https://doi.org/10.2352/ISSN.2470-1173.2021.4.MWSF-300

Attribution of Gradient Based Adversarial Attacks for Reverse Engineering of Deceptions

Michael Goebel[1], Jason Bunk[2], Srinjoy Chattopadhyay[2], Lakshmanan Nataraj[2], Shivkumar Chandrasekaran[1,2], B. S. Manjunath[1,2]
[1]University of California, Santa Barbara
[2]Mayachitra Inc; Santa Barbara, CA

Abstract

Machine Learning (ML) algorithms are susceptible to adversarial attacks and deception both during training and deployment. Automatic reverse engineering of the toolchains behind these adversarial machine learning attacks will aid in recovering the tools and processes used in these attacks. In this paper, we present two techniques that support automated identification and attribution of adversarial ML attack toolchains using Co-occurrence Pixel statistics and Laplacian Residuals. Our experiments show that the proposed techniques can identify parameters used to generate adversarial samples. To the best of our knowledge, this is the first approach to attribute gradient based adversarial attacks and estimate their parameters. Source code and data is available at: https://github.com/michael-goebel/ei_red.

Introduction

Convolutional neural networks (CNNs) are increasingly being used in critical applications, such as self-driving cars and face authentication. Recent works have shown that gradient based attacks can reduce accuracy of visual recognition networks to less than 1%, while minimally perturbing an image. The adversary uses gradient descent through the network to maximize the output at an incorrect label, while minimizing the perturbation to the image. Various attack methods have been produced using this common framework, including Fast Gradient Sign Method (FGSM) [9] and Projected Gradient Descent (PGD) [16]. Works have also been proposed to detect such adversarial samples, but none have been published which can estimate the adversarial setup from image samples. Knowing such parameters would allow for more accurate adversarial retraining against such attacks as well as aid in recovering the tools and processes used in these attacks [1].

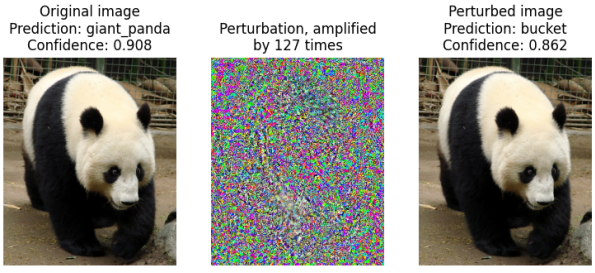


Figure 1: A sample PGD attack against ResNet. Small perturbations against a network with known weights can lead to significant differences in prediction outputs. Scores indicated here are confidence scores from 0-1, where the sum of all scores is equal to 1.

Gradient-descent based adversarial attacks use the gradients of deep neural networks (DNNs) to imperceptibly alter their inputs so as to change the output dramatically. Within this family, there are various strains of algorithms, each with several parameters. In this work, we propose to detect such adversarial attack toolchains and their parameters. Our objectives are two-fold:

1. To attribute an adversarially attacked image to a particular attack toolchain/family,
2. Once an attack has been identified, determine the parameters of the attack so as to facilitate the reverse engineering of these adversarial deceptions.

We will now briefly describe some of the attacks considered for detection and attribution. A deep neural network (DNN) is represented as a function $f : X \rightarrow Y$, where X denotes the input space of data and Y denotes the output space of the classification categories. The training set comprises known pairs (x_t, y_t), where $x_t \in X$ and $y_t \in Y$,and $f()$ is obtained by minimizing a loss function $J(f(x_t), y_t)$. We will consider the following attacks:

1. Fast Gradient Sign Method (FGSM): This attack perturbs a clean image x by taking a fixed step in the direction of the gradient of $J(f(x_t), y_t)$ with respect to x_t.
2. Projected Gradient Descent (PGD): This attack is an improvement over FGSM, where the adversarial samples x' are generated by multiple iterations and intermediate results are clipped so as to keep them within the ε-neighborhood of $x : x'_i = x'_{i-1} - clip_{\varepsilon}(\alpha \cdot sign(\nabla_x J(f(x'_{i-1}, y))))$.

These two attacks are examples of l_∞ attacks, where ε represents the maximum allowable perturbation to any pixel in x. The software repositories of these attacks can be obtained from the following: Advertorch [6], Adversarial Robustness Toolbox [19], Foolbox [22], CleverHans [20]. A PGD example from the Advertorch toolbox is given in Figure 1.

Related Works

Many works have taken the approach of creating more robust networks, for which small changes in input will not significantly change the output classification [2, 11, 13, 14, 21, 23, 24, 27]. Generally, these methods cause a significant decrease in accuracy, for both tampered and untampered images [5]. While these networks are necessary when class estimation is required for all samples, others methods may be more favorable when this requirement is relaxed.

Detection has become another popular approach to circumventing these attacks [4, 7, 8, 10, 17, 12]. Such methods allow

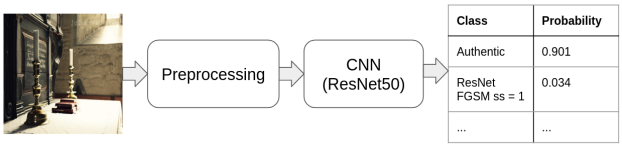


Figure 2: High level model diagram for detection. All models fit into this framework, with different preprocessing methods.

for the classification networks to remain as is, while filtering out adversarial examples before they reach the target network. The methods presented in this paper move a step beyond simple detection, with the addition of attack classification and parameter estimation.

Method

Model

To enhance the artifacts created by adversarial attacks, we consider two preprocessing methods common to image forensic, before training a neural network. A visual summary of our detector is given in Figure 2. As a baseline, we compare these two methods against a method with no preprocessing. The first is a Laplacian high-pass filter. Similar filters have been used for both image resampling detection [15], and general image manipulation detection [3]. In our tests, the following 3x3 filter was applied to each of the RGB channels:

$$h(x,y) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{1}$$

The second preprocessing method investigated is the co-occurrence matrix. Such matrices have been used extensively in detection of steganography [26, 25] as well as in detection of GAN images [18]. For this method, two dimensional histograms of adjacent pixel pairs are constructed for each of the color channels. Below we show the equation for horizontal pairs, where X is a 2D array representing a single color channel. A sample image passed through each mode of processing is shown in Figure 3.

$$C_{i,j} = \sum_{m,n} [X_{m,n} = i][X_{m,n+1} = j] \tag{2}$$

This can be applied to X^T for vertical pairs as well, and on all three channels. These 6 co-occurrence matrices are then stacked into a final input tensor of size $256 \times 256 \times 6$. This tensor is passed to a CNN classifier as a multi-channel image.

Detection, Attribution, and Estimation

For our final output, we would like to tell a user whether or not a query is tampered, what attack method was used, and the parameters for that method. This high level idea described visually in Figure 4. To accomplish this, we train a multiclass network, with each attack and parameter combination as a different label. To form the aggregated sets, such as real vs tampered, we sum the model outputs associated with each set. The set with the largest output is selected as the estimated class.

If the image is predicted to be tampered, we then compute our parameter estimates using the model outputs for the predicted

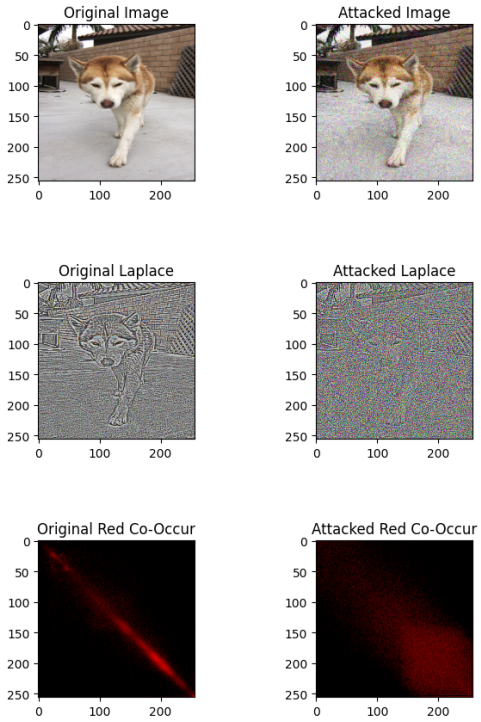


Figure 3: An untampered image, and corresponding PGD attacked image, with a large step size and number of steps to amplify the difference. The adversarial noise added appears across the whole image. The difference in the co-occurrence matrices is notable in the significant increase in spread about the diagonal.

meta-class. A weighted sum is used, with the model outputs as the weights, and the associated class parameters as the values.

$$P_{est} = \frac{\sum_{i \in S} P_i \times y_i}{\sum_{i \in S} y_i} \tag{3}$$

Experiments

Dataset

A full list of the attacks investigated is given in table 1. These attacks are repeated on VGG16 and ResNet50, and each is classified separately. Only the ends of the parameter spectrum are used for training. Parameters which are in-between these ends of the spectrum are seen only at test time. A total of 12 different tampered classes are used at training time, with one additional class for untampered, for a total of 13.

The dataset is constructed from a random selection of ImageNet samples, all resized to 256×256. Attacks are run as targeted, with the new label being randomly selected from the 999 labels which are different than the associated ground truth. The attacks are then run to maximize the network output for the target class.

Model Training

A ResNet50 pretrained on ImageNet was used as our initial network, with the input and output layers modified to accommodate the different input and output sizes for this task. The model

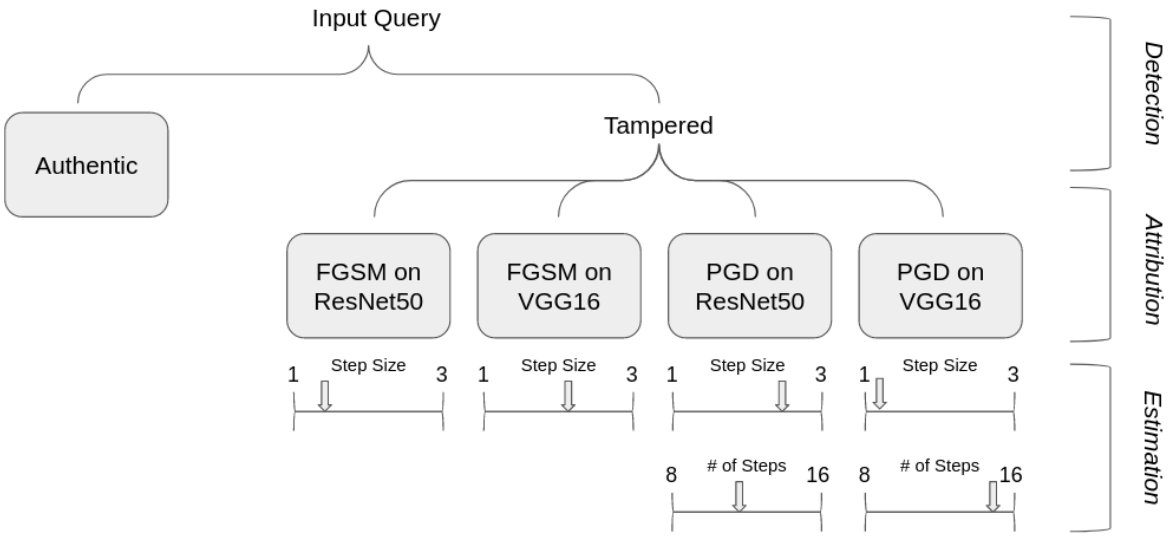


Figure 4: Levels of information provided to the user by our method. Using a single network, we demonstrate results for detection, attribution, and parameter estimation.

Attack	Parameters	Training	Testing
FGSM	ss = 1	X	X
FGSM	ss = 2		X
FGSM	ss = 3	X	X
PGD	ss = 1, ns = 8	X	X
PGD	ss = 1, ns = 12		X
PGD	ss = 1, ns = 16	X	X
PGD	ss = 2, ns = 8		X
PGD	ss = 2, ns = 12		X
PGD	ss = 2, ns = 16		X
PGD	ss = 3, ns = 8	X	X
PGD	ss = 3, ns = 12		X
PGD	ss = 3, ns = 16	X	X

Table 1: Breakdown of the attacks used for training and testing. All attacks are repeated against pretrained VGG16 and ResNet50. "ss" denotes "stride size", assuming an image is in the range [0,255], and "ns" denotes "number of steps".

was trained over 20 epochs, using a batch size of 32, Adam optimizer, and cross-entropy loss. After each of the epochs, the model was evaluated on the validation set. The weights corresponding to the lowest validation loss were saved, and used for the remainder of the tests.

Results

Table 5 shows our results for several different separations of the meta-classes. The co-occurrence and direct methods performed better on average than the Laplace method across the different tasks. Considering all 3 detectors, our methods achieved at least 90% accuracy for each task. Figure 5 shows a t-SNE clustering of the deep features taken from one of these classification networks.

Table 6 shows the results of each method on different estimation tasks. Notably, the Laplace and co-occurrence methods out-perform the baseline direct method in several of the estimation tasks.

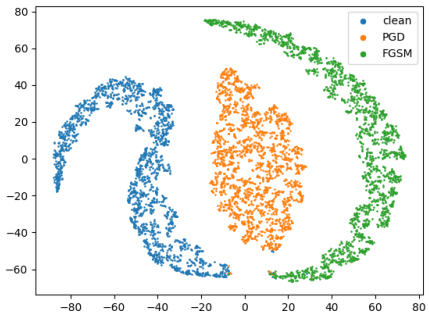


Figure 5: t-SNE results from the co-occurrence model for one of the classification tasks. Features are taken from the penultimate layer of the detection network, and run though t-SNE dimensionality reduction. Clear clusters are seen dividing each class.

Discussion

Across all tasks, the co-occurrence preprocessing tended to perform the best. Especially noteworthy is the difference in performance of the direct method between the classification tasks and the parameter estimation tasks. Of the three, the direct provides the most information too the neural network. While this led to good results on the training classes, the information bottleneck provided by the co-occurrence and Laplace functions may help reduce overfitting.

Conclusion and Future Directions

In this work, we presented several methods for attribution and parameter estimation of select adversarial attacks. This combination of detection, attribution, and parameter estimation was accomplished using a single pass through a multi-class neural network, trained on a sampling of several common adversarial attacks.

While our model was demonstrated effective against several attacks not seen in the training sets, there are a variety of ad-

original_resized	0.864	0.012	0.001	0.001	0.000	0.000	0.000	0.110	0.007	0.005	0.000	0.000	0.000
resnet50_FGSM_ss_1	0.026	0.762	0.004	0.001	0.000	0.000	0.000	0.177	0.021	0.008	0.000	0.000	0.000
resnet50_FGSM_ss_3	0.000	0.002	0.977	0.015	0.000	0.000	0.000	0.000	0.004	0.003	0.000	0.000	0.000
resnet50_PGD_ns_8_ss_1	0.000	0.000	0.042	0.953	0.000	0.003	0.000	0.000	0.000	0.001	0.000	0.000	0.000
resnet50_PGD_ns_8_ss_3	0.000	0.000	0.000	0.000	0.999	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
resnet50_PGD_ns_16_ss_1	0.000	0.000	0.000	0.008	0.003	0.988	0.000	0.000	0.000	0.000	0.000	0.001	0.000
resnet50_PGD_ns_16_ss_3	0.000	0.000	0.000	0.000	0.001	0.000	0.999	0.000	0.000	0.000	0.000	0.000	0.000
vgg16_FGSM_ss_1	0.244	0.018	0.001	0.001	0.000	0.000	0.000	0.704	0.023	0.009	0.000	0.000	0.000
vgg16_FGSM_ss_3	0.007	0.007	0.004	0.000	0.000	0.000	0.000	0.047	0.848	0.087	0.000	0.000	0.000
vgg16_PGD_ns_8_ss_1	0.000	0.001	0.002	0.001	0.000	0.000	0.000	0.002	0.003	0.989	0.000	0.002	0.000
vgg16_PGD_ns_8_ss_3	0.000	0.000	0.000	0.000	0.000	0.001	0.000	0.000	0.000	0.000	0.989	0.009	0.001
vgg16_PGD_ns_16_ss_1	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.003	0.003	0.993	0.000
vgg16_PGD_ns_16_ss_3	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000

Table 2: Confusion matrix for direct method on test dataset. Column labels are in the same order as row labels. Rows indicate ground truth, columns indicate predicted.

original_resized	0.919	0.064	0.003	0.004	0.000	0.001	0.000	0.008	0.000	0.000	0.000	0.000	0.000
resnet50_FGSM_ss_1	0.035	0.943	0.007	0.005	0.000	0.001	0.000	0.008	0.001	0.000	0.000	0.000	0.000
resnet50_FGSM_ss_3	0.000	0.007	0.981	0.009	0.000	0.000	0.000	0.000	0.002	0.000	0.000	0.000	0.000
resnet50_PGD_ns_8_ss_1	0.000	0.000	0.000	0.999	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.000
resnet50_PGD_ns_8_ss_3	0.000	0.000	0.000	0.000	0.733	0.027	0.006	0.000	0.000	0.000	0.199	0.028	0.007
resnet50_PGD_ns_16_ss_1	0.000	0.000	0.000	0.000	0.000	0.560	0.002	0.000	0.000	0.000	0.000	0.424	0.013
resnet50_PGD_ns_16_ss_3	0.000	0.000	0.000	0.000	0.004	0.081	0.297	0.000	0.000	0.000	0.000	0.204	0.414
vgg16_FGSM_ss_1	0.073	0.207	0.005	0.006	0.000	0.000	0.000	0.706	0.001	0.002	0.000	0.000	0.000
vgg16_FGSM_ss_3	0.006	0.009	0.098	0.010	0.000	0.000	0.000	0.005	0.869	0.002	0.000	0.001	0.000
vgg16_PGD_ns_8_ss_1	0.000	0.000	0.000	0.301	0.000	0.000	0.000	0.000	0.000	0.697	0.000	0.003	0.000
vgg16_PGD_ns_8_ss_3	0.000	0.000	0.000	0.000	0.154	0.019	0.004	0.000	0.000	0.000	0.752	0.060	0.012
vgg16_PGD_ns_16_ss_1	0.000	0.000	0.000	0.000	0.000	0.039	0.000	0.000	0.000	0.000	0.000	0.956	0.005
vgg16_PGD_ns_16_ss_3	0.000	0.000	0.000	0.000	0.002	0.082	0.146	0.000	0.000	0.000	0.004	0.214	0.552

Table 3: Confusion matrix for Laplace method on test dataset. Column labels are in the same order as row labels. Rows indicate ground truth, columns indicate predicted.

original_resized	0.941	0.042	0.002	0.001	0.000	0.000	0.000	0.010	0.001	0.002	0.000	0.000	0.000
resnet50_FGSM_ss_1	0.008	0.880	0.003	0.001	0.000	0.000	0.000	0.108	0.000	0.001	0.000	0.000	0.000
resnet50_FGSM_ss_3	0.001	0.001	0.726	0.001	0.000	0.000	0.000	0.000	0.269	0.002	0.000	0.000	0.000
resnet50_PGD_ns_8_ss_1	0.000	0.000	0.000	0.853	0.000	0.000	0.000	0.000	0.000	0.146	0.000	0.000	0.000
resnet50_PGD_ns_8_ss_3	0.000	0.000	0.000	0.000	0.934	0.000	0.000	0.000	0.000	0.000	0.065	0.000	0.000
resnet50_PGD_ns_16_ss_1	0.000	0.000	0.000	0.000	0.000	0.958	0.000	0.000	0.000	0.000	0.000	0.042	0.000
resnet50_PGD_ns_16_ss_3	0.000	0.000	0.000	0.001	0.004	0.000	0.983	0.000	0.000	0.000	0.000	0.000	0.013
vgg16_FGSM_ss_1	0.012	0.746	0.002	0.000	0.000	0.000	0.000	0.238	0.001	0.001	0.000	0.000	0.000
vgg16_FGSM_ss_3	0.001	0.000	0.379	0.001	0.000	0.000	0.000	0.001	0.617	0.002	0.000	0.000	0.000
vgg16_PGD_ns_8_ss_1	0.001	0.000	0.000	0.130	0.000	0.000	0.000	0.000	0.000	0.868	0.000	0.001	0.000
vgg16_PGD_ns_8_ss_3	0.000	0.000	0.000	0.000	0.091	0.000	0.000	0.000	0.000	0.000	0.909	0.000	0.000
vgg16_PGD_ns_16_ss_1	0.000	0.000	0.000	0.000	0.000	0.053	0.000	0.000	0.000	0.000	0.000	0.947	0.000
vgg16_PGD_ns_16_ss_3	0.000	0.000	0.000	0.000	0.000	0.000	0.002	0.000	0.000	0.000	0.000	0.000	0.998

Table 4: Confusion matrix for co-occurrence method on test dataset. Column labels are in the same order as row labels. Rows indicate ground truth, columns indicate predicted.

Meta Classes	Direct	Laplace	Co-occur
Binary Detection	0.921	0.955	**0.970**
Full attribution	**0.907**	0.834	0.808
Original, ResNet, VGG	**0.925**	0.834	0.865
Original, FGSM, PGD	0.919	0.961	**0.978**
Full Classification	**0.928**	0.766	0.835

Table 5: Mean average precision on different meta classification tasks. Full Attribution denotes classification between the original, FGSM ResNet, FGSM VGG, PGD ResNet, and PGD classes. Full Classification refers to accuracy across all 13 classes in the training set.

	Direct	Laplace	Co-Occur
FGSM step size	0.491	**0.469**	0.509
PGD step size	0.567	0.680	**0.535**
PGD number of steps	4.17	3.66	**3.42**

Table 6: Root Mean Squared Error (RMSE) for parameter estimation. Step sizes are sampled from {1,2,3}, and number of steps sampled from {8,12,16}.

ditional attacks to be considered. Furthermore, our method of parameter interpolation is inherently limited to estimating values only within the range of values in the training set. Creation of a more robust model for real-world deployment would require a broader sampling of attack methods, target networks, and attack parameters.

References

[1] DARPA Artificial Intelligence Exploration (AIE) Opportunity, DARPA-PA-19-02-09, Reverse Engineering of Deceptions (RED). https://beta.sam.gov/opp/258cc833c18749de87aba9c129ee2205/view. 1

[2] O. Bastani, Y. Ioannou, L. Lampropoulos, D. Vytiniotis, A. Nori, and A. Criminisi. Measuring neural net robustness with constraints. *arXiv preprint arXiv:1605.07262*, 2016. 1

[3] B. Bayar and M. C. Stamm. A deep learning approach to universal image manipulation detection using a new convolutional layer. In *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*, pages 5–10, 2016. 2

[4] A. N. Bhagoji, D. Cullina, and P. Mittal. Dimensionality reduction as a defense against evasion attacks on machine learning classifiers. *arXiv preprint arXiv:1704.02654*, 2, 2017. 1

[5] N. Carlini and D. Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14, 2017. 1

[6] G. W. Ding, L. Wang, and X. Jin. Advertorch v0. 1: An adversarial robustness toolbox based on pytorch. *arXiv preprint arXiv:1902.07623*, 2019. 1

[7] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017. 1

[8] Z. Gong, W. Wang, and W.-S. Ku. Adversarial and clean data are not twins. *arXiv preprint arXiv:1704.04960*, 2017. 1

[9] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 1

[10] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. McDaniel. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*, 2017. 1

[11] S. Gu and L. Rigazio. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*, 2014. 1

[12] D. Hendrycks and K. Gimpel. Early methods for detecting adversarial images. *arXiv preprint arXiv:1608.00530*, 2016. 1

[13] R. Huang, B. Xu, D. Schuurmans, and C. Szepesvári. Learning with a strong adversary. *arXiv preprint arXiv:1511.03034*, 2015. 1

[14] J. Jin, A. Dundar, and E. Culurciello. Robust convolutional neural networks under adversarial noise. *arXiv preprint arXiv:1511.06306*, 2015. 1

[15] M. Kirchner. Fast and reliable resampling detection by spectral analysis of fixed linear predictor residue. In *Proceedings of the 10th ACM workshop on Multimedia and security*, pages 11–20, 2008. 2

[16] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017. 1

[17] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff. On detecting adversarial perturbations. *arXiv preprint arXiv:1702.04267*, 2017. 1

[18] L. Nataraj, T. M. Mohammed, B. Manjunath, S. Chandrasekaran, A. Flenner, J. H. Bappy, and A. K. Roy-Chowdhury. Detecting gan generated fake images using co-occurrence matrices. *Electronic Imaging*, 2019(5):532–1, 2019. 2

[19] M.-I. Nicolae, M. Sinn, T. N. Minh, A. Rawat, M. Wistuba, V. Zantedeschi, I. M. Molloy, and B. Edwards. Adversarial robustness toolbox v0. 2.2. 2018. 1

[20] N. Papernot, F. Faghri, N. Carlini, I. Goodfellow, R. Feinman, A. Kurakin, C. Xie, Y. Sharma, T. Brown, A. Roy, et al. Technical report on the cleverhans v2. 1.0 adversarial examples library. *arXiv preprint arXiv:1610.00768*, 2016. 1

[21] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pages 582–597. IEEE, 2016. 1

[22] J. Rauber, W. Brendel, and M. Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models. *arXiv preprint arXiv:1707.04131*, 2017. 1

[23] A. Rozsa, E. M. Rudd, and T. E. Boult. Adversarial diversity and hard positive generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 25–32, 2016. 1

[24] U. Shaham, Y. Yamada, and S. Negahban. Understanding adversarial training: Increasing local stability of neural nets through robust optimization. *arXiv preprint arXiv:1511.05432*, 2015. 1

[25] K. Sullivan, U. Madhow, S. Chandrasekaran, and B. Manjunath. Steganalysis for markov cover data with applications to images. *IEEE Transactions on Information Forensics and Security*, 1(2):275–287, 2006. 2

[26] K. Sullivan, U. Madhow, S. Chandrasekaran, and B. S. Manjunath. Steganalysis of spread spectrum data hiding exploiting cover memory. In *Security, Steganography, and Watermarking of Multimedia Contents VII*, volume 5681, pages 38–46. International Society for Optics and Photonics, 2005. 2

[27] S. Zheng, Y. Song, T. Leung, and I. Goodfellow. Improving the robustness of deep neural networks via stability training. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 4480–4488, 2016. 1

Acknowledgements

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Agreement No. HR00111990080. The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

Author Biography

Michael Goebel *received his B.S. and M.S. degrees in Electrical Engineering from Binghamton University in 2016 and 2017. He is currently a PhD student in Electrical Engineering at University of California Santa Barbara.*

Jason Bunk *received his B.S. degree Computational Physics from the University of California, San Diego in 2015, and his M.S. degree in Electrical and Computer Engineering from the University of California, Santa Barbara in 2016. He is currently a Research Staff Member at Mayachitra Inc., Santa Barbara, CA. His recent research efforts include adversarial attacks and defenses, applying deep learning techniques to media forensics, and active learning with neural networks for video activity detection.*

Srinjoy Chattopadhyay *received the B.Tech. degree in electronics and electrical communication engineering and the M.Tech. degree in telecommunication systems engineering from IIT Kharagpur, India, in 2013, and the Ph.D. degree from the Department of Electrical and Computer Engineering, North Carolina State University, USA in 2020. He is currently a Research Staff Member at Mayachitra Inc., Santa Barbara, CA. His current research interests include wireless communications, computer networks, spectral theory of graphs, and network design problems on multilayer interdependent networks*

Lakshmanan Nataraj *received his B.E degree from Sri Venkateswara College of Engineering, Anna university in 2007, and the Ph.D. degree in the Electrical and Computer Engineering from the University of California, Santa Barbara in 2015. He is currently a Senior Research Staff Member at Mayachitra Inc., Santa Barbara, CA. His research interests include malware analysis and image forensics.*

Shivkumar Chandrasekaran *received his Ph.D. degree in Computer Science from Yale University, New Haven, CT, in 1994. He is a Professor in the Electrical and Computer Engineering Department, University of California, Santa Barbara. His research interests are in Computational Mathematics*

B. S. Manjunath *received the Ph.D. degree in Electrical Engineering from the University of Southern California in 1991. He is currently a Distinguished Professor at the ECE Department at the University of California at Santa Barbara. He has coauthored about 300 peer-reviewed articles. His current research interests include image processing, computer vision and biomedical image analysis.*

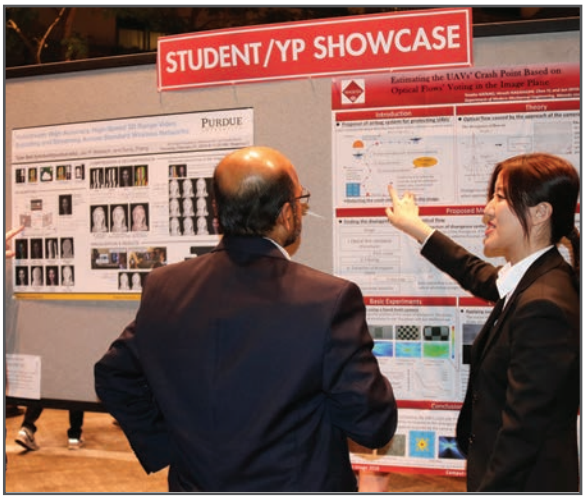
JOIN US AT THE NEXT EI!

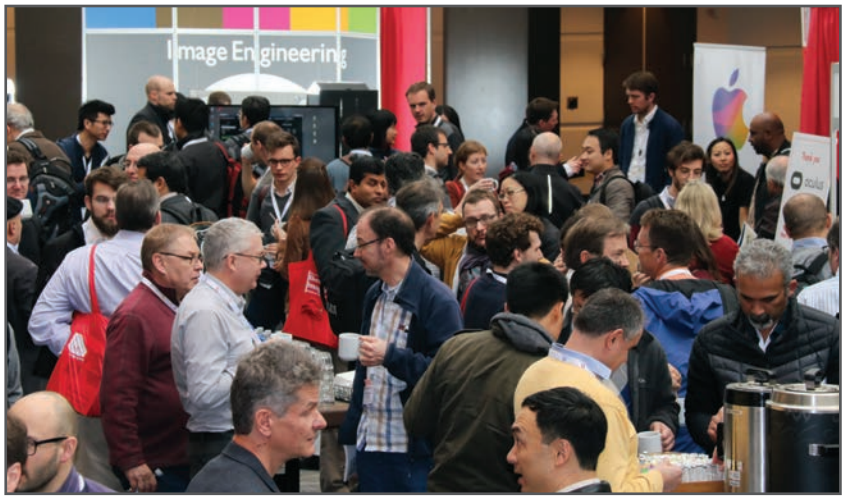
IS&T International Symposium on

Electronic Imaging

SCIENCE AND TECHNOLOGY

Imaging across applications . . . Where industry and academia meet!





• SHORT COURSES • EXHIBITS • DEMONSTRATION SESSION • PLENARY TALKS •
• INTERACTIVE PAPER SESSION • SPECIAL EVENTS • TECHNICAL SESSIONS •

www.electronicimaging.org

IS&T
imaging.org