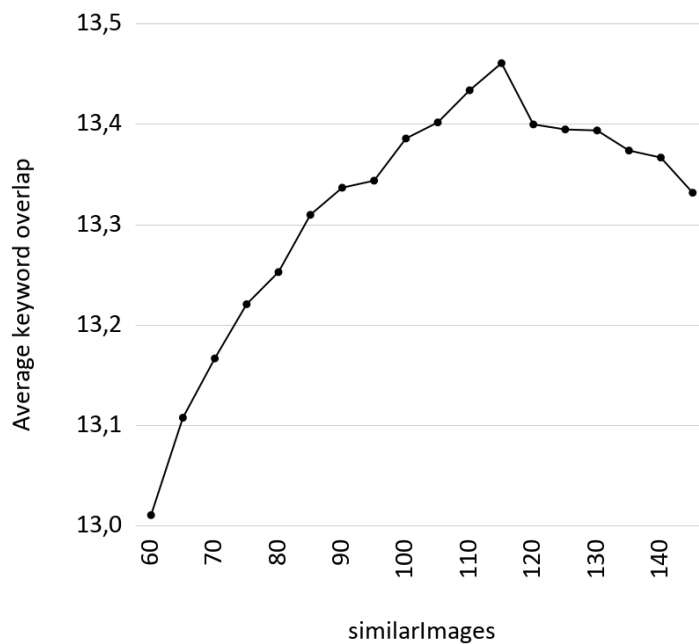


Figure 4.3: Dense search of *similarImages* parameter with step 5. The biggest average overlap of keywords of 13,461 was found with *similarImages* = 115.

4.3 Resulting Keywords and Analysis of Weaknesses

In Section 2.3.2 we defined the character of ideal keywords. Image from our dataset annotated by MUFIN are available in the form of static website⁴. MUFIN does a good job at finding general keywords as well as semantically similar keywords. This can be seen in Figure 4.4, where we can see that MUFIN was able to find relevant keywords that describe the image. We can still see contradictory keywords like "summer" and "winter" or "person" and "nobody". Some of the keywords like "put" or "new" are not relevant to this image. However, in the use case of image keywording for microstock, this is an insignificant problem and can be seen even in manual annotation using the workflow described in Section 2.5.



MUFIN keywords: **fog**, **mountain**, trees, clouds, **forest**, **nature**, formations, **mist**, **view**, seasons, sky, flora, quality, **landscape**, **travel**, put, outdoors, vegetation, **morning**, **scenery**, conifer, nobody, person, daytime, groups, horizontal, **scenic**, region, summer, **cloud**, **winter**, haze, valley, state, day, **weather**, autumn, hills, cover, locations, sunrise, tranquil, color, **fir**, national, british, environment, new, destination, states

Figure 4.4: Example of MUFIN annotation. Keywords highlighted by green color are considered relevant, yellow are partially relevant, and red are irrelevant to the image. Keywords highlighted with bold and underlined font are present in hand-annotation.

4. http://michal.cervenansky.eu/dt_dataset/html/mufin_keywords.html

4. MUFIN FOR MICROSTOCK USE

MUFIN did a good job with a landscape image. Let us take a look at the more difficult image. In Figure 4.5 we can see a night scene in lower quality with two parrots sitting on a sign with a shop in the background. This scene took place in Turkey. We can see that MUFIN is not able to detect what specifically is on the image, and the main focus of the image can be out-weighted by background if the background is the more prominent part of the image. MUFIN was able to output relevant keywords for image background, and we can see that they are all semantically related to travel and shop theme. Two parrots which should be the main focus of the image were ignored. In the result keywords, we can see only very general keyword animal with low rank, which indicates it is more coincidence than related to parrots in the image.



MUFIN keywords: person, continent, market, shop, travel, street, people, outlet, traveller, adult, business, holiday, shopping, buying, group, region, commerce, tourist, tourism, road, leisure, country, food, location, asian, centre, indoors, beach, horizontal, woman, selling, culture, display, show, class, miami, day, objects, island, marketplaces, color, new, district, man, animal, china, city, souvenir, indian, porcelain

Figure 4.5: Example of MUFIN annotation. Keywords highlighted by green color are considered relevant, yellow are partially relevant, and red are irrelevant to the image. Keywords highlighted with bold and underscored font are present in hand-annotation.

5 Enhancing MUFIN Annotation by Seed Keywords

As we saw in Section 4.2, MUFIN provides good results for the usage of finding relevant keywords for images, considering the microstock usage. In this chapter, we introduce the idea of using seed keywords to improve MUFIN performance and explain what the goals of using such a method are. We will look into various possibilities of getting seed keywords in detail, starting with using keywords already written into IPTC metadata of the image, proceeding with parsing keywords from image caption, and finishing with using state-of-the-art neural network models for object detection and image classification.

5.1 Problem Analysis

We believe that the weaknesses of MUFIN annotation could be overcome by giving it a hint about the image. This is possible using relevance feedback described in Chapter 3. However, giving relevance feedback manually would dramatically increase the time and labor complexity of image annotation. Bearing this in mind, we decided for a different approach. We are trying to provide a few relevant keywords already with the initial query before the first annotation is evaluated. We denoted such keywords as seed keywords. These seed keywords help MUFIN to rank keywords that are semantically similar to seed keywords so that these keywords will be a part of the final annotation.

We have thought of multiple approaches how to get seed keywords. The first approach is to ask the user for several keywords respectively use keywords already saved in the IPTC metadata of the image. The next option is to obtain keywords by parsing the image caption. The most interesting approach is to perceive annotation as a computer vision task. Computer vision consists of different problems such as image classification, localization, segmentation, and object detection. We feel that object detection and classification of these detected objects is a promising way of finding relevant seed keywords without needing any input from the user, albeit at the cost of lower precision and computational complexity.

5.2 Seed Keywords from Manual Annotation

Manual annotation of image content is considered the “best case” in terms of accuracy since keywords are selected based on a human determination of the semantic content of images. However, at the same time, it is an effort-intensive and monotonous process [16]. In practice, manual annotation is a compromise of the annotation quality and the time dedicated to this task. We encourage the user to manually add a few important keywords to the IPTC of the image, and these keywords will be used as seed keywords to improve annotation results further.

5.3 Seed Keywords from Image Caption

As already stated in Section 2.5, while uploading an image to the microstock site, the image caption is one of the mandatory fields. The image caption is always manually provided by the user. A good image caption should briefly describe what occurs in the image and what is the subject [17]. We can benefit from this fact by parsing caption into keywords.

In information retrieval system, text normalization is part of text preprocessing. We want to strip off any affixes, a task known as stemming. A further step is to ensure that the resulting form is a known word in a dictionary, a task known as lemmatization [18]. However, we opted not to use neither word lemmatization nor stemming because we do want to change keywords in any way, and removing synonyms also is not our goal.

We applied only the most simplistic approach to parsing image caption. The image caption is loaded directly from the IPTC property of the image file to a string variable and split using space character. Non-alphanumeric characters are removed, and the result is stored in a List structure. From there, we remove 179 *stopwords* defined by NLTK (Natural Language Toolkit) library [19]. Stopwords are English words which does not add much meaning to a sentence. They can safely be ignored without sacrificing the meaning of the sentence [20].

5.4 Seed Keywords from Classification

Image Classification is a fundamental task that attempts to comprehend an entire image as a whole. The goal is to classify the image by assigning it to a single or multiple specific labels. Typically, Image Classification refers to images in which only one object appears and is analysed. In contrast, object detection discussed in Chapter 5.5 involves both classification and localization tasks and is used to analyse more realistic cases in which multiple objects may exist in an image [21].

Convolutional Neural Networks (CNNs) have become the leading solution for image classification tasks in different applications. Although several CNN architectures are available, there is no best architecture regardless the problem at hand. The selection of the most suitable CNN architecture is usually performed by trial and error, which may take much time and computational resources [22]. However, even the most advanced would not be performing without an extensive dataset of training data. We will look in detail to most popular *ImageNet-1K* dataset as well as full *ImageNet-21K* dataset. Thanks to improvements to deep network learning, we are able to use a pre-trained model on such a massive dataset as is the ImageNet-21K. The section will finish with introducing the used CNN model.

5.4.1 Datasets

ImageNet is an image database organized according to the *WordNet* hierarchy, in which each node of the hierarchy is depicted by hundreds and thousands of images [23]. ImageNet-1K serves as the primary dataset for pretraining deep learning models for computer vision tasks. ImageNet-21K dataset, which contains more pictures and classes, is used less frequently for pretraining, mainly due to its complexity and underestimation of its added value compared to standard ImageNet-1K pretraining. ImageNet-1K is a subset of the full ImageNet dataset [24], which consists of 14,197,122 images, divided into 21,841 labels. We shall refer to the full dataset as ImageNet-21K, similar to [25] (although other papers sometimes described it as ImageNet-22K [26]). ImageNet-1K was created by selecting a subset of 1.2M

images from ImageNet-21K that belong to 1,000 mutually exclusive classes. An example of image hierarchy can be seen in Figure 5.1.

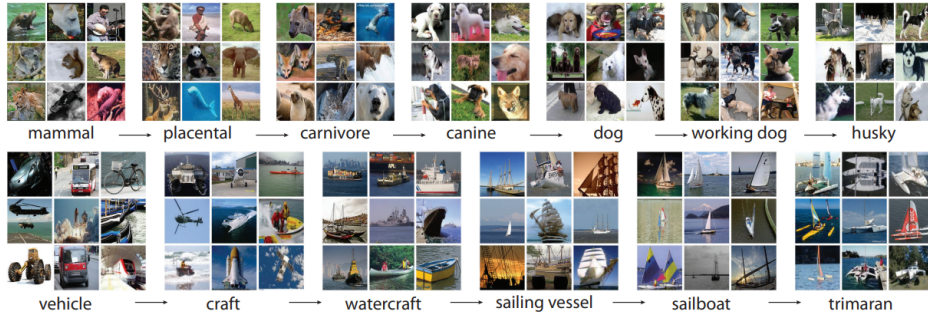


Figure 5.1: A snapshot of two root-to-leaf branches of ImageNet: the top row is from the mammal subtree; the bottom row is from the vehicle subtree. For each synset, 9 randomly sampled images are presented [24].

Even though some previous works showed that pretraining on ImageNet-21K could provide better downstream results for large models [25, 27], pretraining on ImageNet1K remained far more popular. The main reason for this discrepancy is that ImageNet-21K labels are not mutually exclusive – the labels are taken from WordNet [28], where each image is labeled with one label only, not necessarily at the highest possible hierarchy of WordNet semantic tree. For example, the ImageNet-21K dataset contains the labels "chair" and "furniture". A picture with an actual chair can be labeled as "chair", but sometimes be labeled as the semantic parent of "chair", "furniture". Another example can be seen in Figure 5.2. This kind of tagging methodology complicates the training process and makes evaluating models on ImageNet-21K far less accurate. Other challenges of the ImageNet-21K dataset are the lack of official train-validation split, the fact that training is longer than ImageNet-1K and requires highly efficient training schemes, and that the raw dataset is large – 1.2TB [29].



Figure 5.2: Example of inconsistent tagging in ImageNet-21K dataset. Two pictures containing the same animal were labeled differently [29].

5.4.2 Models

Traditional machine learning methods (such as multilayer perception machines, support vector machines, etc.) mostly use shallow structures to deal with a limited number of samples and computing units. When the target objects have rich meanings, the performance and generalization ability of complex classification problems are insufficient. The convolution neural network (CNN) developed in recent years has been widely used in the field of image processing because it is good at dealing with image classification and recognition problems and has brought tremendous improvement in the accuracy of many machine learning tasks. It has become a powerful and universal deep learning model [30].

In the use case of searching for seed keywords, we need our model to be able to classify to very specific classes. That is the reason to use the model pre-trained on ImageNet-21K and not only the ImageNet-1K dataset. However, pretraining such is very difficult and in time of writing this thesis only pre-trained models available on TensorFlowHub¹ are *BiT-M* models with multiple *ResNet* architectures:

- R50x1 – 50 layers, 25.6M parameters
- R50x3 – 50 layers, three times wider than R50x1 architecture
- R101x1 – 101 layers, 44.5M parameters
- R101x3 – 101 layers, three times wider than R101x1 architecture
- R152x4 – 152 layers, four times wider than R152x1 architecture – 928M parameters

The used “Big Transfer” (BiT) model represents a simple paradigm: pre-train on a large supervised source dataset and fine-tune the weights on the target task. In our case, a model with weights fine-tuned for classification was already available. Networks were trained on three different scales of datasets. The largest, BiT-L, is trained on the proprietary JFT-300M dataset [31], which contains 300 M noisily labelled images. For practical usage, the BiT-M model trained on ImageNet-21k was released [25]. This is the model that we also used. As shown in Figure 5.3, we assumed that R152x4 architecture would give us the best results even at the cost longer duration of classification. We have also tried the smallest ResNet-50x1 architecture. We found the results to be worse, however, still acceptable, and usable in time-critical scenarios.

1. <https://tfhub.dev/>

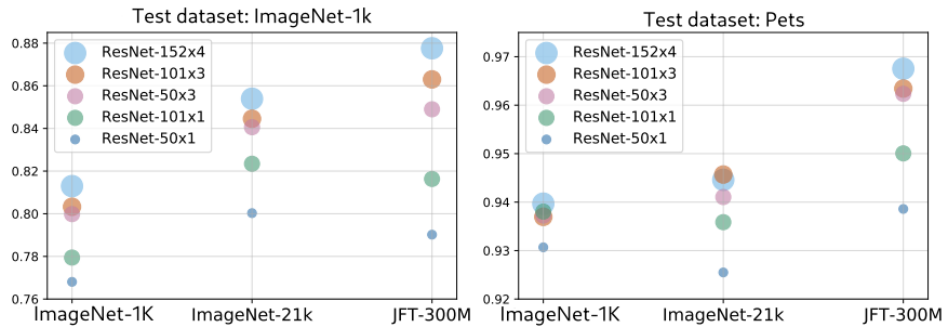


Figure 5.3: Effect of training data (shown on the x-axis) and model size on performance. Note that exclusively using more data or larger models may hurt performance; instead, both need to be increased in tandem [25].

5.5 Positive Feedback from Object Detection

Object recognition is one of the fundamental challenges in computer vision. To gain a complete image understanding, we should not only concentrate on classifying images but also try to precisely estimate the concepts and locations of objects contained in each image [32, 33].

The problem definition of object detection is to determine where objects are located in a given image (object localization) and which category each object belongs to (object classification). This determines the need for a new dataset, which contains information about object location as well as new neural network models.

The pipeline of object detection models can be mainly divided into three stages:

- **Informative region selection.** As different objects may appear in any position of the image and have different aspect ratios or sizes, it is a natural choice to scan the whole image with a multi-scale sliding window. Although this exhaustive strategy can find out all possible positions of the objects, its shortcomings are also evident. Due to a large number of candidate windows, it is computationally expensive and produces too many redundant windows.
- **Feature extraction.** To recognize different objects, we need to extract visual features which can provide a semantic and robust representation. However, due to the diversity of appearances, illumination conditions, and backgrounds, it is challenging to design a robust feature descriptor to manually perfectly describe all kinds of objects.
- **Classification.** Besides, a classifier is needed to distinguish a target object from all the other categories and to make the representations more hierarchical, semantic, and informative for visual recognition [33].

The problem of finding seed keywords is more complex than the basic object detection task. In object detection, we are classifying objects with just several basic classes. The number of classes, as well as performance of object detection, is dependent not only on the model used but mainly on the dataset it was trained on. If the correct class is not available in the dataset, then the object is classified as the most similar class, albeit with a smaller score. This fact can be used in filtering out doubtful results. In the following sections, we will look into available datasets and models more closely.

5.5.1 Datasets

Large scale dataset rich in the number of images, classes, and high number and accuracy of bounding boxes is a prerequisite for building a good performing model for object detection. Building such a

dataset is more difficult than building a classification dataset because it consists not only of labeled images but of precisely drawn boxes around the object, which are labeled. That is a very labor-intensive task. Many datasets are publicly available on [kaggle.com](https://www.kaggle.com/datasets)² or public.roboflow.com³. We will be only focusing on two biggest and most general: The *Microsoft Common Objects in COntext* (MS COCO) dataset and *Open Images V4* dataset by Google.

The Microsoft Common Objects in COntext (MS COCO) dataset contains 91 common object categories, with 82 of them having more than 5,000 labeled instances, Fig. 6. In total, the dataset has 2,500,000 labeled instances in 328,000 images. In contrast to the popular ImageNet dataset [24], COCO has fewer categories but more instances per category.

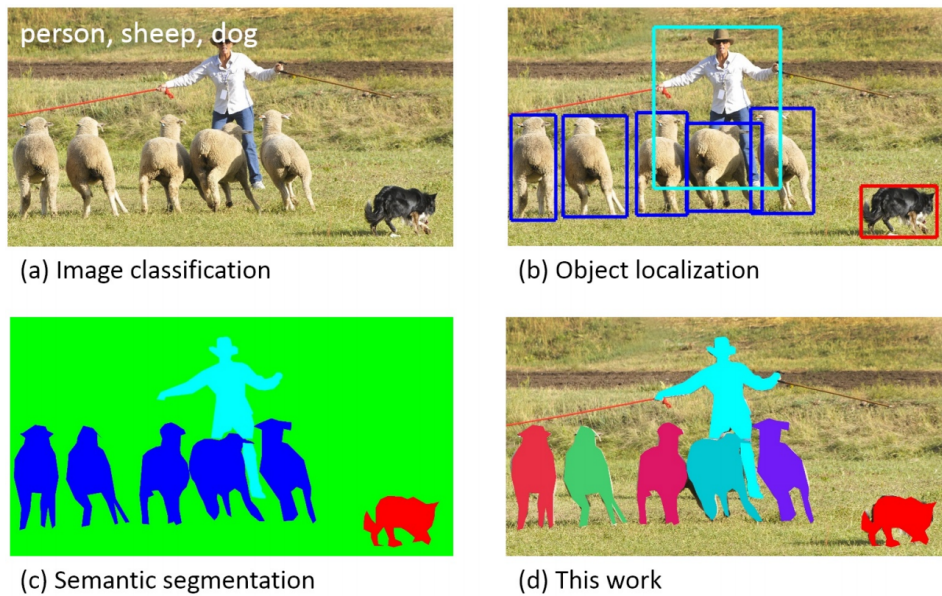


Figure 5.4: While previous object recognition datasets have focused on (a) image classification, (b) object bounding box localization or (c) semantic pixel-level segmentation, COCO focus on (d) segmenting individual object instances [34].

2. <https://www.kaggle.com/datasets>

3. <https://public.roboflow.com/>

5. ENHANCING MUFIN ANNOTATION BY SEED KEYWORDS

This can aid in learning detailed object models capable of precise 2D localization. The dataset is also significantly larger in the number of instances per category than the PASCAL VOC [35] and SUN [36] datasets. Additionally, a critical distinction between the COCO dataset and others is the number of labeled instances per image, which may aid in learning contextual information. MS COCO contains considerably more object instances per image (7.7) as compared to ImageNet (3.0) and PASCAL (2.3). In contrast, the SUN dataset, which contains significant contextual information, has over 17 objects and “stuff” per image but considerably fewer object instances overall [34].

To create a large-scale dataset, a novel pipeline for the gathering was employed with extensive use of Amazon Mechanical Turk. Utilizing over 70,000 worker hours, a vast collection of object instances was gathered, annotated, and organized to drive the advancement of object detection and segmentation algorithms. Emphasis was placed on finding images of objects in natural environments and varied viewpoints [34].

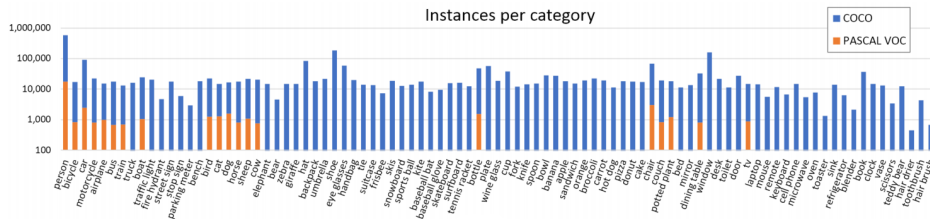


Figure 5.5: Number of annotated instances per category for MS COCO and PASCAL VOC [34].

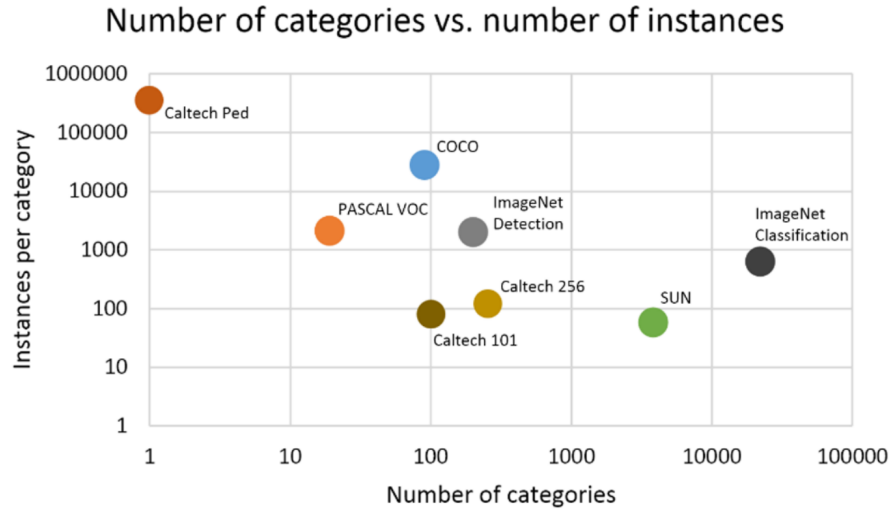


Figure 5.6: Number of categories vs. the number of instances per category for a number of popular object recognition datasets [34].

The number of categories used by the COCO dataset was not sufficient for our usage. We achieved better performance with Open Images V4 dataset by Google. The Open Images V4 offers large scale across several dimensions: 30.1M image-level labels for 19.8k concepts, 15.4M bounding boxes for 600 object classes [37]. Open Images is generally an order of magnitude larger than the other datasets. There are 11 classes in Open Images with more samples than the largest class in COCO. As a particular example, the person class has 257,253 instances in COCO, while Open Images has 3,505,362 instances of the agglomeration of classes referring to person (person, woman, man, girl, boy) [37].

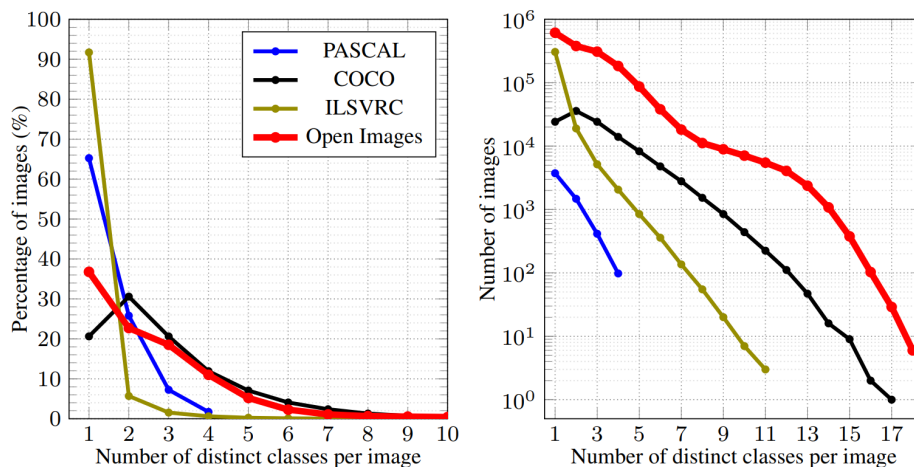


Figure 5.7: Number of distinct classes per image. Normalized (left) and unnormalized (right) histogram of the number of distinct classes per image. Train set for all datasets [37].

Figure 5.7 shows the unnormalized statistics (i.e. with the number of images instead of the percentage). It shows that Open Images has at least one order of magnitude more images than COCO at any point of the curve. For example, Open Images has about 1,000 images with 14 distinct classes, while COCO has 20; ILSVRC has no image with more than 11 classes, and PASCAL has no more than 4. Figure 5.8 displays two images with a large number of classes annotated, to illustrate the variety and granularity that this entails [37].

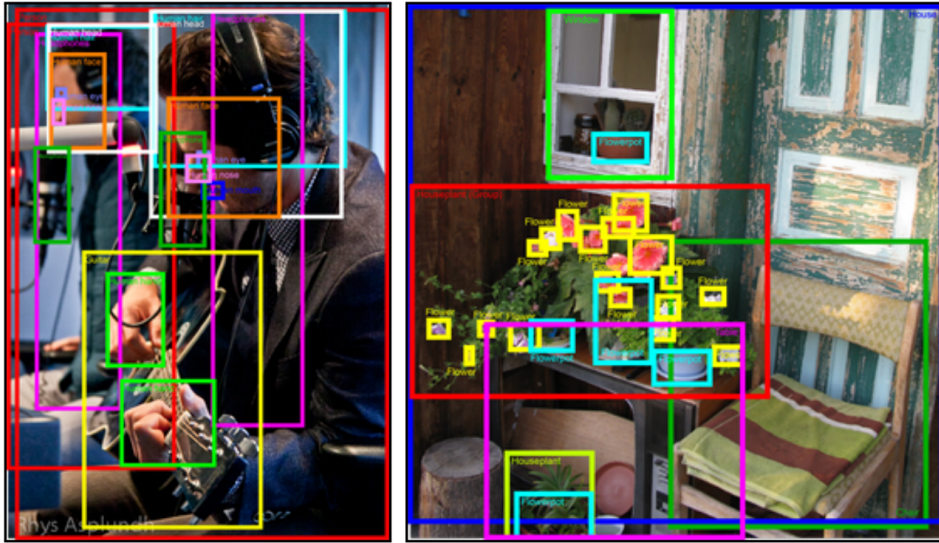


Figure 5.8: Images with a large number of different classes annotated (11 on the left, 7 on the right) [37].

5.5.2 Models

The idea of object detection came in the form classifiers applied to windows sliding over the image (e.g. based on boosting [38] or Deformable Part Models [39])., the concept of “object proposals” [40, 41] was introduced, which reduces the dense grid into just a few thousand windows.

To reduce the search space and complexity, *R-CNN* – Regions with CNN method was proposed, which uses selective search to extract only 2000 regions from the image. These regions are called *region proposals*. Therefore, instead of trying to classify a huge number of regions, only 2000 regions need to be classified [42].

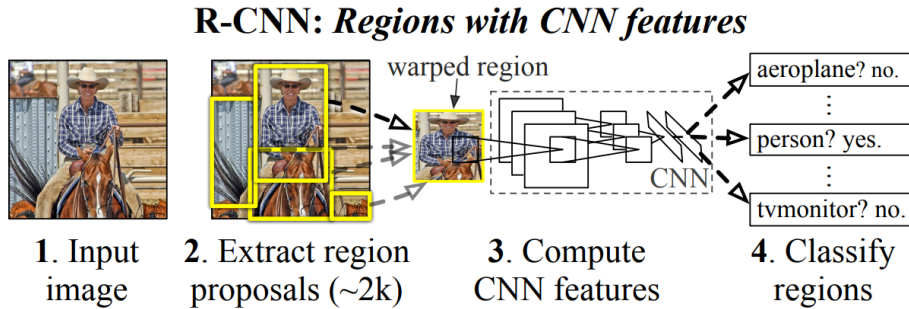


Figure 5.9: Object detection system overview. System (1) takes an input image, (2) extracts around 2000 bottom-up region proposals, (3) computes features for each proposal using a large convolutional neural network (CNN), and then (4) classifies each region [42].

Evolution was Fast R-CNN [43] and Faster R-CNN [44], which generates the proposals with a deep network as well. Faster R-CNN provides very competitive results in terms of accuracy even today. Recently, single-shot detectors were presented to bypass the computational bottleneck of object proposals by regressing object locations directly from a predefined set of anchor boxes (e.g. SSD [45] and YOLO [46]). This results in simpler models that are easier to train.

We decided to use Faster R-CNN architecture due to better availability of pre-trained models and the fact that image annotation is not a use case, where time complexity and performance is as critical as would be video annotation, for example.

Our object detection system, called Faster R-CNN, is composed of two modules. The first module is a deep, fully convolutional network that proposes regions, and the second module is the Fast R-CNN detector [43] that uses the proposed regions. The entire system is single, unified network for object detection, Figure 5.10. Using the recently popular terminology of neural networks with ‘attention’ [47] mechanisms, the RPN module tells the Fast R-CNN module where to look. A Region Proposal Network (RPN) takes an image (of any size) as input and outputs a set of rectangular object proposals, each with an objectness score. To generate region proposals, we slide a small network over the convolutional feature map output by the last shared convolutional layer [44].

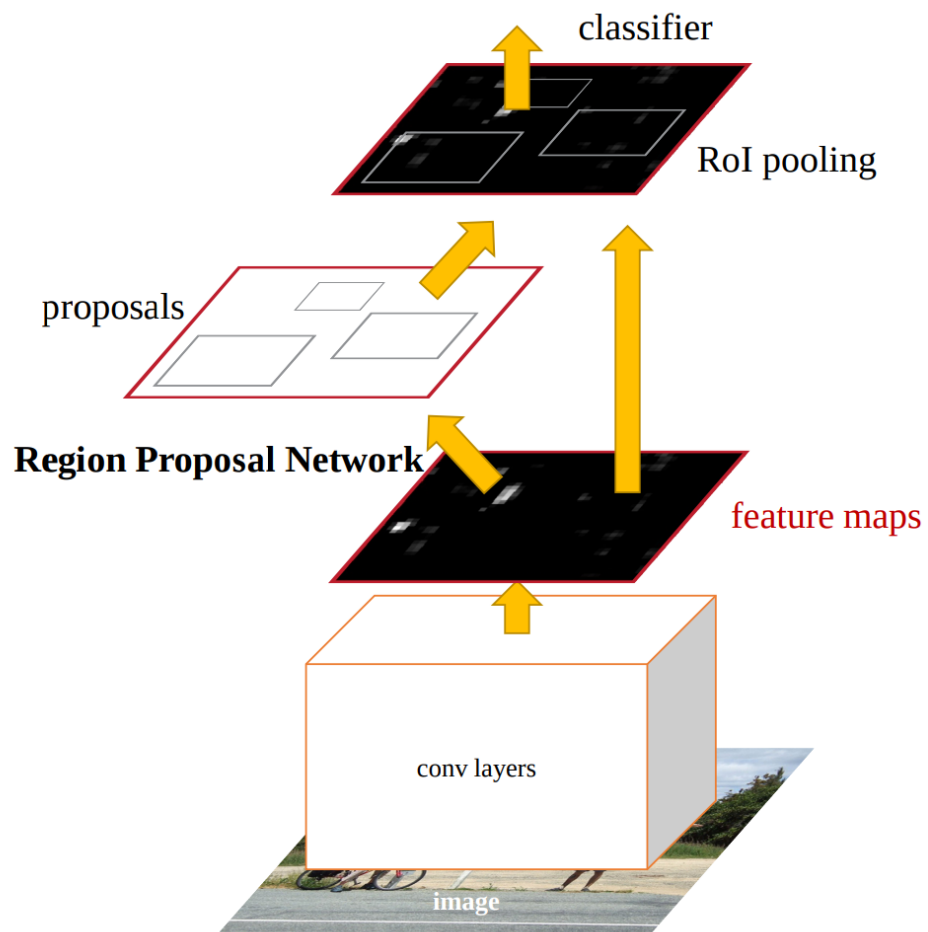


Figure 5.10: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the ‘attention’ of this unified network [44].

6 Implementation of Annotation Pipeline

The pipeline is based on the MUFIN Annotation Framework introduced in Chapter 3. It tries to improve the relevance of obtained keywords by utilizing already available information like the image caption as well as state-of-the-art image recognition techniques such as object detection and image classification. Results obtained by these techniques are part of the resulting annotation and serve as seed keywords for the MUFIN Annotation Framework, which is able to utilize them to improve the relevance of returned keywords. All these keywords are merged together to obtain image annotation. The annotation pipeline is described in Figure 6.1, and we will look into each step of the annotation pipeline in the following sections in more detail.

- Collecting keywords from Image IPTC metadata
- Parsing Image caption into keywords
- Detecting objects using neural networks
- Classifying both the image and detected boxes
- Running MUFIN annotation with seed keywords on the image and cropped boxes
- Merging keywords together into the final result

The annotation pipeline is implemented using the Python programming language. Python was chosen due to the fact that it is platform-independent and provides the best support of machine learning and data processing libraries. Annotation pipeline is able to run on any platform supporting Python 3.6 or 3.7¹ interpreter and the Tensorflow library². We were not able to improve code performance by using an alternative implementation of Python like *PyPY*³, which uses the Just-in-Time compiler to speed up the code. However, due to the fact that the Tensorflow library is not supported in PyPy, we were left with just the basic Python interpreter.

1. <https://www.python.org/>

2. [tensorflow.org/](https://www.tensorflow.org/)

3. <https://www.pypy.org/>

6. IMPLEMENTATION OF ANNOTATION PIPELINE

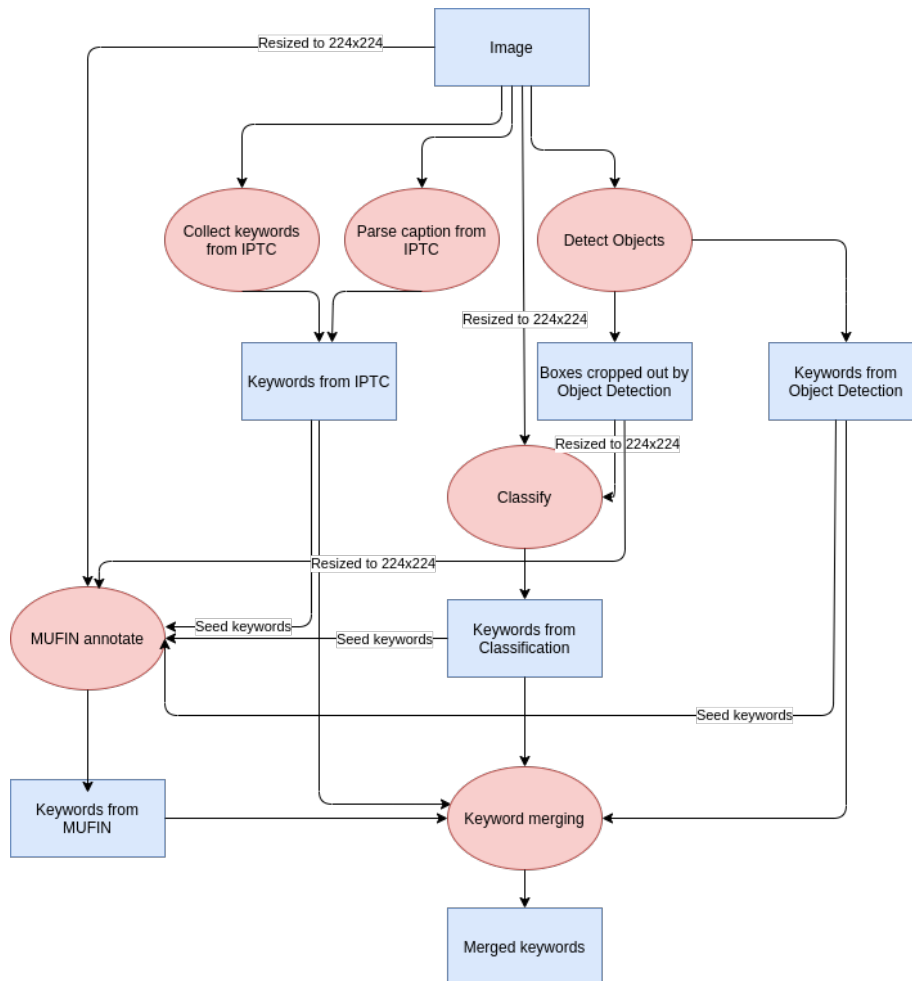


Figure 6.1: Annotation pipeline diagram. Blue color represents data elements and red represents processes or methods.

Implementation is publicly available on Github⁴. Annotation pipeline version intended for usage is in *main* branch of the repository. Branch *test* served only to prepare results for optimization. It can be easily installed using the instructions provided in the *README.md* file. The annotation pipeline was tested on Windows 10 as well as on Fedora 33 platform.

4. <https://github.com/MichalCervenansky/Automatic-image-annotation-for-microstock-sites>

6.1 IPTC Data

First step of annotation pipeline is to use keywords already written down into IPTC metadata of the photo. This enables user to write down a few keywords in advance. We also parse image caption to keywords. Implementation is in *load_from_image.py*⁵ script using slightly modified *iptcinfo3.py*⁶ library. These techniques are described in detail in Sections 5.2, 5.3. Example can be seen in Figure 6.2.



Figure 6.2: Caption:"Breathtaking landscape of rocky island of Zakynthos greece with ship wreck in laguna" resulted into keywords: "breathtaking, landscape, rocky, island, zakynthos, greece, ship, wreck, laguna"

5. https://github.com/MichalCervenansky/Automatic-image-annotation-for-microstock-sites/blob/main/Seed_keywords/load_from_image.py

6. <https://pypi.org/project/IPTCInfo3/>

6.2 Object Detection and Box Cropping

Object detection in general was already described in Section 5.5. Our implementation can be found in *object_detection.py*⁷ script. We have used Object detection model trained on Open Images V4 with ImageNet pre-trained Inception Resnet V2 as image feature extractor downloaded directly from TensorFlowHub.

Firstly, we run the detector on the image with original resolution. Image is preprocessed using *tf.image.convert_image_dtype*⁸. We obtain result in form of tuples consisting of box location, predicted class and score of the prediction. This score describes how certain is the neural network with the predicted class. Score of 1.0 translates to totally certain, while score close to 0.0 is a random guess. We filter out boxes with score higher than parameter `OD_PRECISION_THRESHOLD`, which we set to value 0.5. Then we select number of boxes defined by `OD_MAX_BOXES` parameter, which we set to 5.

These boxes are cropped out to separate images in *object_cropping.py* script. Boxes are cropped out with 50-pixel margin on each side and classified into 600 classes of Open Images V4 dataset. Results are stored in *temp* folder in form of *image_with_boxes.jpg* as shown on Figure 6.3 as well as separate images for each box. Keyword are saved in *OD_results.txt* file.

6.3 Image and Box Classification

In this step, we use ImageNet-21k multi-label classification BiT-M model. We described benefits of this model in Section 5.4. This takes place in the *image_classification.py*⁹ script. Classification is run on original image as well as cropped out boxes. This enables us to obtain more precise class for given box then the label obtained from object detection model. For example, with image of mushroom, the object detection

7. https://github.com/MichalCervenansky/Automatic-image-annotation-for-microstock-sites/blob/main/Seed_keywords/object_detection.py

8. https://www.tensorflow.org/api_docs/python/tf/image/convert_image_dtype

9. https://github.com/MichalCervenansky/Automatic-image-annotation-for-microstock-sites/blob/main/Seed_keywords/image_classification.py

will be able to detect and label "mushroom", while image classification model with a rich vocabulary will propose a label "Boletus reticulatus".

Original image and boxes with detected objects are preprocessed and resized to 224x224 resolution, which is the size of the input layer of classification model. Model is able of batch processing of multiple images at once. We leverage this fact to improve performance. ImageNet21k classes are explicitly loaded from provided *ImageNetLabels21k.txt* file. From this file, we removed synonyms and left only the first word for a given WordNet *synset*, a shortcut for a synonym set.

Classification model is run and obtained results in form of probabilities for each of the given classes are stored in a single *dataframe*¹⁰. This enables us to comfortably sort values by class probability, filter classes with probability bigger than `C_PRECISION_THRESHOLD` parameter and take first `C_MAX_CLASSES` number of classes. Results are saved into *C_results.txt* file.

In the example in Figure 6.3, we can see boxes returned by object detection with labels. These are the 5 most prominent objects found on the image. Then we run classification on the image as well as boxes and obtained 7 keywords that scored better than the defined threshold. We will use the first five in the resulting annotation.

10. <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html>

6. IMPLEMENTATION OF ANNOTATION PIPELINE



Object detection: computer keyboard, laptop,
computer monitor, desk

Classification: monitor, display, portable computer,
notebook, keyboard, personal computer, flat panel display

Figure 6.3: Example of keywords obtain by object detection and classification of the image as well as cropped out boxes.

6.4 MUFIN Annotation and Merging Results

At this point, we already obtained keywords from IPTC metadata, from the object detection process, and from the image and box classification. These keywords in this specific order will be part of the final annotation.

The next step is getting MUFIN annotation for the original image as well as for cropped-out boxes. This is done using *REST API*, where the image is sent in data content of *GET request*. As mention in Section 4.2 we give MUFIN parameter *similarImages* = 115 here. Using parameter *keywords*= and semicolon-separated keywords, we are able to give MUFIN seed keywords introduced in Section 5.1. This is done for

the original image as well as cropped-out boxes. MUFIN annotation of original image uses seed keywords obtained from user-defined keywords and parsed caption first. Secondly, there are keywords from object detection, and last are the results of the classification of the image and all objects. We have chosen this order because we believe that the keywords obtained from the user are the most relevant, while image classification tends to be the least reliable. MUFIN annotation of the single box uses as seed keywords only a single class for a given box given by object detection and classes provided by the classifier. Response in the form of XML is parsed and stored in a text file.

All results are loaded into a single dataframe, grouped by keyword, aggregated using the `minimum()` function, and sorted by *Distance* metric provided by MUFIN. In the configuration file, we set the maximum number of keywords to be returned by variable `MAX_KEYWORDS = 50`. The Resulting 50 keywords will consist of keywords obtained from IPTC metadata, from the object detection process, and from image and boxes classification continuing with keywords from MUFIN annotation till the desired number of keywords is reached.

6.5 Configuration

In our implementation, we considered the option to modify the annotation pipeline very important. This is reflected in `configuration.py`¹¹ file. We choose to store pipeline settings in Python file instead of using intuitive configuration file format such as XML, YAML, or JSON. Thanks to this, the configuration file is easier to edit for a less skilled user and does not require parsing into variables. We also benefited from the ability to rewrite the configuration file using a loop during runtime during the test phase of the annotation pipeline.

11. <https://github.com/MichalCervenansky/Automatic-image-annotation-for-microstock-sites/blob/main/configuration.py>

6. IMPLEMENTATION OF ANNOTATION PIPELINE

Parameters are following:

- `DEBUG` – When enabled, a temp folder is not deleted after the run
- `TEMP_PATH` – Path to the temp folder
- `INITIALDIR` – Defines starting directory of dialog window
- `USE_IPTC` – Sets if metadata from IPTC will be used, more in Section [6.1](#)
- `USE_OD` – Sets if object detection will be used more, in Section [6.2](#)
- `OD_PATH` – Path to the folder with object detection model
- `OD_MAX_BOXES` – Maximum number of boxes returned by the object detection
- `OD_PRECISION_THRESHOLD` – Annotating score used to filter relevant detected objects
- `USE_CL` – Sets if classification will be used more, in Section [6.3](#)
- `C_PATH` – Path to the folder with a classification model
- `C_MAX_CLASSES` – Maximum number of classes returned by classification
- `C_PRECISION_THRESHOLD` – Annotating score used to filter relevant classes
- `ANNOTATOR_URL` – URL to MUFIN Annotation Framework, more in Chapter [3](#)
- `K` – Number of keywords returned by MUFIN
- `SIMILAR_IMAGES` – MUFIN parameter discussed in detail in Section [4.2](#)
- `URL_WITH_PARAMS` – URL to MUFIN with `K` and `SIMILAR_IMAGES` parameters
- `MAX_KEYWORDS` – Maximal number of keywords returned by annotation

7 Evaluation

In this chapter, we will analyse the quality of the annotation returned by the proposed pipeline and compare it to annotation returned by MUFIN annotation framework as well as analyse performance. We also discuss the behaviour of the proposed pipeline in terms of time and memory consumption.

7.1 Comparison to the Manual Annotation

Top-1 accuracy is the standard accuracy measure used in the classification literature. It measures the proportion of examples for which the predicted label matches the single target label. However, the assumption that each image has a single ground truth label from a fixed set of classes is not valid in our use case [48]. As our annotation task uses unlimited vocabulary, we are not able to precisely measure how good our results are. We have employed the principle of comparing the intersection of keywords given by annotation pipeline and hand-annotation from the dataset introduced in Section 4.1.

This dataset contains images that are neither part of the datasets used in the MUFIN annotation framework nor datasets used in training object detection and classification models and can be used as test data. However, while vocabulary used in the dataset hand-annotated images is similar to the vocabulary used by the MUFIN, the vocabulary or, more precisely, the class labels of object detection and classification models is significantly different. This is illustrated on Figure 7.1. After adding IPTC metadata, we can see a substantial increase in not only the median of overlapping keywords but also in the lower quartiles of distribution. This is mainly due to the fact that the person writing the caption used the same vocabulary as while writing keywords. However, we can see that the number of overlapping keywords decreased after adding object detection and classification modules. This is due to the fact that the vocabulary used by these modules is different from the one used in our dataset.

7. EVALUATION

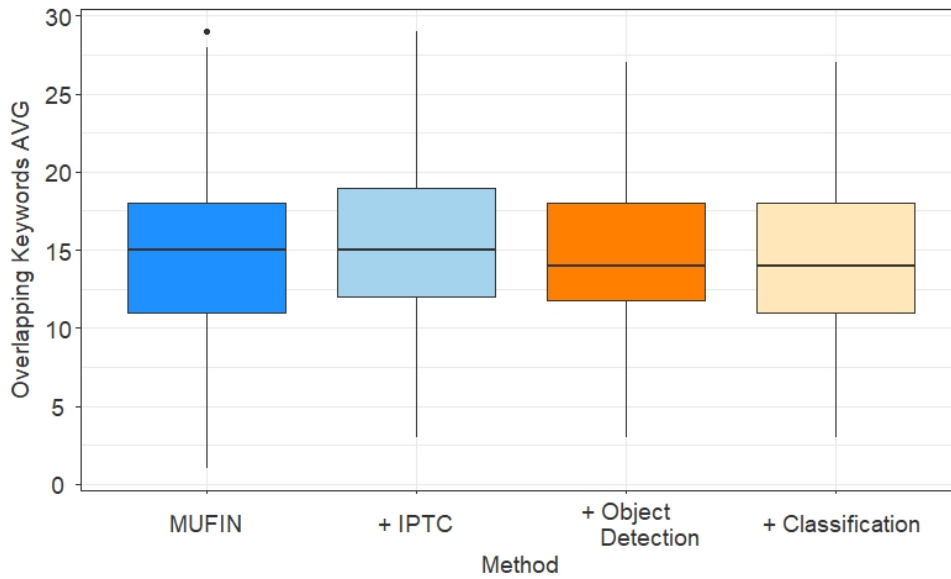


Figure 7.1: Plot shows the average number of overlapping keywords between MUFIN and our dataset.

7.2 Result Analysis

We already discussed deficiencies of the MUFIN annotation framework in Section 4.3. MUFIN focuses on the most prominent part of the image and tends to focus on the background even if the focus of the image is the object in the foreground. We believed that our pipeline is able to solve this issue using object detection and classification of detected objects. Here we present a few examples where the annotation pipeline provided desired results as well as examples where it did not.

Once again, let us remind the picture with two parrots, Figure 7.2. We can see that we obtained the keyword "parrot" from the object detection module and the classification module was able to classify the given box with the label "macaw". In this example, our pipeline was able to fill deficiencies of the MUFIN annotation framework successfully.



MUFIN keywords: person, continent, market, shop, travel, street, people, outlet, traveller, adult, business, holiday, shopping, buying, group, region, commerce, tourist, tourism, road, leisure, country, food, location, asian, centre, indoors, beach, horizontal, woman, selling, culture, display, show, class, miami, day, objects, island, marketplaces, color, new, district, man, animal, china, city, souvenir, indian, porcelain

Pipeline keywords: two, beautiful, parrots, turkish, street, parrot, man, clothing, poll, macaw, pet shop, burqa, chador, person, group, market, adult, continent, shop, activity, women, people, shopping, travel, portrait, get, view, commerce, business, road, celebration, outlet, woman, buying, year, class, occasion, holiday, offs, bird, selling, squat, inside, traditional, century, lay, new, restaurant, leisure, food

Figure 7.2: Keywords obtained from MUFIN and our pipeline. Keywords highlighted by green color are considered relevant, yellow are partially relevant, and red are irrelevant to the image. Keywords highlighted with bold and underscored font are present in hand-annotation.

7. EVALUATION

In Figure 7.3 we can see that the MUFIN annotation is vague. Compared to the result from our annotation framework, the most relevant keywords are either missing or are further away. This is the most profitable image in our dataset, as shown in Figure 2.1.



MUFIN keywords: continent, religion, person, sculpture, building, temple, statue, golden, gold, travel, belief, buddhist, believer, museum, group, creation, attribute, buddhism, art, asian, church, travellers, inside, crown, buddha, location, region, color, display, assets, saint, religious, tourist, visitor, capital, tourism, shrine, figures, monastery, people, residence, christianity, virgin, period, holy, synagogue, sightseeing, holiday, child, treasure

Pipeline keywords: luxury, crown, jewelry, red, satin, castle, slovakia, furniture, crown jewels, continents, person, religion, sculpture, temple, building, museum, structure, travel, exhibit, gold, display, statue, golden, mystic, residence, belief, jewels, treasury, group, exhibition, buddhist, casket, royal, religious, travellers, believers, symbol, sceptre, believer, assets, attribute, creation, decorate, traveller, asian, krone, czech, color, buddhism, kings

Figure 7.3: Keywords obtained from MUFIN and our pipeline. Keywords highlighted by green color are considered relevant, yellow are partially relevant, and red are irrelevant to the image. Keywords highlighted with bold and underscored font are present in hand-annotation.

Here in Figure 7.4 we present our best performer. Photo of tree mushrooms annotated by MUFIN has got only one keyword overlapping with our manual annotation. That is "food". However, our annotation pipeline managed to improve results to 13 overlapping keywords. Due to unlucky cropping out of boxes, one of the boxes was classified as "jellyfish", which is certainly not correct. This is one of the drawbacks of classifying cropped-out objects. Box can be cropped so tightly that the lack of context and surroundings fools the classifier.



MUFIN keywords: **food**, meal, bread, cooking, nourishment, dish, pastry, eat, baked, focus, apple, cuisine, sandwiches, potato, foodstuff, dessert, courses, snack, recipe, person, vegetable, cake, doughs, fried, close, bun, meat, nobody, lunch, breakfast, fruit, gastronomy, bagel, egg, plate, biscuits, mixture, cut, pie, objects, tomato, cheese, traditional, bacon, vertical, fish, hand, prepared, ready, caviar

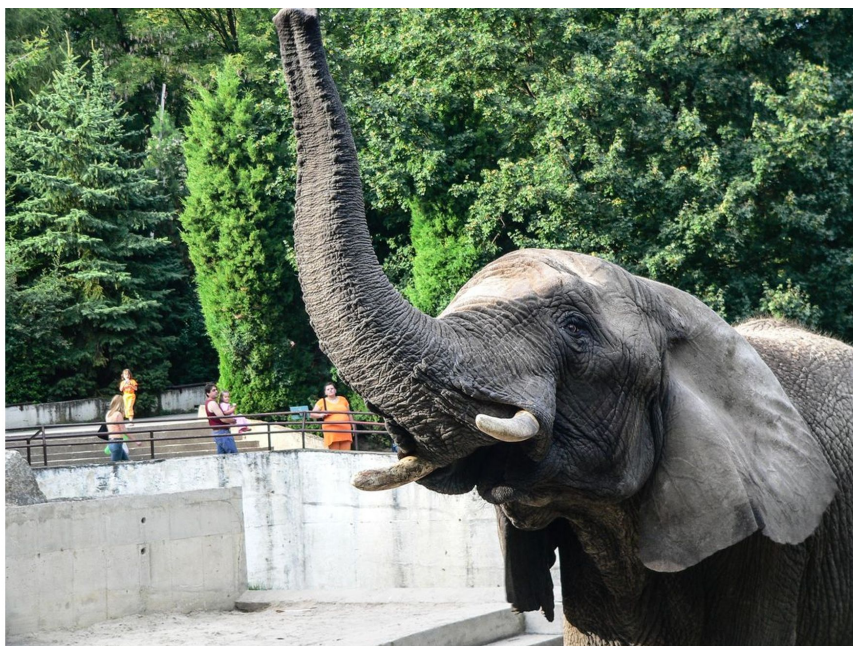
Pipeline keywords: forest, fungus, tree, winter, condition, food, mushroom, jellyfish, mushrooms, beings, change, close, meal, vegetables, pastry, nutrition, object, agaric, sandwich, cooking, bread, burger, mince, christmas, decaying, fungi, edible, sweet, boletus, group, cheeseburger, wood, season, pie, plant, nourishment, outside, detail, decomposing, closeup, autumn, fries, day, baked, white, lunch, nature, cake, photo, dough

Figure 7.4: Keywords obtained from MUFIN and our pipeline. Keywords highlighted by green color are considered relevant, yellow are partially relevant, and red are irrelevant to the image. Keywords highlighted with bold and underscored font are present in hand-annotation.

Lastly, in Figure 7.5 we present our worst performer. This photo of an elephant in a zoo annotated by MUFIN had 20 overlapping keywords with our hand-annotation. However, our pipeline managed to score only 7 overlapping keywords. After a closer look, we, however, see that annotation is still acceptable. A decrease in overlap is more due to different vocabulary than malfunctioning model. We can see that one of the boxes was classified into the label "douglas fir", which is an evergreen conifer species seen in the background. This led to introducing more keywords relevant to the background of the image.

Static web page with our dataset annotated by pipeline is available¹. We saw that our annotation pipeline and usage of seed keywords are able to improve the relevance of returned keywords mainly under challenging scenarios. The pipeline is easily modifiable, and users are welcome to experiment with different object detection and classification models. In our opinion, the weak spot of the pipeline are datasets used for training used models. Mainly ImageNet21k dataset is so rich in classes that it often tends to classify into a class with only a few examples. In our experiments, we saw particular classes, which in our opinion, should have scored worse. One example can be very specific "douglas fir" seen in Figure 7.5. While not irrelevant, it should not score that much higher than the more general "conifer", that did not make it over the threshold.

1. http://michal.cervenansky.eu/dt_dataset/html/final_keywords.html



MUFIN keywords: elephant, animal, continent, vertebrate, mammal, group, pachyderm, african, placental, tusks, person, wildlife, wild, large, national, loxodonta, safari, travel, park, ivory, nature, water, asian, reserve, trunk, tourism, tooth, adult, journey, day, tree, game, eating, river, business, activities, one, herd, zoo, tourist, people, young, daytime, family, masai, outdoors, herbivore, region, facility, photo

Pipeline keywords: old, sad, african, elephant, kept, captured, zoo, tree, douglas fir, diplodocus, travel, water, group, plant, person, states, animal, shore, beach, reap, mountain, river, knowledge, region, structure, stream, summer, nature, holiday, sky, locations, outdoors, place, view, building, wood, ocean, leisure, united, young, woman, palm, natural, color, nobody, day, area, part, objects, coast

Figure 7.5: Keywords obtained from MUFIN and our pipeline. Keywords highlighted by green color are considered relevant, yellow are partially relevant, and red are irrelevant to the image. Keywords highlighted with bold and underscored font are present in hand-annotation.

7.3 Performance Analysis

Target users of our annotation pipeline are photographers, and the pipeline needs to be executable on any personal computer. For testing of performance, we used a laptop configured with Intel® Core™ i7-8650U CPU and 32 GB of RAM. Results are shown in Figure 7.6. We can observe that the performance suffers due to the need to prepare the TensorFlow environment as well as load object detection model and classification model into the memory. Annotation of a single image took 167 seconds, from which approximately 140 seconds took the preparation of the pipeline, and computation of keywords took only 27 seconds. Taking in mind this fact, we encourage users to use our pipeline for batch processing of images. Due to the size of models, the memory usage of the annotation pipeline has peaked at 16 GB. The performance on the computer with less RAM might be decreased.

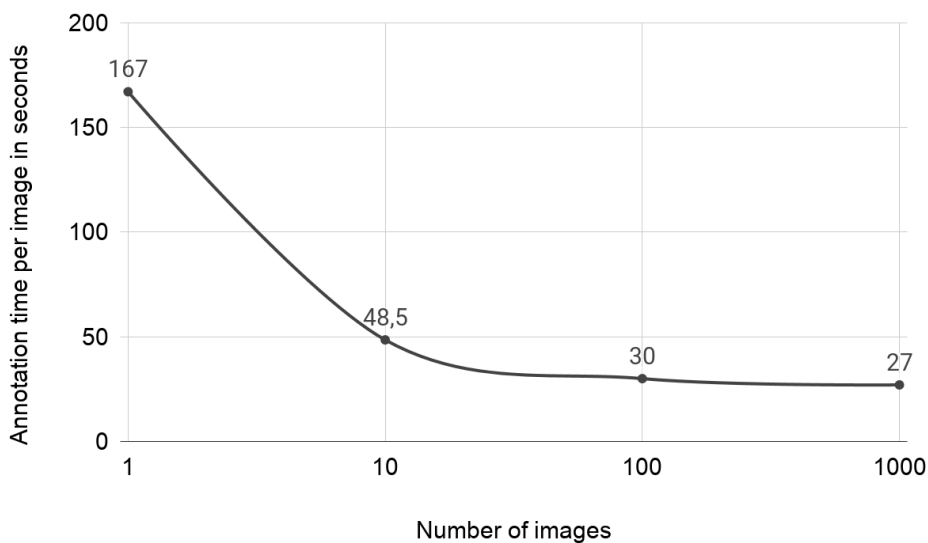


Figure 7.6: Annotation duration depends on number of annotated imaged. We can see that annotation of single image took 167 seconds, while annotating batch of 1,000 images resulted in average annotation time of 27 seconds.

Users with computer setup providing an NVIDIA® GPU card with CUDA® architecture², can strongly benefit from GPU acceleration of object detection and image classification modules. Setting TensorFlow to use GPU acceleration is described in GPU support³ guide. Due to the lack of such GPU on our test computer, we could not measure the effect.

Users who need better performance are able to disable IPTC, object detection, or classification modules. This can be set up in the configuration file described in Section 6.5. Using only MUFIN and IPTC modules still gives interesting results and decreases annotation time to about 5 seconds per image, depending on the internet connection. Users are also able to experiment with using different models. For example, using small *BiT-M R50x1*⁴ model can be done by pointing C_PATH variable to path to unzipped model or downloading it using *download_models.py*⁵ script. We experienced significantly better performance, albeit less relevant results with the BiT-M R50x1 model.

2. <https://developer.nvidia.com/cuda-gpus>

3. <https://www.tensorflow.org/install/gpu>

4. https://tfhub.dev/google/bit/m-r50x1/imagenet21k_classification/1

5. https://github.com/MichalCervenansky/Automatic-image-annotation-for-microstock-sites/blob/main/resources/NN_models/download_models.py

8 Proposed Workflow

In this chapter, we will propose a comfortable and efficient workflow how to handle image annotation task and prepare photos to upload to microstock websites. We will look into the possibilities of using our annotation pipeline for batch processing in the most efficient way.

As already mentioned in Section 2.5, photographers prefer to store caption and keywords directly in IPTC metadata of photos to be able to go through the process of annotating only once and upload images to multiple microstock websites. The first step is opening a batch of images and writing an image caption and optionally a few relevant keywords. This can be done using any image editing software supporting IPTC metadata.

We recommend using XnView MP¹. It is a versatile and powerful photo viewer, image manager as well as image editor. It is available for Windows, MacOS as well as Linux 32 and 64bit platforms. It supports IPTC metadata and various file formats, including unprocessed RAW files directly from a digital camera. One of the benefits is the customizability of XnView MP.

We customized the toolbar and added the "Annotate" button, which runs a simple *shell* script. This script runs *annotate.py* script from the annotation pipeline in a new terminal window to enable us to monitor the progress. The *annotate.py* script is run without any parameters, and it asks for input using a dialog window provided by the operating system, Figure 8.1. In this dialog window, the user selects single or multiple images, and the annotation process starts and shows progress in the terminal. To achieve the same result on the Windows platform, use *Powershell* script instead of Unix shell script.

1. <https://www.xnview.com/en/xnviewmp/>

8. PROPOSED WORKFLOW

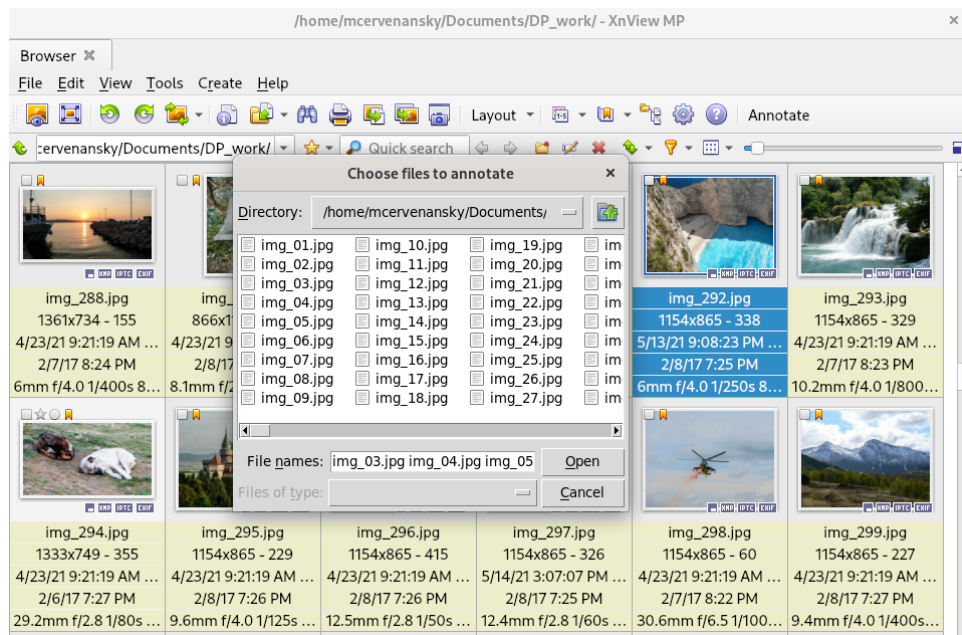


Figure 8.1: Dialog window where user selects single or multiple images. Confirmation by "Open" button starts the annotation process. "Annotate" button is located as the last tool of toolbar.

After the annotation process is finished, obtained keywords are automatically written into IPTC metadata and can be checked using XnView MP. If the user is satisfied with the result, he proceeds with uploading the images to microstock websites. This is done either using the website user interface, using FTP, or using a standalone application. This is unique approach of iStock². After images are uploaded, the user needs to complete the upload form on the microstock website, fill in attributes that are not stored in image metadata like category, and check if the image contains recognizable people. For uploading an image with a recognizable person, there is a possibility to upload signed *Model Release*³. After confirming the upload form, the image is listed as *Waiting for approval*. After approval by the microstock site worker images are listed in the portfolio and available to be purchased.

2. <https://www.istockphoto.com/>

3. https://en.wikipedia.org/wiki/Model_release

9 Conclusion

This thesis addresses the issue of image annotation for the microstock industry. It attempts to bridge the gap between a real-life problem of image annotation and the state-of-the-art research of object detection and image classification techniques. We developed an annotation pipeline optimized for obtaining keyword annotation for microstock usage. This pipeline is based on the MUFIN annotation framework and utilizes state-of-the-art machine learning technologies such as convolutional neural networks to achieve the most relevant annotation. The pipeline is available at GitHub repository¹ and was shared with photographers in the microstock community.

In this thesis, we introduced the reader to the microstock industry and shared tips based on years of our experience in this field. With MUFIN Annotation Framework, we started a discussion about the fully automatic approach to obtaining keyword annotation. We compared the relevance of MUFIN's results to the dataset of 1,000 hand-annotated images already published in the author's microstock portfolio. We proposed an approach of using seed keywords to improve the relevance of MUFIN's results and discussed the possibilities of obtaining them. In short, we described the implementation of the annotation pipeline and analysed relevance obtained annotation as well as pipeline performance. The thesis finished with recommending the most efficient workflow utilizing our annotation pipeline.

Research in the machine learning and image recognition field is rapid, and currently used models could be outclassed by newer, better-performing models in a short amount of time. Thanks to the modular structure of the annotation pipeline, it might be a handy tool to analyse the performance of new models and utilize the best-performing ones.

1. <https://github.com/MichalCervenansky/Automatic-image-annotation-for-microstock-sites>

Bibliography

1. *What is Microstock? | Marcorstock* [online]. 2014 [visited on 2021-05-14]. Available from: <https://marcorstock.com/en/what-is-microstock/>.
2. *What is Microstock Photography? - Definition from Techopedia* [online] [visited on 2021-05-14]. Available from: <http://www.techopedia.com/definition/25894/microstock-photography>.
3. *Top 10 microstocks for contributors in 2021* [online] [visited on 2021-05-14]. Available from: <https://xpiksapp.com/blog/top-microstocks/>.
4. *IPTC Information Interchange Model* [online]. 2020 [visited on 2021-05-14]. Available from: https://en.wikipedia.org/w/index.php?title=IPTC_Information_Interchange_Model&oldid=981934023. Page Version ID: 981934023.
5. G, Leigh. *Complete Guide To Describing & Keywording Photos & Vectors* [online]. 2020 [visited on 2021-04-21]. Available from: <https://blog.123rf.com/guide-describing-and-keywording-photos-and-vectors/>.
6. BATKO, Michal; NOVÁK, David; ZEZULA, Pavel. *MESSIF: Metric Similarity Search Implementation Framework* [online]. Information Society Technologies, 2007 [visited on 2021-05-11]. ISBN 978-2-912335-30-2. Available from: <https://is.muni.cz/publication/712985/cs/MESSIF-Metric-Similarity-Search-Implementation-Framework/Batko-Novak-Zezula>.
7. BATKO, Michal; BOTOŘEK, Jan; BUDÍKOVÁ, Petra; ZEZULA, Pavel. *Content-based annotation and classification framework: A general multi-purpose approach*. 2013. Available from DOI: [10.1145/2513591.2513651](https://doi.org/10.1145/2513591.2513651). Journal Abbreviation: ACM International Conference Proceeding Series Pages: 67 Publication Title: ACM International Conference Proceeding Series.

BIBLIOGRAPHY

8. BUDIKOVA, Petra; BATKO, Michal; ZEZULA, Pavel. ConceptRank for search-based image annotation. *Multimedia Tools and Applications* [online]. 2018, vol. 77, no. 7, pp. 8847–8882 [visited on 2021-05-10]. ISSN 1380-7501, 1573-7721. ISSN 1380-7501, 1573-7721. Available from DOI: [10.1007/s11042-017-4777-8](https://doi.org/10.1007/s11042-017-4777-8).
9. *Image Annotation | DISA Laboratory* [online] [visited on 2021-05-11]. Available from: <http://disa.fi.muni.cz/demo/image-annotation/>.
10. BUDIKOVA, Petra; BATKO, Michal; ZEZULA, Pavel. MUFIN at ImageCLEF 2011: Success or Failure?, pp. 12.
11. DONAHUE, Jeff; JIA, Yangqing; VINYALS, Oriol; HOFFMAN, Judy; ZHANG, Ning; TZENG, Eric; DARRELL, Trevor. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. *arXiv:1310.1531* [cs] [online]. 2013 [visited on 2021-05-11]. Available from: <http://arxiv.org/abs/1310.1531>. arXiv: 1310.1531.
12. LESKOVEC, Jure; RAJARAMAN, Anand; ULLMAN, Jeffrey D. Mining of Massive Datasets, pp. 513.
13. BRIN, Sergey; PAGE, Lawrence. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems* [online]. 1998, vol. 30, no. 1, pp. 107–117 [visited on 2021-05-11]. ISSN 0169-7552. Available from DOI: [10.1016/S0169-7552\(98\)00110-X](https://doi.org/10.1016/S0169-7552(98)00110-X).
14. GYONGYI, Zoltan; GARCIA-MOLINA, Hector; UNIV, Stanford; PEDERSEN, Jan. Combating Web Spam with TrustRank, pp. 12.
15. BUDIKOVA, Petra; BATKO, Michal; ZEZULA, Pavel. Multi-modal Image Retrieval for Search-Based Image Annotation with RF. In: *2018 IEEE International Symposium on Multimedia (ISM)* [online]. Taichung: IEEE, 2018, pp. 52–60 [visited on 2021-05-10]. ISBN 978-1-5386-6857-3. Available from DOI: [10.1109/ISM.2018.00017](https://doi.org/10.1109/ISM.2018.00017).
16. PAGARE, Reena; SHINDE, Anita. A Study on Image Annotation Techniques. *International Journal of Computer Applications*. 2012, vol. 37, pp. 42–45. Available from DOI: [10.5120/4616-6295](https://doi.org/10.5120/4616-6295).

17. *What's in an Image Title?* [online]. 2011 [visited on 2021-04-28]. Available from: <https://www.shutterstock.com/blog/whats-in-an-image-title>.
18. BIRD, Steven; KLEIN, Ewan; LOPER, Edward. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. "O'Reilly Media, Inc.", 2009. ISBN 978-0-596-55571-9. Google-Books-ID: KG1bfiiP1i4C.
19. *Natural Language Toolkit — NLTK 3.6.2 documentation* [online] [visited on 2021-05-03]. Available from: <https://www.nltk.org/>.
20. *Python - Remove Stopwords - Tutorialspoint* [online] [visited on 2021-04-28]. Available from: https://www.tutorialspoint.com/python_text_processing/python_remove_stopwords.htm.
21. WANG, Shuai; SU, Zhendong. Metamorphic Testing for Object Detection Systems. *arXiv:1912.12162 [cs]* [online]. 2019 [visited on 2021-05-10]. Available from: <http://arxiv.org/abs/1912.12162>. arXiv: 1912.12162.
22. DIAS, Lucas V.; MIRANDA, Péricles B. C.; NASCIMENTO, André C. A.; CORDEIRO, Filipe R.; MELLO, Rafael Ferreira; PRUDÊN-CIO, Ricardo B. C. ImageDataset2Vec: An Image Dataset Embedding for Algorithm Selection. *Expert Systems with Applications* [online]. 2021, pp. 115053 [visited on 2021-04-27]. ISSN 0957-4174. Available from DOI: [10.1016/j.eswa.2021.115053](https://doi.org/10.1016/j.eswa.2021.115053).
23. *ImageNet* [online] [visited on 2021-05-16]. Available from: <https://www.image-net.org/>.
24. DENG, Jia; DONG, Wei; SOCHER, Richard; LI, Li-Jia; LI, Kai; FEI-FEI, Li. ImageNet: A large-scale hierarchical image database. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. Available from DOI: [10.1109/CVPR.2009.5206848](https://doi.org/10.1109/CVPR.2009.5206848). ISSN: 1063-6919.
25. KOLESNIKOV, Alexander; BEYER, Lucas; ZHAI, Xiaohua; PUIGSERVER, Joan; YUNG, Jessica; GELLY, Sylvain; HOULSBY, Neil. Big Transfer (BiT): General Visual Representation Learning. *arXiv:1912.11370 [cs]* [online]. 2020 [visited on 2021-03-05].

BIBLIOGRAPHY

- Available from: <http://arxiv.org/abs/1912.11370>. arXiv: 1912.11370.
26. CODREANU, Valeriu; PODAREANU, Damian; SALETORE, Vikram. Scale out for large minibatch SGD: Residual network training on ImageNet-1K with improved accuracy and reduced time to train. *arXiv:1711.04291* [cs, stat] [online]. 2017 [visited on 2021-05-10]. Available from: <http://arxiv.org/abs/1711.04291>. arXiv: 1711.04291.
 27. DOSOVITSKIY, Alexey et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv:2010.11929* [cs] [online]. 2020 [visited on 2021-05-10]. Available from: <http://arxiv.org/abs/2010.11929>. arXiv: 2010.11929.
 28. MILLER, George A. WordNet: a lexical database for English. *Communications of the ACM* [online]. 1995, vol. 38, no. 11, pp. 39–41 [visited on 2021-05-10]. ISSN 0001-0782, 1557-7317. ISSN 0001-0782, 1557-7317. Available from DOI: [10.1145/219717.219748](https://doi.org/10.1145/219717.219748).
 29. RIDNIK, Tal; BEN-BARUCH, Emanuel; NOY, Asaf; ZELNIK-MANOR, Lihi. ImageNet-21K Pretraining for the Masses. *arXiv:2104.10972* [cs] [online]. 2021 [visited on 2021-05-10]. Available from: <http://arxiv.org/abs/2104.10972>. arXiv: 2104.10972.
 30. XIN, Mingyuan; WANG, Yong. Research on image classification model based on deep convolution neural network. *EURASIP Journal on Image and Video Processing* [online]. 2019, vol. 2019, no. 1, pp. 40 [visited on 2021-05-10]. ISSN 1687-5281. Available from DOI: [10.1186/s13640-019-0417-8](https://doi.org/10.1186/s13640-019-0417-8).
 31. SUN, Chen; SHRIVASTAVA, Abhinav; SINGH, Saurabh; GUPTA, Abhinav. Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. *arXiv:1707.02968* [cs] [online]. 2017 [visited on 2021-05-10]. Available from: <http://arxiv.org/abs/1707.02968>. arXiv: 1707.02968.
 32. FELZENSZWALB, Pedro F; GIRSHICK, Ross B; MCALLESTER, David; RAMANAN, Deva. Object Detection with Discriminatively Trained Part Based Models, pp. 20.

33. ZHAO, Zhong-Qiu; ZHENG, Peng; XU, Shou-tao; WU, Xindong. Object Detection with Deep Learning: A Review. *arXiv:1807.05511* [cs] [online]. 2019 [visited on 2021-05-03]. Available from: <http://arxiv.org/abs/1807.05511>. arXiv: 1807.05511.
34. LIN, Tsung-Yi et al. Microsoft COCO: Common Objects in Context. *arXiv:1405.0312* [cs] [online]. 2015 [visited on 2021-04-21]. Available from: <http://arxiv.org/abs/1405.0312>. arXiv: 1405.0312.
35. EVERINGHAM, Mark; VAN GOOL, Luc; WILLIAMS, Christopher K. I.; WINN, John; ZISSERMAN, Andrew. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision* [online]. 2010, vol. 88, no. 2, pp. 303–338 [visited on 2021-05-08]. ISSN 0920-5691, 1573-1405. ISSN 0920-5691, 1573-1405. Available from DOI: [10.1007/s11263-009-0275-4](https://doi.org/10.1007/s11263-009-0275-4).
36. XIAO, Jianxiong; HAYS, James; EHINGER, Krista A.; OLIVA, Aude; TORRALBA, Antonio. SUN database: Large-scale scene recognition from abbey to zoo. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* [online]. San Francisco, CA, USA: IEEE, 2010, pp. 3485–3492 [visited on 2021-05-08]. ISBN 978-1-4244-6984-0. Available from DOI: [10.1109/CVPR.2010.5539970](https://doi.org/10.1109/CVPR.2010.5539970).
37. KUZNETSOVA, Alina et al. The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale. *International Journal of Computer Vision* [online]. 2020, vol. 128, no. 7, pp. 1956–1981 [visited on 2021-04-21]. ISSN 0920-5691, 1573-1405. ISSN 0920-5691, 1573-1405. Available from DOI: [10.1007/s11263-020-01316-z](https://doi.org/10.1007/s11263-020-01316-z). arXiv: 1811.00982.
38. VIOLA, Paul; JONES, Michael. Rapid Object Detection using a Boosted Cascade of Simple Features, pp. 9.
39. FELZENSZWALB, Pedro F.; GIRSHICK, Ross B.; MCALLESTER, David; RAMANAN, Deva. Object Detection with Discriminatively Trained Part-Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2010, vol. 32, no. 9, pp. 1627–1645. ISSN 1939-3539. Available from DOI: [10.1109/TPAMI.2009.167](https://doi.org/10.1109/TPAMI.2009.167). Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

BIBLIOGRAPHY

40. ALEXE, Bogdan; DESELAERS, Thomas; FERRARI, Vittorio. What is an object? In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2010, pp. 73–80. Available from DOI: [10.1109/CVPR.2010.5540226](https://doi.org/10.1109/CVPR.2010.5540226). ISSN: 1063-6919.
41. UIJLINGS, J. R. R.; SANDE, K. E. A. van de; GEVERS, T.; SMEULDERS, A. W. M. Selective Search for Object Recognition. *International Journal of Computer Vision* [online]. 2013, vol. 104, no. 2, pp. 154–171 [visited on 2021-05-10]. ISSN 0920-5691, 1573-1405. Available from DOI: [10.1007/s11263-013-0620-5](https://doi.org/10.1007/s11263-013-0620-5).
42. GIRSHICK, Ross; DONAHUE, Jeff; DARRELL, Trevor; MALIK, Jitendra. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition* [online]. Columbus, OH, USA: IEEE, 2014, pp. 580–587 [visited on 2021-05-10]. ISBN 978-1-4799-5118-5. Available from DOI: [10.1109/CVPR.2014.81](https://doi.org/10.1109/CVPR.2014.81).
43. GIRSHICK, Ross. Fast R-CNN. *arXiv:1504.08083* [cs] [online]. 2015 [visited on 2021-05-10]. Available from: <http://arxiv.org/abs/1504.08083>. arXiv: 1504.08083.
44. REN, Shaoqing; HE, Kaiming; GIRSHICK, Ross; SUN, Jian. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *arXiv:1506.01497* [cs] [online]. 2016 [visited on 2021-05-10]. Available from: <http://arxiv.org/abs/1506.01497>. arXiv: 1506.01497.
45. LIU, Wei; ANGUELOV, Dragomir; ERHAN, Dumitru; SZEGEDY, Christian; REED, Scott; FU, Cheng-Yang; BERG, Alexander C. SSD: Single Shot MultiBox Detector. *arXiv:1512.02325* [cs] [online]. 2016, vol. 9905, pp. 21–37 [visited on 2021-05-10]. Available from DOI: [10.1007/978-3-319-46448-0_2](https://doi.org/10.1007/978-3-319-46448-0_2). arXiv: 1512.02325.
46. REDMON, Joseph; FARHADI, Ali. YOLO9000: Better, Faster, Stronger. *arXiv:1612.08242* [cs] [online]. 2016 [visited on 2021-05-10]. Available from: <http://arxiv.org/abs/1612.08242>. arXiv: 1612.08242.

BIBLIOGRAPHY

47. CHOROWSKI, Jan; BAHDANAU, Dzmitry; SERDYUK, Dmitriy; CHO, Kyunghyun; BENGIO, Yoshua. Attention-Based Models for Speech Recognition. *arXiv:1506.07503 [cs, stat]* [online]. 2015 [visited on 2021-05-10]. Available from: <http://arxiv.org/abs/1506.07503>. arXiv: 1506.07503.
48. SHANKAR, Vaishal; ROELOFS, Rebecca; MANIA, Horia; FANG, Alex; RECHT, Benjamin; SCHMIDT, Ludwig. Evaluating Machine Accuracy on ImageNet, pp. 11.