REBOTNIX VISIONTOOLS

Identifying object recognition errors in datasets and AI models.

Version 1.0

Gary Hilgemann , 02.02.2023

Preface

This document is addressed to users who use REBOTNIX VISIONTOOLS in their daily work. https://rebotnix.com/visiontools

The new VISIONTOOLS Model Analyzer Model is a framework for analyzing error sources in object recognition and segmentation algorithms. This plugin also exists as a plugin with graphical interface with a service connection to the REBOTNIX Low Energy Data Center or as a local installation (onPremise).

The framework is applicable across models and can handle predictions in already created models and weights without the need to know the underlying prediction system beforehand. It can be applied to pre-trained models and applied subsequently at any time by adding a new arbitrary number of GT, also called ground truth.

The best models for new data sets can be identified. Weaknesses in data sets are visualized in VISIONTOOLS.

The prediction system

VISIONTOOLS MA can be used as a substitute for a standard mAP calculation while providing a comprehensive analysis of the strengths and weaknesses of individual models.

We divide the faults into six types and use a new technique to measure the individual fault of each fault in a way that isolates its impact on overall performance per weight generated.

We show that such an analysis is crucial to draw and prove accurate, comprehensive conclusions.

Introduction

Object recognition and instance segmentation are fundamental tasks of computer vision.

Applications range from self-driving cars, robots, driving and sorting workpieces. Industrial production defect detection, recycling, smart city, traffic analysis and much more.

The fields of object recognition have made many advances in recent years. As a rule, the performance in these benchmarks is summarized with a number for all measurement methods. This number combined usually determines the mean average precision, also called mAP - mean average precision.

However, a determined mAP has several weaknesses, not least because of the complexity of the measurement itself.

It is classified within a precision-recognition curve for detections at a certain threshold for overlap with a correctly classified truth , the GroundTruth (GT). The GT is usually always averaged over all classes.

Starting with the well-known COCO data set, this approach became the standard. Mostly a mAP is determined over 10 IOU threshold values within intervals of 0.5 to 0.95 to generate a final MAP. So as an example a threshold value from 0.5 = 5% to 0.95 i.e. 95%.

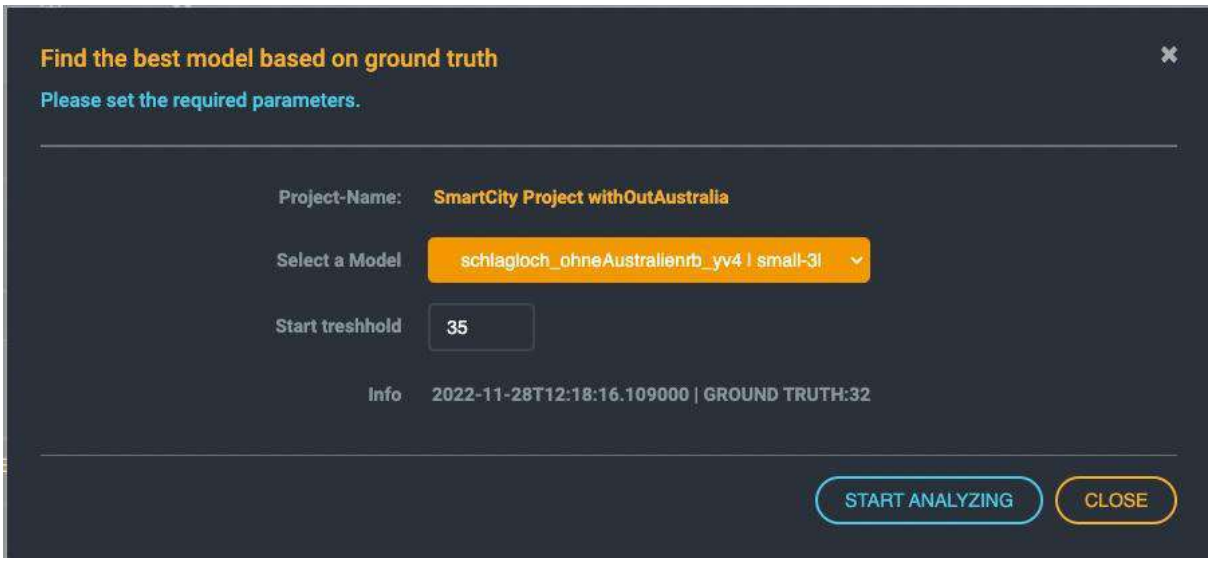
The complexity of this metric presents a particular challenge when we want to analyze the errors in our object detectors, since the error types are interrelated, making it very difficult to estimate how much each error type affects the mAP.

Furthermore, by optimizing for mAP alone, we may unintentionally underestimate the underestimate the relative importance of error types, which can vary depending on the application.

For example, when detecting an object in a robot, correct classification is arguably more important initially than box localization.

In contrast, precise localization can be critical for robotic gripping, where even minor mislocalization can lead to faulty mechanical operation.

REBOTNIX VISIONTOOLS Model Analyzer, starting with version 1.6.0, provides a set of tools to identify detection and segmentation errors to solve these problems.



The analyses include the following sources of error.

1) The grouping of multiple defect types so that comparisons can be made at a glance.

2) Fully isolate each type of error so that there are no confounding variables that can influence a conclusion.

3) Do not require dataset specific annotations to allow comparisons between different datasets.

4) Include all predictions of a model, since considering since considering only a subset hides information.

5) Enable a finer analysis so that the sources of error can be sources of error can be isolated.

Why we need a new analytics tool.

There are many papers analyzing object and instance segmentation faults, but few provide a useful summary of all faults in a model, and none have all the desirable properties we listed above.

The COCO evaluation toolkit attempts to meet the requirement by plotting errors in terms of their impact on the precision-recognition curve. This allows all detections to be used at once, since the precision-recall curve implicitly weights each error based on its confidence.

However, the COCO Toolkit generates graphs with each of the precision-recall curves, which takes a considerable amount of time and thus makes it difficult to present a complex simple comparison.

Perhaps the most critical problem, however, is that the COCO eval toolkit computes errors progressively, which can lead to incorrect conclusions. Finally, the toolkit requires manual annotations that are available for COCO but not necessarily for other datasets as yet unknown. For example, if from a factory A records are ported to another production site factory B. But factory B has minimal different dimensions than in the first location of factory A.

In parallel, we attempt to find an upper bound for AP on these datasets and addresses specific issues with the COCO toolkit. However, this final error reporting is still based on the same progressive scheme of the COCO toolkit.

The Model Analyzer addresses all 5 goals in its analysis, providing a compact but detailed summary of object detection and instance segmentation errors. Each type of error can be represented as a single meaningful number, making it compact enough to fit in a clear table.

We also weight defects based on their impact on overall performance, carefully avoiding the confounding factors present in a MAP.

In addition, the approach is modular enough that the same set of faults can be used for a finer analysis. The end result is a compact, expressive error set that can be used for all existing models as well as all existing models that are trained with any neural network, whether that model is based on CNN-based object detection or on instance segmentation.

With this tool we would like to justify design decisions quantitatively to the chosen model with a meaningful report.

Tools

In object recognition and instance segmentation, one metric is primarily used to assess performance: mean average precision (mAP). While mAP summarizes the performance of a model succinctly performance of a model in a number. It is difficult to separate mAP when there is an error in object detection and instance segmentation.

A False Positive can be a double detection, a misclassification, a mislocalization, a background confusion, or even both a misclassification and a mislocalization. Similarly, a false negative can be a completely missed ground truth, or the potentially correct prediction could simply be misclassified or mislocalized.

These types of errors can have very different effects on mAP, which makes it difficult to diagnose problems with a model based on mAP alone. There is a simple way to determine the significance of a particular error to the overall mAP by simply fixing that error and observing the resulting change in mAP.

We define defects such that fixing all defects still results in 100 mAP, but we weight each error individually, based on the performance of the original model. This has the advantage that confidence and false negatives are included in the calculation, while the magnitudes of the individual error types remain comparable.

Calculation of mAP

Before defining error types, we focus on defining mAP to understand what can cause degradation. To calculate the mAP, we first obtain a list of predictions for each image by the detector.

Each ground truth in the image is then matched with at most one detection. To be considered a positive match, the detection must be of the same class as the ground truth and have an IoU overlap greater than a certain threshold which we set to 0.5 unless otherwise specified.

When multiple detections are considered, the detection with the largest overlap is selected as true-positive, while all others are considered false-positive.

Once each detection matches a ground truth (true positive) or not (false positive), all detections from each image in the dataset are collected and sorted by descending confidence. Then, the cumulative precision and cumulative recall over all detections are calculated as follows:

$$P_c = \frac{TP_c}{TP_c + FP_c} \qquad R_c = \frac{TP_c}{N_{GT}}$$

For all detections with a confidence ≥ c, Pc is the precision, Rc is the recall, T Pc the number of true positives, and FPC the number of false positives. NGT denotes the number of GT examples in the current class.

Define error types

When examining this calculation, there are 3 ways in which our detector can affect mAP: Outputting false positives during the matching step, not outputting true positives (i.e. false negatives), and false calibration (i.e. outputting a higher confidence for a false positive than for a true positive)..

Main Error Types To create a meaningful distribution of errors that captures the components of mAP, we rank all false positives and false negatives in the model of 6 types (see Fig. 1). Note that for some error types (classification and localization).

We use IoU max to denote the maximum IOU overlap of a false positive with a ground truth of the given category. The IoU threshold for the foreground is denoted as tf and the threshold for the background is denoted as tb,
which are set to 0.5 and 0.1 , respectively, unless otherwise specified.

1. **Classification Error**
    a. **Cls**
        GT of the wrong class (i.e. correctly located but incorrectly classified)

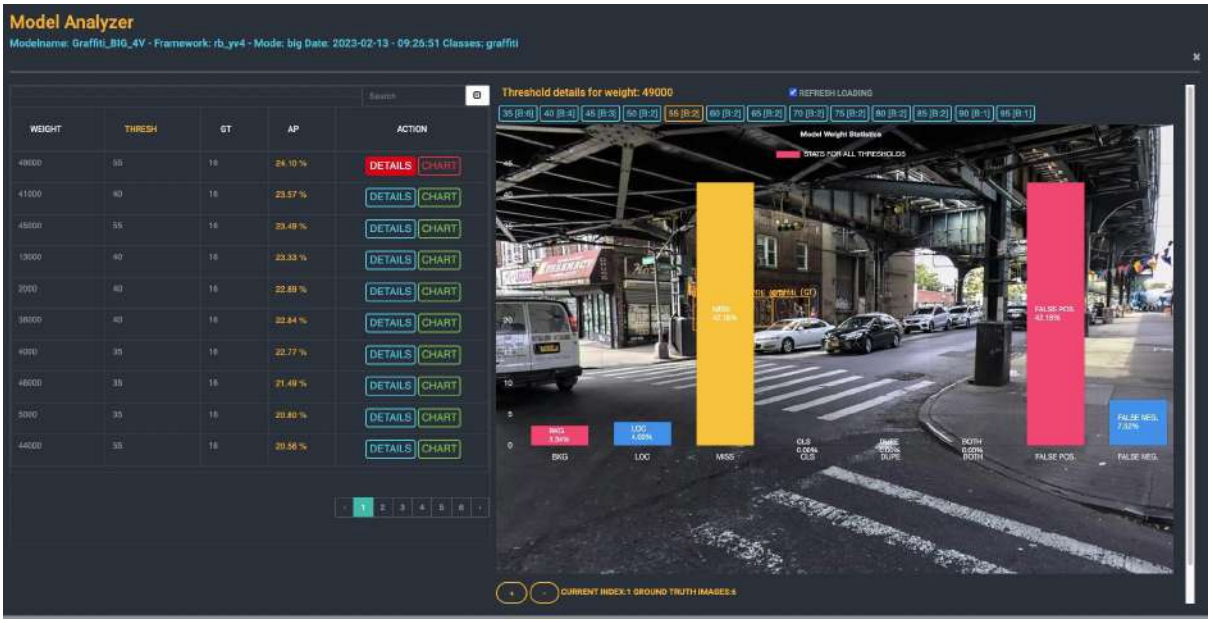
2. **Localization Error**
    a. **Loc** Class (i.e., correctly classified but incorrectly localized)

3. **Both Cls and Loc Error**
    a. tb ≤ IoUmax ≤ tf for GT of the wrong class
    (d. h., misclassified and incorrectly localized)

4. **Duplicate Detection Error**
    a. **Dupe value**
    GT is in the correct class, but another detection with a higher score already matches this GT (i.e., it would be correct if not for a detection with a higher score).

5. **Background Error**
    a. **Bkg**
    b. The background was detected as foreground. There was a detectin where there is no groundtruth.

6. **Missed GT Error**
    a. **Miss**
    All unrecognized ground truth (false negatives) not already identified by classification or localization errors.



Ideally, we would like to have a comprehensive number that indicates how each type of error affects the overall performance of the model. In other words, for each error type, we would like to ask how much that error category affects the performance of my model.

To answer this question, we can consider what the model's performance would be if it had not made this error, and how mAP would have changed as a result. To this end, for each error we need to define a corresponding target that fixes that error.

For example, if a target describe how to turn some false positives into, we could call that after applying the target the calculated AP as APo and then compare it to the value AP to determine the impact of the target (and corresponding error) on performance.

$$\Delta AP_o = AP_o - AP$$

We know that we have covered all the errors in the model when applying them all together gives 100 mAP. In other words, given is the formula O = {o1, . . . , on}:

$$AP_{o_1,\ldots,o_n} = 100 \qquad AP + \Delta AP_{o_1,\ldots,o_n} = 100$$

Referring to the definition of AP, this means that to satisfy Equation 3, the targets used together resolve all false positives and false negatives.

1. Classification

Corrects the class of the discovery (making it a true positive).

2.Localization

Set the localization of the detection to the localization of the GT (making it a true positive). If duplicate detection would occur this way, suppress the detection with the lower score.

3. Cls and Loc

Since we cannot be sure which GT the detector was trying to hit with, we suppress the false positive detection.

4. Duplicate Detection

Suppress duplicate detection.

5.Background

Suppress the detection of the background.

6. Missed GT Target:

Decreases the number of GT (NGT) in the MAP calculation by the number of missed ground truth. This has the effect of stretching the precision-recall curve over a higher recall, essentially acting as if the detector would be just as precise with the missing GT. The alternative to this would be to add new detections, but it is not clear what the score for these new detections should be so that it does not introduces nuisance variables. We discuss this choice further in the Appendix.

Other types of errors

While the previously defined types fully account for all errors in the model.

model, the definition of errors does not clearly distinguish between false positive and negative errors (since cls, loc, and missed errors can all capture false negative errors).

There are cases where a clear separation would be useful, so for these cases we define two separate error types by the target that would deal with each of these errors.

1. false-positive: suppression of all false-positive detections.
2. false-negative: sets NGT to the number of true positive detections.

Both targets together add up to 100 mAP, like the previous 6 targets, but they bind the errors in a different way.

2.3 Limitations in the stepwise calculation of errors

Note that we are careful to compute the errors one at a time (i.e., each AP starts with the AP where no errors were specified).

It will probably never be possible to correct all the localization errors and then start working on the classification errors - there will always be a certain number of errors left in each category. For these reasons, we avoid calculating the errors step by step.

Analysis

We demonstrate the generality and usefulness of our analysis toolbox through detailed analyses for various object detection and instance segmentation models, as well as for various data and annotation sets. We also compare errors based on general ground truth properties, such as object size, and find a number of useful insights. To further explain complicated error cases, we provide detailed analysis for specific error types. All analysis methods used in this paper are available in VISIONTOOLS.

Models

We choose different object detection and instance segmenters based on their ubiquity and/or unique qualities, which allow us to explore trade-offs between different approaches and gain different insights.

We use M-R-CNN as a baseline, as many other approaches are built on the standard R-CNN framework.

We additionally include three such models: Hybrid Task Cascades (HTC), TridentNet, and Mask Scoring R-CNN.

We include HTC due to its strong performance, as it is the winner of the 2018 COCO Challenge.

Finally, we conclude MS R-CNN as a method that specifically focuses on fixing calibration-based errors. Unlike the two-stage R-CNN approaches, we also include three single-stage approaches, which are suitable for representing real-time models.

Datasets

We present our central model overarching analysis on MS-COCO, a widely used and active benchmark. Furthermore, we aim to show the power of our toolbox by including three additional datasets.

Validation of design decisions

The authors of each new object detector or instance segmenter make design decisions that they claim affect the performance of their model in different ways. While the goal is almost always to increase overall MAP, the question is whether the intuitive justification for a design choice is warranted with increased performance.

Comparison of object attributes for fine analysis.

To compare performance for different object attributes such as scale or aspect ratio, the typical approach is to compute mAP on a subset of the detections and ground truth that have the specified attributes.

Missing ground truths, a major problem with densely annotated datasets such as cityscapes. The key challenge with cityscapes is the presence of many small objects, which are notoriously difficult to detect with modern algorithms. On the other hand, the challenge is the lack of ground truth because the detections cannot be properly classified.

This is probably due to the enormous number of classes and the large data set. The trend that densely annotated datasets prone to missing ground truth are also evident in the false-positive and false-negative.

Unavoidable errors

We have found that a large portion of background and localization errors can simply be due to setting an incorrect or uncommented ground truth. Upon closer examination of the most important errors we find that many of the most certain errors are in fact this conclusion due to an oversight in the way ignored detections are handled.

Conclusion

In this document, we have identified meaningful defect types and a way to link and evaluate these defect types with overall performance.

We then created the resulting framework in VISIONTOOLS to independently evaluate decisions within the analysis so that the performance of object attributes can be compared. Likewise, we show the properties of different datasets in their prevalence and their erroneous reason annotations.

VISIONTOOLS model analyzers include methods for isolating and improving recognition errors. Visual interpretability in design decisions of a neural algorithm can be used to provide clearer evidence of the strengths and weaknesses of the model being trained.

References

COCO Analysis Toolkit: http://cocodataset.org/#detection-eval, accessed: 2020-03-01 2, 4, 7
2. Bolya, D., Zhou, C., Xiao, F., Lee, Y.J.: Yolact++: Better real-time instance segmentation. arXiv:1912.06218 (2019) 9, 10, 11
3. Bolya, D., Zhou, C., Xiao, F., Lee, Y.J.: Yolact: real-time instance segmentation. In: ICCV (2019) 9, 11
4. Borji, A., Iranmanesh, S.M.: Empirical upper-bound in object detection and more. arXiv:1911.12451 (2019) 2, 3, 7, 14
5. Chen, K., Pang, J., Wang, J., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Shi, J., Ouyang, W., et al.: Hybrid task cascade for instance segmentation. In: CVPR (2019) 9, 10
6. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: CVPR (2016) 1, 9, 12
7. Divvala, S.K., Hoiem, D., Hays, J.H., Efros, A.A., Hebert, M.: An empirical study of context in object detection. In: CVPR (2009) 2 8. Dollar, P., Wojek, C., Schiele, B., Perona, P.: Pedestrian detection: A benchmark. In: CVPR ´ (2009) 1 9. Dong, H., Yang, G., Liu, F., Mo, Y., Guo, Y.: Automatic brain tumor detection and segmentation using u-net based fully convolutional networks. In: MIUA (2017) 1