Feasible Learning

Juan Ramirez[⋆,1] **Ignacio Hounie**[⋆,2] **Juan Elenter**[⋆,3,◇] **Jose Gallego-Posada**[⋆,1]
Meraj Hashemizadeh[1] **Alejandro Ribeiro**[†,2] **Simon Lacoste-Julien**[†,1,4]

[1]Mila & Université de Montréal [2]University of Pennsylvania [3]Spotify [4]Canada CIFAR AI Chair

Abstract

We introduce Feasible Learning (**FL**), a sample-centric learning paradigm where models are trained by solving a feasibility problem that bounds the loss for each training sample. In contrast to the ubiquitous Empirical Risk Minimization (**ERM**) framework, which optimizes for average performance, **FL** demands satisfactory performance *on every individual data point*. Since any model that meets the prescribed performance threshold is a valid **FL** solution, the choice of optimization algorithm and its dynamics play a crucial role in shaping the properties of the resulting solutions. In particular, we study a primal-dual approach which dynamically re-weights the importance of each sample during training. To address the challenge of setting a meaningful threshold in practice, we introduce a relaxation of **FL** that incorporates slack variables of minimal norm. Our empirical analysis, spanning image classification, age regression, and preference optimization in large language models, demonstrates that models trained via **FL** can learn from data while displaying improved tail behavior compared to **ERM**, with only a marginal impact on average performance.

1 INTRODUCTION

Deep learning trends are shifting toward larger model architectures, as evidenced by GPT-4 (OpenAI, 2023), DALLE-3 (Betker et al., 2023), and Llama-3 (Llama Team, 2024). Larger models are capable of perfectly fitting increasingly large datasets, *memorizing* the data by achieving near-zero loss on all samples (Arpit et al., 2017; Zhang et al., 2017a). In this context, the Empirical Risk Minimization (**ERM**) framework does not specify a preference among the many interpolating solutions. Consequently, the solution recovered in practice depends not only on the learning framework but also on the inductive biases of the chosen optimization algorithm. For instance, Soudry et al. (2018) highlight the role of stochastic gradient descent dynamics in guiding **ERM** toward well-generalizing models.

While research has extensively focused on developing optimization algorithms suited for learning via **ERM** (Kingma & Ba, 2015; Gupta et al., 2018), exploring alternative learning frameworks has received comparatively little attention. Alternatives to **ERM** could be better suited for specific learning scenarios, particularly when it is important to optimize for something other than average performance. Such alternatives may exhibit distinct properties, such as improved uncertainty quantification (Balasubramanian et al., 2014), fairness (Lahoti et al., 2020), or robustness (Mądry et al., 2017).

Given the abundance of interpolating, well-generalizing solutions in modern machine learning problems, why would we limit ourselves to those derived from **ERM**?

In this paper, we introduce a novel learning framework called Feasible Learning (**FL**, §2). **FL** formulates learning as a *feasibility* problem, where we seek a predictor that meets a *bounded loss constraint for all training samples*. Unlike the ubiquitous **ERM** framework, which optimizes for average performance, **FL** demands a minimum performance level for each data point.

Concretely, **FL** formulates an optimization problem with a trivial (constant) objective function, while imposing a constraint on the loss of the predictor $h : \mathcal{X} \to \mathcal{Y}$ for each training sample $(\boldsymbol{x}_i, \boldsymbol{y}_i)_{i=1}^n$:

$$\min_{h \in \mathcal{H}} 0 \quad \text{s.t} \quad \ell(\boldsymbol{y}_i, h(\boldsymbol{x}_i)) \leq \epsilon \;\; \text{for} \;\; i = 1, \ldots, n, \quad \textbf{(FL)}$$

where $\epsilon \geq 0$ is the maximum allowed per-sample loss.

Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258.

⋆Equal contribution. †Equal supervision.
◇Work done while at the University of Pennsilvania.
Correspondance to: juan.ramirez@mila.quebec

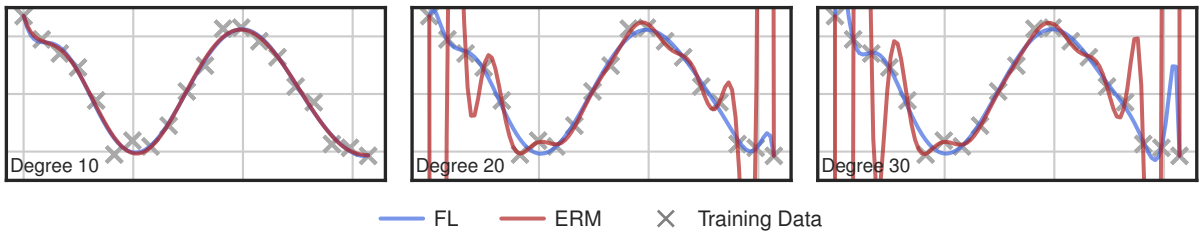


Figure 1: Fitting polynomials of varying degrees: **While ERM tends to overfit with high-degree polynomials, FL still recovers smoother solutions**. This occurs even though non-smooth solutions are also part of the **FL** solution set, highlighting the influence of the optimization algorithm on the final solution. Due to ill-conditioning, we solve both problems using a standard convex optimization solver. The data is generated by adding Gaussian noise ($\sigma = 0.2$) to a cosine wave. **FL**'s constraint value is set to one standard deviation, $\epsilon = \sigma$. See Appendix B for details.

The main properties of **FL** problems are presented in §2 and can be summarized as follows: ① **FL** is inherently *sample-centric*, requiring satisfactory performance across all training samples. This contrasts with the **ERM** framework, which aims to optimize average performance and may overlook poor performance on individual samples. ② **FL** does not establish a preference between feasible models that meet the constraints: while **ERM** seeks solutions with zero training loss, **FL** accepts any solution that satisfies the minimum performance threshold ϵ for each data point. This leads to ③ **FL** inducing functional regularization of the loss function which prevents overfitting to the training data.[1]

Figure 1 illustrates the behavior of **FL** on a polynomial regression task. Both **ERM** and **FL** find solutions that fit the data well. Note that the **ERM** solution is also a valid solution to the **FL** problem. However, **FL** recovers a qualitatively different predictor. Whereas the existence of such non-interpolating solutions is a property of the **FL** problem, finding them in practice depends on the choice of optimization algorithm and its dynamics.

We adopt a primal-dual optimization approach to solve **FL** problems, which amounts to performing a weighted version of **ERM** (§2.1). The weight of each sample corresponds to the Lagrange multiplier associated with its constraint, which is dynamically adjusted based on the "difficulty" of fitting said sample. We favor this approach because ① the algorithmic similarity with **ERM** allows to leverage established techniques for training deep neural networks, ② enabling it to scale to high-dimensional problems with large numbers of constraints; and ③ it does not inherently favor interpolating solutions over non-interpolating ones.

A challenging aspect of **FL** is determining a suitable constraint level ϵ. If set too tightly, the **FL** problem may be infeasible, i.e. it may not admit any solutions. Conversely, setting ϵ too loosely may fail to ensure good performance. To address this, we propose relaxing the **FL** problem with slack variables of minimal norm that would make the problem feasible. We call this approach Resilient Feasible Learning (**RFL**, §3). Our primal-dual approach to solve **RFL** problems (§3.1) enjoys enhanced convergence guarantees compared to primal-dual **FL**.

To gain a deeper understanding of Feasible Learning, we tackle the following questions:

(**Q1**) Can we learn via **FL**? Yes, deep networks trained via **FL** achieve comparable average performance to **ERM** on the train and test sets (§5.1), with equivalent training cost and similar hyperparameter robustness.

(**Q2**) How does **RFL** help? In problems where infeasibility leads to poor optimization dynamics for **FL**, **RFL** alleviates this issue, achieving good performance (§5.2).

(**Q3**) How do **FL** solutions compare to **ERM**? We observe that **FL** produces a more concentrated loss distribution across (training and test) samples, resulting in fewer instances with excessively high losses (§5.3).

The main contributions of our work are as follows:[2]

- We propose Feasible Learning (**FL**), framing learning as a constraint satisfaction problem (§2).
- We introduce Resilient Feasible Learning (**RFL**, §3), a relaxation of **FL** that addresses potential infeasibility issues. We show that **RFL** is equivalent to a non-convex, strongly-concave min-max problem (Proposition 1).
- We provide primal-dual algorithms for solving **FL** (§2.1) and **RFL** (§3.1) problems, which are as cheap as gradient descent on the **ERM** objective (up to the negligible cost of updating the dual variables).
- We perform an empirical exploration of **FL** and **RFL** problems and their solutions (§5), supporting our answers to questions **Q1**-**Q3**.

Scope: We introduce **FL**, present a practical algorithm for solving it, and provide empirical evidence that **FL**

[1] Provided the optimization algorithm of choice does not incentivize reducing the per-sample loss beyond ϵ.

[2] Our code is available at: https://github.com/juan43ramirez/feasible-learning.

is a compelling alternative to the widely used **ERM** framework. However, we do not intend to claim that **FL** outperforms **ERM** or other learning paradigms. As with any multi-objective problem, adequately balancing competing objectives (such as the losses on each sample) depends on the specific application and requirements.

Developing a statistical decision theory framework for analyzing **FL** problems is beyond the scope of our work. In particular, future research should determine which statistical goals **FL** problems are best suited for. Rather, we explore **FL** empirically and investigate its properties.

2 FEASIBLE LEARNING

We consider the problem of learning a predictor $h_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$ with parameters $\boldsymbol{\theta} \in \Theta$ on a labeled dataset $\mathfrak{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$. The quality of the predictions is measured by a differentiable (surrogate) loss function $\ell : \mathcal{Y}^2 \to \mathbb{R}_{\geq 0}$. To simplify the notation, we denote the loss incurred on the i-th data point as $g_i(\boldsymbol{\theta}) \triangleq \ell(\boldsymbol{y}_i, h_{\boldsymbol{\theta}}(\boldsymbol{x}_i))$, and the vectorized version of these losses as $\boldsymbol{g}(\boldsymbol{\theta}) = [g_1(\boldsymbol{\theta}), \dots, g_n(\boldsymbol{\theta})]^\top$.

The Feasible Learning (**FL**) paradigm advocates for learning through solving a *feasibility problem*. Specifically, **FL** considers an optimization problem with a trivial, constant objective while enforcing a loss constraint for each training sample:

$$\min_{\boldsymbol{\theta} \in \Theta} \; 0 \quad \text{s.t} \quad \boldsymbol{g}(\boldsymbol{\theta}) \leq \boldsymbol{\epsilon}, \qquad (\textbf{FL}(\boldsymbol{\epsilon}))$$

where $\boldsymbol{\epsilon} = \epsilon \mathbf{1}$ is the constraint level.[3] In the **FL** framework, a model is acceptable *if and only if* it achieves a sufficiently small loss (given ϵ) on every training point. Choosing ℓ as the squared error imposes a bound on the maximum error per sample, while upper-bounding the per-sample cross-entropy sets a lower bound on the probability the model assigns to the correct label.

When the model class Θ can interpolate the training data, setting $\epsilon = 0$ results in the set of solutions for **FL** matching that of **ERM**—consisting of those that achieve zero training loss on all samples. When $\epsilon > 0$, **FL** admits additional solutions—those that satisfy the constraints but do not necessarily interpolate the data.

In the non-interpolation case, we expect **FL** to outperform **ERM** in terms of the maximum loss over the dataset—potentially at the expense of a heightened average loss. However, this trend may not always manifest in practice due to the inherent challenges of solving non-convex (constrained) optimization problems.

Functional regularization. **FL** does not inherently favor interpolating or non-interpolating solutions, as long as both satisfy the constraints. This introduces a form of *functional* regularization: to prevent excessive minimization of the loss, **FL** regularizes *the loss itself* by not demanding to reduce it beyond a specified threshold ϵ. This approach to regularization is model-agnostic, differing from standard techniques that rely on objectives like smoothness, small norms, or sparsity.

When the model class Θ cannot interpolate the data, certain values of ϵ may result in infeasible **FL** problems—this can occur in tasks with low model capacity or with noisy or mislabeled data. This introduces a trade-off when selecting an appropriate ϵ: tight values can result in infeasible problems, while overly loose values may lead to vacuous constraints, potentially failing to ensure good performance. As appropriate constraint levels depend on the model parametrization, data, and task, selecting them can be challenging in practice.

While our experiments demonstrate that **FL** can recover useful solutions even if they do not satisfy all constraints (§5.1), it remains desirable to make the **FL** framework robust against problem misspecification. Thus, in §3 we propose finding the minimum norm constraint relaxation needed to make the problem feasible.

2.1 Solving FL Problems

Even if the loss function ℓ is convex in its inputs $(\boldsymbol{y}_i, h_{\boldsymbol{\theta}}(\boldsymbol{x}_i))$, it may not be convex in $\boldsymbol{\theta}$. Therefore, **FL**($\boldsymbol{\epsilon}$) is typically a *non-convex* constrained optimization problem with no closed-form solution. Furthermore, the lack of an objective function and the non-convexity of the feasible set preclude the use of standard constrained optimization techniques such as projected gradient descent (Goldstein, 1964) or Frank-Wolfe (Frank & Wolfe, 1956). Instead, we leverage Lagrangian duality. The min-max Lagrangian game associated with **FL**($\boldsymbol{\epsilon}$) is:

$$\min_{\boldsymbol{\theta} \in \Theta} \max_{\boldsymbol{\lambda} \geq \mathbf{0}} \mathfrak{L}_{\text{FL}}(\boldsymbol{\theta}, \boldsymbol{\lambda}) \triangleq \boldsymbol{\lambda}^\top (\boldsymbol{g}(\boldsymbol{\theta}) - \boldsymbol{\epsilon}), \qquad (1)$$

where $\boldsymbol{\lambda} \geq \mathbf{0}$ is the vector of Lagrange multipliers associated with the constraints. We refer to $\boldsymbol{\theta}$ as the *primal* variables, and $\boldsymbol{\lambda}$ as the *dual* variables. **FL** yields a Lagrangian with one multiplier λ_i per datapoint $\boldsymbol{x}_i$.

A simple algorithm for finding min-max points of $\mathfrak{L}_{\text{FL}}$ is to perform gradient descent steps on $\boldsymbol{\theta}$ and projected gradient ascent steps on $\boldsymbol{\lambda}$ (Arrow et al., 1958, GDA). Alternating GDA updates (Zhang et al., 2022) yield:

$$\begin{aligned} \boldsymbol{\lambda}_{t+1} &\leftarrow \Big[\boldsymbol{\lambda}_t + \eta_{\boldsymbol{\lambda}} \underbrace{(\boldsymbol{g}(\boldsymbol{\theta}_t) - \boldsymbol{\epsilon})}_{\nabla_{\boldsymbol{\lambda}} \mathfrak{L}_{\text{FL}}(\boldsymbol{\theta}_t, \boldsymbol{\lambda}_t)} \Big]_+ \\ \boldsymbol{\theta}_{t+1} &\leftarrow \boldsymbol{\theta}_t - \eta_{\boldsymbol{\theta}} \underbrace{\Big[\sum_{i=1}^n \boldsymbol{\lambda}_{t+1}^{(i)} \nabla_{\boldsymbol{\theta}} \, g_i(\boldsymbol{\theta}_t) \Big]}_{\nabla_{\boldsymbol{\theta}} \mathfrak{L}_{\text{FL}}(\boldsymbol{\theta}_t, \boldsymbol{\lambda}_{t+1})}, \end{aligned} \qquad (2)$$

where $[\cdot]_+$ denotes a projection onto $\mathbb{R}^n_{\geq \mathbf{0}}$ to enforce $\boldsymbol{\lambda} \geq 0$, and $\eta_{\{\boldsymbol{x}, \boldsymbol{\lambda}\}}$ are step sizes. We initialize $\boldsymbol{\lambda}_0 = \mathbf{0}$.

[3] Using different constraint levels per sample is possible.

The primal updates in Eq. (2) resemble gradient descent on the **ERM** objective by following the gradients of the per-sample losses. However, unlike **ERM**, where these gradients are weighted equally, **FL** uses the Lagrange multipliers as weights. Since these multipliers are *optimized*, the algorithm dynamically re-weights the importance of each data point throughout training.

The re-weighting works as follows: if $g_i(\boldsymbol{\theta}) > \epsilon$, the corresponding multipler increases; if $g_i(\boldsymbol{\theta}) < \epsilon$, the multiplier decreases, potentially reaching zero. Consequently, data points with consistently high losses result in large multipliers, causing the primal updates to focus on reducing their loss. Conversely, data points with consistently small losses have small or even zero multipliers, allowing them to be largely ignored during optimization. Thus, a given primal update is influenced by the "instantaneous" incentive to satisfy a constraint, reflected in the loss gradient, and the "historical difficulty" of satisfying the constraint, captured in the multiplier. In §5.3, we show how hard samples—such as mislabeled ones—tend to yield large multipliers.

Functional regularization. These optimization dynamics do not aim to satisfy the constraints beyond the prescribed level ϵ. Once a constraint is strictly satisfied, the dual updates reduce the corresponding multiplier, discouraging the primal updates from further minimizing the loss for the corresponding sample.

However, satisfying the constraint for some samples may require achieving a loss tighter than ϵ on others. Additionally, primal-dual methods can overshoot into the interior of the feasible set due to a "delay" between initially meeting the constraint and sufficiently reducing the multiplier to relieve pressure on loss reduction. This overshoot can be mitigated by using PI controllers to update the multipliers instead of relying on gradient ascent (Stooke et al., 2020; Sohrabi et al., 2024).

Infeasible problems. When applying GDA to infeasible problems, the multipliers associated with unsatisfiable constraints will increase indefinitely. This can potentially lead to numerical instability, disrupting optimization. However, this issue does not arise in our proposed method for solving **RFL** problems (§3.1).

The cost of GDA on FL. Since the primal update direction $\nabla_{\boldsymbol{\theta}} \mathfrak{L}_{\text{FL}}$ is a linear combination of the per-datapoint loss gradients $\nabla_{\boldsymbol{\theta}} g_i(\boldsymbol{\theta})$, it can be computed efficiently using automatic differentiation, without needing to store each gradient individually. This makes its computation as efficient as that of the **ERM** loss gradient. Therefore, applying (mini-batch) gradient descent-ascent on $\mathfrak{L}_{\text{FL}}$ is as efficient as performing (mini-batch) gradient descent on the **ERM** loss up to the cost of storing and updating the multipliers. This overhead is negligible when the $\dim(\boldsymbol{\theta})$ is much larger than n.

Practical remarks. More advanced techniques than GDA are often used for solving Lagrangian min-max problems. Standard deep learning optimizers like Adam (Kingma & Ba, 2015) can be used for the primal updates, while recent work suggests using PI control to enhance the optimization dynamics of the multipliers (Stooke et al., 2020; Sohrabi et al., 2024).

We favor alternating GDA over simultaneous GDA as it offers better convergence guarantees under primal strong convexity (Zhang et al., 2022) without incurring additional computational cost (Sohrabi et al., 2024). This approach is particularly relevant for **FL**, where the initial primal update has no effect due to the initialization $\boldsymbol{\lambda}_0 = \mathbf{0}$. Thus, it is essential to first "warm up" $\boldsymbol{\lambda}$, which is naturally achieved by using an alternating scheme that updates the dual variables first.

Machine learning tasks typically involve computations over mini-batches of data. **FL** supports these by: ① sampling a mini-batch and computing the losses for each sample, ② updating the multipliers for the observed samples, and ③ performing a stochastic gradient step for the model. This results in stochastic updates on $\boldsymbol{\theta}$ and coordinate-wise updates on $\boldsymbol{\lambda}$.

3 RESILIENT FEASIBLE LEARNING

The potential misspecification of $\textbf{FL}(\boldsymbol{\epsilon})$ problems can be addressed by relaxing the constraints using *slack variables*, denoted by $\boldsymbol{u}$.[4] Given $\alpha > 0$, we consider the following constrained optimization problem:

$$\min_{\boldsymbol{\theta} \in \Theta, \boldsymbol{u} \geq \mathbf{0}} \frac{\alpha}{2} ||\boldsymbol{u}||^2 \quad \text{s.t} \quad \boldsymbol{g}(\boldsymbol{\theta}) \leq \boldsymbol{\epsilon} + \boldsymbol{u}. \qquad (\textbf{RFL}(\boldsymbol{\epsilon}, \alpha))$$

We call this approach Resilient Feasible Learning (**RFL**) due to its robustness to problem misspecification. If the original **FL** problem is feasible, the corresponding **RFL** problem is equivalent, as the optimal relaxation $\boldsymbol{u}$ will be zero. Crucially, **RFL** *guarantees the existence of a feasible solution*, even when the original **FL** problem is infeasible. We generally favor **RFL** over **FL** in practice since it alleviates the challenge of setting ϵ.

In $\textbf{RFL}(\boldsymbol{\epsilon}, \alpha)$, $u_i > 0$ represents a strict relaxation of the i-th constraint, while $u_i = 0$ indicates that it remains unchanged. The cost of relaxing constraints depends on the norm of the slack variables. Although other norms could be used, we focus on the L_2-norm due to its algorithmic advantages, as demonstrated in Prop. 1.

While one might consider setting $\epsilon = 0$–thus allowing **RFL** to determine the tightest possible loss requirements through the slacks–maintaining a positive ϵ enables regularization, as in **FL**. This prevents the model from overly minimizing per-sample losses, even if *some* data

[4] Similar to soft-margin support vector machines.

points are allowed to violate the constraints. For example, ϵ could demand marginally correct predictions, while $\boldsymbol{u}$ allows mislabeled samples to be misclassified.

Although the parameter α does not change the optimal solution $\boldsymbol{\theta}^*, \boldsymbol{u}^*$ of RFL(ϵ, α), it can affect the dynamics of the algorithm used to solve it. We present a formulation with an arbitrary α to allow flexibility in tuning these dynamics. Its effect is explored in Table 2.

3.1 Solving RFL Problems

As with **FL** problems, we use the Lagrangian approach to solve **RFL** problems. The min-max Lagrangian game associated with RFL(ϵ, α) is given by:

$$\min_{\boldsymbol{\theta}\in\Theta, \boldsymbol{u}\geq \mathbf{0}} \max_{\boldsymbol{\lambda}\geq \mathbf{0}} \underbrace{\frac{\alpha}{2}||\boldsymbol{u}||^2 + \boldsymbol{\lambda}^\top(\boldsymbol{g}(\boldsymbol{\theta}) - \boldsymbol{\epsilon} - \boldsymbol{u})}_{\mathfrak{L}_{\text{RFL}}(\boldsymbol{\theta}, \boldsymbol{u}, \boldsymbol{\lambda})} \quad (3)$$

We now transform Eq. (3) to a problem without slacks.

Proposition 1. *[Proof] For every $\boldsymbol{\theta} \in \Theta$, the following strong duality condition holds:*

$$\min_{\boldsymbol{u}\geq \mathbf{0}} \max_{\boldsymbol{\lambda}\geq \mathbf{0}} \mathfrak{L}_{\text{RFL}}(\boldsymbol{\theta}, \boldsymbol{u}, \boldsymbol{\lambda}) = \max_{\boldsymbol{\lambda}\geq \mathbf{0}} \min_{\boldsymbol{u}\geq \mathbf{0}} \mathfrak{L}_{\text{RFL}}(\boldsymbol{\theta}, \boldsymbol{u}, \boldsymbol{\lambda}) \quad (4)$$

$$= \max_{\boldsymbol{\lambda}\geq \mathbf{0}} \underbrace{\boldsymbol{\lambda}^\top(\boldsymbol{g}(\boldsymbol{\theta}) - \boldsymbol{\epsilon})}_{\mathfrak{L}_{\text{FL}}(\boldsymbol{\theta}, \boldsymbol{\lambda})} - \frac{||\boldsymbol{\lambda}||^2}{2\alpha} \quad (5)$$

As a consequence of Proposition 1, the Lagrangian problem for **RFL** in Eq. (3) can be solved via a quadratically-regularized version of the **FL** Lagrangian:

$$\min_{\boldsymbol{\theta}\in\Theta} \max_{\boldsymbol{\lambda}\geq \mathbf{0}} \mathfrak{L}_\alpha(\boldsymbol{\theta}, \boldsymbol{\lambda}) \triangleq \mathfrak{L}_{\text{FL}}(\boldsymbol{\theta}, \boldsymbol{\lambda}) - \frac{1}{2\alpha}||\boldsymbol{\lambda}||^2. \quad (6)$$

$\mathfrak{L}_\alpha$ is strongly concave on $\boldsymbol{\lambda}$, implying that for a fixed $\boldsymbol{\theta}$, the inner maximization has a unique solution $\boldsymbol{\lambda}^*$ (whereas **FL** may yield an unbounded inner problem).

This formulation is advantageous, as gradient descent-ascent offers convergence guarantees for non-convex, strongly-concave min-max problems (Lin et al., 2020). In particular, strong convexity of the function $\boldsymbol{g}$ yields a linear convergence rate (Chen & Rockafellar, 1997).

Alternating GDA updates on $\mathfrak{L}_\alpha$ yield similar updates to those of primal-dual **FL** (Eq. (2)). The primal update direction remains the same: a linear combination of the per-sample loss gradients, weighted by the multipliers. The dual update includes a weight decay of $1/\alpha$, which "discounts" historical violations, resulting in different dynamics. For example, this prevents the multipliers for unsatisfiable constraints from growing indefinitely.

By analytically solving the inner maximization problem in Eq. (6), we recover the following result:

Proposition 2. *[Proof] For every $\boldsymbol{\theta} \in \Theta$, we have:*

$$\min_{\substack{\boldsymbol{\theta}\in\Theta \\ \boldsymbol{u}\geq \mathbf{0}}} \max_{\boldsymbol{\lambda}\geq \mathbf{0}} \mathfrak{L}_{\text{RFL}}(\boldsymbol{\theta}, \boldsymbol{u}, \boldsymbol{\lambda}) = \min_{\boldsymbol{\theta}\in\Theta} \frac{\alpha}{2} \left\| [\boldsymbol{g}(\boldsymbol{\theta}) - \boldsymbol{\epsilon}]_+ \right\|^2 \quad (7)$$

Therefore, RFL(ϵ, α) can be solved via either ① a non-convex, strongly-concave min-max problem (Eq. (6)), or ② a non-convex **ERM**-style minimization problem with a clamped-and-squared loss (**CSERM**, Eq. (7)). While these two problem formulations are equivalent, we favor the primal-dual approach due to its optimization dynamics, which are explored in §5.2.

4 RELATED WORK

Learning paradigms. **FL** stands in contrast to the standard Empirical Risk Minimization (**ERM**):

$$\min_{\boldsymbol{\theta}\in\Theta} L_{\textbf{ERM}}(\boldsymbol{\theta}) \triangleq \frac{1}{n}\mathbf{1}^\top \boldsymbol{g}(\boldsymbol{\theta}), \quad (\textbf{ERM})$$

which views the learning problem as "choosing from the given set of functions the one which approximates best the supervisor's response" (Vapnik, 1991, p.2). **ERM** operationalizes the notion of "best approximation" through the *average* loss across the training set.

There is a fundamental difference between the goals of the **FL** and **ERM** problems. To illustrate this, consider the case of recommender systems used by streaming or social media platforms. Service providers often prioritize metrics like average click-through rates or watch-time to measure overall system success and user engagement. However, individual users are primarily concerned with how well the recommendations align with *their* personal tastes and preferences. A system that performs well on average might still fail individual users by consistently suggesting irrelevant or inadequate content. **ERM** inherently allows for trade-offs between training samples, allowing models to perform poorly on certain samples as long as they compensate by performing exceptionally well on others.

In contrast to robust (**Rawlsian**) approaches (Lahoti et al., 2020), which minimize the worst-case risk:

$$\min_{\boldsymbol{\theta}\in\Theta} \max_{i\in\{1,\dots,n\}} g_i(\boldsymbol{\theta}), \quad (\textbf{Rawlsian})$$

FL only requires that the upper bound in the per-sample loss is satisfied[5]. Thus, **FL** does not prefer one model over another as long as both satisfy the constraints.

The pursuit of minimizing the average loss in **ERM** or the maximum loss in the **Rawlsian** approach can lead to models that overfit the training data or become overly confident. This excessive reduction in losses can harm generalization, motivating the use of regularization techniques. Unlike traditional methods, which promote parsimony using surrogate criteria like L_p-norms, **FL** explicitly establishes an upper bound *on the loss itself* through the constraint level ϵ.

[5]The **Rawlsian** approach "finds" the tightest ϵ that would ensure feasibility in a corresponding **FL** problem.

Learning through constraints. The use of data samples to constrain the parameter space has long been applied in generative modeling and parametric estimation, dating back at least to the Maximum Entropy principle (Jaynes, 1957). However, moment-constrained approaches like rate-constrained classification (Tong et al., 2020) rely on aggregate statistics of the samples. In contrast, **FL** considers constraints on the samples.

SVMs. Hard-margin Support Vector Machines (**SVMs**) find classifiers that ① correctly classify all points and ② do so with the maximum possible margin. In contrast, **FL** does not explicitly seek the maximum margin. However, it can be interpreted as finding a classifier that guarantees a certain confidence on the surrogate loss ℓ, as determined by ϵ. Moreover, our primal-dual approach allows **FL** to find meaningful solutions even when the problem is infeasible, whereas hard-margin **SVMs** are ineffective for non-separable data.

Constrained ERM. The standard approach in constrained machine learning typically adds constraints to standard training objectives (such as the average loss) to enforce requirements such as fairness (Cotter et al., 2019), sparsity (Gallego-Posada et al., 2022), or safety (Stooke et al., 2020). Thus, constraints are often used to encourage behaviors that drift away from the main learning objective. In contrast, **FL** considers constraints as the primary driving force for learning.

Resilience. Previous work has addressed constraint-level misspecification in constrained **ERM**. Hounie et al. (2024) propose Resilient Constraint Learning, which, like **RFL**, uses slack variables to relax constraints. Their method relaxes constraints based on the *sensitivity* of the objective function to each constraint (i.e., the potential improvement in the objective given a small relaxation). In the case of **FL**, however, the objective is constant, so there is no trade-off between a learning objective and the imposed constraints to consider.

Clamped losses. Using a thresholded loss for regression—where errors below a certain threshold are not penalized—dates back at least to Vapnik (1998, Chapter 6). However, this approach still allows the model to trade off errors between data points that exceed the threshold. In contrast, **FL** does not permit infeasible solutions, thereby preventing such trade-offs. On the other hand, **RFL** is equivalent to **ERM** using a thresholded (and also squared) loss (see Proposition 2). Unlike standard thresholded loss approaches, **RFL** does not regularize the model's complexity.

5 EXPERIMENTS

In this section, we empirically evaluate the Feasible Learning framework, demonstrating that **FL** and **RFL** present a compelling alternative to the widely used **ERM** framework. We demonstrate that models trained via **FL** can learn (§5.1). We also explore the advantages of **RFL** over **FL** (§5.2) and analyze their loss distribution profiles (§5.3). See Appendix B for details on our experimental setup. For comprehensive results, see Appendix C.

Tasks. We train ResNet-18 models (He et al., 2016) for CIFAR10 (Krizhevsky, 2009) classification and for UTKFace (Zhang et al., 2017b) age regression. We also fine-tune an 8 billion parameter Llama-3.1 model (Llama Team, 2024) on a cleaned version of Intel Orca DPO pairs dataset.[6] We use Direct Preference Optimization (DPO) (Rafailov et al., 2024). Finally, we train a Multi-Layer Perceptron for Two-Moons classification. Table 3 in App. B lists each task's training set size, which corresponds to the number of constraints.

As the constraint level ϵ is expressed in terms of the loss, it can be interpreted for each task. **Classification**: we bound the cross-entropy loss, which translates into a lower bound on the predicted probability for the true class.[7] **Regression**: we bound the Squared Error (SE), which corresponds to the difference in years between the predicted and true ages. We normalize the ages to have zero mean and unit variance. **Preference Alignment (DPO)**: The DPO loss constraint is expressed as $\sigma(r(y^+) - r(y^-)) \geq \exp(-\epsilon)$, where y^+ and y^- represent a pair of preferred and dispreferred completions, respectively, r is an implicit reward model defined via log-likelihood ratios, and σ is a sigmoid function.

Methods. We train models via ① **ERM**, ② **CSERM**: Clamped-and-Squared **ERM** (Eq. (7)), ③ **FL**: Feasible Learning, and ④ **RFL**: Resilient Feasible Learning.

Experimental uncertainty. Unless stated otherwise, all reported metrics are averaged over 5 seeds.

Software & Hardware. Our implementations use PyTorch (Paszke et al., 2019) and the Cooper library for constrained optimization (Gallego-Posada et al., 2024). Experiments are run on NVIDIA `L40S` GPUs.

5.1 Can We Learn with Feasible Learning?

We begin by evaluating models trained with **FL** using **ERM**'s primary success criterion: average performance. Despite **FL** tackling a different problem—and irrespective of its effectiveness in solving it—we assess whether **FL** still succeeds in the standard learning task.

Table 1 presents results for a CIFAR10 classification task. We include **FL** and **RFL** under two requirements: $\epsilon = 0$, where the model is required to assign a probabil-

[6] https://huggingface.co/datasets/argilla/distilabel-intel-orca-dpo-pairs

[7] $\hat{p}_{\text{true}} \geq \exp(-\epsilon)$, where $\hat{p}_{\text{true}}$ is the model's predicted probability for the correct label.

Table 1: Final performance for CIFAR10. **FL and RFL achieve comparable average losses and accuracies to ERM, on both the training and test sets.**

Method	ϵ	Train		Test	
		CE Loss	Acc.	CE Loss	Acc.
ERM		0.00	1.00	0.30	0.93
FL	0.51	0.01	1.00	0.34	0.92
RFL	0.51	0.01	1.00	0.35	0.92
CSERM	0.51	0.34	0.98	0.52	0.88
FL	0.00	0.00	1.00	0.33	0.93
RFL	0.00	0.00	1.00	0.34	0.93
CSERM	0.00	0.07	0.99	0.42	0.87

ity of 1 to the correct label, matching ERM's solution set assuming interpolation is possible; and $\epsilon = 0.51$, where a true class probability of 0.6 is required, ensuring correct classification with a small margin.

These results demonstrate that FL and RFL can effectively learn classifiers with only a slight degradation in average performance compared to ERM on both the training and test sets. However, FL and RFL may offer advantages in tail behavior and robustness (§5.3), making this trade-off appealing for certain applications. We observe this trend across all tasks (see Appendix C).

Optimization budget. Notably, FL and RFL achieve comparable performance to ERM *within the same training budget* of 200 epochs. However, it is important to note that poor choices of the dual step size can cause FL and RFL to converge more slowly if chosen too small, or experience degraded performance if set too high.

Robustness. We found that, despite introducing a new hyper-parameter with the dual step size, FL and RFL are ① similarly robust to the choice of the primal step size as ERM, and ② fairly robust to the choice of the dual step size, achieving good performance across multiple orders of magnitude (see Appendix C).

5.2 How Does Resilience Help?

Table 2 presents an ablation study on the choice of RFL's α for the UTKFace age regression task. We select $\epsilon = 0.0$—which is unattainable due to the presence of duplicated samples in the dataset with different labels—to emphasize the benefits of resilience in providing flexibility to satisfy the constraints.

FL's constraints are too restrictive, leading to poor average and maximum performance, significantly worse than ERM. We attribute this to its optimization dynamics, which cause some multipliers to grow indefinitely, destabilizing the optimization process. In contrast, RFL can relax these requirements and achieve performance comparable to ERM. In particular, RFL ($\alpha = 10^{-3}$) outperforms ERM in average train and test errors. Moreover, although RFL relaxes the constraints, potentially allowing for larger maximum errors than FL, it achieves smaller Max SE's, further indicating a failure of FL.

Table 2: Final performance for UTKFace age regression. FL's constraints are too restrictive, resulting in worse performance than ERM. **In contrast, RFL can outperform ERM with appropriate α choices.** $\epsilon = 0.0$. Max SE stands for Maximum per-sample Square Error.

Method	Train		Test	
	MSE	Max SE	MSE	Max SE
ERM	0.03	0.37	0.42	11.11
FL ("$\alpha = \infty$")	0.08	0.87	0.47	12.39
RFL ($\alpha = 1$)	0.05	0.66	0.44	11.14
RFL ($\alpha = 10^{-1}$)	0.05	0.51	0.44	10.91
RFL ($\alpha = 10^{-2}$)	0.02	0.46	0.42	11.33
RFL ($\alpha = 10^{-3}$)	**0.01**	**0.30**	0.38	11.42
RFL ($\alpha = 10^{-4}$)	0.06	2.58	**0.37**	**10.78**

Moreover, we observe that while certain values of α may yield better performance, a wide range of values spanning multiple orders of magnitude can still result in strong performance. In other words, RFL demonstrates relatively low sensitivity to α.

A trade-off in using RFL is that, even though the choice of ϵ becomes less critical, we now need to select an appropriate α. Our findings across various choices of ϵ and tasks indicate that finding suitable α values may require extensive tuning (see Appendix C).

5.3 How do FL Solutions Compare to ERM?

Concentrated loss distribution. Figure 2 presents the Cumulative Density Function (CDF) and Conditional Value at Risk (CVaR) for the loss of test samples in the DPO task. For small losses, ERM's CDF lies above FL's, indicating that ERM has a higher proportion of low-loss samples. Conversely, for larger losses, FL's CDF rises above ERM's, showing that FL has fewer samples with very high losses. This suggests that while ERM performs better on "easy" samples, FL ensures more consistent performance, especially in the tail.

Furthermore, FL consistently achieves lower CVaR values compared to ERM, meaning that the average loss for samples with high losses is lower for FL across all loss percentiles. This highlights FL's sample-centric nature: outlier samples are less severely impacted than in ERM. This property makes FL particularly valuable in applications where consistent performance across all data points is critical. We observed similar behavior for the training set, and across all tasks (see Appendix C).

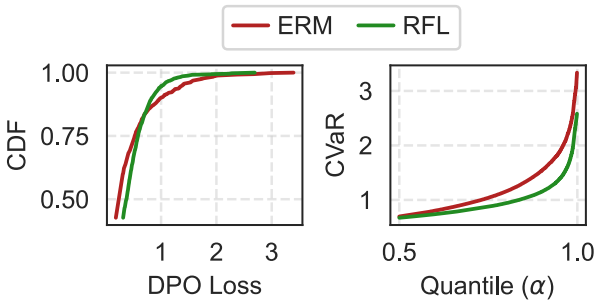


Figure 2: Empirical distribution of validation per-sample DPO losses on a fine-tuned Llama3-8B. **Left**: The empirical Cumulative Density Function (CDF). **Right**: The empirical Conditional Value at Risk (CVaR). **FL results in fewer samples with very high losses and a lower average loss for those samples.** The CVaR represents the average loss for samples exceeding each quantile of the loss distribution.

Multiplier informativity. Despite the absence of an objective, which precludes classical sensitivity or shadow price interpretations of Lagrange multipliers, multipliers can still provide insights into the difficulty of satisfying the corresponding constraint.

Figure 3 illustrates the dual variable informativity in a Two-Moons classification task. Samples near the decision boundary, which are harder to classify, have larger multiplier values at the end of training. Hence, similar to support vectors in **SVMs**, these samples play a more significant role in shaping the classifier, as their higher multipliers give them greater influence in the primal updates. In contrast, points far from the boundary that are easy to classify have near-zero multipliers and contribute less to the resulting classifier.

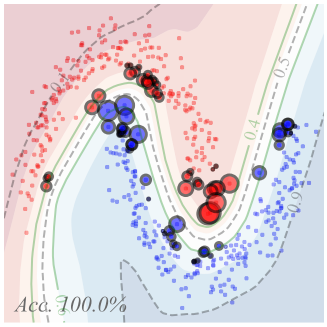


Figure 3: **FL** for a two-dimensional classification task. The marker size of each datapoint is proportional to its corresponding multiplier value at the end of training. **Points near the decision boundary have large multipliers, while those farther away have near-zero ones.** The contours correspond to the level curves of the predicted probabilities.

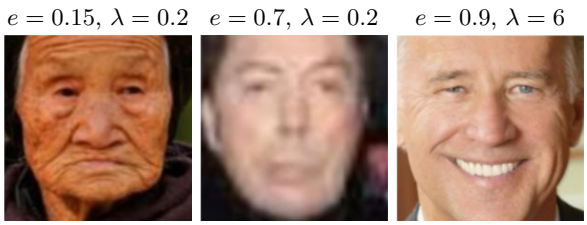


Figure 4: Training samples from UTKFace with large multiplier values at the end of training (top 20), but with low errors. **These samples are memorized by the model, but are difficult to fit**: (a) a sample with an unusually large age, (b) a blurred image, and (c) a repeated subject yet with different labels.

Figure 4 shows the loss and multiplier values for some challenging samples in UTKFace age classification. This figure demonstrates that some examples can achieve a small loss yet have a large multiplier—indicating that they were difficult to fit, but the model ultimately managed to do so. The multipliers can help identify outliers or consistently challenging samples that the final loss value alone may not reveal.

Across various tasks, we observe that many samples have zero-valued multipliers towards the end of training, particularly when using **RFL** with small α values (see Appendix C). This implies that the primal updates eventually rely on only a small subset of the data. Beyond simply identifying "easy" samples, this observation could be leveraged to improve computational efficiency by pruning these samples from the dataset.

6 CONCLUSION

In this work, we introduce Feasible Learning, a novel learning paradigm that frames learning as a constraint satisfaction problem. We show that **FL** problems can be solved using a primal-dual approach, which is as computationally efficient as **ERM** with gradient descent and offers comparable hyperparameter robustness.

FL aligns with the growing demand for user-specific performance as machine learning models are increasingly applied in personalized areas like recommender systems and healthcare. Unlike **ERM**, **FL** directly supports meeting potential regulatory or industry standards that demand a certain level of performance for all users.

We demonstrate that models can learn through **FL**, even when using tools originally developed for **ERM**, such as modern deep learning architectures and mini-batch optimization techniques. Developing algorithmic tools specifically tailored to learning via **FL** is an important direction of future research.

We show that **FL** yields a less heavy-tailed loss distribution than **ERM**. We also highlight the informativity of the Lagrange multipliers, as they correlate with the difficulty of fitting each sample. Other potential benefits of **FL**—which could be explored in future work—may include fairness, due to its sample-centric nature, and calibration, as it does not demand zero training loss.

Acknowledgements

This research was partially supported by an IVADO PhD Excellence Scholarship, the Canada CIFAR AI Chair program (Mila), the NSERC Discovery Grant RGPIN2017-06936, and by Samsung Electronics Co., Ldt. Simon Lacoste-Julien is a CIFAR Associate Fellow in the Learning in Machines & Brains program.

This research was enabled in part by compute resources, software, and technical help provided by Mila.

We thank Pedram Khorsandi, Mansi Rankawat, Motahareh Sohrabi, and Rohan Sukumaran for their feedback on the paper.

References

Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A Closer Look at Memorization in Deep Networks. In *ICML*, 2017. (Cit. on p. 1)

K.J. Arrow, L. Hurwicz, and H. Uzawa. *Studies in Linear and Non-linear Programming*. Stanford University Press, 1958. (Cit. on p. 3)

Vineeth Balasubramanian, Shen-Shyang Ho, and Vladimir Vovk. *Conformal Prediction for Reliable Machine Learning: Theory, Adaptations and Applications*. Newnes, 2014. (Cit. on p. 1)

James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, et al. Improving Image Generation with Better Captions. https://cdn.openai.com/papers/dall-e-3.pdf, 2023. (Cit. on p. 1)

George HG Chen and R Tyrrell Rockafellar. Convergence Rates in Forward–Backward Splitting. *SIAM Journal on Optimization*, 1997. (Cit. on p. 5)

Andrew Cotter, Heinrich Jiang, Maya R Gupta, Serena Wang, Taman Narayan, Seungil You, and Karthik Sridharan. Optimization with Non-Differentiable Constraints with Applications to Fairness, Recall, Churn, and Other Goals. *JMLR*, 2019. (Cit. on p. 6)

Marguerite Frank and Philip Wolfe. An Algorithm for Quadratic Programming. *Naval Research Logistics Quarterly*, 1956. (Cit. on p. 3)

Jose Gallego-Posada, Juan Ramirez, Akram Erraqabi, Yoshua Bengio, and Simon Lacoste-Julien. Controlled Sparsity via Constrained Optimization or: *How I Learned to Stop Tuning Penalties and Love Constraints*. In *NeurIPS*, 2022. (Cit. on p. 6)

Jose Gallego-Posada, Juan Ramirez, Meraj Hashemizadeh, and Simon Lacoste-Julien. Cooper: A Library for Constrained Optimization in Deep Learning. https://github.com/cooper-org/cooper, 2024. (Cit. on p. 6, 13)

Alan A Goldstein. Convex Programming in Hilbert Space. *University of Washington*, 1964. (Cit. on p. 3)

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned Stochastic Tensor Optimization. In *ICML*, 2018. (Cit. on p. 1)

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016. (Cit. on p. 6, 14)

Ignacio Hounie, Alejandro Ribeiro, and Luiz FO Chamon. Resilient Constrained Learning. In *NeurIPS*, 2024. (Cit. on p. 6)

E. T. Jaynes. Information Theory and Statistical Mechanics. *Physical Review*, 1957. (Cit. on p. 6)

Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015. (Cit. on p. 1, 4)

Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical report, University of Toronto, Toronto, Ontario, 2009. (Cit. on p. 6, 14)

P. Lahoti, A. Beutel, J. Chen, K. Lee, F. Prost, N. Thain, X. Wang, and E. Chi. Fairness Without Demographics Through Adversarially Reweighted Learning. In *NeurIPS*, 2020. (Cit. on p. 1, 5)

Tianyi Lin, Chi Jin, and Michael Jordan. On Gradient Descent Ascent for Nonconvex-Concave Minimax Problems. In *ICML*, 2020. (Cit. on p. 5)

AI @ Meta Llama Team. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*, 2024. (Cit. on p. 1, 6, 14)

Aleksander Mądry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. *arXiv preprint arXiv:1706.06083*, 2017. (Cit. on p. 1)

OpenAI. GPT-4 Technical Report. *arXiv:2303.08774*, 2023. (Cit. on p. 1)

Adam Paszke et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, 2019. (Cit. on p. 6, 13)

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. In *NeurIPS*, 2024. (Cit. on p. 6, 14)

Motahareh Sohrabi, Juan Ramirez, Tianyue H. Zhang, Simon Lacoste-Julien, and Jose Gallego-Posada. On PI Controllers for Updating Lagrange Multipliers in Constrained Optimization. In *ICML*, 2024. (Cit. on p. 4)

Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The Implicit Bias of Gradient Descent on Separable Data. *JMLR*, 2018. (Cit. on p. 1)

Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive Safety in Reinforcement Learning by PID Lagrangian Methods. In *ICML*, 2020. (Cit. on p. 4, 6)

Xin Tong, Lucy Xia, Jiacheng Wang, and Yang Feng. Neyman-pearson classification: parametrics and sample size requirement. *JMLR*, 2020. (Cit. on p. 6)

V. Vapnik. Principles of Risk Minimization for Learning Theory. In *NeurIPS*, 1991. (Cit. on p. 5)

Vladimir Naumovich Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998. (Cit. on p. 6)

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding Deep Learning Requires Rethinking Generalization. In *ICLR*, 2017a. (Cit. on p. 1)

Guodong Zhang, Yuanhao Wang, Laurent Lessard, and Roger B Grosse. Near-optimal Local Convergence of Alternating Gradient Descent-Ascent for Minimax Optimization. In *AISTATS*, 2022. (Cit. on p. 3, 4)

Zhifei Zhang, Yang Song, and Hairong Qi. Age Progression/Regression by Conditional Adversarial Autoencoder. In *CVPR*, 2017b. (Cit. on p. 6, 14)

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. **Yes.** Algorithms: see §2.1 for **FL** and §3.1 for **RFL**.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. **Yes.** See *The cost of GDA on* ***FL*** in §2.1.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. **Yes.** See anonymized source code in the supplementary material.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. **Yes.** See Propositions 1 and 2.
   (b) Complete proofs of all theoretical results. **Yes.** See Appendix A.
   (c) Clear explanations of any assumptions. **Yes.**

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). **Yes.** See the `scripts` folder in the code.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). **Yes.** See Appendix B.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). **Yes.** See *Experimental uncertainty* in §5.
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). **Yes.** See *Software & Hardware* in §5.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. **Yes.** See *Tasks* in §5.
   (b) The license information of the assets, if applicable. **Not Applicable.**
   (c) New assets either in the supplemental material or as a URL, if applicable. **Yes.** We include our code.
   (d) Information about consent from data providers/curators. **Not Applicable.**
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. **Not Applicable.**

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. **Not Applicable.**

(b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. **Not Applicable.**

(c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. **Not Applicable.**

Appendix

Table of Contents

A PROOFS

Proof of Proposition 1. For every $\theta \in \Theta$, $\mathfrak{L}_{\text{RFL}}$ is convex in $\boldsymbol{u}$ and concave in $\boldsymbol{\lambda}$. Moreover, we have that $\mathfrak{L}_{\text{RFL}}$ is 0-coercive in $\boldsymbol{u}$ for any $\boldsymbol{\lambda}$, that is, $\mathfrak{L}_{\text{RFL}}(\theta, \boldsymbol{u}, \boldsymbol{\lambda}) \to \infty$ as $\|\boldsymbol{u}\| \to \infty$; and also there exists some fixed $\boldsymbol{u}(\theta)$ (big enough depending on θ) such that $\mathfrak{L}_{\text{RFL}}(\theta, \boldsymbol{u}(\theta), \boldsymbol{\lambda}) \to -\infty$ as $\|\boldsymbol{\lambda}\| \to \infty$. We can thus apply the existence of a saddle-point theorem from (Hiriart-Urruty & Lemaréchal, 1996, Theorem 4.3.1) to obtain:

$$\min_{\boldsymbol{u}\geq\mathbf{0}} \max_{\boldsymbol{\lambda}\geq\mathbf{0}} \mathfrak{L}_{\text{RFL}}(\boldsymbol{\theta}, \boldsymbol{u}, \boldsymbol{\lambda}) = \max_{\boldsymbol{\lambda}\geq\mathbf{0}} \min_{\boldsymbol{u}\geq\mathbf{0}} \mathfrak{L}_{\text{RFL}}(\boldsymbol{\theta}, \boldsymbol{u}, \boldsymbol{\lambda}). \tag{8}$$

Moreover, the first order optimality condition for the inner minimization $\min_{\boldsymbol{u}\geq\mathbf{0}} \mathfrak{L}_{\text{RFL}}(\boldsymbol{\theta}, \boldsymbol{u}, \boldsymbol{\lambda})$ yields $\alpha\boldsymbol{u}^* - \boldsymbol{\lambda} = \mathbf{0}$. Since $\boldsymbol{\lambda} \geq \mathbf{0}$, the first order condition is satisfied at the non-negative $\boldsymbol{u}^* = \boldsymbol{\lambda}/\alpha$. It follows that:

$$\max_{\boldsymbol{\lambda}\geq\mathbf{0}} \min_{\boldsymbol{u}\geq\mathbf{0}} \mathfrak{L}_{\text{RFL}}(\boldsymbol{\theta}, \boldsymbol{u}, \boldsymbol{\lambda}) = \max_{\boldsymbol{\lambda}\geq\mathbf{0}} \mathfrak{L}_{\text{RFL}}\left(\boldsymbol{\theta}, \frac{1}{\alpha}\boldsymbol{\lambda}, \boldsymbol{\lambda}\right) = \max_{\boldsymbol{\lambda}\geq\mathbf{0}} \boldsymbol{\lambda}^\top(\boldsymbol{g}(\boldsymbol{\theta}) - \boldsymbol{\epsilon}) - \frac{1}{2\alpha}||\boldsymbol{\lambda}||^2 \tag{9}$$

□

Proof of Proposition 2. From Proposition 1, it follows that:

$$\min_{\boldsymbol{\theta}\in\Theta} \min_{\boldsymbol{u}\geq\mathbf{0}} \max_{\boldsymbol{\lambda}\geq\mathbf{0}} \mathfrak{L}_{\text{RFL}}(\boldsymbol{\theta}, \boldsymbol{u}, \boldsymbol{\lambda}) = \min_{\boldsymbol{\theta}\in\Theta} \max_{\boldsymbol{\lambda}\geq\mathbf{0}} \min_{\boldsymbol{u}\geq\mathbf{0}} \mathfrak{L}_{\text{RFL}}(\boldsymbol{\theta}, \boldsymbol{u}, \boldsymbol{\lambda}) = \min_{\boldsymbol{\theta}\in\Theta} \max_{\boldsymbol{\lambda}\geq\mathbf{0}} \boldsymbol{\lambda}^\top(\boldsymbol{g}(\boldsymbol{\theta}) - \boldsymbol{\epsilon}) - \frac{1}{2\alpha}||\boldsymbol{\lambda}||^2 \tag{10}$$

The first order optimality condition for $\boldsymbol{\lambda}$ over the non-negative orthant is satisfied at $\boldsymbol{\lambda}^*_{\boldsymbol{\theta}} = \alpha\left[\boldsymbol{g}(\boldsymbol{\theta}) - \boldsymbol{\epsilon}\right]_+$, where $[\cdot]_+$ is an element-wise projection to $\mathbb{R}_{\geq 0}$. Substituting this value of $\boldsymbol{\lambda}$ back in Eq. (10) yields the desired result.

□

B EXPERIMENTAL DETAILS

Our implementations use PyTorch (Paszke et al., 2019) and the Cooper library for constrained optimization (Gallego-Posada et al., 2024).[8] Experiments are run on NVIDIA `L40S` GPUs. Our code is available at: `https://github.com/juan43ramirez/feasible-learning`.

B.1 Polynomial Regression

We consider the problem of fitting a polynomial $p_{\boldsymbol{a}}(x) = \sum_{j=0}^{d} a_j x_i^j$ of degree d to a dataset $\{(x_i, y_i)\}_{i=1}^n$ consisting of n samples. The **ERM** problem in this setting can be expressed as follows:

$$\min_{\boldsymbol{a}} \frac{1}{n}\sum_{i=1}^{n} \left(p_{\boldsymbol{a}}(x_i) - y_i\right)^2, \tag{11}$$

where $\boldsymbol{a} = [a_0, a_1, \ldots, a_d]$ represent the coefficients of the polynomial.

The corresponding **FL** problem is:

$$\min_{\boldsymbol{a}} 0, \quad \text{s.t.} \quad \left(p_{\boldsymbol{a}}(x_i) - y_i\right)^2 \leq \epsilon, \quad i = 1, \ldots, n. \tag{12}$$

We generate data by sampling 20 points from a cosine wave and adding Gaussian noise with a standard deviation of $\sigma = 0.2$. We fit polynomials of degrees 10, 20, and 30, which represent different model parameterization scenarios: the degree 10 polynomial is under-parameterized, and the degree 30 polynomial is over-parameterized.

Due to ill-conditioning, we solve both problems using a standard convex optimization solver: CVXPY's Splitting Conic Solver (O'Donoghue, 2021, SCS). For **ERM**, this solver resorts to a QP solver; for **FL**, note that the problem in Eq. (12) constitutes a second-order cone programming problem over $[p_{\boldsymbol{a}}(x_i) - y_i, \epsilon]$ (Boyd & Vandenberghe, 2004, §4.4.2). **FL**'s constraint value is set to one standard deviation, $\epsilon = \sigma$.

Table 3: Datasets considered throughout this work.

Dataset	Model	Train Size
CIFAR10 (Krizhevsky, 2009)	ResNet-18 (He et al., 2016)	50000
UTKFace (Zhang et al., 2017b)	ResNet-18 (He et al., 2016)	4438
Orca (Mukherjee et al., 2023)	Llama3-8B (Llama Team, 2024)	12859
Two Moons (noise = 0.1)	MLP (2, 70, 70, 2)	1000

B.2 Deep Learning Tasks

Table 3 presents the tasks (datasets and models) considered in our experiments in §5 and Appendix C. We also include the training set size for each dataset, as it corresponds to the number of constraints for each task.

UTKFace. We first perform a 70%-30% train-test split, followed by subsampling 25% of the training set. This is intended to create a simpler task, enabling the ResNet-18 to interpolate most datapoints. However, the resulting dataset still contains some duplicated samples with distinct labels (see §5.2). Additionally, we normalize the age variable to have zero mean and unit variance.

Direct Preference Optimization. Since Large Language Models (LLMs) are primarily pre-trained for next token prediction, fine-tuning their weights to align their outputs with human preferences on specific tasks is often desirable. Preferences are typically provided as an input prompt paired with two outputs, one of which is preferred.

Various techniques exist to increase the likelihood of preferred outputs (see, for example, Kaufmann et al. (2023) and references therein). In particular, Rafailov et al. (2024) propose an effective supervised approach known as Direct Preference Optimization (DPO). This method aims to maximize the log-likelihood ratio between preferred and dispreferred responses while incorporating a regularization term to penalize deviations from the pre-trained model's outputs. This regularization is standard in preference optimization to mitigate overfitting Ouyang et al. (2022), especially given the limited size of preference datasets. The DPO loss is defined as:

$$\ell_{\text{DPO}}(\boldsymbol{x}, y^+, y^-, \pi_\theta, \pi_{\text{ref}}) = \log \sigma \left(\beta \log \frac{\pi_\theta\,(y^+ \mid \boldsymbol{x})}{\pi_{\text{ref}}\ (y^+ \mid \boldsymbol{x})} - \beta \log \frac{\pi_\theta\,(y^- \mid \boldsymbol{x})}{\pi_{\text{ref}}\ (y^- \mid \boldsymbol{x})} \right),$$

where $\boldsymbol{x}$ denotes the input prompt, y^+ and y^- represent the preferred and dispreferred completions, and π_{ref} and π_θ denote the reference (pre-trained) and fine-tuned models. Rafailov et al. (2024) demonstrate that optimizing this supervised loss is equivalent to optimizing the implicit reward $\hat{r}_\theta(x, y) = \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$ within a Bradley & Terry (1952) preference model.

We fine-tune the 7 billion parameter Llama 3.1 model (Llama Team, 2024), and StableLM's Zephyr-3B[9] on the cleaned version of the Intel Orca dpo pairs dataset.[10] This synthetic preference dataset comprises 6k prompts across various domains and tasks, along with their corresponding outputs from ChatGPT and Llama2-13B. In this version of the dataset, ChatGPT is used to score outputs and the preferred choices are designated based on these scores. Because preference datasets are often small, a KL regularization that penalizes deviations from the pre-trained model's outputs is used to mitigate overfitting. In our experiments, the regularization coefficient β was set to 0.1. We use Huggingface Transformer Reinforcement Learning (trl) library.[11]

To reduce hardware requirements for fine-tuning, we apply Low-Rank Adaptation (LoRA), a popular parameter-efficient fine-tuning approach that utilizes a low-rank parametrization of weight matrix updates (Hu et al., 2022). This method decreases the number of learnable parameters, eliminating the need for gradient computation and optimizer state maintenance for most parameters. We further reduce resource requirements by quantizing the pre-trained model to four-bit precision, as proposed by Dettmers et al. (2024) and implemented in Hugging Face's Parameter-Efficient Fine-Tuning library.[12]

[8] https://github.com/cooper-org/cooper
[9] https://huggingface.co/stabilityai/stablelm-zephyr-3b
[10] https://huggingface.co/datasets/argilla/distilabel-intel-orca-dpo-pairs
[11] https://github.com/huggingface/trl
[12] https://github.com/huggingface/peft

Table 4: Primal optimization hyper-parameters. To address numerical issues, we use a step size of $\eta_\theta/10$ for the **CSERM** experiments on the UTKFace and CIFAR-10 datasets, where η_θ denotes the reported primal learning rate.

Dataset	Epochs	Batch Size	Optimizer	Step-size	Weight decay
CIFAR10	200	128	SGD	$1 \cdot 10^{-1}$	$5 \cdot 10^{-4}$
UTKFace	150	128	AdamW	$1 \cdot 10^{-4}$	0
Orca	20	16	AdamW	$5 \cdot 10^{-6}$	0
Two Moons	250	512	AdamW	$5 \cdot 10^{-4}$	0

Table 5: Dual optimization hyper-parameters for **FL** and **RFL** experiments.

Dataset	Optimizer	Step-Size
CIFAR-10	SGD	$1 \cdot 10^{-4}$
UTKFace	SGD	$1 \cdot 10^{-3}$
Orca	SGD	$1 \cdot 10^{-1}$
Two Moons	SGD	$1 \cdot 10^{-2}$

Table 6: Hyperparameter configurations for Orca DPO pairs.

LoRA α	LoRA rank	Scheduler	Warmup Steps	DPO β
1	8	Cosine	200	0.1

Hyper-parameter choices. Table 4 lists the primal hyperparameters used to train our models. We employ a cosine learning rate scheduler for CIFAR10 classification and Orca DPO experiments. Table 5 details the dual hyperparameters employed in training our **FL** and **RFL** models. Additional hyperparameters for our DPO experiments are provided in Table 6.

C ADDITIONAL EXPERIMENTS

This section complements §5. For each subsection—Appendix C.1 for CIFAR10 classification, Appendix C.2 for UTKFace age regression, and Appendix C.3 for direct preference optimization of large language models—we address questions **Q1**-**Q3** from §1. The results presented here support the findings from §5, specifically: ①: models trained with **FL** and **RFL** achieve comparable train and test performance to **ERM**, ②: **RFL** (with appropriate α choices) can succeed where **FL** fails, and ③: Feasible Learning can help shape the distribution of losses, generally producing a less heavy-tailed distribution.

Moreover, we include Figure 6 for CIFAR10 and Figure 8 for UTKFace, which show the training losses recovered by **ERM**, **FL**, **RFL**, and **CSERM** for different choices of the primal and dual learning rates. These figures demonstrate that **FL** and **RFL** are similarly robust to the choice of primal learning rate as **ERM**, while also being fairly robust to the choice of dual learning rate.

C.1 CIFAR10

Can we Learn with Feasible Learning? Table 7 presents the performance at the end of training for the CIFAR10 classification task. We report the outcomes of **ERM**, as well as **FL**, **RFL** ($\alpha = 1$), and **CSERM** under two constraint levels: $\epsilon = 0.51$, which requires the model to assign a probability of at least 0.6 to the correct class, and $\epsilon = 0$, where the model is required to assign a probability of 1 to the correct label, aligning with **ERM**'s solution set if data interpolation is possible.

We observe that **ERM** generates models that nearly interpolate the training data, achieving near-zero loss on most samples. However, the maximum loss indicates that some training points are not correctly classified by **ERM**. Note that these misclassifications are not evident in the accuracy column, as we report accuracy only up to three significant digits for a training dataset of 60000 samples.

At $\epsilon = 0$, **FL** and **RFL** achieve near-perfect training accuracy and similar average and maximum train losses compared to **ERM**. However, their test performance is slightly worse on all metrics. Moreover, **CSERM** yields

Table 7: Final performance for CIFAR10 experiments. **FL and RFL achieve comparable average losses and accuracies to ERM, on both the training and test sets.** ERM, FL, and RFL interpolate the training data in this task, achieving perfect training accuracy and a nearly zero training loss on all samples. This is an extended version of Table 1 in §5.1.

Method	ϵ	Train			Test		
		Avg. CE	Max CE	Acc.	Avg. CE	Max CE	Acc.
ERM		0.002 ± 0.000	2.369 ± 0.647	1.000 ± 0.000	0.298 ± 0.009	12.830 ± 0.443	0.932 ± 0.002
FL	0.00	0.003 ± 0.000	3.084 ± 1.064	1.000 ± 0.000	0.331 ± 0.006	15.533 ± 0.642	0.927 ± 0.001
RFL ($\alpha = 1$)	0.00	0.003 ± 0.000	3.224 ± 1.311	1.000 ± 0.000	0.337 ± 0.012	16.235 ± 1.510	0.927 ± 0.002
CSERM	0.00	0.065 ± 0.091	2.667 ± 0.926	0.988 ± 0.024	0.422 ± 0.136	12.092 ± 0.628	0.870 ± 0.048
FL	0.51	0.015 ± 0.001	4.991 ± 0.814	0.997 ± 0.000	0.342 ± 0.007	14.874 ± 1.541	0.918 ± 0.001
RFL ($\alpha = 1$)	0.51	0.014 ± 0.000	5.713 ± 0.946	0.998 ± 0.000	0.351 ± 0.009	14.139 ± 0.735	0.917 ± 0.001
CSERM	0.51	0.337 ± 0.018	1.724 ± 0.366	0.978 ± 0.010	0.522 ± 0.018	8.770 ± 0.851	0.881 ± 0.014

Table 8: Final performance for CIFAR10 classification. $\epsilon = 0$.

Method	Train			Test		
	Avg. CE	Max CE	Acc.	Avg. CE	Max CE	Acc.
ERM	0.002 ± 0.000	2.369 ± 0.579	1.000 ± 0.000	0.298 ± 0.008	12.830 ± 0.396	0.932 ± 0.002
FL ("$\alpha=\infty$")	0.003 ± 0.000	3.428 ± 1.086	1.000 ± 0.000	0.332 ± 0.007	16.091 ± 1.177	0.927 ± 0.002
RFL ($\alpha=1$)	0.003 ± 0.000	3.673 ± 1.759	1.000 ± 0.000	0.322 ± 0.008	15.013 ± 0.907	0.927 ± 0.002
RFL ($\alpha=10^{-1}$)	0.003 ± 0.000	3.157 ± 0.839	1.000 ± 0.000	0.326 ± 0.007	14.972 ± 0.740	0.928 ± 0.001
RFL ($\alpha=10^{-2}$)	0.005 ± 0.000	3.022 ± 0.787	1.000 ± 0.000	0.306 ± 0.007	12.502 ± 1.120	0.928 ± 0.002
RFL ($\alpha=10^{-3}$)	0.078 ± 0.000	2.695 ± 0.421	1.000 ± 0.000	0.309 ± 0.009	7.168 ± 0.236	0.928 ± 0.002

Table 9: Final performance for CIFAR10 classification. $\epsilon = 0.51$.

Method	Train			Test		
	Avg. CE	Max CE	Acc.	Avg. CE	Max CE	Acc.
ERM	0.002 ± 0.000	2.369 ± 0.579	1.000 ± 0.000	0.298 ± 0.008	12.830 ± 0.396	0.932 ± 0.002
FL ("$\alpha=\infty$")	0.014 ± 0.001	5.422 ± 0.565	0.998 ± 0.000	0.341 ± 0.007	14.671 ± 1.364	0.918 ± 0.003
RFL ($\alpha=1$)	0.014 ± 0.001	5.401 ± 1.052	0.998 ± 0.000	0.345 ± 0.007	13.720 ± 0.604	0.917 ± 0.001
RFL ($\alpha=10^{-1}$)	0.016 ± 0.000	4.763 ± 0.049	0.998 ± 0.000	0.346 ± 0.006	13.393 ± 0.719	0.916 ± 0.001
RFL ($\alpha=10^{-2}$)	0.064 ± 0.004	4.543 ± 0.323	0.997 ± 0.000	0.380 ± 0.006	10.734 ± 0.539	0.914 ± 0.003
RFL ($\alpha=10^{-3}$)	0.479 ± 0.002	2.013 ± 0.127	0.999 ± 0.000	0.646 ± 0.004	5.293 ± 0.238	0.925 ± 0.002

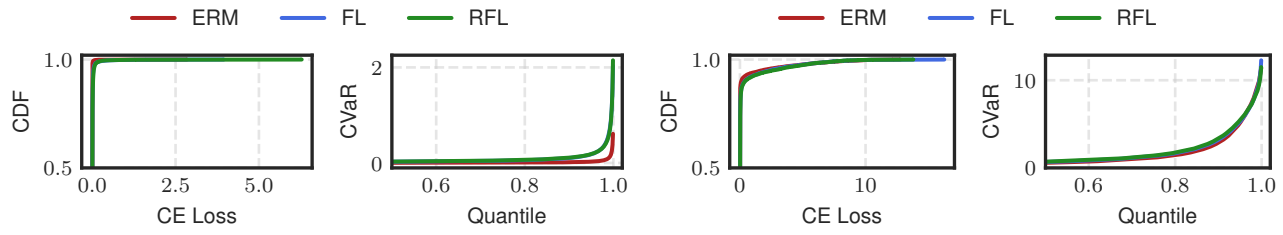
significantly worse training and test performance, which we attribute to potential numerical instabilities caused by squaring the loss functions (leading to exploding or vanishing gradients). Figure 6, presented later, highlights the difficulties in properly tuning the primal learning rate for CSERM.

At $\epsilon = 0.51$, the training performance of FL and RFL remains comparable to that of ERM; however, we observe greater degradation compared to $\epsilon = 0$. This trend extends to the test set as well. These results suggest that the constraint was too loose. A tighter, though not necessarily zero, constraint level could potentially enhance both training and test performance.

How does Resilience Help? Tables 8 and 9 present an ablation study of RFL's α value for the CIFAR10 classification task, with $\epsilon = 0$ and $\epsilon = 0.51$, respectively.

At $\epsilon = 0$, all values of α result in models with near-perfect training accuracy. Smaller α values lead to lower maximum losses, indicating a more compact loss distribution, but with increased average loss. Decreasing α also improves both average and maximum test losses.

From the RFL problem's perspective, different α values do not change the solutions. However, algorithmically, smaller α values lead to more multipliers going to zero, which reduces the pressure to overfit the training data. This reduction in pressure results in a higher average loss, as fewer points are interpolated. However, it also



(a) Train CDF and CVaR.

(b) Test CDF and CVaR.

Figure 5: Empirical distribution of per-sample cross-entropy (CE) losses on a CIFAR10 classification task. **Left**: The empirical Cumulative Density Function (CDF). **Right**: The empirical Conditional Value at Risk (CVaR). The CVaR represents the average loss for samples exceeding each quantile of the loss distribution. $\epsilon = 0.51$. **RFL**'s $\alpha = 1$.

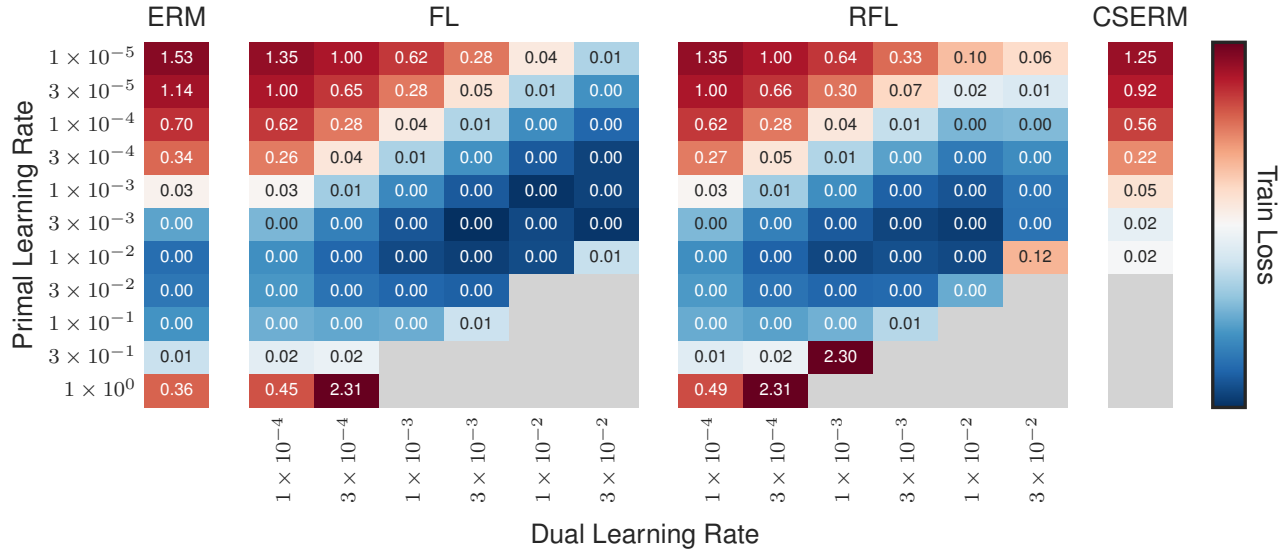


Figure 6: Average training losses at the end of training on the CIFAR10 classification task. **Similar to ERM, both FL and RFL achieve good training losses across a wide range of orders of magnitude for the primal learning rate.** Furthermore, **FL** and **RFL** demonstrate good performance for multiple choices of the dual learning rate. A grey square indicates a configuration that diverged. $\epsilon = 0.51$.

reduces the maximum losses, as overfitting the majority of the data can lead to high-confidence misclassifications of points in the tail, particularly if they are noisy or mislabeled.

At $\epsilon = 0.51$, we observe a slight degradation in both training and test performance compared to **FL** and **RFL** at $\epsilon = 0$. Since the model can effectively interpolate this dataset, we hypothesize that the constraint level of $\epsilon = 0.51$ is too loose to achieve optimal performance. However, to account for potentially misclassified hard-to-fit samples, we hypothesize that some $\epsilon > 0$ may yield better performance than both $\epsilon = 0.51$ and $\epsilon = 0$.

How do FL Solutions Compare to ERM? Figure 5 displays the CDFs and CVaRs for the train and test sets in this task, including **ERM**, **FL**, and **RFL** ($\alpha = 1$). In training, we observe slightly better CDF and CVaR curves for **ERM** compared to **FL** and **RFL**. This aligns with the results in Table 7, indicating that **ERM** outperforms these methods in both average and maximum training losses. A similar trend is observed in the test set.

Hyper-parameter Robustness. Figure 6 shows the training losses at the end of training on the CIFAR10 classification task (with $\epsilon = 0.51$), illustrating the sensitivity of **FL** and **RFL** to the choice of primal and dual learning rates. Similar to **ERM**, both **FL** and **RFL** achieve good training losses across a wide range of orders of magnitude for the primal learning rate. Furthermore, **FL** and **RFL** demonstrate good performance for multiple choices of the dual learning rate. This indicates that **FL** and **RFL** are fairly robust to these hyper-parameters, thus contributing to their applicability in practice.

Table 10: Final performance for UTKFace experiments. **FL and RFL outperform ERM in terms of MSE, on both the training and test sets.**

Method	ϵ	Train		Test	
		Avg. SE	Max SE	Avg. SE	Max SE
ERM		0.060 ± 0.017	0.438 ± 0.021	0.449 ± 0.016	10.335 ± 0.398
FL	0.00	0.026 ± 0.041	0.630 ± 0.190	0.407 ± 0.048	11.361 ± 0.719
RFL ($\alpha = 10^{-3}$)	0.00	0.006 ± 0.002	0.299 ± 0.022	0.379 ± 0.007	11.419 ± 0.589
CSERM	0.00	0.005 ± 0.000	0.343 ± 0.019	0.565 ± 0.000	15.341 ± 0.489
FL	0.02	0.025 ± 0.017	0.863 ± 0.367	0.396 ± 0.004	13.359 ± 2.010
RFL ($\alpha = 10^{-1}$)	0.02	0.011 ± 0.002	0.526 ± 0.031	0.409 ± 0.002	10.473 ± 0.877
CSERM	0.02	0.008 ± 0.001	0.344 ± 0.053	0.572 ± 0.004	15.132 ± 0.489

Table 11: Final performance for UTKFace regression. This is an extended version of Table 2 in §5.2. $\epsilon = 0$.

Method	Train		Test	
	Avg. SE	Max SE	Avg. SE	Max SE
ERM	0.026 ± 0.004	0.368 ± 0.049	0.417 ± 0.004	11.115 ± 1.231
FL ("$\alpha = \infty$")	0.083 ± 0.034	0.868 ± 0.208	0.474 ± 0.041	12.393 ± 0.520
RFL ($\alpha = 1$)	0.050 ± 0.017	0.658 ± 0.321	0.442 ± 0.022	11.139 ± 0.841
RFL ($\alpha = 10^{-1}$)	0.048 ± 0.013	0.511 ± 0.038	0.444 ± 0.017	10.912 ± 0.838
RFL ($\alpha = 10^{-2}$)	0.017 ± 0.014	0.459 ± 0.048	0.420 ± 0.030	11.327 ± 1.275
RFL ($\alpha = 10^{-3}$)	0.006 ± 0.002	0.299 ± 0.020	0.379 ± 0.007	11.419 ± 0.538
RFL ($\alpha = 10^{-4}$)	0.064 ± 0.079	2.584 ± 3.253	0.368 ± 0.012	10.780 ± 0.933

We also observe that large primal learning rate choices cause divergence in **CSERM** runs. We attribute this to the squaring of the loss which can lead to numerical issues for large losses. Additionally, for values that do not diverge, we see degraded performance compared to other approaches. While the squared loss amplifies large losses, it also results in vanishing gradients for small losses, making it harder to overfit the training samples. This situation highlights the practical advantages of **RFL** over **CSERM**, despite them being equivalent problem formulations.

C.2 UTKFace

Can we Learn with Feasible Learning? Table 10 shows the performance at the end of training for the UTKFace regression task. We observe that both **FL** and **RFL** outperform **ERM** in terms of average loss on the training and test sets, while maintaining comparable maximum losses to **ERM**. Additionally, we highlight that **RFL** achieves this performance with low variance in metrics across runs, indicating more robust dynamics than **FL**.

How does Resilience Help? Tables 11 and 12 present an ablation on **RFL**'s α value for the UTKFace regression task, with $\epsilon = 0$ and $\epsilon = 0.02$, respectively.

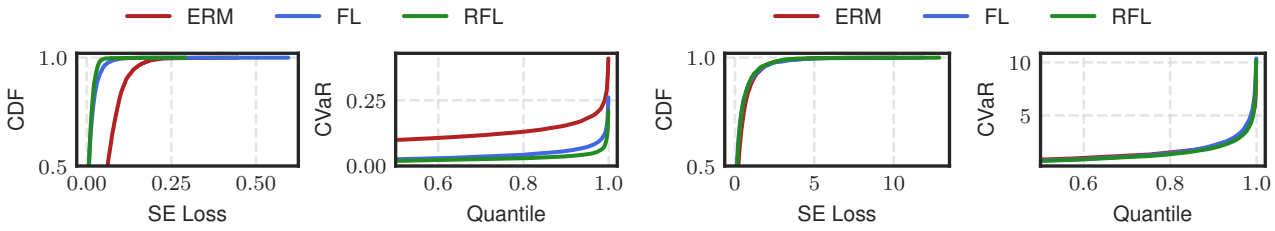
At $\epsilon = 0$, we observe that **RFL** consistently outperforms **FL** on both the training and test sets. The best training performance occurs at $\alpha = 10^{-3}$, while the best test performance is achieved at $\alpha = 10^{-4}$. We attribute **RFL**'s advantage over **FL** to the problem's infeasibility at $\epsilon = 0$ (the dataset contains duplicated samples with different labels) which leads to poor optimization dynamics in **FL**.

At $\epsilon = 0.02$, we observe that **RFL** does not necessarily outperform **FL**. In this scenario, where feasible solutions may exist, **FL** is capable of learning well-performing solutions. However, with appropriate choices of α, **RFL** can still outperform **FL** on both the training and test sets (e.g., $\alpha = 10^{-1}$ for training and $\alpha = 10^{-4}$ for testing). That said, finding these well-performing solutions requires tuning the additional hyper-parameter α.

How do FL Solutions Compare to ERM? Figure 7 shows the CDFs and CVaRs for the UTKFace classification task with $\epsilon = 0.02$. In training, both **FL** and **RFL** outperform **ERM** across the entire loss spectrum, with higher CDFs and lower CVaRs. In validation, the methods demonstrate similar performance, with overlapping curves.

Table 12: Final performance for UTKFace regression. $\epsilon = 0.02$.

Method	Train		Test	
	Avg. SE	Max SE	Avg. SE	Max SE
ERM	0.026 ± 0.004	0.368 ± 0.049	0.417 ± 0.004	11.115 ± 1.231
FL ("$\alpha = \infty$")	0.042 ± 0.004	1.255 ± 0.019	0.399 ± 0.003	15.080 ± 0.984
RFL ($\alpha = 1$)	0.017 ± 0.002	0.443 ± 0.045	0.387 ± 0.007	9.667 ± 1.000
RFL ($\alpha = 10^{-1}$)	0.011 ± 0.002	0.526 ± 0.031	0.409 ± 0.002	10.473 ± 0.877
RFL ($\alpha = 10^{-2}$)	0.023 ± 0.017	0.376 ± 0.031	0.433 ± 0.030	10.695 ± 0.724
RFL ($\alpha = 10^{-3}$)	0.054 ± 0.088	1.524 ± 2.500	0.392 ± 0.023	12.237 ± 1.577
RFL ($\alpha = 10^{-4}$)	0.078 ± 0.089	1.382 ± 2.047	0.381 ± 0.038	11.213 ± 1.674



(a) Train CDF and CVaR.

(b) Test CDF and CVaR.

Figure 7: Empirical distribution of per-sample squared error (SE) losses on a UTKFace age regression task. **Left**: The empirical Cumulative Density Function (CDF). **Right**: The empirical Conditional Value at Risk (CVaR). The CVaR represents the average loss for samples exceeding each quantile of the loss distribution. $\epsilon = 0.02$. **RFL**'s $\alpha = 10^{-1}$.

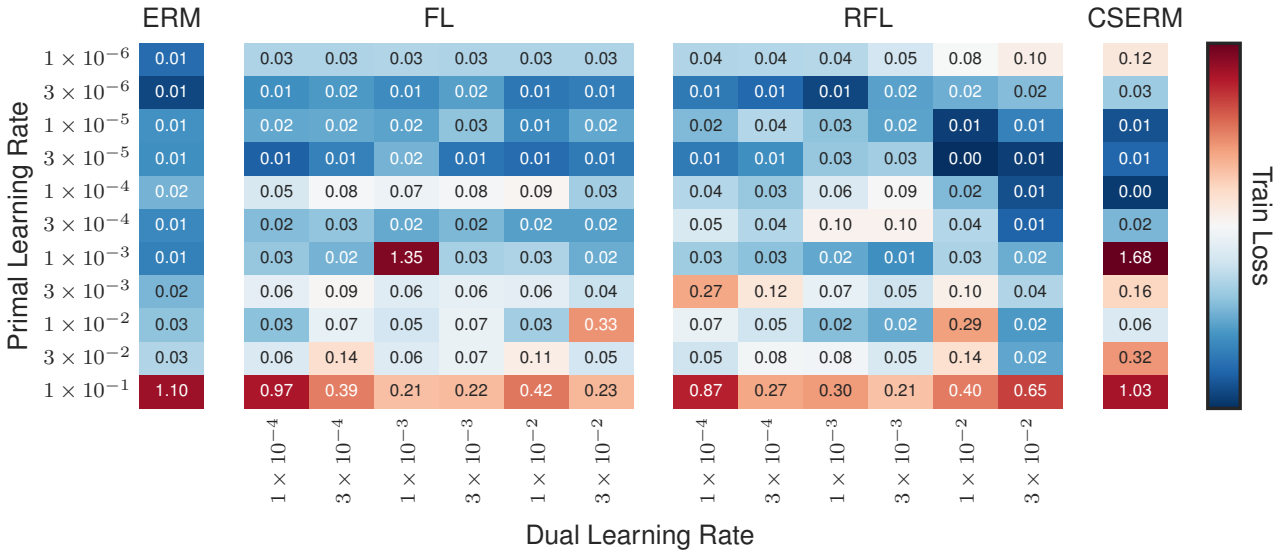
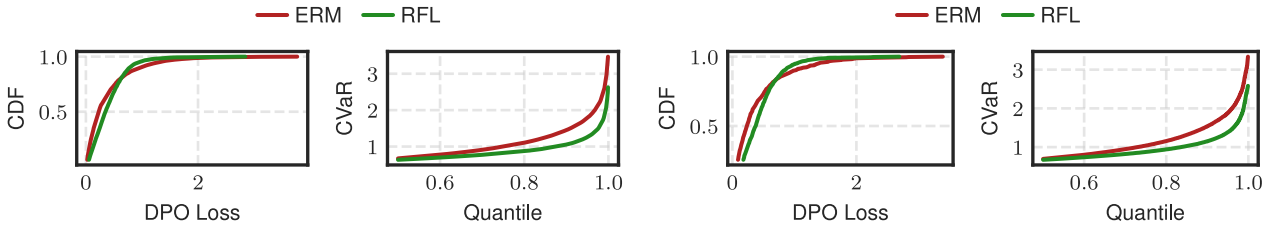


Figure 8: Average training losses at the end of training for UTKFace age regression. **FL and RFL demonstrate good performance across various combinations of primal and dual learning rates.** $\epsilon = 0.02$.

Hyper-parameter Robustness. Figure 8 displays the training losses at the end of training for the UTKFace age regression task (with $\epsilon = 0.02$). Both **FL** and **RFL** exhibit strong performance across a wide range of primal and dual learning rate combinations, covering multiple orders of magnitude. This suggests that **FL** and **RFL** are relatively robust to these hyperparameters.

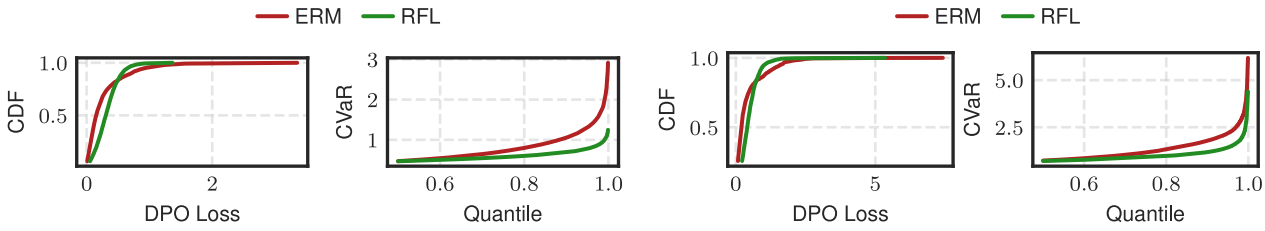
Table 13: DPO losses at the end of training for the Llama and Zephyr models on the direct preference optimization task. **RFL generally leads to reduced maximum losses, suggesting a more compact loss distribution, while ERM achieves better average performance.**

Model	Method	Train		Test	
		Avg. Loss	Max Loss	Avg. Loss	Max Loss
LLaMA	**ERM**	0.387	3.769	0.395	3.395
LLaMA	**RFL**	0.401	2.843	0.424	2.693
Zephyr	**ERM**	0.269	3.354	0.402	7.423
Zephyr	**RFL**	0.321	1.368	0.457	5.350



Train CDF and CVaR. Test CDF and CVaR.

Figure 9: Llama 3.1-8B train and test empirical CDF and CVaR. The test plots in (b) correspond to Fig. 2 in §5.3.



(a) Train CDF and CVaR. (b) Test CDF and CVaR.

Figure 10: Zephyr 3B train and test empirical CDF and CVaR. The plots illustrate the differences in loss distributions between **ERM** and **FL**, with **FL** providing a more compact and stable loss distribution, especially in the tails.

C.3 Direct Preference Optimization

Can we Learn with Feasible Learning? Table 13 shows the performance at the end of training for the direct preference optimization task for Llama and Zephyr models. We report the DPO loss of **ERM** and **RFL** for both models across training and test splits.

For both models, **RFL** results in a slight increase in the average train and test losses compared to **ERM**. However, **RFL** also significantly reduces the maximum losses in both the training and testing phases.

How does Resilience Help? Due to the high computational cost of fine-tuning these billion-parameter models, we did not perform a resilience impact study in the DPO task. Additionally, we did not assess the hyper-parameter robustness of **FL** and **RFL**.

Our main objective is to demonstrate the effectiveness of **RFL** in direct preference optimization, focusing on its ability to shape the loss CDFs. Our results prove that **RFL** can effectively mitigate high losses in preference optimization, even with minimal hyper-parameter tuning.

How do FL Solutions Compare to ERM? Table 13 also shows that **FL** reduces the maximum losses both in training and test splits, highlighting its ability to reduce worst-case losses. Figures 9 and 10 display the CDFs and CVaRs for the train and test sets for the Llama and Zephyr models, respectively. For both models, we observe that **FL** yields a more compact distribution of losses compared to **ERM**, with higher CDF values and lower CVaRs in the tail regions, indicating a reduced risk of large losses.

Supplementary References

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge university press, 2004. (Cit. on p. 13)

Ralph Allan Bradley and Milton E Terry. Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons. *Biometrika*, 1952. (Cit. on p. 14)

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient Finetuning of Quantized LLMs. In *NeurIPS*, 2024. (Cit. on p. 14)

Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Convex Analysis and Minimization Algorithms I: Fundamentals*. Springer science & business media, 1996. (Cit. on p. 13)

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. In *ICLR*, 2022. (Cit. on p. 14)

Timo Kaufmann, Paul Weng, Viktor Bengs, and Eyke Hüllermeier. A Survey of Reinforcement Learning from Human Feedback. *arXiv preprint arXiv:2312.14925*, 2023. (Cit. on p. 14)

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. Orca: Progressive Learning from Complex Explanation Traces of GPT-4. *arXiv preprint arXiv:2306.02707*, 2023. (Cit. on p. 14)

Brendan O'Donoghue. Operator Splitting for a Homogeneous Embedding of the Linear Complementarity Problem. *SIAM Journal on Optimization*, August 2021. (Cit. on p. 13)

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022. (Cit. on p. 14)