Enablers Of Adversarial Attacks in Machine Learning

Rauf Izmailov
Vencore Labs
Basking Ridge, NJ
rizmailov@vencorelabs.com

Shridatt Sugrim
Vencore Labs
Basking Ridge, NJ
ssugrim@vencorelabs.com

Ritu Chadha
Vencore Labs
Basking Ridge, NJ
rchadha@vencorelabs.com

Patrick McDaniel
The Pennsylvania State University
University Park, PA
mcdaniel@cse.psu.edu

Ananthram Swami
U.S. Army Research Laboratory
Adelphi, MD
ananthram.swami.civ@mail.mil

***Abstract*—The proliferation of machine learning (ML) and artificial intelligence (AI) systems for military and security applications creates substantial challenges for designing and deploying such mechanisms that would learn, adapt, reason and act with Dinky, Dirty, Dynamic, Deceptive, Distributed (D5) data. While Dinky and Dirty challenges have been extensively explored in ML theory, the Dynamic challenge has been a persistent problem in ML applications (when the statistical distribution of training data differs from that of test data). The most recent Deceptive challenge is a malicious distribution shift between training and test data that amplifies the effects of the Dynamic challenge to the complete breakdown of the ML algorithms. Using the MNIST dataset as a simple calibration example, we explore the following two questions: (1) What geometric and statistical characteristics of data distribution can be exploited by an adversary with a given magnitude of the attack? (2) What counter-measures can be used to protect the constructed decision rule (at the cost of somewhat decreased performance) against malicious distribution shift within a given magnitude of the attack? While not offering a complete solution to the problem, we collect and interpret obtained observations in a way that provides practical guidance for making more adversary-resistant choices in the design of ML algorithms.**

***Index Terms*—Supervised learning, support vector machines, kernel, data models, feature extraction, training, training data, classification algorithms, machine learning algorithms, learning systems, distribution functions, distortion, adversarial examples, neural networks.**

I. Introduction

Recent developments in the studies of security machine learning algorithms has demonstrated their fundamental vulnerability to adversarial examples. These adversarial examples are maliciously constructed inputs that only deviate a little from legitimately distributed data (i.e. by appearing unmodified to human observers) but yield clearly erroneous model outputs. The implications of this vulnerability for all applications relying on machine learning algorithms (especially applications such as detecting malware, detecting network intrusion, autonomous vehicle navigation or biometric authentication, etc.) are disastrous.

The vulnerability of machine learning models to adversarial examples has been demonstrated across all kinds of algorithms for a diverse set of applications. Biggio et al. demonstrated successful attacks on malware classifiers for PDF files [1] and showed vulnerabilities of machine learning training at the design phase [2]. Nelson et al. explored adversarial attacks for convex-inducing classifiers [3]. Papernot et al. implemented black-box attacks of remote image-classifiers (such as Amazon) [4], [5]. Moosavi et al. and Papernot et al. designed specialized adversarial mechanism for deep neural networks [6], [7]. Kurakin et al. and Sharif et al. demonstrated that adversarial examples can be created with physical world objects [8], in particular, for fooling face recognition software [9].

In order to address these ubiquitous vulnerabilities, a number of protective mechanisms for machine learning models based on iterative training of classification rules under the assumption that they can and will be attacked after deployment, were proposed: [10]–[13]. The ensuing "arms race" of developing new adversarial attacks and the corresponding mitigation mechanisms continue to draw active research efforts.

The sheer scope of these adversarial examples has led to more in-depth studies of the nature and enablers of the discovered vulnerabilities. Tramer et al. discovered special geometric structures of adversarial examples that enabled their *transferability* [14], i.e., the property of an adversarial example designed for one type of machine learning model to be, with a reasonable probability, an adversarial example for another type of machine learning model. Papernot et al. provided a more detailed study of transferability [7], and Goodfellow et al. singled out the linearity of neural networks as the primary cause of their susceptibility to adversarial attacks [15].

In this paper, we focus on the nature and enablers of the

This research was sponsored by the U.S. Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

adversarial attack vulnerabilities, while deliberately refraining from proposing specific mitigation mechanisms for adversarial attacks on machine learning algorithms. We believe that mitigations should be designed *after* the nature of these enablers is sufficiently understood. Specifically, we focus on investigating the enablers of adversarial examples in terms of the interplay of three fundamental elements of machine learning:

1) The type of machine learning algorithm model: We consider three main representatives of these types: a linear algorithm (linear SVM) and two non-linear algorithms (RBF kernel SVM and neural networks (NN)). Although most work on adversarial attacks concentrates only on NNs, we use a wider set of algorithms to explore the scope of transferability.
2) Data featurization: We consider the MNIST dataset [16], a typical example used in many papers. Although it is not very large and has relatively low dimensionality, when compared to data sets like ImageNet [17], we find it useful for observing key adversarial mechanisms described in the body of this paper. We use the 784 pixel features and several subsets of these features.
3) Adversarial attack magnitude: The maximum numerical value of feature modifications that an attacker can apply to available data.

Our goal in this paper is (a) to explore the effects of these fundamental elements on the efficiency of adversarial attacks, (b) to provide explanatory mechanisms for the observed effects, and (c) to formulate, based on the observed effects, several conjectures towards proactive featurization of training data and design of machine learning algorithms under the assumption of inevitable adversarial attacks.

As a result of our study, we identify some of the enablers, i.e., the properties of data distribution on the features of ML algorithms (data featurization) that can be exploited by adversarial attacks. These enablers have low variability of various characteristics of data distribution relative to the magnitude of the attack (e.g., low variability of all classes, low variability of any single class, low margin between the classes). These observations suggest what kind of mitigation approaches should be relatively promising: (a) selection and/or construction of features that are relatively unaffected by the expected magnitude of attacks, and (b) proactive geometric expansion of the training dataset for the relative reduction of the attack magnitude on the underlying data distribution.

The paper is organized in the following way. In Section II, we describe and justify our experiment settings, the dataset used in the experiments, the machine learning algorithms to be studied, and the performance metrics to be collected. In Section III, we describe our experiments and the obtained results. While doing so, we make conjectures about the underlying causes of the observed effects. In Section IV we summarize collected observations and make specific recommendations about machine learning algorithm design to make these algorithms more robust to adversarial attacks. Finally, in Section V, we outline next steps in this research.

II. Experiment Settings

The effect of adversarial examples in machine learning is based on the violation of a fundamental assumption of machine learning itself (i.e., the assumption that test data are drawn from the same statistical distribution as the training data [18]). By engendering malicious distribution shifts from training data (where the algorithm was trained) to test data, adversarial examples confuse almost any well-trained classifier. While an extensive body of literature [19], [20] exists on methods of handling natural low-dimensional and geometrically simple distribution shifts by various mechanisms, they are clearly unable to handle malicious data modifications. Given the fundamental nature of distribution shift as the root cause of adversarial examples, it is reasonable to assume that, if we understand well enough how this distribution shift can be realized, the corresponding general mitigation mechanisms should be applicable across diverse applications that could be vulnerable to such attacks.

As mentioned in Section I, we carried out all our experiments on MNIST dataset [16], consisting of 10 classes of digits (from 0 to 9), rendered in 28*28 pixel images. This dataset has served as a calibration set for a significant number of adversarial examples and methods to protect machine learning models from these adversarial examples. In order to isolate the issue of distribution shift from other effects, we further restrict the scope of our decision rule to the binary classification task of distinguishing between two somewhat similar digits "3" and "8". We expect that the distribution of pixel values between these two digits to be very similar. Thus our binary classification problem models the common case of deciding between two very similarly distributed random variables. Such problems often arise in disciplines ranging from Stochastic signal processing to medical treatment evaluations.

On MNIST dataset, we train and test two SVM algorithms (with linear kernel and with RBF kernel) and one NN algorithm, using the Scikit-learn [21] implementations. They represent two distinct types of algorithms (linear and nonlinear), which allow us to measure the effects of the white-box attacks (i.e., the adversarial examples constructed with complete knowledge of the targeted algorithms) and assess the transferability of the adversarial examples across the algorithms (e.g., the increase of error rates observed for RBF SVM when classifying adversarial examples that were constructed to attack a NN, and vice versa).

Finally, given the common understanding that adversarial examples significantly alter the classification performance of machine learning algorithms with relatively low distortion of data that remain perceptually identical (from the viewpoint of human), we vary the level of distortion (which we call *magnitude*, M) of the attack from zero (there is no attack whatsoever, i.e., the base case) up to 50% of the individual pixel variation (in the test set). This pixel-level bound imposes a perturbation budget for any adversarial example.

Since we have three ML algorithms, we compute and plot nine performance metrics (error rates) for different values of attack magnitude and different methods of feature selection. These nine metrics measure the classification error rate when an adversarial example is classified by one of the three algorithms. For each algorithm we evaluate two cases: adversarial examples created by using the score function of the given classifier (white-box attack) and adversarial examples constructed for one of the other classifiers (black-box attack).

For all cases, the adversarial examples were constructed using an approach similar to the standard FGSM algorithm [15]. Since our method of generating adversarial examples follows a white-box attacker, we assume the attacker has access to the score function $S(x) \in \mathbb{R}$ of the classifier, where x is a 784-dimensional vector and each component represents the color depth at that pixel index. For the SVM algorithms, the score function is part of the algorithm's decision computation for an input vector. In the case of NNs, the score was derived from the class probabilities for each input vector. White-box attackers represent the most capable attackers and are thus a worst case scenario. Since we are preforming a binary classification task, the decision is made by computing the score of a scaled input vector and then classifying it based on the sign of that score.

Since the variation across features was not uniform, scaling was necessary to ensure good classifier performance in the base case [22]. The standard scaler from the Scikit-learn [21] was used. It pre-processes features by removing their mean and scaling them to unit variance. After scaling, the Linear classifier's error rate was $\approx 3\%$, and the error rates of RBF and NN classifiers' were $\approx 1\%$.

In order to generate an adversarial example from an image in test set, we use the FGSM implementation from [23], modified to study the relation between the magnitude of the attack and its effect on performance. In our modification, we pick a feature at random, increase or decrease it by a small value, then recompute the score. If the resulting score is closer to opposite sign (e.g., positives scores decrease bringing them closer to 0), the change is kept. This process is repeated until the sign of the score inverts. The modifications are stored in vector of changes η; thus the adversarial example takes the form $x + \eta$. If the original x has a positive score $S(x) > 0$, the vector η is constructed so that $S(x + \eta) < 0$, and vice versa for the negative score case.

To enforce the magnitude bound, each feature (indexed by $i \in [0, 783]$) is allowed to vary by at most M percent of the image's current value, thus $|x_i * (1 - M)| \leq (|x| + \eta)_i \leq |x_i * (1 + M)|$ for every i. As the iterations of the process progresses, the set of possible features available for modification shrinks. If the budget of allowable modifications is exhausted before the sign of the score has inverted, the process terminates in failure.

We generate adversarial examples for every x in the test set via this iterative procedure with score functions S of our algorithms. We then compute the classification error for the nine cases. While this method does not exactly replicate the FGSM attack, it nevertheless produces examples that cause all classifiers to fail. Since FGSM follows the proper gradient path, instead of randomly traversing the loss function, it produces adversarial examples that are more specific to the classifier being attacked. Our modification of FGSM is built to be less classifier-specific.

III. Experiments and Observations

Our first group of experiments was carried out with the original MNIST dataset, containing 28*28=784 pixels. The left part of Figure 1 shows the results, while the right part of the figure shows the composite average of all the images "3" and "8" in the training data. As it illustrates, the error rates (vertical axis) for almost all combinations of attacks and algorithms predictably increase with the magnitude of the attack (horizontal axis).

Figure 1 also shows that the linear adversarial examples have little impact on the RBF SVM or NN classifiers. Although this seems to contradict the well-established property of transferability of adversarial examples across different ML algorithms, this failure is actually a simple result of the fact that Linear SVM requires significantly *less* pixel changes to produce an adversarial example than are required for either RBF SVM or NN. The adversarial examples for each of our classifiers differ only in how much noise is required to drive the score function to a value that causes the decision to invert. The Linear model's score function on a large feature space is so fragile that it is very easy to drive the score across the threshold with a small amount of noise. Conversely, RBF SVM and NN require much more noise (i.e., more of the perturbation budget) before the classification decision inverts in the large feature space. This observation explains why the RBF SVM examples do not significantly impact the NN; the noise that is being added to the RBF SVM adversarial examples may not be sufficient to influence NN's decision.

We further demonstrate that, when the size of the feature space is reduced (e.g., by applying a mask to the image), the fragility of the linear score function reduces commensurately. Because of this reduction in fragility the impact of the noise is significantly reduced, even for noise levels that corrupt the non-linear classifiers. This observation will be a key factor in some of the following results. In the smaller feature space, more complex non-linear models become *more* fragile in their scoring. This fragility immediately translates to vulnerability to adversarial examples.

For the full feature space, the linear classifier's median number of iterations to generate an adversarial example was 903 across all magnitude cases. In contrast, the median number of iterations for an RBF SVM classifier was 1699 and NN was 1911. While several examples can vary significantly from this median, the general trend is that the RBF SVM and NN classifiers require significantly more changes to produce an adversarial example than the linear because the score function being manipulated is much more complex.

In order to visualize the nature of distribution shift, we computed, for each of the 784 features, the average difference between the original MNIST images and their adversarial modification; the results for both types of SVM and NN are shown in Figure 2. As the figure illustrates, there are several

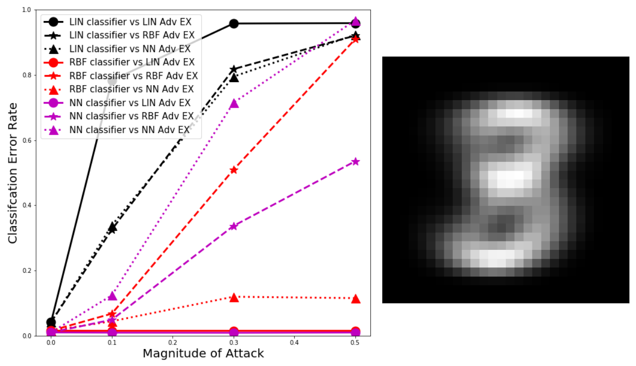

Fig. 1. Adversarial attacks for the full set of MNIST features.

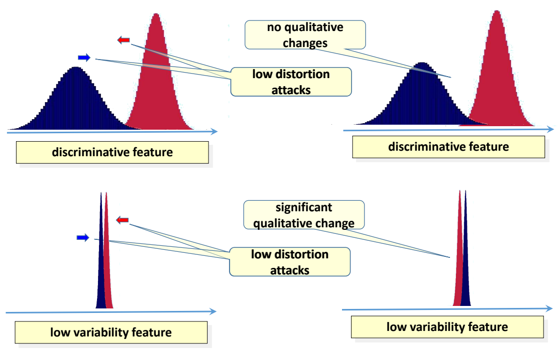

Fig. 3. Low-variability features as enablers of adversarial distribution shift.

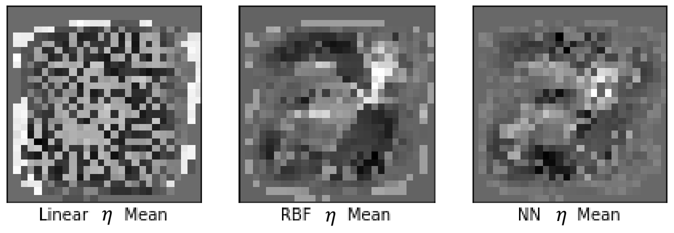

Fig. 2. Distribution shift for the full set of MNIST features.

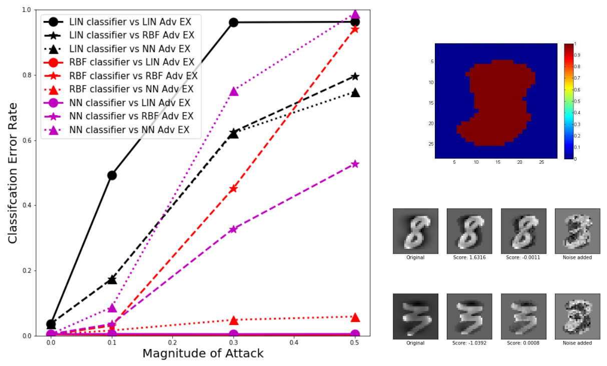

Fig. 4. Adversarial attacks for the high-variability subset of MNIST features.

of features that vary significantly in the peripheral areas of the image, although there are very few informative pixels in those areas (see the right side of Figure 1).

As Figure 3 illustrates, the low-variability features can be actively leveraged in adversarial attacks. These features would typically decrease the performance of the constructed decision rule if they were not pruned by feature selection. For example, the linear classification algorithm would likely assign non-zero weights for many of those features. In the absence of distribution shift, this assignment would not matter much, since such features in aggregate would make an almost constant additive contribution to the constructed decision function, and this additive factor would be incorporated into the offset term. However, the same non-zero weights of these low-variability features can now be actively leveraged by the adversary towards moving the value of the score function in the wrong direction.

The obvious solution to this problem is to diligently remove all low-variability features and rerun the experiments. This procedure would be reasonable as an act of feature selection in any case. The results are shown in the left part of Figure 4. The right part of the figure shows the selected set of high-variability features as a red mask consisting of 298 features, and the effects of that feature selection mask on the original images. The similarity of the error rates for all nine scenarios shown in Figure 1 suggests that the simple act of removing low-variability features, although reasonable, helps very little in terms of mitigation of adversarial attacks.

However, the role of individual features in the construction of adversarial examples suggests that next step should be more aggressive feature selection, where all features with low mutual information are removed as well. Indeed, as Figure 5 illustrates, the low mutual information features (with high overlap between the projections of class distributions on those features) are naturally much more susceptible to adversarial attacks that can further reduce and exploit the low information value of these features.

Given this observation, we conducted another step of feature selection for the given training data where we used the mutual information of individual features to decide which features are to be pruned. This step reduced the set of already selected

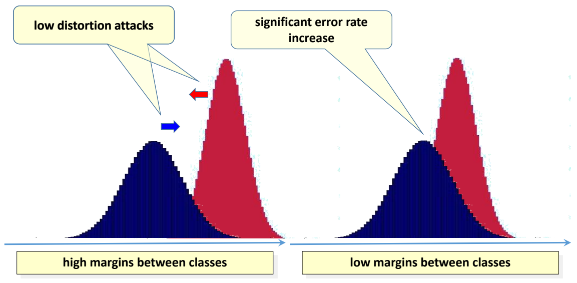
low distortion attacks
significant error rate increase
high margins between classes
low margins between classes

Fig. 5. Low mutual information features as enablers of adversarial distribution shift.

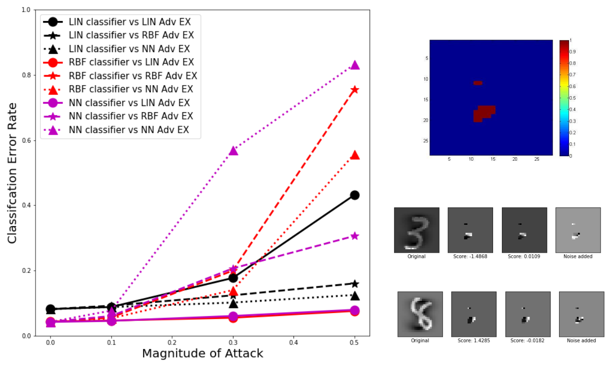

Fig. 6. Adversarial attacks for the high-variability and high mutual information subset of MNIST features.

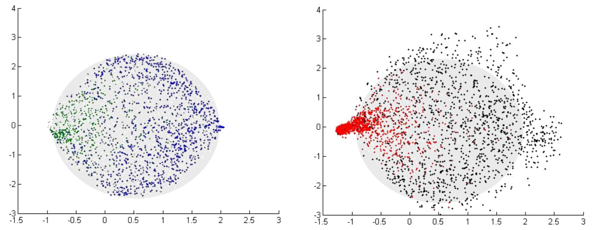

Fig. 7. Class distribution and vulnerabilities to adversarial attacks.

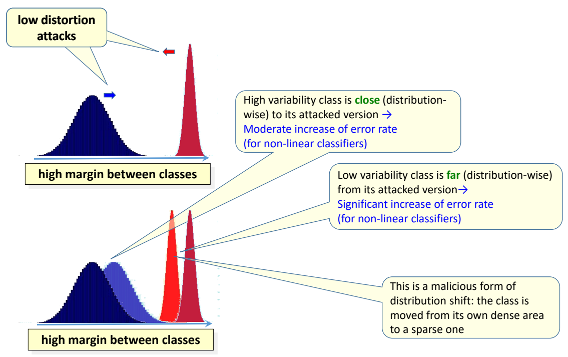
low distortion attacks
high margin between classes
High variability class is close (distribution-wise) to its attacked version →
Moderate increase of error rate (for non-linear classifiers)
Low variability class is far (distribution-wise) from its attacked version→
Significant increase of error rate (for non-linear classifiers)
This is a malicious form of distribution shift: the class is moved from its own dense area to a sparse one
high margin between classes

Fig. 8. Low variability class distributions as enablers of adversarial distribution shift.

298 feature to 17 features shown (along with their effect on original images) in the right part of Figure 6. The left part of Figure 6 illustrates the effects of adversarial attacks on this 17-dimensional space of features. As the figure demonstrates, the result of our more diligent feature selection is the substantial decrease of error rates in the attacked scenarios. The figure also shows that linear algorithm, while being a little worse than nonlinear algorithm in the default (non-attacked) case, exhibits much more relative resistance to adversarial examples as the magnitude of attacks increases.

In order to visualize what keeps enabling adversarial attacks after our substantial pruning of vulnerable features, consider Figure 7. The left part shows the default (non-attacked) distribution of data in the main 2-dimensional subspace after a PCA rotation in its 17-dimensional space (digits “3” are shown in green, and digits “8” in blue). The right part of Figure 7 shows the attacked distribution of data (corresponding to the magnitude 0.5) in the same 2-dimensional subspace (digits “3” are shown in red, and digits “8” in black).

We see that, although both original distributions for digits “3” and “8” are changed by the attacks, their *relative* changes are quite different. The distribution for digits “3” is affected significantly more than the distribution for digits “8”. This difference between the scope of changes of the class distributions is directly reflected in the relative misclassification error rates for these digits. For digits “8”, the analyzed adversarial examples increase the default error rates from 6.0% to 55%, whereas, for digits “3”, the same kind of adversarial examples increase the default error rates from 3.4% to 93%.

The reason for this skewed effect of adversarial examples is illustrated in Figure 8: even with high variability of data projection on the given feature and its high mutual information (i.e., the margin between the classes), a small adversarial shift can significantly increase the statistical distance between the original and attacked distributions if this distribution itself has low variability relative to the magnitude of the attack.

We can now identify yet another enabler of adversarial attacks on our 17-dimensional space of features, namely, low variability of individual class distributions, as illustrated in Figure 7. The asymmetry of class variation on each feature is due to the differences in how each class uses a specific feature. For example, some features (such as the center pixels, or the right side) of an ideal drawing of a three or eight receive the same amount of intensity when drawn. Thus the numeric values of those features will often be the same on average. However, the left side is not equally utilized by three vs. eight. This causes the asymmetry seen in the mean image shown in Figure 1. Thus, there will be significant difference in the variability of values taken on by these features when these values come from the class “3” versus the class “8”.

IV. Discussion

The experiments and observations described in the previous section demonstrate how data featurization can enable various vulnerabilities that can be successfully leveraged for creating adversarial examples. The lessons learned from these experiments are consistent: it is the low variability (relative to the magnitude of the attack) of various characteristics of data distribution (low variability of all classes, low variability of any single class, low margin between the classes) in the given feature space that appears to be one of the key enablers for adversarial examples. The same lessons suggest what kind of mitigation approaches should be relatively promising: (a) selection and/or construction of features that are relatively unaffected by the expected magnitude of attacks, and (b) proactive geometric expansion of the training dataset for the relative reduction of the attack magnitude on the underlying data distribution.

Despite the seemingly narrow scope of the considered problem (binary classification for two digits in MNIST), because of the general nature of adversarial examples, we believe that lessons learned for this case can not only be transferred to other image classification tasks (the features of MNIST, being individual black-and-white pixels exhibit similar behavior to other types of image datasets), but also to other types of applications as well. In particular, they should be transferrable to cyber security applications [24] with proper definitions of the corresponding data and features. Specifically, as opposed to readily available pixels in MNIST and other visual datasets, cyber security applications are based on fundamentally different types of features that have to be "extracted" from aggregated network artifacts, e.g., features such as flow statistics derived from a collection of packets, while more fine-granularity features can be based on individual packets.

The data in the cyber security domain will come from several potential sources, the most important being from the network and the hosts (in addition, external intelligence is expected to play a crucial role). For instance, (a) call sequences could be main features for host-based IDS systems [25]; (b) n-gram sequences extracted from the executable file and information from different sections of PE binaries (such as header, import and export) and strings encoded in program files could be main features for malware detection [26], and (c) various flow statistics (such as number of bytes and/or packets per flow), flow duration, average payload packet length, average number of packets/bytes per second and average time between requests could be main features for network-based IDS such as botnets [26], [27]. We expect that development and selection of all these features (to be addressed in our future work) will be based on the lessons learned from our current work on MNIST images.

V. Conclusion

We have identified several fundamentally related enablers of adversarial examples for machine learning algorithms. Although observed for a simple calibration dataset (MNIST), these enablers appear to be applicable to a wide variety of other applications of machine learning. In subsequent research, we will investigate the following within the context of cyber security applications: (1) the scope of already identified enablers and, potentially, the existence of new enablers, (2) data featurization techniques mitigating the effects of adversarial examples, and (3) proactive modifications of machine learning algorithms based on those data featurization techniques.

References

[1] B. Biggio, I. Corona, D. Maiorca, B. Nelson, P. Laskov, F. Roli, G. Giacinto, and N. Srndi, "Evasion attacks against machine learning at test time," in Machine Learning and Knowledge Discovery in Databases, Springer, 2013, pp. 387-402.

[2] B. Biggio, G. Fumera, and F. Roli. "Security evaluation of pattern classifiers under attack," IEEE Transactions on Knowledge and Data Engineering, 26(4):98417996, 2014.

[3] B. Nelson, B. Rubinstein, L. Huang, A. Joseph, S. Lee, S. Rao, and J. Tygar. "Query strategies for evading convex-inducing classifiers," J. Mach. Learn. Res., 13:1293171332, 2012.

[4] N. Papernot, P. D. McDaniel, and I. J. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," arXiv preprint arXiv:1605.07277, 2016.

[5] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in ACM Asia Conference on Computer and Communications Security, New York, NY, USA, 2017.

[6] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: a simple and accurate method to fool deep neural networks," arXiv preprint arXiv:1511.04599, 2015.

[7] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in IEEE European Symposium on Security and Privacy, 2016.

[8] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," arXiv preprint arXiv:1607.02533, 2016.

[9] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a crime: real and stealthy attacks on state-of-the-art face recognition," in ACM SIGSAC Conference on Computer and Communications Security, New York, NY, USA, 2016.

[10] N. Srndic and P. Laskov, "Practical evasion of a learning-based classifier: a case study," in IEEE Symposium on Security and Privacy, Washington, DC, USA, 2014.

[11] A. Kantchelian, J. Tygar, and A. Joseph, "Evasion and hardening of tree ensemble classifiers," arXiv pre-print, 2015.

[12] B.Li and Y. Vorobeychik,"Feature cross-substitution in adversarial classification," Proceedings of the 27th International Conference on Neural Information Processing Systems. NIPS'14. Cambridge, MA, USA: MIT Press: 2087172095.

[13] B. Li, Y. Vorobeychik, and X. Chen, "A general retraining framework for scalable adversarial classification," arXiv preprint arXiv:1604.02606, 2016.

[14] F. Tramer, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "The space of transferable adversarial examples," arXiv preprint arXiv:1704.03453, 2017.

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

[16] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, 1998, pp. 2278–2324.

[17] J. Deng at al.,"ImageNet: a large-scale hierarchical image database," 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL

[18] V. Vapnik, Statistical Learning Theory. Wiley-Interscience, 1998.

[19] M. Sugiyama and M. Kawanabe, Machine Learning in Non-Stationary Environments: Introduction to Covariate Shift Adaptation. The MIT Press, 2012.

[20] Dataset Shift in Machine Learning. The MIT Press, 2008.

[21] F. Pedregosa et al., "Scikit-learn: machine learning in Python," Journal of Machine Learning Research, 2011.

[22] C. Cortes and V. Vapnik, "Support-vector networks," Machine Learning, volume 2, 1995, pp. 273–297.

[23] N. Papernot, I. Goodfellow, R. Sheatsley, R. Feinman, and P. McDaniel, "cleverhans v1.0.0: an adversarial machine learning library", arXiv preprint arXiv:1610.00768, 2016.

[24] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: a survey," ACM Computing Surveys, Vol. 41, No. 3, July 2009.

[25] K. Rieck, P. Trinius, C. Willems, and T. Holz, "Automatic analysis of malware behavior using machine learning," Journal of Computer Security, Vol. 19, No 4, 2011, pp. 639-668.

[26] A. Shabtai, R. Moskovitch, Y. Elovici, and C. Glezer, "Detection of malicious code by applying machine learning classifiers on static features: a state-of-the-art survey," Information Security Technical Report, Vol. 14, No. 1, 2009, pp. 16–29.

[27] A. Sapello, C. Serban, R. Chadha, and R. Izmailov, "Application of learning using privileged information (LUPI): botnet detection," International Conference on Computer Communication and Networks (ICCCN), 2017.