HALLUCINATION IN OBJECT DETECTION — A STUDY IN VISUAL PART VERIFICATION

Osman Semih Kayhan[⋆] *Bart Vredebregt*[§] *Jan C. van Gemert*[⋆§]

[⋆]Computer Vision Lab, Delft University of Technology and [§]Aiir Innovations

ABSTRACT

We show that object detectors can hallucinate and detect missing objects; potentially even accurately localized at their expected, but non-existing, position. This is particularly problematic for applications that rely on *visual part verification*: detecting if an object part is present or absent. We show how popular object detectors hallucinate objects in a visual part verification task and introduce the first visual part verification dataset: DelftBikes[1], which has 10,000 bike photographs, with 22 densely annotated parts per image, where some parts may be missing. We explicitly annotated an extra object state label for each part to reflect if a part is missing or intact. We propose to evaluate visual part verification by relying on recall and compare popular object detectors on DelftBikes.

Index Terms— Visual part verification, object detection

1. INTRODUCTION

Automatically localizing and detecting an object in an image is one of the most important applications of computer vision. It is therefore paramount to be aware that deep object detectors can hallucinate non-existent objects, and they may even detect those missing objects at their expected location in the image, see Fig. 1. Detecting non-existing objects is particularly detrimental to applications of automatic *visual part verification* or *visual verification*: determining the presence or absence of an object. Examples of visual verification include infrastructure verification in map making, missing instrument detection after surgery, part inspections in machine manufacturing etc. This paper shows how popular deep detectors hallucinate objects in a case study on a novel, specifically created visual object part verification dataset: DelftBikes.

Visual verification as automatic visual inspection is typically used for manufacturing systems with applications such as checking pharmaceutical blister package [1], components on PCBs [2, 3], solder joint [4], parts of railway tracks [5], rail bolts [6], aeronautic components [7, 8], objects [9], and parts under motion [10]. In this paper, we do not focus on a particular application. Instead, we evaluate generic deep object detectors which potentially can be used in several visual inspection applications.

[1]https://github.com/oskyhn/DelftBikes

Fig. 1. Hallucination examples on DelftBikes for Faster RCNN [11], RetinaNet [12] and YOLOv3 [13]. Faster RCNN and RetinaNet detect the front wheel and YOLOv3 predicts the saddle with a high IoU score. Deep object detectors may detect non-existent objects at their expected locations.

There are important differences between visual verification and object detection. An object detector should not detect the same object multiple times. For visual verification, however, the goal is to determine if an object is present or absent, and thus having an existing object detected multiple times is not a problem, as long as the object is detected at least once. This makes recall more important than precision. Moreover, there are differences in how much costs a mistake has. The cost for an existing object that is not detected (false negative) is that a human needs to check the detection. The cost for a missing object that is falsely hallucinated as being present (false positive) is that this object is a wrongly judged as intact and thus may cause accidents in road infrastructure, or may cause incomplete objects to be sent to a customer. The costs for hallucinating missing objects is higher than missing an existing object. These aspects motivate us to not use the evaluation measure of object detection. Object detectors are typically evaluated with mean Average Precision (mAP) and because detections of non-existent objects at lower confidence levels does not significantly impact mAP, the problem of object hallucination has largely been ignored. Here, we propose to evaluate visual verification not with precision but with a cost-weighted variant of recall.

Fig. 2. Example images of our DelftBikes visual verification dataset. Each image has a single bike with 22 bounding box annotated parts. The similar pose, orientation and position can be misleading for context-sensitive detectors as often one or two parts are missing (the saddle in (a), the wheels in (e) etc.).

Object hallucination by deep detectors can be causes by sensitivity to the absolute position in the image [14, 15] while also affected by scene context [16, 17, 18, 19, 20]. Here, we focus on the visual verification task, its evaluation measure, a novel dataset, and a comparison of popular existing detectors. Investigating context is future work.

Existing object detection datasets such as PASCAL VOC [21], MS-COCO[22], Imagenet-det [23], and Open Image [24] have no annotated object parts. Pascal-Parts [25] and GoCaRD [26] include part labels, yet lack information if a part is missing and where, as is required to evaluate visual verification. Thus, we collected a novel visual verification dataset: DelftBikes where we explicitly annotate all part locations and part states as missing, intact, damaged, or occluded.

We have the following contributions:

1. We demonstrate hallucination in object detection for 3 popular object detectors.
2. A dataset of 10k images with 22 densely annotated parts specifically collected and labeled for visual verification.
3. An evaluation criteria for visual verification.

2. DELFTBIKES VISUAL VERIFICATION DATASET

DelftBikes (See Fig. 2) has 10,000 bike images annotated with bounding box locations of 22 different parts where each part is in one of four possible states:

intact: The part is clearly evident and does not indicate any sign of damage. All the images in Fig. 2 have an intact steer.

damaged: The part is broken or has some missing parts. In Fig. 2-g, the front part of the saddle is damaged.

absent: The part is entirely missing and is not occluded. Fig. 2-e has missing front and back wheels.

occluded: The part is partially occluded because of an external object or completely invisible. The saddle in Fig. 2-b is covered with a plastic bag.

The distribution of part states is approximately similar for training and testing set, see Fig. 3. The part state distribution shows 60.5% intact, 19.5% absent, 14% occluded, and 6% damaged. The *front pedal*, *dress guard*, *chain* and *back light* have respectively the highest number of intact, absent, occluded and damaged part states. Note that even if a part is absent or occluded, we still annotate its most likely bounding box location. DelftBikes contains positional and contextual biases. In Fig. 4 where we plot an ellipse for each part in the dataset in terms of their mean position, height and width. It is possible to recognize the shape of a bike, which indicates that there are strong part-to-part position and contextual relations. Its those biases that learning systems may falsely exploit and cause detector hallucinations.

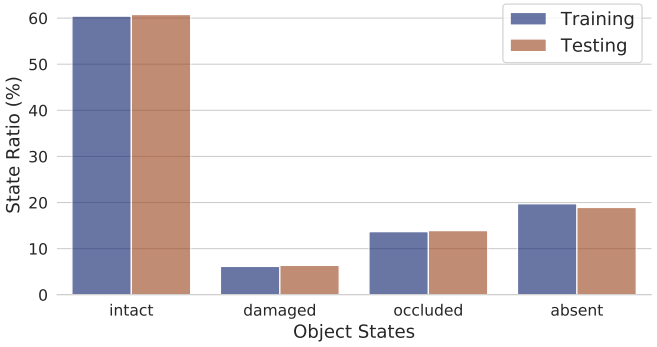

Fig. 3. The distribution of part states for train and test sets in DelftBikes. The ratio of part states are roughly similar for train and test sets. The intact parts have the highest ratio by around 60%. Approximately 20% of parts in the dataset are absent. The damaged and occluded parts constitute 20%.

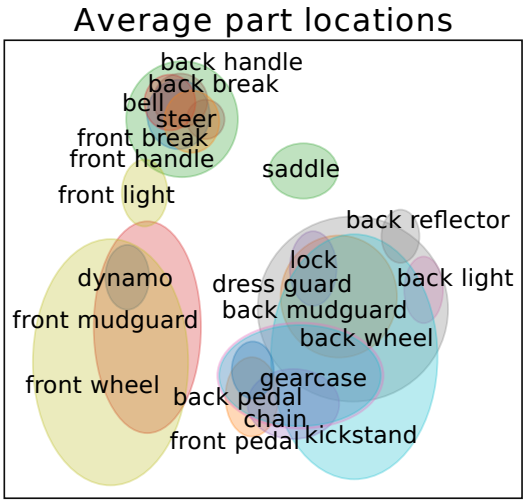

Fig. 4. Averaging position and size for all 22 parts in DelftBikes resembles a bicycle, illustrating the prior in absolute position and the contextual part relations.

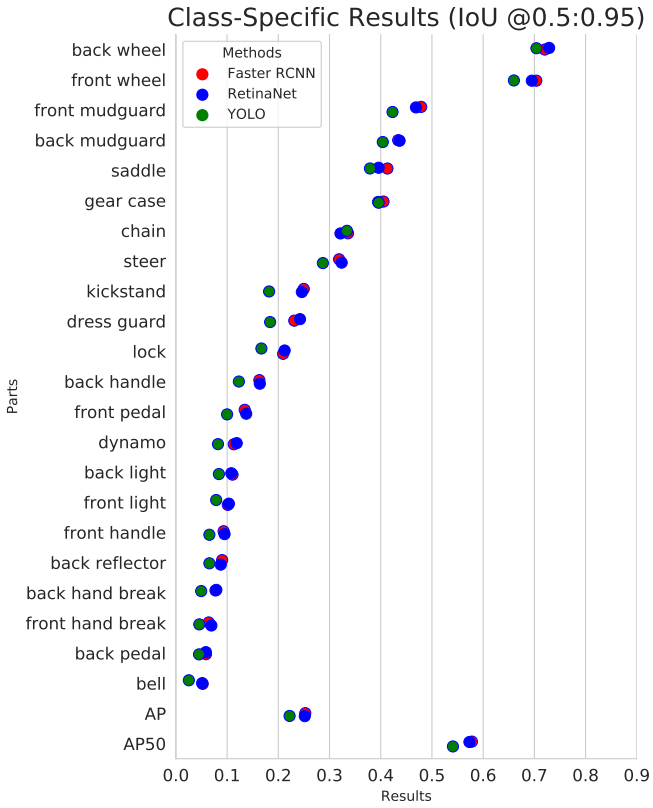

Fig. 5. Object detection results on DelftBikes. Results per category and overall performance. Notice that half of the detections are below 20% AP score. In most of the cases, Faster RCNN and RetinaNet perform similarly and YOLOv3 is behind them.

3. EXPERIMENTS ON DELFTBIKES

The dataset is randomly split in 8k for training and 2k for testing. We use a COCO pretrained models of Faster RCNN [11] and RetinaNet [12]. Both networks have a Resnet-50 [27] backbone architecture with FPN. The networks are finetuned with DelftBikes for 10 epochs using SGD with a initial learning rate of 0.005. The YOLOv3 [13] architecture is trained from scratch for 200 epochs using SGD with an initial learning rate of 0.01. Other hyperparameters are set to their defaults. We group the four part states in two categories for visual verification: (i) *missing* parts consist of absent and occluded states and (ii) *present* parts include intact and damaged states. During training, only parts with *present* states are used.

Detection. We first evaluate traditional object detection using AP. For object detection, the *missing* parts are not used during training nor testing. In Fig. 5, we show results for an IoU of 0.5:0.95 for the 3 detectors. For most of the classes, Faster RCNN and RetinaNet obtain approximately a similar result and YOLOv3 is a bit behind. *Front wheel* and *back wheel* are large and well detected. The small parts like *bell* and *dynamo* have under 12% AP score because they are small parts and often not present. The other parts are below 50% AP, where half of the parts have less than 20% AP, which makes DelftBikes already a challenging and thus interesting object detection dataset.

Recall of *missing* parts. Here, we analyze the hallucination failure of the detectors by evaluating how many non-existing parts they detect in an image. We calculate the IoU score for each detected *missing* part on the test set. We threshold these false detections in terms of their IoU scores to evaluate if the missing parts are still approximately localized. We define the recall score which is the ratio between the number of detected missing part at a given IoU threshold and the total number of missing parts.We show recall for varying IoU threshold for each method in Fig. 6. For a reasonable IoU of 0.5, RetinaNet and YOLOv3 detect approximately 20% of missing parts and Faster RCNN 14%. Without looking at position, (IoU=0), RetinaNet and YOLOv3 detect as much as almost 80% of *missing* parts. Interestingly, Faster RCNN, with similar mAP object detection score as RetinaNet, detects only 32% of missing parts. For Faster RCNN, the most hallucinated part with 14% is *gear case*. For YOLOv3, a missing *dynamo* is most detected and RetinaNet hallucinates most about the *dress guard*.

Evaluating visual verification. For visual verification, we want high recall of *present* parts and low recall of *missing* parts where detecting the same object multiple times does not matter. Besides, wrongly detected *missing* parts (false positives) cost more than not detected *present* parts (false negatives). Thus, our F_{vv} evaluation score is based on recall and inspired by the F_β score [28] so we can weight detection mis-

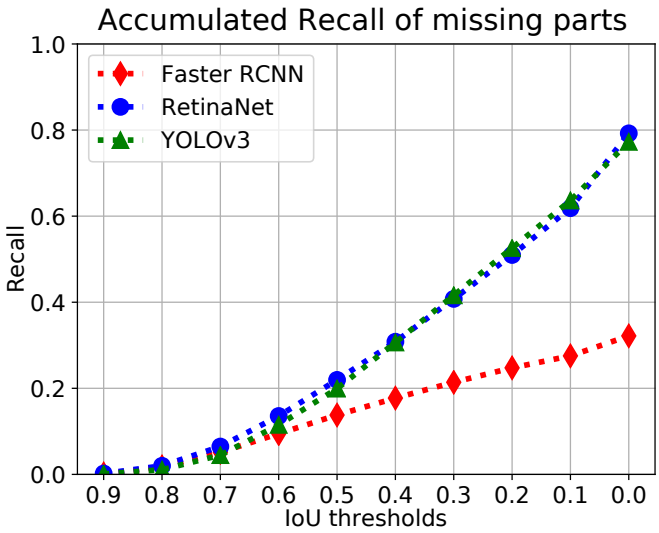

Fig. 6. Recall of *missing* parts on DelftBikes for varying Intersection over Union (IoU). We annotated likely position of missing parts, and the recall of such missing parts should be as low as possible. All methods wrongly detect missing parts at approximately their expected location, as in Fig. 1.

takes differently as

$$F_{vv} = \frac{(1+\beta^2)R^P(1-R^M)}{\beta^2(1-R^M)+R^P}. \quad (1)$$

R^P is the *present* recall and R^M the *missing* recall calculated at a certain IoU threshold. The β parameter allows to weight the detection mistakes, where we set the β parameter to 0.1 so that detections of *missing* parts are 10x more costly than not detected *present* parts.

Visual verification results. Visual verification performance is estimated by using the recall of present and missing parts. We have two setups for visual verification calculation: with and without localization. *Visual verification with localization:* the *present* recall has an IoU threshold of 0.5, where the *missing* recall is less relying on position and we set its IoU threshold to 0.1. *Visual verification without localization:* we set all IoU thresholds to 0. This, in addition, allows us to evaluate a full-image multi-class multi-label classification (MCML) approach. An Imagenet pretrained ResNet-50 architecture is fine-tuned with BCE with logits loss and SGD with an initial learning rate of 0.05 for 15 epochs. After every 5 epoch, the learning rate is reduced by a factor of 10. The network obtains 91% of recall for present parts and 32% of recall for missing parts.

Results are shown in Table 1. For the *with localization* results, Faster RCNN outperforms RetinaNet and YOLO in terms of lower recall of *missing* parts by 28% and a higher F_{vv} score by 72%. RetinaNet and YOLOv3 detects more than 60% of *missing* parts and achieve only 38% and 36% of F_{vv} score respectively. In Fig. 5, the AP scores of Faster RCNN and RetinaNet are quite similar, yet the F_{vv} performance of Faster RCNN is almost 2 times higher than RetinaNet. RetinaNet has 7% more intact recall score than YOLOv3, however, the difference for F_{vv} is only 2%. For the *without localization* results, when the *present* and *missing* IoU thresholds are set to 0, all the methods obtain more than 90% *present* recall. Interestingly, the MCML method, which only needs full image class labels, outperforms RetinaNet and YOLOv3 detectors and performs similar to Faster RCNN.

Method	T^P	T^M	R^P	R^M	F_{vv}
With localization					
Faster RCNN	0.5	0.1	0.83	**0.28**	**0.72**
RetinaNet	0.5	0.1	**0.90**	0.62	0.38
YOLOv3	0.5	0.1	0.83	0.64	0.36
Without localization					
Faster RCNN	0.0	0.0	0.92	**0.32**	**0.68**
RetinaNet	0.0	0.0	**0.99**	0.79	0.21
YOLOv3	0.0	0.0	0.95	0.77	0.23
MCML	0.0	0.0	0.91	0.32	0.68

Table 1. Visual verification of Faster RCNN, RetinaNet, YOLOv3 and MCML for different present (T^P) and missing (T^M) IoU thresholds on DelftBikes. (top) When (T^P, T^M) equals to $(0.5, 0.1)$: RetinaNet has highest recall for *present* parts. Faster RCNN detects the fewest missing parts and has best F_{vv} score. (bottom) When localization is discarded: MCML method outperforms RetinaNet and YOLOv3 and results similarly Faster RCNN in F_{vv} score.

4. DISCUSSION AND CONCLUSION

We show hallucinating object detectors: Detectors can detect objects that are not in the image even with a high IoU score. We show hallucination in the context of a visual part verification task. We introduce DelftBikes, a novel visual verification dataset, with object class, bounding box and state labels. We evaluate visual verification by recall, where the cost of falsely detected missing parts is more expensive than a missing present part. For object detection, Faster RCNN and RetinaNet has similar AP score, however, Faster RCNN is the better for visual verification.

One limitation of our work is that the human annotations for the non-existing parts are partly guesswork. Taking this into account, this makes it even more surprising that detectors predict with such a high IoU score.

5. REFERENCES

[1] R. G. Rosandich, "Automated visual inspection systems," in *Intelligent Visual Inspection*. 1997. 1

[2] Hugo C Garcia and J Rene Villalobos, "Automated refinement of automated visual inspection algorithms," *IEEE T-ASE*, 2009. 1

[3] Dusan Koniar, Libor Hargas, Anna Simonova, Miroslav Hrianka, and Zuzana Loncova, "Virtual instrumentation for visual inspection in mechatronic applications," *Procedia Engineering*, 2014. 1

[4] Tae-Hyeon Kim, Tai-Hoon Cho, Young Shik Moon, and Sung Han Park, "Visual inspection system for the classification of solder joints," *Pattern Recognition*, 1999. 1

[5] Esther Resendiz, John M Hart, and Narendra Ahuja, "Automated visual inspection of railroad tracks," *IEEE transactions on ITS*, 2013. 1

[6] F. Marino, A. Distante, P. L. Mazzeo, and E. Stella, "A real-time visual inspection system for railway maintenance: automatic hexagonal-headed bolts detection," *IEEE Transactions on Systems, Man, and Cybernetics*, 2007. 1

[7] H. Ben Abdallah, I. Jovančević, J.-J. Orteu, and L. Brèthes, "Automatic inspection of aeronautical mechanical assemblies by matching the 3d cad model and real 2d images," *Journal of Imaging*, 2019. 1

[8] Marco San Biagio, Carlos Beltran-Gonzalez, Salvatore Giunta, Alessio Del Bue, and Vittorio Murino, "Automatic inspection of aeronautic components," *Machine Vision and Applications*, 2017. 1

[9] Albert-Jan Baerveldt, "A vision system for object verification and localization based on local features," *Robotics and Autonomous Systems*, 2001. 1

[10] SK Sim, Patrick SK Chua, ML Tay, and Yun Gao, "Recognition of features of parts subjected to motion using artmap incorporated in a flexible vibratory bowl feeder system," *AI EDAM*, 2006. 1

[11] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *NIPS*, 2015. 1, 3

[12] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár, "Focal loss for dense object detection," in *ICCV*, 2017. 1, 3

[13] Ali Farhadi and Joseph Redmon, "Yolov3: An incremental improvement," *CVPR*, 2018. 1, 3

[14] Marco Manfredi and Yu Wang, "Shift equivariance in object detection," in *ECCV workshop*, 2020. 2

[15] O.S. Kayhan and J.C. van Gemert, "On translation invariance in CNNs: Convolutional layers can exploit absolute spatial location," in *CVPR*, 2020. 2

[16] Ehud Barnea and Ohad Ben-Shahar, "Exploring the bounds of the utility of context for object detection," in *CVPR*, 2019. 2

[17] Spyros Gidaris and Nikos Komodakis, "Object detection via a multi-region and semantic segmentation-aware cnn model," in *ICCV*, 2015. 2

[18] Yong Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen, "Structure inference net: Object detection using scene-level context and instance-level relationships," *CVPR*, 2018. 2

[19] Yousong Zhu, Chaoyang Zhao, Jinqiao Wang, Xu Zhao, Yi Wu, and Hanqing Lu, "Couplenet: Coupling global structure with local parts for object detection," in *ICCV*, 2017. 2

[20] Krishna K. Singh, Dhruv Mahajan, Kristen Grauman, Yong Jae Lee, Matt Feiszli, and Deepti Ghadiyaram, "Don't judge an object by its context: Learning to overcome contextual bias," *arXiv:2001.03152*, 2020. 2

[21] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL VOC2012," . 2

[22] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick, "Microsoft coco: Common objects in context," in *ECCV*, 2014. 2

[23] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al., "Imagenet large scale visual recognition challenge," *IJCV*, 2015. 2

[24] Alina Kuznetsova, Hassan Rom, Neil Alldrin, J, Jasper R. R. Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Tom Duerig, and Vittorio Ferrari, "The open images dataset V4: unified image classification, object detection, and visual relationship detection at scale," *CoRR*, vol. abs/1811.00982, 2018. 2

[25] Xianjie Chen, Roozbeh Mottaghi, Xiaobai Liu, Sanja Fidler, Raquel Urtasun, and Alan Yuille, "Detect what you can: Detecting and representing objects using holistic models and body parts," in *CVPR*, 2014. 2

[26] Lukas Stappen, Xinchen Du, et al., "Go-card – generic, optical car part recognition and detection: Collection, insights, and applications," 2020. 2

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *CVPR*, 2016. 3

[28] N. Chinchor, "Muc-4 evaluation metrics," in *MUC4*. 1992, Association for Computational Linguistics. 3