POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

ODIN Web: An interactive dashboard for black-box deep learning error diagnosis

Laurea Magistrale in Computer Science And Engineering - Ingegneria Informatica

Author: Alessandro Mastropasqua

Advisor: Prof. Piero Fraternali

Co-advisors: Federico Milani, Rocio Nahime Torres, Niccolò Zangrando

Academic year: 2021-2022

1. Introduction

Classification, Object Detection, and Instance Segmentation have become key fields in the computer vision research context. Their aim is to recognize the presence, or presence and location, of a specific class within an image in an automatic way using neural models. Over the years, however, the architectures of these models have become progressively complex due to the constant need to increase their performance. This has made their evaluation more and more arduous, leading to evaluate them as "black-boxes" using standard evaluation metrics necessary to perform fair analysis on models and understand their behavior to be able to optimize it for a given dataset or task.
In this way, several papers describe the development of frameworks aimed at enhancing the diagnosis of deep neural models over the standard evaluation metrics, implementing tools with graphical interfaces that support the user during all analyzes by presenting the results cleanly and legibly. The interfaces that these tools present are often sparse and do not allow the user to vary the types of analysis, nor do they offer sets of metrics to fully understand the behavior of the model.

ODIN Framework aims at generalizing and integrating into a unique solution the main approaches to error diagnosis, extending the standard evaluation metrics with custom properties and metrics, a wide range of off-the-shelf metrics, and analysis reports. ODIN is a tool with great potential but with the limit of being python based and accessible only to users capable of programming. Therefore, we felt the need to compensate for this weakness by extending it with a dashboard that can quickly display the most common reports without any programming effort, still allowing the more experienced to refine the analysis as they want. The purpose of the presented work is to introduce ODIN Web, a web-based application capable of compensating for the lack of a complete tool, with an intuitive interface, easy to install, and which supports users in analyzing the results of their models without any programming effort. Finally, to demonstrate the utility and effectiveness of the tool, two types of use cases, applied to the ArtDL(not covered in the executive summary for reasons of space) and PASCAL VOC 2007 datasets, are illustrated.

2. Related Work

2.1. Black-box error diagnosis

To fulfill the tasks that computer vision offers us, it is necessary to use deep neural models, and, to understand the benefits of a given model, it is necessary to analyze and catch the characteristics of its architecture.
The model performance analysis can be performed in two ways: the first approach is to "open the box" to find a link between the input, the various internal layers of the model, its nodes, and the output provided by it. The other, that is what we will discuss in this research and implemented by ODIN, is the one called 'black-box' and consists in associating extra annotations to the input, which are not exploited in the training phase, but capable to make us better understand the model output by analyzing which meta-annotations had the greatest impact on model errors. Our work focuses on the research and study of the most popular tools that offer analysis of neural models, exploiting a black-box approach, through an interactive and effortless programming interface. A first interface approach can be traced back to the *Confusion Wheel* concept introduced by Alsallakh et al. in which authors propose a new graphical analysis that arranges different classes based on a radial layout and use histograms to show the statistics of the true/false positive/negative associated with each class and their prediction confidence. However, the technique cannot support an effective comparison of multiple models. Moving forward over the years, more and more tools such as *ModelTracker*, *Prospector*, *Squares* begin to establish themselves, but they are aimed at expert people, with interfaces that are difficult to interpret for novices and lacking all the necessary analyzes to better understand the performance of the model. *Manifold* is the first example of a tool in which performances were compared on multiple models in an effective way. In other words, the outputs of several models were compared trying to understand the strengths and try to lead to a biased cognition based on their relation to the underlying data. After it, some of the best-known tools in the context of black-box analysis have emerged over the next years. First of all *WhatIfTool*, a web-based tool that leverages a visual interface to help understand class distributions in the dataset and Image Classification model output. A user can manage different characteristics of the input data set to analyze how these changes affect the predictions of the model. Furthermore, the tool provides a fairness analysis and different strategies for optimization. Finally, OpenVino, a stand-alone tool that provides 360-degree support for analyzing the results of a model, made its appearance. It has an intuitive but at the same time functional and complete interface for the analysis of predictions. It focuses on model analysis, optimization, and deployment. Novel visualizations and evaluations are introduced to support hardware optimization and model calibration. Computational graph visualization allows developers to investigate the runtime representation of a model. Calibration techniques enable the acceleration of model performance while decreasing memory impact (keeping into consideration accuracy) and deployment to a target system. Finally, as it could be deduced from the analysis, there are very few tools that allow having an interactive interface and at the same time offer the user the complete customization of the data and the analyzes performed on them. Furthermore, each tool focuses mainly on a certain type of task, completely excluding the other.
For this reason we think that ODIN Framework could be a valid alternative for analyzing model results and their behaviour. In addition, the development of a user-friendly and captivating interface equipped with all the necessary functionalities to better analyze the results of the analyzes, would allow to approach even users who do not have a great deal of experience in programming with python.

2.2. Annotation tools for ML datasets

Training models is one of the key part when it comes to deep learning. Often the model's performance, after being trained, depends on the quality and breadth of data that has been provided to it. For that reason, we can say that the preparation of the training set is crucial to obtain excellent performance from the model and, therefore, it must be supported by a tool that allows a quick, but at the same time efficient, annotation. There are plenty of annotators on

the market, each of which specializes in annotating certain types of data. These data can range from simple text files to images up to multimedia files. The researcher must be able to identify the objective of the model, the development environment, and the effectiveness of the annotator that is going to be used. With *PhotoStuff* authors propose one of the first platform-independent, image annotation tools which use an ontology to provide the expressiveness required to assert the contents of an image, as well information about the image. Over the years, the annotation of datasets has become increasingly fundamental, giving birth to the need for increasingly effective tools that allow the user to annotate huge amounts of data quickly and precisely. In this way, we have identified some of the most relevant proposals that can be found at the moment. First of all *TagTog*, a text annotation tool that can be used to annotate text either automatically or manually. It supports native PDF annotation and includes pre-trained NER templates for automatic text annotation. In addition to the Tagtog tool, the company provides an annotation service through a network of skilled workers in the required sector who will annotate the texts under commission. *ImgLab* is the most used image annotator at the moment, based entirely on web, but it also offers the possibility to be installed locally. It allows multiple tasks including object detection, Semantic and Instance segmentation with a user-friendly interface and rich documentation. After that, *Label Studio* is a complete 360-degree tool that covers all aspects of data annotation, as it offers services for all types of annotations. The installation of the tool could be performed locally or deployed in a cloud instance and has a unique configuration setup called Labeling Config that allows you to design your own customized UI. Finally, *COCO Annotator* provides many distinct features among which: the ability to label an image segment, track object instances, label objects with disconnected visible parts, and efficiently store and export annotations in the well-known COCO format. It also exploits advanced tools, such as DEXTR, MaskRCNN, and Magic Wand to execute automatic annotation. From this survey, it emerged that most of the annotation tools are web-based to avoid, probably, the installation of the software locally and to exploit the computational power of the server to perform more complex tasks needed to prepare the data set. In fact, it is possible to notice how most of the tools based on image annotation prefer an approach based on instance segmentation while not neglecting object detection. In most cases, especially concerning recently developed tools, the annotation process is not totally left in the hands of the user, but instead, the researcher is the person who completes the annotations of images not recognized by the annotation system. Over the years, standard formats have also been set for saving on file (also called ground truth) of annotations. In fact, most of the files, both input and output, are generated as JSON (Javascript Object Notation) files with MS COCO as the standard format. As for the annotations saved in XML, it is assumed as standard format the one dictated by Pascal VOC.

3. An interactive dashboard for ODIN

This work aims to develop a web interface capable of exploiting all the features of the previous work [1] and [2], implementing an application capable of relieving the user of any programming effort, with the aim of:

- providing the user with a simple and intuitive tool in use. For this reason a README guides the user in the installation process: from cloning the repository, to installing the required packages, and starting the dashboard. To relieve the user to those steps, we also provide ODIN Web as a Docker[1] container in order to benefit from the features that dockerizing the app offers us. The decision to offer ODIN also in dockerized format stems from the aim of making ODIN a tool that is simple to use but also to be installed, ensure the possibility to make ODIN accessible in different ways.
- support the user throughout the management of the data set
- support the user in the comprehension of the results provided by the model using meta-annotations, i.e., annotations that do not contribute to model training but can be exploited for understanding performance

In order to meet the requirements setted for

[1] https://www.docker.com/

ODIN, the application flow is divided into two main components:

- *Annotator*: allows the user to access, or if necessary create, the data set to provide to the model. Once created, the user can populate the data set and specify, for each observation, the category, or categories in case of multi-label classification, it represents and the properties that characterize it. Once all the samples are annotated, they will be saved in the right format on the ground truth file.
- *Analyzer*: allows the user to analyze the predictions of one or more models easily and without programming effort. By simply selecting the dataset, the predictions associated with it, and the settings with which you want to perform the analyzes, the user can access all the functions that ODIN Framework already offers, provided with a simple interface and equipped with all the filters necessary for inspection and comprehension of the analyzes at different levels of granularity.

Talking about the architectural part of ODIN Web, it is organized with a client-server architecture to keep the presentation, application, and data logic separated.
Starting from the client-side, here lies everything about the presentation logic. It was chosen to take advantage of Vue.js to implement the ODIN user interface, a javascript framework that exploits the concept of *declarative rendering* to break down the application into modules, making the code cleaner and tidier. Its choice was guided by the fact that it allows to develop a software with a reactive MVVM architecture. In fact, the user interacts with the HTML based web page and normally the whole page has to refresh even if just one object changes. Vue uses a virtual copy of the original DOM that figures out what elements require updating, without re-rendering the entire DOM, greatly improving app performance and speed.
All client side data is retrieved from the server through promise-based HTTP requests to the server-side via Flask made by the Axios JS library extension. ODIN Web uses Flask as web server mainly for its flexibility, scalability for simplistic application, and adaptability to the latest technologies. In addition, the *App Routing* functionality allows to map specific URLs with associated functions that are built to perform a specific task.
The application and data logic resides in the server-side. It is developed entirely in Python, following the line of the previous work [2][3]. A modules structure was designed to maintain a separation of what are the features offered and to facilitate subsequent extensions of the app. All data concerning the analyzes obtained through the client's requests are processed by ODIN Framework, and the results are reorganized to be effortlessly processed by the client. Moreover, in order to not affect the user experience, Redis was used as a memory to store the data structure converted into Bitmap of the instances of the objects associated with the user as regards Dataset, Analyzer and Comparator. It was chosen, first of all for its performance, as it offers response times of less than a millisecond, allowing millions of requests per second. Also, since Redis data resides in memory, it enables data access with low latency and high throughput. The main advantage is that, unlike traditional databases, data in memory does not require going to disk, minimizing engine latency to microsecond levels. In fact, in anticipation of a large number of annotations, it was decided to adopt MongoDB's document-oriented NoSQL database to be able to perform queries instantly without compromising the user experience and limit the number of accesses to the ground truth file. MongoDB is used as it was developed specifically for the management of unstructure data, allowing them to be stored in json format. In this way it is possible to save automatically and without manipulating the annotations provided by the client, in addition it allows to update the schemes quickly.

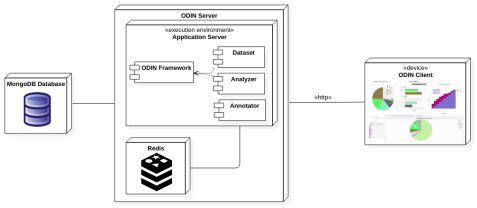

Figure 1: ODIN Web architecture

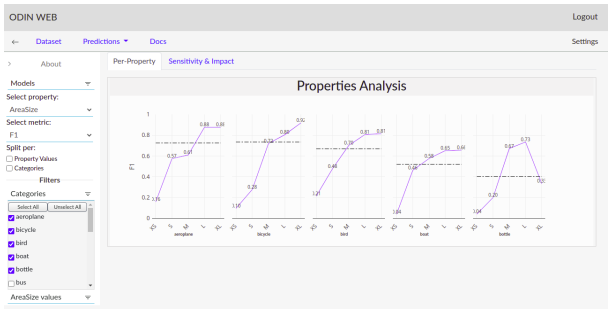

Figure 2: Per-Property Analysis of Faster-RCNN model over a categories subset

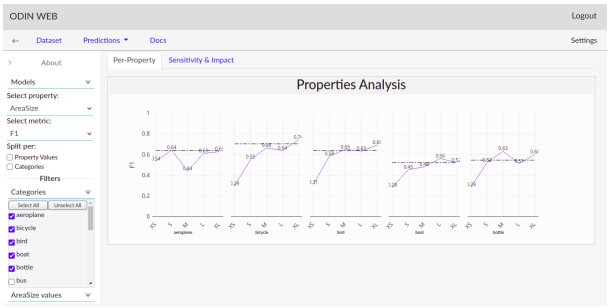

Figure 3: Per-Property Analysis of FPN model over a categories subset

4. ODIN Web in action

ODIN Web functionality is illustrated by applying it on an object detection benchmark: the PASCAL VOC 2007's test set, a time-honored dataset to evaluate performance in object category detection which contains 4952 images depicting 20 different classes. For each annotation, a set of properties is calculated:

- AreaSize: indicates the dimension of its bounding box: XS (extra-small), S (small), M (medium), L (large), and XL (extra-large).
- AspectRatio: is defined as object width divided by its height. Similarly to object size, objects are categorized into extra-tall (XT), tall (T), medium (M), Wide (W), and extra-wide (XW).

In this analysis, we compare two models: Faster-RCNN (with MobileNet backbone) and the same with the addition of the Feature Feature Pyramid Network (FPN) [2]. The models are compared in ODIN Web to catch the strengths and weaknesses of each. In Fig. 2 the Per-Property analysis made over the Area Size property for the Faster-RCNN model is reported. It is possible to notice how the performances decrease as the size of the object decreases. One possible solution, in case the researcher wants to trade some performance aspects with the recognition of the presence of particularly small objects, it could be adopt a different type of model, for example, a Feature Pyramid Network (FPN). Fig. 3 shows the performance of the FPN model and, as expected, it presents higher performance for annotations associated with properties values, such as XS and S, but significantly decreasing performance in cases where the size is larger proving that, for this particular model, performances are less affected by the size of the bounding boxes. Fig. 4 represents the sensitivity and impact

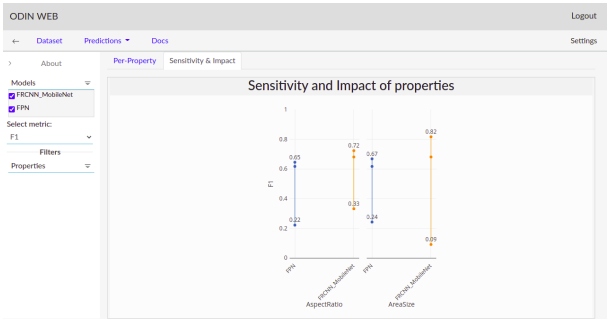

Figure 4: Sensitivity and Impact of the Faster-RCNN and FPN compared

graph that compares the two models simultaneously. As already said, it is possible to see how FPN is much less sensitive and with a lower impact than Faster-RCNN, while as regards aspect ratio property the two models have similar behaviors. From this analysis, we can conclude that the FPN model would be a valid substitute for the Faster-RCNN if the researcher is willing to give up to 6% of the overall performance while having a model that recognizes all types of bounding boxes. By analyzing Fig. 5, it re-

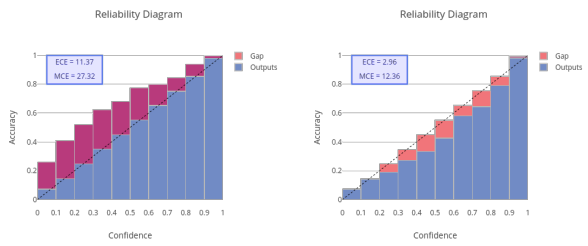

Figure 5: Reliability Diagram of the Faster-RCNN(left) and FPN(right)

[2] The weights are obtain from public repositories.

ports the reliability diagram applied to the two models separately, we can see how, unlike the FPN model, the Faster-RCNN model is not well calibrated. In this case, the model is underestimating its predictions, presenting an expected calibration value(ECE) of 11.37. This condition can be improved by applying some calibration technique such as Bayesian Binning into Quantiles, Platt scaling, or its simplest version, temperature scaling. Focusing on the FPN model, we can notice that it is well calibrated with an expected calibration value(ECE) of 2.96 while overestimating its predictions. This behavior is also noted by analyzing the distribution of False-Positives, in Fig. 6, between the two models, in which FPN commits almost twice as many errors compared to Faster-RCNN.

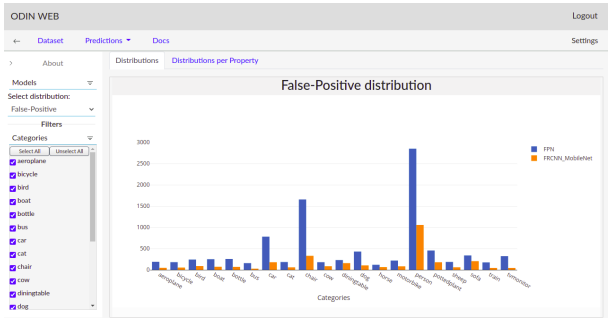

Figure 6: False-Positive distribution of the Faster-RCNN and FPN compared

5. Conclusions

In this work, ODIN Web, a web app that aims to extend the already existing ODIN framework, is presented. ODIN Web offers the possibility to exploit all the functionalities provided by ODIN Framework simply and without any programming effort. Two of the main features of ODIN are provided with a graphical interface: the Annotator can guide the user in a simple but effective way during the entire phase of the dataset population, offering a clean and user-friendly interface. As for the analyzer, it provides all the analyzes implemented by ODIN Framework, allowing the user to view the results in graphical format and providing all the filters necessary to view only the data of interest. It also allows you to compare multiple models at the same time and update the analysis parameters in real-time. In conclusion, it has been demonstrated that in the current state of the art in the field of black-box analysis frameworks there are very few tools that allow having an interactive interface and at the same time offer the user the complete customization of the data and the analyzes performed on them. Therefore, we think that ODIN Web could be a considerable step forward that can allow anyone to interface with the world of computer vision in a simple way but, at the same time, can also support researchers, and not, to extract multiple and useful information for the refinement of their models. As for future works, we expect to expand ODIN web with features already present in the framework, such as: implement a *Visualizer* component to give the user the ability to fully explore the ground truth and predictions of the model, allow the annotator to support textual observations in the annotation phase and expand the types of meta-annotations offered by the annotator: range of values and textual parameters, and implement automatic extraction of certain types of meta-annotations such as: object-count and colors. We also plan to add support analysis for Class Activation Maps (CAMs), both from a quantitative and qualitative point of view.

References

[1] Rocio Nahime Torres, Piero Fraternali, and Jesus Romero. Odin: An object detection and instance segmentation diagnosis framework. In Adrien Bartoli and Andrea Fusiello, editors, Computer Vision – ECCV 2020 Workshops, pages 19–31, Cham, 2020. Springer International Publishing.

[2] Rocio Nahime Torres, Federico Milani, and Piero Fraternali. Odin: Pluggable meta-annotations and metrics for the diagnosis of classification and localization. In Giuseppe Nicosia, Varun Ojha, Emanuele La Malfa, Gabriele La Malfa, Giorgio Jansen, Panos M. Pardalos, Giovanni Giuffrida, and Renato Umeton, editors, Machine Learning, Optimization, and Data Science, pages 383–398, Cham, 2022. Springer International Publishing.

[3] 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.