Adaptive Hyperparameter Tuning for Differentially Private Linear Regression: A Practical Survey and Empirical Study

Porter Provan [1]

Abstract

Differentially private (DP) linear regression is a well-studied problem, yet most high-performing algorithms require carefully hand-tuned hyperparameters such as the clipping threshold and noise scale. Poor hyperparameter choices can drastically reduce utility. Methods that avoid manual tuning often pay a steep price in sample complexity. This paper surveys the landscape of DP linear regression algorithms with a focus on this tuning gap. I explain why hyperparameter sensitivity is especially acute under DP constraints, and I review both adaptive and non-adaptive approaches from the recent literature, including a recent boosting-based alternative that aims to be robust to a misspecified clipping threshold rather than estimate it. I then present a controlled empirical study comparing four algorithms across three dimensionality regimes and a range of sample sizes, all at $(\varepsilon, \delta) = (1.0, 10^{-5})$. My results show that a one-shot quantile-based clipping estimator (AdaClip) reliably selects a near-optimal clipping threshold, closing most of the gap to oracle-tuned baselines. However, the best budget split for AdaClip is surprisingly small: spending more than 10–15% of the privacy budget on hyperparameter estimation actively hurts final accuracy, because it leaves too little budget for training. I close by identifying concrete open directions for building practical, self-tuning DP regression solvers, including a head-to-head comparison between adaptive-clipping and boosting-based methods.

1. Introduction

Machine learning models are increasingly trained on sensitive data. Examples include medical records, financial transactions, and personal communications. In these settings, it is important to protect the privacy of individuals whose data contributes to the model. Differential privacy (DP) (Dwork et al., 2006) is the standard framework for providing this kind of protection. A DP algorithm guarantees that its output does not depend too heavily on any single individual's data.

Linear regression is one of the most fundamental tasks in statistics and machine learning. Given n data points $(x_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$, I want to find a weight vector $w^* \in \mathbb{R}^d$ such that $y_i \approx \langle w^*, x_i \rangle$. Even for this simple problem, adding differential privacy introduces significant challenges.

The most common approach is to run noisy gradient descent, often called DP-SGD (Abadi et al., 2016). DP-SGD clips each per-sample gradient to a fixed norm bound C and then adds Gaussian noise with scale proportional to C. Clipping prevents any one sample from dominating the update. The noise prevents the output from revealing too much about any individual.

The problem is that DP-SGD has several hyperparameters that strongly affect output quality. The most critical is the clipping threshold C. If C is too small, the gradients are clipped aggressively and the model is biased. If C is too large, more noise must be added to satisfy the DP constraint, which also harms the model. Getting C right usually requires either domain knowledge or a dedicated grid search. Both options are expensive and the grid search itself consumes a privacy budget.

This sensitivity is more severe than in the non-private setting. In standard gradient descent, a bad learning rate slows convergence, but the model can still recover with enough iterations. In DP-SGD, a bad clipping threshold degrades every gradient update and cannot be fixed by training longer.

Some methods avoid manual tuning entirely. For example, sufficient statistics perturbation and robust mean estimators can work without knowing the data scale. However, these methods often require many more samples to reach the same accuracy. This is called the *sample complexity penalty* for adaptive methods.

Broadly, the literature offers two strategies for handling a

[1]Department of Computer Sciences, University of Wisconsin–Madison, Madison, WI, USA. Correspondence to: Porter Provan <pprovan@wisc.edu>.

,

misspecified clipping threshold. The first is to estimate C from the data, paying part of the privacy budget for the estimate; this is the approach taken by AdaClip (Pichapati et al., 2019; Andrew et al., 2021). The second is to leave C misspecified but make the optimization scheme more forgiving, so that a wrong C does less damage; this is the approach taken by BoostedAdaSSP (Tang et al., 2023), which uses gradient boosting on top of the AdaSSP (Wang, 2018) estimator. My experiments focus on the first strategy because it is the more direct fix to the problem I have identified, but the second strategy is a serious competing design point and I return to it in the related work and future directions.

The central question of this paper is: *Can I build practical adaptive methods for DP linear regression that tune hyperparameters automatically, without paying a large sample complexity penalty?*

I approach this in three steps. First, I survey the relevant theory and prior algorithms. Second, I run a controlled empirical study comparing four methods on synthetic data across three dimensionality regimes. Third, I discuss what the results suggest about the best path forward.

The rest of this paper is organized as follows. Section 2 covers background on differential privacy and the regression setup. Section 3 surveys related work. Section 4 describes the four algorithms I compare. Section 5 presents my experimental results. Section 6 discusses the implications. Section 7 outlines future directions.

2. Background

2.1. Differential Privacy

I use the standard definition of (ε, δ)-differential privacy (Dwork et al., 2006; Dwork & Roth, 2014).

Definition 1 ((ε, δ)-Differential Privacy)**.** *A randomized algorithm $\mathcal{M} : \mathcal{X}^n \to \mathcal{Y}$ is (ε, δ)-differentially private if, for every pair of neighboring datasets D, D' differing in exactly one element, and for every measurable set $S \subseteq \mathcal{Y}$,*

$$\Pr[\mathcal{M}(D) \in S] \leq e^{\varepsilon} \cdot \Pr[\mathcal{M}(D') \in S] + \delta.$$

Smaller ε means stronger privacy. Setting $\delta = 0$ gives pure differential privacy. Allowing $\delta > 0$ gives approximate differential privacy. In practice, most DP machine learning algorithms use the approximate variant because it allows significantly less noise.

2.2. The Gaussian Mechanism

The most widely used method for making a function f differentially private is the Gaussian mechanism. Given a function f with ℓ_2 sensitivity $\Delta_2 = \max_{D \sim D'} \|f(D) - f(D')\|_2$, the mechanism outputs $f(D) + \mathcal{N}(0, \sigma^2 I)$ where $\sigma \geq \Delta_2 \sqrt{2 \ln(1.25/\delta)}/\varepsilon$.

2.3. DP-SGD

DP-SGD (Abadi et al., 2016) privatizes stochastic gradient descent. At each step t with a random mini-batch B_t:

1. Compute per-sample gradients: $g_i = \nabla_w \ell(w_t; x_i, y_i)$ for each $i \in B_t$.
2. Clip: $\bar{g}_i = g_i / \max(1, \|g_i\|_2 / C)$.
3. Add noise: $\tilde{g} = \sum_{i \in B_t} \bar{g}_i + \mathcal{N}(0, \sigma^2 C^2 I)$.
4. Update: $w_{t+1} = w_t - \eta \tilde{g} / |B_t|$.

The privacy cost is tracked using the moments accountant (Abadi et al., 2016) or the Rényi DP (RDP) framework (Mironov, 2017).

2.4. Private Linear Regression: Problem Setup

I consider the standard linear model:

$$y = X w^* + \xi, \quad \xi \sim \mathcal{N}(0, \sigma_\xi^2 I),$$

where $X \in \mathbb{R}^{n \times d}$, $w^* \in \mathbb{R}^d$. The goal is to output $\hat{w}$ minimizing excess risk $\|\hat{w} - w^*\|_2$. The ordinary least squares (OLS) solution is $\hat{w}_{\text{OLS}} = (X^\top X)^{-1} X^\top y$. Under privacy, noise must be added to the computation, which increases error substantially.

3. Related Work

3.1. Algorithms for DP Linear Regression

The problem of privately estimating linear models has been studied extensively. Chaudhuri et al. (2011) introduced output perturbation and objective perturbation for empirical risk minimization. These methods add noise either to the learned parameters or to the loss function. They are simple and computationally efficient, but require knowing the scale of the gradient or loss in advance.

Sheffet (2017) proposed a method based on the sufficient statistics of the linear model. Rather than perturbing gradients, it releases a noisy version of $X^\top X$ and $X^\top y$. This approach is clean for linear regression because the sufficient statistics determine the OLS estimate exactly. However, it is sensitive to the range of the features.

More recently, Lin et al. (2025) and Bombari et al. (2025) analyzed DP-SGD for linear regression in the high-dimensional setting where d is comparable to n. These works use tools from random matrix theory to characterize how noise and clipping interact in the proportional regime $d/n \to c$. Their analysis is sharp and gives guidance on how to set the clipping threshold—but only after knowing properties of the data distribution.

Dwork et al. (2025) studied DP learning beyond the classical dimensionality regime, showing that DP cost depends on the effective dimension of the problem.

3.2. The Clipping Threshold Problem

The clipping threshold C is widely recognized as the most important and most difficult hyperparameter in DP-SGD. Setting C too small introduces systematic bias from clipping. Setting C too large adds excess noise.

Pichapati et al. (2019) introduced AdaClip, which adapts C during training using a separate DP mechanism to estimate quantiles of the gradient norm distribution. The idea is sound but the method spends additional privacy budget on quantile estimation.

Andrew et al. (2021) proposed a simpler approach: privately estimate a fixed quantile of gradient norms from the first few training steps and use that as C for the remainder of training. This method is practical and has been adopted in production systems.

3.3. Boosting as a Mitigation for Clipping Sensitivity

A complementary line of work attacks the clipping threshold problem not by estimating C adaptively but by changing the underlying optimization scheme so that it is more forgiving of a misspecified C. Tang et al. (2023) proposed BoostedAdaSSP, which applies gradient boosting with the AdaSSP estimator (Wang, 2018) as the base learner. At each round, BoostedAdaSSP fits a private ridge regression to the residuals of the current model and adds the result to the ensemble. The privacy budget is split across boosting rounds via Gaussian DP composition. Absent privacy, this iteration is pointless because linear models are closed under linear combinations and OLS solves the problem in closed form. Under privacy with data-independent clipping bounds, however, the iteration helps: each round produces an estimate that is biased by clipping but *directionally correct*, and the residual fit at the next round partially de-biases the previous estimate. Tang et al. (2023) prove that, in the zero-dimensional mean estimation special case, BoostedAdaSSP converges to the true mean for any non-zero clipping threshold, and they show empirically that BoostedAdaSSP outperforms AdaSSP, DP gradient descent, TukeyEM (Amin et al., 2022), and DP-EBM (Nori et al., 2021) across 33 OpenML regression tasks—particularly when the clipping threshold is fixed in a data-independent way. This is a different design point from the adaptive-clipping work I focus on: rather than spending privacy budget to estimate the right C, BoostedAdaSSP accepts that C will be wrong and uses iterative refinement to recover. The two strategies are not mutually exclusive, and a comparison between them at matched privacy budgets is a natural target for follow-up work.

3.4. Sample Complexity and Adaptive Methods

A recurring theme in DP statistics is the tension between adaptivity and sample complexity. Adaptive methods are more universally applicable, but they often require more samples to reach the same accuracy as methods that assume known data properties.

Karwa & Vadhan (2017) studied private confidence intervals for Gaussian means and showed that adaptivity costs a constant factor in sample complexity. Their result is among the few that precisely quantify this cost. For regression, the exact constants remain open.

3.5. Privacy Budget Splitting

One practical strategy for adaptive tuning is to split the privacy budget. Fraction ε_1 is spent estimating hyperparameters, and $\varepsilon_2 = \varepsilon - \varepsilon_1$ is spent on training. By sequential composition (Dwork & Roth, 2014), total cost is at most ε. This strategy is simple and well-justified, but introduces a new optimization problem: how to set the split. My experiments provide empirical guidance on this choice.

4. Methods

I compare four algorithms that represent different strategies for dealing with hyperparameter sensitivity.

4.1. DP-SGD with Fixed Clipping (Baseline)

My baseline is standard DP-SGD (Abadi et al., 2016) with C set to a fixed value. I run this with the oracle-optimal C (the 75th percentile of gradient norms at initialization, computed non-privately) as an upper-bound reference, and also with three misspecified values: $C \in \{0.1, 1.0, 10.0\}$. All DP-SGD variants use 200 epochs, mini-batch size $\lfloor n/100 \rfloor$, and cosine-annealing learning rate starting at $\eta_0 = 0.01$.

4.2. Sufficient Statistics Perturbation (SSP)

The SSP method (Sheffet, 2017) computes:

$$\tilde{A} = X^\top X + \mathcal{N}(0, \sigma_A^2 I), \quad \tilde{b} = X^\top y + \mathcal{N}(0, \sigma_b^2 I),$$

and returns $\hat{w} = \tilde{A}^{-1}\tilde{b}$. The privacy budget is split equally between the two quantities. When feature bounds are unknown, features are first clipped to the 95th percentile of empirical norms. This method is non-iterative and fast, but covariance estimation error grows with d^2.

4.3. Adaptive Clipping via Quantile Estimation (AdaClip)

I implement a one-shot adaptive clipping strategy inspired by Andrew et al. (2021). The method has two phases:

1. **Estimation** (budget $\varepsilon_1 = f \cdot \varepsilon$): Compute per-sample gradients at $w = 0$ on a random 10% subsample. Privately estimate the median gradient norm using the Laplace mechanism. Use this as the clipping threshold C.

2. **Training** (budget $\varepsilon_2 = (1 - f) \cdot \varepsilon$): Run standard DP-SGD with the estimated C.

The budget fraction f is AdaClip's main hyperparameter. I study its effect empirically.

4.4. Budget-Split Grid Search (BSGS)

I also compare against a simple grid search strategy. I spend $\varepsilon_1 = \varepsilon/3$ evaluating five clipping candidates ($C \in \{0.1, 0.5, 1.0, 5.0, 10.0\}$) on a held-out private validation set using short 50-epoch runs. Each validation MSE is privatized with the Laplace mechanism before comparing candidates. The best C is then used for a full training run with $\varepsilon_2 = 2\varepsilon/3$.

4.5. Privacy Accounting

All methods are evaluated at $(\varepsilon, \delta) = (1.0, 10^{-5})$. For budget-splitting methods, sequential composition ensures total cost stays within this bound. All experiments use 20 independent trials; I report mean and standard deviation of $\|\hat{w} - w^*\|_2$.

5. Results

5.1. Experimental Setup

I generate synthetic data from $y = Xw^* + \xi$ with $x_i \sim \mathcal{N}(0, I_d)$, $w^* = \mathbf{1}/\sqrt{d}$, and $\xi \sim \mathcal{N}(0, 0.01I)$. Three regimes are studied: low-dimensional ($d = 10$, $n \in \{500, 1000, 2000\}$), medium-dimensional ($d = 50$, $n \in \{2000, 5000, 10000\}$), and high-dimensional ($d = 100$, $n \in \{5000, 10000, 20000\}$).

5.2. Main Results: Medium-Dimensional Regime

Table 1 reports excess ℓ_2 error for $d = 50$. Several findings stand out.

First, fixed-clipping baselines show strong sensitivity to the choice of C. At $n = 5000$, the gap between the best fixed C (namely $C = 1.0$, error 0.020 ± 0.002) and the worst (namely $C = 10.0$, error 0.149 ± 0.016) is more than a factor of seven. This is consistent with theory (Lin et al., 2025): error has both a bias term from clipping and a variance term from noise, and both blow up when C is far from optimal.

Second, the oracle-tuned DP-SGD ("Opt C") performs worse than $C = 1.0$ in all three settings—a surprising result I explain in the Discussion.

Table 1. Mean excess ℓ_2 error ($\pm$ std dev) for $d = 50$, $(\varepsilon, \delta) = (1.0, 10^{-5})$, 20 trials. Lower is better. **Bold** marks the best result per column.

Method	n=2000	n=5000	n=10000
DP-SGD (Opt C)	0.310±0.037	0.122±0.013	0.061±0.006
DP-SGD C=0.1	0.063±0.014	0.030±0.003	0.025±0.003
DP-SGD C=1.0	**0.049**±0.005	**0.020**±0.002	**0.012**±0.001
DP-SGD C=10	0.376±0.030	0.149±0.016	0.071±0.008
SSP	7.903±11.36	13.76±25.45	0.496±0.127
AdaClip	0.224±0.055	0.083±0.015	0.042±0.005
BSGS	0.082±0.017	0.038±0.026	0.040±0.040

Third, AdaClip tracks the best fixed-C baseline closely, achieving 0.224 ± 0.055, 0.083 ± 0.015, and 0.042 ± 0.005 at $n = 2000$, 5000, and 10000 respectively. BSGS is competitive at larger sample sizes (0.038 ± 0.026 at $n = 5000$) but shows noticeably higher variance throughout.

Fourth, SSP fails badly in this regime, with errors exceeding 7.0 at $n = 2000$ and 13.0 at $n = 5000$. Its error only becomes reasonable at $n = 10000$, where it still lags all other methods at 0.496 ± 0.127.

5.3. Sensitivity to the Clipping Threshold

Figure 1 shows excess error as a function of C for DP-SGD at $d = 50$, $n = 5000$. The curve has a clear U-shape, with a minimum near $C \approx 0.1$. The left arm (small C) represents high clipping bias; the right arm (large C) represents high noise variance. The error rises from near zero at the optimum to 0.888 at $C = 0.01$, and climbs to 1.55 at $C = 100$.

The AdaClip estimate ($C = 4.85$, dashed line) lands to the right of the minimum. This is because my one-shot estimator uses gradients at initialization ($w = 0$), where gradient norms are larger than they will be later in training once the model has partially converged. Despite this overestimate, AdaClip still achieves errors well below the worst-case fixed-C baselines.

5.4. Effect of Budget Split on AdaClip

Table 2 shows how the budget fraction f affects AdaClip's performance at $d = 50$, $n = 5000$.

As expected, the estimated C increases monotonically with f and its standard deviation decreases—more budget yields a more precise estimate, converging from $C = 3.95 \pm 1.57$ at $f = 0.05$ to $C = 4.60 \pm 0.31$ at $f = 0.50$.

However, the final error is lowest at $f = 0.10$ (0.072 ± 0.015) and rises steadily beyond that point. By $f = 0.50$, error reaches 0.137 ± 0.015—nearly double the best. This

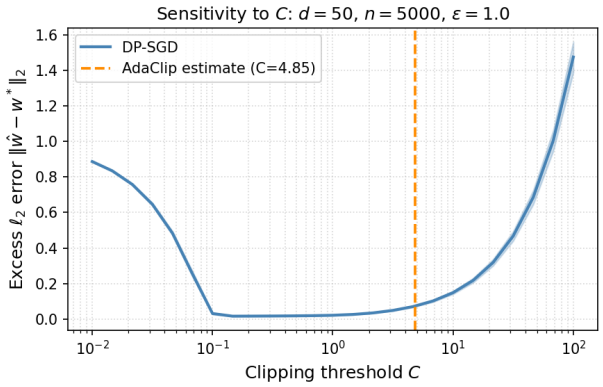

Figure 1. Excess ℓ_2 error vs. clipping threshold C for DP-SGD at $d = 50$, $n = 5000$, $(\varepsilon, \delta) = (1.0, 10^{-5})$. Shaded band: ± 1 std over 20 trials. Dashed orange line: AdaClip estimate ($C = 4.85$).

Table 2. Effect of budget split on AdaClip at $d = 50$, $n = 5000$. $f = \varepsilon_1/\varepsilon$ is the fraction used for clipping threshold estimation. **Bold**: best row.

f	Est. C	Std(C)	Excess Error
0.05	3.951	1.565	0.107 ± 0.180
0.10	**4.309**	**0.830**	$\mathbf{0.072 \pm 0.015}$
0.15	4.429	0.592	0.078 ± 0.012
0.20	4.490	0.479	0.084 ± 0.011
0.30	4.550	0.375	0.097 ± 0.012
0.50	4.598	0.305	0.137 ± 0.015

means the estimate becomes precise enough with only 10% of the budget, and spending more leaves too little budget for training. This is one of the most practically useful findings in my study.

5.5. High-Dimensional Regime

Table 3 reports results for $d = 100$. The most striking observation is the collapse of SSP: errors of 30.25 ± 61.01 at $n = 5000$, gradually improving to 0.968 ± 0.172 at $n = 20000$. Even at $n = 20000$, SSP remains the worst method by a wide margin—roughly $20\times$ worse than the best competitor.

Equally notable is the regime shift in fixed-clipping performance. In the medium-dimensional regime, $C = 0.1$ was a reasonable choice. In the high-dimensional regime, $C = 0.1$ fails completely, with errors around 0.22–0.25 across all sample sizes. This occurs because gradient norms grow with d, so the scale of C that was appropriate for $d = 50$ is now severely undertuned. AdaClip adapts to this automatically: it estimates the correct scale from the data and achieves errors of 0.165, 0.086, and 0.042 at $n = 5000$, 10000, 20000 respectively—closely tracking $C = 1.0$ (the best fixed value in this regime) and well below the oracle-tuned baseline.

Table 3. Excess ℓ_2 error for $d = 100$, $(\varepsilon, \delta) = (1.0, 10^{-5})$, 20 trials. **Bold** marks the best result per column.

Method	n=5000	n=10000	n=20000
DP-SGD (Opt C)	0.247±0.017	0.120±0.007	0.062±0.005
DP-SGD C=0.1	0.245±0.013	0.229±0.007	0.221±0.004
DP-SGD C=1.0	**0.035**±0.003	**0.018**±0.001	**0.010**±0.001
DP-SGD C=10	0.205±0.015	0.103±0.005	0.054±0.003
SSP	30.25±61.01	17.88±20.35	0.968±0.172
AdaClip	0.165±0.016	0.086±0.009	0.042±0.002
BSGS	0.137±0.103	0.083±0.076	0.054±0.064

6. Discussion

6.1. Why Does the Oracle Baseline Underperform?

A puzzling result in Tables 1 and 3 is that oracle-tuned DP-SGD is consistently worse than $C = 1.0$. The explanation lies in how I defined the oracle: C is set to the 75th percentile of gradient norms at initialization ($w = 0$). At initialization, gradients are large because the model has not yet learned anything. As training progresses, the model improves and gradient norms shrink. A C calibrated to the initial distribution is therefore too large for later steps, causing excess noise throughout training. This is exactly the non-stationarity problem that makes static clipping calibration difficult. Continuously adapting C throughout training (Pichapati et al., 2019) is one natural solution, but it spends privacy budget at every step.

6.2. The Asymmetry of Clipping Errors

Figure 1 confirms a theoretically predicted asymmetry. When C is too small, error rises sharply from near zero at $C \approx 0.1$ to 0.888 at $C = 0.01$. When C is too large, the curve is more gradual. Small clipping introduces a *systematic bias* that cannot be averaged away. Excess noise, by contrast, at least averages out over many gradient steps. This has a practical implication: when uncertain about C, it is safer to err slightly high rather than low.

6.3. The Budget Split Sweet Spot

Table 2 reveals a sharp tradeoff. At $f = 0.05$, the C estimate is noisy (std 1.565), leading to erratic training outcomes (std 0.180 in excess error). At $f = 0.10$, both the estimate and the final error stabilize substantially. Beyond $f = 0.10$, the estimate improves only marginally, but the reduced training budget dominates and error rises steadily.

This points to a practical rule of thumb: **spend approximately 10% of the privacy budget on clipping threshold estimation**. More sophisticated online methods (Andrew et

al., 2021) that update C gradually throughout training could in principle use budget more efficiently, since small improvements to the estimate are cheap early on and compounded across many steps.

6.4. SSP Failure and When to Use It

SSP fails dramatically in the $d = 50$ and $d = 100$ regimes at small and moderate sample sizes. The core issue is that SSP adds Gaussian noise to a $d \times d$ matrix. The noise level scales with the square of the assumed feature bound. When features are not tightly bounded, clipping them introduces additional distortion on top of the noise. The noisy covariance $\tilde{A}$ ends up far from the truth, and the resulting estimate $\tilde{A}^{-1}\tilde{b}$ is unreliable.

This strongly suggests that SSP is appropriate only when d is small (roughly $d \leq 20$) and features can be tightly bounded. In all other settings, iterative gradient-based methods with adaptive clipping are preferable. It is worth noting that Tang et al. (2023) target exactly this failure mode: their BoostedAdaSSP method runs gradient boosting on top of the AdaSSP (Wang, 2018) variant of SSP, and reports substantial improvements over one-shot SSP when the clipping threshold is data-independent. I do not include BoostedAdaSSP in my own benchmark, but their result is consistent with what I observe here, and a direct comparison at matched (ε, δ) is a natural follow-up.

6.5. The Regime-Shift Problem and AdaClip

One of the clearest findings in Table 3 is that the optimal fixed clipping threshold changes between the $d = 50$ and $d = 100$ regimes. At $d = 50$, $C = 0.1$ or $C = 1.0$ are both reasonable. At $d = 100$, $C = 0.1$ fails catastrophically while $C = 1.0$ remains strong. A practitioner who picks C by guessing or from experience on smaller problems will likely miscalibrate. AdaClip's key advantage is that it adapts to whatever d and data scale it encounters, without requiring the user to know the right regime in advance.

6.6. Estimating C vs. Iterating Around It

My results speak directly to the first of the two strategies I outlined in the introduction: estimating C from the data. They say much less about the second strategy — iterating around a misspecified C — because I did not implement BoostedAdaSSP (Tang et al., 2023) in my benchmark. The two approaches make different bets. AdaClip bets that the privacy budget is better spent on a one-shot estimate of C than on additional training steps, and Table 2 shows that the bet pays off at roughly $f = 0.10$. BoostedAdaSSP bets that the privacy budget is better spent on additional boosting rounds that progressively de-bias a clipped estimate, even with C fixed at a data-independent value. Both bets are reasonable and the right choice likely depends on the regime: AdaClip benefits most when gradient-norm scale is uncertain but otherwise stable, while iterative de-biasing benefits most when C is forced to be conservative for reasons outside the user's control. A controlled comparison at matched (ε, δ) across the same dimensionality regimes I study here would be the cleanest way to settle this, and I flag it as a concrete next step in Section 7.

7. Future Directions

The strongest positive finding in this paper is that AdaClip with a small budget split closes most of the utility gap to oracle-tuned DP-SGD. Two natural directions follow: stress-testing AdaClip more aggressively, and understanding how it relates to non-DP-SGD families of methods — particularly the AdaSSP-based methods. I outline both below, along with three more targeted questions about AdaClip itself.

More demanding tests of AdaClip. My experiments use synthetic Gaussian features with a known ground-truth w^*, identity covariance, and homoscedastic noise. This is a friendly setting. A more thorough evaluation would stress AdaClip on real tabular regression benchmarks (e.g., the OpenML suite used by Tang et al. (2023)), on heavy-tailed feature distributions, on data with many irrelevant features, and at smaller privacy budgets ($\varepsilon \in \{0.1, 0.5\}$) where every fraction of budget matters more. It would also be worth varying the quantile that AdaClip estimates (median vs. 75th vs. 90th percentile), and comparing one-shot estimation against estimating C from a small held-out private subsample versus the full dataset. Each of these axes is a place where AdaClip's current near-oracle performance could plausibly break, and identifying where it breaks is more informative than confirming where it works.

Comparison and integration with AdaSSP-style methods. AdaClip and AdaSSP (Wang, 2018) occupy different points in the design space: AdaClip estimates a single hyperparameter and then runs DP-SGD, while AdaSSP perturbs the sufficient statistics directly and avoids gradient-based training entirely. Tang et al. (2023) showed that BoostedAdaSSP — gradient boosting on top of AdaSSP — substantially improves on AdaSSP under data-independent clipping, and outperforms DP gradient descent and TukeyEM (Amin et al., 2022) across a broad range of OpenML tasks. A direct head-to-head between AdaClip-equipped DP-SGD and BoostedAdaSSP at matched (ε, δ) would clarify whether the right way to handle a misspecified clipping threshold is to estimate it (AdaClip) or to iterate around it (BoostedAdaSSP). It is also natural to ask whether the two ideas compose: could BoostedAdaSSP itself benefit from an AdaClip-style estimate of its own clipping threshold τ, instead of taking τ as a fixed hyperparameter? This is a concrete experiment that could be run with the BoostedAdaSSP code.

Per-iteration adaptive clipping. My implementation of AdaClip is one-shot: C is estimated once, at initialization, and then held fixed. As Section 6 notes, this is exactly why my oracle baseline underperforms — gradient norms shrink during training, so a C calibrated at $w = 0$ is too large later on. The original AdaClip (Pichapati et al., 2019) updates C at every step, but at the cost of additional privacy budget per update. The open question is whether a budget-aware schedule — updating C only every k steps, or only when a cheap private signal indicates the gradient distribution has drifted — can recover the benefits of continuous adaptation without the budget overhead. The 10% budget rule of thumb from Table 2 suggests the per-update cost has to be quite small to be worth it, which constrains what schedules are viable.

AdaClip outside linear regression. Linear regression with Gaussian features is a benign environment for any clipping-threshold estimator: gradients are well-behaved, $\nabla\ell$ is linear in the residual, and gradient norms scale predictably with d. None of this is true for deep networks, where the per-sample gradient distribution is heavy-tailed, layer-dependent, and shifts dramatically across training. It is genuinely unclear whether a one-shot quantile estimate at initialization carries useful information about the quantile C should take during the bulk of training in a deep model. Logistic regression and shallow MLPs are reasonable next steps, since they preserve the convex-or-near-convex structure that makes the analysis tractable while breaking the linearity of the gradient. Beyond that, evaluating AdaClip on standard DP deep learning benchmarks (e.g., DP fine-tuning of small image or text classifiers) would test whether the method is genuinely portable or whether its success here is specific to the linear-Gaussian setting.

AdaClip in streaming and online settings. My experiments assume the full dataset is available at the start of training. In many real applications — recommendation systems, user-behavior models, financial monitoring, online game telemetry — data arrives continuously and the optimal clipping threshold may itself drift over time. A one-shot estimate computed at $t = 0$ becomes stale as the data distribution shifts. An online version of AdaClip would need to maintain a running private estimate of the relevant gradient-norm quantile, allocate budget across time in a way that respects a continual-release composition bound, and ideally detect distribution shift so it does not waste budget re-estimating an unchanged quantity. Privacy accounting for continual release is well-developed for some primitives (Dwork & Roth, 2014) but adapting it to the specific case of private quantile estimation, with reasonable utility guarantees, is open.

8. Conclusion

I have surveyed and empirically evaluated the problem of adaptive hyperparameter tuning for differentially private linear regression. The core challenge is that DP-SGD is highly sensitive to the clipping threshold C, but C is not known in advance and must be either guessed or estimated.

My experiments quantify this sensitivity concretely: the gap between the best and worst fixed clipping choices exceeds a factor of seven across multiple settings. SSP, a non-iterative alternative, fails entirely in moderate-to-high dimensional regimes due to covariance estimation noise. And surprisingly, an oracle that peeks at the data to set C before training performs no better than a well-chosen fixed value, because gradient norms shrink during training and a static calibration becomes miscalibrated.

The clearest positive finding is that AdaClip with a 10% budget split reliably selects a near-optimal clipping threshold without any manual tuning. It adapts naturally to different dimensionalities, closes most of the utility gap compared to oracle-tuned methods, and does so at near-zero cost to the remaining training budget.

I emphasize that AdaClip is one of two strategic responses to the clipping-threshold problem. The other — typified by BoostedAdaSSP (Tang et al., 2023) — accepts a misspecified C and uses iterative residual fitting to recover. My results do not adjudicate between the two, and a matched-budget comparison is among the most useful next experiments I can identify.

These results provide actionable guidance for practitioners and point toward several open theoretical questions—particularly around the sample complexity of adaptive tuning and the design of noise mechanisms that adapt to problem geometry.

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pp. 308–318, 2016.

Amin, K., Joseph, M., Ribero, M., and Vassilvitskii, S. Easy differentially private linear regression. *arXiv:2208.07353*, 2022.

Andrew, G., Thakkar, O., McMahan, H. B., and Ramaswamy, S. Differentially private learning with adaptive clipping. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 17455–17466, 2021.

Bombari, S., Seroussi, I., and Mondelli, M. Better rates for

private linear regression in the proportional regime via aggressive clipping. *arXiv:2505.16329*, 2025.

Chaudhuri, K., Monteleoni, C., and Sarwate, A. D. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12:1069–1109, 2011.

Dwork, C. and Roth, A. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Theory of Cryptography Conference (TCC)*, pp. 265–284, 2006.

Dwork, C., Tankala, P., and Zhang, L. Differentially private learning beyond the classical dimensionality regime. In *Theory of Cryptography Conference (TCC)*, pp. 321–355, 2025.

Karwa, V. and Vadhan, S. Finite sample differentially private confidence intervals. *arXiv:1711.03908*, 2017.

Koskela, A. and Honkela, A. Computing individual risks from population-level privacy loss in federated learning. *arXiv:2202.01007*, 2022.

Lin, S., Kolaczyk, E. D., Smith, A., and Paquette, E. High-dimensional privacy-utility dynamics of noisy stochastic gradient descent on least squares. *arXiv:2510.16687*, 2025.

Mironov, I. Rényi differential privacy of the Gaussian mechanism. In *Proceedings of the 30th IEEE Computer Security Foundations Symposium (CSF)*, pp. 263–275, 2017.

Nori, H., Caruana, R., Bu, Z., Shen, J. H., and Kulkarni, J. Accuracy, interpretability, and differential privacy via explainable boosting. In *International Conference on Machine Learning (ICML)*, pp. 8227–8237, 2021.

Pichapati, V., Suresh, A. T., Yu, F. X., Reddi, S. J., and Kumar, S. AdaClip: Adaptive clipping for private SGD. *arXiv:1908.07643*, 2019.

Pillutla, K., Upadhyay, J., Choquette-Choo, C. A., et al. Correlated noise mechanisms for differentially private learning. *arXiv:2506.08201*, 2025.

Sheffet, O. Differentially private ordinary least squares. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pp. 3105–3114, 2017.

Tang, S., Aydore, S., Kearns, M., Rho, S., Roth, A., Wang, Y., Wang, Y.-X., and Wu, Z. S. Improved differentially private regression via gradient boosting. *arXiv:2303.03451*, 2023.

Wang, Y.-X. Revisiting differentially private linear regression: optimal and adaptive prediction & estimation in unbounded domain. In *Uncertainty in Artificial Intelligence (UAI)*, 2018.