Counteracting Adversarial Attacks in Autonomous Driving

Qi Sun[1], Arjun Ashok Rao[1], Xufeng Yao[1], Bei Yu[1], **Shiyan Hu**[2]

[1]The Chinese University of Hong Kong
[2]University of Southampton

Vision-Based Object Detection

Classification

- output: class label

Localization

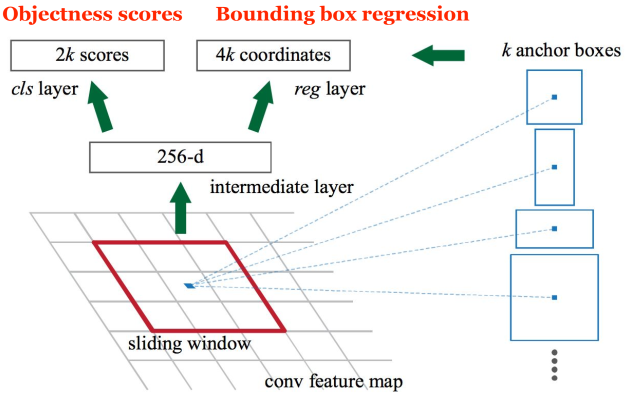
- output: bounding box in image

Object Detection:

- class label l
- bounding box in image, represented as vector (x, y, w, h)

Vision-Based Object Detection

Region Proposal Network (RPN)

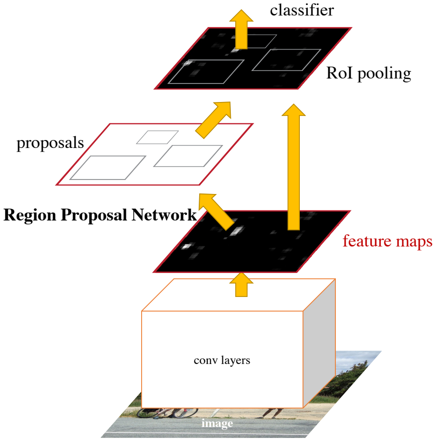


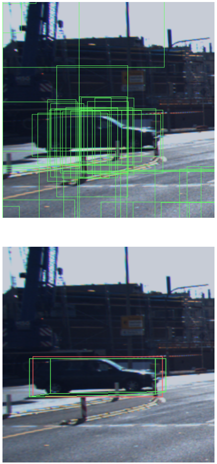
- Generate k boxes, regress label scores and coordinates for the k boxes.
- Use some metrics (*e.g.*, IoU) to measure the qualities of boxes.

Vision-Based Object Detection

Faster R-CNN

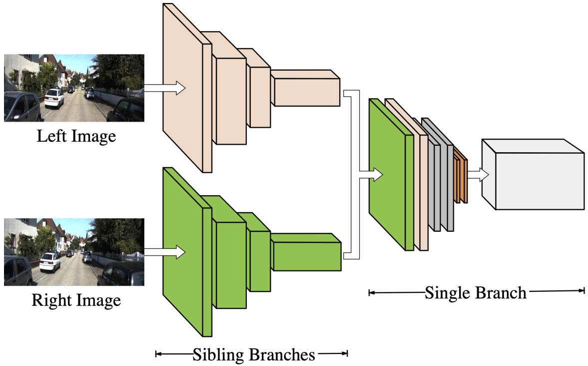
Vision-based object detection model.





Stereo-Based Vision System

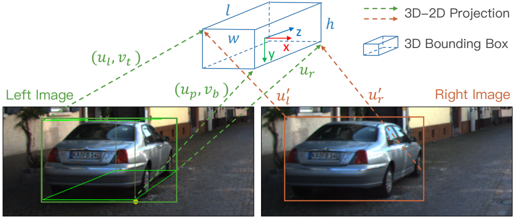
A typical stereo-based multi-task object detection model



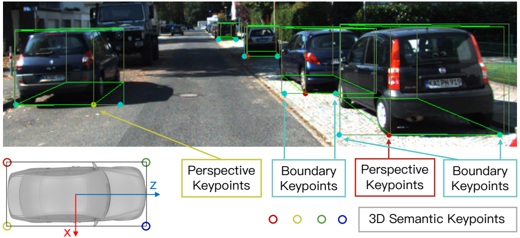
- Two sibling branches (*e.g.*, RPN modules) which use left and right images as inputs.
- A single branch conducts a regression task, *e.g.* predict viewpoint. Sometimes there are several independent single branches.

Stereo-Based Vision System

- Take advantage of left and right images to detect cars.
- Conduct multiple 3D regression tasks based on the joint detection results.



Take advantage of left and right images.



Multiple stereo-based tasks.

Adversarial Attacks

- Vision-based systems suffer from image perturbations (noises, dark light, signs, *etc.*).
- Deep learning models are vulnerable to these perturbations.
- The security risk is especially dangerous for 3D object detection in autonomous driving.
- Adversarial attacks have been widely studied to simulate these perturbations.
- Two typical and widely used attack methods: Fast Gradient Sign Method (FGSM) and Projected Gradient Descent (PGD).

Generate Adversarial Images

Fast Gradient Sign Method (FGSM)

- Direction of gradient: $\text{sign}(\nabla_x L(\theta, x, y))$, with loss function $L(\theta, x, y)$.
- Generates new input image with constrained perturbation δ:

$$\begin{aligned} x' &= x + \delta = x + \epsilon \cdot \text{sign}(\nabla_x L(\theta, x, y)), \\ \text{s.t. } &\|\delta\| \leq \epsilon. \end{aligned} \tag{1}$$

Generate Adversarial Images

Fast Gradient Sign Method (FGSM)

- Direction of gradient: $\mathrm{sign}(\nabla_x L(\theta, x, y))$, with loss function $L(\theta, x, y)$.
- Generates new input image with constrained perturbation δ:

$$\begin{aligned} x' &= x + \delta = x + \epsilon \cdot \mathrm{sign}(\nabla_x L(\theta, x, y)), \\ &\text{s.t. } \|\delta\| \leq \epsilon. \end{aligned} \tag{1}$$

Projected Gradient Descent (PGD)

- Contains several attack steps:

$$x_{t+1} = \prod_{x+S} (x_t + \alpha \cdot \mathrm{sign}(\nabla_x L(\theta, x, y))) \tag{2}$$

Adversarial Training

Traditional Training Method

- The typical form of most adversarial training algorithms involve training of target model on adversarial images.
- Adversarial training methods perform the following min-max training strategy shown as below:

$$\min_{\theta} \max_{\delta} L(x+\delta, \theta; y), \text{ s.t. } \|\delta\|_p \leq \epsilon,$$

where $\|\cdot\|_p$ is the ℓ_p-norm.

Adversarial Training

Traditional Training Method

- The typical form of most adversarial training algorithms involve training of target model on adversarial images.
- Adversarial training methods perform the following min-max training strategy shown as below:

$$\min_{\theta} \max_{\delta} L(x+\delta, \theta; y), \text{ s.t. } \|\delta\|_p \leq \epsilon,$$

where $\|\cdot\|_p$ is the ℓ_p-norm.

Stereo-based Training method

$$\min_{\theta} \max_{\delta_l, \delta_r} L(x_l+\delta_l, x_r+\delta_r, \theta; y),$$

$$\text{s.t. } \|\delta_l\|_p \leq \epsilon, \|\delta_r\|_p \leq \epsilon$$

where x_l and x_r represent left and right images, and δ_l and δ_r represent the perturbations on the left and right images respectively.

Stereo-Based Regularizer

For sibling branches

- Let $f_l(\cdot)$ and $f_r(\cdot)$ denote the features learned from left and right images.
- Distance between left and right images:

$$d\,(x_l, x_r) = \|f_l(x_l) - f_r(x_r)\|_n.$$

- Distance between two images with perturbations:

$$d\,(x_l{+}\delta_l, x_r{+}\delta_r) = \|f_l(x_l{+}\delta_l){-}f_r(x_r{+}\delta_r)\|_n.$$

- Add a margin m to reinforce the optimization of the distance function.

$$d\,(x_l, x_r) = \|f_l(x_l) - f_r(x_r) + m\|_n,$$
$$d\,(x_l + \delta_l, x_r + \delta_r) = \|f_l(x_l + \delta_l) - f_r(x_r + \delta_r) + m\|_n.$$

Stereo-Based Regularizer

For sibling branches

- The distance after attacks should be close to the original distance:

$$L_b = \left| \, d\left(x_l + \delta_l, x_r + \delta_r\right) - d\left(x_l, x_r\right) \, \right|.$$

Stereo-Based Regularizer

For sibling branches

- The distance after attacks should be close to the original distance:

$$L_b = \left| \, d\left(x_l + \delta_l, x_r + \delta_r\right) - d\left(x_l, x_r\right) \right|.$$

For single branch

- The left and right features are used as the joint inputs:

$$L_m = \|f_m(x_l + \delta_l, x_r + \delta_r) - f_m(x_l, x_r)\|_n.$$

Stereo-Based Regularizer

For sibling branches

- The distance after attacks should be close to the original distance:

$$L_b = \left| \, d\left(x_l + \delta_l, x_r + \delta_r\right) - d\left(x_l, x_r\right) \right|.$$

For single branch

- The left and right features are used as the joint inputs:

$$L_m = \|f_m(x_l + \delta_l, x_r + \delta_r) - f_m(x_l, x_r)\|_n.$$

New objective function

$$L = L_o + L_b + L_m,$$

where L_o is the original objective function.

Local Smoothness Optimization

Adversarial Robustness through Local Linearization

- Encourage the loss to behave linearly in the vicinity of training data.
- Approximate the loss function by its linear Taylor expansion in a small neighborhood.
- Take $f_l(\cdot)$ as an example, the first-order Taylor remainder $h_l(\epsilon, x_l)$ is given by :

$$h_l(\epsilon, x_l) = \| \delta_l \nabla_{x_l} f_l(x_l) + f_l(x_l + \delta_l) - f_l(x_l) - \delta_l \nabla_{x_l} f_l(x_l) \|_n.$$

- Define $\gamma_l(x_l, \epsilon)$ as the maximum of $h_l(\epsilon, x_l)$:

$$\gamma_l(\epsilon, x_l) = \max_{\|\delta_l\|_p \leq \epsilon} h_l(\epsilon, x_l). \tag{3}$$

Local Smoothness Optimization

Relaxation of regularizers

- According to the triangle inequality, $\|f_l(x_l + \delta_l) - f_l(x_l)\|_n$ is further relaxed to be:

$$
\begin{aligned}
\| f_l(x_l + \delta_l) - f_l(x_l) \|_n &\approx \| \delta_l \nabla_{x_l} f_l(x_l) + f_l(x_l + \delta_l) - f_l(x_l) - \delta_l \nabla_{x_l} f_l(x_l) \|_n \\
&\leq \| \delta_l \nabla_{x_l} f_l(x_l) \|_n + \| f_l(x_l + \delta_l) - f_l(x_l) - \delta_l \nabla_{x_l} f_l(x_l) \|_n \\
&\leq \| \delta_l \nabla_{x_l} f_l(x_l) \|_n + \gamma_l(x_l, \epsilon),
\end{aligned}
$$

Local Smoothness Optimization

Relaxation of regularizers

- According to the triangle inequality, $\|f_l(x_l+\delta_l)-f_l(x_l)\|_n$ is further relaxed to be:

$$
\begin{aligned}
\| f_l(x_l+\delta_l)-f_l(x_l) \|_n &\approx \| \delta_l \nabla_{x_l} f_l(x_l) + f_l(x_l+\delta_l) - f_l(x_l) - \delta_l \nabla_{x_l} f_l(x_l) \|_n \\
&\leq \| \delta_l \nabla_{x_l} f_l(x_l) \|_n + \| f_l(x_l+\delta_l) - f_l(x_l) - \delta_l \nabla_{x_l} f_l(x_l) \|_n \\
&\leq \| \delta_l \nabla_{x_l} f_l(x_l) \|_n + \gamma_l(x_l, \epsilon),
\end{aligned}
$$

- Accordingly, the regularization term L_b is relaxed as:

$$
\begin{aligned}
L_b &= | \, \|f_l(x_l+\delta_l) - f_r(x_r+\delta_r) + m\|_n - \|f_l(x_l) - f_r(x_r) + m\|_n \, | \\
&\leq \| f_l(x_l+\delta_l) - f_r(x_l) \|_n + \| f_l(x_r+\delta_r) - f_r(x_r) \|_n \\
&\leq \| \delta_l \nabla_{x_l} f_l(x_l) \|_n + \gamma_l(\epsilon, x_l) + \| \delta_r \nabla_{x_r} f_r(x_r) \|_n + \gamma_r(\epsilon, x_r),
\end{aligned}
$$

where $\gamma_l(\epsilon, x_l) = \max_{\|\delta_l\|_p \leq \epsilon} h_l(\epsilon, x_l)$ and $\gamma_r(\epsilon, x_r) = \max_{\|\delta_r\|_p \leq \epsilon} h_r(\epsilon, x_r)$.

Local Smoothness Optimization

Relaxation of regularizers

- The regularization term for the single branch is relaxed as:

$$
\begin{aligned}
L_m &= \| f_m(x_l + \delta_l, x_r + \delta_r) - f_m(x_l, x_r) \|_n \\
&\leq \| \delta_l \nabla_{x_l} f_m(x_l, x_r) + \delta_r \nabla_{x_r} f_m(x_l, x_r) \|_n + \gamma_m(\epsilon, x_l, x_r),
\end{aligned}
$$

where $\gamma_m(\epsilon, x_l, x_r)$ is the maximum of the high-order remainder $h_m(\epsilon, x_l, x_r)$.

Local Smoothness Optimization

Relaxation of regularizers

- The regularization term for the single branch is relaxed as:

$$
\begin{aligned}
L_m &= \| f_m(x_l + \delta_l, x_r + \delta_r) - f_m(x_l, x_r) \|_n \\
&\leq \| \delta_l \nabla_{x_l} f_m(x_l, x_r) + \delta_r \nabla_{x_r} f_m(x_l, x_r) \|_n + \gamma_m(\epsilon, x_l, x_r),
\end{aligned}
$$

where $\gamma_m(\epsilon, x_l, x_r)$ is the maximum of the high-order remainder $h_m(\epsilon, x_l, x_r)$.

- They are defined as follows:

$$
\begin{aligned}
h_m(\epsilon, x_l, x_r) &= \| f_m(x_l + \delta_l, x_r + \delta_r) - f_m(x_l, x_r) \\
&\quad - \delta_l \nabla_{x_l} f_m(x_l, x_r) - \delta_r \nabla_{x_r} f_m(x_l, x_r) \|_n, \\
\gamma_m(\epsilon, x_l, x_r) &= \max_{\|\delta_l\|_p \leq \epsilon, \|\delta_r\|_p \leq \epsilon.} h_m(\epsilon, x_l, x_r).
\end{aligned}
$$

Local Smoothness Optimization

Objective Function

- The Taylor remainders defined above is combined as:

$$L_h = h_l(\epsilon, x_l) + h_r(\epsilon, x_r) + h_m(\epsilon, x_l, x_r).$$

Local Smoothness Optimization

Objective Function

- The Taylor remainders defined above is combined as:

$$L_h = h_l(\epsilon, x_l) + h_r(\epsilon, x_r) + h_m(\epsilon, x_l, x_r).$$

- The first-order gradient terms are combined as:

$$\begin{aligned} L_\nabla = & \parallel \delta_l \nabla_{x_l} f_l(x_l) \parallel_n + \parallel \delta_r \nabla_{x_r} f_r(x_r) \parallel_n \\ & + \parallel \delta_l \nabla_{x_l} f_m(x_l, x_r) + \delta_r \nabla_{x_r} f_m(x_l, x_r) \parallel_n \end{aligned}$$

Local Smoothness Optimization

Objective Function

- The Taylor remainders defined above is combined as:

$$L_h = h_l(\epsilon, x_l) + h_r(\epsilon, x_r) + h_m(\epsilon, x_l, x_r).$$

- The first-order gradient terms are combined as:

$$\begin{aligned} L_\nabla = & \parallel \delta_l \nabla_{x_l} f_l(x_l) \parallel_n + \parallel \delta_r \nabla_{x_r} f_r(x_r) \parallel_n \\ & + \parallel \delta_l \nabla_{x_l} f_m(x_l, x_r) + \delta_r \nabla_{x_r} f_m(x_l, x_r) \parallel_n \end{aligned}$$

- Finally, together with the original loss function L_o, the optimization objective is defined as:

$$\min_{\theta} \left[L_a = L_o + L_\nabla + [\max_{\delta_l,\ \delta_r} L_h] \right]$$
$$\text{s.t. } \|\delta_l\|_p \le \epsilon, \quad \|\delta_r\|_p \le \epsilon,$$

Experimental Settings

- Benchmark: KITTI vehicle dataset (Easy, Moderate, and Hard) $*$.
- Stereo-based object detection model: Stereo R-CNN $\dagger$.
- Adversarial attack methods: FGSM and PGD.
- Baseline defense method: direct adversarial training with FGSM and PGD.

$*$Menze, Moritz, and Andreas Geiger. "Object scene flow for autonomous vehicles." CVPR, 2015.
$\dagger$P. Li, X. Chen, and S. Shen. "Stereo r-cnn based 3d object detection for autonomous driving." CVPR, 2019.

Experimental Results

IEEE/ACM 2020 INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN ICCAD 39th Edition

Adversarial Attacks

Table: Statistical Results of Adversarial Attacks

Model	AP_{2d} (%) ‡ Easy	Moderate	Hard	AOS (%) Easy	Moderate	Hard	AP_{3d} (%) ¶ Easy	Moderate	Hard	AP_{bv} (%) ‖ Easy	Moderate	Hard
`No Attack`	99.28	91.09	78.62	98.42	89.43	76.94	54.10	34.44	28.15	68.24	46.84	39.34
`FGSM`, $\epsilon = 0.7$	88.29	76.45	62.39	87.54	74.11	60.36	40.52	32.94	27.56	15.52	12.19	10.05
`FGSM`, $\epsilon = 2$	76.82	60.49	49.67	74.73	57.84	47.35	26.21	21.35	16.81	13.64	7.7	6.14
`PGD`, $\epsilon = 0.7$	69.55	58.94	48.04	66.72	56.04	45.59	22.52	18.88	15.32	7.02	5.53	4.29
`PGD`, $\epsilon = 2$	53.01	43.11	34.21	51.48	40.23	31.80	9.60	7.61	6.23	3.82	2.22	1.95

‡AP_{2d}: the average detection precision of the 2D bounding box.
AOS: the average orientation similarity of the joint 3D detection.
¶AP_{3d}: the average detection precision of the 3D bounding box.
‖AP_{bv}: the average localization precision of bird's eye view.

Experimental Results

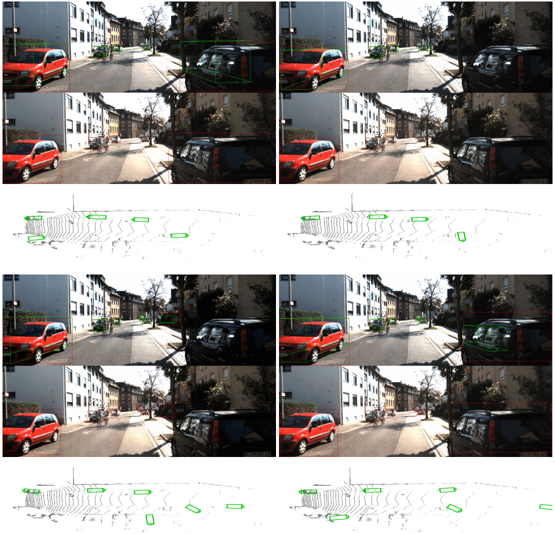
Defense Results

- Attack via FGSM and PGD.
- Defend via our method (`SmoothStereo`) and direct adversarial training.

Table: Statistical Results of Adversarial Defenses

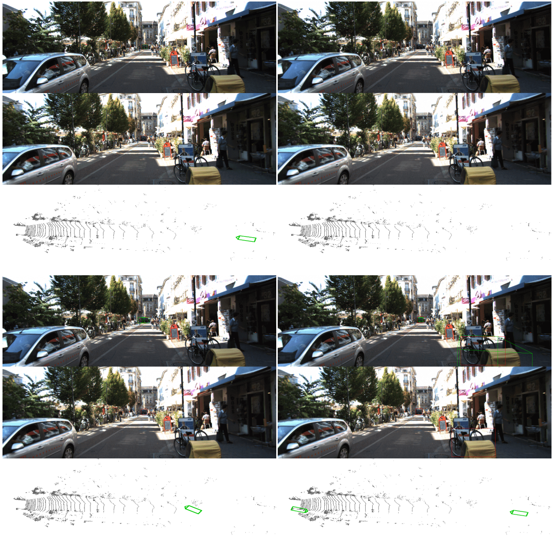
Testing Images	Defense Method	AP_{2d} (%) Easy	AP_{2d} (%) Moderate	AP_{2d} (%) Hard	AOS (%) Easy	AOS (%) Moderate	AOS (%) Hard	AP_{3d} (%) Easy	AP_{3d} (%) Moderate	AP_{3d} (%) Hard	AP_{bv} (%) Easy	AP_{bv} (%) Moderate	AP_{bv} (%) Hard
FGSM, $\epsilon = 0.7$	Direct + FGSM	87.58	81.54	71.53	87.25	80.11	62.42	41.95	30.62	**28.89**	21.57	19.62	16.56
	SmoothStereo	**88.38**	**82.74**	**73.94**	**88.89**	**81.87**	**63.63**	**45.51**	**31.01**	26.61	**24.50**	**20.88**	**18.26**
FGSM, $\epsilon = 2$	Direct + FGSM	84.73	70.82	57.90	**84.13**	69.19	55.61	40.15	30.57	**24.42**	16.21	13.03	10.54
	SmoothStereo	**85.95**	**72.64**	**61.22**	81.65	**74.83**	**60.00**	**41.43**	**31.63**	23.79	**18.25**	**14.76**	**12.53**
PGD, $\epsilon = 0.7$	Direct + PGD	73.37	**61.82**	56.66	73.04	60.46	50.04	**27.47**	20.08	**18.74**	**13.77**	7.10	9.30
	SmoothStereo	**75.67**	61.58	**59.73**	**73.43**	**62.27**	**52.82**	24.88	**20.90**	16.99	12.44	**11.73**	**9.46**
PGD, $\epsilon = 2$	Direct + PGD	54.46	49.11	40.44	53.37	46.23	38.07	14.39	10.38	9.32	5.84	**4.65**	3.29
	SmoothStereo	**55.29**	**49.38**	**41.92**	**53.47**	**47.27**	**40.60**	**18.11**	**12.42**	**9.43**	**6.82**	4.52	**3.94**

Experimental Results



Examples of results on FGSM attacks. The images from upper left to lower right are: ground-truth, FGSM attack with $\epsilon = 2$, defense via direct adversarial training, and defense via our `SmoothStereo`.

Experimental Results



Example of results on PGD attacks. The images from upper left to lower right are: ground-truth, PGD attack with $\epsilon = 2$, defense via direct adversarial training, and defense via our $\texttt{SmoothStereo}$.

Thank You