TALLINN UNIVERSITY OF TECHNOLOGY

Faculty of Information Technology

Anna Grund 211532IABM

HISTORIC IMAGES CLASSIFICATION BASED ON *AJAPAIK* PLATFORM IMAGESETS

Master's thesis

Supervisor: Priit Järv
PhD

Tallinn 2024

TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Anna Grund 211532IABM

AJALOOLISTE PILTIDE KLASSIFITSEERIMINE ***AJAPAIK*** PLATVORMI PILTIDE ALUSEL

Magistritöö

Juhendaja: Priit Järv
PhD

Tallinn 2024

Author's declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author:	Anna Grund	....
		(signature)
Date:	02.01.2024	

Abstract

This thesis aims to address the limitations of manual image categorization on the *Ajapaik* [1] platform by introducing an automated categorization algorithm. Currently, users categorize historical images voluntarily which leads to incomplete and potentially inaccurate categorization. A significant portion of the images lacks categories, obstructing effective categorization-based filtering.

The second goal of the thesis is to enhance the model categorization algorithm by building an intermediate categorization validation layer and excluding inaccurate users' categorization records. The thesis prioritizes quality user input for model training, aiming to enhance long-term prediction accuracy. Emphasis is also placed on scalability, minimizing changes to current processes and codebase. The goal is to ensure easy future maintenance and modifications without significant effort.

As a contribution, we designed, implemented and validated a categorization engine coupled with a *Data Quality Engine* layer to process user feedback before passing it next to the model retraining cycle. This feedback is processed and subsequently incorporated into the model retraining loop, ensuring a continuous improvement scope for the model's performance. The implemented solution has undergone validation and is scheduled for production deployment in January 2024.

The thesis is written in English and contains 79 pages of text, 7 chapters, 26 figures and 13 tables.

Annotatsioon

Antud magistritöö eesmärk on lahendada käsitsi piltide kategoriseerimise piiranguid *Ajapaik* [1] platvormil, tutvustades automatiseeritud kategoriseerimisalgoritmi. Praegu kategoriseerivad kasutajad ajaloolisi pilte vabatahtlikult, mis viib mittetäieliku ja potentsiaalselt ebatäpse kategoriseerimiseni. Arvestatav osa piltidest pole kategooriatega varustatud, takistades tõhusat kategooriapõhist filtreerimist.

Magistritöö teine eesmärk on täiustada mudeli kategoriseerimisalgoritmi, luues vahekihi kategoriseerimise valideerimise tasandi ja välistades ebatäpsed kasutajate kategoriseerimise kirjed. Töö keskendub kvaliteetsele kasutaja sisendile mudeli koolitamiseks, eesmärgiga suurendada pikaajalist ennustustäpsust. Rõhk on ka suurendada skaleeritavust, minimeerides muudatusi praegustesse protsessidesse ja koodibaasi. Eesmärk on tagada lihtne tulevane hooldus ja muudatused ilma märkimisväärse pingutuseta.

Panuseks kavandasime, rakendasime ja valideerisime kategoriseerimismootori koos andmekvaliteedi mootori kihiga (*Data Quality Engine)*, et töödelda kasutajate tagasisidet enne selle edastamist mudeli uuesti koolitamise tsüklisse. See tagasiside töödeldakse ja seejärel integreeritakse mudeli uuesti koolitamise tsüklisse, tagades mudeli jõudluse pideva täiustamise. Rakendatud lahendus on läbinud valideerimise ja on planeeritud tootmistepärasesse kasutussevõttu jaanuaris 2024.

Magistritöö on kirjutatud inglise keeles ja sisaldab 79 lehekülge teksti, 7 peatükki, 26 joonist ja 13 tabelit.

Table of Contents

List of Figures

List of Tables

List of abbreviations and terms

AUM	Area Under the Margin
CNN	Convolutional Neural Network
CPU	Central Processing Unit
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
ML	Machine Learning
CLIP	Contrastive Language Image Pre-training
TL	Transfer Learning
UI	User Interface

1 Introduction

Ajapaik [1] is an online platform based in Estonia that focuses on crowdsourcing and curating historical photographs and images. The platform aims to collect, digitize, and geotag historical photos, making them accessible to the public while also encouraging individuals to contribute their own historical photographs and add valuable context to the images. *Ajapaik* allows users to explore a visual representation of history, connecting past and present by overlaying historical photos on modern-day maps.

The first goal of the thesis is to build an automated images categorization engine which benefits user experience in their every-day *Ajapaik* platform usage. As of today, uploaded historical images are categorized by users manually and voluntarily. Although current platform functionality enables users to perform images categorization, users do not contribute to the process in demanded scale and as a consequence categorization based filtering cannot always be accurate.

Currently, *Ajapaik* enables the categorization of every image using two classification categories: *scene* and *viewpoint elevation.* This implies that any image featured on the platform can be categorized as *scene* (either *interior* or *exterior*) and as *viewpoint elevation* (*ground*, *raised*, or *aerial*).

From 1 121 575 images currently available on *Ajapaik* platform only 43.19% have *scene* category label attached and 37.16% have *viewpoint elevation* label present. Throughout the time the number of images is only growing and there might be an interest in hosting large numbers of historical images coming from museums and art galleries. Given graph (Figure 1) illustrates the tendency of images upload and its categorisation over time:

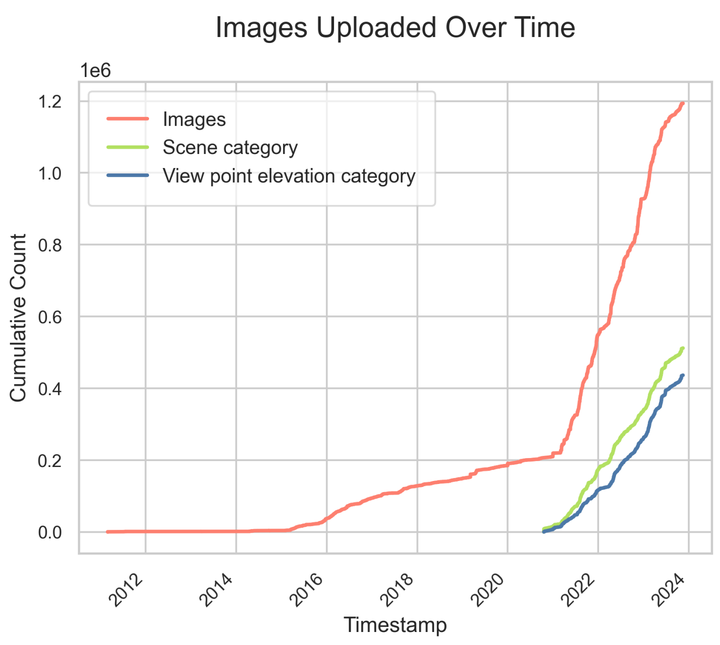


Figure 1. Images upload and categorization over time

The second goal of the thesis is to enhance the automated images categorization algorithm by building an intermediate categorization validation layer and excluding inaccurate users' categorization records. In addition to automated categorization we would still be interested in supplying and training the prediction model with users' input but it is crucial to only provide the model with quality data inputs to enhance its prediction capabilities.

Throughout the course of this thesis, our focus is the future scalability of the model we are developing. We endeavor to ensure that any adjustments to existing processes are kept to a minimum, and that future maintenance or modifications to the model can be seamlessly accomplished, without notable obstacles.

The thesis provides contributions in the following aspects:

- Manually categorized images and composed *ground truth* data.
- Developed image categorization model.
- Developed *Data Quality Engine* algorithm.
- Integrated model into the *Ajapaik* platform with UI adjustments.

The rest of the thesis is organized as follows:

- chapter 2 provides an overview of related work, delving into literature that focuses on enhancing the processing of low-quality and historical images, development of tools for ensuring data quality, as well as overview of comparative datasets hosting historical images.

- chapter 3 delves into the *Ajapaik* platform, exploring its current image categorization solution and explaining existing limitations within the platform.
- chapter 4 provides an overview of *Ajapaik* data, user stories, and technical requirements. It details the solution devised to address challenges and discusses alternative approaches, weighing their pros and cons in relation to the thesis objectives.
- chapter 5 provides an overview of all the results and findings achieved during the thesis. The chapter includes insights into the validation process, covering the evaluation of the developed software system, user interface, and model accuracy.
- chapter 6 summarizes key findings and insights derived from the research.
- chapter 7 outlines potential areas for future exploration and development, extending beyond the current thesis scope.

2 Related work

In this chapter, we present an overview of the relevant literature. This body of work is research of enhancing the processing of low-quality and/or historical images and also addresses the development of data quality tools designed to identify and eliminate inaccurate or faulty users' feedback. In addition, given chapter also provides an overview of comparative datasets hosting historical images.

2.1 Related work on historical images and low-quality images processing

While working on the thesis, we delved into existing research on historical images and/or images with low-quality data, summarizing our findings here.

In the growing landscape of image processing research, a diverse range of studies has been conducted on historical images, defect identification and image categorization. Notable contributions include a Generative Adversarial Network (GAN)-based approach for automating the reconstruction and recognition of low-quality defect images in industrial settings [2]. Another study deals with the detection of degraded characters in historical typewritten documents, introducing a novel clustering-labeling framework for improved results [3]. Object detection techniques in image processing are explored in a separate work, emphasizing the challenges of low-quality images with obstructions [4]. The automatic digitization of historical photographs, specifically focusing on building detection using a Bag-of-Keypoints approach, stands out as a distinctive effort [5]. In the realm of vintage image classification, a study leverages support vector machines, providing a foundation for easily configurable systems [6]. Preservation efforts for historical documents through image processing techniques are evident in a separate study, emphasizing the importance of automated digitization [7]. Urban versus landscape image classification is explored in another work, developing a distinctive procedure for measuring feature distinctiveness [8]. An innovative approach to image labeling through incremental model learning showcases the significance of accurate labeling in computer vision and object categorization [9]. One more study explores challenges and solutions in automated metadata annotation across diverse domains, focusing on use cases related to cultural heritage materials, artworks, and research datasets [10]. The authors of the paper address issues such as insufficient training examples, difficulties in interpreting metaphors and symbols, inconsistent quality across categories, and the importance of adaptability and continuous model updates in the context of machine learning for metadata annotation. Lastly, given study explores the application of CLIP (Contrastive Language Image Pre-training) in Digital Humanities through three case studies, demonstrating its effectiveness in labeling unlabeled visual data, assessing complex visual concepts, and generating labeled training data [11]. The authors discuss challenges, including biases, and suggest that multimodal models like CLIP may lead to a transformative turn in Digital Humanities research by facilitating scalable analysis and integration of images and texts.

Each study contributes uniquely to the broader field, collectively shaping the landscape of image processing and recognition taking into account low quality and resolutions of the input dataset. In the following table (Table 1) we summarise the results of the related work analyses more precisely described in section 2.1.

Research	**Focus/Topic**	**Low-quality input (Y/N)**	**Historical datasets (Y/N)**	**Classification focus (Y/N)**
A Generative Adversarial Network Based Deep Learning Method for Low-Quality Defect Image Reconstruction and Recognition [2]	Automation of reconstruction and recognition of low-quality defect images	Y	N	N
New Framework for Recognition of Heavily Degraded Characters in Historical Typewritten Documents Based on Semi-Supervised Clustering [3]	Detection of degraded characters in historical typewritten documents	Y	Y	N
Prototype analysis of different object recognition techniques in image processing [4]	Object detection techniques in low-quality images with obstructions	Y	N	Y

Detecting Buildings in Historical Photographs Using Bag-of-Keypoints [5]	Automatic digitization of historical photographs, building detection	Y	Y	N
Old fashioned state-of-the-art image classification [6]	Vintage image classification	Y	N	Y
Enhancement of historical documents by image processing techniques [7]	Historical documents automated digitization	Y	Y	N
On image classification: city vs. landscape [8]	Urban versus landscape image classification	N	N	Y
Image labeling via incremental model learning [9]	Image labeling, object categorization	N	N	Y
Automated metadata annotation: What is and is not possible with machine learning [10]	Metadata annotation automation across cultural heritage materials, artworks, and research datasets	Y	Y	Y
A Multimodal Turn in Digital Humanities: Using Contrastive Machine Learning Models to Explore, Enrich, and Analyze Digital Visual Historical Collections [11]	Labeling unlabeled visual historical data, generating labeled training data	Y	Y	Y

Table 1. Related work summary

2.2 Related work on faulty feedback detection

Within this section, we provide an overview of potentially faulty feedback detection and exclusion relevant to *Ajapaik* image categorization.

One study focuses on detecting opinion spam in online product reviews, employing a novel method based on anomalous rating deviations [12]. Another paper explores label noise detection and correction, introducing label noise-robust algorithms and a correction method for classification datasets [13]. This user-friendly system, with automatic hyperparameter selection, quantitatively assesses and qualitatively demonstrates effectiveness in identifying mislabeled instances. A third study explores the AUM (Area Under the Margin) metric to identify mislabeled data in machine learning, achieving high precision and recall rates [14].

These contributions collectively aim to refine machine learning models by enhancing data quality, mitigating misleading feedback, and addressing mislabeled instances across various applications.

2.3 Related work on model architecture

This chapter aims to present an overview of the model architecture and techniques employed in our thesis.

2.3.1 Transfer Learning

Transfer learning (TL), an aspect of Machine Learning (ML), addresses challenges caused by limited training data [15][16]. Traditional ML algorithms work under the assumption of a confined data distribution and might lack their efficiency with smaller data or imagesets. Transfer learning establishes connections between testing (validation) and training samples, which eventually leads to more efficient and trustful outcomes. It involves transferring knowledge from one model's training to enhance another model's performance on a very similar task, proving to be beneficial when annotated data for the given task is limited. TL overcomes data scarcity, facilitates model training and increases generalization across different tasks. TL is recognized as a knowledge transfer approach to enhance traditional ML. It improves task understanding by utilizing the knowledge from related tasks performed at different time periods.

For developing our model for image category predictions, we aim to utilize transfer learning.

2.3.2 Model architecture: MobileNetV2

MobileNetV2 is an advanced mobile architecture that significantly enhances the performance of mobile models across a variety of tasks and benchmarks, accommodating various model sizes [17]. Given framework introduces SSDLite [18] for efficient object detection in mobile models and presents Mobile DeepLabv3 (a reduced form of DeepLabv3), designed mainly for mobile semantic segmentation models. MobileNetV2 uses an inverted residual structure with shortcut connections between thin bottleneck layers, utilizing lightweight depthwise convolutions in the expansion layer to enhance non-linearity while preserving representational its power. The model's unique feature is the decoupling of input/output domains, providing a framework for in-depth analysis.

In the context of image category predictions, we use MobileNetV2 model architecture to develop our model.

2.3.3 Comparative datasets hosting historical images

Throughout the course of our research, we have analyzed several digital platforms that host historical images. The following platforms are available contributors on the international arena:

- **Library of Congress Prints and Photographs Online Catalog:** Hosts thousands of digitized images with catalog records from 54 historic collections [19]
- **New York Public Library/Mid-Manhattan Library Picture Collection:** Stores digitized historical images mostly created before 1923 [20]
- **Art Images for College Teaching (AICT):** Stores ancient, medieval, and Renaissance European art and architecture imagesets [21]
- **British Library in Flickr Commons:** Hosts over 1,000,000 images in the public domain scanned from 17-19th century books, including maps, geological diagrams, illustrations, comical satire, illuminated and decorative letters, colourful illustrations, landscapes, wall-paintings [22]
- **Wellcome Images:** Stores over 100,000 images, including historical content such as manuscripts, paintings, etchings, early photography and advertisements [23]
- **Time & Life Pictures (Getty images):** Hosts over 425,000 digital files of 20th century original prints and negatives archived [24]
- **LIFE photo archive hosted by Google:** Searches for millions of photographs from the LIFE photo archive, stretching from the 1750s to today [25]
- **Mary Evans Picture Library:** Hosts collection of historic pictures. Image collections divided into categories of art and costume, beliefs, events, portraits,

science and medicine, trade and industry, social sciences, places and nature, politics and law, sports and entertainment, places and nature, transport [26]

The following are Estonian-based resources that similarly host collections of historical images:

- **Fotis:** National Archives Photo Database [27]
- **Rahvusarhiiv (Flickr):** National Archives of estonian historical images [27]
- **Museum of Univeristy of Tartu:** Historical images of Tartu [28]

Each of these platforms mentioned above hosts considerably large volumes of historical images. However, it is not obvious to conclude that any of them currently incorporates any similar automated images categorization mechanism as was developed as part of this thesis.

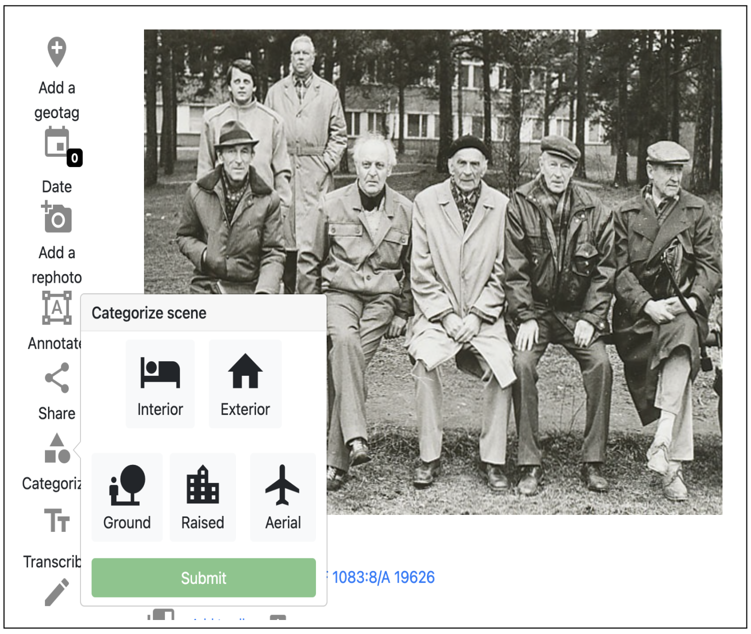
3 Ajapaik platform

In this chapter, we offer a comprehensive exploration of the *Ajapaik* platform, current images categorization solution and platform existing constraints.

3.1 *Ajapaik* platform

Ajapaik is an online platform based in Estonia that focuses on crowdsourcing and curating historical photographs and images. The platform aims to collect, digitize, and geotag historical photos, making them accessible to the public while also encouraging individuals to contribute their own historical photographs and add valuable context to the images. *Ajapaik* allows users to explore a visual representation of history, connecting past and present by overlaying historical photos on modern-day maps.

Current *Ajapaik* functionality enables users to categorize images into two category classes: *scene* and *viewpoint elevation*, each containing subclasses as follows:



```
- scene
    - interior
    - exterior
- view_point_elevation
    - ground
    - raised
    - aerial
```

Figure 2. *Ajapaik* platform available categories

It is worth to note, that subclasses are not mutually excluded explicitly meaning that each image is expected to be categorized by two category classes: *scene* and *viewpoint elevation*.

Example: an image can be described as *exterior* and *aerial* or *interior* and *raised.*

Where:

- ***Exterior:*** Images taken outside of a building.
- ***Interior:*** Images captured within the interior of a building.
- ***Ground:*** Images taken from a viewpoint at ground level.
- ***Raised:*** Images captured from an elevated viewpoint.
- ***Aerial:*** Images taken from a high aerial perspective.

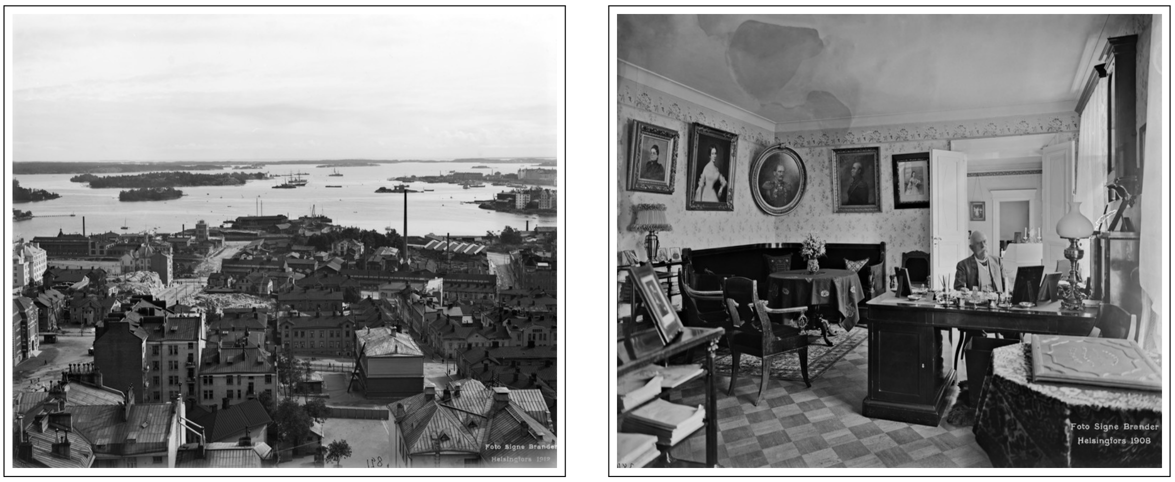


Figure 3. *Ajapaik* image 1 being *exterior* and *aerial* and image 2 being *interior* and *ground*

3.2 *Ajapaik* limitations

On a broader scale, one significant challenge that *Ajapaik* faces relates to user trust and involvement. Within the system, each image is meticulously categorized through the collective efforts of its users, and it is the users themselves who wield the power to shape the final categorization of any given image.

For instance, when a user *A* categorizes an image with the label *interior*, this choice subsequently influences what the next user sees and can search for, as they will encounter images labeled as *interior*. However, if the subsequent user *B*, believes the image is more accurately labeled as *exterior* or purposely would submit faulty feedback, this shift in perception will, once again, have a ripple effect, causing subsequent users to encounter and search for images specifically labeled as *exterior*.

Example:

Image has category labels of *interior* and *ground*:

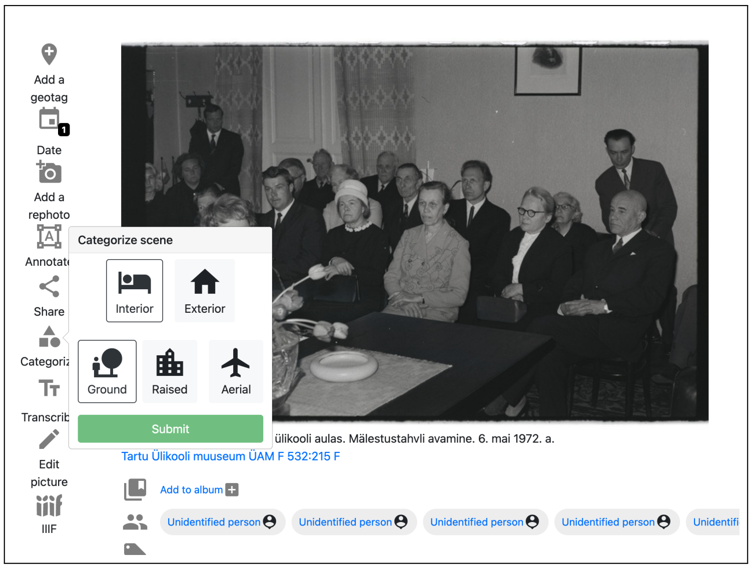


Figure 4. *Ajapaik* image having category labels *interior* and *ground*

After new user categorizes the image as *exterior*, the image no longer belongs to the *interior* category.

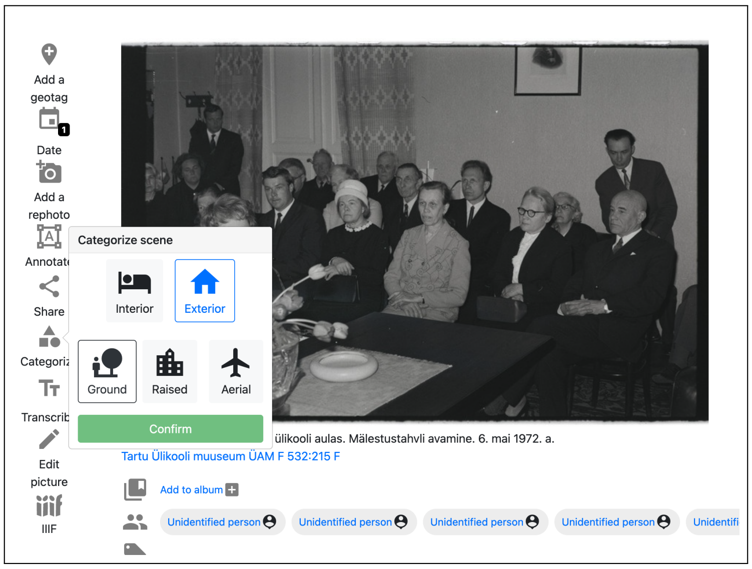


Figure 5. *Ajapaik* image having category labels *exterior* and *ground*

Hence, the ultimate categorization effort effectively determines the image's definitive category, allowing it to be filtered and searched accordingly.

The existing system functionality accommodates the inclusion of potentially erroneous categorizations, preserving them within the system and treating them as the established *ground truth*.

Furthermore, a significant number of images remain uncategorized, essentially avoiding classification and the ability to be discovered through any search filter. In our analysis of the *Ajapaik* platform, we observed that out of the 1 121 575 images currently accessible, only 43.19% feature a *scene* category label, and merely 37.16% include a *viewpoint elevation* label. This finding underscores the necessity for creating an automated categorization solution. Such a solution will ensure that both existing and future image uploads are systematically categorized, greatly enhancing the user experience by simplifying searches and facilitating image grouping.

Total images	**1,121,575**
Images categorized as exterior	74,016
Images categorized as interior	410,371
Images categorized as ground	381,433
Images categorized as raised	31,137
Images categorized as aerial	3,493

Table 2. Categories presence for *Ajapaik* images

Despite the current availability of nearly 1.2 million images on the *Ajapaik* platform, with 43.19% categorized as *scene* and 37.16% as *viewpoint elevation*, it proved challenging to determine the extent of misclassification within the entire categorized dataset. To provide approximate estimates, we randomly selected 3 000 images for each category (*interior*, *exterior*, *ground*, *raised*, *aerial*), totaling 15 000 images. We conducted a thorough manual review to assess the accuracy of user categorization. The results are summarized below (Table 3):

Category	**Incorrectly Categorized**	**Percentage**
Interior	415/3000	13.83%
Exterior	186/3000	6.20%
Ground	204/3000	6.8%
Raised	215/3000	7.17%
Aerial	275/3000	9.17%
Total	**1295/15000**	**8.63%**

Table 3. Incorrect users categorization

4 Analysis and realisation

In this chapter, we provide an overview of *Ajapaik* data (section 4.1), user stories (section 4.3), technical requirements overview (section 4.4) and the solution devised to address the challenge that serves as the driving force behind this thesis (section 4.6). We also discuss alternative approaches, their pros and cons for tackling the thesis objectives (section 4.7).

4.1 *Ajapaik* Data

For our thesis, we were provided with a snapshot of *Ajapaik* images from the platform, consisting of a total of 11 670 images. Our task was to manually categorize these images into five distinct subcategories, namely *exterior* or *interior*, and *aerial*, *ground*, or *raised.* After the manual categorization process, the distribution of images appeared as follows:

Scene	**Image Count**
Interior	3,056
Exterior	8,614
Viewpoint Elevation	**Image Count**
Ground	10,670
Raised	901
Aerial	99

Table 4. *Ajapaik* images category distribution after manual categorization

From manual categorisation results, it could be vividly seen that distribution of categorized images count is very different with *exterior* being 8614 images and *aerial* being only 99 images. To mitigate the issue, we took advantage of open data *Ajapaik* API [29] and managed to fetch additional images where in addition users category choices were present.

Our process of acquisition involved making requests to the *Ajapaik* open API, filtering the retrieved images based on user category suggestions, and reviewing them to ensure the accuracy of user feedback. When necessary, we excluded images that did not meet the required criteria.

This effort resulted in the acquisition of an additional 4856 images for the *raised* category and 3759 images for the *aerial* category.

Categorized images play a pivotal role by serving as the *ground truth*, which is subsequently employed in the initial training and validation processes of models described precisely in chapter 5.

4.2 Planning

In our envisioned approach, we aspire to rectify the *Ajapaik* limitations (described in section 3.2) by incorporating a model prediction component. This element will take on the pivotal role of forecasting image categories and will serve as the definitive authority in this context. Model element will ensure both existing and future images on *Ajapaik* platform will be labeled with model predicted category.

In the implementation process, our focus extends beyond design planning and its implementation to emphasize continuous model improvement through periodical retraining. This retraining involves utilizing users feedback on the model's prior category predictions. A comprehensive explanation of this process can be found in the details outlined in subsection 4.6.3.

Furthermore, while we are enriching the manual predictions by users, we are committed to safeguarding the process against potentially erroneous user feedback. This will be accomplished through our *Data Quality Engine*, which will filter out any flawed user input, ensuring that it does not progress to the subsequent stages of model retraining. Given diagram (Figure 6) provides a high level overview of key components of interest in terms of the thesis work.

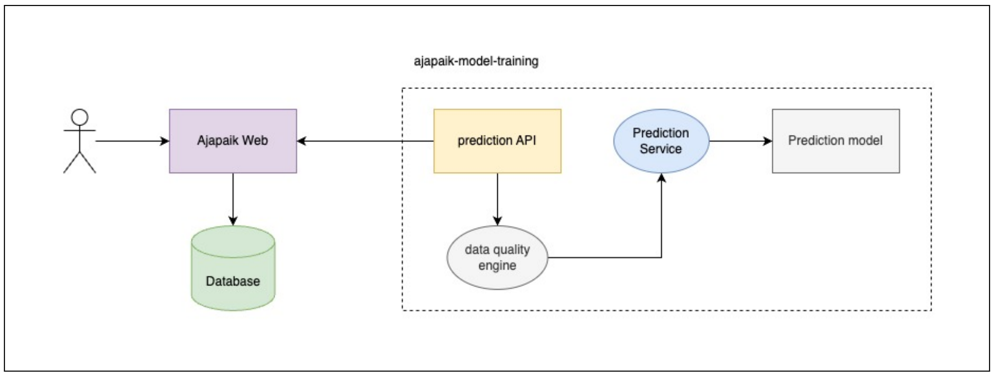


Figure 6. Components of interest, simplified overview

4.3 User stories

To fulfill the objectives of the thesis, we defined the intended user experience on the *Ajapaik* platform. This desired behavior was conceptualized through user stories, presenting the following:

User Story 1	As an *Ajapaik* platform user, I want to easily see what category a model predicts for any image on the *Ajapaik* platform.
Acceptance Criteria	1. User can view model predicted categories for all already existing images on the *Ajapaik* platform. 2. User can view model predicted categories specifically for newly uploaded images on the *Ajapaik* platform. 3. User can view if model's predicted category contradicts the category proposed by a previous user.

Table 5. User Story 1: View model predicted category

User Story 2	As an *Ajapaik* platform user, I want to be able to confirm model predicted categories or propose alternative categories for an image.
Acceptance Criteria	1. User can view model predicted categories for all already existing images on the *Ajapaik* platform. 2. User is able to confirm the predicted categories by selecting the same categories as the model suggested. 3. User is able to propose different categories by choosing alternative categories from what the model suggested.

Table 6. User Story 2: Confirm/propose alternative categories for model categories prediction

4.4 Requirements overview

Before the implementation part, together with the *Ajapaik* team we have set up technical requirements for further software development phase.

View on model category prediction (UI):

- **Toolbox view:**
  - If model category result for an image is present, there exists a model icon indicating categorisation has AI-input

    - If model category result for an image is present, but user has submitted different categorization for an image, there exists a discrepancy icon indicating difference in opinions.
- **Image dialogue view:**
    - If model category result for an image is present, there exists a model icon on related categories indicating categorisation has AI-input. Model predicted categories have black borders.
    - If model category result for an image is present, but user has submitted different categorization for an image, current user categories are active (colored in blue color), model predicted categories have model icon and black borders.
    - If model category result for an image is present, user is able to confirm model predications or propose alternative categories.

Server Side:

- The standalone model component is envisioned to be an independent entity, residing within the *ajapaik-analytics-server*.
- The model component should store its trained model state and seamlessly utilize the saved model upon initialization.
- Model component should be able to predict categories for both *scene* and *viewpoint elevation* category classes.
- Periodically, the model component should retrieve recently uploaded images on the *Ajapaik* platform that lack model categorization.
- The model component should categorize fetched images, encompassing both *scene* and *viewpoint elevation*.
- The model component should seamlessly integrate with the *ajapaik-web* component, allowing the storage of model image predictions in the datastore.
- The model component should conduct periodic retraining. This involves fetching images with user feedback received during a specified time period, processing them, and subsequently conducting model retraining.

4.5 Technologies

During the thesis *Python* [30] as a main development language was used for the implementation model category prediction component - *ajapaik-model-training* [31]. Additionally we took advantage of some *Python* libraries and tools such as:

- *Django* [32]- for building web application

- *Tensorflow* [33] - for building, training, and deploying machine learning and deep learning models
- *Keras* [34] - for developing and training deep learning models
- *Numpy* [35] - for efficient and high-performance numerical computations
- *APScheduler* [36] - for automating and managing task scheduling for image category prediction and model retraining
- *Scikit-learn* [37] - for machine learning and data analysis
- *Typing-extensions* [38] - for adding advanced type hinting capabilities for the codebase
- *Pillow* [39] - for opening, manipulating, and saving various image file formats
- *Plotly* [40] - for creating interactive and visually appealing data visualizations and charts
- *Pandas* [41] - for data manipulation and analysis
- *Requests* [42] - for managing HTTP requests

The *ajapaik-web* [43] project, based on *Django* for the backend and *jQuery* [44] for the frontend, has undergone thesis-related modifications specifically to incorporate model category predictions and gather user feedback on category assignments.

4.6 Implementations

This chapter furnishes a technical overview of the implemented solutions designed to support the objectives and motivations of the thesis.

4.6.1 Introduction of *ajapaik-model-training*

To achieve the objectives of the thesis, a distinct microservice named *ajapaik-model-training*[1] was introduced. This component was intentionally designed as a separate entity from main *ajapaik-web*[2] component for a variety of compelling reasons, aligning with the principles of Software Architecture SOLID [45]. Furthermore, considering technical specifications essential for server performance, this developed component was designated to reside within the *ajapaik-analytics-server*. In contrast to the *ajapaik-prod-server*, the *ajapaik-analytics-server* offers enhanced capabilities in terms of GPU, CPU, and memory resources. The rationale behind the decision to create this component as an isolated service will be elaborated on in the forthcoming subsections.

Firstly, in terms of infrastructure, the *ajapaik-model-training* component is strategically placed within the *ajapaik-analytics-server* to leverage its superior GPU, CPU, and memory

[1]https://github.com/angrun/ajapaik-model-training
[2]https://github.com/Ajapaik/ajapaik-web

resources. This infrastructure choice enhances the performance of the model training process compared to the *ajapaik-prod-server*, thereby optimizing the utilization of hardware capabilities.

Secondly, from an architectural perspective, the isolation of the *ajapaik-model-training* microservice promotes adherence to the SOLID principles. By segregating model training functionalities into a dedicated service, the system achieves better modularity and maintainability.

In the following sections, these reasons will be expounded upon in detail, distinguishing between infrastructure considerations (such as GPU utilization) and architectural considerations, providing a comprehensive understanding of the rationale behind the decision to create *ajapaik-model-training* as a distinct microservice.

Single Responsibility Principle [46]**:** ***ajapaik-model-training***

Argument: Isolating model prediction in the *ajapaik-model-prediction* repository.
Benefit: Streamlined maintenance, consolidated logic, potential for further long-term reuse.

Open-Closed Principle [46]**:** ***ajapaik-model-training***

Argument: Microservice facilitates extension without modifying existing code. Extension for additional category classes.
Benefit: Stability, potential risks mitigation and streamlined feature introduction and enhancements.

Liskov Substitution Principle [46]**:** ***ajapaik-model-training***

Argument: Microservice design aligns with Liskov Substitution Principle for easy substitution.
Benefit: Simplifies integration of new models or transitions to alternative models.

Interface Segregation Principle [46]**:** ***ajapaik-model-training***

Argument: Model prediction as a focused microservice adheres to Interface Segregation Principle.
Benefit: Simplicity and focused implementation.

Dependency Inversion Principle [46]**:** ***ajapaik-model-training***

Argument: Microservice design aligns with Dependency Inversion Principle for component dependencies.
Benefit: Maintenance and modification without impacting other project components.

Taking all of the arguments and benefits into account, *ajapaik-model-training* was therefore introduced as a separate microservice residing in *ajapaik-analytics-server* for greater performance capabilities.

4.6.2 *ajapaik-model-training* model technical specification

As an initial stride toward realizing the thesis objectives, a model for image category prediction was created. As explained earlier (subsection 4.6.1), this model resides within a specific entity called *ajapaik-model-training.*

The implemented model, used for predicting image categories, leverages a convolutional neural network (CNN [47]) architecture, specifically *MobileNetV2*, to recognize and distinguish the visual characteristics of classes. The model's goal is to determine whether a given image portrays an *interior* environment or an *exterior* one (or alternatively *ground*, *raised* or *aerial*).

The model architecture is composed of several layers, including a pre-trained *MobileNetV2* base model and additional fully connected layers. The base model has learned features from a large dataset, which can be beneficial for recognizing class-related patterns. The fully connected layers further process these features to make a final classification decision.

The model is optimized and evaluated using the *Adam* [48] optimizer and cross-entropy loss, while metrics such as accuracy and F1 score are used to assess its performance. It undergoes 20 epochs of training, and early stopping and learning rate reduction strategies are implemented to enhance training efficiency. Once trained, the model is cached for later use and retraining purposes.

ajapaik-model-training model architecture

During the course of the thesis, we have conducted a series of tests (detailed precisely in chapter 5) utilizing different implementations of deep learning models: *MobileNetV2*, *ResNet50* [49], and *AlexNet* [50] with the purpose to eventually keep the model producing most accurate prediction for our problem. Here we aim to provide a small overview of models and their architecture.

MobileNetV2

MobileNetV2 is a lightweight convolutional neural network (*CNN*) designed for efficient use mainly on mobile and embedded devices. It utilizes depthwise separable convolutions to reduce computational complexity while maintaining its accuracy. This makes *MobileNetV2* well-suited for applications where computational resources are limited, without degradation in performance.

ResNet50

ResNet50 (Residual Network) is another convolutional neural network, which introduced an architecture with residual learning blocks. This design facilitates the training of relatively deep neural networks by tackling the vanishing gradient problem. *ResNet50* skip connections allow information to flow directly through the network. Given network also supports training of deep models with thousands of layers.

AlexNet

AlexNet is a pioneering convolutional neural network. Its architecture consists of multiple convolutional and fully connected layers. *AlexNet* played an important role in popularizing deep learning and laid the foundation for subsequent advancements in the field. Despite being a only decade old and having fewer parameters compared to more recent models, *AlexNet* serves as a simpler baseline for comparison, providing a clear reference point for evaluating the performance of more complex and modern architectures in the scope of the thesis.

In the upcoming validation (chapter 5), we assess each implementation and present our conclusive findings.

4.6.3 *ajapaik-model-training* model retraining

In the implementation of the model employed for predicting image categories, a deliberate choice was made to maintain its constant accuracy and relevance by integrating a continuous feedback mechanism. This feedback system is based on users' responses, specifically their evaluations of the model's accuracy in predicting image categories. This dynamic functionality empowers the model to consider and incorporate user feedback, integrating it into the model retraining cycle to enhance and refine its performance over time.

4.6.4 *Data Quality Engine*

To ensure that the model leverages only the most reliable and accurate user feedback for subsequent retraining, our approach involves the incorporation of a so-called filtering layer, referred to as the *Data Quality Engine.* This essential layer serves as a gatekeeper, responsible for filtering out potentially erroneous user feedback, preventing it from being integrated into the model retraining cycle. Throughout the work of this thesis, we implemented various iterations of the *Data Quality Engine*, each designed to enhance the reliability and trustworthiness assured during the validation phase, as explained more in chapter 5.

This intermediate layer represents an innovative addition to the thesis work, diverging from the initial technical specifications. The decision to incorporate an additional filtering layer arose intuitively, driven by the desire to continuously refine the model and leverage user feedback for learning. The introduction of this *Data Quality Engine* resonates with the methodology discussed in the paper "Identifying Mislabeled Data using the Area Under the Margin Ranking" [14].

In the following section, we provide an overview of all the distinct algorithmic versions that were developed and tested throughout the thesis.

***Data Quality Engine v1*: Most Common Verdict**

The given *Data Quality Engine* focuses on the relatively simple "most popular category" principle. Through all possible categorization feedback submitted by users, the given categorization engine aims to exclude the least popular feedback per *image_id*, *user_id*, and forward the rest to the model retraining cycle.

Algorithm 1: *Data Quality Engine v1*: Most Common Verdict

```
Input   : Users' feedback $user\_feedback$
Output  : Cleaned feedback $cleanup\_feedback$, Removed feedback
          $removed\_feedback$
$verdict\_counts\_matrix \leftarrow$ empty matrix;
foreach $feedback$ in $user\_feedback$ do
    $image\_id \leftarrow feedback.image\_id$;
    $verdict \leftarrow feedback.verdict\_scene$;
    $verdict\_counts\_matrix[image\_id][verdict]$ += 1;
$most\_common\_verdicts \leftarrow \{\}$;
foreach $image\_id, verdict\_counts$ in $verdict\_counts\_matrix$ do
    /* findMostCommon determines the most common verdict among all
       verdicts per single image.                                   */
    $most\_common\_verdicts[image\_id] \leftarrow$ findMostCommon($verdict\_counts$);
$cleanup\_feedback \leftarrow []$;
$removed\_feedback \leftarrow []$;
foreach $feedback$ in $user\_feedback$ do
    if $most\_common\_verdicts[feedback.image\_id] == feedback.verdict\_scene$
     then
        append $feedback$ to $cleanup\_feedback$;
    else
        append $feedback$ to $removed\_feedback$;
return $cleanup\_feedback, removed\_feedback$;
```

```
[user_id: 1, image_id: 365, verdict_scene: 1, verdict_view_point_elevation: None]
[user_id: 2, image_id: 365, verdict_scene: 1, verdict_view_point_elevation: None]
[user_id: 3, image_id: 365, verdict_scene: 0, verdict_view_point_elevation: None] # excluded
```

Figure 7. Most Common Verdict algorithm data processing

Given example (Figure 7) indicates that feedback from *user_id*: 3 is excluded from further processing as it does not match the majority of the responses for the given image.

Data Quality Engine v2: anomaly detection

The given *Data Quality Engine* focuses on anomaly detection principle. It employs an anomaly detection technique using Isolation Forest [51] - an anomaly detection algorithm that isolates outliers in a dataset by constructing shallow, random decision trees to exclude potentially faulty user feedback. It processes user feedback data, calculates a ratio-based feature, standardizes it, and identifies potentially faulty user feedback.

The process involves the following steps:

1. ***Feature Engineering:*** We calculate a feature for each user, which is a ratio of *verdict_view_1_count* to the total count of *verdict_view_point_elevation* values (0, 1, and 2) (same for *scene* category).

2. ***Anomaly Detection:*** The Isolation Forest is applied to these features. Users' feedback is considered potentially faulty if the Isolation Forest assigns a prediction score of -1 to their feature vector. The Isolation Forest identifies data points that are far from the norm as anomalies, and a prediction score of -1 indicates that the user's feedback data is an outlier.

3. ***Separation:*** Feedback entries from potentially faulty users are excluded and the entries are considered as potentially faulty feedback.

Algorithm 2: *Data Quality Engine v2*: anomaly detection

```
Input    : Users' feedback $feedback\_data$
Output   : Cleaned feedback $cleaned\_feedback$, Removed feedback $faulty\_feedback$
$user\_data \leftarrow \{\}$;
foreach $entry$ in $feedback\_data$ do
    $user\_id \leftarrow entry.user\_id$;
    if $user\_id$ not in $user\_data$ then
        $user\_data[user\_id] \leftarrow$ {'verdict_scene_0_count': 0, 'verdict_scene_1_count': 0};
    if $entry.verdict\_scene == 0$ then
        $user\_data[user\_id]['verdict\_scene\_0\_count'] += 1$;
    else
        $user\_data[user\_id]['verdict\_scene\_1\_count'] += 1$;
$features \leftarrow []$;
foreach $user\_id, counts$ in $user\_data$ do
    $ratio \leftarrow counts['verdict\_scene\_1\_count']/(counts['verdict\_scene\_0\_count'] +$
      $counts['verdict\_scene\_1\_count'])$;
    $features$.append($[ratio]$);

/* The standard score of a sample x is calculated using formula:  z = (x -
   u) / s; u - the mean of the training samples, s - the standard deviation
   of the training samples.                                                */

$scaler \leftarrow$ StandardScaler()
$scaled\_features \leftarrow scaler$.fit_transform($features$);
$clf \leftarrow$ IsolationForest(contamination=0.05)
$clf$.fit($scaled\_features$);
$faulty\_users \leftarrow \{\}$;
foreach $i, prediction$ in enumerate $clf$.predict($scaled\_features$) do
    if $prediction == -1$ then
        $faulty\_users$.add($list(user\_data$.keys()$)[i]$);
$cleaned\_feedback \leftarrow []$;
$faulty\_feedback \leftarrow []$;
foreach $entry$ in $feedback\_data$ do
    if $entry.user\_id$ not in $faulty\_users$ then
        $cleaned\_feedback$.append($entry$);
    else
        $faulty\_feedback$.append($entry$);
return $cleaned\_feedback, faulty\_feedback$;
```

Data Quality Engine v3: **model prediction involvement**

Given data quality implementation takes advantage of model relatively high accuracy achieved during model initial training. In this implementation, we prioritize the model's high accuracy rate as the primary decision factor for exclusion. Specifically, we retain only the feedback entries that align with the model's predictions, while discarding all others. The method used closely aligns with the principles described in the paper "Identifying Mislabeled Instances in Classification Datasets" [13].

Algorithm 3: *Data Quality Engine v3:* model prediction involvement

```
Input   : Users' feedback $feedback\_data$
Output  : Cleaned feedback $cleanup\_data$, Removed feedback $faulty\_feedbacks$
$faulty\_feedbacks \leftarrow []$;
$cleanup\_data \leftarrow []$;
foreach $feedback$ in $feedback\_data$ do
    $model\_prediction \leftarrow$
      DataQuality.get__image__prediction($feedback.image\_id$);
    if $feedback.verdict\_scene \mathrel{!}= model\_prediction$ then
        $faulty\_feedbacks$.append($feedback$);
    else
        $cleanup\_data$.append($feedback$);
return $cleanup\_data, faulty\_feedbacks$;
```

4.6.5 Model training component integration

Given diagram (Figure 8) provides a comprehensive overview of what was developed during the thesis work.

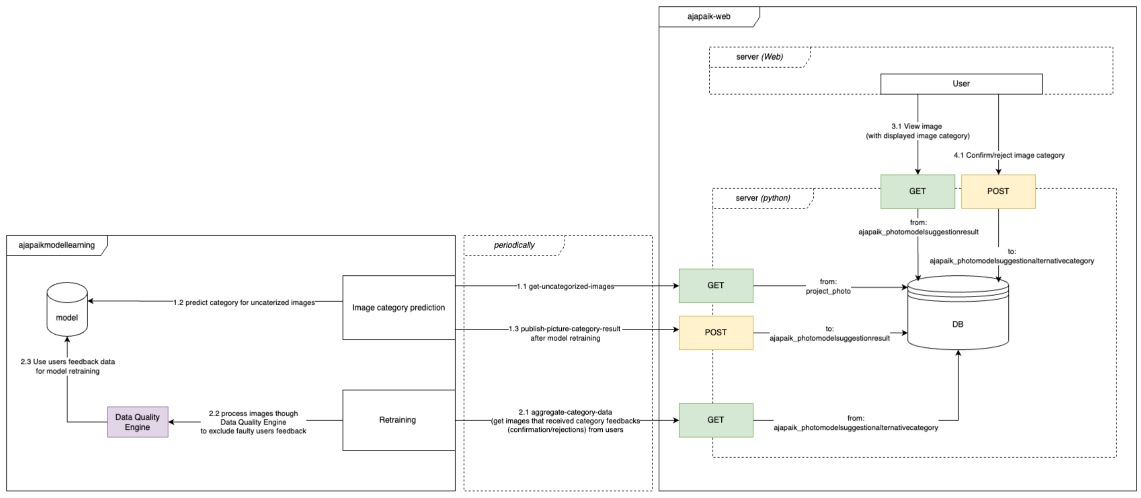

ajapaikmodellearning
model
1.2 predict category for uncaterized images
Image category prediction
2.3 Use users feedback data for model retraining
Data Quality Engine
2.2 process images though Data Quality Engine to exclude faulty users feedback
Retraining
periodically
1.1 get-uncategorized-images
1.3 publish-picture-category-result after model retraining
2.1 aggregate-category-data (get images that received category feedbacks (confirmation/rejections) from users
ajapaik-web
server (Web)
User
3.1 View image (with displayed image category)
4.1 Confirm/reject image category
GET
POST
server (python)
from: ajapaik_photomodelsuggestionresult
to: ajapaik_photomodelsuggestionalternativecategory
GET
from: project_photo
POST
to: ajapaik_photomodelsuggestionresult
DB
GET
from: ajapaik_photomodelsuggestionalternativecategory

Figure 8. *ajapaik-model-training* integration schema

- **1.1 - 1.3:** Given flow is responsible for determining category for yet uncategorized image
  - **1.1:** The *ajapaik-model-training* process regularly performs *GET* requests to the recently introduced */object-categorization/get-uncategorized-images* endpoint. This request fetches a batch of uncategorized images. This recurring call is set to run at intervals of every 1 minute.
  - **1.2:** After retrieving the images, the a*japaik-model-training* component is responsible for forwarding these uncategorized images to the model for image category predictions. Notably, predictions for the *scene* and *viewpoint elevation* categories occur independently, with no mutual exclusion between them.
  - **1.3:** Once the image categories are determined, the model predictions are saved in a newly introduced table *ajapaik_photomodelsuggestionresult*, which resides within the *ajapaik-web* repository. Here's an example of what the rows in the *ajapaik_photomodelsuggestionresult* table look like:

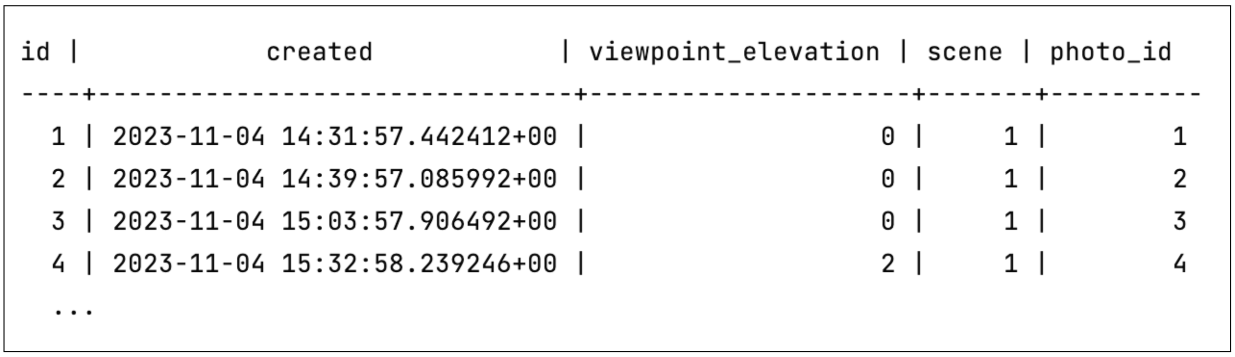
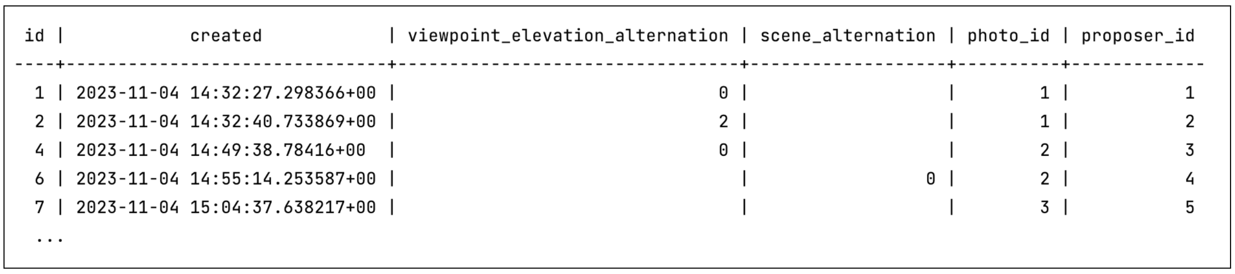

```
id |            created             | viewpoint_elevation | scene | photo_id
----+-------------------------------+---------------------+-------+----------
  1 | 2023-11-04 14:31:57.442412+00 |                   0 |     1 |        1
  2 | 2023-11-04 14:39:57.085992+00 |                   0 |     1 |        2
  3 | 2023-11-04 15:03:57.906492+00 |                   0 |     1 |        3
  4 | 2023-11-04 15:32:58.239246+00 |                   2 |     1 |        4
  ...
```

Figure 9. *ajapaik_photomodelsuggestionresult* table

- **2.1 - 2.3:** The provided flow is responsible for consolidating user feedback on various categories and facilitating subsequent model retraining.
  - **2.1:** In the ongoing *ajapaik-model-training* process, regular GET requests are made to the newly introduced */object-categorization/aggregate-category-data* endpoint. These requests retrieve a batch of images that have received user feedback regarding model-based image categorization. Users have the option to either confirm the model's category prediction or select a different category. This recurring call is scheduled to occur at specified intervals.
  - **2.2:** Subsequently, the *ajapaik-model-training* processes the received feedback through the *Data Quality Engine* (explained in subsection 4.6.4). The aim here is to filter out potentially faulty user feedback, ensuring that it does not influence the subsequent model retraining cycle.
  - **2.3:** In the final step of this stage, *ajapaik-model-training* provides the model with only trustworthy feedback. The model is then retrained using the reliable feedback.

- **3.1:** After the image category has been added to the *ajapaik_photomodelsuggestionresult* table, users can view the suggested class category for a particular image as predicted by the model.

- **4.1:** After the image categories have been added to the *ajapaik_photomodelsuggestionresult* table and are now visible on the *Ajapaik* user interface, users have the option to confirm the suggested category or propose a different one from what the model initially predicted. User feedback is collected and stored in the *ajapaik_photomodelsuggestionalternativecategory*, which is then utilized in the model retraining process. Here's an example of what the rows in the *ajapaik_photomodelsuggestionalternativecategory* table look like:



```
 id |            created            | viewpoint_elevation_alternation | scene_alternation | photo_id | proposer_id
----+-------------------------------+---------------------------------+-------------------+----------+-------------
  1 | 2023-11-04 14:32:27.298366+00 |                               0 |                   |        1 |           1
  2 | 2023-11-04 14:32:40.733869+00 |                               2 |                   |        1 |           2
  4 | 2023-11-04 14:49:38.78416+00  |                               0 |                   |        2 |           3
  6 | 2023-11-04 14:55:14.253587+00 |                                 |                 0 |        2 |           4
  7 | 2023-11-04 15:04:37.638217+00 |                                 |                   |        3 |           5
 ...
```

Figure 10. *ajapaik_photomodelsuggestionalternativecategory* table

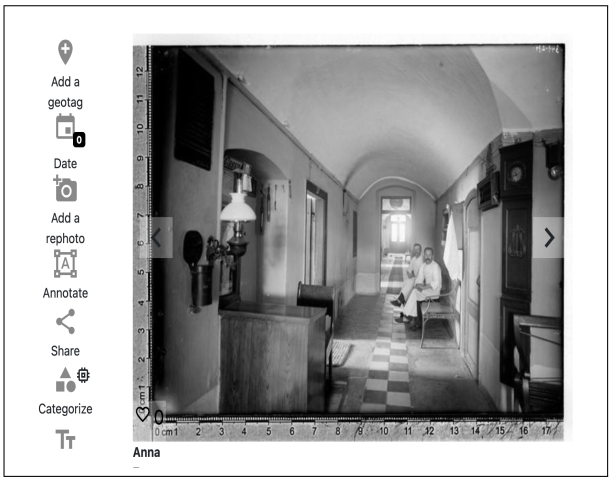
4.6.6 *Ajapaik* UI modifications

As part of the development of the model training component, we have implemented relevant user interface modifications to achieve the following objectives:

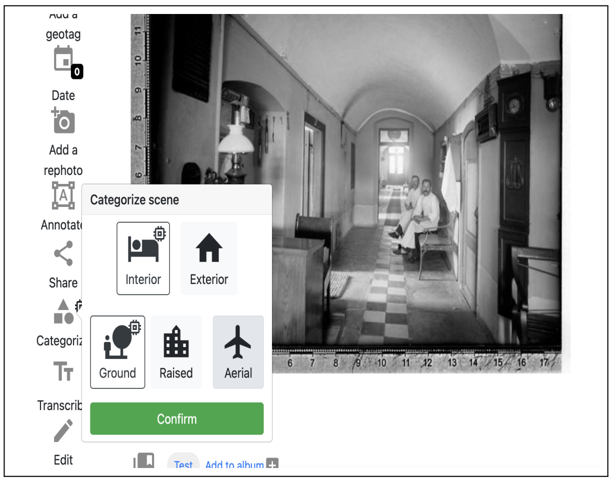
- Display the UI verdict for image category predictions
- Collect user feedback regarding the accuracy of model-based category predictions

Additionally, we have incorporated a feature to show the model's predictions, taking into account the presence or absence of user feedback. The following figures illustrate different scenarios for icon display:

1. After an image has been uploaded and the model successfully predicts a category, but no user feedback has been provided, only the model's category is displayed (Figure 11).



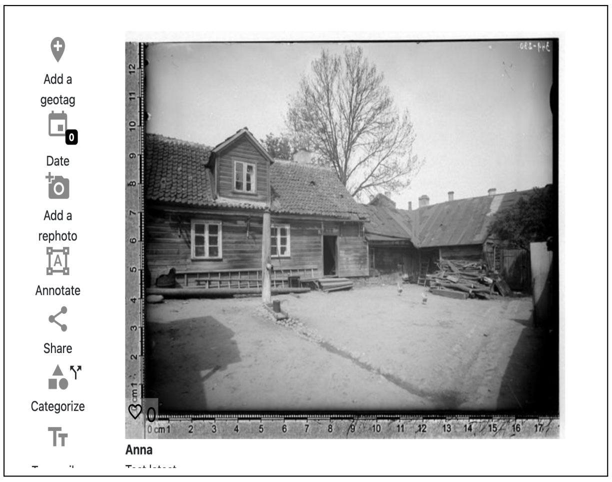
(a) Model result icon on toolbox



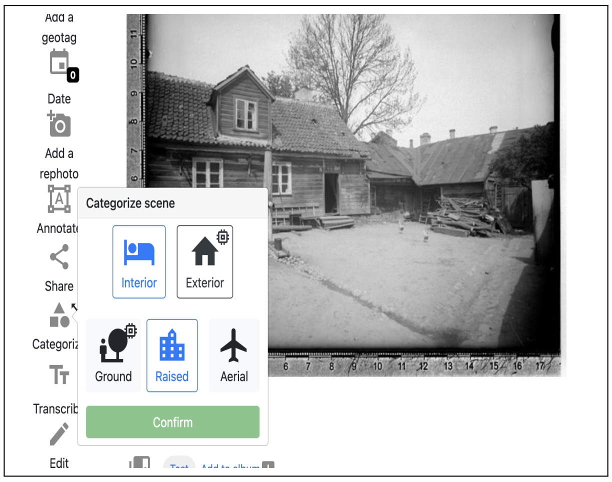
(b) Model result icon on confirmation view

Figure 11. Model categorization result presence icon

2. In cases where both the model and users have suggested categories for an image, but user feedback indicates that the model's prediction is incorrect, this is reflected in the display with the following view (Figure 12).



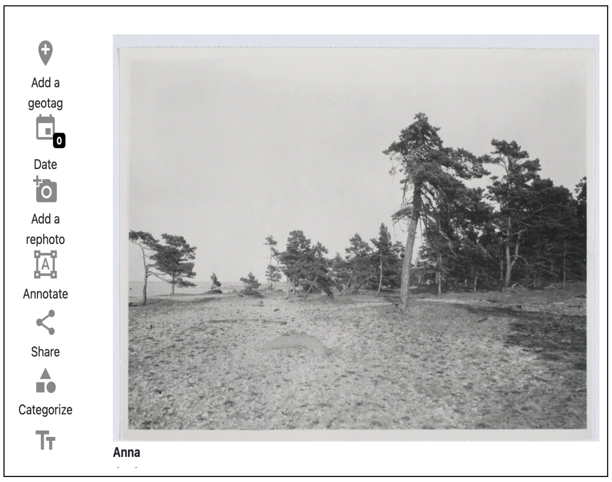
(a) Discrepancy icon on toolbox



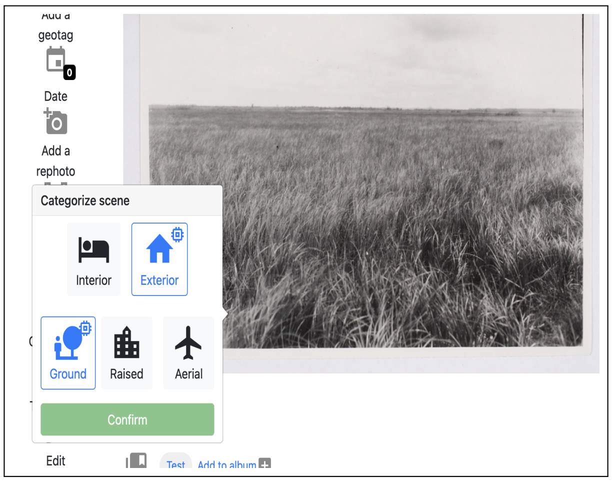
(b) Discrepancy icon on confirmation view

Figure 12. Model and user categorization discrepancy result presence icon

3. When both the model and users suggest the same category for an image, no additional icons are displayed (Figure 13).



(a) Icon absence on toolbox



(b) User and model prediction match

Figure 13. Model and user categorization match, icon absence

4.7 Alternative solutions

Given section provides a small overview of potential alternative solutions for the thesis objections.

4.7.1 Categorization constraint on image upload

One alternative for achieving the thesis goal and supporting image categorization would be introducing compulsory image categorization upon image upload. Users should not be able to upload any new image to the *Ajapaik* platform without proposing categories for the uploaded image.

Implications:

- All newly uploaded images will have categories assigned by the users. However, all already existing images on the *Ajapaik* platform will remain uncategorized.
- The current image upload functionality on *Ajapaik* allows users to upload one image at a time. Considering the potential evolution of the platform, with support for hosting images from archives and museums, the proposed implementation would introduce constraints on how images are uploaded. In the long run, it will become impossible to upload larger volumes of images at a time (as user would be required to manually add categories for each uploaded image), hence damaging the user overall platform experience.
- While the provided alternative guarantees the assignment of categories to newly added images, a mechanism to ensure the accuracy of categorization would still be absent.

4.7.2 Third party integration

Instead of introducing custom trained models, alternative solution would include integration towards third party AI solutions, such as: Google Cloud Vision API [52], Microsoft Azure Computer Vision API [53], Amazon Rekognition [54] or IBM Watson Visual Recognition [55]. These resources provide image analysis services, image labeling/categorization together with pre-trained models.

Implications:

- While these resources offer the advantage of seamless integration and accuracy in category predictions, they also come with associated costs. The non-commercial *Ajapaik* project would require additional funding to incorporate these technologies and sustain their operation.

4.7.3 Large language models

Following the paper "A multimodal turn in Digital Humanities: Using contrastive machine learning models to explore, enrich, and analyze digital visual historical collections" [56], the authors propose that multimodal models, exemplified by ChatGPT [57] and DALL-E [58], hold significant potential for application in tasks aligned with the objectives of our thesis.

Implications:

- These developments surfaced during the initiation of our thesis.
- The article emphasizes that while the results may not necessarily surpass existing methods, it does not imply inferiority. Rather, it underscores the need for separate prototyping and evaluation, which falls beyond the scope of the present work.

5 Results

In this chapter we bring up all of the results and findings achieved during the thesis. We also look into validation process details. Our validation methodology includes an evaluation of a developed software system, user interface, and model accuracy.

5.1 Software System Validation

During the course of the thesis, we developed a distinct component known as *ajapaik-model-training*, tasked with predicting image categories and executing model retraining. Additionally, we integrated it into the existing *ajapaik-web* application. A pivotal objective of our thesis was to not only guarantee the scalability of the model but also to minimize adjustments to the existing processes. We aimed to ensure that future maintenance or modifications to the model could be effortlessly executed. To review the success of our integrated solution, we systematically examine each facet. Architecture solution was reviewed and confirmed together with the *Ajapaik* team.

Scalability

Given that image categories are mutually exclusive and each image is expected to be categorized by both *scene* and *viewpoint elevation* category classes, we implemented two distinct categorization models. Looking ahead, we can benefit from long-term scalability, which makes adding new class category models alongside with enlarging categories within the category classes possible. This process involves composing ground truth data, training the model, adjusting the user interface, and ensuring seamless persistence of new category class records in the UI.

Minimum changes

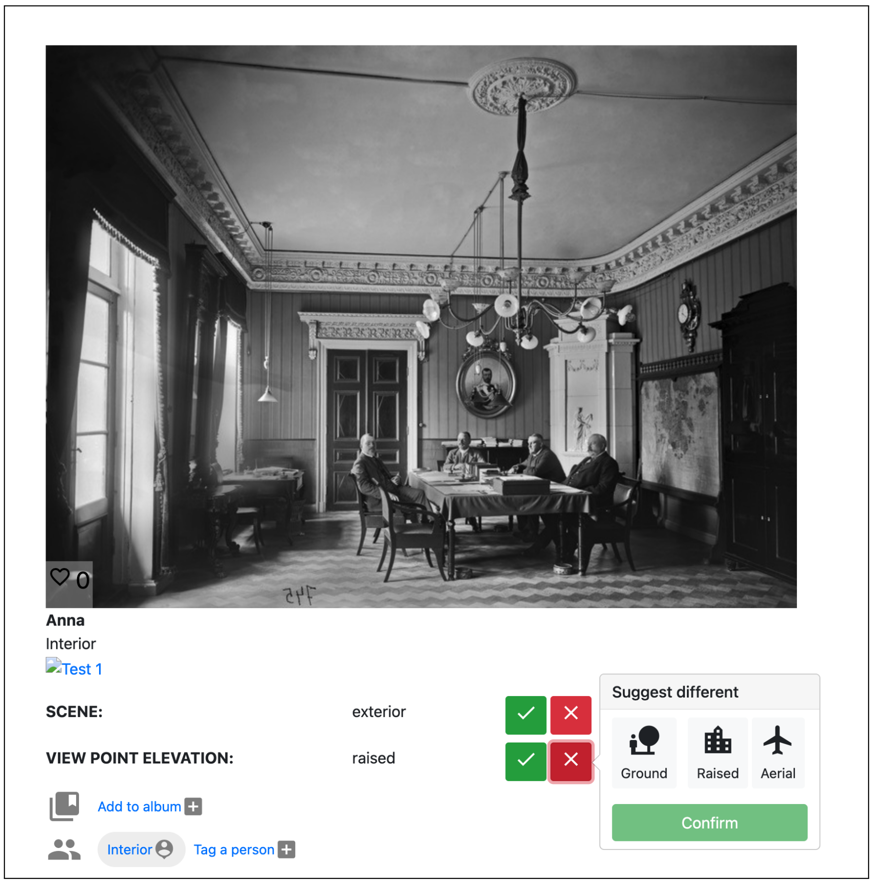
During the development of the *ajapaik-model-training* component and its integration into the *ajapaik-web* application, a focus was placed on minimizing changes to the existing codebase and processes. The objective was to ensure that the addition of the new categorization models and features did not disrupt the flow of the platform. This emphasis on minimum changes facilitates future maintenance and modifications, allowing for an efficient and adaptable system. We have also addressed situations in which the *ajapaik-web* application can maintain its operational status and preserve its state seamlessly, even in instances where the *ajapaik-model-training* has not yet commenced or is temporarily inactive for any reason.

5.2 *Ajapaik* UI validation

Throughout the process of UI modifications, in collaboration with the *Ajapaik* team, we iterated on potential UI solutions. This focused on determining how the model-predicted categories would be presented to users and how users could interact with and provide feedback on these predictions.

5.2.1 UI modifications: version 1

The initial version took the following form:



0
Anna
Interior
Test 1

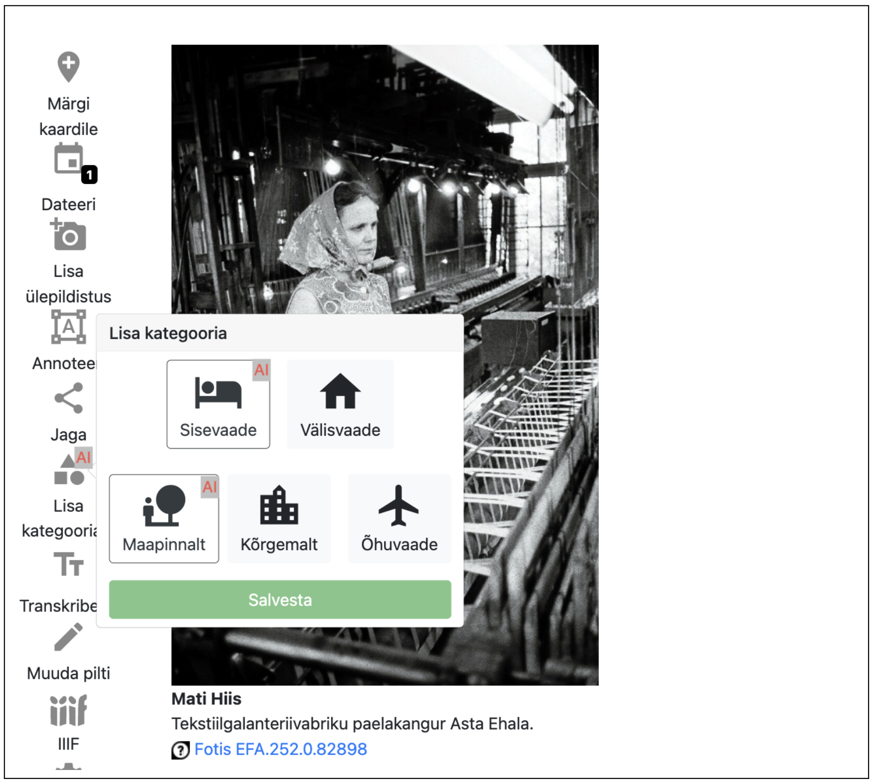
SCENE:	exterior	✓ ✕
VIEW POINT ELEVATION:	raised	✓ ✕

Figure 14. *Ajapaik* UI version 1

Despite the fact that the provided solution explicitly reveals the suggested categories for an image, upon analysis and feedback collaboration with the *Ajapaik* team, it became apparent that the layout was overly intricate. Additionally, it was recognized that the inclusion of confirm/reject buttons was unnecessary, as users should not only have the option to reject the proposed category but also suggest an alternative one. In essence, our goal was to maintain a more minimalistic approach, incorporating any modifications within the framework of the existing categorization input.

5.2.2 UI modifications: version 2

In the second iteration of UI modifications, our objective was to introduce changes by leveraging the already existing categorization solution. We opted to label the model-predicted categories with an "AI" label, indicating that this is the input from model. This allows users to make use of the "Submit" button, enabling them to either confirm the suggested category or propose alternative categories.

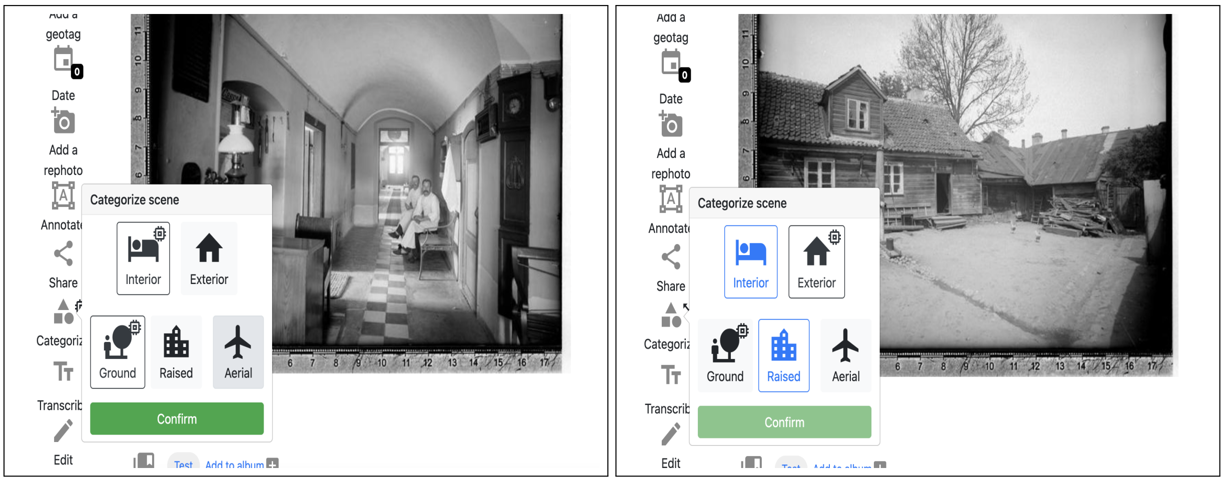


Mati Hiis
Tekstiilgalanteriivabriku paelakangur Asta Ehala.
Fotis EFA.252.0.82898

Figure 15. *Ajapaik* UI version 2

5.2.3 UI modifications: version 3

While the UI was nearly finalized, we chose to enhance the visual aspect slightly and capitalize on pre-existing icons for model input. Furthermore, we introduced an additional layer by displaying variances between model and user opinions using a distinct icon for clarity. The *Ajapaik* development team reviewed and approved the final result, which is as follows:



(a) Model result icon (b) Model result and user result discrepancy

Figure 16. *Ajapaik* UI version 3

5.3 *Data Quality Engine* Validation

5.3.1 *Ground truth* usage and users feedback generation

For conduction further validation, we have utilised previously manually categorised images (*ground truth*) as discussed in details in section 4.1.

To ensure a precise validation of the obtained results, we simulate user feedback specifically for the class category *scene* - *interior* and *exterior* and class category *viewpoint elevation* - *ground*, *raised* and *aerial*. We imitate users' model categories feedback submissions for each category class.

In the validation process, we simulate users categories submissions, as each *imaginary* user was granted the opportunity to submit a single feedback entry for each category class. The generation of user feedback was arbitrary, as our goal was to emulate the scenario where users submit predictions for images currently accessible on the Ajapaik platform.

As a result, we utilized a dataset that encompassed 1500 feedback (for each category class) entries for the random selection of 2038 images, which were gathered from a pool of 100 users.

Taking into account the availability of the ground truth data for image categories, we have the capacity to simulate user feedback, encompassing both accurate and inaccurate responses, and subsequently compare these results with the actual image categories.

The allocation of feedback for each test scenario has been more or less equal, adhering to the following distribution:

Category	**Correct**	**Wrong**
Exterior	371	379
Interior	364	386
Ground	261	239
Raised	243	257
Aerial	288	212

Table 7. Generated users feedback allocations

Where:

- ***[category]__correct:*** represents the count of correct feedback responses for images categorized as *[category]* in reality.
- ***[category]__wrong***: signifies the count of incorrect feedback responses for images categorized as *[category]* in reality.
- ***[category]__correct*** denotes the count of correct feedback responses for images categorized as *[category]* in reality.
- ***[category]__wrong*** indicates the count of incorrect feedback responses for images categorized as *[category]* in reality.

Following the simulation of image category feedback submissions and subsequent model retraining with user input, we proceed to execute a range of implementations of the *Data Quality Engine* layer.

We conduct an analysis to examine which feedback entries are excluded during subsequent model retraining, assess any discernible trends, and compare the outcomes of various *Data Quality Engine* implementations.

For better clarify, we asses *Data Quality Engine* algorithms validation by using recall (Equation 5.1), precision (Equation 5.2) and F1 score (Equation 5.3) calculations as key success indicators.

$$Recall = \frac{TP}{TP + FN} \tag{5.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{5.2}$$

$$F1 = \frac{recall \cdot precision}{recall + precision} \cdot 2 \tag{5.3}$$

Where:

- **True Positives (TP):** The number of feedback marked as incorrect by the *Data Quality Engin*e that were also actually incorrect.
- **True Negatives (TN):** The number of feedback not marked as incorrect by the *Data Quality Engine* that were actually correct.
- **False Positives (FP):** The number of feedback marked as incorrect by the *Data Quality Engine* that were actually correct.
- **False Negatives (FN):** The number of feedback not marked as incorrect by the *Data Quality Engine* that were actually incorrect.

In the ensuing chapters, we delve into each implementation in meticulous detail.

5.4 Model category prediction flow validation

In the upcoming chapter, we validate the accuracy of the model's category prediction flow, integrated with diverse implementations of the *Data Quality Engine*.

To validate the model's category prediction flow, we partitioned the composed *ground truth* images into three distinct categories:

MODEL TRAINING PHASE:

Images employed for the initial model training (33.33% of the *ground truth* dataset).

USERS FEEDBACK AND MODEL RETRAINING PHASE:

Images designated for uploading onto the *Ajapaik* platform, with users subsequently submitting category feedback (simulation of uses category feedback submissions) for a subset of these uploaded images (33.33% of the dataset).

RETRAINED MODEL PREDICTIONS PHASE:

Images reserved for the validation of the retrained model's predictions (33.33% of the dataset).

The accompanying figure (Figure 17) offers a comprehensive overview of the procedures carried out at each of these pivotal steps:

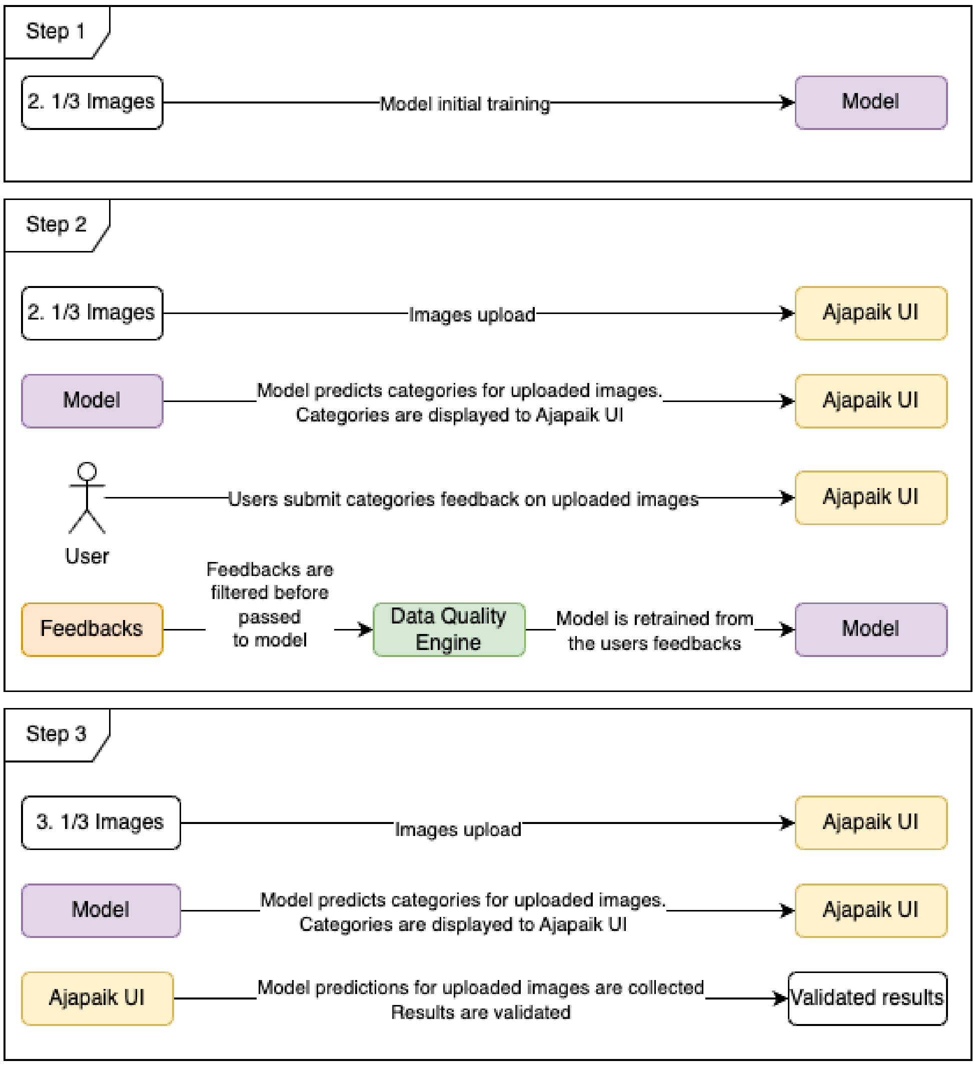

Step 1
2. 1/3 Images
Model initial training
Model
Step 2
2. 1/3 Images
Images upload
Ajapaik UI
Model
Model predicts categories for uploaded images. Categories are displayed to Ajapaik UI
Ajapaik UI
User
Users submit categories feedback on uploaded images
Ajapaik UI
Feedbacks
Feedbacks are filtered before passed to model
Data Quality Engine
Model is retrained from the users feedbacks
Model
Step 3
3. 1/3 Images
Images upload
Ajapaik UI
Model
Model predicts categories for uploaded images. Categories are displayed to Ajapaik UI
Ajapaik UI
Ajapaik UI
Model predictions for uploaded images are collected Results are validated
Validated results

Figure 17. Validation Schema

Where:

MODEL TRAINING PHASE:

Initialization and Training: When the model starts, it sets aside one-third of the available images for validation.

Outcome: This crucial step marks the beginning of model training, enabling predictions of image categories on the *Ajapaik* platform.

USERS FEEDBACK AND MODEL RETRAINING PHASE:

Data Ingestion: To facilitate the validation process, one-third of the available images is uploaded through the *Ajapaik* user interface. Given the current UI's constraint of single-image uploads, a temporary functionality has been introduced to enable high-volume image uploads.

Categorization: Upon successful image uploads, the model takes charge, systematically predicting categories for previously uncategorized images at defined intervals. These categories are promptly displayed on the user interface and stored in *Ajapaik* datastore *ajapaik_photomodelsuggestionresult*.

Simulated User Feedback: To enhance the model's learning process, we simulate user feedback submissions for the images uploaded in the previous step. The feedback is stored in the *ajapaik_photomodelsuggestionalternativecategory* table and later retrieved during the model retraining cycle. The precise methodology for generating user feedback is comprehensively detailed in subsection 5.3.1.

Retraining: During the subsequent model retraining cycle, the accumulated feedback is harnessed to refine the model's capabilities. This step results in the model being retrained, incorporating user feedback.

RETRAINED MODEL PREDICTIONS PHASE:

Additional Data Upload: In this phase, a last one-third of the images designated for validation (one third of the available images) are uploaded via the *Ajapaik* user interface.

Recategorized Output: Following the image uploads, the model, now retrained, predicts categories for previously uncategorized images. The categorization results are displayed on the user interface.

Validation and Comparison: The validation process encompasses the collection of categorization verdicts from the retrained model. These verdicts are meticulously compared against *ground truth* data, and the results are visually presented. This step serves the dual purpose of assessing the retrained model's accuracy and comparing it with the accuracy of the initial model.

The validation approach involves a lot of automation, making the process much smoother. Typically, it takes approximately 15 minutes to configure and verify the results.

In the following chapter we will follow these steps and apply them for different configurations of *Ajapaik* model implementation. Table 8 presents an overview of the test configurations executed throughout the validation process. The results summary of the validation is outlined in subsection 5.4.10, while detailed and specific results can be found in Table 12 and Table 13.

Validation set up	**Data quality engine version**	**Model architecture**	**Image augmentation**
Model category prediction flow validation: No Data Quality Engine Involved (subsection 5.4.1)	None	MobileNetV2	Original volume
Model category prediction flow validation: No Data Quality Engine involved, additional image generation (subsection 5.4.2)	None	MobileNetV2	Additional image generation
Model category prediction flow validation: No Data Quality Engine Involved, images downsampling (subsection 5.4.3)	None	MobileNetV2	Images downscaling - even volumes
Model category prediction flow validation: No Data Quality Engine Involved, viewpoint elevation images additional sourcing (subsection 5.4.4)	None	MobileNetV2	Images downscaling and additional sourcing - even volumes
Model category prediction flow validation: Data Quality Engine v1 (subsection 5.4.5)	v1: most common verdict	MobileNetV2	Images downscaling and additional sourcing - even volumes
Model category prediction flow validation: Data Quality Engine v2 (subsection 5.4.6)	v2: anomaly detection	MobileNetV2	Images downscaling and additional sourcing - even volumes

Model category prediction flow validation: Data Quality Engine v3 (subsection 5.4.7)	v3: model predictions involvement	MobileNetV2	Images downscaling and additional sourcing - even volumes
Model category prediction flow validation: Data Quality Engine v3 (subsection 5.4.8)	v3: model predictions involvement	ResNet50	Images downscaling and additional sourcing - even volumes
Model category prediction flow validation: Data Quality Engine v3 (subsection 5.4.9)	v3: model predictions involvement	AlexNet	Images downscaling and additional sourcing - even volumes

Table 8. Validation set up overview

5.4.1 Model category prediction flow validation: No *Data Quality Engine* Involved

The validation process outlined above focuses exclusively on assessing the model's category prediction flow without the integration of the *Data Quality Engine* layer. In this context, our objective is to ascertain the influence of incorporating the *Data Quality Engine* and to gauge the ultimate impact on the model, culminating in the final validation results.

VALIDATION CONFIGURATION:

MODEL TRAINING PHASE:

Model used: MobileNetV2

Images initially utilized during model training phase - (one third of possible amount from composed *ground truth* imageset for each category):

- *scene:*
  - *interior:* 1019
  - *exterior:* 2872

- *viewpoint elevation:*

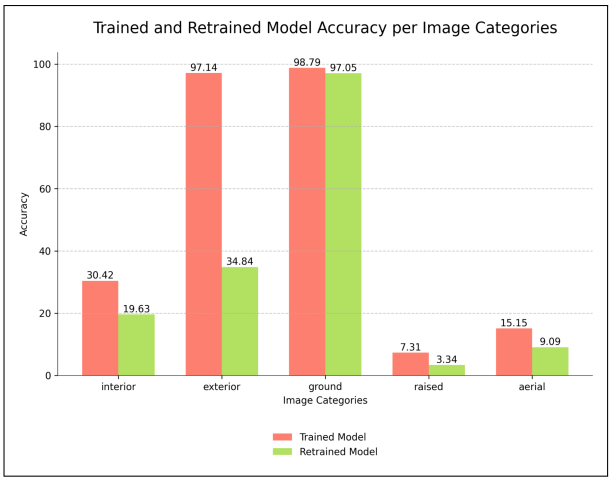
- *ground:* 3557
- *raised:* 301
- *aerial:* 33

USERS FEEDBACK AND MODEL RETRAINING PHASE:

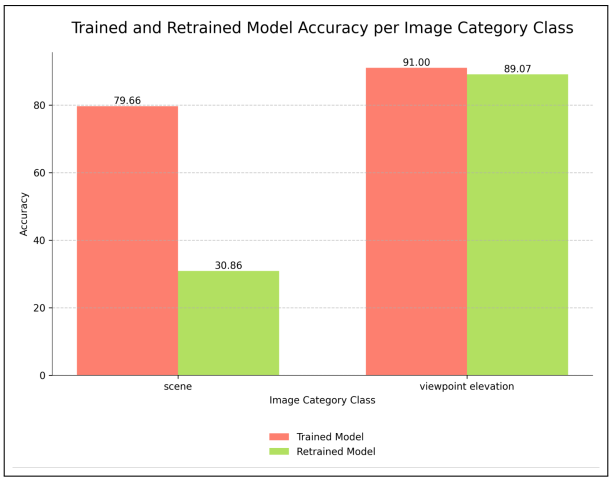
A second batch of images was uploaded, and the model's category predictions were displayed on the *Ajapaik* user interface. Subsequently, we simulated users feedback on image categories, and the model underwent a retraining process.

RETRAINED MODEL PREDICTIONS PHASE:

A fresh set of images was uploaded for validation purposes after the model had been retrained from users feedback. The results of this validation phase are outlined as follows:



(a) Per category



(b) Per category class

Figure 18. Validation results: No *Data Quality Engine* Involved

Conclusions

The outcomes of this experiment indicate that the initial model training process, involving uneven number of images per category class, has a significant impact on the accuracy of category prediction for the *interior*, *aerial* and *raised* classes. Furthermore, it is worth noting a significant decrease in the model's accuracy rate following the incorporation of unfiltered user feedback and subsequent retraining. In this context, it is clear that the distribution of images varies significantly, making it challenging to achieve uniform accuracy across all category classes. To mitigate the disparities in the number of images for different categories, our intention is to augment the image data by generating additional samples for categories with low image count.

5.4.2 Model category prediction flow validation: No *Data Quality Engine* involved, additional image generation

In light of the findings derived from the previous validation process (as visible in Figure 5.4.1), it becomes evident that there is a pressing requirement to achieve equilibrium in the quantities of *interior* and *exterior* and *ground*, *raised* and *aerial* images provided to the model during its initial training phase.

To fulfill this requirement, we have produced extra images for categories with a limited image count, ensuring alignment with the category boasting the highest number of images in both the *scene* and *viewpoint elevation* classes.

- *scene:*
  - *interior:* 2872 (+ 1853 generated images)
  - *exterior:* 2872

- *viewpoint elevation:*
  - *ground:* 3557
  - *raised:* 3557 (+ 3256 generated images)
  - *aerial:* 3557 (+ 3524 generated images)

- The image generation techniques employed encompass:
  - Image resizing
  - Image rotation
  - Image horizontal flipping
  - Image brightness adjustment
  - Image zooming

VALIDATION CONFIGURATION:

MODEL TRAINING PHASE:

Model used: MobileNetV2

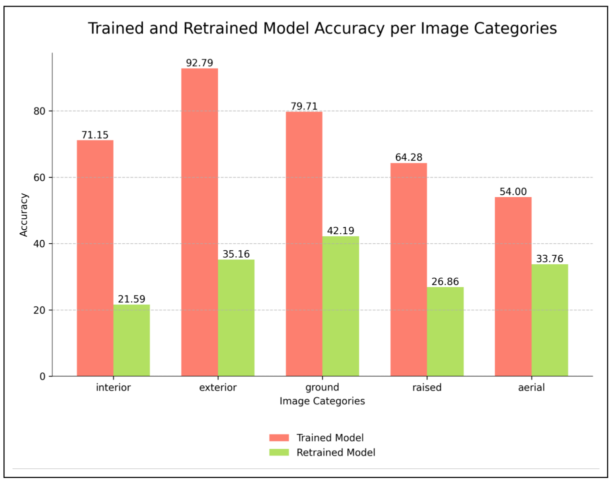
Images initially utilized during the model training phase.

USERS FEEDBACK AND MODEL RETRAINING PHASE:

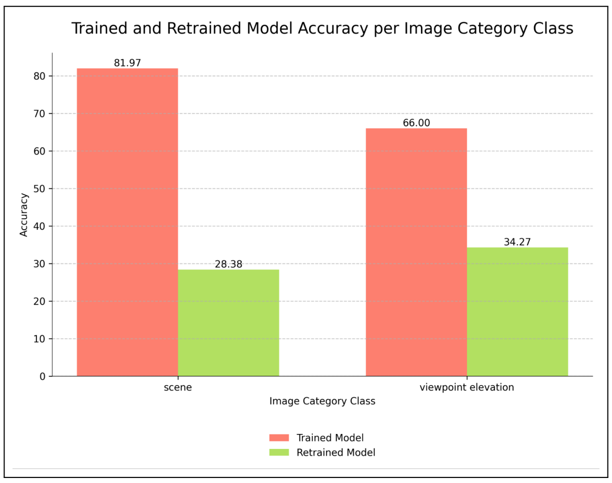
A second batch of images was uploaded, and the model's category predictions were displayed on the *Ajapaik* user interface. Subsequently, we simulated users feedback on image categories, and the model underwent a retraining process.

RETRAINED MODEL PREDICTIONS PHASE:

A fresh set of images was uploaded for validation purposes after the model had been retrained from users feedback. The results of this validation phase are outlined as follows:



(a) Per category



(b) Per category class

Figure 19. Validation results: No *Data Quality Engine* Involved, additional image generation

Conclusions

The generation of supplementary *interior*, *raised* and *aerial* category images has yielded remarkable improvements in the accuracy of the model after its initial training. It is noteworthy, however, that this enhancement has been accompanied by a marginal decline in the accuracy of *exterior* and *ground* image prediction. This decline can be attributed to the generated images closely resembling the source dataset, thereby lacking the diversity needed for robust training.

After retraining the model with the inclusion of user feedback categories, a consistent decline in accuracy rates may become apparent. As a next step, we aim to execute a validation where image count would be downsampled.

5.4.3 Model category prediction flow validation: No *Data Quality Engine* Involved, images downsampling

In this context, we have opted to conduct an additional experiment, whereby the supplied images for the initial model training have been deliberately downsized. This downsizing entails using a fixed count of images matching the category with the lowest image counts.

- *scene:*
  - *interior:* 1019
  - *exterior:* 1019 (- 1853 ground truth images)

- *viewpoint elevation:*
  - *ground:* 33 (- 3523 ground truth images)
  - *raised:* 33 (- 268 ground truth images)
  - *aerial:* 33

VALIDATION CONFIGURATION:

MODEL TRAINING PHASE:

Model used: MobileNetV2

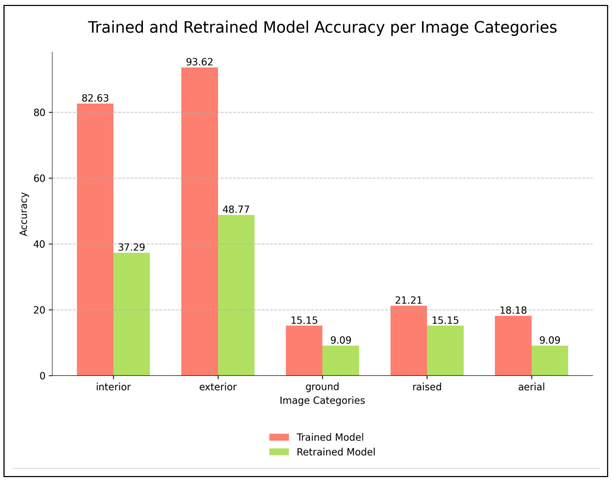
Images initially utilized during the model training phase.

USERS FEEDBACK AND MODEL RETRAINING PHASE:

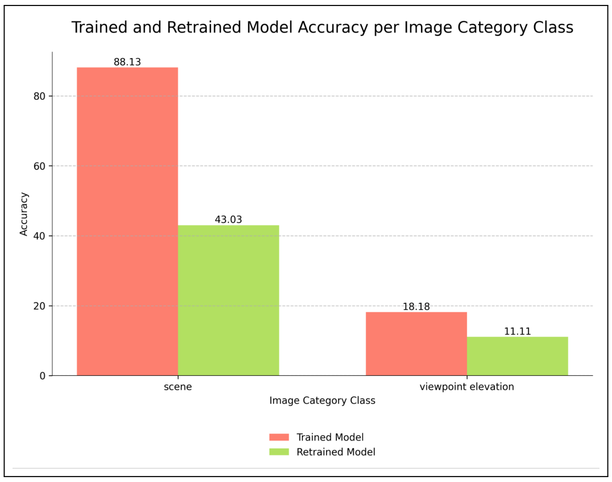
A second batch of images was uploaded, and the model's category predictions were displayed on the *Ajapaik* user interface. Subsequently, we simulated users feedback on image categories, and the model underwent a retraining process.

RETRAINED MODEL PREDICTIONS PHASE:

A fresh set of images was uploaded for validation purposes after the model had been retrained from users feedback. The results of this validation phase are outlined as follows:



(a) Per category



(b) Per category class

Figure 20. Validation results: No *Data Quality Engine* Involved, images downsampling

Conclusions

The validation process has shown that downsampling the data has notably enhanced the performance, particularly in the *scene* category class. However, when it came to the downsampled *viewpoint elevation* images, their limited quantity posed a challenge. The training process could not reach successful completion due to the scarcity of this specific data, resulting in validation results that fell considerably short of the expected ideal.

5.4.4 Addition: Model category prediction flow validation: No ***Data Quality Engine*** Involved, ***viewpoint elevation*** images additional sourcing

Due to a significant shortage of images in the *raised* and *aerial* categories among the initially provided image snapshots, we took the initiative to acquire additional images through the *Ajapaik* openAPI. Additional outsourcing became vital as we have revealed the initially provided snapshot from *Ajapaik* lacks diversity of images for *viewpoint elevation* category class (explained in details in section 4.1)

To maintain consistency and standardize our dataset, we opted to work with a common number of 1019 images for each category, which allowed us to proceed with our validation process.

- *scene:*
  - *interior:* 1019
  - *exterior:* 1019

- *viewpoint elevation:*
  - *ground:* 1019
  - *raised:* 1019
  - *aerial:* 1019

VALIDATION CONFIGURATION:

MODEL TRAINING PHASE:

Model used: MobileNetV2

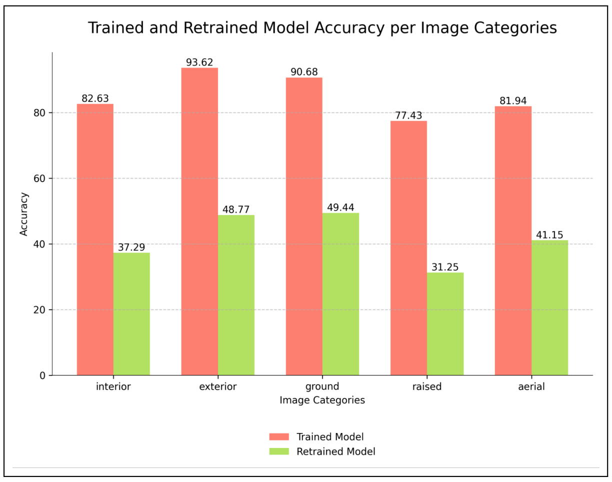
Images initially utilized during the model training phase.

USERS FEEDBACK AND MODEL RETRAINING PHASE:

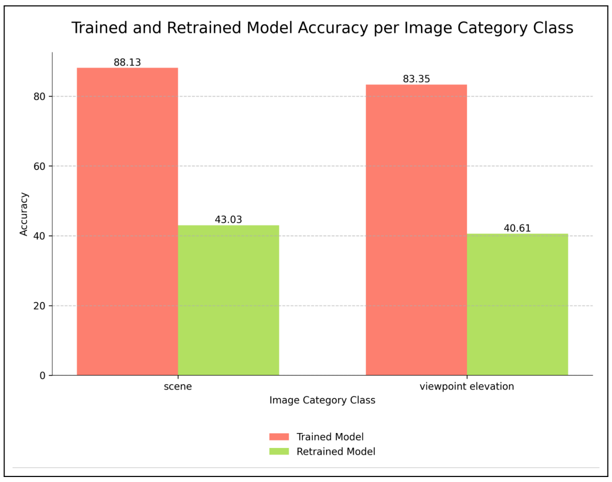
A second batch of images was uploaded, and the model's category predictions were displayed on the *Ajapaik* user interface. Subsequently, we simulated users feedback on image categories, and the model underwent a retraining process.

RETRAINED MODEL PREDICTIONS PHASE:

A fresh set of images was uploaded for validation purposes after the model had been retrained from users feedback. The results of this validation phase are outlined as follows:



(a) Per category



(b) Per category class

Figure 21. Validation results: No *Data Quality Engine* Involved, *viewpoint elevation* images additional sourcing

Conclusions

The validation process has shown considerable increase in accuracy for *viewpoint elevation* category class. Despite that, model accuracy after the retraining cycle remains under the desired bar. All following validation runs will therefore be executed using *Data Quality Engine* implementations maintaining constant images count for each category (1019 images).

5.4.5 Model category prediction flow validation: *Data Quality Engine* v1 - most common verdict

Within the given validation process, we employ the *Data Quality Engine* v1 (subsection 4.6.4) algorithm to effectively exclude feedback that pertains to faulty data.

VALIDATION CONFIGURATION:

MODEL TRAINING PHASE:

Model used: MobileNetV2

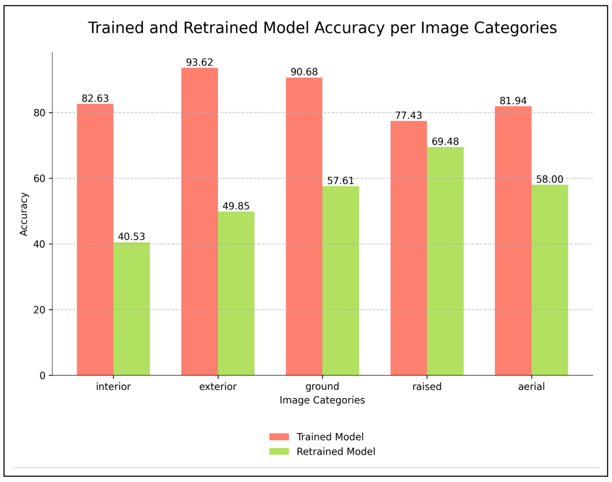
Images initially utilized during the model training phase.

USERS FEEDBACK AND MODEL RETRAINING PHASE:

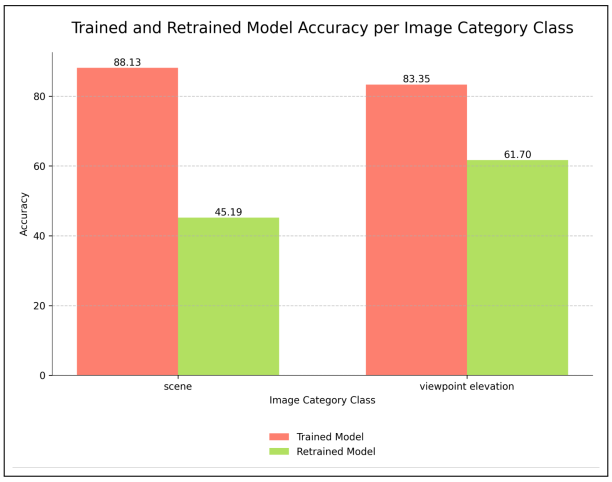
A second batch of images was uploaded, and the model's category predictions were displayed on the *Ajapaik* user interface. Subsequently, we simulated users feedback on image categories, and the model underwent a retraining process.

RETRAINED MODEL PREDICTIONS PHASE:

A fresh set of images was introduced for validation purposes after the model had been retrained based on user feedback. The results of this validation phase are outlined as follows:



(a) Per category



(b) Per category class

Figure 22. Validation results: *Data Quality Engine* v1- most common verdict

Conclusions

Within the given validation flow, *Data Quality Engine* v1 managed to exclude 195 feedback for *scene* class and 135 for *viewpoint elevation* class (out for 1500 total feedback for each category class). The precision, recall and F1 results are outlined in Table 9.

Category class	TP	TN	FP	FN	Precision	Recall	F1 Score
Scene	102	575	93	730	0.52	0.12	0.2
Viewpoint elevation	75	574	60	791	0.56	0.1	0.15

Table 9. *Data Quality Engine v1*: precision, recall, F1 score overview

The improvements are only marginal, with the overall accuracy rising from 44.30% to

45.19%. As explained in subsection 4.6.4 The *Data Quality Engine*'s implementation primarily emphasizes the most frequently encountered category in feedback. In light of this, if an image is associated with a higher frequency of faulty categories or is exclusively linked to faulty categories, it will be directed to the model and integrated into the model retraining process resulting in the model being negatively impacted.

5.4.6 Model category prediction flow validation: *Data Quality Engine* v2 - anomaly detection

Within the given validation process, we employ the *Data Quality Engine* v2 (subsection 4.6.4) algorithm to effectively exclude feedback that pertains to faulty data.

VALIDATION CONFIGURATION:

MODEL TRAINING PHASE:

Model used: MobileNetV2

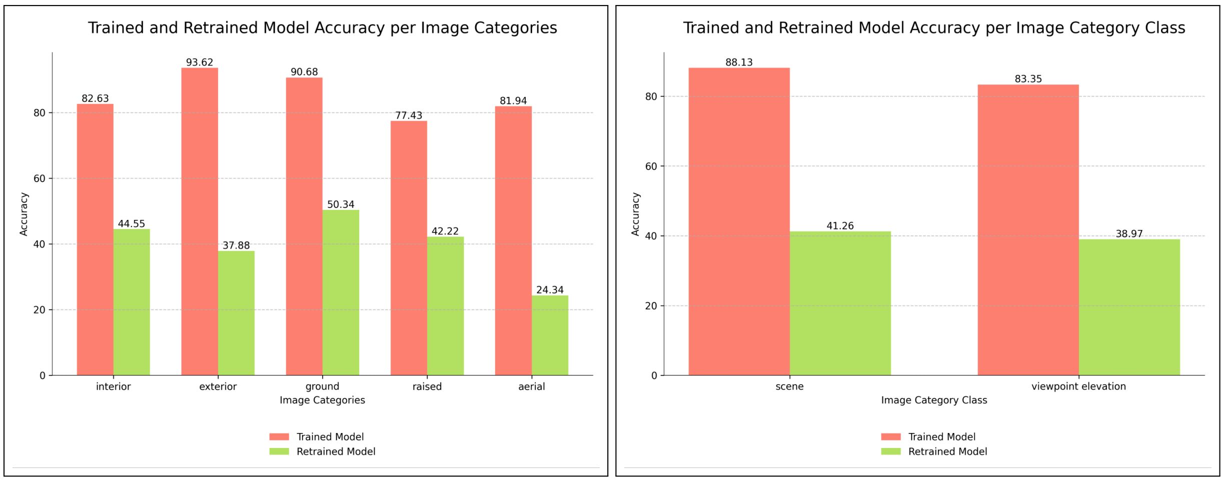
Images initially utilized during the model training phase.

USERS FEEDBACK AND MODEL RETRAINING PHASE:

A second batch of images was uploaded, and the model's category predictions were displayed on the *Ajapaik* user interface. Subsequently, we simulated users feedback on image categories, and the model underwent a retraining process.

RETRAINED MODEL PREDICTIONS PHASE:

A fresh set of images was introduced for validation purposes after the model had been retrained based on user feedback. The results of this validation phase are outlined as follows:



(a) Per category (b) Per category class

Figure 23. Validation results: *Data Quality Engine* v2 - anomaly detection

Conclusions

Within the given validation flow, *Data Quality Engine* v2 managed to exclude 99 feedback for *scene* class and 105 for *viewpoint elevation* class (out for 1500 total feedback). The precision, recall and F1 results are outlined in Table 10.

Category class	TP	TN	FP	FN	Precision	Recall	F1 Score
Scene	79	679	20	722	0.8	0.1	0.18
Viewpoint elevation	69	498	36	897	0.66	0.07	0.13

Table 10. *Data Quality Engine v2*: precision, recall, F1 score overview

This decline in accuracy can be attributed to the conservative filtering approach of *Data Quality Engine* v2, which is thoroughly described in subsection 4.6.4. As for the model's performance, it is evident that it continues to be adversely affected, as only a handful of feedback entries are being categorized as faulty and consequently not excluded from consideration.

5.4.7 Model category prediction flow validation: *Data Quality Engine* v3 (*MobileNetV2* model) - model predictions involvement

Within the given validation process, we employ the *Data Quality Engine v3* (subsection 4.6.4) algorithm to effectively exclude feedback that pertains to faulty data. *Data Quality Engine* v3 algorithm utilizes *MobileNetV2* model architecture.

VALIDATION CONFIGURATION:

MODEL TRAINING PHASE:

Model used: MobileNetV2

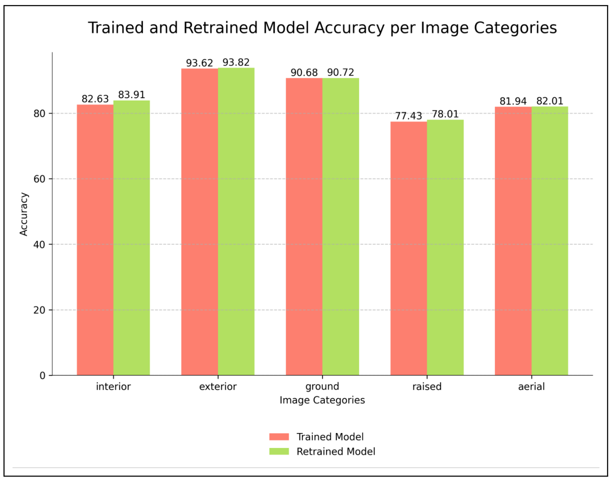
Images initially utilized during the model training phase.

USERS FEEDBACK AND MODEL RETRAINING PHASE:

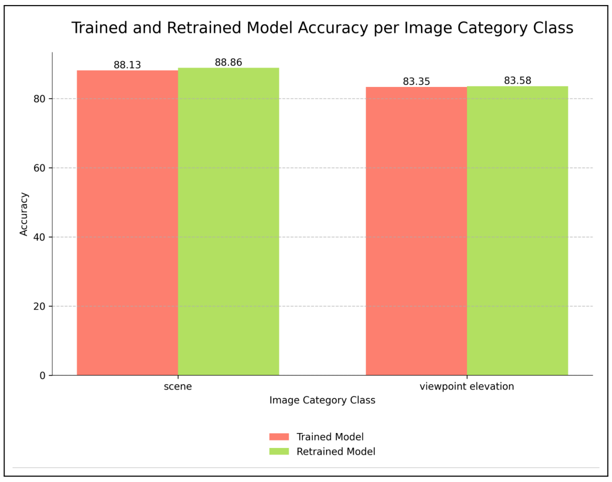
A second batch of images was uploaded, and the model's category predictions were displayed on the *Ajapaik* user interface. Subsequently, we simulated users feedback on image categories, and the model underwent a retraining process.

RETRAINED MODEL PREDICTIONS PHASE:

A fresh set of images was introduced for validation purposes after the model had been retrained based on user feedback. The results of this validation phase are outlined as follows:



(a) Per category



(b) Per category class

Figure 24. Validation results: *Data Quality Engine* v3 (*MobileNetV2* model) - model predictions involvement

Conclusions

Prior to forwarding user feedback for model retraining, the *Data Quality Engine* managed to exclude 670 feedback for *scene* class and 704 for *viewpoint elevation* class (out for 1500 total feedback per each category). The precision, recall and F1 results are outlined in Table 11.

Category class	TP	TN	FP	FN	Precision	Recall	F1 Score
Scene	601	701	69	129	0.9	0.82	0.85
Viewpoint elevation	654	567	50	229	0.93	0.74	0.82

Table 11. *Data Quality Engine v3*: precision, recall, F1 score overview

The results showed a modest improvement, albeit still noticeable. With future iterations of the retraining process, we can anticipate further enhancements in the model's accuracy.

5.4.8 Model category prediction flow validation: ***Data Quality Engine*** v3 (***ResNet50*** model) - model predictions involvement

Within the given validation process, we employ the *Data Quality Engine* v3 (subsection 4.6.4) algorithm to effectively exclude feedback that pertains to faulty data. *Data Quality Engine* v3 algorithm utilizes *ResNet50* model architecture.

VALIDATION CONFIGURATION:

MODEL TRAINING PHASE:

Model used: ResNet50

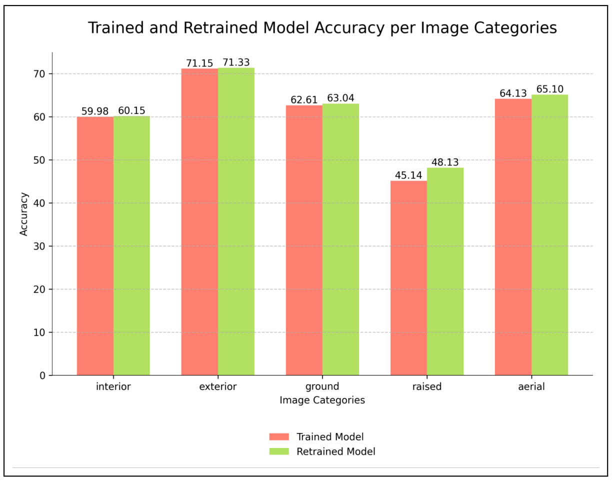
Images initially utilized during the model training phase.

USERS FEEDBACK AND MODEL RETRAINING PHASE:

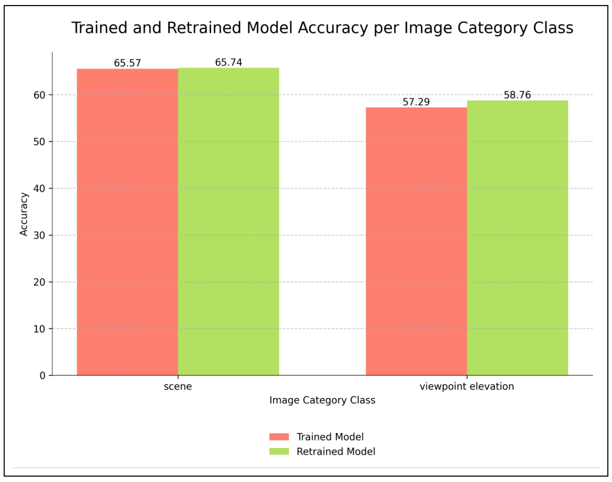
A second batch of images was uploaded, and the model's category predictions were displayed on the *Ajapaik* user interface. Subsequently, we simulated users feedback on image categories, and the model underwent a retraining process.

RETRAINED MODEL PREDICTIONS PHASE:

A fresh set of images was uploaded for validation purposes after the model had been retrained from users feedback. A fresh set of images was introduced for validation purposes after the model had been retrained based on user feedback. The results of this validation phase are outlined as follows:



(a) Per category



(b) Per category class

Figure 25. Validation results: *Data Quality Engine* v3 (*ResNet50* model) - model predictions involvement

Conclusions

Despite its notable reliability and high performance in addressing image categorization tasks, we observed a noticeable decline in accuracy across all categories. Despite the unexpected nature of this outcome, we acknowledge it and proceed to the next phase, where we aim to perform validation of AlexNet model.

5.4.9 Model category prediction flow validation: *Data Quality Engine* v3 (*AlexNet* model) - model predictions involvement

Within the given validation process, we employ the *Data Quality Engine* v3 (subsection 4.6.4) algorithm to effectively exclude feedback that pertains to faulty data. *Data Quality Engine* v3 algorithm utilizes *AlexNet* model architecture.

VALIDATION CONFIGURATION:

MODEL TRAINING PHASE:

Model used: AlexNet

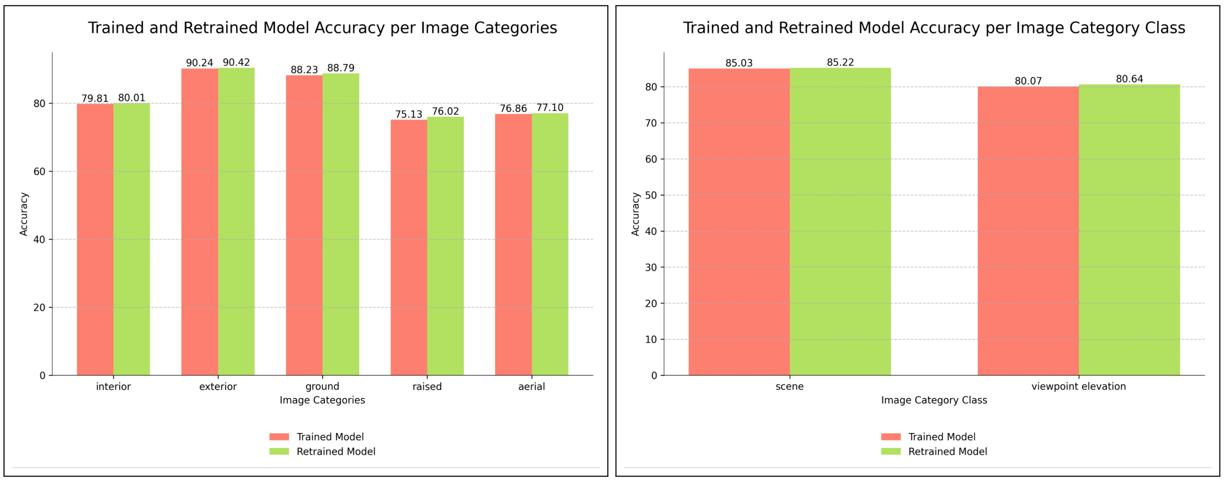
Images initially utilized during the model training phase.

USERS FEEDBACK AND MODEL RETRAINING PHASE:

A second batch of images was uploaded, and the model's category predictions were displayed on the *Ajapaik* user interface. Subsequently, we simulated users feedback on image categories, and the model underwent a retraining process.

RETRAINED MODEL PREDICTIONS PHASE:

A fresh set of images was introduced for validation purposes after the model had been retrained based on user feedback. The results of this validation phase are outlined as follows:



(a) Per category

(b) Per category class

Figure 26. Validation results: *Data Quality Engine* v3 (*AlexNet* model) - model predictions involvement

Conclusions

While AlexNet reached high level of model accuracy, approaching the performance achieved by MobileNetV2, it is crucial to recognize that AlexNet is an older architecture. The observed high accuracy of AlexNet can be attributed to its pioneering role in the field, but over time, more modern architectures, such as MobileNetV2, have emerged, surpassing it in terms of overall performance.

5.4.10 Validation summary

After conducting numerous validation tests with various configurations - including initial training image volumes, model architecture, the presence of the *Data Quality Engine*, and its version — we can confidently conclude that the optimal result was obtained when utilizing *Data Quality Engine* version 3. In this setup, the model architecture was MobileNetV2, and the number of input training images was balanced at 1019 for each category (as detailed in subsection 5.4.7).

This configuration yielded impressive performance, achieving an accuracy of 88.13% for the *scene* category class and 83.35% for *viewpoint elevation.* Following retraining, the model's accuracy experienced a slight increase, reaching 88.86% for *scene* and 83.58% for *viewpoint elevation.*

Throughout the validation process, it became evident that accuracy per category suffered when the image volume was significantly lower than other classes. Thus, maintaining an even number of images during the initial training of the model proved crucial. Although initially exploring additional image generation to balance category counts, we found that downsizing existing images and acquiring supplementary images for categories with lower counts, without resorting to image generation, produced more promising results. Consequently, this approach was preferred and adopted.

A surprising finding emerged regarding the accuracy of the ResNet50 model. Despite its well-established reliability and high performance in image categorization tasks, we observed a noticeable decline in accuracy across all categories when employing this particular model architecture.

Table 12 and Table 13 in Appendix 7 provide a comprehensive overview of the achieved results across various validation configuration setups.

5.5 Alternative Validation

Having conducted software, UI, and model accuracy validations, we recognize additional potential in undertaking validation with actual users once the model is live. The prospect of validating user feedback on image categories in a natural, non-simulated setting holds promise. However, this type of validation would necessitate a more extended timeframe, given the current level of activity among *Ajapaik* users. A more expansive monitoring window may be essential during this validation phase.

6 Conclusions

The first goal of the thesis was to build an automated images categorization engine which benefits user experience in their every-day *Ajapaik* [1] platform usage. The second goal of the thesis was to enhance the model categorization algorithm by building an intermediate categorization validation layer and excluding inaccurate users' categorization records.

For achieving all of the goals mentioned above we have designed, implemented and validated an images categorization engine coupled with a *Data Quality Engine* layer to process user feedback before passing it next to the model retraining cycle. We have validated and developed integration with various set-up configurations and developed algorithms for *Data Quality Engine*. We assessed the precision of the initially trained model by considering the images utilized during training and their respective quantities. We also measured the accuracy of the model after retraining with various implementations of *Data Quality Engine* algorithms.

Following our thesis, we conclude that the best initial accuracy of the model was achieved when an equal number of images (1019 images per category) were provided, without employing additional image generation techniques. The initial accuracy for *scene* category images reached 88.13%, while for *viewpoint elevation* category images, it reached 83.55%. Through the utilization of the *Data Quality Engine*, we determined that the most effective implementation for the algorithm is "model predictions involvement." This approach excels in ensuring that users' faulty feedback is filtered out before entering the retraining cycle. Thanks to this algorithm, the model accuracy after retraining saw a slight enhancement, reaching 88.86% for *scene* category images and 83.58% for *viewpoint elevation*. While this improvement may appear modest, we hold the perspective that any number approaching 100% signifies a noteworthy success.

Though we believe, during the thesis we have achieved some good accuracy numbers for initially trained model and for the model after retraining cycle, it is still not enough to treat this result as final. We see potential in improving *Data Quality Engine* v3 (model predictions involvement) by increasing involvement aka weight in users feedback and hence not solely rely on model predictions only.

7 Future work

Despite the impressive results achieved by the *Data Quality Engine v3*, which is based on model predictions, and its contribution to model retraining, leading to enhanced accuracy, there remains a need to further enhance the data quality engine.

Even though the initial model achieved an accuracy of nearly 90%, it is essential to acknowledge that this still means that approximately 10 out of 100 images might be predicted incorrectly. The current implementation would discard user feedback if it does not align with the model's prediction, even when users are providing accurate feedback by trying to improve the model.

In additional, we see potential in applying and validating multimodal principles that combine language models and computer vision, as discussed in "A multimodal turn in Digital Humanities" paper [56].

Our developed implementation represents the first step toward a more advanced solution, one that takes into account both the model's predictions and user input. This approach ensures that exclusions are not solely based on the model's verdict.

Bibliography

[1] *Ajapaik*. URL: `https://ajapaik.ee/?page=1`. (accessed: 29.11.2023).

[2] Yiping Gao, Liang Gao, and Xinyu Li. “A Generative Adversarial Network Based Deep Learning Method for Low-Quality Defect Image Reconstruction and Recognition”. In: *IEEE Transactions on Industrial Informatics* 17.5 (2021), pp. 3231–3240. DOI: `10.1109/TII.2020.3008703`.

[3] Stefan Pletschacher, Jianying Hu, and Apostolos Antonacopoulos. “A New Framework for Recognition of Heavily Degraded Characters in Historical Typewritten Documents Based on Semi-Supervised Clustering”. In: *2009 10th International Conference on Document Analysis and Recognition*. 2009, pp. 506–510. DOI: `10.1109/ICDAR.2009.267`.

[4] N. Neelima, A. SriKrishna, and K Gangadhara Rao. “Prototype analysis of different object recognition techniques in image processing”. In: *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*. 2017, pp. 2882–2892. DOI: `10.1109/ICECDS.2017.8389982`.

[5] Natália C. Batista, Ana Paula B. Lopes, and Arnaldo de A. Araújo. “Detecting Buildings in Historical Photographs Using Bag-of-Keypoints”. In: *2009 XXII Brazilian Symposium on Computer Graphics and Image Processing*. 2009, pp. 276–283. DOI: `10.1109/SIBGRAPI.2009.31`.

[6] A. Barla, F. Odone, and A. Verri. “Old fashioned state-of-the-art image classification”. In: *12th International Conference on Image Analysis and Processing, 2003.Proceedings.* 2003, pp. 566–571. DOI: `10.1109/ICIAP.2003.1234110`.

[7] Neetu Mittal, Arjun Sehgal, and Sunil Kumar Khatri. “Enhancement of historical documents by image processing techniques”. In: *2017 6th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*. 2017, pp. 630–635. DOI: `10.1109/ICRITO.2017.8342504`.

[8] A. Vailaya, A. Jain, and Hong Jiang Zhang. “On image classification: city vs. landscape”. In: *Proceedings. IEEE Workshop on Content-Based Access of Image and Video Libraries (Cat. No.98EX173)*. 1998, pp. 3–8. DOI: `10.1109/IVL.1998.694464`.

[9] Yanyun Qu et al. “Image labeling via incremental model learning”. In: Oct. 2010, pp. 1573–1576. DOI: `10.1109/ICIP.2010.5653562`.

[10] Mingfang wu et al. “Automated metadata annotation: What is and is not possible with machine learning”. In: *Data Intelligence* 5 (Sept. 2022), pp. 1–17. DOI: 10.1162/dint_a_00162.

[11] Thomas Smits and Melvin Wevers. “A Multimodal Turn in Digital Humanities: Using Contrastive Machine Learning Models to Explore, Enrich, and Analyze Digital Visual Historical Collections”. In: *Digital Scholarship in the Humanities* 38 (3 Sept. 2023), pp. 1267–1280. DOI: 10.1093/llc/fqad008. URL: https://doi.org/10.1093/llc/fqad008.

[12] David Savage et al. “Detection of opinion spam based on anomalous rating deviation”. In: *Expert Systems with Applications* 42 (July 2015). DOI: 10.1016/j.eswa.2015.07.019.

[13] Nicolas Muller and Karla Markert. “Identifying Mislabeled Instances in Classification Datasets”. In: July 2019, pp. 1–8. DOI: 10.1109/IJCNN.2019.8851920.

[14] Geoff Pleiss et al. *Identifying Mislabeled Data using the Area Under the Margin Ranking.* Jan. 2020.

[15] A. Hosna, E. Merry, J. Gyalmo, et al. “Transfer learning: a friendly introduction”. In: *Journal of Big Data* 9 (2022), p. 102. DOI: 10.1186/s40537-022-00652-w.

[16] François Chollet. *Deep learning with Python.* Manning Publications, 2017.

[17] Mark Sandler et al. “MobileNetV2: Inverted Residuals and Linear Bottlenecks”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), pp. 4510–4520. URL: https://api.semanticscholar.org/CorpusID:4555207.

[18] PyTorch Contributors. *PyTorch SSDLite Model Documentation.* (accessed: 21.12.2023). 2023. URL: https://pytorch.org/vision/main/models/ssdlite.html.

[19] *Library of Congress Pictures Collection.* Library of Congress. URL: https://www.loc.gov/pictures/. (accessed: 17.12.2023).

[20] *New York Public Library.* New York Public Library. URL: https://www.nypl.org/. (accessed: 17.12.2023).

[21] *University of Michigan Image Collections.* University of Michigan Library. URL: https://quod.lib.umich.edu/cgi/i/image/image-idx?c=aict. (accessed: 18.12.2023).

[22] *British Library Flickr Photostream.* British Library. URL: https://www.flickr.com/photos/britishlibrary/. (accessed: 18.12.2023).

[23] *Wellcome Collection Search Works.* Wellcome Collection. URL: https://wellcomecollection.org/search/works. (accessed: 18.12.2023).

[24] *Getty Images - Media and Sports Collections.* (accessed: 18.12.2023). Getty Images. URL: `https://www.gettyimages.com/collections/media-and-sports`.

[25] *Google Images - Life Magazine Hosted.* (accessed: 18.12.2023). Google Images. URL: `https://images.google.com/hosted/life`.

[26] *Mary Evans Picture Library.* (accessed: 18.12.2023). Mary Evans Picture Library. URL: `https://www.maryevans.com/`.

[27] *Estonian State Archives - Fotis.* (accessed: 18.12.2023). Estonian State Archives. URL: `https://www.ra.ee/fotis/`.

[28] *University of Tartu DSpace Collection.* (accessed: 18.12.2023). University of Tartu. URL: `https://dspace.ut.ee/collections/c3bbb89b-6b56-4a70-861e-7b71ef324f82`.

[29] Ajapaik Development Team. *Ajapaik Open Data.* URL: `https://opendata.ajapaik.ee/photos/`. (accessed: 30.11.2023).

[30] Guido van Rossum. *Python.* URL: `https://www.python.org`. (accessed: 29.11.2023).

[31] Anna Grund. *ajapaik-model-training.* URL: `https://github.com/angrun/ajapaik-model-training`. (accessed: 29.11.2023).

[32] Django Software Foundation. *Django, Python-based web framework.* URL: `https://www.djangoproject.com/`. (accessed: 29.11.2023).

[33] Google Brain Google. *TensorFlow machine learning platform.* URL: `https://www.tensorflow.org/`. (accessed: 29.11.2023).

[34] François Chollet. *Keras artificial neural network library.* URL: `https://keras.io/`. (accessed: 29.11.2023).

[35] Travis Oliphant. *Numpy library for large, multi-dimensional arrays and matrices.* URL: `https://numpy.org/`. (accessed: 29.11.2023).

[36] Alex Grönholm. *APScheduler library for scheduling code executions.* URL: `https://pypi.org/project/APScheduler/`. (accessed: 29.11.2023).

[37] David Cournapeau. *Scikit-learn machine learning library.* URL: `https://scikit-learn.org/stable/`. (accessed: 29.11.2023).

[38] Guido van Rossum, Jukka Lehtosalo, Łukasz Langa and Michael Lee. *typing-extensions library for adding advanced type hinting capabilities for the codebase.* URL: `https://pypi.org/project/typing-extensions/`. (accessed: 29.11.2023).

[39] Jeffrey A. Clark (Alex). *Pillow for opening, manipulating, and saving various image file formats.* URL: `https://pypi.org/project/Pillow/`. (accessed: 30.11.2023).

[40] Alex Johnson, Jack Parmer, Chris Parmer, and Matthew Sundquist. *Plotly for creating interactive and visually appealing data visualizations and charts.* URL: https://plotly.com/dash/. (accessed: 30.11.2023).

[41] The Pandas Development Team. *Pandas for data manipulation and analysis.* URL: https://pypi.org/project/pandas/. (accessed: 30.11.2023).

[42] Kenneth Reitz. *Requests for managing HTTP requests.* URL: https://pypi.org/project/requests/. (accessed: 30.11.2023).

[43] Ajapaik Development Team. *Ajapaik web project.* URL: https://github.com/Ajapaik/ajapaik-web. (accessed: 30.11.2023).

[44] The jQuery Team. *JavaScript library.* URL: https://jquery.com/. (accessed: 30.11.2023).

[45] Robert C. Martin. *Design Principles and Design Patterns.* Object Mentor, Inc, 2000.

[46] Wang Haoyu and Zhou Haili. “Basic Design Principles in Software Engineering”. In: *2012 Fourth International Conference on Computational and Information Sciences.* 2012, pp. 1251–1254. DOI: 10.1109/ICCIS.2012.91.

[47] Chu He et al. “Lifting Scheme-Based Deep Neural Network for Remote Sensing Scene Classification”. In: *Remote Sensing* 11 (Nov. 2019), p. 2648. DOI: 10.3390/rs11222648.

[48] François Chollet. *Adam optimiser.* URL: https://keras.io/api/optimizers/adam/. (accessed: 30.11.2023).

[49] François Chollet. *ResNet model architecture.* URL: https://keras.io/api/applications/resnet/. (accessed: 30.11.2023).

[50] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems.* Curran Associates, Inc., 2012, pp. 1097–1105.

[51] Fei Tony Liu, Kai Ting, and Zhi-Hua Zhou. “Isolation Forest”. In: Jan. 2009, pp. 413–422. DOI: 10.1109/ICDM.2008.17.

[52] Google Cloud. *Google Cloud Vision API.* (accessed: 21.12.2023). URL: https://cloud.google.com/vision.

[53] Microsoft Azure. *Microsoft Azure Computer Vision API.* (accessed: 21.12.2023). URL: https://azure.microsoft.com/en-us/products/ai-services/ai-vision.

[54] Amazon Web Services. *Amazon Rekognition.* (accessed: 21.12.2023). URL: https://aws.amazon.com/ru/rekognition/.

[55] IBM. *IBM Watson Visual Recognition Documentation.* (accessed: 21.12.2023). URL: https://developer.ibm.com/components/watson-visual-recognition/.

[56] Thomas Smits and Melvin Wevers. “A multimodal turn in Digital Humanities: Using contrastive machine learning models to explore, enrich, and analyze digital visual historical collections”. In: *Digital Scholarship in the Humanities* 38.3 (2023), pp. 1267–1280. DOI: `10.1093/llc/fqad008`.

[57] OpenAI. *ChatGPT: OpenAI’s Conversational Language Model.* (accessed: 21.12.2023). 2022. URL: `https://openai.com/blog/chatgpt`.

[58] OpenAI. *DALL-E 2: A Powerful Custom Image Generation Model.* (accessed: 21.12.2023). 2022. URL: `https://openai.com/dall-e-2`.

Appendices

Appendix 1 - Non-exclusive licence for reproduction and publication of graduation thesis [1]

[1]The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.a and 1.b of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

Appendix 2 - Validation results overview

Implementation	**Accuracy %**						
Image count during validation	INTERIOR	EXTERIOR	GROUND	RAISED	AERIAL	SCENE TOTAL	VIEW TOTAL
No Data Quality Engine - Interior: 3056 - Exterior: 8614 - Ground: 10 670 - Raised: 901 - Aerial: 99	**Train:** 30.42% **Retrain:** 19.63%	**Train:** 97.14% **Retrain:** 34.84%	**Train:** 98.79% **Retrain:** 97.05%	**Train:** 7.31% **Retrain:** 3.34%	**Train:** 15.15% **Retrain:** 9.09%	**Train:** 79.66% **Retrain:** 30.86%	**Train:** 91.00% **Retrain:** 89.07%
No Data Quality Engine (additional images generation) - Interior: 8614 - Exterior: 8614 - Ground: 10 670 - Raised: 10 670 - Aerial: 10 670	**Train:** 71.15% **Retrain:** 21.59%	**Train:** 92.79% **Retrain:** 35.16%	**Train:** 79.71% **Retrain:** 42.19%	**Train:** 64.28% **Retrain:** 26.86%	**Train:** 54.00% **Retrain:** 33.76%	**Train:** 81.97% **Retrain:** 28.38%	**Train:** 66.00% **Retrain:** 34.27%
No Data Quality Engine (downsampling + outsourcing for viewpoint elevation - Interior: 3056 - Exterior: 3056 - Ground: 3056 - Raised: 3056 - Aerial: 3056	**Train:** 82.63% **Retrain:** 37.29%	**Train:** 93.62% **Retrain:** 48.77%	**Train:** 90.68% **Retrain:** 49.44%	**Train:** 77.43% **Retrain:** 31.25%	**Train:** 81.94% **Retrain:** 41.15%	**Train:** 88.13% **Retrain:** 43.03%	**Train:** 83.35% **Retrain:** 40.61%
Data Quality Engine v1 (most common verdict) - Interior: 3056 - Exterior: 3056 - Ground: 3056 - Raised: 3056 - Aerial: 3056	**Train:** 82.63% **Retrain:** 40.53%	**Train:** 93.62% **Retrain:** 49.85%	**Train:** 90.68% **Retrain:** 57.61%	**Train:** 77.43% **Retrain:** 69.48%	**Train:** 81.94% **Retrain:** 58.00%	**Train:** 88.13% **Retrain:** 45.19%	**Train:** 83.35% **Retrain:** 61.70%
Data Quality Engine v2 (anomaly detection) - Interior: 3056 - Exterior: 3056 - Ground: 3056 - Raised: 3056 - Aerial: 3056	**Train:** 82.63% **Retrain:** 44.55%	**Train:** 93.62% **Retrain:** 37.88%	**Train:** 90.68% **Retrain:** 50.34%	**Train:** 77.43% **Retrain:** 42.22%	**Train:** 81.94% **Retrain:** 24.34%	**Train:** 88.13% **Retrain:** 41.26%	**Train:** 83.35% **Retrain:** 38.97%
Data Quality Engine v3 (model prediction involvement) MobileNetV2 model - Interior: 3056 - Exterior: 3056 - Ground: 3056 - Raised: 3056 - Aerial: 3056	**Train:** 82.63% **Retrain:** 83.91%	**Train:** 93.62% **Retrain:** 93.82%	**Train:** 90.68% **Retrain:** 90.72%	**Train:** 77.43% **Retrain:** 78.01%	**Train:** 81.94% **Retrain:** 82.01%	**Train:** 88.13% **Retrain:** 88.86%	**Train:** 83.35% **Retrain:** 83.58%
Data Quality Engine v3 (model prediction involvement) ResNet50 model - Interior: 3056 - Exterior: 3056 - Ground: 3056 - Raised: 3056 - Aerial: 3056	**Train:** 82.63% **Retrain:** 60.15%	**Train:** 93.62% **Retrain:** 71.33%	**Train:** 90.68% **Retrain:** 63.04%	**Train:** 77.43% **Retrain:** 48.13%	**Train:** 81.94% **Retrain:** 65.10%	**Train:** 88.13% **Retrain:** 65.74%	**Train:** 83.35% **Retrain:** 58.76%
Data Quality Engine v3 (model prediction involvement) AlexNet model - Interior: 3056 - Exterior: 3056 - Ground: 3056 - Raised: 3056 - Aerial: 3056	**Train:** 82.63% **Retrain:** 80.01%	**Train:** 93.62% **Retrain:** 90.42%	**Train:** 90.68% **Retrain:** 88.79%	**Train:** 77.43% **Retrain:** 76.02%	**Train:** 81.94% **Retrain:** 77.10%	**Train:** 88.13% **Retrain:** 85.22%	**Train:** 83.35% **Retrain:** 80.64%

Table 12. Validation results overview: accuracy

Implementation	**F1 score**						
Image count during validation	INTERIOR	EXTERIOR	GROUND	RAISED	AERIAL	SCENE TOTAL	VIEW TOTAL
No Data Quality Engine - Interior: 3056 - Exterior: 8614 - Ground: 10 670 - Raised: 901 - Aerial: 99	**Train:** 0.44 **Retrain:** 0.13	**Train:** 0.88 **Retrain:** 0.43	**Train:** 0.95 **Retrain:** 0.94	**Train:** 0.01 **Retrain:** 0.01	**Train:** 0.002 **Retrain:** 0.002	**Train:** 0.76 **Retrain:** 0.35	**Train:** 0.87 **Retrain:** 0.86
No Data Quality Engine (additional images generation) - Interior: 8614 - Exterior: 8614 - Ground: 10 670 - Raised: 10 670 - Aerial: 10 670	**Train:** 0.8 **Retrain:** 0.35	**Train:** 0.84 **Retrain:** 0.47	**Train:** 0.69 **Retrain:** 0.28	**Train:** 0.59 **Retrain:** 0.23	**Train:** 0.56 **Retrain:** 0.28	**Train:** 0.82 **Retrain:** 0.41	**Train:** 0.61 **Retrain:** 0.26
No Data Quality Engine (downsampling + outsourcing for viewpoint elevation - Interior: 3056 - Exterior: 3056 - Ground: 3056 - Raised: 3056 - Aerial: 3056	**Train:** 0.87 **Retrain:** 0.4	**Train:** 0.89 **Retrain:** 0.46	**Train:** 0.82 **Retrain:** 0.36	**Train:** 0.79 **Retrain:** 0.26	**Train:** 0.78 **Retrain:** 0.33	**Train:** 0.88 **Retrain:** 0.43	**Train:** 0.8 **Retrain:** 0.32
Data Quality Engine v1 (most common verdict) - Interior: 3056 - Exterior: 3056 - Ground: 3056 - Raised: 3056 - Aerial: 3056	**Train:** 0.87 **Retrain:** 0.43	**Train:** 0.89 **Retrain:** 0.48	**Train:** 0.82 **Retrain:** 0.52	**Train:** 0.79 **Retrain:** 0.63	**Train:** 0.78 **Retrain:** 0.51	**Train:** 0.88 **Retrain:** 0.46	**Train:** 0.8 **Retrain:** 0.55
Data Quality Engine v2 (anomaly detection) - Interior: 3056 - Exterior: 3056 - Ground: 3056 - Raised: 3056 - Aerial: 3056	**Train:** 0.87 **Retrain:** 0.43	**Train:** 0.89 **Retrain:** 0.39	**Train:** 0.82 **Retrain:** 0.35	**Train:** 0.79 **Retrain:** 0.36	**Train:** 0.78 **Retrain:** 0.2	**Train:** 0.88 **Retrain:** 0.41	**Train:** 0.8 **Retrain:** 0.3
Data Quality Engine v3 (model prediction involvement) MobileNetV2 model - Interior: 3056 - Exterior: 3056 - Ground: 3056 - Raised: 3056 - Aerial: 3056	**Train:** 0.87 **Retrain:** 0.88	**Train:** 0.89 **Retrain:** 0.89	**Train:** 0.82 **Retrain:** 0.82	**Train:** 0.79 **Retrain:** 0.79	**Train:** 0.78 **Retrain:** 0.78	**Train:** 0.88 **Retrain:** 0.89	**Train:** 0.8 **Retrain:** 0.82
Data Quality Engine v3 (model prediction involvement) ResNet50 model - Interior: 3056 - Exterior: 3056 - Ground: 3056 - Raised: 3056 - Aerial: 3056	**Train:** 0.87 **Retrain:** 0.64	**Train:** 0.89 **Retrain:** 0.68	**Train:** 0.82 **Retrain:** 0.54	**Train:** 0.79 **Retrain:** 0.47	**Train:** 0.78 **Retrain:** 0.58	**Train:** 0.88 **Retrain:** 0.66	**Train:** 0.8 **Retrain:** 0.53
Data Quality Engine v3 (model prediction involvement) AlexNet model - Interior: 3056 - Exterior: 3056 - Ground: 3056 - Raised: 3056 - Aerial: 3056	**Train:** 0.87 **Retrain:** 0.84	**Train:** 0.89 **Retrain:** 0.86	**Train:** 0.82 **Retrain:** 0.77	**Train:** 0.79 **Retrain:** 0.73	**Train:** 0.78 **Retrain:** 0.75	**Train:** 0.88 **Retrain:** 0.85	**Train:** 0.8 **Retrain:** 0.75

Table 13. Validation results overview: F1 score

Appendix 3 - Repository link

ajapaik-model-training repository: https://github.com/angrun/ajapaik-model-training