Adversarial Reprogramming of Pretrained Neural Networks for Fraud Detection

Lingwei Chen
Pennsylvania State University
State College, PA, USA
lgchen@mix.wvu.edu

Yujie Fan
Case Western Reserve University
Cleveland, OH, USA
yxf370@case.edu

Yanfang Ye
Case Western Reserve University
Cleveland, OH, USA
yanfang.ye@case.edu

ABSTRACT

Machine learning models have been widely used for fraud detection, while developing and maintaining these models often suffers from significant limitations in terms of training data scarcity and constrained resources. To address these issues, in this paper, we leverage the machine learning vulnerability to adversarial attacks, and design a novel model *AdvRFD* that **Adv**ersarially **R**eprograms an ImageNet classification neural network to perform **F**raud **D**etection task. Specifically, AdvRFD first embeds transaction features into a host image to construct new ImageNet data, and then learns a universal perturbation to be added to all inputs, such that the outputs of the pretrained model can be accordingly mapped to the final detection decisions for all source transactions. Extensive experiments on two transaction datasets made over Ethereum and credit cards have demonstrated that AdvRFD is effective to detect fraud using limited data and resources.

CCS CONCEPTS

• **Security and privacy** → *Web application security*; • **Computing methodologies** → *Neural networks.*

KEYWORDS

Fraud detection, Adversarial reprogramming, Perturbation

ACM Reference Format:
Lingwei Chen, Yujie Fan, and Yanfang Ye. 2021. Adversarial Reprogramming of Pretrained Neural Networks for Fraud Detection. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3459637.3482053

1 INTRODUCTION

In the e-commerce age, financial transactions have shot up manifolds, and so have cases of fraud. The Consumer Sentinel Network maintained by Federal Trade Commission (FTC) received 3.2 million reports of identity theft and online fraud in 2019, causing huge economic losses and damages [25]. Given such a large amount of fraud and sophistication of fraudsters adept at exploiting loopholes in systems, machine learning has been increasingly applied over transactions to detect new fraud earlier than rule-based approaches and thus alleviate the evolving threats it poses [4, 7, 12, 14, 20]. While many finance and security companies successfully deploy machine learning models in their fraud detection services, developing and maintaining these models often suffers from significant limitations in terms of training data scarcity for subtle feature learning, and constrained computing resources for deployment [5]. With this in mind, a learning-effective yet cost-efficient machine learning model may be in need to address these issues.

Due to the inherent machine learning vulnerability to adversarial attacks [2, 8, 13, 17], adversarial reprogramming [6] has been recently proposed to repurpose a machine learning model trained in a source domain to perform a target-domain task through learning a universal perturbation to the target-domain data without modifying the source-domain model parameters, where the domains and tasks can be completely different. Different from traditional adversarial attacks exploiting model's linear approximations to cause high misclassification rate, adversarial reprogramming relies on the nonlinear interactions of the input and the perturbation, which can be satisfied by a source-domain model of nonlinear deep structure [18]. In other words, an additive offset to a deep neural network's input would be sufficient on its own to repurpose the network to a new task without the need of model fine-tuning [6]. As such, adversarial reprogramming of pretrained neural networks yields the prospective advantages for our fraud detection problem: (1) it can leverage the powerful learning capabilities of deep neural networks to extract expressive patterns from subtle transaction features with much less effort than training from scratch or transfer learning, and (2) make the source-domain model resources reusable for shared compute from different fraud detection tasks and benefit the application scenarios with limited resources.

In this paper, we explore a novel perspective of fraud detection, and present a model *AdvRFD* to adversarially reprogram an ImageNet classification neural network for fraud detection. On one hand, ImageNet classification neural networks have been undergoing vibrant evolution in their nonlinear deep structures and learning capabilities; on the other hand, a wide range of such high-performance pretrained networks can be easily available for straightforward leverages. Given a pretrained model of this kind and a host image randomly selected from ImageNet, AdvRFD first injects features extracted from transactions into the host image to construct new image data, and then learns a universal perturbation to be added to all inputs, such that the outputs of the pretrained model can be accordingly mapped to the final detection decisions (i.e., valid or fraudulent) for all source transactions. This naturally leads to the following two goals for AdvRFD: (1) input transformation to embed transaction features and perturbation into the host image for the adversarial task, and (2) output transformation to

CIKM '21, November 1–5, 2021, Virtual Event, QLD, Australia

ACM ISBN 978-1-4503-8446-9/21/11…$15.00
https://doi.org/10.1145/3459637.3482053

map ImageNet classes to fraud detection classes. Different
recent works [6, 16, 26], we conceal the visibility of features
perturbation for general applications, and investigate the impa
different output mapping methods on detection performance

2 BACKGROUND AND RELATED WORK

2.1 Machine Learning-based Fraud Detectio

A fraud detection model using machine learning attempts to ide
fraud through building a classification model based on the lal
training transactions and feature representations, such as ge
tive adversarial networks (GAN) [7], hybrid ensemble [12, 20]
graph neural network (GNN) [14]. Without loss of generality, we define transactions as $\mathcal{D}$, and denote feature extraction with the function $\phi : \mathcal{D} \rightarrow \mathcal{X}$, where $\mathcal{X} \subseteq \mathbb{R}^k$ is the feature space. The fraud detection can be thus stated in the form of $F : \mathcal{X} \rightarrow \mathcal{Y}$, which outputs confidence scores in the class space $\mathcal{Y} = \{-1, +1\}$ representing the valid and fraudulent classes respectively, such that the predicted class of sample $\mathbf{x} \in \mathcal{X}$ is derived from

$$y^* = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} F_y(\mathbf{x}) \tag{1}$$

where $F_y(\mathbf{x})$ is the confidence score of $y \in \mathcal{Y}$ class.

2.2 Adversarial Reprogramming

In general, adversarial attacks apply different perturbations to different inputs [8, 24], while recently, universal perturbation has been explored to be added on different inputs to craft adversarial examples [1, 9, 15]. Similarly, adversarial reprogramming repurposes a model for a new task by learning a universal perturbation to the inputs [16]. Given a model performing a source-domain task, for inputs $\tilde{\mathbf{x}}$ it produces outputs $S(\tilde{\mathbf{x}})$. An attacker would like to perform a target-domain task, for inputs $\mathbf{x}$ it computes outputs $T(\mathbf{x})$. Adversarial reprogramming proceeds by learning transformation functions $H_s(\cdot; \boldsymbol{\theta})$ and $H_t(\cdot)$ that map between the two tasks, where H_s transforms target-domain inputs $\mathbf{x}$ into source-domain inputs $\tilde{\mathbf{x}} = H_s(\mathbf{x}; \boldsymbol{\theta})$, while H_t maps source-domain outputs $S(H_s(\mathbf{x}; \boldsymbol{\theta}))$ back to target-domain outputs $T(\mathbf{x})$, such that

$$T(\mathbf{x}) = H_t(S(H_s(\mathbf{x}; \boldsymbol{\theta}))) \tag{2}$$

where $\boldsymbol{\theta}$ is the trainable universal perturbation. Unlike transfer learning [3, 19], adversarial reprogramming keeps model parameters unchanged and simply learns $\boldsymbol{\theta}$ to perform adversarial task.

3 PROPOSED MODEL

3.1 Overview of AdvRFD

In our work, we consider that we have access to a self-deployed ImageNet classification neural network, and leverage adversarial reprogramming of this model for fraud detection. As such, the source-domain task is ImageNet classification, and the target-domain task is fraud detection. Accordingly, we define $\mathbf{x}$ to be transaction features, $T(\mathbf{x})$ a fraud detection function, $\tilde{\mathbf{x}}$ ImageNet images, and $S(\tilde{\mathbf{x}})$ a pretrained ImageNet model. Based on these definitions, the input transformation function $H_s(\cdot; \boldsymbol{\theta})$ then comprises embedding $\mathbf{x}$ to a host ImageNet image to formulate new image data $\tilde{\mathbf{x}}$, and learning a universal perturbation $\boldsymbol{\theta}$ to be added to $\tilde{\mathbf{x}}$, while the output transformation function $H_t(\cdot)$ further maps ImageNet classes to fraud

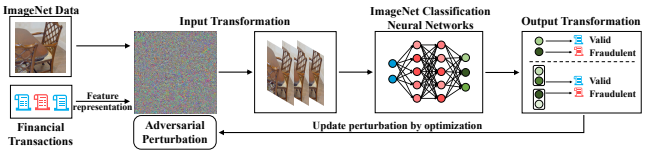

Figure 1: The overview of our proposed model AdvRFD.

detection classes. Given training transactions, AdvRFD proceeds with input and output transformations, while the perturbation $\boldsymbol{\theta}$ is updated via evaluating the loss between the output and the input's ground truth. The overview of AdvRFD is illustrated in Fig. 1.

3.2 Input Transformation

Input transformation is to convert the transaction data to the input space of ImageNet classification, which includes transaction feature embedding and perturbation formulation. Since we consider that we have access to the neural networks that would be reprogrammed, the distribution of the features and the magnitude of this perturbation need not be constrained for adversarial reprogramming to work. However, the idea is more general here: AdvRFD could repurpose any self-deployed or public image classification services, while those real-life machine learning services may be aware of the potential attacks to avoid computational resource abuse. As such, similar to most previous adversarial attacks, we would like to construct our adversarial reprogramming that is imperceptible, such that the visibility of transaction features and adversarial perturbation is concealed to the deployed models.

3.2.1 Transaction feature embedding. Generally, each transaction $\mathbf{x} \in \mathcal{X}$ is represented by a k-dimensional feature vector:

$$\mathbf{x} = < x_1, x_2, x_3, \cdots, x_k > \tag{3}$$

where $\mathbf{x} \in \mathbb{R}^k$ and the value of x_i depends on the possible features derived from transactions, such as transaction number, value, time, purchase data, account information, or customer behaviors. x_i may thus correspond to either 0/1 binary values indicating the absence or presence of one feature, or numeric values specifying the size, degree, or difference of feature variables. Considering that each pixel value for images from ImageNet data is initially scaled to $[0, 1]$ before being further transformed, x_i of numeric value needs to be normalized in the range of $[0, 1]$ to reasonably perform transaction feature embedding over ImageNet Images.

Different from recent works [6, 26] that simply placed the target-domain data in the center of the images padded by perturbation, we scatter transaction features in a host image $\tilde{\mathbf{x}} \in \mathbb{R}^{n \times n \times 3}$ to conceal their visibility. More importantly, as high-frequency areas in images like noises and edges rapidly change in space, we further extract pixels in such areas to host features and enhance the invisibility of feature embedding. Let $\mathbf{C} \in \mathbb{R}^{n \times n \times 3}$ mark the high-frequency areas in the host image $\tilde{\mathbf{x}}$ by zeroing out the pixels in the low-frequency areas (e.g., using Discrete Wavelet Transform (DWT) [22]). k pixels are then randomly selected to store k feature values from $\mathbf{x}$, i.e., $\tilde{\mathbf{x}}^k_{\mathbf{C}} \oplus \mathbf{x}$, where $\oplus$ implies feature embedding operation; a mask $\mathbf{M} \in \mathbb{R}^{n \times n \times 3}$ is accordingly formulated with 0 for k feature locations and 1 for others to avoid transaction features being perturbed in

$\tilde{\mathbf{x}}$. Note that, once k pixels are designated, pixel i will be firmly associated with the same feature x_i for all data samples.

3.2.2 Perturbation formulation. The perturbation in adversarial reprogramming is formulated not specific to a single image, but to be added to all input images [6], which can be defined as:

$$\tilde{\boldsymbol{\theta}} = \epsilon \cdot \tanh(\boldsymbol{\theta} \odot \mathbf{M}) \tag{4}$$

where $\boldsymbol{\theta} \in \mathbb{R}^{n \times n \times 3}$ is the universal perturbation to be learned, $\mathbf{M}$ is a mask as defined above, $\odot$ denotes the element-wise product, and $\tanh(\cdot)$ bounds the perturbation to be in $(-1, 1)$. Here we also introduce a hyper-parameter ϵ to govern the magnitude of the perturbation. The input transformation can thus be presented as:

$$\tilde{\mathbf{x}} = H_s(\mathbf{x}; \boldsymbol{\theta}) = \text{clip}(\tilde{\mathbf{x}}_{\mathbf{C}}^{k} \oplus \mathbf{x} + \epsilon \cdot \tanh(\boldsymbol{\theta} \odot \mathbf{M})) \tag{5}$$

where clip$(\tilde{\mathbf{x}})$ function performs per-pixel clipping of the image $\tilde{\mathbf{x}}$ to limit each pixel value to the range of $[0, 1]$. In our method, both the parameter ϵ and clip function contribute to the invisibility of the formulated perturbation.

3.3 Output Transformation

For the new input $\tilde{\mathbf{x}}$, the ImageNet classification neural network outputs $\tilde{y} = S(\tilde{\mathbf{x}})$ with $\tilde{y} \in \{0, 1, \cdots, 999\}$, while for the source transaction $\mathbf{x}$, the fraud detection function outputs $y = T(\mathbf{x})$ with $y \in \{0, 1\}$. To transform the ImageNet classes to fraud detection classes, we specify output transformation function $H_t(\cdot)$ using either hard coded mapping or soft coded mapping method to investigate their impacts on adversarial reprogramming effectiveness. Specifically, hard coded mapping simply assigns two randomly-selected class outputs of ImageNet to predict valid and fraudulent classes respectively. Soft coded mapping, similar to multi-label mapping [26], forms two class sets of ImageNet with the same number of non-repeated classes each, and averages the outputs for both sets to report fraud detection decisions respectively.

3.4 Optimization

The optimization of AdvRFD can be formulated as

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\text{argmax}}(-\log p(H_t(\tilde{y})|\tilde{\mathbf{x}}) + \lambda \|\boldsymbol{\theta}\|_F^2) \tag{6}$$

where $p(H_t(\tilde{y})|\tilde{\mathbf{x}})$ indicates the probability that an image input $\tilde{\mathbf{x}}$ transformed from transaction $\mathbf{x}$ being classified as $\tilde{y}$, which can be mapped to fraud; λ is the regularization parameter for a perturbation norm penalty to reduce overfitting. Eq. (6) shows that $\boldsymbol{\theta}$ is the only parameter for our problem, which can be updated by optimizing this loss using Adam. As the host image and transaction features are constants during reprogramming, feature embedding can be pre-processed to decrease training cost. Based on the trained model, we can easily perform the fraud detection that needs only input and output transformations, while leaving the majority of computation to the deployed ImageNet classification neural networks.

4 EXPERIMENTAL RESULTS AND ANALYSIS

4.1 Experimental Setup

We test AdvRFD on two transaction datasets made over Ethereum[1] (9,841 38-d transactions with 7,662 valid and 2,179 fraud) and credit cards[2] (1,012 30-d transactions with 520 valid and 492 fraud). We randomly select 80% of the samples for training, while the remaining is used for testing, and we report mean accuracy and F1-score of 5 runs for all experiments. The parameter setting is specified as 0.01 initial learning rate, and 5×10^{-4} L2 regularization on the perturbation. We evaluate the impact of ϵ and host images on fraud detection performance in Section 4.3.

We employ five pretrained ImageNet classification neural networks, including DenseNet-121 and DenseNet-161 [11], ResNet-50 and ResNet-101 [10], and Inception-V3 [23] for reprogramming. For baselines, we compare AdvRFD with Ensemble [12], GAN [7], GNN [14], and three ImageNet classification neural networks (using transfer learning with binary outputs).

4.2 Ablation Study on Output Transformation

We first perform an ablation study to decide the mapping method that better benefits AdvRFD. For hard coded mapping, we randomly choose two ImageNet classes; for soft coded mapping, we randomly divide 1,000 ImageNet classes into two sets. Based on this setting, we evaluate AdvRFD on Ethereum using Inception-V3 with $\epsilon = 0.3$. From Fig. 2 we can see that hard coded mapping slightly outperforms soft coded mapping on test accuracy and F1-score. When we look into training procedure, we find hard coded mapping reaches a stable loss at epoch 12, while soft coded mapping needs 15 epochs to achieve the comparable performance causing higher training time. Thus, hard coded mapping leads to faster and better convergence results, and we use it in AdvRFD for the following evaluation.

4.3 Evaluation of AdvRFD

4.3.1 Fraud detection effectiveness. In our experiments, we evaluate AdvRFD on Ethereum under different perturbation magnitudes ϵ and host images $\tilde{\mathbf{x}}$. In particular, we test the results of AdvRFD for fraud detection with $\epsilon \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$ respectively, while the host images $\tilde{\mathbf{x}}$ being randomly selected from ImageNet data, including dining table, rickshaw, spindle, tank and polecat. The experimental results are shown in Fig. 3. Though different parameter settings contribute to different test results, which will be further discussed in the following subsections, AdvRFD successfully achieves the goal of reprogramming five well-trained ImageNet classification neural networks to perform fraud detection. Averagely, our model obtains 95.96% test accuracy and 90.91% F1-score.

4.3.2 Impact of perturbation magnitudes. We investigate the sensitivity of perturbation magnitude ϵ on the performance of AdvRFD, which is illustrated in Fig. 3(a)-(b) (run on a host image of dining table). Generally, when we enlarge ϵ, the test effectiveness increases. We observe that the accuracy and F1-score rise to a high level at $\epsilon = 0.3$ for all pretrained networks and then either slightly increase or drop when ϵ changes from 0.3 to 0.5. Notably, the larger ϵ increases the risk of perturbation being recognized by the pretrained models. Thus, we use $\epsilon = 0.3$ throughout the following evaluations to keep a good trace-off between fraud detection effectiveness and perturbation invisibility for adversarial reprogramming.

4.3.3 Impact of host images. It is also interesting to see if different host images affect the performance of AdvRFD. Since Inception-V3

[1] https://www.kaggle.com/vagifa/ethereum-frauddetection-dataset

[2] https://www.kaggle.com/mlg-ulb/creditcardfraud

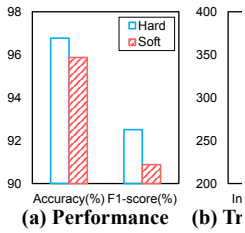

Figure 2: Output maı

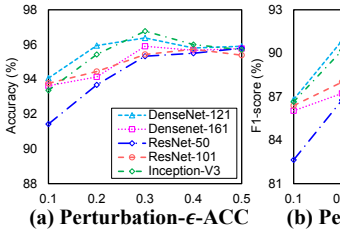

Figure 3: Evaluation with

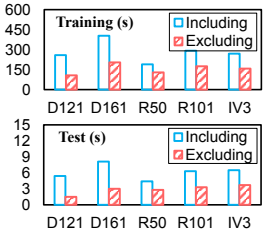

Figure 4: Cost efficiency.

Table 1: Comparisons of baselines (TR–training time)

Method	Pretrained	Ethereum			Credit Cards		
		AC (%)	F1 (%)	TR (s)	AC (%)	F1 (%)	TR (s)
Ensemble [12]	-	88.31	65.11	-	92.37	91.17	-
GAN [7]	-	91.63	86.88	-	93.41	92.50	-
GNN [14]	-	93.82	87.83	-	93.46	92.72	-
DenseNet-161 [11]	-	90.26	85.16	1,196	83.37	82.61	159
ResNet-101 [10]	-	90.74	85.63	825	82.81	81.46	95
Inception-V3 [23]	-	91.65	87.79	717	82.28	81.15	66
AdvRFD	DenseNet-121	96.38	91.92	259	93.64	92.87	22
	DenseNet-161	95.90	90.83	404	**94.09**	**93.13**	34
	ResNet-50	95.33	89.56	190	92.59	91.82	17
	ResNet-101	95.43	89.70	292	92.98	92.09	25
	Inception-V3	**96.77**	**92.75**	270	93.96	92.80	23

outperforms other pretrained models at $\epsilon = 0.3$, we report the test accuracy and F1-score of AdvRFD that repurposes Inception-V3 for fraud detection over five different host images selected from ImageNet. As shown in Fig. 3(c), the evaluation results slightly vary in different host images. However, the standard deviations of accuracy and F1-score fall within $[0.001, 0.003]$, which imply that the performance difference is not statistically significant and AdvRFD is loosely coupled with host images.

4.4 Comparisons with Baselines

We compare AdvRFD with baselines, including Ensemble [12], GAN [7], GNN [14], DenseNet-161 [11], ResNet-101 [10], and Inception-V3 [23]. The comparative results are illustrated in Table 1. Compared to ensemble, regular networks like GAN and GNN improve the detection effectiveness; however, DenseNet-161, ResNet-101, and Inception-V3 with much deeper structure significantly underperform due to limited data for fine-tuning. By contrast, AdvRFD manages to leverage the powerful feature extraction capability of ImageNet classifiers and the facilitation from perturbation to achieve state-of-the-art accuracy and F1-score for fraud detection: (1) most of pretrained models reprogrammed by AdvRFD outperform other baselines; (2) Inception-V3 and DenseNet-161 obtain the best results on Ethereum and credit cards respectively. The only trainable parameter for AdvRFD is perturbation tensor, while transfer learning suffers from huge computations to fine-tune the whole networks. AdvRFD thus takes less time (12 epochs), while transfer learning requires at least 30 epochs for training.

4.5 Evaluation of Cost Efficiency

In addition to less training data requirement and less training effort than transfer learning, here we further demonstrate the practicality of AdvRFD for fraud detection with cost efficiency in computing resources. We report computing time of AdvRFD on Ethereum including/excluding the consumption of the deployed ImageNet networks, which is illustrated in Fig. 4. Different from other baselines, when we extract the time cost consumed by the ImageNet models, all the costs enjoy a drastic drop. This implies that a major part of model training and test time are spent on shared computes. This is especially beneficial for larger datasets. In other words, AdvRFD enables reusable resources, significantly reduces computing cost for individual tasks, and thus addresses the real-world challenge for fraud detection with limited resources.

5 APPLICABILITY

AdvRFD provides a learning-effective yet cost-efficient solution for fraud detection. Thus, AdvRFD holds the applicability of reprogramming high-performance neural networks to perform fraud detection tasks in practice. Similar to the recent work in machine learning that focused on building dynamically connected networks with reusable components [21], we may consider the source-domain models as our reusable components to perform the shared computes disentangled out of different tasks, while these components can be self-deployed in local or cloud servers for complete and easy access, or reached through pay-per-use online services. Though we demonstrate adversarial reprogramming of image-domain neural networks on fraud detection, both source-domain models and target-domain tasks can be in a wider range.

6 CONCLUSION

In this paper, we propose AdvRFD to adversarially reprogram pretrained classifiers to perform fraud detection task. We conduct experimental studies on real-world transactions to evaluate the performance of AdvRFD, which validate its fraud detection effectiveness and cost efficiency. The adversarial reprogramming leveraged by AdvRFD falls into the category of white-box adversarial attacks, which may not be feasible to repurpuse the access-limited online services. We leave the investigation on black-box adversarial reprogramming as our future work, yet it does not impact the great value and validity of AdvRFD for fraud detection in practice.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable feedback. Y. Fan and Y. Ye's work is partially supported by the NSF under grants IIS-2107172, IIS-2140785, IIS-2027127, IIS-2040144, IIS-1951504, CNS-1940859, CNS-1814825, and OAC-1940855, the NIJ 2018-75-CX-0032.

REFERENCES

[1] Shumeet Baluja and Ian Fischer. 2018. Learning to attack: Adversarial transformation networks. In *Thirty-second aaai conference on artificial intelligence*.

[2] Lingwei Chen, Yanfang Ye, and Thirimachos Bourlai. 2017. Adversarial machine learning in malware detection: Arms race between evasion attack and defense. In *2017 European Intelligence and Security Informatics Conference (EISIC)*. IEEE, 99–106.

[3] Grégoire Mesnil Yann Dauphin, Xavier Glorot, Salah Rifai, Yoshua Bengio, Ian Goodfellow, Erick Lavoie, Xavier Muller, Guillaume Desjardins, David Warde-Farley, Pascal Vincent, et al. 2012. Unsupervised and transfer learning challenge: a deep learning approach. In *ICML Workshop*. 97–110.

[4] Daniel de Roux, Boris Perez, Andrés Moreno, Maria del Pilar Villamil, and César Figueroa. 2018. Tax fraud detection for under-reporting declarations using an unsupervised machine learning approach. In *SIGKDD*. 215–222.

[5] Luca Demetrio, Scott E Coull, Battista Biggio, Giovanni Lagorio, Alessandro Armando, and Fabio Roli. 2020. Adversarial EXEmples: A Survey and Experimental Evaluation of Practical Attacks on Machine Learning for Windows Malware Detection. *arXiv preprint arXiv:2008.07125* (2020).

[6] Gamaleldin F Elsayed, Ian Goodfellow, and Jascha Sohl-Dickstein. 2018. Adversarial reprogramming of neural networks. *arXiv preprint arXiv:1806.11146* (2018).

[7] Ugo Fiore, Alfredo De Santis, Francesca Perla, Paolo Zanetti, and Francesco Palmieri. 2019. Using generative adversarial networks for improving classification effectiveness in credit card fraud detection. *Information Sciences* 479 (2019), 448–455.

[8] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[9] Tejus Gupta, Abhishek Sinha, Nupur Kumari, Mayank Singh, and Balaji Krishnamurthy. 2019. A Method for Computing Class-wise Universal Adversarial Perturbations. *arXiv preprint arXiv:1912.00466* (2019).

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[11] Forrest Iandola, Matt Moskewicz, Sergey Karayev, Ross Girshick, Trevor Darrell, and Kurt Keutzer. 2014. Densenet: Implementing efficient convnet descriptor pyramids. *arXiv preprint arXiv:1404.1869* (2014).

[12] Eunji Kim, Jehyuk Lee, Hunsik Shin, Hoseong Yang, Sungzoon Cho, Seung-kwan Nam, Youngmi Song, Jeong-a Yoon, and Jong-il Kim. 2019. Champion-challenger analysis for credit card fraud detection: Hybrid ensemble and deep learning. *Expert Systems with Applications* 128 (2019), 214–224.

[13] Xiaoting Li, Lingwei Chen, and Dinghao Wu. 2021. Turning Attacks into Protection: Social Media Privacy Protection Using Adversarial Attacks. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*. SIAM, 208–216.

[14] Zhiwei Liu, Yingtong Dou, Philip S Yu, Yutong Deng, and Hao Peng. 2020. Alleviating the inconsistency problem of applying graph neural network to fraud detection. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1569–1572.

[15] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1765–1773.

[16] Paarth Neekhara, Shehzeen Hussain, Shlomo Dubnov, and Farinaz Koushanfar. 2018. Adversarial Reprogramming of Text Classification Neural Networks. *arXiv preprint arXiv:1809.01829* (2018).

[17] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. 2016. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*. IEEE, 372–387.

[18] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. 2017. On the expressive power of deep neural networks. In *international conference on machine learning*. PMLR, 2847–2854.

[19] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y Ng. 2007. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning*. 759–766.

[20] Kuldeep Randhawa, Chu Kiong Loo, Manjeevan Seera, Chee Peng Lim, and Asoke K Nandi. 2018. Credit card fraud detection using AdaBoost and majority voting. *IEEE access* 6 (2018), 14277–14284.

[21] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538* (2017).

[22] Mark J Shensa. 1992. The discrete wavelet transform: wedding the a trous and Mallat algorithms. *IEEE Transactions on signal processing* 40, 10 (1992), 2464–2482.

[23] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2818–2826.

[24] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013).

[25] Maruti Techlabs. 2020. How Machine Learning Is Enhancing Fraud Detection. In *https://marutitech.medium.com/how-machine-learning-is-enhancing-fraud-detection-694f3a2237f*.

[26] Yun-Yun Tsai, Pin-Yu Chen, and Tsung-Yi Ho. 2020. Transfer learning without knowing: Reprogramming black-box machine learning models with scarce data and limited resources. *arXiv preprint arXiv:2007.08714* (2020).