Supplementary Material for "Connecting the Dots: Detecting Adversarial Perturbations Using Context Inconsistency"

Shasha Li[1], Shitong Zhu[1], Sudipta Paul[1],
Amit Roy-Chowdhury[1], Chengyu Song[1], Srikanth Krishnamurthy[1],
Ananthram Swami[2], and Kevin S Chan[2]

[1] University of California, Riverside, USA
{sli057, szhu014, spaul007}@ucr.edu,
{amitrc}@ece.ucr.edu, {csong, krish}@cs.ucr.edu
[2] US Army CCDC Army Research Lab
{ananthram.swami.civ, kevin.s.chan.civ}@mail.mil

In this supplementary material, we 1) provide a tabular representation of the numbers used for the plots in the paper; 2) describe the architecture of the auto-encoders; 3) explain how we extend the state-of-the-art adversarial perturbation detection method FeatureSqueeze to defend object detection system; 4) describe how we apply non-deep Co-occurGraph to defend object detection system using cooccurrence relations inside the scene images; 5) include results of the detection performance of our proposed method against digital perturbations generated by various generation mechanisms; 6) provide results on the comparison of our proposed method with others that use context inconsistency to detect adversarial perturbations.

1 Values in the Plots

In the paper, some experimental results have been provided as plots for better visualization. We provide a table for each plot in this supplementary material. Tab. 1 and Tab. 2 correspond to the upper part and the lower part of Fig.8(a). Tab. 3 corresponds to Fig.8(b). Tab. 4 and Tab. 5 correspond to the upper part and lower part of Fig.8(c). Some entries are missing due to inadequate number of samples. For example, there are no entries for digital hiding attack for images with 6 objects in Tab. 4 because there are only 14 hiding-attacked images and the AUC reported would not be accurate. We report AUC when we have at least 50 attacked samples.

2 Architecture of the Auto-encoders

For each category, we use a separate auto-encoder to learn the distribution of its context profile. The architecture of the auto-encoders is identical and is shown in Fig. 1. The input to the auto-encoder is the context profile $x = [r, \gamma_{u1}, \gamma_{u2}, \gamma_{r1}, \gamma_{r2}]$. We denote the height and width of the input as H and W.

#Proposals	Digital Perturbations			Physical Perturbations		
	Miscategorization	Hiding	Appearing	Miscategorization	Hiding	Appearing
FeatureSqueeze [15]:						
1-3	0.751	0.628	0.587	0.762	0.679	0.647
3-6	0.712	0.626	0.614	0.749	0.633	0.653
6-9	0.748	0.612	0.609	0.784	0.654	0.688
9-12	0.727	0.576	0.629	0.767	0.672	0.692
Our method:						
1-3	0.830	0.977	0.843	0.940	0.955	0.950
3-6	0.902	0.995	0.859	0.983	0.982	0.977
6-9	0.933	0.999	0.903	0.993	0.998	0.985
9-12	0.950	0.983	0.929	0.996	1.000	0.991

Table 1: The detection performance against different attacks w.r.t. the number of proposals on the perturbed objects in PASCAL VOC dataset.

#Proposals	Digital Attack			Physical Attack		
	Miscategorization	Hiding	Appearing	Miscategorization	Hiding	Appearing
FeatureSqueeze [15]:						
1-3	0.704	0.692	0.594	0.670	0.678	0.502
3-6	0.656	0.679	0.569	0.719	0.692	0.528
6-9	0.584	-	0.562	0.653	0.641	0.552
9-12	0.616	-	0.521	0.682	-	0.556
Our method:						
1-3	0.896	0.961	0.804	0.918	0.938	0.952
3-6	0.947	0.992	0.876	0.985	0.982	0.973
6-9	0.978	-	0.90	0.983	0.999	0.989
9-12	0.988	-	0.932	0.995	-	0.987

Table 2: The detection performance against different attacks w.r.t. the number of proposals on the perturbed objects in MS COCO dataset.

IoU	PASCAL VOC		MS COCO	
	Digital Appearing	Physical Appearing	Digital Appearing	Physical Appearing
FeatureSqueeze [15]:				
0.0	0.605	0.653	0.614	0.550
0.0-0.1	0.606	0.605	0.557	0.552
0.1-0.2	0.592	0.642	0.549	0.518
0.2-0.3	0.602	0.752	0.521	0.478
0.3-0.4	0.590	0.640	0.504	0.586
0.4-0.5	0.594	0.644	0.510	0.474
Our method:				
0.0	0.748	0.939	0.769	0.977
0.0-0.1	0.872	0.945	0.827	0.970
0.1-0.2	0.879	0.966	0.849	0.978
0.2-0.3	0.906	0.980	0.850	0.984
0.3-0.4	0.905	0.986	0.855	0.996
0.4-0.5	0.924	0.994	0.910	0.990

Table 3: The detection performance against appearing attacks w.r.t. the overlap (IoU) between the perturbed region and some ground truth object in PASCAL VOC and MS COCO

$W = 5$ since there are 5 feature vectors in x and H equals to the dimension of the RoI pooling feature. A fully connected layer is first used to compress the node features (r) and edge features ($[\gamma_{u1}, \gamma_{u2}, \gamma_{r1}, \gamma_{r2}]$) separately. This is followed by two convolution layers, wherein the node and edge features are combined to learn the joint compression. Two fully connected layers are then used to

#Objects	Digital Perturbation			Physical Perturbations		
	Miscategorization	Hiding	Appearing	Miscategorization	Hiding	Appearing
FeatureSqueeze [15]:						
1	0.724	0.627	0.600	0.726	0.617	0.657
2	0.715	0.624	0.574	0.806	0.679	0.635
3	0.733	0.610	0.661	0.834	0.716	0.631
4	0.760	0.615	0.584	0.806	0.683	0.578
5	0.740	0.612	0.611	0.879	0.789	0.640
6	0.778	-	0.666	0.825	0.735	0.675
Our method:						
1	0.927	0.994	0.829	0.986	0.987	0.966
2	0.901	0.986	0.838	0.972	0.940	0.979
3	0.888	0.960	0.810	0.913	0.898	0.977
4	0.889	0.969	0.813	0.984	0.976	0.987
5	0.890	0.958	0.902	0.980	1.000	0.986
6	0.912	-	0.968	0.987	0.998	0.995

Table 4: The detection performance against different attacks w.r.t. the number of objects in the scene images in PASCAL VOC dataset.

#Object	Digital Attack			Physical Attack		
	Miscategorization	Hiding	Appearing	Miscategorization	Hiding	Appearing
FeatureSqueeze [15]:						
1	0.683	0.681	0.590	0.674	0.701	0.565
2	0.677	0.676	0.573	0.692	0.688	0.550
3	0.693	0.683	0.562	0.714	0.636	0.539
4	0.676	0.691	0.584	0.707	0.749	0.532
5	0.662	0.676	0.528	0.654	0.596	-
6	0.699	0.683	0.611	0.751	0.621	-
Our method:						
1	0.976	0.991	0.853	0.993	0.957	0.984
2	0.964	0.987	0.824	0.984	0.967	0.975
3	0.922	0.972	0.884	0.982	0.967	0.967
4	0.891	0.938	0.882	0.986	0.984	0.936
5	0.952	0.963	0.903	0.995	0.992	0.995
6	0.965	0.983	0.909	0.991	0.994	0.997

Table 5: The detection performance against different attacks w.r.t. the number of objects in the scene images in COCO dataset.

further compress the joint features. These layers form a bottleneck that drives the encoder to learn the true relationships between the features and get rid of redundant information.

3 Extending FeatureSqueeze to Region-level Perturbation Detection

3.1 FeatureSqueeze

FeatureSqueeze [15] proposes to squeeze the search space available to an adversary, driven by the observation that the feature input spaces are often unnecessarily large, which provides extensive opportunities for an adversary to construct adversarial examples. There are two feature squeezing methods used in their implementation: a) reducing the color bit depth of each pixel; b) spatial smoothing. By comparing a DNN model's prediction on the original input with that on

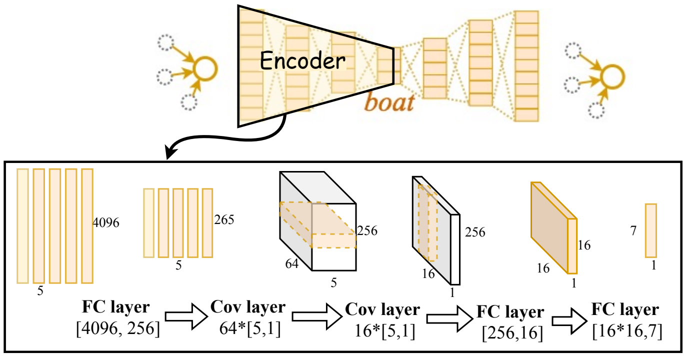

Fig. 1: Auto-encoder structure. One auto-encoder is learned for each category. The structure of the auto-encoders is identical.

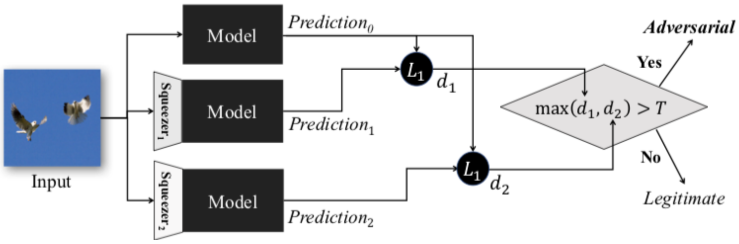

Fig. 2: This figure is from paper [15]. "The model is evaluated on both the original input and the input after being pre-processed by feature squeezers. If the difference between the models prediction on a squeezed input and its prediction on the original input exceeds a threshold level, the input is identified to be adversarial."

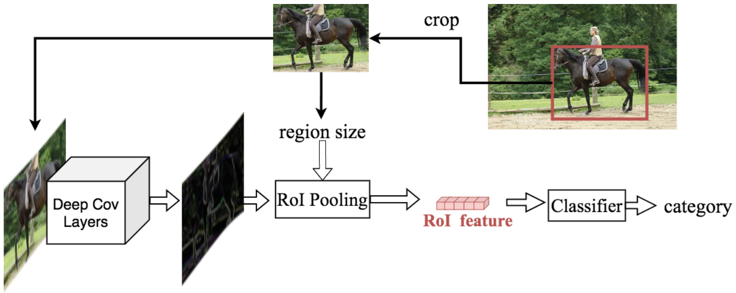

Fig. 3: Extending the DNN of FeatureSqueeze to region-level classification

squeezed ones, feature squeezing detects adversarial examples with high accuracy and few false positives. The framework of FeatureSqueeze [15] is shown in Fig. 2.

3.2 Extending to Region-level Detection

To detect perturbed regions inside scene images, the DNN model of FeatureSqueeze is required to operate on region-level. We crop the ground-truth regions, denoted as r, as the input to the DNN model. The output of the DNN model is the predicted category. To deal with region inputs with various size, we use RoI pooling [4] (box size equals to input region size) as the last feature extraction layer as shown in Fig. 3. Softmax function [2] is used as the last layer and cross entropy loss [5] is used as the objective loss function.

3.3 Implementation Details

We initialize the feature extractor with the weights pretrained on ImageNet. Momentum optimizer with momentum 0.9 is used to train the classifier. The learning rate is 1e-4 and decays every 80k iterations at decay rate 0.1. Training ends after 240k iterations. The final classification accuracy for the 20 categories in PASCAL VOC dataset is 95.6%. The final classification accuracy for the 80 categories in MS COCO dataset is 87.1%. The accuracy is not high because MS COCO is biased among categories, for example, more than 100k person instances v.s. less than 1k hair dryer instances. Even after we balance the number of samples among different categories, the performance is not good because some categories have too few examples, like the hair dryer category. The hyperparameters used for feature squeezing are exactly the same as the authors' GitHub implementation [12].

4 Co-occurGraph for Misclassification Attack Detection

We consider a non-deep model as baseline where co-occurrence statistics are used to detect misclassification due to adversarial perturbation. This approach uses the inconsistency between prior relational information obtained from the training data and inferred relational information conditioned on misclassified detection to detect the presence of adversarial perturbation. As the co-occurrence statistics of background class cannot be modeled, this approach is not applicable for detecting hiding and appearing attacks.

Prior Relational Information. Same as [1], we use the co-occurrence frequency of different categories of objects in the training data to obtain the prior relational information. Co-occurrence statistics gives an estimate of how likely two object classes will appear together in an image.

Graphical Representation. To encode the relational information of different classes of objects present in an image, we represent each image as an undirected graph $G = (V, E)$. Here, a node in V represents a single proposed region by the region proposal network. The edges $E = \{(i, j)|$ if region v_i and v_j are linked$\}$ represent the relationships between the regions. We formulate a tree structure graph where the region of interest is connected with all other proposed regions. The estimate of class probabilities of each proposed region generated by the object

detection model is used as the node potential and the co-occurrence statistics is used as the edge potential.
Detection of Misclassification Attack. For each image instance in test-set, we estimate its class conditional relatedness with other classes by making conditional inference on the representative graph. Conditional inference gives the pairwise conditional distribution of classes for each edge, which we use to obtain the posterior relational information of that image conditioned on the misclassified label. Based on the inconsistency among the prior relational information and posterior relational information, we detect if there is any misclassification attack.
Implementation Details. We use the Faster R-CNN [10] as the object detection and region proposal generation module. For each image, we consider top 20 proposed regions based on the class confidence score. To formulate the graph and make conditional inference, we use the publicly available UGM Toolbox [11].

5 Detection performance w.r.t. various perturbation generation mechanisms

In the paper, we show our proposed method is effective in detecting six different perturbation attacks, i.e., digital miscategorization attack, digital hiding attack, digital appearing attack, physical miscategorization attack, physical hiding attack and physical appearing attack. These attacks are different in terms of their attack goals and perturbation forms. Other defense papers also evaluate their defense methods w.r.t different perturbation generation mechanisms. Our defense strategy is dependent on the contextual information, and therefore should not rely heavily on the mechanism to generate the perturbation. We validate our hypothesis by testing our method against different perturbation generation mechanisms. The results in Tab. 6 show that our method is consistently effective against all the perturbation generation mechanisms.

As stated in the paper, COCO has few examples for certain categories. To make sure we have enough number of context profiles to learn the distribution, out of all the 80 categories, we choose 10 categories with the largest number of context profiles extracted. These 10 categories are “car”, “diningtable”, “chair”, “bowl”, “giraffe”, “person”, “zebra”, “elephant”, “cow”, “cat”. We also choose “stop sign” category because attacks on stop signs have gained long-lasting attentions. In addition to “background”, we have in total 12 categories and learn 12 autoencoders separately. We use these 12 autoencoders and evaluate misclassifications to these categories in our experiments.

6 Comparison with other context inconsistency based adversarial defense methods

The general notion of using context has been used to detect anomalous activities[16, 14, 3, 7]. When it comes to adversarial perturbation detection, spatial context has been used to detect adversarial perturbations against semantic segmentation [13]. Temporal context has been used to detect adversarial perturbation

Perturbation Generation Mechanism	PASCAL VOC	MS COCO
FeatureSqueeze [15]:		
FGSM [6]	0.788	0.678
BIM [9]	0.724	0.681
Our method:		
FGSM	0.947	0.915
BIM	0.938	0.959

Table 6: The detection performance against digital miscategorization attacks w.r.t. different perturbation generation mechanisms on PASCAL VOC and MS COCO

against video classification [8]. Context inconsistency has never been used to detect adversarial examples against objection detection systems. Essentially, our approach utilizes different kinds of context, including the spatial one from these prior works and object-level inter-relationships for the first time, as discussed in Tab. 7.

Detection	Temporal	Spatial	Object-object	Object-background	Object-scene	Task
Video[8]	✓					video classification
Seg[13]		✓				semantic segmentation
Our method		✓	✓	✓	✓	object detection

Table 7: Comparison with other context inconsistency based adversarial detection methods

References

1. Bappy, J.H., Paul, S., Roy-Chowdhury, A.K.: Online adaptation for joint scene and object classification. In: European Conference on Computer Vision. pp. 227–243. Springer (2016)
2. Bishop, C.M.: Pattern recognition and machine learning. Springer (2006)
3. Cao, N., Lin, C., Zhu, Q., Lin, Y.R., Teng, X., Wen, X.: Voila: Visual anomaly detection and monitoring with streaming spatiotemporal data. IEEE Transactions on Visualization and Computer Graphics **24**(1), 23–33 (2017)
4. Girshick, R.: Fast r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1440–1448 (2015)
5. Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. MIT press (2016)
6. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
7. Hasan, M., Choi, J., Neumann, J., Roy-Chowdhury, A.K., Davis, L.S.: Learning temporal regularity in video sequences. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 733–742. IEEE (2016)
8. Jia, X., Wei, X., Cao, X.: Identifying and resisting adversarial videos using temporal consistency. arXiv preprint arXiv:1909.04837 (2019)
9. Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial examples in the physical world. arXiv preprint arXiv:1607.02533 (2016)

10. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems. pp. 91–99 (2015)
11. Schmidt, M.: UGM: Matlab code for undirected graphical models
12. uvasrg: Featuresqueezing. https://github.com/uvasrg/FeatureSqueezing.git (2018)
13. Xiao, C., Deng, R., Li, B., Yu, F., Liu, M., Song, D.: Characterizing adversarial examples based on spatial consistency information for semantic segmentation. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 217–234 (2018)
14. Xu, D., Song, R., Wu, X., Li, N., Feng, W., Qian, H.: Video anomaly detection based on a hierarchical activity discovery within spatio-temporal contexts. Neurocomputing **143**, 144–152 (2014)
15. Xu, W., Evans, D., Qi, Y.: Feature squeezing: Detecting adversarial examples in deep neural networks. arXiv preprint arXiv:1704.01155 (2017)
16. Zhu, Y., Nayak, N.M., Roy-Chowdhury, A.K.: Context-aware activity recognition and anomaly detection in video. IEEE Journal of Selected Topics in Signal Processing **7**(1), 91–101 (2012)