Understanding the Limits of Unsupervised Domain Adaptation via Data Poisoning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Unsupervised domain adaptation (UDA) enables cross-domain learning without target domain labels by transferring knowledge from a labeled source domain whose distribution differs from the target. However, UDA is not always successful and several accounts of 'negative transfer' have been reported in the literature. In this work, we prove a simple lower bound on the target domain error that complements the existing upper bound. The bound shows the insufficiency of minimizing source domain error and marginal distribution mismatch for a guaranteed reduction in the target domain error, due to the possible increase of induced labeling function mismatch. This insufficiency is further illustrated through simple distributions for which the same UDA approach succeeds, fails, and may succeed or fail with an equal chance. Motivated from this, we propose novel data poisoning attacks to fool UDA methods into learning representations that produce large target domain error. We evaluate the effect of these attacks on popular UDA methods using benchmark datasets where they have been previously shown to be successful. Our results show that poisoning can significantly decrease the target domain accuracy, dropping it to almost 0% in some cases, with the addition of only 10% poisoned data in the source domain. The failure of UDA methods demonstrates the limitations of UDA at guaranteeing cross-domain generalization consistent with the lower bound. Thus, evaluation of UDA methods in adversarial settings such as data poisoning can provide a better sense of their robustness in scenarios unfavorable for UDA.

1 Introduction

The problem of domain adaptation (DA) arises when the training and the test data distributions are different which violates the common assumption of learning. In this paper, we focus on unlabeled DA (UDA), which is a special case of DA when no labeled information from the target domain is available. This setting is useful for applications where obtaining large-scale well-curated datasets is both time consuming and costly. Inspired from the seminal works [1, 2] which proved an upper bound on the target domain error in UDA setting, several algorithms have been proposed. Many of these rely on learning a domain invariant representation by minimizing the error on the source domain and different divergence measures of the marginal feature distributions of the source and the target domain, such as total variation distance, Jensen-Shannon divergence [9, 29, 33], Wasserstein distance [27, 13, 7], and maximum mean discrepancy [17, 15, 18]. The success of these algorithms is argued in terms of minimization of the upper bound proposed in [1] and significant improvement in the target domain accuracy on benchmark UDA tasks was demonstrated.

Despite this success on benchmark datasets, some works [9, 14, 32, 30] have presented evidences of failure of these methods under different conditions. Recent works have tried to explain this apparent failure of UDA methods by proposing new upper bounds [32, 6] on the target domain error while

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

others have demonstrated the cause of failure with experiments where learning a domain invariant representation that minimizes the source domain error causes an increase in the error of the ideal joint hypothesis [14] leading to an increase in the upper bound proposed by [1]. However, a large upper bound does not guarantee failure of UDA methods. Thus in this work, we propose a lower bound on the target domain error that provides a necessary condition for the success of learning under UDA setting and therefore complements the known upper bound. Our lower bound is dependent on the difference between the labeling functions for source and target domain data induced by the representation map. For cases when the induced labeling functions match on source and target domain data (i.e., favorable case), the success of UDA is explained using the upper bounds proposed by previous works [1, 32]. For a representation that aligns source and target domain data and minimizes error on the source but induces labeling functions that don't agree on the source and target domain data (i.e., unfavorable case), our lower bound explains the failure of UDA. There is still another case (i.e., ambiguous case) of data distributions where success and failure of UDA are equally likely. This is due to the lack of labeled information from the target domain. In this case, a small amount of misinformation about the target domain labels can turn the distribution into the unfavorable case causing a significant increase in the target domain error.

Motivated from the analysis and the lower bound, we evaluate the effect of addition of small amount of adversarially crafted data on current UDA methods. For this purpose, We propose novel data poisoning attacks, using mislabeled and clean-label points, to study how few poisoned points can fool UDA methods into learning a representation that incurs high target domain error. We evaluate the effect of our attacks on popular UDA methods using benchmark datasets where they were previously shown to be very effective. Our poisoning attacks cause UDA methods to either align incorrect classes from the two domains or prevent correct classes from being very close in the representation space. Both of these lead to the failure of UDA methods at reducing target domain error. Our results show a significant drop in the target domain accuracy for current UDA methods, dropping to almost 0% in some cases, with just 10% poison data in the source domain. This dramatic failure of UDA methods demonstrates the limits of current UDA methods and suggests that the future UDA methods may be evaluated non only on clean benchmark datasets but also in adversarial setting such as poisoning to demonstrate their success in unfavorable scenarios.

Our contributions are as follows:

- We prove a simple lower bound on the target domain error providing a necessary condition for the success of learning in UDA setting. The bound shows the failure of learning a domain invariant representation that minimizes source domain error at guaranteeing target generalization.
- We present example distributions where UDA succeeds, fails, and where success and failure are equally likely. The last case opens door to adversarial attacks such as data poisoning.
- We propose several novel data poisoning attacks that use clean-label and mislabeled points to demonstrate the dramatic failure of UDA methods at target generalization on benchmark datasets. Our poisoning attacks can be used to obtain a better sense of robustness of the future UDA methods beyond simple benchmark datasets.

2 Background and related work

Notations and settings: Let $\mathcal{X}$ be the data domain and $\mathcal{D}$ the distribution over $\mathcal{X}$ with the corresponding pdf $p(x)$. We assume there is a deterministic labeling function $f : \mathcal{X} \to [0, 1]$ for the given binary classification task. The $f(x)$ can be interpreted as $Pr[y = 1|x]$. Let $g : \mathcal{X} \to \mathcal{Z}$ denote the representation map that maps an input instance x to its features where $\mathcal{Z}$ is called feature or representation space. Let $h : \mathcal{Z} \to [0, 1]$ be a hypothesis for binary classification on the representation space. Note that the representation map g induces a distribution over $\mathcal{Z}$ denoted by $Pr_{\tilde{\mathcal{D}}}[B] = Pr_{\mathcal{D}}[g^{-1}(B)]$ and the corresponding density function $\tilde{p}(z)$ on $\mathcal{Z}$ [2]. The g also induces the labeling function

$$\tilde{f}(z) = E_D[f(x)|g(x) = z] \tag{1}$$

for any B such that $g^{-1}(B)$ is $\mathcal{D}$-measurable. The misclassification error $e(h)$ w.r.t. the induced labeling function is $e(h) = E_{\tilde{D}}[|\tilde{f}(z) - h(z)|]$. Similarly, we define $e(\tilde{f}, \tilde{f}') = E_{\tilde{D}}[|\tilde{f}(z) - \tilde{f}'(z)|]$ and $e(h, h') = E_{\tilde{D}}[|h(z) - h'(z)|]$. The distributions $\tilde{p}$ and the labeling functions $\tilde{f}$ for the source

and the target domains will be written as $\tilde{p}_S$, $\tilde{p}_T$, $\tilde{f}_S$ and $\tilde{f}_T$, respectively. The total variation distance is $D_1(\tilde{p}, \tilde{p}') = \int_{\mathcal{Z}} |\tilde{p}(z) - \tilde{p}'(z)| dz$.

Analysis of Domain Adaptation: Several previous works have studied the problem of DA and have provided conditions under which DA is possible [2, 1, 19, 20, 8, 3]. [1, 2] proposed an upper bound on the target domain error which is the inspiration of many UDA algorithms. Recent works [32, 6] have improved the upper bounds on target error and have proposed a lower bound dependent on the labeling functions in the input space. Using this lower bound failure for UDA was demonstrated in the setting when marginal label distributions are different between the two domains. Our lower bound on the other hand, is dependent on the labeling functions induced by the representation, for the source and target domains. This suggests that if a representation induces labeling functions which disagree on source and target domains then DA provably fails even if the representation is domain invariant and minimizes error on the source domain. Thus our lower bound directly explains the observations of failure of UDA in many previous works [9, 14, 30, 32].

Algorithms for UDA: Most algorithms [9, 29, 16, 33] for UDA learn a representation that is invariant to the domain shift while minimizing error on source domain. The popular adversarial approach to domain adaptation is DANN [9] uses a discriminator to distinguish points from source and target domains based on their representations. Another popular method CDAN[16], uses classifier output along with representations to identify domains of a points. Recently IW-DAN and IW-CDAN [6] were proposed as extensions of the original DANN and CDAN with an importance weighting scheme to minimize the mismatch between the labeling distributions of the two domains. A different approach, MCD [25] uses two task specific classifiers as discriminators to align the two domains. This method adversarially trains the representation to minimize the disagreement between the two classifiers on the target domain data (classifier discrepancy) while training the classifiers to maximize this discrepancy. Another recent approach, SSL [31] uses self-supervised learning tasks (e.g. rotation angle prediction) to better align the two domains. In this work, study the effect of poisoning on these methods since they are quite effective at various UDA tasks.

Data poisoning: Data poisoning [4, 23, 11, 5, 12, 28] is a training time attack where the attacker has access to the training data which will be used by the victim for training. Attacker's intention is to modify the training data such that victim's model behaves as the attacker intended, after training. Several works [34, 24, 26, 21, 10] have shown the limitations of supervised learning methods in presence of poisoned data. Recently, effects of poisoning were also studied against training methods that produce certifiably robust classifiers [22]. However, all these methods consider poisoning in presence of a single domain. We are the first to study the effect of poisoning in the UDA setting with two domains and training using popular UDA algorithms.

3 When does unsupervised domain adaption fail?

3.1 Lower bound on the target-domain loss

Most adversarial DA methods learn a domain invariant representation $g : \mathcal{X} \to \mathcal{Z}$ by minimizing error on the source domain and penalizing the mismatch between the **marginal** source and target distributions, since the conditional distribution $\tilde{p}_T(z|y)$ for target domain is unavailable in the UDA setting. Some works [30, 32, 14] have shown this not to be sufficient at guaranteeing target generalization. Recent works have proposed a new upper bound [32] or argued failure in terms of the upper bound in [1, 2] ($e_T(h) \leq \min\{e_T(\tilde{f}_S, \tilde{f}_T)), e_S(\tilde{f}_S, \tilde{f}_T))\} + e_S(h) + D_1(\tilde{p}_S, \tilde{p}_T)$) being vacuous. However, large upper bound does not explain failure. Here, we derive a simple lower bound on the target domain error to explain the failure of UDA methods at guaranteeing target generalization

Theorem 1. *For all $h \in H$ and $g \in G$,*

$$e_T(h) \geq \max\{e_S(\tilde{f}_S, \tilde{f}_T), e_T(\tilde{f}_S, \tilde{f}_T)\} - e_S(h) - D_1(\tilde{p}_S, \tilde{p}_T) \tag{2}$$

The proof is in Appendix A. Even though our bound depends on the total variation distance which is an overly strict measure of distance and is difficult to estimate, the bound still explains limitations of UDA. Extending the bound to different divergence metrics is left as future work. The following corollary is immediate from the theorem:

Corollary 2. *For all $h \in H$, $g \in G$, $e_T(h) \geq \max\{e_S(\tilde{f}_S, \tilde{f}_T), e_T(\tilde{f}_S, \tilde{f}_T)\} - e_S(h) - \sqrt{0.5 D_{KL}(\tilde{p}_S || \tilde{p}_T)}$.*

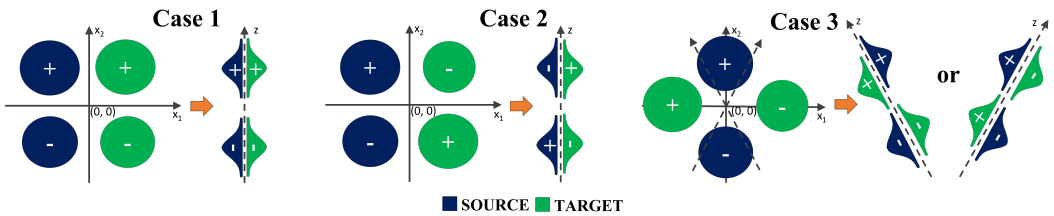


Figure 1: Illustrative cases for UDA. In UDA, one can observe the source data (blue blobs), the source labels (class + and -), and the target data (green blobs) but not the target labels. The optimal linear representation $g(x) = u^T x$ from the input space in $\mathbb{R}^2$ to the feature space in $\mathbb{R}$ (dashed line) that minimizes the source loss and the alignment loss in Eq. 3 can be computed accurately. Depending on the original data distribution, the optimal representation can be successful (Case 1), fail (Case 2), or undetermined (Case 3). Case 3 has two global minima and poisoning can make the bad solution (right) to be chosen.

This is due to the Pinsker's inequality: $D_1(p, p') \leq \sqrt{0.5 D_{KL}(p||p')}$. The interpretation of Eq. 2 is as follows. Since f_T is not observable, the goal of UDA is to minimize the observable source classification error $e_S(h)$ and the domain mismatch $D_1(\tilde{p}_S, \tilde{p}_T)$ (or other metrics such as $d_{H\Delta H}(\tilde{p}_s, \tilde{p}_T)$):

$$\min_{g,h} \; e_s(h) + D_1(\tilde{p}_S, \tilde{p}_T). \tag{3}$$

This leads to maximization of the 2nd and the 3rd term in the RHS of Eq. 2. With overparameterized models such as deep neural nets, it is often possible to get $e_S(h) \simeq 0$ and $D_1(\tilde{f}_S, \tilde{f}_T) \simeq 0$, and consequently, $e_T(h) \geq e(\tilde{f}_S, \tilde{f}_T)$. By making $\tilde{f}_S$ and $\tilde{f}_T$ disagree in the representation space, it is possible to make the target domain error $e_T(h)$ **provably** large.

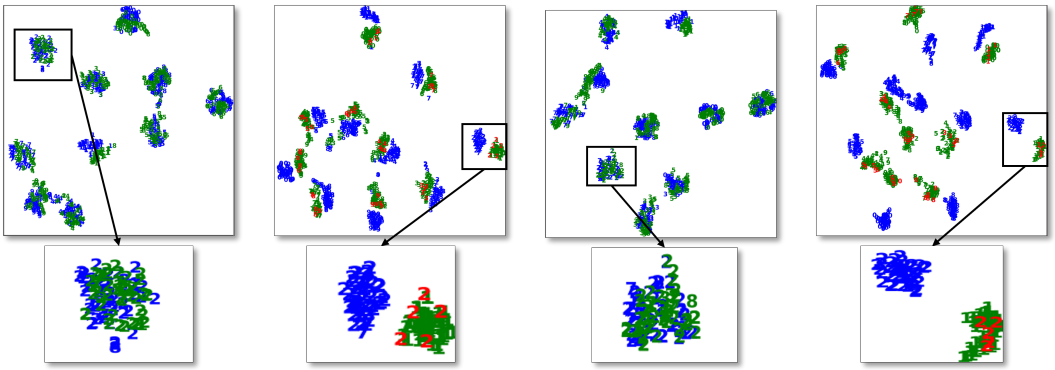
Corollary 3. *For all $h \in H$ and $g \in G$*

$$|e_T(h) - e_S(\tilde{f}_S, \tilde{f}_T)| \leq e_S(h) + D_1(\tilde{p}_S, \tilde{p}_T), \;\; \text{and} \;\; |e_T(h) - e_T(\tilde{f}_S, \tilde{f}_T)| \leq e_S(h) + D_1(\tilde{p}_S, \tilde{p}_T). \tag{4}$$

This is obtained by combining the upper and the lower bounds. Thus a UDA method can only guarantee the target error $e_T(h)$ to be close to the labeling function mismatch $e(\tilde{f}_S, \tilde{f}_T)$. Whether $e(\tilde{f}_S, \tilde{f}_T)$ becomes larger or smaller after solving Eq. 3 is data/model dependent. For concreteness, we analyze three cases of successful and failed DA using the following illustrative examples.

3.2 Illustrative examples of UDA failure

Assume the mixture-of-Gaussian distributions $p_S(x)$ and $p_T(x)$ for the source and the target domain in the input space, shown as blue and green blobs in Figure 1. We consider a linear representation map $g(x) = u^T x$ from the input space $\mathcal{X} \subset \mathbb{R}^2$ to the feature space $\mathcal{Z} \subset \mathbb{R}$ (dashed line) that minimizes the source classification loss plus the marginal mismatch loss in Eq. 3. The optimal solution g to the minimization problem can be found accurately using a mix of analytical and numerical optimization (detailed in Appendix B). We demonstrate three cases of data distributions. In Case 1 (favorable case), the true labeling function for source and target is $f_S((x_1, x_2)) = f_T((x_1, x_2)) = I[x_2 \geq 0]$, that is, f is 1 in the upper halfspace and 0 in the bottom halfspace. One can verify (Appendix B) that the optimal u is the vertical direction ($u = [0, 1]^T$) and the best hypothesis is $h(z) = I[z \geq 0]$, in which case $e_S(h) = 0$ and $D_1(\tilde{p}_S, \tilde{p}_T) = 0$. That is, a perfect source classification and a perfect alignment of the marginals are achieved. Furthermore, the true labeling function in $\mathcal{Z}$ is $\tilde{f}_S(z) = \tilde{f}_T(z) = I[z \geq 0]$ (from Eq. 1) and therefore $e(\tilde{f}_S, \tilde{f}_S) = 0$ as well as the target loss $e_T(h) = 0$. In other words, the representation that minimizes Eq. 3 simultaneously **minimizes** $e(\tilde{f}_S, \tilde{f}_S) = 0$, achieving the goal of reducing $e_T(h)$. However in Case 2 (unfavorable case), the true labeling function for target is upside-down $f_T((x_1, x_2)) = I[x_2 \leq 0]$. Since UDA does not use the target label nor the true labeling function, the optimal g is exactly the same as Case 1 ($u = [0, 1]^T$), in which case we still have $e_S(h) = 0$ and $D_1(\tilde{p}_S, \tilde{p}_T) = 0$, but the labeling function mismatch becomes $e(\tilde{f}_S, \tilde{f}_S) = 1$ as well as $e_T(h) = 1$ which is the worst case. In other words, the representation that minimizes Eq. 3 simultaneously **maximizes** $e(\tilde{f}_S, \tilde{f}_S) = 0$, totally failing at the goal of reducing $e_T(h)$. Case 3 exemplifies an ambiguous case. The optimal projection minimizing e_S is still the vertical direction $u = [0, 1]^T$ but the optimal projection minimizing the alignment loss $D_1(\tilde{p}_S, \tilde{p}_T)$ can be either of



(a) DANN trained on clean data (b) DANN trained on poisoned data (c) CDAN trained on clean data (d) CDAN trained on poisoned data

SOURCE TARGET POISON

Figure 2: (Best viewed in color.) t-SNE embedding of the data in the representation space (for MNIST→ USPS task) learnt using DANN and CDAN on clean and poisoned source domain data. Without poisoning, correct classes (data from source class 2 is zoomed in) from two domains are aligned ((a) and (c)). Presence of poisoned data fools the methods into aligning incorrect classes from the two domains ((b) and (d)). The mismatch between the source and target classes is dependent on the labels of the poison data (due to which, target class 1 aligned to source class 2).

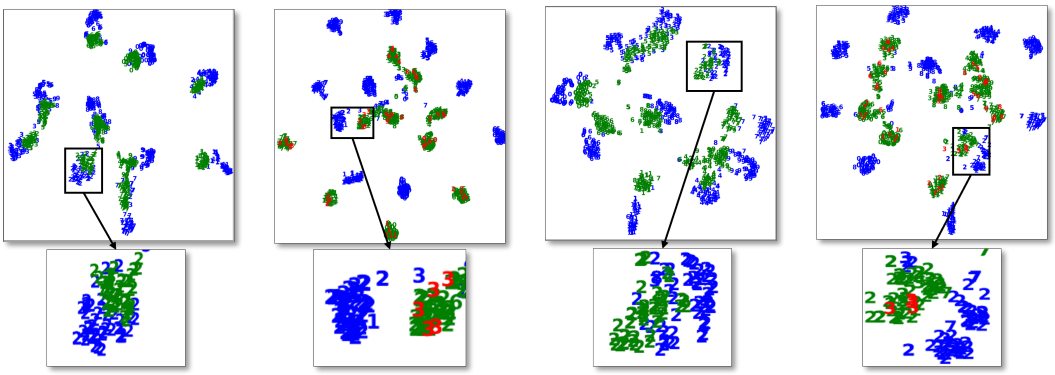
the $\pm 45°$ directions ($u = [\pm 1/\sqrt{2}, 1/\sqrt{2}]^T$) with no preference of one over the other. Therefore the optimal solution u for Eq. 3 that trades off the source error and the mistmatch loss has two equal-valued global solutions $[\pm u_1, u_2]^T$ for some u_1, u_2. One solution (Figure 1, Case 3, left) yields a small $e(\tilde{f}_S, \tilde{f}_S)$ and the other (Figure 1, Case 3, right) yields a large $e(\tilde{f}_S, \tilde{f}_S)$. As can be intuitively seen, UDA is successful in the former but fails in the latter, and which equal-valued solution will be chosen is undetermined. Thus a slight nudge via additional information (such as target labels) or misinformation (such as poisoning) can determine success or failure of UDA methods. Details of the analysis and the results are in Appendix B.

4 Breaking unsupervised domain adaptation methods with data poisoning

In this section, we present novel data poisoning attacks to evaluate the ease with which current UDA methods can be fooled into producing a representation that leads to a large error on the target domain. We propose three methods to generate poisoned data which will be added to the clean source domain data. The first method adds mislabeled data as poisons. Under this attack we evaluate two approaches (a) adding mislabeled source domain data (wrong-label correct-domain poisoning) and (b) adding mislabeled target domain data (wrong-label incorrect-domain poisoning) as poisons. The second method adds images from the source domain watermarked with images from the target domain with incorrect labels (watermarking attack) as poisons. The last method adds poisons with clean labels. We evaluate two approaches for this attack (a) using source domain data (clean-label correct-domain poisoning) and (b) using target domain data (clean-label wrong-domain poisoning) to initialize the poison data. The intuitive pictures of how these poisoning attacks hurt UDA methods are shown in Figs. 3, 5 and 6. We evaluate the effect of poisoning on popular UDA methods namely, DANN[9], CDAN[16], MCD[25], SSL[31] (with rotation-angle prediction task), IW-DAN[6], and IW-CDAN[6]. We compare the difference in the target accuracy attainable by these methods when using clean versus poisoned data. Two benchmark datasets are used in our experiments, namely Digits and Office-31. We evaluate four tasks using SVHN, MNIST, MNIST_M and USPS datasets under Digits and six tasks under the Office-31 using Amazon (A), DSLR (D) and Webcam (W) datasets. For all experiments, we train UDA algorithms on clean and poisoned source domain data and use neural networks whose architectures are similar to those used by previous works (see Appendix E).

4.1 Poisoning using mislabeled source and target domain data

In this experiment, mislabeled data is added to the clean source domain as poison. The labels provided by the attacker to the poison data are chosen to fool UDA methods into learning a representation

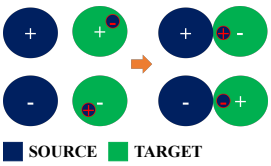


(a) MCD trained on clean data (b) MCD trained on poisoned data (c) SSL trained on clean data (d) SSL trained on poisoned data

SOURCE TARGET POISON

Figure 4: (Best viewed in color.) t-SNE embedding of the data in the representation space (for MNIST→ USPS task) learnt using MCD and SSL on clean and poisoned source domain data. Without poisoning, correct classes (data from source class 2 is zoomed in) from two domains are aligned ((a) and (c)). Presence of poisoned data prevents the methods from aligning correct classes from the two domains ((b) and (d)).

that incurs large error on target domain. We use two simple and effective labeling functions for this. For the Digits, we use a labeling function that systematically labels the poison data to the class next to their true class (e.g. poison points with true class one are labeled as two, points with true class two are labeled as three and so on). For Office-31 dataset, we use a labeling function that assigns the poison data the label of the closest (in the representation space learnt using the clean source domain data) incorrect source domain class. In this experiment, attacker is limited to adding only 10% poisoned data with respect to the size of the available target domain data (Experiments with different poison percentages are shown in the Appendix C). In wrong-label correct-domain poisoning, mislabeled source domain data are used as poisons. The results in rows marked with $\text{Poison}_{\text{source}}$ in Tables 1 and 2, show that UDA methods suffer only minor decrease in target domain accuracy with this approach. This happens because the presence of large amount of correctly labeled source domain data prevents the small amount poisoned data to affect the performance of UDA methods. A similar effect is observed with label flipping attacks in single domain poisoning [21, 24]. However, if the relative size of poisoned data was larger or comparable to clean source domain data, poisoning will be effective. This is observed in Table 2 when Amazon is the target data. Amazon dataset is roughly 5 times bigger that both DSLR and Webcam datasets. Due to this difference the permissible amount of poisoned data (10% of the target domain) makes the size of clean and poisoned data comparable, leading to successful poisoning. Thus, this poisoning approach causes UDA methods to fail in presence of large amount of poisoned data.



SOURCE TARGET

Figure 3: Poisoning with mislabeled target domain data causes discriminator based UDA approaches to align wrong classes (+ to -) from the two domains, leading to significant decline in the target domain accuracy.

In the wrong-label wrong-domain approach mislabeled target domain data is used for poisoning. The effect of poisoning on discriminator based methods [9, 16] is shown in Fig 3. The domain discriminator used in these methods is maximally confused when marginal distributions of the source and target domains are aligned. However, alignment of the marginal distributions does not ensure alignment of the conditional distributions[32, 16]. The objective of achieving low source domain error pushes the generator to correctly classify the poison data. Thus placing the poison and source data with labels close in the representation space. Since the poison data is mislabeled target domain data, poisoning makes UDA methods align wrong source and target domain classes. This leads to a significant decline in the target domain accuracy. This is also evident from the t-SNE embedding for DANN and CDAN methods in Fig. 2. In absence of poisoning, correct source and target classes are aligned (Fig. 2 (a) and (c)), where as in presence of poisoning wrong classes from the two domains are closer (Fig. 2 (b) and (d)). For MCD[25], which uses use classifier

Table 1: Decrease in the target domain accuracy for UDA methods trained on poisoned source domain data (with poisons sampled from source/target domains) compared to accuracy attained with clean data on the Digits tasks (mean±s.d. of 5 trials).

Method	Data	SVHN → MNIST	MNIST → MNIST_M	MNIST → USPS	USPS → MNIST
Source only	Clean	72.42±1.44	39.05±2.30	87.13±1.75	78.6±1.45
DANN	Clean	78.05±1.15	76.22±2.38	92.17±0.73	92.73±0.71
	$\text{Poison}_{\text{source}}$	70.26±2.84	69.98±3.49	93.44±0.84	92.08±0.68
	$\text{Poison}_{\text{target}}$	**1.46±1.12**	**0.48±0.04**	**0.97±0.53**	**5.83±0.82**
CDAN	Clean	79.19±0.70	73.88±1.10	93.92±0.97	95.94±0.71
	$\text{Poison}_{\text{source}}$	73.67±4.19	73.36±1.31	92.06±0.59	92.85±0.31
	$\text{Poison}_{\text{target}}$	**12.27±5.02**	**0.59±0.12**	**1.92±0.42**	**2.96±0.71**
MCD	Clean	96.18±1.53	93.95±0.33	89.96±2.04	88.34±2.50
	$\text{Poison}_{\text{source}}$	85.86±5.66	93.33±0.71	87.99±1.05	83.19±2.98
	$\text{Poison}_{\text{target}}$	**0.97±0.94**	**0.37±0.06**	**0.66±0.16**	**2.07±0.69**
SSL	Clean	66.85±2.30	92.76±0.91	88.69±1.28	82.23±1.59
	$\text{Poison}_{\text{source}}$	61.97±1.62	91.35±1.13	85.74±2.92	82.56±0.84
	$\text{Poison}_{\text{target}}$	**0.31±0.03**	**0.36±0.02**	**7.76±1.52**	**9.88±1.07**

Table 2: Decrease in the target domain accuracy for UDA methods trained on poisoned source domain data (with poisons sampled from source/target domains) compared to accuracy attained with clean data on the Office tasks (mean±s.d. of 3 trials)

Method	Dataset	A → D	A → W	D → A	D → W	W → A	W → D
Source Only	Clean	79.61	73.18	59.33	96.31	58.75	99.68
DANN	Clean	84.06	85.41	64.67	96.08	66.77	99.44
	$\text{Poison}_{\text{source}}$	79.11±0.35	83.98±1.19	44.31±2.94	95.22±0.22	43.35±1.65	96.58±0.87
	$\text{Poison}_{\text{target}}$	**59.83±0.20**	**63.18±1.96**	**17.58±0.39**	**76.43±0.62**	**19.82±0.33**	**84.20±0.71**
CDAN	Clean	89.56	93.01	71.25	99.24	70.32	100
	$\text{Poison}_{\text{source}}$	90.16±0.61	90.94±0.13	53.68±0.37	98.45±0.07	57.27±0.57	99.66±0.23
	$\text{Poison}_{\text{target}}$	**71.88±0.20**	**71.94±0.76**	**11.19±1.47**	**86.37±0.36**	**18.54±0.45**	**89.08±1.23**
IW-DAN	Clean	84.3	86.42	68.38	97.13	67.16	100
	$\text{Poison}_{\text{source}}$	81.25±0.91	83.27±0.45	50.76±1.58	96.68±0.29	48.31±2.02	99.73±0.12
	$\text{Poison}_{\text{target}}$	**61.64±0.53**	**63.43±1.14**	**15.69±1.76**	**80.29±0.07**	**26.54±0.48**	**88.62±0.23**
IW-CDAN	Clean	88.91	93.23	71.9	99.3	70.43	100
	$\text{Poison}_{\text{source}}$	89.83±0.31	90.77±1.27	57.51±0.06	98.41±0.07	61.16±1.21	99.66±0.12
	$\text{Poison}_{\text{target}}$	**72.62±0.42**	**70.15±2.21**	**14.36±0.66**	**88.26±0.15**	**22.36±0.96**	**87.75±0.53**

discrepancy to detect and align source and target domains, our poisoned data prevents the method from detecting target examples. This happens because the term that minimizes the error on poisoned source domain reduces the discrepancy of the classifiers on poison data, which are the target domain data. Thus, both the generator and discriminator (in the form of two classifiers) become optimal and there is no signal for the generator to align the two domains. The t-SNE embedding in Fig. 4 shows this effect. In presence of poisoned data Fig. 4 (b) we see twenty distinct clusters rather than just ten (we have ten classes in the Digits) as seen in the absence of poisoning Fig. 4 (a). In SSL[31], the generator must work well on the main task, i.e., should correctly classify all data in the poisoned source domain data. The auxiliary task ensures that representations of the source and target domains become similar as seen on clean data (Fig. 4 (c)). But in presence of poisoned data, similar representations of correct source and target domain classes leads to a drop in the accuracy of the main task on the poisoned data. This creates a conflict between the main and auxiliary tasks due to which correct source and target domain classes cannot be aligned Fig. 4 (d). The objective of making the main task accurate on poisoned data, leads to target domain data being assigned the labels of the poisoned points. There by leading to a significant drop in the target-domain accuracy. The results of wrong-label wrong-domain poisoning are in rows marked with $\text{Poison}_{\text{target}}$ in Tables 1 and 2 which show a significant reduction in the target domain accuracy compared to the accuracy obtained on clean data consistent. On Digits poisoning makes the target domain accuracy close to 0% on most tasks. On Office-31 poisoning causes at least 20% reduction in the target domain accuracy in most cases. The reason for tasks in Office-31 are less hurt by poisoning is because of the use of a pre-trained representation. Similar to previous works for Office-31, we use all labeled source and unlabeled target domain data for training and fine tune from a representation pre-trained on ImageNet dataset. The slowly changing representation pre-trained on a massive dataset weakens the effect of poisoning but cannot eliminate it. For the two tasks D → W and W → D, the fine tuned

Table 3: Decrease in target accuracy when training different domain adaptation methods on poisoned watermarked data in comparison to the target accuracy obtained with clean data on the Digits dataset.

Method	Dataset	SVHN → MNIST	MNIST → MNIST_M	MNIST → USPS	USPS → MNIST
DANN	Clean	78.05±1.15	76.22±2.38	92.17±0.73	92.73±0.71
	Poisoned$_\alpha$	68.76±3.91$_{0.05}$	27.36±15.77$_{0.05}$	91.84±0.55$_{0.10}$	88.93±4.36$_{0.10}$
		57.96±5.84$_{0.10}$	7.19±2.59$_{0.10}$	85.51±3.01$_{0.20}$	78.29±8.52$_{0.20}$
		33.33±4.38$_{0.15}$	4.73±0.38$_{0.15}$	39.29±1.34$_{0.30}$	41.52±7.43$_{0.30}$
CDAN	Clean	79.19±0.70	73.88±1.10	93.92±0.97	95.94±0.71
	Poisoned$_\alpha$	65.77±4.82$_{0.05}$	55.47±3.87$_{0.05}$	92.05±0.96$_{0.10}$	86.53±1.55$_{0.10}$
		57.57±3.11$_{0.10}$	7.37±1.26$_{0.10}$	86.54±2.43$_{0.20}$	77.39±4.84$_{0.20}$
		44.83±4.09$_{0.15}$	6.68±1.64$_{0.15}$	88.67±0.44$_{0.30}$	79.54±7.02$_{0.30}$
MCD	Clean	96.18±1.53	93.95±0.33	89.96±2.04	88.34±2.50
	Poisoned$_\alpha$	74.96±3.20$_{0.05}$	92.18±0.78$_{0.05}$	6.75±4.81$_{0.10}$	30.35±2.30$_{0.10}$
		35.85±3.23$_{0.10}$	85.38±3.57$_{0.10}$	0.77±0.22$_{0.20}$	11.34±0.77$_{0.20}$
		17.01±1.52$_{0.15}$	70.34±11.49$_{0.15}$	0.71±0.22$_{0.30}$	3.28±0.94$_{0.30}$
SSL	Clean	66.85±2.30	92.76±0.91	88.69±1.28	82.23±1.59
	Poisoned$_\alpha$	44.64±2.01$_{0.05}$	53.33±13.48$_{0.05}$	32.38±10.77$_{0.10}$	34.72±1.71$_{0.10}$
		10.86±1.21$_{0.10}$	26.64±10.1$_{0.10}$	6.12±2.13$_{0.20}$	21.86±1.01$_{0.20}$
		3.4±1.11$_{0.15}$	12.14±4.66$_{0.15}$	2.42±0.41$_{0.30}$	11.90±0.81$_{0.30}$

representation trained just on the clean source dataset (Source Only in Table 2) achieves very high accuracy indicating the domains are already well aligned in terms of conditional distributions as well. As a result UDA methods can easily align correct classes and only suffer a drop close to 10% which is the amount of poisoned data added. However, this is not an interesting case for evaluation of UDA methods as domains can be aligned just by training on the source domain data without the need of target domain data.

4.2 Poisoning with mislabeled watermarked data

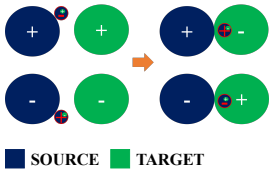


Figure 5: Successful poisoning with mislabeled watermarked data prevents UDA approaches from aligning correct classes from the source and target domains.

In this experiment, we evaluate the effect of using poisoned data that looks like the source domain data. The poisoned data is generated by superimposing an image from the source domain with an image from the target domain. This method of generating poison data is known as watermarking [26]. To generate watermarked poison data we select an image from the target domain (t) and a base image from the source domain (s) such that it has the same class as the target domain image and lies closest to the target image (in the input space). The poisoned image (p) is obtained by a convex combination of the base and target images i.e., $p = \alpha t + (1 - \alpha)s$ where $\alpha \in [0, 1]$. α is selected such that the target image is not visible in the poison image ensuring the poisoned image looks like the image from the source domain. We use the same labeling function as discussed in the previous section to label the poisoned image and add 10% poison data in the source. The illustrative picture of the effect of poisoning in this scenario is presented in Fig. 5. Successful poisoning in this case works just like in the previous experiment i.e., by making the representations of the data from wrong classes in source and target domains similar for DANN/CDAN, by reducing the discrepancy between the classifiers on target data for MCD and inducing a conflict between the supervised and auxiliary task for SSL. The t-SNE embedding showing the effect of poisoning (Fig. 9) and watermarked poison data (Fig. 8) are shown in the Appendix. We evaluate the effectiveness of this method on the Digits dataset for different values of α. The results in Table 3 show a significant decrease in the target domain accuracy even with a small watermarking percentage for all methods except CDAN. This is because the success of CDAN is dependent on the correctness of the pseudo-labels on the target domain data (output of the classifier), which are used in the discriminator. Correct pesudo-labels provide CDAN a positive reinforcement to align correct classes from the two domains, leading to a failure of poisoning. However, as we increase the amount of watermarking, quality of pseudo labels deteriorate. Thus providing a negative reinforcement to CDAN, causing an alignment of wrong classes from the two domains.

4.3 Poisoning using clean-label source and target data

In this experiment, correctly labeled data is used for poisoning. To generate clean-label poison data that can affect the performance of UDA methods we must affect the features of the poison data. This requires solving a bilevel optimization problem [10, 21, 22] which we present in the Appendix. Due to the high computational complexity involved in solving the bilevel problem, we propose to use a simple alternating optimization to demonstrate the feasibility of clean label poisoning attack against UDA. We use the setting of previous works[10, 26] and consider misclassification of a single target domain test point $(x_{\text{test}}^{\text{target}}, y_{\text{test}}^{\text{target}})$ rather than affecting accuracy of the entire target domain as done in previous sections. Let $u = \{u_1, ..., u_n\}$ denote the poisoned data. To ensure clean label, each poison point u_i must have a bounded perturbation from a base point x_i^{base} i.e, $\|u_i - x_i^{\text{base}}\| = \|\delta_i\| \leq \epsilon$ and has label of the base i.e., y_i^{base}. Thus, $\hat{\mathcal{D}}^{\text{poison}} = \{(u_i, y_i^{\text{base}})\}_{i=1}^{N_{\text{poison}}}$, $\hat{\mathcal{D}}_{\text{source}} = \{(x_i^{\text{source}}, y_i^{\text{source}})\}_{i=1}^{N_{\text{source}}}$ and $\hat{\mathcal{D}}_{\text{target}} = \{(x_i^{\text{target}}, y_i^{\text{target}})\}_{i=1}^{N_{\text{target}}}$. The clean-label poison data u is such that when the victim uses $\hat{\mathcal{D}}^{\text{source}} \bigcup \hat{\mathcal{D}}^{\text{poison}}$ and $\hat{\mathcal{D}}^{\text{target}}$ for UDA, the target domain test point $x_{\text{test}}^{\text{target}}, y_{\text{test}}^{\text{target}}$ is misclassified. The optimization problem for clean-label attack is as follows.

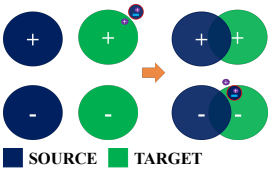


Figure 6: Successful clean-label poisoning attack aligns the target point (purple) close to the wrong class (-).

$$\min_{u} \sum_{i=1}^{N_{\text{poison}}} \left[\|g(x_{\text{test}}^{\text{target}};\theta) - g(u_i;\theta)\|_2^2 + \lambda\|x_i^{\text{base}} - u_i\|\right], \min_{\theta} \mathcal{L}_{\text{UDA}}(\hat{\mathcal{D}}^{\text{source}} \bigcup \hat{\mathcal{D}}^{\text{poison}}, \hat{\mathcal{D}}^{\text{target}};\theta). \quad (5)$$

The first problem minimizes the distance between the representations of the poison and the target domain test data (first term) while ensuring the poison data is not too far from the base data (second term). The second problem optimizes the parameters of the representation using UDA methods. The attack success is evaluated by solving the second problem in Eq. 5 from scratch and evaluating the classification of $x_{\text{test}}^{\text{target}}$. This is illustrated in Fig. 6. Left part shows the case before retraining using the poison data generated from Eq. 5 and the right part shows how poisoning induces misclassification. We use two approaches for poisoning. The first uses source domain data and second uses target domain data to as base data. We add 1% poisoned data and test the effect of poisoning on a two-class (3 vs 8) domain adaptation problem on MNIST→ MNIST_M (see Appendix E for details). The results in Fig. 7 show that using target domain data as base data is significantly more successful under small permissible perturbation (ϵ). Using base data from the source domain requires larger distortion to keep the poison data close to the target point in the representation space and is hence less successful. This experiment shows the feasibility of clean label attacks against UDA methods. We believe attack success can be further boosted by solving the bilevel level problem (Eq. 7 in Appendix) and is left as future work.

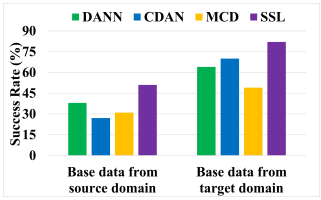


Figure 7: Attack success rate of clean-label poisoning using base data from source/target for a two-class problem in MNIST → MNIST_M.

5 Conclusion

We study the important problem of UDA and highlight the limitations of learning under this setting. We propose a simple lower bound on the target domain error for UDA, dependent on the labeling function induced by the representation. The lower bound demonstrates that learning a domain invariant representation while minimizing error on the source domain cannot guarantee good generalization on the target domain. We analyze a simple model and showe existence of cases where UDA can naturally succeed or fail. The analysis also highlights a case where, without access to any labeled target domain data, success and failure are equally likely. In such a case, presence of even small amount of poisoned data can make the data distribution unfavorable for UDA methods, making them fail dramatically in comparison to the case without poisoning. We propose novel data poisoning attacks and use them to demonstrate the failure of popular UDA methods with small amount of poisoned data. Our results suggest that performance of a UDA method in presence of poisoned data provides an indication of how well the method aligns the conditional distributions across the two domains.Thus, we believe our attacks can be used for evaluating UDA methods, beyond simple benchmark datasets, to reveal their robustness in data distributions unfavorable for UDA.

References

[1] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1):151–175, 2010.

[2] Shai Ben-David, John Blitzer, Koby Crammer, Fernando Pereira, et al. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19:137, 2007.

[3] Shai Ben-David and Ruth Urner. On the hardness of domain adaptation and the utility of unlabeled target samples. In *International Conference on Algorithmic Learning Theory*, pages 139–153. Springer, 2012.

[4] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012.

[5] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[6] Remi Tachet des Combes, Han Zhao, Yu-Xiang Wang, and Geoff Gordon. Domain adaptation with conditional distribution matching and generalized label shift. *arXiv preprint arXiv:2003.04475*, 2020.

[7] Nicolas Courty, Rémi Flamary, Amaury Habrard, and Alain Rakotomamonjy. Joint distribution optimal transportation for domain adaptation. *arXiv preprint arXiv:1705.08848*, 2017.

[8] Shai Ben David, Tyler Lu, Teresa Luu, and Dávid Pál. Impossibility theorems for domain adaptation. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 129–136. JMLR Workshop and Conference Proceedings, 2010.

[9] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

[10] W Ronny Huang, Jonas Geiping, Liam Fowl, Gavin Taylor, and Tom Goldstein. Metapoison: Practical general-purpose clean-label data poisoning. *arXiv preprint arXiv:2004.00225*, 2020.

[11] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 19–35. IEEE, 2018.

[12] Yujie Ji, Xinyang Zhang, and Ting Wang. Backdoor attacks against learning systems. In *2017 IEEE Conference on Communications and Network Security (CNS)*, pages 1–9. IEEE, 2017.

[13] Jaeho Lee and Maxim Raginsky. Minimax statistical learning with wasserstein distances. *arXiv preprint arXiv:1705.07815*, 2017.

[14] Hong Liu, Mingsheng Long, Jianmin Wang, and Michael Jordan. Transferable adversarial training: A general approach to adapting deep classifiers. In *International Conference on Machine Learning*, pages 4013–4022. PMLR, 2019.

[15] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pages 97–105. PMLR, 2015.

[16] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. *arXiv preprint arXiv:1705.10667*, 2017.

[17] Mingsheng Long, Jianmin Wang, Guiguang Ding, Jiaguang Sun, and Philip S Yu. Transfer joint matching for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1410–1417, 2014.

[18] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Unsupervised domain adaptation with residual transfer networks. *arXiv preprint arXiv:1602.04433*, 2016.

[19] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms. *arXiv preprint arXiv:0902.3430*, 2009.

[20] Yishay Mansour and Mariano Schain. Robust domain adaptation. *Annals of Mathematics and Artificial Intelligence*, 71(4):365–380, 2014.

[21] Akshay Mehra and Jihun Hamm. Penalty method for inversion-free deep bilevel optimization. *arXiv preprint arXiv:1911.03432*, 2019.

[22] Akshay Mehra, Bhavya Kailkhura, Pin-Yu Chen, and Jihun Hamm. How robust are randomized smoothing based defenses to data poisoning? *arXiv preprint arXiv:2012.01274*, 2020.

[23] Shike Mei and Xiaojin Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[24] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 27–38, 2017.

[25] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3723–3732, 2018.

[26] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. *arXiv preprint arXiv:1804.00792*, 2018.

[27] Jian Shen, Yanru Qu, Weinan Zhang, and Yong Yu. Wasserstein distance guided representation learning for domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[28] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Clean-label backdoor attacks. 2018.

[29] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7167–7176, 2017.

[30] Zirui Wang, Zihang Dai, Barnabás Póczos, and Jaime Carbonell. Characterizing and avoiding negative transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11293–11302, 2019.

[31] Jiaolong Xu, Liang Xiao, and Antonio M López. Self-supervised domain adaptation for computer vision tasks. *IEEE Access*, 7:156694–156706, 2019.

[32] Han Zhao, Remi Tachet Des Combes, Kun Zhang, and Geoffrey Gordon. On learning invariant representations for domain adaptation. In *International Conference on Machine Learning*, pages 7523–7532. PMLR, 2019.

[33] Han Zhao, Shanghang Zhang, Guanhang Wu, José MF Moura, Joao P Costeira, and Geoffrey J Gordon. Adversarial multiple source domain adaptation. *Advances in neural information processing systems*, 31:8559–8570, 2018.

[34] Chen Zhu, W Ronny Huang, Ali Shafahi, Hengduo Li, Gavin Taylor, Christoph Studer, and Tom Goldstein. Transferable clean-label poisoning attacks on deep neural nets. *arXiv preprint arXiv:1905.05897*, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Sec. 3 and 4
   (b) Did you describe the limitations of your work? [Yes] See Sec. 3
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Sec. 3 and Appendix A
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Appendix E
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix E
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix E
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]