Appendix

In this Appendix, we provide more implementation details in Section 7, defense against decision-based DFME in Section 8, more experiment results in Section 9, baseline description in Section 10. We then provide math derivation, including Lagrangian duality derivation, Wasserstein gradient flow for DRO and derivations of the worst-case test data simulation in Section 11.

7 More Implementation Details

7.1 Teacher Training Details

For LeNet-5 on MNIST, following [51], we train the target model for 10 epochs. Following [51], for ResNet34-8x and GoogLeNet on CIFAR10/100 datasets, we train the target model for 200 epochs. For epochs less than 80, we set its learning rate to be 0.1, for epoch 80 to 120, we set its learning rate to be 0.01. For epochs 120 to 200, we set its learning rate to be 0.001. For the score-based DFME clone model training, we follow the procedure in [51]. For decision-based DFME clone model training, we follow the procedure in [47].

7.2 Computing Resources

We use Nvidia-A6000 GPU and Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz to do experiments

8 Defense against decision-based DFME

The representative hard-label DFME methods are DFMS-HL [47] and ZSDB3KD [54]. In the following, we briefly describe how our proposed method can defend against those two attack methods.

8.1 Defending against DFMS-HL by Perturbing the Label Prediction

We denote the ground-truth label for the query data $\boldsymbol{x}$ as $\vec{y}(\boldsymbol{x})$, which is predicted by the teacher without perturbation. The predicted label by the target victim model with parameters $\boldsymbol{\theta}_T$ is $y(\boldsymbol{x} + h_{\boldsymbol{\omega}}(\boldsymbol{x}, \boldsymbol{\epsilon}))$. Suppose the label is wrongly predicted to be $y'(\boldsymbol{x})$ with the perturbed input, i.e., $y'(\boldsymbol{x}) \neq \vec{y}(\boldsymbol{x})$. Thus, minimizing the loss $\mathcal{L}(\mathcal{C}(\boldsymbol{x}, \boldsymbol{\phi}_C), y'(\boldsymbol{x})))$ would make the attacker learn incorrect information.

8.2 Brief Introduction of ZSDB3KD

We briefly describe the method in the following. Intuitively, the distance between a training sample to the teacher's decision boundary is usually larger than the distance between a randomly generated noise image to the boundary since the teacher is trained to distinguish the training samples maximally [54]. The goal of ZSDB3KD is to move the random noise away from the decision boundary gradually.

Suppose the closest point to the point $\boldsymbol{q}_0^n$ (the random noise with prediction label n) on the decision boundary is $\boldsymbol{q}_*^n$, the data point $\boldsymbol{q}_0^n$ is updated as:

$$\boldsymbol{q}_0^n = \boldsymbol{q}_0^n - \xi \nabla T(\boldsymbol{q}_*^n), \tag{14}$$

Where ξ is the step size and $\nabla T(\boldsymbol{q}_0^n)$ is the gradient at the decision boundary of the teacher model via zeroth-order gradient, defined as:

$$\nabla T(\boldsymbol{q}_0^n) = \frac{1}{m} \sum_{i=1}^{m} sign(\boldsymbol{q}_0^n + \delta \boldsymbol{\mu}_i) \boldsymbol{\mu}_i, \tag{15}$$

where $sign(\boldsymbol{q}_0^n + \delta \boldsymbol{\mu}_i)$ is defined as the following

$$sign(\boldsymbol{q}_0^n + \delta \boldsymbol{\mu}_i) = \begin{cases} +1, & T(\boldsymbol{q}_0^n + \delta \boldsymbol{\mu}_i) = n; \\ -1, & Otherwise. \end{cases} \tag{16}$$

After a certain amount of iterations, the attacker can obtain a certain amount of pseudo samples from the target model T by the above approach. Then, with the proposed approach in [54] to construct

the pseudo data output probabilities based on the sample's distance to the target model's decision boundary. Finally, the attacker will train a clone model based on the generated pseudo samples and their class probabilities.

8.3 Defending against ZSDB3KD by Disturbing the Zero-Order Gradient Estimation

By adding data-dependent random perturbation, the zeroth-order gradient estimation at the decision boundary in Eq. (15) becomes:

$$\nabla T(\boldsymbol{q}_0^n) = \frac{1}{m}\sum_{i=1}^{m} sign(\boldsymbol{q}_0^n + \delta\boldsymbol{\mu}_i + h_{\boldsymbol{\omega}}(\boldsymbol{q}_0^n + \delta\boldsymbol{\mu}_i, \boldsymbol{\epsilon}))\boldsymbol{\mu}_i. \tag{17}$$

Since the synthetic noisy data are often nearer to the decision boundary than real training data [54], thus adding data-dependent random perturbation would easily make the prediction to change the sign function in Eq. (16) and Eq. (15). This wrong information would incorrectly update the pseudo sample in Eq. (14) so that the generated samples are not informative for the attacker to learn a good clone model.

9 More Experimental Results

9.1 Application on data-based model extraction (DBME) Attack

In this section, we apply our method (MeCo) on data-based model extraction (DBME) attack, where an attacker can access a small subset of the in-distribution training data [41], or a relevant surrogate dataset [36] of the target model. MeCo is orthogonal to existing DBME defense methods, we integrate MeCo with EDM [22] as an example. We use Knockoff Nets [37] and Jacobian Based Dataset Augmentation (JBDA) [41] to extract the target victim model. We present the results in Table 6.

- Knockoff Nets [37] extracts the target black-box model with a relevant surrogate dataset to query the target model. Then, the attacker trains a clone model with the surrogate dataset and the target model predictions on the surrogate dataset.
- Jacobian Based Dataset Augmentation (JBDA) [41]. Attacker first uses a small subset of in-distribution data examples to query the target model to construct a labeled dataset. The attacker then trains the clone model on the constructed labeled dataset. The attacker also augments the dataset with additional synthetic examples by perturbing the raw data input with the Jacobian of the loss function.

Table 6: Clone model accuracy after applying defense methods against DBME on **CIFAR-10** by combing MeCo with EDM [22]

Dataset	KnockoffNets			JBDA		
	undefended	EDM	EDM + MeCo	undefended	EDM	EDM + MeCo
MNIST	90.18	51.34	**46.19**	88.48	79.04	**22.49**
CIFAR10	85.39	68.50	**59.18**	22.03	22.01	**21.81**
CIFAR100	53.04	41.16	**35.71**	4.09	3.69	**3.17**

9.2 More DFME defense results

For MNIST in Table 7, we follow [54] and use LeNet5 as the target model, and use LeNet5, LeNet5-Half, and LeNet5-1/5 as the clone network architectures. The LeNet5-Half and LeNet5-1/5 networks have half and 1/5 number of convolutional filters in each layer of that in LeNet-5.

Table 7: Clone model accuracy after applying different defense methods on **MNIST** with LeNet5 as the target model.

Attack	Defense	Clone Model Architecture		
		LeNet5	LeNet5-Half	LeNet5-1/5
DFME	undefended ↓	$98.76 \pm 0.27\%$	$96.65 \pm 0.43\%$	$94.62 \pm 0.69\%$
	RandP ↓	$92.25 \pm 0.32\%$	$91.86 \pm 0.49\%$	$90.37 \pm 0.73\%$
	P-poison ↓	$88.34 \pm 0.78\%$	$86.09 \pm 0.96\%$	$84.98 \pm 1.07\%$
	GRAD ↓	$87.22 \pm 0.70\%$	$85.38 \pm 0.91\%$	$84.23 \pm 1.16\%$
	MeCo ↓	**85.07 ± 0.87%**	**82.93 ± 1.27%**	**82.57 ± 1.53%**
MAZE	undefended ↓	$98.09 \pm 0.26\%$	$95.91 \pm 0.36\%$	$89.41 \pm 0.38\%$
	(RandP, C) ↓	$92.77 \pm 0.41\%$	$88.71 \pm 0.90\%$	$86.38 \pm 0.95\%$
	P-poison ↓	**85.34 ± 0.78%**	$87.05 \pm 0.86\%$	$86.18 \pm 0.97\%$
	GRAD ↓	$85.63 \pm 0.91\%$	$87.31 \pm 0.75\%$	$86.42 \pm 0.78\%$
	MeCo ↓	$88.57 \pm 0.87\%$	**86.99 ± 0.78%**	**85.03 ± 0.68%**

Table 8: Clone model accuracy after applying different defense methods against existing DFME methods on **MiniImageNet** with ResNet34-8x as the target model.

Attack	Defense	Clone Model Architecture		
		ResNet18-8X	MobileNetV2	DenseNet121
DFMS-HL	undefended ↓	$46.72 \pm 4.86\%$	$40.35 \pm 4.97\%$	$38.71 \pm 3.85\%$
	RandP ↓	$45.09 \pm 4.93\%$	$39.51 \pm 4.83\%$	$38.08 \pm 3.95\%$
	P-poison ↓	$45.16 \pm 5.03\%$	$39.06 \pm 4.72\%$	$37.78 \pm 4.26\%$
	GRAD ↓	$45.32 \pm 5.21\%$	$39.17 \pm 4.85\%$	$37.85 \pm 4.32\%$
	MeCo ↓	**39.23±4.83%**	**35.81 ± 4.69%**	**32.30 ± 4.56%**
DFME	undefended ↓	$35.89 \pm 3.97\%$	$28.71 \pm 3.25\%$	$25.05 \pm 3.68\%$
	RandP ↓	$30.76 \pm 4.09\%$	$22.06 \pm 3.83\%$	$20.23 \pm 3.97\%$
	P-poison ↓	$29.36 \pm 4.23\%$	$21.83 \pm 3.77\%$	$20.01 \pm 3.89\%$
	GRAD ↓	$29.87 \pm 3.76\%$	$21.65 \pm 3.75\%$	$19.82 \pm 3.77\%$
	MeCo ↓	**23.29 ± 3.83%**	**17.83±3.67%**	**16.73 ± 3.88%**

9.3 Change the Target Model as GoogLeNet

Table 9: Clone model accuracy after applying different defense methods on **CIFAR10** against existing DFME methods with **GoogLeNet** as the target model

Attack	Defense	Clone Model Architecture		
		ResNet18-8X	MobileNetV2	DenseNet121
DFMS-HL	undefended ↓	$61.38 \pm 1.93\%$	$60.23 \pm 1.82\%$	$58.37 \pm 2.19\%$
	RandP ↓	$61.06 \pm 2.18\%$	$59.82 \pm 1.87\%$	$58.03 \pm 2.31\%$
	P-poison ↓	$60.73 \pm 1.95\%$	$59.07 \pm 1.93\%$	$58.09 \pm 2.33\%$
	GRAD ↓	$59.41 \pm 1.90\%$	$58.72 \pm 1.91\%$	$58.18 \pm 1.98\%$
	MeCo ↓	**53.77 ± 2.21%**	**53.89 ± 1.96%**	**53.28 ± 2.20%**
DFME	undefended ↓	$74.67 \pm 1.35\%$	$75.23 \pm 1.51\%$	$70.96 \pm 0.78\%$
	RandP ↓	$70.05 \pm 1.97\%$	$70.54 \pm 1.72\%$	$66.78 \pm 2.07\%$
	P-poison ↓	$67.32 \pm 1.89\%$	$69.28 \pm 1.82\%$	$64.90 \pm 2.01\%$
	GRAD ↓	$69.32 \pm 1.91\%$	$66.32 \pm 1.97\%$	$65.73 \pm 1.77\%$
	MeCo ↓	**54.53 ± 2.17%**	**50.28 ± 2.32%**	**59.42 ± 2.51%**
MAZE	undefended ↓	$23.18 \pm 2.37\%$	$19.01 \pm 1.09\%$	$21.28 \pm 3.16\%$
	RandP ↓	$22.03 \pm 2.16\%$	$17.24 \pm 1.16\%$	$19.23 \pm 1.38\%$
	P-poison ↓	$21.28 \pm 2.33\%$	$17.28 \pm 1.22\%$	**17.76 ± 1.16%**
	GRAD ↓	$20.79 \pm 2.07\%$	$16.96 \pm 1.27\%$	$18.09 \pm 1.34\%$
	MeCo ↓	**19.65 ± 2.61%**	**14.16 ± 2.30%**	$18.31 \pm 2.75\%$

9.4 Hyperparameter Analysis

We evaluate the hyperparameter sensitivity for γ in Table 10. We can observe that the model utility increases as γ increases with a trade-off of a decrease in defense performance. We additionally present the sensitivity analysis of Q in Table 11. Results show that $Q = 2$ performs the best. With the increase of Q, the simulated test data distribution may become harder to learn; thus, the benign accuracy slightly worsens.

Table 10: Sensitivity analysis of γ with ResNet34-8X as the target model and ResNet18-8X as the clone model on CIFAR10.

Method	l_1 norm model utility ↓	defense performance ↓
$\gamma = 0.0$	0.128 ± 0.007	$51.29 \pm 2.19\%$
$\gamma = 0.5$	0.099 ± 0.006	$51.68 \pm 1.96\%$
$\gamma = 1.0$	0.08 ± 0.005	$55.16 \pm 1.77\%$

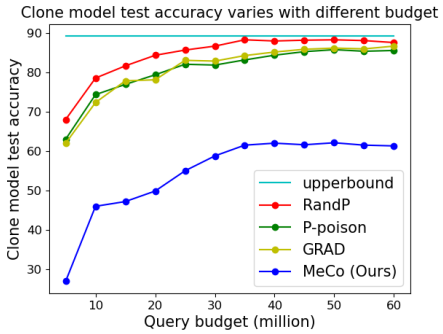

Figure 5: Clone model test accuracy on CIFAR10 (the lower, the better) varies with query budgets after applying different defense methods to the victim model. MeCo (Ours) significantly outperforms various defense methods.

Table 11: Sensitivity analysis of Q with ResNet34-8X as the target model and ResNet18-8X as the clone model.

Method	CIFAR10
$Q = 1$	$93.97 \pm 0.72\%$
$Q = 2$	$94.17 \pm 0.56\%$
$Q = 3$	$93.81 \pm 0.78\%$
$Q = 5$	$93.36 \pm 0.67\%$

9.5 Ablation Study

Effect of DRO We evaluate how much improvement DRO can bring to the model utility preservation in Table 12. We can observe that DRO significantly improves the model utility by $5.3\% - 5.6\%$ compared to the method without DRO training. The results are close to the standard training without defense.

Table 12: Ablation study of DRO on different datasets with ResNet34-8x as the target model.

Method	Dataset	
	CIFAR10	CIFAR100
Standard ↑	$94.91 \pm 0.37\%$	$76.71 \pm 1.25\%$
MeCo ↑	$94.17 \pm 0.56\%$	$75.36 \pm 0.68\%$
MeCo w/o DRO ↑	$88.59 \pm 0.42\%$	$69.93 \pm 0.32\%$

9.6 Clone model test accuracy varies with different query budgets

9.7 Running Time and Memory Cost Analysis

We provide the test time running time analysis in Table 13. MeCo achieves up to $37\times$ speed up on CIFAR10 and $172\times$ speed up on CIFAR100 compared with P-poison; achieves speed up of $17\times$ on CIFAR10 and speed up of $49\times$ on CIFAR100 compared with GRAD. This is because P-poison and GRAD solve computationally expensive optimization problems during testing. In contrast, MeCo does not need this optimization.

Table 13: Running time (seconds) during deployment

Algorithm	CIFAR10	CIFAR100
P-poison	223.68 ± 2.78	1126.83 ± 8.71
GRAD	107.29 ± 1.66	317.83 ± 4.29
MeCo	$\mathbf{6.06 \pm 0.80}$	$\mathbf{6.53 \pm 0.82}$

We additionally evaluate the test time memory consumption compared with different methods in Table 14, MeCo uses less memory compared with P-poison since P-poison needs backpropagation and optimization and during test time. GRAD needs much more memory since GRAD needs double backpropogation, which is memory intensive.

Table 14: Running memory consumption during deployment

Algorithm	CIFAR10
P-poison	2.2 GB
GRAD	10.83 GB
MeCo	2.03 GB

9.8 Training time analysis

We compare MeCo training efficiency to baselines in Table 15. MeCo increases the training cost by $1.3\times$ cost. However, since MeCo substantially improves the test time running efficiency, it is worth this running cost.

Table 15: Training Time on CIFAR10

Algorithm	one epoch training time (seconds)
Standard	37.68 ± 1.25
MeCo	86.68 ± 1.19

10 Baseline

- **RandP** [38]: Random target model output perturbation: we randomly perturb the prediction logits with random noise $\widehat{\boldsymbol{a}} = \boldsymbol{a} + \boldsymbol{\xi}$; where $\boldsymbol{a}$ is the logit, and it is calculated from the probability outputs $\boldsymbol{y}$ as $\boldsymbol{a} = \log \frac{\boldsymbol{y}}{1-\boldsymbol{y}}$; $\boldsymbol{\xi}$ is the random noise vector. The logits are renormalized as $\widehat{\boldsymbol{y}} = \frac{1}{1+e^{-\widehat{\boldsymbol{a}}}}$ as outputs for the attacker.
- **Prediction Poisoning (P-poison)** [38]: They propose a perturbation objective, named Maximizing Angular Deviation (MAD). The main goal of MAD approach is to perturb the prediction probabilities $\boldsymbol{y}_T$ of the target model so that it results in an adversarial perturbed gradient that maximally deviates from the original gradient of target model.
- **GRAD** [33]: is a gradient redirection method so that the adversary gradient update direction could be in any arbitrary direction.

11 Math derivation

11.1 Lagrangian Duality Derivation

The principle oof Lagrangian duality [6] is to transform the constraints through augmenting the optimization objective function with a weighted sum of the constraint

$$\min_{\boldsymbol{\theta}_T,\, \boldsymbol{\omega}} \sup_{\mu \in \mathcal{P}} \mathbb{E}_{\boldsymbol{x}\sim\mu} \mathcal{L}(\boldsymbol{\theta}_T, \boldsymbol{x} + h_{\boldsymbol{\omega}}(\boldsymbol{x}, \boldsymbol{\epsilon}), y) \tag{18}$$

$$\text{s.t. } \mathcal{P} = \{\mu : \mathbb{KL}(\mu||\mu_0) \leq \beta\} \tag{19}$$

$$\mathbb{E}_{\boldsymbol{x}\sim\mu}||T(\boldsymbol{x} + h_{\boldsymbol{\omega}}(\boldsymbol{x}, \boldsymbol{\epsilon})) - T(\boldsymbol{x})||_1 \leq \tau \tag{20}$$

Then, we can get the equivalent constrained optimization:

$$\begin{aligned} \min_{\boldsymbol{\theta}_T,\, \boldsymbol{\omega}} \sup_{\mu} [&\mathbb{E}_{\boldsymbol{x}\sim\mu}\, \mathcal{L}(\boldsymbol{\theta}_T, \boldsymbol{x} + h_{\boldsymbol{\omega}}(\boldsymbol{x}, \boldsymbol{\epsilon}), y) - \mathbb{KL}(\mu||\mu_0)] \\ &+\gamma \mathbb{E}_{\boldsymbol{x}\sim\mu}||T(\boldsymbol{x} + h_{\boldsymbol{\omega}}(\boldsymbol{x}, \boldsymbol{\epsilon})) - T(\boldsymbol{x})||_1 \end{aligned} \tag{21}$$

where γ is the Lagrange multiplier for the model utility constraints. The coefficient before the $\mathbb{KL}$ divergence $\mathbb{KL}(\mu||\mu_0)$ is set to 1 since it can be dealt with WGF defined in Section 11.2.

11.2 Gradient Flow for DRO derivation

Given a batch of raw training data $\mathcal{D}_{tr} = \{(\boldsymbol{x}_0^1, y^1), (\boldsymbol{x}_0^2, y^2), \cdots, (\boldsymbol{x}_0^N, y^N)\}$, where N is the batch size. We denote $\boldsymbol{x}_t^i$ as the i^{th} datapoint in the perturbed batch data after t evolution steps. The raw training data is sampled from the random variable X_0, i.e., $\{\boldsymbol{x}_0^1, \boldsymbol{x}_0^2, \cdots, \boldsymbol{x}_0^N\} \sim X_0$. The random variable X_0 has probability distribution μ_0, i.e., $X_0 \sim \mu_0$. At perturbation time t, the training data $\mathcal{D}_{tr}$ is perturbed as random variable X_t, i.e., $\{\boldsymbol{x}_t^1, \boldsymbol{x}_t^2, \cdots, \boldsymbol{x}_t^N\} \sim X_t$. The probability distribution of random variable X_t has probability measure μ_t, i.e., $X_t \sim \mu_t$. We define the empirical measure on the perturbed training data at time t as $\hat{\mu}_t = \frac{1}{N}\sum_{i=1}^{i=N} \delta(\boldsymbol{x}_t^i)$; where δ is the Dirac measure. We model the training data perturbation process as continuous WGF in probability measure space,

i.e., we use the continuous $(\mu_t)_{t\geq 0}$ to model the gradual change of the training data probability distribution. The gradual change of probability measures $(\mu_t)_{t\geq 0}$ will in turn determine the training data distribution perturbation process $(X_t)_{t\geq 0}$ in Euclidean space.

Before we introduce how we formulate the perturbation as WGF, we first present some preliminaries. In calculus of variations, the first variation is defined as following:

Definition 11.1 (First Variations)**.** [34] The first variation of a functional $F(\mu)$ is defined as the following functional at μ

$$\frac{\delta F}{\delta \mu}(\mu) = \lim_{\epsilon \to 0} \frac{F(\mu + \epsilon\psi) - F(\mu)}{\epsilon}, \tag{22}$$

where ψ could be any arbitrary function.

By the above defined calculus of variation [34], the first variation for the KL divergence and $V(\mu)$ are [3]:

$$\frac{\delta \mathbb{KL}(\mu||\pi)}{\delta \mu} = \log \frac{\mu}{\pi} + 1;$$
$$\frac{\delta V(u)}{\delta \mu} = U(\boldsymbol{x}, \boldsymbol{\theta}_T) = -\mathcal{L}(\boldsymbol{\theta}_T, \boldsymbol{x}, y)$$

Then, we present the Wasserstein gradient flow in probability measure space. We denote $\mathcal{P}_2(\mathbb{R}^d)$ as the space of probability measures on $\mathbb{R}^d$ with finite second-order moments. Each element $\mu \in \mathcal{P}_2(\mathbb{R}^d)$ is a probability measure represented by its density function $\mu : \mathbb{R}^d \to \mathbb{R}$ with respect to Lebesgue measure dx. The Wasserstein distance between two probability measures $\mu_1, \mu_2 \in \mathcal{P}_2(\mathbb{R}^d)$ is defined as:

$$W_2(\mu_1, \mu_2) = \left(\min_{\omega \in \prod(\mu_1, \mu_2)} \int ||\boldsymbol{x} - \boldsymbol{x}'||^2 d\omega(\boldsymbol{x}, \boldsymbol{x}') \right)^{1/2},$$

where $\prod(\mu_1, \mu_2) = \{\omega | \omega(A \times \mathbb{R}^d) = \mu_1(A), \omega(\mathbb{R}^d \times B) = \mu_2(B)\}$. ω is the joint probability measure with marginal measure of μ_1 and μ_2 respectively. Thus, $\mathbb{W}^2 = (\mathcal{P}_2(\mathbb{R}^d), W_2)$ forms a metric space.

Definition 11.2 (Wasserstein Gradient Flow)**.** [3]. Suppose we have a Wasserstein space $\mathbb{W}^2 = (\mathcal{P}_2(\mathbb{R}^d), W_2)$. A curve of $(\mu_t)_{t\geq 0}$ of probability measures is a Wasserstein gradient flow for functional F if it satisfies

$$\partial_t \mu_t = \nabla_{W_2} F(\mu_t) := div\left(\mu_t \nabla \frac{\delta F}{\delta \mu}(\mu_t)\right), \tag{23}$$

where $:=$ means defined as, and $div(\boldsymbol{r}) := \sum_{i=1}^{d} \partial_{\boldsymbol{z}^i} \boldsymbol{r}^i(\boldsymbol{z})$ is the divergence operator of a vector-valued function $\boldsymbol{r} : \mathbb{R}^d \to \mathbb{R}^d$, where $\boldsymbol{z}^i$ and $\boldsymbol{r}^i$ are the i th element of $\boldsymbol{z}$ and $\boldsymbol{r}$; ∇ is the gradient of a scalar-valued function. $\nabla_{W_2} F(\mu_t) := div(\mu_t \nabla \frac{\delta F}{\delta \mu}(\mu_t))$ is the Wasserstein gradient of functional F at μ_t, where $\frac{\delta F}{\delta \mu}(\mu_t)$ is the first variation of F at μ_t. The first variation of a *functional* in function space is analogous to the first-order gradient of a *function* in Euclidean distance. Intuitively, the above defined WGF states that the probability measure μ_t moves along the steepest path of the functional $F(\mu)$ in the Wasserstein space of probability measures towards the target probability distribution.

11.3 Derivations of Test Data Distribution Simulation

We derive the solution to DRO in kernel space in the following:

$$\frac{\delta \mathbb{KL}(\mu||\pi)}{\delta \mu} = \log \frac{\mu}{\pi} + 1;$$
$$\frac{\delta V(u)}{\delta \mu} = U(\boldsymbol{x}, \boldsymbol{\theta}_T) = -\mathcal{L}(\boldsymbol{\theta}_T, \boldsymbol{x}, y).$$

The target worst-case test data distribution is defined as $\pi \propto e^{-U}$; where the energy function $U(\boldsymbol{x}, \boldsymbol{\theta}_T)$ is defined above.

We transform the Wasserstein gradient $\nabla_{W_2} F(\mu_t)$ in Eq. (9) by the transformation $\mathcal{K}_\mu \nabla_{W_2} F(\mu_t)$ with the integral operator $\mathcal{K}_\mu f(\boldsymbol{x}) = \int K(\boldsymbol{x}, \boldsymbol{x}') f(\boldsymbol{x}') d\mu(\boldsymbol{x}')$; where we use $\mathcal{H}$ to denote the RKHS space induced by the kernel K. Then, we transform the WGF Eq. (9) into the following kernelized WGF [29]:

$$\partial_t \mu_t = div(\mu_t \mathcal{K}_{\mu_t} \nabla \frac{\delta F}{\delta \mu}(\mu_t)). \tag{24}$$

This equation reveals that the random variable X_t, which represents the perturbed training data distribution at continuous time t follows the following differential equation [29, 8]:

$$\frac{dX}{dt} = -[\mathcal{K}_\mu \nabla \frac{\delta F}{\delta \mu}(\mu_t)](X). \tag{25}$$

The kernelized Wasserstein gradient can be derived as the following equation:

$$\mathcal{K}_{\mu_t} \nabla \log \frac{\mu_t}{\pi}(\boldsymbol{x}) := \int K(\boldsymbol{x}, \cdot) \nabla \log \frac{\mu_t}{\pi} = \int K(\boldsymbol{x}, \cdot) \nabla U d\mu_t - \int \nabla_2 K(\boldsymbol{x}, \cdot) d\mu_t \tag{26}$$

For the above equation, we adopt integration by parts technique in the second identity. ∇_2 is defined as the gradient of the kernel w.r.t. the second argument.

We then plug Eq. 26 into Eq. 25 to obtain the following differential equation for the continuous gradual change of the training data distribution:

$$\frac{dX}{dt} = -\int K(\boldsymbol{x}, \cdot) \nabla U d\mu_t + \int \nabla_2 K(\boldsymbol{x}, \cdot) d\mu_t \tag{27}$$

We further discretize the above Eq. (27) and view each training data as one particle, and arrive at the following training data distribution perturbation update equation:

$$\boldsymbol{x}_{t+1}^i - \boldsymbol{x}_t^i = -\frac{\alpha}{N} \sum_{j=1}^{j=N} [\underbrace{k(\boldsymbol{x}_t^i, \boldsymbol{x}_t^j) \nabla_{\boldsymbol{x}_t^j} U(\boldsymbol{x}_t^j, \boldsymbol{\theta}_T)}_{\text{smoothed gradient}} + \underbrace{\nabla_{\boldsymbol{x}_t^j} k(\boldsymbol{x}_t^i, \boldsymbol{x}_t^j)}_{\text{repulsive term}}], \tag{28}$$

12 Theoretical Analysis

since the input is perturbed, then the loss function estimation is noisy for the attacker. Consequently, the gradient estimation is noisy for the attacker. The attacker adopts the following inaccurate loss function.

$\mathcal{L}_C^{\Delta}(\boldsymbol{\theta}_C) = KL(T(x + h^{\omega}(x, \epsilon); \boldsymbol{\theta}_T), C(x; \boldsymbol{\theta}_C))$

where $T(x, \boldsymbol{\theta}_T)$ is the target victim model with parameters $\boldsymbol{\theta}_T$, $C(x; \boldsymbol{\theta}_C)$ is the clone model with parameters $\boldsymbol{\theta}_C$ and $h_\omega(x, \epsilon)$ is the data-dependent perturbation.

The ground truth loss function without input perturbation (inaccessible to the attackers since attacker does not know the amount of noise added to the input) is shown in the following

$\mathcal{L}_C(\boldsymbol{\theta}_C) = KL(T(x; \boldsymbol{\theta}_T), C(x; \boldsymbol{\theta}_C))$

The attacker optimizes the following loss function.

$\mathcal{L}_{\mathcal{C}}{}^* = \min_{\boldsymbol{\theta}_C} \mathcal{L}_C^{\Delta}(\boldsymbol{\theta}_C)$

Due to the noisy loss function, attacker loss function gradient becomes noisy and inaccurate, shown as the following:

We model the attacker loss gradient as the following:

$G_t(\boldsymbol{\theta}_C) = \nabla_{\boldsymbol{\theta}_C} \mathcal{L}_{\mathcal{C}}{}^{\Delta}(\boldsymbol{\theta}_C) = \nabla_{\boldsymbol{\theta}_C} \mathcal{L}_C(\boldsymbol{\theta}_C) + B_t(\boldsymbol{\theta}_C) + N_t(\boldsymbol{\theta}_C)$

where $\nabla_{\boldsymbol{\theta}_C} \mathcal{L}_C^{\Delta}(\boldsymbol{\theta}_C)$ is the actual gradient adopted by the attacker,

$\nabla_{\boldsymbol{\theta}_C}\mathcal{L}_C(\boldsymbol{\theta}_C)$ is the ground truth loss gradient without input perturbation, which is inaccessible to attacker,

$B_t(\boldsymbol{\theta}_C)$ are the gradient bias due to the loss function introduced by the perturbation generator h_w.

$N_t(\boldsymbol{\theta}_C)$ is the random variable introduced by the randomness in the data samples. For analysis convenience, the expectation of the noise is assumed to be zero, i.e. $\mathbb{E}N_t(\boldsymbol{\theta}_C) = 0$.

where the attacker updates the clone model as $\boldsymbol{\theta}_C^{t+1} = \boldsymbol{\theta}_C^t - \alpha G_t$; where α is the learning rate.

Assumptions

Assumption 12.1. We assume Polyak- Łojasiewicz (PL) condition [20] on the attacker loss function $\mathcal{L}_C(\boldsymbol{\theta}_C)$ $|\nabla_{\boldsymbol{\theta}_C}\mathcal{L}_{\mathcal{C}}(\boldsymbol{\theta}_C)| \geq 2\omega(\mathcal{L}_{\mathcal{C}}(\boldsymbol{\theta}_C))$

Assumption 12.2. We assume the following smoothness condition for the loss function $\mathcal{L}_C(\boldsymbol{\theta}_C)$ by following [2]:

$\mathcal{L}C(\boldsymbol{\theta}_C^2) \leq \mathcal{L}C(\boldsymbol{\theta}_C^1) + \langle \nabla_{\boldsymbol{\theta}_C}\mathcal{L}_C(\boldsymbol{\theta}_C^1), \boldsymbol{\theta}_C^2 - \boldsymbol{\theta}_C^1\rangle + \frac{A}{2}|\boldsymbol{\theta}_C^2 - \boldsymbol{\theta}_C^1|^2$

Assumption 12.3. We assume the following for gradient bias and randomness

$\mathbb{E}||N_t(\boldsymbol{\theta}_C)||^2 \leq D||\mathcal{L}_C(\boldsymbol{\theta}_C) + B_t(\boldsymbol{\theta}_C)||^2 + \rho^2$

$||B_t(\boldsymbol{\theta}_C)||^2 \leq d||\nabla_{\boldsymbol{\theta}_C}\mathcal{L}_C(\boldsymbol{\theta}_C)||^2 + \tau^2$ $(0 \leq d < 1)$

where D, d, ρ, τ are all constants.

Theory

Theorem 12.4. *Assume the attacker loss $\mathcal{L}_C(\boldsymbol{\theta}_C)$ function satisfy assumption 2 and 3. Then, the attacker loss function satisfies the following inequality:*

$\mathbb{E}[\mathcal{L}_{\mathcal{C}}(\boldsymbol{\theta}_C^{t+1})] \leq \mathcal{L}_{\mathcal{C}}(\boldsymbol{\theta}_C^t) + \frac{\alpha}{2}(d-1)|\nabla_{\boldsymbol{\theta}_C}\mathcal{L}_C(\boldsymbol{\theta}_C)|^2 + \frac{\alpha}{2}\tau^2 + \frac{\alpha^2 A\rho^2}{2}$

Proof. By adopting assumption 2 and 3. We set $\boldsymbol{\theta}_C^1 = \boldsymbol{\theta}_C^{t+1}$ and $\boldsymbol{\theta}_C^2 = \boldsymbol{\theta}_C^t$ with $\boldsymbol{\theta}_C^{t+1} = \boldsymbol{\theta}_C^t - \alpha G_t$

$\mathbb{E}[\mathcal{L}_{\mathcal{C}}(\boldsymbol{\theta}_C^{t+1})] \leq \mathcal{L}_{\mathcal{C}}(\boldsymbol{\theta}_C^t) - \alpha\langle \nabla_{\boldsymbol{\theta}_C}\mathcal{L}_C(\boldsymbol{\theta}_C), \mathbb{E}(G_t)\rangle + \frac{\alpha^2 A}{2}\left(\mathbb{E}|G_t|^2\right)$

The above equation can be equivalently converted into the following inequality due to the fact: $\mathbb{E}|G_t|^2 = \mathbb{E}|G_t - \mathbb{E}G_t + \mathbb{E}G_t|^2 = \mathbb{E}|G_t - \mathbb{E}G_t|^2 + 2\mathbb{E}[(G_t - \mathbb{E}G_t)\mathbb{E}G_t] + \mathbb{E}|\mathbb{E}G_t|^2 = \mathbb{E}|G_t - \mathbb{E}G_t|^2 + \mathbb{E}|\mathbb{E}G_t|^2$

$\mathbb{E}[\mathcal{L}_{\mathcal{C}}(\boldsymbol{\theta}_C^{t+1})] \leq \mathcal{L}_{\mathcal{C}}(\boldsymbol{\theta}_C^t) - \alpha\langle \nabla_{\boldsymbol{\theta}_C}\mathcal{L}_C(\boldsymbol{\theta}_C), \mathbb{E}(G_t)\rangle + \frac{\alpha^2 A}{2}\left(\mathbb{E}|G_t - \mathbb{E}G_t|^2 + \mathbb{E}|\mathbb{E}G_t|^2\right)$

$= \mathcal{L}_{\mathcal{C}}(\boldsymbol{\theta}_C^t) - \alpha\langle \nabla_{\boldsymbol{\theta}_C}\mathcal{L}_{\mathcal{C}(\boldsymbol{\theta}_C)}, \mathbb{E}(G_t)\rangle + \frac{\alpha^2 A}{2}\left(\mathbb{E}|N_t|^2 + \mathbb{E}|\mathcal{L}_{\mathcal{C}}(\boldsymbol{\theta}_C) + B_t|^2\right)$

$\leq \mathcal{L}_{\mathcal{C}}(\boldsymbol{\theta}_C^t) - \alpha\langle \nabla_{\boldsymbol{\theta}_C}\mathcal{L}_{\mathcal{C}(\boldsymbol{\theta}_C)}, \mathbb{E}(G_t)\rangle + \frac{\alpha^2 A}{2}\left((D+1)\mathbb{E}|\mathcal{L}_{\mathcal{C}}(\boldsymbol{\theta}_C) + B_t|^2 + \rho^2\right)$

$\mathbb{E}[\mathcal{L}_{\mathcal{C}}(\boldsymbol{\theta}_C^{t+1})] \leq \mathcal{L}_{\mathcal{C}}(\boldsymbol{\theta}_C^t) + \frac{\alpha}{2}\left(-2\langle \nabla_{\boldsymbol{\theta}_C}\mathcal{L}_{\mathcal{C}(\boldsymbol{\theta}_C)}, \mathbb{E}(G_t)\rangle + |\nabla_{\boldsymbol{\theta}_C}\mathcal{L}_C(\boldsymbol{\theta}_C) + B_t|^2\right) + \frac{A\alpha^2\rho^2}{2}$

$= \mathcal{L}_{\mathcal{C}}(\boldsymbol{\theta}_C^t) + \frac{\alpha}{2}(-|\nabla_{\boldsymbol{\theta}_C}\mathcal{L}_C(\boldsymbol{\theta}_C)|^2 + |B_t|^2) + \frac{A\alpha^2\rho^2}{2}$

$\leq \mathcal{L}_{\mathcal{C}}(\boldsymbol{\theta}_C^t) + \frac{\alpha}{2}(d-1)|\nabla_{\boldsymbol{\theta}_C}\mathcal{L}_C(\boldsymbol{\theta}_C)|^2 + \frac{A\alpha^2\rho^2}{2}$ □

Theorem 12.5. *With the assumption 3, the convergence error of attacker loss function can be estimated as the following:*

$L_C^T \leq (1 - \alpha\omega(1-d))^T L_C^0 + \frac{\tau^2 + A\alpha\rho^2}{\omega(1-d)}$

Proof. We define $L_C^t = \mathcal{L}_{\mathcal{C}}(\boldsymbol{\theta}_C^t) - \mathcal{L}_C^*$. Then, we apply the assumption 1 and got the following:

$L_C^{t+1} = (1 - \alpha\omega(1-d))L_C^t + \frac{\alpha}{2}\tau^2 + \frac{\alpha^2 A\rho^2}{2}$

We set a constant $\kappa = \frac{\tau^2 + A\alpha\rho^2}{\omega(1-d)}$ and the above equation can be rearranged as the following inequality.

$L_C^T - \kappa \leq (1 - \alpha\omega(1-d))^T(L_C^0 - \kappa)$

Therefore, $L_C^T \leq (1 - \alpha\omega(1-d))^T L_C^0 + \frac{\tau^2 + A\alpha\rho^2}{\omega(1-d)}$ □

Remark After analyzing the theoretical aspects, it becomes evident that the accumulation of gradient estimation errors leads to a deviation of the final estimation error $L_C^T := \mathcal{L}_{\mathcal{C}}(\boldsymbol{\theta}_C^T) - \mathcal{L}_{\mathcal{C}}{}^*$ from the ground truth. The first term in the above inequality $(1 - \alpha\omega(1 - d))^T L_C^0 \to 0$. This deviation occurs due to the increase in $\frac{\tau^2 + A\alpha\rho^2}{\omega(1-d)}$ caused by higher gradient bias τ and gradient randomness ρ. Consequently, the model extraction attacker's extracted model diverges from the optimal stolen model.