2019 IEEE International Conference on Artificial Intelligence Testing (AITest)

An Analytical Framework for Security-Tuning of Artificial Intelligence Applications Under Attack*

Koosha Sadeghi, Ayan Banerjee, and Sandeep K. S. Gupta
IMPACT Lab at CIDSE, Arizona State University, Tempe, Arizona, USA
Email: {k.sadeghi, abanerj3, and sandeep.gupta}@asu.edu

***Abstract*—Machine Learning (ML) algorithms, as the core technology in Artificial Intelligence (AI) applications, such as self-driving vehicles, make important decisions by performing a variety of data classification or prediction tasks. Attacks on data or algorithms in AI applications can lead to misclassification or misprediction, which can fail the applications. For each dataset separately, the parameters of ML algorithms should be tuned to reach a desirable classification or prediction accuracy. Typically, ML experts tune the parameters empirically, which can be time consuming and does not guarantee the optimal result. To this end, some research suggests an analytical approach to tune the ML parameters for maximum accuracy. However, none of the works consider the ML performance under attack in their tuning process. This paper proposes an analytical framework for tuning the ML parameters to be secure against attacks, while keeping its accuracy high. The framework finds the optimal set of parameters by defining a novel objective function, which takes into account the test results of both ML accuracy and its security against attacks. For validating the framework, an AI application is implemented to recognize whether a subject's eyes are open or closed, by applying k-Nearest Neighbors (kNN) algorithm on her Electroencephalogram (EEG) signals. In this application, the number of neighbors (k) and the distance metric type, as the two main parameters of kNN, are chosen for tuning. The input data perturbation attack, as one of the most common attacks on ML algorithms, is used for testing the security of the application. Exhaustive search approach is used to solve the optimization problem. The experiment results show k = 43 and cosine distance metric is the optimal configuration of kNN for the EEG dataset, which leads to 83.75% classification accuracy and reduces the attack success rate to 5.21%.**

***Keywords*-artificial intelligence; machine learning; security; perturbation attack; parameters tuning; optimization;**

I. Introduction

Artificial Intelligence (AI) widely utilizes Machine Learning (ML) algorithms in developing real-world applications in various fields, such as transportation [1], health-care [2], communication [3], security [4], and multimedia [5]. ML algorithms make key decisions in AI applications through various classification or prediction tasks. Any tampering with the decision process can cause application failure, which can result in financial loss or even fatality. This emerging research area, known as *adversarial machine learning*, where the ML algorithms face any attack interfering with their operation, is drawing more attention [6, 7].

*This work has been partly funded by CNS grant #1218505, IIS grant #1116385, and NIH grant #EB019202.

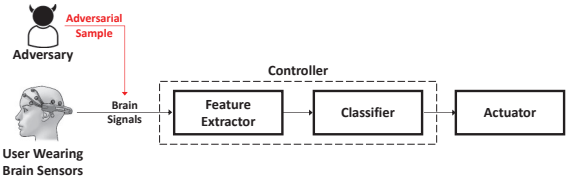


Figure 1: A generic model of ML algorithms in AI applications [8].

The attacks on AI applications, can be studied under two main categories: 1) *passive* (stealing information about the application), and 2) *active* (interrupting the application's normal operation). There are two approaches for active attacks: 1) corrupting the operational data of application [9], or 2) altering the application's components (e.g. its ML algorithms) [10]. The focus of this paper is the first approach, where an adversary perturbs the available data or manufactures new data, to obtain *adversarial samples*. In *presentation attacks*, the adversarial samples are applied as input, to cause errors in the ML classification or prediction results (Fig. 1). As shown in Table I, there are two main approaches for improving the accuracy or security of ML algorithms, which are: 1) modifying the training data or 2) improving the ML algorithm. This paper focuses on ML algorithm improvement in AI applications that perform classification tasks through supervised learning. In some applications, there is a trade-off between accuracy and security of ML algorithms. For example, in a centroid-based classifier [11], by setting the radius of a given class's boundary to zero, no adversarial sample can fit in the class (i.e. maximum security), but at the same time the accuracy is minimum (i.e. 50% for two classes). So, there are some debates on negative effects of security enhancement on the classification accuracy. The research question can be stated as follows: *Is there any solution for improving the security of AI applications, while avoiding accuracy reduction?*

A. Related Works & Research Gap

Table I shows neither empirical nor analytical tuning methods for ML algorithm improvement, aim toward enhancing the accuracy and security at the same time. Since empirical tuning methods requires manual configuration of

978-1-7281-0492-8/19/$31.00 ©2019 IEEE
DOI 10.1109/AITest.2019.00012

Table I: Related works on improving the classification process to enhance the accuracy or the security.

Focus			Method	Accuracy	Security
1) Data			Data Augmentation (Krell et al. 2017 [12])	✓	✗
			Adversarial Training (Goodfellow et al. 2014 [13])	✓	✓
			Generative Adversarial Network (Samangouei et al. 2018 [14])	✓	✓
2) Algorithm	**2-1) Tuning**	**2-1-1) Empirical**	Empirical Configuration (Prusa et al. 2017 [15])	✓	✗
		2-1-2) Analytical	Bayesian Optimization (Snoek et al. 2012 [16])	✓	✗
			Nelder-Mead Method (Albelwi et al. 2017 [17])	✓	✗
			Exhaustive Search (Sadeghi et al. 2017 [11])	✓	✗
			Exhaustive Search (Sadeghi et al. 2017 [11])	✗	✓
	2-2) Design		Classifier Randomization (Alabdulmohsin et al. 2014 [18])	✓	✓
			Enclosing the Target Class (Kantchelian et al. 2014 [19])	✓	✓
			Multiple Classifier System (Biggio et al. 2015 [20])	✓	✓
			Defensive Distillation (Papernot et al. 2016 [21])	✓	✓
			Region-Based Classification (Cao and Gong 2017 [22])	✓	✓

the parameters through trial-and-error by an expert in ML, which is time consuming and does not ensure obtaining an optimal configuration, analytical methods are drawing more attention [16, 17]. In the analytical tuning approach, there is only our previous work that configures the radius in the centroid-based classifier to optimize either the accuracy or the security, but not both at the same time [11]. *To the best of our knowledge, there is no research on tuning ML algorithms to achieve high classification or prediction accuracies, facing perturbation attacks.*

B. Proposed Research & Problem Statement

This work proposes an analytical framework for optimizing the accuracy and security of ML algorithms facing perturbation attacks. In this framework, the parameters of a given ML algorithm should be set in a way to maximize both accuracy and security. The framework operates in three main phases: 1) testing the ML accuracy, 2) testing the ML security, and 3) optimizing the ML parameters for maximum accuracy and security. In the third phase, a new objective function is defined that combines the test results from the first two phases. The main research problem can be formally presented as the following optimization problem (Eq. 1):

$$\begin{aligned} \underset{\varphi}{\text{maximize}} \quad & \lambda f(\varphi) - (1-\lambda) g(\varphi), \\ \text{subject to} \quad & 0 \leq \lambda \leq 1 \end{aligned} \tag{1}$$

where, $\varphi = \{\varphi_1, \varphi_2 \dots \varphi_n\}$, $f(\varphi)$, and $g(\varphi)$ are the classifier's parameters set, classifier's accuracy, and adversary's success rate (i.e. inverse of the classifier's security), respectively. Also, λ is a coefficient, which indicates the importance of accuracy and security for each AI application.

C. Solution Approach & Research Outline

In this research, we solve the optimization problem in Eq. 1 for an exemplary AI application, which uses k-Nearest Neighbors (kNN) classifier to recognize open or closed eyes states in a subject according to her Electroencephalogram (EEG) signals. Solving the optimization problem has two phases: 1) calculating the f and g values, and 2) finding φ to maximize the objective function. For calculating f, labeled test are fed to the kNN classifier, and based on the observed output the classification accuracy is obtained. For calculating g, the perturbation attack is simulated, where an adversary attempts to force the kNN to classify the perturbed input data as the open eyes class data. The probability of a successful adversarial attempt is calculated by random guessing the input data, which gives the g value. At last, for solving the optimization problem, exhaustive search approach is used to find the optimal φ. In this approach, the accuracy and security are calculated for all possible combinations of the two main parameters of kNN (i.e. number of neighbors and distance metric type) to find a combination that favors both accuracy and security. The experiment shows k = 43 and cosine distance metric is the optimal configuration for the kNN applied on the EEG dataset, which leads to 83.75% classification accuracy and reduces the attack success rate to 5.21%. A ML designer can utilize this framework to estimate the robustness of a given ML algorithm and customize the algorithm for desired accuracy and security levels. The contributions of the work can be summarized as follows:

- Modeling AI applications under attack and setting evaluation metrics for analyzing the AI accuracy and security.
- Proposing an analytical framework for security-tuning of AI applications under perturbation attacks.
- Security-tuning a kNN-based EEG classifier, to validate the framework for real-world applications.

II. AI Applications Under Attack

An attack on an AI application can be specified by an *attack scenario* consisting of: 1) the AI application under attack (i.e. *target system*), 2) the adversary (i.e. *attack system*), and 3) the defense mechanism (i.e. *defense system*). Given some knowledge about the target system, an adversary plans for an *attack strategy*, such as adding noise to the input data, to cause misclassification in the target application. Based on the launched attack, defensive changes in the target system (e.g. adding a filter to reduce data perturbations or tuning the ML parameters), may mitigate the attack [23].

A. Target System Model

As seen in Fig. 1, ML algorithms in the target AI applications, operate in two phases [24]: 1) feature extraction and 2) classification. The features are fed to the classifier

Table II: Volume calculation methods in security evaluation.

Category	Method	Comments
Calculus-Based	Integration [11]	Feasible for explicitly known classifier's function.
	Known Shape Spaces [26]	
Grid-Based [27]	Fine-Grid	Feasible for low dimensional input data domain.
	Grid Walk	
Geometry-Based [27]	Monte-Carlo [26, 28]	Available solutions for unknown classifier's function and relatively high dimensional data domain.
	Random Walk	
	Hit-and-Run	
	Convex Intersection	

for either training or testing. In supervised learning, labeled raw data is used to train the ML algorithm, in a way to match significant amount of the raw data samples to their corresponding labels. At the test time, labels are assigned to unlabeled raw data, by passing through the trained classifier [25]. Perturbation attacks are test-time attacks, which can be launched by adding noise to the unlabeled test data or manufacturing a new test data sample from scratch.

B. Attack System Model

The attack system model has four main specifications [29]: 1) *knowledge*, 2) *capability*, 3) *goal*, and 4) *strategy*. The adversary's knowledge about the AI application including its data or algorithms, has a direct impact on the attack success rate. This knowledge can be *limited* or *full*. Capability indicates the amount of alteration that can be made by an adversary during the attack, such as altering the training data or the ML algorithm. In perturbation attacks, it is assumed that the adversary can only alter the raw input data. Adversary's goal can be either: 1) *indiscriminate* or 2) *targeted*. In an indiscriminate perturbation attack, the perturbed data is misclassified in any classes, while in the targeted version, it should be recognized as a data sample of a specific class [30]. An attack strategy describes the detailed mechanism of an attack. In perturbation attack strategies, an adversary obtains the adversarial sample by altering a non-target data sample (e.g. by adding crafted perturbation, a.k.a. *adversarial noise*), to be accepted as the target class data. Other types of presentation attack, such as zero-effort [31] and replay [32] attacks, are out of the paper's scope.

C. Defense System Model

There are two main approaches for defense strategies [33]: 1) *proactive*, when the target system is prepared for potential threats before attack or 2) *reactive*, where strategies are applied after the occurrence of an attack, such as *machine unlearning* after poisoning attacks [34]. The classification process improvement strategies (Table I), fall in the proactive category. These strategies prevent access inside the target class boundaries by compacting the target boundaries through modifying its training process [35] or ML algorithm [36]. Another type of proactive strategies adds specialized detectors to the target system to block the adversarial samples [37], which is out of the paper's scope.

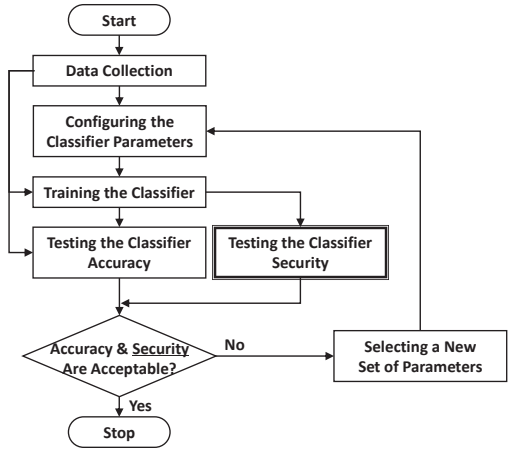


Figure 2: The analytical framework for security-tuning.

III. Security-Tuning Framework

Previous research mainly focus on optimizing the classifier's accuracy in AI applications (Table I). In our framework, in addition to accuracy, the security of the classifier is also tested toward finding solutions to mitigate attacks. Fig. 2 illustrates the flowchart of the framework. The main objective of the framework is to keep both accuracy and security of the classifier high by tuning its parameters. In the framework, various parameters configurations for the classifier are continuously tested through a closed loop, until reaching the desirable level of accuracy and security. As mentioned in Sec. I-B, the framework is in charge of solving the following optimization problem: $\lambda f(\varphi) - (1-\lambda)g(\varphi)$, where φ, $f(\varphi)$, and $g(\varphi)$ are the parameters set, accuracy, and adversary's success rate, respectively.

A. Terminology & Evaluation Metrics

In this part, some terms and metrics are defined, which are used for describing the framework:

- **Input Data**: raw data (e.g. EEG signal) that is fed to an AI application for classification.
- **Adversarial Sample**: Any presented data as input at the test time, which causes intentional misclassification.
- **Target Class**: A class in the data domain, which adversarial samples are supposed to fall inside its boundaries.
- **Accuracy (Acc.)**: The number of correct classification of the test data samples, out of the total number of trials.
- **Attack Success Rate (SR)**: SR indicates the percentage of adversarial samples, which are misclassified as the target class out of the total number of adversarial attempts.

B. Accuracy & Security Evaluation

The common way of testing the classification accuracy is to apply a pool of test data and observe the classification outputs. However, there is no universal method for testing the classifier's security. The main reason is the presence of several factors, such as data distribution, ML algorithm, and

Table III: Optimization methods.

Category	Method	Comments
Gradient-Based	Direct Methods	Require explicitly known objective function.
	Indirect Methods	
Gradient-Free	Exhaustive Search [11]	No need for explicit objective function, but time consuming for large data domains and no guarantee for optimal solution.
	Genetic Algorithm	
	Particle Swarm	
	Bayesian Optimization [16]	
	Simulated Annealing	
	Nelder-Mead Algorithm [17]	

adversary's capability, which have direct influence on the vulnerability level of a classifier under attack. To avoid this issue, we define a ground-truth for security evaluation by setting the adversary's specifications to their lowest level [38]. In this case, the adversary has no knowledge about the target system and can only change the input data (black-box scenario). Also, it can only generate the adversarial samples through random guessing to be accepted in any other classes. With this ground-truth, the security of any classifier can be uniformly tested and used for comparison. As seen in Table II, there are various approaches for calculating the attack success rate through random guessing of the input data. In this case, from geometrical viewpoint, by calculating the volume of the target class and dividing it by the volume of the whole data domain, the success rate can be obtained.

C. Optimization

Table III lists various search methods that can be applied to find the optimized parameters set (φ^*) [39]. The parameters search space for classic classifiers, such as kNN, is tractable, while more complex classifiers, such as deep neural networks, have an extremely vast search space, which makes the optimization procedure highly challenging. Fig. 3 is a geometrical representation of the optimization problem, which illustrates the changes in class boundaries w.r.t. the changes in kNN parameters (the implementation details in Sec. IV). Only two extracted features from training data samples of both eyes open/closed classes are selected for demonstration purposes. Also, the distance metric parameter is set to "Mahalanobis", while the number of nearest neighbors (k) can change from 1 to the total number of training samples (due to high sensitivity of class boundaries to k using Mahalanobis distance metric, this metric is chosen for the demonstration). In our attack scenario, the adversary attempts to guess a data sample with features inside the open eyes class (i.e. the gray areas in Fig. 3). From geometrical viewpoint, shrinking the target class area reduces the chance of the adversary in guessing samples inside the target class. However, too much reduction in the class volume can lead to rejection of the true class members (i.e. higher false negatives) and deteriorate the classification accuracy. For instance, when k = 51 (Fig. 3f), kNN has the highest classification accuracy (83.75%), while the attack Success Rate (SR) is also relatively high (3.05%). By reducing the k to 21, the open eyes class volume is shrunk, which leads to lower SR (1.85%) or in other terms higher security. However, the classification accuracy is also diminished (78.75%). The security-tuning framework, by considering both accuracy and security and their trade-offs, can find the optimal configuration, which is k = 51 (among the 16 configurations in Fig. 3) with objective function value equal to 40.35 (since only two features have been chosen in Fig. 3 experiments, the results are different from Table IV). It is also seen, when k = 151, kNN has the lowest accuracy and security (Fig. 3p).

IV. Implementation Details

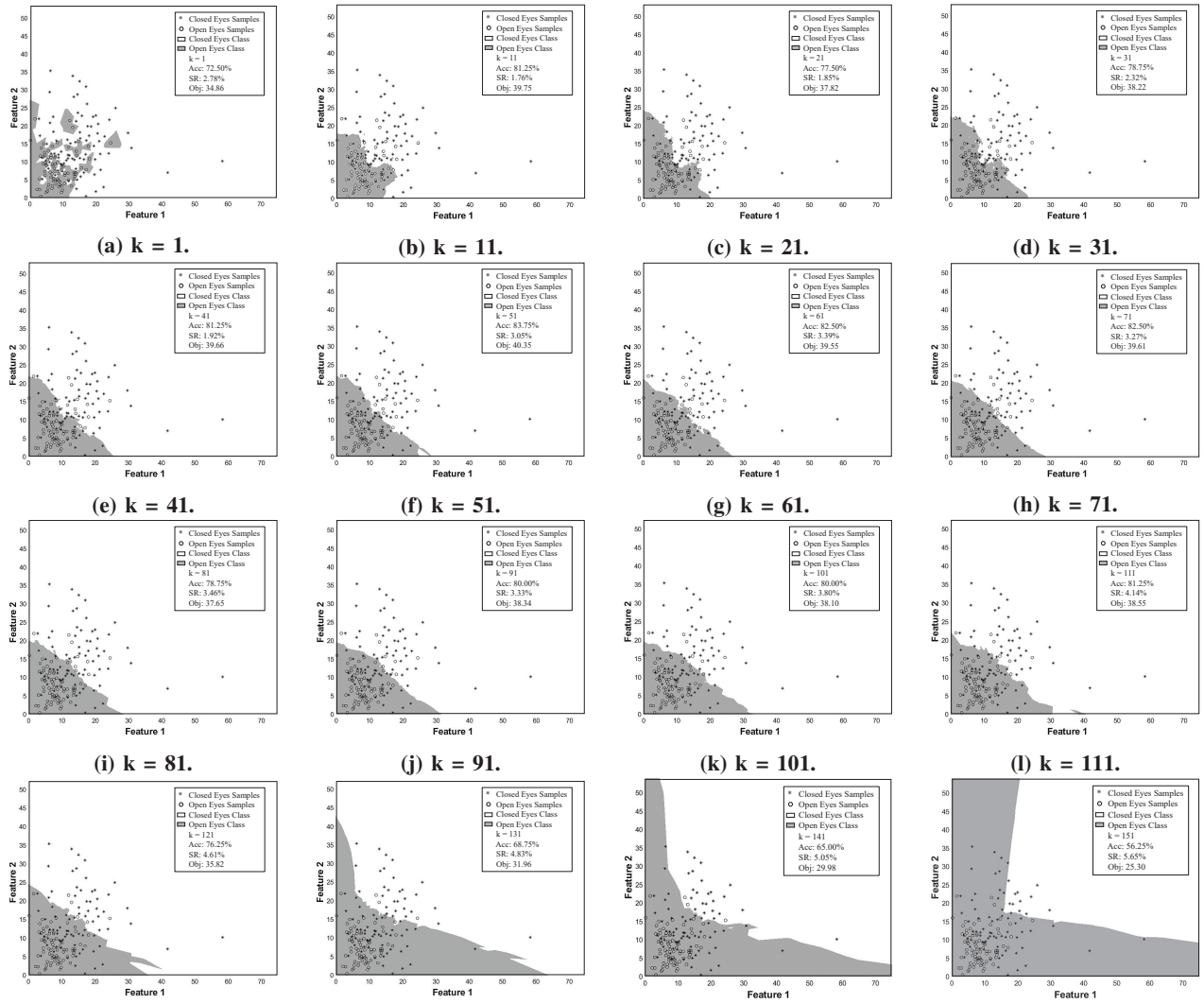
In this section, the detailed description of implementing an AI application under perturbation attack is provided, which utilizes the framework for security-tuning:

A. AI Application

As the research case study, a Brain-Machine Interface (BMI) application is developed to detect open/closed eyes states according to a subject's Electroencephalogram (EEG) signals. This application can be used for fatigue detection in drivers [1]. The application consists of three main components (Fig. 1): 1) sensor, 2) controller, and 3) actuator.

Data Collection Using Sensors: Electrical currents through neurons caused by brain activities produce potential differences in the order of microvolts (5-100 μV), which when measured by a sensor result in EEG waves. Typically the amplitude of EEG signals are measured by placing EEG electrodes on the surface of the scalp. In our experiment, an on-line dataset is used, which contains raw EEG data from 109 subjects collected through 64 channels with sampling rate of 160 Hz [40, 41]. Two minutes data of a randomly selected subject is chosen, which contains 1-min of EEG data when eyes are open and 1-min for closed eyes. According to the International 10-20 System [42], "O1" channel is selected for interpreting the subject's visual input.

Controller: The two main modules in the controller are: 1) *feature extractor* and 2) *classifier*. After acquiring raw signals, for feature extraction, fast Fourier transform (8 to 13Hz) is applied on each half a second of data ($0.5 \times 160 = 80$ data samples), which provides 240 feature vectors (with size six each) for both classes in total [43]. Two third of the data is kept as the training set and the rest of it forms the test set. The extracted feature vectors are mapped to a specific mental state by the ML classifier. A common supervised learning algorithm (i.e. k-Nearest Neighbors (kNN)) is applied on the vectors to classify them in either open or closed eyes classes. According to ML research, kNN reaches decent classification accuracies with low processing time [44]. A distance metric is used (e.g. Euclidean distance) to find the k nearest training vectors to each test input. The class label for the majority of the k vectors is assigned to the test input. kNN has two main parameters that need to be configured for the best results: 1) number of nearest neighbors (k) and 2)



(a) k = 1. **(b) k = 11.** **(c) k = 21.** **(d) k = 31.**

(e) k = 41. **(f) k = 51.** **(g) k = 61.** **(h) k = 71.**

(i) k = 81. **(j) k = 91.** **(k) k = 101.** **(l) k = 111.**

(m) k = 121. **(n) k = 131.** **(o) k = 141.** **(p) k = 151.**

Figure 3: The impact of the number of nearest neighbors (k) on the classes boundaries ($\lambda = 0.5$, distance metric: "Mahalanobis", Acc.: Accuracy, SR: Attack Success Rate, & Obj.: Objective Function Value).

the distance metric type (e.g. Euclidean). The objective of the security-training process is to find the optimal k (φ_1) and distance metric (φ_2) for maximizing accuracy and security.

Actuator: An actuator triggers an action or provides feedback to the subject. In our application, the actuator produces a warning feedback to the subject, whenever the classification results from the controller indicates closed eyes state.

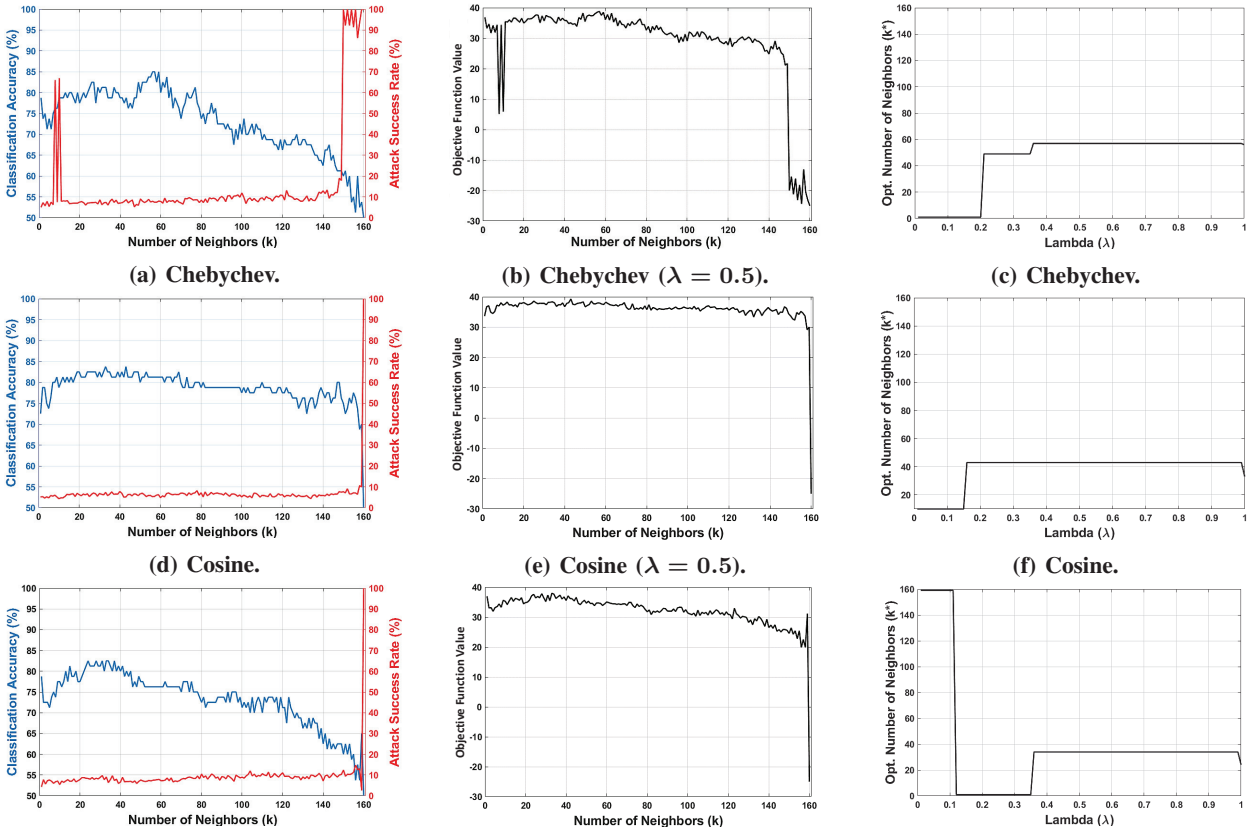
B. Threat Model

The threat model for the attack simulation is determined based on the adversary's attributes:

Adversary's Knowledge: In our experiment, a black-box attack scenario is assumed, where for each input test data, an adversary has only access to its class label determined by the ML classifier. The adversary has no knowledge about the AI application, such as classifier's type or architecture. Also, the training and testing data remains unknown for the adversary, and only the type of data (i.e. signal), its size, and range of the signal values are available.

Adversary's Capability: Full capability is assumed for the adversary, which means any signal with 80 data samples length and sample value range $[-687, 846]$ can be fed to the target system (the range is obtained from the minimum and maximum signal value in the whole EEG dataset). In this case, an adversarial sample may not represent any meaningful signal. Also, it is assumed that the adversary can test unlimited number of data samples on the target system.

Adversary's Goal: According to the simulation dataset, the adversary's goal is to guess an input data sample, to be classified as the open eyes classed, which is considered as a targeted attack. This attack causes the recognition of eyes



(a) Chebychev. (b) Chebychev ($\lambda = 0.5$). (c) Chebychev.

(d) Cosine. (e) Cosine ($\lambda = 0.5$). (f) Cosine.

(g) Euclidean. (h) Euclidean ($\lambda = 0.5$). (i) Euclidean.

Figure 4: Accuracy (Acc.) and attack success rate (SR) w.r.t. number of neighbors (k) (left column), objective function output w.r.t. k for lambda = 0.5 (middle column), and optimal number of neighbors w.r.t. lambda (right column).

open state in the subject by the kNN, while they are closed in reality. The misrecognition fails the driver's fatigue detection application and may cause accident.

Adversary's Strategy: Random sampling is chosen for generating adversarial samples. By access to the class labels, some random values between -687 and 846 (including the bounds) are assigned to the elements of an 80 long vector, which represents a signal. The obtained signal is tested on the target system, and this process is repeated till the random input to be classified as open eyes class.

C. Defense Mechanism

Tuning the kNN parameters is chosen as the only defense strategy, which depends on the performance of the ML classifier in rejecting the adversarial samples.

V. AI Application Testing Results

The experiment for security-tuning of AI applications has three main phases: 1) accuracy testing, 2) security testing, and 3) solving the optimization problem. Two separate test datasets are prepared for accuracy and security testing, respectively. The test data for accuracy testing is selected from the EEG dataset (Sec. IV-A), while the test data for security testing is randomly generated from scratch (Sec. IV-B).

A. Accuracy Testing

For evaluating the classification accuracy, $240 \times 1/3 = 80$ labeled test samples (40 samples for each class) are fed to the kNN classifier and according to the outputs, the accuracy for each given configuration is calculated. The testing results of the top three configurations with the highest accuracy and security are illustrated in Fig. 4. The left column diagrams in Fig. 4 show the kNN classification accuracy based on the number of neighbors, which can change from one to the total number of training samples. Each diagram illustrates the accuracy for a specific distance metric type.

B. Security Testing

For security testing, the adversary's behavior is simulated by generating signal samples through random guessing. The chance of the adversary to guess a sample inside the target class (i.e. adversary's success rate) is equal to the volume of the target class (i.e. open eyes class) divided by the volume of the whole data domain (1534^{80}), which is extremely large. Finding these volumes for classifier's without explicitly known classification function and large data domains is challenging. As seen in Table II, calculus-

and grid-based techniques can not be used for our application. In our experiment, we use a geometry-based method (i.e. Monte Carlo algorithm), to estimate the volume of the target class, and subsequently the adversary's success rate. In implementation of the Monte Carlo algorithm, thousand signal samples are generated and fed to the kNN. Based on the number of samples recognized as the target class data out of thousand samples, the attack SR is obtained. The left column diagrams in Fig. 4 show the perturbation attack success rate based on the number of neighbors (i.e. marked on the right vertical axis). Each diagram illustrates the SR for a specific distance metric type. In most cases, when the k is increased, the SR is also increased; however, SR for kNNs using some distance metrics, such as Chebychev, do not follow the pattern and show more randomness.

C. Optimization Problem

For solving the optimization problem, since obtaining the explicit objective function is costly and the search domain is relatively small, the gradient-free exhaustive search approach is selected (Table III). In this case, all possible configurations of kNN should be implemented. The number of neighbors can change from 1 to the total number of training samples (i.e. $240 \times 2/3 = 160$ for this experiment). Also, there are eleven common distance metric for finding samples in the vicinity of the test sample in kNN, which are *cityblock*, *Chebychev*, *correlation*, *cosine*, *Euclidean*, *Hamming*, *Jaccard*, *Mahalanobis*, *Minkowski*, *seuclidean*, and *Spearman*. For each number of neighbors choice, all the mentioned distance metrics are applied to evaluate the kNN accuracy and adversary's success rate. The middle column diagrams in Fig. 4 show the value of the objective function, $\lambda f(\varphi_1, \varphi_2) - (1-\lambda) g(\varphi_1, \varphi_2)$, when $\lambda = 0.5$, for all possible number of nearest neighbors (k) for the three selected distance metrics (Chebychev, cosine, and Euclidean). The k related to the highest value indicates the optimal number of neighbors for $\lambda = 0.5$ and the corresponding distance metric. Table IV lists the optimal number of neighbors for each distance metric, when lambda is equal to 0.5 (both accuracy and security are equally important in the optimization). It is seen that $k = 43$ and cosine distance metric provides the highest accuracy and security among all the configurations. Hamming and Jaccard distance metrics could not classify the EEG features (accuracy = 50% and attack success rate = 100% for any k), so they are eliminated from the results in Table IV. Finally, the right column diagrams in Fig. 4 illustrate the optimal k for various values of lambda ($\lambda \in [0, 1]$).

Table IV: Experiment Results ($\lambda = 0.5$, k*: optimal k, Acc.: Accuracy, SR: Attack Success Rate, & Obj.: Objective Function Value).

Distance Metric	k*	Acc. (%)	SR (%)	Obj.
Cityblock	15	80.00	4.58	37.71
Chebychev (Fig. 4a, 4b, & 4c)	57	85.00	7.41	38.80
Correlation	119	68.75	1.58	33.59
Cosine (Fig. 4d, 4e, & 4f)	**43**	**83.75**	**5.21**	**39.27**
Euclidean (Fig. 4g, 4h, & 4i)	34	82.50	6.37	38.07
Mahalanobis	11	78.75	6.19	36.28
Minkowski	28	82.50	7.32	37.59
Seuclidean	31	82.50	7.83	37.34
Spearman	127	68.75	4.12	32.32

VI. Discussion & Conclusion

In this paper, an analytical framework for security-tuning of AI applications is proposed. The framework finds the optimal configurations of ML algorithms in AI applications, which not only ensure some level of robustness against attacks, but also keeps the accuracy high. For validating the framework, an AI application is developed that detects eyes state in subjects by applying the kNN algorithm on their EEG signals. For testing the security of the application, a perturbation attack is simulated. The framework finds the optimal number of neighbors and distance metric type for the kNN to mitigate the attack. Considering only the classification accuracy of the application (Table IV), k = 56 and Chebychev distance metric are chosen as the configuration with the highest accuracy (i.e. 85%). However, the attack success rate on kNN using this configuration is equal to 7.92%, which is relatively high. Using our framework, the obtained optimal configuration is $k = 43$ and cosine distance metric (Table IV), with 83.75% accuracy and 5.21% vulnerability under the attack. It is seen by losing 1.25% accuracy, the kNN gains 2.71% more security. In applications that accuracy and security have the same level of importance ($\lambda = 0.5$), our suggested configuration has better performance. The experiment results show the framework ensures the maximum accuracy and security for ML applications, by obtaining their optimal configuration. At last, the usage of the framework can be generalized to test and optimize other ML-based AI applications. Recently, deep learning algorithms have drawn lots of attention in various fields, such as self-driving cars. These algorithms typically have several parameters (e.g. φ = {*filter size*, *number of filters*, *stride*, *padding*, *pooling method*, *number of fully connected layers*, ...} in convolutional neural networks [45]), which can not efficiently be set using empirical methods; while, the proposed analytical framework can be used as a standard methodology for tuning. However, due to large data (e.g. colored images) and parameters domains, more advanced security testing and optimization methods are required to find the optimal configuration, which is left for future work.

References

[1] Koosha Sadeghi, Ayan Banerjee, Javad Sohankar, and Sandeep K. S. Gupta. Safedrive: An autonomous driver safety application in aware cities. In *PerCom Workshops*, pages 1–6. IEEE, 2016.

[2] Junghyo Lee, Ayan Banerjee, and Sandeep K. S. Gupta. MT-Diet: Automated smartphone based diet assessment with infrared images. In *PerCom*, pages 1–6. IEEE, 2016.

[3] Prajwal Paudyal, Ayan Banerjee, and Sandeep K. S. Gupta. Sceptre: a pervasive, non-invasive, and programmable gesture recognition technology. In *Proceedings of the 21st International Conference on Intelligent User Interfaces*, pages 282–293. ACM, 2016.

[4] Javad Sohankar, Koosha Sadeghi, Ayan Banerjee, and Sandeep K. S.

Gupta. E-BIAS: A pervasive EEG-based identification and authentication system. *Proceedings of the 10th ACM Symposium on QoS and Security for Wireless and Mobile Networks (Q2SWinet)*.

[5] Koosha Sadeghi, Ayan Banerjee, Javad Sohankar, and Sandeep K. S. Gupta. Optimization of brain mobile interface applications using IoT. In *High Performance Computing (HiPC), 2016 IEEE 23rd International Conference on*, pages 32–41. IEEE, 2016.

[6] Sandeep K. S. Gupta, Tridib Mukherjee, and Krishna Kumar Venkatasubramanian. *Body area networks: Safety, security, and sustainability*. Cambridge University Press, 2013.

[7] Daniel Lowd and Christopher Meek. Adversarial learning. In *SIGKDD*, pages 641–647. ACM, 2005.

[8] Sandeep K. S. Gupta, Ayan Banerjee, Mohammad Javad Sohankar Esfahani, and Seyed Koosha Sadeghi Oskooyee. Brain-mobile interface optimization using internet-of-things, July 5 2018. US Patent App. 15/857,794.

[9] Koosha Sadeghi, Javad Sohankar, Ayan Banerjee, and Sandeep K. S. Gupta. A novel spoofing attack against electroencephalogram-based security systems. In *Advanced and Trusted Computing (ATC), The 14th IEEE Conference on*, pages 1–6, 2017.

[10] Scott Alfeld, Xiaojin Zhu, and Paul Barford. Data Poisoning Attacks against Autoregressive Models. In *AAAI*, pages 1452–1458, 2016.

[11] Koosha Sadeghi, Ayan Banerjee, Javad Sohankar, and Sandeep K. S. Gupta. Performance and Security Strength Trade-Off in Machine Learning Based Biometric Authentication Systems. In *ICMLA*, pages 1045–1048. IEEE, 2017.

[12] Mario Michael Krell and Su Kyoung Kim. Rotational data augmentation for electroencephalographic data. In *EMBC, 2017 39th Annual International Conference of the IEEE*, pages 471–474, 2017.

[13] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. *arXiv preprint arXiv:1412.6572*, 2017.

[14] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. *arXiv preprint arXiv:1805.06605*, 2018.

[15] Joseph D. Prusa and Taghi M. Khoshgoftaar. Deep Neural Network Architecture for Character-Level Learning on Short Text. In *FLAIRS Conference*, pages 353–358, 2017.

[16] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[17] Saleh Albelwi and Ausif Mahmood. A framework for designing the architectures of deep convolutional neural networks. *Entropy*, 19(6):242, 2017.

[18] Ibrahim M Alabdulmohsin, Xin Gao, and Xiangliang Zhang. Adding robustness to support vector machines against adversarial reverse engineering. In *Proc. of the 23rd ACM Intl. Conference on Information and Knowledge Management*, pages 231–240. ACM, 2014.

[19] Alex Kantchelian et al. Large-margin convex polytope machine. In *Advances in Neural Information Processing Systems*, pages 3248–3256, 2014.

[20] Battista Biggio et al. One-and-a-half-class multiple classifier systems for secure learning against evasion attacks at test time. In *International Workshop on Multiple Classifier Systems*, pages 168–180. Springer, 2015.

[21] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *S&P*, pages 582–597. IEEE, 2016.

[22] Xiaoyu Cao and Neil Zhenqiang Gong. Mitigating evasion attacks to deep neural networks via region-based classification. *arXiv preprint arXiv:1709.05583*, 2017.

[23] Lemonia Dritsoula, Patrick Loiseau, and John Musacchio. A Game-Theoretic Analysis of Adversarial Classification. *IEEE Transactions on Information Forensics and Security*, 12(12):3094–3109, 2017.

[24] Prajwal Paudyal, Junghyo Lee, Ayan Banerjee, and Sandeep K. S. Gupta. Dyfav: Dynamic feature selection and voting for real-time recognition of fingerspelled alphabet using wearables. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces*, pages 457–467. ACM, 2017.

[25] Junghyo Lee, Prajwal Paudyal, Ayan Banerjee, and Sandeep K. S. Gupta. FIT-EVE&ADAM: Estimation of velocity & energy for automated diet activity monitoring. In *ICMLA*, pages 1071–1074. IEEE, 2017.

[26] Koosha Sadeghi, Ayan Banerjee, Javad Sohankar, and Sandeep K. S. Gupta. Geometrical analysis of machine learning security in biometric authentication systems. In *ICMLA*, pages 309–314. IEEE, 2017.

[27] Miklós Simonovits. How to compute the volume in high dimension? *Mathematical programming*, 97(1-2):337–374, 2003.

[28] Koosha Sadeghi, Ayan Banerjee, Javad Sohankar, and Sandeep K. S. Gupta. Toward Parametric Security Analysis of Machine Learning Based Cyber Forensic Biometric Systems. In *ICMLA*, pages 626–631. IEEE, 2016.

[29] Battista Biggio et al. Evasion attacks against machine learning at test time. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 387–402. Springer, 2013.

[30] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017.

[31] Denis Foo Kune et al. Ghost talk: Mitigating EMI signal injection attacks against analog sensors. In *S&P*, pages 145–159. IEEE, 2013.

[32] Simon Eberz et al. Broken hearted: How to attack ECG biometrics. *Proc. of the 2017 Network and Distributed Systems Symposium*, 2017.

[33] Battista Biggio, Giorgio Fumera, and Fabio Roli. Pattern recognition systems under attack: Design issues and research challenges. *Intl. J. of Pattern Recognition and Artificial Intelligence*, 28(07), 2014.

[34] Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *S&P*, pages 463–480, 2015.

[35] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on Machine Learning*, pages 833–840. Omnipress, 2011.

[36] Lemonia Dritsoula, Patrick Loiseau, and John Musacchio. A game-theoretic analysis of adversarial classification. *Information Forensics and Security, IEEE Transactions on*, 12(12):3094–3109, 2017.

[37] Koosha Sadeghi, Ayan Banerjee, Javad Sohankar, and Sandeep K. S. Gupta. Geometrical Analysis of Machine Learning Security in Biometric Authentication Systems. In *ICMLA*, pages 309–314. IEEE, 2017.

[38] Sandeep K. S. Gupta, Ayan Banerjee, Seyed Koosha Sadeghi Oskooyee, and Mohammad Javad Sohankar Esfahani. Framework for security strength and performance analysis of machine learning based biometric systems, October 18 2018. US Patent App. 15/949,469.

[39] José L. Ribeiro Filho, Philip C. Treleaven, and Cesare Alippi. Genetic-algorithm programming environments. *Computer*, 27(6):28–43, 1994.

[40] Gerwin Schalk, Dennis J. McFarland, Thilo Hinterberger, Niels Birbaumer, and Jonathan R. Wolpaw. BCI2000: a general-purpose brain-computer interface (BCI) system. *Biomedical Engineering, IEEE Transactions on*, 51(6):1034–1043, 2004.

[41] Ary L. Goldberger et al. Physiobank, physiotoolkit, and physionet components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215—-e220, 2000.

[42] Luis Fernando Nicolas-Alonso and Jaime Gomez-Gil. Brain computer interfaces, a review. *Sensors*, 12(2):1211–1279, 2012.

[43] Koosha Sadeghi, Junghyo Lee, Ayan Banerjee, Javad Sohankar, and Sandeep K. S. Gupta. Permanency analysis on human electroencephalogram signals for pervasive brain-computer interface systems. In *EMBC, 39th Annual Intl. Conf. of the IEEE*, pages 767–770, 2017.

[44] Zhenyun Deng, Xiaoshu Zhu, Debo Cheng, Ming Zong, and Shichao Zhang. Efficient kNN classification algorithm for big data. *Neurocomputing*, 195:143–148, 2016.

[45] U. Rajendra Acharya, Shu Lih Oh, Yuki Hagiwara, Jen Hong Tan, and Hojjat Adeli. Deep convolutional neural network for the automated detection and diagnosis of seizure using EEG signals. *Computers in biology and medicine*, 100:270–278, 2018.