
Testing Calibration in Nearly-Linear Time

Lunjia Hu
Harvard University
lunjia@alumni.stanford.edu

Arun Jambulapati
University of Michigan
jmbllpati@gmail.com

Kevin Tian
University of Texas at Austin
kjtian@cs.utexas.edu

Chutong Yang
University of Texas at Austin
cyang98@utexas.edu

Abstract

In the recent literature on machine learning and decision making, *calibration* has emerged as a desirable and widely-studied statistical property of the outputs of binary prediction models. However, the algorithmic aspects of measuring model calibration have remained relatively less well-explored. Motivated by [BGHN23a], which proposed a rigorous framework for measuring distances to calibration, we initiate the algorithmic study of calibration through the lens of property testing. We define the problem of *calibration testing* from samples where given n draws from a distribution \mathcal{D} on (predictions, binary outcomes), our goal is to distinguish between the cases where \mathcal{D} is perfectly calibrated or ε -far from calibration. We make the simple observation that the empirical smooth calibration linear program can be reformulated as an instance of minimum-cost flow on a highly-structured graph, and design an exact dynamic programming-based solver for it which runs in time $O(n \log^2(n))$, and solves the calibration testing problem information-theoretically optimally in the same time. This improves upon state-of-the-art black-box linear program solvers requiring $\Omega(n^\omega)$ time, where $\omega > 2$ is the exponent of matrix multiplication. We also develop algorithms for tolerant variants of our testing problem improving upon black-box linear program solvers, and give sample complexity lower bounds for alternative calibration measures to the one considered in this work. Finally, we present experiments showing the testing problem we define faithfully captures standard notions of calibration, and that our algorithms scale efficiently to accommodate large sample sizes.

1 Introduction

Probabilistic predictions are at the heart of modern data science. In domains as wide-ranging as forecasting (e.g. predicting the chance of rain from meteorological data [MW84, Mur98]), medicine (e.g. assessing the likelihood of disease [Doi07]), computer vision (e.g. assigning confidence values for categorizing images [VDDP17]), and more (e.g. speech recognition [AAA⁺16] and recommender systems [RRSK11]), prediction models have by now become essential components of the decision-making pipeline. Particularly in the context of critical, high-risk use cases, the interpretability of prediction models is therefore paramount in downstream applications. That is, how do we assign meaning to the predictions our model gives us, especially when the model is uncertain?

We focus on perhaps the most ubiquitous form of prediction modeling: binary predictions, represented as tuples (v, y) in $[0, 1] \times \{0, 1\}$ (where the v coordinate is our prediction of the likelihood of an event, and the y coordinate is the observed outcome). We model prediction-outcome pairs in the binary prediction setting by a joint distribution \mathcal{D} over $[0, 1] \times \{0, 1\}$, fixed in the following discussion. In

this context, *calibration* of a predictor has emerged as a basic desideratum. A prediction-outcome distribution \mathcal{D} is said to be calibrated if

$$\mathbb{E}_{(v,y)\sim\mathcal{D}}[y \mid v = t] = t \text{ for all } t \in [0, 1]. \quad (1)$$

That is, calibration asks that the outcome is 1 exactly 60% of the time, when the model returns a prediction $v = 0.6$. While calibration (or approximate variants thereof) is a relatively weak requirement on a meaningful predictor, as it can be achieved by simple models,¹ it can still be significantly violated in practice. For example, interest in calibration in the machine learning community was spurred by [GPSW17], which observed that many modern deep learning models are far from calibrated. Moreover, variants of calibration have been shown to have strong postprocessing properties for fairness constraints and loss minimization [HKRR18, DKR⁺21, GKR⁺22], which has garnered renewed interest in calibration by the theoretical computer science and statistics communities.

The question of measuring the calibration of a distribution is subtle; even a calibrated distribution incurs measurement error due to sampling. For example, consider the *expected calibration error*, used in e.g. [NCH15, GPSW17, MDR⁺21a, RT21b] as a ground-truth measure of calibration:

$$\text{ECE}(\mathcal{D}) := \mathbb{E}_{(v,y)\sim\mathcal{D}} \left[\left| \mathbb{E}_{(v',y)\sim\mathcal{D}} [y \mid v' = v] - v \right| \right].$$

Unfortunately, the empirical ECE is typically meaningless; if the marginal density of v is continuous, we will almost surely only observe a single sample with each v value. Further, [KF08] observed that ECE is discontinuous in v . In practice, binned variants of ECE are often used as a proxy, where a range of v is lumped together in the conditioning event. However, hyperparameter choices (e.g. the number of bins) can significantly affect the quality of binned ECE variants as a distance measure [KLM19, NDZ⁺19, MDR⁺21a].² Moreover, as we explore in this paper, binned calibration measures inherently suffer from larger sample complexity-to-accuracy tradeoffs, and are less faithful to ground truth calibration notions in experiments than the calibration measures we consider.

Recently, [BGHN23a] undertook a systematic study of various measures of distance to calibration proposed in the literature. They proposed information-theoretic tractability in the *prediction-only access (PA) model*, where the calibration measure definition can only depend on the joint prediction-outcome distribution (rather than the features of training examples),³ as a desirable criterion for calibration measures. Correspondingly, [BGHN23a] introduced Definition 1 as a ground-truth notion for measuring calibration in the PA model, which we also adopt in this work.⁴

Definition 1 (Lower distance to calibration). *Let \mathcal{D} be a distribution over $[0, 1] \times \{0, 1\}$. The lower distance to calibration (LDTC) of \mathcal{D} , denoted $\underline{\text{dCE}}(\mathcal{D})$, is defined by*

$$\underline{\text{dCE}}(\mathcal{D}) := \inf_{\Pi \in \text{ext}(\mathcal{D})} \mathbb{E}_{(u,v,y)\sim\Pi} |u - v|,$$

where $\text{ext}(\mathcal{D})$ is all distributions Π over $(u, v, y) \in [0, 1] \times [0, 1] \times \{0, 1\}$ satisfying the following.

- The marginal distribution of (v, y) is \mathcal{D} .
- The marginal distribution (u, y) is perfectly calibrated, i.e. $\mathbb{E}_{\Pi}[y|u] = u$.

Definition 1 has various beneficial aspects: it is convex in v , computable in the PA model, and (as shown by [BGHN23a]) polynomially-related to various other calibration measures, including some which require feature access, e.g. the *distance to calibration* (DTC, Eq. (1), [BGHN23a]). Roughly, the DTC of a distribution is the tightest lower bound on the ℓ_1 distance between v and any calibrated function of the features, *after taking features into account*. The LDTC is the analog of this feature-aware measure of calibration when limited to the PA model, so it does not depend on features. We focus on Definition 1 as our ground-truth measure in the remainder of the paper.

¹The predictor which ignores features and always return the population mean is calibrated, for example.

²For example, [NDZ⁺19] observed that, in their words, “dramatic differences in bin sensitivity” can occur “depending on properties of the (distribution) at hand,” a sentiment echoed by Section 5 of [MDR⁺21a].

³This access model is practically desirable because it abstracts away the feature space, which can lead to significant memory savings when our goal is only to test the calibration of model predictions. Moreover, this matches conventions in the machine learning literature, e.g. loss functions are typically defined in the PA model.

⁴We note that [BGHN23a] introduced an *upper distance to calibration*, also defined in the PA model, which they showed is quadratically-related to the $\underline{\text{dCE}}$ in Definition 1. However, the upper distance does not satisfy basic properties such as continuity, making it less amenable to estimation and algorithm design.

1.1 Our results

We initiate the algorithmic study of the *calibration testing* problem, defined as follows.

Definition 2 (Calibration testing). *Let $\varepsilon \in [0, 1]$. We say algorithm \mathcal{A} solves the ε -calibration testing problem with n samples, if given n i.i.d. draws from a distribution \mathcal{D} over $[0, 1] \times \{0, 1\}$, \mathcal{A} returns either “yes” or “no” and satisfies the following with probability $\geq \frac{2}{3}$.⁵*

- \mathcal{A} returns “no” if $\text{dCE}(\mathcal{D}) \geq \varepsilon$.
- \mathcal{A} returns “yes” if $\text{dCE}(\mathcal{D}) = 0$.

In this case, we also call \mathcal{A} an ε -calibration tester.

To our knowledge, we are the first to formalize the calibration testing problem in Definition 2, which is natural from the perspective of *property testing*, an influential paradigm in statistical learning [Ron08, Ron09, Gol17]. In particular, there is an $\varepsilon_n = \Theta(n^{-1/2})$ so that it is information-theoretically impossible to solve the ε_n -calibration testing problem from n samples (see Lemma 5), so a variant of Definition 2 with an exact distinguishing threshold between “calibrated/uncalibrated” is not tractable. Hence, Definition 2 only requires distinguishing distributions \mathcal{D} which are “clearly uncalibrated” (parameterized by a threshold ε) from those which are perfectly calibrated.

We note that a variant of Definition 2 where dCE is replaced by variants of the ECE was recently proposed by [LHHD23]. However, due to the aforementioned discontinuity and binning choice issues which plague the ECE, [LHHD23] posed as an explicit open question whether an alternative calibration metric makes for a more appropriate calibration testing problem, motivating our Definition 2. Indeed, Proposition 9 of [LHHD23] shows that without smoothness assumptions on the data distribution, it is impossible to solve the ECE calibration testing problem from finite samples.⁶

Our first algorithmic contribution is a nearly-linear time algorithm for calibration testing.

Theorem 1. *Let $n \in \mathbb{N}$, and $\varepsilon = \Omega(\varepsilon_n)$, where $\varepsilon_n = \Theta(n^{-1/2})$ is minimal such that it is information-theoretically possible to solve the ε_n -calibration testing problem (Definition 2) with n samples. There is an algorithm which solves the ε -calibration testing problem with n samples, running in time $O(n \log^2(n))$.*

The lower bound on the acceptable range of ε_n in Theorem 1 is well-known, and recalled in Lemma 5 for completeness. Our main contribution is to prove the upper bound (i.e., achieving $O(\varepsilon_n)$ -calibration testing) in Theorem 1 by designing a new algorithm for computing $\text{smCE}(\hat{\mathcal{D}}_n)$, the *smooth calibration error* (Definition 3), an alternative calibration measure, of an empirical distribution $\hat{\mathcal{D}}_n$.

Definition 3 (Smooth calibration error). *Let W be the set of Lipschitz functions $w : [0, 1] \rightarrow [-1, 1]$. The smooth calibration error of distribution \mathcal{D} over $[0, 1] \times \{0, 1\}$, denoted $\text{smCE}(\mathcal{D})$, is*

$$\text{smCE}(\mathcal{D}) = \sup_{w \in W} \left| \mathbb{E}_{(v,y) \sim \mathcal{D}} [(y - v)w(v)] \right|.$$

It was shown in [BGHN23a] that $\text{smCE}(\mathcal{D})$ is a constant-factor approximation to $\text{dCE}(\mathcal{D})$ for all \mathcal{D} on $[0, 1] \times \{0, 1\}$ (see Lemma 4). Additionally, the empirical smCE admits a representation as a linear program with an $O(n) \times O(n)$ -sized constraint matrix encoding Lipschitz constraints.⁷ Thus, [BGHN23a] proposed a simple procedure for estimating $\text{smCE}(\mathcal{D})$: draw n samples from \mathcal{D} , and solve the associated linear program on the empirical distribution. While there have been significant recent runtime advances in the linear programming literature [LS14, CLS21, vdBLSS20, vdBLL⁺21], all state-of-the-art black-box linear programming algorithms solve linear systems involving the constraint matrix, which takes $\Omega(n^\omega)$ time, where $\omega > 2.371$ [WXXZ23] is the current exponent of

⁵As is standard in property testing problems, the success probability of either a calibration tester or a tolerant calibration tester can be boosted to $1 - \delta$ for any $\delta \in (0, 1)$ at a $k = O(\log(\frac{1}{\delta}))$ overhead in the sample complexity. This is because we can independently call k copies of the tester and output the majority vote, which succeeds with probability $\geq 1 - \delta$ by Chernoff bounds, so we focus on $\delta = \frac{1}{3}$.

⁶The work [LHHD23] considered k -class prediction tasks, extending our focus on binary classification ($k = 2$), which we believe is an exciting future direction. However, their Proposition 9 holds even when $k = 2$.

⁷Formally, the number of constraints in the smCE linear program is $O(n^2)$, but we show that in the hard-constrained setting, requiring that “adjacent” constraints are met suffices (see Lemma 1).

matrix multiplication. Even under the best-possible assumption that $\omega = 2$, the strategy of exactly solving a linear program represents an $\Omega(n^2)$ quadratic runtime barrier for calibration testing.

We bypass this barrier by noting that the smCE linear program is highly-structured, and can be reformulated as minimum-cost flow on a planar graph. We believe this observation is already independently interesting, as it opens the door to using powerful software packages designed for efficiently solving flow problems to measure calibration in practice. Moreover, using recent theoretical breakthroughs in graph algorithms [DGG⁺22, CKL⁺22] as a black box, this observation readily implies an $O(n \cdot \text{polylog}(n))$ -time algorithm for solving the smooth calibration linear program.

However, these aforementioned algorithms are quite complicated, and implementations in practice are not available, leaving their relevance to empirical calibration testing unclear at the moment. Motivated by this, in Section 2 we develop a custom solver for the minimum-cost flow reformulation of empirical smooth calibration, based on dynamic programming. Our theoretical runtime improvement upon [DGG⁺22, CKL⁺22] is by at least a large polylogarithmic factor, and moreover our algorithm is simple enough to implement in practice, where it attains faster runtimes than general-purpose commercial solvers on moderate or large dataset sizes, evaluated in Section 3.

We further define a *tolerant* variant of Definition 2 (see Definition 4), where we allow for error thresholds in both the “yes” and “no” cases; “yes” is the required answer when $\text{dCE}(\mathcal{D}) \leq \varepsilon_2$, and “no” is required when $\text{dCE}(\mathcal{D}) \geq \varepsilon_1$. Our algorithm in Theorem 1 continues to serve as an efficient tolerant calibration tester when $\varepsilon_1 \geq 4\varepsilon_2$, formally stated in Theorem 3. This constant-factor loss comes from a similar loss in the relationship between smCE and dCE, see Lemma 4. We make the observation that a constant factor loss in the tolerant testing parameters is inherent following this strategy, via a lower bound in Lemma 13. Thus, even given infinite samples, computing the smooth calibration error cannot solve tolerant calibration testing all the way down to the information-theoretic threshold $\varepsilon_1 \geq \varepsilon_2$. To develop an improved tolerant calibration tester, we directly show how to approximate the LDTC of an empirical distribution, our second main algorithmic contribution.

Theorem 2 (Informal, see Theorem 4, Corollary 3). *Let $n \in \mathbb{N}$, and let $\varepsilon_1 - \varepsilon_2 = \Omega(\varepsilon_n)$, where $\varepsilon_n = \Theta(n^{-1/2})$ is minimal such that it is information-theoretically possible to solve the ε_n -calibration testing problem (Definition 4) with n samples. There is an algorithm which solves the $(\varepsilon_1, \varepsilon_2)$ -tolerant calibration testing problem with n samples, running in time $O(\frac{n \log(n)}{(\varepsilon_1 - \varepsilon_2)^2}) = O(n^2 \log(n))$.*

While Theorem 2 is slower than Theorem 1, it directly approximates the LDTC, making it applicable to tolerant calibration testing. We mention that state-of-the-art black-box linear programming based solvers, while still applicable to (a discretization of) the empirical LDTC, require $\Omega(n^{2.5})$ time [vdBLL⁺21], even if $\omega = 2$. This is because the constraint matrix for the ε -approximate empirical LDTC linear program has dimensions $O(\frac{n}{\varepsilon}) \times O(n)$, resulting in an $\approx \frac{1}{\varepsilon} = \Omega(\sqrt{n})$ overhead in the dimension of the decision variable. We prove Theorem 2 in Appendix C, where we use recent advances in minimax optimization [JT23] and a custom combinatorial rounding procedure to develop a faster algorithm, improving state-of-the-art linear programming runtimes by an $\Omega(\sqrt{n})$ factor.

In Appendix D, we complement our algorithmic results with lower bounds (Theorems 5, 6) on the sample complexity required to solve variants of the testing problem in Definition 2, when dCE is replaced with different calibration measures. For several widely-used distances in the machine learning literature, including binned and convolved variants of ECE [NCH15, BN23], we show that $\tilde{\Omega}(\varepsilon^{-2.5})$ samples are required to the associated ε -calibration testing problem. This demonstrates a statistical advantage of our focus on dCE as our ground-truth notion for calibration testing.

We corroborate our theoretical findings with experimental evidence on real and synthetic data in Section 3. First, on a simple Bernoulli example, we show that dCE and smCE testers are more reliable measures of calibration than a recently-proposed binned ECE variant. We then apply our smCE tester to postprocessed neural network predictions to test their calibration levels, validating against the findings in [GPSW17]. Finally, we implement our method from Theorem 1 on our Bernoulli dataset, showing that it scales to high dimensions and runs faster than both a linear program solver from CVXPY for computing the empirical smCE, as well as a commercial minimum-cost flow solver from Gurobi Optimization (combined with our reformulation in Lemma 2).⁸

⁸Our code is included in the supplementary material.

1.2 Our techniques

Theorems 1 and 2 follow from designing custom algorithms for approximating empirical linear programs associated with the smCE and dCE of a sampled dataset $\widehat{\mathcal{D}}_n := \{(v_i, y_i)\}_{i \in [n]} \sim_{\text{i.i.d.}} \mathcal{D}$. In both cases, generalization bounds from [BGHN23a] show it suffices to approximate the value of the empirical calibration measures to error $\varepsilon = \Omega(n^{-1/2})$, though our solver in Theorem 1 will be exact.

We begin by explaining our strategy for estimating $\text{smCE}(\widehat{\mathcal{D}}_n)$ (Definition 3). By definition, the smooth calibration error of $\widehat{\mathcal{D}}_n$ can be formulated as a linear program,

$$\min_{x \in [-1, 1]^n} \frac{1}{n} \sum_{i \in [n]} x_i (v_i - y_i), \text{ where } |x_i - x_j| \leq |v_i - v_j| \text{ for all } (i, j) \in [n] \times [n]. \quad (2)$$

Here, $x_i \in [-1, 1]$ corresponds to the weight on v_i , and there are $2 \binom{n}{2}$ constraints on the decision variable x , each of which corresponds to a Lipschitz constraint. We make the simple observation that every Lipschitz inequality constraint can be replaced by two constraints of the form $x_i - x_j \leq |v_j - v_i|$ (with i, j swapped). Moreover, the box constraints $x \in [-1, 1]^n$ can be handled by introducing a dummy variable x_{n+1} and writing $\max(x_i - x_{n+1}, x_{n+1} - x_i) \leq 1$, after penalizing x_{n+1} appropriately in the objective. Notably, this substitution makes every constraint the difference of two decision variables, which is enforceable using the edge-vertex incidence matrix of a graph. Finally, the triangle inequality implies that we only need to enforce Lipschitz constraints in (2) corresponding to adjacent i, j . After making these simplifications, the result is the dual of a minimum-cost flow problem on a graph which is the union of a star and a path; this argument is carried out in Lemma 2.

Because of the sequential structure of the induced graph, we show in Appendix B.2 that a dynamic programming-based approach, which maintains the minimum-cost flow value after committing to the first $i < n$ flow variables in the graph recursively, succeeds in computing the value (2). To implement each iteration of our dynamic program in polylogarithmic time, we rely on a generalization of the classical segment tree data structure that we develop in Appendix B.3; combining gives Theorem 1.

On the other hand, the linear program corresponding to the empirical dCE is more complex (with two types of constraints), and to our knowledge lacks the graphical structure to be compatible with the aforementioned approach. Moreover, it is not obvious how to use first-order methods, an alternative linear programming framework suitable when only approximate answers are needed, to solve this problem more quickly. This is because the empirical dCE linear program enforces hard constraints to a set that is difficult to project to under standard distance metrics. To develop our faster algorithm in Theorem 2, we instead follow an “augmented Lagrangian” method where we lift the constraints directly into the objective as a soft-constrained penalty term. To prove correctness of this lifting, we follow a line of results in combinatorial optimization [She13, JST19]. These works develop a “proof-by-rounding algorithm” framework to show that the hard-constrained and soft-constrained linear programs have equal values, summarized in Appendix C.1 (see Lemma 14).

To use this augmented Lagrangian framework, it remains to develop an appropriate rounding algorithm to the feasible polytope for the empirical dCE linear program, which enforces two types of constraints: marginal satisfaction of (v, y) , and calibration of (u, y) (using notation from Definition 1). In Appendix C.3, we design a two-step rounding procedure, which first fixes the marginals on the (v, y) coordinates, and then calibrates the u coordinates without affecting any (v, y) marginal.

1.3 Related work

The calibration performance of deep neural networks has been studied extensively in the literature (e.g. [GPSW17, MDR⁺21b, Rt21a, BGHN23b]). Measuring the calibration error in a meaningful way can be challenging, especially when the predictions are not naturally discretized (e.g. in neural networks). Recently, [BGHN23a] addresses this challenge using the *distance to calibration* as a central notion. They consider a calibration measure to be *consistent* if it is polynomially-related to the distance to calibration. Consistent calibration measures include the smooth calibration error [KF04], Laplace kernel calibration error [KSJ18], interval calibration error [BGHN23a], and convolved ECE [BN23].⁹

⁹The calibration measure we call the *convolved ECE* in our work was originally called the *smooth ECE* in [BN23]. We change the name slightly to reduce overlap with the smooth calibration error (Definition 3), a central object throughout the paper.

On the algorithmic front, substantial observations were made by [BGHN23a] on linear programming characterizations of calibration measures such as the LDTC and smooth calibration. While there have been significant advances on the runtime frontier of linear programming solvers, current runtimes for handling an $n \times d$ linear program constraint matrix with $n \geq d$ remain $\Omega(\min(nd + d^{2.5}, n^\omega))$ [CLS21, vdBLL⁺21, JSWZ21]. Our constraint matrix is roughly-square and highly-sparse, so it is plausible that e.g. the recent research on sparse linear system solvers [PV21, Nie22] could apply to the relevant Newton’s method subproblems and improve upon these rates. Moreover, while efficient estimation algorithms have been proposed by [BGHN23a] for (surrogate) interval calibration error and by [BN23] for convolved ECE, these algorithms require suboptimal sample complexity for solving our testing task in Definition 2 (see Appendix D). To compute their respective distances to error ε from samples, these algorithms require $\Omega(\varepsilon^{-5})$ and $\Omega(\varepsilon^{-3})$ time. As comparison, under this parameterization Theorems 1 and 2 require $\tilde{O}(\varepsilon^{-2})$ and $\tilde{O}(\varepsilon^{-4})$ time, but can solve stronger testing problems with the same sample complexity, experimentally validated in Section 3.

Notation. Throughout, \mathcal{D} denotes a distribution over $[0, 1] \times \{0, 1\}$. When \mathcal{D} is clear from context, we let $\widehat{\mathcal{D}}_n = \{(v_i, y_i)\}_{i \in [n]}$ denote a dataset of n independent samples from \mathcal{D} and, in a slight abuse of notation, the distribution with probability $\frac{1}{n}$ for each (v_i, y_i) . We say d is a *calibration measure* if it takes distributions on $[0, 1] \times \{0, 1\}$ to the nonnegative reals $\mathbb{R}_{\geq 0}$, so $\underline{\text{dCE}}$ (Definition 1) and smCE (Definition 3) are both calibration measures. We use \tilde{O} and $\tilde{\Omega}$ to hide polylogarithmic factors in the argument. We denote $[n] := \{i \in \mathbb{N} \mid i \leq n\}$. We denote matrices in boldface throughout. For any $\mathbf{A} \in \mathbb{R}^{m \times n}$, we refer to its i^{th} row by $\mathbf{A}_{i \cdot}$ and its j^{th} column by $\mathbf{A}_{\cdot j}$. For a set S identified with rows of a matrix \mathbf{A} , we let \mathbf{A}_S denote the row indexed by $S \in S$, and use similar notation for columns. For a directed graph $G = (V, E)$, we define its edge-vertex incidence matrix $\mathbf{B} \in \{-1, 0, 1\}^{E \times V}$ which has a row corresponding to each $e = (u, v) \in E$ with $\mathbf{B}_{eu} = 1$ and $\mathbf{B}_{ev} = -1$. When G is undirected, we similarly define $\mathbf{B} \in \{-1, 0, 1\}^{E \times V}$ with arbitrary edge orientations.

2 Smooth calibration

In this section, we overview our main result on approximating the smooth calibration of a distribution on $[0, 1] \times \{0, 1\}$, deferring some aspects of the proof to Appendix B. We first show that the linear program corresponding to the smooth calibration of an empirical distribution can be reformulated as an instance of minimum-cost flow on a highly-structured graph. We then explain our dynamic programming approach to solving this minimum-cost flow problem and state a runtime guarantee. Finally, we give our main result on near-linear time calibration testing, Theorem 1.

Throughout this section, we fix a dataset under consideration, $\widehat{\mathcal{D}}_n := \{(v_i, y_i)\}_{i \in [n]} \subset [0, 1] \times \{0, 1\}$, and the corresponding empirical distribution (which, in an abuse of notation, we also denote $\widehat{\mathcal{D}}_n$), i.e. we use $(v, y) \sim \widehat{\mathcal{D}}_n$ to mean that $(v, y) = (v_i, y_i)$ with probability $\frac{1}{n}$ for each $i \in [n]$. We also assume without loss of generality that the $\{v_i\}_{i \in [n]}$ are in sorted order, so $0 \leq v_1 \leq \dots \leq v_n \leq 1$. Recalling Definition 3, the associated empirical smooth calibration linear program is

$$\begin{aligned} \text{smCE}(\widehat{\mathcal{D}}_n) &:= \max_{x \in [-1, 1]^n} b^\top x, \\ \text{where } |x_i - x_j| &\leq v_j - v_i \text{ for all } (i, j) \in [n] \times [n] \text{ with } i < j, \\ \text{and } b_i &:= \frac{1}{n}(y_i - v_i) \text{ for all } i \in [n]. \end{aligned} \tag{3}$$

We first make a simplifying observation, which shows that it suffices to replace the Lipschitz constraints in (3) with only the Lipschitz constraints corresponding to adjacent indices (i, j) .

Lemma 1. *If $x, v \in \mathbb{R}^n$, where v has monotonically nondecreasing coordinates, and $|x_i - x_{i+1}| \leq v_{i+1} - v_i$ for all $i \in [n - 1]$, then $|x_i - x_j| \leq v_j - v_i$ for all $(i, j) \in [n] \times [n]$ with $i < j$.*

We now reformulate (3) as a (variant of a) minimum-cost flow problem.

Lemma 2. *Consider an instance of (3). Let $G = (V, E)$ be an undirected graph on $n + 1$ vertices labeled by $V := [n + 1]$, and with $2n - 1$ directed edges E defined and with edge costs as follows.*

- For all $i \in [n - 1]$, there is an edge between vertices $(i, i + 1)$ with edge cost $v_{i+1} - v_i$.

- For all $i \in [n]$, there is an edge between vertices $(i, n + 1)$ with edge cost 1.

Let $c \in \mathbb{R}^E$ be the vector of all edge costs, let $d \in \mathbb{R}^{n+1}$ be the demand vector which concatenates $-b$ in (3) with a last coordinate set to $\sum_{i \in [n]} b_i$, and let $\mathbf{B} \in \{-1, 0, 1\}^{E \times V}$ be the edge-vertex incidence matrix of G . Then the problem

$$\min_{\substack{f \in \mathbb{R}^E \\ \mathbf{B}^\top f = d}} c^\top |f| := \sum_{e \in E} c_e |f_e| \quad (4)$$

has the same value as the empirical smooth calibration linear program (3).

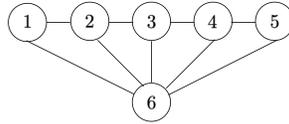


Figure 1: Example graph G for $n = 5$ with $n + 1 = 6$ vertices and $2n - 1 = 9$ edges.

Proof. By Lemma 1, solving (3) is equivalent to solving

$$\min_{x \in [-1, 1]^n} -b^\top x, \text{ where } |x_i - x_{i+1}| \leq v_{i+1} - v_i \text{ for all } i \in [n - 1], \quad (5)$$

We create a dummy variable x_{n+1} , and rewrite (5) as

$$\min_{x \in \mathbb{R}^{n+1}} \sum_{i \in [n]} -b_i(x_i - x_{n+1}), \text{ where } |x_i - x_{i+1}| \leq v_{i+1} - v_i \text{ for all } i \in [n - 1], \quad (6)$$

and $-1 \leq x_i - x_{n+1} \leq 1$ for all $i \in [n]$.

Next, consider a directed graph $\tilde{G} = (V, \tilde{E})$ with $4n - 2$ edges which duplicate the undirected edges described in the lemma statement in both directions. Let $\tilde{\mathbf{B}} \in \{-1, 0, 1\}^{\tilde{E} \times V}$ be the edge-vertex incidence matrix of \tilde{G} , and let $\tilde{c} \in \mathbb{R}^{\tilde{E}}$ be the edge cost vector so that both edges in \tilde{E} corresponding to $e \in E$ have the same cost c_e . Then (6) is equivalent to the linear program $\max_{x \in \mathbb{R}^{n+1}} d^\top x$ such that $\tilde{\mathbf{B}}x \leq c$, where d is described as in the lemma statement. The dual of this linear program is

$$\min_{\substack{f \in \mathbb{R}_{\geq 0}^{\tilde{E}} \\ \tilde{\mathbf{B}}^\top f = d}} \tilde{c}^\top f, \quad (7)$$

a minimum-cost flow problem on \tilde{G} . In particular, based on the way we defined $\tilde{\mathbf{B}}$, we can check that $\tilde{\mathbf{B}}^\top f$ encodes the net flow at each vertex of \tilde{G} , which is set according to the demand vector d in the above optimization problem. Next, for each pair of directed edges (e', e'') in \tilde{G} corresponding to some $e \in E$, note that an optimal solution to (7) will only put nonzero flow on one of e' or e'' , else we can achieve a smaller cost by canceling out redundant flow. Therefore, we can collapse each pair of directed edges into a single undirected edge, where we allow the flow variable f to be negative but charge its magnitude $|f|$ in cost, proving equivalence of (7) and (4) as claimed. \square

We believe this observation in Lemma 2 is already interesting, as it lets us to use specialized graph algorithms to achieve faster runtimes in both theory and practice for solving (3). By using the special structure of the graph (the union of a star and path), we show in Appendix B that we can develop a more efficient custom algorithm for this problem. Specifically, we show how to replace the constrained problem (4) with an unconstrained problem on only the path edges, of the form

$$\min_{f \in \mathbb{R}^{n-1}} A(f) := |d_1 + f_1| + |d_n - f_{n-1}| + \sum_{i \in [n-2]} |f_i - f_{i+1} - d_{i+1}| + \sum_{i \in [n-1]} c_i |f_i|. \quad (8)$$

We prove the following result in Appendix B.2.

Proposition 1. *There is an algorithm which computes a minimizer $f \in \mathbb{R}^{n-1}$ to A in (8), as well as the minimizing value $A(f)$, in time $O(n \log^2(n))$.*

Our algorithm for establishing Proposition 1 is based on dynamic programming, and recursively represents partial solutions to A as a piecewise-linear function. We implement updates to this representation via a *segment tree* data structure in polylogarithmic time, giving our overall solution.

Corollary 1. *There is an algorithm which computes the value of (3) in time $O(n \log^2(n))$.*

Proof. This is immediate from Lemma 2, the equivalence between the constrained problem (4) and the unconstrained problem (8) established in Appendix B.2, and Proposition 1. \square

We now describe how to build upon Corollary 1 to give an algorithm for proving Theorem 1, using a result from [BGHN23a] which bounds how well the smooth calibration of an empirical distribution approximates the smooth calibration of the population.

Lemma 3 (Corollary 9.9, [BGHN23a]). *For $\varepsilon \in (0, 1)$, there is an $n = O(\frac{1}{\varepsilon^2})$ such that if $\widehat{\mathcal{D}}_n$ is the empirical distribution over n i.i.d. draws from \mathcal{D} , with probability $\geq \frac{2}{3}$, $|\text{smCE}(\mathcal{D}) - \text{smCE}(\widehat{\mathcal{D}}_n)| \leq \varepsilon$.*

Further, we recall the smooth calibration error is constant-factor related to the LDTC.

Lemma 4 (Theorem 7.3, [BGHN23a]). *For any distribution \mathcal{D} over $[0, 1] \times \{0, 1\}$, we have $\frac{1}{2}\text{dCE}(\mathcal{D}) \leq \text{smCE}(\mathcal{D}) \leq 2\text{dCE}(\mathcal{D})$.*

Proof of Theorem 1. We take $n = O(\frac{1}{\varepsilon^2})$ samples so Lemma 3 ensures $|\text{smCE}(\mathcal{D}) - \text{smCE}(\widehat{\mathcal{D}}_n)| \leq \frac{\varepsilon}{2}$ with probability $\geq \frac{2}{3}$. We then compute $\beta = \text{smCE}(\widehat{\mathcal{D}}_n)$ using Corollary 1, and return “yes” iff $\beta \leq \frac{\varepsilon}{4}$, which distinguishes between the two cases in Definition 2 via Lemma 4. \square

3 Experiments

In this section, we present experiments on synthetic data and CIFAR-100 supporting our argument that dCE and smCE are reliable measures of calibration for use in defining a testing problem. We then evaluate our custom algorithms from Section 2 and Appendix B, showing promising results on their runtimes outperforming standard packages for linear programming and minimum-cost flow. The experiments in the first and third part of this section are run on a 2018 laptop with 2.2 GHz 6-Core Intel Core i7 processor. The experiments in the second part are run on a cluster using 2x AMD EPYC 7763 64-Core Processor and a single NVIDIA A100 PCIE 40GB.

Synthetic dataset. In our first experiment, we considered the ability of ε -d-testers (Definition 5) to detect the miscalibration of a synthetic dataset, for various levels of $\varepsilon \in \{0.01, 0.03, 0.05, 0.07, 0.1\}$, and various choices of $d \in \{\text{smCE}, \text{dCE}, \text{ConvECE}\}$.¹⁰ The synthetic dataset we used is n independent draws from \mathcal{D} , where a draw $(v, y) \sim \mathcal{D}$ first draws $v \sim_{\text{unif.}} [0, 1 - \varepsilon^*]$, and $y \sim \text{Bern}(v + \varepsilon^*)$, for $\varepsilon^* := 0.01$.¹¹ Note that $\text{dCE}(\mathcal{D}) = \varepsilon^* = 0.01$, by the proof in Lemma 13. In Table 1, where the columns index n (the number of samples), for each choice of d we report the smallest value of ε such that a majority of 100 runs of an ε -d-tester report “yes.” For $d = \text{ConvECE}$, we implemented our tester by running code in [BN23] to compute ConvECE and thresholding at $\frac{\varepsilon}{2}$. For $d \in \{\text{smCE}, \text{dCE}\}$, we used the standard linear program solver from CVXPY [DB16, AVDB18] and again thresholded at $\frac{\varepsilon}{2}$. We remark that the CVXPY solver, when run on the dCE linear program, fails to produce stable results for $n > 2^9$ due to the size of the constraint matrix. As seen from Table 1, both smCE and dCE testers are more reliable estimators of the ground truth calibration error ε^* than ConvECE.

Table 1: Calibration testing thresholds (smallest passed on half of 100 runs).

n	$2^6 + 1$	$2^7 + 1$	$2^8 + 1$	$2^9 + 1$	$2^{10} + 1$	$2^{11} + 1$
smCE	0.07	0.05	0.03	0.03	0.01	0.01
dCE	0.03	0.01	0.01			
cECE	0.1	0.1	0.07	0.07	0.05	0.03
Ground Truth	0.01	0.01	0.01	0.01	0.01	0.01

¹⁰We implemented ConvECE using code from [BN23], which automatically conducts a search for σ .

¹¹This is a slight variation on the synthetic dataset used in [BGHN23a].

Table 2: Empirical smCE on postprocessed DenseNet40 predictions (median over 20 runs)

\mathcal{D}	$\mathcal{D}_{\text{base}}$	\mathcal{D}_{iso}	$\mathcal{D}_{\text{temp}}$
Empirical smCE	0.2269	0.2150	0.1542

In Figure 2, we plot the median error with error bars for each calibration measure, where the x axis denotes $\log_2(n - 1)$, and results are reported over 100 runs.

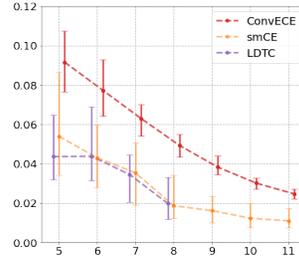


Figure 2: The 25% quantile, median, and 75% quantile (over 100 runs) for smCE, dCE and cECE respectively. The x -axis is for dataset with size $2^x + 1$.

Postprocessed neural networks. In [GPSW17], which observed modern deep neural networks may be very miscalibrated, various strategies were proposed for postprocessing network predictions to calibrate them. We evaluate two of these strategies using our testing algorithms. We trained a DenseNet40 model [HLvdMW17] on the CIFAR-100 dataset [Kri09], producing a distribution $\mathcal{D}_{\text{base}}$, where a draw $(v, y) \sim \mathcal{D}_{\text{base}}$ selects a random example from the test dataset, sets y to be its label, and v to be the prediction of the neural network. We also learned calibrating postprocessing functions f_{iso} and f_{temp} from the training dataset, the former via isotonic regression and the latter via temperature scaling. These induce (ideally, calibrated) distributions $\mathcal{D}_{\text{iso}}, \mathcal{D}_{\text{temp}}$, where a draw from \mathcal{D}_{iso} samples $(v, y) \sim \mathcal{D}_{\text{base}}$ and returns $(f_{\text{iso}}(v), y)$, and $\mathcal{D}_{\text{temp}}$ is defined analogously. The neural network and postprocessing functions were all trained by adapting code from [GPSW17].

We computed the median smooth calibration error of 20 runs of the following experiment. In each run, for each $\mathcal{D} \in \{\mathcal{D}_{\text{base}}, \mathcal{D}_{\text{iso}}, \mathcal{D}_{\text{temp}}\}$, we drew 256 random examples from \mathcal{D} , and computed the average smooth calibration error smCE of the empirical dataset using a linear program solver from CVXPY. We report our findings in Table 2. We also compared computing smCE using the CVXPY solver and a commercial minimum-cost flow solver from Gurobi Optimization [Opt23] (on the objective from Lemma 2) in this setting. The absolute difference between outputs is smaller than 10^{-5} in all cases, verifying that minimum-cost flow solvers accurately measure smooth calibration.

Qualitatively, our results (based on smCE) agree with findings in [GPSW17] (based on binned variants of ECE), in that temperature scaling appears to be the most effective postprocessing technique.

smCE tester. Finally, we evaluated the efficiency of our proposed approaches to computing the empirical smCE. Specifically, we measure the runtime of four solvers for computing (3): a linear program solver from CVXPY, a commercial minimum-cost flow solver from Gurobi Optimization, a naïve implementation of our algorithm from Corollary 1 using Python, and a slightly-optimized implementation using the PyPy package [PyP19]. We use the same experimental setup as in Table 1, i.e. measuring calibration of a uniform predictor on a miscalibrated synthetic dataset, with $\varepsilon^* = 0.01$.¹² In Table 3, we report the average runtimes for each trial (across 10 runs), varying the sample size. Again, the absolute difference between the outputs of all methods is negligible ($\leq 10^{-9}$ in all cases). As seen in Table 3, our custom algorithm (optimized with PyPy) outperforms standard packages from CVXPY and Gurobi Optimization starting from moderate sample sizes. We believe that Table 3 demonstrates that our new algorithms are a scalable, reliable way of testing calibration, and that these performance gains may be significantly improvable by further optimizing our code.

¹²We found similar runtime trends when using our algorithms to test calibration on the postprocessed neural network dataset, but the runtime gains were not as drastic as the sample size $n = 2^8$ was smaller in that case.

Acknowledgements

We thank Edgar Dobriban for pointing us to the reference [LHHD23], and Yang P. Liu and Richard Peng for helpful discussions on the segment tree data structure in Section B.3. We also thank Yue Zhao for advice on running our experiments.

Table 3: Runtimes (in seconds) for computing the value of (3), using various solvers

n	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}
CVXPY LP solver	0.105	0.370	1.58	6.51	45.7	245
Gurobi minimum-cost flow solver	0.063	0.179	0.238	0.539	1.45	3.19
Solver from Corollary 1	0.177	0.389	0.899	2.01	4.66	10.6
Solver from Corollary 1 with PyPy	0.079	0.115	0.176	0.307	0.621	2.05

References

- [AAA⁺16] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Erich Elsen, Jesse H. Engel, Linxi Fan, Christopher Fougner, Awni Y. Hannun, Billy Jun, Tony Han, Patrick LeGresley, Xiangang Li, Libby Lin, Sharan Narang, Andrew Y. Ng, Sherjil Ozair, Ryan Prenger, Sheng Qian, Jonathan Raiman, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Chong Wang, Yi Wang, Zhiqian Wang, Bo Xiao, Yan Xie, Dani Yogatama, Jun Zhan, and Zhenyao Zhu. Deep speech 2 : End-to-end speech recognition in english and mandarin. In *Proceedings of the 33rd International Conference on Machine Learning, ICML 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 173–182. JMLR.org, 2016.
- [AVDB18] Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.
- [BGHN23a] Jarosław Błasiok, Parikshit Gopalan, Lunjia Hu, and Preetum Nakkiran. A unifying theory of distance from calibration. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 1727–1740, 2023.
- [BGHN23b] Jarosław Błasiok, Parikshit Gopalan, Lunjia Hu, and Preetum Nakkiran. When does optimizing a proper loss yield calibration? *arXiv preprint arXiv:2305.18764*, 2023.
- [BN23] Jarosław Błasiok and Preetum Nakkiran. Smooth ECE: Principled reliability diagrams via kernel smoothing. *arXiv preprint arXiv:2309.12236*, 2023.
- [Can22] Clément L. Canonne. Topics and techniques in distribution testing: A biased but representative sample. *Foundations and Trends® in Communications and Information Theory*, 19(6):1032–1198, 2022.
- [CKL⁺22] Li Chen, Rasmus Kyng, Yang P Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 612–623. IEEE, 2022.
- [CLS21] Michael B. Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. *J. ACM*, 68(1):3:1–3:39, 2021.
- [CST21] Michael B. Cohen, Aaron Sidford, and Kevin Tian. Relative lipschitzness in extragradient methods and a direct recipe for acceleration. In *12th Innovations in Theoretical Computer Science Conference, ITCS 2021*, volume 185 of *LIPICs*, pages 62:1–62:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.
- [DB16] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

- [DGG⁺22] Sally Dong, Yu Gao, Gramoz Goranci, Yin Tat Lee, Richard Peng, Sushant Sachdeva, and Guanghao Ye. Nested dissection meets ipms: Planar min-cost flow in nearly-linear time. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022*, pages 124–153. SIAM, 2022.
- [DKR⁺21] Cynthia Dwork, Michael P. Kim, Omer Reingold, Guy N. Rothblum, and Gal Yona. Outcome indistinguishability. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, 2021*, pages 1095–1108. ACM, 2021.
- [Doi07] Kunio Doi. Computer-aided diagnosis in medical imaging: historical review, current status and future potential. *Computerized medical imaging and graphics*, 31(4–5):198–211, 2007.
- [GKR⁺22] Parikshit Gopalan, Adam Tauman Kalai, Omer Reingold, Vatsal Sharan, and Udi Wieder. Omnipredictors. In *13th Innovations in Theoretical Computer Science Conference, ITCS 2022*, volume 215 of *LIPICs*, pages 79:1–79:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.
- [Gol17] Oded Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017.
- [GPSW17] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017.
- [HKRR18] Úrsula Hébert-Johnson, Michael P. Kim, Omer Reingold, and Guy N. Rothblum. Multicalibration: Calibration for the (computationally-identifiable) masses. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1944–1953. PMLR, 2018.
- [HLvdMW17] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, pages 2261–2269. IEEE Computer Society, 2017.
- [IS03] Y. I. Ingster and I. A. Suslina. *Nonparametric Goodness-of-Fit Testing under Gaussian Models*, volume 169 of *Lecture Notes in Statistics*. Springer-Verlag, New York, 2003.
- [JST19] Arun Jambulapati, Aaron Sidford, and Kevin Tian. A direct $\tilde{O}(1/\varepsilon)$ iteration parallel algorithm for optimal transport. *Advances in Neural Information Processing Systems*, 32, 2019.
- [JSWZ21] Shunhua Jiang, Zhao Song, Omri Weinstein, and Hengjie Zhang. A faster algorithm for solving general lps. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 823–832, 2021.
- [JT23] Arun Jambulapati and Kevin Tian. Revisiting area convexity: Faster box-simplex games and spectrahedral generalizations. *arXiv preprint arXiv:2303.15627*, 2023.
- [KF04] Sham M. Kakade and Dean P. Foster. Deterministic calibration and nash equilibrium. In John Shawe-Taylor and Yoram Singer, editors, *Learning Theory*, pages 33–48. Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.
- [KF08] Sham M. Kakade and Dean P. Foster. Deterministic calibration and nash equilibrium. *J. Comput. Syst. Sci.*, 74(1):115–130, 2008.
- [KLM19] Ananya Kumar, Percy Liang, and Tengyu Ma. Verified uncertainty calibration. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, pages 3787–3798, 2019.

- [Kri09] Alex Krizhevsky. Learning multiple layers of features from tiny images. <https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf>, 2009. Accessed: 2024-01-31.
- [KSJ18] Aviral Kumar, Sunita Sarawagi, and Ujjwal Jain. Trainable calibration measures for neural networks from kernel mean embeddings. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2805–2814. PMLR, 10–15 Jul 2018.
- [LHHD23] Donghwan Lee, Xinmeng Huang, Hamed Hassani, and Edgar Dobriban. T-cal: An optimal test for the calibration of predictive models. *Journal of Machine Learning Research*, 24:1–72, 2023.
- [LS14] Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in \tilde{O} (v rank) iterations and faster algorithms for maximum flow. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014*, pages 424–433. IEEE Computer Society, 2014.
- [MDR⁺21a] Matthias Minderer, Josip Djolonga, Rob Romijnders, Frances Hubis, Xiaohua Zhai, Neil Houlsby, Dustin Tran, and Mario Lucic. Revisiting the calibration of modern neural networks. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021*, pages 15682–15694, 2021.
- [MDR⁺21b] Matthias Minderer, Josip Djolonga, Rob Romijnders, Frances Hubis, Xiaohua Zhai, Neil Houlsby, Dustin Tran, and Mario Lucic. Revisiting the calibration of modern neural networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 15682–15694. Curran Associates, Inc., 2021.
- [Mur98] Allan H. Murphy. The early history of probability forecasts: Some extensions and clarifications. *Weather and forecasting*, 13(1):5–15, 1998.
- [MW84] Allan H. Murphy and Robert L. Winkler. Probability forecasting in meteorology. *Journal of the American Statistical Association*, 79(387):489–500, 1984.
- [NCH15] Mahdi Pakdaman Naeini, Gregory F. Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015*, pages 2901–2907. AAAI Press, 2015.
- [NDZ⁺19] Jeremy Nixon, Michael W. Dusenberry, Linchuan Zhang, Ghassen Jerfel, and Dustin Tran. Measuring calibration in deep learning. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2019*, pages 38–41. Computer Vision Foundation / IEEE, 2019.
- [Nie22] Zipei Nie. Matrix anti-concentration inequalities with applications. In *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 568–581. ACM, 2022.
- [Opt23] Gurobi Optimization. Minimum-cost flow. <https://gurobi-optimization-gurobi-optimods.readthedocs-hosted.com/en/latest/mods/min-cost-flow.html>, 2023. Accessed: 2024-05-21.
- [PV21] Richard Peng and Santosh S. Vempala. Solving sparse linear systems faster than matrix multiplication. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021*, pages 504–521. SIAM, 2021.
- [PyP19] PyPy. Pypy. <https://www.pypy.org/>, 2019. Accessed: 2024-05-21.
- [QM22] Benjamin Qi and Dustin Miao. Segment tree beats. <https://usaco.guide/adv/segtree-beats?lang=cpp>, 2022. Accessed: 2024-05-20.

- [Ron08] Dana Ron. Property testing: A learning theory perspective. *Found. Trends Mach. Learn.*, 1(3):307–402, 2008.
- [Ron09] Dana Ron. Algorithmic and analysis techniques in property testing. *Found. Trends Theor. Comput. Sci.*, 5(2):73–205, 2009.
- [RRSK11] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor. *Recommender Systems Handbook*. Springer New York, 2011.
- [Rt21a] Rahul Rahaman and alexandre thiery. Uncertainty quantification and deep ensembles. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 20063–20075. Curran Associates, Inc., 2021.
- [RT21b] Rahul Rahaman and Alexandre H. Thiéry. Uncertainty quantification and deep ensembles. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021*, pages 20063–20075, 2021.
- [She13] Jonah Sherman. Nearly maximum flows in nearly linear time. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013*, pages 263–269. IEEE Computer Society, 2013.
- [She17] Jonah Sherman. Area-convexity, l_∞ regularization, and undirected multicommodity flow. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017*, pages 452–460. ACM, 2017.
- [vdBLL⁺21] Jan van den Brand, Yin Tat Lee, Yang P. Liu, Thatchaphol Saranurak, Aaron Sidford, Zhao Song, and Di Wang. Minimum cost flows, mdps, and ℓ_1 -regression in nearly linear time for dense instances. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, 2021*, pages 859–869. ACM, 2021.
- [vdBLSS20] Jan van den Brand, Yin Tat Lee, Aaron Sidford, and Zhao Song. Solving tall dense linear programs in nearly linear time. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020*, pages 775–788. ACM, 2020.
- [VDDP17] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. Deep learning for computer vision: A brief review. *Computational Intelligence and Neuroscience*, 2018, 2017.
- [WXXZ23] Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. New bounds for matrix multiplication: from alpha to omega. *CoRR*, abs/2307.07970, 2023.

A Additional preliminaries

We first introduce some notation used in Appendices B, C, and D. When applied to a vector, we let $\|\cdot\|_p$ denote the ℓ_p norm for $p \geq 1$. We let $\mathbb{0}_d$ and $\mathbb{1}_d$ denote the all-zeroes and all-ones vectors in dimension d . We let $\Delta^d := \{x \in \mathbb{R}_{\geq 0}^d \mid \|x\|_1 = 1\}$ denote the probability simplex in dimension d . The i^{th} coordinate basis vector is denoted e_i . We say $\tilde{x} \in \mathbb{R}$ is an ε -additive approximation of $x \in \mathbb{R}$ if $|\tilde{x} - x| \leq \varepsilon$. For a set $S \subset \mathbb{R}$, we say another set $T \subset \mathbb{R}$ is an ε -cover of S if for all $s \in S$, there is $t \in T$ with $|s - t| \leq \varepsilon$. For $a, b \in \mathbb{R}$ with $a \leq b$, we let $\text{clip}_{[a,b]}(t) := \min(b, \max(a, t))$ denote the result of projecting $t \in \mathbb{R}$ onto $[a, b]$.

We next formally define our tolerant variant of the calibration testing problem in Definition 2.

Definition 4 (Tolerant calibration testing). *Let $0 \leq \varepsilon_2 \leq \varepsilon_1 \leq 1$. We say algorithm \mathcal{A} solves the $(\varepsilon_1, \varepsilon_2)$ -tolerant calibration testing problem with n samples, if given n i.i.d. draws from a distribution \mathcal{D} over $[0, 1] \times \{0, 1\}$, \mathcal{A} returns either “yes” or “no” and satisfies the following with probability $\geq \frac{2}{3}$.*

- \mathcal{A} returns “no” if $\underline{\text{dCE}}(\mathcal{D}) \geq \varepsilon_1$.
- \mathcal{A} returns “yes” if $\underline{\text{dCE}}(\mathcal{D}) \leq \varepsilon_2$.

In this case, we also call \mathcal{A} an $(\varepsilon_1, \varepsilon_2)$ -tolerant calibration tester.

Note that an algorithm which solves the $(\varepsilon_1, \varepsilon_2)$ -tolerant calibration testing problem with n samples also solves the ε_1 -calibration testing problem with the same sample complexity. Moreover, we give a simple impossibility result on parameter ranges for calibration testing.

Lemma 5. *Let $0 \leq \varepsilon_2 \leq \varepsilon_1 \leq \frac{1}{2}$ satisfy $\varepsilon_1 - \varepsilon_2 = \varepsilon$. There is a universal constant C_{coin} such that, given $n \leq \frac{C_{\text{coin}}}{\varepsilon^2}$ samples from a distribution on $[0, 1] \times \{0, 1\}$, it is information-theoretically impossible to solve the $(\varepsilon_1, \varepsilon_2)$ -tolerant calibration testing problem.*

Proof. Suppose $\varepsilon \leq \frac{1}{10}$, else we can choose C_{coin} small enough such that $n < 1$. We consider two distributions over $[0, 1] \times \{0, 1\}$, \mathcal{D} and \mathcal{D}' , with $\underline{\text{dCE}}(\mathcal{D}) \geq \varepsilon_1$ but $\underline{\text{dCE}}(\mathcal{D}') \leq \varepsilon_2$, so if \mathcal{A} succeeds at tolerant calibration testing for both \mathcal{D} and \mathcal{D}' , we must have $d_{\text{TV}}(\mathcal{D}^{\otimes n}, (\mathcal{D}')^{\otimes n}) \geq \frac{1}{3}$, where we denote the n -fold product of a distribution with $\cdot^{\otimes n}$. Else, \mathcal{A} cannot return different answers from n samples with probability $\geq \frac{2}{3}$, as required by Definition 4. Specifically, we define $\mathcal{D}, \mathcal{D}'$ as follows.

- To draw $(v, y) \sim \mathcal{D}$, let $v = \frac{1}{2} + \varepsilon_1$ and $y \sim \text{Bern}(\frac{1}{2})$.
- To draw $(v, y) \sim \mathcal{D}'$, let $v = \frac{1}{2} + \varepsilon_1$ and $y \sim \text{Bern}(\frac{1}{2} + \varepsilon)$.

We claim that $\underline{\text{dCE}}(\mathcal{D}) = \varepsilon_1$. To see this, let $\Pi \in \text{ext}(\mathcal{D})$ and $(u, v, y) \sim \Pi$, so $\mathbb{E}_{(u,v,y) \sim \Pi}[u] = \frac{1}{2}$ because u is calibrated. By Jensen’s inequality, we have:

$$\mathbb{E}_{(u,v,y) \sim \Pi}[|u - v|] \geq \left| \mathbb{E}_{(u,v,y) \sim \Pi}[u] - \left(\frac{1}{2} + \varepsilon_1\right) \right| = \varepsilon_1.$$

The equality case is realized when $u = \frac{1}{2}$ with probability 1, proving the claim. Similarly, $\underline{\text{dCE}}(\mathcal{D}') = \varepsilon_2$. Finally, let $\pi := \text{Bern}(\frac{1}{2})$, $\pi' := \text{Bern}(\frac{1}{2} + \varepsilon)$, and $\pi^{\otimes n}, (\pi')^{\otimes n}$ denote their n -fold product distributions. Pinsker’s inequality shows that it suffices to show that $d_{\text{KL}}(\pi^{\otimes n} \| (\pi')^{\otimes n}) \leq \frac{1}{5}$ to contradict our earlier claim $d_{\text{TV}}((\mathcal{D})^{\otimes n}, (\mathcal{D}')^{\otimes n}) \geq \frac{1}{3}$. To this end, we have

$$\begin{aligned} d_{\text{KL}}(\pi^{\otimes n} \| (\pi')^{\otimes n}) &= n \cdot d_{\text{KL}}(\pi \| \pi') = \frac{n}{2} \left(\log \left(\frac{\frac{1}{2}}{\frac{1}{2} + \varepsilon} \right) + \log \left(\frac{\frac{1}{2}}{\frac{1}{2} - \varepsilon} \right) \right) \\ &= \frac{n}{2} \log \left(\frac{1}{1 - 4\varepsilon^2} \right) \leq \frac{n}{2} \cdot 5\varepsilon^2 \leq \frac{1}{5}, \end{aligned}$$

where the first line used tensorization of d_{KL} , and the last chose C_{coin} small enough. \square

We also generalize Definitions 2 and 4 to apply to an arbitrary calibration measure.

Definition 5 (d testing). Let d be a calibration measure. For $\varepsilon \in \mathbb{R}_{\geq 0}$, we say algorithm \mathcal{A} solves the ε -d-testing problem (or, \mathcal{A} is an ε -d tester) with n samples, if given n i.i.d. draws from a distribution \mathcal{D} over $[0, 1] \times \{0, 1\}$, \mathcal{A} returns either “yes” or “no” and satisfies the following with probability $\geq \frac{2}{3}$.

1. \mathcal{A} returns “no” if $d(\mathcal{D}) \geq \varepsilon$.
2. \mathcal{A} returns “yes” if $d(\mathcal{D}) = 0$.

For $0 \leq \varepsilon_2 \leq \varepsilon_1$, we say algorithm \mathcal{A} solves the $(\varepsilon_1, \varepsilon_2)$ -tolerant d testing problem (or, \mathcal{A} is an $(\varepsilon_1, \varepsilon_2)$ -tolerant d tester) with n samples, if given n i.i.d. draws from a distribution \mathcal{D} over $[0, 1] \times \{0, 1\}$, \mathcal{A} returns either “yes” or “no” and satisfies the following with probability $\geq \frac{2}{3}$.

1. \mathcal{A} returns “no” if $d(\mathcal{D}) \geq \varepsilon_1$.
2. \mathcal{A} returns “yes” if $d(\mathcal{D}) \leq \varepsilon_2$.

B Appendix for Section 2

B.1 Deferred lemma proofs

Proof of Lemma 1. This follows from the triangle inequality:

$$|x_i - x_j| \leq \sum_{k=i}^{j-1} |x_k - x_{k+1}| \leq \sum_{k=i}^{j-1} v_{k+1} - v_k = v_j - v_i.$$

□

B.2 Dynamic programming

In this section, we give our dynamic programming approach to solving (4), which establishes Proposition 1. The graph G in (4) is the union of a path P on n vertices and a star S to the $(n+1)^{\text{th}}$ vertex. Let us identify the edges in P with $[n-1]$ (where edge i corresponds to vertices $(i, i+1)$), and the edges in S with $[2n-1] \setminus [n-1]$. We first make a simplifying observation, which is that given the coordinates of a flow variable $f \in \mathbb{R}^E$ on the edges in the path P , there is a unique way to set the values of f on the edges in the star S so that the demands $\mathbf{B}^\top f = d$ are satisfied. Concretely, we require

$$\begin{aligned} f_{n-1+i} &= d_i + f_i - f_{i-1} \text{ for all } 2 \leq i \leq n-1, \\ f_n &= d_1 + f_1, \text{ and } f_{2n-1} = d_n - f_{n-1}. \end{aligned}$$

Hence, minimizing the constrained problem in (4) is equivalent to minimizing the following unconstrained problem on the first $n-1$ flow variables, stated in (8) and reproduced here:

$$\min_{f \in \mathbb{R}^{n-1}} A(f) := |d_1 + f_1| + |d_n - f_{n-1}| + \sum_{i \in [n-2]} |f_i - f_{i+1} - d_{i+1}| + \sum_{i \in [n-1]} c_i |f_i|.$$

We now solve (8). We first define a sequence of partial functions $\{A_j : \mathbb{R} \rightarrow \mathbb{R}\}_{j \in [n-1]}$ by

$$\begin{aligned} A_1(z) &:= |d_1 + z| + c_1 |z|, \\ A_j(z) &:= \min_{\substack{f \in \mathbb{R}^j \\ f_j = z}} |d_1 + f_1| + \sum_{i \in [j-1]} |f_i - f_{i+1} - d_{i+1}| + \sum_{i \in [j-1]} c_i |f_i| \text{ for all } 2 \leq j \leq n-2, \\ A_{n-1}(z) &:= \min_{\substack{f \in \mathbb{R}^j \\ f_j = z}} A(f). \end{aligned} \tag{9}$$

In other words, $A_j(z)$ asks to minimize the partial function in (8) over the first j flow variables $\{f_i\}_{i \in [j]}$, corresponding to all terms in which these flow variables participate, subject to fixing $f_j = z$. We make some preliminary observations about the partial functions $\{A_j\}_{j \in [n-1]}$.

Lemma 6. For all $j \in [n - 2]$, A_j is a convex, continuous, piecewise linear function with at most $j + 2$ pieces, and A_{n-1} is a convex, continuous, piecewise linear function with at most $n + 2$ pieces.

Proof. We first establish convexity by induction; the base case $j = 1$ is clear. We next observe that

$$\begin{aligned} A_j(z) &= c_j|z| + \min_{w \in \mathbb{R}} |w - z - d_j| + A_{j-1}(w) \text{ for all } 2 \leq j \leq n - 2, \\ A_{n-1}(z) &= |d_n - z| + c_{n-1}|z| + \min_{w \in \mathbb{R}} |w - z - d_{n-1}| + A_{n-2}(w). \end{aligned} \tag{10}$$

In other words, each partial function A_j can be recursively defined by first minimizing the first $j - 2$ flow variables for a fixed value $f_{j-1} = w$, and then taking the optimal choice of w . Moreover, supposing inductively A_{j-1} is convex, A_j is the sum of a convex function and a partial minimization over a jointly convex function of (w, z) , so it is also convex, completing the induction.

To see that A_j is continuous (assuming continuity of A_{j-1} inductively), it suffices to note A_j is the sum of a continuous function and a partial minimization over a continuous function in two variables.

We now prove the claims about piecewise linearity, and the number of pieces. Clearly, A_1 is continuous and piecewise linear with at most 3 pieces. Next, for some $2 \leq j \leq n - 2$, suppose A_{j-1} is piecewise linear with vertices $\{v_i\}_{i \in [j]}$ in nondecreasing order (possibly with repetition) and slopes $\{t_i\}_{i=0}^j$, so t_i is the slope of the segment of A_{j-1} between v_i and v_{i+1} , and t_0 and t_j are the leftmost and rightmost slopes. For convenience we define $v_0 := -\infty$ and $v_{j+1} := \infty$. Consider the function

$$\min_{w \in \mathbb{R}} |w - z - d_j| + A_{j-1}(w). \tag{11}$$

For all values z satisfying $v_i \leq z + d_j \leq v_{i+1}$ where $0 \leq i \leq j$, the function $|w - z - d_j| + A_{j-1}(w)$ is piecewise linear with vertices $v_1, \dots, v_i, z + d_j, v_{i+1}, \dots, v_j$, and correspondingly ordered slopes $t_0 - 1, t_1 - 1, \dots, t_i - 1, t_i + 1, \dots, t_j + 1$. The minimizing w in (11) corresponds to any $v \in \{v_i\}_{i \in [j]} \cup \{z + d_j\}$ where the slope switches from nonpositive to nonnegative, which is either a fixed vertex $v = v_k$ for the entire range $v_i \leq z + d_j \leq v_{i+1}$, or the new vertex $z + d_j$ for this entire range.

In the former case, we have

$$\min_{w \in \mathbb{R}} |w - z - d_j| + A_{j-1}(w) = |v_k - z - d_j| + A_{j-1}(v_k), \tag{12}$$

which up to a constant additive shift is $|z - (v_k - d_j)|$, a linear function in z in the range $v_i \leq z + d_j \leq v_{i+1}$ because the sign of $z - (v_k - d_j)$ does not change. In the latter case, we have

$$\min_{w \in \mathbb{R}} |w - z - d_j| + A_{j-1}(w) = A_{j-1}(z + d_j), \tag{13}$$

which again is a linear function in z in the range $v_i \leq z + d_j \leq v_{i+1}$ by induction. In conclusion, (11) is linear in each range $z \in [v_i - d_j, v_{i+1} - d_j]$, so it is piecewise linear with at most $j + 1$ pieces; adding $c_j|z|$, which introduces at most 1 more piece, completes the induction. An analogous argument holds for $j = n - 1$, but we potentially introduce two more pieces due to adding $|d_n - z| + c_{n-1}|z|$.

For convenience, we now describe how to update the slopes and vertices going from the piecewise linear function A_{j-1} to A_j . Also, as above suppose the vertices of A_{j-1} are $\{v_i\}_{i \in [j]}$ and the corresponding slopes are $\{t_i\}_{i=0}^j$. Then because we argued (11) is linear in each range $z \in [v_i - d_j, v_{i+1} - d_j]$, it has vertices $\{v_i - d_j\}_{i \in [j]}$. If $z \in [v_i - d_j, v_{i+1} - d_j]$ and $t_i \in [-1, 1]$, then we are in the case of (13) and the corresponding slope in this range is t_i . Otherwise, if $t_i \leq -1$ we are in the case of (12) with slope -1 , and if $t_i \geq 1$ the slope of the piece is similarly 1 . We then add $c_j|z|$ (and $|d_n - z|$ if $j = n - 1$). In summary, the new slopes and vertices are as follows.

- If $j \leq n - 2$, the new vertices are $\{v_i - d_j\}_{i \in [j]} \cup \{0\}$. If $v_k - d_j \leq 0 \leq v_{k+1} - d_j$ for some $0 \leq k \leq j$, using our convention $v_0 = -\infty$ and $v_{j+1} = \infty$, the new slopes are

$$\begin{aligned} &\{\text{clip}_{[-1,1]}(t_i) - c_j\}_{0 \leq i < k} \cup \{\text{clip}_{[-1,1]}(t_k) - c_j\} \\ &\cup \{\text{clip}_{[-1,1]}(t_k) + c_j\} \cup \{\text{clip}_{[-1,1]}(t_i) + c_j\}_{k < i \leq j}. \end{aligned} \tag{14}$$

- If $j = n - 1$, the new vertices are $\{v_i - d_j\}_{i \in [j]} \cup \{0\} \cup \{d_n\}$. If $v_k - d_j \leq 0 \leq v_{k+1} - d_j$ and $v_h - d_j \leq d_n \leq v_{h+1} - d_n$ for some $0 \leq h, k \leq j$, the new slopes are

$$\begin{aligned} & \left\{ \text{clip}_{[-1,1]}(t_i) - c_j \iota_{v_{i+1} \leq d_j} + c_j \iota_{v_i \geq d_j} \right\}_{\substack{0 \leq i \leq j \\ i \notin \{h,k\}}} \\ & \cup \left\{ \text{clip}_{[-1,1]}(t_k) - c_j \right\} \cup \left\{ \text{clip}_{[-1,1]}(t_k) + c_j \right\} \\ & \cup \left\{ \text{clip}_{[-1,1]}(t_h) - 1 \right\} \cup \left\{ \text{clip}_{[-1,1]}(t_h) + 1 \right\}, \end{aligned} \tag{15}$$

where we let $\iota_{\mathcal{E}}$ denote the 0-1 indicator variable of an event \mathcal{E} .

The ordering of these vertices and their slopes are uniquely determined, because they are sorted similarly due to convexity, which implies nondecreasing slopes as z increases. Finally, we note that assuming the invariant that at least one $\{t_i\}_{i=0}^j$ is nonnegative and at least one is nonpositive (which holds in the first iteration), in either of the cases (14) or (15) this invariant is preserved, since clipping preserves signs, the smallest slope decreases, and the largest slope increases. \square

We require one additional property of the slope updates (14).

Lemma 7. For $j \in [n - 1]$, let $\{t_i\}_{i=0}^j$ be the nondecreasing slopes of A_j . Then $t_0 \leq -1$ and $t_j \geq 1$.

Proof. By observation, the smallest and largest slopes of A_1 (9) are $-1 - c_1$ and $1 + c_1$. Hence, assuming inductively the lemma statement is true for A_{j-1} , the slope updates (14) result in smallest and largest slopes $-1 - c_j$ and $-1 + c_j$ in A_j , completing the induction. \square

We next observe that, by storing a constant amount of information in each iteration, we can work backwards from an optimal solution to A_{n-1} and recover all flow variables which realized this value.

Lemma 8. Let $\{v_i\}_{i \in [j]}$ and $\{t_i\}_{i=0}^j$ be the nondecreasing vertices and slopes of A_{j-1} for some $2 \leq j \leq n - 1$. Suppose we know v_ℓ and v_r for $\ell, r \in [j]$, defined such that $t_{\ell-1} \leq -1$ but $t_\ell \geq -1$, and similarly $t_{r-1} \leq 1$ but $t_r \geq 1$. Then given a value of z , we can compute in $O(1)$ time

$$\operatorname{argmin}_{w \in \mathbb{R}} |w - z - d_j| + A_{j-1}(w).$$

Proof. Note that existence of v_ℓ, v_r is guaranteed by Lemma 7. We consider three cases.

First, if $z + d_j \in [v_\ell, v_r]$, we claim $w = z + d_j$. To see this, recall that if $z + d_j \in [v_i, v_{i+1}]$, we proved in Lemma 6 that the slopes of $|w - z - d_j| + A_{j-1}(w)$ (in w) are $t_0 - 1, t_1 - 1, \dots, t_i - 1, t_i + 1, \dots, t_j + 1$. Hence, the assumptions imply $t_i \in [-1, 1]$, so a vertex of $|w - z - d_j| + A_{j-1}(w)$ where the slope changes from nonpositive to nonnegative (i.e. the minimizing argument w) is $w = z + d_j$, as claimed.

To handle the other two cases, if $z + d_j \leq v_\ell$, then the above calculation shows the new optimal vertex is $w = v_\ell$; similarly, if $z + d_j \geq v_r$ the new optimal vertex is $w = v_r$. \square

We now describe an interface for a data structure that we use to efficiently implement the updates (14) and (15), whose existence we prove in the following Appendix B.3.

Lemma 9. There is a data structure, Segment Tree, which initializes a vector $t \in \mathbb{R}^n$ to $t \leftarrow \mathbb{0}_n$, and supports the following operations each in $O(\log n)$ time.

- Query(i): Return t_i .
- Add(ℓ, r, c): Update $t_i \leftarrow t_i + c$ for all $\ell \leq i \leq r$.
- Set(ℓ, r, c): Update $t_i \leftarrow c$ for all $\ell \leq i \leq r$.

Given access to SegmentTree, we now describe how to solve (8) in nearly-linear time, proving Proposition 1.

Proof of Proposition 1. We first describe how to compute all of the vertices and slopes $\{v_i\}_{i \in [n-1]}$, $\{t_i\}_{i=0}^{n-1}$ of A_{n-2} in time $O(n \log^2(n))$. The proof of Lemma 6 shows that the vertices are

$$\left\{ - \sum_{k=j+1}^{n-2} d_k \right\}_{j=0}^{n-2}, \quad (16)$$

where we treat the empty sum as 0. Moreover, the vertex $-\sum_{k=j+1}^{n-2} d_k$ is the new vertex which was inserted (as 0) when computing A_j , and then advanced through the remaining iterations. We sort the vertices (16) into nondecreasing order, keeping track of the resulting permutation $\pi : [n-2] \cup \{0\} \rightarrow [n-1]$, i.e. if the vertex $-\sum_{k=j+1}^{n-2} d_k$ is the i^{th} smallest in (16), then $\pi(j) = i$. This step takes time $O(n \log n)$ and does not dominate.

We next initialize a SegmentTree (Lemma 9) with n vertices. We update it through the first $n-2$ recursive computations of slopes, via (14), keeping track of a time counter j , i.e. after we are done updating the slopes of A_j the time counter increments. The state of SegmentTree at the end of time j is as follows. For all $k \in [j]$, letting $\pi(h)$ be the next largest value after $\pi(k)$ amongst $\{\pi(k')\}_{k' \in [j]}$, we require that t_i equals the slope of the segment to the right of the vertex inserted at time k in A_j for all the coordinates $\pi(k) + 1 \leq i \leq \pi(h)$. In other words, SegmentTree stores all of the slopes of A_j in its coordinates (with redundancies due to vertices which will be inserted after time j), and π maps the times vertices are inserted to their coordinate values in SegmentTree.

We next show how to maintain this state in $O(\log^2(n))$ time per time increment. In iteration j , the update (14) requires us to clip all previous slopes to the range $[-1, 1]$, subtract c_j from all slopes in the range $[1, \pi(j)]$, and add c_j to all slopes in the range $[\pi(j) + 1, n]$. We first use Query to perform binary searches for the values ℓ, r as defined in Lemma 7. We then sequentially apply

$$\text{Set}(1, \ell, -1), \text{Set}(r + 1, n, 1), \text{Add}(1, \pi(j), -c_j), \text{Add}(\pi(j) + 1, n, c_j).$$

The dominant runtime term is the cost of $O(\log(n))$ calls to Query to perform the binary search, giving the claimed $O(\log^2(n))$ runtime per time increment. We can now call Query n times to compute all slopes and vertices of A_{n-2} . We will also store the values of v_ℓ and v_r at time j , which takes $O(1)$ time given ℓ, r, π , and partial sums which can be precomputed in time $O(n)$.

Given these slopes and vertices, it is straightforward to apply (15) to compute all slopes and vertices of A_{n-1} in time $O(n)$, at which point we can find $z := \operatorname{argmin}_{z \in \mathbb{R}} A_{n-1}(z)$. Next, by using our stored values of v_ℓ and v_r in every iteration, we can then use Lemma 8 to compute all the optimal flow values realizing $A_{n-1}(z)$. Finally, we can compute the optimal value (8) in time $O(n)$. \square

B.3 Implementation of SegmentTree

In this section, we develop a data structure known as a *segment tree* which plays a vital role in our main algorithm. In particular, it allows us to prove Lemma 9. While this data structure is well-known folklore in the competitive programming community (see e.g. an overview of this technique in [QM22]), we provide a full description and proof for completeness.

For integers ℓ and r satisfying $\ell \leq r$, we use $[\ell : r]$ to denote the set $\{\ell, \ell + 1, \dots, r\}$.

Lemma 10 (Segment tree). *Let G be a semigroup with an identity element e , where the semigroup product of $a, b \in G$ is denoted by $a \cdot b$ or ab and is not necessarily commutative. Let v be an array of length n , where each element of v is initialized to be the identity element e of G . There is a data structure \mathcal{D} , called a segment tree, that can perform each of the following operations in $O(\log n)$ time (assuming a semigroup product can be computed in constant time).*

1. *Access(i): given $i \in [1 : n]$, return the i^{th} element in v .*
2. *Apply(g, ℓ, r): given $\ell, r \in [1 : n]$ satisfying $\ell \leq r$, and given a semigroup element $g \in G$, for each index $i \in [\ell : r]$, replace $v[i]$ with $g \cdot v[i]$.*

Before proving Lemma 10, we first use it to prove Lemma 9.

Proof of Lemma 9. We apply the data structure in Lemma 10 to a specific semigroup G defined as follows. The elements of G are functions $\tau : \mathbb{R} \rightarrow \mathbb{R}$, where the identity element e is the identity

function $e(u) = u$ for every $u \in \mathbb{R}$, and the semigroup product is defined as function composition: $(a \cdot b)(u) = a(b(u))$ for every $a, b \in G$ and $u \in \mathbb{R}$. The semigroup G consists of the following functions: add_c and set_c for every $c \in \mathbb{R}$. These functions are defined as follows:

$$\text{add}_c(u) = u + c, \quad \text{set}_c(u) = c, \quad \text{for every } u \in \mathbb{R}.$$

It is easy to check that these functions are closed under composition:

$$\begin{aligned} \text{add}_c \cdot \text{add}_{c'} &= \text{add}_{c+c'}, \\ \text{set}_c \cdot \text{set}_{c'} &= \text{set}_c, \\ \text{add}_c \cdot \text{set}_{c'} &= \text{set}_{c+c'}, \\ \text{set}_c \cdot \text{add}_{c'} &= \text{set}_c. \end{aligned}$$

Therefore, G is a valid semigroup. We can now implement the operations Query, Add, Set in Lemma 9 using the operations Access and Apply in Lemma 10 as follows.

To implement $\text{Query}(i)$, we run $\text{Access}(i)$ to obtain its output $g = v[i] \in G$, and return $g(0)$.

To implement $\text{Add}(\ell, r, c)$, we run $\text{Apply}(\text{add}_c, \ell, r)$.

To implement $\text{Set}(\ell, r, c)$, we run $\text{Apply}(\text{set}_c, \ell, r)$.

The correctness of this implementation can be shown inductively. At initialization, $v[i] = e$, and thus $\text{Query}(i)$ returns $e(0) = 0$, which is the correct value of t_i at initialization. It remains to show inductively that after each Add and Set operation, the output of $\text{Query}(i)$, i.e., $v[i](0)$, is the intended value of t_i . Indeed, after an Add operation, for any $i \in [\ell : r]$, the element $v[i]$ is updated to $v[i]' := (\text{add}_c \cdot v[i])$, and thus

$$v[i]'(0) = (\text{add}_c \cdot v[i])(0) = \text{add}_c(v[i](0)) = v[i](0) + c = t_i + c,$$

which is the intended new value of t_i . For $i \notin [\ell : r]$, the element $v[i]$ remains unchanged, and thus $v[i](0)$ remains unchanged. This is as desired because the new value of t_i is intended to be the same as the old value. Combining these two cases, we have shown that the Add operation maintains that $v[i](0)$ is the intended value of t_i for every $i \in [1 : n]$. We can similarly show that the Set operation also has this property, and thus our implementation is correct. The running time guarantee of the implementation follows directly from the running time guarantee in Lemma 10. \square

We prove Lemma 10 by describing the construction of the segment tree data structure and analyzing its correctness (Lemma 11) and efficiency (Lemma 12). By appending to the array an appropriate number of auxiliary entries, we can assume without loss of generality that the array length n is a power of 2, i.e., $n = 2^r$ for a positive integer r . The data structure is implemented using a complete binary tree T with depth r , where the 2^r leaves correspond to the $n = 2^r$ entries in the array.

More specifically, we use $\text{seg}(\tau) \subseteq [1 : n]$ to denote the set of indices i that a node $\tau \in T$ is associated with. For the root τ_0 of the tree T , we set $\text{seg}(\tau_0) = [1 : n] = [1 : 2^r]$, and for its two children τ_1, τ_2 we set $\text{seg}(\tau_1) = [1 : n/2] = [1 : 2^{r-1}]$ and $\text{seg}(\tau_2) = [n/2 + 1 : n] = [2^{r-1} + 1 : 2^r]$. In general, for any non-leaf node τ and its two children τ_1, τ_2 , we set $\text{seg}(\tau_1)$ as the first half of $\text{seg}(\tau)$, and set $\text{seg}(\tau_2)$ as the second half. In particular, for every leaf τ , $\text{seg}(\tau)$ is a singleton set consisting of a unique index $i \in [1 : n]$, and we say τ is the (unique) leaf corresponding to index i . See Figure 3 for an example with depth $r = 3$.

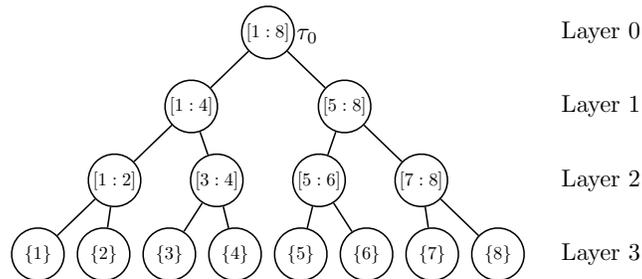


Figure 3: Example of segment tree with depth $r = 3$.

At every node τ of the tree, we maintain a semigroup element $g_\tau \in G$ initialized to be the identity element e .

Access. We implement $\text{Access}(i)$ as follows. Let $\tau_0 \rightarrow \dots \rightarrow \tau_r$ be the directed path from the root τ_0 to the leaf τ_r corresponding to index i . Return the semigroup product $g_{\tau_0} g_{\tau_1} \dots g_{\tau_r}$.

Apply. We implement $\text{Apply}(g, \ell, r)$ recursively. That is, we implement $\text{Apply}(g, \ell, r, \tau)$ in Algorithm 1 which takes a node $\tau \in T$ as an additional input. We then define $\text{Apply}(g, \ell, r)$ to be $\text{Apply}(g, \ell, r, \tau_0)$, where the additional input is set as the root τ_0 of the tree T .

Algorithm 1 $\text{Apply}(g, \ell, r, \tau)$

```

1: if  $\text{seg}(\tau) \cap [\ell : r] = \emptyset$  then
2:   return
3: end if
4: if  $\text{seg}(\tau) \subseteq [\ell : r]$  then
5:    $g_\tau \leftarrow g \cdot g_\tau$ 
6:   return
7: end if
8: Let  $\tau_1, \tau_2$  be the two children of  $\tau$ 
9:  $g_{\tau_1} \leftarrow g_\tau \cdot g_{\tau_1}$ 
10:  $g_{\tau_2} \leftarrow g_\tau \cdot g_{\tau_2}$ 
11:  $g_\tau \leftarrow e$ 
12:  $\text{Apply}(g, \ell, r, \tau_1)$ 
13:  $\text{Apply}(g, \ell, r, \tau_2)$ 

```

Correctness. At initialization, each array element $v[i]$ is initialized to be the identity element e . The semigroup element g_τ stored at each tree node τ is also initialized to be e , so $\text{Access}(i)$ returns the correct value e . It remains to show that $\text{Access}(i)$ still returns the correct value of $v[i]$ after each Apply operation. This is established in the following lemma:

Lemma 11. For $i \in \{1, \dots, n\}$, let $\tau_0 \rightarrow \dots \rightarrow \tau_r$ be the directed path from the root τ_0 to the leaf τ_r corresponding to index i . Let $\hat{g}_{\tau_0}, \dots, \hat{g}_{\tau_r} \in G$ denote the current states of the semigroup elements $g_{\tau_0}, \dots, g_{\tau_r}$ stored in the data structure. Then after we call $\text{Apply}(g, \ell, r)$, we have

$$g_{\tau_0} \dots g_{\tau_r} = \begin{cases} g \cdot \hat{g}_{\tau_0} \dots \hat{g}_{\tau_r}, & \text{if } i \in [\ell : r]; \\ \hat{g}_{\tau_0} \dots \hat{g}_{\tau_r}, & \text{if } i \notin [\ell : r]. \end{cases}$$

Proof. It is clear that $[1 : n] = \text{seg}(\tau_0) \supseteq \dots \supseteq \text{seg}(\tau_r) = \{i\}$. If $i \in [\ell : r]$, let τ_j be the first node among τ_0, \dots, τ_r such that $\text{seg}(\tau_j) \subseteq [\ell, r]$. We can inductively show that for every $j' = 1, \dots, j$, right before we make the recursive call to $\text{Apply}(g, \ell, r, \tau_{j'})$, we have

$$g_{\tau_i} = \begin{cases} e, & \text{if } i < j', \\ \hat{g}_{\tau_0} \dots \hat{g}_{\tau_{j'}}, & \text{if } i = j', \\ \hat{g}_{\tau_i}, & \text{if } i > j'. \end{cases}$$

When we call $\text{Apply}(g, \ell, r, \tau_j)$, since $\text{seg}(\tau_j) \subseteq [\ell : r]$, Line 5 is executed and the function returns after that. Now we have

$$g_{\tau_i} = \begin{cases} e, & \text{if } i < j, \\ g \cdot \hat{g}_{\tau_0} \dots \hat{g}_{\tau_j}, & \text{if } i = j, \\ \hat{g}_{\tau_i}, & \text{if } i > j. \end{cases}$$

This implies $g_{\tau_0} \dots g_{\tau_r} = g \cdot \hat{g}_{\tau_0} \dots \hat{g}_{\tau_r}$, as desired.

Similarly, if $i \notin [\ell, r]$, let τ_j be the first node among τ_0, \dots, τ_r such that $\text{seg}(\tau_j) \cap [\ell : r] = \emptyset$. We can show that

$$g_{\tau_i} = \begin{cases} e, & \text{if } i < j; \\ \hat{g}_{\tau_0} \dots \hat{g}_{\tau_j}, & \text{if } i = j; \\ \hat{g}_{\tau_i}, & \text{if } i > j. \end{cases}$$

This implies $g_{\tau_0} \dots g_{\tau_r} = \hat{g}_{\tau_0} \dots \hat{g}_{\tau_r}$, as desired. □

Efficiency. The following result establishes the running time guarantee in Lemma 10.

Lemma 12. *Both Access and Apply run in $O(\log n)$ time.*

Proof. It is clear that Access runs in time $O(r) = O(\log n)$. When we run $\text{Apply}(g, \ell, r, \tau)$, the recursive calls at Lines 12-13 are made only when $[\ell : r]$ intersects but does not contain $\text{seg}(\tau)$, i.e., $\emptyset \subsetneq \text{seg}(\tau) \cap [\ell : r] \subsetneq \text{seg}(\tau)$. There are at most 2 such nodes τ at each level of the binary tree, so the total number of such nodes τ is $O(\log n)$. This implies that Apply runs in time $O(\log n)$. \square

B.4 Tolerant testing via smooth calibration

For completeness, we first make the simple (but to our knowledge, new) observation that, while the constants in Lemma 4 are not necessarily tight, there is a constant gap between dCE and smCE .

Lemma 13. *Suppose for constants $B \geq A > 0$, it is the case that $A \cdot \text{dCE}(\mathcal{D}) \leq \text{smCE}(\mathcal{D}) \leq B \cdot \text{dCE}(\mathcal{D})$ for all distributions \mathcal{D} over $[0, 1] \times \{0, 1\}$. Then, $\frac{B}{A} \geq \frac{3}{2}$.*

Proof. First, we claim that $A \leq 1$. To see this, let $(v, y) \sim \mathcal{D}$ be distributed where $v = \frac{1}{2}$ with probability 1, and $y \sim \text{Bern}(\frac{1}{2} + \varepsilon)$ for some $\varepsilon \in [0, \frac{1}{2}]$. Clearly, $\text{smCE}(\mathcal{D}) = |\frac{1}{2} - (\frac{1}{2} + \varepsilon)| = \varepsilon$. Moreover, $\text{dCE}(\mathcal{D}) = \varepsilon$, which follows from the same Jensen's inequality argument as in Lemma 5, so this shows that $A \leq 1$. Next, we claim that $B \geq \frac{3}{2}$, concluding the proof. Consider the joint distribution over (u, v, y) in Table B.4, and let \mathcal{D} be the marginal of (v, y) . It is straightforward

Probability mass	u	v	y	$w(v)$
$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2} - \varepsilon$	1	1
$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	0	$1 - \varepsilon$

to check (u, y) is calibrated and $\mathbb{E}|u - v| = \frac{\varepsilon}{2}$, so $\text{dCE}(\mathcal{D}) \leq \frac{\varepsilon}{2}$. Moreover, $\text{smCE}(v, y) \geq \frac{3\varepsilon}{4}$, as witnessed by the Lipschitz weight function $w(v)$ in Table B.4, finishing our proof that $B \geq \frac{3}{2}$:

$$\text{smCE}(v, y) \geq \mathbb{E}[(y - v)w(v)] = \frac{1}{2} \left(\left(\frac{1}{2} + \varepsilon \right) \cdot 1 \right) + \frac{1}{2} \left(\left(-\frac{1}{2} \right) \cdot (1 - \varepsilon) \right) = \frac{3\varepsilon}{4}.$$

\square

Using these claims, we now give our tolerant calibration tester in the regime $\varepsilon_1 > 4\varepsilon_2$.

Theorem 3. *Let $0 \leq \varepsilon_2 \leq \varepsilon_1 \leq 1$ satisfy $\varepsilon_1 > 4\varepsilon_2$, and let $n \geq C_{\text{tct}} \cdot \frac{1}{(\varepsilon_1 - 4\varepsilon_2)^2}$ for a universal constant C_{tct} . There is an algorithm \mathcal{A} which solves the $(\varepsilon_1, \varepsilon_2)$ -tolerant calibration testing problem with n samples, which runs in time*

$$O(n \log^2(n)).$$

Proof. Throughout the proof, let $\alpha := \frac{\varepsilon_1}{2} - 2\varepsilon_2 > 0$. Consider the following algorithm.

1. Sample $n \geq \frac{C_{\text{tct}}}{\alpha^2}$ samples to form an empirical distribution $\widehat{\mathcal{D}}_n$, where C_{tct} is chosen large enough so that Lemma 3 guarantees $|\text{smCE}(\mathcal{D}) - \text{smCE}(\widehat{\mathcal{D}}_n)| \leq \frac{\alpha}{2}$ with probability $\geq \frac{2}{3}$.
2. Call Corollary 1 to obtain β , the value of $\text{smCE}(\widehat{\mathcal{D}}_n)$.
3. Return “yes” if $\beta \leq 2\varepsilon_2 + \frac{\alpha}{2}$, and return “no” otherwise.

Conditioned on the event $|\text{smCE}(\mathcal{D}) - \text{smCE}(\widehat{\mathcal{D}}_n)| \leq \frac{\alpha}{2}$, we show that the algorithm succeeds in tolerant calibration testing. First, if $\text{dCE}(\mathcal{D}) \leq \varepsilon_2$, then $\text{smCE}(\mathcal{D}) \leq 2\varepsilon_2$ by Lemma 4, and therefore by the assumed success of Lemma 3, the algorithm will return “yes.” Second, if $\text{dCE}(\mathcal{D}) \geq \varepsilon_1$, then $\text{smCE}(\mathcal{D}) \geq \frac{\varepsilon_1}{2}$ by Lemma 4, and similarly the algorithm returns “no” in this case. Finally, the runtime is immediate from Corollary 1; all other steps take $O(1)$ time. \square

C Lower distance to calibration

In this section, we provide our main result on approximating the lower distance to calibration of a distribution on $[0, 1] \times \{0, 1\}$. We provide details on a framework for lifting constrained linear programs to equivalent unconstrained counterparts in Appendix C.1. In Appendix C.2, we next state preliminary definitions and results from [BGHN23a] used in our algorithm. In Appendix C.3, we then develop a rounding procedure compatible with a linear program which closely approximates the empirical lower distance to calibration. Finally, in Appendix C.4, we use our rounding procedure to design an algorithm for calibration testing, which solves the problem for a larger range of parameters than Theorem 4 (i.e. the entire relevant parameter range), at a quadratic runtime overhead.

C.1 Rounding linear programs

In this section, we give a general framework for approximately solving linear programs, following similar developments in the recent combinatorial optimization literature [She13, JST19]. Roughly speaking, this framework is a technique for losslessly converting a constrained convex program to an unconstrained one, provided we can show existence of a rounding procedure compatible with the constrained program in an appropriate sense. We begin with our definition of a rounding procedure.

Definition 6 (Rounding procedure). *Consider a convex program defined on the intersection of convex set \mathcal{X} with linear equality constraints $\mathbf{A}x = b$:*

$$\min_{\substack{x \in \mathcal{X} \\ \mathbf{A}x = b}} c^\top x. \quad (17)$$

We say Round is a $(\tilde{\mathbf{A}}, \tilde{b}, p)$ -equality rounding procedure for $(\mathbf{A}, b, c, \mathcal{X})$ if $p \geq 1$, and for any $x \in \mathcal{X}$, there exists $x' := \text{Round}(x) \in \mathcal{X}$ such that $\mathbf{A}x' = b$, $\tilde{\mathbf{A}}x' = \tilde{b}$, and

$$c^\top x' \leq c^\top x + \left\| \tilde{\mathbf{A}}x - \tilde{b} \right\|_p. \quad (18)$$

Intuitively, rounding procedures replace the hard-constrained problem (17) with its soft-constrained variants, i.e. the soft equality-constrained

$$\min_{x \in \mathcal{X}} c^\top x + \left\| \tilde{\mathbf{A}}x - \tilde{b} \right\|_p, \quad (19)$$

for some $(\tilde{\mathbf{A}}, \tilde{b}, p)$ constructed from the corresponding hard-constrained problem instance (parameterized by $\mathbf{A}, b, c, \mathcal{X}$). Leveraging the assumptions on our rounding procedure, we now show how to relate approximate solutions to these problems, generalizing Lemma 1 of [JST19].

Lemma 14. *Let x be an ε -approximate minimizer to (19), and let Round be a $(\tilde{\mathbf{A}}, \tilde{b}, p)$ -equality rounding procedure for $(\mathbf{A}, b, c, \mathcal{X})$. Then $x' := \text{Round}(x)$ is an ε -approximate minimizer to (17).*

Proof. We first claim that a minimizing solution to (19) satisfies the constraints $\mathbf{A}x = b$. To see this, given any $x \in \mathcal{X}$, we can produce $x' \in \mathcal{X}$ with $\mathbf{A}x' = b$ and such that x' has smaller objective value in (19). Indeed, letting $x' := \text{Round}(x)$, (18) guarantees

$$c^\top x' = c^\top x + \left\| \tilde{\mathbf{A}}x' - \tilde{b} \right\|_p \leq c^\top x + \left\| \tilde{\mathbf{A}}x - \tilde{b} \right\|_p,$$

as claimed. Now let $x^* \in \mathcal{X}$ satisfying $\mathbf{A}x^* = b$ minimize (19). Then, if x is an ε -approximate minimizer to (19) and $x' = \text{Round}(x)$, we have the desired claim from $\mathbf{A}x' = b$, and

$$c^\top x' = c^\top x + \left\| \tilde{\mathbf{A}}x' - \tilde{b} \right\|_p \leq c^\top x + \left\| \tilde{\mathbf{A}}x - \tilde{b} \right\|_p \leq c^\top x^* + \left\| \tilde{\mathbf{A}}x^* - \tilde{b} \right\|_p = c^\top x^* + \varepsilon.$$

□

In the remainder of the section, we apply our rounding framework to a hard-constrained linear program, in the $p = 1$ geometry. To aid in approximately solving the soft-constrained linear programs arising from our framework, we use the following procedure from [JT23], building upon the recent

literature for solving box-simplex games at accelerated rates [She17, JST19, CST21]. In the statement of Proposition 2, we use the notation

$$\|\mathbf{A}\|_{p \rightarrow q} := \max_{x \in \mathbb{R}^n, \|x\|_p \leq 1} \|\mathbf{A}x\|_q.$$

Notice that in particular, $\|\mathbf{A}\|_{1 \rightarrow 1}$ is the largest ℓ_1 norm of any column of \mathbf{A} .

Proposition 2 (Theorem 1, [JT23]). *Let $\mathbf{A} \in \mathbb{R}^{n \times d}$, $b \in \mathbb{R}^d$, $c \in \mathbb{R}^n$, and $\varepsilon > 0$. There is an algorithm which computes an ε -approximate saddle point to the box-simplex game*

$$\min_{x \in [-1, 1]^n} \max_{y \in \Delta^d} x^\top \mathbf{A}y - b^\top y + c^\top x, \quad (20)$$

in time

$$O\left(\text{nnz}(\mathbf{A}) \cdot \frac{\|\mathbf{A}\|_{1 \rightarrow 1} \log d}{\varepsilon}\right).$$

We also require a standard claim on converting minimax optimization error to error on an induced minimization objective. To introduce our notation, we say that $x \in \mathcal{X}$ is an ε -approximate minimizer of $f : \mathcal{X} \rightarrow \mathbb{R}$ if $f(x) - \min_{x' \in \mathcal{X}} f(x') \leq \varepsilon$. We call $(x, y) \in \mathcal{X} \times \mathcal{Y}$ an ε -approximate saddle point to a convex-concave function $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ if its duality gap is at most ε , i.e.

$$\max_{y' \in \mathcal{Y}} f(x, y') - \min_{x' \in \mathcal{X}} f(x', y) \leq \varepsilon.$$

Lemma 15. *Let $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ be convex-concave for compact \mathcal{X}, \mathcal{Y} , and let $g(x) := \max_{y \in \mathcal{Y}} f(x, y)$ for $x \in \mathcal{X}$. If (x, y) is an ε -approximate saddle point to f , x is an ε -approximate minimizer to g .*

Proof. Let $y' := \operatorname{argmax}_{y \in \mathcal{Y}} f(x, y)$ and $x' := \operatorname{argmin}_{x' \in \mathcal{X}} f(x', y)$. The conclusion follows from

$$\min_{x^* \in \mathcal{X}} g(x^*) = \min_{x^* \in \mathcal{X}} \max_{y^* \in \mathcal{Y}} f(x^*, y^*) = \max_{y^* \in \mathcal{Y}} \min_{x^* \in \mathcal{X}} f(x^*, y^*) \geq \min_{x' \in \mathcal{X}} f(x', y),$$

where we used strong duality (via Sion's minimax theorem), so that

$$g(x) - \min_{x^* \in \mathcal{X}} g(x^*) \leq g(x) - f(x', y) = f(x, y') - f(x', y) \leq \varepsilon.$$

□

The following corollary of Proposition 2 will be particularly useful in our development, which is immediate using Lemma 15, upon negating the box-simplex game (20), exchanging the names of the variables (x, y) , (b, c) , and explicitly maximizing over $y \in [-1, 1]^n$, i.e.

$$\min_{x \in \Delta^d} \langle c, x \rangle + \|\mathbf{A}x - b\|_1 = \min_{x \in \Delta^d} \max_{y \in [-1, 1]^n} \langle c, x \rangle + y^\top (\mathbf{A}x - b).$$

Corollary 2. *Let $\mathbf{A} \in \mathbb{R}^{n \times d}$, $b \in \mathbb{R}^n$, $c \in \mathbb{R}^d$, and $\varepsilon > 0$. There is an algorithm which computes an ε -approximate minimizer to $\min_{x \in \Delta^d} c^\top x + \|\mathbf{A}x - b\|_1$, in time*

$$O\left(\text{nnz}(\mathbf{A}) \cdot \frac{\|\mathbf{A}\|_{1 \rightarrow 1} \log d}{\varepsilon}\right).$$

C.2 LDTC preliminaries

In this section, we collect preliminaries for our testing algorithm based on estimating the LDTC. First, analogously to Lemma 3, we recall a bound from [BGHN23a] on the deviation of the empirical estimate of $\underline{\text{dCE}}(\mathcal{D})$ from the population truth which holds with constant probability.

Lemma 16 (Theorem 9.10, [BGHN23a]). *For any $\varepsilon \in (0, 1)$, there is an $n = O(\frac{1}{\varepsilon^2})$ such that, if $\widehat{\mathcal{D}}_n$ is the empirical distribution over n i.i.d. draws from \mathcal{D} , with probability $\geq \frac{2}{3}$,*

$$\left| \underline{\text{dCE}}(\mathcal{D}) - \underline{\text{dCE}}(\widehat{\mathcal{D}}_n) \right| \leq \varepsilon.$$

Next, given a set $U \subset [0, 1]$, we provide an analog of Definition 1 which is restricted to U .

Definition 7 (U-LDTC). Let $U \subset [0, 1]$, and let \mathcal{D} be a distribution over $[0, 1] \times \{0, 1\}$. Define $\text{ext}^U(\mathcal{D})$ to be all joint distributions Π over $(u, v, y) \in U \times [0, 1] \times \{0, 1\}$, with the following properties.

- The marginal distribution of (v, y) is \mathcal{D} .
- The marginal distribution (u, y) is perfectly calibrated, i.e. $\mathbb{E}_\Pi[y|u] = u$.

The U-lower distance to calibration (U-LDTC) of \mathcal{D} , denoted $\underline{\text{dCE}}^U(\mathcal{D})$, is defined by

$$\underline{\text{dCE}}^U(\mathcal{D}) := \inf_{\Pi \in \text{ext}^U(\mathcal{D})} \mathbb{E}_{(u,v,y) \sim \Pi} |u - v|.$$

Note that if we require $\{0, 1\} \subset U$, then $\text{ext}^U(\mathcal{D})$ is always nonempty, because we can let $u = y$ with probability 1. We also state a helper claim from [BGHN23a], which relates $\underline{\text{dCE}}^U$ to $\underline{\text{dCE}}$.

Lemma 17 (Lemma 7.11, [BGHN23a]). Let \mathcal{D} be a distribution over $[0, 1] \times \{0, 1\}$, and let U be a finite $\frac{\varepsilon}{2}$ -covering of $[0, 1]$ satisfying $\{0, 1\} \subseteq U$. Then, $\underline{\text{dCE}}(\mathcal{D}) \leq \underline{\text{dCE}}^U(\mathcal{D}) \leq \underline{\text{dCE}}(\mathcal{D}) + \varepsilon$.

To this end, in the rest of the section we define, for any $\varepsilon \in (0, 1)$,

$$U_\varepsilon := \{0, 1\} \cup \left\{ \frac{i\varepsilon}{2} \mid i \in \left[\left\lceil \frac{2}{\varepsilon} \right\rceil \right] \right\}, \quad (21)$$

which is an $\frac{\varepsilon}{2}$ -cover of $[0, 1]$ satisfying $|U_\varepsilon| = O(\frac{1}{\varepsilon})$. Finally, we state a linear program, derived in [BGHN23a], whose value equals $\underline{\text{dCE}}^U(\mathcal{D})$, when the first marginal of \mathcal{D} is discretely supported.

Lemma 18 (Lemma 7.6, [BGHN23a]). Let $U, V \subset [0, 1]$ be discrete sets, where $\{0, 1\} \subset U$, and let \mathcal{D} be a distribution over $V \times \{0, 1\}$, where for $(v, y) \in V \times \{0, 1\}$ we denote the probability of $(v, y) \sim \mathcal{D}$ by $\mathcal{D}(v, y)$. The following linear program with $2|U||V|$ variables $\Pi(u, v, y)$ for all $(u, v, y) \in U \times V \times \{0, 1\}$, is feasible, and its optimal value equals $\underline{\text{dCE}}^U(\mathcal{D})$:

$$\begin{aligned} & \min_{\Pi \in \mathbb{R}_{\geq 0}^{2|U||V|}} \sum_{(u,v,y) \in U \times V \times \{0,1\}} |u - v| \Pi(u, v, y) \\ & \text{such that } \sum_{u \in U} \Pi(u, v, y) = \mathcal{D}(v, y), \text{ for all } (v, y) \in V \times \{0, 1\}, \\ & \text{and } (1 - u) \sum_{v \in V} \Pi(u, v, 1) = u \sum_{v \in V} \Pi(u, v, 0), \text{ for all } u \in U. \end{aligned}$$

C.3 Rounding for empirical U-LDTC

In this section we fix a dataset under consideration,

$$\widehat{\mathcal{D}}_n := \{(v_i, y_i)\}_{i \in [n]} \subset [0, 1] \times \{0, 1\},$$

and the corresponding empirical distribution, also denoted $\widehat{\mathcal{D}}_n$, where $(v, y) \sim \widehat{\mathcal{D}}_n$ means $(v, y) = (v_i, y_i)$ with probability $\frac{1}{n}$ for each $i \in [n]$. We let $V := \{v_i\}_{i \in [n]}$ be identified with $[n]$ in the natural way. Moreover, for a fixed parameter $\varepsilon \in (0, 1)$ throughout, we let $U := U_\varepsilon$ defined in (21). Finally, we denote $m := |U| = O(\frac{1}{\varepsilon})$, and let $\mathbf{U} \in [0, 1]^{m \times m}$ be the diagonal matrix whose diagonal entries correspond to U . We also identify elements of U with $j \in [m]$ in an arbitrary but consistent way, writing $u_j \in [0, 1]$ to mean the j^{th} element of U according to this identification.

We next rewrite the linear program in Lemma 18 into a more convenient reformulation.

Lemma 19. The linear program in Lemma 18 can equivalently be written as:

$$\underline{\text{dCE}}^U(\widehat{\mathcal{D}}_n) := \min_{\substack{x \in \mathcal{X} \\ \mathbf{M}x = \frac{1}{n} \mathbf{1}_n \\ \mathbf{U} \mathbf{B}_0 x_0 = (\mathbf{I}_m - \mathbf{U}) \mathbf{B}_1 x_1}} c^\top x, \text{ where } \mathcal{X} := \Delta^{2mn} \text{ and we denote } x = \begin{pmatrix} x_0 \in \mathbb{R}^{mn} \\ x_1 \in \mathbb{R}^{mn} \end{pmatrix}, \quad (22)$$

where we define $c \in \mathbb{R}^{2mn}$, $\mathbf{M} \in \mathbb{R}^{n \times 2mn}$, and $\mathbf{B}_0, \mathbf{B}_1 \in \mathbb{R}^{m \times mn}$ by

$$c_{(i,j,k)} := |u_j - v_i| \text{ for all } (i, j, k) \in [n] \times [m] \times \{0, 1\},$$

$$\mathbf{M}_{i',(i,j,k)} := \begin{cases} 1 & y_i = k, i' = i \\ 0 & \text{else} \end{cases} \text{ for all } i' \in [n], (i, j, k) \in [n] \times [m] \times \{0, 1\},$$

and $[\mathbf{B}_0]_{j',(i,j,k)} := \begin{cases} 1 & j = j', k = 0 \\ 0 & \text{else} \end{cases} \text{ for all } j' \in [m], (i, j, k) \in [n] \times [m] \times \{0, 1\},$

$$[\mathbf{B}_1]_{j',(i,j,k)} := \begin{cases} 1 & j = j', k = 1 \\ 0 & \text{else} \end{cases} \text{ for all } j' \in [m], (i, j, k) \in [n] \times [m] \times \{0, 1\}.$$

Proof. This is clear from observation, but we give a brief explanation of the notation. First, $x \in \mathcal{X}$ represents the density function of our joint distribution Π over $U \times V \times \{0, 1\}$, and has $2mn$ coordinates identified with elements $(i, j, k) \in V \times U \times \{0, 1\} \equiv [n] \times [m] \times \{0, 1\}$. We let the subset of coordinates with $k = 0$ be denoted $x_0 \in \mathbb{R}^{mn}$, defining x_1 similarly. Recalling the definition of the linear program in Lemma 18, $x_{(i,j,k)}$ is indeed reweighted by $c_{(i,j,k)} = |u_j - v_i|$.

Next, \mathbf{M} represents the marginal constraints in Lemma 18, and enforcing $\mathbf{M}x = \frac{1}{n} \mathbb{1}_n$ is equivalent to the statement that, for each $i' \in [n]$, the sum of all entries (i, j, k) of x with $i = i'$ and $k = y_i$ is $\frac{1}{n}$, since that is the probability density assigned to $(v_{i'}, y_{i'})$ by the distribution $\widehat{\mathcal{D}}_n$.

Lastly, the j^{th} calibration constraint in Lemma 18 is enforced by the j^{th} row of the equation $\mathbf{U}\mathbf{B}_0x_0 = (\mathbf{I}_m - \mathbf{U})\mathbf{B}_1x_1$, which reads $u_j \langle [\mathbf{B}_0]_{j,:}, x_0 \rangle = (1 - u_j) \langle [\mathbf{B}_1]_{j,:}, x_1 \rangle$. We can check by the definitions of $[\mathbf{B}_0]_{j,:}$, $[\mathbf{B}_1]_{j,:}$ that this is consistent with our earlier calibration constraints. □

We give a convenient way of visualizing the marginal and calibration constraints described in Lemma 19. For convenience, we identify each $x \in \Delta^{2mn}$ with an $m \times 2n$ matrix

$$\text{mat}(x) = \mathbf{X} = (\mathbf{X}_0 \in \mathbb{R}^{m \times n} \quad \mathbf{X}_1 \in \mathbb{R}^{m \times n}), \quad (23)$$

where \mathbf{X}_0 consists of entries of x_0 arranged in a matrix fashion (with rows corresponding to $[m] \equiv U$ and columns corresponding to $[n] \equiv V$), and similarly \mathbf{X}_1 is a rearrangement of x_1 , recalling (22). When explaining how we design our rounding procedures to modify \mathbf{X} to satisfy constraints, it will be helpful to view entries of \mathbf{X} as denoting an amount of physical mass which we can move around.

There are $2n$ columns in \mathbf{X} , corresponding to pairs $(i, k) \in V \times \{0, 1\}$; among these, we say n columns are “active,” where column (i, k) is active iff $y_i = k$, and we say the other n columns are “inactive.” Following notation (23), the marginal constraints $\mathbf{X} = \frac{1}{n} \mathbb{1}_n$ simply ask that the total amount of mass in each active column is $\frac{1}{n}$, so there is no mass in any inactive column since $x \in \Delta^{2mn}$.

Moreover, there are m rows in \mathbf{X} , each corresponding to some $j \in U$. If we let ℓ_j denote the amount of mass on $[\mathbf{X}_0]_{j,:}$ and r_j the amount of mass on $[\mathbf{X}_1]_{j,:}$, the j^{th} calibration constraint simply asks that $u_j \ell_j = (1 - u_j) r_j$, i.e. it enforces balance on the amount of mass in each row’s two halves.

Finally, for consistency with Definition 6, the linear program in (22) can be concisely written as

$$\min_{\substack{x \in \mathcal{X} \\ \mathbf{A}x=b}} c^\top x, \text{ where } \mathbf{A} := \begin{pmatrix} \mathbf{M} \\ \mathbf{B} \end{pmatrix}, \mathbf{B} := (\mathbf{U}\mathbf{B}_0 \quad -(\mathbf{I}_m - \mathbf{U})\mathbf{B}_1), b := \begin{pmatrix} \frac{1}{n} \mathbb{1}_n \\ 0_m \end{pmatrix} \quad (24)$$

and c, \mathcal{X} are as defined in (22). In the rest of the section, following Definition 6, we develop an equality rounding procedure for the equality-constrained linear program in (24) in two steps.

1. In Lemma 20, we first show how to take $x \in \mathcal{X}$ with $\|\mathbf{M}x - \frac{1}{n} \mathbb{1}_n\|_1 = \Delta$, and produce $x' \in \mathcal{X}$ such that $\mathbf{M}x' = \frac{1}{n} \mathbb{1}_n$ (i.e. x' now satisfies the marginal constraints) and $\|x - x'\|_1 = O(\Delta)$.
2. In Lemma 22, we then consider $x \in \mathcal{X}$ such that, following the notation (22), $\|\mathbf{U}\mathbf{B}_0x_0 - (\mathbf{I}_m - \mathbf{U})\mathbf{B}_1x_1\|_1 = \Delta$. We show how to produce $x' \in \mathcal{X}$ such that $\mathbf{M}x = \mathbf{M}x'$ (i.e. the marginals of x' are unchanged), $\mathbf{U}\mathbf{B}_0x'_0 = (\mathbf{I}_m - \mathbf{U})\mathbf{B}_1x'_1$ (i.e. x' is calibrated), and $\langle c, x' - x \rangle = O(\Delta)$.

Our rounding procedure uses Lemma 20 to satisfy the marginal constraints in (22), and then applies Lemma 22 to the result to satisfy the calibration constraints in (22) without affecting the marginal constraints. By leveraging the stability guarantees on these steps, we can show this is indeed a valid rounding procedure in the sense of (18). We now give our first step for marginal satisfaction.

Lemma 20 (Marginal satisfaction). *Following notation in (22), let $x \in \mathcal{X}$ satisfy $\|\mathbf{M}x - \frac{1}{n} \mathbb{1}_n\|_1 = \Delta$. There is an algorithm which runs in time $O(mn)$, and returns x' with*

$$\mathbf{M}x' = \frac{1}{n} \mathbb{1}_n, \|x - x'\|_1 \leq 2\Delta.$$

Proof. Recall for $i \in [n]$, we say column i of \mathbf{X}_0 is active if $y_i = 0$, and similarly column i of \mathbf{X}_1 is active if $y_i = 1$. We call I the set of n inactive columns, and partition A , which we call the set of n active columns, into three sets $A^>$, $A^=$, and $A^<$, where $A^>$ are the columns whose sums are $> \frac{1}{n}$, $A^<$ are the columns whose sums are $< \frac{1}{n}$, and $A^=$ are the remaining columns. Hence, every column of \mathbf{X} belongs to I , $A^>$, $A^=$, or $A^<$. Note that until $|A^|=n$, we can never have $A^< = \emptyset$, since this means all column sums in A are $\geq \frac{1}{n}$ (with at least 1 strict inequality), contradicting $x \in \mathcal{X}$.

We first take columns $i \in A^>$ one at a time, and pair them with an arbitrary column in $i' \in A^<$, moving mass from column i arbitrarily to column i' until either column i or column i' enters $A^=$. We charge this movement to the marginal constraints corresponding to i and i' , since the constraints were violated by the same amount as the mass being moved. After this process is complete, $A^>$ is empty, and we only moved mass from columns originally in $A^>$ to columns originally in $A^<$.

Next, we take columns $i \in I$ one at a time, and pair them with an arbitrary column $i' \in A^<$, moving mass until either column i is 0_m or column i' enters $A^=$. We can charge half this movement to the marginal constraint corresponding to i' , since the sign of the marginal violation stays the same throughout. Hence, the overall movement is $\leq 2\Delta$. After this is complete, all columns in I are 0_m and all columns in A are in $A^=$, so we can return $x' \in \Delta^{2mn}$ corresponding to the new matrix.

It is clear both steps of this marginal satisfaction procedure take $O(mn)$ time, since we can sequentially process columns in A^- until they enter $A^=$, and will never be considered again. \square

We next describe a procedure which takes $x \in \Delta^{2mn}$, and modifies it to satisfy the calibration constraints $\mathbf{U}\mathbf{B}_0x = (\mathbf{I}_m - \mathbf{U})\mathbf{B}_1x$ without changing the marginals $\mathbf{M}x$. We first provide a helper lemma used in our rounding procedure, which describes how to fix the j^{th} marginal constraint.

Lemma 21. *Let $x \in \Delta^{2mn}$ and $\mathbf{X} := \text{mat}(x)$ as defined in (23). Let $j \in [m]$ correspond to an element $u_j \in U$, let $\ell_j := \|\mathbf{X}_0\|_{j,1}$, $r_j := \|\mathbf{X}_1\|_{j,1}$, and let $\Delta_j := |u_j\ell_j - (1-u_j)r_j|$. There exists $j' \in [m]$ such that we can move mass from only \mathbf{X}_j to $\mathbf{X}_{j'}$, resulting in $\mathbb{R}^{m \times 2n} \ni \mathbf{X}' \equiv x' \in \Delta^{2mn}$ such that $\mathbf{M}x' = \mathbf{M}x$, $u_j\|\mathbf{X}'_0\|_{j,1} = (1-u_j)\|\mathbf{X}'_1\|_{j,1}$, and $\langle c, x' - x \rangle \leq \Delta_j$.*

Proof. Without loss of generality, suppose that the row $j = 1$ corresponds to $u_j = 0$, and $j = m$ corresponds to $u_j = 1$. We split the proof into two cases, depending on the sign of $u_j\ell_j - (1-u_j)r_j$.

Case 1: $u_j\ell_j > (1-u_j)r_j$. We let $j' = 1$, i.e. we only move mass from the j^{th} row to the first row. Specifically, we leave $\mathbf{X}_1\|_{j,1}$ unchanged, and move mass from $\mathbf{X}_0\|_{j,1}$ to $\mathbf{X}_0\|_{1,1}$, making sure to only move mass in the same column. The total amount of mass we must delete from $\mathbf{X}_0\|_{j,1}$ is

$$\ell_j - \frac{1-u_j}{u_j} \cdot r_j = \frac{u_j\ell_j - (1-u_j)r_j}{u_j} = \frac{\Delta_j}{u_j}.$$

Our strategy is to arbitrarily move mass within columns until we have deleted $\frac{\Delta_j}{u_j}$ total mass. If we denote the mass moved in column $i \in [n]$ as δ_{ij} , and let x' be the result after the move,

$$\langle c, x' - x \rangle \leq \sum_{i \in [n]} |c_{(i,1,0)} - c_{(i,j,0)}| \delta_{ij} = \sum_{i \in [n]} ||u_j - v_i| - |u_1 - v_i|| \delta_{ij} \leq \sum_{i \in [n]} u_j \delta_{ij} = \Delta_j.$$

Here, the first inequality was the triangle inequality, the first equality used the definition of c in (22), the second inequality used $u_1 = 0$ and the triangle inequality, and the last used $\sum_{i \in [n]} \delta_{ij} = \frac{\Delta_j}{u_j}$.

Case 2: $u_j \ell_j < (1 - u_j)r_j$. This case is entirely analogous; we move mass arbitrarily from row j to row m , i.e. the last row with $u_m = 1$. The amount of mass we must move is

$$r_j - \frac{u_j}{1 - u_j} \ell_j = \frac{(1 - u_j)r_j - u_j \ell_j}{1 - u_j} = \frac{\Delta_j}{1 - u_j}.$$

Again denoting the amount of mass moved from column $i \in [n]$ as δ_{ij} , the claim follows:

$$\langle c, x' - x \rangle \leq \sum_{i \in [n]} \|u_j - v_i\| - |u_m - v_i| \delta_{ij} \leq \sum_{i \in [n]} (1 - u_j) \delta_{ij} = \Delta_j.$$

□

By iteratively applying Lemma 21, we have our marginal-preserving calibration procedure.

Lemma 22 (Marginal-preserving calibration). *Following the notation (24), given $x \in \Delta^{2mn}$ with $\|\mathbf{B}x\|_1 = \Delta$, we can compute x' with $\mathbf{M}x' = \mathbf{M}x$, $\mathbf{B}x' = \mathbf{0}_m$, and $\langle c, x' - x \rangle \leq \Delta$ in $O(mn)$ time.*

Proof. It suffices to apply Lemma 21 to each row $i \in [m]$. All of the movement in the rows $i \in [2, m - 1]$ are independent of each other, and do not affect the imbalance in the rows $i \in \{1, m\}$ when we have finished applying Lemma 21, since e.g. $u_1 \ell_1 = 0$ regardless of how much mass is moved to $[\mathbf{X}_0]_1$, and a similar property holds for the m^{th} row. The total change in $\langle c, x' - x \rangle$ is thus boundable by $\sum_{j \in [m]} \Delta_j \leq \Delta$, and applying Lemma 21 to each row takes $O(n)$ time. Finally, $\mathbf{M}x = \mathbf{M}x'$ follows because we only move mass within the same column, so no marginal changes. □

By combining Lemma 20 with Lemma 22, we can complete our rounding procedure.

Lemma 23. *Let $(\mathbf{A}, b, c, \mathcal{X})$ be defined as in (22), (24), and let $(\tilde{\mathbf{A}}, \tilde{b}) := (4\mathbf{A}, 4b)$. There exists Round, a $(\tilde{\mathbf{A}}, \tilde{b}, 1)$ -equality rounding procedure for $(\mathbf{A}, b, c, \mathcal{X})$, running in $O(mn)$ time.*

Proof. Throughout the proof, let $\Delta_{\mathbf{M}} := \|\mathbf{M}x - \frac{1}{n} \mathbb{1}_n\|_1$ and $\Delta_{\mathbf{B}} := \|\mathbf{B}x\|_1$, following the notation (24). We also denote the total violation by

$$\Delta := \|\tilde{\mathbf{A}}x - \tilde{b}\|_1 = 4\Delta_{\mathbf{M}} + 4\Delta_{\mathbf{B}}.$$

We first apply Lemma 20 to x to produce \tilde{x} satisfying $\|x - \tilde{x}\|_1 \leq 2\Delta_{\mathbf{M}}$ and $\mathbf{M}\tilde{x} = \frac{1}{n} \mathbb{1}_n$, in $O(mn)$ time. Note that, because $\|\mathbf{B}\|_{1 \rightarrow 1} \leq 1$ since all columns of \mathbf{B} are 1-sparse, we have

$$\|\mathbf{B}\tilde{x}\|_1 \leq \|\mathbf{B}x\|_1 + \|\mathbf{B}\|_{1 \rightarrow 1} \|x - \tilde{x}\|_1 \leq \Delta_{\mathbf{B}} + 2\Delta_{\mathbf{M}}.$$

Next, we apply Lemma 22 to \tilde{x} , resulting in x' with $\mathbf{M}x' = \frac{1}{n} \mathbb{1}_n$, $\mathbf{B}x' = \mathbf{0}_m$, and $\langle c, x' - \tilde{x} \rangle \leq \Delta_{\mathbf{B}} + 2\Delta_{\mathbf{M}}$, in $O(mn)$ time. Recalling the definition (19), we have the conclusion from $\|c\|_{\infty} \leq 1$, so

$$\begin{aligned} c^\top(x' - x) &\leq c^\top(\tilde{x} - x) + c^\top(x' - \tilde{x}) \\ &\leq \|c\|_{\infty} \|\tilde{x} - x\|_1 + c^\top(x' - \tilde{x}) \leq 2\Delta_{\mathbf{M}} + \Delta_{\mathbf{B}} + 2\Delta_{\mathbf{M}} \leq \Delta. \end{aligned}$$

□

We conclude by applying the solver from Corollary 2 to our resulting unconstrained linear program.

Proposition 3. *Let $\varepsilon \geq 0$. We can compute $x \in \mathcal{X}$, an ε -approximate minimizer to (22), in time*

$$O\left(\frac{n \log(n)}{\varepsilon^2}\right).$$

Further, the objective value of x in (22) is a 2ε -additive approximation of $\text{dCE}(\hat{\mathcal{D}}_n)$.

Proof. Observe that for $\tilde{\mathbf{A}} = 4\mathbf{A}$, we have $\|\tilde{\mathbf{A}}\|_{1 \rightarrow 1} \leq 8$ and $\text{nnz}(\tilde{\mathbf{A}}) = O(mn)$, since no column is more than 2-sparse and all entries of \mathbf{A} are in $[-1, 1]$. Further, recalling the definition of U from (21), we have $m = O(\frac{1}{\varepsilon})$. So, Corollary 2 shows we can compute an ε -approximate minimizer to

$$\min_{x \in \Delta^{2mn}} c^\top x + \|\tilde{\mathbf{A}}x - \tilde{b}\|_1$$

within the stated runtime. The rest of the proof follows using Round from Lemma 23, where we recall $|\text{dCE}(\hat{\mathcal{D}}_n) - \text{dCE}^U(\hat{\mathcal{D}}_n)| \leq \varepsilon$ due to our definition of U and Lemma 17. □

C.4 Testing via LDTC

We now give analogs of Theorems 1 and 3, using our solver in Proposition 3.

Theorem 4. *Let $0 \leq \varepsilon_2 \leq \varepsilon_1 \leq 1$ satisfy $\varepsilon_1 > \varepsilon_2$, and let $n \geq C_{\text{tct}} \cdot \frac{1}{(\varepsilon_1 - \varepsilon_2)^2}$ for a universal constant C_{tct} . There is an algorithm \mathcal{A} which solves the $(\varepsilon_1, \varepsilon_2)$ -tolerant calibration testing problem with n samples, which runs in time*

$$O\left(\frac{n \log(n)}{(\varepsilon_1 - \varepsilon_2)^2}\right).$$

Proof. Throughout the proof, let $\alpha := \varepsilon_1 - \varepsilon_2 > 0$. Consider the following algorithm.

1. For $|U| = m \geq \frac{6}{\alpha}$, sample $n \geq \frac{C_{\text{tct}}}{\alpha^2}$ samples to form an empirical distribution $\widehat{\mathcal{D}}_n$, where C_{tct} is chosen so Lemma 3 guarantees $|\underline{\text{dCE}}(\mathcal{D}) - \underline{\text{dCE}}(\widehat{\mathcal{D}}_n)| \leq \frac{\alpha}{6}$ with probability $\geq \frac{2}{3}$.
2. Call Proposition 2 with $\varepsilon \leftarrow \frac{\alpha}{6}$ to obtain β , an $\frac{\alpha}{3}$ -additive approximation to $|\underline{\text{dCE}}(\widehat{\mathcal{D}}_n)|$.
3. Return “yes” if $\beta \leq \varepsilon_2 + \frac{\alpha}{2}$, and return “no” otherwise.

Conditioned on the event that $|\underline{\text{dCE}}(\mathcal{D}) - \underline{\text{dCE}}(\widehat{\mathcal{D}}_n)| \leq \frac{\alpha}{6}$, we show that the algorithm succeeds. First, if $\underline{\text{dCE}}(\mathcal{D}) \leq \varepsilon_2$, then $\underline{\text{dCE}}(\widehat{\mathcal{D}}_n) \leq \varepsilon_2 + \frac{\alpha}{6}$ by assumption, and so $\beta \leq \varepsilon_2 + \frac{\alpha}{2}$ by Proposition 2, so the tester will return “yes.” Second, if $\underline{\text{dCE}}(\mathcal{D}) \geq \varepsilon_1$, then $\underline{\text{dCE}}(\widehat{\mathcal{D}}_n) \geq \varepsilon_1 - \frac{\alpha}{6}$ by assumption, so $\beta \geq \varepsilon_1 - \frac{\alpha}{2}$ by Proposition 2 and similarly the tester will return “no” in this case. Finally, the runtime is immediate from Proposition 3 and the definition of α . \square

Theorem 4 has the following implication for (standard) calibration testing, by letting $\varepsilon_2 = 0$.

Corollary 3. *Let $n \in \mathbb{N}$ and let $\varepsilon_n \in (0, 1)$ be minimal such that it is information-theoretically possible to solve the ε_n -calibration testing problem with n samples. For some $\varepsilon = \Theta(\varepsilon_n)$, there is an algorithm \mathcal{A} which solves the ε -calibration testing problem with n samples, which runs in time*

$$O(n^2 \log(n)).$$

D Sample complexity lower bounds for calibration measures

Recent works [BN23, BGHN23a] have introduced other calibration measures (e.g. the convolved ECE and interval CE), given efficient estimation algorithms for them, and showed that they are polynomially related to the lower distance to calibration $\underline{\text{dCE}}$. Therefore, an alternative approach to the (non-tolerant) testing problem for $\underline{\text{dCE}}$ is by reducing it to testing problems for these measures. The main result of this section is that this approach leads to suboptimal sample complexity: the testing problems for these measures cannot be solved given only $O(\varepsilon^{-2})$ data points $\{(v_i, y_i)\}_{i \in [n]}$.

To establish this sample complexity lower bound, we construct a perfectly calibrated distribution \mathcal{D}_0 and a family of miscalibrated distributions \mathcal{D}_θ parameterized by θ belonging to a finite set. We use $\mathcal{D}_0^{\otimes n}$ (and $\mathcal{D}_\theta^{\otimes n}$) to denote the joint distribution of n independent examples from \mathcal{D}_0 (and \mathcal{D}_θ). In Lemma 24, we show that the total variation distance between $\mathcal{D}_0^{\otimes n}$ and the mixture $\mathbb{E}[\mathcal{D}_\theta^{\otimes n}]$ of $\mathcal{D}_\theta^{\otimes n}$ is small unless n is large, and thus distinguishing them requires large sample complexity. Consequently, the testing problem for a calibration measure has large sample complexity if it assigns every \mathcal{D}_θ a large calibration error. Finally, we show every \mathcal{D}_θ indeed has large convolved ECE and interval CE, establishing sample complexity lower bounds for these measures in Theorems 5 and 6.

To construct \mathcal{D}_0 and \mathcal{D}_θ , we consider t values $\{u_i\}_{i \in [t]} \in [\frac{1}{3}, \frac{2}{3}]$ where $u_i = \frac{1}{3} + \frac{i}{3t}$ for $i \in [t]$. We will determine the value of $t \in \mathbb{N}$ later. We also define the following distribution, a perfectly calibrated distribution which is related to the miscalibrated synthetic dataset used in Section 3.

Definition 8. *The distribution \mathcal{D}_0 of $(v, y) \in [0, 1] \times \{0, 1\}$ is defined such that the marginal distribution of v is uniform over $\{u_i\}_{i \in [t]}$ and $\mathbb{E}_{\mathcal{D}_0}[y|v] = v$.*

Fix $\alpha \in (0, \frac{1}{3})$. For $\theta \in \{-1, 1\}^t$, we define distribution \mathcal{D}_θ of $(v, y) \in [0, 1] \times \{0, 1\}$ such that the marginal distribution of v is uniform over $\{u_i\}_{i \in [t]}$ and $\mathbb{E}_{\mathcal{D}_\theta}[y|v = u_i] = u_i + \theta_i \alpha$. In other words,

each conditional distribution given v is miscalibrated by α , but the bias takes a random direction. We now follow a standard approach by [IS03] to bound the total variation between our distributions.

Lemma 24. For any $t \in \mathbb{N}$ and $\alpha \in (0, \frac{1}{3})$,

$$d_{\text{TV}}(\mathcal{D}_0^{\otimes n}, \mathbb{E}_\theta[\mathcal{D}_\theta^{\otimes n}]) \leq \frac{1}{2} \sqrt{\exp\left(\frac{11\alpha^4 n^2}{t}\right) - 1}.$$

Here, to construct the mixture distribution $\mathbb{E}_\theta[\mathcal{D}_\theta^{\otimes n}]$, we first draw $\theta \sim_{\text{unif.}} \{-1, 1\}^t$, and then draw n independent examples from \mathcal{D}_θ . We denote the distribution of the n examples by $\mathbb{E}_\theta[\mathcal{D}_\theta^{\otimes n}]$.

Proof. By a standard inequality between the total variation distance and the χ^2 distance, we have

$$d_{\text{TV}}(\mathcal{D}_0^{\otimes n}, \mathbb{E}_\theta[\mathcal{D}_\theta^{\otimes n}]) \leq \frac{1}{2} \sqrt{\chi^2(\mathbb{E}_\theta[\mathcal{D}_\theta^{\otimes n}] \| \mathcal{D}_0^{\otimes n})}. \quad (25)$$

By Ingster's method [IS03] (see also Section 3.1 of [Can22]),

$$\chi^2(\mathbb{E}_\theta[\mathcal{D}_\theta^{\otimes n}] \| \mathcal{D}_0^{\otimes n}) = \mathbb{E}_{\theta, \theta'} \left[\left(\sum_{i=1}^t \sum_{j \in \{0,1\}} \frac{\mathcal{D}_\theta(u_i, j) \mathcal{D}_{\theta'}(u_i, j)}{\mathcal{D}_0(u_i, j)} \right)^n \right] - 1, \quad (26)$$

where the expectation is over θ, θ' drawn i.i.d. $\sim_{\text{unif.}} \{-1, 1\}^t$. For every $i \in [t]$, we have

$$\frac{\mathcal{D}_\theta(u_i, 1) \mathcal{D}_{\theta'}(u_i, 1)}{\mathcal{D}_0(u_i, 1)} = \frac{\left(\frac{u_i + \theta_i \alpha}{t}\right) \left(\frac{u_i + \theta'_i \alpha}{t}\right)}{\frac{u_i}{t}} = \frac{u_i}{t} + \frac{(\theta_i + \theta'_i) \alpha}{t} + \frac{\theta_i \theta'_i \alpha^2}{u_i t},$$

and similarly

$$\frac{\mathcal{D}_\theta(u_i, 0) \mathcal{D}_{\theta'}(u_i, 0)}{\mathcal{D}_0(u_i, 0)} = \frac{\left(\frac{(1-u_i - \theta_i \alpha)}{t}\right) \left(\frac{(1-u_i - \theta'_i \alpha)}{t}\right)}{\frac{1-u_i}{t}} = \frac{1-u_i}{t} - \frac{(\theta_i - \theta'_i) \alpha}{t} + \frac{\theta_i \theta'_i \alpha^2}{(1-u_i)t}.$$

Adding up the two equations, we get

$$\sum_{j \in \{0,1\}} \frac{\mathcal{D}_\theta(u_i, j) \mathcal{D}_{\theta'}(u_i, j)}{\mathcal{D}_0(u_i, j)} = \frac{1}{t} + \frac{\theta_i \theta'_i \alpha^2}{t} \left(\frac{1}{u_i} + \frac{1}{1-u_i} \right) \leq \frac{1}{t} + \frac{9\theta_i \theta'_i \alpha^2}{2t},$$

where the last inequality uses the fact that $u_i \in [\frac{1}{3}, \frac{2}{3}]$. Plugging this into (26), we get

$$\begin{aligned} \chi^2(\mathbb{E}_\theta[\mathcal{D}_\theta^{\otimes n}] \| \mathcal{D}_0^{\otimes n}) &\leq \mathbb{E}_{\theta, \theta'} \left[\left(1 + \frac{9\alpha^2}{2t} \sum_{i=1}^t \theta_i \theta'_i \right)^n \right] - 1 \\ &\leq \mathbb{E}_{\theta, \theta'} \left[\exp\left(\frac{9\alpha^2 n}{2t} \sum_{i=1}^t \theta_i \theta'_i\right) \right] - 1 \\ &= \prod_{i=1}^t \mathbb{E}_{\theta, \theta'} \left[\exp\left(\frac{9\alpha^2 n}{2t} \theta_i \theta'_i\right) \right] - 1 \\ &\leq \prod_{i=1}^t \mathbb{E}_\theta \left[\exp\left(\frac{81\alpha^4 n^2}{8t^2} \theta_i^2\right) \right] - 1 \quad (\text{by Hoeffding's lemma}) \\ &= \exp\left(\frac{81\alpha^4 n^2}{8t}\right) - 1. \end{aligned}$$

Plugging this into (25) completes the proof. \square

D.1 Lower bound for convolved ECE

We now introduce the definition of convolved ECE from [BN23], and show that for every $\theta \in \{-1, 1\}^t$, \mathcal{D}_θ has a large convolved ECE in Lemma 26. This allows us to prove our sample complexity lower bound for convolved ECE in Theorem 5, by applying Lemma 24.

Definition 9 (Convolved ECE [BN23]). Let $\pi_R : \mathbb{R} \rightarrow [0, 1]$ be the periodic function with period 2 satisfying $\pi_R(v) = v$ if $v \in [0, 1]$, and $\pi_R(v) = 2 - v$ if $v \in [1, 2]$. Consider a distribution \mathcal{D} over $[0, 1] \times \{0, 1\}$. For $(v, y) \sim \mathcal{D}$, define random variable $\hat{v} := \pi_R(v + \eta)$, where η is drawn independently from $\mathcal{N}(0, \sigma^2)$ for a parameter $\sigma \geq 0$. The σ -convolved ECE is defined as follows:

$$\text{cECE}_\sigma(\mathcal{D}) := \mathbb{E}[\mathbb{E}[(y - v)|\hat{v}]],$$

where the outer expectation is over the marginal distribution of \hat{v} , and the inner expectation is over the conditional distribution of (y, v) given \hat{v} . It has been shown in [BN23] that $\text{cECE}_\sigma(\mathcal{D}) \in [0, 1]$ is a nonincreasing function of $\sigma \geq 0$ and there exists a unique $\sigma^* \geq 0$ satisfying $\text{cECE}_{\sigma^*}(\mathcal{D}) = \sigma^*$. The convolved ECE $\text{cECE}(\mathcal{D})$ is defined to be $\text{cECE}_{\sigma^*}(\mathcal{D})$.

We also mention that the following relationship is known between cECE and dCE.

Lemma 25 (Theorem 7, [BN23]). For any distribution \mathcal{D} over $[0, 1] \times \{0, 1\}$, it holds that

$$\frac{1}{2} \text{dCE}(\mathcal{D}) \leq \text{cECE}(\mathcal{D}) \leq 2\sqrt{\text{dCE}(\mathcal{D})}.$$

We have the following lower bound on $\text{cECE}(\mathcal{D}_\theta)$:

Lemma 26. For integer $t \geq 3$, choose $\alpha = \frac{1}{t\sqrt{\ln t}}$. Then for every $\theta \in \{-1, 1\}^t$,

$$\text{cECE}(\mathcal{D}_\theta) \geq \frac{1}{100t\sqrt{\ln t}}.$$

Proof. It suffices to show that $\text{cECE}_\sigma(\mathcal{D}_\theta) \geq \frac{1}{100t\sqrt{\ln t}}$ whenever $\sigma \leq \frac{1}{100t\sqrt{\ln t}}$.

Consider $(v, y) \sim \mathcal{D}$ and $\hat{v} = \pi_R(v + \eta)$, where η is drawn independently from $\mathcal{N}(0, \sigma^2)$. By standard Gaussian tail bounds, we have

$$\Pr\left[|\eta| \geq \frac{1}{6t}\right] \leq \frac{1}{t^2}. \quad (27)$$

Next, consider a function $\ell : [0, 1] \rightarrow [t]$ such that $\ell(\hat{v}) \in \text{argmin}_{i \in [t]} |u_i - \hat{v}|$. Let \mathcal{E} denote the event that $v = u_{\ell(\hat{v})}$. Let $\mathbb{1}_\mathcal{E}$ and $\mathbb{1}_{-\mathcal{E}}$ be the indicators of \mathcal{E} and its complement, respectively. We have

$$\begin{aligned} \mathbb{E}[(y - v)\mathbb{1}_\mathcal{E} | \hat{v}, v] &= \mathbb{1}_\mathcal{E} \mathbb{E}[y - v | \hat{v}, v] && (\mathbb{1}_\mathcal{E} \text{ is fully determined by } v \text{ and } \hat{v}) \\ &= \mathbb{1}_\mathcal{E} \mathbb{E}[y - v | v] && (y \text{ is independent of } \hat{v} \text{ given } v) \\ &= \mathbb{1}_\mathcal{E} \mathbb{E}[y - v | v = u_{\ell(\hat{v})}] = \mathbb{1}_\mathcal{E} \theta_{\ell(\hat{v})} \alpha. \end{aligned}$$

Taking expectation over v conditioned on \hat{v} , we have

$$|\mathbb{E}[(y - v)\mathbb{1}_\mathcal{E} | \hat{v}]| = |\Pr[\mathcal{E} | \hat{v}] \theta_{\ell(\hat{v})} \alpha| = \Pr[\mathcal{E} | \hat{v}] \alpha.$$

We also have

$$\left| \mathbb{E}[(y - v)\mathbb{1}_{-\mathcal{E}} | \hat{v}] \right| \leq \mathbb{E}[|(y - v)\mathbb{1}_{-\mathcal{E}}| | \hat{v}] \leq \Pr[-\mathcal{E} | \hat{v}].$$

Therefore,

$$|\mathbb{E}[y - v | \hat{v}]| \geq \Pr[\mathcal{E} | \hat{v}] \alpha - \Pr[-\mathcal{E} | \hat{v}].$$

Taking expectations over \hat{v} , we have

$$\text{cECE}_\sigma(\mathcal{D}_\theta) = \mathbb{E}[|\mathbb{E}[y - v | \hat{v}]|] \geq \Pr[\mathcal{E}] \alpha - \Pr[-\mathcal{E}]. \quad (28)$$

Whenever \mathcal{E} does not occur, it must hold that $|v - \hat{v}| \geq \frac{1}{6t}$, which can only hold when $|\eta| \geq \frac{1}{6t}$. Therefore, by plugging (27) into (28), we get

$$\text{cECE}_\sigma(\mathcal{D}_\theta) \geq \left(1 - \frac{1}{t^2}\right) \alpha - \frac{1}{t^2} \geq \frac{1}{100t\sqrt{\ln t}}. \quad \square$$

Theorem 5. If \mathcal{A} is an ε -cECE tester with n samples (Definition 5), for $\varepsilon \in (0, \frac{1}{3})$, then

$$n = \Omega\left(\frac{1}{\varepsilon^{2.5} \ln^{0.25}(\frac{1}{\varepsilon})}\right).$$

Proof. Without loss of generality, assume that $\varepsilon \leq 10^{-3}$. Let $t \geq 3$ be the largest integer satisfying $\varepsilon \leq \frac{1}{100t\sqrt{\ln t}}$. We choose $\alpha = \frac{1}{t\sqrt{\ln t}}$.

By Lemma 26, we have $\text{cECE}(\mathcal{D}_\theta) \geq \varepsilon$ for every $\theta \in \{-1, 1\}^t$. By the guarantee of the tester, we have

$$d_{\text{TV}}(\mathcal{D}_0^{\otimes n}, \mathbb{E}_\theta[\mathcal{D}_\theta^{\otimes n}]) \geq \frac{1}{3}.$$

Combining this with Lemma 24, we get $n = \Omega(\alpha^{-2}\sqrt{t}) = \Omega(t^{2.5} \ln t)$, so the claim holds. \square

D.2 Lower bound for (surrogate) interval CE

The *interval calibration error* was introduced in [BGHN23a] as a modified version of the popular *binned ECE* to obtain a polynomial relationship to the lower distance to calibration (dCE). To give an efficient estimation algorithm, [BGHN23a] considered a slight variant of the interval calibration error, called the *surrogate interval calibration error*, which preserves the polynomial relationship. Below we include the definition of the surrogate interval calibration error, and its polynomial relationship with dCE . We then establish our sample complexity lower bound (Theorem 6) for surrogate interval CE by showing that every \mathcal{D}_θ has a large surrogate interval CE (Lemma 28).

Definition 10 ([BGHN23a]). For a distribution \mathcal{D} over $[0, 1] \times \{0, 1\}$ and an interval width parameter $w > 0$, the random interval calibration error is defined to be

$$\text{RintCE}(\mathcal{D}, w) := \mathbb{E}_r \left[\sum_{j \in \mathbb{Z}} |\mathbb{E}_{(v,y) \sim \mathcal{D}}[(y-v)\mathbb{1}(v \in I_{r,j}^w)]| \right], \quad (29)$$

where the outer expectation is over r drawn uniformly from $[0, w)$ and $I_{r,j}^w$ is the interval $[r + j\varepsilon, r + (j+1)\varepsilon)$. Note that although the summation is over $j \in \mathbb{Z}$, there are only finitely many j that can contribute to the sum (which are the j that satisfy $I_{r,j}^w \cap [0, 1] \neq \emptyset$). The surrogate interval calibration error is defined as follows:

$$\text{SintCE}(\mathcal{D}) := \inf_{k \in \mathbb{Z}_{\geq 0}} \left(\text{RintCE}(\mathcal{D}, 2^{-k}) + 2^{-k} \right).$$

Lemma 27 (Theorem 6.11, [BGHN23a]). For any distribution \mathcal{D} over $[0, 1] \times \{0, 1\}$, it holds that

$$\text{dCE}(\mathcal{D}) \leq \text{SintCE}(\mathcal{D}) \leq 6\sqrt{\text{dCE}(\mathcal{D})}.$$

Lemma 28. For $t \in \mathbb{N}$, let $\alpha = \frac{1}{3t}$. Then for every $\theta \in \{-1, 1\}^t$, it holds that $\text{SintCE}(\mathcal{D}_\theta) \geq \frac{1}{3t}$.

Proof. It suffices to prove that $\text{RintCE}(\mathcal{D}_\theta, w) \geq \frac{1}{3t}$ whenever $w < \frac{1}{3t}$, where we recall the definition (29). Fix some $r \in [0, 1]$. Every u_i belongs to the interval I_{r,j_i}^w for a unique $j_i \in \mathbb{Z}$. Since the interval width w is smaller than the gap between u_i and $u_{i'}$ for distinct i, i' , we have $j_i \neq j_{i'}$. Therefore,

$$\begin{aligned} \sum_{j \in \mathbb{Z}} |\mathbb{E}_{(v,y) \sim \mathcal{D}}[(y-v)\mathbb{1}(v \in I_{r,j}^w)]| &\geq \sum_{i \in [t]} |\mathbb{E}_{(v,y) \sim \mathcal{D}}[(y-v)\mathbb{1}(v \in I_{r,j_i}^w)]| \\ &= \sum_{i \in [t]} |\mathbb{E}_{(v,y) \sim \mathcal{D}}[(y-v)\mathbb{1}(v = u_i)]| \\ &= \sum_{i \in [t]} \Pr[v = u_i] \mathbb{E}[y-v | v = u_i] \\ &= \frac{1}{3t}. \end{aligned}$$

Plugging this into (29), we get $\text{RintCE}(\mathcal{D}_\theta, w) \geq \frac{1}{3t}$. \square

Theorem 6. If \mathcal{A} is an ε -SintCE tester with n samples (Definition 5), for $\varepsilon \in (0, \frac{1}{3})$, then

$$n = \Omega\left(\frac{1}{\varepsilon^{2.5}}\right).$$

Proof. Choose $t \geq 1$ to be the largest integer satisfying $\frac{1}{3t} \geq \varepsilon$, and choose $\alpha = \frac{1}{3t}$. By Lemma 28, $\text{SintCE}(\mathcal{D}_\theta) \geq \varepsilon$ for every $\theta \in \{-1, 1\}^t$. By the guarantee of the tester, we have

$$d_{\text{TV}}(\mathcal{D}_0^{\otimes n}, \mathbb{E}_\theta[\mathcal{D}_\theta^{\otimes n}]) \geq \frac{1}{3}.$$

Combining this with Lemma 24, we get $n = \Omega(\alpha^{-2}\sqrt{t}) = \Omega(\varepsilon^{-2.5})$. □

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [Yes]

Justification: We claimed a novel testing problem for measuring calibration, faster algorithms for solving it, and lower bounds for alternative measures. This is precisely what we provide.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: We explain that our basic tester does not solve the tolerant testing problem in all regimes, and that our more powerful tolerant tester has a slower runtimes. We also state that our practical implementation is preliminary and could be optimized.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory Assumptions and Proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We give full, detailed proofs of all theoretical claims.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental Result Reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide a complete description of our experimental setups, and all of the relevant code in the supplementary material for reproducibility.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: As previously mentioned, we give full descriptions and provide our code.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental Setting/Details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: See above.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment Statistical Significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We replicated our experiments across multiple runs and provide error bars.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments Compute Resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We describe all our computing environments and provide runtime metrics.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code Of Ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: We read the ethics guidelines and believe we meet them.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader Impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: We provide a new tool for assessing calibration, a measure of interpretability of models, which is not tied to any particular societal application, and do not foresee a pathway to use this tool for negative societal impact.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: We only use simple synthetic or public datasets for our experiments.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We made sure to cite all relevant code and data used in our experiments.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: We do not release new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: Our paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: Our paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.