All You Need is RAW: Defending Against Adversarial Attacks with Camera Image Pipelines (Supplementary Material)

Yuxuan Zhang, Bo Dong, Felix Heide

Princeton University

In this supplemental document, we present additional information and results supporting the findings made in the main manuscript. We first formally define the attack methods used for validation in our main manuscript, see Section 1. In Section 2, additional details on the S and F operators are provided. Next, in Section 3, we provide additional qualitative comparisons of the proposed defense method compared to state-of-the-art approaches. In Section 4, we investigate the proposed defense method in the presence of a super-white box attack. Section 5 discusses how to modify the proposed method to be robust to the BPDA attack. Finally, in Section 6, we provide additional details on the baseline defense methods used in our experiments.

1 Attack Algorithms

In this section, we provide formal definitions for the attack algorithms used to validate the proposed defense method.

1. **FSGM:** The FSGM [3] is an attack method that, for each pixel, determines in which direction the pixel intensity should be moved according to the gradient of the loss function. Mathematically, the adversarial perturbation δ is defined by FGSM as

$$\delta = \epsilon \cdot \text{sign}(\nabla_x L(x, y)),$$

   where ϵ is the maximum perturbation allowed for an attack, which is commonly sufficiently small for undetectable; x and y are a benign image and the corresponding ground truth label, respectively.

2. **BIM:** The BIM [8] attack is an enhanced version of FGSM. Instead of taking one single step of size ϵ, it iteratively searches for an optimal perturbation with multiple smaller steps α. Then, the obtained perturbation is clipped by the predefined ϵ. Formally, BIM is defined as

$$x_i = \text{clip}_{\epsilon,x}(x_{i-1} + \alpha \cdot \text{sign}(\nabla_x L(x, y))),$$

where $x_0 = x$. The number of iteration used is a hyperparameter, and more iterations typically lead to a stronger attack.

3. **PGD:** Similar to the BIM attack, the PGD [11] attack interatively optimizes the adversarial examples generated. With the initial $x_0 = x$, the method iteratively find x_i by

$$x^i = \Pi_{x+S}(x^{i-1} + \alpha \cdot \text{sign}(\nabla_x L(f_\theta(x^{i-1}), y))), \tag{1}$$

where Π_{x+S} denotes projecting perturbations into the set S. We refer the reader to [11] for additional details. The scalar α is the step size used in each iteration, and ϵ denotes the maximum perturbation allowed.

4. **DeepFool:** Similar to BIM and PGD, the DeepFool [13] attack generates adversarial perturbations iteratively, intending to obtain the perturbations with minimal distortion. Specifically, it finds the closest decision boundary by multiple linearizations of the classifier and then generates the perturbations based on the decision boundary.

5. **C&W:** The C&W [1] attack is an optimization-based attack method, which generates the adversarial perturbations by solving the following problem

$$\begin{aligned} \min \quad & ||\delta||_p + c \cdot f(x+\delta) \\ s.t. \quad & x + \delta \in [0,1]^n, \end{aligned} \tag{2}$$

where f is a objective function that is designed to mislead the example x to be misclassified; $||\cdot||_p$ denotes l_p norm; and c is a constant, estimated by binary search.

6. **NewtonFool:** NewtonFool [6] attack is essentially a variant of the DeepFool attack. In contrast to DeepFool generating perturbations based on linearization approximation, the NewtonFool is based on nonlinear constraints, allowing for a significantly faster generation of adversarial patterns. Hence, with the same time budget, more iterations can be performed, resulting in stronger attacks.

7. **DAG:** The DAG [18] attack is designed to attack semantic segmentation and object detection models. Essentially, it is similar to optimization-based classification attacks, generating adversarial perturbation by solving the following problem

$$\forall n, \arg\max_c \{f_c(\mathbf{X} + \mathbf{r}, t_n)\} \neq l_n, \tag{3}$$

where X is an image which contains N recognition targets $\mathcal{T} = \{t_1, t_2, \ldots, t_N\}$; $\mathcal{L} = \{l_1, l_2, \ldots, l_n\}$ is the ground-truth class labels of $\mathcal{T}$, *i.e.*, l_n is the ground-truth label of t_n; $\mathbf{f}(\mathbf{X}, t_n) \in \mathbb{R}^C$ denote the classification score vector on the n-th recognition target; $\mathbf{r}$ denotes an adversarial perturbation to be estimated.
The form of $\mathcal{T}$ is defined based on a specific task. For the image classification task, $\mathcal{T}$ only contains one element, *i.e.*, the entire image. $\mathcal{T}$ becomes all pixels

for semantic segmentation tasks, while it becomes all proposals for object detection tasks.

Intuitively, the goal of the Eq. 3 is to make the predictions of all targets incorrect by estimating an adversarial perturbation $\mathbf{r}$. In doing so, adversarial labels $\mathcal{L}' = \{l'_1, l'_2, \ldots, l'_n\} \quad \forall i, l_i \neq l'_i$ are required, which are generated by randomly sampling from other incorrect classes. Under this setting, the loss function covering all targets can be written as

$$L(\mathbf{X}, \mathcal{T}, \mathcal{L}, \mathcal{L}') = \sum_{n=1}^{N} \left[f_{l_n}(\mathbf{X}, t_n) - f_{l'_n}(\mathbf{X}, t_n) \right]. \tag{4}$$

Minimizing L can be achieved by making every target an incorrect prediction, *i.e.*, suppressing the confidence of the original correct class $f_{l_n}(\mathbf{X} + \mathrm{r}, t_n)$, while increasing the confidence of the desired (adversarial) incorrect class $f_{l'_n}(\mathbf{X} + \mathrm{r}, t_n))$.

2 Additional Details on S and F Operator

The S operator offers the functionality of a conventional (hardware) ISP pipeline using a sequence of cascaded sub-modules. In particular, the proposed S operator consists of the following components: Bayer Demosaicking, color balancing, white balancing, contrast enhancement, Gamma adjustment, and colorspace conversion sub-modules. We provide the implementation details of each sub-module below:

1. **Bayer Demosaicking**: As discussed in the main manuscript, a color filter array (CFA) sits on a matrix of small potential wells. When light passes through the CFA layer, a mosaic pattern of the three stimulus RGB colors is generated, called RAW image. A Bayer pattern mosaic is the most commonly used one, alternating R-G-G-B 2×2 superpixels. To reconstruct trichromatic intensity values from a RAW image, a demosaicking algorithm is required. In our implementation, we use the DDFaPD demosaicking algorithm [12] offered by the Python colour-demosaicing library.
2. **Color Balance** The purpose of color balancing is to recover the color characteristics of the original scene. We achieve the color balancing by using the SimpleColorBalance [10] algorithm offered by the OpenCV library.
3. **White Balance** We further adjust the color temperatures of RGB images with a white balance method to make the color look more natural. We use a white balancing method provided by the OpenCV library, mainly based on the White Patch algorithm and Gray World algorithm [16].
4. **Contrast Enhancement** Contrast plays a critical role in separating the dark and bright areas of an image. An improvement in the contrast increases this separation, making objects more distinguishable. In our pipeline, the

Contrast Limited Adaptive Histogram Equalization(CLAHE) algorithm [15] is leveraged for contrast enhancement.

5. **Gamma Adjustment**: In this step, we use a non-linear Gamma correction function to adjust the image luminance (*i.e.*, brightness level). In our pipeline, we use the default gamma correction algorithm [5] offered by the OpenCV library.
6. **Colorspace Conversion**: Pixel values are converted to a specific colorspace, e.g., sRGB, before compression storage or further processing, e.g., JPEG.

The F operator is an encoder-decoder network to map an RGB image to its corresponding intermediate RAW measurements. The details of the network architecture are shown in Table 1

Table 1: Architecture description of the F Operator.

Layer	Type	In-Channel	Out-Channel	Kernel
1st	Conv+Relu	3	32	3×3
2nd	Conv+Relu	32	64	3×3
3rd	Conv+Relu	64	128	3×3
4th	UpConv+Relu	128	64	3×3
5th	UpConv+Relu	64	32	3×3
6th	UpConv+Sigmoid	32	1	3×3

3 Additional Qualitative Results

This section provides additional qualitative comparisons between the proposed methods and state-of-the-art defense baselines. For input-transformation-based adversarial defense methods, a balance between transformation fidelity and adversarial pattern removal is essential. A higher transformation fidelity indicates more details of the original image are kept, including adversarial patterns. In contrast, aggressive pattern removal could result in a severe loss of detail. Our method achieves this balance by a weighted sum between the G and S operator. The S operator is designed for faithfully reconstructing high-frequency details, while the G operator is trained to mitigate adversarial patterns. As shown in Figure 1, our approach offers better transformation fidelity than Comdefend, TVM, Pixel Deflection, and Resizing & Padding methods. Even though the JPEG-Defense method keeps more high-frequency details than the ones offered by our approach, it suffers from removing adversarial patterns, resulting in the worst performance among all compared methods.

4 Super-white Box Attack

We conduct a super-white attack experiment to provide insights into the benefits of the proposed non-differentiable S operator. A white-box setting means the attack methods have full access to a target model, including network architecture,

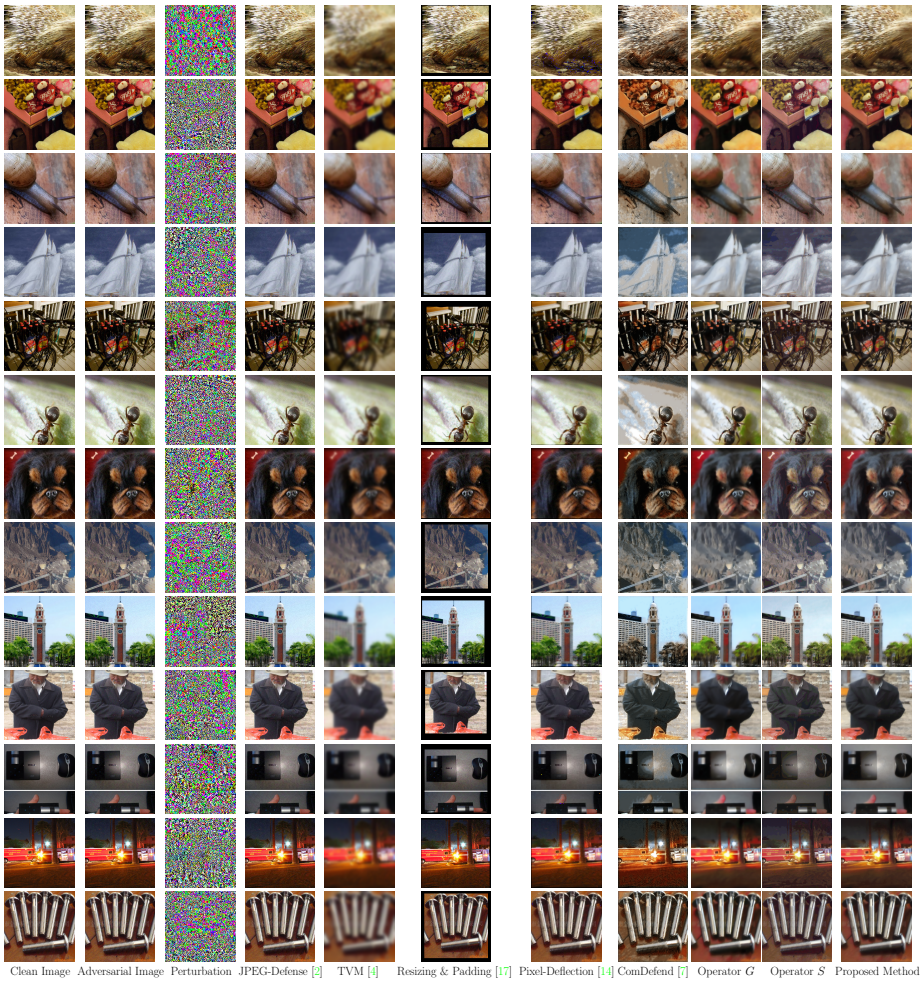

Fig. 1: Qualitative comparison of the proposed method along with both G and S operators against state-of-the-art defense methods on the ImageNet dataset, see text.

	Super-FSGM		Super-PGD		Super-BIM		Super-DeepFool		Super-C&W		Super-NewtonFool
	$2/255\uparrow$	$4/255\uparrow$	$2/255\uparrow$	$4/255\uparrow$	$2/255\uparrow$	$4/255\uparrow$	$L_\infty\uparrow$	$L_2\uparrow$	$L_\infty\uparrow$	$L_2\uparrow$	$L_\infty\uparrow$
ComDefend [7]	7.43	5.12	1.33	0.80	1.54	0.82	1.83	1.70	2.47	1.02	0.45
Proposed method w/ G only	11.97	7.82	3.68	1.83	4.02	2.35	3.27	2.11	4.93	1.36	0.83
Proposed Method	**62.88**	**55.25**	**65.05**	**63.97**	**64.49**	**60.56**	**68.62**	**59.93**	**68.74**	**65.13**	**38.41**

Table 2: We evaluate the defense performance against a super-white attack setting, where we assume the preprocessing model is exposed and the adversary can calculate the gradient for the differentiable part of the preprocessing model to construct stronger adversarial attacks. In such a setting, the differentiable preprocessing module (*i.e.*, ComDefend and G only) fail while our method does not collapse, demonstrating the benefit of exploiting the conventional ISP operator S in our method.

learned weights, and even training data, yet not to the preprocessing module used to transform input. In contrast, preprocessing modules and target models are exposed in the super-white box attack setting, allowing the adversary to calculate the gradient for adversarial attacks. In Table 2, we demonstrate the defense performances of three competing approaches on the ImageNet in Top-1 accuracy under the super-white box attack setting. We see that the defense methods with differentiable preprocessing modules (*i.e.*, ComDefend and the proposed method with G only) cannot function effectively under the super-white box attack setting. The attack only impacts the proposed method marginally, validating the benefit of the non-differentiable S operator.

5 Defense against BPDA Attack

In this section, we discuss the modifications of the proposed method to defend against BPDA attacks. For a non-differentiable target model, BPDA and derivative attacks build a differentiable proxy to mimic its functions. Then, the gradients of the created proxy are leveraged to attack the non-differentiable target model, *i.e.*obfuscated gradients. As such, the adversary cannot build an adequate proxy. In order to tackle such an attack, we modify our transformation algorithm from a determined algorithm to a randomized algorithm.

The modifications are listed in Algorithm 2; see original Algorithm 1. Specifically, at the inference time, instead of transforming the input image and directly feeding it to the downstream model, we apply our method N times sequentially, where the output of each transformation is the input for the next one. We add small random noise perturbations to the RGB and Raw images during each transformation. Here, two randomization effects are introduced: the number of times applying the transformation, N, is also random. The introduced randomization makes it hard to build an accurate proxy for providing effective gradients. Note that the new algorithm only requires modification at the inference stage, whereas the training stage remains the same; hence there is no need to retrain the model.

Algorithm 1 Original Inference Algorithm

Require: Input Image I, Operator S, F, G, Hyper-parameter ω

$Raw \leftarrow F(I)$
$I \leftarrow \omega * G(Raw) + (1 - \omega) * S(Raw)$

Algorithm 2 Modified Inference Algorithm

Require: Input Image I, Operator S, F, G, Hyper-parameter ω

$N \leftarrow random(0, 20)$
$i \leftarrow 0$
while $i \neq N$ **do**
    $n_I \leftarrow Gaussian(\mu = 0, \sigma = 0.05)$
    $n_{Raw} \leftarrow Gaussian(\mu = 0, \sigma = 0.05)$
    $Raw \leftarrow F(I + n_I)$
    $Raw \leftarrow Raw + n_{Raw}$
    $I \leftarrow \omega * G(Raw) + (1 - \omega) * S(Raw)$
    $i \leftarrow i + 1$
end while

The defense performance against BPDA is reported in Table 3. Specifically, the proposed modification improves the defending accuracy from 14.28 to 38.85. Also, compared to the baseline input transformation methods, the proposed method is the only method that shows robustness to the BPDA attack. The effectiveness of the modification for all other attacks is shown in Table 4, and we observe that the defense performances under other attacks decrease slightly after applying the modification. In other words, the modification significantly increases the robustness of the proposed method under the BPDA attack, at the cost of slightly decreasing the defense performance under other attacks. Note that the relative ranking of the proposed defense method versus baseline defense methods remains the same, with the proposed defense method outperforming baselines with a large margin before and after the modification.

Defense Method	JPEG-Defense	TVM	Randomized Resizing & Padding	Pixel-Deflection
	0.08	6.39	2.66	1.87
	ComDefend	HGD	Proposed Method (Unmodified)	Proposed Method (modified)
	0.03	0.03	14.28	38.85

Table 3: **Quantitative comparison with baseline input-transformation defense methods under BPDA attack:** The proposed modification improves the defending accuracy from 14.28 to 38.85, and the resulting method significantly outperforms the baselines.

6 Baseline Defense Methods

In this section, we provide additional details on the baseline adversarial defense methods used in our experiment. These baselines methods are state-of-the-art input-transformation defense methods with released code.

1. **JPEG-Defense [2]:** The JPEG-Defense method is a defense method based on input transformation. The intuition behind the method is that the compression process may remove high-frequency adversarial perturbations. A hyper-parameter of this method is the compression ratio, which affects the adversarial defense accuracy. A high compression ratio results in high image quality and less removal of adversarial perturbations. In our experiment,

	FSGM		PGD		BIM		DeepFool		C&W		NewtonFool	BPDA
	2/255 ↑	4/255 ↑	2/255 ↑	4/255 ↑	2/255 ↑	4/255 ↑	L_∞ ↑	L_2 ↑	L_∞ ↑	L_2 ↑	L_∞ ↑	L_∞ ↑
	ResNet-101											
JPEG-Defense [2]	33.14	20.71	45.19	21.74	36.78	8.5	53.16	45.69	59.06	52.01	24.65	0.08
TVM [4]	43.75	40.02	45.46	44.35	44.86	41.93	47.69	39.89	45.51	40.44	22.6	6.39
Randomized Resizing & Padding [17]	45.21	34.97	45.38	27.75	40.04	18.04	73.06	62.47	66.53	59.87	27.93	2.66
HGD [9]	54.75	43.85	55.26	50.05	56.74	48.61	64.34	58.13	59.98	52.88	27.70	0.03
Pixel-Deflection [14]	54.56	35.14	60.68	34.86	58.71	41.91	75.97	64.13	66.29	60.91	28.81	1.87
ComDefend [7]	48.21	36.51	53.28	48.38	51.39	42.01	63.68	55.62	58.53	50.38	26.46	0.03
Proposed Method w/o modification	66.02	58.85	68.34	66.17	66.91	63.01	72.04	63.52	71.40	67.33	40.96	14.28
Proposed Method w/ modification	61.87	54.42	63.79	59.03	61.29	57.35	68.80	56.42	68.13	61.65	35.88	38.85
	InceptionV3											
JPEG-Defense [2]	31.97	20.25	43.34	21.15	34.68	8.55	51.20	43.49	55.00	50.39	24.06	0.12
TVM [4]	42.47	37.23	42.75	41.61	42.80	39.71	45.21	37.39	43.27	37.51	23.05	4.58
Randomized Resizing & Padding [17]	41.86	34.49	43.41	25.60	39.42	16.62	70.24	58.65	63.24	55.62	27.55	2.09
HGD [9]	52.83	40.99	50.35	47.62	56.02	47.78	60.33	56.61	59.55	52.0	26.84	0.03
Pixel-Deflection [14]	51.42	34.27	56.13	32.49	56.18	39.13	71.16	61.58	61.94	57.58	28.01	1.56
ComDefend [7]	47.00	35.34	49.99	46.15	48.74	39.58	60.01	52.47	55.85	47.70	25.44	0.03
Proposed Method w/o modification	63.03	56.34	65.69	63.03	64.77	59.49	69.25	60.04	66.97	64.69	38.01	12.11
Proposed Method w/ modification	59.89	53.02	61.15	59.44	58.07	57.92	66.48	55.84	61.25	60.30	35.63	36.43

Table 4: **Quantitative Comparisons for the proposed method w/ and w/o modification for BPDA.** The modification significantly increase the robustness of the proposed method under BPDA attack, at the cost of slightly decreasing the defense performance under other attacks. The relative ranking of the proposed defense method versus baseline defense methods remains the same, with the proposed defense method outperforming baselines with a large margin both before and after modification.

we set the compression ratio to 75%, which is consistent with the setting proposed in [2].

2. **ComDefend [7]:** Similar to JPEG-Defense, ComDefend also aims to remove unnecessary details of the input image through compression. But unlike JPEG compression, ComDefend learns an auto-encoder to compress and reconstruct the image. Specifically, an encoder maps the input image to a compressed latent space, which is then fed into the decoder for reconstruction.

3. **TVM [4]:** The method works by randomly selecting a small set of pixels and reconstructing the most total-variation regularized image that is consistent with the selected pixels. A hyperparameter of this method is the number of pixels to be selected. Selecting a large portion of pixels keeps more image details but also keeps more adversarial perturbations. In our experiment, we tune the hyperparameter to achieve the best performance.

4. **Randomized Resizing & Padding [17]:** The method aims to mitigate the effect of adversarial perturbations through randomization. Given an input image, two randomization operations are performed. In particular, the method first randomly resizes the input image. Then, it randomly pads the input images with zeros.

5. **Pixel-Deflection [14]:** The method works by forcing the image to match natural image statistics. In doing so, the algorithm locally corrupts the image by redistributing pixel values by pixel deflection. Then, a subsequent wavelet-based denoising operation softens this corruption.

References

1. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks (2017) 2
2. Dziugaite, G.K., Ghahramani, Z., Roy, D.M.: A study of the effect of JPG compression on adversarial images. CoRR **abs/1608.00853** (2016) 5, 7, 8
3. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples (2015) 1
4. Guo, C., Rana, M., Cisse, M., Van Der Maaten, L.: Countering adversarial images using input transformations. ICLR (2018) 5, 8
5. Huang, S.C., Cheng, F.C., Chiu, Y.S.: Efficient contrast enhancement using adaptive gamma correction with weighting distribution. IEEE transactions on image processing **22**(3), 1032–1041 (2012) 4
6. Jang, U., Wu, X., Jha, S.: Objective metrics and gradient descent algorithms for adversarial examples in machine learning. In: Proceedings of the 33rd Annual Computer Security Applications Conference. pp. 262–277 (2017) 2
7. Jia, X., Wei, X., Cao, X., Foroosh, H.: Comdefend: An efficient image compression model to defend adversarial examples. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6084–6092 (2019) 5, 6, 8
8. Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial examples in the physical world (2017) 1
9. Liao, F., Liang, M., Dong, Y., Pang, T., Hu, X., Zhu, J.: Defense against adversarial attacks using high-level representation guided denoiser. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1778–1787 (2018) 8
10. Limare, N., Lisani, J.L., Morel, J.M., Petro, A.B., Sbert, C.: Simplest color balance. Image Processing On Line **1**, 297–315 (2011) 3
11. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks (2019) 2
12. Menon, D., Andriani, S., Calvagno, G.: Demosaicing with directional filtering and a posteriori decision. IEEE Transactions on Image Processing **16**(1), 132–141 (2006) 3
13. Moosavi-Dezfooli, S.M., Fawzi, A., Frossard, P.: Deepfool: a simple and accurate method to fool deep neural networks (2016) 2
14. Prakash, A., Moran, N., Garber, S., DiLillo, A., Storer, J.: Deflecting adversarial attacks with pixel deflection. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (2018) 5, 8
15. Reza, A.M.: Realization of the contrast limited adaptive histogram equalization (clahe) for real-time image enhancement. Journal of VLSI signal processing systems for signal, image and video technology **38**(1), 35–44 (2004) 4
16. Rizzi, A., Gatta, C., Marini, D.: Color correction between gray world and white patch. In: Human Vision and Electronic Imaging VII. vol. 4662, pp. 367–375. International Society for Optics and Photonics (2002) 3
17. Xie, C., Wang, J., Zhang, Z., Ren, Z., Yuille, A.: Mitigating adversarial effects through randomization. arXiv preprint arXiv:1711.01991 (2017) 5, 8
18. Xie, C., Wang, J., Zhang, Z., Zhou, Y., Xie, L., Yuille, A.: Adversarial examples for semantic segmentation and object detection (2017) 2