Executive Summary of the Thesis

ODIN Web: A web-based tool for image annotation, inference, and model evaluation

Laurea Magistrale in Computer Science And Engineering - Ingegneria Informatica

Author: Simona Malegori

Advisor: Prof. Piero Fraternali

Academic year: 2022-2023

1. Introduction

Computer Vision (CV) is a field of study, whose purpose is to replicate, by means of Machine Learning (ML) and Deep Learning (DL) methods, the ability of humans to extract information and interpret the visual world around them. Computer Vision exploits the vast amount of imagery data available, which has to be annotated with the "truth" the model should learn from. Being image annotation a laborious task, typically manual, several tools, that provide functionalities to annotate images and create high-quality datasets, have been implemented. Thanks to the progress made in this field, complex models based on Deep Neural Networks (DNNs) have now achieved exceptional results. This, however, added complexity to the model performance evaluation, which is necessary for their understanding and improvement. Studies on model performance analysis yielded the development of several tools that exploit "black-box" analysis techniques to achieve in-depth diagnosis of complex models. These advancements led Computer Vision to acquire an increasing impact in real-world scenarios applications. This combination of Computer Vision and practical applications has revealed the need for tools to allow individuals with no technical knowledge to approach Computer Vision models. In this context, the objective is now the implementation of intuitive tools that cover the entire pipeline of model development, from image annotation and model analysis to the practical application of the Computer Vision model. ODIN Web, is a user-friendly web-based tool for dataset management, image annotation related to image classification, object detection, and instance segmentation, and model performance investigation. For this evaluation task, it leverages ODIN, a framework for extensive "black-box" analysis. Starting from that, this work expands ODIN Web by implementing a user system, for collaboration and role-based access control, integrating model inference, and providing geo-visualization functionalities for datasets of geolocalized satellite images. The relevance of the implemented tool was demonstrated by illustrating its usage in some real-application scenarios.

2. Related Work

Computer Vision (CV) is a subfield of Artificial Intelligence (AI), whose purpose is to replicate the ability of humans to extract information and interpret the visual world around them. Computer Vision is based on a comprehensive set of different tasks, which the models are trained to solve. The fundamental ones are the following: *Image Classification (IC)*, which is the task of determining what object classes are present in an image, *Object Detection (OD)*, which is the task of identifying which objects are present and localizing them on the image with a bounding box, and *Image Segmentation*, which can be declined into *Semantic Segmentation (SS)*, the task of assigning a semantic label to each pixel in an image and *Instance Segmentation (IS)* the problem of detecting and precisely outlining each object of interest.

Computer Vision counts a variety of real-world application domains, such as the environmental monitoring field. Many studies in this area have focused on exploiting Computer Vision for environmental protection through the detection of waste and illegal landfills, by leveraging the large amount of data acquired through Remote Sensing (RS) performed by Earth Observation (EO) satellites and unmanned aerial vehicles (UAVs) [2]. Several RS problems have been the subject of research, such as landfill or sparse waste detection and monitoring, illegal dumping distribution characterization, and mapping of areas with a high risk of illegal waste dumping. Important results have been achieved through traditional CV architectures applied to multispectral images for object-based and pixel-based classification. Then, the employment of Deep Learning CV models yielded additional results in the environmental monitoring field, by exploiting optical and multispectral images.

Fundamental to the understanding of model behaviors and the improvement of the model itself, is the model performance evaluation. With the increase in the elaborateness of the architectures, "black-box" analysis has become preferable to "open-box" analysis, thanks to its reduced complexity. This analysis is based on the computation of performance metrics, such as accuracy, precision, recall, and F1 score. It includes evaluation through different types of investigations such as curve analysis, false positives analysis, and reliability analysis. Moreover, by considering extra properties of the input data (meta-annotations), the analysis can be enriched by computing their impact on the model performance, which provides insights into the model behavior. Over the years several published tools for DNNs evaluation implemented "black-box" analysis techniques. Pioneer in the analysis of the impact of meta-annotations was the work of *Hoiem et al.* [3], which demonstrated how breaking down standard metrics into sub-metrics linked to a metadata value helps in understanding the model behavior. The *COCO API* framework addressed error diagnosis for both OD and IS tasks, introducing the use of custom input properties to differentiate the computation of the mean average precision. The *What-If-Tool* allows to analyze the performance of ML systems in hypothetical situations by visualizing the effect of several features across multiple models and subsets of the input data. Finally, *OpenVINO* [1] is an environment for analyzing, optimizing, evaluating, and deploying DL models. It includes "black-box" analysis through metrics for IC, OD, IS, and SS tasks, and supports calibration techniques.

The training of ML and DL models requires large amounts of annotated data from which models extract patterns and learn how to solve certain tasks. The quality of the input data greatly impacts the model performance, thus data annotation assumes a substantial role in the preparation of training and testing datasets. The image annotation process is typically manual which causes the construction of a high-quality dataset to be a labor-intensive and time-consuming task. Over the years, different tools implemented diverse functionalities to assist the annotators. One of the first tools for image annotation was *PhotoStuff* a platform that addresses annotation integrating ontologies in the process of assigning concepts to regions in an image. Subsequently, were published several tools providing user-friendly graphical interfaces for the annotation of data sets, supporting different CV tasks with instruments such as bounding boxes, polygons lines, points, and others. Examples of those tools are *LabelMe*, *VoTT*, *ImageTagger*, and *VIA*. These tools also provide import/export of annotations in different formats. Finally, some more complex tools, such as *CVAT*, *V7*, *Make Sense* and *Labelbox*, implement additional aids, like automatic detection of shapes during annotation, and even integrate semi-automatic annotation by leveraging some CV models.

3. Proposed Solution

The ODIN framework [5–7] is a Python "black-box" diagnosis tool that allows the investigation of errors and model performance through a wide range of metrics and diagnostic reports, including the analysis of subsets of the input by exploiting meta-annotations.
ODIN Web is a web-based application [4], whose aim is to provide a web interface, accessible to non-technical users, to exploit the ODIN framework. The application allows to create and manage datasets, to annotate images according to different Computer Vision tasks, and to investigate model performance. The tasks supported by ODIN Web are *binary classification*, *multi-class multi-label* and *multi-class single-label classification*, *object detection*, and *instance segmentation*. The purpose of this work is to extend ODIN Web with further functionalities to meet the following requirements. ODIN Web must provide a collaborative user system, with user management and role-based access control. In particular, the users should be of two types, i.e., admins and simple users. While admins should have full rights to create/delete datasets, edit dataset settings, register users, and manage their access to datasets, simple users should have limited rights and should be able to access only the datasets they have been authorized to. The tool must implement a model inference system with non-blocking execution. In particular, it should allow the user to run one or more of the available models on the images of a target dataset. The computed predictions can be then used in the analyzer to evaluate the model and in the annotator to visualize the predictions corresponding to the images. The tool should allow the uploading of annotations and predictions produced outside of the tool. Finally, ODIN Web should implement a geo-visualization system for datasets of geolocalized satellite images. In particular, inside these datasets, upon uploading predictions in the required GeoJSON format, a map visualization section should be enabled. The map should display the perimeter of the area of interest, the predictions in the form of color-coded bounding boxes, and the Class Activation Maps (CAMs), i.e., the contours of the areas the model focused on. From the map, it should be possible to open in the annotator the image corresponding to a selected box.

4. Implementation

ODIN Web was implemented with a three-tier client-server architecture (Fig.1). It is accessible for the users on the client side through a Web Browser of choice. The *ODIN Web Client* communicates, through the HTTP protocol, with the *ODIN Server* where three dockers are deployed containing respectively: the *Application Server*, with the application logic, the *Database server*, on which resides the **MongoDB** database instance, and the *Ortophoto Server*, that runs the *Ortophoto Provider*. A configuration in multiple dockers was chosen, since **Docker** allows for code isolation, portability, and fast and consistent delivery.

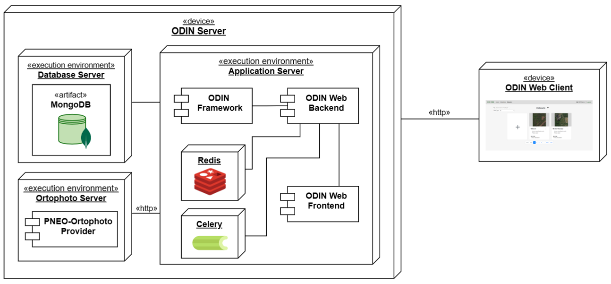

Figure 1: ODIN Web Deployment Architecture

The ODIN Web backend was implemented in Python, with the web framework **Flask**, a lightweight and flexible micro web framework that handles the mapping of URLs to application functions and HTTP communication. The logic was divided into multiple modules that allow the separation of the various functionalities and also reflect the concepts involved in the ODIN Web domain model. In particular, the following modules were implemented.
The *User* module handles admins, users, and user management features. Users are stored on the **MongoDB** database in the dedicated *users* collection. MongoDB is a NoSQL document-based database that was chosen for ODIN Web because of its schemaless approach and the JSON-like storing format.
The *Dataset* component implements the datasets management functionalities. A dataset is a composite object that contains multiple pieces of information. Each dataset of each admin has a dedicated directory that stores the configuration files, the ground truth file, and other directories to store predictions and geolocalized predictions.

The *Annotator* module is the one used to manage everything related to the annotation process. The module includes other two specific components that handle annotations respectively for classification and localization tasks. Annotations are stored in the MongoDB database inside the *observations* and *annotations* collections.
The *Analyzer* module handles the analysis process and is divided into four components: the *Classification Analyzer* handles analysis requests related to classification tasks, the *Localization Analyzer* handles analysis requests related to localization tasks, the *Comparator* handles the comparison of models, finally the *Analyzer* handles the analysis more in general. All these components leverage the ODIN Framework which is installed on the server.
The *Model Executor* module is the one used to manage model inference. It runs the execution in the background on a celery worker, allowing the rest of the application to remain available. **Celery** is a task queue system, integrable with Flask, for real-time processing and task scheduling, that allows to handle background tasks execution.
The *Geovisualizer* module handles the requests and the data regarding the geo-visualization functionalities.
Finally, the *Redis* component manages the data, relative to Datasets and Analyzer instances, stored for caching purposes on the **Redis** server. Redis is an open-source, in-memory data structure store that can be used, among other things, as a fast caching memory.

The ODIN Web frontend was implemented with the **Vue.js** Javascript framework, which provides a component-based programming model and also extends the standard HTML with a declarative rendering syntax to reduce DOM manual manipulation. Moreover, Vue.js is reactive and it efficiently updates the interface, by re-rendering only the necessary parts. The HTTP communication is handled through the promise-based HTTP Client, **Axios**. Axios provides asynchronous call handling and body serialization and deserialization to/from JSON. The user application interface of ODIN Web comprises different pages, that are built following a component-based approach. Indeed, components allow to split of the user interface into independent and reusable pieces, which can then be assembled and nested to realize more complex components. At the root of the ODIN Web application, there is the *App* component, which includes the ones implementing the different views, such as *Login*, *Datasets*, *Dataset Page*, *Users Management*, *Dataset Creation*, *Annotator*, *Predictor* and *Map Visualizer*.

5. Results and Validation

Following the requirements ODIN Web was implemented resulting in a web application characterized by the following main sections: *User Management*, *Dataset Management*, *Annotator*, *Analyzer*, *Map Visualizer*, and *Predictor*.
After the user logs in, ODIN Web displays the homepage, which corresponds to the *Datasets* page (Fig.2). A navigation bar and a breadcrumb trail can be used to quickly navigate the application sections. In the *Datasets* page is displayed a horizontal paginated list of datasets cards. An admin has access to all the datasets he created and is allowed to create more by providing a name and a description, selecting the target task, defining extra properties, and associating a collection of images. A simple user, instead, can access only the datasets he was authorized to by the admin. By selecting a card the page of the relative dataset is opened displaying its information, a button to reach the *Dataset Settings* and four other buttons to access the *Annotator*, the *Analyzer*, the *Map Visualizer*, and the *Predictor*.

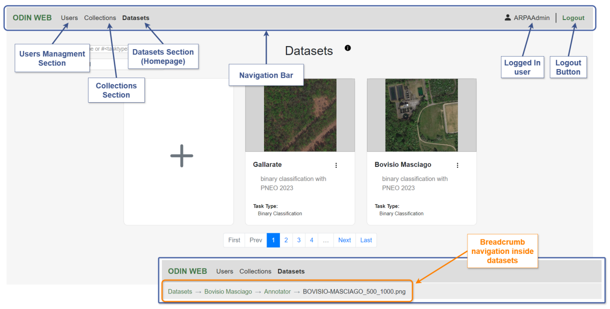

Figure 2: Datasets page

The *Users* section (Fig.3), available to admins, is dedicated to the user management. An admin can use a dedicated form to register a new user. In the top-right area of the page, the list of supervised users is shown. In the lower half of the page, the admin can select a target dataset, and the corresponding users, authorized and not authorized to it, appear in two lists. The admin can authorize some users by selecting them and then clicking on the button to move them to the authorized. Similarly, the admin can revoke the authorization from some users.

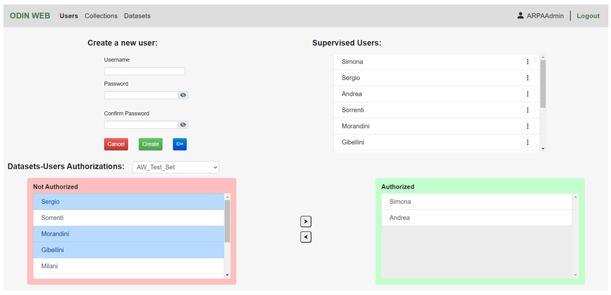

Figure 3: User Management

The *Annotator* displays a filterable paginated grid of the images contained in the dataset which the user can navigate (Fig.4). The user can access the *Annotation Editor* by clicking on a desired image.

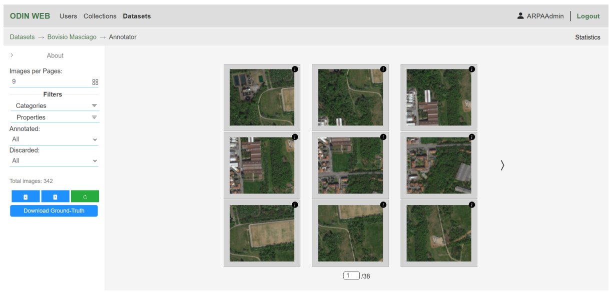

Figure 4: Annotator

The *Annotation Editor* (Fig.5) displays the image in the center. On the right the section for annotating categories and properties. The annotation interface and tools change based on the target task of the dataset. If the predictions and/or the CAMs are available they can be respectively visualized through the relative toggle switch and the layer controller.

The *Analyzer* (Fig.6) comprises two macro sections, the *Dataset Analysis*, analysis of categories and properties distribution within the dataset, and the *Predictions Analysis*, enabled if the predictions are available, which gives access to multiple metrics and graphical analyses, and to model comparison.

The *Map Visualizer* (Fig.7) is accessible if the dataset is geolocalized and predictions in GeoJ-

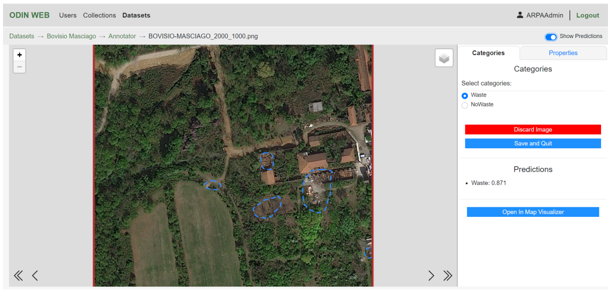

Figure 5: Annotation Editor with predictions and CAMs (binary classification)

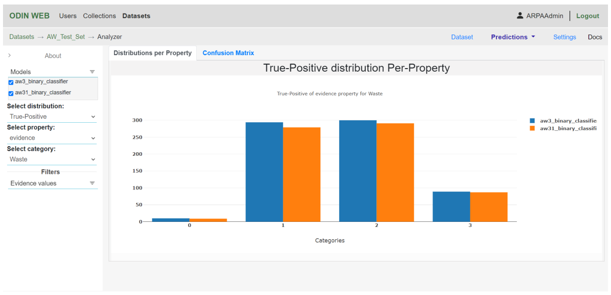

Figure 6: Analyzer: comparison of two models on TP Distribution

SON format have been uploaded. Once entered, the map is displayed, by default the Google Maps one. Over the map are rendered the perimeter of the area of interest, the predictions in the form of color-coded bounding boxes, and the activation maps (CAMs), i.e., the contours of the areas the model focused on. The user can use the layers controller, and the interactive legend to customize the visualization. By hovering each bounding box, a tooltip is opened with coordinates and annotation information. By right-clicking on a box the user can open the image in the *Annotator*.

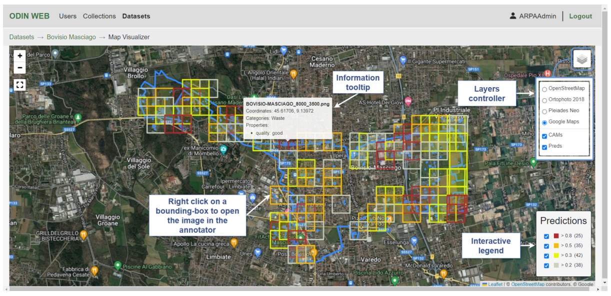

Figure 7: Map Visualizer

The *Predictor* (Fig.8) displays a dropdown menu from which the user can select the desired model and run the inference on the dataset. The models the user can choose from, are those previously selected in the *Dataset Settings*. While the model is running a progress bar is displayed. Under it are reported the name of the image that is currently being processed and the estimated remaining time to inference completion. While the model is running the user can freely navigate throughout the rest of the website.

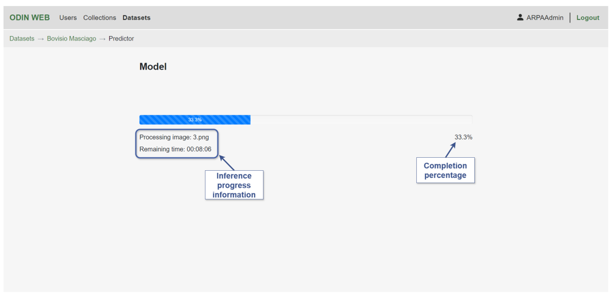

Figure 8: Predictor

ODIN Web has been validated, to verify its effectiveness and compliance with the requirements, internally through a set of different application use cases, and externally by a team of photo interpreters from ARPA Lombardia (Environmental Protection Agency of the Region of Lombardy). Some of the use cases in which it was tested are verification of the predictions for datasets corresponding to 20 Lombardy's municipalities with PNEO satellite images, model performance analysis of two waste detection binary classifiers on an AerialWaste 2.0 test set, and model performance analysis of three target detection binary classifiers on an AREU dataset of aerial images from UAVs. ODIN Web was validated by the ARPA team of photo interpreters, who employed it to re-annotate and correct the AerialWaste 1.0 dataset.

6. Conclusions

In this work, the ODIN Web application has been extended. ODIN Web is a tool that integrates, in a web-based application, dataset management, image annotation for Computer Vision tasks, and model performance investigation and comparison through the ODIN framework. The tool has been extended with further functionalities realizing a web-based tool that covers the most steps in the model development pipeline, from data annotation to the application of Computer Vision models. The web application was enriched with a user system involving admins and simple users, that allows for collaboration and role-based access control of datasets. Then, model inference execution was integrated, and a geo-visualization interface for geolocalized satellite images and predictions was implemented. The tool was tested and validated internally and externally by ARPA Lombardia. Future works will focus on the integration of external models through REST API and the automatic retrieval of geolocalized images.

References

[1] Alexander Demidovskij, Artyom Tugaryov, Andrej Kashchikhin, Alexander Suvorov, Yaroslav Tarkan, Fedorov Mikhail, and Gorbachev Yury. Openvino deep learning workbench: towards analytical platform for neural networks inference optimization. In *Journal of physics: Conference series*, volume 1828, page 012012. IOP Publishing, 2021.

[2] Piero Fraternali, Luca Morandini, and Sergio Luis Herrera González. Solid waste detection in remote sensing images: A survey. *arXiv preprint arXiv:2402.09066*, 2024.

[3] Derek Hoiem, Yodsawalai Chodpathumwan, and Qieyun Dai. Diagnosing error in object detectors. In *European conference on computer vision*, pages 340–353. Springer, 2012.

[4] Alessandro Mastropasqua. ODIN Web: an interactive dashboard for black-box deep learning error diagnosis. Master's thesis, Politecnico di Milano, ING - Scuola di Ingegneria Industriale e dell'Informazione, 2022.

[5] Rocio Nahime Torres, Piero Fraternali, and Jesus Romero. Odin: An object detection and instance segmentation diagnosis framework. In *Computer Vision–ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*, pages 19–31. Springer, 2020.

[6] Rocio Nahime Torres, Federico Milani, and Piero Fraternali. Odin: Pluggable meta-annotations and metrics for the diagnosis of classification and localization. In *International Conference on Machine Learning, Optimization, and Data Science*, pages 383–398. Springer, 2021.

[7] Niccolò Zangrando. The ODIN framework, a tool for image classification diagnosis. Master's thesis, Politecnico di Milano, ING - Scuola di Ingegneria Industriale e dell'Informazione, 2021.