Reducing Confidence Along Adversarial Directions: Maximizing Entropy on Self-Generated Perturbations

Anonymous Author(s)
Affiliation
Address
email

Abstract

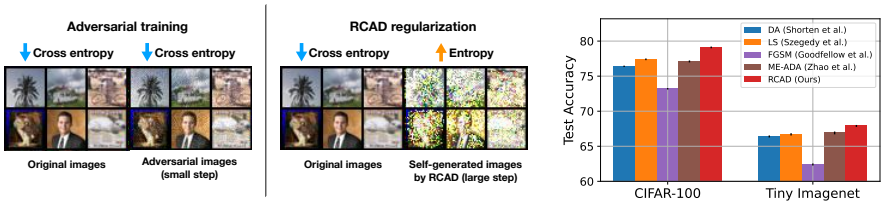
Supervised learning methods trained with maximum likelihood objectives often overfit on training data. Most regularizers that prevent overfitting look to increase confidence on additional examples (e.g., data augmentation, adversarial training), or reduce it on training data (e.g., label smoothing). In this work we propose a complementary regularization strategy that reduces confidence on self-generated examples. We call our regularizer RCAD: Reducing Confidence along Adversarial Directions, and the key idea behind it is to reduce confidence on out-of-distribution examples lying along directions adversarially chosen to increase training loss. In contrast to adversarial training, our method does not try to robustify the model to output the original label, but rather regularizes it to have reduced confidence on points generated using much larger perturbations than in conventional adversarial training. RCAD can be easily integrated into training pipelines with a few lines of code. Despite its simplicity, we find on many classification benchmarks that RCAD can be added to existing techniques (e.g., label smoothing, MixUp training) to increase test accuracy by 1–3% in absolute value, with more significant gains in the low data regime. We also provide a theoretical analysis that helps to explain these benefits in simplified settings, showing that RCAD can provably help the model unlearn spurious features in the training data.

1 Introduction

The standard supervised learning formulation aims to train models to make accurate predictions on fresh test examples drawn from the same distribution as training data. Unfortunately, it is well known that maximizing the likelihood of the training data alone may result in overfitting. Prior work broadly considers two approaches to combat this issue. Some methods train on additional examples, e.g., generated via augmentations [44, 7, 56] or adversarial updates [11, 28, 2]. Others modify the objective by using alternative losses and/or regularization terms (e.g., label smoothing [47, 31], MixUp [59], robust objectives [55, 17]). In effect, these prior approaches either make the model's predictions more certain on new training examples or make the distribution over potential models less certain.

Existing regularization methods can be seen as providing a certain inductive bias for the model, e.g., the model's weights should be small (i.e., weight decay), the model's predictions should vary linearly between training examples (i.e., MixUp). In this paper we identify a different inductive bias: the model's predictions should be less confident on out-of-distribution inputs that look nothing like the training examples. We turn this form of inductive bias into a simple regularizer, whose benefits are complementary to existing regularization strategies. To instantiate such a method, we must be able to sample out-of-distribution examples. For this, we propose a simple approach: generating adversarial examples [30, 11] using very large step sizes (orders-of-magnitude larger than traditional adversarial training [28]). Hence, we first perturb the training points using the training loss gradient, and then maximize predictive entropy on these adversarially perturbed examples.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.



(a) (b)

Figure 1: **Reducing confidence along adversarial directions** (RCAD) is a simple and efficient regularization technique to improve test performance. *(Left)* For RCAD, examples are generated by taking a large step ($10\times$ typical for adversarial examples) along the gradient direction. We see that generated images thus look very different from the original, with accentuated spurious components responsible for the model's flipped predictions on adversarial images. *(Right)* RCAD achieves greater test accuracy than data augmentation (DA), label smoothing (LS), and methods that minimize cross-entropy on adversarial examples: adversarial training via FGSM [11] and ME-ADA [63].

In contrast to adversarial training, our method does not try to robustify the model to output the original label, but rather regularizes it to have reduced confidence on examples generated via a large step size (Figure 1a). As shown in Figure 1b, this can lead to significant improvements in in-distribution test accuracy unlike adversarial training [28, 29, 3, 2], which tends to decrease in-distribution test performance [39, 60, 52]. Compared with semi-supervised learning methods [12, 54], our method will not require an additional unlabeled dataset (it generates one automatically), and different from prior works [54, 64] it will also train the model to be less confident on these examples.

As the self-generated samples in our procedure no longer resemble in-distribution data, we are uncertain of their labels and train the model to predict a uniform distribution over labels on them, following the principle of maximum entropy [20]. A few prior works [47, 35, 43, 8] have also considered using the same principle to prevent the memorization of training examples, but only increase entropy on *iid* sampled *in-distribution* labeled/unlabeled data. In contrast, we study the effect of maximizing entropy on self-generated *out-of-distribution* examples along adversarial directions.

The main contribution of this work is a training procedure we call RCAD: Reducing Confidence along Adversarial Directions. RCAD improves test accuracy (for classification) and log-likelihood (for regression) across multiple supervised learning benchmarks. Importantly, we find that the benefits of RCAD are complementary to prior methods: Combining RCAD with alternative regularizers (e.g., augmentation [44, 56], label smoothing [47], MixUp training [59]) further improves performance. Our method requires adding $\sim$5 lines of code and is computationally efficient (with training time at most $1.3\times$ standard). We provide a theoretical analysis that helps to explain RCAD's benefits in a simplified setting, showing that RCAD unlearns spurious features in the training data, thereby improving accuracy on unseen examples.

2 Related Work

Below we survey common regularization techniques in machine learning, as well as other methods that utilize entropy maximization in training for various purposes (differing from our own).

Data augmentation. A common strategy to improve test accuracy (particularly on image tasks) is to augment the training data with corruptions [44] or surface level variations [7, 56] (e.g., rotations, random crops). Some methods [29, 28, 11] further augment data with imperceptibly different adversarial examples or interpolated samples [59]. Others [37, 34] sample new examples from generative models that model the marginal density of the input distribution. Also related are semi-supervised methods, which assume access to an extra set of unlabeled data and train the model to have more confident predictions on them [12, 29]. All these methods *minimize entropy* [13] on the augmented or unlabeled data to improve generalization [59, 63]. In contrast, our method *maximizes entropy* on perturbed samples along the adversarial direction.

Adversarial training. Deep learning models are vulnerable to worst-case perturbations that can flip the model's predictions [11, 41]. Adversarial training was proposed to improve robustness to such attacks by reducing worst-case loss in small regions around the training samples [28, 25, 39]. More recently, Zhao et al. [63] proposed maximum entropy adversarial data augmentation (ME-ADA), which uses an information bottleneck principle to identify worst-case perturbations that both maximize training loss and predictive entropy. Similar to adversarial training ME-ADA still

Table 1: **Regularization objectives**: We summarize prior works that employ adversarial examples or directly regularize model's predictions $p_{\boldsymbol{w}}(\mathrm{y} \mid \mathbf{x})$ along with the scalar hyperparameters (in $[\cdot]$) associated with each.

name	objective
cross entropy	$\min_{\boldsymbol{w}} -\sum_{\mathbf{x},\mathrm{y}\in\hat{\mathcal{D}}} \log p_{\boldsymbol{w}}(\mathrm{y} \mid \mathbf{x})$
label smoothing $[\epsilon]$ [31]	$\min_{\boldsymbol{w}} -\sum_{\mathbf{x},\mathrm{y}\in\hat{\mathcal{D}}} \Big((1-\epsilon)\log p_{\boldsymbol{w}}(\mathrm{y} \mid \mathbf{x}) + \sum_{y'\neq \mathrm{y}} \frac{\epsilon}{\lvert\mathcal{Y}\rvert-1} \log p_{\boldsymbol{w}}(y' \mid \mathbf{x})\Big)$
Adv. training $[\alpha]$ [28]	$\min_{\boldsymbol{w}} -\sum_{\mathbf{x},\mathrm{y}\in\hat{\mathcal{D}}} \log p_{\boldsymbol{w}}(\mathrm{y} \mid \mathbf{x} - \alpha\cdot\mathrm{sign}(\nabla_{\mathbf{x}} \log p_{\boldsymbol{w}}(\mathrm{y} \mid \mathbf{x})))$
ME-ADA $[\alpha, \beta]$ [63]	$\min_{\boldsymbol{w}} -\sum_{(\mathbf{x},\mathrm{y})\in\hat{\mathcal{D}}\cup\hat{\mathcal{D}}'} \log p_{\boldsymbol{w}}(\mathrm{y} \mid \mathbf{x})$ where, for a distance metric $C_{\boldsymbol{w}}{:}(\mathcal{X}\times\mathcal{Y})\times(\mathcal{X}\times\mathcal{Y})\mapsto\mathbb{R}$ $\hat{\mathcal{D}}' \triangleq \{(\tilde{\boldsymbol{x}},\mathrm{y}) \mid \tilde{\boldsymbol{x}} \triangleq \sup_{\boldsymbol{x}_0} -\log p_w(\boldsymbol{x}_0 \mid \mathrm{y}) + \alpha\mathcal{H}_{\boldsymbol{w}}(\boldsymbol{x}_0) - \beta C_{\boldsymbol{w}}((\boldsymbol{x}_0,\mathrm{y}),(\mathbf{x},\mathrm{y})),\ \forall(\mathbf{x},\mathrm{y})\in\hat{\mathcal{D}}\}$
RCAD (ours) $[\alpha, \lambda]$	$\min_{\boldsymbol{w}} \sum_{\mathbf{x},\mathrm{y}\in\hat{\mathcal{D}}} \left(-\log p_{\boldsymbol{w}}(\mathrm{y} \mid \mathbf{x}) - \lambda\cdot\mathcal{H}_{\boldsymbol{w}}(\mathbf{x} - \alpha\cdot\nabla_{\mathbf{x}} \log p_{\boldsymbol{w}}(\mathrm{y} \mid \mathbf{x}))\right)$

minimizes cross-entropy to output the same label on new points. There are two key differences between above methods and RCAD; *(i)* rather than minimizing cross entropy loss on adversarial examples, we *increase* model's uncertainty on the self-generated examples; and *(ii)* we take much *larger* steps—so large that unlike adversarial images, the generated example is no longer similar to the original one (Figure 1a). We show that these differences are important to improve test performance since adversarial training has been shown to hurt generalization on non-adversarial in-distribution test points [39, 51, 60]. Further, RCAD has a lower (at most $1/5^{\text{th}}$) computational cost compared to multi-step adversarial training procedures [63, 60].

Robust objectives. In order to improve robustness to noise and outliers in the data, a common approach is to modify the objective by considering risk averse loss functions [e.g., 17, 40, 48] or incorporating regularization terms such as the l_2/l_1 norms [24, 50]. Our method is similar to these approaches in that we propose a new regularization term. However, whereas most regularization terms are applied to the model's weights or activations, ours is directly applied to the model's predictions. Another effective loss function for classification problems is label smoothing [47, 6, 21], which uniformly increases model's predictive uncertainty in a region around training samples [9]. In contrast, RCAD increases entropy only on examples generated along the adversarial direction that has been shown to comprise of spurious features [18, 5], thereby *unlearning* them.

Entropy maximization. Finally, we note that our work builds upon prior work that draws connections between entropy maximization in supervised [43, 8, 35] and reinforcement learning [14]. For example, Pereyra et al. [35] apply a hinge form of confidence penalty directly on training data which is similar to label smoothing in principle and performance (Table 2 in [35]). Other works like [8, 43] also adapt the principle of entropy maximization but do so either on additional unlabeled data or a subset of the training samples. More recently [36] show that maximizing entropy on interpolated samples from the same class improves out-of-distribution uncertainty quantification. In contrast to the above, we minimize cross-entropy loss on training data and only maximize entropy on samples generated along the adversarial direction. Our experimental results also span a wider set of benchmarks and presents significant gains $+1$–3% complementary to methods like label smoothing. We also theoretically analyze our objective for the class of linear predictors and show how RCAD can mitigate vulnerability to spurious correlations.

3 RCAD: Reducing Confidence Along Adversarial Directions

We now introduce our regularization technique of reducing confidence along adversarial directions (RCAD). This section will only describe the objective and an algorithm for optimizing it; Section 5 presents a more formal discussion of RCAD and in Section 4 we provide our empirical study.

Notation. We are given a *training* dataset $\hat{\mathcal{D}} \triangleq \left\{(\mathbf{x}^{(i)}, \mathrm{y}^{(i)})\right\}_{i=1}^{N}$ where $\mathbf{x}^{(i)} \in \mathcal{X}$, $\mathrm{y}^i \in \mathcal{Y}$, are sampled *iid* from a joint distribution $\mathcal{D}$ over $\mathcal{X}\times\mathcal{Y}$. We use $\hat{\mathcal{D}}$ to denote both the training dataset and an empirical measure over it. The aim is to learn a parameterized distribution $p_{\boldsymbol{w}}(\mathrm{y} \mid \mathbf{x}),\ \boldsymbol{w}\in\mathcal{W}$ where the learnt model is typically obtained by maximizing the log-likelihood over $\hat{\mathcal{D}}$. Such a solution is referred to as the maximum likelihood estimate (MLE): $\hat{\boldsymbol{w}}_{\mathrm{mle}} \triangleq \arg\max_{\boldsymbol{w}\in\mathcal{W}} \mathbb{E}_{\hat{\mathcal{D}}} \log p_{\boldsymbol{w}}(\mathrm{y} \mid \mathbf{x})$. In the classification setting, we measure the performance of any learned solution $\hat{\boldsymbol{w}}$ using its accuracy on the *test* (population) data: $\mathbb{E}_{\mathcal{D}}\left[\mathbb{1}(\arg\max_{y'} p_{\hat{\boldsymbol{w}}}(y' \mid \mathbf{x}) = \mathrm{y})\right]$.

Solely optimizing the log-likelihood over the training data can lead to poor test accuracy, since the estimate $\hat{\boldsymbol{w}}_{\mathrm{mle}}$ can overfit on noise in $\hat{\mathcal{D}}$ and fail to generalize to unseen samples. Many algorithms aim to mitigate overfitting by either designing suitable loss functions replacing the log-likelihood objective, or by augmenting the training set with additional data (see Section 2). Our main contribution

is a new data-dependent regularization term that will depend not just on the model parameters but also on the training dataset. We reduce confidence on out-of-distribution samples that are obtained by perturbing the original inputs along the direction that adversarially maximizes training loss.

In the paragraphs that follow we describe the methodology and rationale behind our objective that uses the following definition of model's predictive entropy when $\mathcal{Y}$ is a discrete set: $\mathcal{H}_{\boldsymbol{w}}(x) \triangleq -\sum_{y \in \mathcal{Y}} p_{\boldsymbol{w}}(y \mid \mathbf{x}) \log p_{\boldsymbol{w}}(y \mid \mathbf{x})$. In cases where $\mathcal{Y}$ is continuous, for example in regression tasks, we will use the differential form of predictive entropy: $-\int_{\mathcal{Y}} p_{\boldsymbol{w}}(y \mid \mathbf{x}) \log p_{\boldsymbol{w}}(y \mid \mathbf{x})\, dy$.

Reducing Confidence Along Adversarial Directions. The key idea behind RCAD is that models should not only make accurate predictions on the sampled data, but also make uncertain predictions on examples that are very different from training data. We use directions that adversarially maximize the training loss locally around the training points to construct these out-of-distribution examples that are different from the training data. This is mainly because adversarial directions have been known to comprise of spurious features [18] and we want to regularize the model in a way that makes it uncertain on these features. We first describe how these examples are generated, and then describe how we train the model to be less confident.

RCAD: Reducing Confidence along Adversarial Directions

```
def rcad_loss(x, y, α, λ):
  loss = − model(x).log_prob(y)
  x_adv = x + α * loss.grad(x)
  entropy = model(x_adv).entropy()
  return loss - λ * entropy
```

We generate an out-of-distribution example $\tilde{\mathbf{x}}$ by taking a large gradient of the MLE objective with respect to the input $\mathbf{x}$, using a step size of $\alpha > 0$:

$$\tilde{\mathbf{x}} \triangleq \mathbf{x} - \alpha \cdot \nabla_{\mathbf{x}} \log p_{\boldsymbol{w}}(\mathrm{y} \mid \mathbf{x}) \tag{1}$$

We train the model to make unconfident predictions on these self-generated examples by maximizing the model's predictive entropy. We add this entropy term $\mathcal{H}_{\boldsymbol{w}}(\tilde{\mathbf{x}})$ to the standard MLE objective, weighted by scalar $\lambda > 0$, yielding the final RCAD objective:

$$\hat{\boldsymbol{w}}_{\text{rcad}} \triangleq \underset{\boldsymbol{w} \in \mathcal{W}}{\arg\max}\ \mathbb{E}_{\hat{\mathcal{D}}}\left[\log p_{\boldsymbol{w}}(\mathrm{y} \mid \mathbf{x}) + \lambda \cdot \mathcal{H}_{\boldsymbol{w}}\left(\mathbf{x} - \alpha \cdot \nabla_{\mathbf{x}} \log p_{\boldsymbol{w}}(\mathrm{y} \mid \mathbf{x})\right)\right] \tag{2}$$

In adversarial training, adversarial examples are generated by solving a constrained optimization problem [28, 39]. Using a first-order approximation, the adversarial example $\tilde{\mathbf{x}}$ generated by one of the simplest solvers [11] has a closed form resembling Equation 1. We note that RCAD is different from the above form of adversarial training in two ways. First, RCAD takes a step size that is $10\times$ larger, so that the resulting example no longer resembles the training examples (Figure 1a). Second, whereas adversarial training updates the model to be more confident on the new example, RCAD trains the model to be less confident. Our experiments in Section 4 (Figure 4b) show that these differences are important for improving test accuracy.

Informal understanding of RCAD. For image classification, image features that are pure noise and independent of the true label can still be spuriously correlated with the labels for a finite set of *iid* drawn examples [45]. Such features are usually termed *spurious features* [62]. Overparameterized neural networks have a tendency to overfit on spurious features in the training examples [58, 42]. In order to generate unrealistic out-of-distribution samples, we pick examples that are far away from the true data point along the adversarial direction because *(i)* adversarial directions are comprised of noisy features that are spuriously correlated with the label on a few samples [18]; and *(ii)* maximizing entropy on samples with an amplified feature forces the model to quickly unlearn that feature. Thus, by taking large step size α we exacerbate the spurious features in our self-generated example $\tilde{\mathbf{x}}$. When we maximize predictive entropy over our generated examples we force the model to *unlearn* the spurious correlations. Hence, the trained model generalizes better and achieves higher prediction performance on unseen test examples. In Section 5 we build on this informal understanding to present a more formal analysis on the benefits of RCAD.

4 Experiments: Does RCAD Improve Test Performance?

Our experiments aim to study the effect that RCAD has on test accuracy, both in comparison to and in addition to existing regularization techniques and adversarial methods, so as to understand the degree to which its effect is *complementary* to existing methods.

Benchmark datasets. We use six image classification benchmarks. In addition to CIFAR-10, CIFAR-100 [23], SVHN [33] and Tiny Imagenet [26], we modify CIFAR-100 by randomly sub-

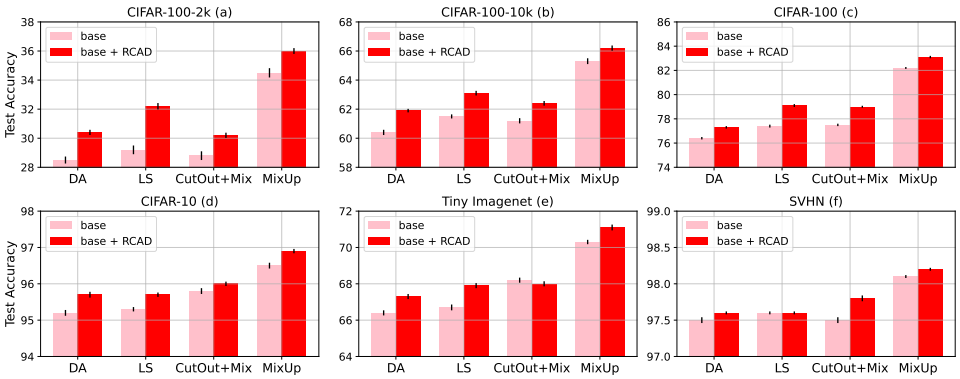


Figure 2: **Main results on supervised image classification benchmarks:** We plot the mean test accuracy and 95% confidence intervals over 10 independent runs for models trained with base methods: Data Augmentation (DA), Label Smoothing (LS), CutOut/CutMix (CutOut+Mix) augmentation, MixUp and compare them with the test accuracies of the models trained with the RCAD objective in Equation 2, in addition to the base methods.

sampling 2,000 and 10,000 training examples (from the original 50,000) to create CIFAR-100-2k and CIFAR-100-10k. These smaller datasets allow us to study low-data settings where we expect the generalization gap to be larger. CIFAR-100(-2k/10k) share the same test sets. If the validation split is not provided by the benchmark, we hold out 10% of our training examples for validation.

Implementation details[1]**.** Unless specified otherwise, we train all methods using the ResNet-18 [16] backbone, and to accelerate training loss convergence we clip gradients in the l_2 norm (at 1.0) [61, 15]. We train all models for 200 epochs and use SGD with an initial learning rate of 0.1 and Nesterov momentum of 0.9, and decay the learning rate by a factor of 0.1 at epochs $100, 150$ and 180 [7]. We select the model checkpoint corresponding to the epoch with the best accuracy on validation samples as the final model representing a given training method. For all datasets (except CIFAR-100-2k and CIFAR-100-10k for which we used 32 and 64 respectively) the methods were trained with a batch size of 128. For details on algorithm specific hyperparameter choices refer to Appendix B.

Baselines. The primary aim of our experiments is to study whether entropy maximization along the adversarial direction shrinks the generalization gap. Hence, we explore baselines commonplace in deep learning that either *(i)* directly constrain the model's predictive distribution on observed samples (label smoothing [31]) or *(ii)* implicitly regularize the model by training on additional images generated in an adversarial manner. For the first, our main comparisons are to label smoothing [47]), standard data augmentation [44], cutout data augmentation [56, 7], and MixUp [59] training. For the second, we compare with adversarial training[28] that uses FGSM [11] to perturb the inputs. Additionally, we compare RCAD with two recent approaches that use adversarial examples in different ways: adversarial data augmentation (ADA) [53] and maximum entropy adversarial data augmentation (ME-ADA) [63]. We summarize these baselines in Table 1.

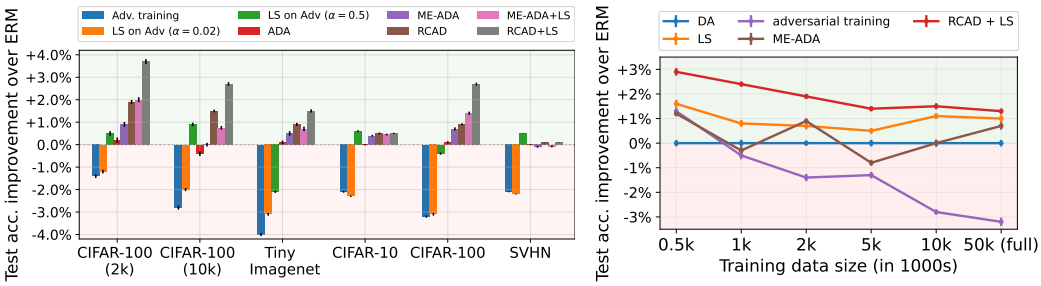
4.1 How much does RCAD improve test accuracy in comparison and in addition to other regularization methods?

Figure 2 presents the main empirical findings of RCAD over six image classification benchmarks and across four baseline regularizers. Across all datasets, we observe that training with the RCAD objective improves the test accuracy over the baseline method in $22/24$ cases. The effects are more pronounced on datasets with fewer training samples. For example, on CIFAR-100-2k, adding RCAD on top of label smoothing boosts the performance by $\approx 3\%$. These results also show that the benefits of RCAD are complementary to prior methods—RCAD + MixUp outperforms MixUp, and RCAD + augmentation/smoothing outperforms both. We test for statistical significance using a 1-sided p-value, finding that $p \ll 1e{-}3$ in $22/24$ comparisons. In summary, these results show that our proposed regularizer is complementary to prior regularization techniques.

4.2 How effectively does RCAD improve test accuracy compared to adversarial training?

Our next set of experiments compares different ways of using adversarial examples. RCAD maximizes the model's predictive entropy on unlabeled examples along the adversarial direction (obtained with a large step-size ≈ 0.5). In contrast, other methods we compare against (Adversarial Training [28],

[1]Code for this work can be found in the supplementary submission.



(a) (b)

Figure 3: *(Left)* **How should we use adversarial examples to improve test accuracy?** We compare RCAD with adversarial baselines including label smoothing on adversarial samples (LS on Adv), measuring the improvement in test accuracy relative to ERM. We also compare RCAD with ME-ADA when combined with label smoothing. *(Right)* **RCAD is more effective in low-data regime.** We compare the test accuracy improvement over ERM for baselines: Label Smoothing (LS), Data Augmentation (DA), adversarial training, and ME-ADA with RCAD + LS on different sub-samples of CIFAR-100 training set ($0.5\text{k} \rightarrow 50\text{k}$). We find RCAD achieves the largest gains in the low data regime. In both plots, we plot the mean and 95% confidence intervals over 10 independent runs.

ADA [53], ME-ADA [53]) minimize the cross entropy loss on examples obtained by adversarially perturbing the inputs without changing the original labels (using a much smaller step size ≈ 0.05). Additionally, we look at the baseline that performs label smoothing on these adversarial samples (LS on Adv) – a relaxed version of RCAD. We evaluate all methods by measuring their test accuracy improvement over empirical risk minimization (ERM), noting that some of these methods were proposed to optimize robustness, a different metric. We show results in Figure 3a and note that RCAD outperforms adversarial training and LS on Adv with small step-sizes ($\alpha = 0.02$), ADA and ME-ADA by significant margins on all benchmarks. We numerically verify that RCAD statistically outperforms the best baseline ME-ADA by computing 1-sided p-values (p=0.009 on CIFAR-10, $p < 1e{-}4$ on others). Next, we look at LS on Adv with large $\alpha = 0.5$. Since this is the same value of α used by RCAD for most datasets, this method is equivalent to performing label smoothing on examples generated by RCAD. Since label smoothing is a relaxed form of entropy maximization it is not surprising that the performance of this method is similar to (if not better than) RCAD on a few benchmarks. Finally, when both RCAD and ME-ADA are equipped with label smoothing, both methods improve, but the benefit of RCAD persists ($p = 0.0478$ on CIFAR-10, $p < 1e{-}4$ on others).

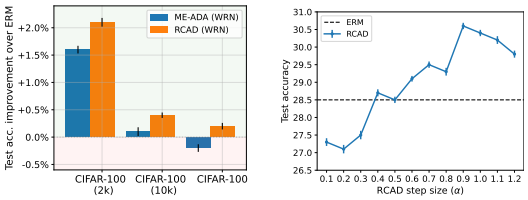
4.3 How effective is RCAD in low training data regime?

Since supervised learning methods are more susceptible to overfitting when training samples are limited [10], we analyze the effectiveness of RCAD in low-data regime. To verify this, we sample small training datasets of size 0.5k, 1k, 2k, 5k and 10k from the 50,000 training samples in CIFAR-100. The test set for each is the same as that of CIFAR-100. We train the baseline regularizers and RCAD on each of these datasets and compare the observed improvements in test accuracies relative to ERM. We show our results for this experiment in Figure 3b. Straight away we observe that RCAD yields positive improvements in test accuracy for any training dataset size. In contrast, in line with the *robust overfitting* phenomena described in Raghunathan et al. [39, 38], Zhang et al. [60], adversarial training (with FGSM [11]) is found to be hurtful in some settings. Next we observe that compared to typical regularizers for supervised learning like label smoothing and data augmentation, the regularization effect of RCAD has a stronger impact as the size of the training data reduces. Notice that RCAD has a steeper upward trend (right$\rightarrow$left) compared to the next best method label smoothing, while outperforming each baseline in terms of absolute performance values.

4.4 Additional Results and Ablations

In addition to the experiments below on wider architectures, effect of step size α, and regression tasks, we conduct more analysis and experiments for RCAD (e.g., evaluating its robustness to adversarial perturbations and distribution shifts). For these and other experimental details refer to Appendix B, C.

Results on a larger architecture. We compare the test accuracies of RCAD and ME-ADA (most competitive baseline from ResNet-18 experiments) when trained with the larger backbone Wide ResNet 28-10 [57] (WRN) on CIFAR-100 and its derivatives. We plot these test accuracies relative to ERM trained with WRN in Figure 4a. Clearly, the benefit of RCAD over ME-ADA still persists albeit with slightly diminished absolute performance differences compared to ResNet-18 in Figure 3a.



(a) (b)

Negative log-likelihood (NLL) on test data					
Method	Boston	Concrete	Energy	Wine	Yacht
ERM	2.85 ± 0.06	3.25 ± 0.07	**1.85** ± 0.03	**0.97** ± 0.04	1.66 ± 0.04
ME-ADA	3.01 ± 0.04	**3.14** ± 0.03	1.99 ± 0.06	1.11 ± 0.05	1.78 ± 0.04
RCAD (ours)	**2.60** ± 0.05	**3.12** ± 0.04	**1.75** ± 0.07	1.09 ± 0.05	**1.54** ± 0.05

(c)

Figure 4: **RCAD can be used with larger networks and is also effective on regression tasks.** *(Left)* We compare the improvements in test accuracy (over ERM) for RCAD and ME-ADA when all methods use Wide ResNet 28-10 [57]. *(Center)* For CIFAR-100-2k, we show the effect on test accuracy of the step size α that RCAD takes while generating out-of-distribution examples along the adversarial direction. *(Right)* We compare the negative log-likelihood on test samples for RCAD with baselines ERM and ME-ADA on five regression datasets from the UCI repository. For all the above we show the mean result and 95% confidence intervals across 10 runs.

Effect of step size α on test accuracy. In Figure 4b we plot the test accuracy of RCAD on CIFAR-100-2k as we vary the step size α. We see that RCAD is effective only when the step size is sufficiently large (> 0.5), so that new examples do not resemble the original ones. When the step size is small (< 0.5), the perturbations are small and the new example is indistinguishable from the original, as in adversarial training. Our theory in Section 5 also agrees with this observation.

Results on regression tasks. On five regression datasets from the UCI repository we compare the performance of RCAD with ERM and ME-ADA in terms of test negative log-likelihood (NLL) ($\downarrow$ is better). To be able to compute NLL, we model the predictive distribution $p_{\boldsymbol{w}}(\mathrm{y} \mid \mathbf{x})$ as a Gaussian and each method outputs two parameters (mean, variance of $p_{\boldsymbol{w}}(\mathrm{y} \mid \mathbf{x})$) at each input $\mathbf{x}$. For the RCAD regularizer we use the differential form of entropy. Results are shown in Figure 4c from which we see that RCAD matches/improves over baselines on 4/5 regression datasets.

5 Analysis: Why Does RCAD Improve Test Performance?

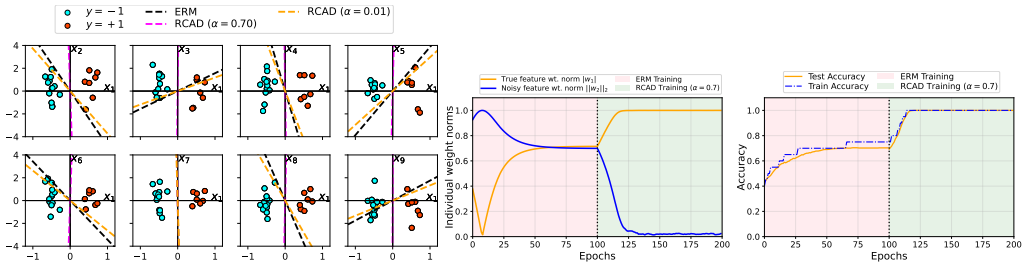
In this section, we present theoretical results that provide a more formal explanation as to why we see a boost in test accuracy when we regularize a classifier using RCAD. We analyze the simplified problem of learning l_2-regularized linear predictors in a fully specified binary classification setting. Specifically, our analysis will show that, while linear predictors trained with standard ERM can have a high dependence on spurious features, further optimization with the RCAD objective will cause the classifier to unlearn these spurious features.

The intuition behind our result can be summarized as follows: when RCAD generates examples after taking a large step along adversarial directions, a majority of the generated examples end up lying close to the decision boundary along the true feature, and yet a portion of them would significantly depend on the noisy spurious features. Thus, when RCAD maximizes the predictive entropy on these new examples, the classifier weights that align with the spurious components are driven toward smaller values. Detailed proofs for the analysis that follows can be found in Appendix A.

Setup. We consider a binary classification problem with a joint distribution $\mathcal{D}$ over $(\mathbf{x}, \mathrm{y}) \in \mathcal{X} \times \{0, 1\}$ where $\mathcal{X} \subseteq \mathbb{R}^{d+1}$ with $\mathbf{x} = [\mathrm{x}_1, \mathbf{x}_2]^\top$, $\mathrm{x}_1 \in \mathbb{R}$, $\mathbf{x}_2 \in \mathbb{R}^d$ and for some $\beta > 0$, $\mathcal{D}$ is:

$$\mathrm{y} \sim \mathrm{Unif}\{-1, 1\}, \quad \mathrm{x}_1 \sim \mathcal{N}(\beta \cdot \mathrm{y}, \sigma_1^2), \quad \mathbf{x}_2 \sim \mathcal{N}(\mathbf{0}, \Sigma_2), \quad \text{and let } \tilde{\Sigma} \triangleq \begin{pmatrix} \sigma_1^2 & \mathbf{0} \\ \mathbf{0} & \Sigma_2 \end{pmatrix}. \tag{3}$$

We borrow this setup from Chen et al. [5] which analyzes self-training/entropy minimization on unlabeled out-of-distribution samples. While their objective is very different from RCAD, their setup is relevant for our analysis since it captures two kinds of features: x_1, which is the *true* univariate feature that is predictive of the label y, and $\mathbf{x}_2$, which is the high dimensional *noise*, that is not correlated with the label under the true distribution $\mathcal{D}$, but w.h.p. is correlated with the label on finite sampled training dataset $\hat{\mathcal{D}} = \{(\mathbf{x}^{(i)}, \mathrm{y}^{(i)})\}_{i=1}^n \sim \mathcal{D}^n$. We wish to learn the class of homogeneous linear predictors $\mathcal{W} \triangleq \{\boldsymbol{w} \in \mathbb{R}^{d+1} : \langle \boldsymbol{w}, \mathbf{x} \rangle = w_1 \cdot \mathrm{x}_1 + \boldsymbol{w}_2 \cdot \mathbf{x}_2, \|\boldsymbol{w}\|_2 \leqslant 1\}$. For mathematical convenience we make three modifications to the RCAD objective in Equation 2 that are common in works analyzing logistic classifiers and entropy based objectives [46, 5]: *(i)* we minimize the margin loss $l_\gamma(\boldsymbol{w}; (\mathbf{x}, \mathrm{y})) \triangleq \max(0, \gamma - \mathrm{y} \cdot \langle \boldsymbol{w}, \mathbf{x} \rangle)$ (instead of the negative log-likelihood) over $\mathcal{D}'$ with some $\gamma > 0$; *(ii)* we consider the constrained optimization problem in Equation 4 as opposed



(a) (b) (c)

Figure 5: **RCAD corrects ERM solution:** *(Left)* We plot the training data and high-dimensional decision boundaries for all three estimates $\hat{\boldsymbol{w}}_{\text{erm}}$ and $\hat{\boldsymbol{w}}_{\text{rcad}}$ with $\alpha = 0.01, 0.70$. Here, we plot the linear boundary projected onto the set of planes $\{\mathrm{x}_1, \mathrm{x}_i\}_{i=2}^{i=8}$ spanned by the true feature x_1 and noisy features $\mathrm{x}_2, \ldots, \mathrm{x}_8$. Across the ERM and RCAD training iterations, we plot the weight norms along the true feature $|w_1|$ and noisy dimensions $\|\boldsymbol{w}_2\|_2$ in the *(Center)* plot, and on the *(Right)* we plot the train/test accuracies.

to the Lagrangian form of RCAD in Equation 2 [27]; and *(iii)* we use Lemma 5.1 and replace $\mathcal{H}_{\boldsymbol{w}}(\mathbf{x}+\alpha\cdot\nabla_{\mathbf{x}} l_\gamma(\boldsymbol{w};(\mathbf{x},\mathrm{y})))$ with $\exp(-|\langle \boldsymbol{w}, \mathbf{x}+\alpha\cdot\nabla_{\mathbf{x}} l_\gamma(\boldsymbol{w};(\mathbf{x},\mathrm{y}))\rangle|)$. Note that the optimal $\boldsymbol{w}^* \in \mathcal{W}$ that has the best test accuracy and lowest population margin loss is given by $\boldsymbol{w}^* \triangleq [1, 0, \ldots, 0]^\top$.

$$\begin{aligned} \max_{\boldsymbol{w}} \quad & \mathcal{M}_{\hat{\mathcal{D}}}(\boldsymbol{w}) \triangleq \mathbb{E}_{\hat{\mathcal{D}}} \ \exp(-|\langle \boldsymbol{w}, \mathbf{x} + \alpha \cdot \nabla_{\boldsymbol{x}} l_\gamma(\boldsymbol{w};(\mathbf{x},\mathrm{y}))\rangle|) \\ \text{s.t.} \quad & \mathbb{E}_{\hat{\mathcal{D}}} \ \mathbb{1}(l_\gamma(\boldsymbol{w};(\mathbf{x},\mathrm{y})) > 0) \leqslant \rho/2, \quad \boldsymbol{w} \in \mathcal{W} \end{aligned} \tag{4}$$

Lemma 5.1 ([4, 46], informal)**.** *$\mathcal{H}_{\boldsymbol{w}}(\mathbf{x}) = \mathcal{H}_{\text{bin}}((1+\exp(-\langle \boldsymbol{w}, \mathbf{x}\rangle))^{-1})$ where $\mathcal{H}_{\text{bin}}(p)$ is the entropy of $\text{Bern}(p)$ distribution. Thus $\mathcal{H}_{\boldsymbol{w}}(\mathbf{x}) \approx \exp(-|\langle \boldsymbol{w}, \mathbf{x}\rangle|)$, as both exhibit same tail behavior.*

We analyze the RCAD estimate $\hat{\boldsymbol{w}}_{\text{rcad}} \triangleq \boldsymbol{w}^{(T)}$ returned after T iterations of projected gradient ascent on the non-convex objective in Equation 4, initialized with $\boldsymbol{w}^{(0)}$ satisfying the constraints in Equation 4. Given the projection $\Pi_{\mathcal{S}}$ onto the convex set $\mathcal{S}$, and learning rate $\eta > 0$, the update rule for $\boldsymbol{w}^{(t)}$ is as follows: $\tilde{\boldsymbol{w}}^{(t+1)} = \boldsymbol{w}^{(t)} + \eta \cdot \nabla_{\boldsymbol{w}^{(t)}} \mathcal{M}_{\hat{\mathcal{D}}}(\boldsymbol{w})$, and $\boldsymbol{w}^{(t+1)} = \Pi_{\|\boldsymbol{w}\|_2 \leqslant 1} \tilde{\boldsymbol{w}}^{(t+1)}$. Since the initialization $\boldsymbol{w}^{(0)}$ satisfies the constraint in Equation 4, it has a low margin loss on training samples and using a margin based generalization gap [22] stated in Lemma 5.2 we can conclude w.h.p. $\boldsymbol{w}^{(0)}$ has low test error ($\leqslant \rho$). This, in turn tells us that $\boldsymbol{w}^{(0)}$ should have learnt the true feature x_1 to some extent (Lemma 5.3). Intuitively, if the classifier is not trained at all, then adversarial directions would be less meaningful since they may include a higher component of the true feature x_1, as opposed to noisy $\mathbf{x}_2$. To obtain $\boldsymbol{w}^{(0)}$, we simply minimize loss l_γ on $\hat{\mathcal{D}}$ using projected (onto $\|\boldsymbol{w}\|_2 \leqslant 1$) gradient descent and the ERM solution $\hat{\boldsymbol{w}}_{\text{erm}}$ serves as the initialization $\boldsymbol{w}^{(0)}$ for RCAD's gradient ascent iterations. Note that $\hat{\boldsymbol{w}}_{\text{erm}}$ still significantly depends on spurious features (see Figure 5). Furthermore, this dependence is unavoidable and worse when n is small, or when the noise dimension d is large.

Lemma 5.2 (generalization gap)**.** *With probability $1-\delta$ over $\hat{\mathcal{D}}$, $\forall \boldsymbol{w} \in \mathcal{W}$, $\mathbb{E}_{\mathcal{D}}\left[\mathbb{1}(\mathrm{y}\cdot\langle \boldsymbol{w}, \mathbf{x}\rangle < 0)\right] \leqslant \mathbb{E}_{\hat{\mathcal{D}}}\left[\mathbb{1}(\mathrm{y}\cdot\langle \boldsymbol{w}, \mathbf{x}\rangle < \gamma)\right] + \tilde{\mathcal{O}}\left(\frac{\beta+\sqrt{\log(1/\delta)\|\tilde{\Sigma}\|_{\text{op}}}+\sqrt{\|\tilde{\Sigma}\|_*}}{\gamma\sqrt{n}}\right)$. $\tilde{\mathcal{O}}$ hides* poly-log *factors in $\beta, \gamma, \|\tilde{\Sigma}\|$.*

Lemma 5.3 (true feature is partially learnt before RCAD)**.** *If $\boldsymbol{w}^{(0)}$ satisfies constraint in Equation 4, and when $n \gtrsim \frac{\beta^2+\log(1/\delta)\cdot\|\tilde{\Sigma}\|_{\text{op}}+\|\tilde{\Sigma}\|_*}{\gamma^2\rho^2}$, with probability $1-\delta$ over $\hat{\mathcal{D}}$, $w_1^{(0)} \geqslant \frac{\text{erfc}^{-1}(2\rho)\cdot\sqrt{2\sigma_{\min}(\tilde{\Sigma})}}{\beta}$.*

We are now ready to present our main results in Theorem 5.4 which states that after $T = \mathcal{O}(\log(1/\epsilon))$ gradient ascent iterations of optimizing the RCAD objective $\hat{\boldsymbol{w}}_{\text{rcad}}$ is ϵ close to $\boldsymbol{w}^*$, since it *unlearns* the spurious feature ($\|\boldsymbol{w}_2^{(T)}\|_2 \leqslant \epsilon$) and *amplifies* the true feature ($|w_1^{(T)}| \geqslant \sqrt{1-\epsilon^2}$). Note that Theorem 5.4 suggests a minimum step size α in arriving at a sufficient condition for RCAD to unlearn spuriousness and improve test performance This is also in line with our empirical findings (Figure 4b). Since $\alpha = \Theta(\gamma)$, most of the points generated by RCAD lie close to the decision boundary along x_1, except for the ones with noisy features that are correlated with the classifier weights.

Theorem 5.4 ($\hat{\boldsymbol{w}}_{\text{rcad}} \to \boldsymbol{w}^*$)**.** *If $\beta = \Omega(\|\tilde{\Sigma}\|_{\text{op}})$ and $\exists\, c_0, K > 0$, such that $\beta \gtrsim \alpha \gtrsim \max(\gamma, \|\tilde{\Sigma}\|_{\text{op}})$ and $\gamma + c_0 \cdot \sigma_{\min}(\tilde{\Sigma}) \geqslant \beta$, then with $n \gtrsim \frac{\beta^2+\log(1/\delta)\cdot\|\tilde{\Sigma}\|_{\text{op}}+\|\tilde{\Sigma}\|_*}{\gamma^2 \text{erfc}\left(K\alpha/\sqrt{2\sigma_{\min}(\tilde{\Sigma})}\right)^2} + \frac{\log 1/\delta}{\epsilon^4}$, after $T = \mathcal{O}\left(\log(1/\epsilon)\right)$ gradient ascent iterations, $|w_1^{(T)}| \geqslant \sqrt{1-\epsilon^2}$ and $\|\boldsymbol{w}_2^{(T)}\|_2 \leqslant \epsilon$, with probability $1-\delta$ over $\mathcal{D}'$.*

The key idea behind our guarantee above is an inductive argument. We show that, if the class separation β and step size α are sufficiently larger than any noise variance, then RCAD monotonically increases the weight norm along the true feature and monotonically decreases it along the noisy dimensions with each gradient ascent update. This is because at any given instant the gradient of RCAD objective $\mathcal{M}_{\mathcal{D}}(\boldsymbol{w})$ with respect to w_1 always points in the same direction, while the one with respect to $\boldsymbol{w}_2$ is sufficiently anti-correlated with the direction of $\boldsymbol{w}_2$. We formalize this argument in Lemma 5.5. It is also easy to see why the update will improve test accuracy monotonically, thus satisfying the constraint in Equation 4 with high probability.

Lemma 5.5 ($w_1, \boldsymbol{w}_2$ update)**.** *If α, β, γ and sample size n satisfy the noise conditions in Theorem 5.4, then $\langle \frac{\partial \mathcal{M}_{\mathcal{D}}(\boldsymbol{w})}{\partial w_1^{(t)}}, w_1^{(t)} \rangle > 0$, and $|\tilde{w}_1^{(t+1)}| > |w_1^{(t)}|$. On the other hand, $\exists c_1 > 0$ such that $\langle \nabla_{\boldsymbol{w}_2^{(t)}} \mathcal{M}_{\mathcal{D}}(\boldsymbol{w}^{(t)}), \boldsymbol{w}_2^{(t)} \rangle \leqslant -c_1 \cdot \|\boldsymbol{w}_2^{(t)}\|_2^2$. Consequently $\exists \eta$, such that $\|\tilde{\boldsymbol{w}}_2^{(t+1)}\|_2 / \|\boldsymbol{w}_2^{(t)}\|_2 < 1$.*

Empirically validating our theoretical results in the toy setup. Now, using the same setup as our theoretical study, we check if taking the ERM solution and updating it with RCAD truly helps in unlearning the spurious feature $\mathbf{x}_2$ and amplifying the true feature x_1. With $d = 8, \gamma = 2.0, \beta = 0.5, \sigma_1 = 0.1, \Sigma = \mathbf{I}_d$ we collect training dataset $\hat{\mathcal{D}}$ of size 20 according to the data distribution in Equation 3. First, we obtain the ERM solution $\hat{\boldsymbol{w}}_{\mathrm{erm}}$ by optimizing the margin loss on $\hat{\mathcal{D}}$ for 100 projected gradient descent iterations. Then, with $\hat{\boldsymbol{w}}_{\mathrm{erm}}$ as initialization we optimize the RCAD objective in Equation 4 for 100 additional projected gradient ascent iterations to obtain RCAD estimate $\hat{\boldsymbol{w}}_{\mathrm{rcad}}$. For RCAD, we try both small ($\alpha = 0.01$) and large ($\alpha = 0.7$) values of the step size. Using the results of this study in Figure 5 we shall now verify our main theoretical claims:

(i) **RCAD unlearns spurious features learnt by ERM (Theorem 5.4):** RCAD training with large step size $\alpha = 0.7$ corrects ERM's decision boundaries across all noisy dimensions i.e., $\hat{\boldsymbol{w}}_{\mathrm{rcad}}$ has almost no dependence on noisy dimensions $\{\mathrm{x}_i\}_{i=2}^{i=8}$ (Figure 5a). On the other hand, the ERM solution has a significant non-zero component across all noisy dimensions (except x_7).

(ii) **RCAD is helpful only when step size is large enough (Theorem 5.4):** In Figure 5a we see that RCAD estimate with small step size $\alpha = 0.01$ has a higher dependence on spurious features compared to ERM. Thus optimizing RCAD with a very small α leads to an even poor solution than ERM. This is because, the perturbed data point still has a significant component of the true feature, which the model is forced to unlearn when RCAD tries to maximize entropy over the perturbed point.

(iii) **The weight norms $|w_1|$ and $\|\boldsymbol{w}_2\|_2$ change monotonically through RCAD iterations (Lemma 5.5):** In Figure 5b we see that the norm along the true feature $|w_1|$ increases monotonically through RCAD iterations, whereas $\|\boldsymbol{w}_2\|_2$ decreases monotonically, until $|w_1| \approx 1$ and $\|\boldsymbol{w}_2\|_2 \approx 0$. Thus, RCAD successfully recovers optimal $\boldsymbol{w}^*$. Based on this, we also see the test accuracy increase monotonically through RCAD iterations; reaching 100% by the end of it (Figure 5c). In contrast, since the ERM solution depends on spurious features, its accuracy does not improve beyond 70% in the first 100 iterations. We find that training ERM for more iterations only improves training accuracy.

The analysis above explains why RCAD improves test performance in the linear setting and it is inline with our intuition of RCAD's ability to unlearn spurious features. To check if our intuition also generalizes to deeper networks we further study RCAD's behaviour in a non-linear setting using a different toy example in Appendix D. Furthermore, while our analysis is restricted to linear classifiers, linear models can provide a rough proxy for neural network learning dynamics via the neural tangent kernel (NTK) approximation [19]. This strategy has been used in a number of prior works [32, 49, 1] and extending the analysis to the NTK setting may be possible in future work.

6 Conclusion

In this paper we propose RCAD, a regularization technique that maximizes a model's predictive entropy on out-of-distribution examples. These samples lie along the directions that adversarially maximize the loss locally around the training points. Our experiments on image classification benchmarks show that RCAD not only improves test accuracy by a significant margin, but that it can also seamlessly compose with prior regularization methods. We find that RCAD is particularly effective when learning from limited data. Finally, we present analyses in a simplified setting to show that RCAD can help the model unlearn noisy features. Some current limitations of our work are that RCAD slows training by 30% and our theoretical analysis is limited to the linear case, which would be interesting directions to address/expand on in future work—particularly in light of the significant empirical benefits of RCAD for improving generalization.

References

[1] Cao, Y. and Gu, Q. (2019). Generalization bounds of stochastic gradient descent for wide and deep neural networks. *Advances in neural information processing systems*, 32.

[2] Carlini, N., Athalye, A., Papernot, N., Brendel, W., Rauber, J., Tsipras, D., Goodfellow, I., Madry, A., and Kurakin, A. (2019). On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705*.

[3] Carlini, N. and Wagner, D. (2017). Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE.

[4] Chen, T., Zhang, Z., Liu, S., Chang, S., and Wang, Z. (2020a). Robust overfitting may be mitigated by properly learned smoothening. In *International Conference on Learning Representations*.

[5] Chen, Y., Wei, C., Kumar, A., and Ma, T. (2020b). Self-training avoids using spurious features under domain shift. *Advances in Neural Information Processing Systems*, 33:21061–21071.

[6] Church, K. W. and Gale, W. A. (1991). A comparison of the enhanced good-turing and deleted estimation methods for estimating probabilities of english bigrams. *Computer Speech & Language*, 5(1):19–54.

[7] DeVries, T. and Taylor, G. W. (2017). Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*.

[8] Dubey, A., Gupta, O., Raskar, R., and Naik, N. (2018). Maximum-entropy fine grained classification. *Advances in neural information processing systems*, 31.

[9] Gao, Y., Wang, W., Herold, C., Yang, Z., and Ney, H. (2020). Towards a better understanding of label smoothing in neural machine translation. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 212–223.

[10] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

[11] Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

[12] Grandvalet, Y. and Bengio, Y. (2004). Semi-supervised learning by entropy minimization. *Advances in neural information processing systems*, 17.

[13] Grathwohl, W., Wang, K.-C., Jacobsen, J.-H., Duvenaud, D., Norouzi, M., and Swersky, K. (2019). Your classifier is secretly an energy based model and you should treat it like one. *arXiv preprint arXiv:1912.03263*.

[14] Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR.

[15] Hardt, M., Recht, B., and Singer, Y. (2016). Train faster, generalize better: Stability of stochastic gradient descent. In *International Conference on Machine Learning*, pages 1225–1234. PMLR.

[16] 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.

[17] Hertz, J., Krogh, A., and Palmer, R. G. (2018). *Introduction to the theory of neural computation*. CRC Press.

[18] Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., and Madry, A. (2019). Adversarial examples are not bugs, they are features. *Advances in neural information processing systems*, 32.

[19] Jacot, A., Gabriel, F., and Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31.

[20] Jaynes, E. T. (1957). Information theory and statistical mechanics. *Physical review*, 106(4):620.

[21] Johnson, W. E. (1932). Probability: The deductive and inductive problems. *Mind*, 41(164):409–423.

[22] Kakade, S. M., Sridharan, K., and Tewari, A. (2008). On the complexity of linear prediction: Risk bounds, margin bounds, and regularization. *Advances in neural information processing systems*, 21.

[23] Krizhevsky, A. et al. (2009). Learning multiple layers of features from tiny images. *Citeseer*.

[24] Krogh, A. and Hertz, J. A. (1992). A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957.

[25] Kurakin, A., Goodfellow, I., and Bengio, S. (2016). Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*.

[26] Le, Y. and Yang, X. (2015). Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3.

[27] Li, D. (1995). Zero duality gap for a class of nonconvex optimization problems. *Journal of Optimization Theory and Applications*, 85(2):309–324.

[28] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

[29] Miyato, T., Maeda, S.-i., Koyama, M., and Ishii, S. (2018). Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993.

[30] Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. (2016). Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582.

[31] Müller, R., Kornblith, S., and Hinton, G. (2019). When does label smoothing help? *arXiv preprint arXiv:1906.02629*.

[32] Nagarajan, V. and Kolter, J. Z. (2019). Uniform convergence may be unable to explain generalization in deep learning. *Advances in Neural Information Processing Systems*, 32.

[33] Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. *Citeseer*.

[34] Odena, A. (2016). Semi-supervised learning with generative adversarial networks. *arXiv preprint arXiv:1606.01583*.

[35] Pereyra, G., Tucker, G., Chorowski, J., Kaiser, Ł., and Hinton, G. (2017). Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*.

[36] Pinto, F., Yang, H., Lim, S.-N., Torr, P., and Dokania, P. K. (2021). Mix-maxent: Improving accuracy and uncertainty estimates of deterministic neural networks. In *NeurIPS 2021 Workshop on Distribution Shifts: Connecting Methods and Applications*.

[37] Poursaeed, O., Jiang, T., Yang, H., Belongie, S., and Lim, S.-N. (2021). Robustness and generalization via generative adversarial training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15711–15720.

[38] Raghunathan, A., Xie, S. M., Yang, F., Duchi, J., and Liang, P. (2020). Understanding and mitigating the tradeoff between robustness and accuracy. In *International Conference on Machine Learning*, pages 7909–7919. PMLR.

[39] Raghunathan, A., Xie, S. M., Yang, F., Duchi, J. C., and Liang, P. (2019). Adversarial training can hurt generalization. *arXiv preprint arXiv:1906.06032*.

[40] Rockafellar, R. T., Uryasev, S., et al. (2000). Optimization of conditional value-at-risk. *Journal of risk*, 2:21–42.

[41] Sabour, S., Cao, Y., Faghri, F., and Fleet, D. J. (2016). Adversarial manipulation of deep representations. In *ICLR (Poster)*.

[42] Sagawa, S., Raghunathan, A., Koh, P. W., and Liang, P. (2020). An investigation of why overparameterization exacerbates spurious correlations. In *International Conference on Machine Learning*, pages 8346–8356. PMLR.

[43] Saito, K., Kim, D., Sclaroff, S., Darrell, T., and Saenko, K. (2019). Semi-supervised domain adaptation via minimax entropy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8050–8058.

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

[45] Singla, S. and Feizi, S. (2021). Salient imagenet: How to discover spurious features in deep learning? In *International Conference on Learning Representations*.

[46] Soudry, D., Hoffer, E., Nacson, M. S., Gunasekar, S., and Srebro, N. (2018). The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878.

[47] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

[48] Tamar, A., Glassner, Y., and Mannor, S. (2015). Optimizing the cvar via sampling. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

[49] Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J., and Ng, R. (2020). Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547.

[50] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288.

[51] Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., and Madry, A. (2018). Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*.

[52] Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., and Madry, A. (2019). Robustness may be at odds with accuracy. *arXiv: Machine Learning*.

[53] Volpi, R., Namkoong, H., Sener, O., Duchi, J., Murino, V., and Savarese, S. (2018). Generalizing to unseen domains via adversarial data augmentation. *arXiv preprint arXiv:1805.12018*.

[54] Wang, D., Shelhamer, E., Liu, S., Olshausen, B., and Darrell, T. (2020). Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*.

[55] Wang, X., Jiang, Y., Huang, M., and Zhang, H. (2013). Robust variable selection with exponential squared loss. *Journal of the American Statistical Association*, 108(502):632–643.

[56] Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., and Yoo, Y. (2019). Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6023–6032.

[57] Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. *arXiv preprint arXiv:1605.07146*.

[58] Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2021a). Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115.

[59] Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. (2018). mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*.

[60] Zhang, H., Yu, Y., Jiao, J., Xing, E., El Ghaoui, L., and Jordan, M. (2019a). Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, pages 7472–7482. PMLR.

[61] Zhang, J., He, T., Sra, S., and Jadbabaie, A. (2019b). Why gradient clipping accelerates training: A theoretical justification for adaptivity. *arXiv preprint arXiv:1905.11881*.

[62] Zhang, Y., Gong, M., Liu, T., Niu, G., Tian, X., Han, B., Schölkopf, B., and Zhang, K. (2021b). Adversarial robustness through the lens of causality. *arXiv preprint arXiv:2106.06196*.

[63] Zhao, L., Liu, T., Peng, X., and Metaxas, D. (2020). Maximum-entropy adversarial data augmentation for improved generalization and robustness. *Advances in Neural Information Processing Systems*, 33:14435–14447.

[64] Zhou, A. and Levine, S. (2021). Training on test data with bayesian adaptation for covariate shift. *arXiv preprint arXiv:2109.12746*.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets?
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Sec. 6.
   (c) Did you discuss any potential negative societal impacts of your work? [No] While supervised learning might be used in applications with both positive and negative societal impacts, our empirical and theoretical contributions do not dictate these potential applications.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] We provide them in the Appendix, as part of the supplementary submission.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We document these in the Appendix, as part of the supplementary submission.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] For our results in Section 4, we provide error bars for the estimated test performance.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Details in Appendix.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We use public datasets and cite the creators.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]