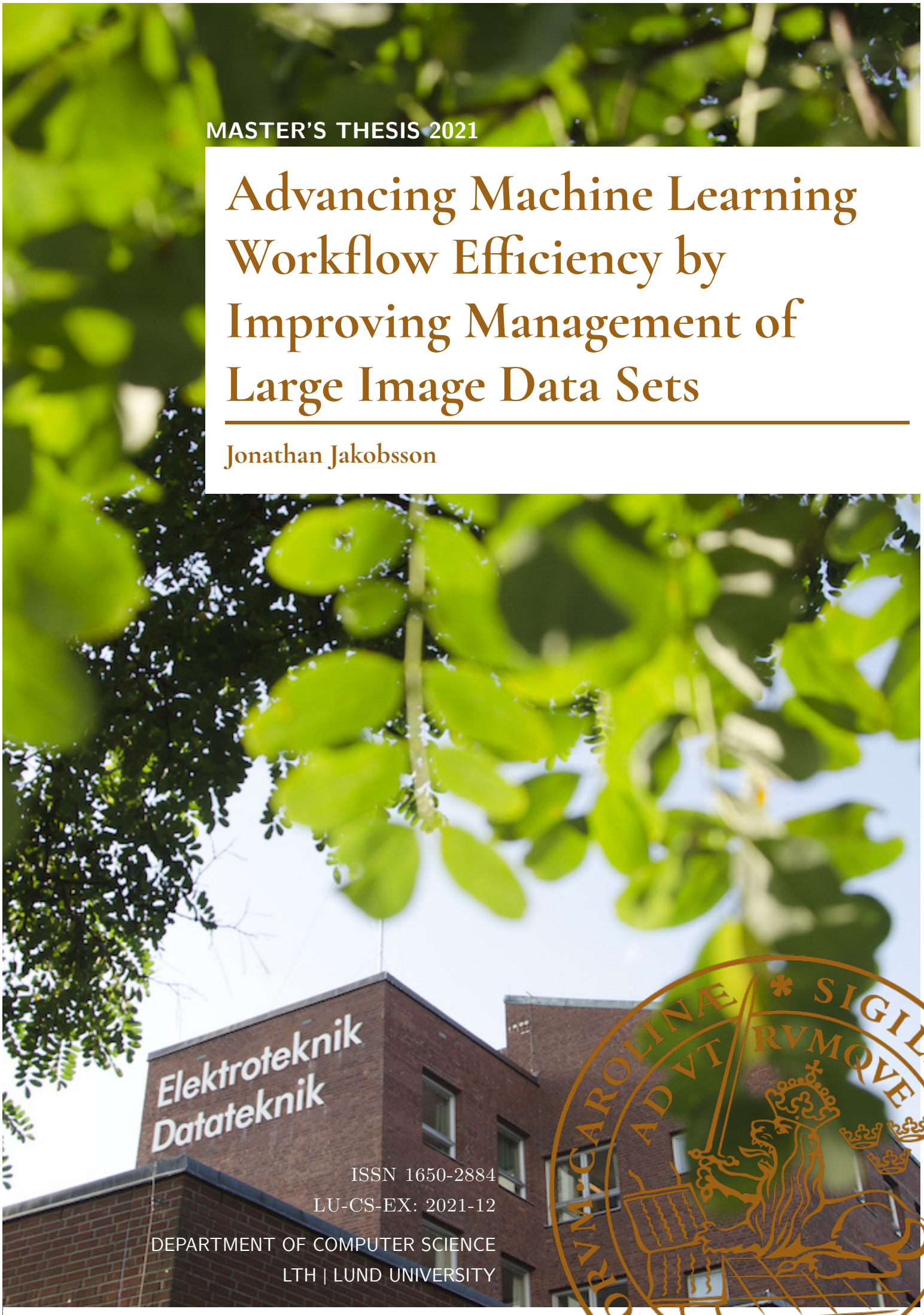
MASTER'S THESIS 2021

Advancing Machine Learning Workflow Efficiency by Improving Management of Large Image Data Sets

Jonathan Jakobsson



ISSN 1650-2884
LU-CS-EX: 2021-12

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2021-12

Advancing Machine Learning Workflow Efficiency by Improving Management of Large Image Data Sets

Effektivisering av maskininlärnings process genom förbättrad hantering av stora bilddata set

Jonathan Jakobsson

Advancing Machine Learning Workflow Efficiency by Improving Management of Large Image Data Sets

Jonathan Jakobsson
elt15jja@student.lu.se

June 2, 2021

Master's thesis work carried out at SiB Solutions AB.

Supervisors: Alma Orucevic-Alagic, alma.orucevic-alagic@cs.lth.se
Sverrir Valgeirsson, sverrir.valgeirsson@sibsolutions.se

Examiner: Mathias Haage, mathias.haage@cs.lth.se

Abstract

Due to the rising use of Machine Learning in the tech industry the last decade and the increasing amount of publicly available data, the demand for data and Machine Learning scientists has grown as well. The lack of resources prevents many companies from utilizing Machine Learning strategies in their projects. This thesis explores a workflow at SiB Solutions with the goal of creating and maintaining several image data sets used for adapting Machine Learning models for various use cases. However, the workflow is currently deemed unfeasible due to the amount resources required to operate. In order to propose improvements to the workflow, the components of the workflow are analyzed and the issues of the components are identified. The issues identified are mainly large amounts of manual labor, high competence requirement due to lack of manageability of data sets. An improved workflow is proposed for each of the components by researching causes and solutions the the issues. The result is a proposed workflow that is fully automated with the exception to two cases: manual correction in cases where the automatic data pipeline makes mistakes and when a special customization of the data set, beyond the regular customization of the workflow, is required. An example implementation is made for the components of the workflow that are deemed to be the major sources of inefficiency as a proof of concept. The automated workflow, along with the addition of easily interpretable data set specifications, provide the ability to effortlessly generate and recreate data sets, provides improved management of the data sets. Thus, the amount of required resources to operate the workflow, in terms of time and competence, is reduced.

Keywords: MSc, CS, Machine Learning Workflow, Object detection, Image Processing, Data-driven workflow, Data Science, Image Data Sets

Acknowledgements

Special thanks goes to Alma Orucevic-Alagic (Department of Computer Science at LTH) for excellent feedback, supervision and suggestions throughout this project which helped immensely. A big thanks to all my co-workers at SiB Solutions AB and my supervisor Sverrir Valgeirsson (SiB Solutions) for providing me with the opportunity to perform this thesis project as the company and providing me with excellent supervision and guidance. I would also like to thank Mathias Haage (Department of Computer Science at LTH) for taking the time to examine this project

Contents

Chapter 1
Introduction

In the modern world of computer science, the concept known as "Big data" [1] introduces new challenges that pertain to the management of the vast amount of data that has become widely available with modern information collection and distribution. In the context of image and video data, in order to properly utilize all the information collected by various systems, companies and software systems have come to rely on automatic processing by machine-learning (ML) for automatic detecting and classification of objects. Companies within various industrial fields have started to utilize this type of processing. Examples of such fields include video-surveillance [2], medical imagery analysis [3] and automatic vehicle control [4]. Furthermore, image processing techniques such as object detection and image classification has become more accessible to the general public within the last decade. This is, to a great extent, due to large companies such as Google [5] and Facebook [6] supporting ML through the development of high-level frameworks that enable more accessible utilization of ML techniques for automatic media processing [7].

In regards to image processing using such frameworks, a ML model is a neural network or algorithm tuned through the provision of training data which adapts the parameters of the model to make predictions about unseen input data in a specific use case[8, pp. 14–15]. Such models are utilized by the high-level frameworks to effectively process data [9]. But in order to fully utilize the effectiveness of intelligent ML processing, the ML models, require a massive amount of training data to reach their full potential in terms of accuracy of the predictions [10]. This leads to collections of image training data, also known as training data sets, to become increasingly large in size due to the number of images required to effectively adapt the model. Apart from size, another challenge is to effectively manage such data sets as more data is added. As data is continuously added to the data set, the expanding data sets become harder to manage. Notably, in terms of processing massive amounts of incoming data in order to verify and utilize it to expand existing data sets. The purpose of the verification is the detection of invalid data, for example, data that is either presented in an incorrect format, mislabeled or has been incorrectly placed into data set. This is important since adding

invalid data can have a notably negative impact on the accuracy of the model when used to conceive a ML model [11]. Thus it becomes necessary to prevent this by managing the data sets more carefully with method such as version control and verifying the data before approving it for admission into the data set. However, if it is done through backing up the data set at various points, it means storing multiple version of the same data set which is is often already impractically large.

This has prompted the creation of management tools for handling the storage and version control of large data files [12, 13] that can be utilized for easier management of image data sets. While tools like this has greatly contributed to remedy the issues, there are still critical issues related to the management of data sets left to solve. Mainly the process of extending the data set and the generation of subsets from the data pool. Both of which require a substantial investment in terms of time and knowledge of the process.

This highlights a core issue for the many industries relying on ML advancement: high labor cost for data management and an extensive need for technical expertise in the ML and data science area. 2017 IBM published a report [14] describing how the demand for data science skill is rising alongside technology but the job market cannot supply the demand in relation to this issue. Further, many recent news-articles [15] and reports [16] claims that the increasing shortage and rising demand for AI/ML engineers and data scientists indicates that the cost imposed by utilizing ML strategies makes it inaccessible to many industries and companies due to competence shortage.

An initiative to continuously improve ML models has been dubbed Automated machine learning (AutoML [17]) and has gained popularity in recent years. The goals of which partially aims to remedy the issues related to the lack of expertise and labor cost by automating the iteration process of creating a ML model for specific function [17]. The main objective is described as the ability to make decisions in a data-driven, objective, and automated way to solve the provided issue [18, p. ix]. The user would simply provide data and gain a solutions to the problem. In other words, it strives to effectively eliminate the need for human involvement in the process beyond the initial setup and problem definement.

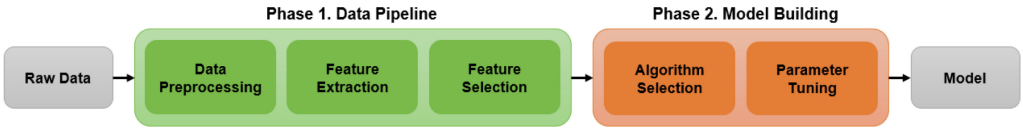


Figure 1.1: The general high-level workflow of the AutoML initiative. Source: [19]

The figure above visualizes a high-level simplified workflow of the AutoML iteration process from incoming raw data to a tuned model. The second phase, labeled “Phase 2: Model Building”, of the simplified workflow consists of the generation and tuning of the model used for predictions. However, this thesis applies to the following case, where the model is already created and tuning is implemented by one of the mentioned high-level ML frameworks. Therefore, the area prone for improvement is the data pipeline specified, defined as the first phase of the workflow described in the paragraph below.

The first phase, labeled “Phase 1. Data Pipeline”, consists of the three following substeps: Processing the data by extracting all information required for generating the data set in the “Data Preprocessing” step, defining the the features of the data in the “Feature Extraction” step and finally selecting which features the model should eventually learn to recognize in the “Feature Selection” step. These features can be very ambiguous in their definition so finding features which can act as input parameters for the second phase is crucial to the process. This phase is the main focus of this thesis. This AutoML data pipeline workflow should endeavor to implement a data-driven solution that processes raw data and finally serve a fully prepared data sets, with minimum supervision, ready to be deployed directly into the second phase. Ideally it would act as black box, void of human interaction [18, p. 181]. This thesis aims to propose such a workflow.

1.1 Problem Summary

SiB Solutions AB is a startup company based in Lund, Sweden, that aims to reduce the amount of errors in the logistics sector by providing services that offer solutions and optimizations to problem in supply chains. The company was founded in 2017 and has, as of the time of writing, 23 employees and is expanding rapidly into the global logistics sector. The services are based on video analytics supported by artificial intelligence which is used to detect and ideally prevent errors as they occur. A practical example of this is the ability to prevent the incorrect items from being packaged into parcel before it has been sent by identifying the inside using a video feed and cross referencing it with the intended content in the parcel content note.

To accomplish this, the company relies on the object detection technology in order to automatically analyze the movement of objects in real-time of desired object using a live video-feed for many various situations and environments. To handle the large amount of incoming video data, the company currently utilizes object detection using machine-learning model (ML models). These ML models are pre-designed in order to be utilized by a high-level framework and pre-trained on a large general image data set in order tweak the models parameters for use within image processing, as is expanded upon in Section 2.1.3. However, in order to adapt a model to one of multiple specific use cases desired by the company, a data set for the specific circumstance the model will handle needs to be crafted. The crafted data sets are constantly expanding and may share common image data since use cases often overlap.

A practical example of this is cardboard boxes and containers which are both items that are consistently present within warehouses and logistic companies’ other areas of operation. Both items are generally of interest to track which means that most of the data sets share a great deal of images containing the above-mentioned items. Additionally, to gain the expected effectiveness of the adaptation of the models, the data set needs to be quite large. This is due to the sheer effectiveness increased training data amount has in terms of an ML models prediction accuracy [10] and the requirement for accuracy is high in order to provide the desired confidence in the models conclusions about any given circumstance. In conclusion, the company faces several challenges when expanding the data set due to the following factors: The cross-dependency of data sets, the large amount of data and the varying scenery of the

various sources.

To continuously improve the model accordingly, the company has a workflow to build and extend current data sets used for training using image data from the cameras supervising the scene. The current process is visualized on a high-level in Figure 1.2.

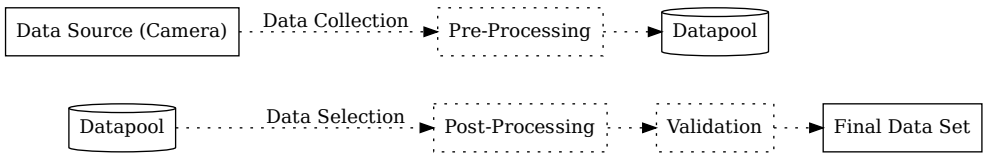


Figure 1.2: Current Dataflow with the manual components dotted

Image and video data is fed continuously to a central location. Operators can then manually collect and process the data from cameras in order to add the data to the main Datapool which contains all the processed data collected. To then make a training data set, data is manually selected from this main data pool. It is then prepared through manual means to fit the required data formats and validated by means of manual judgment to detect any general issues or flaws that might impact the final model.

Several issues can be identified in the current process. The main reason for ineffectiveness is the amount of manual intervention required. The manual solutions in data-science has the disadvantages of being expensive in terms of time. Not only due to the initial time investment during the collection but additionally the introduction of human errors. Human errors not only including oversights and mishandling but also sync and communication errors between multiple data operators. Examples of such being: referring to the same object by different names, storing data in different locations and in different formats. All of which cause major difficulties for the operator crafting the data set along with the understanding of the use case and general ML knowledge required to decide what the data set should consist of. This translates into massive time, communication and competence requirements due to the massive amount of incoming data. Thus, processing images and selecting a subset are the two potential candidates for major improvement here. Another is the unnecessary creation of copies of existing data which could also be stored more efficiently. As previously mentioned, data sets share a large amount of images and images are copied from the main data pool in order to be added to a new data set. And lastly: major improvements could be made to the management of the subsets. Since the data set are crafted manually, they have no consistent system of expansion, means to reproduce or clear descriptions of content unless the craftsman of the data set chose to create a description and include it. In which case the result can have substantial differences due to the lack of standard procedure. An optimal pipeline would instead act as an interactive black box as the user would feed image data and then query for finished data sets, visualized below.

A deeper analysis of the current and proposed workflow is presented in Section 4.1.

This thesis aims to alleviates the issues by proposing an alternative workflow. Evaluate how the proposed solutions affects the efficiency of the workflow in terms of time required, how

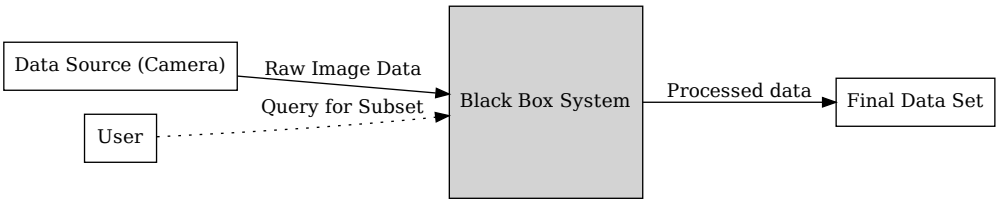


Figure 1.3: Ideal Dataflow

easily it can be managed with limited ML competence, the extent the data sets can be reproduced and how clear the contents of a data set can be presented. Ideally, human interaction should be limited as much as possible.

1.2 Related Work

1.2.1 Workflow analysis

The main idea of this project is to analyze and improve a workflow by breaking down and analyzing the individual components of the process. This is done by establishing characteristics of individual components such as context, goal and requirements by researching use-cases [20, pp. 83–85]. Enhancements may be proposed that accomplish the desired goal more effectively [20, pp. 86–88].

1.2.2 AutoML

A priority when researching process improvement solutions shall be to strive for a high automation degree as the amount of human resources in this case is extremely limited. As such, the project project aims to further the evolving initiative of AutoML by proposing a strategy for the data-driven data-collection pipeline [19].

1.2.3 Feature Extraction

To effectively categorize raw image data, a system for classifying the image by its identifying features needs to be employed. Techniques for categorization will be employed and evaluated for the entire image and for any objects the image might contain such as proposed by the [21] article. To extract the features: techniques using saliency [22] (similarity), extraction using ML-strategies [23, 24] and Scale-invariant feature transform [25] will be implemented to then be evaluated on how effective respective method is in regards to categorization.

1.2.4 Image Data Modeling

One of the main difficulties in building an image database is how to effectively model the data in order to effectively and accurately query the main data pool for the requested data. In order to accomplish this, the following research paper [26] presents a strategy to effectively model the image database for effective queries. However, it lacks implementations of many modern feature extraction techniques such as *Multi-label image classification* [27] when having images with multiple object and supports for all techniques mentioned to extract features. In addition, articles such as [28] argue that the next step in handling "big data" lies in the algorithms that allow for data management, not in the storage systems themselves. In other words, the process of retrieving and storing data, not in the implementation or performance of the storage system or database itself. Therefore, the solutions proposed with strive to lessen the human component of the system by utilizing automation.

1.2.5 Subset Management

Additionally, this project aims to expand upon several methods focused on allowing for automatic generation of a data set consisting from a larger datapool. This will try to build on works such as [29] to allow for humanly understandable queries to select the desired subset. While generation of the data set, automatic sanity check will be evaluated as such procedures is part of the data pipeline. Such methods include: *Automatic image quality assessment* [30], which tries to eliminate images of poor quality and other flaws such as imbalanced data which might compromise the effectiveness of training performed using the data set.

Chapter 2
Background

This chapter will present the solutions proposed to improve the workflow. Firstly, all the theoretical background on the related subjects will be provided. Next, the issues of the current workflow will be identified and analyzed in order to then research and evaluate possible solutions for the separate issues. The solutions will be based on research in addition to the desired features of the final workflow from the company. Finally, the proposed workflow will be presented.

2.1 Theory

To fully understand the implementation, the techniques used for automatic data processing and categorization will be introduced. Not all of the covered techniques will be employed in the final solutions, but they are evaluated or mentioned as part of a solution in the conclusion.

2.1.1 Object Detection and Image Classification

Image classification is the process of classifying or describe the contents of an image (assigning labels) based on the content either manually or automatically using image processing [31]. To perform this process automatically using a machine- or deep-learning approach, processes can utilizes convolutional neural networks in order to process the image and automatically extract features presented in Section 2.1.2. The features are then utilized for comparison to the models learned features of known classes presented in Section 2.1.3.

The object detection method, on the other hand, aims to detect and classify all known objects in an image. So in contrast to image classification, object detection is not limited to only classifying an image containing a single object but tries to provide information about the location and identity of multiple objects in the image. This project will provide a proposed solution for creating either an image classification or object detection data set. While

the models utilizing each approach share many similarities in architecture, they differ in processing strategy as described below.

For object detection, there are several approaches to defining the location of an object. Specifying coordinates for the mid-point of the object, coordinates for a bounding box surrounding the object or specifying a contour/mask around the object in question [32]. The approach this thesis will refer to is the bounding box since is the most common input/output for object detection models [33] and for literature as stated in [34]. Networks such as Region-based convolutional networks (R-CNN) employ a strategy called recognition using regions [24] that separates the image into multiple predefined regions. Regions are then individually classified (using the image classification strategy explained above) in order to determine the location and nature of the desired object if it is known to the model.

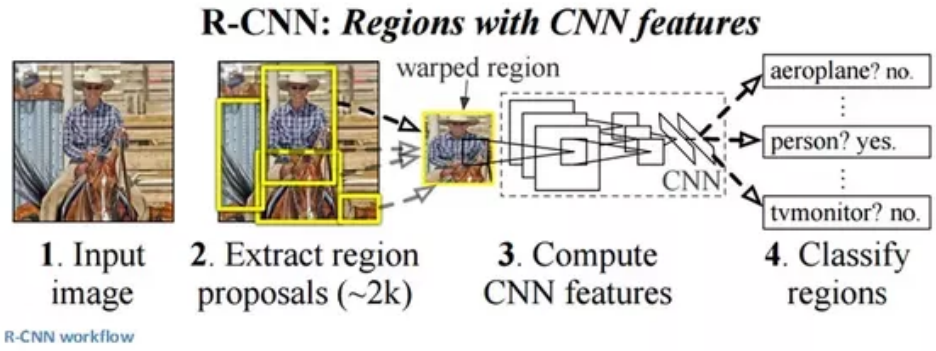


Figure 2.1: The "recognition using regions" paradigm of the Region-based convolutional network visualized. Source: [24]

As Figure 2.1 demonstrates, the R-CNN model will classify many regions individually to establish the bounding box of any object.

Other approaches such as the sliding window paradigm continuously classifies a portion of the image instead of fixed regions [35]. This makes model more likely to detect object that might be placed in a position the fixed regions are unequipped to handle. However, the the ability to detect object of varying sizes is limited to the specified regions as the regions are placed in fixed locations that are either included in the detection or not.

Figure 2.2: The "sliding window" paradigm of the Region-based convolutional network visualized. Source: [35]

In order to compare different object detection model, the most common method is using AP (average precision) or mAP (mean Average Precision) which is defined as the combination of precision and recall [36]. There are still issues with relying on mAp for object detection in video. While it fails to convey a variety of factors when it comes to object detection in video, such as the ability to continuously detect an object in order to track it and the latency of the detection (as a heavy delay might cause the detection bounding-box to appear out of sync) [37]. However, for still images, mAP is the standard for measuring models (shown in Equation 2.2).

$$P = \frac{TP}{TP + FP}, \quad R = \frac{TP}{TP + FN}, \quad AP = P * R \tag{2.1}$$

where:

mAP = "mean Average Precision",
AP = "Average Precision",
P = "Precision",
R = "Recall",
TP = "True Positive" (Predicted positive correctly),
TN = "True Negative" (Predicted negative correctly),
FP = "False Positive" (Predicted negative as positive),
FN = "False Negative" (Predicted positive as negative)

Eq. 2.1: Equation for Average Precision (AP).

For a prediction to be considered a TP, the area of intersect of the predicted bounding box need to be above a certain threshold as presented in Figure 2.3. To be able to properly compare the performance in relation to the precision of the bounding boxes produced by a model; the standard COCO-style mAp [38] is used. This measurement is the average of all AP's for the 10 threshold values in the interval [0.5, 0.55, 0.6... 0.95] in order to provide a more nuanced analysis of the performance.

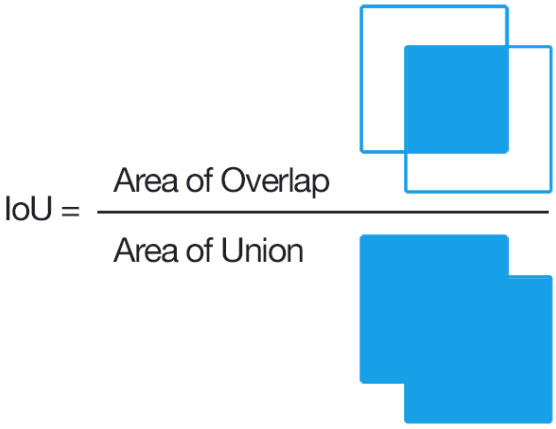


Figure 2.3: Visualization of the Intersect over Union (IoU) between a predicted bounding bow and the actual bounding box. Source: [39]

IoU adds an indicator of how accurate the predicted bounding box location of any correctly found object is. The AP is then evaluated for each class which produces the final mAP for the entire model as shown in Equation 2.2.

$$TP = \sum_{n=1}^{predictions} \begin{cases} 1, & IoU_n \geq TRH \\ 0, & otherwise \end{cases} \quad , \quad mAP = \frac{\sum_{n=1}^{classes} AP_n}{classes} \tag{2.2}$$

where:

AP = "Average Precision",
mAP = "Mean Average Precision"
TP = "True Positive" (Predicted positive correctly),
IoU = "Intersect of Union"
THR = "Threshold for IoU"

Eq. 2.2: Equation for Mean Average Precision (mAP).

Depending on the use-case, the desired mAp can vary. If the use-case of the model requires precise tracking, the mAp metrics with high IoU thresholds are considered more important as the bounding box needs to be precise for the object to be properly tracked. Whereas use-cases when the only priority is to simply detect the presence of an object, mAp with a low IoU threshold, or even plain precision (P), may be a sufficient metric for the intended purpose.

2.1.2 Feature Extraction

In order to effectively categorize an image, an analysis must be performed of the content of said image to collect the defining features [40]. Human perception is adept at this analysis as a human can quickly analyze an image and provide information as to the color, texture and shape. However machines are not limited to humanly understandable features and image classifying machine learning models utilizes feature extraction techniques to effectively classify an image. So if a specific feature in the images is desired in a data set being crafted by a human, it would be greatly benefit the human if the feature was expressed in a manner that the human can understand. This would provide a greater ability to customize a data set based on these features. Therefore, the solution will strive to include techniques to extract features for both humans and machines.

Image features for humans

Techniques to classify data in humanly comprehensible term is commonly referred to as bag-of-words (BoW) where an attempt is made to produce a number of words that can effectively describe the data. In the case of images, these words can be more easily comprehensible for humans than raw numerical data.
Generally, there are two types of BoW. Firstly, local/conceptual, which uses techniques similar to that of the object detection machine-learning networks by extracting the features of regions. The regions that contain clearly definable features can be categorized by techniques such as saliency or Scale Invariant Feature Transform (SIFT). These techniques are primarily

used by computers and neural networks to describe features and will be elaborated on in Section 2.1.2 which elaborates on image features used in the ML approach. While the features described by the techniques may be humanly describable through words. It is more commonly the case that the individual feature can only be comprehensible by a machine.

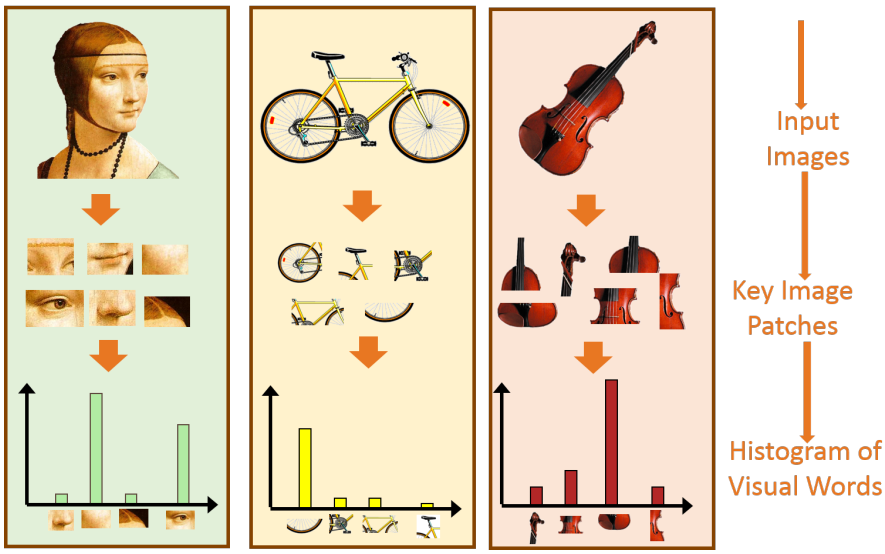


Figure 2.4: Simplified example of extracting key patches for word categorization of the image. Source: [41]

The second type, contextual, instead uses the background features of an image in order to express the context of the image, in other words, the "scene". By using the features of the background to categorize the image with other scenes with similar background features the method can effectively name the scene as described in [21]. Examples of this can be seen in Figure 2.5 below.

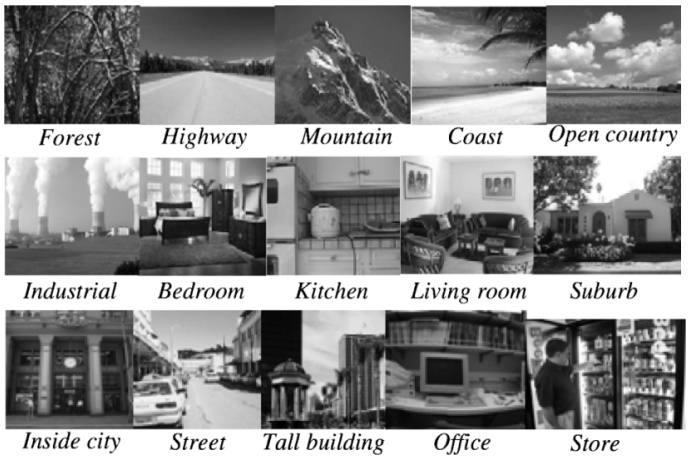


Figure 2.5: Describing words generated using the related features of the images context. Source: [21]

The words produced by such procedures can be added to the images metadata and utilized to categorize the images in order to allow for human inspection as exemplified in Section 2.1.5.

In summary: humans are generally much more proficient in recognizing and describing the context of an image while machines can generally more clearly define the detailed features of an image. In both cases, humans are generally useful in describing images to other humans and machine features can be interpreted by other machines. The challenging part is the connection between the two.

Feature representations for machines

There are many different feature descriptors that can be used. One of the most common is the Scale Invariant Feature Transform feature descriptor [25] due to several factors such as it being one of the oldest feature descriptors. It also is relatively simple to compare SIFT feature descriptors [42]. SIFT is an extremely broad area of the feature description and can be summarized as techniques to summarize an area of an image into a description of the areas feature. The descriptor is created by normalizing the vectors (and by extension the matrix) defining the area to detect local minima and maxima of contrasting parts of the image. The any local minima or maxima is the midpoint of a feature descriptor and is known as keypoints [25]. Once a keypoint has been identified, a description of the area including and surrounding will be established through pooling the matrix through a kernel, producing the SIFT description which will include descriptions of the areas features. Color content and local gradient direction which can describe edges and lines can be described with this description. The descriptors can then be compared simply by cartesian distance to find a similar feature descriptor [25]. These descriptions may also be clustered through different means of nearest neighbor search strategies such as k-means or Best-Bin-First method [43]. There are also other effective methods for extracting descriptions of features such as ORB (Oriented, Features for Accelerated Segment Test and Rotated Binary Robust Independent Elementary Features) which aims to increase performance of the feature detection algorithm while also considering orientation and rotation of the feature descriptions [44].

While image classification and object detection ML models use custom feature descriptors and not SIFT, they do use a similar approach to feature extraction as described above. That is by matching features with similar features to that of the training data in order to be able to effectively predict the class of an image, given that it recognizes the features from a certain class in the training set [45]. The Figure 2.6 image visualizes how a R-CNN model uses this techniques on various regions of an image to establish the possible identity of the region and, in the process, the bounding box of the object inside if present [23].

Saliency

Saliency detection is an image processing technique that aims to determine contrasting regions of an Image. In other words, objects that would stand out to a human [22, 47]. This can be done by considering many aspects of the the image that differs. A simple example in terms of ordinary brightness contrast would be a bright object in a otherwise completely dark image. However, alternative contrast that could be taken into account are other attributes such as edges, color and sharpness amongst others.

Below is an example of fine-grained saliency analysis performed on an image. This yields

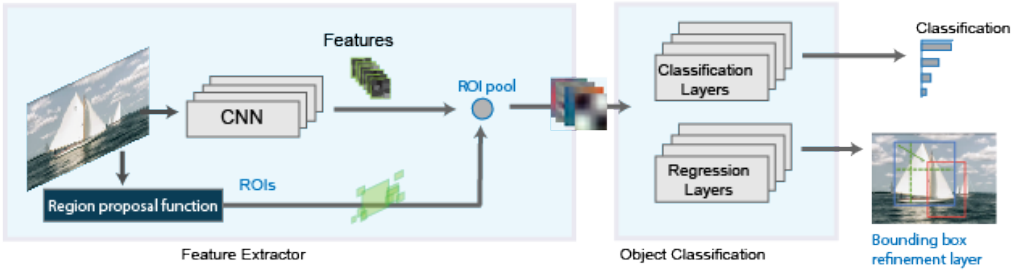


Figure 2.6: Example of how the fast-rcnn model structure utilizes feature extraction to classify objects. Source: [46]

a salinecy map which describes every pixel by the amount it contrasts its surrounding pixels in terms of brightness, color and edges [48].

Figure 2.7: Fine-grained of an image next to the original saliencymap visualized.

2.1.3 Model Training

A neural-network that has had its parameters tweaked automatically using a set of data in order to accomplished a specific task is generally referred to as a ML model and the procedure of automatically tweaking the parameters as training [49]. There are a number of approaches and algorithms that are utilized to train ML models. But in regards to image classification and object detection, nearly all object detection frameworks use algorithms that stem from Stochastic Gradient Decent optimizer (SGD)[50] which is an iterative technique to optimize a function. In case of model training, the amount of incorrect predictions of the training data set act as the loss function to be reduced or optimized and the parameters of the model will be adjusted in order to accomplish this. This means that the model will gradually gain an improved ability to make accurate predictions on the training data set and thereby similar data. Section 2.1.1 that explores object detection and image classification elaborates on what is considered an error in this case but can be simplified as images classified correctly. Since the algorithm relies heavily on the input data, the importance of quality data increases as the algorithm will automatically adjust the model incorrectly in case of erroneous data.

Transfer Learning:
Since the training a large scale object detection model is extremely computationally demanding, many of the common frameworks for using object detection utilize transfer learn-

ing [51]. This allows machine-learning models trained for a specialized task to be repurposed for performing a, similar, but not identical task [52]. In the case of image classification, this technique is commonly employed to limit the necessary amount of training since objects to be recognized generally share numerous features. Examples of such features are: outlines, contrasting colors to the background and uniform color schemes.

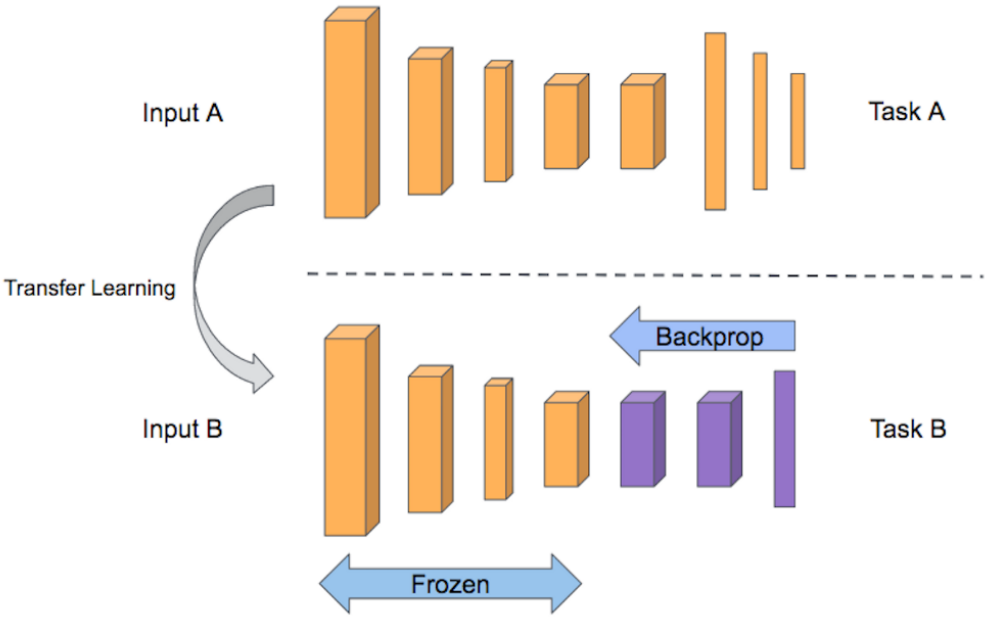


Figure 2.8: Simplified overview of a classifier models layers when performing transfer learning. Source: [53]

Figure 2.8 visualizes how only a portion of the entire model is actually trained while the majority of the model is frozen when using transfer learning since the heavy part of the training has already been performed. This yields steady base layers, adaptive classification layers and much faster training of the model overall.

However a major risk when using transfer learning is the risk of overfitting/bias [54]. Since only a portion of the models weights are adjusted during training they require more drastic modification to adapt to the features of the training set. Therefore, if the training data only consists of a small quantity of images and thereby small quantity of features, the model will recognize those features and those features only. For example: if a model is trained to detect birds, but only given images of birds in broad daylight, it might not be able to recognize birds in rain och dusk. A method to negate the negative effects of this is through introducing augmentations to the training images in order to make the model more robust to the changes in the image that should not effect the result. This is explored further in Section 2.1.6. The result would be the same if one class of objects is over-represented is the data set compared to other classes, it would negate the other classes if favor of the over-represented class. This means that variety and quantity in the training set is required to negate said bias [54]. Erroneous data is also more harmful as a result. Due to the aggressive adjusting of the weights, data that is causing the model to learn incorrect features are more detrimental to the accuracy of the model. In summary, data sets used as training input in transfer learning require a

large data amount and is gravely affected by erroneous data.

2.1.4 Version control of data sets

A standardized practice in the software development environment is to implement version control into the development of software in the form of a Version Control System(VCS) [55, pp. 1–2]. This does not only provide a distribution and a backup for the codebase if the VCS endpoint is on an external server, but a system for safely implementing changes as well. However, the version control of image data sets provide a greater challenge as the files of which the data set consists of are large binary files. This makes distribution of the data set problematic as the bandwidth and storage space of the recipiant machine of the data set might cause a issue. Therefore, many solutions to this offer the option to instead version control the metadata of the any large binary files instead of the binary files themselves [56]. This remedies the restrains put on storage and bandwidth as the metadata is commonly in text format which is of insignificant size in contrast to the binary files.

2.1.5 Image Metadata

Metadata can be considered as any information about an image that is provided alongside the image. It can generally come in two forms, either externally as a description alongside the image, or embedded inside the image data. These two types of metadata will be discussed from the image perspective.

Embedded Image Metadata:

Embedded metadata is information about an image which is included within the binary representation of the file. Formats for such metadata includes: “Exchangeable image file format” (Exif) [57] for jpg and wav files, “Extensible Metadata Platform” (XMP) which is most common in png files and many others. The content can be split in to three main categories: technical, descriptive and administrative metadata.[57]. Examples of the extracted metadata for two images are provided in Appendix A and will be discussed here.
Technical data includes information such as how the image shall be interpreted by software and hardware by specifying color profiles, specification on the camera that captured the image and other visual specifications of the image like contrast, saturation and sharpness. Additionally, applications that may have modified the image often leaves a stamp of the interaction like we see in the example, there is a record that the image has been modified in the image manipulation program Adobe Photoshop [58] roughly half a year subsequent to capture.
Basic descriptive information may also be included as it allows for the image to be more effectively categorized and utilized like we see in the example. Since the metadata usually is kept quite short as Exif is limited to 64 kb [59], long descriptions or other verbose data is commonly put as external metadata. Additionally, it allows for more convenient parsing.
While not present in the example in Appendix A, the image may also contain administrative metadata such as copyright information and other credits meant to inform about legality or other useful information.

External Image metadata

This type of metadata may contain anything, such as an image description, as long as the information is closely bundled with the image [60]. A common approach is storage in a separate file but the metadata also stored in a database or use other storage solutions. If it is kept in a separate file, the related file has the same name with the exception of the file extension. In relation to this project, the external metadata of focus is descriptive information about the Image, human readable or simply binary characteristics, that allow for categorization of the data. This type of data is covered in Section 2.1.2 which explores feature descriptions. A strategy to collect the data categorized by extracted features, outlined in [61], uses the non-human understandable representations of the features to find similar features in other images metadata and thereby connect the images.

Another import type of metadata that is commonly located in an external file are "image annotations" which describes the name located in an image for image classification and all the objects of interest for object detection [62]. These annotations are crucial for the process of training detection machine-learning models as models trained using object detection frameworks are trained using various methods of stochastic gradient decent. But in order to accomplish this, the training data provided will need to specify the correct annotations as specified in Section 2.1.3 for use in ML model training. Listing 2.1.5 is the specification for the YOLO image annotation format [63] which is used to train model using the "You only look once" (YOLO) object detection framework [64]. Listing 2.1.5 is an example of a annotation file, we can see that the image that the annotation specification is based on contains two objects, one smaller object of class 2 and a larger object of class 5. Classes are specified in a separate file for YOLO.

Code Listing 2.1: Yolo format specification.

```
<object-class> <x> <y> <width> <height>
```

Code Listing 2.2: Yolo format example.

```
2 0.3 0.3 0.2 0.2
5 0.8 0.2 0.5 0.5
```

2.1.6 Image Post-Processing

While post-processing is not a main focus of this thesis, it is still a necessary step in the workflow to support as it is crucial to the preparation of a data set. Therefore, the following

topics will not be featured extensively and any implementation made of the post processing step will not aim to improve the current related process. It will instead focus on necessary functionality rather then be the subject to improvement.

Image augmentation

When performing training on a model using a small to medium data set, a common issue is over-fitting or bias in the model. This might occur when a model is trained extensively on the same data which might cause the weights of the model to heavily favor the images in the training data set and become oblivious as to the features of any other images that does not exactly match the features of the training data set [54]. In order to combat this issue; a common approach involves augmenting the images of the data set in order to create several copies of each image [65]. Each copy is altered in comparison to the original while maintaining the core features of the image in order for the model to be able to learn and focus on the identifying features and disregard the irrelevant features. Examples of such images can be found in Figure 2.9 below.

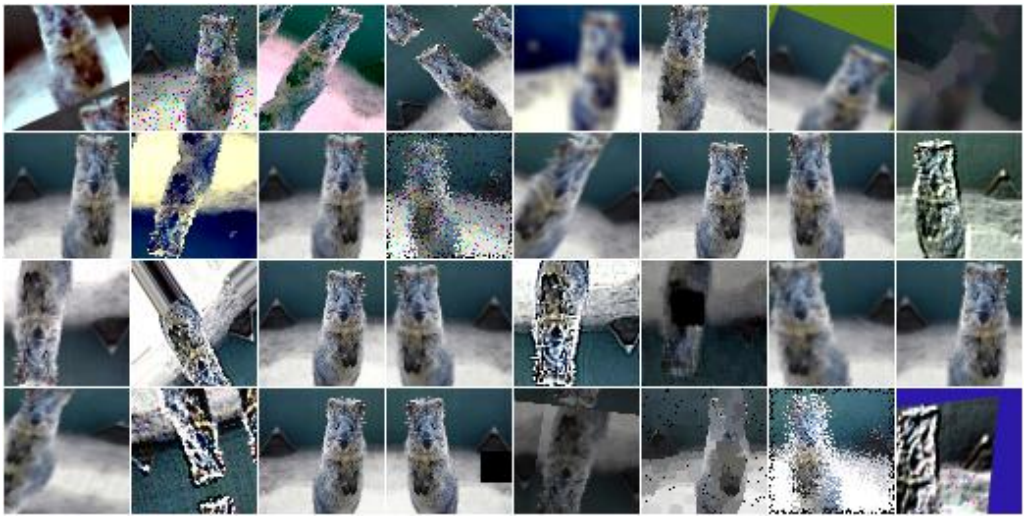


Figure 2.9: Augmented images created from a single image to artificially extend a data set. Source: [66]

A practical example of this would be to alter the brightness of an image. As in many common use-cases, the model should not let brightness affect the evaluation of the identity of an object. Otherwise, the evaluation might be affected based on the time of day the provided image was taken on or the amount of light in the room.

Verification

Before any training is performed on a data set, it is necessary do perform a sanity check to catch any obvious potential flaws in the data set. This includes more obvious error such as: incorrect metadata for an image, obscured or irrelevant images and excessive classes which might hurt the accuracy. Another common sanity check to perform is for data balancing issues as the weights of the model can become over biased towards certain classes if the data set is heavily unbalanced in the number of data entries for several classes [67]. Therefore, it is important to exclude data from classes where there is an overwhelming amount of data, despite the importance of a large amount of training data. This issue can also be negated trough only augmenting the less featured class in order to gain a proper balance through artificial means [68].

Compression

To reduce the amount of space the generated data set will utilize on the disk, the final data set has to be compressed. The conventional format in linux/GNU-based software is the basic tar format [69] that utilizes a single file to store an archive in reduced size containing the files. However, some frameworks such as Tensorflow object detection API, which is used by the company, provides support for the final data set to be prepared in a custom format. Such a format is tensorflows TFRrecord [70] which is used in order to optimize training by limiting the need to read from individual files during collection of input from the data set and only storing relevant information. This specific format is a compression of the images in additions to their required metadata such as resolution and any classification or annotation.

2.1.7 Data Management System

To be able to store and use a large amount incoming data, a Data Management System (DMS) has implemented is often used to provide a workflow for the data [71]. The DMS should handle both extension of the current data pool and the management of the current data by making it easily accessible for any user trying to interact with data by structuring the storage.

Databases and queries

To effectively combat many of the problems that come with Big data, namely volume, management and lack of data overview, it is crucial for any system dealing with said problems to implement a DMS [71]. This DMS is used to store and query all in incoming data. A query, in relation to DMS, is a request for desired data using specified parameters [72, p. 10]. These parameters can be of any form as long as it is supported by the data storage system. If the data is stored in tables, as in the relational DMS [73], queries can be the names of entries on rows and contents in certain tables. But in other structures, it could be categories or various attributes the data might have. Queries can also be extended by the system to include data the query did not specifically ask for, but may be implicitly desired by the user. This can

be done using lexical extension by analyzing keywords of the query and including synonyms (for example: including human, man, woman in a query for person) or related words (for example: including wood, forest, shrub in query for tree) [26]. Figure 2.10 shows how a query can be extended using keywords in order to provide the user to effective query the database for data using searches and not query the exact data but a description of the desired result. In relation to images, unlike numerical data, image data can easily be described by a human and there is potential to find similar images by examining related features and similarities as expanded upon in Section 2.1.2.

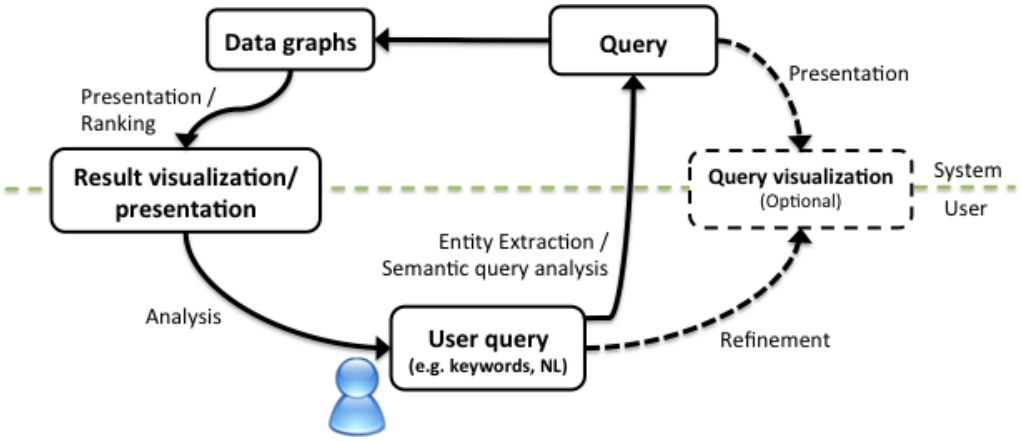


Figure 2.10: Overview of the data retrieval process in a data management system. Source: [74]

Data modeling

To effectively provide the desired answer to queries it is paramount to structure the data in accordance to the desired query system [72, pp. 23–34]. Otherwise, providing a desired result to any query will be impossible if the data cannot be found. There are many types and implementations of the different kinds of databases which are suited for specific types of data and specific requirements for the designated data management system. The data contained in the database is stored either in-memory (in RAM [75]) or on-disc with each method having its advantages and disadvantages. While in-memory databases gain the advantages of read speed and efficiency in data-cycles, which is especially useful when having to perform continuous operations on the data stored in the database, it is limited by the amount of memory on the actual host [76]. Such databases are commonly used for caching and configuration as the size of the data is the limiting factor.

Image Storage

Images are not as easily stored as numerical or string data as it is considered a Binary Large Object (BLOB) data [77]. Not only is the storage space required very large, since images are often stored as an unstructured object, the data cannot be easily sorted [78]. This problem is exacerbated due to the fact that many database types, which can save binary data, has a size limit the binary objects which means that it has to be stored in chunks [79]. This makes the process of reading and writing images to such a database extremely slow since it has to

make a read for every chunk. A common approach is to instead use a BLOB storage [77] or file store to store the images only, then store a reference or URL link to the image location in another database, for example, a relational database. This technique allows for the image data for become more easily sortable and accessible [78].

Hierarchical Storage Format

Hierarchical storage structures uses a data model that is very similar to file systems as data is stored in groups and subgroups [80]. It was originally developed to provide a storing data that requires more flexible structuring which the relational storage methods popular at the time (circa 1990) did not have. The problem being that queries for large subsets were unnecessarily inefficient since all the entries in a table had to be redundantly filtered in order to provide the results when they could be stored in groups instead [81]. Since image data often requires flexibility, several solutions opt to use this structure since it allows for storage that uses features for identification where flexibility is prioritized [82, 83]. A popular options which uses this data management system approach when it comes to storing large data sets is HDF5 [84]. Hierarchical Data Format #5 (HDF5) is a flexible storage format which attempts to provide a compromise of the normal storage options. It is built for high volume, complex data and flexibility in store options. Two useful features when it comes to using the format for image data is the ability to entirely or partially load the database into memory, which allows for quick operations on the data, and built in support for metadata for bootstrapped metadata which can be stored alongside any data entry [84].

Chapter 3
Methodology

This section will provide the method of which this project will utilize in the endeavor to improve the current workflow. The current workflow needs to be analyzed, research of improvements needs to be performed and proposed solutions needs to be tested. Finally, the proposed solutions shall be evaluated on the basis of the performed experiments and theoretical implementation in comparison to the desired result and the current workflow.

3.1 Thesis Goal

This thesis will attempt to answer the following research questions after presenting the findings:

1. What are the major issues with the management of machine learning image data sets at the company?
2. What solutions can be proposed alleviate the identified issues?
3. How do the proposed solutions affect the effectiveness of the workflow at the company?

The answer to these questions will be presented in the Analysis Section 4 and summarized in the Project Goals Section 5.3 of the Results 5 to then be discussed in the Discussion Section 6.

3.2 Assessment Methods

As previously mentioned in the Introduction Section 1, the previous management of the current workflow at the company has prior to this project assessed the workflow to be unfeasible to be the large amount of resources required to operate. The resources referred to is

namely the amount of labor hours put in by the employees and the required competence of said employees to effectively operate the workflow. Therefore, the measure of assessment of which the research questions will be answered in the context of is primarily: the total amount manual time of the workflow which can be eliminated and to what degree the required competence can be reduced. However, not all solutions will be possible to evaluate through this measurement as issues unrelated to time and competence will most likely be found. The solutions to the issues will be evaluated based on the nature of the issue. In such cases, the assessment will primarily be whether the solutions alleviates the issue or not.

3.3 Approach

In order to answer the research questions posed in Section 3.1, this project will employ the methodology of that of the action research [85]. This was decided due the nature of the workflow which requires both highly theoretical and practical understanding. Both of which can be gained through a combination of analysis and practical experimentation in order to understand the issues and propose solutions. As described in [86]: the action approach is highly beneficial for complex information systems where close analysis of the issues and experimentation can benefit a project with the goal of improving the system by closely iterating experimental solutions and evaluating them. The focus will foremost be to find solutions for the issues located in the most crucial component through research and experimentation and continue to apply the same process with the less impactful components until a complete workflow is achieved. The crucial components being the components with the topmost potential for performance gain in regards to the assessment method stated in Section 3.2. Generally, the components with the highest labor cost and competence requirement.

Practically, this will be done by firstly identifying the causes of inefficiency of the workflow and grouping them into problem areas as described in Section 3.3.1. Each component can then be analyzed individually in order to identify the issues in each problem area as described in Section 3.3.2. Once issues have been established, research will be employed proposing a solutions to the each problem area which attempts to alleviate the issues. This will be done trough research and experiments as described in Sections 3.3.4 and 3.3.5. Based on the research, experimental implementations of components will be performed to the extent possible as described in Section 3.3.5. Finally, the approach for the evaluation of the experimental and practical implementations will be presented in Section 3.3.6.

3.3.1 Problem Area Identification

In order to properly propose a solution, it is paramount to analyze the current workflow closely and evaluate which components are the cause of the specified problem. This includes investigating the desired use-cases of the company and comparing them to the other typical workflows from other similar systems. The important components of the workflow will be identified and grouped into problem areas where an area can be defined as a separate process where a solutions can be proposed for the area independently from the other areas. This process is visualized in Figure 3.1 below.

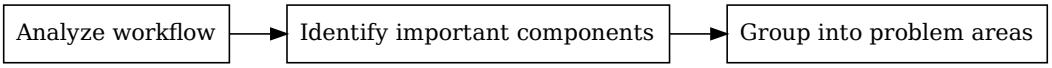


Figure 3.1: Sector identification process

3.3.2 Problem Area Analysis

Once the areas have been isolated they require individual analysis to access the requirements for the desired solution. A proposed system for performing this analysis is to tally the different tasks on the sector [87]. Then specify the conditions and cost necessary for the task to be completed. Conditions in this case could be potential blockers of the task and cost is the time in addition to resources associated with said task. This will provide the necessary scope required by the evaluation process specified below and allow for prioritization of the research effort. In other words, to clarify the important issues to address and what solutions should be evaluated, as non-essential parts of the process do not solicit equally comprehensive evaluations and may even be excluded from the proposed workflow entirely if the effort required is to vast or assessed to be out of scope of the project. In this context, non-essential part are components with less potential for improvement or components which has less impact in terms of time and competence. Such as the performance of the storage, while it is important for the storage to be easily manageable, the read and write performance is negligible compared to the speed of the internet connection to the webserver itself. The steps of this process are visualized below in Figure 3.2.

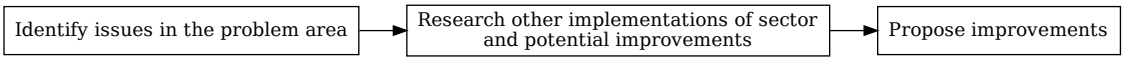


Figure 3.2: Analysis to be performed for each area

3.3.3 Solution Cost-Benefit Estimation

The next step is to analyze the scope and cost in terms of time of the potential solution. The solution may not be so costly to implement that it outweighs the potential benefits of the solution. However, a prioritized property of any solutions evaluated is the automation potential as the projects overall ambition is to limit the necessity of human interaction in the

process. As specified by the "AutoML" ambition 1.2.2, automation is the prioritised objective. This process is visualized below in Figure 3.3.

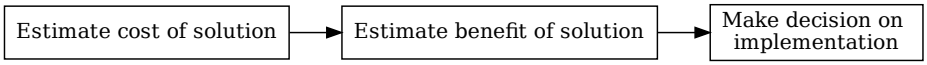


Figure 3.3: Estimation for each solution

3.3.4 Research

In parallel with the workflow evaluation, obtaining information about the technologies and processes involved through research is vital to the project. This research is performed by first studying similar systems and investigating techniques that could be employed to achieve the desired result. Then, estimating how sufficiently the researched solution would satisfy the requirements, how expensive in terms of time it would be to implement and finally if the solutions is feasible, to include in the final solution. All the necessary theory and knowledge should also be researched in order to be able to make a qualified judgment on final implementation choice. If no such judgment can be made, or if two potential solutions are considered equal, the solutions will be subject to evaluation through experimentation. The research shall be conducted by seeking relevant research journals and workflow implementation documentation in the field. Additionally, desired features desired by the company and suggestions made by the company should be considered. This is a critical step as the reason for creating the proposed workflow is foremost to aid the company. The output of this step in the approach will be a specification for a proposed workflow which will act as a guideline for an experimental implementation of the entire workflow.

3.3.5 Implementation

Subsequent to the completion of a complete theoretical proposed workflow, the goal of this step is to implement the proposed workflow. Based on the research performed in the Research Section 3.3.4, the components will be subject to attempted implementation. If the the implementation cannot be performed or is deemed to not function according to desired performance in regards to the methods of assessment, the proposed workflow will be adjusted based on the experiment to achieve an implementation that does produce the desired result. To the extent possible, the components to be implemented in order of potential performance gain, meaning the component were the major causes of inefficiency are located, until the complete workflow is implemented. The implementation is primarily performed as a proof of concept, the goal of the implementation is not creating a perfectly optimized solutions but instead a working solution that alleviates the identified issues to the greatest extent possible and can guarantee the validity of the proposed workflow.

3.3.6 Evaluation

The action research approach generally used small iteration with evaluation of experimental implementations to receive feedback continuously [85]. This project will apply a variation of this where research is performed to reach a potential solutions for the cause of an issue as specified in the Reseach Step in Section 3.3.5. An attempted experimental implementation of the solution will the be performed. If deemed successful in regards to the Assessment Methods of Section 3.2, the solution will be solidified as part of the proposed workflow. If deemed unsuccessful, an another alternate solution will be found to replace the ineffective solution through additional research and be subject to the same process. This will be performed for each issue within the scope, starting with the must crucial issue in terms of potential performance gain, and will continue until a full implemented workflow is achieved which can then be evaluated separately. As stated in the Assessment Methods Section 3.2, the focus of the evaluation in this project is primarily to evaluate how much of the system now is fully automated, thus removing the required time and required human competence to operate the workflow.

The experimental implementations will only act as a proof of concept and will not aim to be completely optimal in terms of performance. This means that there are several threats to validity in terms of uncertain effects of the proposed implementation such as the possibility that the final proposed solutions are not possible to implement optimally, are excessively expensive to implement or are simply ineffective. These possibilities and risks will be explored in the Threats to Validity Section of the Discussion 6.2.

Chapter 4
Analysis

The analysis will consist of two parts. Firstly the issues with the current workflow will be analyzed by using the methodology presented in problem area analysis in Section 3.3.2 in order to answer research question 1 in Problem Analysis Section 4.1. Based on the issues identified, solutions will be proposed in Section 4.2 in order to answer research question 2.

4.1 Problem Analysis

This section will use the following structure for each subsection: Firstly, the problem area potentially prone to improvement will be introduced in the overview. The current system for the sector at the company will be outlined and potential issues clarified. The value of the sector will also be estimated here as a more trivial part of the workflow will naturally have solutions that is less prioritized for optimization and may even circumvent the need for any major research. For example, the final solution proposed will not incorporate any security due to it being a part of the workflow of no concern to this project. Secondly, the required and the desired features will be discussed. All the features that will be considered out of scope will also be mentioned here to allow for a potential discussion on improvement potential during the evaluation. Finally, the potential solution(s) will be discussed. This includes: any related research, similar solution for the problem area implementations and potential improvements that can be found. This process is presented in Figure 3.2

4.1.1 Data Upload

This is first part of the process, albeit the part with the least requirements in the entire workflow but still a necessity for the process. It is the ability to upload one or several images to a central location, preferably from anywhere. This should act as the entrypoint for the entire application since all the images received by this sector shall be stored and subjected to the rest of the process as it is implemented.

Required and Desired Features

The requirements for this part should be an application that allows for image and video uploads. Ideally this should be done via a stream since the media files may be quite large and will be continuously uploaded in the use-case of the the company. This means that file-size is a potential issue, so compression before the upload and decompression after upload is a much desired feature to limit the strain on the connection since bandwidth might be a bottleneck. As with all plain file uploads, security is a crucial point of interest if this will be used in an environment open from outside networks, but is considered outside the scope of this thesis due to time restraints and lack of relevance to the automation.

4.1.2 Image Pre-Processing

In this area, the images shall be processed in order to extract and gather all the information required to store the image in a category and to include the image in a data set. Currently, this process in done manually for new data sets where a model has not yet been created to partially process the images automatically. This includes manually opening the images in a software used to label images, draw the bounding boxes of the objects and create the meta-data specifying the object.
Assuming the worst case scenario where no prior data exists, there is a high number of objects in the image with various sizes and classes, every object in the image needs to be labeled and cropped into an individual image in case an image classification data set is needed; this processing will most likely take at least 5 minutes per image due to the amount of objects that requires processing. This means that addition of a batch of 100 images will take more than 8 hours of manual labor to process the batch. The human factor amplifies this problem due to the nature of the occupation. It requires that simultaneous handling of a large quantity of data and is extremely repetitive which can massively increase the chance of human error. To remedy the mentioned issues, an common attempt might be to distribute the labor across several employees. However, this could create additional issues of its own, mainly the management of all the outputted data due to the difference in processing strategies humans naturally have. The same object might receive different identifiers, bounding boxes might have different amounts of padding surrounding the object and even the outputted files might be places in different structures. This could result a greater amount of added overhead labor cost due to the amount of constant data management or data reconstruction required after the process.

When a model has been created, object can be automatically detected and labeled but since the model is constantly improving, the amount of images that are completely correct in the beginning of this process will be very few [54]. This is due to the low accuracy of the model the small amount of of training data will cause. This means that every image has to manually verified, and potentially corrected, to avoid pollution of the data set with erroneous data.

Required and Desired Features

In an ideal solutions for this area, any incoming images would be automatically labeled without previous knowledge of the source, content or desired objects to annotate. This is not feasible since the desired annotations and objects to annotate is completely subjective to the current use-case but it will be the main goal of the entire workflow. Not to mention that without prior knowledge or data related to the image content, any method involving ML that requires training can be ruled out since training can not be performed without initial data. A more feasible goal is to extract any necessary data required to cluster the images based of different properties. This would mean that the data set could be generated by only specifying the mutual properties of the images. Therefore, the solutions to this sector will be closely related and dependent to the implementations of the subset generation problem area Section 4.1.4.

4.1.3 Storage

Naturally, all data that enters the workflow needs to be stored and accessible by the other components of the workflow that utilize the data from the data pool. This is currently done by storing the media files as plain files on a common server categorized by source. However, this introduces a major issue related to the management of the files. Particularly as processed and unprocessed files are stored together, desired data is hard to collect and process as there is no clear system for categorizing. Often causing data to be copied and stored in multiple locations simultaneously as data sets are created.

Required and Desired Features

Requirements for the storage solutions is that it can be accessed from multiple tenants and that no data should be stored multiple times. E.g if two data sets are stored with shared images, all images should be stored only once. But it should also be able to support all the features desired by the solutions proposed in Sections 4.1.4 for subset generation and 4.1.2 for image pre-processing to allow for storage and sorting based on the data provided and desired by said sections. For example: if the proposed solutions for subset generation relies the use of a certain feature descriptor for generating the subsets, the data storage solutions must be capable of handling the proposed feature descriptor.

4.1.4 Subset Generation

To take advantage of all the data collected, a data set has to be generated. This data set should consist of a specified subset of the large data pool since the desired classes for a training set change depending on the use-case for model trained using the data set. Currently, this is done by manually collecting the desired data in the plain files, copying the files into a separate directory and finally removing the unwanted classes from the metadata. Or in other words,

manually altering the metadata to fit the specifications of the model in terms of format and content. This process is extremely time consuming as all data has to be collected manually to customize the data set according to the desired specifications. As mentioned previously this requires a great deal of knowledge and attention to detail as the data set is very vulnerable to pollution by any errors. Another problem is the transfer and recreation of the complete data set. Since the data set in created manually, it is near impossible to recreate in from the source and sharing it means manually distributing the files. This is a big issue since the size of the complete data set could be substantially large. When performing augmentation, this problem is exacerbated since the size could potentially multiply by a factor of five up to twenty in cases where a large number of augmentation techniques are employed. Furthermore, since there is currently no method check the methods of augmentation used, it becomes increasingly difficult to monitor the content of the data set. Thus, there are three issues which should be the goal to to remedy for any proposed solution. The time consumed by manual collection, lack of content overview and the copying required to share the data set.

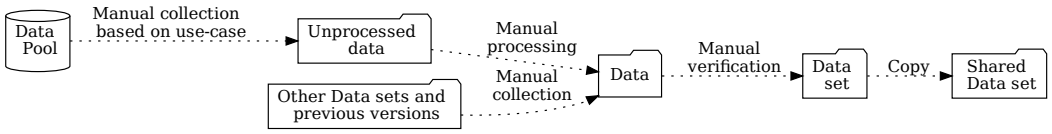


Figure 4.1: Subset generation for the current workflow

Required and Desired Features

To alleviate the issue of manual data collection, an interface which can provide a data set according to easily provided parameters is the aim of the desired solution. These parameters should ideally include as many alternatives as possible, as more options will increase amount of customizability. Some vital alternatives should include: classes (object types), amount of data and type of image data. Where type of image data should include specifying either object detection or image classification and ideally the format of the metadata provided as elaborated on in the theory chapter on external metadata in Section 2.1.5. These parameters could be shared to allow for regeneration on the same data set from the data pool. Thus, removing the need to transfer the entire data set by indirectly sharing the content. This would allow for easy sharing and distribution of the data set as you could simply share these parameters instead of the entire data set. In addition, if the parameters can be clearly read, they can provide the needed overview of the content that the system currently lacks.

4.1.5 Post-processing

After a data set has been created for training, it is necessary to perform post-processing in the form of extending, sanity checks and tweaks depending on the desired features of the finished model as outlined in the image post-processing Section 4.1.4. In the current workflow, the data set is already extended by generating augmented copies of images to artificially increase

the amount of data and thereby extend the data set. In terms of verification, the content of the data set is visually scanned for any potential errors and and images of subpar quality such as blurry images. Any imbalance in the data amount, e.g different amount of images, for each class is manually accounted for by reducing the amount of data of the classes that are disproportionate and extending the others to a higher degree.

Required and Desired Features

There is no hard requirements for this problem area but any solutions should strive to automatically make qualified assessments and fix any obvious faults the data set might contain. If the solutions does not succeed in doing so reliably, it should at least make recommendations to the requestee on types of actions that could improve the quality of the data set. Compression however, is naturally a feature that must be supported to circumvent unnecessary waste of storage space. Augmentation of the image is already performed automatically but it does lack two fundamental features: reproducibility and information. Since the augmentations are performed randomly using a distribution of values, it is statistically impossible to recreate this without additional information. For example, if the brightness of an image is augmented: it is common to apply values from a guassian distribution to avoid producing too many outlying images. The lack of information stems from the fact that the data set currently contains no information regarding on what extending was performed on it. This would provide a solution to the monitoring issue first outlined in the subset generation area analysis in Section 4.1.4.

4.2 Proposed Workflow

The proposed solution will be based on a combination of the research performed and experience gained from attempting to implement a prototype. Due to the magnitude of the project, the entire pipeline was not be fully implemented but the full specifications and techniques of the proposed pipeline will be presented in this section.

The prototypes backend will be implemented in Python which is one of the preferred languages for machine learning due to the existing frameworks and bindings to libraries for ML (such as Tensorflow [88]) and image processing (such as Pillow [89] and OpenCV [90]). This will not be optimal in terms of speed and efficiency but that is of no concern as stated in the desired features The frontend is implemented in javascript that allows for easy integration with both website functionality and back-end connections.

Figures 4.2 and 4.3 below are examples of images that could be uploaded to the process. Scene images, which can be seen in Figure 4.2, are images that could contain one or multiple objects and will be processed to extract any individual object. However, they will also be processed as an individual image in order to attempt grouping with other images depicting the same or a similar scene. Object images, which can be seen in Figure 4.3, are images only containing a single object and will need to be categorized into an existing group of objects or define a new group of objects.

Figure 4.2: Examples of "Scene images"

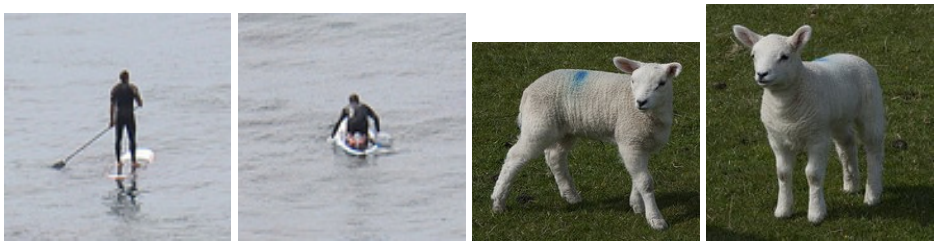


Figure 4.3: Examples of "Object images"

4.2.1 Data Upload

In order to support access to system for multiple users and devices uploads will be handled using POST requests to a webserver through a rest API. This will allow uploads of any content from multiple peers in any format to the process. Other benefits of this is that a webinterface can be added to allow for inspection and modification of the data.

Compression of the data before upload needs to be handles by the respective applications and/or users uploading to the service. However, when uploading to the server via a web browser, compression can be handled directly using a client side compression library such as JSZip [91].

Figure 4.4 is an overview of the data collection process where data is able to be collected from multiple sources and saved in the main data pool storage. This webserver was implemented in the prototype implementation. See Appendix C for the API-reference of the proposed API implemented in the prototype.

4.2.2 Image Pre-Processing

The proposed pre-processing will consist of three steps. Firstly, an attempt to process the image to extract all parts of the image which will contain an object. This will only apply to scene images. Object images will contain only the one object which makes this step redundant. Secondly, processing to collect all the data and feature descriptions necessary to group the image with other images of similar content and origin. Finally, utilizing the data collected to match the images into the existing groups which shares the most similarities.

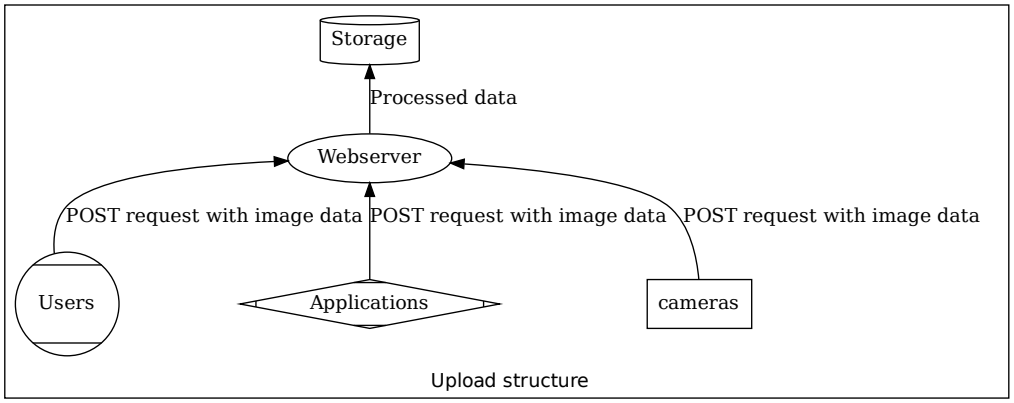


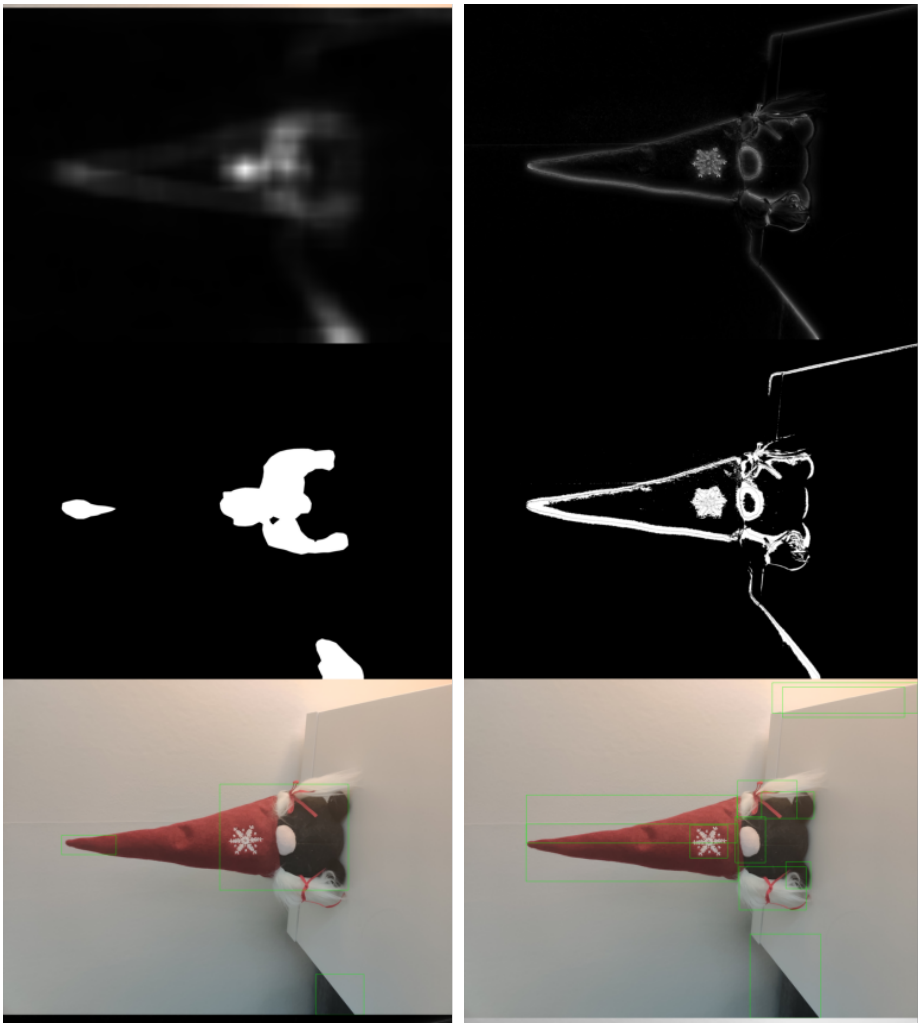
Figure 4.4: The suggested structure of the access to the process.

Object Detection

The first attempt to retrieve the object images will be to utilize a "efficientdet_d4" object detection model [92] with coco data set classifiers [93]. While this will commonly not yield any result except very common objects, since the model can only detect objects in the training data. However, most objects of interest are often subgroups of any of the more general Coco classes [93]E.1 and will give us the object image without the need for any additional processing. For example, while the coco cannot distinguish certain types of dogs, it could simply classify dogs as a dog which gives an important clue that two object both being detected as the same class are closely related.

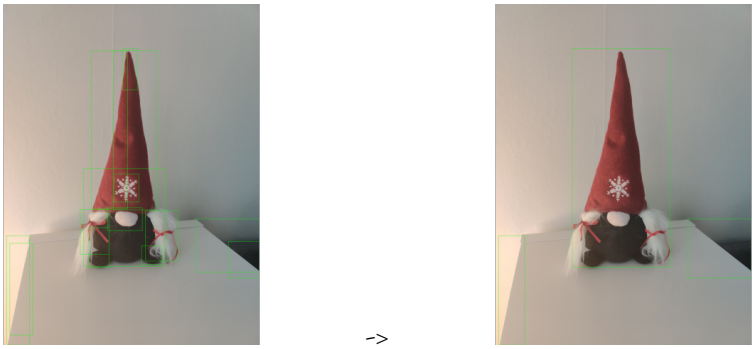
However, in the event of an object that does not match any common item class, several saliency maps will be created based on fine-grained and residual saliency plus corner detections by contrast [47, 48] as shown in Section 2.1.2 which elaborates on the used saliency techniques. In the Figure 4.5a, all steps in the saliency object detection is visualized. First the saliency map is created by residual saliency detection, then all the areas are distinguished by highlighting the areas with a high density of highlighted pixels and finally, bounding boxes will then be drawn to encapsulate the distinct regions that will emerge. This process is repeated for the fine-grained saliency as shown in Figure 4.5b. A collection of all the bounding boxes found through saliency detection is create and visualized in Figure 4.5c. The overlapping bounding boxes will then combined to remove bounding boxes that encapsulate the same object as visualized in Figure 4.5d, leaving only the final bounding boxes of found objects. However, bounding boxes that only appear in one type of saliency map will be ignored to avoid clutter that may occur from using many types of saliency maps.

In the bounding boxes of object found above, two of the resulting bounding boxes are clearly not objects. However they will only be grouped with similar images so it was judged to be better to be generous with the object detection and end up with more uninteresting images rather than risk omitting positive ones.



(a) Detection of objects using residual saliency

(b) Detection of objects using fine-grained saliency



(c) All found bounding boxes

->

(d) Bounding boxes combined

Figure 4.5: Combining bounding boxes from performed saliency analysis to obtain object bounding boxes

Data Collection

The main data that is subject to collection in this step is features and metadata. Metadata is collected through exif metadata extraction (see examples in Appendix A). However, images that has previously been uploaded then to have had their metadata stripped for privacy and integrity reasons [94] so this is mostly helpful when uploading original content.

The most helpful piece of information collected is feature descriptions, namely SIFT, SUFT, FAST and ORB feature descriptions [95] which will be saved and utilized in the feature matching step as defined in Section 2.1.2. Additionally, if the "efficientnet_b4" [96] models with the ImageNet classifiers [97, 98] yield any useful information such any ImageNet class that shares features, this will be saved as additional information. ImageNet has a greater amount of classes than Coco which is used in object detection data collection but can only be used for image classification. This is also the reason for different models utilized as efficientnet is utilized for classification and efficientdet utilized for object detection. Appendix A contains examples of the metadata which is collected and below is an example of features collected.

Code listing 4.1 below is a small example that extracts three feature descriptor types and outputs an example of the feature descriptor and the keypoint describing the feature descriptor.

```
''' Short example of the feature extraction only '''
import cv2
from pprint import pprint

# ...
# Get the image buffer from webserver or database
# ...

img_cv = cv2.imdecode(
    np.frombuffer(img_b, np.uint8),
    cv2.IMREAD_COLOR)

for d, detector in [('SIFT', cv2.SIFT_create()),
                    ('BRISK/FAST', cv2.BRISK_create()),
                    ('ORB', cv2.ORB_create())]:

    # find the keypoints and descriptors with the specified detector
    kp1, des1 = detector.detectAndCompute(img_cv, None)

    print(f'Keypoint example for {d}:')
    pprint({
        'Point':kp1[0].pt,
        'Octave':kp1[0].octave,
        'Angle':kp1[0].angle,
        'Response':kp1[0].response,
        'Size':kp1[0].size
        }
    )
    print(f'Descriptor example for {d}:')
    pprint(des1[0])
    print()
```

Code Listing 4.1: Example of feature extraction

```
'Keypoint example for SIFT:'
{'Angle': 0.12317657470703125,
 'Octave': 3801599,
 'Point': (3.9903314113616943, 166.71286010742188),
```

```
 'Response': 0.017094777897000313,
 'Size': 1.892791986465454}
'Descriptor example for SIFT:'
array([ 51.,   9.,   0.,   0.,   0.,   0.,   0.,   0.,149.,149., 26.,
         2.,   0.,   0.,   0.,   0., 29., 73.,149., 94.,   0.,   0.,
         0.,   0.,   0.,   0., 56., 83.,   0.,   0.,   0.,   1., 43.,
         4.,   0.,   0.,   0.,   0.,   0.,   0.,149., 61.,   4.,   0.,
         0.,   0.,   0.,   2., 74., 30., 26.,   8.,   0.,   0.,   2.,
        13.,   1.,   0.,   5.,   4.,   0.,   0., 11., 24., 43.,   0.,
         0.,   0.,   0.,   0.,   0.,   1.,149.,   0.,   0.,   0.,   0.,
         0.,   0., 29., 91.,   0.,   0.,   0.,   0.,   0., 28.,103.,
         0.,   0.,   0.,   0.,   0.,   0., 71.,117., 38.,   0.,   0.,
         0.,   0.,   0.,   0.,   4.,149.,   1.,   0.,   0.,   0.,   0.,
         0.,107., 65.,   1.,   6.,   5.,   0.,   0., 35.,149.,   0.,
         0., 17., 28.,   0.,   0., 36., 61.], dtype=float32)

'Keypoint example for BRISK:'
{'Angle': 106.77054595947266,
 'Octave': 0,
 'Point': (91.41263580322266, 51.645721435546875),
 'Response': 92.01216125488281,
 'Size': 14.063562393188477}
'Descriptor example for BRISK:'
array([254, 255, 239, 243, 240, 224,   0, 129,
       131, 135,  31, 255, 249, 231, 158, 127,
       142, 121, 196,  49, 140, 195,  56, 142,
       131,  35,  97, 198,  25, 109, 124, 227,
         9,   5, 144,  16, 134,  27, 199,  24,
         4,  16, 154, 113,  51,  38, 249, 187,
         0, 237,  50,  25,   4, 195, 192,  64,
       144, 108, 100,  16,  92, 136, 153,  18], dtype=uint8)

'Keypoint example for ORB:'
{'Angle': 281.38720703125,
 'Octave': 0,
 'Point': (391.0, 207.0),
 'Response': 0.00020538090029731393,
 'Size': 31.0}
'Descriptor example for ORB:'
array([120,  61, 248, 228, 184, 111,  85, 127,
        51, 175, 172,  25,  55,  55,  66,  18,
       136, 214, 237,   0,  56,  75, 209,  10,
       219, 199, 116,  57, 225, 224, 206, 107], dtype=uint8)
```

Code Listing 4.2: Stdout from the example 4.1

Feature Matching

In order to group the images based on features the process will rely on matching the features detected. This is done by comparing relative distance of the features from each other, with non-binary feature descriptions like SIFT and SURF, normal trigonometric distance measure can be used [25]. However for binary descriptors like ORB and BRIEF, the hamming distance [99] has to be utilized [100] since trigonometrical distance does not apply.

To determine similarities between images, this thesis proposes a combination of distance between the best feature matches and percentage of feature matches. Since there may be an great number of matches, but a vast distance between them or a low number of matches but a close relation. We also like for the score to be normalized between 0 and 1 in order to more easily compare the score to other methods of comparison. This gives us the final algorithm below:

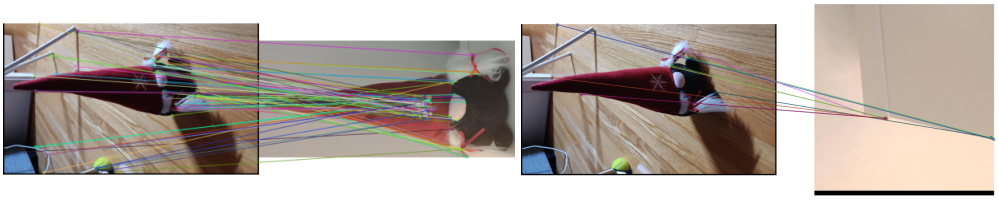
$$M * min(1.0, C1/uDist) * min(1.0, C2/mDist) \tag{4.1}$$

where:

M = max(Hits/max(possible, 200), 1),
$uDist$ = Average distance,
$mDist$ = Median distance,
$C1$ = 60 for norm distance and 2000 for hamming,
$C2$ = 45 for norm distance and 2000 for hamming,

Eq. 4.1: Algorithm for matching score.

Below we can see an example of three images that are compared to each other using SIFT, ORB and BRIEF features. We can see that the images that contain the toy shown in Figure 4.6a has much more connected features than the images compared in Figure 4.6b where one of the images does not. In addition, these features are much more close distance wise then the features of the image of a desk-corner which gives it a greater score in the algorithm.



(a) Comparison of two images with a toy. SIFT score: ≈ 0.42, ORB score: ≈ 0.34.

(b) Comparison of an image with a toy and a corner image. Sift score: ≈ 0.016, ORB score ≈ 0.029

Figure 4.6: Visualization of three images with feature comparison where similar features are connected. (50 features max in order to improve visibility). Scores are calculated using Equation 4.1

Other than feature description matching, we can also match metadata features. The most useful being camera model and geological data which can be used the location of where the photo was initially captured. This is done by reverse geocoding the coordinates obtained from the metadata and utilizing a reverse geocoder [101, 102]. Utilizing this data is an effective indication that the image shares common objects with other images from the same location.

4.2.3 Storage

In order to store the data sets created, a storage system has to be proposed. The proposed storage system will store both the images, data sets and any additional information required. Any information stored is naturally sorted and categorized in order to be easily accessible by the other areas of the workflow.

Storage Type

The chosen storage format is the in-memory storage file format HDF5 [84] which supports storage of any numpy ndarray-object [103], into which any image can be converted into in Python. Since the other components of the workflow require that data is accessed often due to the amount of continuous processing the fast i/o of hdf5 will be useful. Another beneficial features of the storage type is the ability to store arbitrary attributes alongside any data set object. This will allow for the extracted information to be stored as attributes of the image inside the database.

Storage Structure

The HDF5 storage system is very similar to that of the Unix filesystem in that a path is separated by forwardslashes and each path can either be a data set, link or a group. This is very similar to the structure and behavior of objects, links and directories in Unix. A link is stored in the same manner as a data set except that it contains a reference to other data set or ground instead of containing data directly.
Fully using the feature of hdf5, all images and subimages will be stored in the "/all" group. This is to avoid unnecessarily moving and coping the images since they will remain in the same location, no matter what respective group they belong to.

Other than "/all", two more base groups will be created, "/scene" which will contain all scene groups and "/object" which will contain all object groups. These groups will only contain a link to all members belonging to the group. As mentioned above, this allows for any image to belong to many group, move groups and easily be accessed without any copy having to exist. This is the preferred method of dynamic data as a hardlink (direct reference) or softlink (reference to direct reference) requires a very small amount of storage space [80].

Figure 4.7 above demonstrates how the storage will be structured after adding two images. An image on an ocean with two humans and an image of a forest with two birds. This will require the creation of two scene groups and respectively two image groups. Each containing links to images belonging to the group, drawn as dotted lines in the graph.

Since no previous data exists, the groups cannot be named during creation but can be named manually or by some other method later on. New additions will be matched up and added to existing groups by the pre-processing steps. If no groups are a good match: a new group will be added.

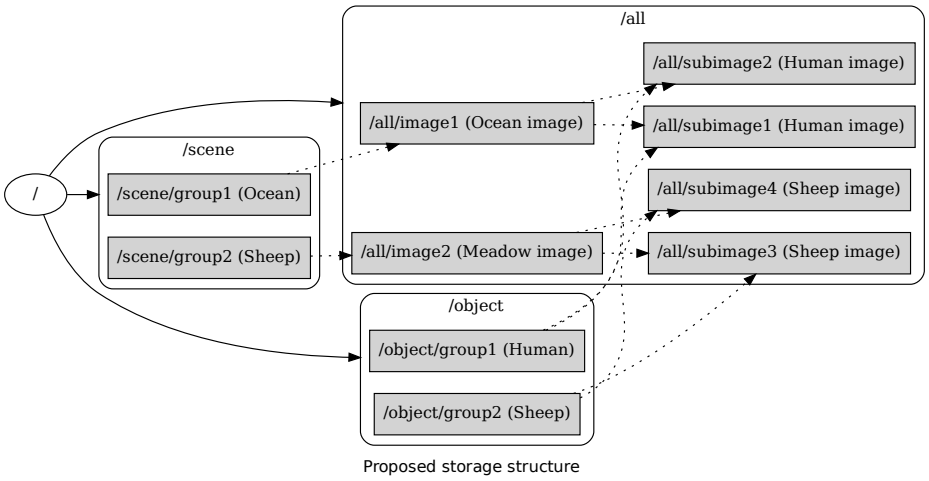


Figure 4.7: The suggested storage structure of the images.

4.2.4 Subset Generation

To alleviate the issues of having to create multiple copies of the same data set in order to share it and having to version control the entire data set to expand it. The proposed solution will instead introduce a recipe or prescription for collecting the data set directly from the database described in the previous steps. The text based recipe will therefore be the object subject to version control instead of the binary files directly which will improve both time and storage efficiency. This project proposes two different forms to represent the data set that shall be generated from the database. One being a description of the content of the data set while the other is a pure list of the exact content. Examples of of such descriptions are collected in Appendix B.

The description will be used to generate an exact list of what images and and classes the data set will contain. As shows in the examples shown in Appendices B.1, B.2 this description aims to give any user reading the description a clear view of what the data set will ultimately contain. It contains many parameter to allow for inclusion and exclusion of images based on the data collected in the pre-processing step. Since the feature descriptions are stored in the database as well, images could potentially even be excluded or included based on features of the image.

However, since the descriptions and attributes of any image are subject to change during the course of the database's existence, the content of the generated data set could change. This would break the requirement for reproducible results and would be severely problematic to version control. Therefore, the workflow will instead generate a report of the exact content the data set allowing for the recreation of the data set based on this report. See examples of this in Appendices B.3, B.4. This will allow for generation of an exact replica of the data set and allow for inspection and version control any changes should occur to the data set due to alteration of any image attributes or addition/deletion of image data.

4.2.5 Post-processing

The current workflow already utilizes augmentation to extend the data set usage. This is done using the python package imgaug [66] which augments the images and automatically modifies the bounding boxes in the augmented pictures to adapt to any shift or displacement. However, nearly all augmentations are based on a random distribution factor to be able to get a random intensity of augmentation which contributes to making the data more varied [104]. This causes the augmented images in the augmented data set to become distinctly different each time a new generation is performed. To solve this and the issue of lack of content overview is the data set, this thesis proposes additional instructions in the data set description which specifies the exact augmentations performed along with a seed [105]. The introduction of a set seed causes the random number generator which decides the exact intensity of any given augmentation to always yield the same result [105]. Thus we will have repoducibility and overview of the content both.

To then liberate any user from having to repeatedly re-specify or provide exact augmentation specifications, the specification will follow a standard augmentation description which can then be modified as desired. These augmentation descriptions follow the imgaug [66] functions for augmentation and specifies type, distribution and intensity. The standard configuration of augmentations chosen is based on the RandAugment method [106] which is also used as the pytouch augmentation method [107]. The method allows for the specification of an augmentation severity, dubbed M in the equation, which can be a value or distributed from a minimum of 0 to a maximum of 30. The severity of any augmentation is then calculated based on the following equation below in Equation 4.2:

$$SF = 0.1 + \frac{M * 1.8}{M_{max}}, \quad SF = 0.1 + \frac{T * M * 1.8}{M_{max}}, S = V * (SF + 1) \tag{4.2}$$

where:

SF = Severity Factor,
M = M resulting from the provided M value/distribution,
M_{max} = Maxiumum M value (30 unless modified),
T = Span variable (either -1 or 1)
S = Severity
V = Normal value

Eq. 4.2: Equation for Severity Factor and severity.

For example: GammaContrast normal value is 1, M is uniformly distributed from 0 to 12 causing a uniformly distributed severity factor of |0| to |0.82| which in turns causes the GammaContrast value to be uniformly distributed from 0.18 to 1.82.

The specification thus only needs to contain, the severity distribution and the amount of augmentation copies that shall be provided. Examples of this can be seen in Appendices B.1, B.2, B.3, B.4.

Chapter 5
Results

This section provides a summary of the proposed workflow in the Analysis section 4 and an overview of the proof of concept implementation created. The effectiveness of the workflow will be evaluated in terms of the metrics stated in Section 3, by testing where an example implementation has been made and by estimation where it has not. This will provide the basis of which the research question 3 will be answered in Section 6. The Project Goals Section 5.3 will then provide an final summary of results in regards to the research questions.

5.1 Summary of Proposed Workflow

The Figures 5.1 and 5.2 summarize the proposed pipeline for expanding the data pool and for creating a data set based on data from the data pool.

For the data set expansion in Figure 5.1, the user will upload images to the webserver, the images are then subject to pre-processing. The images are then stored and the user will provide a name for the groups automatically created. The only two manual parts of the model is the correction of the data that the automatic process has collected and the naming of the groups that the automatic process has collected. Since the images are automatically grouped with similar images, naming and recognizing irrelevant groups for the data set would be fast. However, fixing incorrectly labeled images and moving them into correct groups would be as time consuming as labeling the images manually in the old workflow. But this only has to be done once and, as the processing improves, ideally not at all.

For data set generation in Figure 5.2, the user supplies a data set specification file, receives a data set content description file in return and finally provides the file in return for the finished data set. The only manual part of this process would be the creation of the initial data set description file. If the desired data set has special requirements or needs manual correction this would also be done manually by altering the generated data set content specification file but this should merely be a possibility and not the general rule. The data set specification file would be subject to version control and can be shared with other users, providing them

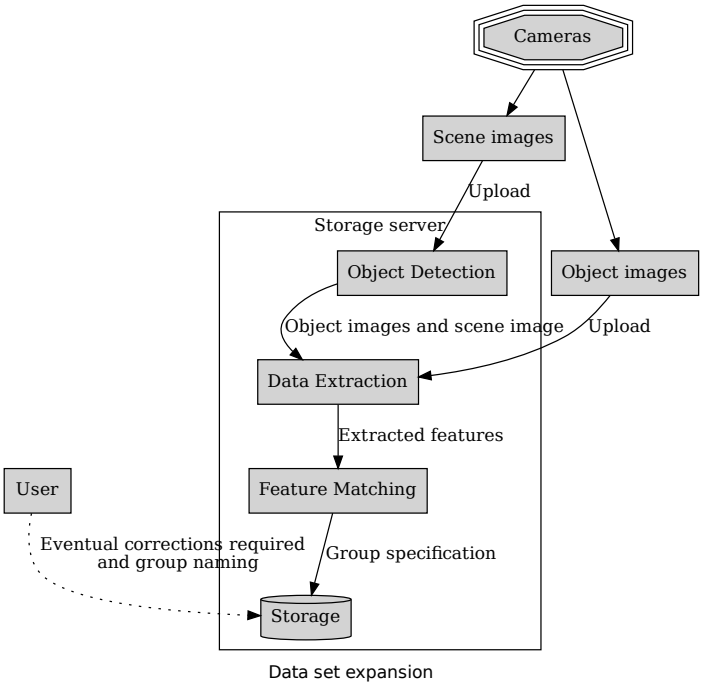


Figure 5.1: The suggested process of adding an image to the data pool.

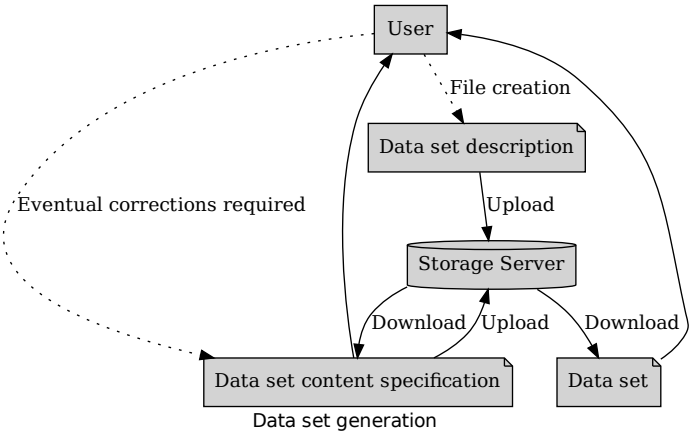


Figure 5.2: The suggested process of generating a data set from the data pool.

with the possibility of generating a clone of the data set on demand.

5.2 Experimental Implementation

An experimental implementation of the Data Upload, Image Pre-processing and Storage steps was implemented. A webserver application was created using the micro web framework Flask [108] containing an API for uploading, viewing and sorting the images. The webserver uses the hd5f database[109] as a backend for storing all the uploaded images. Interaction with the webserver can be done either by a webinterface through a browser or by GET and POST [110] requests which allows both human and applications to interact with the system. The full reference for the implemented API can be found in Appendix C and an example of the webinterface shown in Figure 5.3.

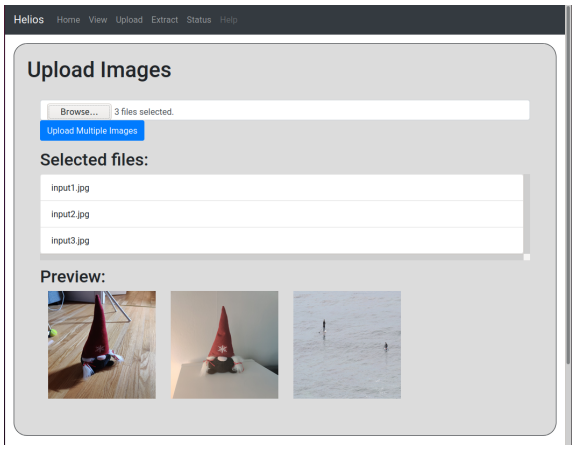


Figure 5.3: The webinterface created for the application. Other views are shown in Appendix D

The Pre-Processing was implemented according to the specifications in the image pre-processing Section 4.2.2 of the proposed workflow. Data collection by ML processing was performed utilizing the Tensorflow Object Detection API [62] for object detection data and image recognition. Additionally, opencv [90] was utilized for saliency object detection and feature extraction. However, only the saliency types residual saliency and fine-grained saliency was implemented and only the feature descriptors SIFT, SURF and ORB was collected due to the implementation only being a proof of concept. Metadata was collected using a combination of pillow [89] and a reverse geocoder [102] to establish the geographical location the image was captured at.

Scene images and object images are automatically grouped in categories according to the structure in Figure 4.7. This is done by first calculating the similarity score with existing categories using the feature descriptors collected and the algorithm in Equation 4.1. If no category is a good match, e.g getting a score above the threshold of 0.3, a new category is created and the image placed in it.

Overall the prototype workflow functions as intended. However, the pre-processing step is limited to only fully accurately processing images that do not contain many objects and

the objects are contrasting to the background. In such cases the object are often found automatically with no need for manual correction but the processing struggles with complex images with many objects and images of poor quality. Images with similar content will almost always be grouped together but color is a very crucial part of the feature detection which means that the same objects of different color will not be grouped together very often. This could be remedied by adding more feature descriptions to the equation presented in Equation 4.1 and by improving the algorithm itself. However, any lack of correction performed will be an improvement over the previous workflow which means that the processing should be applied in either case since any correction will only have to be performed once.

5.3 Project Goals

This section will provide a final summary of the established answers to the research questions presented in Section 3.1.

5.3.1 Current Issues

The first goal of the project was to find the major issues caused by the current management of image data sets as posed by the research question 1. As stated in the Methodology section 3, the approach to identify the issues was to analyze the current workflow by individually analyzing the components of the workflow. The following areas where improvement can be made has been identified an analyzed in the Analysis Section 4: Data Upload, Image Pre-Processing, Storage, Subset Generation and Post-Processing. Before the project was initiated, the current workflow was deemed unfeasible by the company management due to the amount of resources required to operate and competence required to do so. Therefore, the focus was to find the underlying causes of these issues. Analysis of the components in the Analysis Section 4 yielded multiple underlying causes of these issues. Primarily, that many steps in the current workflow consisted of manual, very time consuming and repetitive tasks. This caused the employees performing the tasks to repeatedly produce human errors, all of which requires manual detected to be found and to be manually corrected, further increasing the amount of manual labor in the workflow.

Another issue identified was that there was no clear framework or guidelines of how to perform the tasks. This caused an increase in both the required competence as the employee would have to know how to perform the manual steps. Alternatively the employee could try to ascertain the required steps through experimentation, which increases the amount of errors, or through communication which additionally increases manual labor and introduces the possibility of miscommunication. Additionally, no generalized approach to structure the data caused issues in utilizing the data as the employees has to manually search for previously processed data set to then manually extract and adapt the data to their own data set. The fact that most of the data sets does not contain descriptions of content causes further difficulties. As a consequence, the required competence level required is raised as knowledge on what data is desired, where to find the data and how to adapt the data is a requisite.

Apart from the major issues presented above, other issues were also identified but was es-

tablished to be unrelated to the two major issues of manual labor and competence required. Such issues include inefficient storage by producing multiple copies of the data sets and lack of reproducibility to to due to random parameters in the post-processing.

5.3.2 Proposed Solutions

The next goal of the project was to propose solutions that would alleviate the identified issues as posed by research question 2. An ideal solution was deemed to be a fully automated workflow which would drastically reduce manual labor and competence as there would be no need to perform any manual tasks. The ideal workflow would produces the desired data sets that are fully reproducible on demand and are accompanied a humanly comprehendible overview of the contents. A workflow with the ambition to achieve these ideals was proposed and presented in the Proposed workflow Section 4.2 and a summary was presented in the Workflow Summary Section 5.1. To accomplish full automation, the proposed workflow introduced a system for automatic collection of data, automatic image processing using feature matching and automatic post-processing. Additionally, to further alleviate the issues of required competence and lack of manageability, the proposed workflow also contains a database solution for storage of the data and a system for data set generation through the introduction of content descriptions and specifications.

5.3.3 Effectiveness

The final research question posed 3, inquires as to how the proposed solutions affect the effectiveness of the workflow of the company. This will be primarily answered in the context of the assessment methods of measuring effectiveness stated in the Metrics Section 3.2, e.g, how the solutions affect the manual work and competence required to operate the workflow. The theoretical implementation ensures that the workflow is fully operable without any manual intervention and therefore eliminates any required competence to operate the workflow. While no manual labor would be required to operate the system, manual labor would still be required to setup and maintain the system in the case of the undesired output of the system. This is made clear by the practical implementation of the data processing component of the workflow. While it can effectively cluster images of that are uncluttered and of high quality, the experimental implementation has limitations during more complex circumstances as stated in the Experimental Implementation Section 5.2.

Thus, correction of the errors produced by the system would require the same level of competence as the previous workflow and would be equally difficult to correct. However, the system ensures a consumption of a drastically reduced number of labor hours. Since the storage system is now standardized, meaning that no conversion of data has to be performed to use image data of different origin. This further removed the possibilities for human errors as the proposed workflow manages the storage and categorization of data. If the proposed workflow is evaluated, assuming the worst case scenario of the performance of the data processing component: the automatic clustering of the images is not successful in effectively producing any clusters. The proposed workflow would still improve the performance compared to the previous workflow as the correction is less demanding and only requires to be performed once since the produced result is now automatically managed.

The other components of the experimental implementation, namely the data upload and the data storage is fully automated and successfully eliminated the required resources in regards to the manual labor and the competence required and can therefore be deemed effective.

Chapter 6
Discussion

This section will provide thoughts and reflections of the project as a whole. Potential threats to validity will be presented along with suggestions on improvements to the proposed workflow and future work in the same area based on the results of the project.

6.1 Project

In my opinion, many parts of the project can be considered a success. The current workflow was effectively analyzed to determine the causes of the previously determined inefficiency of the current workflow by the company. Further, the research and experimentation performed in order to produce the theoretical proposed workflow was equally successful in producing results. All of which could be conducted according to the Approach Section 3.3 Where the project encountered the majority of the issues was in the experimental implementation of the workflow. While the project initially had planned to implement a complete workflow according to the specifications of the proposed workflow, the implementation of the image clustering in the data processing component required a more complex solution than initially estimated. This caused the initial scope of the project to be reduced and shift focus from the a full experimental implementation to the evaluation and validity of the proposed workflow.

While the experimental implementation has limitations, as discussed in the Improvement Suggestions Section 6.3, it is my opinion that it proves the proposed workflow viability of the proposed workflow. Since proposed alternative solutions to the components that were deemed to be the major causes of inefficiency were implemented in the experimental implementation. Further, the proposed solutions to the components that were not subject to experimental implementation are not, in my opinion, as complex as the image pre-processing to implement. For example the post-processing component of the proposed workflow, while it required research and experimentation to improve the simplicity and usability of the component, the practical implementation would primarily be a matter of providing parameters

to an existing system of extending the data set. The subset generation component would not be as trivial but the major requirement for a successful implementation is a categorized storage, which the experimental implementation includes. Therefore, I would argue that the primary sources on uncertainties in the proposed workflow, has been alleviated trough solutions which can established be effective to the extent of which the experimental implementation currently operates. Meaning, the proposed workflow can be established to improve the current performance and reduce the required resources to operate if implemented.

6.2 Threats to Validity

Since many parts of the proposed workflow were not implemented, there could be the possibility that the proposed solutions cannot be implementable due to unforeseen factors. Such factors could include, the possibility the component are more expensive to implement in terms of labor, time or hardware than estimated or the solution is not as effective as estimated. However, most of the proposed implementations of the components are based on similar existing systems and the specifications of the theoretically proposed solutions contain suggested techniques and frameworks that can be utilized to achieve the desired result.

A threat to validity from employing the proposed workflow is the nature of the use-case of the company. The complexity of the images admitted into the current workflow is quite low. The objects are commonly distinguishable from the background and the resolutions and quality of the images is high. If this was not the case, the requirements of the pre-processing step drastically increase in order to handle this increased complexity. Techniques such as more advanced edge detection could be necessary to effectively process and automatically detect in this case. Moreover, while the proposed workflow and experimental implementations both support multiple sources of data simultaneously, increased amount of sources and data would but additional demands on the workflow in order to handle the increased flow of data. This, together with many different use-cases, could lead to variations of this project not having equal performance utilizing identical techniques, meaning the proposed workflow might have to be adapted in order to fit other processes.

Further potential threats to validity is the specification for generating the subsets B and the algorithm for determining the parameters of the system for extending an existing data set 4.2 in the proposed post-processing component. Since the proposed subset generation and the post-processing component workflow were not included in the experimental implementation, there is the possibility that both method are not effective for alleviating the issues caused by the same components of the current workflow. However, the algorithm for determining the parameters for extending the data set is based on an algorithm adopted by large existing frameworks, as presented in the proposed workflow Section 4.2.5, which increases the probable validity of the proposed solution.

6.3 Improvement Suggestions and Future Work

For the practical implementation, the major area of improvement would be the clustering technique used to cluster the various image groups in the pre-processing of the proposed workflow. The experimental method for clustering images has many limitations as it currently only produces the desired result on images of high content quality. That is, in images which the content is uncluttered and clearly distinguishable from the background. Furthermore, while the experimental implementation works to a adequate extent in the conditions of the use-case at the company which are comparatively simple compare to a move advanced use-case. Since the company does not require any closely related sub groups such as the same item separated by color or size, the required complexity of the clustering algorithm is heavily reduced. If a version the proposed workflow is to be produced capable of correctly functioning during more complex circumstances, the clustering method would need to collect additional data through further method and utilize the increased knowledge to provide a more qualified assessment.

While not in the scope of this Thesis project, if computational performance is a requirement for a future project based on this project. Several of the technologies chosen for the experimental implementation are doubtlessly subject to replacement as mentioned in the Proposed Workflow Section 4.2. The programming language Python used in the experimental implementation provides various options for implementation but, as a dynamically typed language with automatic garbage collection, it has very poor performance in comparison to most statically typed languages [111]. Additionally, the database system chosen, HD5F, has the possibility of extending the database with dynamically formatted data e.g data of various size and content [84]. While this is useful during experimentation since output format of various feature extraction methods can vary drastically, the dynamic extension is extremely computationally heavy if the storage space is not pre-allocated [112]. To remedy this, space could be pre-allocated if the exact format of the data stored is fixed and prior to storing the data. For example, if the exact amount of extracted features to store from the pre-processing step is determined, storage space for the data can be allocated before the processing is performed.

Another potential area of improvement is the balance of customization versus the usability of the system which is touched upon in the Threats to Validity Section 6.2. If the amount of customization options available is excessive, the level of competence required is increased drastically since the user has to be aware of the outcome of any customization imposed to the data set. Therefore, the customization in the proposed solution is very limited and the output is very specific to the use-case of the company in order to keep the competence required to operate low. This will cause any change in the use case to become very expensive since the system has to be adapted to maintain the level of automation proposed to alleviates the issues identified. The consequence of this could potentially be the loss of any gain in productivity generated by the automatic workflow since the competence required to effectively adapt the workflow to the given circumstances could be greater than the competence required to operate the manual workflow.

Based on the discussion in the paragraphs above, it is clear that the workflow proposed is cur-

rently tailored for the specific behavior required by the use case of the company. To achieve a solution that can be applied to a more general case: future continuations of the project should therefore strive to strike an effective balance between customizability and simplicity to reap the full benefits of the proposed. Additionally, as discussed above, the future continuations should aim to implement a clustering method capable of effectively handling more complex data in order to achieve full automation.

Chapter 7
Conclusion

In conclusion, the goals of the project were partially achieved. While a complete implementation of an improved workflow was not achieved, a full theoretical implementation was attained through research and experimentation. The proposed workflow alleviates most of the issues found with the current workflow and could drastically enhance the current performance of managing image data sets at the company if implemented. The validity of the proposed workflow was strengthened by performing an experimental implementation of the main components of the proposed workflow, namely the data upload, collection, storage and processing. Thus, while the full effect of the entire proposed workflow cannot be practically validated, many of the proposed solutions to the issues of the existing workflow can be established to be effective as a result.

The next future step of the project would naturally be the full implementation of the proposed workflow to be able to effectively evaluate the effectiveness of the proposed solutions. This will provide the possibility to adapt the proposed workflow according to the evaluation. There can also be solutions found to eliminate the final manual steps in the workflow to achieve a fully automated system. Particularly, the system for clustering and categorizing the image data has major potential for improvements. Such improvements should aim to give the proposed workflow the possibility to automatically collect and present humanly interpretable information while producing more specific clusters of images. This is not solved in the proposed workflow without manual correction and could possibly be fully automated with improvements. The clustering of images itself could be optimized additionally by utilizing more computationally expensive techniques for feature extraction while maintaining the desired performance of the system.

References

[1] Venkat Gudivada, Ricardo Baeza-Yates, and Vijay Raghavan. "Big Data: Promises and Problems". In: *IEEE Computer* 48 (Mar. 2015), pp. 20 –23. DOI: 10.1109/MC.2015.62.

[2] P. K. Mishra and G. P. Saroha. "A study on video surveillance system for object detection and tracking". In: *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*. 2016, pp. 221–226.

[3] Zhuoling Li et al. "CLU-CNNs: Object detection for medical images". In: *Neurocomputing* 350 (2019), pp. 53 –59. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2019.04.028. URL: http://www.sciencedirect.com/science/article/pii/S0925231219305521.

[4] D. M. Gavrila and V. Philomin. "Real-time object detection for "smart" vehicles". In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 1. 1999, 87–93 vol.1. DOI: 10.1109/ICCV.1999.791202.

[5] *Google company introduction*. Accessed: 2020-11-12. URL: https://about.google/.

[6] *Facebook company introduction*. Accessed: 2020-11-12. URL: https://about.fb.com/company-info/.

[7] Z. Zhao et al. "Object Detection With Deep Learning: A Review". In: *IEEE Transactions on Neural Networks and Learning Systems* 30.11 (2019), pp. 3212–3232. DOI: 10.1109/TNNLS.2018.2876865.

[8] Sebastian Raschka. *Python machine learning*. Packt publishing ltd, 2015.

[9] Martín Abadi et al. "Tensorflow: A system for large-scale machine learning". In: *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*. 2016, pp. 265–283.

[10] Chen Sun et al. "Revisiting Unreasonable Effectiveness of Data in Deep Learning Era". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017.

[11] A. L'Heureux et al. "Machine Learning With Big Data: Challenges and Approaches". In: *IEEE Access* 5 (2017), pp. 7776–7797. DOI: 10.1109/ACCESS.2017.2696365.

[12] *DVC: Dataset version control tool*. Accessed: 2020-10-31. URL: https://dvc.org/doc.

[13] *An open source Git extension for versioning large files*. Accessed: 2020-12-20. URL: https://git-lfs.github.com/.

[14] Miller. S; Hughes. D. *'The quant crunch: how the demand for data science skills is disrupting the job market'*. Tech. rep. viewed 17 Nov 2020. Burning Glass Technologies: Boston, 2017. URL: http://burning-glass.com/wp-content/uploads/The_Quant_Crunch.pdf.

[15] Joe McKendrick. *Artificial intelligence skills shortages re-emerge from hiatus*. Accessed: 2020-11-17. 2020. URL: https://www.zdnet.com/article/artificial-intelligence-skills-shortages-re-emerge/.

[16] *RELX Emerging Tech Executive Report 2020*. Accessed: 2020-11-17. URL: https://stories.relx.com/relx-emerging-tech-2020/index.html.

[17] *AutoML workhow at ICML 14*. Accessed: 2021-03-21. URL: https://sites.google.com/site/automlwsicml14/.

[18] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, eds. *Automated Machine Learning : Methods, Systems, Challenges*. English. Springer Nature, 2019, pp. ix, 181. DOI: 10.1007/978-3-030-05318-5. URL: https://library.oapen.org/bitstream/id/02773536-62c8-428d-9172-c3195c2e602d/1007149.pdf.

[19] Alexandre Quemy. "Two-stage optimization for machine learning workflow". In: *Information Systems* 92 (2020), p. 101483. ISSN: 0306-4379. DOI: https://doi.org/10.1016/j.is.2019.101483. URL: http://www.sciencedirect.com/science/article/pii/S0306437919305356.

[20] Alec Sharp and Patrick McDermott. *Workflow Modeling: Tools for Process Improvement and Application Development*. 1st. USA: Artech House, Inc., 2001. ISBN: 1580530214.

[21] T. Li et al. "Contextual Bag-of-Words for Visual Categorization". In: *IEEE Transactions on Circuits and Systems for Video Technology* 21.4 (2011), pp. 381–392. DOI: 10.1109/TCSVT.2010.2041828.

[22] Piotr Dabkowski and Yarin Gal. "Real Time Image Saliency for Black Box Classifiers". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017, pp. 6967–6976. URL: https://proceedings.neurips.cc/paper/2017/file/0060ef47b12160b9198302ebdb144dcf-Paper.pdf.

[23] Zhaowei Cai et al. "A Unified Multi-scale Deep Convolutional Neural Network for Fast Object Detection". In: vol. 9908. Oct. 2016, pp. 354–370. ISBN: 978-3-319-46492-3. DOI: 10.1007/978-3-319-46493-0_22.

[24] R. Girshick et al. "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 580–587. DOI: 10.1109/CVPR.2014.81.

[25] T. Lindeberg. "Scale Invariant Feature Transform". In: *Scholarpedia* 7.5 (2012). revision #149777, p. 10491. DOI: 10.4249/scholarpedia.10491.

[26] X. Li et al. "Modeling Image Data for Effective Indexing and Retrieval in Large General Image Databases". In: *IEEE Transactions on Knowledge and Data Engineering* 20.11 (2008), pp. 1566–1580. DOI: 10.1109/TKDE.2008.56.

[27] F. Lyu et al. “Coarse to Fine: Multi-label Image Classification with Global/Local Attention”. In: *2018 IEEE International Smart Cities Conference (ISC2)*. 2018, pp. 1–7. DOI: 10.1109/ISC2.2018.8656664.

[28] S. Madden. “From Databases to Big Data”. In: *IEEE Internet Computing* 16.3 (2012), pp. 4–6. DOI: 10.1109/MIC.2012.50.

[29] Iztok Savnik. “Index Data Structure for Fast Subset and Superset Queries”. In: *Availability, Reliability, and Security in Information Systems and HCI*. Ed. by Alfredo Cuzzocrea et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 134–148. ISBN: 978-3-642-40511-2.

[30] Xiaodi Guan et al. “Entropy Based Data Expansion Method for Blind Image Quality Assessment”. In: *Entropy* 22 (Dec. 2019), p. 60. DOI: 10.3390/e22010060.

[31] Dengsheng Lu. “A Survey of Image Classification Methods and Techniques for Improving Classification Performance”. In: *International Journal of Remote Sensing* 28 (Mar. 2007), pp. 823 –870. DOI: 10.1080/01431160600746456.

[32] A. Behl et al. “Bounding Boxes, Segmentations and Object Coordinates: How Important is Recognition for 3D Scene Flow Estimation in Autonomous Driving Scenarios?” In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2593–2602. DOI: 10.1109/ICCV.2017.281.

[33] *Dive into Deep Learning: object detection and bounding boxes*. Accessed: 2021-03-21. URL: https://d2l.ai/chapter_computer-vision/bounding-box.html#bounding-box.

[34] Li Liu et al. “Deep Learning for Generic Object Detection: A Survey”. In: Sept. 2018.

[35] Peter M. Roth. “On-line Conservative Learning”. In: (Apr. 2011).

[36] Kemal Oksuz et al. “Localization Recall Precision (LRP): A New Performance Metric for Object Detection”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

[37] Huizi Mao, Xiaodong Yang, and William J. Dally. “A Delay Metric for Video Object Detection: What Average Precision Fails to Tell”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019.

[38] Shengkai Wu, Xiaoping Li, and Xinggang Wang. “IoU-aware Single-stage Object Detector for Accurate Localization”. In: *CoRR* abs/1912.05992 (2019). arXiv: 1912.05992. URL: http://arxiv.org/abs/1912.05992.

[39] *The Confusing Metrics of AP and mAP for Object Detection / Instance Segmentation*. Accessed: 2020-11-12. URL: https://medium.com/@yanfengliux/the-confusing-metrics-of-ap-and-map-for-object-detection-3113ba0386ef.

[40] Dong ping Tian. “A Review on Image Feature Extraction and Representation Techniques”. In: *International Journal of Multimedia and Ubiquitous Engineering* 8 (4 2013), pp. 385–396. URL: https://www.earticle.net/Article/A202352.

[41] *Generating and Applying a Bag of Visual Words Model for Image Classification*. Accessed: 2020-11-18. URL: http://www.deepcore.io/2017/04/18/generating-and-applying-a-bag-of-visual-words-model-for-image-classification/#comment-219.

[42] Ebrahim Karami, Siva Prasad, and Mohamed S. Shehata. "Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images". In: *CoRR* abs/1710.02726 (2017). arXiv: 1710.02726. URL: http://arxiv.org/abs/1710.02726.

[43] D. G. Lowe and J. S. Beis. "Shape Indexing Using Approximate Nearest-Neighbour Search in High-Dimensional Spaces". In: *2013 IEEE Conference on Computer Vision and Pattern Recognition*. Los Alamitos, CA, USA: IEEE Computer Society, 1997, p. 1000. DOI: 10.1109/CVPR.1997.609451. URL: https://doi.ieeecomputersociety.org/10.1109/CVPR.1997.609451.

[44] Ethan Rublee et al. "ORB: an efficient alternative to SIFT or SURF". In: Nov. 2011, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

[45] Z. Dai et al. "CNN Descriptor Improvement Based on L2-Normalization and Feature Pooling for Patch Classification". In: *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. 2018, pp. 144–149. DOI: 10.1109/ROBIO.2018.8665330.

[46] *Getting Started with R-CNN, Fast R-CNN, and Faster R-CNN*. Accessed: 2020-11-19. URL: https://ch.mathworks.com/help/vision/ug/getting-started-with-r-cnn-fast-r-cnn-and-faster-r-cnn.html.

[47] J. Shi et al. "Hierarchical Image Saliency Detection on Extended CSSD". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.4 (2016), pp. 717–729. DOI: 10.1109/TPAMI.2015.2465960.

[48] Carola Figueroa Flores et al. "Saliency for Fine-grained Object Recognition in Domains with Scarce Training Data". In: *CoRR* abs/1808.00262 (2018). arXiv: 1808.00262. URL: http://arxiv.org/abs/1808.00262.

[49] *Amazon web services: Training ML Models*. Accessed: 2020-12-20. URL: https://docs.aws.amazon.com/machine-learning/latest/dg/training-ml-models.html.

[50] Léon Bottou. "Large-Scale Machine Learning with Stochastic Gradient Descent". In: *Proceedings of COMPSTAT'2010*. Ed. by Yves Lechevallier and Gilbert Saporta. Heidelberg: Physica-Verlag HD, 2010, pp. 177–186. ISBN: 978-3-7908-2604-3.

[51] Qianru Sun et al. "Meta-transfer learning for few-shot learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 403–412.

[52] Lisa Torrey and Jude Shavlik. *Transfer learning*. IGI global, 2010, pp. 242–264.

[53] *Transfer learning using Tensorflow*. Accessed: 2020-11-19. URL: https://medium.com/@subodh.malgonde/transfer-learning-using-tensorflow-52a4f6bcde3e.

[54] Wei Zhao. "Research on the deep learning of the small sample data based on transfer learning". In: *AIP Conference Proceedings*. Vol. 1864. 1. AIP Publishing LLC. 2017, p. 020018.

[55] Jon Loeliger and Matthew McCullough. *Version Control with Git: Powerful Tools and Techniques for Collaborative Software Development*. O'Reilly Media, Inc., 2012. ISBN: 1449316387.

[56] Maxwell Ogden, Karissa McKelvey, Mathias Buus Madsen, et al. "Dat-Distributed Dataset Synchronization And Versioning". In: (2017). URL: https://github.com/datprotocol/whitepaper/blob/master/dat-paper.pdf.

[57] D. P. Gangwar and A. Pathania. "Authentication of Digital Image using Exif Metadata and Decoding Properties". In: 2018.

[58] *Adobe photoshop website*. Accessed: 2020-11-30. URL: https://www.adobe.com/products/photoshop.html.

[59] *Exif fileformat doc*. Accessed: 2020-11-30. URL: http://www.fileformat.info/format/jpeg/egff.htm.

[60] *IPTC: Photo Metadata*. Accessed: 2021-03-22. URL: https://iptc.org/standards/photo-metadata/photo-metadata/.

[61] R. Shekhar and C. V. Jawahar. "Word Image Retrieval Using Bag of Visual Words". In: *2012 10th IAPR International Workshop on Document Analysis Systems*. 2012, pp. 297–301. DOI: 10.1109/DAS.2012.96.

[62] *Tensorflow 2 Object Detection API Tutorial: Label maps*. Accessed: 2021-03-23. URL: https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/training.html#create-label-map.

[63] *Yolo format specification*. Accessed: 2020-11-30. URL: https://github.com/ultralytics/yolov3/wiki/Train-Custom-Data.

[64] Joseph Redmon et al. *You Only Look Once: Unified, Real-Time Object Detection*. 2016. arXiv: 1506.02640 [cs.CV].

[65] X. Ke, J. Zou, and Y. Niu. "End-to-End Automatic Image Annotation Based on Deep CNN and Multi-Label Data Augmentation". In: *IEEE Transactions on Multimedia* 21.8 (2019), pp. 2093–2106. DOI: 10.1109/TMM.2019.2895511.

[66] *imgaug Python Documentation*. Accessed: 2020-11-18. URL: https://imgaug.readthedocs.io/en/latest/.

[67] David A. Cieslak and Nitesh V. Chawla. "Learning Decision Trees for Unbalanced Data". In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Walter Daelemans, Bart Goethals, and Katharina Morik. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 241–256. ISBN: 978-3-540-87479-9.

[68] Biao Leng, Kai Yu, and Jingyan QIN. "Data augmentation for unbalanced face recognition training sets". In: *Neurocomputing* 235 (2017), pp. 10 –14. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2016.12.013. URL: http://www.sciencedirect.com/science/article/pii/S0925231216314886.

[69] *Documentation on the tar format from gnu.org*. Accessed: 2020-11-16. URL: https://www.gnu.org/software/tar/manual/html_node/Standard.html.

[70] *Tensorflow's documentation page on the tfrecord data format*. Accessed: 2020-11-16. URL: https://www.tensorflow.org/tutorials/load_data/tfrecord.

[71] Jinchuan Chen et al. "Big data challenge: a data management perspective". In: *Frontiers of Computer Science* 7.2 (2013), pp. 157–164. ISSN: 2095-2236. DOI: 10.1007/s11704-013-3903-7. URL: https://doi.org/10.1007/s11704-013-3903-7.

[72] C Ballard et al. *Data modeling techniques for data warehousing*. 1998, p. 10. URL: http://members.aol.com/fmcguff/dwmodel/index.htm.

[73] E.F. Codd. "Relational Database: A Practical Foundation for Productivity". In: *Readings in Artificial Intelligence and Databases*. Ed. by John Mylopolous and Michael Brodie. San Francisco (CA): Morgan Kaufmann, 1989, pp. 60–68. ISBN: 978-0-934613-53-8. DOI: https://doi.org/10.1016/B978-0-934613-53-8.50009-1. URL: https://www.sciencedirect.com/science/article/pii/B9780934613538500091.

[74] *How to use Linked Data: Linked Data Search*. Accessed: 2020-12-07. URL: https://en.wikitolearn.org/index.php?title=Course:How_to_use_Linked_Data/Interaction_with_Linked_Data/Linked_Data_Search&veaction=edit&vesection=3.

[75] Peter Haugen et al. "A Basic Overview of Commonly Encountered types of Random Access Memory (RAM)". In: (). URL: https://www.rose-hulman.edu/Class/ee/yoder/ece332/Papers/RAM\%20Technologies.pdf.

[76] *What is an in-memory database?* Accessed: 2020-12-13. URL: https://aws.amazon.com/nosql/in-memory/.

[77] *Azure Documentation: Blob image data*. Accessed: 2021-03-28. URL: https://docs.microsoft.com/en-us/azure/storage/blobs/storage-upload-process-images?tabs=dotnet.

[78] *Datamatic: Store Images in the Database*. Accessed: 2021-03-28. URL: https://www.datanamic.com/support/storeimagesinthedatabase.html.

[79] Jim Gray. *To BLOB or Not To BLOB: Large Object Storage in a Database or a Filesystem*. Tech. rep. MSR-TR-2006-45. 2006, p. 10. URL: https://www.microsoft.com/en-us/research/publication/to-blob-or-not-to-blob-large-object-storage-in-a-database-or-a-filesystem/.

[80] *Hdf5 Manual: Groups and links*. Accessed: 2021-03-01. URL: http://davis.lbl.gov/Manuals/HDF5-1.8.7/UG/09_Groups.html.

[81] Michael J Kamfonas. "Recursive Hierarchies: The Relational Taboo!" In: *The Relational Journal* (1992). URL: Cachedatwayback-machine:https://web.archive.org/web/20081108151540/http://www.kamfonas.com/id3.html.

[82] D. L. Swets and Juyang Weng. "Hierarchical discriminant analysis for image retrieval". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21.5 (1999), pp. 386–401. DOI: 10.1109/34.765652.

[83] E. Yildirim, S. Duan, and X. Qi. "A Distributed Deep Memory Hierarchy System for Content-based Image Retrieval of Big Whole Slide Image Datasets". In: *2019 IEEE/ACM Workshop on Memory Centric High Performance Computing (MCHPC)*. 2019, pp. 43–49. DOI: 10.1109/MCHPC49590.2019.00013.

[84] *HDF5 group solutions: HDF5 format introduction*. Accessed: 2020-11-18. URL: https://www.hdfgroup.org/solutions/hdf5/.

[85] Jean McNiff. *You and your action research project*. Routledge, 2016, pp. 20–22.

[86] David Avison et al. "Action Research". In: *COMMUNICATIONS OF THE ACM* 42.1 (1999), p. 95.

[87] Gregory Mentzas, Christos Halaris, and Stylianos Kavadias. "Modelling business processes with workflow systems: an evaluation of alternative approaches". In: *International Journal of Information Management* 21.2 (2001), pp. 123 –135. ISSN: 0268-4012. DOI: https://doi.org/10.1016/S0268-4012(01)00005-6. URL: http://www.sciencedirect.com/science/article/pii/S0268401201000056.

[88] *Tensorflow API Documentation*. Accessed: 2021-02-28. URL: https://www.tensorflow.org/api_docs.

[89] *Pillow: Documenation*. Accessed: 2021-02-28. URL: https://pillow.readthedocs.io/en/stable/.

[90] *OpenCV-python: Documentation*. Accessed: 2021-02-28. URL: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_tutorials.html.

[91] *JSZip library description*. Accessed: 2021-02-28. URL: https://stuk.github.io/jszip/.

[92] Mingxing Tan, Ruoming Pang, and Quoc V. Le. "EfficientDet: Scalable and Efficient Object Detection". In: *CoRR* abs/1911.09070 (2019). arXiv: 1911.09070. URL: http://arxiv.org/abs/1911.09070.

[93] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. 2015. arXiv: 1405.0312 [cs.CV].

[94] N. Hassan and R. Hijazi. *Digital Privacy and Security Using Windows: A Practical Guide*. Apress, 2017, pp. 57–59. ISBN: 9781484227985. URL: https://books.google.se/books?id=UNXPtAEACAAJ.

[95] *Opencv documentation: Feature descriptors*. Accessed: 2021-03-27. URL: https://docs.opencv.org/master/db/d27/tutorial_py_table_of_contents_feature2d.html.

[96] Mingxing Tan and Quoc V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *CoRR* abs/1905.11946 (2019). arXiv: 1905.11946. URL: http://arxiv.org/abs/1905.11946.

[97] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[98] *Imagenet: class identifiers list*. Accessed: 2021-03-27. URL: https://gist.github.com/yrevar/942d3a0ac09ec9e5eb3a.

[99] R. W. Hamming. "Error detecting and error correcting codes". In: *The Bell System Technical Journal* 29.2 (1950), pp. 147–160. DOI: 10.1002/j.1538-7305.1950.tb00463.x.

[100] *Opencv documentation: Feature matching*. Accessed: 2021-03-27. URL: https://docs.opencv.org/master/dc/dc3/tutorial_py_matcher.html.

[101] *Latlong.net: Reverse geocoding*. Accessed: 2021-03-27. URL: https://www.latlong.net/Show-Latitude-Longitude.html.

[102] *Github: offline reverse geocoder*. Accessed: 2021-03-27. URL: https://gist.github.com/yrevar/942d3a0ac09ec9e5eb3a.

[103] *Numpy: ndarray documentation*. Accessed: 2021-02-28. URL: https://numpy.org/doc/stable/reference/arrays.html.

[104] *imgaug Python Documentation: Stochastic Parameters*. Accessed: 2021-03-28. URL: https://imgaug.readthedocs.io/en/latest/source/parameters.html#stochastic-parameters.

[105] *imgaug Python Documentation: seed() Function*. Accessed: 2021-03-28. URL: https://imgaug.readthedocs.io/en/latest/source/api_imgaug.html#imgaug.imgaug.seed.

[106] Ekin D. Cubuk et al. "RandAugment: Practical data augmentation with no separate search". In: *CoRR* abs/1909.13719 (2019). arXiv: 1909.13719. URL: http://arxiv.org/abs/1909.13719.

[107] *Pytorch RandAugment documentation*. Accessed: 2021-03-6. URL: https://github.com/ildoonet/pytorch-randaugment/blob/master/RandAugment/augmentations.py.

[108] *Flask python web framework Documentation*. Accessed: 2021-03-28. URL: https://flask.palletsprojects.com/en/1.1.x/.

[109] *H5py Documentation*. Accessed: 2021-03-28. URL: https://docs.h5py.org/en/stable/.

[110] *w3wchool: HTTP Request Methods*. Accessed: 2021-03-28. URL: https://www.w3schools.com/tags/ref_httpmethods.asp.

[111] KR Srinath. "Python–the fastest growing programming language". In: *International Research Journal of Engineering and Technology* 4.12 (2017), pp. 354–357.

[112] *BASTet documentation: HDF5 I/O Performance*. Accessed: 2021-04-22. URL: https://biorack.github.io/BASTet/HDF5_format_performance.html.

[113] *Computer science at LTH thesis template*. Accessed: 2020-11-29. URL: https://bitbucket.org/flavius_gruian/msccls/src/master/.

Appendices

Appendix A
Embedded Image Metadata

This appendix includes the embedded metadata of this papers cover image.

Figure A.1: Coverphoto of this thesis. Source: [113]

Code Listing A.1: Metadata (including Exif) for cover image

```
{
    "jfif": "257",
    "jfif_version": "(1, 1)",
    "jfif_unit": "0",
    "jfif_density": "(150, 150)",
    "exif": {
```

```
        "ExifVersion": "0221",
        "ShutterSpeedValue": "8.643855995239512",
        "ApertureValue": "4.643856000573703",
        "DateTimeOriginal": "'2014:08:13 07:47:21'",
        "DateTimeDigitized": "'2014:08:13 07:47:21'",
        "ExposureBiasValue": "0.0",
        "MaxApertureValue": "3.6",
        "SubjectDistance": "4294967295.0",
        "MeteringMode": "3",
        "LightSource": "0",
        "Flash": "16",
        "FocalLength": "18.0",
        "ExifImageWidth": "680",
        "DigitalZoomRatio": "1.0",
        "FocalLengthIn35mmFilm": "27",
        "SceneCaptureType": "0",
        "ExifImageHeight": "1000",
        "Contrast": "0",
        "Saturation": "0",
        "Sharpness": "2",
        "SubjectDistanceRange": "0",
        "ImageDescription": "'E-huset pa LTH. Bilden ar tankt
           att anvandas som bakgrundsbild med en textplatta i
           ovre delen.'",
        "Make": "'NIKON CORPORATION'",
        "SensingMethod": "2",
        "Model": "'NIKON D7000'",
        "FileSource": "\u0003",
        "ExposureTime": "0.0025",
        "XResolution": "150.0",
        "YResolution": "150.0",
        "FNumber": "5.0",
        "SceneType": "\u0001",
        "ExposureProgram": "3",
        "CustomRendered": "0",
        "ISOSpeedRatings": "200",
        "ResolutionUnit": "2",
        "PhotometricInterpretation": "2",
        "ExposureMode": "0",
        "WhiteBalance": "0",
        "BodySerialNumber": "'6426664'",
        "LensSpecification": "(18.0, 105.0, 3.5, 5.6)",
        "LensModel": "'18.0-105.0 mm f/3.5-5.6'",
        "Software": "'Adobe Photoshop CS5.1 Windows'",
        "DateTime": "'2015:02:27 08:41:26'",
        "GainControl": "0",
        "Orientation": "1",
        "ExifOffset": "342"
    },
    "dpi": "(150, 150)",
    "photoshop": {
        "ExifVersion": "0221",
        "ShutterSpeedValue": "8.643855995239512",
        "ApertureValue": "4.643856000573703",
```

```
        "DateTimeOriginal": "'2014:08:13 07:47:21'",
        "DateTimeDigitized": "'2014:08:13 07:47:21'",
        "ExposureBiasValue": "0.0",
        "MaxApertureValue": "3.6",
        "SubjectDistance": "4294967295.0",
        "MeteringMode": "3",
        "LightSource": "0",
        "Flash": "16",
        "FocalLength": "18.0",
        "ExifImageWidth": "680",
        "DigitalZoomRatio": "1.0",
        "FocalLengthIn35mmFilm": "27",
        "SceneCaptureType": "0",
        "ExifImageHeight": "1000",
        "Contrast": "0",
        "Saturation": "0",
        "Sharpness": "2",
        "SubjectDistanceRange": "0",
        "ImageDescription": "'E-huset pa LTH. Bilden ar tankt
           att anvandas som bakgrundsbild med en textplatta i
           ovre delen.'",
        "Make": "'NIKON CORPORATION'",
        "SensingMethod": "2",
        "Model": "'NIKON D7000'",
        "FileSource": "\u0003",
        "ExposureTime": "0.0025",
        "XResolution": "150.0",
        "YResolution": "150.0",
        "FNumber": "5.0",
        "SceneType": "\u0001",
        "ExposureProgram": "3",
        "CustomRendered": "0",
        "ISOSpeedRatings": "200",
        "ResolutionUnit": "2",
        "PhotometricInterpretation": "2",
        "ExposureMode": "0",
        "WhiteBalance": "0",
        "BodySerialNumber": "'6426664'",
        "LensSpecification": "(18.0, 105.0, 3.5, 5.6)",
        "LensModel": "'18.0-105.0 mm f/3.5-5.6'",
        "Software": "'Adobe Photoshop CS5.1 Windows'",
        "DateTime": "'2015:02:27 08:41:26'",
        "GainControl": "0",
        "Orientation": "1",
        "ExifOffset": "342"
    },
    "icc_profile": "Adobe RGB (1998)"
}
```

Figure A.2: Image of a piece of curio taken using an android smartphone. Source: Lund, Sweden (personal photograph)

Code Listing A.2: Geolocation metadata for image above. We can see DMS coordinates: “55°42’15.5"N 13°10’27.9"E” and that the photo was taken “2020-11-30 11:50:9”

```
{
    "_comment": "[Other metadata]",
    "GPSInfo":
        {
        "GPSAltitude": 78.4,
        "GPSAltitudeRef": "",
        "GPSDateStamp": "2020:11:30",
        "GPSLatitude": "(55.0, 42.0, 15.5663)",
        "GPSLatitudeRef": "N",
        "GPSLongitude": "(13.0, 10.0, 27.9371)",
        "GPSLongitudeRef": "E",
        "GPSProcessingMethod": "ASCII\x00\x00\x00CELLID\x00",
        "GPSTimeStamp": "(11.0, 50.0, 9.0)"
    },
    "comment_": "[Other metadata]"
}
```

Appendix B
Proposed data set descriptions

Code Listing B.1: Description of object data set

```
---
## Request creation of content

# Data set characteristics
name: Example Object Data set
description: >
  An object data set with the to identify humans
  and sheep in order to help sheep herders distinguish
  between the two.
type: object
format: yolo
dimensions: # Output dimensions
  width: 640
  height: 480

# augmentation, yes if n > 0
augmentation:
  n: 30
  Mmin: 0
  Mmax: 12
  M_dist: uniform

# Groups to include
classes:
  Human:
    include:
      description:
        - "Human"
```

```
          - "Person"
          - "Man"
          - "Woman"
          - "Child"
        metadata:
          location:
            - "Sweden"
            - "Lund"

    exclude:

  Sheep:
    include:
      description:
        - "Sheep"
        - "Ram"

  exclude:
    size:
      above: 0.5
      below: 0.04
    metadata:
      contrast: 0
    description:
      - "Rainy"
```

Code Listing B.2: Description of scene data set

```
---
## Request creation of content

# Data set characteristics
name: Example Scene Data set
description: >
  A data set with the to detect humans and sheep
  in order to help sheep herders count employees
  and herd size on sunny days outdoors.
type: scene
format: yolo
dimensions:
  width: 640
  height: 480

# augmentation, yes if n > 0
augmentation:
  n: 30
  Mmin: 0
  Mmax: 12
  M_dist: uniform

# Groups to include
classes:
```

```
  Human:
    include:
      description:
        - "Human"
        - "Person"
        - "Man"
        - "Woman"
        - "Child"
      metadata:
        location:
          - "Sweden"
          - "Lund"

    exclude:
      description:
        - "Indoors"

  Sheep:
    include:
      description:
        - "Sheep"
        - "Ram"

    exclude:
      size:
        above: 0.5
        below: 0.04
      metadata:
        contrast: 0
      description:
        - "Rainy"
```

Code Listing B.3: Content of object data set

```
---
# Description of the dataset
name: Example Object Data set
description: >
  An object data set with the to identify humans
  and sheep in order to help sheep herders distinguish
  between the two.
type: object
class_count: 2
image_count: 4
format: yolo
size_mb: 400
dimensions:
  width: 640
  height: 480

# The classes the dataset will consist of
classes:
```

```
  Human:
    count: 2
  Sheep:
    count: 2

# The images and classes making up the dataset
content:
  subimage1: Human
  subimage2: Human
  subimage3: Sheep
  subimage4: Sheep

# Augmentation
augmentation:
  seed: 0
  n: 30
  aug:
    strategy: SomeOf
    sigma:
      dist: uniform
      min: 0
      max: 1.0
    children:
      GammaContrast:
        args: [(0.18, 1.82)]
      Alpha:
        args: [(0.1, 0.82), AllChannelsHistogramEqualization()]
      Solarize:
        args: [1, (23, 233)]
      Rotate:
        args: [(-30, 30)]
      UniformColorQuantizationToNBits:
        args: [(2, 8)]
      MultiplyHueAndSaturation:
        args: [(0.18, 1.82)]
      MultiplyBrightness:
        args: [(0.18, 1.82)]
      Sharpen:
        args: [(0.0, 0.82),1.0]
      Affine:
        args: [(-30, 30)]
      Cutout:
        args: [(20, 80), 0.005, False, 'gaussian']
      aug:
        children:
          strategy: SomeOf
          sigma:
            dist: uniform
            min: 0
            max: 1.0
          TranslateX:
            args: [(-82, 82)]
          TranslateY:
            args: [(-82, 82)]
```

Code Listing B.4: Object of scene data set

```
---
# Description of the dataset
name: Example Scene Data set
description: >
  A data set with the to detect humans and sheep
  in order to help sheep herders count employees
  and herd size on sunny days outdoors.
type: scene
class_count: 2
image_count: 2
format: yolo
size_mb: 20
dimensions:
  width: 640
  height: 480

# The classes the dataset will consist of
classes:
  Human:
    count: 2
  Sheep:
    count: 2

# The images and classes making up the dataset
content:
  image1:
    object1:
      class: Human
      x: 0.2
      y: 0.4
      width: 0.15
      height: 0.15
    object2:
      class: Human
      x: 0.2
      y: 0.4
      width: 0.15
      height: 0.15

    image2:
      object1:
        class: Sheep
        x: 0.2
        y: 0.4
        width: 0.15
        height: 0.15
      object2:
        class: Sheep
        x: 0.2
```

```
        y: 0.4
        width: 0.15
        height: 0.15

# Augmentation
augmentation:
  seed: 0
  n: 30
  aug:
    strategy: SomeOf
    sigma:
      dist: uniform
      min: 0
      max: 1.0
    children:
      GammaContrast:
        args: [(0.18, 1.82)]
      Alpha:
        args: [(0.1, 0.82), AllChannelsHistogramEqualization()]
      Solarize:
        args: [1, (23, 233)]
      Rotate:
        args: [(-30, 30)]
      UniformColorQuantizationToNBits:
        args: [(2, 8)]
      MultiplyHueAndSaturation:
        args: [(0.18, 1.82)]
      MultiplyBrightness:
        args: [(0.18, 1.82)]
      Sharpen:
        args: [(0.0, 0.82),1.0]
      Affine:
        args: [(-30, 30)]
      Cutout:
        args: [(20, 80), 0.005, False, 'gaussian']
      aug:
        children:
          strategy: SomeOf
          sigma:
            dist: uniform
            min: 0
            max: 1.0
          TranslateX:
            args: [(-82, 82)]
          TranslateY:
            args: [(-82, 82)]
```

Appendix C
API Reference

CHAPTER

ONE

DOCUMENTATION FOR THE SERVER API

Resource	Operation	Description
	POST /api/generate/dsetspec	
	POST /api/generate/dset	
	POST /api/command/sort	
	GET /api/list/eventdates	
	GET /api/get/info	
	POST /api/upload	
	GET /api/list/images/(category_name)/(group_name)	
	GET /api/list/groups/(category_name)	
	GET /api/list/events/(date)	
	GET /api/get/attributes/(image_name)	
	GET /api/get/image/(image_name)	

1.1 Api details

`POST /api/generate/dsetspec`

Generate a dataset specification based on the dataset content file provided

```
curl -F dataset_description.yaml http://localhost:5000/generate/dsetspec
```

Example request:

```
POST /generate/dsetspec HTTP/1.1
Host: 127.0.0.1:5000
Content-Type: text/yaml
Content-Disposition: text/yaml; filename="dataset_description.yaml"
```

Example response:

```
    HTTP/1.1 200 OK
    Vary: Accept
    Content-Type: text/yaml
    Content-Disposition: text/yaml; filename="dataset_content.yaml"

.. sourcecode:: http

    HTTP/1.1 500 ERROR
    Vary: Accept
    Content-Type: application/json
```

(continues on next page)

(continued from previous page)

```
    {
        "error": "..."
    }
```

Request Headers

- Accept – the response content type depends on *Accept* header

Status Codes

- 200 OK – success
- 500 Internal Server Error – could not generate content specification

POST /api/generate/dset

Generate a dataset specification based on the dataset content file provided

```
curl -F dataset_content.yaml http://localhost:5000/generate/dset
```

Example request:

```
POST /generate/dset HTTP/1.1
Host: 127.0.0.1:5000
Content-Type: text/yaml
Content-Disposition: text/yaml; filename="dataset_content.yaml"
```

Example response:

```
    HTTP/1.1 200 OK
    Vary: Accept
    Content-Type: gzip
    Content-Disposition: gzip; filename="dataset.tar.gz"

.. sourcecode:: http

    HTTP/1.1 500 ERROR
    Vary: Accept
    Content-Type: application/json

    {
        "error": "..."
    }
```

Request Headers

- Accept – the response content type depends on *Accept* header

Status Codes

- 200 OK – success
- 500 Internal Server Error – could not generate dataset

POST /api/command/sort

Trigger a sort of the database

```
curl -X http://localhost:5000/command/sort
```

Example request:

```
POST /command/sort HTTP/1.1
Host: 127.0.0.1:5000
Accept: application/json
```

Example response:

```
    HTTP/1.1 200 OK
    Vary: Accept
    Content-Type: application/json

    {
        "status": "ok"
    }

.. sourcecode:: http

    HTTP/1.1 500 ERROR
    Vary: Accept
    Content-Type: application/json

    {
        "error": "..."
    }
```

Request Headers

- Accept – the response content type depends on *Accept* header

Status Codes

- 200 OK – success
- 500 Internal Server Error – could not sort

GET /api/list/eventdates

List all dates in the eventlog

```
curl http://localhost:5000/api/get/groups/scene
```

Example request:

```
GET /users/123/posts/web HTTP/1.1
Host: 127.0.0.1:5000
Accept: application/json
```

Example response:

```
    HTTP/1.1 200 OK
    Vary: Accept
    Content-Type: application/json

    {
        "data": {
            "groups":["unsorted", "group0", "group1"]
```

(continues on next page)

(continued from previous page)

```
        }
    }

.. sourcecode:: http

    HTTP/1.1 404 NOT FOUND
    Vary: Accept
    Content-Type: application/json

    {
        "error": "no such category found"
    }
```

Request Headers

- Accept – the response content type depends on *Accept* header

Status Codes

- 200 OK – success
- 404 Not Found – no such category found

GET /api/get/info

Return system information.

```
curl http://localhost:5000/api/get/info
```

Example request:

```
GET /api/get/info HTTP/1.1
Host: 127.0.0.1:5000
Accept: application/json
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
    "data": {
        "DATABASE": "/tmp/imagedb.h5",
        "SIZEGB": "0.00GB",
        "IMAGES": 0,
        "SCENE_GROUPS": 0,
        "OBJECT_GROUPS": 0,
        "UNSORTED_IMAGES": 0,
        "TESTING": true,
        "DEBUG": true,
        "ENV": "development",
        "ALLOWED_IMG_EXTENSIONS": [
            "jpeg",
            "jpg",
            "png"
        ],
        "ANALYZER_THREADS": 1
```

(continues on next page)

(continued from previous page)

```
    }
}
```

Request Headers

- Accept – the response content type depends on *Accept* header

Status Codes

- 200 OK – success

POST /api/upload

Receive an image or images to put into the database

```
curl -F 'avatar.png=@/home/jtrack/Pictures/avatar.png' -F 'sib_location.
↪png=@/home/jtrack/Pictures/sib_location.png' http://localhost:5000/api/
↪upload
```

Example request:

```
    POST /api/upload HTTP/1.1
    Host: 127.0.0.1:5000
    Accept-Encoding: gzip
    Content-Type: multipart/form-data;boundary="files[]"

    --files[]
    Content-Disposition: form-data; name="image"; filename="image1.jpg"

    --files[]
    Content-Disposition: form-data; name="image"; filename="image2.png"
    Content-Type: image/png

.. sourcecode:: http

    POST /api/upload HTTP/1.1
    Host: 127.0.0.1:5000
    Accept-Encoding: gzip
    Content-Type: image/png
    Content-Disposition: form-data; name="image"; filename="image1.jpg"
```

Example response:

```
    HTTP/1.1 200 OK
    Location: /view
    Content-Type: application/json
    Content-Length: length

    {
        {
                "filename": "image1.jpg",
                "id": "image0"
        },
        {
                "filename": "image2.png",
                "id": "image1"
        }
```

(continues on next page)

(continued from previous page)

```
    }

.. sourcecode:: http

    HTTP/1.1 400 Redirect
    Location: /view
    Vary: Accept
```

Request Headers

- Accept – the response content type depends on `Accept` header

Status Codes

- 200 OK – no error, get infolist json
- 400 Bad Request – no file part in POST request, redirect to /upload

GET /api/list/images/ (*category_name*) **/**
group_name List all images in the group *group_name* under category *category_name*

```
curl http://localhost:5000/api/list/images/scene/group0
```

Example request:

```
GET /api/list/images/<category_name>/<group_name> HTTP/1.1
Host: 127.0.0.1:5000
Accept: application/json
```

Example response:

```
    HTTP/1.1 200 OK
    Vary: Accept
    Content-Type: application/json

    {
        "data": {
            "images":["image0", "image1"]
        }
    }

.. sourcecode:: http

    HTTP/1.1 404 NOT FOUND
    Vary: Accept
    Content-Type: application/json

    {
        "error": "no such category found"
    }
```

Request Headers

- Accept – the response content type depends on `Accept` header

Status Codes

- 200 OK – success

- 404 Not Found – no such category found

GET /api/list/groups/ (*category_name*)
List all groups in the category *category_name* e.g 'scene' or 'object'

```
curl http://localhost:5000/api/list/groups/scene
```

Example request:

```
GET /api/list/groups/<category_name> HTTP/1.1
Host: 127.0.0.1:5000
Accept: application/json
```

Example response:

```
    HTTP/1.1 200 OK
    Vary: Accept
    Content-Type: application/json

    {
        "data": {
            "groups":["unsorted", "group0", "group1"]
        }
    }

.. sourcecode:: http

    HTTP/1.1 404 NOT FOUND
    Vary: Accept
    Content-Type: application/json

    {
        "error": "no such category found"
    }
```

Request Headers

- Accept – the response content type depends on `Accept` header

Status Codes

- 200 OK – success
- 404 Not Found – no such category found

GET /api/list/events/ (*date*)
List all events on the specified *date*

```
curl http://localhost:5000/api/list/events/"2021-04-06"
```

Example request:

```
GET /list/events/<date> HTTP/1.1
Host: 127.0.0.1:5000
Accept: application/json
```

Example response:

```
    HTTP/1.1 200 OK
    Vary: Accept
    Content-Type: application/json

    {
    "data": {
        events: [
            "06:23:35 - INFO - 'Database initialized'",
            "06:23:35 - INFO - 'Website initialized'",
            "07:21:41 - REQUEST - 'From 127.0.0.1: GET /api/get/info?'"
            ]
        }
    }

.. sourcecode:: http

    HTTP/1.1 404 NOT FOUND
    Vary: Accept
    Content-Type: application/json

    {
        "error": "no entry for date found"
    }
```

Request Headers

- Accept – the response content type depends on `Accept` header

Status Codes

- 200 OK – success
- 404 Not Found – no entry for date found

GET /api/get/attributes/(*image_name*)

Get the attributes of an image from the database using the unique *image_name*.

```
curl http://localhost:5000/api/get/attributes/image0
```

Example request:

```
GET /api/get/attributes/image0 HTTP/1.1
Host: 127.0.0.1:5000
Accept: image/jpeg; image/png
```

Example response:

```
    HTTP/1.1 200 OK
    Vary: Accept
    Content-Type: image/png

.. sourcecode:: http

    HTTP/1.1 404 NOT FOUND
    Vary: Accept
    Content-Type: application/json

    {
```

(continues on next page)

(continued from previous page)

```
        "error": "not found"
    }
```

Request Headers

- Accept – the response content type depends on `Accept` header

Status Codes

- 200 OK – success
- 404 Not Found – no such image found

GET /api/get/image/ (*image_name*)
Get an image from the database using the unique *image_name*.

```
curl http://localhost:5000/api/get/image/image0
```

Example request:

```
GET /api/get/image/<image_name> HTTP/1.1
Host: 127.0.0.1:5000
Accept: image/jpeg; image/png
```

Example response:

```
    HTTP/1.1 200 OK
    Vary: Accept
    Content-Type: image/png

.. sourcecode:: http

    HTTP/1.1 404 NOT FOUND
    Vary: Accept
    Content-Type: application/json

    {
        "error": "not found"
    }
```

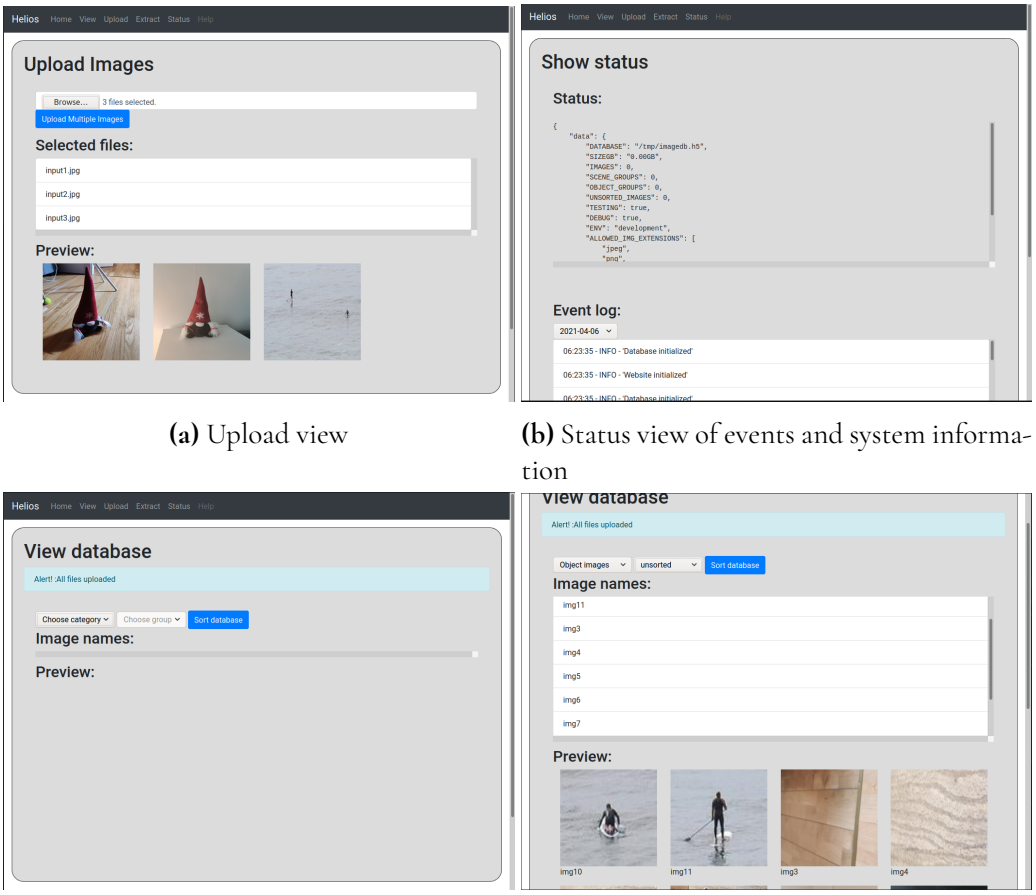
Request Headers

- Accept – the response content type depends on `Accept` header

Status Codes

- 200 OK – no error, redirect to /view
- 404 Not Found – no such image found

Appendix D
Interface



(a) Upload view

(b) Status view of events and system information

(c) Database view

(d) Database view of images in a group

Figure D.1: Examples of the interface used for a user to more easily interact with the workflow.

Appendix E

Detection Model Classes

Code Listing E.1:
Coco class listing

```
1  person
2  bicycle
3  car
4  motorbike
5  aeroplane
6  bus
7  train
8  truck
9  boat
10 traffic light
11 fire hydrant
12 stop sign
13 parking meter
14 bench
15 bird
16 cat
17 dog
18 horse
19 sheep
20 cow
21 elephant
22 bear
23 zebra
24 giraffe
25 backpack
26 umbrella
27 handbag
28 tie
29 suitcase
30 frisbee
31 skis
32 snowboard
33 sports ball
34 kite
35 baseball bat
36 baseball glove
37 skateboard
38 surfboard
39 tennis racket
40 bottle
41 wine glass
42 cup
43 fork
44 knife
45 spoon
46 bowl
47 banana
48 apple
49 sandwich
50 orange
51 broccoli
52 carrot
53 hot dog
54 pizza
55 donut
56 cake
57 chair
58 sofa
59 pottedplant
60 bed
61 diningtable
62 toilet
63 tvmonitor
64 laptop
65 mouse
66 remote
67 keyboard
68 cell phone
69 microwave
70 oven
71 toaster
72 sink
73 refrigerator
74 book
75 clock
76 vase
77 scissors
78 teddy bear
79 hair drier
80 toothbrush
```

EXAMENSARBETE Advancing machine learning workflow efficiency by improving management of large image datasets
STUDENT Jonathan Jakobsson
HANDLEDARE Alma Orucevic-Alagic (LTH)
EXAMINATOR Mathias Haage (LTH)

Effektivisering av maskininlärnings-process genom förbättrad hantering av stora bilddata-set

POPULÄRVETENSKAPLIG SAMMANFATTNING **Jonathan Jakobsson**

Med ökat nyttjande av maskininlärningsmetoder för databehandling, kommer även ökade krav på hanteringen av dataflödet. Detta arbete har eftersträvat att förbättra hanteringen av stora bilddata-set och därmed reducera mängden resurser som krävs för att underhålla ett bilddata-flöde för användning inom maskininlärning.

Genom att använda maskininlärningsmodeller för att bearbeta bilddata kan stora mängder data bearbetas automatiskt. Denna metod kommer dock inte utan kostnad. För att maskininlärningsmodellerna ska prestera och kunna producera korrekta förutsägelser om framtida data behövs stora mängder träningsdata för att förbereda modellerna. Detta arbete har analyserat ett arbetsflöde på företaget SiB Solutions AB där målet är att kontinuerligt bygga och expandera bildträningsdataset för att anpassa modeller för olika användningsfall. Företagets nuvarande arbetsflöde har avbrutits då kostnaden i form av arbetskraft resulterar i att arbetsflödet blir för olönsamt. Examensarbetet föreslår en lösning genom att analysera det nuvarande arbetsflödet, i syfte att konstatera vilka brister som finns, anledningen till dessa samt vilka krav som finns på flödet. Anledningen till bristerna kan sammanfattas som mycket repetitiva arbetssysslor och få riktlinjer vilket ökar kravet på kunskap om maskininlärning och erfarenhet inom arbetsflödet. Efter analys genomfördes efterforskning och experiment för att nå

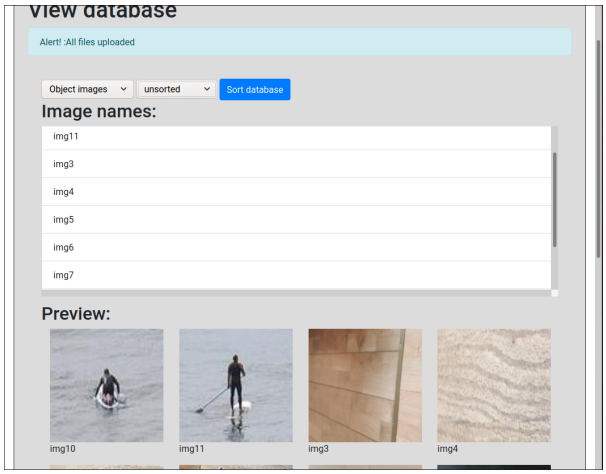


Figure 1: Användargränssittet för den experimentella implementationen av arbetsflödet. Vyn som visas på bilden är till för att inspektera bilder i databasen.

förslag på lösningar till dessa brister för att sedan kunna göra en sammanställning av ett förslag på ett förbättrat arbetsflöde. Resultatet blev en specifikation på ett helt automatiserat system

EXAMENSARBETE Advancing machine learning workflow efficiency by improving management of large image datasets
STUDENT Jonathan Jakobsson
HANDLEDARE Alma Orucevic-Alagic (LTH)
EXAMINATOR Mathias Haage (LTH)

som använder sig av flera olika tekniker för att minska manuella moment. Det togs även fram en experimentell implementation av en del av flödet. De tekniker som används för att åstadkomma detta är, bl.a. automatisk bildbehandling för att bearbeta och sortera bilddata, en databas för att kunna effektivt spara och sortera bilddata och specifikationer för dataset för att förbättra hanteringen av dessa. Det nya arbetsflödet har potential att markant reducera kostnad förknippad med drift och underhåll av stora bilddata-set för maskininlärning.