machine learning & knowledge extraction

MDPI

Article

Surrogate Object Detection Explainer (SODEx) with YOLOv4 and LIME

Jonas Herskind Sejr *, Peter Schneider-Kamp and Naeem Ayoub

Department of Mathematics & Computer Science, University of Southern Denmark, 5230 Odense, Denmark; petersk@imada.sdu.dk (P.S.-K.); nomi@imada.sdu.dk (N.A.)
* Correspondence: sejr@imada.sdu.dk; Tel.: +45-52342918

Abstract: Due to impressive performance, deep neural networks for object detection in images have become a prevalent choice. Given the complexity of the neural network models used, users of these algorithms are typically given no hint as to how the objects were found. It remains, for example, unclear whether an object is detected based on what it looks like or based on the context in which it is located. We have developed an algorithm, Surrogate Object Detection Explainer (SODEx), that can explain any object detection algorithm using any classification explainer. We evaluate SODEx qualitatively and quantitatively by detecting objects in the COCO dataset with YOLOv4 and explaining these detections with LIME. This empirical evaluation does not only demonstrate the value of explainable object detection, it also provides valuable insights into how YOLOv4 detects objects.

Keywords: object detection; Explainable Artificial Intelligence; YOLO; LIME

Citation: Sejr, J.H.; Schneider-Kamp, P.; Ayoub, N. Surrogate Object Detection Explainer (SODEx) with YOLOv4 and LIME. *Mach. Learn. Knowl. Extr.* **2021**, *3*, 662–671. https://doi.org/10.3390/make3030033

Academic Editors: Jochen Garcke and Ribana Roscher

Received: 29 June 2021
Accepted: 3 August 2021
Published: 6 August 2021

1. Introduction

YOLO (you only look once) [1,2] is the most popular object detection algorithm, and the results from YOLO are quite impressive. After training on a body of labeled images (labeled with bounding boxes and classes), YOLO precisely detects objects of any kind in milliseconds and can, thus, even be used for real-time object detection in videos. YOLO, thereby, processes input similar to human sight, and it is one of the tools that really provide an impression that Artificial Intelligence (AI) is already among us.

YOLO is not based on a clear theoretical model for object detection but represents the cumulative result of many incremental versions tested and tuned in practical use. In other words, YOLO is a fine-tuned aggregation of methods from object detection that has been demonstrated to work well on real images.

YOLO uses the whole image for object detection, and a user of YOLO is not given any hint as to how a detection was made, for example, what pixels or areas of the image were used to detect a given object.

Through experiments, we have seen YOLO detect very blurry objects. Initially, we found it hard to grasp how YOLO was able to do this and whether something was wrong with the data or the trained model. We were also interested in using YOLOs class score (an indication of how much YOLO believes in the detection) to make a statement about the detected object. Specifically, we wanted to detect broken objects. Assuming broken objects would not follow the visual patterns of normal objects, we assumed they would get a lower class score, in other words, a kind of object outlier detection. In both situations, we found it challenging not to know which pixels YOLO uses to make the detections. We did not know whether YOLO uses the overall structure of the detected object, a single recognized pattern within the object, the context of the object (pixels outside the object), or a complex combination of all of them. These challenges motivated us to look into explaining YOLO.

Since YOLO is a moving target that is continuously changed and improved, we wanted to develop an explainer for YOLO that is independent of YOLOs internals (i.e., a model-agnostic, black box, object detection explainer [3]). Our algorithm, Surrogate Object

Detection Explainer (SODEx), defines a surrogate classifier model from a single object detection in YOLO and explains an object detection using a black-box classifier explainer such as LIME [4]. For each detected object, we explain the prediction by determining and highlighting the regions that contributed the most and the least to the class score.

In this article, we explain YOLOv4 [2] with LIME, but our method is general and independent of either. The SODEx algorithm can explain any object detection algorithm that provides a bounding box and a score using any model-agnostic classifier explainer.

Our experiments on YOLOv4 and COCO dataset ([5]) demonstrate how explanations from SODEx provide valuable insight into both YOLO and the detected objects.

In summary, our contributions are:

- To the best of our knowledge, the first instance of an object detection explainer.
- An abstract algorithm that explains virtually any object detector with any classification explainer.
- Insights on YOLOs object detection exemplified with explanations for images from the COCO dataset.

2. Background

2.1. YOLO

Deep Neural Networks (DNNs) are popular for developing object detection algorithms. Object detection algorithms extract important information to solve computer vision problems such as object classification, localization, and recognition [6]. In the last two decades, many deep neural network models for object detection have evolved, improving the intelligence of machine vision systems. Examples of such are faster R-CNN [7], RetinaNet [8], Single Shot MultiBox Detector (SSD) [9], and You Only Look Once (YOLO) [1]. Among the aforementioned state-of-the-art object detection algorithms, YOLO stands out regarding its ability to run on low-power devices, its real-time performance, and its accuracy [10].

At its core, YOLO is based on a Convolution Neural Network (CNN). This layer predicts a fixed number of boxes and class probabilities. YOLO divides the image into $S \times S$ grid cells. Each grid cell is responsible for detecting a fixed number of objects with their center inside the grid cell. The output of the core layer of YOLO is a large number of boxes and class probabilities, each in the form:

$$[p_c, b_x, b_y, b_h, b_w, c_1, c_2, ...]$$

where b_x and b_y are the coordinates of the center of a detected object, b_h and b_w are the height and width of the bounding box, c_1, c_2... are the class probabilities, and p_c is the object probability, YOLOs estimated probability that there is an object in the given position. The core neural network is trained on images where objects are labeled with bounding boxes, object probability 1, and class probability 0 for all but for the correct class, which is labeled by 1. The loss function balances the bounding box precision, the object probability precision, and the class probability precision.

Having the neural network detect a fixed number of objects, on the one hand, eases training and allows different parts of the network to specialize in different shapes, etc. On the other hand, it results in many detected objects, some overlapping and some with very low object probability.

After predicting a large number of objects, YOLO first filters away objects with a class c_{score} score below some pre-set threshold,

$$c_{score} = p_c \cdot c_x$$

where c_x is the class probability for a given class. c_{score} indicates how likely it is that there is an object of the specific class in the given position.

In the third part of the algorithm, denoted non-maximum suppression, YOLO first removes low probability objects. Then, to remove overlapping objects that might detect the

same real object in the image, YOLO keeps the objects with the highest object probabilities p_c and removes objects that overlap with an intersection of union (iou) above a pre-set threshold. iou is defined as follows:

$$iou(I_1, I_2) = \frac{|I_1 \cap I_2|}{|I_1 \cup I_2|}$$

where I_1 and I_2 are images as collections of pixels.

Finally, YOLO returns a list of bounding boxes, each with the predicted class and the corresponding class score.

The above is a description of YOLOs overall components and structure, but YOLO comes in multiple versions with variations, particularly in the core neural network.

The YOLO algorithm was first introduced by [1], in which they unify the region classification proposals into a single neural network for class probabilities and bounding boxes prediction. To increase robustness of the algorithm, [11] proposed YOLO9000 (also called YOLOv2). They added features such as a high-resolution classifier, fine gradients, dimension clustering, and added batch normalization for faster learning and detection. Ref. [12] introduced a darknet-53 based YOLOv3 model, which increases accuracy and real-time performance of YOLO-DNN (The Convolutional Neural Network in the core of YOLO). Finally, and recently, [2] added extra features into the YOLO network and introduced a CSPdarknet-53 based YOLOv4, which further improves the speed and accuracy of the network. In this article, we use and explain YOLOv4.

2.2. LIME

Models in supervised learning have become increasingly complex. Consequently, many users find it hard to explain the predictions of a model, i.e., to understand why a model predicts as it does. Especially deep learning models have been criticized for their lack of interpretability.

This has led to a new type of method that seeks to explain previously uninterpretable models [13]. The primary value propositions of explanations are: Increased trust from users, ensuring ethical and fair decision making, additional data insight, insight into model transferability, and model debugging.

A plethora of methods exists that will explain uninterpretable models. Some of these explanations take their outset in the specific model, while others are model agnostic. Some explain the whole model while others explain locally, e.g., single predictions.

A popular type of explanations are local model-agnostic explanations. We use Local Interpretable Model Explanation (LIME) [4], which is a local model-agnostic method for explaining single predictions.

One major advantage of model-agnostic explanations is that the users do not need to understand the model being explained. This makes it possible to explain very complex models for which it may be impossible to interpret the internals. The advantage with local models is that the model being explained may globally be very complex but locally (i.e., around a single prediction) is simpler and easier to explain. A model may, for instance, locally be assumed to behave approximately linear. This assumption is at the core of LIME.

LIME explains a single prediction with a linear surrogate model. The surrogate model is trained with a version of LASSO [14] to enforce sparsity, as sparsity is associated with higher interpretability. By weighting each sample in training with an exponential kernel on the L2 (or cosine distance for text) distance to the object under explanation, the surrogate model is localized.

The linear model coefficients are finally interpreted as feature importance, either in favor of (positive) or against (negative) the predicted class.

When used for explaining images, LIME first segments the image into superpixels using a segmentation algorithm of choice, e.g., Quickshift [15]. LIME then defines a set of superpixel features, each running from 0, meaning the superpixel is greyed out to 1, meaning the superpixel is untouched. An explanation, therefore, is a weighting of the

superpixels and can, e.g., be visualized by showing top and bottom superpixels in green and red, cf. Figure 1.

(**a**) Top-Bottom explanatory superpixels. (**b**) Superpixel explanation weights.

Figure 1. ($c_{score} = 0.91, i_{ratio} = 3.914, w_{diff} = 0.00554$) YOLOv4 has detected a person in a street. With a class score of 0.91 YOLOv4 is relatively confident that there is a person in this position. YOLOv4 has primarily used pixels inside the detected box in the detection ($i_{ratio} = 3.914$), and pixels inside the detected box are on average more positive towards the object than pixels outside the box ($w_{diff} = 0.00554$). Looking at the explanations we see that YOLOv4 primarily used the upper body, the visible hand, and the visible leg to detect the person. The visible lower arm and a region close to the leg can put YOLOv4 in doubt about the detection.

3. Method

We propose an abstract algorithm, Surrogate Object Detection Explainer (SODEx), which can explain any object detection algorithms with any classifier explainer. In this article, we instantiate this abstract algorithm to explain YOLOv4 with LIME.

LIME explains image classification as described in Section 2.2, but YOLOv4 detects objects in an image and not image classes (Section 2.1). To explain a detected object in YOLOv4 (the object under explanation), we introduce a surrogate binary classifier for the object under explanation (Algorithm 1).

Algorithm 1 Surrogate Binary Classifier (SBC)

function $SBC_{oue}(I)$ ▷ Object under explanation (oue)
  $objects \leftarrow$ YOLO.FIND_OBJECTS(I)
  if *objects* is empty **then**
    return 0
  end if
  $iou_{max} \leftarrow -1$
  $c_{score} \leftarrow 0$
  for all $object \in objects$ **do**
    $iou \leftarrow$ IOU$(object.bbox, oue.bbox)$
    if $iou > IOU_{MIN} \wedge iou > iou_{max} \wedge object.class = oue.class$ **then**
      $iou_{max} \leftarrow iou$
      $c_{score} \leftarrow object.score$
    end if
  end for
  return c_{score}
end function

3.1. Surrogate Binary Classifier (SBC)

SBC takes in an image and derives a score that indicating how likely it is the image contains the object under explanation (c_{score}).

With YOLOv4, the surrogate detects every object in the image. For each detected object the surrogate classifier calculates the Intersection Over Union (iou) (see Section 2.1).

If at least one object was detected that has an *iou* with the object under explanation above some threshold IOU_{MIN} (set to 0.4 in our experiments), the object with the highest *iou* is assumed to be the object under explanation, and the class score from YOLOv4 is returned as the class score c_{score}.

3.2. Surrogate Object Detection Explainer (SODEx)

Explaining a detected object in YOLOv4 is now just constructing and explaining the surrogate binary classifier for the object under explanation, Algorithm 2.

Algorithm 2 Surrogate Object Detection Explainer (SODEx)

```
function SODEx(obj)
    seg_alg ← QUICKSHIFT                    ▷ or another segmentation algorithm
    classifier ← SBC_obj
    explanation ← LIME.EXPLAIN(classifier, seg_alg, obj)
end function
```

3.3. What Are We Explaining?

Since SODEx explains the surrogate model and not directly the object detection with YOLOv4, the natural question is, what are we really explaining? When we explain a classifier with LIME, LIME implicitly defines a measure for how much "influence" each feature (in this case pixels) locally has on the class probability. Since the SBC retrieves a class score for the object under explanation from YOLOv4, explaining the SBC is similar to explaining the class score from YOLOv4. In other words, SODEx explains YOLOv4's class score, which indicates how much YOLOv4 considers it likely that the box contains an object of the predicted class. We believe this is well aligned with how users will perceive an explanation of an object.

An inherent limitation is that the SBC cannot ensure that the returned object probability stems from YOLOv4 detecting the same object: YOLOv4 might, for some variations of the image, not even detect the object, in which case the SBC returns probability 0. This adds to the uncertainty of the explanation but does not change what SODEx explains.

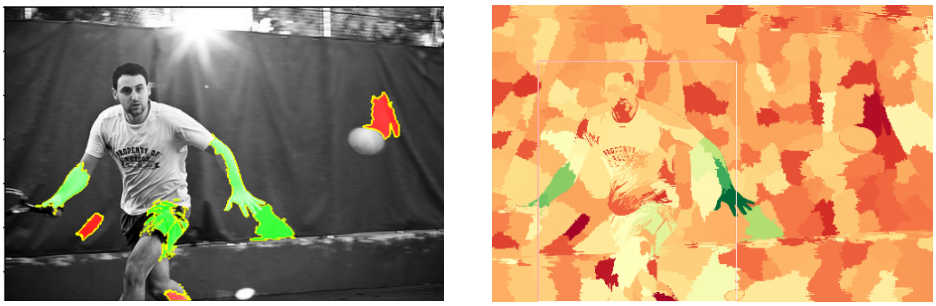
4. Qualitative Evaluation of Explanations

To demonstrate how explanations from SODEx work, we have trained YOLOv4 to recognize persons in the COCO dataset and explained with SODEx how YOLOv4 does the detection.

For our experiments, we filtered all images from the COCO 2017 training and validation dataset referenced from the person-with-keypoints annotation files. We further kept only those images that contained at least one real (i.e., not annotated with "iscrowd") person of reasonable size (here, with an area between $\frac{1}{3}$ and $\frac{2}{3}$ of the total image area). This filtering was implemented after the first qualitative evaluations of explanations in order to ensure a certain quality standard for the images showing persons, as well as for the stability and quality of the explanations obtained.

We fine-tuned YOLOv4 with the filtered training images to recognize the person class. Then we applied SODEx on the filtered validation images to explain all detected objects of the person class. We use the fine-tuned YOLOv4 model for object detection and LIME for explanations, with Quickshift [15] as the image segmentation algorithm to be used by LIME. We have experimented with different parameter settings of YOLOv4, LIME and Quickshift, but to prevent overfitting the results presented in this paper are the result of the default settings in the implementations we have used (Our implementation of SODEx is available at https://github.com/JonasSejr/SODEx, accessed on 4 August 2021). SODEx itself only has one parameter, IOU_{MIN}, which is set to 0.4 in our experiments.

Figures 1–4 show four images explained with SODEx, LIME, and Quickshift.

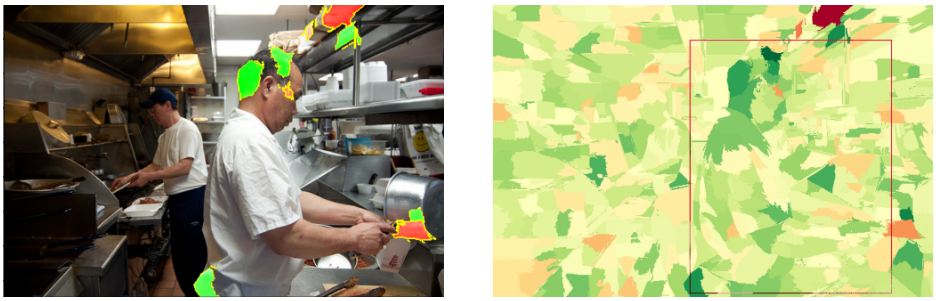


(**a**) Top-Bottom explanatory superpixels. (**b**) Superpixel explanation weights.

Figure 2. ($c_{score} = 0.97, i_{ratio} = 1.501, w_{diff} = 0.00131$) YOLOv4 has detected a tennis player and is very confident about the detection. A hand, arms, and a part of the leg are the most influential regions when YOLOv4 detects this object. Three regions in the background have a negative effect on the detection. This could either be the result of random noise or YOLOv4 use of the context of the detected object.

(**a**) Top-Bottom explanatory superpixels. (**b**) Superpixel explanation weights.

Figure 3. ($c_{score} = 0.92, i_{ratio} = 1.92, w_{diff} = 0.00044$) The images show the detection of the leftmost person. The legs are the most influential when YOLOv4 detects the object. Again, the head has less importance than the lower body and arms. A region that is part of the person behind the detected person has the highest negative effect on the detection. It makes sense that YOLOv4 is confused by the person behind, or, in other words, removing the person would make YOLOv4 more confident.



(**a**) Top-Bottom explanatory superpixels. (**b**) Superpixel explanation weights.

Figure 4. ($c_{score} = 0.98, i_{ratio} = 0.99, w_{diff} = 0.00052$) YOLOv4 is very confident in the detection of the rightmost person. Contrary to the other detections, the head has the highest weight. Arms are less important even though they are very visible in the image.

The images to the left show the original image with the most important superpixels highlighted. The superpixels contributing positively are highlighted in green, while those

that contribute negatively are highlighted in red. In other words, removing (i.e., greying out) the green superpixels will make YOLOv4 less sure in its detection of the object under explanation, while removing the red pixels will make YOLOv4 more sure in its detection.

The images to the right are heat maps visualizing the weight of each pixel in the explanation from LIME. The heat map also shows the detected bounding box. The in–out importance ratios (i_{ratio}) and in–out weight differences w_{diff} (Section 5), as well as class scores (c_{score}) are given in the caption.

The general impression when qualitatively evaluating the explanations for the top-ranked (w.r.t. class score) 200 objects is that most objects are detected based on positively contributing superpixels inside the detected bounding boxes and, typically, on the people detected. This is the case in Figures 1–4 with the exception of a few superpixels. In many cases, some superpixels close to the person are also relevant. These, however, typically affect the detection negatively. This is, for example, the case in Figure 3, where another person behind the detected person affects the detection negatively, i.e., it confuses YOLOv4 that there is another person in such proximity. We see the same issue in Figure 1, where a superpixel close to the leg of the person contributes negatively to the detection.

In general, we can conclude from the qualitative review of the explanations that extremities (arms and legs) seem to play an important role when YOLOv4 detects people. This could be because these are large recognizable structures that most images in the training set feature. The head and face seem to be used less frequently, even though it is, e.g., the most important factor in Figure 4.

In quite a few images, we also have observed superpixels with negative contributions in regions we cannot relate to the detected object. This hints at a certain amount of context-dependency of the object detection.

Before evaluating the explanations of the images, we expected that YOLOv4 would not only use superpixels on the detected object but also superpixels outside such that, e.g., if there were a bicycle, the probability to detect a person might increase. This does not seem to be the case.

5. Quantitative Evaluation of Explanations

With the impression from the qualitative evaluation in mind, we set out to look at the general tendencies when YOLOv4 detects objects. To do this, we defined two statistics: the in–out importance ratio (i_{ratio}) and the in–out weight difference w_{diff}. The in–out importance ratio (i_{ratio}) denotes how important (in either negative or positive direction) pixels inside the bounding box are (the internal causes) compared to the pixels outside (the context causes).

The in–out importance ratio defines the ratio between the sum of absolute weights of the pixels inside the box divided by the sum of absolute weights of the pixels outside, i.e., how much more important is the bounding box versus the context in determining if there is an object.

$$i_{ratio} = \frac{\sum_{p \in box} |weight(p)|}{|box|} \Big/ \frac{\sum_{p \notin box} |weight(p)|}{|box^c|}$$

The in–out weight difference is defined as the difference in the average weight inside minus outside and tells us if the bounding box is determined from inbox pixels or from out-of-box pixels.

$$w_{diff} = \frac{\sum_{p \in box} weight(p)}{|box|} - \frac{\sum_{p \notin box} weight(p)}{|box^c|}$$

Figure 5 visualizes the distribution of the in–out importance ratio in our test dataset. It is clear that, in this dataset, YOLOv4 primarily uses the inbox pixels in detecting the objects. None of the bounding boxes detected in the test set have higher average importance of pixels outside the box. The image with the highest score is the image presented in Figure 1 with nearly four times more weight of the pixel inside the detected box.

Furthermore, when we look at the real values of the weights (Figure 6), the average weight inside the bounding boxes is higher than outside, i.e., not only are the pixels inside the bounding box more important, but they also in general, affect YOLOs trust in its prediction positively compared to out of box pixels. This emphasizes that for this dataset, YOLO primarily detects base on the looks of the object. In a few cases, though, the context is more important. This can be attributed to the fact that superpixels sometimes cross the boundaries of the bounding box and, in some cases, especially for low class scores, we have seen somewhat random explanations.

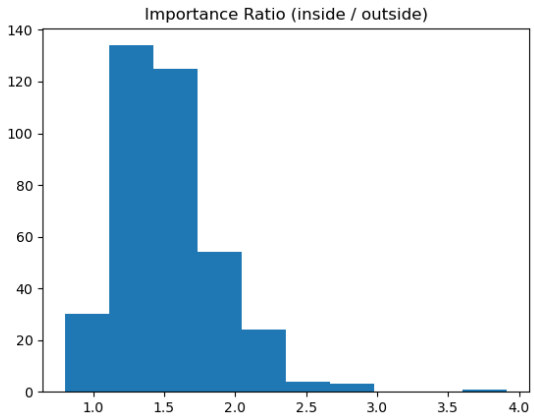


Figure 5. in–out importance ratio measures the importance (absolute weight) inside the detected bounding box relative to the importance of pixels outside the detected box. The figure show that for every detected object in the test set, the pixels inside the bounding box contribute more to the decision.

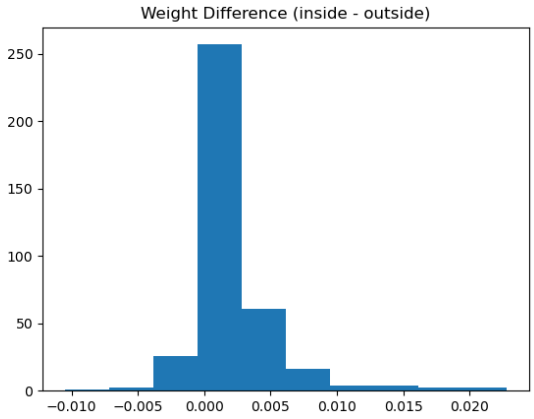


Figure 6. in–out weight difference is the average explanatory weight inside the detected bounding box minus the average explanatory weight of pixels outside the detected bounding box. The figure shows that pixels inside the detected box in general contribute more in favor of there being an object of the specific type in the specific position.

6. Conclusions and Outlook

We have developed an algorithm, SODEx, that can provide black box explanations to object detection algorithms. We do not have a way to evaluate explanations quantitatively, but our experiments with SODEx, YOLOv4, LIME, and QuickShift on selected images from the COCO dataset show explanations that correlate with our intuition. The most important pixels are typically legs, arms, and heads. Quantitatively, we have seen that when we look at absolute pixel importance and pixel weight on the given dataset, pixels inside the detected box are more important in detecting the object and, in general, are more in favor of the detected object than pixels outside the box.

We chose to explain object detection based on the YOLOv4s class score, which is a number that indicates how much YOLOv4 believes the object is located in the specific position and belongs to the predicted class. Using SODEx for other object detectors, the user will have to determine a score that will indicate the confidence of the detection.

Our experiments show that the explanations using YOLOv4s class score seem reasonable. However, other aspects could have been used, e.g., the deviation of the box could be taken into account in the surrogate classifier. The advantage of using a single statistic variable is that it is simple and concise, and there is no need for parameter tuning. Future research could look into explaining other aspects of object detectors and YOLOv4.

Our experiments use LIME and Quickshift for explaining, and, therefore, the explanation's semantics has to be interpreted relative to how LIME sees an explanation and how Quickshift divides the image. Other explainers will provide different results. When we explain YOLOv4 trained on the COCO dataset, it is a philosophical question whether we explain YOLOv4 or the COCO dataset. Therefore, it would be interesting to see research that includes experiments with SODEx using a different explainer, a different object detector or a different dataset.

As an example, it is conceivable that, while YOLOv4 was found to rely mostly on pixels inside the bounding box for the COCO dataset, it might be that YOLOv4 on another dataset would pay more attention to the context of the objects.

Our contribution initiates and enables such types of experiments, as it provides an easy approach to explaining object detectors. We look forward to seeing future research in explainable object detection.

Author Contributions: Conceptualization, J.H.S. and N.A.; methodology, J.H.S., P.S.-K. and N.A.; software, J.H.S., P.S.-K. and N.A.; validation, J.H.S., P.S.-K. and N.A.; formal analysis, J.H.S., P.S.-K. and N.A.; investigation, J.H.S., P.S.-K. and N.A.; resources, J.H.S., P.S.-K. and N.A.; data curation, J.H.S., P.S.-K. and N.A.; writing—original draft preparation, J.H.S., N.A.; writing—review and editing, P.S.-K.; visualization, J.H.S. and N.A.; supervision, P.S.-K.; project administration, J.H.S., P.S.-K. and N.A.; funding acquisition, P.S.-K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Drones4Energy.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: https://cocodataset.org/ (accessed on 1 August 2021).

Acknowledgments: The authors would like to acknowledge Department of Mathematics & Computer Science, University of Southern Denmark.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
2. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. Yolov4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934.
3. Barredo Arrieta, A.; Diaz Rodriguez, N.; Del Ser, J.; Bennetot, A.; Tabik, S.; Barbado González, A.; Garcia, S.; Gil-Lopez, S.; Molina, D.; Benjamins, V.R.; et al. Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI. *Inf. Fusion* **2019**, *58*. [CrossRef]
4. Ribeiro, M.T.; Singh, S.; Guestrin, C. Why Should I Trust You? Explaining the Predictions of Any Classifier. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, WA, USA, 13–17 August 2016.
5. Lin, T.; Maire, M.; Belongie, S.J.; Bourdev, L.D.; Girshick, R.B.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision*; Springer: Cham, Switzerland, 2014; pp. 740–755.
6. Ayoub, N.; Gao, Z.; Chen, B.; Jian, M. A synthetic fusion rule for salient region detection under the framework of DS-evidence theory. *Symmetry* **2018**, *10*, 183. [CrossRef]
7. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *39*, 1137–1149. [CrossRef] [PubMed]

8. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 7 August 2017; pp. 2980–2988.
9. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*; Springer: Cham, Switzerland, 2016; pp. 21–37.
10. Ayoub, N.; Schneider-Kamp, P. Real-Time On-Board Deep Learning Fault Detection for Autonomous UAV Inspections. *Electronics* **2021**, *10*, 1091. [CrossRef]
11. Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 7263–7271.
12. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
13. Burkart, N.; Huber, M.F. A Survey on the Explainability of Supervised Machine Learning. *J. Artif. Intell. Res.* **2021**, *70*, 245–317. [CrossRef]
14. Tibshirani, R. Regression Shrinkage and Selection Via the Lasso. *J. R. Stat. Soc. Ser. B* **1996**, *58*, 267–288. [CrossRef]
15. Vedaldi, A.; Soatto, S. Quick Shift and Kernel Methods for Mode Seeking. In *Computer Vision—ECCV 2008*; Forsyth, D., Torr, P., Zisserman, A., Eds.; Springer: Berlin/Heidelberg, Germany, 2008.