Adversarial Attacks On Multi-Agent Communication - Supplementary Materials

James Tu[1,2*] Tsunhsuan Wang[3*] Jingkang Wang[1,2] Sivabalan Manivasagam[1,2]
Mengye Ren[1,2] Raquel Urtasun[1,2]

[1]Waabi [2]University of Toronto [3]MIT
{jtu, wangjk, manivasagam, mren, urtasun}@cs.toronto.edu
johnsonwang0810@gmail.com

1. Additional Implementation Details

1.1. V2VNet

LiDAR Feature Extraction First, raw LiDAR point clouds are preprocessed to filter out points outside the region of interest (ROI), namely $[0m, 100m]$, $[-40m, 40m]$ for the x and y coordinates. Point clouds are then voxelized into $15.625cm$ density voxels using bilinear interpolation to calculate the weighting of each point in nearby voxels. The 3D voxel volume is then processed as a bird's eye view image with a 2D CNN which produces an intermediate representation to be shared.

BEV Feature Aggregation Upon receiving intermediate BEV features from other vehicles, the receiver first warps each image into its own coordinate frame such that messages are spatially aligned and features outside the receivers ROI are discarded. The messages are then fused using a graph neural network (GNN). The GNN performs 3 rounds of message passing where GNN node states are updated with a convolutional gated recurrent unit at each step. After the final iteration, an MLP outputs a post-aggregation BEV representation.

BEV Detection Following aggregation, the BEV image is processed with 4 multi-scale convolutional blocks similar to InceptionNet [5] to capture different levels of contextual information. Finally, a detection header outputs bounding box proposals which are then processed with non-maximum suppression.

Learning To train the model we use cross entropy loss for proposal classification and smooth L1 loss for bounding box regression. During training, hard negative mining is used to select 20 hard negatives for each sample. We first pretrain the detection model without any fusion and then freeze the weights of the LiDAR feature extraction network to train the network with fusion. Both stages are trained with the same annotations and using Adam [2] with learning rate 0.001.

1.2. ShapeNet Dataset

Objects are placed onto the same table which is also taken from the ShapeNet dataset and we use the same background with RGB (200, 200, 200). All images in this dataset is rendered using Habitat-sim [4] and we use the same lighting set up in every picture, with 5 light sources around the center of the table. When sub sampling 50 meshes from each class, we take the ones with the highest number of vertices to sample high quality meshes. Each agent uses a pinhole camera with a focal length of 1.0 units. Objects are placed on to a 5x3 grid on the table and we designate regularly spaced points for object placement. During placement, we sample one of these locations and apply a uniformly random offset bounded by 0.2 in each direction and perform collision checking to ensure objects placements are valid.

1.3. ShapeNet Model

Image Feature Extraction: Our ShapeNet detection model closely follows prior work on active vision for drones [1]. The model starts with a 2D U-Net [3] consisting of 4 blocks in the encoder and decoder. Each block consists of two groups of

*Equal contribution.

	Clean			Perturbed		
Agents	2	4	6	2	4	6
Feature Fusion	82.19	89.93	92.94	7.55	52.31	76.18
Input Fusion	81.03	88.18	91.28	42.71	69.95	80.29
Output Fusion	80.32	86.69	89.76	45.27	58.71	64.82

Table 1. Performance on clean and perturbed data for input fusion, output fusion, and intermediate feature fusion. Attack becomes with weaker with more benign agents in all fusion methods.

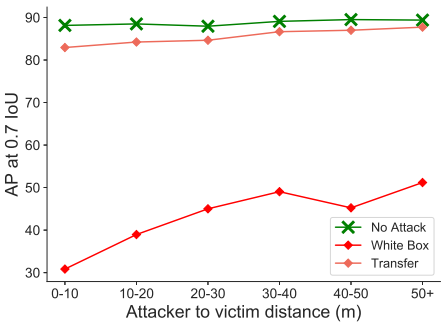

Figure 1. Attacks become stronger as attacker gets closer to victim.

convolution, group normalization [7], and ReLU activation. In the encoder, we down sample inputs using stride 2 convolution in the first convolution of each block. In the decoder, we up sample features using bilinear interpolation before each block. We set the initial number of channels to be 48.

3D Feature Aggregation Following the UNet, we use the pose information of the camera and the depth values of the pixels to unproject the pixel features into 3D voxels. For voxelization we use a $64 \times 32 \times 64$ grid where height is the second dimension. If multiple pixels map to the same voxel, we apply mean pooling to aggregate the features. After unprojecting into a common 3D coordinate frame, each agent broadcasts and the features are then aggregated with mean pooling.

3D Detection Following aggregation, a 3D U-Net similar is used to process the voxel features. The 3D U-Net is similar to the 2D U-Net with 4 blocks in the encoder and decoder. Finally, a detection header processes the features to generate proposals for each voxel.

Learning To train the model, we use Adam optimizer with learning rate 0.001, cross entropy loss for classification, and smooth L1 loss for regression. During training, we employ hard negative mining and mine 10 hard negatives for each sample.

2. Additional Results

Other Fusion Methods Aside from sharing learned intermediate representations, agents can alternatively share raw sensory inputs or predicted outputs. We follow prior implementations [6] and perform input fusion by directly overlaying LiDAR sweeps from other agents before running inference. For output fusion, we overlay output bounding boxes from other agents and then perform non-maximum suppression to select a single box for an instance.

We conduct additional experiments of attacks on these fusion methods in Table 1 and designate one attacker amongst a variable number of agents in the communication network. For input fusion we apply perturbations to each LiDAR point with $\epsilon = 10cm$ and for output fusion we perturb bounding box parameters with $\epsilon = 20cm$ and also add $N//8$ fake bounding boxes where N is the size of the unperturbed bounding box set. We find that the trend of increased robustness with more agents still holds. However, it is difficult to compare across different fusion methods fairly as it is unclear how to set fair ϵ constraints for all settings.

Attacker Distance In the V2V setting, agents perceive the world with a limited viewing range. Therefore, an attacker can only influence a victim SDV where their viewing ranges overlap. Thus, we expect stronger attacks when the attacker is closer to the victim and the overlap is maximized. We verify this intuition in Figure 1 where we plot the detection performance after attack versus the distance between the attacker and the victim. Observe that attacks become stronger when the attacker is close to the victim.

Cross Inference Another way to showcase the effectiveness of our proposed domain adaptation is to use the surrogate model to process features from the victim model and evaluate the performance, which we call *Cross Inference*. Specifically, this

AP @ 0.7 2 Agents	ShapeNet		V2V	
	Cross Inference ↑	Transfer Attack ↓	Cross Inference ↑	Transfer Attack ↓
Original Model	66.28	0.37	82.19	7.55
Surrogate w/o DA	0.51	66.21	2.47	81.34
Surrogate w/ DA	48.08	42.59	72.02	72.45

Table 2. Domain adaptation (DA) ablation. Cross inference refers to the surrogate model doing inference with the original model's intermediate feature maps. Note that a transfer attack with the original model is equivalent to a white box attack. Without domain adaptation, the surrogate cannot use the original model's features for inference and thus cannot produce transferable perturbations.

is evaluating the outputs of $G'(F(x))$ and we present results with a single attacker and victim in Table 2. Without domain adaptation, the surrogate model is not able to interpret features generated by the victim model to produce accurate detection outputs. However, with domain adaptation, the cross inference results are significantly better.

Qualitative Examples - V2V We provide more qualitative examples of the V2V setting in Figure 2. Furthermore, for our proposed online attack, we demonstrate results on a snippet of the dataset and present a video of the attack in the attached video *Supplementary Video.mp4*.

Qualitative Examples - ShapeNet We provide more qualitative examples in the ShapeNet setting in Table 3. The feature volumes are visualized from bird's eye view. After an imperceivable perturbation to the transmitted feature map, the output detections are severely degraded. For these visualizations, we projected the 3D bounding boxes onto the images.

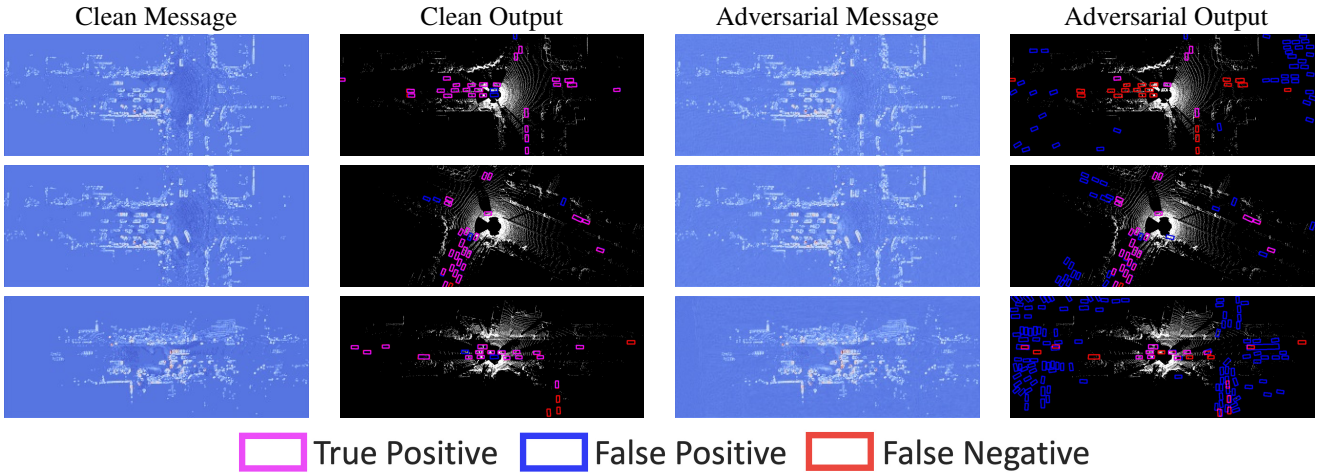

Figure 2. Qualitative examples of perturbing the transmitted feature map to attack bird's eye view vehicle detection. With imperceivable perturbations on the messages, the detection output can be severely degraded.

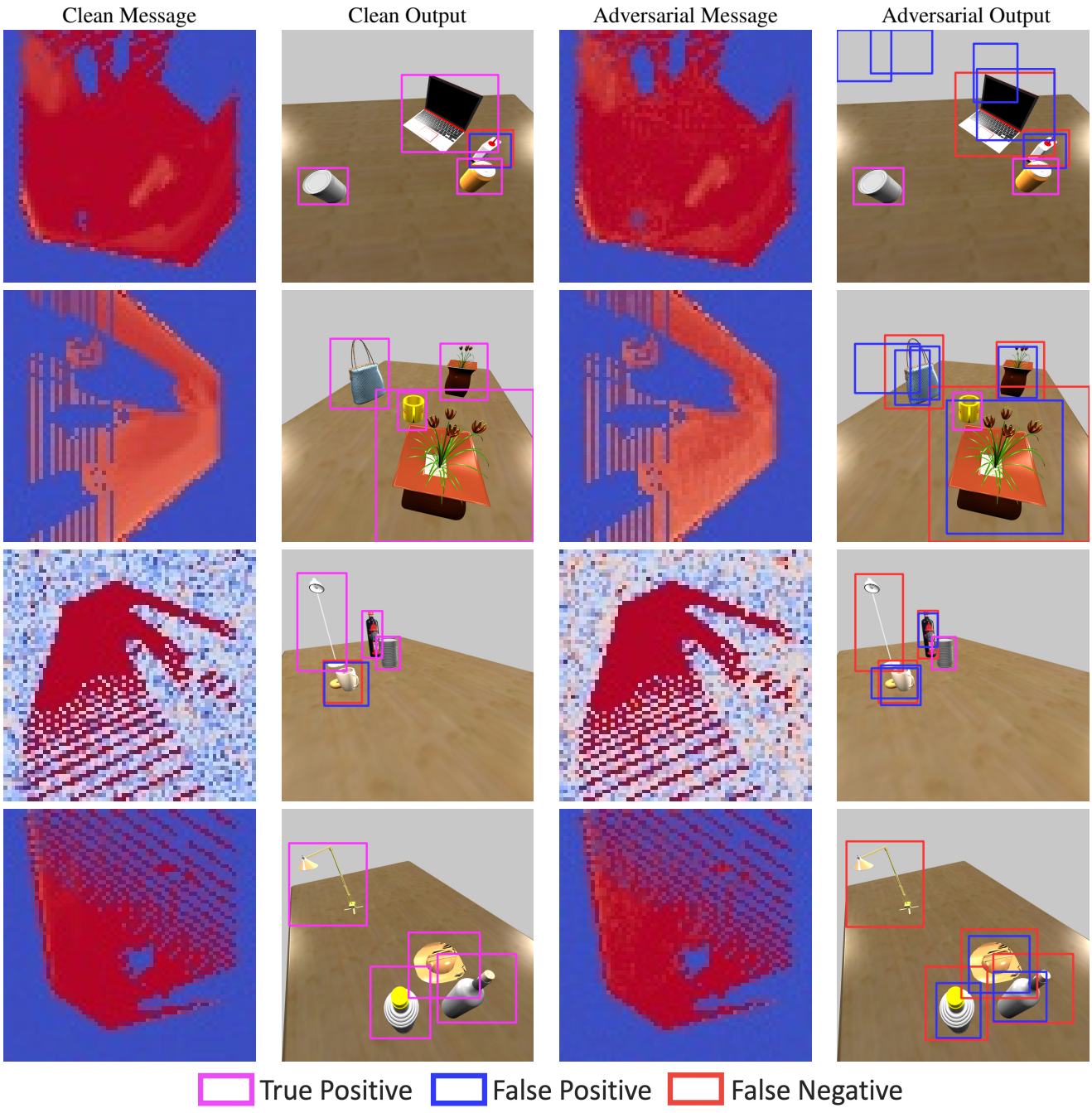

Figure 3. Qualitative examples of perturbing the transmitted feature map to attack 3D object detection on shapenet objects. With imperceivable perturbations on the messages, the detection output can be severely degraded.

References

[1] Ricson Cheng, Ziyan Wang, and Katerina Fragkiadaki. Geometry-aware recurrent neural networks for active visual recognition. In *NeurIPS*. 2018. 1

[2] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1

[3] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 1

[4] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9339–9347, 2019. 1

[5] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 1

[6] Tsun-Hsuan Wang, Sivabalan Manivasagam, Ming Liang, Bin Yang, Wenyuan Zeng, and Raquel Urtasun. V2vnet: Vehicle-to-vehicle communication for joint perception and prediction. *arXiv*, 2020. 2

[7] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 2