Adversarial Machine Learning in e-Health: attacking a Smart Prescription System

Article

Accepted version

S. Gaglio, A. Giammanco, G. Lo Re, M. Morana

Proceedings of AI*IA 2021 Advances in Artificial Intelligence: XX International Conference of the Italian Association for Artificial Intelligence, Milan, Italy, 2021.

Publisher: Springer

Adversarial Machine Learning in e-Health: attacking a Smart Prescription System

Salvatore Gaglio[1,2], Andrea Giammanco[2], Giuseppe Lo Re[1,2], Marco Morana[1,2]

[1]Smart Cities and Communities National Lab CINI - Consorzio Interuniversitario Nazionale per l'Informatica
[2]Dept. of Engineering, University of Palermo, Palermo, Italy
{salvatore.gaglio, andrea.giammanco, giuseppe.lore, marco.morana}@unipa.it

Abstract. Machine learning (ML) algorithms are the basis of many services we rely on in our everyday life. For this reason, a new research line has recently emerged with the aim of investigating how ML can be misled by adversarial examples. In this paper we address an e-health scenario in which an automatic system for prescriptions can be deceived by inputs forged to subvert the model's prediction. In particular, we present an algorithm capable of generating a precise sequence of moves that the adversary has to take in order to elude the automatic prescription service. Experimental analyses performed on a real dataset of patients' clinical records show that a minimal alteration of the clinical records can subvert predictions with high probability.

Keywords: Adversarial Machine Learning · Healthcare · Evasion attacks.

1 Introduction

Machine learning algorithms are extensively adopted in scenarios where the analysis of large amounts of data is mandatory [28]. Typically, the higher the confidence of the automatic learning algorithm on its predictions, the higher the trust that the practitioner has towards the model, orienting his decisions accordingly. In recent years, a novel research line, named Adversarial Machine Learning (AML), is studying how to exploit the same optimization mechanism at the core of ML algorithms with an opposite intent: to let the model be sure, with high confidence, about an erroneous prediction. In particular, *Adversarial examples* are defined as "those that change the verdicts of machine learning systems but not those of humans" [7]. Because of the immediateness for a human to verify the appearance of a certain image and evaluate the correctness of the classifier, image processing has been for several years the most common scenario in which the effectiveness of adversarial examples can be demonstrated. In this context, the attacks are aimed at creating noise patterns [29] that exhibit two main characteristics: their superimposition over the original image is invisible to the human eye, and they cause an error in the classification algorithm. Moreover, algorithms for adversarial images corruption can heavily exploit the very

large number of features (i.e., all the pixels of the image), as well as their scale of variability depending on the adopted encoding, so that the ascent along the gradient can proceed simultaneously in multiple directions at the same time. In other application domains, understanding the best way to corrupt the input with adversarial noise can be very challenging. For instance, considering a malware detection algorithm based on the API calls made by a software [15], one possible adversarial noise may consist in adding innocuous calls [5], while preserving the malicious behavior of the software. In an ambient intelligence scenario [4], sensors' raw data can be altered through a vector of carefully selected real values, in order to let a smart anomaly detection system [12] raise false irregularity alerts regarding users' behaviors, or interrupt the operation of an intelligent energy-saving system [13]. Assuming the presence of a Reputation Management System capable of identifying malicious entities in a sharing environment [2, 3], changing the released feedback patterns can refresh the bad reputation of an adversary. Conjugated in Online Social Networks, slight modifications in spammers behavior (e.g., inflating the number of innocuous tweets) can hide their malicious intent to an intelligent detector [11]. Systems relying on smartphones sensors to recognize the activities carried out by users [9, 10], may be trained on corrupted labeled data and fail in their identification task worsening the end services provided. In other domains, guaranteeing that the final verdict of the human remains unchanged is not straightforward. In a *healthcare* scenario, for example, it would mean that an expert clinician should not alter his judgment in the face of an altered clinical record. However, if the adversary's move consists in altering the patient's record, it is highly likely that the final decision made by the clinician will change. Nonetheless, perturbed clinical records may still be regarded as adversarial examples, as they share both the final goal to fool a machine learning algorithm, and the methodology used to get to the specific noise through the formulation of an optimization problem. In this paper, we address this issue and show how an adversary may alter binary entries in the clinical record of a patient in order to elude a smart prescription system. In this scenario, given that economic return is one of the most common motivations to conduct adversarial attacks in the healthcare domain [17], we can imagine as adversary an agent of a pharmaceutical company that produces a particular active ingredient, and wants to increase the sales by artificially inflate the number of prescriptions. In order to elude the smart prescription system, we propose an algorithm capable of generating the precise sequence of moves that the adversary has to take, i.e., which binary entries on the clinical record of the patient need to be flipped. In particular, we assume that the target model to evade is a neural network, whose parameters can be reasonably emulated by probing the smart prescription service as a black box [6]. The remainder of the paper is organized as follows. Section 2 discusses recent studies in the field of *AML*. Section 3 outlines the *healthcare* scenario considered as case study. Section 4 formalizes the model of the adversary. Section 5 describes the algorithm to generate the adversarial perturbation for the clinical records of the patients. Section 6 presents the experimental analysis to validate our proposal. Section 7 draws the conclusions.

2 Related Work

In recent years, *AML* has been the subject of studies from multiple fields of inquiry [6]. These are spearheaded by the image processing field, given the easy demonstrability of how well-designed adversarial examples are potentially lethal. In [29] for example, the authors formulate a method to create a perturbation which is strictly constrained in space, in order to craft a sort of sticker that is similar to real-world noise. By applying such stickers to danger road signs, state-of-art object detectors are led to completely different predictions, which can be fatal for self-driving cars. Considering the speech recognition field, in [21] multiple denoising strategies are leveraged to defend against attempts of altering the semantics of the sentences. These attempts reveal their full malevolent potential when cast on popular systems (e.g., Alexa), in order to redirect users to fraudulent sites instead of performing their intended requests. The problem of malware detection is studied in [24], where a set of instructions are injected after the *return* statement of ActionScript programs in order to inflate the detection of false negatives (i.e., malicious applications classified as benign) without altering the behavior of the code. In [5] is presented a similar approach, where the malevolent behavior in terms of Windows API calls is kept fixed, while the addition of benign calls serves the purpose to inflate the recognition of false negatives. The security of machine learning algorithms in mobile edge computing scenarios is addressed in [30], where false data are injected in the training set of the models in order to alter the aggregated results computed by the server, and in turn, the service offered to end users. The authors propose a graph matching algorithm to filter outliers according to the distance between the graph inferred from data, and the graph extracted from popular location based social networks. Network intrusion detection systems are examined in [8], where an autoencoder is employed to generate features resembling the benign class, which an attacker can use to circumvent automatic detectors. With regard to the *healthcare* domain, in [26] the authors evaluated the impact of several attack algorithms against models trained on a dataset containing ten vital signs of patients, showing how both attacks during training and test phase can have perilous implications. However, it is not formulated a precise sequence of steps the adversary has to make in order to achieve his goal, given that the perturbation is a real number which is difficult to interpret, and thus, inject into the data in a realistic scenario.

3 Case Study

Electronic healthcare represents an ideal scenario to describe an adversary attack because of the strong economic interests that move the pharmaceutical production. In this paper we consider a typical scenario, schematically represented in Figure 1, where Alice and Bob are the doctor and the patient respectively. Bob reports his symptoms to Alice, who compiles a medical record also including his personal information, so that a decision about which treatment to prescribe can be made. In order to refine her decision, Alice relies on a trusted *Smart*

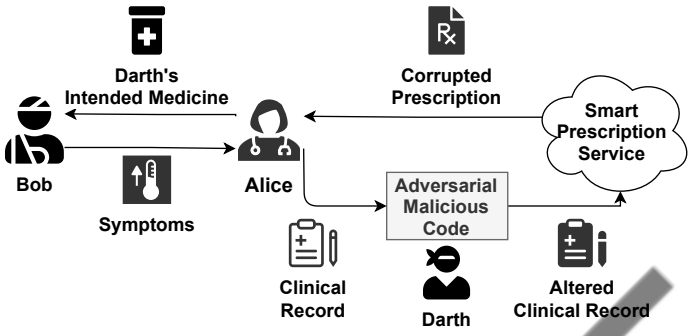

Fig. 1. A Smart Prescription System with adversary.

Prescription Service on cloud, which is able to reason about the information in the medical record and suggest appropriate treatment. It is assumed that the model beneath this service is a neural network. In this context, a pharmaceutical company infiltrator named Darth, gains an economic return when a drug of its corporation is prescribed. Darth suggests Alice to install a software in the host that will interact with the cloud service, whose stated purpose is to optimize response times and effectiveness of the prescription system. Actually, this program performs an *AML* algorithm able to elude the *Smart Prescription Service* and induce the prescription of Darth's intended treatment. In particular, this middleware software identifies a restricted set of features which have to be altered in order to deceive the predictor in the cloud. Altering just a few features is a characteristic of the utmost importance in this scenario, since the *Smart Prescription Service* may return a detailed report including the altered clinical record received as input, which therefore needs to contain no striking changes in order not to make any doctor suspicious. Finally, Alice will weigh her judgment based on the response of the intelligent service, resulting in the prescription of Darth's intended drug with high probability. The following subsections describe the attack.

4 Threat Model

Following the guidelines proposed in [6], in this section we frame the model of the adversary according to three main aspects: what is the pursued security breach (attacker's goal); what is the degree of acquired knowledge on the problem domain (attacker's knowledge); what are the concrete viable actions to achieve the malevolent intent (attacker's capability).

Attacker's goal: the adversary carries out an *integrity* violation of the predictive algorithm in order to flip its belief without disrupting the system in the whole, thus protecting himself from the risk of being caught. The attack specificity is *indiscriminate*, since the adversary does not make any distinction between the patients he wants to fool at his benefit. Accordingly, the error speci-

ficity is *specific*, as the target label, i.e., the active principle, that the adversary wants to be prescribed to raise an economic return for his company, is prefixed.

Attacker's knowledge: we assume the adversary has perfect knowledge on the domain at hand, in other words, he conducts a *white-box* attack. The parameters (weights and biases) and hyperparameters (number of layers, number of neurons per layer) of the neural network under attack are known to the adversary, as well as the feature representation of the data. What needs not to be necessarily known are the portion of samples being part of the training set, and certain hyperparameters of the training process such as the batch size, the number of epochs, the learning rate, the weight decay factor adopted as regularizer, and the momentum coefficient for gradient descent. Although these assumptions may seem highly unlikely, it is common practice to test the strength of a machine learning model against the worst case scenario, so that under real and softer conditions the security of the system should not decrease. Moreover, in light of the transferability property of adversarial attacks [16], the model's parameters can be estimated by querying repeatedly, until a surrogate model capable of providing the same answers as the target model can be built.

Attacker's capability: the attack influence is *exploratory*, as the adversary has no access to the training data, and can only corrupt data belonging to the test set. The data manipulation constraints strongly depend on the particular scenario we are addressing in this study, where data are in the form of binary feature vectors. It is thus clear that the adversary has to respect the range of allowed values in the clinical records of the patients, i.e., values either of 0 or 1.

5 Methodology

The target *Smart Prescription Service* is a neural network, whose structure will be further investigated in Section 6.1. The pseudocode for the proposed strategy to create an effective perturbation against this model is provided in Algorithm 1. Our approach leverages the logic behind the Fast Gradient Sign Method (FGSM) [20], by first computing the gradient of the model's loss function $L(\theta, x, y)$ with respect to the input vector x, ground truth label y, and trained parameters θ. It then selects the least amount of features whose perturbation leads to the most precipitous step taken along the direction of the gradient. In what follows we retrace the complete logical flow of the proposed algorithm.

This algorithm is executed by the malicious code injected by Darth into Alice's PC. The first step of the attack consists in computing the forward pass of the neural network w.r.t. the input vector x Darth wants to perturb. If the hypothesis of the model $\hat{y}$ is different from the target label y_{target}, Darth's objective is to flip the predicted label for the input vector x to the intended one. In other terms, the goal of the attack is to find the perturbation δ such that $h_\theta(x + \delta) \neq \hat{y}$. First, Darth puts in place a revised version of FGSM [20]. The traditional approach aims at climbing up the gradient by adding the perturbation $\xi = \epsilon \cdot sign(\nabla_x L(\theta, x, y))$, for a given $\epsilon > 0$, so that the perturbation vector ξ is composed by values equal to $\pm\epsilon$. ∇_x symbolizes the gradient taken w.r.t. the

Algorithm 1 Binary Adversarial Perturbation

Input:
x: the input binary feature vector to perturb;
y: the ground truth label for x;
$mask$: a binary indicator of alterable features;
θ: trained parameters of the neural network;
ψ: number of binary features the adversary may perturb;
y_{target}: the desired output label.

Output:
δ: perturbation to add to the input sample such that: $h_\theta(x+\delta) = y_{target} \neq \hat{y}$.

$\hat{y} \leftarrow h_\theta(x)$
$\delta \leftarrow zeros\_like(x)$
if $\hat{y} == y_{target}$ **then**
  return δ
$\xi \leftarrow \nabla_x L(\theta, x, y)$
$\xi_{ranked} \leftarrow sort\_descending(\xi,\ key = abs)$
$flippable \leftarrow mask\ \&\ (x \oplus sign(\xi))$
$counter \leftarrow 0$
for $value \in \xi_{ranked}$ **do**
  if $counter == \psi$ **then**
    break
  $idx \leftarrow \xi.index(value)$
  if $flippable[idx] == 1$ **then**
    if $x[idx] == 0$ **then**
      $\delta[idx] \leftarrow +1$
    else if $x[idx] == 1$ **then**
      $\delta[idx] \leftarrow -1$
    $counter \leftarrow counter + 1$
return δ

input vector. In this paper we decided to consider only the term $\xi = \nabla_x L(\theta, x, y)$, so that each single $\xi_i \in \mathbb{R}$. The reason lies in the need to select only a small subset of the input features to perturb. Opposed to the image processing domain, where each pixel may be perturbed with a small step in its scale of representation, in different domains where the features may assume a limited set of values (in this case, binary values), each single perturbation added to the input features has to be selected with care. Therefore, from the real-valued perturbations vector ξ, Darth has to craft a binary perturbation mask to add to the input sample in order to flip the neural network's prediction. Initially, Darth sorts the perturbation vector ξ according to the absolute values of its components. The higher the value in ξ for a specific feature, the higher the contribute along the error that its perturbation will induce, thus becoming the optimum target. Let us now suppose that the application domain imposes some constraints on the features that may be altered; we represent these constraints in the form of a binary *mask* as input to our algorithm, where the presence of a 1 indicates that the related feature may be perturbed. This *mask* explicates those features whose alteration is risky, because they can easily lead to the possibility of being disclosed during

the attack. Another input parameter to the attack algorithm is the maximum number of features ψ that Darth may alter from the input vector. A single input feature x_i may be altered in two cases:

1. $x_i == 1$ and the perturbation which results from ascending along the gradient of the loss function has negative sign, i.e. $sign(\xi_i) = -$, so that x_i may be flipped by adding $\delta_i = -1$. In other words, a feature value of 1 can be altered to 0 only if the sign of the gradient along that feature is negative;
2. $x_i == 0$ and the perturbation has positive sign instead, i.e. $sign(\xi_i) = +$, so that in order to flip x_i, $\delta_i = +1$ may be added.

To achieve this aim, we first take the sign of the perturbation vector ξ, $sign(\xi)$, which is then XORed with the input vector x: $x \oplus sign(\xi)$, so as to obtain a True value only when the input feature is 1 and the perturbation has negative sign, and vice versa. We represent $sign(\xi)$ as a binary vector where 1 stands for the sign '+' and 0 for '-'. The result of this operation is then processed with a bit-wise AND with the input $mask$: $flippable = mask \ \& \ (x \oplus sign(\xi))$. This operation results in a binary vector, $flippable$, which signals all those features that, if altered, make the neural network increase the error, because their alteration is concordant with the direction of the loss's gradient. Finally, having the list of features he may alter to deceive the neural network, Darth chooses the ψ features with maximum absolute value, in order to take the gradient's sharpest stride.

In order to achieve the attack, Darth must have a deep knowledge of the medical domain he is going to infiltrate. This implies the awareness of both the set of alterable features to compose the $mask$ and, most importantly, the parameters θ of the model beneath the smart prescription service. The latter can be achieved by probing the service as a black box, and building a surrogate model which responds in the most similar manner to the smart prescription service [6]. This approach finds his justification in the demonstrated transferability property of adversarial attacks [16]. When Alice provides Bob's clinical record to Darth, he first decides a threshold ψ of maximum binary feature values to perturb. He computes the loss' gradient ∇_x of the surrogate model's parameters with respect to Bob's record. The most proficient features to alter are those which posses three properties: they do not appear in the $mask$ of inconvenient features; they have the highest correspondent module in ∇_x; their alteration is concordant with the respective sign of ∇_x. The perturbed clinical record is then provided to the smart prescription service, which will return, with high probability, a report to Alice containing Darth's intended medicine as the suggested prescription. It is Darth's concern to select ψ as a good trade-off between an higher probability of subverting the smart prescriber prediction, and a lower probability of raising Alice's doubts towards the model's outcome.

6 Experimental Analysis

In order to validate our proposal, we adopted the AMR-UTI dataset[1] [18,23,27], which contains electronic health records of patients with urinary tract infections. Each record consists of demographic information, past clinical data such as previous antibiotic exposure or resistance, and the antibiotic prescription chosen by a clinician to treat the patient. This dataset, allows to train a model able to prescribe the so-called "empiric antibiotic treatment", which the patient should take while waiting the necessary three days for the accurate response from his urinal specimen analysis. In our scenario, the interest of the adversary lies in altering the treatment chosen by the model, simultaneously respecting any contraindications w.r.t. the patient's clinical status. In particular, we considered the patients who were treated with a first-line antibiotic, which is one of two classes: nitrofurantoin (NIT) and trimethoprim-sulfamethoxazole (SXT). The authors of the dataset provided a train/test division based on the years: specimen samples of the train set have been collected during the years 2007-2013, whereas the specimen in the test set refers to the period 2014-2016. Respecting this original division, the train set of first-line prescriptions contains 6815 samples, while the test set contains 2618 samples. Among the train set, 1892 samples received an empirical prescription of nitrofurantoin (NIT), and 4923 the trimethoprim-sulfamethoxazole (SXT). In the test set, 1358 samples where prescribed nitrofurantoin (NIT), the remaining 1260 trimethoprim-sulfamethoxazole (SXT).

Among the features exposed in the AMR-UTI dataset, we considered the patients' demographic information as *"not-corruptible"* (which we model through the input *mask* in Algorithm 1), in the sense that the adversary has no interest in altering these information in the clinical record of the patient, because of their ease of counter-proofing with reality. By performing other preprocessing steps which are released as part of the source code[2], we resort to a set of 564 binary features which are represented with different time granularities. In this paper, we restrict the analysis to the features registered within 180 days, also because this is the time window most commonly shared between all the features, for a total amount of 135. Finally, we select the κ best features according to the chi-square independence test [25], where κ is considered as one of the hyperparameters whose exploration will be further described in the next subsection. Having fixed a specific value for κ, we remove all those samples with equal binary features values but different label.

6.1 The classification network

Experiments were performed starting from an existing neural network[3], which we extended by adding the cross entropy loss function, the softmax activation layer, the momentum gradient descent, the regularization through weight decay,

[1] `https://www.physionet.org/content/antimicrobial-resistance-uti/1.0.0/`
[2] `https://github.com/agiammanco94/AIxIA2021`
[3] `https://github.com/RafayAK/NothingButNumPy`

Table 1. Ranges of values for the hyperparameters explored with Random Search.

Category	Hyperparameter	Min	Max
Network	Number of layers	2	7
	Number of neurons in 1st layer	50	500
Dataset	Number of κ features	20	70
Training	Learning rate α	$1e^{-4}$	$1e^{-3}$
	Weight decay λ	$1e^{-5}$	$1e^{-3}$
	Momentum ν	$1e^{-3}$	$8e^{-1}$

the random search algorithm for hyperparameters tuning [19], the FGSM [20], and the attack algorithm proposed in this work. We employ the random search approach [19] to explore different structures for the neural network (in terms of number of layers, and number of neurons per layer) and different configurations of the training phase (the learning rate, weight decay, and momentum factors for the gradient descent algorithm). Table 1 shows the range of values explored for each of the hyperparameters: in each experiment, a uniform probability with Min and Max as extremes is sampled for every hyperparameter. In particular, once the number of neurons for the first layer had been selected, the neurons for the subsequent layers are halved, given that a preliminary experimental evaluation proved this architectural choice to be more effective. The number of neurons in the last layer is equal to 2, since there are two classes (NIT and SXT) in the problem we are addressing. The activation functions employed are the ReLU for all the intermediate layers, and the Softmax for the output layer. This choice led to the adoption of the weights initialization procedure described in [22], which has been proved to be the optimal choice to combine with ReLU layers.

The *f-score* measure [19] has been employed to evaluate the effectiveness of the neural network classification; to be more specific, *f-score* values have been computed for each of the two classes separately, thus by assuming NIT and SXT as the positive class in turn. Then, in order to evaluate the effectiveness of the attack algorithm, the analysis was restricted to the portion of samples of the test set that the neural network identifies correctly, and we measured the error percentage of the model w.r.t. the corrupted input samples of a specific class as:

$$error < class >= \frac{|h_\theta(x+\delta) \neq class \ \& \ h_\theta(x) = class = y|}{|h_\theta(x) = class = y|},$$

where h_θ is the hypothesis of the model with the trained parameters θ, x is the set of samples in the test set, δ is the perturbation created with the attack algorithm, and y is the ground truth class.

6.2 Results and discussion

We ran 50 batches of experiments where the neural network architecture hyperparameters are sampled from ranges shown in Table 1. For each batch, 50 different samplings of training hyperparameters have been explored while keeping fixed the network structure sharing such values in all neurons, so resulting

Table 2. F-score and errors of the best performing experiment in each group.

Exp.	f-score NIT	f-score SXT	error NIT					error SXT				
			$\psi = 1$	$\psi = 2$	$\psi = 3$	$\psi = 4$	$\psi = 5$	$\psi = 1$	$\psi = 2$	$\psi = 3$	$\psi = 4$	$\psi = 5$
1	0.80	0.72	0.74	1.00	1.00	1.00	1.00	0.71	0.96	1.00	1.00	1.00
2	0.73	0.70	0.58	0.87	0.97	1.00	1.00	0.78	1.00	1.00	1.00	1.00
3	0.64	0.67	0.31	0.70	0.89	0.96	1.00	0.29	0.60	0.84	0.95	1.00

Table 3. Hyperparameters of the most significant experiments.

Experiment	Neurons	κ	α	λ	ν
1	$145, 2$	29	$6.70e^{-4}$	$1.71e^{-4}$	$4.38e^{-1}$
2	$177, 2$	42	$5.40e^{-4}$	$1.36e^{-4}$	$7.49e^{-3}$
3	$215, 2$	59	$5.18e^{-4}$	$3.25e^{-5}$	$7.65e^{-1}$

in a total number of 2500 configurations. Results have been analyzed according to the value of κ; in particular we considered three ranges of values, i.e., κ in $[20; 30]$, $[31; 50]$, and $[51; 70]$. For the sake of clarity, in Table 2 we present the most significant results from each group, while the corresponding hyperparameters are reported in Table 3. In particular, Table 2 shows the *f-scores* of the selected models w.r.t. the two classes, as well as the error percentage due to the injection of $\psi \in [1, 2, 3, 4, 5]$ binary feature values into the test data with our attack algorithm. When the dataset is preprocessed in order to select only the $\kappa = 29$ most meaningful features (first row), our attack procedure with $\psi = 3$ allowed to completely mislead the neural network for all the test data. This result can be due to the extreme sparsity of the AMR-UTI dataset, in which the vast majority of the binary features have value 0. For such a reason, when a binary feature value is flipped from 0 to 1 in the direction of the loss' gradient, it is extremely likely that the new feature becomes "characteristic" for the target class, thus flipping the label with high probability. As the number of features considered increases, an higher quantity of features needs to be perturbed in order to completely subvert the predictions, in particular, for $\kappa = 42$ and $\kappa = 59$ (second and third row of Table 2), the best performances of the algorithm are achieved by altering 4 and 5 features respectively.

In order to provide a more in-depth analysis of the features that have actually been altered in the experiments carried out, Figure 2 shows the percentage of times that a given feature has been chosen by our algorithm, and the corresponding success rate in deceiving the model. The two heatmaps are computed aggregating the results of the three experiments reported in Table 2. It is important to note that the percentages of feature selections depicted in the first heatmap have unitary sum for a fixed value of ψ, meaning that the shown set of features contains all the perturbed ones. Instead, the percentages of success due to feature perturbations represented in the second heatmap sum to the error rate of the model, e.g., when $\psi = 5$ the percentages of success add up to 1, since results shown full model deception in all the experiments by altering 5 features. The most selected feature (38) is related to breathing difficulties, and it has been chosen for the 13.67% of times across both all the experiments and

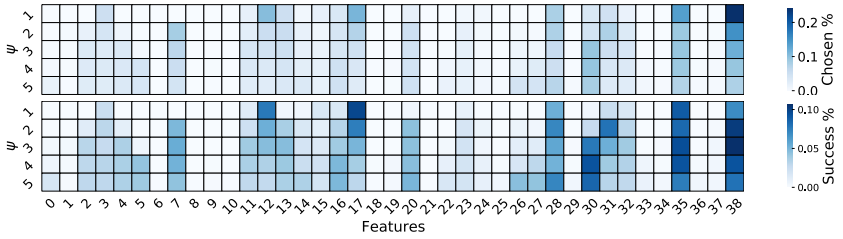

Fig. 2. Heatmaps of the perturbed features by our attack algorithm, and the respective success percentage in changing the prediction of the model.

ψ values, leading to success in 8.89% of cases w.r.t. the total of all other perturbation attempts. The motivation behind this fact can be traced back to the adverse effects of the two active principles. Indeed, among the side effects of nitrofurantoin assumption[4] there is pulmonary toxicity, which is instead absent in trimethoprim-sulfamethoxazole's side effects[5]. In light of this consideration, our approach realized the shrewd behavior of the doctor prescribing SXT treatment for patients who have recently experienced breathing complications.

7 Conclusions

In this paper, we proposed an algorithm for the generation of adversarial examples in scenarios with electronic health records in the form of binary data. In particular, we studied how an adversary may alter the medical record of a patient in order to fool an intelligent system for antibiotic prescription. The experimental results showed that even only modifying three fields in the patient record, a trained neural network can almost always be induced into suggesting a prearranged treatment. As part of our future works, we want to eliminate the time granularity as input parameter to filter the dataset. For example, if the adversarial noise produced by an attack algorithm suggests to modify a feature with time granularity equal to 14 days from 0 to 1, then, all the features falling in the same category and with a granularity > 14 should be set to 1. Moreover, we plan to define an automatic strategy for dynamically choosing the number of ψ features to perturb based on the magnitude of the gradient, so that ψ does not need to be specified as input to the approach. Finally, we want to investigate the feasibility of our approach in other smart environments such as university campuses [1], where adversarial attacks aim at disrupting the provision of intelligent services to users [14].

[4] https://www.msdmanuals.com/professional/infectious-diseases/bacteria-and-antibacterial-drugs/nitrofurantoin

[5] https://www.msdmanuals.com/professional/infectious-diseases/bacteria-and-antibacterial-drugs/trimethoprim-and-sulfamethoxazole

References

1. Agate, V., Concone, F., Ferraro, P.: Wip: Smart services for an augmented campus. In: 2018 IEEE International Conference on Smart Computing. pp. 276–278 (2018)
2. Agate, V., De Paola, A., Gaglio, S., Lo Re, G., Morana, M.: A framework for parallel assessment of reputation management systems. In: 17th International Conference on Computer Systems and Technologies. pp. 121–128 (2016)
3. Agate, V., De Paola, A., Lo Re, G., Morana, M.: A simulation software for the evaluation of vulnerabilities in reputation management systems. ACM Transactions on Computer Systems (TOCS) **37**(1-4), 1–30 (2021)
4. Agate, V., Ferraro, P., Gaglio, S.: A cognitive architecture for ambient intelligence systems. In: AIC. pp. 52–58 (2018)
5. Al-Dujaili, A., Huang, A., Hemberg, E., O'Reilly, U.M.: Adversarial deep learning for robust detection of binary encoded malware. In: 2018 IEEE Security and Privacy Workshops (SPW). pp. 76–82 (2018)
6. Biggio, B., Roli, F.: Wild patterns: Ten years after the rise of adversarial machine learning. Pattern Recognition **84**, 317–331 (2018)
7. Buckner, C.: Understanding adversarial examples requires a theory of artefacts for deep learning. Nature Machine Intelligence pp. 1–6 (2020)
8. Chen, J., Wu, D., Zhao, Y., Sharma, N., Blumenstein, M., Yu, S.: Fooling intrusion detection systems using adversarially autoencoder. Digital Communications and Networks (2020)
9. Concone, F., Gaglio, S., Lo Re, G., Morana, M.: Smartphone data analysis for human activity recognition. In: Conference of the Italian Association for Artificial Intelligence. pp. 58–71. Springer (2017)
10. Concone, F., Lo Re, G., Morana, M.: A fog-based application for human activity recognition using personal smart devices. ACM Transactions on Internet Technology (TOIT) **19**(2), 1–20 (2019)
11. Concone, F., Lo Re, G., Morana, M., Ruocco, C.: Twitter spam account detection by effective labeling. In: ITASEC (2019)
12. De Paola, A., Ferraro, P., Gaglio, S., Lo Re, G., Morana, M., Ortolani, M., Peri, D.: A context-aware system for ambient assisted living. In: Ochoa, S.F., Singh, P., Bravo, J. (eds.) Ubiquitous Computing and Ambient Intelligence. pp. 426–438. Springer International Publishing, Cham (2017)
13. De Paola, A., Ferraro, P., Lo Re, G., Morana, M., Ortolani, M.: A fog-based hybrid intelligent system for energy saving in smart buildings. Journal of Ambient Intelligence and Humanized Computing **11**(7), 2793–2807 (2020)
14. De Paola, A., Gaglio, S., Giammanco, A., Lo Re, G., Morana, M.: A multi-agent system for itinerary suggestion in smart environments. CAAI Transactions on Intelligence Technology (2021)
15. De Paola, A., Gaglio, S., Lo Re, G., Morana, M.: A hybrid system for malware detection on big data. In: IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). pp. 45–50 (2018)
16. Demontis, A., Melis, M., Pintor, M., Jagielski, M., Biggio, B., Oprea, A., Nita-Rotaru, C., Roli, F.: Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks. In: 28th {USENIX} Security Symposium ({USENIX} Security 19). pp. 321–338 (2019)
17. Finlayson, S.G., Bowers, J.D., Ito, J., Zittrain, J.L., Beam, A.L., Kohane, I.S.: Adversarial attacks on medical machine learning. Science **363**(6433) (2019)

18. Goldberger, A.L., Amaral, L.A., Glass, L., Hausdorff, J.M., Ivanov, P.C., Mark, R.G., Mietus, J.E., Moody, G.B., Peng, C.K., Stanley, H.E.: Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. circulation **101**(23), e215–e220 (2000)
19. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016)
20. Goodfellow, I., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: International Conference on Learning Representations (2015)
21. Guo, Q., Ye, J., Chen, Y., Hu, Y., Lan, Y., Zhang, G., Li, X.: INOR—An Intelligent noise reduction method to defend against adversarial audio examples. Neurocomputing **401**, 160–172 (2020)
22. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (2015)
23. Kanjilal, S., Oberst, M., Boominathan, S., Zhou, H., Hooper, D.C., Sontag, D.: A decision algorithm to promote outpatient antimicrobial stewardship for uncomplicated urinary tract infection. Science Translational Medicine **12**(568) (2020)
24. Maiorca, D., Demontis, A., Biggio, B., Roli, F., Giacinto, G.: Adversarial Detection of Flash Malware: Limitations and Open Issues. Computers & Security **96** (2020)
25. McHugh, M.L.: The chi-square test of independence. Biochemia medica **23**(2), 143–149 (2013)
26. Newaz, A.I., Haque, N.I., Sikder, A.K., Rahman, M.A., Uluagac, A.S.: Adversarial attacks to machine learning-based smart healthcare systems. In: GLOBECOM 2020 - 2020 IEEE Global Communications Conference. pp. 1–6 (2020)
27. Oberst, M., Boominathan, S., Zhou, H., Kanjilal, S., Sontag, D.: Amr-uti: Antimicrobial resistance in urinary tract infections (version 1.0.0). Physionet (2020)
28. Verbraeken, J., Wolting, M., Katzy, J., Kloppenburg, J., Verbelen, T., Rellermeyer, J.S.: A survey on distributed machine learning. ACM Computing Surveys (CSUR) **53**(2), 1–33 (2020)
29. Xue, M., Yuan, C., He, C., Wang, J., Liu, W.: NaturalAE: Natural and robust physical adversarial examples for object detectors. Journal of Information Security and Applications **57**, 102694 (2021)
30. Zhao, P., Huang, H., Zhao, X., Huang, D.: P^3: Privacy-preserving scheme against poisoning attacks in mobile-edge computing. IEEE Transactions on Computational Social Systems (2020)