Toward Dataset Distillation for Regression Problems

Jamie Mahowald [*1] **Ravi Srinivasan** [1] **Atlas Wang** [1]

Abstract

Dataset distillation is a growing technique that compresses large datasets into smaller synthetic datasets while preserving learning characteristics. However, it remains under-studied for regression problems. This paper presents a theoretical framework for regression dataset distillation using bilevel optimization, where inner loops optimize model parameters on distilled data, while the outer loops refine the dataset itself. For regularized linear regression, we derive closed-form solutions and show approximation guarantees when the number of features is greater than the size of the distilled dataset, using Polyak-Łojasiewicz properties to yield linear rates. Numerical experiments support our predictions with high determination, validating our theory while reducing dataset size by an order of magnitude.

1. Introduction

Dataset distillation is an algorithm in machine learning that uses a given model to compress a large dataset into a smaller, more efficient dataset that retains the essential information of the original dataset. Traditional compression methods decide what to compress based on properties of the data itself – spectral properties, linear transformations of arbitrary vectors, or norm-preservation, to name a few. These methods and the properties they preserve are general, and they usually suffice for a wide range of tasks. Distillation, however, leverages the model's learning process to identify and preserve the features most relevant *to the task of training the model*, producing a dataset tailored to that task.

This aim can be formalized as follows: Let $\mathbf{x}$ be an existing dataset, and let $f_{\theta(\mathbf{x})}$ a model known to perform well when trained on that dataset. The distillation process generates a synthetic dataset $\tilde{\mathbf{x}}$, whose elements are not elements of $\mathbf{x}$, but rather objects synthesized from $\mathbf{x}$ through some iterative training process. The size of this dataset $|\tilde{\mathbf{x}}| = m$ is much smaller than $|\mathbf{x}| = n$. The process is successful if f performs similarly when trained on this smaller dataset $\tilde{\mathbf{x}}$ as it does when trained on $\mathbf{x}$ at the task of evaluating $\mathbf{x}$-like testing data: symbolically, if $f_{\theta(\tilde{\mathbf{x}})}(\mathbf{x}) \approx f_{\theta(\mathbf{x})}(\mathbf{x})$. What is extraordinary about this process is that, through the distilled data, the model can also tell us what about the original data it is paying attention to.

There are several species of dataset distillation defined according to their objective (Ruonan Yu et al., 2023). The most relevant to this paper is **performance matching**, which aims to synthesize a dataset $\tilde{\mathbf{x}}$ such that a neural network $f_{\theta(\tilde{\mathbf{x}})}$ trained on $\tilde{\mathbf{x}}$ achieves *minimal loss* when evaluated on the original dataset $\mathbf{x}$, usually with the loss function ℓ defined in the original model architecture. Here $\mathcal{L}$ depends on $\ell(\mathbf{x}; \theta(\tilde{\mathbf{x}}))$, the model's loss when evaluating the original dataset on the parameters trained by the synthesized dataset.

Distillation has chiefly been used to synthesize datasets that train models for classification problems, with validation on datasets like CIFAR, MNIST, and more recently ImageNet. Furthermore, implementations of distillation create and save the *entire training trajectory*, rather than simply a final dataset, undermining the practicality of easy storing and sharing. To the extent that it has been used for regression problems, it is usually to recast a classification problem as a regression problem to exploit certain properties, like the approximation of gradient descent training by kernel ridge regression. Regresison problems, however, are useful in their own right: emerging fields such as scientific machine learning and time-series forecasting rely explicitly on these settings. Distillation should be leveraged to relieve issues like interpretability and training costs for these areas, just as it has in mainstream ML settings. Recognizing that a robust distillation theory for these practical

[*]Equal contribution [1]The University of Texas at Austin, Austin, TX, USA. Correspondence to: Jamie Mahowald <mahowald.jamie@gmail.com>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

fields requires a strong foundation, we build out properties of dataset distillation for a basic regression task.

1.1. Analyzing the algorithm

Bilevel optimization. The structure of this algorithm (see Algorithm 1) follows a bilevel optimization problem. The "inner loop" (lines 7-9) optimizes a weight matrix to the distilled data, and the "outer loop" (lines 3-14) optimizes the distilled data to the original data via the weight matrix. Symbolically, given an original dataset $\mathbf{x}$, the algorithm creates

$$\tilde{\mathbf{x}} = \arg\min_{\tilde{\mathbf{x}}} \mathcal{L}_o(\mathbf{x}, \theta^*(\tilde{\mathbf{x}})) \quad s.t. \quad \theta^*(\tilde{\mathbf{x}}) \in \arg\min_{\theta} \mathcal{L}_i(\theta, \tilde{\mathbf{x}}), \tag{1}$$

where $\mathcal{L}_o$ and $\mathcal{L}_i$ are outer and inner loss functions, respectively. The distilled dataset itself functions as a learnable parameter matrix, similar to how neural network weights are learnable parameters. In line 2, we randomly initialize this distilled dataset matrix, before optimizing it to become the most effective small dataset for training. This initialization approach mirrors how we would randomly initialize weight matrices in a neural network before training them toward their optimal values.

Rather than use the entire original dataset for each distillation step, we sample representative batches to guide the optimization process. To begin an outer loop, we sample q model weights from some distribution $p(\theta_0)$. We use several initializations per distillation step to ensure that the distilled dataset is robust to the choice of initialization. Within the inner loop, the distilled dataset is held fixed; it is only updated at the end of the distillation step (lines 12-13).

Loop structures. Both loops run optimization via gradient descent on their respective loss functions.* In the inner loop, we run T gradient descent steps on the distilled dataset for each weight matrix, updating the weights according to the loss function. This is intended to reflect a mini-training run on the distilled data. In the outer loop, this trained mini-network is evaluated on the original dataset, with its loss accumulated. The distilled dataset is then updated according to the gradient of the accumulated loss with respect to the distilled data. This is analogous to how a weight matrix is updated according to the gradient of the loss with respect to the weights.

In the first few distillation steps, θ_T is expected to fit the randomly initialized distilled data and therefore perform poorly on real data, and the initial updates to the distilled data should be large. However, as the algorithm "fills in" the distilled data according to line 12, the distilled data should train weights that perform better and better on the original dataset. We also allow the learning rate to update alongside the distilled data, but that analysis is not discussed in depth here.

2. Regression

The regression problem is a classic data-science task that aims to learn a mapping from input features to a continuous space of target values. In this section, we discuss how dataset distillation can be applied to regression problems, focusing on the theoretical basis of the distillation process and its convergence properties. The goal will be to establish that a model trained on a distilled dataset can approximate a model trained on the original dataset. To begin analysis of regression problems, we study linear regression, including regularized and kernel-ridge regression. We leave nonlinear analysis for future work.

2.1. Problem setup

We adopt the following setup from (Wang et al., 2020)'s motivation for dataset distillation: consider a linear regression problem defined by a dataset $\mathbf{x}$ composed of n input-target pairs, $\mathbf{x} = \{(x_i, y_i)\}_{i=1}^n$, where each input $x_i \in \mathbb{R}^{d_{\text{in}}}$ and target $y_i \in \mathbb{R}^{d_{\text{out}}}$ for some positive integers d_{in} and d_{out} (for now, we consider one-dimensional targets, or $d_{\text{out}} = 1$, and denote $d = d_{\text{in}}$). We assume $n > d$, i.e., there are more samples than features per sample. We represent $\mathbf{x}$ by the data matrices $X = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}^\top \in \mathbb{R}^{n \times d}, Y = \begin{bmatrix} y_1 & \cdots & y_n \end{bmatrix} \in \mathbb{R}^n, \mathbf{x} = (X, Y)$. We want to regress a function $f : \mathbb{R}^d \to \mathbb{R}$ such that $f(x_i) \approx y_i$ for all i.

Regression loss. Let Φ be a feature transformation on the inputs X. We use the regularized mean-squared error to measure how well a linear prediction $f_\theta(x_i) = \Phi(x_i)\ \theta$ parameterized by a d-dimensional weight matrix θ fits the dataset:

$$\ell(\mathbf{x}, \theta) = \ell((X, Y), \theta) = \frac{1}{2n} ||\Phi(X)\ \theta - Y||^2 + \frac{\lambda}{2} ||\theta||^2 \tag{2}$$

*Some newer works are analyzing non-GD methods, particularly for outer-loop optimization. Here we stick with gradient descent.

where $||\cdot||$ is the Euclidean norm for vectors and the Frobenius norm for matrices, and the regularization term penalizes large model weights. The simplest variant assumes $\lambda = 0$ and $\Phi(x) = x$, giving a traditional, unregularized regression problem. This loss function will be used in both lines 8 and 10 of Algorithm 1.

Distillation. Based on Algorithm 1, the distillation process builds out a dataset $\tilde{\mathbf{x}} = (\tilde{X}, \tilde{Y})$ of size $m << n$, so that $\tilde{X}$ and $\tilde{Y}$ are $m \times d$ and $m \times 1$ matrices, respectively. It's helpful to see m as a "tunable" hyperparameter in the positive integers $< n$ that has different properties before and after d. Then, a function $f_{\theta(\tilde{\mathbf{x}})}$ regressed on $\tilde{\mathbf{x}}$ should satisfy $f_{\theta(\tilde{\mathbf{x}})}(X) \approx f_{\theta(\mathbf{x})}(X) \approx Y$: that is, with $f_{\theta(\mathbf{x})}$ and $f_{\theta(\tilde{\mathbf{x}})}$ trained respectively by $\theta(\mathbf{x})$ and $\theta(\tilde{\mathbf{x}})$, we'd have $\ell(\mathbf{x}, \theta(\tilde{\mathbf{x}})) \approx \ell(\mathbf{x}, \theta(\mathbf{x})) <$ some small error.

2.2. Inner-loop convergence

We begin by analyzing inner-loop performance before turning to the outer loop.

Definition 2.1 (Inner-loop loss function)**.** Let $\tilde{\mathbf{x}}$ be a distilled dataset of size m with inputs $\tilde{X} \in \mathbb{R}^{m \times d}$ and outputs $\tilde{Y} \in \mathbb{R}^d$. We define the regularized, linear, *mean-squared error inner-loop loss function* as

$$\ell(\tilde{\mathbf{x}}, \theta) = \frac{1}{2m}||\tilde{X}\theta - \tilde{Y}||^2 + \frac{\lambda_I}{2}||\theta||^2.$$

Definition 2.2 (Inner-loop gradient descent step)**.** Let $\tilde{\eta}$ be a distillation learning rate hyperparameter, and let $\tilde{\mathbf{x}}$ be a distilled dataset at any step. Let $\lambda_I \geq 0$ be an inner-loop regularization parameter. The *inner-loop gradient descent (I-GD) step* is defined as the update of the weights θ_t at step t given the distilled dataset $\tilde{\mathbf{x}}$:

$$\theta_{t+1} \leftarrow \theta_t - \tilde{\eta}\nabla_{\theta_t}\ell(\tilde{\mathbf{x}}, \theta_t) \quad \text{to minimize} \quad \ell(\tilde{\mathbf{x}}, \theta). \tag{3}$$

Following A.4, this gives an alternating, combinatorial, closed-form expression for θ_t in terms of the distilled data $\tilde{\mathbf{x}}$ and inner-loop hyperparameters $\beta = 1 - \tilde{\eta}\lambda_I$ and $\gamma = \tilde{\eta}/m$:

$$\theta_t = \left(\underbrace{\beta I - \gamma \tilde{X}^\top \tilde{X}}_{=:C}\right)^t \theta_0 + \underbrace{\sum_{j=0}^{t-1} (-1)^j \binom{t}{j} \beta^{t-j} \left(\gamma \tilde{X}^\top \tilde{X}\right)^j \tilde{X}^\top \tilde{Y}}_{=:D_t}. \tag{4}$$

Exact performance matching. In preparation for our weaker convergence conditions, we define a necessary condition for the distilled dataset to achieve the *exact* same performance as the original dataset.

Lemma 2.3 (Exact inner-loop equality)**.** *For any amount of I-GD steps, $X\theta_t = Y$ exactly only if $m \geq d$.*

Sketch of proof. For $X(C^t\theta_0 + D_t) = Y$, we have two constraints:

- $XC^t\theta_0 = 0$. This point is necessary for the equality to hold *for any* θ_0. If instead we allowed $XC\theta_0 + XD_t = Y$, for fixed C and D_t and variable θ_0, we would still need $XC = 0$, which again forces $C = 0$ when X is full rank. Substituting, $X\left(\beta I - \gamma \tilde{X}^\top \tilde{X}\right)^t \theta_0 = \mathbf{0}$.

  Just as in the one-I-GD-step case, this equality is satisfied only when $I = \frac{\gamma}{\beta}\tilde{X}^\top \tilde{X}$ (with a full-rank X). However, as shown in Sec 2.2, the exponentiation tends to push C^t toward zero.

- $XD_t = Y$. This term represents the target alignment achieved through the distillation process over several runs of the outer loop.

The first constraint forces $m \geq d$ as a necessary condition for any distilled dataset of size m to achieve the *exact same* performance as the original. Note that $\tilde{X}^\top \tilde{X}$ is positive semi-definite with k eigenvalues $\sigma_1^2, \sigma_2^2, \ldots, \sigma_k^2$, all ≥ 0. If $m \geq d$, then, assuming the number of data points n is greater than the dimensionality d, since $\tilde{X}$ is full rank,

$$k = \text{rank}(\tilde{X}^\top \tilde{X}) = \text{rank}(\tilde{X}) = d = \text{rank}(X),$$

regardless of the number of I-GD steps.

If $m < d$, these equalities no longer hold, and $\text{rank}(\tilde{X}^\top \tilde{X})$ is guaranteed only to be some positive integer $k \leq m$.

Approximate performance matching. If we relax the call for perfection, and we instead study the convergence in loss by distilled data toward by using the original data, we start seeing benefits that don't depend on rank when running multiple I-GD steps on a distilled dataset.

Lemma 2.4 (Inner-loop convergence by I-GD)**.** *Let $\{\sigma_i^2\}$ be the set of eigenvalues of $\tilde{X}^\top \tilde{X}$, and let $\tilde{\eta} < 2m/(\sigma_{\max}^2 + m\lambda_I)$. As t increases, the inner-loop prediction $\tilde{X}\theta$ approaches $\tilde{X}D_t$.*

Proof. With $\tilde{X}\theta = \tilde{X}(C^t\theta_0 + D_t)$, the eigenvalues of $C = \beta I - \gamma \tilde{X}^\top \tilde{X}$ are $\{\beta - \gamma\sigma_i^2\}$ for $i = 1, \ldots, k$. For C^t to converge to $\mathbf{0}$ as t increases, all eigenvalues must be less than 1 in absolute value, so that $|\beta - \gamma\sigma_i^2| < 1$, or $0 < \gamma\sigma_i^2 + \tilde{\eta}\lambda_I < 2$ for all i. Thus, if we choose $\tilde{\eta} < 2m/(\sigma_i^2 + m\lambda_I)$ for all i — or equivalently

$$\tilde{\eta} < 2m/(\sigma_{\max}^2 + m\lambda_I), \tag{5}$$

where $\sigma_{\max} = \max_i \{\sigma_i\}$ — then $\gamma = \tilde{\eta}/m$ will "push" all eigenvalues of C below β.

With this condition satisfied, the relationship between matrix powers and their spectral norms guarantees $||C^t|| \leq \rho^t$, where $\rho = \max_i |\beta - \gamma\sigma_i^2| < 1$. This relation implies that C^t decreases exponentially with t to 0, so that $X(C^t + D_t) \to XD_t$. □

While a positive regularization term λ_I reduces the range of admissible learning rates $\tilde{\eta}$, it also contracts the spectral radius of C: the regularized radius $|\beta - \gamma\sigma_i^2| = |1 - \gamma\sigma_i^2 - \tilde{\eta}\lambda_I|$ is smaller than the unregularized radius $|1 - \gamma\sigma_i^2|$ for all i. The hyperparameter λ_I should therefore be tuned to balance this tradeoff.

2.3. Outer-loop convergence

Structure of the outer loop. At this point, we consider the outer loop and note that $\tilde{\mathbf{x}}$ is updated only once per distillation step (in line 12 of Algorithm 1). While we had used an arbitrary distilled dataset $\tilde{\mathbf{x}}$ for the inner loop, we now name the distilled dataset at the s^{th} distillation step $\tilde{\mathbf{x}}_s = (\tilde{X}_s, \tilde{Y}_s)$.

Within the s^{th} distillation step, for each network initialization $q = 1, \ldots, Q$, a run of T I-GD steps gives us a set of fixed parameters $\left\{\theta_{T,s}^{(q)}\right\}_{q=1}^{Q}$, each associated with a different initialization $\theta_0^{(q)}$ but with the same distilled dataset $\tilde{\mathbf{x}}_{s-1}$. We then compute the outer-loop loss $\mathcal{L}_s^{(q)}$ of those parameters on the *original data* $\mathbf{x}$ to gauge how well the model trained on distilled data fits the original data. In practice, we use a representative batch $\mathbf{x}_B$ instead of the full $\mathbf{x}$, where $\|X_B\theta - Y_B\| \approx \|X\theta - Y\|$. We then update the distilled dataset $\tilde{\mathbf{x}}_s$ using the outer-loop gradient descent (O-GD) step, which minimizes the sum of the outer-loop losses across all initializations.

Definition 2.5 (Outer-loop loss function)**.** Let $\tilde{\mathbf{x}} = (X, Y)$ be a real dataset, and let $\tilde{\mathbf{x}} = (\tilde{X}, \tilde{Y})$ be a set of distilled data. Let $\theta_T = \theta_T(\tilde{\mathbf{x}})$ be a set of parameters trained on $\tilde{\mathbf{x}}$ with a run of T inner-loop gradient descent steps. Let $||\cdot||$ denote the Frobenius norm. Let λ_O be the outer-loop regularization term, a hyperparameter. The regularized, linear, mean-squared error *outer-loop loss function* is defined as

$$\mathcal{L}(\mathbf{x}, \theta_T) = \frac{1}{2n}\|X\theta_T - Y\|^2 + \frac{\lambda_O}{2}(\|\tilde{X}\|_F^2 + \|\tilde{Y}\|_F^2). \tag{6}$$

Definition 2.6 (Outer-loop gradient descent step)**.** Let $\mathcal{L}_s^{(q)}$ be the outer-loop loss function for the q^{th} network initialization at distillation step $s+1$, and let α_x be the meta-gradient learning rate. The *outer-loop gradient descent (O-GD) step* is defined as the update of the distilled dataset $\tilde{\mathbf{x}}_{s+1} = (\tilde{X}_{s+1}, \tilde{Y}_{s+1})$ at step $s+1$:

$$\tilde{X}_{s+1} \leftarrow \tilde{X}_s - \alpha_x \nabla_{\tilde{X}_s}\left(\sum_{q=1}^{Q} \mathcal{L}_s^{(q)}\right) \quad \text{and} \quad \tilde{Y}_{s+1} \leftarrow \tilde{Y}_s - \alpha_x \nabla_{\tilde{Y}_s}\left(\sum_{q=1}^{Q} \mathcal{L}_s^{(q)}\right). \tag{7}$$

The outer-loop loss depends on $\tilde{X}$ through $\theta_{T,s}^{(q)}$ and through $\frac{\lambda_O}{2}\|\tilde{X}\|_F^2$, so its gradient with respect to $\tilde{X}_s$ is given by

$$\nabla_{\tilde{X}}\mathcal{L} = \frac{\partial \mathcal{L}}{\partial \theta}\frac{\partial \theta}{\partial \tilde{X}} + \left.\frac{\partial \mathcal{L}}{\partial \tilde{X}}\right|_{\text{direct}},$$

where $\frac{\partial \mathcal{L}}{\partial \theta} = \frac{1}{n}X^\top(X\theta - Y)$, $\frac{\partial \theta}{\partial \tilde{X}} = \frac{\partial}{\partial \tilde{X}}[C^T\theta_0 + D_T]$, and $\frac{\partial \mathcal{L}}{\partial \tilde{X}}|_{\text{direct}} = \lambda_O \tilde{X}$.

Linear convergence of $\mathcal{L}$. We aim to find conditions under which $\tilde{\mathbf{x}}$ converges to a minimizing $\tilde{\mathbf{x}}^*$ under gradient descent. At a minimum, exploiting the properties of the PL condition, we can show that if the step size is small enough, then the outer-loop loss function converges linearly to a local minimum. We first state two lemmas for general functions, whose proofs are found in Appendix A.6:

Lemma 2.7 (Local strong convexity)**.** *Consider the function $f = Q + R$, where Q is a polynomial and $R(x) = \frac{\lambda}{2}||x||^2$ with regularization parameter λ. Let $\lambda_{\min}^{(Q)}$ be the smallest eigenvalue of $\nabla^2 Q$, the Hessian of Q, and let y be some point in the same domain as x.*

Suppose that for some $M \in [0, \lambda)$ and $r > 0$, we have that $\lambda_{\min}^{(Q)} \geq -M$ for all x in the r-neighborhood of y. Then there exists $\mu > 0$, namely $\lambda - M$, such that f is locally μ-strongly convex in $\{x : ||x - y|| \leq r\}$.

Lemma 2.8 (Outer-loop PL condition)**.** *Let x^* be a local minimizer of f. Under the conditions of Lemma 2.7, for all x with $||x - x^*|| \leq r$, the outer-loop objective $f(x) = Q(x) + R(x)$ satisfies the PL inequality,*

$$\frac{1}{2}\|\nabla f(x)\|^2 \geq \mu(f(x) - f(x^*)), \tag{8}$$

where $\mu = \lambda - M > 0$.

Consider the outer-loop loss function

$$\mathcal{L}(\tilde{X}) = Q(\tilde{X}) + R(\tilde{X}), \quad \text{where} \quad Q(x) = \frac{1}{2n}\|X\theta_T - Y\|^2 \quad \text{and} \quad R(x) = \frac{\lambda_O}{2}(\|x\|_F^2 + \|\tilde{Y}\|_F^2).$$

Note that $\theta_T \approx D_T$ for large T, which is polynomial in x with degree $2T - 1$ (see Lemma 2.4), so that $Q(x)$ is polynomial in x with degree $4T - 2$.

Theorem 2.9 (Linear convergence of outer-loop)**.** *Consider the outer-loop gradient descent update given in Defintion 2.6. Let L be the Lipschitz constant of the gradient of $\mathcal{L}$ with respect to $\tilde{X}$. Let $\tilde{X}^*$ be a local minimizer of $\mathcal{L}$.*

Suppose $\mathcal{L}$ satisfies the conditions of Lemma 2.8, with M the minimum eigenvalue of the Hessian of $\mathcal{L}$ in a region of radius r around $\tilde{X}^*$. If the step size satisfies $\alpha_x \leq 1/L$, then for all $\tilde{X}^*$ with $||\tilde{X}_s - \tilde{X}^*||_F \leq r$,

$$\mathcal{L}(\tilde{X}_s) - \mathcal{L}(x^*) \leq \left(1 - \frac{\lambda_O - M}{L}\right)^s (\mathcal{L}(\tilde{X}_0) - \mathcal{L}(x^*)), \tag{9}$$

Proof. This follows directly from the standard convergence result for gradient descent under the PL condition (Karimi et al., 2020). Since $\mathcal{L}$ satisfies the PL inequality with constant μ and has L-Lipschitz gradient, gradient descent with step size $\alpha_x = 1/L$ achieves the stated linear convergence rate. □

From Theorem 2.9, we get a testable prediction: during the main training run (i.e., excluding warmup and cooldown), plotting $\mathcal{L}_s = \ell(\mathbf{x}, \theta_{T,s})$ on a log-scale against s should produce a straight line whose slope is $\log(1 - \alpha_x \mu_T)$. Below, we show several parameterizations of the distillation process along with their empirical $\mu_T = (1 - e^{\text{slope}})/\alpha_x$ values, visualized in A.7. In this setting, $\alpha_x = 0.0025$, $\tilde{\eta}_0 = 0.05$, $b = 32$, and $\{d, m, n\} = \{1000, 100, 99\}$.

Inner-loop steps T	**100**	**125**	**150**	**175**
Slope of log fit	-0.0052 ± 0.0005	-0.0066 ± 0.0006	-0.0073 ± 0.0007	-0.0103 ± 0.0009
R^2 **of log fit**	0.9974	0.9972	0.9973	0.9967
Empirical μ_T	2.087 ± 0.182	2.639 ± 0.252	2.914 ± 0.279	4.097 ± 0.359

As shown, larger inner-loop step counts T lead to faster outer-loop convergence as measured by the empirical PL constant μ_T. A higher μ_T corresponds to a sharper landscape, suggesting faster reduction of suboptimality. This suggests using higher T values as budget permits, especially for regimes demanding sharper generalization or fast convergence.

3. Conclusion

Using the bilevel optimization structure of dataset distillation, we show how distillation operates in the context of regularized regression. Under reasonable assumptions, the outer objective satisfies a Polyak-Łojasiewicz condition inherited from the

base loss, from which we obtain linear convergence rates for gradient-based distillation under mild smoothness. These results position distilled datasets as a principled, hardware-agnostic alternative to data-subset selection for rapid training. We can also begin to apply regression to compute-heavy areas of machine learning that are based on regression, like time series forecasting and scientific machine learning.

In future work, we aim to answer two open questions. First, we intend to relate the admissible range around the optimal solution to hyperparameters like T, m, the choice of initialization. For instance, since T exponentiates in the inner-loop weights, we expect that larger T will produce an ill-conditioned Hessian of $\mathcal{L}(\theta)$, introducing a tradeoff between inner-loop performance and outer-loop convergence that should be optimized.

Second, we aim to convert the regularized *linear* regression problem into a regularized *kernel* regression problem, otherwise known as kernel ridge regression. This can be done by replacing $\tilde{X}$ in the inner loop with $\Phi(\tilde{X})$ for some feature transformation Φ, to induce the kernel $k(\tilde{X}, \tilde{X}') = \langle \Phi(\tilde{X}), \Phi(\tilde{X}') \rangle$. Recent work has shown that, as the width of a neural network grows, training the network with gradient descent becomes equivalent to kernel ridge regression with the neural tangent kernel (NTK) (Jacot et al., 2020). This work will allow us to extend the theory on regression-based distillation to any gradient-descent problem that can be adequately approximated by a kernel ridge regression problem, such as classification and generative modeling.

References

Chen, Y., Huang, W., and Weng, T.-W. Provable and efficient dataset distillation for kernel ridge regression. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=WI2VpcBdnd.

Dong, T., Zhao, B., and Lyu, L. Privacy for free: How does dataset condensation help privacy?, 2022. URL https://arxiv.org/abs/2206.00240.

Goetz, J. and Tewari, A. Federated learning via synthetic data, 2020. URL https://arxiv.org/abs/2008.04489.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Ivakhnenko, A. G., Lapa, V. G., and McDonough, R. N. Cybernetics and forecasting techniques, 1967. URL https://api.semanticscholar.org/CorpusID:60378835.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks, 2020. URL https://arxiv.org/abs/1806.07572.

Karimi, H., Nutini, J., and Schmidt, M. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition, 2020. URL https://arxiv.org/abs/1608.04636.

Lee, D. B., Lee, S., Ko, J., Kawaguchi, K., Lee, J., and Hwang, S. J. Self-supervised dataset distillation for transfer learning, 2024. URL https://arxiv.org/abs/2310.06511.

Li, G., Zhao, B., and Wang, T. Awesome dataset distillation. https://github.com/Guang000/Awesome-Dataset-Distillation.

Light, J., Liu, Y., and Hu, Z. Dataset distillation for offline reinforcement learning, 2024. URL https://arxiv.org/abs/2407.20299.

Loo, N., Hasani, R., Lechner, M., and Rus, D. Dataset distillation with convexified implicit gradients, 2023. URL https://arxiv.org/abs/2302.06755.

Moser, B. B., Raue, F., Palacio, S., Frolov, S., and Dengel, A. Latent dataset distillation with diffusion models, 2024. URL https://arxiv.org/abs/2403.03881.

Nguyen, T., Chen, Z., and Lee, J. Dataset meta-learning from kernel ridge-regression, 2021. URL https://arxiv.org/abs/2011.00050.

Ruonan Yu, Songhua Liu, and Xinchao Wang. Dataset Distillation: A Comprehensive Review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, October 2023. URL https://arxiv.org/pdf/2301.07014.

Schmidhuber, J. Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4 (2):234–242, 1992. doi: 10.1162/neco.1992.4.2.234.

Wang, T., Zhu, J.-Y., Torralba, A., and Efros, A. A. Dataset Distillation. 2020. URL https://arxiv.org/abs/1811.10959. arXiv preprint: 1811.10959.

Yang, W., Zhu, Y., Deng, Z., and Russakovsky, O. What is dataset distillation learning?, 2024. URL https://arxiv.org/abs/2406.04284.

Zhao, Y., Zhao, H., Wen, B., Ong, Y.-S., and Zhou, J. T. Video set distillation: Information diversification and temporal densification, 2024. URL https://arxiv.org/abs/2412.00111.

Zheng, T. and Li, B. Differentially private dataset condensation, 2023. URL https://openreview.net/forum?id=H8XpqEkbua_. OpenReview preprint.

Zhou, M., Yin, Z., Shao, S., and Shen, Z. Self-supervised dataset distillation: A good compression is all you need, 2024. URL https://arxiv.org/abs/2404.07976.

A. Appendix

A.1. Related work

Dataset distillation was first introduced in full by Wang et al. in 2018 (Wang et al., 2020). This paper proposed the original algorithm, analyzed it briefly on a simple linear case, and proved its effectiveness on standard CIFAR10 and MNIST classification problems, showing that the latter can be compressed into just one image per class. Their idea builds on **knowledge distillation** – the transfer of "knowledge," broadly defined, from larger to smaller models.

Ivakhnenko and Lapa originated this concept in 1965 (Ivakhnenko et al., 1967), Schmidhuber introduced it to neural networks in 1992 (Schmidhuber, 1992), and Hinton et al. formalized it (Hinton et al., 2015) in 2015.

Since (Wang et al., 2020), hundreds of papers have been published on all stages of distillation. Particularly active areas include **kernel-based regression**, including techniques in ridge regression (Chen et al., 2024; Nguyen et al., 2021) and convexified implicit gradients (Loo et al., 2023). These exploit the ease of analyzing kernel methods against deep neural networks to derive provable guarantees for gradient computation. Other areas include new frameworks – like **self-supervised distillation** (Zhou et al., 2024; Lee et al., 2024) and **reinforcement learning-based distillation** (Light et al., 2024) – and distillation based on **generative models** (Moser et al., 2024).

Areas of application that take advantage of shorter processing times for large amounts of data include **video**, **super-resolution**, **medicine**, **domain adaptation**, and **text** (specifically natural language). For instance, Zhao et al. (Zhao et al., 2024) exploit redundancies in video data, as informed by the model, to reduce the size of the data while retaining its action. Other areas exploit interesting properties of dataset distillation, like **security guarantees** (Dong et al., 2022), **mechanistic interpretability** (Yang et al., 2024), and **federated learning** (Goetz & Tewari, 2020). **Privacy** researchers in particular have noted that the distillation process can obscure or abstract the individually identifiable features of input data while allowing the model to remain performant (Zheng & Li, 2023). This concept is related **differential privacy**, a mathematically rigorous approach that allows training on dataset distributions while concealing any one individual's input. Differential privacy ensures that removing any single data point does not significantly alter the overall training distribution, preventing the revelation of individual features.

Lastly, we owe a huge debt to Awesome-Dataset-Distillation (Li et al.), a superbly maintained GitHub repository that collects and classifies almost every paper written on distillation to date, including many that aren't cited in this paper. We also owe much of the background research to Yu et al.'s 2023 survey (Ruonan Yu et al., 2023).

A.2. Index of symbols

- Original data: $\mathbf{x} = (X \in \mathbb{R}^{n\times d}, Y \in \mathbb{R}^n)$ given.
- Inner- and outer-loop regularization: λ_I, λ_O given.
- Learning rates: $\tilde{\eta}$, α_x, α_η given.
- Initial weights: $\theta_0 \in \mathbb{R}^d$ sampled.
- Distilled data: $\tilde{X}_0 \in \mathbb{R}^{m\times d}$, $\tilde{Y}_0 \in \mathbb{R}^m$ sampled.
- Inner loss: $\ell(\tilde{\mathbf{x}}, \theta) = \frac{1}{2m}||\tilde{X}\theta - \tilde{Y}||_2^2 + \frac{\lambda_I}{2}||\theta||^2$.
- Inner hyperparameters: $\gamma = \tilde{\eta}/m, \quad \beta = 1 - \tilde{\eta}\lambda_I$.
- Inner components: $C_s^t = \beta I - \gamma \tilde{X}_s^\top \tilde{X}_s, \quad D_{t,s} = \sum_{j=0}^{t-1}(-1)^j \binom{t}{j} \beta^{t-j} \left(\gamma \tilde{X}_s^\top \tilde{X}_s\right)^j \tilde{X}_s^\top \tilde{Y}_s$.
- Inner gradient descent: $\theta_{t,s} = \theta_{t-1,s} - \tilde{\eta}\nabla_{\theta_{t-1,s}} \ell(\tilde{\mathbf{x}}, \theta_{t-1}) = C_s^t \theta_0 + D_{t,s}$.
- Outer loss: $\mathcal{L}_s^{(q)} = \ell(\mathbf{x}, \theta_{T,s}^{(q)}) = \frac{1}{2n}||X\theta_{T,s}^{(q)} - Y||_2^2 + \frac{\lambda_O}{2}(||\tilde{X}||_F^2 + ||\tilde{Y}||^2)$.
- Outer gradient descent: $\tilde{X}_{s+1} = \tilde{X}_s - \alpha_x \nabla_{\tilde{X}_s}(\sum_{q=1}^Q \mathcal{L}_s^{(q)}), \quad \tilde{Y}_{s+1} = \tilde{Y}_s - \alpha_y \nabla_{\tilde{Y}_s}(\sum_{q=1}^Q \mathcal{L}_s^{(q)})$.

A.3. Algorithm

Algorithm 1 Dataset Distillation

Require: Original dataset: $\mathbf{x} = (X \in \mathbb{R}^{n \times d_{in}}, Y \in \mathbb{R}^{n \times d_{out}})$; $p(\theta_0)$: Distribution of initial weights; m: Number of synthetic examples to create; b: Batch size for real data

Require: $\tilde{\eta}_0$: Initial distillation learning rate; α_x: Learning rate for distilled data; α_η: Learning rate for the distillation learning rate; $\ell(x, \theta)$: Loss function;

Require: S: Number of distillation steps; Q: Number of network initializations per distillation step; T: Number of inner-loop gradient descent steps per network.

Set $\tilde{\eta} \leftarrow \tilde{\eta}_0$
Initialize distilled set $\tilde{\mathbf{x}}_0 = (\tilde{X}_0 \in \mathbb{R}^{m \times d_{\text{in}}}, \tilde{Y}_0 \in \mathbb{R}^{m \times d_{\text{out}}})$ randomly.
for outer gradient-descent step $s = 1, \ldots, S$ **do**
  Sample batch $\mathbf{x}_B = (X_B, Y_B)$ of b input-target pairs from original dataset.
  Sample q initial network parameters $\{\theta_0^{(q)}\}_{q=1}^{Q}$ from $p(\theta_0)$.
  for $q = 1, \ldots, Q$ **do**
    for inner gradient-descent step $t = 1, \ldots, T$ **do**
      Update weights on inner loss: $\theta_t^{(q)} = \theta_{t-1}^{(q)} - \tilde{\eta} \nabla_{\theta_{t-1}^{(q)}} \ell(\tilde{\mathbf{x}}, \theta_{t-1}^{(q)})$.
    end for
    Compute outer loss: $\mathcal{L}_s^{(q)} = \ell(\mathbf{x}_B, \theta_T^{(q)})$.
  end for
  Update distilled data: $\tilde{\mathbf{x}}_s \leftarrow \tilde{\mathbf{x}}_{s-1} - \alpha_x \nabla_{\tilde{\mathbf{x}}_{s-1}} \left(\sum_q \mathcal{L}_s^{(q)} \right)$.
  Update learning rate: $\tilde{\eta}_s \leftarrow \tilde{\eta}_{s-1} - \alpha_\eta \nabla_{\tilde{\eta}_{s-1}} \left(\sum_q \mathcal{L}_s^{(q)} \right)$.
end for
return $\tilde{\mathbf{x}}, \tilde{\eta}$

A.4. Inner-loop gradients

At this point, we're looking for a closed-form expression of the weight matrix θ_t after t I-GD steps within a run of the inner loop (remember that $\tilde{X}$ does not update until the *end* of the inner loop, after all these gradients have been calculated – thus in the first run of the outer loop, these gradients are calculated using the randomly initialized $\tilde{X}_0$ and $\tilde{Y}_0$). The updated matrix after a single I-GD step becomes

$$\theta_1 = \theta_0 - \tilde{\eta} \nabla_{\theta_0} \ell(\tilde{\mathbf{x}}, \theta_0) = \theta_0 - \frac{\tilde{\eta}}{m} \tilde{X}^\top (\tilde{X} \theta_0 - \tilde{Y}) + \lambda \theta_0 = \left(I(1 - \tilde{\eta}\lambda) - \frac{\tilde{\eta}}{m} \tilde{X}^\top \tilde{X} \right) \theta_0 + \frac{\tilde{\eta}}{m} \tilde{X}^\top \tilde{Y}.$$

This relation generalizes to

$$\theta_t = \left(I(1 - \tilde{\eta}\lambda) - \frac{\tilde{\eta}}{m} \tilde{X}^\top \tilde{X} \right) \theta_{t-1} + \frac{\tilde{\eta}}{m} \tilde{X}^\top \tilde{Y}.$$

To simplify notation, we define $\gamma = \eta/m$, $\alpha = 1 - \tilde{\eta}\lambda$, $M = (\alpha I - \gamma A)$, $A = \tilde{X}^\top \tilde{X}$, and $B = \tilde{X}^\top \tilde{Y}$. Then,

$$\begin{aligned}
\theta_1 &= M\theta_0 + \gamma B, \\
\theta_2 &= M\theta_1 + \gamma B \\
&= M[M\theta_0 + \gamma B] + \gamma B \\
&= M^2\theta_0 + (M + I)\gamma B, \\
\theta_3 &= M\theta_2 + \gamma B \\
&= M^3\theta_0 + (M^2 + M + I)\gamma B \\
&\vdots \\
\theta_t &= M^t\theta_0 + \gamma \sum_{k=0}^{t-1} M^k B \\
&= (\alpha I - \gamma A)^t \theta_0 + \sum_{k=0}^{t-1} \binom{t}{j} \alpha^{t-j} (-\gamma A)^j B
\end{aligned}$$

A.5. Outer-loop gradients

By the chain rule,

$$\begin{aligned}
\frac{\partial ||X\theta_T^{(q)} - Y||^2}{\partial \tilde{X}} &= \underbrace{\frac{\partial ||X\theta_T^{(q)} - Y||^2}{\partial \theta_T^{(q)}}}_{\text{(I)}} \cdot \underbrace{\frac{\partial \theta_T^{(q)}}{\partial \tilde{X}}}_{\text{(II)}} + \frac{\lambda_O}{2} \frac{\partial ||\tilde{X}||_F^2}{\partial \tilde{X}} \\
\frac{\partial ||X\theta_T^{(q)} - Y||^2}{\partial \tilde{Y}} &= \underbrace{\frac{\partial ||X\theta_T^{(q)} - Y||^2}{\partial \theta_T^{(q)}}}_{\text{(I)}} \cdot \underbrace{\frac{\partial \theta_T^{(q)}}{\partial \tilde{Y}}}_{\text{(III)}} + \frac{\lambda_O}{2} \frac{\partial ||\tilde{Y}||_F^2}{\partial \tilde{Y}}.
\end{aligned}$$

Notice that (I) $= 2X^\top(X\theta_T^{(q)} - Y) \in \mathbb{R}^{d \times 1}$ is the same for either gradient. For (II) and (III), we first consider $T = 1$, so that $\theta_T = \theta_1 = \theta_0 - \gamma \tilde{X}^\top(\tilde{X}\theta_0 - \tilde{Y})$:

$$\frac{\partial \theta_1^{(q)}}{\partial \tilde{X}} = -\gamma \left[\frac{\partial \tilde{X}^\top}{\partial \tilde{X}} (\tilde{X}\theta_0 - \tilde{Y}) + \tilde{X}^\top \frac{\partial (\tilde{X}\theta_0)}{\partial \tilde{X}} \right], \qquad \frac{\partial \theta_1^{(q)}}{\partial \tilde{Y}} = \gamma \tilde{X}^\top.$$

Sorting out the tensor products and reintroducing step notation yields

$$\nabla_{\tilde{X}_{s-1}} \left(\sum_{q=1}^{Q} \mathcal{L}_s^{(q)} \right) = -\frac{\tilde{\eta}}{nm} \sum_{q=1}^{Q} \left[(\tilde{X}_{s-1}\theta_0^{(q)} - \tilde{Y}_{s-1})(X\theta_1^{(q)} - Y)^\top X + \tilde{X}_{s-1} X^\top (X\theta_1^{(q)} - Y) \theta_0^{(q)\top} \right], \tag{10}$$

$$\nabla_{\tilde{Y}_{s-1}} \left(\sum_{q=1}^{Q} \mathcal{L}_s^{(q)} \right) = 2\gamma \tilde{X}_s X^\top (X\theta_1 - Y). \tag{11}$$

If $T > 1$, the first part of the chain rule, (I) = $2X^\top(X\theta_T^{(q)} - Y)$ remains the same, with $\theta_T^{(q)}$ as given in (A.2). The $\tilde{Y}$ derivative is straightforward. Since C doesn't depend on $\tilde{Y}$ in the relation $\theta_T = C^T\theta_0 + D_{T,s}$, we can write

$$\frac{\partial \theta_T^{(q)}}{\partial \tilde{Y}} = \frac{\partial D_{T,s}}{\partial \tilde{Y}} = \sum_{i=1}^{T} \binom{T}{j} \gamma_s^j (-\tilde{X}_s^\top \tilde{X}_s)^{j-1} \tilde{X}.$$

The $\tilde{X}$ derivative is significantly more involved:

$$\frac{\partial \theta_T}{\partial \tilde{X}} = \frac{\partial C^T \theta_0}{\partial \tilde{X}} + \frac{\partial D_T}{\partial \tilde{X}} \to \frac{\partial D_T}{\partial \tilde{X}} \text{ as } T \text{ increases, where}$$

$$\frac{\partial D_T}{\partial \tilde{X}} = \sum_{j=1}^{T} \binom{T}{j} \gamma^j \left[\sum_{k=0}^{j-2} (-\tilde{X}^\top \tilde{X})^k \otimes (-\tilde{X}^\top \tilde{X})^{j-2-k} \cdot 2\tilde{X} \cdot (\tilde{X}^\top \tilde{Y}) + (-\tilde{X}^\top \tilde{X})^{j-1} \tilde{Y} \mathbf{e}^\top \right].$$

These derivatives will yield a tensor of shape $d \times (m \times d)$, whereby multiplying by (I) in transpose gives a gradient of shape $m \times d$.

A.6. Outer-loop convergence lemmas

We state general conditions in the appendix, then apply them to our case of the outer-loop loss function in the main paper.

Lemma A.1 (Local strong convexity)**.** *Consider the function $f = Q + R$, where Q is a polynomial and $R(x) = \frac{\lambda}{2}||x||^2$ with regularization parameter λ. Let $\lambda_{\min}^{(Q)}$ be the smallest eigenvalue of $\nabla^2 Q$, the Hessian of Q, and let y be some point in the same domain as x.*

Suppose that for some finite $M \in [0, \lambda)$ and $r > 0$, we have that $\lambda_{\min}^{(Q)} \geq -M$ for all x in the r-neighborhood of y. Then there exists $\mu > 0$, namely $\lambda - M$, such that f is locally μ-strongly convex in $\{x : ||x - y|| \leq r\}$.

Proof. Consider x in this region. Set $\mu = \lambda - M > 0$. Then, for all x in this region,

$$\nabla_x^2 f(x) = \nabla_x^2 Q(x) + \lambda I \geq (-M + \lambda) I = \mu I.$$

Thus f is strongly convex in the r-neighborhood of y. □

Lemma A.2 (Outer-loop PL condition)**.** *Let x^* be a local minimizer of f. Under the conditions of Lemma A.1, for all x with $||x - x^*|| \leq r$, the outer-loop objective $f(x) = Q(x) + R(x)$ satisfies the PL inequality,*

$$\frac{1}{2}\|\nabla f(x)\|^2 \geq \mu(f(x) - f(x^*)), \tag{12}$$

where $\mu = \lambda - M > 0$.

Proof. From Lemma A.1, we have that f is μ-strongly convex in a neighborhood of the optimum. For strongly convex functions, the PL inequality holds with the same constant μ (see (Karimi et al., 2020)). Specifically, for any μ-strongly convex function,

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2}\|y - x\|^2.$$

Taking $y = x^*$, a local minimizer of f, and using $\nabla f(x^*) = 0$, we have

$$f(x^*) \geq f(x) + \langle \nabla f(x), x^* - x \rangle + \frac{\mu}{2}\|x^* - x\|^2,$$

which rearranges to

$$f(x) - f(x^*) \leq -\langle \nabla f(x), x^* - x \rangle - \frac{\mu}{2}\|x^* - x\|^2.$$

By Cauchy-Schwarz, $-\langle \nabla f(x), x^* - x \rangle \leq \|\nabla f(x)\|\|x^* - x\|$.

And by Young's inequality, $\|\nabla f(x)\|\|x^* - x\| \leq \frac{1}{2\mu}\|\nabla f(x)\|^2 + \frac{\mu}{2}\|x^* - x\|^2$. Therefore,

$$f(x) - f(x^*) \leq \frac{1}{2\mu}\|\nabla f(x)\|^2, \tag{13}$$

which gives us the PL inequality. □

A.7. Convergence plots

For the following experiment, real data ($X \in \mathbb{R}^{n \times d}, Y \in \mathbb{R}^n$) are generated by sampling $X, \theta \sim N(0, I_d)$ and $Y = X\theta + \eta$ for Gaussian noise η. We use the hyperparameters batch size $b = 32$, network samples per I-GD step $q = 30$, $\alpha_x = 0.0025$, $\alpha_\eta = 0.00025$, $\tilde{\eta}_0 = 0.05$, with an 80/20 train-test split. We also introduce the regularization parameter $\lambda_O = \lambda_I = 0.001$.

Reporting loss over distillation steps is an appropriate metric because the goal of distillation – good performance on the original dataset – is reported in the outer-loop testing loss.

For this run, we also introduce a few minor structural changes. Namely, we increasing the number of network initializations slightly whenever real loss increases over a certain threshold (e.g., $q \leftarrow \min\{q+1, \text{max\_q}\}$ if $\text{loss}_t \geq 1.02 \cdot \text{loss}_{t-1}$ for some hyperparameter max_q). We also scheduling the number t of I-GD steps per network initialization by $t \leftarrow t + \text{int}\left(\frac{1}{2}t \cdot (1 - e^{-s/300})\right)$ to increase slightly as the number of O-GD steps increases.

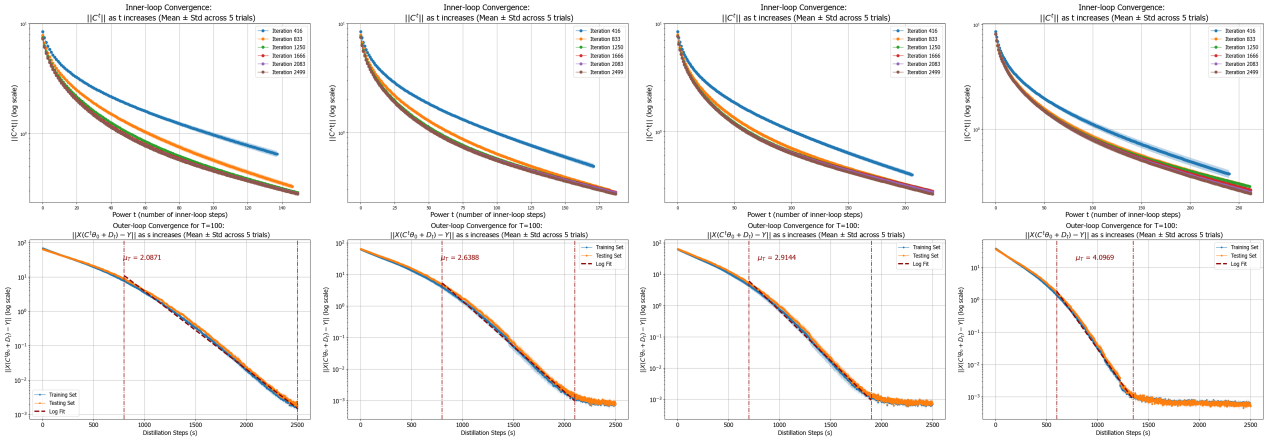

Figure 1. Top: Inner-loop convergence for $T = \{100, 125, 150, 175\}$ across select outer-loop (s) values. Convergence is consistently quicker for later outer-loop steps. The uneven inner-loop steps reflect modest increases across the full training run, a slight architectural boost to performance.
Bottom: Outer-loop convergence for the same T, where $S = 2500$. We include log fit for the main training run, excluding warmup and cooldown. For consistency, we define the training run so that the R^2 value is comparable across all trials; notice that the required warmup and cooldown times decrease as T grows.