ODIN: an Object Detection and Instance Segmentation Diagnosis Framework

Rocio Nahime Torres[1], Piero Fraternali[1], and Jesus Romero[1]

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

Abstract. Object detection and instance segmentation are major tasks in Computer Vision and have substantially progressed after the introduction of Deep Convolutional Neural Network (DCNN). Analyzing the performance of DCNNs is an open research issue, addressed with attention techniques that inspect the response of inner network layers to input stimuli. A complementary approach relies on the black-box diagnosis of errors, which exploits ad hoc metadata on the input data set and factors the performance into indicators sensible or impacted by specific facets of the input (e.g., object size, presence of occlusions, image acquisition conditions, etc). In this paper we present an open source error diagnosis framework for object detection and instance segmentation that helps model developers to add meta-annotations to their data sets, to compute performance metrics split by meta-annotation values, and to visualize diagnosis reports. The framework accepts the popular PASCAL VOC and MS COCO input formats, is agnostic to the training platform, and can be extended with application- and domain-specific meta-annotations and metrics with almost no coding.

Keywords: Object detection, instance segmentation, metrics, evaluation

1 Introduction

Object detection is a key problem in Computer Vision, defined as the task of locating objects of a given class in images [12]. The output of object detection is the specification of the image region where the object appears, normally encoded as the coordinates of a rectangular bounding box. A finer-grain object detection task is instance segmentation, in which the output is an annotation of the image that associates each pixel to the object instance it belongs to [8]. Object detection and image segmentation have innumerable applications, including medical image analysis, surveillance, environment monitoring, autonomous driving, and more [1]. Performances of object detection and instance segmentation methods have rapidly increased, thank to Deep Learning methods and tools. Progress is not only ascribed to improved model architecture and training methods but also to the availability of data sets and open challenges. Data sets widely accepted as standard for benchmarking performance include MS COCO [10], OpenImages [9], and PASCAL VOC [3]. Most benchmarks rely on standardized metrics to enable the comparison of competing methods. The most widely used ones stem from the indicators used for classification: precision, recall, accuracy and their derivatives. True positives are determined by Intersection over Union (IoU), a metric that quantifies the "goodness" of a detection [13]. The availability of recognized performance evaluation standards has played a key role in the progress of the field, enabling researches to compare their results on a fair ground. However, the metrics used for assessing the end-to-end performance of a method and for comparing methods with a black-box approach may not be the most adequate ones for understanding the inner behavior of an architecture and for optimizing it for a certain data set or task. The analysis of a DCNN can be pursued in two different and complementary 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. A notable example is the use of attention models that capture the essential region of the input that have most impact on the inference [18]. On the other hand, it is possible to investigate the performance of a model by enriching the annotations of the images with metadata (or *meta-annotations*, i.e., annotations of the 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 differentiates the AP metrics based on the size of the detected object (small, medium or large), permitting researchers to focus their improvement on the sub-classes of objects where they expect the most gain. The work in [7] is a pioneering effort to systematically integrate meta-annotations in the evaluation of detectors. The authors design a tool suite that allows developers, provided a data set with certain object characteristics, exploit such metadata to evaluate the influence on detection of such dimensions as occlusion, object size, aspect ratio, visibility of parts, viewpoint, localization error, confusion with other objects and with background.

In this paper, we follow the line of [7] and implement a novel tool suite for supporting the diagnosis of errors in object detection and instance segmentation components. The contribution of our work can be summarized as follows:

– We realize an integrated framework for error diagnosis in object detection and instance segmentation components. The framework applies to image data sets in the popular MS PASCAL VOC and COCO formats. It accepts ground truth annotations in the form of bounding boxes and segmentation masks. Its meta-annotation user interface (Fig. 1) supports the enrichment of ground truth annotations with custom meta-annotations representing features of interest. It computes and visualizes custom metrics that highlight the influence and impact of object characteristics on detection or segmentation performance (Fig. 3 to Fig. 6).

– We give the framework a plug&play architecture, whereby model developers can easily add their own meta-annotations and custom metrics with no coding besides the actual implementation of the metrics function.
– We showcase the use of the proposed tool suite in the diagnosis of segmentation performances in the RacePlane dataset [16]. The data set contains several meta-annotations such as sunlight intensity, time of the day, or weather conditions.
– We release the code of the framework publicly[1]. The tool suite is developed in Python, integrates directly with the MS COCO data set format and with the PASCAL VOC format via parsing, and is agnostic to the model training platforms.

2 Related Work

Object detection is the task of recognizing an object of a given class and localize it in the image. A detector distinguishes the object from the background and outputs its position as a bounding box [12]. Instance segmentation has the same goal as object detection, but outputs a pixel-level segmentation mask of each object instance instead of a bounding box per object [8]. Early methods started by computing areas of the image most likely to contain an object (Regions of Interest, ROIs), extracted descriptors for each ROI using features such as SIFT [11] or HOG [2] and then fed such descriptors to a classifier to predict a class label for each ROI. The use of region proposals based on fixed-size sliding windows and of hand-crafted descriptors were the most relevant limitations of such first-generation approaches [6]. The breakthrough of CNNs in the image classification task boosted a novel generation of object detectors, in which both features extraction and region proposals identification were formulated as learning tasks [5]. A further progress occurred with the elimination of the distinct steps for ROI extraction and classification, given the unified prediction of bounding boxes and class probabilities directly from images [15]. The improvement in architectures and models has been fostered by the availability of benchmarking data sets and challenges, thanks to the common practice of publishing standardized methods to measure performance. The popular Pascal VOC 2012 [3] data set for object detection features 20 categories, 11.500+ annotated images and 10.000+ images without annotations. MS COCO [10] contains 80 categories with 123.000+ annotated images and 40.670 images without annotations. Object detection performance is measured with standard metrics such as mean Average Precision (mAP), which relies on the Intersection Over Union (IoU) measure to determine True and False Positives, based on the overlap between a detected and a ground truth object. For example, PASCAL VOC 2012 employs a IoU threshold of 0.5 and MS COCO uses both 0.5 and 0.75 values. Other commonly used metrics are True Positive Rate (TPR) and False Positives Per Image (FPPI). A survey of the metrics commonly employed for object detection and a discussion of the problems associated with their implementation is performed in [13], which also

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

provides a standard implementation easily adapted to the different image annotation formats found in commonly used object detection data sets. The above mentioned indicators afford a quantitative assessment of the overall performance but do not provide insight on how object characteristics affect performance. A first step in the direction of supporting deeper insight by means of finer grain metrics is found in MS COCO, in which mAP is differentiated based on object size into mAP_{small}, mAP_{medium} and mAP_{big}. A more general approach is presented in [7], which specifically focuses on the design of metrics supported by a software framework for object detection error diagnosis. The authors designed an open-source framework for evaluating objects proposals in greater detail, thanks to metadata that expose characteristics that may affect performance, such as occlusion, object size, aspect ratio, visibility of parts, viewpoint, localization error, confusion with other objects and with background. The framework is implemented in MatLab and is equipped with a meta-annotated data set to put the diagnosis approach to work.

In this paper, we proceed along the line of [7]. We extend the set of implemented metrics and visualizations to add support for instance segmentation, modernize the code adapting it the most popular data formats, and give the framework hooks for the easy integration of novel meta-annotations and metrics.

3 An Object Detection and Instance Segmentation Error Diagnosis Framework

The ODIN framework supports the development of object detection and instance segmentation models by enabling designers to add application-specific meta-annotations to the input, evaluate standard metrics on inputs grouped by meta-annotation values, assess custom metrics that exploit meta-annotations, and visualize the results.

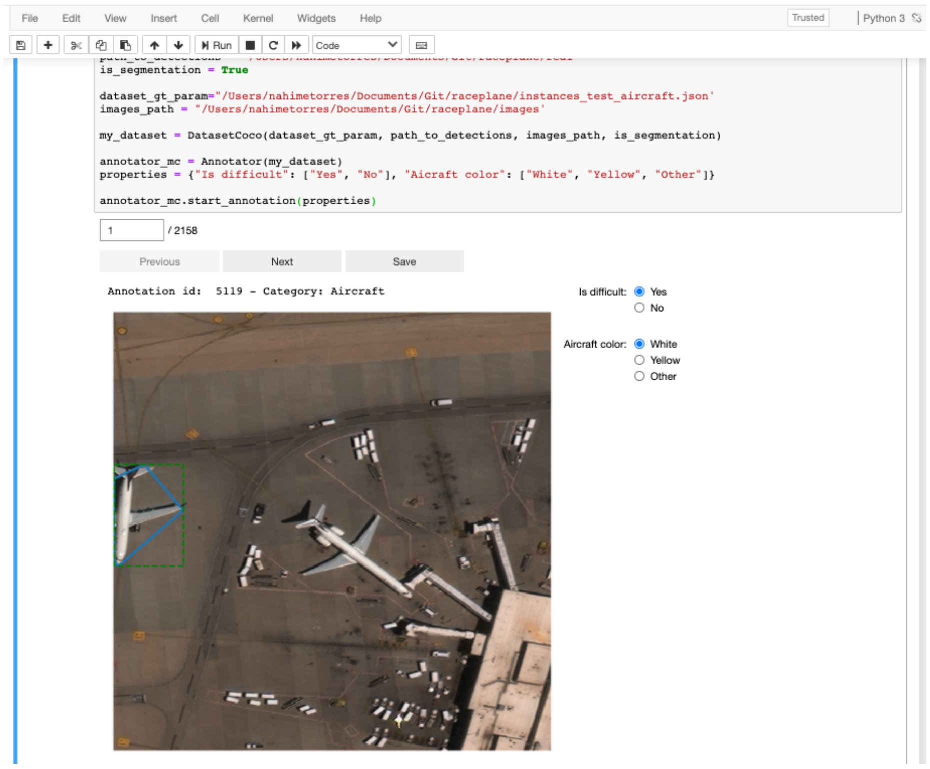
3.1 Input and Meta-Annotations

The ODIN framework accepts as input an image data set (PASCAL VOC and MS COCO format) and a list of objects predictions. Objects predictions can be expressed as bounding boxes for object detection tasks or as pixel masks for instance segmentation tasks. Besides the standard annotations consisting of the object class and location, developers can provide meta-annotations that denote task- or domain-specific characteristics of the whole image or of a ROI. Meta-annotations can convey additional information about:

– ROI size (extra-small, small, medium, large, extra-large) or aspect ratio (extra-tall, tall, medium, wide, extra-wide). If not provided, values can be calculated by the framework based on the areas and shapes of the annotations.
– Object visibility conditions: for example, occlusion level (none, low, medium, high), truncation (no, yes), viewpoint (bottom, front, top, side, rear).

– Object parts, which may denote specific parts of the object in view.
– Image context: e.g., image technical parameters or acquisition conditions such as weather or lighting.

The addition of meta-annotation is supported by a Jupyter Notebook that given a set of images and a set of valid meta-annotation values allows the developer to iterate on the images and select the appropriate value, which is saved in the format used for evaluation. Figure 1 shows the interface of the meta-annotation editor.



```
is_segmentation = True

dataset_gt_param="/Users/nahimetorres/Documents/Git/raceplane/instances_test_aircraft.json'
images_path = "/Users/nahimetorres/Documents/Git/raceplane/images'

my_dataset = DatasetCoco(dataset_gt_param, path_to_detections, images_path, is_segmentation)

annotator_mc = Annotator(my_dataset)
properties = {"Is difficult": ["Yes", "No"], "Aicraft color": ["White", "Yellow", "Other"]}

annotator_mc.start_annotation(properties)
```

1 / 2158

Annotation id: 5119 - Category: Aircraft

Is difficult: [✓ Yes] No

Aircraft color: [✓ White] Yellow Other

Fig. 1. The user interface of the meta-annotation editor

The following example shows the code needed to create a meta-annotation session for two custom properties (`Is difficult` and `Airplane color`).

```
from emd_with_classes.classes import Dataset, Annotator

path_to_detections = "../detections"
dataset_gt_path="../coco.json'
```

```
5  images_path = "../images'
6
7  is_segmentation = True
8
9  my_dataset = Dataset(dataset_gt_path,
10                         path_to_detections, images_path, is_segmentation)
11 annotator_mc = Annotator(my_dataset)
12 properties = {"Is difficult": ["Yes", "No"],
               "Aicraft color": ["White", "Yellow", "Other"]}
13 annotator_mc.start_annotation()
```

The code imports the necessary modules (line 1), declares the physical paths to inputs and outputs (lines 3-5), specifies the type of task (detection or segmentation, line 7), instantiates the data set and task object (line 9), creates an instance of the annotator (line 11), declares the names and admitted values of the novel meta-annotations, and finally starts the annotation interface whereby the user can actually tag the images (line 13).

3.2 Metrics

ODIN supports both standard metrics for object detection and instance segmentation assessment, their restriction to specific values of properties expressed by the meta-annotations, and metrics focused on the analysis of false positives. The values of all metrics are reported using diagrams of multiple types, which can be visualized and saved. Developers can run all metrics, or a subset thereof, for a single class or for a set of classes.

Intersection over Union IoU is the intersection between the ground truth B_{gt} and the predicted object location B_{pr} divided by the union of the two (Eq. 1). IoU is a base measure from which all the other metrics are computed. A threshold on the IoU value of a predicted location is used to consider it as a True Positive (TP) or False Positive (FP).

$$IoU = \frac{B_{GT} \cap B_{pr}}{B_{GT} \cup B_{pr}} \tag{1}$$

Precision, Recall, PR curve and Average Precision The precision and recall metrics have the usual definition (Eq. 2). The PR curve plots the precision vs recall for all the values of the IoU threshold and can be computed for all the classes, per class, or on a subset of the classes [14].

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

Average Precision (AP) summarizes the PR curve as a single value, i.e., the precision averaged for all recall values from 0 to 1, equivalent to the Area Under the PR Curve. A common approximation (Interpolated AP) used in the

PASCAL VOC data set [4] 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 $\tilde{r}$.

An alternative definition of AP, normalized AP, is defined to cope with class unbalance [7]. Normalized AP uses a definition of precision in which the cardinality of the class is replaced by the average cardinality of all the classes in the data set. Normalized AP is the default metrics used in ODIN for multi-class diagnosis.

AP per property For a given set of classes (from 1 to all) the AP can be computed only for object proposals having a certain property value.

Property sensitivity and impact For each property value, the worst and best performing image subsets can be computed, with the maximum and minimum AP achieved. The difference between the maximum and minimum AP highlights the sensitivity of AP w.r.t. the property, while the maximum w.r.t. the overall AP provides insight on the impact of the property onto the AP.

False positive impact The per-class analysis of wrong object detection is supported, including:

- Confusion with background or with unlabeled objects (IoU lower than 0.1)
- Poor localization (L) for predictions with the correct class with an IoU lower than the threshold (default is 0.5); the indicator also counts duplicated predictions for the same ground truth object.
- Confusion with similar objects (S), when a similarly annotation is provided.
- Confusion with other objects (O).

For each identified issue the percentage of predictions that fall into that category is reported and also the absolute AP improvement by removing all false positives of each type.

3.3 Data set Visualizator

A visualization interface, realized as a Jupyter Notebook, enables the inspection of the data set and of the results of a diagnosis session. The visualization can be executed at multiple levels:

– All the images.
– All the images of a certain class.
– All the images with a certain meta-annotation value.
– All the images of a given class with a meta-annotation value.

3.4 Extending the framework

ODIN has been designed with a “plug&play” architecture, which facilitates the addition of metrics. Adding a new metrics requires extending the provided Analyzer class and implementing the wrapper method that calls the computation of the metrics.

```
class MyCustomAnalyzer(Analyzer):
    def _evaluation_metric(self, gt, proposals, match):
        # Returns the desired metric value and the std_devivation
        # To be used by the different plots

        # Parameters:
        #   gt: contains the a list of the GT objects
        #   proposals: contains a list of proposals
        #   both contain: {image, category, segemetation/bbox}
        #   match: association between gt objects and proposals

        # Returns:
        #   metric_value: the calculated value in the set
        #   std_deviation: of the metric (None: does not apply)

        #TODO: call metrics computation code...
        metric_value = #...
        std_deviation = # ...
        return metric_value, std_deviation
```

The following example shows the code needed to create an instance of the novel metrics class and execute the analysis.

```
from emd_with_classes.classes import Dataset
import MyCustomAnnotator

path_to_detections = "../detections"
dataset_gt_path="../coco.json'
images_path = "../images'
res_path = "../results"
is_segmentation = True

my_dataset = Dataset(dataset_gt_path, path_to_detections,
        images_path, is_segmentation)

my_analyzer =
MyCustomAnalyzer(my_dataset, results_path=res_path)
my_analyzer.analyze_property( "Is difficult")
```

The code imports the necessary modules (lines 1-2), declares the physical paths to inputs and outputs of the detector (lines 4-7), specifies the type of task (line 8), creates an instance of the data set (line 10), creates an instance of the analyzer (line 12) and finally starts the analysis for some of the meta-annotations previously added to the data set (line 14).

In practice, the addition of new meta-annotations and metrics to ODIN requires only a simple wrapper around the implementation of the metrics.

4 Use Case

In this section we illustrate the functionality of ODIN by applying it to an instance segmentation benchmark: the RarePlanes [16] data set of satellite images for aircraft detection. RarePlanes comprises real and synthetic images associated with object detection and instance segmentation annotations provided in MS COCO format. It contains 50,000 images and $\approx 630,000$ (14,707 real and 629,551 synthetic) annotations. It also features multiple meta-annotations: aircraft length, wing span, wing shape, wing position, Federal Aviation Administration (FAA) wing span class, propulsion, number of engines, number of vertical stabilizers, canards, aircraft type and weather acquisition conditions (Fig. 2). We performed the output analysis of the task of aircraft instance segmentation for small, medium and large civil transportation vehicles. To this end, we replicated the instance segmentation experiments of [16] using Detectron2 [17] with the configuration files associated with the data set[2]. The predictions obtained by the Detectron2.COCOEvaluator component were processed by ODIN.

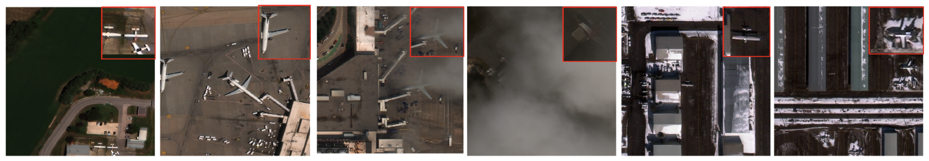


Fig. 2. Example of RarePlanes images with different weather conditions: Clear (the first two), Cloud or Haze (the middle ones), Snow (the last two)

Figure 2 shows a few examples of images acquired in different weather conditions, a meta-annotation that affects the visibility of the planes. Figure 3 reports the AP in different weather conditions for the different classes of civil airplanes. The diagram highlights that the conditions with more difficulties are snow for the small objects and cloud for mid size ones. Since in the approach of [16] data can be augmented by generating images synthetically, this finding can be taken into account when creating new examples.

[2] Models and files published at: https://github.com/aireveries/RarePlanes/tree/master/models

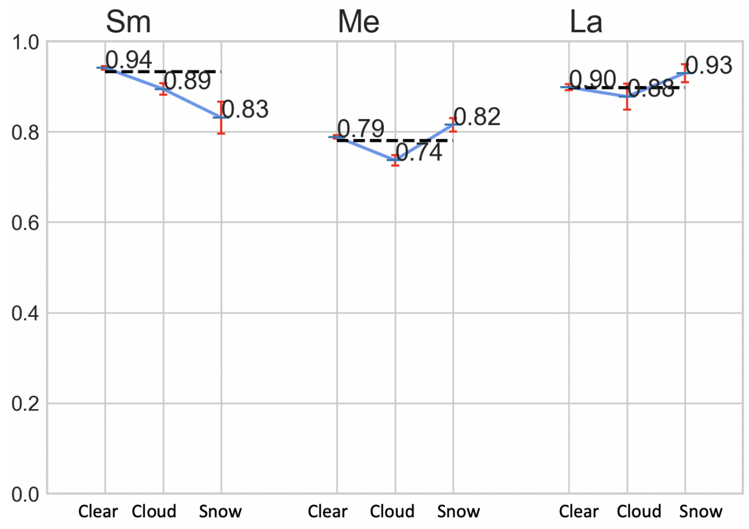


Fig. 3. AP variation under different weather conditions

Figure 4 shows how the model responds to the aspect ratio of the masks. The different shapes and sizes affect each category in a specific way. For example, the small civil aircraft class has better performances when objects are extra-wide, while the other two categories exhibit a different pattern. From a visual inspection of the examples of each class with extra wide aspect ratio (Fig. 4) we observed that while an elongated geometry is prevalent for the small civil airplanes, for medium and large ones it seems correlated to the mask orientation combined with truncation. Based on such observation, one can design measures to improve the detection of this type of images, by applying suitable data augmentation techniques.

Figure 5 (left) shows the relative impact of different objects characteristics. Propulsion, wing type and number of engines display much larger influence on the overall performance. The inspection of the propulsion property shows that it has a high imbalance across the aircraft classes (Fig. 5 right). Fig. 5 also shows that the impact of truncation is lower than that of the aspect ratio, which suggest to prioritize data augmentation by giving precedence to image rotation over the creation of artificial truncation.

Figure 6 reports the false positive distribution across classes and the distribution of the different error types explained in Section 3.2 for each class. In all cases, localization is responsible of a minority of errors, with the minimum incidence for large airplanes. This is an indicator of the good localization capabilities of the model and thus shifts the focus of improvement to the classification step. The small civil aircraft objects are the ones with more confusion with background or unlabeled objects, while the large civil aircraft ones present less errors

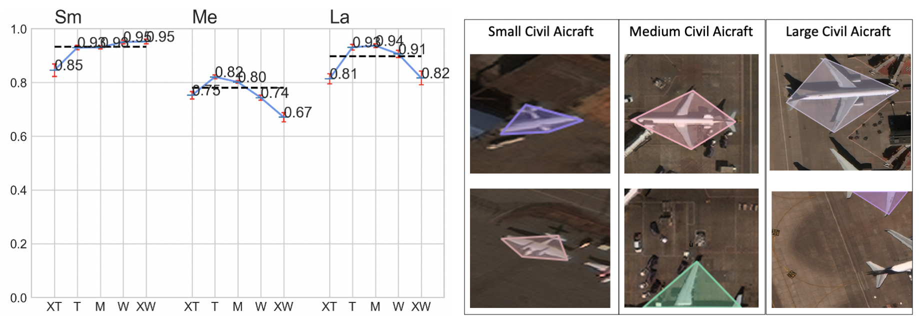


Fig. 4. AP variation under different different aspect ratios for each category and examples of small medium and large airplanes with extra wide aspect ratio

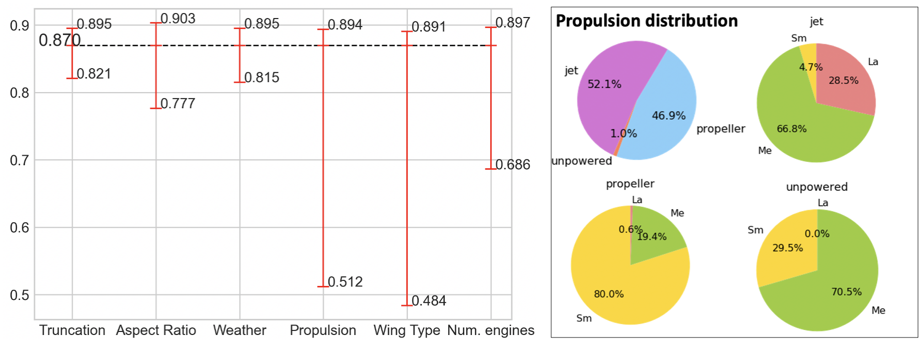


Fig. 5. Relative impact of different properties and distribution of the propulsion values (propeller, jet, unpowered) across all images and per class (small, mid, large)

of such types. To some extent, the real dimensions of the objects impact on how likely it is to confuse them with other unlabeled objects.

5 Conclusions

In this paper we have described a framework for the investigation of errors in object detection and instance segmentation applications. The framework implements the most common metrics for such tasks and enables the association of meta-annotations to the input, so that the analysis of the performance can be focused on arbitrary subsets of the input characterized by critical values of the meta-annotations. We have illustrated the output of the framework on the RarePlanes satellite image data set, which features both generic and domain-specific meta-annotations. The framework is implemented in Python and released

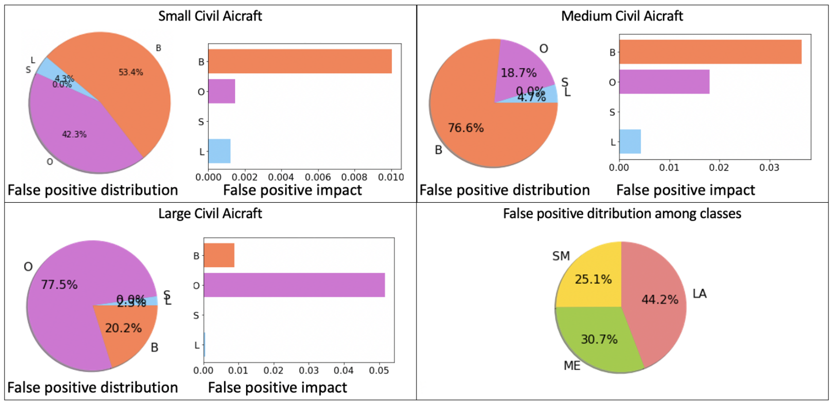


Fig. 6. Analysis of False Positives. L (light blue): location error, B: (orange) confusion with background or unlabeled objects B, O (purple) confusion with other classes

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 adding support to classification tasks and on extending the library of metrics implementations with further classes for specific applications, e.g., human pose detection. In addition, we will add support for the automatic extraction of specific types of meta-annotations, such as the geographical coordinates, date and time of image acquisition, and the orientation of the instance segmentation mask. We also plan to integrate the analysis of attention, by computing the position and extent of the CAM w.r.t. to the object bounding box or segmentation mask to support the optimization of weakly supervised models.

References

1. Abiodun, O.I., Jantan, A., Omolara, A.E., Dada, K.V., Mohamed, N.A., Arshad, H.: State-of-the-art in artificial neural network applications: A survey. Heliyon **4**(11), e00938 (2018). https://doi.org/https://doi.org/10.1016/j.heliyon.2018.e00938, `http://www.sciencedirect.com/science/article/pii/S2405844018332067`
2. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05). vol. 1, pp. 886–893. IEEE (2005)
3. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html (2012)
4. 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`

5. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition. p. 580–587. CVPR '14, IEEE Computer Society, USA (2014). https://doi.org/10.1109/CVPR.2014.81, `https://doi.org/10.1109/CVPR.2014.81`
6. Harzallah, H., Jurie, F., Schmid, C.: Combining efficient object localization and image classification. In: 2009 IEEE 12th international conference on computer vision. pp. 237–244. IEEE (2009)
7. Hoiem, D., Chodpathumwan, Y., Dai, Q.: Diagnosing error in object detectors. In: European conference on computer vision. pp. 340–353. Springer (2012)
8. Kirillov, A., He, K., Girshick, R., Rother, C., Dollar, P.: Panoptic segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
9. Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J.R.R., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Kolesnikov, A., Duerig, T., Ferrari, V.: The open images dataset V4. Int. J. Comput. Vis. **128**(7), 1956–1981 (2020). https://doi.org/10.1007/s11263-020-01316-z, `https://doi.org/10.1007/s11263-020-01316-z`
10. 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)
11. Lindeberg, T.: Scale invariant feature transform (2012)
12. Liu, L., Ouyang, W., Wang, X., Fieguth, P.W., Chen, J., Liu, X., Pietikäinen, M.: Deep learning for generic object detection: A survey. Int. J. Comput. Vis. **128**(2), 261–318 (2020). https://doi.org/10.1007/s11263-019-01247-4, `https://doi.org/10.1007/s11263-019-01247-4`
13. Padilla, R., Netto, S., da Silva, E.: A survey on performance metrics for object-detection algorithms. In: International Conference on Systems, Signals and Image Processing (IWSSIP) (07 2020). https://doi.org/10.1109/IWSSIP48289.2020
14. 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)
15. Redmon, J., Divvala, S.K., Girshick, R.B., Farhadi, A.: You only look once: Unified, real-time object detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016. pp. 779–788. IEEE Computer Society (2016). https://doi.org/10.1109/CVPR.2016.91, `https://doi.org/10.1109/CVPR.2016.91`
16. Shermeyer, J., Hossler, T., Van Etten, A., Hogan, D., Lewis, R., Kim, D.: Rareplanes: Synthetic data takes flight. arXiv preprint arXiv:2006.02963 (2020)
17. Wu, Y., Kirillov, A., Massa, F., Lo, W.Y., Girshick, R.: Detectron2. `https://github.com/facebookresearch/detectron2` (2019)
18. 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`