Iterative Teaching by Data Hallucination

Zeju Qiu[1,3,*], **Weiyang Liu**[1,2,*], **Tim Z. Xiao**[4], **Zhen Liu**[5], **Umang Bhatt**[2,6]
Yucen Luo[1], **Adrian Weller**[2,6], **Bernhard Schölkopf**[1]
[1]Max Planck Institute for Intelligent Systems, Tübingen, [2]University of Cambridge, [3]Technical University of Munich
[4]University of Tübingen, [5]Mila, Université de Montréal, [6]The Alan Turing Institute

Abstract

We consider the problem of iterative machine teaching, where a teacher sequentially provides examples based on the status of a learner under a discrete input space (*i.e.*, a pool of finite samples), which greatly limits the teacher's capability. To address this issue, we study iterative teaching under a continuous input space where the input example (*i.e.*, image) can be either generated by solving an optimization problem or drawn directly from a continuous distribution. Specifically, we propose data hallucination teaching (DHT) where the teacher can generate input data intelligently based on labels, the learner's status and the target concept. We study a number of challenging teaching setups (*e.g.*, linear/neural learners in omniscient and black-box settings). Extensive empirical results verify the effectiveness of DHT.

1 Introduction

Machine teaching [1, 2] seeks a training dataset of minimal size such that a learner can learn a target concept based on this minimal dataset. Compared to machine learning where a learner is provided with a dataset to find the optimal parameters, machine teaching studies the inverse problem where the goal is to find a minimal dataset with which the learner can converge to the given target parameters. A deeper understanding towards machine teaching is essential in many applications, such as crowd sourcing [3, 4, 5, 6], optimal education [1], model robustness [7, 8, 9, 10], curriculum learning [11] and dataset distillation [12].

Depending on the type of learner, machine teaching can be carried out batch-wise (*i.e.*, the teacher provides the dataset to the learner in one shot) or iteratively (*i.e.*, the teacher provides data to the learner iteratively and adaptively). Motivated by the dominance of iterative learners (*e.g.*, almost all types of neural networks), we study the problem of iterative machine teaching (IMT) [13] where the teacher feeds data intelligently based on the learner's status in every iteration such that the learner can converge to the target concept within minimal iterations. The minimal number of such iterations is defined as *iterative teaching dimension*. Vanilla IMT [13] iteratively selects examples from a fixed pool (*i.e.*, dataset), which, however, is inherently a difficult combinatorial problem computationally prohibitive to solve. [14] addresses this problem by finding a continuous teaching signal – the label space. Despite its simplicity, label synthesis teaching still imposes a strong constraint on the teaching space, limiting its capability of faster convergence. To avoid the combinatorial problem of example selection while enjoying the flexibility of a continuous teaching space, we propose data hallucination teaching (DHT), where the teacher generates from a continuous space the input data by conditioning on the learner's status. An intuitive comparison between IMT and DHT is given in Figure 1.

Another motivation behind DHT comes from the promising results of approximating a dataset with synthetic prototypes, such as dataset distillation [12, 15] and dataset condensation [16]. DHT shares the same spirit as dataset approximation in the sense that both aim to guide the learner to

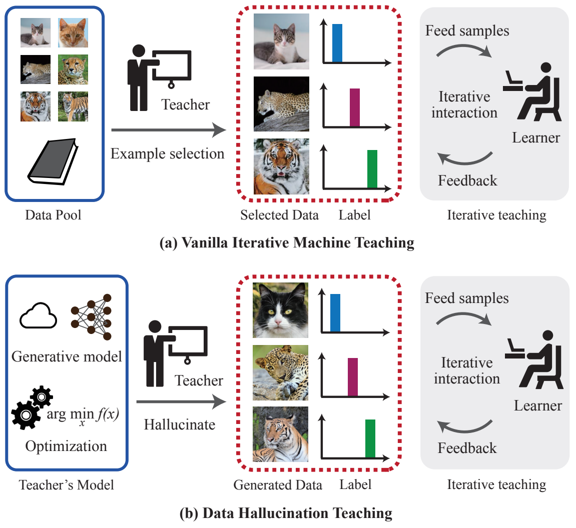

Figure 1: Comparison of vanilla iterative machine teaching and the proposed data hallucination teaching.

[*]Zeju Qiu and Weiyang Liu contributed equally to this work. This work was done when Zeju Qiu did an internship with Weiyang Liu at MPI. Correspondence to Weiyang Liu (`wl396@cam.ac.uk`).

some target concept with synthetic samples. Different from dataset approximation, DHT takes one step further by taking the specific iterative optimization algorithm into account and seeks to generate a sequence of examples (with ordering information) rather than a synthetic dataset.

DHT can also be viewed as a natural generalization of IMT, extending the original discrete teaching space to a continuous one. Such a generalization introduces more modeling flexibility but meanwhile makes the teaching process more challenging. To tackle this challenge, we study both greedy teaching policy and parameterized teaching policy. Particularly for the parameterized one, we propose multiple teacher formulations (*e.g.*, generative models) and multiple teacher's action spaces (*e.g.*, Mixup sample space [17]). We emphasize that DHT is quite different from standard generative models which usually capture static data distributions. In contrast, DHT models a *dynamically changing* data distribution, which depends on the learner's status and is used for fast convergence rather than reconstruction.

The intuition behind the benefits of the continuous teaching space in DHT comes from the empirical success of Mixup [17, 18] and data augmentation [19, 20, 21, 22]. Mixup uses a linear interpolation between two samples for training neural networks. Data augmentation perturbs the inputs, *e.g.*, images, in a small neighborhood around the original input. Both methods can be viewed as a continuous perturbation in the high-dimensional input space and special cases of DHT. Even if it is only a small subset of the continuous input space being considered, the empirical generalization performance can be significantly improved. Therefore, the original discrete input space can be sub-optimal for teaching, for which we propose to explore the continuous input space in order to improve the learner's convergence.

Specifically, we study DHT under both omniscient scenario, where the teacher knows everything (particularly the optimal learner parameters) about the learner, and black-box scenario, where the teacher has no prior knowledge of the optimal learner parameters. We theoretically prove that DHT can achieve exponential teachability (ET) [13], and empirically show that DHT achieves much faster convergence than a random teacher (*i.e.*, SGD) and IMT [13].

Most significantly, we formulate the problem of teaching black-box neural learners as *performative teaching*, which is a novel application of performativity [23] to iterative teaching. Specifically, performative teaching assumes that the teaching target will shift based on the teacher's action. This is exactly the scenario when we perform iterative teaching in the representation space of neural networks. We show that under this formulation, DHT is able to teach deep neural learners in a fully black-box manner on realistic datasets such as CIFAR-10 and CIFAR-100. We believe that it is the very first time that iterative teaching can be used to teach black-box nonlinear learners in realistic settings while achieving significant empirical performance gain.

Our contribution can be briefly summarized as follows:

- We propose a novel teaching framework – **data hallucination teaching**, where the teacher iteratively generates synthetic training data depending on the learner's status. DHT yields a highly flexible teaching space.
- In the DHT framework, we comprehensively study the greedy and parameterized policies under both the omniscient and black-box scenarios.
- We propose a novel performative formulation for iterative teaching, which assumes a dynamically changing teaching target. The formulation is shown to be a natural fit for teaching black-box neural learners.
- For the first time, we are able to apply iterative teaching to black-box neural learners on realistic datasets. Significant performance gain is observed empirically.
- We demonstrate faster convergence of DHT versus SGD and other baselines, both theoretically and empirically.

2 Related Work

Machine teaching. The study of machine teaching begins with the batch setting [24, 1, 25, 26], where the teacher simply prepares a dataset of minimal size to the learner towards some target concept. Efforts have been made on the teaching behavior of different types of learner, such as version space learners [27, 28], linear learners [25], kernel learners [29], reinforcement learner [30], active learners [31, 32], teacher-aware learners [33] and forgetful learners [34, 35]. Iterative (or sequential) machine teaching [13, 35, 36, 26, 37, 14] studies iterative learners by considering the specific optimization algorithm that the learner uses. The teaching performance is measured by the learner's convergence. Machine teaching has diverse applications in reinforcement learning [38, 39, 9, 40], human-in-the-loop learning [41, 42, 43], crowd sourcing [3, 5, 6] and cyber security [44, 7, 45, 46, 47]. Sharing similar spirits, cooperative communication [48, 49, 50] also studies the interaction between a teacher and a learner as well as how information can be transmitted efficiently.

Data augmentation. In deep learning, data augmentation is ubiquitous [19, 51, 52, 22] and plays a crucial role in regularizing neural networks and improving generalization. Without it, the training set can be easily fitted and training loss will be minimized to zero even with random labels [53]. Data augmentation is the *de facto* choice in image recognition [19, 54, 55] and also one of the key ingredients to the success of contrastive learning [22, 56].

Dataset approximation. How to approximate a dataset with a few representative prototypes that can be used for training remains an open problem and is actively studied in coreset [57, 58, 59, 60], dataset pruning [61, 62], dataset distillation [12, 15] and dataset condensation [16]. However, dataset approximation typically considers one batch of data, while DHT constructs a sequence of data samples.

3 Data Hallucination Teaching

3.1 Problem Settings

Teaching protocol. We generally follow the teaching protocol in [13, 14]. This section mostly considers the omniscient scenario. That is, both the teacher and the learner observe the same sample $\mathcal{A}$ and share the same feature space, which represents $\mathcal{A}$ as $\boldsymbol{x}$ with the label $\boldsymbol{y}$. The teacher knows all the information about the learner, including the model parameters $\boldsymbol{w}^i$ at the i-th iteration, the learning rate η_i, the loss function ℓ and the optimization algorithm (usually we consider SGD). The teacher can only feed examples $(\boldsymbol{x}^i, \boldsymbol{y}^i)$ to the learner at the i-th iteration.

Teacher's objective. In the omniscient scenario, the teacher aims to provide examples to the learner in every iteration such that the learner parameters $\boldsymbol{w}$ converge to the desired parameters $\boldsymbol{w}^*$ as quickly as possible. We typically use $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})}\{\ell(\boldsymbol{x},\boldsymbol{y}|\boldsymbol{w})\}$. The teacher seeks to optimize the following objective for gradient decent learner:

$$\min_{\{(\boldsymbol{x}^1,\boldsymbol{y}^1),\cdots,(\boldsymbol{x}^T,\boldsymbol{y}^T)\}} T \\ \text{s.t.} \begin{cases} d(\boldsymbol{w}^T, \boldsymbol{w}^*) \leq \epsilon \\ \boldsymbol{w}^{t+1} = \boldsymbol{w}^t - \eta_t \frac{\partial \ell(\boldsymbol{x}^{t+1},\boldsymbol{y}^{t+1}|\boldsymbol{w}^t)}{\partial \boldsymbol{w}^t} \end{cases} \tag{1}$$

where $d(\cdot,\cdot)$ denotes some discrepancy measure (*e.g.*, Euclidean distance or cosine similarity). The above optimization is in general intractable and ϵ is hard to set in practice, so we usually resort to a simpler teacher's objective:

$$\min_{\{(\boldsymbol{x}^1,\boldsymbol{y}^1),\cdots,(\boldsymbol{x}^T,\boldsymbol{y}^T)\}} d(\boldsymbol{w}^T, \boldsymbol{w}^*) \tag{2}$$

where T is the prescribed termination iteration. This minimization aims to find a teaching trajectory of length T such that the distance between $\boldsymbol{w}^T$ and $\boldsymbol{w}^*$ reaches the minimum.

Learner's objective. The learner minimizes its loss function ℓ with examples given by the teacher. If the teacher feeds one example at a time, gradient descent learners use

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^t - \eta_t \frac{\partial \ell(\boldsymbol{x}^{t+1},\boldsymbol{y}^{t+1}|\boldsymbol{w}^t)}{\partial \boldsymbol{w}^t} \tag{3}$$

where ℓ can be any regression or classification loss.

3.2 Greedy Teaching Policy

We start with the simplest greedy teaching policy which approximates Equation 2 with T-times one-step minimization. DHT aims to generate the teaching example $(\boldsymbol{x},\boldsymbol{y})$. To simplify the problem, we uniformly sample a label $\boldsymbol{y}$ and synthesize the corresponding data $\boldsymbol{x}$ for teaching. After simplification, this leads to the one-step optimization:

$$\min_{\boldsymbol{x}^{t+1}\in\mathcal{X},\boldsymbol{y}^{t+1}\sim\mathbb{U}} \eta_t^2 \left\| \frac{\partial \ell(\boldsymbol{x}^{t+1},\boldsymbol{y}^{t+1}|\boldsymbol{w}^t)}{\partial \boldsymbol{w}^t} \right\|_2^2 \\ - 2\eta_t \langle \boldsymbol{w}^t - \boldsymbol{w}^*, \frac{\partial \ell(\boldsymbol{x}^{t+1},\boldsymbol{y}^{t+1}|\boldsymbol{w}^t)}{\partial \boldsymbol{w}^t} \rangle \tag{4}$$

where $\boldsymbol{x}^{t+1}$ is optimized within $\mathcal{X}$ (*e.g.*, pixel space $[0, 255]$) and $\boldsymbol{y}$ is sampled uniformly from the discrete label space. Equation 4 can be directly used to teach any linear learner such as least square regression and logistic regression. Despite its simplicity, the greedy policy is computationally expensive if $\boldsymbol{x}$ is high-dimensional (*e.g.*, images).

3.3 Parameterized Teaching Policy

The greedy policy considers only a one-step update for the learner, hence is inevitably sub-optimal. However, considering all the possible combinations of $\boldsymbol{x}$ in multiple steps is computationally infeasible, especially when the teaching space $\mathcal{X}$ is continuous. To address this, here we study a parameterized teaching policy. The central idea is to parameterize the teacher by a neural network $\boldsymbol{\theta}$, and the teaching policy is denoted as $\pi_{\boldsymbol{\theta}}$. With a parameterized policy, we can easily consider multiple-step updates for the learner. For example, when taking v-step updates for the learner into account, the teacher optimizes $\min_{\pi_{\boldsymbol{\theta}}} \|\boldsymbol{w}^{t+v} - \boldsymbol{w}^*\|^2$.

3.3.1 Data Transformation

To simplify the problem, we start with a data transformation policy (at the t-th iteration) $\tilde{\boldsymbol{x}} = \pi_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{w}^t,\boldsymbol{w}^*)$ that transforms a randomly sampled data point $(\boldsymbol{x},\boldsymbol{y})$ to a teaching example $(\tilde{\boldsymbol{x}},\boldsymbol{y})$. To learn $\boldsymbol{\theta}$, we have that

$$\min_{\boldsymbol{\theta}} \|\boldsymbol{w}^v(\boldsymbol{\theta}) - \boldsymbol{w}^*\|_2^2 + \alpha \sum_{i=1}^{v} \ell(\pi_{\boldsymbol{\theta}}(\boldsymbol{x}^i,\boldsymbol{y}^i,\boldsymbol{w}^i_{\text{SG}},\boldsymbol{w}^*),\boldsymbol{y}^i|\boldsymbol{w}^i_{\text{SG}}) \\ \text{s.t. } \boldsymbol{w}^v(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{w}} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})}\big\{\ell\big(\pi_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{w},\boldsymbol{w}^*),\boldsymbol{y}|\boldsymbol{w}\big)\big\}$$

where α is a hyperparameter and the learner is initialized at $\boldsymbol{w}^0$. The policy $\pi_{\boldsymbol{\theta}}$ keeps transforming the randomly sampled data based on both the current learner parameters and the target parameters in order to improve the learner's convergence to $\boldsymbol{w}^*$. To solve this bi-level optimization, we can simply unroll the inner optimization with v steps of stochastic gradient descent, which enables the gradient to flow back to $\boldsymbol{\theta}$ when solving the outer minimization. This shares the same spirit as meta-learning [63] and back-propagation through time in recurrent networks [64]. We note that the greedy policy is the special case of $v = 1$. In order to amplify the learning signal, we introduce an auxiliary intermediate loss minimization into the teacher's objective. With this auxiliary term, the teacher will favor the teaching trajectory that not only quickly guides the learner to $\boldsymbol{w}^*$ but also well minimizes the learner's loss function. In the implementation, we simplify the gradient in the auxiliary term by replacing $\boldsymbol{w}^i$ with $\boldsymbol{w}^i_{\text{SG}} = \text{StopGradient}(\boldsymbol{w}^i)$.

3.3.2 Generative Modeling

The data transformation policy builds a learner-conditioned deterministic mapping from existing data to teaching examples, and it does not model the underlying distribution of teaching examples. Moreover, the data transformation policy is likely to generate unrealistic samples that do not match the underlying data distribution $p(\boldsymbol{x})$. To this end, we study the generative teaching policy. The central idea is to parameterize the teaching policy with a generative model and impose a distribution divergence constraint, *i.e.*,

$\text{Div}(p(\pi), p(\boldsymbol{x})) \leq \epsilon$, which is used to force teaching examples to be similar to the empirical data distribution.

GAN-based teacher. One of the simplest ways to perform generative modeling is to use generative adversarial networks (GANs) [65]. Therefore, we parameterize the teacher as a generator and introduce an additional discriminator $D(\cdot)$ to close the gap between synthetic teaching examples and real data. Specifically, the teacher model optimizes

$$\begin{aligned}&\min_{\boldsymbol{\theta}} \max_{D} \mathbb{E}_{\tilde{\boldsymbol{x}}\sim p_\pi(\boldsymbol{z})} \log\big(1-D(\pi_{\boldsymbol{\theta}}(\boldsymbol{z}))\big) + \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})} \log\big(D(\boldsymbol{x})\big)\\ &+\|\boldsymbol{w}^v(\boldsymbol{\theta})-\boldsymbol{w}^*\|_2^2 + \alpha\sum_{i=1}^{v} \ell\big(\pi_{\boldsymbol{\theta}}(\boldsymbol{z},\boldsymbol{x}^i,\boldsymbol{y}^i,\boldsymbol{w}^i_{\text{SG}},\boldsymbol{w}^*),\boldsymbol{y}^i|\boldsymbol{w}^i_{\text{SG}}\big)\\ &\text{s.t. } \boldsymbol{w}^v(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{w}} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})}\big\{\ell\big(\pi_{\boldsymbol{\theta}}(\boldsymbol{z},\boldsymbol{x},\boldsymbol{y},\boldsymbol{w},\boldsymbol{w}^*),\boldsymbol{y}\big)|\boldsymbol{w}\big\}\end{aligned}$$

where $\boldsymbol{z}$ is a noise vector following a normal distribution and we sometimes omit the input arguments $(\boldsymbol{x},\boldsymbol{y},\boldsymbol{w},\boldsymbol{w}^*)$ for $\pi_{\boldsymbol{\theta}}$ for notational simplicity. Similar to learning the data transformation policy, we unroll v steps of SGD for the inner optimization and put $\boldsymbol{w}^v(\boldsymbol{\theta})$ into the outer min-max optimization for end-to-end training. This outer min-max problem can be solved following standard GAN training.

VAE-based teacher. We take advantage of a pretrained variational autoencoder (VAE) [66] to realize the distribution divergence constraint. Specifically, we first pretrain a VAE on the full dataset to capture the joint distribution $p(\boldsymbol{x},\boldsymbol{y})$, and then let the teaching policy to generate data in the latent space of the VAE. The objective for the teacher is

$$\begin{aligned}&\min_{\boldsymbol{\theta}} \|\boldsymbol{w}^v-\boldsymbol{w}^*\|_2^2 + \alpha\sum_{i=1}^{v} \ell\big(p_{\boldsymbol{\psi}}(\pi_{\boldsymbol{\theta}})|\boldsymbol{w}^i_{\text{SG}}\big) + \text{KL}\big(\pi_{\boldsymbol{\theta}}||p(\boldsymbol{u})\big)\\ &\text{s.t. } \boldsymbol{w}^v(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{w}} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})}\big\{\ell(\pi_{\boldsymbol{\theta}}(\boldsymbol{u},\boldsymbol{x},\boldsymbol{y},\boldsymbol{w},\boldsymbol{w}^*),\boldsymbol{y})|\boldsymbol{w}\big\}\end{aligned}$$

where we denote the fixed standard Gaussian prior for VAE's latent variable $\boldsymbol{u}$ as $p(\boldsymbol{u})$ and the decoder as $p_{\boldsymbol{\psi}}(\boldsymbol{x},\boldsymbol{y}|\boldsymbol{u})$. $\text{KL}(\cdot||\cdot)$ is the Kullback–Leibler divergence. VAE essentially serves as a bridge between the latent space and the input data space, and the teacher generates the latent code which is then mapped to raw data by the decoder. Compared to GAN-based teacher, VAE-based teacher enjoys stronger training stability and also avoids the problem of mode collapse. Since the GAN-based teacher is jointly trained with the learner, it may produce examples that achieve better teaching performance but yield weaker semantic meaning.

3.4 Theoretical Insights and Discussions

Similar to sample selection [13, 35] and label synthesis [14], we now show that the greedy DHT can provably achieve ET. We consider two types of linear learners here: $\ell_{\text{LSR}}(\boldsymbol{x},y|\boldsymbol{w}) = \frac{1}{2}(\langle\boldsymbol{w},\boldsymbol{x}\rangle - y)^2$ for the least square regression (LSR) learner and $\ell_{\text{LR}}(\boldsymbol{x},y|\boldsymbol{w}) = \log(1+\exp\{-y\langle\boldsymbol{w},\boldsymbol{x}\rangle\})$ for the logistic regression (LR) learner. For simplicity, we consider the case where the label is a scalar. For LSR, the gradient *w.r.t.* $\boldsymbol{w}$ of a single sample $(\boldsymbol{x},y)$ is $\nabla_{\boldsymbol{w}}\ell = (\langle\boldsymbol{w},\boldsymbol{x}\rangle - y)\boldsymbol{x}$. For LR learners, the gradient is $\nabla_{\boldsymbol{w}}\ell = \frac{-y\boldsymbol{x}}{1+\exp(y\langle\boldsymbol{w},\boldsymbol{x}\rangle)}$. Then we define $g(\tilde{\boldsymbol{x}})$ as the teaching gradient ratio that is used to quantify the scale difference between the gradient of a normal sample and that of a teaching example. $\mathcal{T}_{\boldsymbol{x}\rightarrow\tilde{\boldsymbol{x}}}$ is defined as the transformation operator that maps $\boldsymbol{x}$ to $\tilde{\boldsymbol{x}}$, *i.e.*, $\tilde{\boldsymbol{x}} = \mathcal{T}_{\boldsymbol{x}\rightarrow\tilde{\boldsymbol{x}}} \circ \boldsymbol{x}$. We have

$$\nabla_{\boldsymbol{w}}\ell(\tilde{\boldsymbol{x}},y|\boldsymbol{w}) = g(\tilde{\boldsymbol{x}}) \cdot \mathcal{T}_{\boldsymbol{x}\rightarrow\tilde{\boldsymbol{x}}} \circ \nabla_{\boldsymbol{w}}\ell(\boldsymbol{x},y|\boldsymbol{w}). \quad (5)$$

For LSR, we have that $g(\tilde{\boldsymbol{x}}) = \frac{\langle\boldsymbol{w},\boldsymbol{x}\rangle-y}{\langle\boldsymbol{w},\tilde{\boldsymbol{x}}\rangle-y}$. For LR, we have that $g(\tilde{\boldsymbol{x}}) = \frac{1+\exp(y\langle\boldsymbol{w},\tilde{\boldsymbol{x}}\rangle)}{1+\exp(y\langle\boldsymbol{w},\boldsymbol{x}\rangle)}$. We note that $g(\tilde{\boldsymbol{x}})$ is important for the convergence of the learner and also largely determines the teacher's ability to achieve ET. Following prior work [13, 35], ET is defined as the ability for the teacher to guide the learner to converge to $\boldsymbol{w}^*$ at an exponential rate.

Theorem 1 (Exponential teachability of DHT)**.** *Assume that the learner loss ℓ_i has the property of interpolation, L_i-Lipschitz, and convexity. f is order-1 μ strongly convex. Then DHT can achieve ET if $\mathcal{T}_{\boldsymbol{x}\rightarrow\tilde{\boldsymbol{x}}}$ is an scaling mapping, i.e., $\mathcal{T}_{\boldsymbol{x}\rightarrow\tilde{\boldsymbol{x}}} \circ \boldsymbol{x} = \beta\boldsymbol{x}$ and β is adjusted such that $g(\tilde{\boldsymbol{x}}) = c_1\|\boldsymbol{w}^t - \boldsymbol{w}^*\|$. Specifically, we have that*

$$\mathbb{E}\{\|\boldsymbol{w}^T-\boldsymbol{w}^*\|^2\} \leq (1-c_1\eta_t\bar{\mu}+c_1^2\eta_t^2 L_{\max})^T \left\|\boldsymbol{w}^0 - \boldsymbol{w}^*\right\|^2 \quad (6)$$

in which $L_{\max} = \max_i L_i$ and $\bar{\mu} = \sum_i \mu_i/n$. It implies that $\mathcal{O}((\log\frac{1}{c_0})^{-1}\log(\frac{1}{\epsilon}))$ samples are needed to achieve $\mathbb{E}\{\|\boldsymbol{w}^T - \boldsymbol{w}^*\|^2\} \leq \epsilon$. We let $c_0 = 1 - c_1\eta_t\bar{\mu} + c_1^2\eta_t^2 L_{\max}$ and c_1 is adjusted such that $0 < c_1\eta_t < \bar{\mu}/L_{\max}$ holds.

Theorem 1 validates the importance of $g(\tilde{\boldsymbol{x}})$ in achieving ET. When $\mathcal{T}_{\boldsymbol{x}\rightarrow\tilde{\boldsymbol{x}}}$ is a scaling mapping, DHT recovers the case of label synthesis [14]. Thus, DHT can always achieve better convergence rate than label synthesis. Further, DHT enjoys all the theoretical guarantees for label synthesis. When $\mathcal{T}_{\boldsymbol{x}\rightarrow\tilde{\boldsymbol{x}}}$ is a nonlinear mapping, DHT will become very flexible and potentially a better convergence rate can be derived.

4 Black-box DHT for Neural Learners

In this section, we discuss how DHT can be used to teach neural learners in a black-box setting. Black-box teaching for neural teachers has long been an open challenge in iterative machine teaching. We start by studying how parameterized DHT can be extended to the black-box setting and then introduce a novel alternative – performative teaching which can naturally be used to teach neural learners.

Black-box teaching is generally difficult due to two aspects. First, the optimal learner parameters $\boldsymbol{w}^*$ are no longer given and how to find a good surrogate to measure the distance to $\boldsymbol{w}^*$ is crucial (essentially when the learner is nonlinear and non-convex). In general, the goal of black-box teaching is to improve the learner's generalizability instead of its convergence to some $\boldsymbol{w}^*$. Therefore, we usually seek to find a surrogate for $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathbb{P}_{\text{real}}}\{\ell(\boldsymbol{x},\boldsymbol{y}|\boldsymbol{w})\}$ where $\mathbb{P}_{\text{real}}$ denotes the underlying joint data and label distribution. Second, the teaching space of DHT is huge and how to properly reduce the teaching space to a reasonably small yet sufficiently effective one is important. Black-box teaching shares a similar ultimate goal to knowledge distillation [67].

4.1 Mixup-based Teaching

We propose a black-box DHT based on the data augmentation space in Mixup [17]. The basic idea is to learn a teacher that produces the learner-conditioned Mixup coefficients.

Surrogate target. We use a simple surrogate to measure the distance to $\boldsymbol{w}^*$: the validation accuracy on a held-out validation data set that is not used for training the learner. Recent studies in neural architecture search [68, 69, 70], meta-learning [71] and automated machine learning [72] validate the effectiveness of such a surrogate to approximate the distance to a generalizable $\boldsymbol{w}^*$.

Teaching space. We restrict the teacher's action space to the data augmentation space in Mixup. Specifically, the teaching policy outputs the interpolation between two samples:

$$\pi_{\boldsymbol{\theta}}\big((\boldsymbol{x}_1,\boldsymbol{y}_1),(\boldsymbol{x}_2,\boldsymbol{y}_2),\boldsymbol{w}^t\big)=\lambda_{\boldsymbol{\theta}}\boldsymbol{x}_1+(1-\lambda_{\boldsymbol{\theta}})\boldsymbol{x}_2 \tag{7}$$

where $\lambda_{\boldsymbol{\theta}}=h_{\boldsymbol{\theta}}((\boldsymbol{x}_1,\boldsymbol{y}_1),(\boldsymbol{x}_2,\boldsymbol{y}_2),\boldsymbol{w}^t)$. $h_{\boldsymbol{\theta}}(\cdot)$ is a neural network parameterized by $\boldsymbol{\theta}$ and outputs the Mixup coefficient λ for mixing $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. The teaching example is $\tilde{\boldsymbol{x}}=\lambda_{\boldsymbol{\theta}}\boldsymbol{x}_1+(1-\lambda_{\boldsymbol{\theta}})\boldsymbol{x}_2$, and its label is $\tilde{\boldsymbol{y}}=\lambda_{\boldsymbol{\theta}}\boldsymbol{y}_1+(1-\lambda_{\boldsymbol{\theta}})\boldsymbol{y}_2$. Specifically, we can learn a teacher network $h(\cdot)$ that either outputs a continuous value within $[0,1]$ or outputs a discrete value (*e.g.*, $\{0,0.25,0.5,0.75,1\}$). For the latter case, the teacher is a classifier for the discrete Mixup coefficients.

Unrolling. We first formulate the learning of the teacher as a bi-level optimization similar to the one in Section 3.3.1. The outer optimization is to minimize the empirical risk on the validation set $\mathcal{D}_a$ and the inner optimization is to minimize the empirical risk on the training set $\mathcal{D}_r$. We have

$$\min_{\boldsymbol{\theta}}\mathbb{E}_{(\boldsymbol{x}_a,\boldsymbol{y}_a)\sim\mathcal{D}_a}\big\{\ell(\boldsymbol{x}_a,\boldsymbol{y}_a|\boldsymbol{w}^v(\boldsymbol{\theta}))\big\}$$
$$\text{s.t. }\boldsymbol{w}^v(\boldsymbol{\theta})=\arg\min_{\boldsymbol{w}}\mathbb{E}_{(\boldsymbol{x}_r,\boldsymbol{y}_r)\sim\mathcal{D}_r}\big\{\ell\big(\pi_{\boldsymbol{\theta}}(\boldsymbol{x}_r,\boldsymbol{y}_r,\boldsymbol{w}),\tilde{\boldsymbol{y}}_r|\boldsymbol{w}\big)\big\}$$

which can be solved by unrolling a few gradient descent steps of the inner minimization to the outer minimization, similar to works [63, 70]. $\pi_{\boldsymbol{\theta}}$ consists of a network $h_{\boldsymbol{\theta}}(\cdot)$ that outputs the Mixup coefficient. Here the empirical risk on the validation set serves as a proxy to the distance to $\boldsymbol{w}^*$.

Policy gradient. Alternatively, we can also resort to the policy gradient approach [73]. We can use the accuracy on the validation set as the reward signal R. Thus simply maximizing this terminal reward: $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}):=\mathbb{E}_{\pi_{\boldsymbol{\theta}}}\{R\}$ leads to the update rule for the teacher: $\boldsymbol{\theta}\leftarrow\boldsymbol{\theta}+\eta\cdot\nabla_{\boldsymbol{\theta}}J(\boldsymbol{\theta})$ where $\nabla_{\boldsymbol{\theta}}J(\boldsymbol{\theta})=\sum_t\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(a_t|s_t)R$ and (a_t,s_t) is the state-action pair at the t-th iteration. For the state features, we use the current learner's predictions of representative samples. For the action space, we use a discrete Mixup coefficient space $\{0,0.5,1\}$ to reduce the search space. The overall training is conceptually similar to prior work [68].

4.2 Performative Teaching

The concept of performativity has been studied primarily in economics [74, 75] and recently in machine learning [23, 76]. When supporting consequential decision-making, predictive models can produce actions that influence the outcome they aim to predict at the beginning. These predictions are called *performative*.

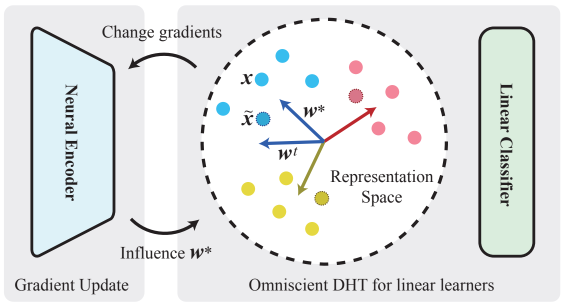

Figure 2: Performative teaching for black-box neural learners.

As shown in Figure 2, we decompose a neural network into two components: neural encoder $g_1(\cdot)$ which is used to extract features, and linear classifier $g_2(\cdot)$ which is used to obtain class labels. Suppose we teach the last-layer classifier of a neural network with omniscient DHT and the teaching example will thus change the gradients for updating the neural encoder. Then after the neural encoder gets updated, the teaching target $\boldsymbol{w}^*$ will also be shifted because the data representation changes. The entire process is iteratively executed. Inspired by the striking connection between performativity and iterative teaching in the representation space, we introduce performative teaching where the teaching target $\boldsymbol{w}^*$ will change dynamically according to the teacher's action. In the context of teaching black-box neural learners, performative teaching is formulated as

$$\min_{(\boldsymbol{x}_t,\boldsymbol{y}_t)} d(\boldsymbol{w}^t,\boldsymbol{w}^*(t)),\quad \text{s.t. }\boldsymbol{w}^*(t)\sim\mathcal{M}(\boldsymbol{x}_{t-1},\boldsymbol{y}_{t-1}) \tag{8}$$

where $\boldsymbol{w}^*(t)$ is the target parameters at the t-the iteration and $\mathcal{M}(\boldsymbol{x}_i,\boldsymbol{y}_i)$ denotes the distribution of the target learner parameters that is dynamically dependent on the teaching example $(\boldsymbol{x}_i,\boldsymbol{y}_i)$. In training, Equation 8 is solved alternately with the gradient update for the neural encoder.

If we want to use performative teaching to train neural learners in practice, we still need to consider a few unresolved problems. First, we have to estimate $\boldsymbol{w}^*$ in each iteration. Because the teaching is performed for linear classifiers, estimating $\boldsymbol{w}^*$ is relatively easy. We simply run a few more gradient descent steps to update the last-layer linear classifiers with the neural encoder fixed, and the resulting classifier weights are viewed as an approximate $\boldsymbol{w}^*$. Second, we need to develop a concrete algorithm to minimize $d(\boldsymbol{w}^t,\boldsymbol{w}^*(t))$ even if $\boldsymbol{w}^*(t)$ can be estimated. We resort to the simplest greedy teaching algorithm. In order to preserve the semantic meaning of the original feature $\boldsymbol{x}$ (given its ground truth label $\boldsymbol{y}$) and to remove potential degenerate solutions, we optimize the teaching example in an ϵ-neighborhood of $\boldsymbol{x}$, *e.g.*, $\|\tilde{\boldsymbol{x}}-\boldsymbol{x}\|\leq\epsilon$. Combining all the pieces, we summarize our performative teaching algorithm for training a

Algorithm 1 Performative teaching for neural learners

```
1. Randomly initialize the neural network. We denote the neural weights of $g_1$ as $\boldsymbol{v}$, and the neural weights of $g_2$ as $\boldsymbol{w}$;
for $i = 1, 2, \cdots, T_1$ do
    2. Form a mini-batch of $m$ samples and perform inference to extract features, denoted as $(\boldsymbol{x}_1^i, \boldsymbol{y}_1^i), \cdots, (\boldsymbol{x}_m^i, \boldsymbol{y}_m^i)$.
    3. $\boldsymbol{w}_{\text{buffer}} \leftarrow \boldsymbol{w}$.
    4. Fix $\boldsymbol{v}$ and update $\boldsymbol{w}$ by minimizing the empirical risk on the training set (e.g., a few SGD steps).
    5. $\boldsymbol{w}^* \leftarrow \boldsymbol{w}$ and then $\boldsymbol{w} \leftarrow \boldsymbol{w}_{\text{buffer}}$.
    for $j = 1, 2, \cdots, m$ do
        6. Solve the greedy teaching problem for the $j$-th sample:
            $\tilde{\boldsymbol{x}}_j^i = \arg\min_{\boldsymbol{x}} \eta_t^2 \left\| \frac{\partial \ell(\boldsymbol{x}, \boldsymbol{y}_j^i | \boldsymbol{w})}{\partial \boldsymbol{w}} \right\|_2^2 - 2\eta_t \langle \boldsymbol{w} - \boldsymbol{w}^*, \frac{\partial \ell(\boldsymbol{x}, \boldsymbol{y}_j^i | \boldsymbol{w})}{\partial \boldsymbol{w}} \rangle$    (9)
            s.t. $\left\| \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|} - \frac{\boldsymbol{x}_j^i}{\|\boldsymbol{x}_j^i\|} \right\| \leq \epsilon, \ \|\boldsymbol{x}\| = \|\boldsymbol{x}_j^i\|$
    end
    7. Use SGD to update the neural network ($\boldsymbol{w}$ and $\boldsymbol{v}$) by replacing $(\boldsymbol{x}_1^i, \boldsymbol{y}_1^i), \cdots, (\boldsymbol{x}_m^i, \boldsymbol{y}_m^i)$ with $(\tilde{\boldsymbol{x}}_1^i, \boldsymbol{y}_1^i), \cdots, (\tilde{\boldsymbol{x}}_m^i, \boldsymbol{y}_m^i)$.
end
```

black-box neural learners in Algorithm 1. After we obtain the teaching examples $\tilde{\boldsymbol{x}}_j^i$ with greedy DHT, we replace the original $\boldsymbol{x}_j^i$ with $\tilde{\boldsymbol{x}}_j^i$ during training. This is essentially to add a perturbation to the original feature and the backward gradients to update the network will be affected. In practice, the computational overhead is reasonably small as long as we use a small number of steps to estimate $\boldsymbol{w}^*$ in each iteration. Instead of using a ϵ-neighborhood of the original feature in the Euclidean space, we use a ϵ-neighborhood on the hypersphere with the radius being the norm of the original feature in Equation 8. This is inspired by the observation in [77, 78, 79, 80, 81, 82] that angular distance in the representation space tends to model semantic difference.

Performing DHT iteratively in the representation space can provide additional information for the last-layer classifiers, leading to better convergence of the last-layer classifiers. This will in turn improve the backward gradients of the loss *w.r.t.* the representation which is responsible for updating the network encoder. From an optimization perspective, performative teaching shares similar spirits to Lookahead optimizer [83] in the sense that both methods use information about future steps, which can be viewed as an approximate form of $\boldsymbol{w}^*$. With the surrogate knowledge of $\boldsymbol{w}^*$ for the last-layer classifier, DHT implicitly encodes more information about the loss landscape and may help the neural encoder to avoid some poor local minima.

5 Experiments and Results

We evaluate DHT on several widely used image classification datasets: for white-box and black-box teaching in the logistic regression, we test our policies on synthetic half-moon and MNIST; for black-box teaching in deep neural networks, we test our methods on MNIST, CIFAR-10, and

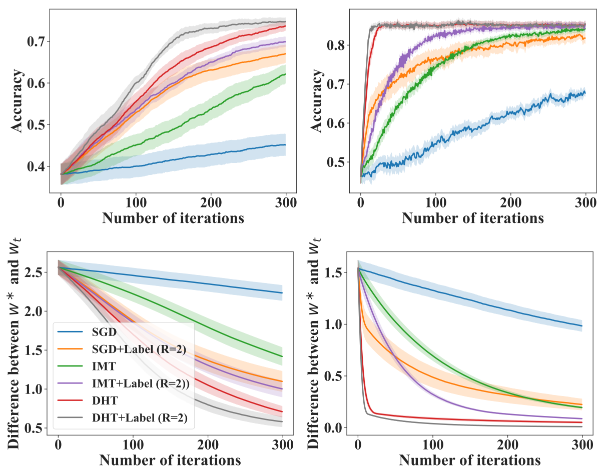

Figure 3: Omniscient teaching with or without label synthesis [14]. Convergence comparison between our greedy teaching policy with several other baseline methods. Top: half-moon. Bottom: MNIST.

CIFAR-100. Full experiment details and additional experimental results can be found in Appendix.

5.1 Omniscient Teaching

In the omniscient teaching scenario, we seek to optimize the convergence speed to a target classifier w^*, and because the target classifier exhibits good classification performance, we also measure the convergence of the testing accuracy. We initialize all the models with the same architecture and model weights. All experiments are repeated ten times with different seeds. We compare random teacher (*i.e.*, SGD), samples selected by IMT, and samples generated by DHT. We update the student model with a standard SGD optimizer, a learning rate of 0.001 and compare the convergence behavior in the first 300 iterations.

Greedy teaching policy. For greedy teaching, we additionally optimize the labels with [14], *i.e.*, after one sample has been selected or generated, we further optimize the corresponding label by fixing the sample. The results are summarized in Figure 3. The baseline methods SGD+Label and IMT+Label correspond to variants of the LAST framework [14]. For samples generated by DHT, we constrain the value of each dimension to be within the minimum and maximum of the original dataset; for the optimized one-hot labels, we constraint its values to be positive $y_i > 0$ and the magnitude to satisfy $\|\tilde{\boldsymbol{y}}\|_2 \leq 2$. Constraining the magnitude to 2 gives the teacher more flexibility when generating labels than the original one-hot label space. We observe that with or without label synthesis, greedy DHT achieves the fastest convergence and outperforms both SGD and IMT.

Parameterized teaching policy. The results of data transformation policy are given in Figure 4. We generally observe parameterized policy exhibits faster convergence than greedy policy, and DHT again outperforms both SGD and IMT by a significant margin. It is worth mentioning that we do not impose any constraint on the output space when syn-

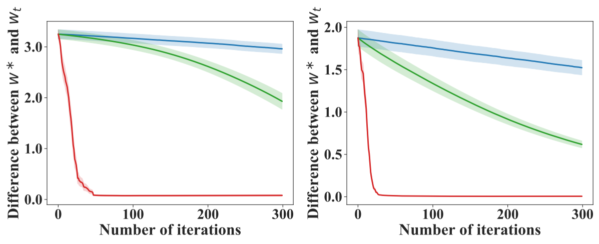

Figure 4: Convergence of data transformation policy. Left: binary classification on half-moon. Right: 3/5 classification on MNIST.

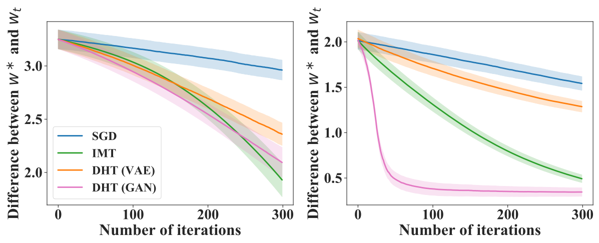

Figure 5: Convergence of GAN-based and VAE-based generative modeling policy. Left: half-moon. Right: MNIST.

thesizing samples, resulting in out-of-distribution samples that are not semantically interpretable for humans (*e.g.*, on MNIST it appears to be random noise images).

Next, we discuss the difference between GAN-based and VAE-based teaching policies. The weight convergence is given in Figure 5 and some generated samples are exemplified below in Figure 6 and Figure 7. The GAN-based teacher is able to outperform the VAE-based teacher in both experiments. In general, we observe that by synthesizing samples directly in the image space, the teacher has more freedom in conveying information, causing the student to learn faster. However, this also means that the data synthesized by a GAN-based teacher might appear to be visually very different from the original data set. A VAE-based teacher can synthesize samples that appear much closer to the original data distribution at the cost of teaching performance. We observe in Figure 6 and Figure 7 that the synthesized samples are nearly differentiable from the original data set. One interesting observation is that the GAN-based teacher implicitly learns a meaningful teaching policy. First, samples are synthesized that have certain stylistic characteristics of the original MNIST dataset but do not contain any meaningful semantic information. During this stage, the student can efficiently converge to w^*; afterward, the GAN-based teacher generates samples that look more similar to the original dataset, and we can interpret it as a fine-tuning process. The evolution of the generated samples during the teaching process can be seen in Figure 7.

5.2 Black-box Teaching

Parameterized teaching. We start with the black-box version of parameterized teaching. We use the same teaching objective as the parameterized teaching in the omniscient setting and remove any information about the target classifier w^*, *i.e.*, by relying solely on the empirical classification loss on the validation set. Surprisingly, when teaching a linear logistic regression learner, the knowledge about the target classifier is not indispensable to achieve fast convergence in terms of the test accuracy (see Figure 8).

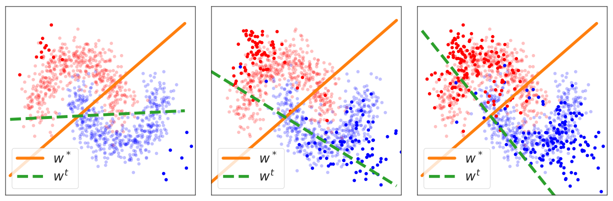

Figure 6: Visualization of the data synthesized by a VAE-based teacher after iteration 20, 150 and 280. The orange line indicates the target classifier w^*; the green dashed line indicates the student classifier. Different colors indicate different classes; points with lower opacity represent the ground truth data distribution.

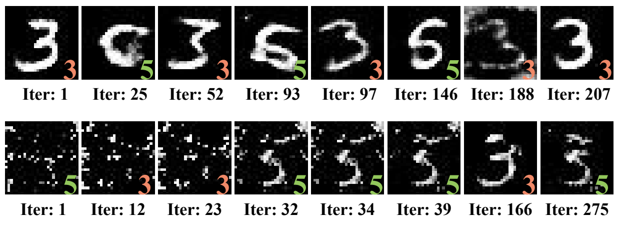

Figure 7: MNIST samples synthesized by the DHT teacher conditioned on GT labels (3/5) during the training process. Upper row: samples synthesized by the VAE-based teacher. Lower row: samples synthesized by the GAN-based teacher.

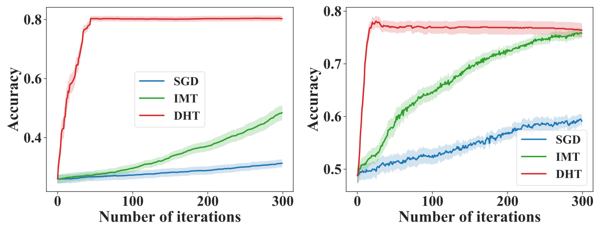

Figure 8: Convergence of black-box parameterized teaching policy with the full teaching space. Left: half-moon. Right: MNIST.

Mixup-based teaching. We empirically evaluate our mixup-based teaching with unrolling and policy gradient in Table 1. The learner is a simple CNN model optimized by a standard Adam optimizer on CIFAR-10 without data augmentation for 50 epochs. We compare our Mixup-based teaching policy with standard empirical risk minimization (ERM), ERM with continuous mixup data augmentation (*i.e.*, interpolation coefficient sampled from a beta distribution, denoted as cMixup), and ERM with discrete mixup data augmentation (*i.e.*, mixing coefficient discretized to $\{0, 0.5, 1\}$, denoted as dMixup). With policy gradient, we observe that our DHT teacher is able to find a

Method	Accuracy (%)
ERM	61.75
cMixup	66.27
dMixup	65.80
Unrolling	65.18
Policy gradient	**67.64**

Table 1: Results on CIFAR-10.

Dataset	Learner	SGD	Random Policy	DHT
MNIST	MLP	92.45 ± 0.07	92.47 ± 0.06	$\mathbf{95.02 \pm 0.04}$
CIFAR-10	CNN-3	87.30 ± 0.28	87.17 ± 0.17	$\mathbf{88.77 \pm 0.35}$
	CNN-6	90.34 ± 0.10	90.20 ± 0.09	$\mathbf{91.61 \pm 0.23}$
	CNN-9	91.10 ± 0.26	91.12 ± 0.12	$\mathbf{92.30 \pm 0.13}$
	CNN-15	91.85 ± 0.28	91.67 ± 0.13	$\mathbf{92.44 \pm 0.15}$
CIFAR-100	CNN-3	62.10 ± 0.29	62.04 ± 0.11	$\mathbf{62.69 \pm 0.37}$
	CNN-6	65.02 ± 0.24	64.96 ± 0.17	$\mathbf{66.81 \pm 0.17}$
	CNN-9	67.05 ± 0.29	67.19 ± 0.23	$\mathbf{69.23 \pm 0.34}$
	CNN-15	68.39 ± 0.39	68.49 ± 0.17	$\mathbf{68.96 \pm 0.36}$

Table 2: Testing accuracy (%) of performative teaching. Multiple types of neural learners (*e.g.*, MLP and CNN) are considered.

policy that greatly facilitates the convergence of the student model and outperforms the other baselines. We characterize the model states with model features that are obtained through querying the student model, similar to [14]. We also notice that unrolling does not perform as well as the baselines, and we suspect that this is because the space of Mixup coefficients is highly non-smooth. There exist many poor local minima that prevent the unrolling approach from finding a good solution to the bi-level optimization.

Performative teaching for neural learners. We comprehensively evaluate the performance of the performative teaching by conducting image classification experiments with similar architectures and settings as [80]. For CIFAR-10 and CIFAR-100, we start the training with the learning rate of 0.1 and divide it by 10 at iteration 20k, 30k and 37.5k. The training stops at iteration 42.5k. For MNIST, we start the training with a learning rate of 0.001 and train for 39k iterations. We use a standard SGD optimizer with weight decay. The batch size is set to 128 and only basic data augmentation is performed. Multiple network architectures are used to serve as the learner and the specific architectures are given in Appendix. Results are given in Table 2.

We compare our performative teaching with two other baselines. The first one is vanilla SGD optimization, where no teaching takes place during the training. This is to demonstrate the clean performance gain obtained by performative teaching. From Table 2, we can clearly see that DHT consistently outperforms vanilla SGD by a considerable margin. To further verify whether DHT indeed teaches useful information or not, we construct a random policy in the exactly same action space of the performative teacher, *i.e.*, we omit the teaching process, but instead uniformly sample a new point on the same ϵ-neighborhood on the hypersphere of the representation space. The only difference between random policy and performative teaching is how we generate $\tilde{\boldsymbol{x}}$, and we note that the space to generate $\tilde{\boldsymbol{x}}$ is the same for both. This comparison shows that the performance gain does not result from implicit data augmentation in the representation space, as can be seen from Table 2 that a random policy does not have a noticeable effect on the final performance. All the results in Table 2 are averaged over 5 runs and the standard deviations of accuracy is also given to make sure that the performance gain is not due to randomness. The performance gain of performative teaching is evident across

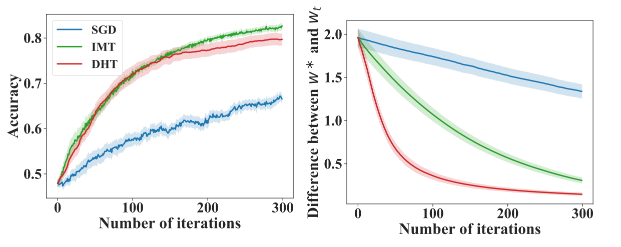

Figure 9: Convergence of privacy-preserving teaching on MNIST.

all datasets and all different neural learners. We emphasize that we do not have $\boldsymbol{w}^*$ for the neural networks, so performative teaching can be used to teach any neural network on any dataset. We only consider a simple greedy DHT in performative teaching, and the teaching performance could be further improved with an advanced teaching algorithm.

5.3 Privacy-preserving Teaching via constrained DHT

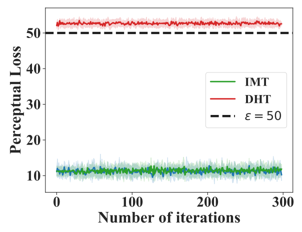

Figure 10: Private perceptual distance of the synthesized samples during teaching. ϵ is a prescribed distance threshold.

In practical applications, generating samples that are semantically distinct from the original data distribution could be beneficial. For example, in the medical domain, we wish not to reveal sensitive information that might contain in the original dataset. As a proof of concept, we regard privacy preservation as some distance constraints on the feature space, *i.e.*, the generated samples are at least ϵ-away (in a semantic latent space) from a pre-defined privacy set. The teaching objective is

$$\begin{aligned} \min_{\boldsymbol{\theta}} \ & \|\boldsymbol{w}^v(\boldsymbol{\theta}) - \boldsymbol{w}^*\|_2^2 + \sum_{t=1}^{v} \ell(\boldsymbol{\pi}_{\boldsymbol{\theta}}, \boldsymbol{y}) | \boldsymbol{w}^t) \\ & \quad + \max\{0, \epsilon - \|\phi(\boldsymbol{\pi}_{\boldsymbol{\theta}}) - \phi(\boldsymbol{x})\|_2^2\} \\ \text{s.t. } \ & \|\phi(\boldsymbol{\pi}_{\boldsymbol{\theta}}) - \phi(\boldsymbol{x})\|_2^2 \leq \epsilon \end{aligned} \tag{10}$$

where $\phi(\cdot)$ is a pretrained neural network for computing perceptual distance. Here, we demonstrate that it is possible to achieve similar teaching performance by only synthesizing samples that satisfy the privacy constraints (see Figure 9). We also show the private perceptual distance during the teaching in Figure 10 in which the private perceptual distance is defined as the minimal distance between the samples in the privacy set and the synthesized sample.

6 Concluding Remarks

In this paper, we introduce a novel data hallucination teaching framework and demonstrate, both theoretically and empirically, that DHT achieves promising teaching performance in both omniscient and black-box settings. We also highlight that a novel performative teaching formulation is proposed for teaching black-box neural learners. Experiments show that DHT is able to achieve significant performance gains when teaching black-box neural learners.

References

[1] X. Zhu, "Machine teaching: An inverse problem to machine learning and an approach toward optimal education.," in *AAAI*, 2015. 1, 2

[2] X. Zhu, A. Singla, S. Zilles, and A. N. Rafferty, "An overview of machine teaching," *arXiv preprint arXiv:1801.05927*, 2018. 1

[3] A. Singla, I. Bogunovic, G. Bartok, A. Karbasi, and A. Krause, "Near-optimally teaching the crowd to classify.," in *ICML*, 2014. 1, 2

[4] A. Singla, I. Bogunovic, G. Bartók, A. Karbasi, and A. Krause, "On actively teaching the crowd to classify," in *NeurIPS Workshop on Data Driven Education*, 2013. 1

[5] Y. Zhou, A. R. Nelakurthi, and J. He, "Unlearn what you have learned: Adaptive crowd teaching with exponentially decayed memory learners," in *KDD*, 2018. 1, 2

[6] Y. Zhou, A. R. Nelakurthi, R. Maciejewski, W. Fan, and J. He, "Crowd teaching with imperfect labels," in *WWW*, 2020. 1, 2

[7] S. Alfeld, X. Zhu, and P. Barford, "Data poisoning attacks against autoregressive models.," in *AAAI*, 2016. 1, 2

[8] S. Alfeld, X. Zhu, and P. Barford, "Explicit defense actions against test-set attacks," in *AAAI*, 2017. 1

[9] A. Rakhsha, G. Radanovic, R. Devidze, X. Zhu, and A. Singla, "Policy teaching via environment poisoning: Training-time adversarial attacks against reinforcement learning," in *ICML*, 2020. 1, 2

[10] Y. Ma, X. Zhang, W. Sun, and J. Zhu, "Policy poisoning in batch reinforcement learning and control," in *NeurIPS*, 2019. 1

[11] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *ICML*, 2009. 1

[12] T. Wang, J.-Y. Zhu, A. Torralba, and A. A. Efros, "Dataset distillation," *arXiv preprint arXiv:1811.10959*, 2018. 1, 2

[13] W. Liu, B. Dai, A. Humayun, C. Tay, C. Yu, L. B. Smith, J. M. Rehg, and L. Song, "Iterative machine teaching," in *ICML*, 2017. 1, 2, 3, 4

[14] W. Liu, Z. Liu, H. Wang, L. Paull, B. Schölkopf, and A. Weller, "Iterative teaching by label synthesis," in *NeurIPS*, 2021. 1, 2, 3, 4, 6, 8, 16

[15] G. Cazenavette, T. Wang, A. Torralba, A. A. Efros, and J.-Y. Zhu, "Dataset distillation by matching training trajectories," in *CVPR Workshops*, 2022. 1, 2

[16] B. Zhao, K. R. Mopuri, and H. Bilen, "Dataset condensation with gradient matching.," in *ICLR*, 2021. 1, 2

[17] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017. 2, 5

[18] V. Verma, A. Lamb, C. Beckham, A. Najafi, I. Mitliagkas, D. Lopez-Paz, and Y. Bengio, "Manifold mixup: Better representations by interpolating hidden states," in *ICML*, 2019. 2

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017. 2

[20] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," *arXiv preprint arXiv:1712.04621*, 2017. 2

[21] T. Dao, A. Gu, A. Ratner, V. Smith, C. De Sa, and C. Ré, "A kernel theory of modern data augmentation," in *ICML*, 2019. 2

[22] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *ICML*, 2020. 2

[23] J. Perdomo, T. Zrnic, C. Mendler-Dünner, and M. Hardt, "Performative prediction," in *ICML*, 2020. 2, 5

[24] X. Zhu, "Machine teaching for bayesian learners in the exponential family," in *NeurIPS*, 2013. 2

[25] J. Liu, X. Zhu, and H. G. Ohannessian, "The teaching dimension of linear learners," in *ICML*, 2016. 2

[26] F. Mansouri, Y. Chen, A. Vartanian, J. Zhu, and A. Singla, "Preference-based batch and sequential teaching: Towards a unified view of models," in *NeurIPS*, 2019. 2

[27] Y. Chen, A. Singla, O. Mac Aodha, P. Perona, and Y. Yue, "Understanding the role of adaptivity in machine teaching: The case of version space learners," in *NeurIPS*, 2018. 2

[28] B. Tabibian, U. Upadhyay, A. De, A. Zarezade, B. Schölkopf, and M. Gomez-Rodriguez, "Enhancing human learning via spaced repetition optimization," *Proceedings of the National Academy of Sciences*, 2019. 2

[29] A. Kumar, H. Zhang, A. Singla, and Y. Chen, "The teaching dimension of kernel perceptron," *arXiv preprint arXiv:2010.14043*, 2020. 2

[30] X. Zhang, S. K. Bharti, Y. Ma, A. Singla, and X. Zhu, "The teaching dimension of q-learning," *arXiv preprint arXiv:2006.09324*, 2020. 2

[31] C. Wang, A. Singla, and Y. Chen, "Teaching an active learner with contrastive examples," in *NeurIPS*, 2021. 2

[32] T. Peltola, M. M. Çelikok, P. Daee, and S. Kaski, "Machine teaching of active sequential learners," in *NeurIPS*, 2019. 2

[33] L. Yuan, D. Zhou, J. Shen, J. Gao, J. L. Chen, Q. Gu, Y. N. Wu, and S.-C. Zhu, "Iterative teacher-aware learning," in *NeurIPS*, 2021. 2

[34] A. Hunziker, Y. Chen, O. Mac Aodha, M. G. Rodriguez, A. Krause, P. Perona, Y. Yue, and A. Singla, "Teaching multiple concepts to a forgetful learner," in *NeurIPS*, 2019. 2

[35] W. Liu, B. Dai, X. Li, Z. Liu, J. Rehg, and L. Song, "Towards black-box iterative machine teaching," in *ICML*, 2018. 2, 4

[36] L. Lessard, X. Zhang, and X. Zhu, "An optimal control approach to sequential machine teaching," in *AISTATS*, 2019. 2

[37] Z. Xu, B. Chen, C. Li, W. Liu, L. Song, Y. Lin, and A. Shrivastava, "Locality sensitive teaching," in *NeurIPS*, 2021. 2

[38] S. Tschiatschek, A. Ghosh, L. Haug, R. Devidze, and A. Singla, "Learner-aware teaching: Inverse reinforcement learning with preferences and constraints," in *NeurIPS*, 2019. 2

[39] P. Kamalaruban, R. Devidze, V. Cevher, and A. Singla, "Interactive teaching algorithms for inverse reinforcement learning," *arXiv preprint arXiv:1905.11867*, 2019. 2

[40] L. Haug, S. Tschiatschek, and A. Singla, "Teaching inverse reinforcement learners via features and demonstrations," in *NeurIPS*, 2018. 2

[41] E. Johns, O. Mac Aodha, and G. J. Brostow, "Becoming the expert-interactive multi-class machine teaching," in *CVPR*, 2015. 2

[42] Y. Chen, O. Mac Aodha, S. Su, P. Perona, and Y. Yue, "Near-optimal machine teaching via explanatory teaching sets," in *AISTATS*, 2018. 2

[43] O. Mac Aodha, S. Su, Y. Chen, P. Perona, and Y. Yue, "Teaching categories to human learners with visual explanations," in *CVPR*, 2018. 2

[44] S. Mei and X. Zhu, "Using machine teaching to identify optimal training-set attacks on machine learners.," in *AAAI*, 2015. 2

[45] X. Zhang, X. Zhu, and S. Wright, "Training set debugging using trusted items," in *AAAI*, 2018. 2

[46] X. Zhang, Y. Ma, A. Singla, and X. Zhu, "Adaptive reward-poisoning attacks against reinforcement learning," in *ICML*, 2020. 2

[47] X. Zhang, X. Zhu, and L. Lessard, "Online data poisoning attacks," in *L4DC*, 2020. 2

[48] P. Wang, J. Wang, P. Paranamana, and P. Shafto, "A mathematical theory of cooperative communication," in *NeurIPS*, 2020. 2

[49] P. Shafto, J. Wang, and P. Wang, "Cooperative communication as belief transport," *Trends in Cognitive Sciences*, 2021. 2

[50] J. Wang, P. Wang, and P. Shafto, "Sequential cooperative bayesian inference," in *ICML*, 2020. 2

[51] J. Wei and K. Zou, "Eda: Easy data augmentation techniques for boosting performance on text classification tasks," in *EMNLP*, 2019. 2

[52] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," in *Interspeech*, 2019. 2

[53] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning (still) requires rethinking generalization," *Communications of the ACM*, vol. 64, no. 3, pp. 107–115, 2021. 2

[54] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. 2

[55] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015. 2

[56] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *CVPR*, 2020. 2

[57] I. W. Tsang, J. T. Kwok, P.-M. Cheung, and N. Cristianini, "Core vector machines: Fast svm training on very large data sets.," *Journal of Machine Learning Research*, vol. 6, no. 4, 2005. 2

[58] T. Campbell and T. Broderick, "Bayesian coreset construction via greedy iterative geodesic ascent," in *ICML*, 2018. 2

[59] O. Sener and S. Savarese, “Active learning for convolutional neural networks: A core-set approach,” in *ICLR*, 2018. 2

[60] B. Mirzasoleiman, J. Bilmes, and J. Leskovec, “Coresets for data-efficient training of machine learning models,” in *ICML*, 2020. 2

[61] A. Angelova, Y. Abu-Mostafam, and P. Perona, “Pruning training sets for learning of object categories,” in *CVPR*, 2005. 2

[62] A. Lapedriza, H. Pirsiavash, Z. Bylinskii, and A. Torralba, “Are all training examples equally valuable?,” *arXiv preprint arXiv:1311.6510*, 2013. 2

[63] C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” in *ICML*, 2017. 3, 5

[64] P. J. Werbos, “Generalization of backpropagation with application to a recurrent gas market model,” *Neural networks*, 1988. 3

[65] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks,” *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020. 4

[66] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *arXiv preprint arXiv:1312.6114*, 2013. 4

[67] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” *arXiv preprint arXiv:1503.02531*, 2015. 4

[68] B. Zoph and Q. V. Le, “Neural architecture search with reinforcement learning,” *arXiv preprint arXiv:1611.01578*, 2016. 5

[69] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, “Progressive neural architecture search,” in *ECCV*, 2018. 5

[70] H. Liu, K. Simonyan, and Y. Yang, “Darts: Differentiable architecture search,” *arXiv preprint arXiv:1806.09055*, 2018. 5, 15

[71] M. Andrychowicz, M. Denil, S. G. Colmenarejo, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas, “Learning to learn by gradient descent by gradient descent,” in *NeurIPS*, 2016. 5

[72] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, “Autoaugment: Learning augmentation strategies from data,” in *CVPR*, 2019. 5

[73] R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” *Machine learning*, 1992. 5

[74] K. Healy, “The performativity of networks,” *European Journal of Sociology/Archives Européennes de Sociologie*, vol. 56, no. 2, pp. 175–205, 2015. 5

[75] D. A. MacKenzie, F. Muniesa, L. Siu, *et al.*, *Do economists make markets?: on the performativity of economics*. Princeton University Press, 2007. 5

[76] M. Hardt, M. Jagadeesan, and C. Mendler-Dünner, “Performative power,” *arXiv preprint arXiv:2203.17232*, 2022. 5

[77] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, “Sphereface: Deep hypersphere embedding for face recognition,” in *CVPR*, 2017. 6

[78] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu, “Cosface: Large margin cosine loss for deep face recognition,” in *CVPR*, 2018. 6

[79] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, “Arcface: Additive angular margin loss for deep face recognition,” in *CVPR*, 2019. 6

[80] W. Liu, R. Lin, Z. Liu, L. Liu, Z. Yu, B. Dai, and L. Song, “Learning towards minimum hyperspherical energy,” in *NeurIPS*, 2018. 6, 8, 15

[81] W. Liu, Z. Liu, Z. Yu, B. Dai, R. Lin, Y. Wang, J. M. Rehg, and L. Song, “Decoupled networks,” in *CVPR*, 2018. 6

[82] B. Chen, W. Liu, Z. Yu, J. Kautz, A. Shrivastava, A. Garg, and A. Anandkumar, “Angular visual hardness,” in *ICML*, 2020. 6

[83] M. Zhang, J. Lucas, J. Ba, and G. E. Hinton, “Lookahead optimizer: k steps forward, 1 step back,” in *NeurIPS*, 2019. 6

[84] G.-H. Lin and M. Fukushima, “Some exact penalty results for nonlinear programs and mathematical programs with equilibrium constraints,” *Journal of Optimization Theory and Applications*, 2003. 13

[85] S. Vaswani, A. Mishkin, I. Laradji, M. Schmidt, G. Gidel, and S. Lacoste-Julien, “Painless stochastic gradient: Interpolation, line-search, and convergence rates,” in *NeurIPS*, 2019. 14

Appendix

Table of Contents

A Proof of Theorem 1

From the $(t+1)$-th gradient update with the greedy DHT teacher, we have that

$$\begin{aligned}\left\|\boldsymbol{w}^{t+1}-\boldsymbol{w}^*\right\|^2 &= \left\|\boldsymbol{w}^t-\eta_t\nabla_{\boldsymbol{w}^t}\ell(\tilde{\boldsymbol{x}}_{i_t},y_{i_t}|\boldsymbol{w}^t)-\boldsymbol{w}^*\right\|^2\\ &= \left\|\boldsymbol{w}^t-\eta_t g(\tilde{\boldsymbol{x}}_{i_t})\cdot\mathcal{T}_{\boldsymbol{x}\rightarrow\tilde{\boldsymbol{x}}}\circ\nabla_{\boldsymbol{w}^t}\ell(\boldsymbol{x}_{i_t},y_{i_t}|\boldsymbol{w}^t)-\boldsymbol{w}^*\right\|^2\\ &= \Bigg\|\boldsymbol{w}^t-\eta_t \underbrace{g(\tilde{\boldsymbol{x}}_{i_t})\cdot\beta}_{\overset{\text{def}}{=}g_s(\tilde{\boldsymbol{x}}_{i_t})}\cdot\nabla_{\boldsymbol{w}^t}\ell(\boldsymbol{x}_{i_t},y_{i_t}|\boldsymbol{w}^t)-\boldsymbol{w}^*\Bigg\|^2\end{aligned} \tag{11}$$

where $\tilde{\boldsymbol{x}}$ is the data generated by DHT and i_t denotes a randomly sampled index from the pool in the t-th iteration. Because $\mathcal{T}_{\boldsymbol{x}\rightarrow\tilde{\boldsymbol{x}}}$ is a scaling mapping, we have that $g_s(\tilde{\boldsymbol{x}})$ is generally defined as

$$g_s(\tilde{\boldsymbol{x}})=\beta\cdot g(\tilde{\boldsymbol{x}})=\beta\cdot\frac{\|\nabla_{\boldsymbol{w}}\ell(\boldsymbol{x},y|\boldsymbol{w})\|}{\|\nabla_{\boldsymbol{w}}\ell(\tilde{\boldsymbol{x}},y|\boldsymbol{w})\|}\cdot\frac{\|\tilde{\boldsymbol{x}}\|}{\|\boldsymbol{x}\|}, \tag{12}$$

which, for different linear learners, can be instantiated as

$$\begin{aligned}\text{LSR learner: }\; g_s(\tilde{\boldsymbol{x}}) &= \frac{\beta\langle\boldsymbol{w},\boldsymbol{x}\rangle-\beta\cdot y}{\beta\langle\boldsymbol{w},\boldsymbol{x}\rangle-y}\\ \text{LR learner: }\; g_s(\tilde{\boldsymbol{x}}) &= \frac{\beta+\beta\cdot\exp(\beta\cdot y\langle\boldsymbol{w},\boldsymbol{x}\rangle)}{1+\exp(y\langle\boldsymbol{w},\boldsymbol{x}\rangle)}\end{aligned} \tag{13}$$

which can be controlled by adjusting the value of β. β can be dependent on $\boldsymbol{w}$, so it can be different in different iterations. Intuitively, since we can adjust β to equivalently adjust the learning rate, we can therefore provably have a better convergence rate. Concretely, we have that

$$\begin{aligned}\left\|\boldsymbol{w}^{t+1}-\boldsymbol{w}^*\right\|^2 = \left\|\boldsymbol{w}^t-\boldsymbol{w}^*\right\|^2 &- 2\eta_t g_s(\tilde{\boldsymbol{x}}_{i_t})\langle\nabla_{\boldsymbol{w}^t}\ell(\boldsymbol{x}_{i_t},y_{i_t}|\boldsymbol{w}^t),\boldsymbol{w}^t-\boldsymbol{w}^*\rangle\\ &+\eta_t^2(g_s(\tilde{\boldsymbol{x}}_{i_t}))^2\left\|\nabla_{\boldsymbol{w}^t}\ell(\boldsymbol{x}_{i_t},y_{i_t}|\boldsymbol{w}^t)\right\|^2\end{aligned} \tag{14}$$

which can be simplified as (by denoting $\nabla_{\boldsymbol{w}^t}\ell(\boldsymbol{x}_{i_t},y_{i_t}|\boldsymbol{w}^t)$ as $\nabla\ell_{i_t}(\boldsymbol{w}^t)$):

$$\left\|\boldsymbol{w}^{t+1}-\boldsymbol{w}^*\right\|^2=\left\|\boldsymbol{w}^t-\boldsymbol{w}^*\right\|^2-2\eta_t g_s(\tilde{\boldsymbol{x}}_{i_t})\langle\nabla\ell_{i_t}(\boldsymbol{w}^t),\boldsymbol{w}^t-\boldsymbol{w}^*\rangle+\eta_t^2(g_s(\tilde{\boldsymbol{x}}_{i_t}))^2\left\|\nabla\ell_{i_t}(\boldsymbol{w}^t)\right\|^2. \tag{15}$$

Because we know that the synthesized data $\boldsymbol{x}_{i_t}$ generated by the greedy DHT policy is the solution to the following minimization:

$$\tilde{\boldsymbol{x}}_{i_t}=\arg\min_{\boldsymbol{x}'_{i_t}}\left\{\eta_t^2(g_s(\boldsymbol{x}'_{i_t}))^2\left\|\nabla_{\boldsymbol{w}^t}\ell(\boldsymbol{x}_{i_t},y_{i_t}|\boldsymbol{w}^t)\right\|^2-2\eta_t g_s(\boldsymbol{x}'_{i_t})\langle\nabla_{\boldsymbol{w}^t}\ell(\boldsymbol{x}_{i_t},y_{i_t}|\boldsymbol{w}^t),\boldsymbol{w}^t-\boldsymbol{w}^*\rangle\right\}, \tag{16}$$

then we plug a new $\tilde{\boldsymbol{x}}''_{i_t}$ which satisfies $g_s(\tilde{\boldsymbol{x}}''_{i_t})=c_1\left\|\boldsymbol{w}^t-\boldsymbol{w}^*\right\|$ to Eq. (15) and have the following inequality:

$$\begin{aligned}\left\|\boldsymbol{w}^{t+1}-\boldsymbol{w}^*\right\|^2 &\leq \left\|\boldsymbol{w}^t-\boldsymbol{w}^*\right\|^2-2\eta_t g_s(\tilde{\boldsymbol{x}}''_{i_t})\langle\nabla\ell_{i_t}(\boldsymbol{w}^t),\boldsymbol{w}^t-\boldsymbol{w}^*\rangle+\eta_t^2(g_s(\tilde{\boldsymbol{x}}''_{i_t}))^2\left\|\nabla\ell_{i_t}(\boldsymbol{w}^t)\right\|^2\\ &= \left\|\boldsymbol{w}^t-\boldsymbol{w}^*\right\|^2-2\eta_t c_1\left\|\boldsymbol{w}^t-\boldsymbol{w}^*\right\|\langle\nabla\ell_{i_t}(\boldsymbol{w}^t),\boldsymbol{w}^t-\boldsymbol{w}^*\rangle\\ &\quad+\eta_t^2c_1^2\left\|\nabla\ell_{i_t}(\boldsymbol{w}^t)\right\|^2\left\|\boldsymbol{w}^t-\boldsymbol{w}^*\right\|^2\end{aligned} \tag{17}$$

which holds because $\tilde{\boldsymbol{x}}_{i_t}$ leads to the minimal $\left\|\boldsymbol{w}^{t+1}-\boldsymbol{w}^*\right\|^2$ and $\tilde{\boldsymbol{x}}''_{i_t}$ has to result in a larger or equal $\left\|\boldsymbol{w}^{t+1}-\boldsymbol{w}^*\right\|^2$.

Next, we apply the convexity of $f(\cdot)$ and the order-1 strong convexity [84] of $\ell_{i_t}(\cdot)$, and therefore have that (let $\mu_{i_t}=0$ when ℓ_{i_t} is not order-1 strongly convex):

$$-\langle\nabla\ell_{i_t}(\boldsymbol{w}^t),\boldsymbol{w}^t-\boldsymbol{w}^*\rangle\leq\ell_{i_t}(\boldsymbol{w}^*)-\ell_{i_t}(\boldsymbol{w}^t)-\frac{\mu_{i_t}}{2}\left\|\boldsymbol{w}^t-\boldsymbol{w}^*\right\| \tag{18}$$

which results in

$$\begin{aligned}\left\|\boldsymbol{w}^{t+1}-\boldsymbol{w}^*\right\|^2 &\leq \left\|\boldsymbol{w}^t-\boldsymbol{w}^*\right\|^2+2\eta_t c_1\left\|\boldsymbol{w}^t-\boldsymbol{w}^*\right\|\left(\ell_{i_t}(\boldsymbol{w}^*)-\ell_{i_t}(\boldsymbol{w}^t)-\frac{\mu_{i_t}}{2}\left\|\boldsymbol{w}^t-\boldsymbol{w}^*\right\|\right)\\ &\quad+c_1^2\left\|\nabla\ell_{i_t}(\boldsymbol{w}^t)\right\|^2\left\|\boldsymbol{w}^t-\boldsymbol{w}^*\right\|^2\\ &= \left\|\boldsymbol{w}^t-\boldsymbol{w}^*\right\|^2+2\eta_t c_1\left\|\boldsymbol{w}^t-\boldsymbol{w}^*\right\|(\ell_{i_t}(\boldsymbol{w}^*)-\ell_{i_t}(\boldsymbol{w}^t))-\eta_t c_1\mu_{i_t}\left\|\boldsymbol{w}^t-\boldsymbol{w}^*\right\|^2\\ &\quad+c_1^2\left\|\nabla\ell_{i_t}(\boldsymbol{w}^t)\right\|^2\left\|\boldsymbol{w}^t-\boldsymbol{w}^*\right\|^2.\end{aligned}$$

Considering the condition that ℓ_i is L_i-Lipschitz continuous and denoting $L_{\max} = \max_i L_i$, we then have

$$\begin{aligned}\left\|\boldsymbol{w}^{t+1}-\boldsymbol{w}^*\right\|^2 \leq \left\|\boldsymbol{w}^{t}-\boldsymbol{w}^*\right\|^2 + 2\eta_t c_1 \left\|\boldsymbol{w}^{t}-\boldsymbol{w}^*\right\| (\ell_{i_t}(\boldsymbol{w}^*) \\ - \ell_{i_t}(\boldsymbol{w}^t)) - \eta_t c_1 \mu_{i_t} \left\|\boldsymbol{w}^{t}-\boldsymbol{w}^*\right\|^2 + \eta_t^2 c_1^2 L_{\max}^2 \left\|\boldsymbol{w}^{t}-\boldsymbol{w}^*\right\|^2.\end{aligned}$$

The interpolation condition [85] indicates that $\boldsymbol{w}^*$ is the minimum for all functions ℓ_i, which is equivalent to $\ell_i(\boldsymbol{w}^*) \leq \ell_i(\boldsymbol{w}^t)$ for all i. Therefore, we have that $(\ell_{i_t}(\boldsymbol{w}^*) - \ell_{i_t}(\boldsymbol{w}^t)) \leq 0$. Finally we arrive at

$$\begin{aligned}\left\|\boldsymbol{w}^{t+1}-\boldsymbol{w}^*\right\|^2 &\leq \left\|\boldsymbol{w}^{t}-\boldsymbol{w}^*\right\|^2 - \eta_t c_1 \mu_{i_t} \left\|\boldsymbol{w}^{t}-\boldsymbol{w}^*\right\|^2 + \eta_t^2 c_1^2 L_{\max}^2 \left\|\boldsymbol{w}^{t}-\boldsymbol{w}^*\right\|^2 \\ &= (1-\mu_{i_t}\eta_t c_1 + \eta_t^2 L_{\max}^2 c_1^2) \left\|\boldsymbol{w}^{t}-\boldsymbol{w}^*\right\|^2.\end{aligned} \tag{19}$$

Taking expectation *w.r.t.* i_t, we obtain that

$$\begin{aligned}\mathbb{E}\{\left\|\boldsymbol{w}^{t+1}-\boldsymbol{w}^*\right\|^2\} &\leq \mathbb{E}_{i_t}\{(1-\mu_{i_t}\eta_t c_1 + \eta_t^2 L_{\max}^2 c_1^2) \left\|\boldsymbol{w}^{t}-\boldsymbol{w}^*\right\|^2\} \\ &= (1-\mathbb{E}_{i_t}\{\mu_{i_t}\}\eta_t c_1 + \eta_t^2 L_{\max}^2 c_1^2) \left\|\boldsymbol{w}^{t}-\boldsymbol{w}^*\right\|^2 \\ &= (1-\bar{\mu}\eta_t c_1 + \eta_t^2 L_{\max}^2 c_1^2) \left\|\boldsymbol{w}^{t}-\boldsymbol{w}^*\right\|^2.\end{aligned} \tag{20}$$

Using recursion, we have that

$$\mathbb{E}\{\left\|\boldsymbol{w}^{T}-\boldsymbol{w}^*\right\|^2\} \leq (1-\bar{\mu}\eta_t c_1 + \eta_t^2 c_1^2 L_{\max}^2)^T \left\|\boldsymbol{w}^{0}-\boldsymbol{w}^*\right\|^2 \tag{21}$$

where we usually make $\eta_t c_1$ a constant such that $(1-\bar{\mu}\eta_t c_1 + \eta_t^2 c_1^2 L_{\max}^2)$ also becomes a constant between 0 and 1. This is equivalent to the statement in the theorem that at most $\lceil (\log \frac{1}{1-c_1\eta_t\bar{\mu}+\eta_t^2 c_1^2 L_{\max}})^{-1} \log(\frac{1}{\epsilon}\|\boldsymbol{w}^0 - \boldsymbol{w}^*\|^2) \rceil$ iterations are needed to achieve the ϵ-approximation, namely $\mathbb{E}\{\|\boldsymbol{w}^T - \boldsymbol{w}^*\|^2\} \leq \epsilon$. The proof is concluded. $\square$

B Experimental Details

Experiments on MNIST. For teaching logistic regression learners, we do not use the original MNIST dataset but use a fixed projection matrix $\boldsymbol{P}$ ($\boldsymbol{P} \in \mathbb{R}^{784\times 24}$) to downscale the flattened 784-dimensional MNIST image data to a 24-dimensional feature vector. Experiments are performed on the 24D feature vectors. For visualization, we un-project the 24D feature vectors to the original image shape using the pseudo-inverse matrix $\boldsymbol{P}^{+}$ ($\boldsymbol{P}^{+} \in \mathbb{R}^{24\times 784}$). For teaching a logistic regression learner, we use 1100 (1000/100) images from class 3 and 5. For teaching neural learners with a performative teaching policy, we use the full MNIST dataset without any data augmentation.

Experiments on Half-moon. For half-moon, we use the built-in function from scikit-learn to generate 1000 (800/200) sample points with a Gaussian noise of 0.2.

Performative Teaching. The training schedule has been elaborated on in the main paper. The employed network architectures are described in Table 3. The MLP used for MNIST training has two layers (input dimension - 128 - output dimension). There are in total three hyperparameters: we denote the number of update on $\boldsymbol{w}$ to obtain $\boldsymbol{w}^*$ as n_w, the number of feature update to obtain $\tilde{\boldsymbol{x}}_j^i$ as m and the ϵ-neighbourhood. In our experiments, we use a m of 15, n_w of 5 and ϵ of 0.1.

Layer	CNN-3	CNN-6	CNN-9	CNN-15
Conv1.x	$[3\times 3, 64]\times 1$	$[3\times 3, 64]\times 2$	$[3\times 3, 64]\times 3$	$[3\times 3, 64]\times 5$
Pool1	2×2, Max Pooling, Stride 2			
Conv2.x	$[3\times 3, 128]\times 1$	$[3\times 3, 128]\times 2$	$[3\times 3, 128]\times 3$	$[3\times 3, 128]\times 5$
Pool2	2×2, Max Pooling, Stride 2			
Conv3.x	$[3\times 3, 256]\times 1$	$[3\times 3, 256]\times 2$	$[3\times 3, 256]\times 3$	$[3\times 3, 256]\times 5$
Pool3	2×2, Max Pooling, Stride 2			
Fully Connected	256	256	256	256

Table 3: Network architecture specification of the used CNN models. Note, we use the same CNN network architecture definition as [80] with one additional CNN-3 following the same principle. $[3\times 3, 64]\times 2$ denotes 2 cascaded 2D convolution layers with 64 3×3 filters.

Mixup-based Teaching (Unrolling). The training schedule has been elaborated on in the main paper. The teaching objective of mixup-based teaching is inspired by [70]:

$$\begin{aligned}&\min_{\boldsymbol{v}} \mathcal{L}_{val}(\boldsymbol{w}^*(\boldsymbol{v}), \boldsymbol{v})\\ &\text{s.t.}\quad \boldsymbol{w}^*(\boldsymbol{v}) = \operatorname*{argmin}_{\boldsymbol{w}} \mathcal{L}_{train}(\boldsymbol{w}, \boldsymbol{v})\end{aligned}$$

with $\boldsymbol{w}$ as the student weight and $\boldsymbol{v}$ as the teacher weight. Following [70], we also perform first-order and second-order optimizaton. The result reported in the main paper is obtained by using second-order optimization.

$$\begin{aligned}&\nabla_{\boldsymbol{v}}\mathcal{L}_{val}(\boldsymbol{w}, \boldsymbol{v}) &\qquad& \nabla_{\boldsymbol{v}}\mathcal{L}_{val}(\boldsymbol{w} - \xi\nabla_{\boldsymbol{w}}\mathcal{L}_{train}(\boldsymbol{w}, \boldsymbol{v}), \boldsymbol{v})\\ &\nabla_{\boldsymbol{w}}\mathcal{L}_{train}(\boldsymbol{w}, \boldsymbol{v}) && \nabla_{\boldsymbol{w}}\mathcal{L}_{train}(\boldsymbol{w}, \boldsymbol{v})\end{aligned}$$

Mixup-based Teaching (Policy Gradient). We use a simple MLP with two layers (input dimension (3) - 128 - output dimension (3)) and dropout as policy network. We perform one step of optimization after running two epochs of training.

- **Reward Signal**: $\ell(\boldsymbol{x}_{val}, \boldsymbol{y}_{val}|\boldsymbol{w}^v)$
- **Reward**: $R = r_T(s_{1\ldots T}, a_{1\ldots T})$
- **Objective Function**: $J(\boldsymbol{\theta}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\{R\}$
- **Teacher Update Function**: $\nabla_{\boldsymbol{\theta}}J(\boldsymbol{\theta}) = \sum_t \nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(a_t|r_t)R$
- **State**: model features (obtained through query student model) consists of current iteration, average training loss and best validation loss. The best validation loss is updated every 100 iterations.
- **Action**: discretise the λ into discrete action space $[0, 0.5, 1.0]$

C Additional Experimental Results

The objective of DHT is to reduce the distance between the student model w_t with some desired w^*. Therefore, we only show the convergence of the weight difference in the main paper. w^* is obtained by using a model with the same initial weight as the student model and training it until convergence. We use a learning rate of 0.001 and a standard SGD optimizer with no momentum and weight decay to optimize the logistic regression learner. Since the w^* is associated with good test accuracy on the test dataset, we further show the convergence of test accuracy.

C.1 Teaching Logistic Regression on Half-moon Data with Greedy DHT

We observe similar behavior between the weight convergence and the accuracy convergence of the examined methods using a greedy teaching policy on half-moon data. We use an Adam optimizer with step size of 0.02, factor for average gradient of 0.8 (β_1), factor for average squared gradient of 0.999 (β_2) and update each sample for 300 iterations. We perform early stopping when the loss converges. We constrain the sample value to be within the maximum and minimum values of the original dataset. We use the same optimizer (expect a lower learning rate of 0.001) to optimize the label. We also tested different magnitudes, but the teaching effectiveness degenerates with a smaller magnitude. Note, the original LAST framework [14] does not impose any constraint to achieve good teaching results.

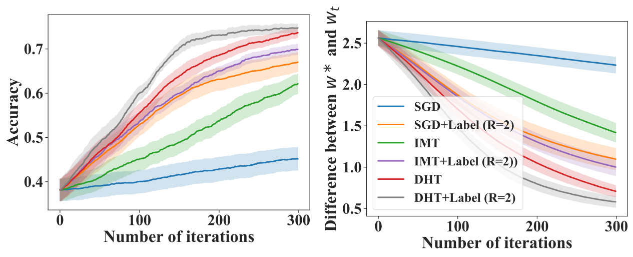

Figure 11: Omniscient teaching with or without label synthesis. Convergence comparison between our greedy teaching policy with several other baseline methods on half-moon.

C.2 Teaching Logistic Regression on MNIST with Greedy DHT

We observe similar behavior between the weight convergence and the accuracy convergence of the examined methods using a greedy teaching policy on MNIST data. We use an AMSGrad optimizer with a step size of 0.02, a factor for average gradient of 0.8 (β_1), a factor for average squared gradient of 0.999 (β_2) and update each sample for 300 iterations. We use AMSGrad because we found it converges faster for our 24D data. We perform early stopping when the loss converges. We constrain the sample value to be within the maximum and minimum values of the original dataset. We use the same optimizer (expect a lower learning rate of 0.001) to optimize the label. We constrain the magnitude of the label to be 2.

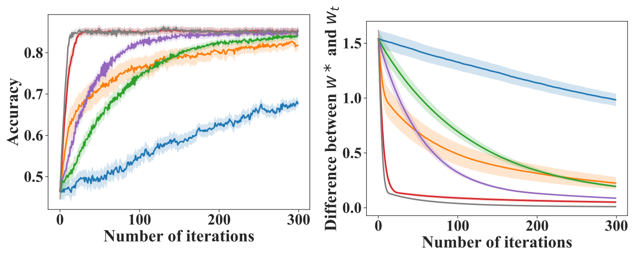

Figure 12: Omniscient teaching with or without label synthesis. Convergence comparison between our greedy teaching policy with several other baseline methods on MNIST.

C.3 Teaching Logistic Regression on Half-moon Data with Data Transformation

We observe similar behavior between the weight convergence and the accuracy convergence of the examined methods using a data transformation policy on half-moon data. Generally, we notice a much steeper convergence than the greedy teaching policy. We optimize the teacher using an Adam optimizer with a learning rate of 0.002, a factor for average gradient of 0.9

(β_1), a factor for average squared gradient of 0.999 (β_2) and train for 1000 iterations. For each iteration, we perform 40 steps of unrolling.

Model input. Model input is the current student weight $\boldsymbol{w_t}$, the difference to the target model weight $\boldsymbol{w_t} - \boldsymbol{w}^*$, one random sample/label pair from the original dataset $(\boldsymbol{x}, \boldsymbol{y})$. The synthesized sample $\tilde{\boldsymbol{x}}$ is conditioned on label $\boldsymbol{y}$.

Teacher architecture. The teacher is a simple MLP with three layers (input dimension (8) - 32 - 16 - output dimension (2)) and ReLU activation.

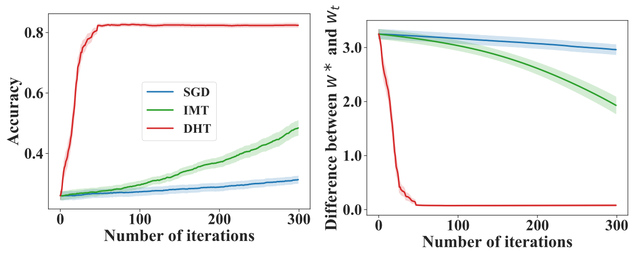

Figure 13: Accuracy and weight convergence using data transformation policy in binary classification on half-moon.

C.4 Teaching Logistic Regression on MNIST with Data Transformation

We observe similar behavior between the weight convergence and the accuracy convergence of the examined methods using a data transformation policy on MNIST data. Generally, we notice a much steeper convergence than the greedy teaching policy. We optimize the teacher using an Adam optimizer with a learning rate of 0.002, a factor for average gradient of 0.9 (β_1), a factor for average squared gradient of 0.999 (β_2) and train for 1000 iterations. For each iteration, we perform 40 steps of unrolling.

Model input. Model input is the current student weight $\boldsymbol{w_t}$, the difference to the target model weight $\boldsymbol{w_t} - \boldsymbol{w}^*$, one random sample/label pair from the original dataset $(\boldsymbol{x}, \boldsymbol{y})$. The synthesized sample $\tilde{\boldsymbol{x}}$ is conditioned on label $\boldsymbol{y}$.

Teacher architecture. The teacher is a MLP with five layers (input dimension (82) - 128 - 256 - 512 - 512 - 1024 - output dimension (24)), ReLU activation and 1D batch normalization.

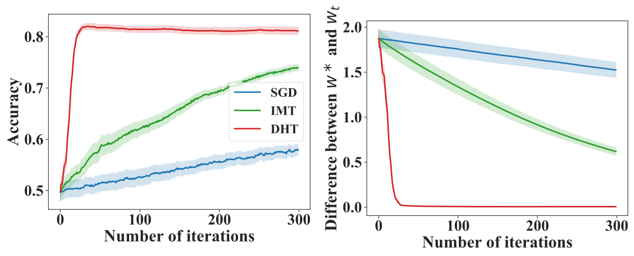

Figure 14: Accuracy and weight convergence using data transformation policy in 3/5 classification on MNIST.

C.5 Teaching Logistic Regression on Half-moon Data with Generative Modeling

We observe similar behavior between the weight convergence and the accuracy convergence of the examined methods using a VAE-based generative teacher on half-moon data. In general, the performance is slightly worse than the IMT baseline but still significantly outperforms optimizing using random samples (SGD).

Model input. Model input for the teacher is the current student weight $\boldsymbol{w_t}$, the difference to the target model weight $\boldsymbol{w_t} - \boldsymbol{w}^*$, one random sample/label pair from the original dataset $(\boldsymbol{x}, \boldsymbol{y})$. The synthesized sample $\tilde{\boldsymbol{x}}$ is conditioned on label $\boldsymbol{y}$. The pre-trained VAE models takes one random sample/label pair $(\boldsymbol{x}, \boldsymbol{y})$ as input.

Teacher architecture. The VAE-based teacher utilizes a pre-trained VAE model, with a simple MLP with three layers (input dimension (4) - 128 - 256 - 128 - output dimension) as encoder and another MLP with three layers (input dimension - 128 - 256 - 128 - output dimension) as decoder.

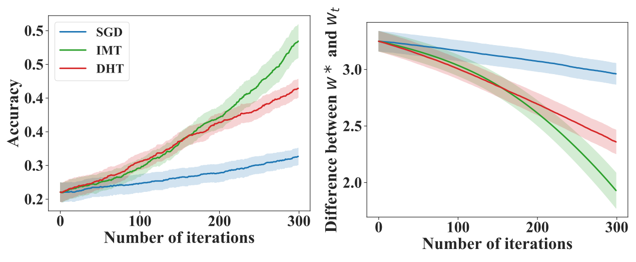

Figure 15: Accuracy and weight convergence using VAE-based teacher in binary classification on half-moon.

We observe similar behavior between the weight convergence and the accuracy convergence of the examined methods using a GAN-based generative teacher on half-moon data. In general, the performance is comparable with that of the IMT baseline and significantly outperforms optimizing using random samples (SGD).

Model input. Model input is the current student weight $\boldsymbol{w_t}$, the difference to the target model weight $\boldsymbol{w_t} - \boldsymbol{w}^*$ and one random label from the original dataset $\boldsymbol{y}$. The synthesized sample $\tilde{\boldsymbol{x}}$ is conditioned on label $\boldsymbol{y}$.

Teacher architecture. The GAN-based teacher is a simple MLP with three layers (input dimension (8) - 32 - 16 - output dimension (2)) and ReLU activation. The additional discriminator is a simple MLP with two layers (input dimension (4) - 8 - output dimension'), leaky ReLU (0.2) and a drop out layer (0.3).

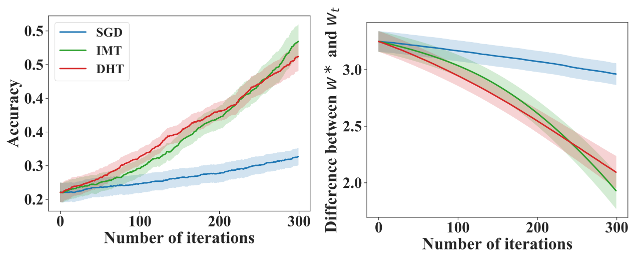

Figure 16: Accuracy and weight convergence using GAN-based teacher in binary classification on half-moon.

C.6 More Qualitative Results of Generative Modeling Policy

We visualize the data distribution of the synthesized data by a VAE-based teacher after we finish teaching. We observe that by using a VAE-based teacher, we can generate data samples with similar distribution in the x- and y-coordinates of both classes as the original data distribution. The generated samples are more widespread and match the original data distribution better; however, sometimes, samples are synthesized outside of the original data distribution.

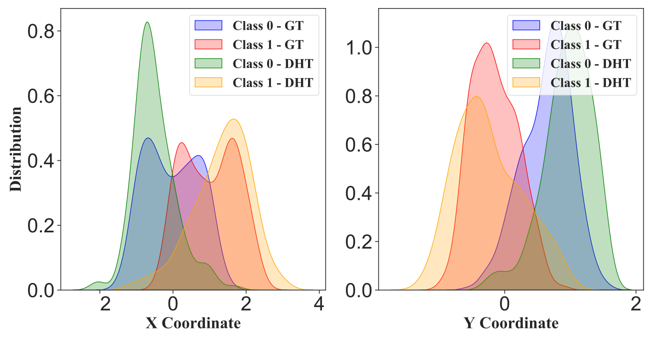

Figure 17: Visualization of the data distribution of the synthesized data by a VAE-based teacher after 300 iterations. GT means the ground truth.

We also visualize the data distribution of the synthesized data by a GAN-based teacher after we finish teaching. We observe that by using a GAN-based teacher, we can generate data samples with similar distribution in the x- and y-coordinates of class 1 as the original data distribution. Samples generated for class 0 are more clustered but still lie completely within the original data distribution. We do not interpret these clustered samples as an indication of mode collapse but rather view these samples as the most informative ones that can cause the model to converge faster to the desired w^*. Since we also observe this kind of clustered generation when using a greedy teaching policy.

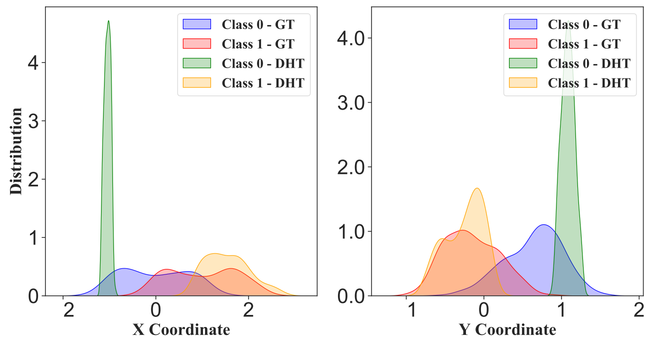

Figure 18: Visualization of the data distribution of the synthesized data by a GAN-based teacher after 300 iterations. GT means the ground truth.

Here we visualize the synthesized samples together with the ground truth data distribution, the target classifier w^* and the student classifier w_t at different epochs. We can clearly notice the difference between the generated data samples and also its effect on the student classifier. By synthesizing more clustered data samples using a GAN-based teacher, the student classifier is able to converge faster to w^*. When training a GAN-based teacher, there exists this trade-off between generating samples that will lead to faster convergence (clustered samples) and samples that are more similar to the original data distribution (spread samples). The advantage of using a VAE-based teacher is that the training is relatively stable, and the generated samples are much closer to the original dataset because we only teach in the feature space.

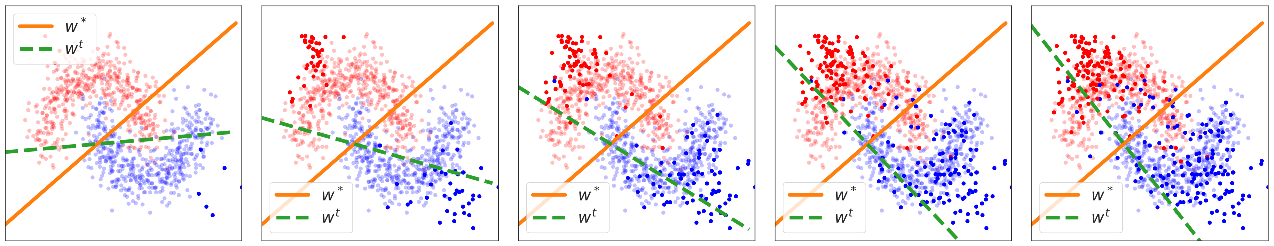

Figure 19: Visualization of the data synthesized by a VAE-based teacher after iteration 10, 80, 150, 240 and 290. The orange line indicates the target classifier w^*; the green dashed line indicates the student classifier. Different colors indicate different classes; points with lower opacity represent the ground truth data distribution.

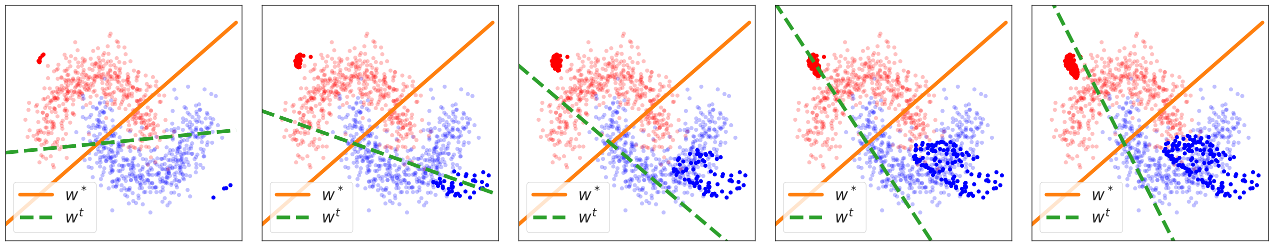

Figure 20: Visualization of the data synthesized by a GAN-based teacher after iteration 10, 80, 150, 240 and 290. The orange line indicates the target classifier w^*; the green dashed line indicates the student classifier. Different colors indicate different classes; points with lower opacity represent the ground truth data distribution.

C.7 Teaching Logistic Regression on MNIST with Generative Modeling

We observe similar behavior between the weight convergence and the accuracy convergence of the examined methods using a VAE-based generative teacher on MNIST data. In general, the performance is worse than the IMT baseline but still outperforms optimizing using random samples (SGD).

Model input. Model input for the teacher is the current student weight $\boldsymbol{w_t}$, the difference to the target model weight $\boldsymbol{w_t} - \boldsymbol{w}^*$, one random sample/label pair from the original dataset $(\boldsymbol{x}, \boldsymbol{y})$. The synthesized sample $\tilde{\boldsymbol{x}}$ is conditioned on label $\boldsymbol{y}$. The pre-trained VAE models takes one random sample/label pair $(\boldsymbol{x}, \boldsymbol{y})$ as input.

Teacher architecture. The VAE-based teacher utilizes a pre-trained VAE model, with a simple CNN with two 2D convolutional layers as encoder and another CNN with two 2D transposed convolution layers.

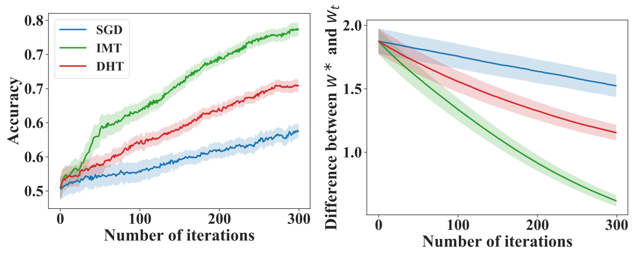

Figure 21: Accuracy and weight convergence using VAE-based teacher in 3/5 classification on MNIST.

We observe similar behavior between the weight convergence and the accuracy convergence of the examined methods using a GAN-based generative teacher on MNIST data. In general, the performance is comparable with that of the IMT baseline and significantly outperforms optimizing using random samples (SGD).

Model input. Model input for the teacher is the current student weight $\boldsymbol{w_t}$, the difference to the target model weight $\boldsymbol{w_t} - \boldsymbol{w}^*$, one random sample/label pair from the original dataset $(\boldsymbol{x}, \boldsymbol{y})$. The synthesized sample $\tilde{\boldsymbol{x}}$ is conditioned on label $\boldsymbol{y}$. The pre-trained VAE models takes one random sample/label pair $(\boldsymbol{x}, \boldsymbol{y})$ as input.

Teacher architecture. The GAN-based teacher utilizes a conditional deep convolutional GAN (DCGAN) to directly generate MNIST images with the original size (28×28). The synthesized images are then downscaled using the same projection matrix as the other data to teach the logistic regression learner. The generator consists of three blocks: each block consists of 2D transposed convolution layers, 2D batch normalization, and ReLU activation. Block x upscales the input feature ($\mathbb{R}^{N \times D}$) to a sample feature map ($\mathbb{R}^{N \times 128 \times 3 \times 3}$); block y upscales the label embedding ($\mathbb{R}^{N \times n_{cls}}$) to a label feature map ($\mathbb{R}^{N \times 128 \times 3 \times 3}$). Both feature maps are concatenated and inserted into the third block xy and upscaled to generate MNIST-like samples (($\mathbb{R}^{N \times 1 \times 28 \times 28}$)). The discriminator operates in a similar fashion: block x downscales the input image ($\mathbb{R}^{N \times 1 \times 28 \times 28}$) to a sample feature map ($\mathbb{R}^{N \times 32 \times 14 \times 14}$); block y downscales the one-hot label embedding ($\mathbb{R}^{N \times 10 \times 28 \times 28}$) to a label feature map ($\mathbb{R}^{N \times 32 \times 14 \times 14}$). Both feature maps are concatenated and inserted into the third block xy to make the prediction about real or fake.

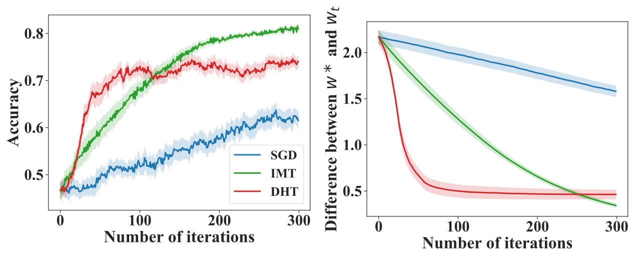

Figure 22: Accuracy and weight convergence using GAN-based teacher in 3/5 classification on MNIST.

C.8 Teaching Logistic Regression on Half-moon Data with Parameterized Black-box Teaching

We observe similar behavior between the weight convergence and the accuracy convergence of the examined methods using a parametrized black-box teaching policy on half-moon data. Even without the knowledge about w^*, the DHT is able to significantly outperform the IMT baseline. It is worth mentioning that here we use a surrogate objective of optimizing the classification loss, *i.e.*, we do not explicitly optimize the difference between w_t and w^*. Here, we just use a general w^* with good accuracy performance. In general, this teaching policy will not lead to a faster convergence to w^*.

Model input. Model input for the teacher is the current student weight $\boldsymbol{w_t}$ and one random sample/label pair from the original dataset $(\boldsymbol{x}, \boldsymbol{y})$. The synthesized sample $\tilde{\boldsymbol{x}}$ is conditioned on label $\boldsymbol{y}$. The pre-trained VAE models takes one random sample/label pair $(\boldsymbol{x}, \boldsymbol{y})$ as input.

Teacher architecture. The teacher is a simple MLP with three layers (input dimension (6) - 32 - 16 - output dimension (2)) and ReLU activation.

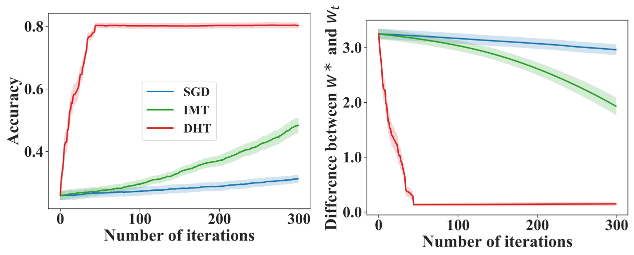

Figure 23: Accuracy and weight convergence using using parametrized black-box teaching policy in binary classification on half-moon.

C.9 Teaching Logistic Regression on MNIST with Parameterized Black-box Teaching

We observe similar behavior between the weight convergence and the accuracy convergence of the examined methods using a parametrized black-box teaching policy on half-moon data. Even without the knowledge about w^*, the DHT is able to significantly outperform the IMT baseline.

Model input. Model input for the teacher is the current student weight $\boldsymbol{w_t}$ and one random sample/label pair from the original dataset $(\boldsymbol{x}, \boldsymbol{y})$. The synthesized sample $\tilde{\boldsymbol{x}}$ is conditioned on label $\boldsymbol{y}$. The pre-trained VAE models takes one random sample/label pair $(\boldsymbol{x}, \boldsymbol{y})$ as input.

Teacher architecture. The teacher is a MLP with five layers (input dimension 58) - 128 - 256 - 512 - 512 - 1024 - output dimension (24)), ReLU activation and 1D batch normalization.

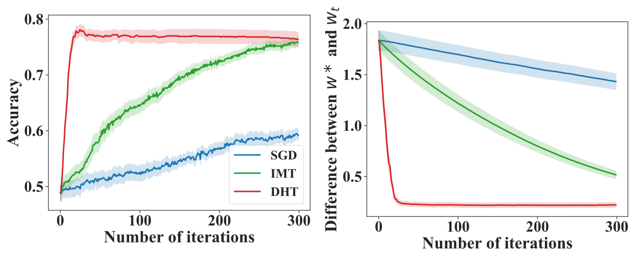

Figure 24: Accuracy and weight convergence using parametrized black-box teaching policy in 3/5 classification on MNIST.