ODIN: pluggable meta-annotations and metrics for the diagnosis of classification and localization

Rocio Nahime Torres[1], Federico Milani[1], and Piero Fraternali[1]

Politecnico di Milano
Piazza Leonardo da Vinci, 32, Milano, Italy
{rocionahime.torres, federico.milani, piero.fraternali}@polimi.it

Abstract. Machine Learning (ML) tasks, especially Computer Vision (CV) ones, have greatly progressed after the introduction of Deep Neural Networks. Analyzing the performance of deep models is an open issue, addressed with techniques that inspect the response of inner network layers to given inputs. A complementary approach relies on ad-hoc metadata added to the input and used to factor the performance into indicators sensitive to specific facets of the data. We present ODIN an open source diagnosis framework for generic ML classification tasks and for CV object detection and instance segmentation tasks that lets developers add meta-annotations to their data sets, compute performance metrics split by meta-annotation values, and visualize diagnosis reports. ODIN is agnostic to the training platform and input formats and can be extended with application- and domain-specific meta-annotations and metrics with almost no coding. It integrates a rapid annotation tool for classification and object detection data sets. In this paper, we exemplify ODIN through CV tasks, but the tool can be used for generic ML classification.

Keywords: computer vision · metrics · evaluation · diagnosis

1 Introduction

In ML the availability of data sets and open challenges allows the creation of baselines essential for progress. Benchmarks rely on standard metrics to compare alternative methods. However, the metrics used for assessing the end-to-end performance with a black-box approach may not be the most adequate ones for understanding the behavior of an architecture and for optimizing it for a certain data set or task. The analysis of performance can be pursued in two ways. On one side, model interpretation techniques aim at "opening the box" to assess the relationship between the input, the inner layers and the output. For example attention models capture the essential region of the input that have most impact on the inference [31]. On the other hand, it is possible to associate the images with *meta-annotations*, i.e., annotations that do not contribute to model training but can be exploited for understanding performance. Such performance-driven meta-annotations enable the computation of task- and data set-specific metrics that may help the diagnosis. As an example of meta-annotations used

to fine tune standard metrics, the MS COCO data set [15] differentiates the Average Precision (AP) metric based on the size of the detected object (small, medium or large), permitting researchers to focus their improvement on the sub-classes where they expect the most gain. The work in [9] is a pioneering effort to exploit meta-annotations in the evaluation of detectors. The described method and tool help developers evaluate the impact on performance of selected dimensions of the data (object size, aspect ratio, visibility of parts, viewpoint) or of the errors (occlusion, wrong localization, confusion with other objects and with background). In [28] we followed the line of [9] and implemented a preliminary version of ODIN, a tool supporting the diagnosis of errors in object detection and instance segmentation components allowing the plug-in of meta-annotation types and metrics. In this paper, we extend the work in [28] with novel metrics and analysis reports and expand the analysis to classification, in addition to object detection and instance segmentation. The contribution of our work can be summarized as follows:

- We present ODIN, a tool for error diagnosis applicable to generic ML classification tasks and to CV object detection and instance segmentation tasks.
- We exploit the plug&play architecture of ODIN and add a broad set of off-the-shelf metrics and reports, also for classification tasks.
- We extend the user interface supporting the editing of annotations and meta-annotations (Fig. 1) to classification tasks, in addition to localization ones. The new User Interface (UI) can be used not only for meta-annotation editing but also for the creation of classification and object detection data sets.
- We add confidence calibration [7] as a novel type of analysis.
- We showcase the use of ODIN in the diagnosis of two classification scenarios.
- We release the code publicly[1]. ODIN is developed in Python, integrates directly with the MS COCO data set and is agnostic to the training platforms.

2 Related Work

Deep learning models are typically used and evaluated as black-boxes. Performance analysis exploits standard metrics (Accuracy, Precision, Recall, F1-Score or Average Precision) implemented off-the-shelf in most evaluation frameworks. Several works [18,10,21,20] explain the standard metrics and discuss issues and best practices. The standard metrics asses a model end-to-end and thus the problem arises of how to analyse its behavior to diagnose weaknesses and improve representation power and inference accuracy. The investigation of model behavior is pursued with two complementary approaches. One line of research aims at improving model interpretability, by studying the internal representations of deep models and their relation to the input [32]. An alternative approach is to study how the properties of the input samples influence performance. The latter approaches aim at factoring out performance indicators based on the properties

[1] `https://github.com/rnt-pmi/odin`

of the input samples and at identifying the characteristics of the inputs with greatest impact on performance.

A first step towards the diagnostic use of objects properties is found in the MS COCO data set, where the computation of mean Average Precision (mAP) is differentiated based on object size: $\text{mAP}_{\text{small}}$, $\text{mAP}_{\text{medium}}$ and mAP_{big}. This distinction can help diagnose problems and apply proper techniques, e.g. multi-scale object detection [5], to improve localization ability. The work in [9] introduces a more general approach to analyse errors in object detectors. The proposed framework exploits diagnosis-oriented metadata (called *meta-annotations*, in this paper) that can affect the model accuracy. The authors employ a fixed set of such meta-annotations: occlusion, size, shape, aspect ratio, and parts visibility. They show how decomposing the standard metrics into sub-metrics associated to a meta-annotation value helps understand model failures and focus redesign where the margin of improvement is higher. In [25] the authors present a diagnostic tool tailored to the study of pose estimation errors. The tool analyzes the influences of fixed object characteristics (e.g., visibility of parts, size, aspect ratio) on the detection and pose estimation performance and enables the study of the impact of different types of pose-related False Positives. In [1] the focus is on the localization of temporal actions in videos. The diagnosis method and tool allow False Positive (FP) and False Negative (FN) analyses and the estimation of the sensitivity of mAP-based metric to six action characteristics: context size, context distance, agreement, coverage, length, and the number of instances. In [19] the authors apply the original diagnostic methodology of [9] to the case of semantic segmentation but do not provide a public implementation of their toolkit. REVISE (REvealing VIsual biaSEs) [29] is a tool that permits the investigation of bias in image data sets along three dimensions: object-based (size, context, or diversity), gender-based, and geography-based (location). The set of properties is fixed and the tool aims at revealing biases in the visual data sets rather than at supporting the diagnosis of errors by models, addressing data set curators rather than model builders. TIDE [2] is a tool for error diagnosis in object detection and instance segmentation applicable across data sets and output prediction files. The focus is on the finer classification of detection and segmentation error types and on the provision of compact error summaries and meaningful impact reports. Unlike ODIN, TIDE purposely avoids resorting to meta-annotations of the input. Finally, the most recent version of the object detection evaluation toolkit by Padilla et al. [22] generalizes the previous version: it makes the framework independent of the input formats, adds novel object detection metrics and bounding box formats, and provides a novel spatio-temporal metric for object detection in video.

In some critical ML applications, where the output of the model is used to make complex and risky decisions, the reliability of predictions is crucial. This property can be investigated by assessing the extent to which the predicted probability estimates of outcomes reflect the true correctness likelihood, a process called *confidence calibration* [7,13]. Confidence calibration is not normally

employed in current neural model design and validation practices, where output confidence values are simply cut-off at a threshold to decide the output class.

ODIN aims at generalizing and integrating into a unique solution the previous approaches to error diagnosis for neural models. It supports classification, object detection and instance segmentation, allows the addition of custom meta-annotations and metrics, and includes a wide range of off-the shelf metrics and analysis reports. It combines error impact sensitivity and confidence calibration analysis. It can be used to study both model performance and data set bias. In [28], we presented a preliminary version developed along the line of [9], which featured support for instance segmentation tasks, adapting the input to the most common data set formats and providing an easy-to-use Python implementation. The version presented in this paper extends our previous work [28] by: 1) adding support for classification tasks in addition to object detection and instance segmentation thus making ODIN applicable to generic ML classification problems; 2) increasing the number of metrics available off-the-shelf; 3) integrating a novel type of analysis for confidence calibration, supporting the evaluation of the confidence error; 4) integrating a GUI for input annotation so to provide a one stop solution spanning all phases from training set preparation to model evaluation.

To the best of our knowledge no other tool offers such a complete set of functions for dissecting the performance of ML and CV models.

3 The ODIN Error Diagnosis Framework

The ODIN framework supports the development of (image and generic) classification, object detection and instance segmentation models by enabling designers to add application-specific meta-annotations to data sets, evaluate standard metrics on inputs and outputs grouped by meta-annotation values, assess custom metrics that exploit meta-annotations, evaluate the confidence calibration error, and visualize a variety of diagnostic reports.

3.1 Metrics

ODIN supports both standard metrics for (image) classification, object detection and instance segmentation assessment and their restriction to specific properties expressed by the meta-annotation values. Developers can run all metrics, or a subset thereof, for a single class or a set of classes. The values of all metrics are reported using diagrams of multiple types, which can be visualized and saved. Table 1 summarizes the implemented metrics for each task.

Normalization The metrics that depend on the number of true positives TP (e.g., precision, average precision, accuracy, recall) can be defined in two variants: with and without normalization [9]. Normalization copes with class unbalance. The number of TP is affected by the size of a class as follows. If T_c is the fraction of objects detected with confidence of at least c and N_j is the size of class j, then $\text{TP} = T_c \cdot N_j$. Thus, for the same detection rate, the metrics that depend on TP grow with N_j, which may be undesirable if classes are unbalanced. Normalization

Table 1: Metrics implemented for Binary/Single Label Classification, Multi Label Classification, Object Detection and Instance Segmentation

Metric / Analysis		Binary	SL Class.	ML Class.	Obj. Det.	Ins. Seg.
Base Metrics	Accuracy	X	X	X	-	-
	Precision	X	X	X	X	X
	Recall	X	X	X	X	X
	Average Precision	X	X	X	X	X
	ROC AUC	X	X	X	-	-
	Precision Recall AUC	X	X	X	X	X
	F1 Score AUC	X	X	X	X	X
	F1 Score	X	X	X	X	X
	Custom	X	X	X	X	X
Curves	Precision Recall	X	X	X	X	X
	F1 Score	X	X	X	X	X
	ROC	X	X	X	-	-
Confusion Matrix		X	X	X	-	-
Metric per Property Value		X	X	X	X	X
Property Distribution		X	X	X	X	X
Sensitivity and Impact Analysis		X	X	X	X	X
False Positives Analysis		-	X	X	X	X
FP, TP, FN, TN Distribution		-	X	X	X	X
Calibration Analysis		X	X	X	X	X
Base Report	Total value	X	X	X	X	X
	Per-category value	X	X	X	X	X
	Per-property value	X	X	X	X	X

replaces N_j with a constant N, which represents the size that each class would have in balanced conditions ($n_{obs}/n_{classes}$). Normalization is optional and can be enabled both class-wise and for specific meta-annotation values.

Thresholding Some of the curves supported by ODIN require computing values of the corresponding metric for different threshold values. A threshold t defines the confidence value above which a prediction is considered positive. For object detection and instance segmentation, a threshold t_{IoU} applies to the Intersection over Union (IoU) value between the proposals and ground truth objects.

Accuracy represents the fraction of correct predictions of a model (Eq. 1).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

Precision, Recall, PR curve, PR AUC and Average Precision The precision and recall metrics have the usual definition (Eq. 2). The PR curve plots the precision vs recall at different thresholds and can be computed for all classes, per class, or on a subset of the classes [23].

$$Precision = \frac{TP}{TP + FP}, Recall = \frac{TP}{TP + FN} \tag{2}$$

Average Precision (AP) summarizes the PR curve as a single value, i.e., the precision averaged for recall values from 0 to 1, equivalent to the Area Under the PR Curve. The Interpolated AP approximation used in PASCAL VOC [6] is computed as the sum of the maximum precision values for recall greater than the current sampling value, weighted by the recall delta (Eq. 3 and 4).

$$\sum_{r=0}^{1}(r_{n+1} - r_n)\, p_{interpol}(r) \tag{3}$$

with

$$p_{interpol}(r) = \max_{\tilde{r}:\tilde{r}\geq r_{n+1}} p(\tilde{r}) \tag{4}$$

where $p(\tilde{r})$ is precision at recall $\tilde{r}$.

F1 Score, F1 Curve, F1 AUC The F1 score is the harmonic mean of Precision and Recall (Eq. 5). The F1 Curve plots the F1 score over all threshold values. The area under such curve defines the F1 AUC metric [16].

$$F1Score = 2 \cdot \frac{p \cdot r}{p + r} \tag{5}$$

ROC Curve, ROC AUC The Receiver Operating Characteristic (ROC) Curve plots the tradeoff between True Positive Rate (TPR) and False Positive Rate (FPR) (Eq. 6). The area under the curve summarizes it into a single value (ROC AUC) [26].

$$TPR = \frac{TP}{TP + FN}, FPR = \frac{FP}{FP + TN} \tag{6}$$

Confusion Matrix presents TN, FN, TP, FP in a matrix form where position (i, j) contains the number of samples in group i predicted as group j.

Custom metrics ODIN has a "plug&play" architecture so that adding new metrics requires extending the `Analyzer` class and providing the wrapper method that calls the code of the metrics.

3.2 Analysis Reports

ODIN supports several types of analysis based on the metrics described in Section 3.1. The user can restrict the analysis to a subset of the classes, to the value of a meta-annotation (henceforth called *property*) or to specific metrics. An example of property is *object size* with values *small*, *medium* and *large*.

Property Distribution Data set bias can be analysed by visualizing the distribution of property values over the whole data set and over individual classes.

Metrics per property The values of a metric can be disaggregated by class and then by property value. For example, the metric M can be computed and visualized for the class *car* and then for the property values *small*, *medium* and *large*. The class mean value is reported too.

Property sensitivity and impact Given a metric, its value is computed limited to the subset of the input data corresponding to each property value. The maximum and minimum values are reported. The difference between the maximum and minimum value highlights the sensitivity of the metrics w.r.t. the property. The difference between the maximum value and the overall value of the metrics suggests the impact of the property on the specific metrics.

FP, TP, FN, TN Distribution The distribution of False Positives, True Positive, False Negatives and True Negatives by class can be reported.

False Positives Analysis The per-class analysis of errors is supported. For detection and segmentation the FP errors are based on: confusion with background (B), poor localization (L), confusion with similar classes (S) and confusion with other objects (O) [28]. For classification we included: 1) Confusion with input samples without annotated class (W). 2) Confusion with similar classes based on a user-defined class similarity relation (S). 3) Confusion with non similar classes (G). The percentage of predictions that fall into each error type is reported along with the absolute improvement obtained by removing the FP of each type.

Calibration Analysis It relies on the confidence histogram and on the reliability diagram [4]. Both plots have the confidence divided into buckets (e.g., 0-0.1, 0.11-0.2, ..., 0.91-1) on the abscissa. The confidence histogram shows the percentage of positive predicted samples that fall into each confidence range. The reliability diagram indicates, for each confidence range, the average accuracy of the positive samples in that range. When a classifier is well-calibrated, its probability estimates can be interpreted as correctness likelihood, i.e., of all the samples that are predicted with a probability estimate of 0.6, around 60% should belong to the positive class [7]. ODIN reports the Expected Calibration Error (ECE) (Eq. 7) and the Maximum Calibration Error (MCE) (Eq. 8)

$$ECE = \sum_{m=1}^{M} \frac{B_m}{n} acc(B_m) - conf(B_m) \tag{7}$$

$$MCE = max_{m\epsilon(1..M)} |acc(B_m) - conf(B_m)| \tag{8}$$

where n is the number of samples in the data set, M is the number of buckets (each of size 1/M) and B_m denotes the set of indices of observations whose prediction confidence falls into the interval m.

Base Report tabulates the chosen metrics: total, per-class and per-property value. The total shows both the micro- and macro-averaged values: the first computes the value by counting the total TP, FP and FN; the latter computes the metrics for each class and then performs an unweighted mean.

3.3 Annotator

Meta-annotations can be automatically extracted (e.g., image color space) or manually provided. Meta-annotation editing is supported by a Jupyter Notebook (Figure 1) that given the data samples (e.g., images) and the meta-annotation

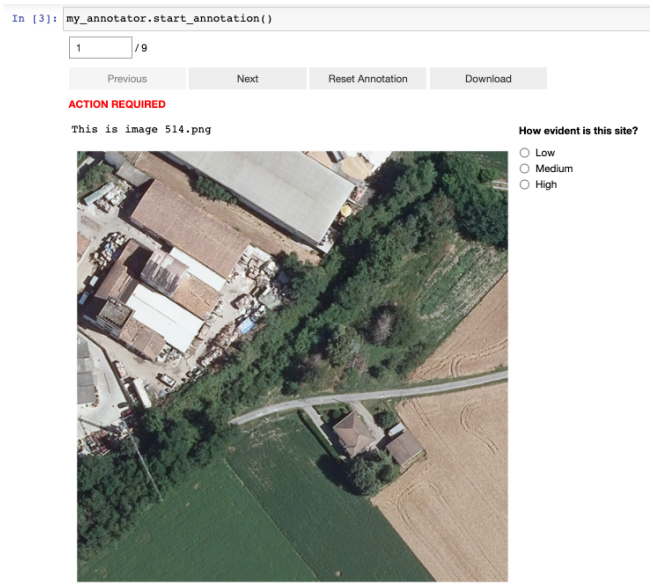

Fig. 1: Interface of the meta-annotation editor. The user can navigate between the images to: add new annotations, reset the current annotations or download the current image.

values allows the developer to iterate on the samples and select the appropriate value, which is saved in the chosen evaluation format. The following example shows the code needed to create, for a classification data set, a meta-annotation session for a custom property (`Evidence`). The code declares the paths to the inputs and outputs (lines 1-2), specifies the task type (line 4), instantiates the data set (line 6), creates a custom visualization (lines 8-16) and a custom validation function (lines 18-19), declares the properties to annotate (with property type, values, optional message) (line 21), creates an instance of the annotator (line 23), with an optional custom display function (if not set, the default visualization is used) and the optional validation function (useful to guide the user). Finally it starts the annotator (line 24). Similar code is used to instantiate an editor for a detection or segmentation task.

```
1 dataset_gt = '../gt.json'
2 images_path ='../images'
3
4 classific_type = TaskType.CLASSIFICATION_BINARY
5
6 my_dataset = DatasetClassification(dataset_gt, classific_type,
  ↪  observations_abs_path=images_path, for_analysis=False)
7
8 def custom_display_function(obs_record):
9     print(f"This is image {obs_record['file_name']}")
```

```
    path_img = os.path.join(images_path, obs_record['file_name'])

    img = Image.open(path_img) # read img from path and show it
    plt.figure(figsize=(10, 10))
    plt.axis('off')
    plt.imshow(img)
    plt.show()

def validate_function(observation_record):
    return 'Evidence' in observation_record

properties = {"Evidence": (MetaPropertiesType.UNIQUE, ["Low",
↪  "Medium", "High"], "How evident is this site?")}

my_annotator = AnnotatorClassification(my_dataset, properties,
↪  custom_display_function=custom_display_function,
↪  validate_function=validate_function)
my_annotator.start_annotation()
```

Data set generation In addition to editing meta-annotations, ODIN also supports the creation of a classification or object detection data set. The annotator can be configured to associate training labels to data samples and to draw bounding boxes over images and label them. The resulting data set is saved in a standard format and can be analysed with the illustrated diagnosis functions.

Data set visualizer A GUI realized as a Jupyter Notebook enables the inspection of the data set. The visualization can be executed on all the samples, limited to the samples of a class, limited to the samples with a certain meta-annotation value, and limited to the samples of a class with a given meta-annotation value.

4 ODIN in action

This section exemplifies the use of ODIN for the evaluation of: 1) a binary classifier of aerial images for predicting the presence of illegal landfills and 2) a multi-label classifier of painting images based on their iconography elements (e.g., Christian Saints).

Illegal landfills For each image the presence or absence of a suspicious site is predicted. Over 1,000 geographical coordinates of illegal waste dumps were provided by experts, each position associated with the evidence level (low, medium, high) and the extension level (low, medium, high). Other 2,000 coordinates were randomly chosen in the same region as negative samples. The evidence and extension meta-annotations are only present for the positive samples. For each location, an image was extracted with the size randomly sampled from three scales (600, 800 or 1,000 pixels) to provide a different amount of context around the center position. The scale constitutes an automatically extracted meta-annotation available for all samples. Given the application, we focus the illustration on the

most relevant metric: recall. Initially, prediction was realized with ResNet-50 [8]. Figure 2 shows the distribution of the scale property among the test images (left) and plots the recall value variation with the scale value (middle). The analysis reveals sensitivity of recall to the scale. Based on this observation, a second model was trained using the same ResNet-50 as backbone augmented with Feature Pyramid Network links [14,24]. The results of such a model show that the sensitivity to the scale reduces (right).

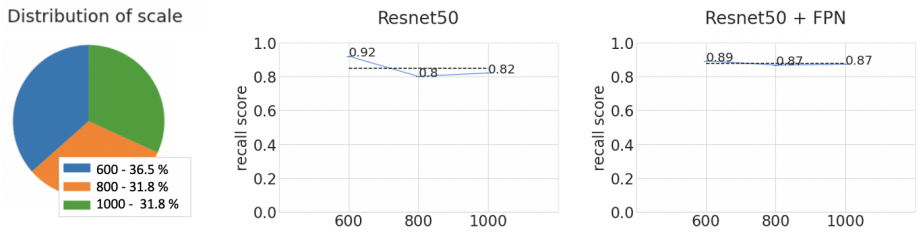

Fig. 2: Distribution of scale property and the analysis of recall for this property.

Figure 3 compares the sensitivity and impact on recall of the three meta-annotations for the ResNet-50+FPN classifier. Now the scale is the least sensitive property whereas the extension has the largest impact; it thus becomes the focus of the next improvement cycle.

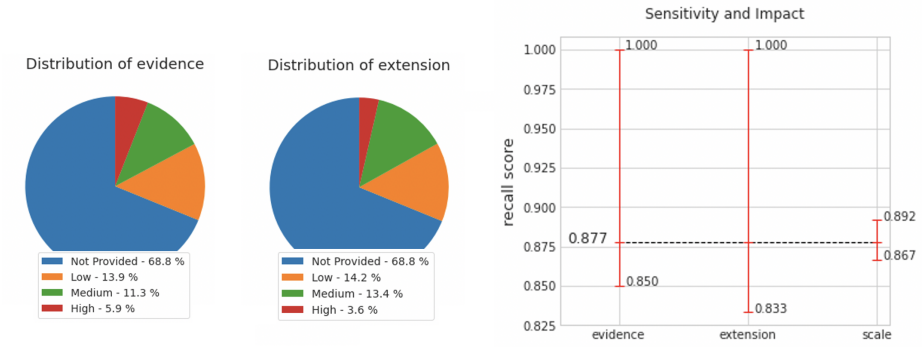

Fig. 3: Distribution of evidence and extension (left). Impact and sensitivity of meta-annotations on recall in ResNet-50+FPN architecture (right).

The predictions are used by local authorities to scan a large territory. Given the operational cost of field inspections, assessing the quality of the probability estimates is important. Figure 4 shows the confidence histogram and the reliability diagram. In most cases, the model is moderately more confidence than it

should be, except in the 0.4-0.5 bucket, where the over-confidence is the highest, giving an MCE of 56. Yet this bin has a small effect on the ECE (7.01) given the low amount of samples in this range, as can be seen in the confidence histogram.

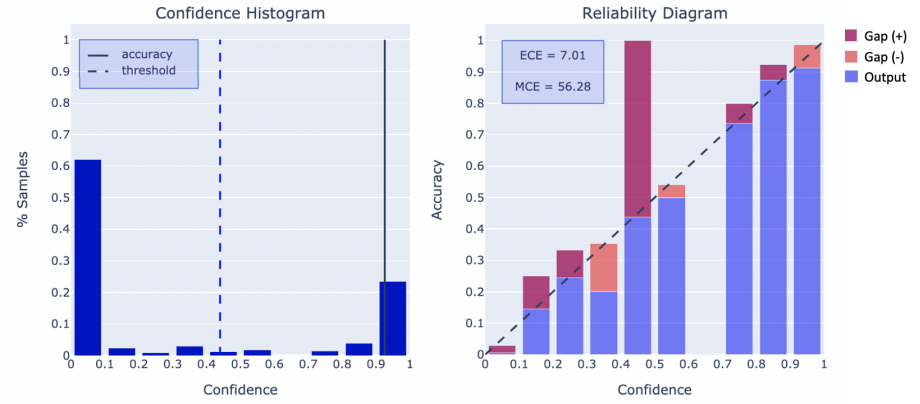

Fig. 4: Confidence histogram and reliability diagram.

ArtDL The ArtDL data set, presented in [17], exemplifies the multi-class multi-label classification case. It comprises 42,279 painting images in 10 classes related to the iconography of Christian Saints. Predictions are made with a trained ResNet-50 model on 1,864 test images made publicly available. Three meta-annotations are used: the color space and the source collection, which determine a great variability in the image quality, and the number of characters depicted in the paintings, which describe the complexity of the scene. All properties are acquired automatically: the source collection is set during the content crawling phase, the color space is found by post-processing the images and the number of characters is counted by extracting the estimated poses from the images with OpenPose [3]. The number of characters is divided into three ranges (0-1, 2-4, 5+). Figure 5 shows the distribution of the three properties in the data set.

Figure 6 shows the analysis of the #characters property. For most classes, the F1 score deteriorates as the complexity of the scene increases. As an example, the F1-Score of Saint Jerome is $\sim 83\%$ when he is the only element in a painting and it drops to $\sim 43\%$ when one to three other characters are present. Saint Dominic and Saint Anthony of Padua share a similar behaviour. For other classes, such as Virgin Mary or Saint Sebastian, the performance drop is rather limited. This is explained by the fact that those two characters are associated with very distinctive visual elements, e.g., Baby Jesus for Virgin Mary or the arrows for Saint Sebastian; such strong symbols make them recognizable even when they appear in a crowded scene or in a polyptych.

Figure 7 shows the distribution of FP predictions over all classes and the FP analysis of two classes: Anthony of Padua and Mary Magdalene. For both

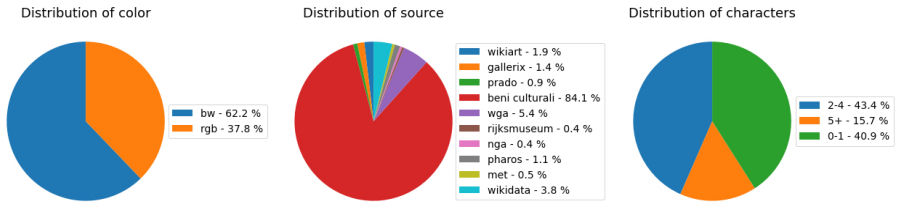

Fig. 5: Distribution of the color, source and #characters in the ArtDL test set. For each property, the distribution of values is reported, e.g., 37.8% of image are RGB and 43.4% of images contain 2 to 4 characters.

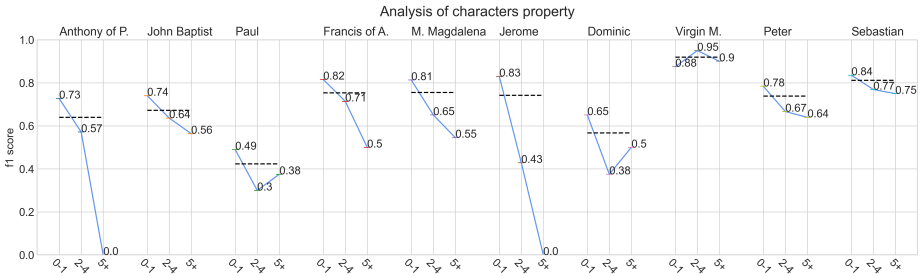

Fig. 6: Analysis of the people property on the ArtDL data set for the F1-Score metric. For each class, the performance calculated on the specific property value is reported. As expected, complex scenes, i.e. images depicting many figures, lead to a decrease of model performance.

classes, most errors occur when predicting images that contain similar classes (S), respectively Saint Francis of Assisi and Virgin Mary, whereas the confusion with unlabeled images (W) is irrelevant. The same analysis also shows the performance gain for the F1-score if the FP errors are mitigated, e.g., a $\sim 7\%$ improvement for Mary Magdalene if she is not confused with Virgin Mary.

The analysis shown in Figures 6 and 7 suggests the application of Fine-Grained Visual Categorization (FGVC) techniques, such as attention aware data augmentation [11,12] or specific attention modules [27,30], to deal with the similarity between classes. These methods should help the model focus on the subtle iconographic symbols that make each class unique, overcoming the issues exposed by the FP analysis.

5 Conclusions and Future Work

In this paper, we have described a framework for the analysis of errors and the visualization of a variety of diagnostic reports in ML and CV tasks. The illustrated work extends a previous preliminary version that focused only on object detection [28] and limited the off-the-shelf analysis to the Average Precision metric.

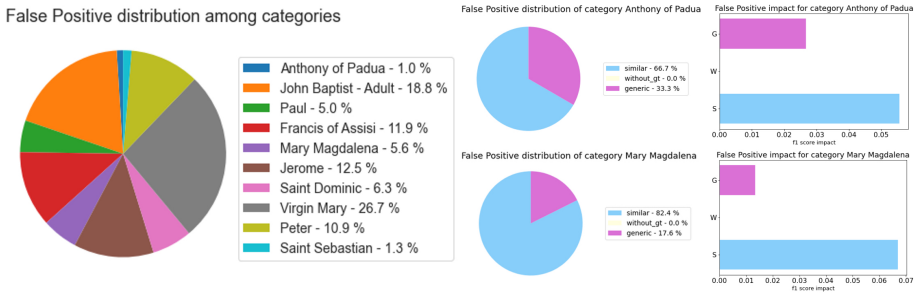

Fig. 7: FP distribution across classes (left) and FP analysis for two classes (right). In the FP analysis, the distribution and impact of the errors are reported.

Now ODIN supports also classification tasks, implements all the most common metrics, and assists with an automatically generated interface the association of arbitrary meta-annotations to the input data also for classification. In addition, an annotator GUI is provided to create a data set from scratch, for both object detection and classification. A new type of analysis, confidence calibration, has been added too. Performance analysis can be focused on standard or custom metrics and on arbitrary subsets of the input characterized by critical values of the meta-annotations. We have illustrated the output on two different settings, scene classification for remote sensing (illegal landfills) and multi-label image classification for cultural heritage (Christian Iconography), explaining how the analysis reports provide insight useful for improvement. For space reasons, other metrics available in ODIN (e.g., Pre/Rec curve, ROC curve, confusion matrix) were not illustrated.

ODIN is implemented in Python and released as open source. Its plug&play architecture permits the addition of novel meta-annotations and custom metrics with minimal coding effort. Our future work will concentrate on extending the library of metrics implementations with further classes for specific applications, e.g., human pose detection and temporal series analysis. In addition, we will add support for the automatic extraction of specific types of meta-annotations from images, such as the geographical coordinates, date and time of acquisition, lighting conditions, etc.. We also plan to integrate the analysis of attention, by computing the position and extent of the CAM [33] w.r.t. to the object bounding box or segmentation mask, with the final goal of supporting the optimization of weakly supervised models.

Acknowledgements This work is partially supported by the project "PRECEPT - A novel decentralized edge-enabled PREsCriptivE and ProacTive framework for increased energy efficiency and well-being in residential buildings" funded by the EU H2020 Programme, grant agreement no. 958284.

References

1. Alwassel, H., Heilbron, F.C., Escorcia, V., Ghanem, B.: Diagnosing error in temporal action detectors. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 256–272 (2018)
2. Bolya, D., Foley, S., Hays, J., Hoffman, J.: Tide: A general toolbox for identifying object detection errors. arXiv preprint arXiv:2008.08115 (2020)
3. Cao, Z., Hidalgo Martinez, G., Simon, T., Wei, S., Sheikh, Y.A.: Openpose: Realtime multi-person 2d pose estimation using part affinity fields. IEEE Transactions on Pattern Analysis and Machine Intelligence (2019)
4. DeGroot, M.H., Fienberg, S.E.: The comparison and evaluation of forecasters. Journal of the Royal Statistical Society: Series D (The Statistician) **32**(1-2), 12–22 (1983)
5. Deng, Z., Sun, H., Zhou, S., Zhao, J., Lei, L., Zou, H.: Multi-scale object detection in remote sensing imagery with convolutional neural networks. ISPRS Journal of Photogrammetry and Remote Sensing **145**, 3–22 (2018). https://doi.org/https://doi.org/10.1016/j.isprsjprs.2018.04.003, `https://www.sciencedirect.com/science/article/pii/S0924271618301096`, deep Learning RS Data
6. Everingham, M., Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. Int. J. Comput. Vision **88**(2), 303–338 (Jun 2010). https://doi.org/10.1007/s11263-009-0275-4, `https://doi.org/10.1007/s11263-009-0275-4`
7. Guo, C., Pleiss, G., Sun, Y., Weinberger, K.Q.: On calibration of modern neural networks. In: International Conference on Machine Learning. pp. 1321–1330. PMLR (2017)
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition (2015)
9. Hoiem, D., Chodpathumwan, Y., Dai, Q.: Diagnosing error in object detectors. In: European conference on computer vision. pp. 340–353. Springer (2012)
10. Hossin, M., Sulaiman, M.: A review on evaluation metrics for data classification evaluations. International Journal of Data Mining & Knowledge Management Process **5**(2), 1 (2015)
11. Hu, T., Qi, H., Huang, Q., Lu, Y.: See better before looking closer: Weakly supervised data augmentation network for fine-grained visual classification. arXiv preprint arXiv:1901.09891 (2019)
12. Imran, A., Athitsos, V.: Domain adaptive transfer learning on visual attention aware data augmentation for fine-grained visual categorization. In: International Symposium on Visual Computing. pp. 53–65. Springer (2020)
13. Kumar, A., Sarawagi, S., Jain, U.: Trainable calibration measures for neural networks from kernel mean embeddings. In: International Conference on Machine Learning. pp. 2805–2814. PMLR (2018)
14. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection (2017)
15. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) Computer Vision – ECCV 2014. pp. 740–755. Springer International Publishing, Cham (2014)
16. Lipton, Z.C., Elkan, C., Naryanaswamy, B.: Optimal thresholding of classifiers to maximize f1 measure. In: Calders, T., Esposito, F., Hüllermeier, E., Meo, R. (eds.)

Machine Learning and Knowledge Discovery in Databases. pp. 225–239. Springer Berlin Heidelberg, Berlin, Heidelberg (2014)
17. Milani, F., Fraternali, P.: A data set and a convolutional model for iconography classification in paintings (2020)
18. Monteiro, F.C., Campilho, A.C.: Performance evaluation of image segmentation. In: International Conference Image Analysis and Recognition. pp. 248–259. Springer (2006)
19. Nekrasov, V., Shen, C., Reid, I.: Diagnostics in semantic segmentation. arXiv preprint arXiv:1809.10328 (2018)
20. Novaković, J.D., Veljović, A., Ilić, S.S., Papić, Ž., Milica, T.: Evaluation of classification models in machine learning. Theory and Applications of Mathematics & Computer Science **7**(1), 39–46 (2017)
21. Padilla, R., Netto, S.L., da Silva, E.A.: A survey on performance metrics for object-detection algorithms. In: 2020 International Conference on Systems, Signals and Image Processing (IWSSIP). pp. 237–242. IEEE (2020)
22. Padilla, R., Passos, W.L., Dias, T.L., Netto, S.L., da Silva, E.A.: A comparative analysis of object detection metrics with a companion open-source toolkit. Electronics **10**(3), 279 (2021)
23. Raghavan, V., Bollmann, P., Jung, G.S.: A critical investigation of recall and precision as measures of retrieval system performance. ACM Transactions on Information Systems (TOIS) **7**(3), 205–229 (1989)
24. Rahimzadeh, M., Attar, A., Sakhaei, S.M.: A fully automated deep learning-based network for detecting covid-19 from a new and large lung ct scan dataset. medRxiv (2020)
25. Redondo-Cabrera, C., López-Sastre, R.J., Xiang, Y., Tuytelaars, T., Savarese, S.: Pose estimation errors, the ultimate diagnosis. In: European Conference on Computer Vision. pp. 118–134. Springer (2016)
26. Sokolova, M., Japkowicz, N., Szpakowicz, S.: Beyond accuracy, f-score and roc: A family of discriminant measures for performance evaluation. vol. Vol. 4304, pp. 1015–1021 (01 2006). https://doi.org/10.1007/11941439_114
27. Sun, G., Cholakkal, H., Khan, S., Khan, F., Shao, L.: Fine-grained recognition: Accounting for subtle differences between similar classes. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 12047–12054 (2020)
28. Torres, R.N., Fraternali, P., Romero, J.: Odin: An object detection and instance segmentation diagnosis framework. In: European Conference on Computer Vision. pp. 19–31. Springer (2020)
29. Wang, A., Narayanan, A., Russakovsky, O.: Revise: A tool for measuring and mitigating bias in visual datasets. In: European Conference on Computer Vision. pp. 733–751. Springer (2020)
30. Zhang, F., Li, M., Zhai, G., Liu, Y.: Multi-branch and multi-scale attention learning for fine-grained visual categorization. In: International Conference on Multimedia Modeling. pp. 136–147. Springer (2021)
31. Zhang, J., Bargal, S.A., Lin, Z., Brandt, J., Shen, X., Sclaroff, S.: Top-down neural attention by excitation backprop. Int. J. Comput. Vision **126**(10), 1084–1102 (Oct 2018). https://doi.org/10.1007/s11263-017-1059-x, `https://doi.org/10.1007/s11263-017-1059-x`
32. Zhang, Q.s., Zhu, S.C.: Visual interpretability for deep learning: a survey. Frontiers of Information Technology & Electronic Engineering **19**(1), 27–39 (2018)
33. Zhou, B., Khosla, A., A., L., Oliva, A., Torralba, A.: Learning Deep Features for Discriminative Localization. CVPR (2016)