Optimizing Neural Network Training and Quantization with Rooted Logistic Objectives

Zhu Wang **Praveen Raj Veluswami** * **Harsh Mishra** * **Sathya N. Ravi**
Department of Computer Science, University of Illinois Chicago

Abstract

First-order methods are widely employed for training neural networks that are used in practical applications. For classification of input features, Cross-Entropy based loss functions are often preferred since they are differentiable everywhere. Recent optimization results show that the convergence properties of first-order methods such as gradient descent are intricately tied to the separability of datasets and the induced loss landscape. We introduce Rooted Logistic Objectives (RLO) to improve practical convergence behavior with benefits for downstream tasks. We show that our proposed loss satisfies strict convexity properties and has better condition number properties that will benefit practical implementations. To evaluate our proposed RLO, we compare its performance on various classification benchmarks. Our results illustrate that training procedure converges faster with RLO in many cases. Furthermore, on two downstream tasks viz., post-training quantization and finetuning on quantized space, we show that it is possible to ensure lower performance degradation while using reduced precision for sequence prediction tasks in large language models over state of the art methods.

1 Introduction

Neural networks have become a necessity to enable various real-world applications, especially in large scale settings. An appropriate model with trainable parameters is chosen with the information of the domains or use-cases pertaining to the applications [Devlin et al., 2018, Radford et al., 2021, Caron et al., 2021]. The parameters of the model are then iteratively updated to find optimal solutions of a mathematically valid loss function. The loss function is defined on data points based on the application under consideration [Ghosh et al., 2017, Lin et al., 2017, Kavalerov et al., 2021, Hui et al., 2023]. Once the iterative procedure terminates (or is terminated with stopping conditions), the model parameters w can be used to make predictions on unseen points. Thus, it is crucial to understand how different algorithms behave during optimization phase, also called the training procedure. In large scale setting, first order methods are preferred since they require the least computing resources, and are easier to implement with Automatric Differentiation packages [Paszke et al., 2017, Loshchilov and Hutter, 2018, Reddi et al., 2019]. Naturally, the success and efficiency of first order methods depend on the landscape properties of the loss function when are applied on samples in datasets [Deng et al., 2009, Karras et al., 2019].

Factors that may lead to poor optimization landscape. Consider the task of classification in which a dataset $\mathcal{D}$ is represented as a set of pairs (x, y), where x denote features, and y denote corresponding classes or labels [Pranckevičius and Marcinkevičius, 2017, Singh et al., 2017, Zhang and Liu, 2023]. In binary classification, the task is to categorize x into one of two classes using model parameters after optimization. Here, it is known the rate of convergence of (stochastic) gradient descent or its variants – the de-facto first order method – to the optimal solution is primarily influenced by two factors: **1.** *condition number* of the loss function $\mathcal{L}$ [Overton, 2001]: this number gives an insight into the structure and properties of the dataset at various parameter settings. A lower condition number implies that gradients provide a good local approximation of the loss function which makes optimization faster [Hazimeh et al., 2022, Boob et al., 2023]. For a one layer neural network model, this condition number

*Contributed to the project as a student at UIC.

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

is the ratio of largest to smallest eigenvalue of the positive semidefinite matrix $X^\top X$ where X is data matrix in which (x, y) pairs are appropriately stacked as rows/columns; **2.** recent work have shown that *separability* of $\mathcal{D}$ is an important factor to consider for modeling and training purposes [Shamir, 2021, Tarzanagh et al., 2023]. Separability measures or quantifies the extent to which a model can distinguish between two features x_1, x_2 from different classes $y_1, y_2, y_1 \neq y_2$ in the dataset $\mathcal{D}$. A highly separable dataset is easier to classify, and the optimization process is expected to converge faster. Indeed, separability is inherent to the dataset, and so without employing extra pre-processing steps like normalization [Ioffe and Szegedy, 2015, Wu et al., 2021], augmentation [Shorten and Khoshgoftaar, 2019, Yarats et al., 2020], over-parametrization (more than one layer) [Du et al., 2018, Buhai et al., 2020], the level of separability is determined by the distribution from which $\mathcal{D}$ was sampled from.

In addition to concerns related to convergence of first order methods, it turns out that condition number plays a prominent role in deciding the success of model deployment in real world use cases. Recently, [Dong et al., 2019, Frantar et al., 2022, Chee et al., 2024, Tseng et al., 2024] proposed quantization of large language models (LLM) parameters for efficient inference. One key finding is that quantization strategies that use the estimate of the Hessian are preferable in practice. In other words, well-conditioned loss landscape is suitable for both post-training quantization and finetuning on quantized space [Dettmers et al., 2024, Yin et al., 2024].

If we optimize a well-conditioned loss function, then is it easy to quantize? Yes! To see why, assume for simplicity that our algorithm converges to a stationary point that satisfies second order optimality conditions – that is, Hessian at the stationary point is positive definite. With this assumption, and Taylor expansion, the loss values in the neighborhood increase steadily in all directions as we move away from the stationary point if the eigenvalues of the Hessian have small variance (i.e., have low condition number), thus making it easier to quantize since it is easier to sample nearby points with similar loss values, see Figure 1. Due to ill-conditioning, a point along the vertical direction from w^* may indicate a larger loss value while using cross-entropy (CE) loss, which is undesirable for quantization purposes. Indeed, our intuition that a small condition number of the Hessian is sufficient for quantization is applicable in nonconvex settings also, albeit it might not be necessary. Even in nonconvex settings, recent work on Hessian-based pruning algorithms such as [Wang et al., 2019, Wang et al., 2020] highlight the importance of the Hessian matrix and loss landscape on pruning weights in deep learning models.

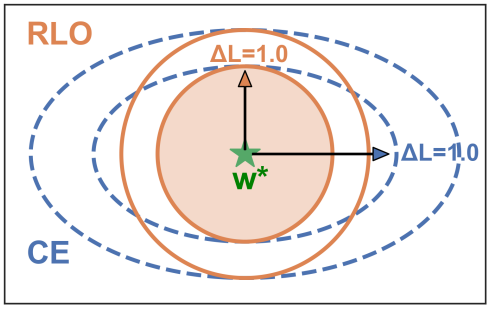


Figure 1: Visual illustration of the proposed RLO vs CE in two dimensional parameter space.

Our Contributions. We provide a plug-in replacement for log based loss functions for supervised classification and unsupervised generation tasks with provable benefits for optimizing parameters in learning models.

- First, we use a natural approximation to $(-\log)$ that is bounded from below and show that the approximation has desirable theoretical properties such as convexity and smoothness for practical optimization. Our theoretical analysis shows that the proposed *RLO* is an upper bound on popular cross entropy loss and has a lower condition number, achieving provable acceleration when first order methods are used for optimization.
- Second, we apply our loss to various datasets, architecture combinations and show that it can lead to significant empirical benefits for classifications. In image classifications, we show that the training time with our proposed RLO is much less than CE or Focal loss. It also provides 1.44% - 2.32 % gains over CE loss and 5.78 % - 6.66 % gains over Focal loss in term of test accuracy.
- Third, on post-training quantization of next word prediction models that are used in language modeling, we find that using RLO can lead to significantly lower perplexity values which is desirable. Our implementation of models and loss can be found at: `https://github.com/ellenzhuwang/rooted_loss`.

2 Preliminaries

Logistic regression is the task of finding a vector $w \in \mathbb{R}^d$ which approximately minimizes the empirical logistic loss [Ji and Telgarsky, 2018]. While logistic regression can be seen as a single-layer neural network, deep neural networks contain multiple such layers stacked together.

Each layer captures increasingly complex features from the input data. This hierarchical structure allows deep networks to model complex relationships.

Given datapoints $(x_i, y_i), i = 1, \ldots, n$, where $x_i \in \mathbb{R}^d$ denotes the features in d dimensions, and $y_i \in \{+1, -1\}$ is the binary label. We parametrize the prediction function of a new sample x as,

$$f(x) := \mathbb{P}(y = \pm 1|x) = \sigma(\pm w^\top x) \tag{1}$$

where σ is the sigmoid function. This is convenient because the maximum likelihood estimator of $w \in \mathbb{R}^d$ can be obtained by minimizing the *negative* log-likelihood function of w [Sur and Candès, 2019], written as,

$$\mathcal{L}_{\mathrm{LR}}(w) := \frac{1}{n} \sum_{i=1}^{n} \left[\log \left(1 + \exp \left(-y_i w^\top x_i \right) \right) \right]. \tag{2}$$

The cross-entropy (CE) loss is one of the most commonly used loss functions for training deep neural networks, most notably in multi-class classification problems. Given datapoints as (x_i, y_{ic}), where $c \in [C]$, C is the number of classes, $y_{ic} \in \{0, 1\}$ is a binary indicator to specify whether class c is in example i. Following the equation 2, multi-class multi-label CE loss is written as,

$$\mathcal{L}_{\mathrm{CE}}(w) := -\frac{1}{n} \sum_{i=1}^{n} \sum_{c=1}^{C} y_{ic} \log \left(\frac{\exp(w_c^\top x_i)}{\sum_{j=1}^{C} \exp(w_j^\top x_i)} \right), \tag{3}$$

where $w_j^\top x_i$ represents the prediction score for the i-th example and the j-th class. If $y_i \in \mathbb{R}^C$ is 1-hot at $c' \in [C]$ i.e., $y_{ic} = 0 \ \forall \ c \neq c'$, then the inner sum becomes $y_{ic'} \log \left(\frac{\exp(w_{c'}^\top x_i)}{\sum_{j=1}^{C} \exp(w_j^\top x_i)} \right)$ – CE loss.

3 Rooted Logistic Objective Function

In this section, we discuss mathematical properties of our loss function that may affect convergence behavior and connections to recently introduced loss functions.

3.1 Intuition: From Logistic Objective to Rooted Logistic Objective

Logistic for binary classification and log-sum-exp function in cross-entropy loss is a smooth approximation to the element-wise maximum function, where smoothness is desirable since gradient-based optimizers are commonly used for optimization. In this work, we will use the Taylor approximation of the natural logarithm function as follows: 1. for a fixed $u \in \mathbb{R}_+$, the derivative of u^v is given by $u^v \log(u)$ by Chain rule, 2. now observe that by evaluating the derivative at $v = 0$, we obtain $\log(u)$, and 3. finally, plugging the above two in the definition of derivative we have that $\log(u) = \lim_{v \downarrow 0} \frac{u^v - 1}{v} = \lim_{k \uparrow \infty} k \left(u^{1/k} - 1 \right)$. Thus, for training purposes, we propose using a fixed sufficiently large k with the following approximation to the log function: $\log(u) \approx k(u)^{\frac{1}{k}} - k$.

Our approximation expresses $\log(u)$ in terms of a function raised to the power of $\frac{1}{k}$. The constant k provides a degree of freedom that can be adjusted to fine-tune the approximation. Building on this approximation, we introduce a novel loss function, **R**ooted **L**ogistic **O**bjective function (**RLO**) denoted by $\mathcal{L}^k_{\mathrm{RLO}}(w)$. We modify the traditional logistic loss by incorporating the above approximation $\mathcal{L}^k_{\mathrm{RLO}}(w)$ defined as,

$$\frac{1}{n} \sum_{i=1}^{n} k \cdot \left[l_i^k(w) := \left(1 + \exp \left(-y_i w^\top x_i \right) \right)^{\frac{1}{k}} \right]. \tag{4}$$

Similarly, our multi-class rooted loss l_i^k for a single point (x_i, y_i) is defined as $\sum_{c=1}^{C} y_{ic} \left(\frac{\exp(w_c^\top x_i)}{\sum_{j=1}^{C} \exp(w_j^\top x_i)} \right)^{\frac{1}{k}}$.

Logistic loss plays a pivotal role in penalizing prediction errors, particularly for the true class denoted as y_i in classification tasks. One of its notable characteristics is the high loss and large gradient when w misclassifies a point, that is, when $\left| w^\top x_i \right|$ is large and $\mathrm{sign}(y_i) \neq \mathrm{sign}(w^\top x_i)$. This sharp gradient is beneficial in gradient-based optimization methods, such as gradient descent, because it promotes larger update steps, accelerating the convergence towards the optimal solutions. Moreover, when we consider the gradient contributions from incorrect classes, the "signal" coming from the gradient is weaker, so such optimization schemes may be less effective in driving the probabilities for these classes to zero. Specifically, optimization algorithms might struggle or take longer to drive the predicted probabilities of these incorrect classes towards zero. In simpler terms, while logistic loss is adept at penalizing mistakes for the true class, it might be gentler or slower in correcting overconfident incorrect predictions. Thus, in cases where the initialization is far away from the optimal solution or when the loss values are high, RLO is intended to provide gradients with larger magnitude.

3.2 Convexity of RLO

Standard logistic regression function in equation 2 has favorable convexity properties for optimization. Specifically, it is strictly convex with respect to parameters w, for more details, see [Freund et al., 2018]. By using Chain and Product rules, we can calculate the gradient

$\nabla_w l_i^k$ for a single point (x_i, y_i) as,

$$\begin{aligned}\nabla_w l_i^k(w) &= \frac{l_i^k(w)}{k} \cdot \frac{\exp\left(-y_i w^\top x_i\right)}{1+\exp\left(-y_i w^\top x_i\right)} \cdot (-y_i x_i) \\ &= \frac{l_i^k(w)}{k} \cdot \frac{1}{\exp\left(y_i w^\top x_i\right)+1} \cdot (-y_i x_i) \quad (5) \\ &= -g(w, x_i) \cdot y_i x_i, \quad (6)\end{aligned}$$

where $g(w, x_i) := \sigma(y_i w^\top x_i) \cdot l_i^k(w)/k \geq 0$. Applying product rule, we have that the Hessian $\nabla^2 l_i^k(w)$ for a single point (x_i, y_i) as,

$$\nabla^2 l_i^k(w) = h(w, x_i) \cdot x_i x_i^\top, \quad (7)$$

where $h(w, x_i) := l_i^k(w) \cdot \sigma(y_i w^\top x_i) \cdot \left[1 - \sigma(y_i w^\top x_i) \cdot (1 - 1/k)\right] > 0$ since both $\sigma(\cdot), 1/k \in (0, 1)$. We have included the full proof of Hessian in the Appendix A.1. With these calculations, we have the following result:

Lemma 3.1. *$\mathcal{L}_{RLO}^k(w)$ is a strictly convex function whenever $k > 1$ as is considered here.*

Note that, our result is novel because standard composition rules for convex optimization do not apply. This is due to the fact the function $(\cdot)^{\frac{1}{k}}$ is a *concave* function in the nonnegative orthant. Numerically, the main advantage is that the condition number of $\mathcal{L}_{\text{RLO}}^k(w)$ is independent of data, while $\mathcal{L}_{\text{LR}}^k(w)$ can be quite ill-conditioned for inseparable datasets due to the $\log(\cdot)$ function. More details can be found in Chapter 12 of [Overton, 2001].

While strict convexity is true for both Logistic and RLO loss functions, the following result says that the full batch RLO is guaranteed to be as conditioned as Logistic objective function by comparing the coefficient of the Hessian term $x_i x_i^\top$ in RLO and Logisitic objectives:

Lemma 3.2. *Let $r_i := h_{RLO}(w_i^*, x_i^*)/h_{LR}(w, x_i) \in \mathbb{R}_{\geq 0}$. Then if $k \leq \exp(l_i^k(w_i^*))$ then $r_i > 1$ where w^* is the optimal parameters for LR.*

Above, Lemma 3.2 states that as long as k is not chosen to be too large, the gradient directions may provide sufficient descent needed for fast convergence. This property makes it ideal for solving classification problems. From Lemma 3.1 and 3.2, we can conclude that there is a range of values of k that provides better conditioning for individual data points. It is beneficial when using stochastic algorithms that use a random mini-batch of samples at each iteration instead of the full dataset to compute gradient. Using Lemma 3.2, we have the following result that shows that RLO loss functions have a better (or lower) condition number compared to LR. Please see Appendix A.1 for the proof.

Lemma 3.3. *Let M be a symmetric positive definite matrix, and let $\kappa(M)$ denote the condition number defined as the ratio of largest and smallest eigenvalues of M. We have that $\kappa\left(\nabla^2 \mathcal{L}_{LR}(w)\right) \geq \kappa\left(\nabla^2 \mathcal{L}_{RLO}^k(w)\right)$ for any w as long as $k \leq \exp(l_i^k(w_i^*))$.*

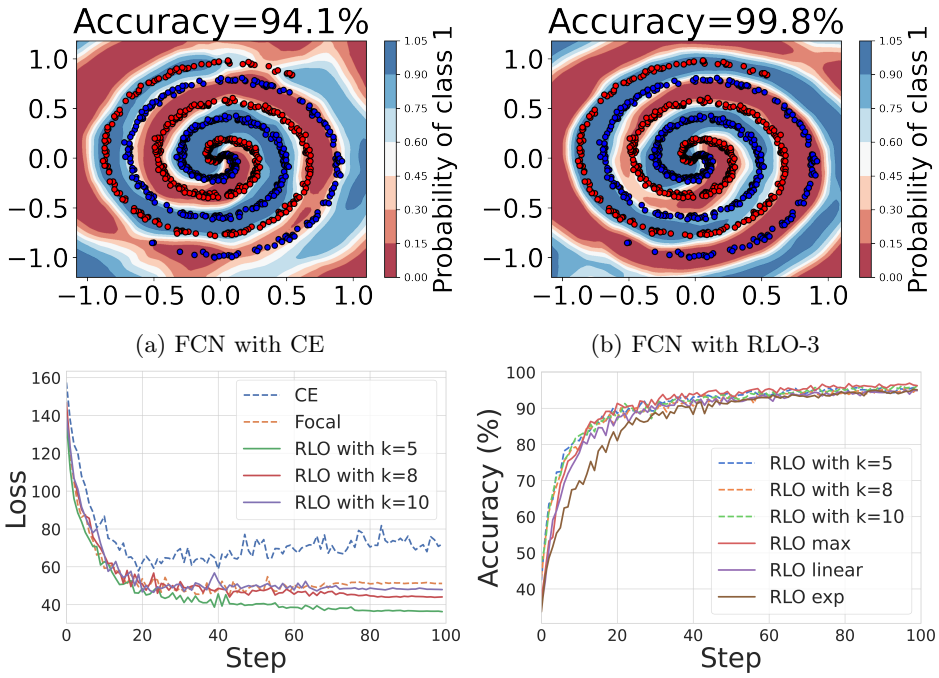
Our proof uses the fact that $h_{\text{RLO}}(w, x_i) \geq h_{\text{LR}}(w, x_i) + \delta_i$ from Lemma 3.2, matrix representation of Hessian of these two loss functions as in equation 10 and equation 13 along with the variational definition of largest and smallest eigenvalues to bound the condition number. Please see Appendix A.2 for the details.

Why does condition number lead to faster optimization for nonconvex models? Consider the case that the models are twice differentiable with respect to w, and w^* is approximate locally optimal solution, that is, $\nabla_w \mathcal{L}(w^*) \approx 0$. By Taylor's theorem, when w is close to w^*, $\mathcal{L}$ is approximately equal to a quadratic function defined by its Hessian $\nabla^2 \mathcal{L}(w^*)$. Hence, with a well conditioned, we may be able to get faster convergence to locally optimally solutions.

Generalization properties of RLO. Assuming that points $x_i \in \mathbb{R}^d$ are bounded i.e., $\|x\| \leq B_x$ and that there is an bounded optimal solution $\|w\| \leq B_o$, we expect that the generalization bounds for LR in equation 2 to hold for RLO in equation 4. This is because of the fact that asymptotically – when $k \uparrow +\infty$ – the Hessian coefficient of RLO is at most 1, which guarantees that the gradient is Lipschitz continuous [Lei et al., 2019, Bartlett and Mendelson, 2002].

Role of Hessian on Quantization. Recent studies in [Chee et al., 2024, Tseng et al., 2024] suggest that the worst case performance of models after quantization is governed by Hessian of the loss function. The basic intuition is that when the trace of Hessian of the loss function is small, then quantization operation has a negligible effect on the weights. Formally, please see Lemma 3 in [Chee et al., 2024].

Specifically, the Hessian matrix's role in quantization is twofold. Firstly, it helps analyze how sensitive the model's output is to changes in each parameter. Parameters with high sensitivity have a disproportionate impact on model performance when quantized, necessitating careful adjustments. Secondly, a well-conditioned Hessian, indicated by a small trace, suggests that quantization operations have a negligible effect on the weights, assuming the loss function is convex. Even though the result assumes that the Hessian is positive semidefinite, or in other words, the loss is convex, we have evidence that RLO provides approximately optimal weights with clustered eigenvalues. Moreover, this assumption can be guaranteed if we use a first order learning method that can converge to solutions w that satisfies second order sufficiency conditions



(a) FCN with CE

(b) FCN with RLO-3

(c) RN-50 on CIFAR-100

(d) RN-50 on CIFAR-10

Figure 2: RLO performance on synthetic dataset and CIFAR-10/100 training. (a) the intervening white line between the red and blue regions denotes the decision boundary. (b) RLO obtains more stable validation loss. (c) Comparisons of RLO with various setups, exponential increases up to max outperforms others.

[Curtis et al., 2021] to train/finetune networks.

3.3 Relationship with Tsallis entropy (TE) and Fenchel-Young (FY) Losses

In Section 4.1 of [Martins et al., 2022], TE is introduced through the use of β - logarithm and β - exponential functions. Our proposed RLO function is equivalent to a TE function by setting $1-\beta = 1/k$ where k is the parameter in RLO. To our knowledge, there are no existing works that use a *loss* function for training purposes. We find that the only similarity in [Martins et al., 2022] is the approximation of the log function using power functions (RLO) in the definition of TE and used TE as a regularizer for prediction. We will now describe the differences in detail.

Application Context: We use RLO directly as the objective function for training neural networks. In contrast, [Martins et al., 2022] utilizes the TE primarily as a regularizer on the distribution of outputs of a neural network. In particular, proposed RLO: $\mathbb{R} \to \mathbb{R}$ is a real valued one dimensional function whereas the loss $L : \mathcal{F} \times \mathcal{M} \to \mathbb{R}$ in Definition 3 of [Martins et al., 2022] quantifies the descrepancy between two distributions viz., predicted distribution $\mathcal{F}$ and desired output distribution $\mathcal{M}$.

Connection to FY Losses: For finite dimensional setting, we refer to [Blondel et al., 2020]. Specifically, RLO is applicable to multiclass classification settings with one label for each sample in training data corresponding to Logistic, CRF, (and to some extent, Structured Perceptron, Structured Hinge) losses, please see details in Table 1 in [Blondel et al., 2020]. [Blondel et al., 2020] has shown FY loss for various prediction functions. This is shown by using convex conjugate to decompose the loss into individual functions on predictions, desired output (in training data), and inner product between them. When the prediction function is softmax transformation of the neural network output, then the FY loss automatically becomes the CE loss given by $CE = \log\left[1 + \sum_{c \neq c'} \exp(\cdot)\right]$ where the sum is over all classes except for label c'.

Furthermore, using TE to regularize (instead of Shannon entropy) the prediction function in binary classification gives us sparse sigmoid, please see Section 6.2 in [Blondel et al., 2020] – different from the prediction function we use (which is sigmoid). Proposition 8 in [Martins et al., 2022] gives us the general expression for multiclass setup, once again different from our setup. We use the same prediction function as softmax function as in [Martins et al., 2022] but a different loss function.

With $0 < \beta = 1/k < 1$, our loss defined by RLO is $\bar{L} = \frac{\left[1+\sum_{c\neq c'} \exp(\cdot)\right]^{\beta}-1}{\beta}$. We will now show that $\bar{L}$ is an upper bound to CE using elementary concepts. Consider the following two functions $f_1(x) = \log(1+x), f_2(x) = \frac{(1+x)^{\beta}-1}{\beta}$ with $\mathbb{R} \ni x \geq 0$, and their difference $f_3(x) = f_2(x) - f_1(x)$. Then, we have that: 1. Value of f_3 at 0: $f_3(0) = f_2(0) - f_1(0) = 0$, and 2. Derivative of f_3: $f_3'(x) = f_2'(x) - f_1'(x) = (1+x)^{\beta-1} - \frac{1}{1+x} = \frac{(1+x)^{\beta}-1}{1+x} \geq 0$ since $(1+x)^{\beta} \geq 1$ for all $x \geq 0$. Thus, the difference is an increasing function in nonnegative reals, and then we have $\min_{x\geq 0} f_3(x) = h(0) = 0$. Now, since both f_1, f_2 are nonnegative for all $x \geq 0$, we have the desired upper bound result. We can show that ratio of f_1 and f_2 is at most 1 also. For algorithmic applications, this means that approximate solutions of $\bar{L}$ is a subset of those of CE, so we may conveniently use the same accuracy parameter used in CE applications while using RLO. In summary, in multiclass classification considered in this paper, our loss function is provably an upper bound of FY losses in [Blondel et al., 2020, Martins et al., 2022].

4 Experiments

In this section, we illustrate the experiments of using our proposed RLO on multiple architectures of models on various benchmark datasets. First, we evaluate RLO against CE loss and Focal loss [Lin et al., 2017] by training state-of-the-arts deep models, e.g., ResNet [He et al., 2016], ViT [Dosovitskiy et al., 2020] and Swin [Liu et al., 2021], on image classification tasks. Second, we quantify the benefits of RLO for post-training quantization on large language models, such as OPT [Zhang et al., 2022] and Llama [Touvron et al., 2023] family, and pruning vision models. Third, we compared rooted logistic regression with standard logistic regression on a swiss-roll synthetic dataset and 4 benchmark datasets from UCI machine learning repository in Appendix A.4.1. Finally, we showcase the application of image generations using RLO in Appendix A.5.

Datasets. (1) Swiss-roll Synthetic dataset Setup: We used the popular 2-class spiral with 1500 samples [Hui et al., 2023] which we randomly selected 70% data as train set and the remaining 30% data for testing. **(2) Image datasets:** We conducted image classification experiments on CIFAR-10/100 [Krizhevsky et al., 2009] for training from the scratch, and Tiny-ImageNet [mnmoustafa, 2017], ImageNet [Russakovsky et al., 2015] and Food-101 [Bossard et al., 2014] for fine-tuning with RLO. In addition, we used CIFAR-10 for pruning experiments. **(3) Text datasets:** We evaluated quantization on language generation tasks on the following datasets, WikiText2 [Merity et al., 2016], Penn Treebank (PTB) [Marcus et al., 1994], and C4 [Raffel et al., 2020].

4.1 DNNs for classification with RLO

Experiments setups: (1) We implemented three different layers (2, 3, 4) fully-connected neural networks (FCN) on a synthetic dataset. **(2)** For the vision models in image classification tasks, such as multi-class classification, we trained and fine-tuned the ViT-B [Dosovitskiy et al., 2020], ResNet-50 [He et al., 2016], and Swin-B [Liu et al., 2021] models. The approximation parameter k of our proposed RLO is chosen from the set $\{5, 8, 10\}$. We trained on CIFAR-10 and CIFAR-100 for 350 epochs with ViT, ResNet and Swin. We fine-tuned these models on Tiny-ImageNet and Food-101 for 15 epochs. We conducted the above experiments on 3 NVIDIA RTX 2080Ti GPUs. We used 3 different seeds for all experiments in image classification and illustrate the average results. To evaluate our proposed RLO, we used cross-entropy (CE) loss and focal loss as baselines. More details of the implementation are provided in Appendix A.4.2.

4.1.1 Observations on FCNs decision boundaries.

For simpler interpretation of parameters after training, we used the synthetic setups and visualized the decision boundaries learned by RLO compared with CE. Figure 6c and 2b shows the decision boundaries obtained using RLO and CE. The intervening white line between the red and blue regions denotes the decision boundary, acting as a visual partition between different classifications induced by the trained models. We can see that the margins which are the distances from datapoints to the decision boundary are larger in models trained using RLO in most of regions. Hence, RLO is beneficial to separate data points while also converging at a faster rate. We have provided more results and visualizations in Appendix A.4.2.

4.1.2 Nonconvex Optimization Benefits

(1) Performance gains: Here we evaluated the effectiveness of RLO for training deep networks – which is usually a nonconvex optimization problem. Table 1 shows that RLO performs the best and the second best accuracy across all datasets and network architectures. Specifically, training with RLO can lead to (approximately) 1.44% - 2.32 % gains over CE and 5.78 % - 6.66 % gains over focal loss in terms of test accuracy. Additionally, Figure 2c shows that using RLO it is possible to train much faster compared to

Dataset	Model	CE		Focal		RLO-5		RLO-8		RLO-10	
		Time	Acc	Time	Acc	Time	Acc	Time	Acc	Time	Acc
CIFAR-10	ViT	18.87	93.68	78.76	92.62	16.21	93.79	15.97	93.92	16.77	**94.10**
	ResNet	24.45	96.25	92.46	94.58	20.81	96.84	19.81	96.32	21.27	**97.16**
	Swin	26.74	92.80	38.76	91.81	22.23	92.77	22.99	**93.24**	22.95	93.10
CIFAR-100	ViT	47.32	78.30	65.29	76.43	32.46	**79.31**	35.73	79.70	31.88	79.05
	ResNet	26.34	82.98	19.98	80.52	26.27	83.88	25.79	83.26	24.81	**84.10**
	Swin	35.16	76.53	68.65	75.49	31.46	78.26	30.92	**78.80**	31.28	78.61
Tiny-INet	ViT	920.94	84.7	821.65	85.08	901.68	**86.05**	908.69	85.73	905.26	85.38
	ResNet	253.92	73.39	245.74	73.95	257.85	**74.19**	259.85	74.05	255.94	74.1
	Swin	950.72	88.85	952.54	88.22	932.56	88.88	928.37	88.74	926.53	**88.91**
Food-101	ViT	669.43	90.24	677.32	89.57	664.05	90.32	664.85	**91.80**	667.96	91.72
	ResNet	186.45	88.12	180.35	85.40	189.56	88.79	189.75	89.20	183.87	**89.93**
	Swin	718.01	90.21	700.89	88.64	723.89	91.26	723.25	**91.64**	717.04	91.52

Table 1: Test performance (top-1 accuracy) for image classifications on different datasets. Time is averaged one epoch training time in seconds. Note that, k values are 5, 8, and 10. Our RLO obtains the best and second best accuracy in all datasets and models.

Dataset	Model	CE	Focal	Square	Sparsemax	RLO-5	RLO-8	RLO-10
CIFAR-10	RN	96.25	94.58	94.69	95.86	96.84	96.32	**97.16**
	ViT	93.68	92.62	93.18	93.23	93.79	93.92	**94.10**
CIFAR-100	RN	82.98	80.52	81.20	81.90	83.88	83.26	**84.10**
	ViT	78.30	76.43	76.5	78.12	79.31	**79.70**	79.05
ImageNet	RN	78.12	76.96	77.92	78.24	79.30	**79.43**	79.21
	ViT	83.18	82.76	82.54	83.12	84.18	84.50	**84.69**

Table 2: More test performance for image classifications on extensive datasets and more baseline loss functions. Time is averaged one epoch training time in seconds. Note that, CE is cross-entropy for short. k values are 5, 8, and 10. Our RLO obtains the best and second best accuracy in all datasets and models.

CE and focal losses. Our results indicate that our proposed RLO can accelerate neural networks training and also provide performance improvements across many datasets and model architectures. Finally, Figure 3a shows that while using RLO loss, it is possible to find parameters such that the Hessian matrix is better conditioned compared to CE loss functions. This shows that RLO is suitable in situations where downstream applications such as fine-tuning and unlearning have significant value.

(2) Generalization benefits: Figure 2c shows that the validation loss using CE increases over iterations which is undesirable. In contrast, we can see that validation loss decreases over time on the same dataset and model while training with RLO, which indicates that RLO can mitigate overfitting concerns. More results and visualizations are in Appendix A.4.2.

4.2 Efficient tuning with RLO

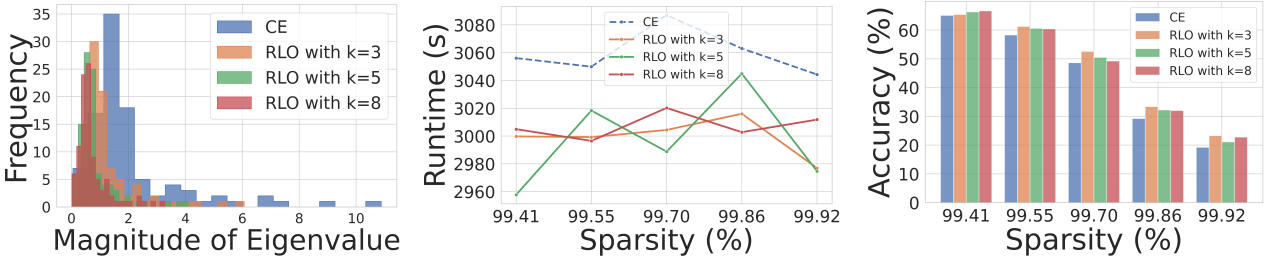
4.2.1 LLMs Quantization with RLO

Now we demonstrate that RLO provides a suitable landscape which can mitigate the impact of reducing precision when quantizing in two different settings: *(1) Finetuning with RLO and Quantizing:* we finetuned OPT family with CE and RLO on Wikitext2, and quantization the finetuned models with QUIP [Chee et al., 2024], and *(2) Finetuning on quantized space with RLO:* we finetuned quantized Llama family with CE and RLO on Wikitext2 with LLMTools [Yin et al., 2024]. We conducted all our experiments of (1) on NVIDIA V100 GPUs and (2) on NVIDIA A100 GPUs. More details on training can be found in Appendix A.4.3.

Observations: In both the experimental setups we considered, our proposed RLO loss function consistently outperformed the cross-entropy loss in terms of perplexity across all datasets and different model families from size 125m to 7b. Our results are shown in Table 3. The table clearly indicates that improvement was observed not only in the models at full precision but also under quantization at 4-bit, 3-bit, and 2-bit. Remarkably, the most significant gains in performance are in the 2-bit quantized models, such as quite lower perplexity from 818.94 to 347.73 on C4 with OPT-125m. These results illustrate the benefits of a smoother loss

Bits	Dataset	OPT-125M			OPT-1.3b			Llama2-7b		
		CE	RLO-5	RLO-10	CE	RLO-5	RLO-10	CE	RLO-5	RLO-10
16	Wiki	27.58	**21.23**	22.31	14.83	14.08	**13.4**	5.68	**5.32**	5.41
	PTB	45.77	**35.95**	37.23	22.71	21.16	**20.60**	10.83	**9.96**	10.20
	C4	43.71	**38.04**	42.06	25.23	24.17	**23.2**	7.17	**6.86**	6.92
4	Wiki	28.59	**22.53**	23.09	15.98	15.23	**14.49**	5.81	**5.64**	5.66
	PTB	47.18	**38.06**	39.97	24.18	22.75	**22.15**	11.02	10.53	**10.39**
	C4	44.12	43.37	**40.15**	25.85	24.41	**23.28**	7.84	**7.01**	7.19
3	Wiki	32.03	**24.89**	26.34	16.71	15.91	**15.46**	5.92	**5.82**	5.85
	PTB	56.69	**45.61**	48.92	26.18	25.15	**24.49**	11.88	**10.99**	11.16
	C4	100.41	**49.07**	55.73	28.31	27.83	**27.6**	8.17	**7.29**	7.35
2	Wiki	127.37	101.16	**98.17**	28.41	27.37	**27.13**	6.89	6.77	**6.69**
	PTB	334.5	268.92	**250.85**	273.26	**211.8**	238.62	12.36	**11.86**	12.01
	C4	818.94	470.66	**347.73**	632.49	**335.93**	412.31	8.84	**8.26**	8.30

Table 3: Text generation performance (perplexity ↓) for LLMs on different datasets. Note that, k values are 5 and 10. Our RLO obtains the lowest perplexity in all datasets and models.



(a) Magnitude of Eigenvalues (b) Prune RN20 on CIFAR10 (c) Prune RN20 on CIFAR10

Figure 3: (a) is the histogram of RLO vs CE spectrum at w^* for a fixed architecture, and dataset. It shows that the eigenvalues of solutions found by RLO are close to each other, so the landscape is well conditioned compared to solutions found using CE loss. (b) and (c) are the results of ResNet20 in CIFAR10 in terms of test accuracy and runtime in seconds of pruned on different sparsity levels of RLO vs CE. Note that RLO achieved consistently better performance with less runtime.

landscape and a well-conditioned optimization process in LLM quantization. Models trained using our loss function can preserve the performance of LLMs even under significant computational constraints. In terms of speed, the quantization time of OPT-125m with RLO is only 7.19 second on one V100 which is 2 second less than using CE loss. Therefore, RLO demonstrates the potential of the efficient deployment of LLMs with limited computational resources in practical scenarios.

4.2.2 ResNet Pruning with RLO.

We performed experiments with gem-miner (a pruning method) [Sreenivasan et al., 2022] on ResNet-20 on CIFAR-10 and fine-tuned the pruned models with RLO. We evaluated the performance of RLO in different sparsity levels of pruned models. Our main focus is on high sparsity regime from 99.41% to 99.92% since they are harder instances.

Takeaway: As shown in Figure 3c, in the higher sparsity level, RLO performs better in terms of accuracy. It highlights that RLO encourage the network to learn more robust features even when a large number of weights are pruned.

4.3 Ablation studies

4.3.1 How to choose the parameter k in practice?

Our proposed RLO includes a hyperparameter k as defined in equation 4. A natural approach to using RLO is not to select a fixed value for k but to gradually increase it, as the limiting case corresponds to the minimum norm solution. To explore the impact of k, we conducted experiments with various setups, including using constant values, exponential increases, linear increases, and exponential increases up to a maximum value as in Lemma 3.2, followed by fixing the value to that maximum. From results shown in Table 1 and 7, we can see that the best k values varies across datasets

Bits	Dataset	Llama3-8b			Llama3-70b		
		CE	RLO-5	RLO-10	CE	RLO-5	RLO-10
4	Wiki	6.3	6.1	**6.0**	3.6	**3.4**	**3.4**
	C4	11.5	**11.0**	11.1	7.1	**6.8**	6.9
3	Wiki	7.5	**7.2**	7.3	4.7	**4.3**	4.5
	C4	13.1	**12.9**	**12.9**	8.1	**8.0**	7.9
2	Wiki	8.2	**7.9**	8.0	6.8	6.6	**6.5**
	C4	17.5	17.3	**17.1**	10.2	**10.0**	10.1

Table 4: Text generation performance (perplexity ↓) for LLMs on different datasets (Continued). Note that, k values are 5 and 10. Our RLO obtains the lowest perplexity in all datasets and models.

Model	# Param	CE		Focal		RLO-10	
		Train	Test	Train	Test	Train	Test
ResNet-34	21.3M	99.79 ± 0.09	93.87 ± 0.05	98.94 ± 0.23	92.24 ± 0.12	**99.88 ± 0.02**	**94.13 ± 0.07**
ResNet-50	23.7M	**100.00 ± 0.00**	94.22 ± 0.13	99.08 ± 0.09	92.50 ± 0.11	**100.00 ± 0.00**	**95.76 ± 0.04**
ResNet-101	42.7M	**100.00 ± 0.00**	94.89 ± 0.08	99.46 ± 0.13	92.79 ± 0.15	**100.00 ± 0.00**	**95.82 ± 0.03**
ViT-S	18.6M	90.43 ± 0.30	87.61 ± 0.34	88.17 ± 0.35	85.93 ± 0.29	**91.30 ± 0.15**	**88.25 ± 0.09**
ViT-B	88.4M	94.50 ± 0.13	92.18 ± 0.25	94.88 ± 0.10	90.62 ± 0.12	**96.04 ± 0.13**	**93.61 ± 0.04**
ViT-L	232.5M	95.32 ± 0.16	94.25 ± 0.10	93.81 ± 0.06	92.81 ± 0.14	**97.42 ± 0.05**	**94.81 ± 0.08**

Table 5: Ablations on model architectures, including running time and test performance (accuracy ± standard deviations) on CIFAR-10. Time is averaged over one epoch in seconds. k value is 10. Our RLO obtains the best train and test accuracy in all models.

and neural network architectures. Our results in Figure 2d indicate that constant values perform better in the first few epochs, but exponential increases up to a maximum value can achieve the best performance in term of accuracy – so it is not necessary to accurately choose k to use RLO. Finally, we can see that k is much more smaller than number of samples or feature dimensions as in Lemma 3.2. We have included results with more k values are in Appendix A.4.2.

4.3.2 Is RLO sensitive to model architectures/sizes?

First, in Table 1, we present our results on the performance of RLO for image classification tasks for various combinations of datasets and deep neural network models. We can see that when $k \in \{5, 8, 10\}$, the performance gain over CE and focal methods is significantly improved. In fact, with respect to test accuracy, RLO is better in almost all of our evaluation settings. Second, we performed experiments with different k values on LLMs quantization. Both k values obtain lower perplexities across all datasets and models. Third, for further ablation, we chose the k value to be 10 and compared with the baselines under different model architectures of ResNet and ViT. The ablation results in Table 5 suggest that RLO resoundingly performs better, even under different architectures of the same model. Both the training as well as the test accuracy under RLO-10 are better than those trained with CE and focal losses. Thus, RLO guarantees performance gain across different hyperparameters and model architectures.

5 Conclusions and Future work

We conducted comprehensive evaluations of a new class of loss functions for prediction tasks in both shallow and deep networks. This class exhibits favorable optimization and generalization properties for high-dimensional data. Our results build on recent findings that the standard logistic loss $L_{\mathrm{LR}}(\cdot)$ needs adjustment for improved convergence and generalization ([Sur and Candès, 2019]). The minimizers of these losses coincide as k is increased. Our experiments show that rooted logistic loss performs better with first-order methods. Future work includes investigating the convergence rate to the max classifier and the generalization properties of RLO, as well as exploring the dependence of k on excess risk and generalization bounds. Analyzing the rate of convergence in terms of prediction and generalization using Hölder smoothness seems to be a promising direction [Hanneke et al., 2023, Emami et al., 2020]. Incorporating RLO in Second-order methods like Sophia [Liu et al., 2023] is an open problem.

Impact Statements: This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

6 Acknowledgments

We thank anonymous reviewers for constructive feedback and suggestions for improvements. We thank Electronic Visualization Laboratory for hosting V100 GPUs and Balajee Vamanan for providing access to the computational resources.

References

[Arora et al., 2017] Arora, S., Ge, R., Liang, Y., Ma, T., and Zhang, Y. (2017). Generalization and equilibrium in generative adversarial nets (GANs). In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 224–232. PMLR.

[Asuncion and Newman, 2007] Asuncion, A. and Newman, D. (2007). Uci machine learning repository.

[Bartlett and Mendelson, 2002] Bartlett, P. L. and Mendelson, S. (2002). Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482.

[Blondel et al., 2020] Blondel, M., Martins, A. F., and Niculae, V. (2020). Learning with fenchel-young losses. *Journal of Machine Learning Research*, 21(35):1–69.

[Boob et al., 2023] Boob, D., Deng, Q., and Lan, G. (2023). Stochastic first-order methods for convex and nonconvex functional constrained optimization. *Mathematical Programming*, 197(1):215–279.

[Bossard et al., 2014] Bossard, L., Guillaumin, M., and Van Gool, L. (2014). Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*.

[Brock et al., 2018] Brock, A., Donahue, J., and Simonyan, K. (2018). Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*.

[Buhai et al., 2020] Buhai, R.-D., Halpern, Y., Kim, Y., Risteski, A., and Sontag, D. (2020). Empirical study of the benefits of overparameterization in learning latent variable models. In *International Conference on Machine Learning*, pages 1211–1219. PMLR.

[Caron et al., 2021] Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., and Joulin, A. (2021). Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660.

[Chee et al., 2024] Chee, J., Cai, Y., Kuleshov, V., and De Sa, C. M. (2024). Quip: 2-bit quantization of large language models with guarantees. *Advances in Neural Information Processing Systems*, 36.

[Cubuk et al., 2020] Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. (2020). Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703.

[Curtis et al., 2021] Curtis, F. E., Robinson, D. P., Royer, C. W., and Wright, S. J. (2021). Trust-region newton-cg with strong second-order complexity guarantees for nonconvex optimization. *SIAM Journal on Optimization*, 31(1):518–544.

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

[Dettmers et al., 2024] Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. (2024). Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

[Devlin et al., 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[Dong et al., 2019] Dong, Z., Yao, Z., Gholami, A., Mahoney, M. W., and Keutzer, K. (2019). Hawq: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 293–302.

[Dosovitskiy et al., 2020] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

[Du et al., 2018] Du, S. S., Zhai, X., Poczos, B., and Singh, A. (2018). Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*.

[Emami et al., 2020] Emami, M., Sahraee-Ardakan, M., Pandit, P., Rangan, S., and Fletcher, A. (2020). Generalization error of generalized linear models in

high dimensions. In *International Conference on Machine Learning*, pages 2892–2901. PMLR.

[Fang et al., 2022] Fang, T., Sun, R., and Schwing, A. (2022). Diggan: Discriminator gradient gap regularization for gan training with limited data. *Advances in Neural Information Processing Systems*, 35:31782–31795.

[Frantar et al., 2022] Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. (2022). Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*.

[Freund et al., 2018] Freund, R. M., Grigas, P., and Mazumder, R. (2018). Condition number analysis of logistic regression, and its implications for standard first-order solution methods. *arXiv preprint arXiv:1810.08727*.

[Ghosh et al., 2017] Ghosh, A., Kumar, H., and Sastry, P. S. (2017). Robust loss functions under label noise for deep neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31.

[Hanneke et al., 2023] Hanneke, S., Kontorovich, A., and Kornowski, G. (2023). Near-optimal learning with average h\" older smoothness. *arXiv preprint arXiv:2302.06005*.

[Hazimeh et al., 2022] Hazimeh, H., Mazumder, R., and Saab, A. (2022). Sparse regression at scale: Branch-and-bound rooted in first-order optimization. *Mathematical Programming*, 196(1-2):347–388.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[Hui et al., 2023] Hui, L., Belkin, M., and Wright, S. (2023). Cut your losses with squentropy. *ICML*.

[Ioffe and Szegedy, 2015] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr.

[Ji and Telgarsky, 2018] Ji, Z. and Telgarsky, M. (2018). Risk and parameter convergence of logistic regression. *arXiv preprint arXiv:1803.07300*.

[Karras et al., 2020a] Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J., and Aila, T. (2020a). Training generative adversarial networks with limited data. In *Proc. NeurIPS*.

[Karras et al., 2018] Karras, T., Laine, S., and Aila, T. (2018). Flickr faces hq (ffhq) 70k from stylegan. *CoRR*.

[Karras et al., 2019] Karras, T., Laine, S., and Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410.

[Karras et al., 2020b] Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. (2020b). Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119.

[Kavalerov et al., 2021] Kavalerov, I., Czaja, W., and Chellappa, R. (2021). A multi-class hinge loss for conditional gans. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 1290–1299.

[Khosla et al., 2011] Khosla, A., Jayadevaprakash, N., Yao, B., and Fei-Fei, L. (2011). Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO.

[Krizhevsky et al., 2009] Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

[Kurach et al., 2019] Kurach, K., Lučić, M., Zhai, X., Michalski, M., and Gelly, S. (2019). A large-scale study on regularization and normalization in gans. In *International conference on machine learning*, pages 3581–3590. PMLR.

[Lei et al., 2019] Lei, Y., Dogan, Ü., Zhou, D.-X., and Kloft, M. (2019). Data-dependent generalization bounds for multi-class classification. *IEEE Transactions on Information Theory*, 65(5):2995–3021.

[Lin et al., 2017] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

[Lindholm et al., 2022] Lindholm, A., Wahlström, N., Lindsten, F., and Schön, T. B. (2022). *Machine learning: a first course for engineers and scientists*. Cambridge University Press.

[Liu et al., 2023] Liu, H., Li, Z., Hall, D., Liang, P., and Ma, T. (2023). Sophia: A scalable stochastic second-order optimizer for language model pretraining. *arXiv preprint arXiv:2305.14342*.

[Liu et al., 2021] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021). Swin

transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022.

[Loshchilov and Hutter, 2018] Loshchilov, I. and Hutter, F. (2018). Decoupled weight decay regularization. In *International Conference on Learning Representations*.

[Marcus et al., 1994] Marcus, M., Kim, G., Marcinkiewicz, M. A., MacIntyre, R., Bies, A., Ferguson, M., Katz, K., and Schasberger, B. (1994). The penn treebank: Annotating predicate argument structure. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

[Martins et al., 2022] Martins, A. F., Treviso, M., Farinhas, A., Aguiar, P. M., Figueiredo, M. A., Blondel, M., and Niculae, V. (2022). Sparse continuous distributions and fenchel-young losses. *Journal of Machine Learning Research*, 23(257):1–74.

[Merity et al., 2016] Merity, S., Xiong, C., Bradbury, J., and Socher, R. (2016). Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

[mnmoustafa, 2017] mnmoustafa, M. A. (2017). Tiny imagenet.

[Nie and Patel, 2020] Nie, W. and Patel, A. B. (2020). Towards a better understanding and regularization of gan training dynamics. In *Uncertainty in Artificial Intelligence*, pages 281–291. PMLR.

[Overton, 2001] Overton, M. L. (2001). *Numerical computing with IEEE floating point arithmetic*. SIAM.

[Paszke et al., 2017] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch.

[Pranckevičius and Marcinkevičius, 2017] Pranckevičius, T. and Marcinkevičius, V. (2017). Comparison of naive bayes, random forest, decision tree, support vector machines, and logistic regression classifiers for text reviews classification. *Baltic Journal of Modern Computing*, 5(2):221.

[Qi, 2020] Qi, G.-J. (2020). Loss-sensitive generative adversarial networks on lipschitz densities. *International Journal of Computer Vision*, 128(5):1118–1140.

[Radford et al., 2021] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

[Radford et al., 2015] Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

[Raffel et al., 2020] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

[Reddi et al., 2019] Reddi, S. J., Kale, S., and Kumar, S. (2019). On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*.

[Russakovsky et al., 2015] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

[Shamir, 2021] Shamir, O. (2021). Gradient methods never overfit on separable data. *The Journal of Machine Learning Research*, 22(1):3847–3866.

[Shorten and Khoshgoftaar, 2019] Shorten, C. and Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48.

[Singh et al., 2017] Singh, Y. K., Singh, N. D., et al. (2017). Binary face image recognition using logistic regression and neural network. In *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, pages 3883–3888. IEEE.

[Sreenivasan et al., 2022] Sreenivasan, K., Sohn, J.-y., Yang, L., Grinde, M., Nagle, A., Wang, H., Xing, E., Lee, K., and Papailiopoulos, D. (2022). Rare gems: Finding lottery tickets at initialization. *Advances in neural information processing systems*, 35:14529–14540.

[Sur and Candès, 2019] Sur, P. and Candès, E. J. (2019). A modern maximum-likelihood theory for high-dimensional logistic regression. *Proceedings of the National Academy of Sciences*, 116(29):14516–14525.

[Tarzanagh et al., 2023] Tarzanagh, D. A., Li, Y., Thrampoulidis, C., and Oymak, S. (2023). Transformers as support vector machines. *arXiv preprint arXiv:2308.16898*.

[Touvron et al., 2023] Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. (2023). Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

[Tseng et al., 2024] Tseng, A., Chee, J., Sun, Q., Kuleshov, V., and Sa, C. D. (2024). Quip: Even better llm quantization with hadamard incoherence and lattice codebooks.

[Wang et al., 2019] Wang, C., Grosse, R., Fidler, S., and Zhang, G. (2019). Eigendamage: Structured pruning in the kronecker-factored eigenbasis. In *International conference on machine learning*, pages 6566–6575. PMLR.

[Wang et al., 2020] Wang, C., Zhang, G., and Grosse, R. (2020). Picking winning tickets before training by preserving gradient flow. *arXiv preprint arXiv:2002.07376*.

[Wu et al., 2021] Wu, Y.-L., Shuai, H.-H., Tam, Z.-R., and Chiu, H.-Y. (2021). Gradient normalization for generative adversarial networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6373–6382.

[Yarats et al., 2020] Yarats, D., Kostrikov, I., and Fergus, R. (2020). Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International conference on learning representations*.

[Yin et al., 2024] Yin, J., Dong, J., Wang, Y., De Sa, C., and Kuleshov, V. (2024). Modulora: Finetuning 2-bit llms on consumer gpus by integrating with modular quantizers. *TMLR*.

[Zhang and Liu, 2023] Zhang, M. and Liu, K. (2023). On regularized sparse logistic regression. *arXiv preprint arXiv:2309.05925*.

[Zhang et al., 2022] Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., et al. (2022). Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

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. In section 3.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. Yes. In section 3.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. Yes. Our implementation of models and loss can be found at: `https://github.com/ellenzhuwang/rooted_loss`.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. Yes. In section 3.
   (b) Complete proofs of all theoretical results. Yes. In Appendix A.1.
   (c) Clear explanations of any assumptions. Yes. In section 3.

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. Our implementation of models and loss can be found at: `https://github.com/ellenzhuwang/rooted_loss`
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). Yes. In Section 4.
   (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. In Section 4.
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). Yes. In Section 4.

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.
   (b) The license information of the assets, if applicable. Yes.
   (c) New assets either in the supplemental material or as a URL, if applicable. Not Applicable.
   (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.

A Appendix

A.1 Proof of Lemma 3.2.

We first simplify gradient and Hessian of LR in equation 2 and RLO in equation 4 one by one.

Recall that the LR of a single sample $(x_i, y_i) \in \mathbb{R}^d \times \{+1, -1\}$ is given by,

$$l_i(w; x_i, y_i) := \log\left(1 + \exp\left(-y_i w^\top x_i\right)\right). \tag{8}$$

By using chain rule to differentiate loss in equation 8 with respect to w, we obtain the gradient $\nabla_w l_i(w; x_i, y_i)$ as,

$$\begin{aligned}\nabla_w l_i(w; x_i, y_i) &= -y\left[\frac{\exp(-y_i w^\top x_i)}{1 + \exp(-y_i w^\top x_i)}\right] x_i = -y_i\left[\frac{1 + \exp(-y_i w^\top x_i) - 1}{1 + \exp(-y_i w^\top x_i)}\right] x_i \\ &= -y_i\left[1 - \frac{1}{1 + \exp(-y_i w^\top x_i)}\right] x_i. \end{aligned} \tag{9}$$

Now applying the quotient rule and then chain rule once again to each coordinate of the gradient function in equation 9 and arranging the columns appropriately we obtain the hessian $\nabla^2 \mathcal{L}_{\text{LR}}(w; x_i, y_i)$ as,

$$\begin{aligned}\nabla^2 l_i(w; x_i, y_i) &= -y_i\left[-y_i \cdot \frac{\exp(-y_i w^\top x_i)}{\left(1 + \exp(-y_i w^\top x_i)\right)^2}\right] x_i x_i^\top \\ &= \left[\frac{\exp(-y_i w^\top x_i)}{\left(1 + \exp(-y_i w^\top x_i)\right)^2}\right] x_i x_i^\top \quad \left(\text{since } y_i^2 = 1\right). \end{aligned} \tag{10}$$

Now, $z^\top x_i x_i^\top z = \left(z^\top x_i\right)^2 \geq 0$ for any $z \in \mathbb{R}^d$, $\exp(\cdot) > 0$ for any finite argument, and input features $x_i \neq 0$, so we have that the hessian $\nabla^2 l_i(w; x_i, y_i)$ is positive definite. Moreover, since sum of positive definite matrices are positive definite, we have that $\mathcal{L}_{\text{LR}}$ is a strictly convex function.

We now repeat the calculations for RLO in similar way. Recall from equation equation 4 that the RLO of a single sample single sample $(x_i, y_i) \in \mathbb{R}^d \times \{+1, -1\}$ is given by,

$$l_i^k(w; x_i, y_i) = k \cdot \left(1 + \exp\left(-y_i w^\top x_i\right)\right)^{\frac{1}{k}}. \tag{11}$$

By using chain rule to differentiate loss in equation 11 with respect to w, we obtain the gradient $\nabla_w l_i^k(w; x_i, y_i)$ as,

$$\begin{aligned}\nabla_w l_i^k(w; x_i, y_i) &= -y\left[\frac{\exp(-y_i w^\top x_i)}{\left(1 + \exp(-y_i w^\top x_i)\right)^{1-\frac{1}{k}}}\right] x_i = -y_i\left[\frac{1 + \exp(-y_i w^\top x_i) - 1}{\left(1 + \exp(-y_i w^\top x_i)\right)^{1-\frac{1}{k}}}\right] x_i \\ &= -y_i\left[\left(1 + \exp(-y_i w^\top x_i)\right)^{\frac{1}{k}} - \left(1 + \exp(-y_i w^\top x_i)\right)^{\frac{1}{k}-1}\right] x_i. \end{aligned} \tag{12}$$

Now applying the quotient rule and then chain rule once again to each coordinate of the gradient function in the two terms in equation 12, arranging the columns appropriately and simplifying, we obtain the hessian $l_i^k(w; x_i, y_i)$ as,

$$\nabla^2 l_i^k(w; x_i, y_i) = \frac{\exp(-y_i w^\top x_i)}{\left(1 + \exp\left(-y_i w^\top x_i\right)\right)^{1-\frac{1}{k}}} \cdot \left[\frac{1}{k} + \frac{\left(1 - \frac{1}{k}\right)}{1 + \exp(-y_i w^\top x_i)}\right] \cdot x_i x_i^\top. \tag{13}$$

Once again, all the coefficents are positive since $1/k < 1$, and so $\nabla^2 \mathcal{L}_{\text{RLO}}(w; x_i, y_i)$ is positive definite. Whence, $\mathcal{L}_{\text{RLO}}^k(w; x_i, y_i)$ is a strictly convex function for any $k > 1$.

We are now ready to compare the scalar coefficients h of Hessian of LR in equation 10 and RLO in equation 13. As the first step, we note that,

$$\frac{1}{k} + \frac{\left(1 - \frac{1}{k}\right)}{1 + \exp(-y_i w^\top x_i)} > \frac{1}{k}$$

We consider the under-approximation of the RLO hessian coefficient given by ignoring the second term inside the square parenthesis in equation 13. This is the main novelty in our analysis. The under-approximated hessian can be written as,

$$\underline{\nabla^2} l_i^k(w; x_i, y_i) = \frac{1}{k} \cdot \left[\frac{\exp(-y_i w^\top x_i)}{\left(1 + \exp\left(-y_i w^\top x_i\right)\right)^{1-\frac{1}{k}}} \right] x_i x_i^\top. \tag{14}$$

Let us use r_i to denote the ratio of under-approximation of Hessian coefficient in equation 14 of RLO and ratio of hessian coefficient of LR in equation equation 13. Recall that we need $r_i > 1$ for some k to finish the proof. Now we proceed with the calculation of r_i as follows,

$$\begin{aligned} r_i :&= \left[\frac{1}{k} \cdot \frac{\exp(-y_i w^\top x_i)}{\left(1 + \exp\left(-y_i w^\top x_i\right)\right)^{1-\frac{1}{k}}} \right] \div \left[\frac{\exp(-y_i w^\top x_i)}{\left(1 + \exp(-y_i w^\top x_i)\right)^2} \right] \\ &= \frac{1}{k} \times \frac{\cancel{\exp(-y_i w^\top x_i)}}{\left(1 + \exp\left(-y_i w^\top x_i\right)\right)^{1-\frac{1}{k}}} \times \frac{\left(1 + \exp(-y_i w^\top x_i)\right)^2}{\cancel{\exp(-y_i w^\top x_i)}} \\ &= \frac{\left(1 + \exp(-y_i w^\top x_i)\right)^{1+\frac{1}{k}}}{k}. \end{aligned} \tag{15}$$

Now, note that $r_i > 1$ in the above equation equation 15 if and only if,

$$r_i > 1 \iff \left(1 + \exp(-y_i w^\top x_i)\right)^{1+\frac{1}{k}} > k \iff \left(1 + \frac{1}{k}\right) \log\left(1 + \exp(-y_i w^\top x_i)\right) > \log(k)$$

Note that the last inequality in the above equation is satisfied if

$$\left(1 + \frac{1}{k}\right) \cdot \log\left(1 + \exp(-y_i w^\top x_i)\right) > \log\left(1 + \exp(-y_i w^\top x_i)\right) > \log(k). \tag{16}$$

Using the last inequality, we see that $r_i > 1$ if $k \leq \left(1 + \exp(-y_i w^\top x_i)\right)$, and we have the desired result. This concludes the proof of Lemma 3.2 in the main paper. □

Furthermore, by summing over the dataset or indices i, and minimizing over w we can get an upper bound on k in terms of the total loss function also. With this we can say that RLO is strictly better than LR objective from the optimization perspective.

A.2 Proof of Lemma 3.3.

From equation 10 and equation 13, we can write the hessian matrix functions $\nabla^2 \mathcal{L}_{LR}, \nabla^2 \mathcal{L}_{RLO}^k$ of Logistic Regression in equation 2, and equation 4 as,

$$\nabla^2 \mathcal{L}_{LR}(w) = X D(w) X^T, \quad \nabla^2 \mathcal{L}_{RLO}(w) = X D^k(w) X^T, \tag{17}$$

where $D(w), D^k(w) \in \mathbb{R}^{d \times d}$ are diagonal matrices with i-th diagonal entries $d_i(w), d_i^k(w)$ equal to the scalar coefficients in equation 10 and equation 13. By assumption, and Lemma 3.2, we know that $d_i^k > d_i$. This means that the matrix $\Delta^k := D^k - D \succ 0$ is a positive definite matrix i.e., diagonal entries given by $\delta_i := d_i^k - d_i > 0$ are positive for any w. To avoid notation clutter, we will just use D, D^k for the diagonal matrices while keeping in mind that they are not constant wrt parameters w.

The variational definition of smallest eigenvalue $\lambda_{\min}$ of the hessian matrices are given by,

$$\begin{aligned} \lambda_{\min, RLO} = \min_{z:\|z\|_2^2=1} z^\top X D^k X^\top z &= \min_{z:\|z\|_2^2=1} z^\top X (D + \Delta^k) X^\top z \\ &= \min_{z:\|z\|_2^2=1} \left(z^\top X D X^\top z + z^\top X \Delta^k X^\top z \right) \\ &\geq \min_{z:\|z\|_2^2=1} \left(z^\top X D X^\top z \right) + \min_{z:\|z\|_2^2=1} \left(z^\top X \Delta^k X^\top z \right) \\ &= \lambda_{\min, LR} + \underline{\delta}^k > \lambda_{\min, LR}, \end{aligned} \tag{18}$$

where we used the fact that $\min_z [f(z) + g(z)] \geq \min_z f(z) + \min_z g(z)$ for any two (smooth, nonnegative) functions f and g, to obtain the inequality in equation 18. Similarly, by using the variational definition of largest eigenvalue along with the fact that $\max_z [f(z) + g(z)] \leq \max_z f(z) + \max_z g(z)$ for any two (smooth, nonnegative) functions f and g, we can show that $\lambda_{\max,RLO} < \lambda_{\max,LR}$. We can obtain the desired result in Lemma 3.3 by noting that,

$$\kappa\left(\nabla^2 \mathcal{L}^k_{RLO}(w)\right) = \frac{\lambda_{\max,RLO}}{\lambda_{\min,RLO}} < \frac{\lambda_{\max,LR}}{\lambda_{\min,RLO}} < \frac{\lambda_{\max,LR}}{\lambda_{\min,LR}} = \kappa\left(\nabla^2 \mathcal{L}_{LR}(w)\right). \tag{19}$$

This concludes the proof of Lemma 3.3 in the main paper.

A.3 Pseudocode for RLO

We provide the example codes of our proposed RLO in Figure 4. Figure 4a is to calculated rooted loss and gradients following Eq 4 and 6. Figure 4b is to obtain rooted loss and can be optimized by any optimizer in PyTorch, which is easy to use for training deep neural networks.

```
def root_grad(x):
    temp = 1. / (1. + np.exp(b * np.dot(A, x)))
    templ = np.power(1. + np.exp(-b * np.dot(A, x)), 1/self.kparam)
    grad = - np.dot(A.T, b * temp * templ) / n + lbda * x
    return grad

def root_loss(x):
    bAx = b * np.dot(A, x)
    return self.kparam * (np.mean(np.power(1. + np.exp(- bAx),1/kparam))) + lbda * norm(x) ** 2 / 2.
```

(a) Example code to calculate rooted loss and gradients.

```
def root_loss(output, target, k, m):

    n = target.shape[0]
    prob = F.softmax(output, dim=1)
    root = torch.pow((prob[range(n), target]), 1 / k)
    root = m * (1 - root)
    loss = torch.mean(root)

    return loss
```

(b) Example code for using PyTorch optimizer.

Figure 4: The example code block to use our proposed RLO.

A.4 Experiment Details

In this section, we provide additional experiments of our intuitions mentioned in Section 3.1. Moreover, we demonstrate the details of the experiment corresponding to Section 4.

A.4.1 Intuition: Shallow Logistic regression vs Rooted Logistic regression

Datasets: The empirical studies are conducted on the following 4 benchmark classification datasets, which can be found in the publicly available UCI machine learning repository ([Asuncion and Newman, 2007]): Wine, Ionosphere, Madelon and Specheart, see Table 6.

Experiments setups: The baseline is standard logistic regression. To showcase the benefits of RLO, we run the experiments with different numbers of k$\in [3, 20]$ for the proposed rooted logistic regression. Note that, for all the datasets except Specheart, we use the same number of iterations (200) and learning rate (0.01) for all the experiment settings. For Specheart, we increase the number of iterations to 1000 for better convergence and higher accuracy. We also evaluate standard logistic regression as well as RLO with/without ℓ_2 regularization. The Wine dataset contains 3 classes, so we use the One-vs-All approach for binary classification. The reported results are obtained by averaging the results from these three separate binary classification tasks. All the other

Dataset	#Samples	#Attributes
Wine	178	13
Specheart	267	44
Ionosphere	351	34
Madelon	4400	500

Table 6: Dataset information for regression.

datasets contains 2 classes, and undergo the standard binary classification task using the 4 regression methods. The classification performance for all the different settings is performed using K-fold cross-validation, with the number of folds being 5. The results shown are averaged across the 5 folds.

Results: Table 7 shows the test accuracy for all the datasets under the different regression settings. For RLO, we also show the top 3 k values which achieved the highest accuracy. As seen in the table, for all the datasets, RLO with/without ℓ_2 regularization outperforms standard logistical regression with/without ℓ_2 in term of accuracy on test set. Specifically, RLO with ℓ_2 regularization consistently achieves higher accuracy rates for different values of k. Hence, we conclude that our proposed RLO is beneficial to accelerate the training and also provide improvements. Figure 5 shows the convergence performance for all the top 3 performing k values of RLO, with/without ℓ_2 regularization, compared to LR with and without ℓ_2. Again, it clear that RLO helps in faster convergence, compared to the standard logistic regression for Wine, Ionosphere, Madelon and Specheart datasets.

	LR	LR - L2		RLO	RLO - L2
Dataset	Test Acc.	Test Acc.	k	Test Acc.	Test Acc.
Wine	90 ± 4.15	89.44 ± 5.66	3	**97.22 ± 1.75**	94.55 ± 3.51
			11	82.77 ± 1.23	**95.55 ± 2.22**
			13	91.66 ± 5.55	**95 ± 5.09**
Ionosphere	81.4 ± 2.73	83.94 ± 2.1	4	85.07 ± 1.12	**86.47 ± 1.12**
			3	**86.47 ± 1.69**	85.63 ± 0.56
			16	84.5 ± 0.00	**86.19 ± 0.56**
Madelon	52.03 ± 1.9	50.83 ± 1.51	6	**54.36 ± 0.71**	52.75 ± 0.97
			9	**54.13 ± 0.58**	51.8 ± 1.43
			19	52.36 ± 1.14	**54.13 ± 1.42**
Specheart	80.49 ± 3.92	88.25 ± 1.69	20	84 ± 3.10	**88.5 ± 1.83**
			15	82.75 ± 1.83	**88 ± 1.49**
			13	82.99 ± 2.44	**88 ± 1.00**

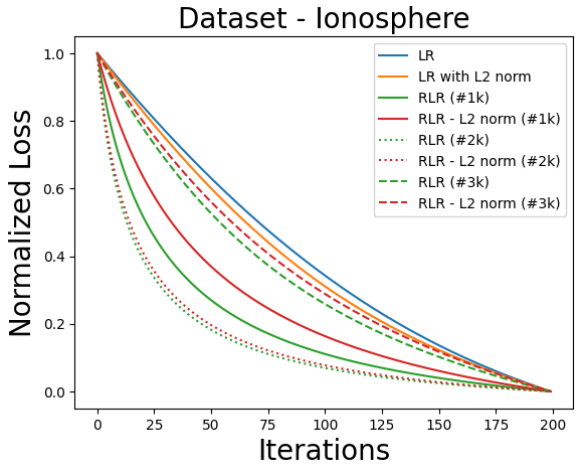
Table 7: Testing Accuracy from 5-Fold Cross Validation, using Shallow Logistic regression vs Rooted Logistic regression (RLO). Top 3 values of k are shown for RLO. RLO with/without ℓ_2 regularization outperforms Shallow Logistic regression with/without ℓ_2, in term of accuracy on test sets of all 4 datasets.

A.4.2 DNNs for Classification with RLO

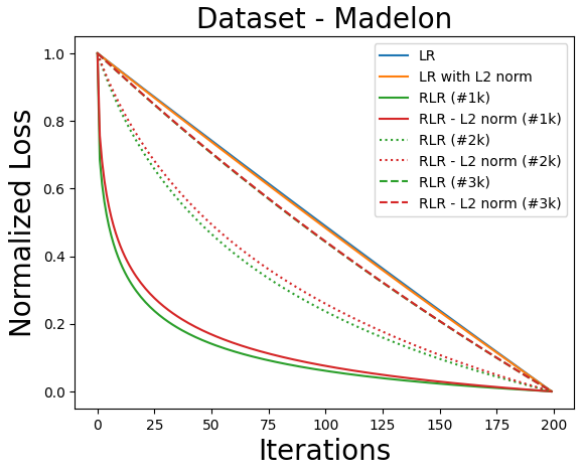
Dataset: We conduct image classification experiments to test the performance of RLO. In particular, we use CIFAR-10/100 for training from the scratch, and Tiny-ImageNet, ImageNet1k and Food-101 ([Bossard et al., 2014])for finetuning. The dataset information are shown in Table 8

Dataset	#Images	#Image size	#Classes
CIFAR-10	60,000	32	10
CIFAR-100	60,000	32	100
Tiny-ImageNet	100,000	64	200
Food-101	101,000	512	101
ImageNet1k	1281167	256	1000

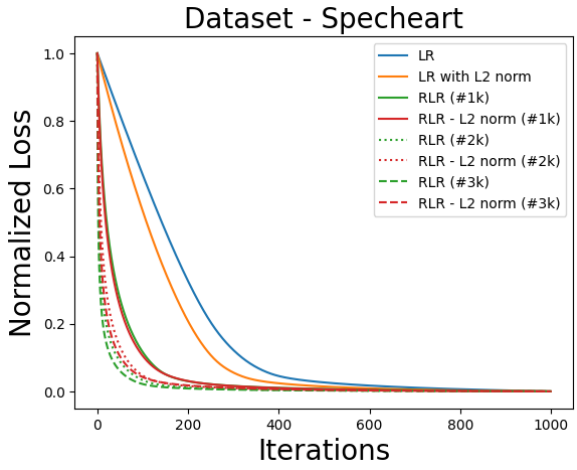
Table 8: Dataset information for image classification.



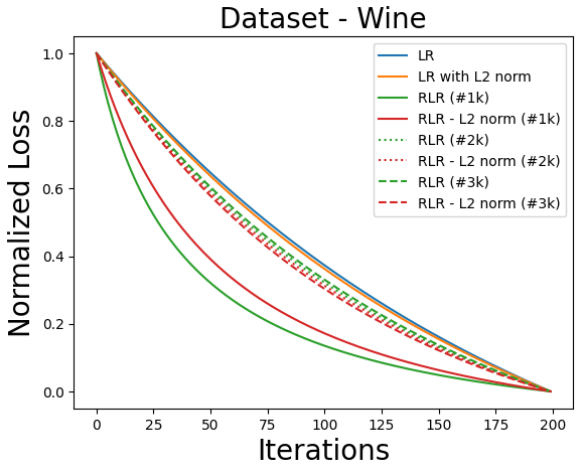
(a) Ionosphere



(b) Madelon



(c) Specheart



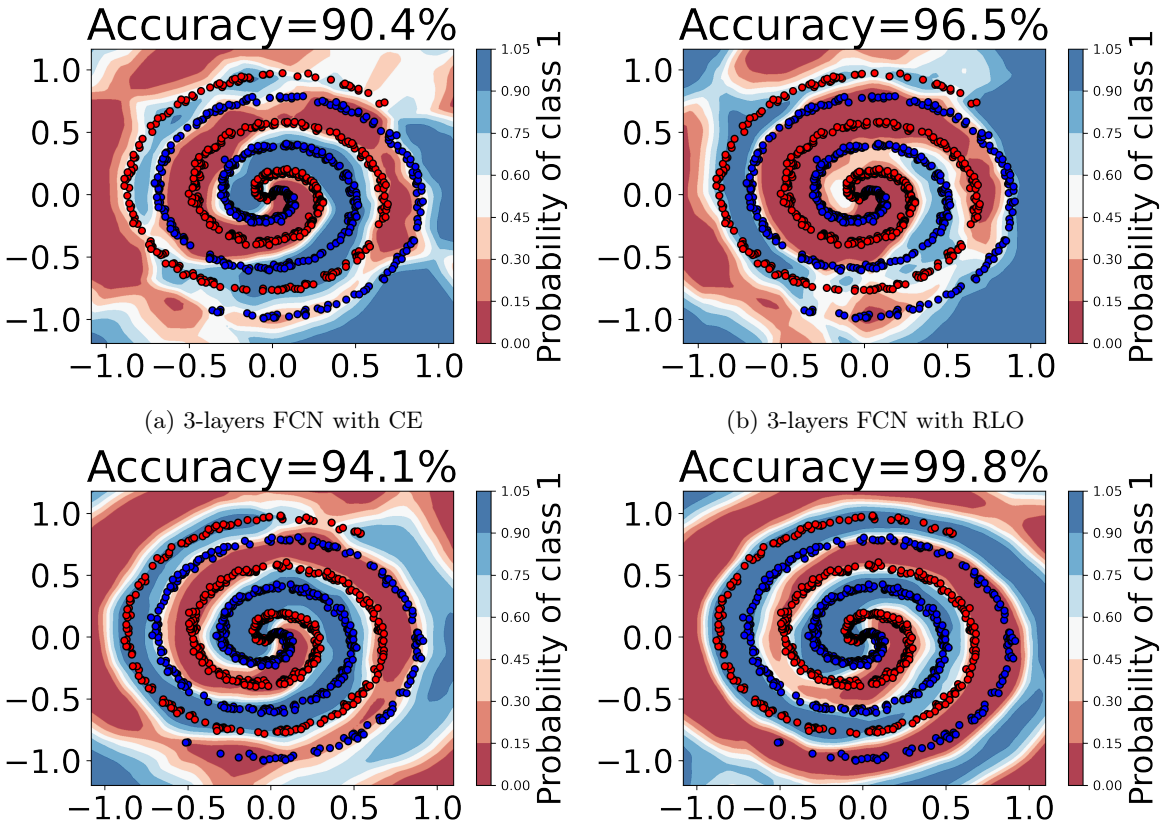
(d) Wine

Figure 5: The rate of convergence over iterations of standard logistic regression and rooted logistic regression, for the train sets of Ionoshphere, Madelon, Specheart and Wine. The lines for the rooted logistic regression show the convergence for the value of top 3 $k's$ which gives the best test accuracy, as mentioned in Table 7. For RLO, solid lines show the normalized loss with the best k value. The dotted and dashed lines of the same color depict the second and the third performing k value.

Implementation details: In the synthetic settings for binary classification, we implemented three different layers (2, 3, 4) fully-connected neural networks (FCN). The training iterations are 1000, 100, and 50 respectively. We use the same hidden size of 100, learning rate as 0.01 and k of 3 for three FCNs.

For the vision models in image classification tasks, as multi-class classification, we train and finetune on ViT-B ([Dosovitskiy et al., 2020]), ResNet-50 ([He et al., 2016]), and Swin-B ([Liu et al., 2021]) models. The k parameters of our proposed RLO are chosen from the set {5, 8, 10}. We train the three models on CIFAR-10 and CIFAR-100 for 200 epochs with ViT and 100 epochs with ResNet and Swin. The batch size is 512 and learning rate is 1e-4. Moreover, we finetune the models which pre-trained on ImageNet ([Deng et al., 2009]) on Tine-ImageNet and Food-101 for 10 epochs with batch size of 256 and learning rate of 1e-5. We use AdamW optimizer designed by ([Loshchilov and Hutter, 2018]). In addition, we apply RandAugment ([Cubuk et al., 2020]) as augmentation strategy in finetuning steps. We train and fine-tune both on 3 NVIDIA RTX 2080Ti GPUs. To evaluate the effectiveness of our proposed RLO, we use cross-entropy (CE) loss and focal loss as baselines.

Results of FCN: As results shown in Figure 6, in (a) and (b), we train a 3-layer FCN for 100 iterations with CE and RLO. In (c) and (d), we train a 4-layer FCN for 50 iterations with CE and RLO. RLO outperforms CE in term of accuracy in all settings. The decision boundaries are larger in RLO, which indicates that RLO is beneficial to separate data points.



(a) 3-layers FCN with CE

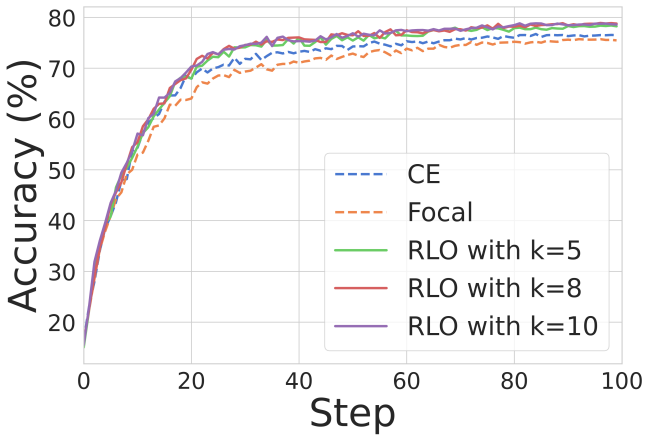
(b) 3-layers FCN with RLO

(c) 4-layers FCN with CE

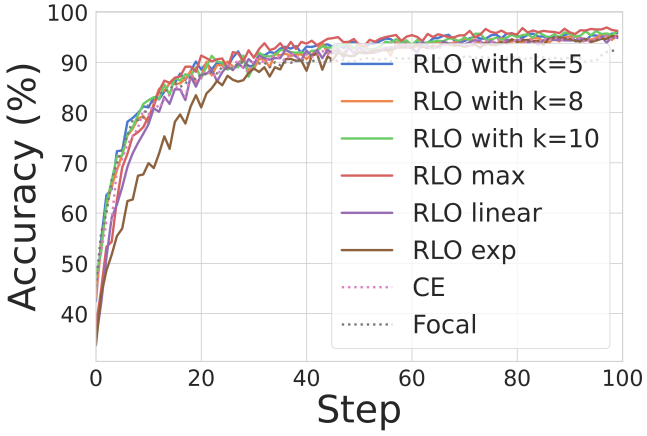
(d) 4-layers FCN with RLO

Figure 6: The color demonstrated the estimated probability of a class label being identified as 1, as aligned with the scale located on the right side in the figures. The intervening white line between the red and blue regions denotes the decision boundary.

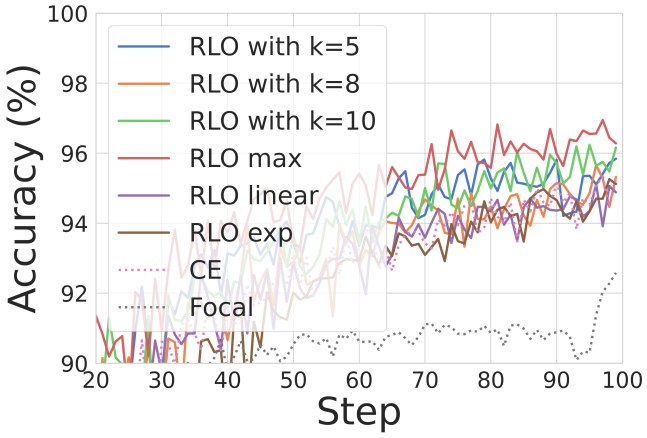
Results of Image Classification: We include more detailed on training accuracy over epochs with different models and setups of k values. As results shown in Figure 7, RLO outperforms CE and Focal consistently in term of accuracy on various datasets. Moreover, Figure 7c indicates that all setups of k can achieve better performance and RLO with k of exponential increases up to the maximum value followed Lemma 3.2 produced the highest accuracy. It provides the therotical and practical insights of choosing k values.



(a) SWIN on CIFAR-100



(b) ResNet50 on CIFAR-10



(c) ResNet50 on CIFAR-10 after 20 epochs

Figure 7: RLO performance on image classfication tasks with different models. We compare RLO with CE and Focal, and different setups of k values. RLO outperforms CE and Focal on different models and dataets. RLO with k of exponential increases up to the maximum value performs the best with ResNet-50 on CIFAR-10.

A.4.3 LLMs Qantization with RLO

Implementation details: For finetuning OPT [Zhang et al., 2022], we use the pre-trained models from huggingface [1]. We integrate RLO to trainer to fairly compare with CE. Batch size is 16 and max sequence length is 256. We use learning rate as 1e-05 with linear schedule and AdamW optimizer. We finetune all models with RLO and CE for 3 epochs. For quantization on the finetuned models with QUIP [Chee et al., 2024], we use ldlqRG as quantization method with incoherence processing. For Finetuning with quantizer with RLO on Llama [Touvron et al., 2023]: we finetune quantized Llama family with CE and RLO on Wikitext2 with LLMTools [Yin et al., 2024], and we follow the parameters and settings same as LLMTools [2].

A.5 Applying RLO for Generative Models

Similarly, the landscape of objective function that are used for generating or sampling points similar to x have also been under investigation [Qi, 2020]. For this task, various architectures have been proposed with (discrete) convolution or smoothing operators as the building blocks, such as DCGAN [Radford et al., 2015], BigGAN [Brock et al., 2018], StyleGAN [Karras et al., 2020b]. These smoothing based architectures called Generators gradually transform a random signal to $\tilde{x}$, a "fake" or synthetic sample. Then, a classification architecture called Discriminator is used to assign the probability of $\tilde{x}$ being a real sample from $\mathcal{D}$. While separability might not be the deciding factor in training the overall models, conditioning of loss functions used to train the Discriminator is crucial in determining the success of first order algorithms, and thereby the sampling process to obtain $\tilde{x} \sim x \in \mathcal{D}$ [Arora et al., 2017].

Generative models were studied as *statistical* problem where the goal is, given a training dataset $x_i, i = 1, 2...n$, learn a parametric model of its distribution $p(x)$. For an appropriate parametric model f_θ, we need θ such that $f_\theta(z) \approx x$, where z is usually a Gaussian vector to approximate some x_i through the transformation f_θ. For sampling, given a mapping f_θ, synthetic data points can be generated by sampling a Gaussian vector z and computing $f_\theta(z)$. This overcomes some of the architectural restrictions of f_θ. This property is leveraged to come up with Generative Adversarial Networks (GANs), see Chapter 10 in [Lindholm et al., 2022].

GANs are a class of models that help the synthesize data points from the model using f_θ which gets a Gaussian vector z as an input. GANs are trained by comparing these synthetic samples with real samples from the training data x_i. The comparison is done by a critic, e.g., a binary classifier g_η which judges the authenticity of the samples. It is an adversarial game where the generator's parameters θ are continuously updated to synthesize data close to reality while the classifier such as the discriminator wants to label them correctly as fake. The result is a generator that has successfully learned to generate data that the discriminator labels as real. The generator tries to maximize the classifier loss with respect to θ while the classifier tries to minimize the loss with respect to η. This leads to a rooted minmax problem with loss that is similar equation 4, written as,

$$\min_\theta \max_\eta V_k(f_\theta, g_\eta) = \mathbb{E}_{x\sim p_{data}(x)}[k\left(g_\eta(x)\right)^{1/k}] \tag{20}$$

$$+\mathbb{E}_{z\sim p_z(z)}[k\left(1 - g_\eta(f_\theta(z))\right)^{1/k}]. \tag{21}$$

It is possible to modify our GAN loss to incorporate regularization terms [Kurach et al., 2019] such as Gradient norm [Wu et al., 2021, Fang et al., 2022], Jacobian [Nie and Patel, 2020] to further stabilize the parameters during training the generative model.

A.5.1 Experiment details of GANs with RLO

Datast: For our image generation experiments with StyleGAN, we use FFHQ dataset [Karras et al., 2018] and the Stanford Dogs dataset [Khosla et al., 2011], see details in Table 9.

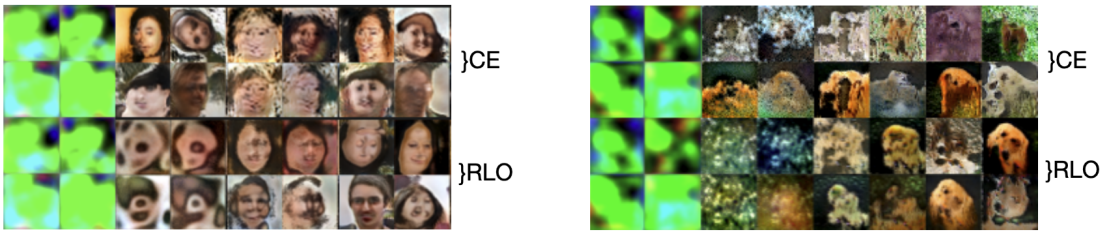
Image generation setup: we use the version of StyleGAN capable of being trained by limited training data, as proposed by [Karras et al., 2020a]. We evaluate the effectiveness of RLO by replacing the original loss and compare it to StyleGAN's CE loss, for different values of k. To compare the efficacy of the models trained using RLO and CE loss, we take a large image from the original dataset, and compute its projection on the latent

[1] https://huggingface.co/facebook

[2] https://github.com/kuleshov-group/llmtools

Dataset	#Images	#Image size	#Classes
Stanford Dogs	20,580	-	120
FFHQ	70,000	1024	-

Table 9: Dataset information for GAN.



(a) FFHQ - Progressive Generation

(b) Stanford Dogs - Progressive Generation

Figure 8: Results on FFHQ dataset and Stanford Dogs dataset. (a) FID score vs training time for both cross-entropy loss and RLO-2 setup. (c) FID score vs training time for both cross-entropy loss and RLO-11 setup. In (b) and (d), the top 8×2 row contains four instances of the image generation (Each image is a part of the 2×2 grid containing four images) using CE loss. The bottom 8×2 represents the same with the RLO setup at the same instances.

space using our model snapshots from the initial and final stages of the training. We then use these projections to generate an image using their respective models.

All training is run on 3 NVIDIA RTX 2080Ti GPUs, using 200 64×64 dimension images from the FFHQ dataset, and 55 64×64 images from the Stanford Dogs dataset. We use a mini-batch of 32 images and learning rate of 1e-4, for both the baseline, as well as our setup. We further evaluate the efficacy of RLO by replacing the original loss with our derived rooted loss function and compare it to StyleGAN's cross-entropy loss, for different values of k. To compare the efficacy of the models trained using RLO and cross-entropy loss, we take a large image from the original dataset, and compute its projection on the latent space using our model snapshots from the initial and final stages of the training. We then use these projections to generate an image using their respective models.

A.5.2 Results of GANs with RLO

The setup with RLO trained on the FFHQ dataset produces a lower FID of 81.2, which means better quality images, than the one with the CE with FID of 89.7. The progressive image generation while training these models are illustrated in Figure 8a. Images produced by RLO (bottom 2 rows) seems to be slightly better at the final stages of the training. Similar FID vs time comparison and progressive image generation for the Stanford Dogs dataset is shown in Figure 8, where the FID scores for the two models are close. Finally, in Figure 9, for a target image, we show the images obtained from the projections using the initial and final stages of training. For both FFHQ and Stanford Dogs dataset, we can see that the images generated using the final stage RLO models (bottom image, in the last column), produce details that are closer to the target image than CE. More generated images are shown in Appendix A.5.2.

We demonstrate more progressively generated images in Figure 10 for FFHQ dataset. We show the generated images with different values of k and also compare to the cross-entropy which is our baseline. The results show that our proposed RLO is able to generate high-quality images with training on limited data.

(a) Results with FFHQ Dataset.

(b) Results with Stanford Dogs Dataset

Figure 9: For each setup, the target image can be seen on the left-most image. To its right, the first row shows the generated images with the projection obtained from the initial and final stages of the training respectively, with CE. The second row is the result of replacing it with RLO.

(a) Cross-entropy based training

(b) RLO - 2

(c) RLO - 8

(d) RLO - 11

Figure 10: The progressive generation of images shown with cross-entropy loss and RLO at different values of k using the FFHQ dataset.