Training Set Camouflage

Ayon Sen[1], Scott Alfeld[2], Xuezhou Zhang[1], Ara Vartanian[1], Yuzhe Ma[1], and Xiaojin Zhu[1]

[1] University of Wisconsin-Madison
{ayonsn, zhangxz1123, aravart, yzm234, jerryzhu}@cs.wisc.com
[2] Amherst College
salfeld@amherst.edu

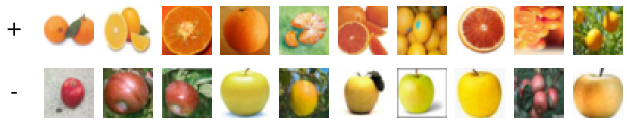
Abstract. We introduce a form of steganography in the domain of machine learning which we call training set camouflage. Imagine Alice has a training set on an illicit machine learning classification task. Alice wants Bob (a machine learning system) to learn the task. However, sending either the training set or the trained model to Bob can raise suspicion if the communication is monitored. Training set camouflage allows Alice to compute a second training set on a completely different – and seemingly benign – classification task. By construction, sending the second training set will not raise suspicion. When Bob applies his standard (public) learning algorithm to the second training set, he approximately recovers the classifier on the original task. Training set camouflage is a novel form of steganography in machine learning. We formulate training set camouflage as a combinatorial bilevel optimization problem and propose solvers based on nonlinear programming and local search. Experiments on real classification tasks demonstrate the feasibility of such camouflage.

Keywords: Machine Teaching · Adversarial Learning · Steganography

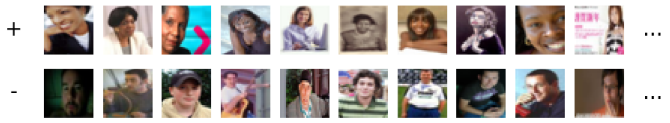
1 Introduction

Look at the classification training set shown in Figure 1a. The top row contains instances of class positive (+), and the bottom shows instances of class negative (-). These images can be fed into a machine learner to learn a model which will successfully classify future, previously unseen instances (images) as + or -. If you think that the task is fruit image classification (orange vs. apples) then you have already been successfully fooled, in a sense to be made precise below. The actual intended task is to classify woman vs. man, with samples shown in Figure 1b. Indeed, a standard logistic regression learner [26] trained on only the images in Figure 1a achieves high gender classification accuracy on the images in Figure 1b.

In this paper, we consider an agent Alice who has a secret classification task (e.g., classifying images of women and men) and a corresponding private training set (women and men images). Alice wants to train a second agent, Bob, on the secret task. However, the communication channel between them has an eavesdropper we refer to as a third agent Eve. Eve takes the role of a data verifier,



(a) Camouflaged training set



(b) Secret classification task

Fig. 1: Example of training set camouflage

who will terminate communication (and refuse to deliver the data to Bob) if she is suspicious of what Alice is sending. Sending the private training set would reveal Alice's intention; sending the model parameters directly will also raise suspicion. Alice must camouflage the communication for it to look mundane to Eve, while avoiding excessive coding tricks with Bob beforehand. In the present work, we show how Alice can construct a camouflaged training set on a *cover task* which (i) does not look suspicious to Eve, and (ii) results in Bob learning an accurate model for the secret task. In the previous example, Eve noticed that Alice sent images of apples and oranges which seems benign, and knew nothing of Alice's secret task of women vs men.

Hiding information in plain sight such that its presence is not suspected is known as steganography. Steganography is not new. In the fifth century BCE messengers would have their heads shaved and a message written on their scalp. Regrowing their hair served to hide the message which would only be revealed because the intended recipient knew to shave the messenger's head [40]. In more modern times, steganographic techniques are used to detect unauthorized distribution of digital media [15].

Note that, steganography is different from crypotgraphy [31, 61], where the goal is to hide the data content. In cryptography, the communicating agents have access to some particular key (pairs) which is used to encrypt and decrypt data. Cryptography cannot be used if someone monitoring the data can alter the data or stop the data transmission entirely. In such cases, steganography becomes important because we do not want any intervening eavesdropper to become suspicious and stop the data transmission.

The role and capabilities of the eavesdropper are key in selecting how to hide information. Eve can be either passive and merely observes traffic [13], or active and tries to modify the hidden message [51, 14]. In this manuscript we assume a

passive observer, whose only ability is to refuse to deliver Alice's message. To our knowledge, steganography for machine learning in this context is new. In the area of adversarial learning [5], however, much work has been done investigating how an agent can assert control over a learner by manipulating input data. We note that training set camouflage differs from so called training-time or "poisoning attacks" [36] in two primary ways: (i) Alice aims to communicate information to Bob about a potentially completely unrelated task, not affect his behavior on the original task and (ii) Alice specifically aims to avoid detection by Eve.

Due to the widespread use of machine learning in sensitive fields ranging from social media to health care, the study of the security ramifications of using ML techniques is well studied [3, 47]. The work presented herein adds to this conversation, as we reveal an additional avenue of attack. For example, Bob might be a model that classifies job applicants as "should hire" and "shouldn't hire". The company may have many records (collected over years) of job applicants and how they performed. It is expected from Alice to select a subset of these records and present to Bob, with the idea that training on the complete set is too time consuming. But Alice may be a malicious agent and wants Bob to actually learn some additional bias (e.g., racial, gender etc.). In such a scenario, Alice will select a subset of records that satisfies her goals while Eve's responsibility is to verify the data sent by Alice to Bob. Our specific contributions in this paper are as follows: (i) We propose a general mathematical framework for defining how Alice can achieve training set camouflage. (ii) We formulate a nonlinear-program based approach for performing Alice's task for a general class of learner (Bob) and eavesdropper (Eve), and two combinatorial-search based approaches for arbitrary learners/eavesdroppers.

2 Training Set Camouflage

In this section we describe the three agents Bob, Alice and Eve, and formulate a camouflage optimization problem for Alice, parametrized by Bob and Eve's definitions.

The agent Bob uses a standard learning algorithm $\mathcal{A} : \mathcal{D} \mapsto \mathcal{H}$ which, given a training set D, learns a hypothesis $\mathcal{A}(D)$ in a hypothesis space $\mathcal{H}$. The resulting hypothesis maps instances in the input space $\mathcal{X}$ to the output space $\mathcal{Y}$. This can be multi-class classification (three or more classes) or regression, though in the present work we focus on binary classification. We assume that Bob's learning algorithm is "open source". That is, all information about $\mathcal{A}$ is known to all agents. However, Bob and Alice have shared knowledge on class naming: which class is positive and which negative. For K-class classification this shared knowledge requires $O(K \log K)$ bits, as Alice must communicate a mapping from K classes to K classes. For example, when Alice sends Bob orange and apple images for the secret task of woman vs man, Alice must communicate to Bob whether orange maps to woman and apple to man, or vice versa.

Alice is an agent who wants to train Bob. She has a secret classification task and the corresponding private dataset D_S. In addition, she has access to

a public pool of n instances $\mathcal{C} = \{(\mathbf{x}_i, y_i)_{1:n}\}$ (the *camouflage pool*) drawn i.i.d. from $\mathbb{Q}_{(\mathbf{x},y)}$ which we call the *cover data distribution*. Note that this is not the distribution from which D_S is drawn. In the preceding example, $\mathbb{Q}_{(\mathbf{x},y)}$ is the distribution over orange and apple images, whereas D_S is a collection of photographs of women and men.

Alice seeks to select a camouflaged training set $D \subset \mathcal{C}$ which she will send to Bob for training. Alice wants Bob to succeed on the secret task, thus she seeks to find a D which minimizes the risk of Bob's resulting model:

$$\mathcal{L}_{\mathcal{A}}(D) = \frac{1}{|D_S|} \sum_{(\tilde{\mathbf{x}},\tilde{y}) \in D_S} \ell(\mathcal{A}(D)(\tilde{\mathbf{x}}), \tilde{y})$$

where $\ell(\hat{y}, y)$ is a loss function. Note the private D_S is used to define the quality of D. Also, we note that D_S and D are in different domains.

Eve is a passive eavesdropper. She observes the dataset D which Alice sends to Bob, but does not alter it. Her goal is to determine if D is normal or suspicious. Specifically, Eve is suspicious of any data unless it appears to be an i.i.d. sample drawn from $\mathbb{Q}_{(\mathbf{x},y)}$. She has access to the camouflage pool $\mathcal{C}$, but like Alice she cannot draw additional instances from $\mathbb{Q}_{(\mathbf{x},y)}$. Eve uses $\mathcal{C}$ and a detection function $\Psi : \mathcal{D} \times \mathcal{D} \mapsto \mathbb{R}_+$ to examine D. Here $\mathcal{D}$ is the domain of all possible training sets. She flags D as suspicious if $\Psi(\mathcal{C}, D) \geq 0$. It is useful to think of Ψ as a two-sample statistical test that aims to determine whether $\mathcal{C}, D$ come from the same underlying distribution. We envision Alice as Eve's adversary. As with Bob, we assume that Eve is "open source". Namely, Alice can see Eve's detection function Ψ.

The agents Alice and Eve have conflicting goals. Alice aims to secretly communicate to Bob by way of constructing a training set, while Eve is suspicious and will flag any data sent to Bob if it looks suspect. Bob dutifully learns a model and for our purposes has no explicit intentions. We make explicit each Agent's knowledge in Table 1.

Table 1: Information available to different agents

Agent	**Secret Set** D_S	**Camouflage Pool** $\mathcal{C}$	**Bob's Learner** $\mathcal{A}$	**Detection Function** Ψ	**Camouflaged Training Set** D
Bob	No	Yes/No	Yes	Yes/No	Yes
Alice	Yes	Yes	Yes	Yes	Yes
Eve	No	Yes	Yes	Yes	Yes

With the agents defined, we can now formulate Alice's goal:

$$\begin{aligned} &\operatorname*{argmin}_{D \subset \mathcal{C}} \frac{1}{|D_S|} \sum_{(\tilde{\mathbf{x}},\tilde{y}) \in D_S} \ell(\mathcal{A}(D)(\tilde{\mathbf{x}}), \tilde{y}) \\ &\text{s.t. } \Psi(\mathcal{C}, D) < 0 \end{aligned} \tag{1}$$

That is, she seeks a camouflaged training set D from the cover data pool. D should not be flagged as suspicious by Eve. D should also make Bob learn well, similar to as if Alice directly gave Bob her private data set D_S. An example of the training set camouflage in action is shown in Figure 2.

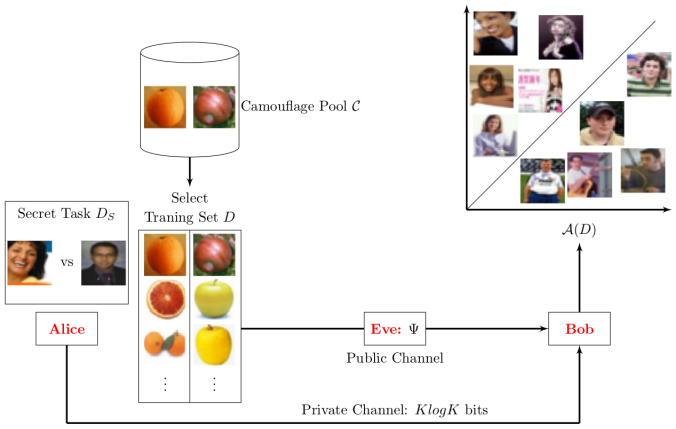


Fig. 2: Training set camouflage framework. We show the three agents along with the classification task, camouflage pool, camouflage training set and Eve's detection function

3 Solving the Camouflage Problem

In this section, we propose three methods of solving the optimization problem defined in (1). We first show how the optimization problem can be reduced to a nonlinear programming problem for a broad class of learners. We relax the resulting optimization problem to one which is computationally efficient to solve. We then present two combinatoric methods as heuristic methods applicable to any learner.

3.1 Nonlinear Programming (NLP)

We assume Bob's machine learning algorithm $\mathcal{A}$ solves a convex optimization problem. Specifically, Bob performs regularized empirical risk minimization. This covers a wide range of learners such as support vector machines [23], logistic regression [26], and ridge regression [24]. Let Θ be Bob's hypothesis space, ℓ his loss function, and λ his regularization parameter, respectively. Let $m := |D|$ be given. We convert Alice's optimization problem (1) into a nonlinear programming problem as follows.

Step 1. Using the definition of Bob, we rewrite (1) as

$$\begin{aligned} \min_{D\subset\mathcal{C},\hat{\theta}\in\Theta} \quad & \frac{1}{|D_S|}\sum_{(\tilde{\mathbf{x}},\tilde{y})\in D_S}\ell(\hat{\theta},\tilde{\mathbf{x}},\tilde{y}) \\ \text{s.t.} \quad & \hat{\theta}=\operatorname*{argmin}_{\theta\in\Theta}\sum_{(\mathbf{x},y)\in D}\ell(\theta,\mathbf{x},y)+\frac{\lambda}{2}\|\theta\|^2 \\ & \Psi(\mathcal{C},D)<0, \\ & |D|=m. \end{aligned} \tag{2}$$

We make note that in both levels of this bilevel optimization problem (the upper and lower levels corresponding with Alice and Bob, respectively) $\ell(\cdot)$ is being minimized. That is, Alice and Bob both seek to minimize the loss of Bob's resulting model. Due to its combinatorial nature, this is a computationally difficult problem to solve.

Step 2. Since Bob's learning problem (the lower level optimization problem) is assumed to be convex, satisfying its Karush-Kuhn-Tucker (KKT) conditions is necessary and sufficient for a point to be optimal [63, 44]. Thus we replace the lower level optimization problem in (2) with the KKT conditions to obtain a single-level optimization problem:

$$\begin{aligned} \min_{D\subset\mathcal{C},\hat{\theta}\in\Theta} \quad & \frac{1}{|D_S|}\sum_{(\tilde{\mathbf{x}},\tilde{y})\in D_S}\ell(\hat{\theta},\tilde{\mathbf{x}},\tilde{y}) \\ \text{s.t.} \quad & \sum_{(\mathbf{x},y)\in D}\nabla\ell(\hat{\theta},\mathbf{x},y)+\lambda\hat{\theta}=0, \\ & \Psi(\mathcal{C},D)<0, \\ & |D|=m. \end{aligned} \tag{3}$$

While now a single level optimization problem, selecting a subset $D\subset\mathcal{C}$ is still a combinatorial problem and computationally expensive to solve. In what comes next we relax this problem to one of continuous optimization.

Step 3. We introduce binary indicator variable b_i for each instance $(\mathbf{x}_i,y_i)\in\mathcal{C}$. A value of 1 indicates that the instance is a member of the training set D. Also dropping the hat on $\hat{\theta}$ for simplicity. This yields:

$$\begin{aligned} \min_{\theta\in\Theta;b_1,\ldots,b_{|\mathcal{C}|};b_i\in\{0,1\}} \quad & \frac{1}{|D_S|}\sum_{(\tilde{\mathbf{x}},\tilde{y})\in D_S}\ell(\theta,\tilde{\mathbf{x}},\tilde{y}) \\ \text{s.t.} \quad & \sum_{i=1}^{n}b_i\nabla\ell(\theta,\mathbf{x}_i,y_i)+\lambda\theta=0, \\ & \Psi(\mathcal{C},\{b_i(\mathbf{x}_i,y_i)|(\mathbf{x}_i,y_i)\in\mathcal{C},b_i\neq 0\})<0 \\ & \sum_{i=1}^{n}b_i=m. \end{aligned} \tag{4}$$

This is known as a Mixed Integer Non-Linear Optimization Problem (MINLP) [12]. MINLP problems are generally hard to solve in practice. However, phrasing the problem in this way yields a natural relaxation. Namely we relax b_i to be continuous in $[0, 1]$, resulting in the following non-linear optimization problem:

$$\begin{aligned} \min_{\theta \in \Theta; b_1, \ldots, b_n \in [0,1]} \quad & \frac{1}{|D_S|} \sum_{(\tilde{\mathbf{x}}, \tilde{y}) \in D_S} \ell(\theta, \tilde{\mathbf{x}}, \tilde{y}) \\ \text{s.t.} \quad & \sum_{i=1}^{n} b_i \nabla \ell(\theta, \mathbf{x}_i, y_i) + \lambda \theta = 0, \\ & \Psi(\mathcal{C}, b_1, \ldots, b_{|\mathcal{C}|}) < 0, \\ & \sum_{i=1}^{n} b_i = m. \end{aligned} \tag{5}$$

Note that in this equation we scale the gradient of the loss function for each $(\mathbf{x}_i, y_i)$ by the corresponding b_i. This b_i indicates the importance of an instance in the training set. In essence, the learner is training on a "soft" version of the dataset, where each training example is weighted. Similarly, when calculating the detection function we weigh each instance in the training set by its corresponding b_i. The exact nature of this weighing depends on the detection function itself. We further note that the nonlinear optimization problem is non-convex. As such, Alice must seed her solver with some initial $\{b_i\}$. This is discussed further in Section 4.

After solving this (continuous) optimization problem, Alice must round the $\{b_i\}$'s into binary indicators so that she can select a training set to send to Bob. Alice uses a rounding procedure that proposes $m + 1$ candidate training sets $D^{(1)}, \ldots, D^{(m+1)}$ from the continuous solution $\{b\}$. The candidate training sets include (1) the training set $D^{(1)}$ consisting of the m items with the largest b values, (2) the seed training set before running optimization, (3) $m-1$ other training sets that "interpolate" between 1 and 2. Alice then checks $D^{(1)}, \ldots, D^{(m+1)}$ for feasibility (satisfying Ψ) and picks the best one. Note the seed training set is feasible, hence Alice is guaranteed to have a solution. The interpolation scheme ensures that Alice will find a solution no worse than the seed set.

Concretely, let S be the m-item seed training set and $\mathcal{C} \backslash S$ be the remaining items. Alice sorts items in S by their b values. Separately, Alice sorts items in $\mathcal{C} \backslash S$ by their b values. Then, Alice starts from S and sequentially swaps the least-valued item in S with the largest-valued item in $\mathcal{C} \backslash S$. She performs m swaps. This produces the $m+1$ candidate training sets, including the original S. It can be shown that the m items with the largest b values will be one of the training sets.

3.2 Uniform Sampling

For any learner Bob, even one which does not solve a convex empirical risk minimizing problem discussed above, Alice has a simple option for finding a

training set. Let Alice's budget B denote the number of times Alice is able to train the classifier $\mathcal{A}$. She first creates B training sets $D^{(1)}, \ldots, D^{(B)}$, each by sampling m points uniformly without replacement from her camouflage pool $\mathcal{C}$, such that each $D^{(j)}$ successfully bypasses Eve i.e., $\Psi(\mathcal{C}, D^{(j)}) < 0$. Among these B training sets, she then picks the $D^{(j)}$ with the lowest objective value in (1). This procedure captures what Bob would learn if given each feasible training set.

3.3 Beam Search

We now describe a heuristic beam search algorithm [53] to approximately solve Alice's optimization problem (1). This process is similar to uniform sampling, described above, but instead of independently generating a new training set every time, Alice performs a local search to augment a proposed training set incrementally.

The state space consists of all training sets $D \subset \mathcal{C}$ such that $|D| = m$ and $\Psi(\mathcal{C}, D) < 0$. Two training sets that differ by one instance are considered neighbors. For computational efficiency, we do not consider the entire set of neighbors at each step. Instead, we evaluate a randomly selected subset of neighbors for each training set in the beam. The beam $\mathcal{D}$ is initialized by selecting w training sets at random. The width (w) of the beam is fixed beforehand. From the union of evaluated neighbors and training sets in the current beam, we select the top w training sets (based on the value of the objective function in (1)) to reinitialize the beam and discard the rest. Note that training sets which would be flagged by Eve are not present in the statespace (because Alice has full knowledge of Eve, she need not consider any set that Eve would reject). We continue the search process until a pre-specified search budget B (number of times the classifier $\mathcal{A}$ is trained) is met. Algorithm 1 shows the search procedure with random restarts.

Algorithm 1 Beam Search for Solving the Camouflage Problem

Input: Camouflage Pool: $\mathcal{C}$, Risk: $\mathcal{L}_{\mathcal{A}}$, Beam Width: w, Budget: B, Neighborhood Function: $\mathcal{N}$, Size: m, Detection Function: Ψ, Restarts: R
for $r = 1 \rightarrow R$ **do**
  $\mathcal{D} \leftarrow w$ randomly selected subsets of size m from $\mathcal{C}$ such that $\Psi(\mathcal{C}, D) < 0$
  while budget B/R not exhausted **do**
    $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{N}(\mathcal{D}, \mathcal{C}, \Psi)$, the neighbors
    $\mathcal{D} \leftarrow w$ training sets from $\mathcal{D}$ with smallest $\mathcal{L}_{\mathcal{A}}(D)$ values
  end while
end for
return the best D found within total budget

4 Experiments

We investigated the effectiveness of training set camouflage through empirical experiments on real world datasets. Our results show that camouflage works on a variety of image and text classification tasks: Bob can perform well on the secret task after training on the camouflaged training set, and the camouflaged training set passes Eve's test undetected. We start by discussing the three agents.

Bob. We considered the logistic regression learning algorithm for Bob. Logistic regression is a popular learner and is regularly used in practice. Bob set the weight of the regularization parameter to 1.

Eve. The training set camouflage framework is general with respect to Eve's detection function. For our experiments we used Maximum Mean Discrepancy (**MMD**) [20] as the core of Eve's detection function. We used **MMD** as it is a popular and widely used two-sample test [17]. Unfortunately **MMD** cannot be directly applied to the camouflage framework as its application requires that the two samples have the same size. We introduce **MMD** and how Eve used it in Appendix A. The level-α for this detection function was set to 0.05 (i.e., the probability of incorrectly rejecting a benign training set is 5%).

Alice. We considered three different Alices. Each of them used one of the proposed solvers. For each secret task, Alice had access to multiple camouflage candidate tasks. Alice can run her solver on each of these tasks separately and then select the best one, but this would be time consuming and thus instead she started by identifying a suitable camouflage task. For this purpose, all three Alices used uniform sampling (as this is the easiest algorithm to implement, and makes the weakest assumptions) with a search budget of $80,000$ (divided equally among candidate tasks). This meant that Alice stopped after training the logistic regression learner $80,000$ times. For each candidate task Alice identified a training set using this budget. Then she selected the best task (as her cover task) based on the loss on the secret set.

Next, all three Alices used their respective solvers (NLP, beam search and uniform sampling) to find a camouflaged training set. We assumed that all of them were allotted a fixed amount of time for this purpose. This time was set as the time required to run the NLP solver.

The Alice who used the NLP solver seeded the solver with the camouflaged training set found during the candidate task identification phase. The Alice who used the beam search solver performed random restarts each with a per-restart budget of $B/R = 16,000$. Here the width of the beam was $w = 10$ and for each training set in the beam, 50 randomly selected neighbors were evaluated during each iteration. It should be noted that both beam search and uniform sampling are stochastic in nature. We run the Alices who used these solvers five times. We then report the average. Alice constructed camouflaged training sets of size $m = 2, 20$ and 50, and set the loss ℓ to logistic loss with natural logarithm. All experiments were run on an Intel(R) Core(TM) i7-7700T CPU @2.90GHz machine, using one thread.

Evaluation metrics. As is standard to estimate generalization performance of a learned model, we used a separate test set, generated from the same dis-

tribution as the secret set D_S and not known to any agent, to estimate Bob's generalization error when trained on Alice's camouflaged training set D. We compare these values to two additional quantities: ("random") when Bob is trained on a uniform sample of size m from the cover data distribution, which we expect to perform poorly; and ("oracle") when Bob is trained directly on Alice's secret set D_S, ignoring Eve's presence. The oracle gives us an estimate on how much performance Bob is losing due to using the camouflage framework to fool Eve.

4.1 Datasets

We performed experiments for four secret tasks: WM (CIFAR-100 [41]), GP (OpenImages [39]), CA (20-newsgroups [28]) and DR (All The News dataset [60]). The two letters in the acronym represent the two classes in the corresponding task (see Table 2). The first two tasks were image classification while the remaining two were text classification. For the image tasks we selected eight candidate cover tasks. Six of them were from the MNIST handwritten digits: 17, 71, 25, 52, 69 and 96. The other two were from the CIFAR-100 dataset: OA and AO. Similarly for the text tasks we also selected eight candidate cover tasks. All of them were from the 20-newsgroups dataset: BH, HB, IM, MI, AM, MA, MX and XM. As before the acronyms here represent the class names.

Table 2: Summary of secret sets and camouflage pools.

Dataset	**Type**	**# Features**	**class 1**	**class 2**	**# class 1**	**# class 2**
WM	Image	2048	**w**oman	**m**an	500	500
GP	Image	2048	hand**g**un	**p**hone	400	400
CA	Text	300	**c**hristian	**a**theist	599	480
DR	Text	300	**d**emocratic	**r**epublican	800	800
17	Image	2048	digit **1**	digit **7**	600	600
25	Image	2048	digit **2**	digit **5**	600	600
69	Image	2048	digit **6**	digit **8**	600	600
OA	Image	2048	**o**range	**a**pple	600	600
BH	Text	300	**b**aseball	**h**ockey	994	999
IM	Text	300	**i**bm	**m**ac	982	963
AM	Text	300	**a**utos	**m**otorcycles	990	996
MX	Text	300	**m**s-windows	windows **x**	985	988

For images we used ResNet [22] to generate feature vectors of dimension 2048. For this purpose we removed the output layer and used the values found in the penultimate layer of the network. For text we used Word2Vec [50] to generate feature vectors of dimension 300 by averaging over the word vectors in an article. We also removed punctuation and stop words before generating the word vectors. A summary of the secret sets and camouflage pools can be found in Table 2. As mentioned previously, we kept a held out test set for each of the secret tasks. The number of class 1 and class 2 instances were 100/100,

100/100, 398/319 and 200/200 respectively for WM, GP, CA and DR. Here the two numbers (num1/num2) represent the number of instances in class1 and class2 respectively.

Table 3: Logistic loss ($\frac{1}{|D_S|}\sum_{(\tilde{\mathbf{x}},\tilde{y})\in D_S}\log(1+\exp(-\tilde{y}w^\top\tilde{\mathbf{x}}))$) after performing Uniform Sampling search with search budget $10,000$ for image secret tasks. The best results for each secret task is shown in bold.

m	Secret \ Camouflage	17	71	25	52	69	96	OA	AO
2	WM	0.671	0.631	0.643	0.638	0.671	0.640	**0.606**	0.647
	GP	0.481	0.541	0.458	**0.443**	0.516	0.463	0.541	0.558
20	WM	0.790	0.611	0.672	0.688	0.798	0.679	**0.584**	0.731
	GP	0.480	0.510	0.433	0.390	0.632	**0.337**	0.510	0.531
50	WM	0.874	0.614	0.705	0.772	1.116	0.802	**0.606**	0.856
	GP	0.565	0.479	0.473	**0.387**	1.047	0.421	0.479	0.506

Alice first selected a suitable camouflage task for each of the secret tasks. For each candidate task she used a search budget of $10,000$ (for a total of $80,000$ budget). The results of this phase are shown in Table 3 and 4. For $m = 2$ the camouflage tasks selected for WM, GP, CA and DR were OA, 52, XM and HB respectively. Similarly for $m = 20$ the selected camouflage tasks were OA, 96, BH and BH respectively. OA, 52, HB and BH were the selected camouflage tasks respectively when $m = 50$. It should be noted that the logistic error reported in the tables are large (> 0.693) in some cases indicating that some of these cover tasks will perform worse than random chance on secret tasks. However, this was not true for the selected cover tasks. The top three camouflaged training sets for GP ($m = 20$) identified during this phase are shown in Figure 3.

Table 4: Logistic loss after performing Uniform Sampling search with search budget $10,000$ for text secret tasks. The best results for each secret task is shown in bold.

m	Secret \ Camouflage	BH	HB	IM	MI	AM	MA	MX	XM
2	CA	0.6845	0.6846	0.6868	0.6862	0.6861	0.6862	0.6844	**0.6843**
	DR	0.6889	**0.6886**	0.6891	0.6893	0.6888	0.6887	0.6890	0.6894
20	CA	**0.672**	0.673	0.676	0.675	0.676	0.674	0.675	0.675
	DR	**0.681**	0.684	0.682	0.683	0.682	0.682	0.682	0.683
50	CA	0.671	**0.669**	0.672	0.671	0.674	0.670	0.671	0.671
	DR	**0.677**	0.681	0.679	0.680	0.678	0.680	0.683	0.679

(a) Samples from the secret task Handgun vs. Phone (GP)

(b) Camouflaged training set using 9 vs. 6

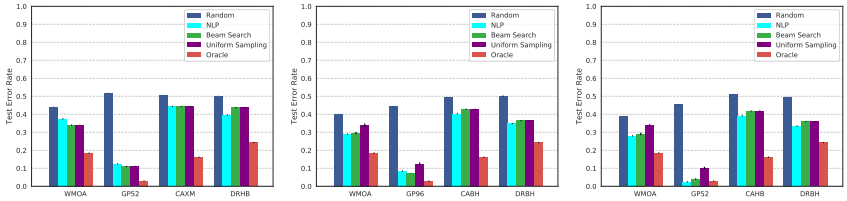
(c) Camouflaged training set using 5 vs 2

(d) Camouflaged training set using 2 vs. 5

Fig. 3: Samples of GP secret set, and the top three camuflaged training set found during the candidate selection phase for $m = 20$.

4.2 Results

For $m = 2$, the NLP solver ran for 23363, 33763, 48 and 44 seconds respectively for WMOA, GP96, CABH and DRBH. The solver ran for 29150, 65697, 50 and 57 seconds respectively for WMOA, GP96, CABH and DRBH when $m = 20$. The run time was 39656, 171637, 126 and 193 seconds respectively when $m = 50$. We present our results for all three solvers in Figure 4. For the text secret tasks, Alice could not find a better camouflaged training set using either beam search or uniform sampling than the one found during the initial run of uniform sampling (with a total budget of 80,000). To explore the sensitivity of beam search and uniform sampling regarding the time budget, we ran both solvers for an additional two hours. But the results only improved marginally. We observe that Alice, using any of the three solvers can find much better camouflage training sets than random and in many cases approach oracle performance. Note



(a) $m = 2$ (b) $m = 20$ (c) $m = 50$

Fig. 4: Test error rates found by solving the camouflage framework. We also show random and oracle error for comparison. Error bars are also shown. All three solvers were run for the same amount of time.

that Alice's solutions do not trigger Eve's suspicion function. This shows that such subterfuges are plausible in practice and can actually yield good results from Alice's point of view. We note that Alice yields the best results when $m = 50$ in most of the experiments, but this may not hold for larger values of m e.g., when m is equal to the size of the camouflage pool. We plan to run further experiments to understand the effect of m.

Figure 1 shows the result of WMOA when Bob's learner is logistic regression and the solver is NLP ($m = 20$). Visually, the camouflaged training set D bears no resemblance to the secret training set D_S. This is true for the text camouflage experiments as well, where articles in the camouflaged training sets have no obvious semantic connection to the secret task. See Table 5 for results on the text experiment CABH. This is indeed bad news for human Eves: not only did camouflage fooled MMD detector, it will also likely fool human inspectors.

Table 5: Camouflage results for the CABH experiment with $m = 20$ for the NLP solver

Sample of Secret Set		**Sample of Camouflaged Training Set**	
Class	**Article**	**Class**	**Article**
Christianity	...Christ that often causes christians to be very critical of themselves and other christinas. We...	Baseball	...Boys, hats off to any Cubs fan who can actually muster up the courage to put down Braves fans. I...
	...I've heard it said that the accounts we have of Christs life and ministry in the Gospels were...		... NPR's Morning Edition aired a report this morning to get (4/19) on Hispanic/Latin American players in MLB...
Atheism	...This article attempts to provide a general introduction to atheism. Whilst I have tried to be...	Hockey	... Would Kevin Dineen play for the Miami Colons??? As a Flyers fan, I resent you making Kevin Dineen...
	...Science is wonderful at answering most of our questions. I'm not the type to question scientific...		...Good point - there haven't even been any recent posts about ULf! Secretly, I'm convinced that he is responsible ...

5 Related Work

Concealing the existence of messages is known as steganography. One illustration of steganography (first presented in 1983 in [55]) is where prisoners Alice and Bob wish to devise an escape plan. All their communication is observed by the

adversary (the warden, Eve) who will thwart their plan as soon as she detects any sign of hidden message.

Steganography has multiple real-world applications including secret communication [64], feature tagging elements [48], and copyright protection [48]. Although many different data formats can be used for steganography, images [51, 29] are by far the most popular format due to their popularity on the internet and the fact that they are rich with noise-insensitive information. Image steganography can be broadly classified into spatial domain, transform domain, spread spectrum and model based [56], and has been thoroughly studied. On the other side, steganalysis is the study of detecting the existence of hidden messages (using steganography). Identifying such messages in text by looking at patterns in texts, odd language and unusual white space was explored in [14]. The authors of [18, 32, 51] explore the detection of hidden messages in images.

A study of steganography from a complexity-theoretic point of view is presented in [25, 52]. An information-theoretic model for such a setup is presented in [13]. This complexity-theoretic security notion is similar to modern cryptography and they try to define a secure stegosystem such that the stegotext is computationally indistinguishable from the covertext. In such a scenario a new term called steganographic secrecy of stegosystem is introduced which is defined as the inability of a polynomial-time adversary (Eve) to distinguish between observed distributions of unaltered covertext and stegotexts. To the best of our knowledge, steganographic techniques have not been used in the domain of training sets for machine learning models.

Steganography is often confused with cryptography [31, 61], however the goal of these two systems are completely different. The goal of cryptography is to ensure confidentiality of data in communication and storage processes. Hiding the existence of sensitive data is not the end goal here (unlike steganography). According to Kerckhoffs's principle [33, 34], this confidentiality must not rely on the obfuscation of the encoding scheme, but only on the secrecy of the decryption key.

One particular branch of cryptography we highlight is homomorphic encryption [54]. Consider a situation where you seek to delegate some computation to another computer (e.g., using a cloud computation service to a perform machine learning task). You would like to utilize their computation power, but you do not trust them with your private data. Homomorphic encryption allows a method by which you can encrypt your data prior to sending it. The untrusted computer will then perform its operations on the encrypted data, returning to you the result (e.g., a learned model). You then decrypt the result, yielding what the remote computer would have computed had you provided your original (unencrypted) data. A homomorphic cryptosystem which supports arbitrary computation on ciphertexts is known as fully homomorphic encryption (FHE). The first plausible construction of such a system was proposed in [57]. This scheme supports both addition and multiplication operations on ciphertexts, which in turn makes possible to construct circuits for arbitrary computations. Some second generation solutions were proposed in [7, 6, 46, 19].

In our setting, encryption (homomorphic or otherwise) is not enough to solve Alice's task. After Alice has transmitted her data to Bob, Bob learns a model. Alice's goal is not only for Eve to not know the model (which could easily be achieved by Alice simply sending an encrypted model), but also for Eve not to be suspicious. Eve believes that Alice is drawing data points i.i.d. from some distribution and thus data encrypted by standard methods will cause alarm. We do note that the relatively new method of "honey encryption" [30] may be a useful alternative approach for Alice, which we leave as future work.

The idea of constructing a dataset keeping a particular machine learning algorithm and a target model in mind is known as machine teaching. Machine teaching is the inverse of machine learning and has applications in various fields [65, 44]. In particular, machine teaching has applications in the domain of adversarial learning which studies the use of machine learning in security-sensitive domains. Numerous attacks against various machine learners have been explored, highlighting the security ramifications of using machine learning in practice [27, 4, 3, 16, 42, 59].

In the work presented herein, Alice can be thought of as "attacking" the learner Bob, in that she aims to provide a dataset which causes Bob to learn a model with parituclar properties. We highlight how this differs from the classical adversarial learning framework in two ways. First, Alice is not perturbing an existing training set, but rather generating one. Thus, this is more akin to the Machine Teaching framework. Second is the presence of Eve. Namely, Alice is trying not only to affect Bob's resulting model, but also to hide her involvement from a third party eavesdropper. In spirit, this is similar to the adversarial learning work performed on intrusion detection systems [35, 37]. In terms of the details of the mathematics, our framework and strategies for solving Alice's optimization problem more closely follow [49].

Within adversarial machine learning, a line of research has posed the problem of learning in the presence of adversaries in game theoretic contexts [45, 8, 16, 9, 10, 21, 10]. [16, 43, 1] specifically address a learner's defense strategy in various contexts. Randomization has also been explored as a method of defense [11, 62], as well as in the context of machine teaching [2]. Our work contributes to this conversation as Eve can be seen as a form of defense for Bob.

6 Conclusion and Discussions

We introduced the training set camouflage setting where a carefully constructed training set can be sent over an open channel with the intention of training a machine learner on a secret classification task. Using this framework, an agent can hide the intention of the secret task from a third party observer. Our experimental results show that training set camouflage is indeed a plausible threat. We present three approaches to solve the optimization problem. We observe that all three solvers perform well but both NLP and beam search outperform uniform sampling in all cases. The NLP solver often performs a bit better than beam search. This suggests that for the logistic regression learner NLP is Alice's

preferred solver of choice. However, the NLP solver cannot be applied to all possible learners (non-convexity prevents the application of KKT conditions). Thus in such cases beam search becomes the preferred solver.

We note that **MMD** is stronger with larger sample sizes. It will be harder for Alice to fool Eve given a large camouflage pool C and also if she is forced to select a large camouflaged training set D. **MMD** is also stronger with smaller feature dimensions [20]. Also, it is harder for Alice to fool Eve if she increases the value of α. Since α is the upper bound of the probability of the Type I error for the null hypothesis i.e., the camouflage pool and camouflaged training set come from the same distribution, increasing α allows Eve to become more suspicious. As future work we plan to devise defensive strategies against Alice. In such scenarios it is advisable to assume that Eve's detection function is known to the attacker (Kerckhoffs's principle [33, 34]) which we make here.

We note that camouflage seems easier for Alice to do if the cover task is in fact somewhat confusable, presumably because she can generate different decision boundaries by picking from overlapping camouflage items. This can be imagined easily in the 2D case with two overlapping point clouds forming the cover task. In such a scenario any separable secret task (no overlap between the secret task instances) can be taught to Bob by Alice. One interesting open question is whether there is a *universal* cover task for all secret tasks. We also note that achieving Alice's goal becomes much harder in the multi-class setting as finding a cover task becomes more challenging.

As mentioned previously, Bob fixed his learning hyperparameters (e.g., regularization parameter of the logistic regression). This was done for speed. However, nothing prevents Bob from using cross validation [38]. Cross validation is popular technique used in machine learning where the learner is trained multiple times on different subsets of the whole training set to tune the hyperparameters of the learner. Alice would simply emulate the same cross validation while optimizing the camouflaged training set. This can be easily done in beam search and uniform sampling, at the cost of more computation. Unfortunately significant modifications will be required for NLP.

Also, the loss function ℓ used by Alice and Bob is the same, as seen in the upper and lower optimization problems in (2). It is straightforward to allow different losses. For example, Bob may learn with the logistic loss since it is a standard learner, while Alice uses 0-1 loss to directly optimize Bob's accuracy.

We note that training set camouflage can be extended to cross modality correspondence, e.g., use an image camouflage pool while the secret classification task is to classify text articles. Alice and Bob can communicate via the private channel to establish the correspondance between images features and text features. Another possible way to extend the camouflage pool is to allow perturbed instances as well.

Acknowledgment This work is supported in part by NSF 1545481, 1704117, 1623605, 1561512, and the MADLab AF Center of Excellence FA9550-18-1-0166.

References

1. Alfeld, S., Zhu, X., Barford, P.: Explicit defense actions against test-set attacks. In: AAAI. pp. 1274–1280 (2017)
2. Balbach, F.J., Zeugmann, T.: Teaching Randomized Learners. In: International Conference on Computational Learning Theory (2006)
3. Barreno, M., Nelson, B., Joseph, A.D., Tygar, J.: The security of machine learning. Machine Learning **81**(2), 121–148 (2010)
4. Barreno, M., Nelson, B., Sears, R., Joseph, A.D., Tygar, J.D.: Can Machine Learning Be Secure? In: Proceedings of the 2006 ACM Symposium on Information, computer and communications security (2006)
5. Biggio, B., Roli, F.: Wild patterns: Ten years after the rise of adversarial machine learning. arXiv preprint arXiv:1712.03141 (2017)
6. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical gapsvp. In: Advances in cryptology–crypto 2012, pp. 868–886. Springer (2012)
7. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. ACM Transactions on Computation Theory (TOCT) **6**(3), 13 (2014)
8. Brückner, M., Kanzow, C., Scheffer, T.: Static Prediction Games for Adversarial Learning Problems. The Journal of Machine Learning Research (2012)
9. Brückner, M., Scheffer, T.: Nash Equilibria of Static Prediction Games. In: Advances in neural information processing systems (2009)
10. Brückner, M., Scheffer, T.: Stackelberg Games for Adversarial Prediction Problems. In: ACM SIGKDD (2011)
11. Bulò, S.R., Biggio, B., Pillai, I., Pelillo, M., Roli, F.: Randomized Prediction Games for Adversarial Machine Learning. IEEE Transactions on Neural Networks and Learning Systems (2016)
12. Bussieck, M.R., Pruessner, A.: Mixed-integer nonlinear programming. SIAG/OPT Newsletter: Views & News **14**(1), 19–22 (2003)
13. Cachin, C.: An information-theoretic model for steganography. In: Information Hiding. pp. 306–318. Springer (1998)
14. Chandramouli, R.: A mathematical approach to steganalysis. In: Proc. SPIE. vol. 4675, pp. 14–25 (2002)
15. Cox, I.J., Kalker, T., Pakura, G., Scheel, M.: Information transmission and steganography. In: IWDW. pp. 15–29. Springer (2005)
16. Dalvi, N., Domingos, P., Sanghai, S., Verma, D., et al.: Adversarial Classification. In: ACM SIGKDD (2004)
17. Dziugaite, G.K., Roy, D.M., Ghahramani, Z.: Training generative neural networks via maximum mean discrepancy optimization. arXiv preprint arXiv:1505.03906 (2015)
18. Fridrich, J.: Feature-based steganalysis for jpeg images and its implications for future design of steganographic schemes. In: International Workshop on Information Hiding. pp. 67–81. Springer (2004)
19. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Advances in Cryptology–CRYPTO 2013, pp. 75–92. Springer (2013)
20. Gretton, A., Borgwardt, K.M., Rasch, M.J., Schölkopf, B., Smola, A.: A kernel two-sample test. Journal of Machine Learning Research **13**(Mar), 723–773 (2012)
21. Hardt, M., Megiddo, N., Papadimitriou, C., Wootters, M.: Strategic Classification. In: ACM ITCS (2016)

22. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE CVPR. pp. 770–778 (2016)
23. Hearst, M.A., Dumais, S.T., Osuna, E., Platt, J., Scholkopf, B.: Support vector machines. IEEE Intelligent Systems and their applications **13**(4), 18–28 (1998)
24. Hoerl, A.E., Kennard, R.W.: Ridge regression: Biased estimation for nonorthogonal problems. Technometrics **12**(1), 55–67 (1970)
25. Hopper, N.J., Langford, J., Von Ahn, L.: Provably secure steganography. In: Annual International Cryptology Conference. pp. 77–92. Springer (2002)
26. Hosmer Jr, D.W., Lemeshow, S., Sturdivant, R.X.: Applied logistic regression, vol. 398. John Wiley & Sons (2013)
27. Huang, L., Joseph, A.D., Nelson, B., Rubinstein, B.I., Tygar, J.: Adversarial Machine Learning. In: AISEC (2011)
28. Joachims, T.: A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. Tech. rep., Carnegie-mellon univ pittsburgh pa dept of computer science (1996)
29. Johnson, N.F., Jajodia, S.: Exploring steganography: Seeing the unseen. Computer **31**(2) (1998)
30. Juels, A., Ristenpart, T.: Honey encryption: Security beyond the brute-force bound. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 293–310. Springer (2014)
31. Katz, J., Menezes, A.J., Van Oorschot, P.C., Vanstone, S.A.: Handbook of applied cryptography. CRC press (1996)
32. Ker, A.D.: Steganalysis of lsb matching in grayscale images. IEEE signal processing letters **12**(6), 441–444 (2005)
33. Kerckhoffs, A.: la cryptographie militaire (part i). vol. 9, pp. 5–38 (1883)
34. Kerckhoffs, A.: la cryptographie militaire (part ii). vol. 9, pp. 161–191 (1883)
35. Kloft, M., Laskov, P.: A poisoning attack against online anomaly detection. In: NIPS Workshop on Machine Learning in Adversarial Environments for Computer Security. Citeseer (2007)
36. Kloft, M., Laskov, P.: Online anomaly detection under adversarial impact. In: AISTATS. pp. 405–412 (2010)
37. Kloft, M., Laskov, P.: Online anomaly detection under adversarial impact (2011)
38. Kohavi, R., et al.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: IJCAI. vol. 14 (2), pp. 1137–1145. Montreal, Canada (1995)
39. Krasin, I., Duerig, T., Alldrin, N., Ferrari, V., Abu-El-Haija, S., Kuznetsova, A., Rom, H., Uijlings, J., Popov, S., Veit, A., Belongie, S., Gomes, V., Gupta, A., Sun, C., Chechik, G., Cai, D., Feng, Z., Narayanan, D., Murphy, K.: Openimages: A public dataset for large-scale multi-label and multi-class image classification. Dataset available from https://github.com/openimages (2017)
40. Krenn, R.: Steganography and steganalysis (2004)
41. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images (2009)
42. Laskov, P., Kloft, M.: A Framework for Quantitative Security Analysis of Machine Learning. In: Proceedings of the 2nd ACM workshop on Security and artificial intelligence (2009)
43. Letchford, J., Vorobeychik, Y.: Optimal Interdiction of Attack Plans. In: AAMAS (2013)
44. Liu, J., Zhu, X.: The teaching dimension of linear learners. Journal of Machine Learning Research **17**(162), 1–25 (2016)

45. Liu, W., Chawla, S.: A Game Theoretical Model for Adversarial Learning. In: Data Mining Workshops, 2009. ICDMW'09. IEEE International Conference on (2009)
46. López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: Proceedings of the forty-fourth annual ACM symposium on Theory of computing. pp. 1219–1234. ACM (2012)
47. Lowd, D., Meek, C.: Adversarial learning. In: ACM SIGKDD. pp. 641–647. ACM (2005)
48. Maganbhai, P.A.K., Chouhan, K.: A study and literature review on image steganography. International Journal of Computer Science and Information Technologies **6** (2015)
49. Mei, S., Zhu, X.: Using machine teaching to identify optimal training-set attacks on machine learners. In: Twenty-Ninth AAAI Conference on Artificial Intelligence (2015)
50. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
51. Queirolo, F.: Steganography in images. Final Communications Report. **3** (2011)
52. Reyzin, L., Russell, S.: More efficient provably secure steganography. Department of computer science Boston University (2003)
53. Rich, E., Knight, K.: Artificial intelligence. McGraw-Hill, New (1991)
54. Rivest, R.L., Adleman, L., Dertouzos, M.L.: On data banks and privacy homomorphisms. Foundations of secure computation **4**(11), 169–180 (1978)
55. Simmons, G.J.: The prisoners problem and the subliminal channel. In: Advances in Cryptology. pp. 51–67. Springer (1984)
56. Singh, K.U.: A survey on image steganography techniques. International Journal of Computer Applications **97**(18) (2014)
57. Smart, N.P., Vercauteren, F.: Fully homomorphic encryption with relatively small key and ciphertext sizes. In: International Workshop on Public Key Cryptography. pp. 420–443. Springer (2010)
58. Steinwart, I.: On the influence of the kernel on the consistency of support vector machines. Journal of machine learning research **2**(Nov), 67–93 (2001)
59. Tan, K.M., Killourhy, K.S., Maxion, R.A.: Undermining an Anomaly-Based Intrusion Detection System Using Common Exploits. In: Recent Advances in Intrusion Detection (2002)
60. Thompson, A.: All the news. https://www.kaggle.com/snapcrack/all-the-news (2017)
61. Van Tilborg, H.C., Jajodia, S.: Encyclopedia of cryptography and security. Springer Science & Business Media (2014)
62. Vorobeychik, Y., Li, B.: Optimal Randomized Classification in Adversarial Settings. In: AAMAS (2014)
63. Wu, H.C.: The karush–kuhn–tucker optimality conditions in an optimization problem with interval-valued objective function. European Journal of Operational Research **176**(1), 46–59 (2007)
64. Zhang, L., Wu, J., Zhou, N.: Image encryption with discrete fractional cosine transform and chaos. In: Information Assurance and Security, 2009. IAS'09. Fifth International Conference on. vol. 2, pp. 61–64. IEEE (2009)
65. Zhang, X., Zhu, X., Wright, S.: Training set debugging using trusted items. In: AAAI (2018)

7 Appendix A: MMD as Eve's Detection Function

One critical component of our camouflage framework is Eve's detection function Ψ — how she determines if a training set is suspicious or not. Eve's detection function is a two-sample test as its goal is to discern if the two sets $\mathcal{C}, D$ are drawn from the same distribution or not. In what follows we discuss using Maximum Mean Discrepancy (**MMD**) [20] as Eve's detection function, as we do in our experiments. **MMD** is a widely used two-sample test [17], but, of course other detection functions can be used in (1). We first review basic **MMD** following [20]. Let p and p' be two Borel probability measures defined on a topological space $\mathcal{Z}$. Given a class of functions $\mathcal{F}$ such that $f : \mathcal{Z} \mapsto \mathbb{R}, f \in \mathcal{F}$, **MMD** is defined as $\mathbf{MMD}(p, p') = \sup_{f \in \mathcal{F}}(E_{\mathbf{z}}[f(\mathbf{z})] - E_{\mathbf{z}'}[f(\mathbf{z}')])$. Any unit ball in a reproducing kernel Hilbert space (RKHS) can be used as the function class $\mathcal{F}$ if the kernel is universal (e.g., Gaussian and Laplace kernels [58]). Using this function space, **MMD** is a metric. This means $\mathbf{MMD}(p, p') = 0 \Leftrightarrow p = p'$. Computing **MMD** requires the expectations to be known, which generally, is not the case in practice. We obtain an empirical estimation by replacing the population expectations with empirical mean computed on i.i.d. samples $Z = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$ and $Z' = \{\mathbf{z}'_1, \ldots, \mathbf{z}'_m\}$ from p and p', respectively. We define

$$\mathbf{MMD}(Z, Z') = \left[\frac{1}{n^2}\sum_{i,j=1}^{n} k(\mathbf{z}_i, \mathbf{z}_j) - \frac{2}{nm}\sum_{i,j=1}^{n,m} k(\mathbf{z}_i, \mathbf{z}'_j) + \frac{1}{m^2}\sum_{i,j=1}^{m} k(\mathbf{z}'_i, \mathbf{z}'_j)\right]^{\frac{1}{2}}$$

where k is the kernel of the RKHS. Let $d = |\mathbf{MMD}(Z, Z') - \mathbf{MMD}(p, p')|$. Gretton *et. al.* show that $P\left(d > 2\left(\sqrt{\frac{K}{n}} + \sqrt{\frac{K}{m}}\right) + \epsilon\right) \leq 2e^{-\frac{\epsilon^2 nm}{2K(n+m)}}$, where K is an upperbound on the kernel values. We convert the above bound into a one-sided hypothesis testing procedure. Under the null hypothesis $p = p'$ we have $\mathbf{MMD}(p, p') = 0$. We consider positive deviations of $\mathbf{MMD}(Z, Z')$ from $\mathbf{MMD}(p, p')$. Equating the RHS with α (probability of incorrectly stating $p \neq p'$ also known as the type I error) gives a hypothesis test of level-α, where solving ϵ as a function of α gives $\alpha = e^{-\frac{\epsilon^2 nm}{2K(n+m)}} \Rightarrow \epsilon = \sqrt{\frac{2K(n+m)}{nm}\log\frac{1}{\alpha}}$. We retain the null hypothesis if $\mathbf{MMD}(Z, Z') - T < 0$, where the threshold is $T = 2\left(\sqrt{\frac{K}{n}} + \sqrt{\frac{K}{m}}\right) + \sqrt{\frac{2K(n+m)}{nm}\log\frac{1}{\alpha}}$. This also defines Eve's detection function ($\Psi(\mathcal{C}, D)$) at level-α: $\Psi(\mathcal{C}, D) \equiv \mathbf{MMD}(\mathcal{C}, D) - T$. If $\Psi(\mathcal{C}, D) \geq 0$ then Eve realizes that D is not drawn i.i.d. from $\mathbb{Q}_{(\mathbf{x},y)}$ and flags it as suspicious.

For all our experiments Eve used the RBF kernel $k(\mathbf{z}_i, \mathbf{z}_j) = \exp\left(-\frac{\|\mathbf{z}_i - \mathbf{z}_j\|^2}{2\sigma^2}\right)$. Eve set σ to be the median distance between points in the camouflage pool as proposed in [20]. Eve also included the scaled class label as a feature dimension: $[\mathbf{x}_i, c\mathbb{1}\{y_i = 1\}]$ where $c = \max_{k,l \text{ such that } y_k = y_l} \|\mathbf{x}_k - \mathbf{x}_l\|$ and $\mathbb{1}\{\cdot\}$ is the indicator function. This augmented feature enables Eve to monitor both features and labels. When using the NLP solver Alice only has to consider instances from camouflage pool. She calculated **MMD** in the following manner:

$$\mathbf{MMD}_b(Z, b_1, \ldots, b_{|Z|}) = \left[\frac{1}{n^2}\sum_{i,j=1}^{n} k(\mathbf{z}_i, \mathbf{z}_j) - \frac{2}{n\sum_{i=1}^{n} b_i}\sum_{i,j=1}^{n} b_i k(\mathbf{z}_i, \mathbf{z}_j) + \frac{1}{(\sum_{i=1}^{n} b_i)^2}\sum_{i,j=1}^{n} b_i b_j k(\mathbf{z}_i, \mathbf{z}_j)\right]^{\frac{1}{2}} \tag{6}$$