RC-Mixup: A Data Augmentation Strategy against Noisy Data for Regression Tasks

Anonymous Author(s)

ABSTRACT

We study the problem of robust data augmentation for regression tasks in the presence of noisy data. Data augmentation is essential for generalizing deep learning models, but most of the techniques like the popular Mixup are primarily designed for classification tasks on image data. Recently, there are also Mixup techniques that are specialized to regression tasks like C-Mixup. In comparison to Mixup, which takes linear interpolations of pairs of samples, C-Mixup is more selective in which samples to mix based on their label distances for better regression performance. However, C-Mixup does not distinguish noisy versus clean samples, which can be problematic when mixing and lead to suboptimal model performance. At the same time, robust training has been heavily studied where the goal is to train accurate models against noisy data through multiple rounds of model training. We thus propose our data augmentation strategy RC-Mixup, which *tightly integrates C-Mixup with multi-round robust training methods for a synergistic effect*. In particular, C-Mixup improves robust training in identifying clean data, while robust training provides cleaner data to C-Mixup for it to perform better. A key advantage of RC-Mixup is that it is *data-centric* where the robust model training algorithm itself does not need to be modified, but can simply benefit from data mixing. We show in our experiments that RC-Mixup significantly outperforms C-Mixup and robust training baselines on noisy data benchmarks and can be integrated with various robust training methods.

ACM Reference Format:

Anonymous Author(s). 2024. RC-Mixup: A Data Augmentation Strategy against Noisy Data for Regression Tasks. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (KDD '24).* ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Deep learning is widely used in applications that perform regression tasks including manufacturing, climate prediction, and finance. However, one of the challenges is a lack of enough training data, and data augmentation techniques have been proposed as a solution for better generalizing the trained models. A representative technique is Mixup [24, 25, 52, 53], which mixes two samples by linear interpolation to estimate the label of any sample in between. However, Mixup is primarily designed for classification tasks mainly on image data along with most of the other data augmentation techniques [14, 27] and does not readily perform well on regression tasks where the goal is to predict real numbers.

Recently, there is an increasing literature on data augmentation designed for regression tasks, and C-Mixup [50] is the state-of-the-art Mixup-based method. In order to avoid arbitrarily-incorrect labels, each sample is mixed with a neighboring sample in terms of label distance where the selection follows the proposed sampling probability calculated using a Gaussian kernel that is configured using a bandwidth parameter. The larger the parameter the wider the distribution, which means that mixing can be done with distant neighbors in terms of label distance. In a sense, the bandwidth indicates whether mixup should be performed, and to what degree. C-Mixup has theoretical guarantees for generalization and handling correlation shifts in the data.

At the same time, robustness against noise is becoming increasingly important for regression tasks. For example, in semiconductor manufacturing, the layer thickness in a 3D semiconductor needs to be predicted for defect detection using a model. In this case, labels (e.g., layer thickness) can be noisy due to erroneous or malfunctioning measurement equipment, leading to a decline in prediction model performance and thus revenue. Hence, global semiconductor companies make great efforts to ensure that their models are robust against noise. To address this challenge, there is a recent line of multi-round robust training [43, 44] where noisy samples are removed or fixed based on their loss values through multiple model trainings.

We thus contend that integrating C-Mixup with robust training methods is desirable, but this is not trivial. C-Mixup is not explicitly designed to be robust against noisy data and may be prone to incorrect mixing because any out-of-distribution samples are mixed just the same way as in-distribution samples. C-Mixup is robust against correlation shifts, which assume the same data distribution, but unfortunately cannot cope with noise in general. More fundamentally, data augmentation is to add data, while robust training is to clean (select or refurbish) data, so the two seem to even be contradictory operations. A naïve approach is to run the two methods in sequence, e.g., run C-Mixup and then robust training or in the other ordering. However, either C-Mixup will end up running on noisy data or robust training would not benefit from augmented data.

We propose the novel data augmentation strategy of *tightly integrating C-Mixup and multi-round robust training for a synergistic effect* (see Figure 1). We call our framework RC-Mixup to emphasize the robustness of C-Mixup. Each robust training round typically consists of cleaning (Step 1) and model updating (Step 3) steps. Between these two steps, we run C-Mixup (Step 2) so that it benefits from the intermediate clean data that is identified by the cleaning. In addition, the model updating step now benefits from the augmented data produced by C-Mixup and produces a more accurate regression model that can clean data better. Another benefit of this

KDD '24, August 25–29, 2024, Barcelona, Spain

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

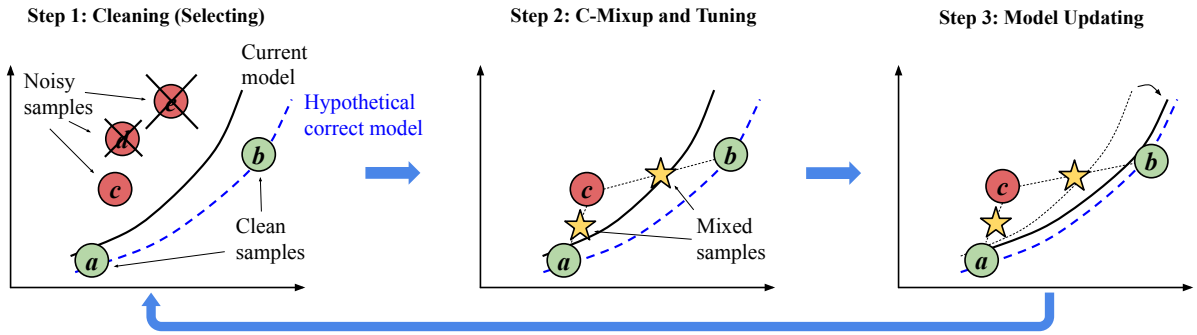


Figure 1: RC-Mixup tightly integrates C-Mixup with multi-round robust training techniques for a synergistic effect: C-Mixup improves robust training in identifying clean data, while robust training provides (intermediate) clean data for C-Mixup. Suppose the x-axis is the only feature, and the y-axis is the label. Also, there are two clean samples a and b and three noisy samples c, d, and e. In Step 1, suppose that cleaning removes d and e (the exact outcome depends on the robust training technique). In Step 2, we perform C-Mixup possibly with bandwidth tuning to generate mixed samples. Here we mix the sample pairs (a, c) and (b, c) to generate the mixed samples denoted as star shapes. Notice that C-Mixup selectively mixes samples that have closer labels, so in this example (a, b) are not mixed. In Step 3, the augmented samples can be used to train an improved regression model, which can then be used for better cleaning in the next round.

integration is that it is *data-centric* where the robust training algorithm itself does not need to be modified because RC-Mixup is only augmenting the data. Hence, RC-Mixup is compatible with any existing multi-round robust training algorithm like Iterative Trimmed Loss Minimization (ITLM) [43], O2U-Net [18], and SELFIE [44]. A technical challenge is efficiency where we would like to keep C-Mixup's bandwidth up-to-date during the multiple rounds in robust training. We propose to periodically update the bandwidth, but doing so reliably by evaluating candidate bandwidth values for several robust training rounds before choosing the one to use. We can optionally speedup this process by simply updating the bandwidth in one direction with some tradeoff in performance as well.

We perform extensive experiments of RC-Mixup on various regression benchmarks and show how it significantly outperforms C-Mixup in terms of robustness against noise. While RC-Mixup utilizes a small validation set, we show it is sufficient to use the robust training to generate one from the training set. In addition, RC-Mixup also outperforms existing robust training techniques that do not augment their data.

Summary of Contributions: (1) We propose RC-Mixup, the first selective mixing framework for regression tasks that is also robust against noisy data. (2) We tightly integrate the state-of-the-art C-Mixup with multi-round robust training techniques where C-Mixup utilizes intermediate clean data and has a synergistic effect with robust training. (3) We perform extensive experiments and show how RC-Mixup significantly outperforms baselines by utilizing this synergy on noisy real and synthetic datasets.

2 BACKGROUND

Notations. Let $\mathcal{D} = \{(x, y)\} \sim P$ be the training set where $x \in X$ is a d-dimensional input sample, and $y \in Y$ is an e-dimensional label. Let $\mathcal{D}^v = \{(x^v, y^v)\} \sim P^t$ be the validation set, where P^t is the distribution of the test set. Let θ model parameters, f_θ be a regression model, and ℓ_θ the loss function that returns a performance score comparing $f_\theta(x)$ with the true label y using Mean Squared Error (MSE) loss.

Mixup. Mixup [53] takes a linear interpolation between any pair of samples x_i and x_j with the labels y_i and y_j to produce the new sample $\lambda x_i + (1-\lambda)x_j$ with the label $\lambda y_i + (1-\lambda)y_j$ where $\lambda \sim Beta(\alpha, \alpha)$. According to Zhang et al. [53], mixing all samples outperforms Empirical Risk Minimization on many classification datasets. This strategy works well for classification where the labels are one-hot encoded where many samples may have the same label. In regression, however, such linear interpolations are not suitable as labels are in a continuous space instead of a discrete space where one-hot encodings cannot be used. Here two distant samples may have labels that are arbitrarily different, and simply mixing them could result in intermediate labels that are very different than the actual label.

C-Mixup. C-Mixup [50] overcomes the limitation of Mixup and is the state-of-the-art approach for regression tasks on clean data. C-Mixup proposes a sampling probability distribution for each sample based on the label distance between a neighbor using a symmetric Gaussian kernel. C-Mixup selects a sample to mix following a sampling probability and generates a new sample and label pair similar to Mixup. The sampling probability introduced by C-Mixup is:

$$P((x_j, y_j)|(x_i, y_i)) \propto \exp\left(-\frac{d(y_i, y_j)}{b^2}\right) \quad (1)$$

where $d(y_i, y_j)$ is the label distance and b is the bandwidth of a kernel function. Since the values of a probability mass function sum to one, C-Mixup normalizes the calculated values. The bandwidth is the key parameter to tune. As the bandwidth increases, the probability distribution becomes more uniform, thereby making C-Mixup similar to Mixup. As the bandwidth decreases, samples are only mixed with their nearest neighbors, and C-Mixup eventually becomes identical to Empirical Risk Minimization (ERM).

Robust Training. There is an existing literature [7, 16, 20, 41, 43, 44] on robust training where the goal is to perform accurate model training against noisy data. While there are many approaches, we focus on *multi-round robust training* where noisy data is either removed or fixed progressively during the model training iterations.

As a default, we use Iterative Trimmed Loss Minimization (ITLM) [43] as a representative clean sample selection method, although we can use other methods as we demonstrate in Section 6.5. ITLM repeatedly removes a certain percentage of the training data where the intermediate model's predictions differ from the labels and solves the following optimization problem:

$$\min_{S:|S|=\lfloor \tau n \rfloor} \sum_{s_i \in S} \ell_\theta(s_i)$$

where $S \subseteq \mathcal{D}$ is the current clean data, s_i is a sample in S, ℓ_θ is a loss function, and τ is a parameter that indicates the ratio of clean samples in the training data. ITLM is widely used because of its simplicity and scalability.

However, any other multi-round robust training technique [6, 18, 44] can be used as well. We demonstrate this point in Section 6.5 where we replace ITLM with O2U-Net [18], which is a different clean sample selection method, and SELFIE [44], which refurbishes labels of unclean samples.

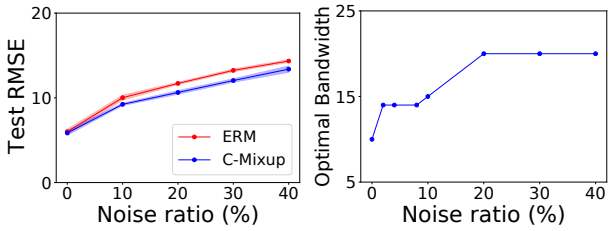
3 C-MIXUP VULNERABILITY TO NOISY DATA

C-Mixup assumes that all samples are clean data and performs mixup using a fixed bandwidth. Although C-Mixup is known to be robust against covariate shifts in the data where the distribution $P(X)$ may change, but not $P(Y|X)$, the data is still assumed to be clean. In comparison, we consider noisy data where $P(Y|X)$ may change as well.

We demonstrate how C-Mixup is vulnerable to such general noise in Figure 2a. We experiment on the Spectrum dataset for manufacturing and add random Gaussian noise to the labels (see more details in Section 6). We evaluate C-Mixup with the tuned bandwidth on noisy data. This model is then evaluated on clean test data. We compare C-Mixup with using ERM only and compare the Root Mean Squared Error (RMSE, see definition in Section 6) results where a lower value is better. As the noise ratio increases, the RMSE of C-Mixup trained on noisy data increases. In addition, regardless of the noise ratio, the RMSE of C-Mixup increases as much as ERM's RMSE does, showing the vulnerability of C-Mixup.

To better understand how C-Mixup is affected by noise, we also determine the optimal bandwidth, which results in the best model performance across the different noise ratios as shown in Figure 2b using the same experimental setup. As a result, as the noise ratio increases, the optimal bandwidth also increases. This result is counter-intuitive at first glance because it seems like noisy data should be mixed less. However, we suspect that mixing also has a dilution effect where out-of-distribution samples have less impact on the model training if they are mixed with other clean samples. We do not claim that this trend always holds, and the point is that mixing has a non-trivial effect on noisy data. Thus, it becomes difficult to find a single bandwidth that works best for both clean and noisy data.

The C-Mixup paper [50] also performs a robust training experiment against label noise, but it assumes a fixed noise ratio and does not necessarily show the entire story. The more extensive results here suggest that C-Mixup is not designed to handle noise and can benefit from robust training methods. We thus propose a natural extension by integrating C-Mixup with robust training.



(a) (b)

Figure 2: We evaluate C-Mixup on noisy data where we add label noise to the Spectrum dataset. A lower RMSE means better model performance. (a) As the noise ratio increases, both ERM and C-Mixup gradually perform worse where their performance gap does not change much. (b) In addition, the optimal bandwidth may actually increase where mixing dilutes out-of-distribution data from negatively impacting the model performance.

4 PROBLEM DEFINITION

We formulate our problem as solving the bilevel optimization problem of cleaning and augmenting samples during model training:

$$\min_{\theta} \sum_{s_i \in \texttt{C-Mixup}(S_c, b, \alpha)} \ell_\theta(s_i) \tag{2}$$

$$S_c = \underset{S=\{s_i | p_i=1\}}{\arg\min} \sum_{i=1}^{n} \ell_\theta(s_i)\, p_i \tag{3}$$

$$\text{s.t.} \sum_{i=1}^{n} p_i = \tau n$$

$$p_i \in \{0, 1\}, i = 1, ..., n$$

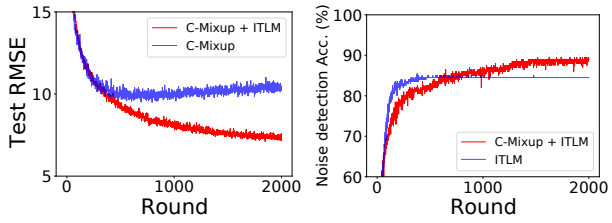
where p_i indicates whether the data sample s_i is selected or not, τ is the ratio of clean samples in the training data, S is the selected samples, $\texttt{C-Mixup}(S)$ is an augmented result of S by C-Mixup, b is the bandwidth, and α is the Mixup parameter.

5 RC-MIXUP

We explain how we interleave data augmentation (C-Mixup) with robust training, empirically analyze how the two techniques have a synergistic effect, propose bandwidth tuning techniques, and present the entire RC-Mixup algorithm.

5.1 C-Mixup and Robust Training Integration

We solve the bilevel optimization problem by interleaving C-Mixup with the robust training rounds. This approach is common in other bilevel optimization works [31, 40]. Recall that robust training iteratively cleans data and then updates its regression model until the data is clean enough. For each round, we can perform C-Mixup



(a) (b)

Figure 3: Robust training benefits C-Mixup and vice versa. (a) As robust training iteratively cleans the data, C-Mixup's model performance improves. (b) Using C-Mixup within robust training helps it remove noisy data better compared to when not using C-Mixup.

between the data cleaning and model updating steps as in Figure 1. We explain how this sequence is beneficial to both methods.

Data Cleaning benefits C-Mixup. We provide a simple empirical analysis of how data cleaning during robust training benefits C-Mixup in Figure 3a. We experiment on the Spectrum dataset used above and add random Gaussian noise to the labels, independently per label. We start with a 10% noise ratio and run ITLM, which progressively cleans the data. For each round, we evaluate C-Mixup by training a model on the mixed data. As a result, when combining C-Mixup with robust training, the model's RMSE on the validation set clearly improves with each round, unlike when using C-Mixup only. In addition, while using C-Mixup only has a slight performance decrease after convergence due to overfitting to the noisy data, C-Mixup with robust training does not have this problem.

C-Mixup benefits Data Cleaning Performance. We also empirically analyze how C-Mixup improves robust training by making it identify clean data better in Figure 3b. We use the same experimental setup as above and evaluate the noise detection accuracy of ITLM, which is the percent of noisy samples that are correctly identified by robust training. We compare ITLM with when it is combined with C-Mixup. As a result, C-Mixup improves ITLM's noise detection accuracy by up to 5%.

5.2 Dynamic Bandwidth Tuning

As we analyzed in Figure 2b, the optimal bandwidth of C-Mixup may vary as the data is progressively cleaned via robust training, so we would like to dynamically adjust this value both accurately and efficiently. Recall that the bandwidth adjusts the level of mixing where a larger bandwidth means that samples are mixed more with neighbors. Following the original C-Mixup setup [50], we consider a fixed set of bandwidth candidates $B = \{b_1, b_2, \ldots, b_{|B|}\}$ and aim to select the best one.

Our strategy is to update the best bandwidth after every $L > 0$ rounds of robust training. In the initial warm-up phase, we opt for a higher bandwidth to accommodate the relatively greater noise ratio. Subsequently, for each update, we evaluate bandwidth candidates within B on the next $N > 0$ rounds and then choose the one that results in the best model performance. Analytically predicting model performance after multiple rounds is challenging because the cleaned data itself keeps on changing. While this strategy has an overhead, we later show in Section 6.3 that only one or two initial updates are needed to achieve most of the performance benefits.

One way to further reduce the overhead is to avoid the bandwidth searching altogether and instead simply decay the bandwidth by a certain ratio (say by 10%) every L epochs based on the observation that the optimal bandwidth tends to decrease for lower noise ratios. Naturally there is be a tradeoff between runtime and performance, which we show in Section 6.1. In addition, we now need to figure out the right decay rate. Nonetheless, if lowering runtime is critical, the decaying strategy can be a viable option.

5.3 Overall Algorithm

Algorithm 1 shows the overall RC-Mixup algorithm when using ITLM as the robust training method. We first initialize model parameters using C-Mixup (Steps 2–4). The initial bandwidth is tuned on the validation set. This bootstrapping is important because the later steps are designed to gradually update its value as the data is cleaned progressively, so we need to start from a bandwidth value that is optimal on the noisy data first. Next, for each round, robust training cleans the data, we run C-Mixup and update the model on the cleaned data (Steps 18–20). After every L rounds, we also update the bandwidth of C-Mixup by evaluating the possible bandwidths on N rounds and choosing the one that results in the lowest validation set RMSE (Steps 8–16). We repeat the entire process until the model converges. Algorithm 2 shows the clean sample selection algorithm in ITLM, and Algorithm 3 shows the C-Mixup algorithm.

Overhead on Robust Training. In comparison to robust training, RC-Mixup adds the overhead of mixing samples via C-Mixup during each round. For every L rounds, there is also the overhead of tuning the bandwidth using N rounds of training. In our experiments, we observe that the tuning overhead is small because the bandwidth only needs to be updated once or twice.

6 EXPERIMENTS

We provide experimental results for RC-Mixup. We evaluate the regression models trained on the augmented training sets on separate test sets. We use Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE) for measuring model performance where lower values are better. We report the mean and the standard deviation (± in tables) for results of five random seeds. We use PyTorch [35], and all experiments are performed using Intel Xeon Silver 4210R CPUs and NVIDIA Quadro RTX 8000 GPUs.

Metrics. RMSE is defined as $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$. MAPE is defined as $\frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100$ and is a measure of prediction performance of forecasting methods.

Experimental environment. We conducted all experiments using Intel(R) Xeon(R) Silver 4210R CPUs @ 2.40GHz, and eight NVIDIA Quadro RTX 8000 GPUs equipped with 48GB VRAM on Linux Ubuntu 18.04.5. We employed Python 3.8.13, Scikit-learn version 0.24.2, and PyTorch version 1.7.1 for our evaluations.

Datasets. We use one synthetic and three real datasets. The Spectrum dataset [10] is synthetic and contains spectrum data generated

Algorithm 1: The RC-Mixup algorithm.

Input: Training data $\mathcal{D}$, validation set $\mathcal{D}^v$, bandwidth update interval L, number of bandwidth update rounds N, possible bandwidths $B = \{b_1, \ldots, b_{|B|}\}$, clean ratio τ, Mixup parameter α, initial bandwidth b, initial model parameters θ

// Warm-up phase

for round = 1 to *warm-up rounds* **do**
  S_{mix} = C-Mixup($\mathcal{D}, b, \alpha$)
  Update model parameters θ on S_{mix}

$i \leftarrow 0$

// Clean & Update phase

while not converge **do**
  if i % L == 0 **then**
    // Update bandwidth
    for b_j in B **do**
      $\theta_{b_j} \leftarrow \theta$
      for round = 1 to N **do**
        C = CleanSelection($\mathcal{D}, \theta_{b_j}, \tau$)
        S_{mix} = C-Mixup(C, b_j, α)
        Update model parameters θ_{b_j} on S_{mix}
    (b^*, θ_{b^*}) = Bandwidth and corresponding model parameters with the lowest RMSE
    $\theta \leftarrow \theta_{b^*}$
    $b \leftarrow b^*$
  else
    C = CleanSelection($\mathcal{D}, \theta, \tau$)
    S_{mix} = C-Mixup(C, b, α)
    Update model parameters θ on S_{mix}
  $i = i + 1$

Output: θ

Algorithm 2: The clean sample selection algorithm.

Input: Dataset D, model parameters θ, clean ratio τ

sortIdx = argSort($\ell_\theta(D)$) (ascending order)

$C \leftarrow$ first $\tau|D|$ elements of D[sortIdx]

Output: C

by applying light waves on 4-layer 3D semiconductors and measuring the returning wavelengths. This data is used to predict layer thickness without touching the semiconductor itself. The NO2 emissions dataset [1] contains traffic and meteorological information around roads and is used to predict NO2 concentration. The Airfoil dataset [11] contains aerodynamic and acoustic test results for airfoil blade sections in a wind tunnel. Finally, the Exchange-Rate [29] is a time-series dataset and contains daily exchange rates from 8 countries. The three real datasets are from [50] for a fair comparison. Table 1 shows the dimension and size information of the datasets.

Noise Injection. We inject noise to labels using two methods: (1) Gaussian noise, which adds to each label a value sampled from the Gaussian distribution $N(0, m^2\sigma^2)$, where m is the noise magnitude

Algorithm 3: The C-Mixup algorithm.

Input: Dataset D, bandwidth b, Mixup parameter α

$S \leftarrow []$

for (x_i, y_i) in D **do**
  Sample (x_j, y_j) using Equation 1 and b
  Sample $\lambda \sim Beta(\alpha, \alpha)$
  $\tilde{x} = \lambda x_i + (1 - \lambda) x_j$
  $\tilde{y} = \lambda y_i + (1 - \lambda) y_j$
  $S \leftarrow S \cup \{(\tilde{x}, \tilde{y})\}$

Output: S

Table 1: Settings for the four datasets.

Dataset	Data dim.	Label dim.	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{D}^v\rvert$
Spectrum	226	4	2,000	500
NO2	7	1	200	200
Airfoil	5	1	1,000	400
Exchange-Rate	168×8	8	4,373	1,518

(see default values in the appendix), and σ is the standard deviation of labels of training set and (2) labeling flipping noise, which subtracts a maximum label value by each label as in classification [37]. We use a noise rate of 10–40% where the default is 30%. We do not set the noise ratio to be larger than 50% to prevent the noise from dominating the data.

Baselines. We consider two types of baselines:

- *Individual methods*: performing ITLM [43] only ("Rob. training") and performing C-Mixup [50] only ("C-Mixup").
- *Simple combinations*: performing robust training first and C-Mixup in sequence ("R→C"), performing C-Mixup and then robust training in sequence ("C→R"), and performing RC-Mixup without dynamic bandwidth tuning ("C→R+C").

Parameters. For robust training, we assume that the clean ratio τ is known for each dataset. If τ is unknown, it can be inferred with cross validation [32, 51]. For the C-Mixup, C→R, and C→R+C baselines, we use a single bandwidth value and tune it using a grid search. We choose other parameters using a grid search on the validation set or following C-Mixup's setup. More details on parameters are in the appendix.

6.1 Model Performance and Runtime Results

We compare the overall performance RC-Mixup with the two types of baselines on all the datasets as shown in Table 2. For the individual method baselines, we observe that C-Mixup is indeed vulnerable to noise, and that robust training is less affected by the noise, but still needs improvement. The simple combination baselines C→R and C→R+C increasingly outperform the individual methods as they start to take advantage of both methods. However, RC-Mixup performs the best by also dynamically tuning the bandwidth. We also note that our results are near-optimal. As a reference, the (RMSE, MAPE) results in an optimal setting where we only use clean data are: (6.961, 6.000) for Spectrum and (0.535, 13.209) for

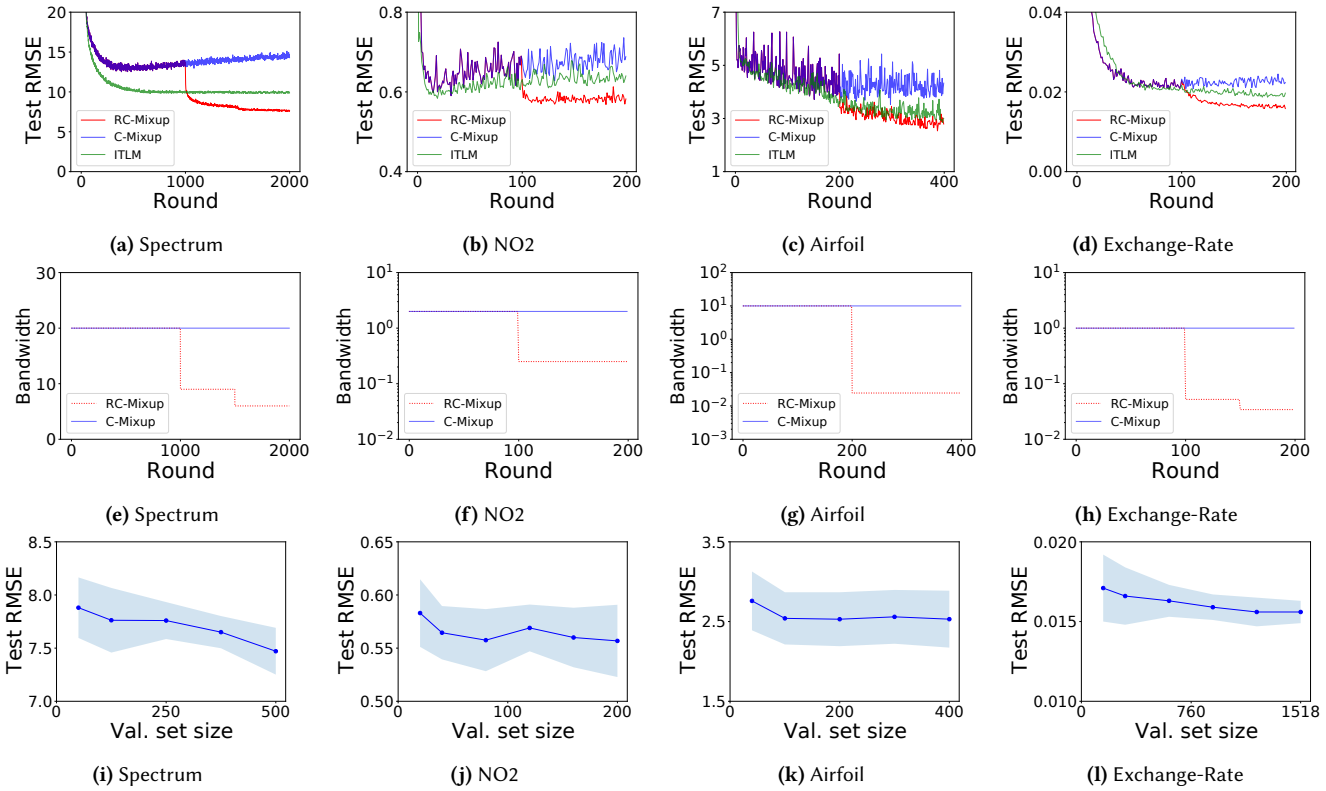


Figure 4: (a)-(d) RC-Mixup model training convergence results. (e)-(h) RC-Mixup dynamic bandwidth tuning results. The tuned bandwidth values vary slightly depending on the random seed, and we show the bandwidth averaged across five random seeds (dotted red lines). (i)-(l) RC-Mixup performance results while varying the validation set size.

NO2. The RC-Mixup results are similar to these results and cannot be further improved.

We also show how RC-Mixup's model training converges and how its bandwidth is tuned in Figures 4a–4d. For all four datasets, RC-Mixup clearly converges to lower RMSE values for the same number of epochs compared to C-Mixup and robust training. RC-Mixup's RMSE values drastically decrease around the middle of each figure because robust training and bandwidth tuning are applied at that point. Once robust training is applied, the model training is now done on cleaned data, so the model performance improves significantly afterwards. We then show how RC-Mixup dynamically adjusts its bandwidth for the four datasets with five random seeds in Figures 4e–4h. For these datasets, the bandwidth values vary slightly depending on the random seed, and we show the bandwidth averaged across five random seeds (dotted red lines). The decreasing trends of the tuned bandwidths are consistent with Figure 2b, where as the data is cleaned, a smaller bandwidth is more effective. We do not claim this trend always holds, and the point is that RC-Mixup is able to find the right bandwidth in any situation.

Runtime. The computational cost of RC-Mixup varies depending on the number of bandwidth candidates or bandwidth update frequency. For the Spectrum dataset, the runtimes of all methods are as follows: robust training (ITLM) only: 52s, C-Mixup only: 244s, C→R: 161s, C→C+R: 250s, and RC-Mixup: 578s. Although RC-Mixup requires roughly twice the runtime of C-Mixup, the substantial performance improvement of RC-Mixup justifies its overhead. In general, while the efficiency decreases as the search space grows, we can obtain performance improvements with only 4–7 bandwidth candidates in practice. As a reference, C-Mixup also searches for the optimal bandwidth among 6–10 bandwidth candidates using a grid search.

Bandwidth Decaying Results. We also show RC-Mixup's performance and runtime when using bandwidth decaying in Table 3. Here we decrease the bandwidth by 10% per update. As a result, there is a tradeoff between runtime and performance where the decaying strategy is 1.4-2.6x faster than the original RC-Mixup, but has slightly worse RMSE and MAPE results that are still better than those of the baselines.

6.2 Varying Noise

We evaluate how robust RC-Mixup is against different types and levels of noise on the Spectrum dataset in Table 4. We sample different levels of Gaussian noise by varying the noise magnitude n and also use label flipping. As a result, RC-Mixup consistently performs better than C-Mixup as it is less affected by noise with its bandwidth tuning. We also evaluate RC-Mixup's robustness while varying the noise rate on the Spectrum dataset from 10% to 40% in

Table 2: RC-Mixup performance compared to the five baselines on the real and synthetic datasets.

Dataset	Method	RMSE	MAPE
Spectrum	C-Mixup	$12.125_{\pm 0.200}$	$10.840_{\pm 0.246}$
	Rob. training	$9.756_{\pm 0.541}$	$7.228_{\pm 0.221}$
	R→C	$9.055_{\pm 0.445}$	$6.806_{\pm 0.335}$
	C→R	$8.024_{\pm 0.321}$	$6.468_{\pm 0.211}$
	C→R+C	$8.755_{\pm 0.187}$	$7.031_{\pm 0.067}$
	RC-Mixup	$\mathbf{7.471}_{\pm 0.220}$	$\mathbf{5.930}_{\pm 0.165}$
NO2	C-Mixup	$0.586_{\pm 0.033}$	$14.604_{\pm 1.283}$
	Rob. training	$0.574_{\pm 0.028}$	$14.418_{\pm 1.551}$
	R→C	$0.573_{\pm 0.023}$	$14.340_{\pm 1.258}$
	C→R	$0.562_{\pm 0.038}$	$14.020_{\pm 1.562}$
	C→R+C	$0.566_{\pm 0.033}$	$14.203_{\pm 1.549}$
	RC-Mixup	$\mathbf{0.557}_{\pm 0.034}$	$\mathbf{13.816}_{\pm 1.440}$
Airfoil	C-Mixup	$3.438_{\pm 0.218}$	$2.093_{\pm 0.185}$
	Rob. training	$2.760_{\pm 0.329}$	$1.529_{\pm 0.069}$
	R→C	$3.226_{\pm 0.260}$	$1.833_{\pm 0.136}$
	C→R	$2.721_{\pm 0.463}$	$1.492_{\pm 0.235}$
	C→R+C	$2.699_{\pm 0.381}$	$1.501_{\pm 0.167}$
	RC-Mixup	$\mathbf{2.530}_{\pm 0.357}$	$\mathbf{1.398}_{\pm 0.123}$
Exchange-Rate	C-Mixup	$0.0216_{\pm 0.0018}$	$2.2931_{\pm 0.2153}$
	Rob. training	$0.0180_{\pm 0.0008}$	$1.7715_{\pm 0.1213}$
	R→C	$0.0165_{\pm 0.0015}$	$1.5825_{\pm 0.1821}$
	C→R	$0.0162_{\pm 0.0011}$	$1.5553_{\pm 0.1428}$
	C→R+C	$0.0162_{\pm 0.0011}$	$1.5495_{\pm 0.1214}$
	RC-Mixup	$\mathbf{0.0156}_{\pm 0.0007}$	$\mathbf{1.4692}_{\pm 0.0811}$

Table 3: RC-Mixup with bandwidth decaying ("With BD") compared to the RC-Mixup on the real and synthetic datasets.

Dataset	Method	RMSE	MAPE	Runtime(s)
Spectrum	RC-Mixup	$7.471_{\pm 0.220}$	$5.930_{\pm 0.165}$	578
	With BD	$7.903_{\pm 0.144}$	$6.284_{\pm 0.126}$	247
NO2	RC-Mixup	$0.557_{\pm 0.034}$	$13.816_{\pm 1.440}$	14
	With BD	$0.556_{\pm 0.029}$	$13.974_{\pm 1.318}$	7
Airfoil	RC-Mixup	$2.530_{\pm 0.357}$	$1.398_{\pm 0.123}$	188
	With BD	$2.582_{\pm 0.244}$	$1.446_{\pm 0.111}$	73
Exchange-Rate	RC-Mixup	$0.0156_{\pm 0.0007}$	$1.4692_{\pm 0.0811}$	512
	With BD	$0.0158_{\pm 0.0010}$	$1.5093_{\pm 0.1116}$	371

Table 5 and observe results that are consistent with when using the default noise rate of 30%.

6.3 Parameter Analysis

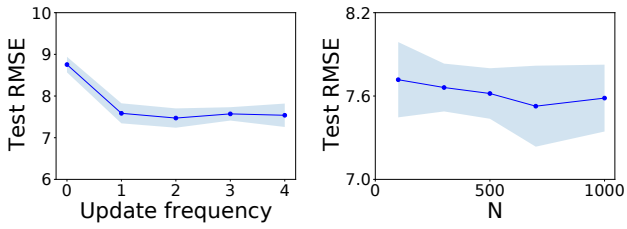
We vary the bandwidth tuning parameters L and N to see how RC-Mixup performs in different scenarios on the Spectrum dataset. The results on the other datasets are similar. As we decrease L, we update the bandwidth more frequently, which means that it is more likely to be up-to-date. Figure 5a indeed shows that the RMSE decreases against the number of updates, but only for the first one or two updates. Increasing N means that we evaluate the bandwidths using more epochs before selecting one of them. Figure 5b shows that a higher N leads to lower RMSE, but with diminishing returns. Hence, RC-Mixup achieves sufficient performance improvements even when both L and N are small, which means that the additional overhead for bandwidth tuning of RC-Mixup is not large.

Table 4: RC-Mixup robustness against different types and levels of noise on the Spectrum dataset.

Noise Type	Magnitude (n value)	Method	RMSE	MAPE
Gaussian	Low (1)	C-Mixup	10.044	9.031
		RC-Mixup	7.442	6.029
	Medium (2)	C-Mixup	12.125	10.840
		RC-Mixup	7.471	5.930
	High (3)	C-Mixup	13.074	11.660
		RC-Mixup	7.941	6.240
Label Flipping	n/a	C-Mixup	18.259	17.520
		RC-Mixup	7.893	6.124

Table 5: RC-Mixup robustness against different Gaussian noise rates on the Spectrum dataset.

Noise Rate	Method	RMSE	MAPE
10%	C-Mixup	$9.291_{\pm 0.129}$	$8.210_{\pm 0.066}$
	RC-Mixup	$6.100_{\pm 0.118}$	$5.090_{\pm 0.119}$
20%	C-Mixup	$10.721_{\pm 0.202}$	$9.583_{\pm 0.232}$
	RC-Mixup	$6.729_{\pm 0.155}$	$5.493_{\pm 0.095}$
40%	C-Mixup	$13.376_{\pm 0.369}$	$12.257_{\pm 0.325}$
	RC-Mixup	$8.524_{\pm 0.342}$	$6.784_{\pm 0.293}$



(a) Varying L **(b) Varying N**

Figure 5: RC-Mixup performance when varying L and N used for bandwidth tuning on the Spectrum dataset.

6.4 Validation Set Construction and Size

We evaluate RC-Mixup by varying the validation set size on all the four datasets in Figures 4i–4l. As a result, RC-Mixup works reasonably well even for smaller validation set sizes. If there is no clean validation set readily available, we can utilize the given robust

Table 6: RC-Mixup performance when using a clean validation set, a noisy validation set that is cleaned with robust training (RT), and a noisy validation set.

Dataset	Validation set	RMSE	MAPE
Spectrum	Clean	$7.471_{\pm 0.220}$	$5.930_{\pm 0.165}$
	Cleaned with RT	$7.739_{\pm 0.262}$	$6.209_{\pm 0.225}$
	Noisy	$8.176_{\pm 0.544}$	$6.625_{\pm 0.659}$
Airfoil	Clean	$2.530_{\pm 0.357}$	$1.398_{\pm 0.123}$
	Cleaned with RT	$2.684_{\pm 0.453}$	$1.471_{\pm 0.136}$
	Noisy	$2.692_{\pm 0.244}$	$1.547_{\pm 0.167}$

training method to construct a validation set ourselves. In Table 6, we compare RC-Mixup's RMSE on two datasets when using (1) a clean validation set; (2) a noisy validation set that is cleaned with robust training first; and (3) a noisy validation set that is a subset of the training set. We compare with (3) just as a reference. As a result, RC-Mixup using (2) has slightly lower, but a comparable performance as when using (1), which means that RC-Mixup can still perform well without a given clean validation set with an additional overhead of cleaning the noisy validation set.

6.5 Other Robust Training Methods

To show the generality of RC-Mixup, we integrate C-Mixup with the two other robust training methods O2U-Net [18] and SELFIE [44] and provide more evaluation results. RC-Mixup can be plugged into any other multi-round robust training method as well.

O2U-Net Integration. O2U-Net consists of two phases for selecting clean samples: a pre-training phase and a cyclical training phase between underfitting and overfitting. During the pre-training phase, we train the model using C-Mixup. In the cyclical training phase, we further train this pre-trained model by adjusting the learning rate cyclically to obtain the clean samples. We then return the pre-trained C-Mixup model trained on the final clean samples with bandwidth tuning.

SELFIE Integration. SELFIE refurbishes the labels of unclean samples with low predictive uncertainty by assigning them the most frequent class among the previous q predictions. SELFIE was designed for classification, so we extend it to work for regression.

While SELFIE calculates uncertainty using the entropy of the predictive categorical distribution, this approach cannot be directly applied in a regression setting due to the absence of the categorical distribution. Instead of using entropy, we quantify uncertainty by assessing the variance of predictive labels over the past q predictions. This approach is consistent with other common regression techniques [12, 30], where the uncertainty corresponds to the variation in predictions across multiple instances or models.

When refurbishing a label of an unclean sample, we average the previous predictions instead of finding the most frequent label. The reason is that in regression, labels have continuous values, so it makes less sense to find the most frequent labels as in classification.

Finally, since SELFIE refurbishes the labels for every mini-batch, the label distances between two labels change, which means the sampling probabilities for C-Mixup may change as well. However, updating the distances between label pairs requires significant amounts of computation, and we thus choose not to update the sampling probabilities once they are initially computed. This approach is reasonable because only a fraction of labels that have low uncertainties are actually refurbished. Even if we do update the sampling probabilities, our experiments show that they have a minor impact on the model's performance. Nonetheless, incrementally updating the sampling probabilities efficiently is an interesting future work.

Table 7: RC-Mixup performance using the robust training methods O2U-Net [18] and SELFIE [44] on the four datasets.

Dataset	Method	RMSE	MAPE
Spectrum	C-Mixup only	$12.125_{\pm 0.200}$	$10.840_{\pm 0.246}$
	O2U-Net only	$9.386_{\pm 0.360}$	$7.867_{\pm 0.321}$
	RC-Mixup w/ O2U-Net	$8.372_{\pm 0.316}$	$6.702_{\pm 0.144}$
	SELFIE only	$9.040_{\pm 0.225}$	$7.000_{\pm 0.178}$
	RC-Mixup w/ SELFIE	$7.701_{\pm 0.138}$	$6.234_{\pm 0.186}$
NO2	C-Mixup only	$0.586_{\pm 0.033}$	$14.604_{\pm 1.283}$
	O2U-Net only	$0.564_{\pm 0.033}$	$14.052_{\pm 1.617}$
	RC-Mixup w/ O2U-Net	$0.554_{\pm 0.022}$	$13.782_{\pm 1.285}$
	SELFIE only	$0.559_{\pm 0.043}$	$14.002_{\pm 1.580}$
	RC-Mixup w/ SELFIE	$0.550_{\pm 0.030}$	$13.562_{\pm 1.247}$
Airfoil	C-Mixup only	$3.438_{\pm 0.218}$	$2.093_{\pm 0.185}$
	O2U-Net only	$3.026_{\pm 0.199}$	$1.783_{\pm 0.073}$
	RC-Mixup w/ O2U-Net	$2.795_{\pm 0.379}$	$1.574_{\pm 0.145}$
	SELFIE only	$2.713_{\pm 0.426}$	$1.508_{\pm 0.174}$
	RC-Mixup w/ SELFIE	$2.569_{\pm 0.358}$	$1.408_{\pm 0.165}$
Exchange-Rate	C-Mixup only	$0.0216_{\pm 0.0018}$	$2.2931_{\pm 0.2153}$
	O2U-Net only	$0.0203_{\pm 0.0015}$	$2.0909_{\pm 0.1676}$
	RC-Mixup w/ O2U-Net	$0.0162_{\pm 0.0011}$	$1.5523_{\pm 0.1407}$
	SELFIE only	$0.0165_{\pm 0.0004}$	$1.5741_{\pm 0.0666}$
	RC-Mixup w/ SELFIE	$0.0154_{\pm 0.0009}$	$1.4535_{\pm 0.1137}$

Evaluation. We now evaluate RC-Mixup using O2U-Net and SELFIE on the four datasets in Table 7. As a result, RC-Mixup improves both C-Mixup and robust training as in Table 7 demonstrating the synergy between the two for all four datasets.

7 RELATED WORK

There are largely two branches of work for data augmentation in regression. One is semi-supervised regression [23, 26, 28, 54] where the goal is to utilize unlabeled data for training. Another branch is data augmentation when there is no unlabeled data, which is our research focus. Most data augmentation techniques are tailored to classification and more recently a few are designed for regression.

Data Augmentation for Classification. There are largely three approaches: generative models, policies, and Mixup techniques. Generative models including GANs [14] and VAEs [27] are popular in classification where the idea is to generate realistic data that

cannot be distinguished from the real data by a discriminator. Another approach is to use policies [9], which specify fixing rules for transforming the data. However, a major assumption is that the labels of these generated samples are the same, which does not necessarily hold in a regression setting where most samples may have different labels. Mixup [24, 25, 47, 52, 53] takes the alternative approach of generating both data and labels together by mixing existing samples with different labels assuming linearity between training samples [5, 48].

Data Augmentation for Regression. There is a new and increasing literature on data augmentation for regression using Mixup techniques. Although the original Mixup paper mentions that its techniques can easily be extended to regression, the linearity of a regression model is limited where mixing samples with all others may not be beneficial and even detrimental to model performance. Hence, the key is to figure out how to limit the mixing. RegMix [19] learns for each sample how many nearest neighbors in terms of data distance it should be mixed with for the best model performance using a validation set. The state-of-the-art C-Mixup [50] uses a Gaussian kernel to generate a sampling probability distribution for each sample based on label distances using a global bandwidth and selects a sample to mix based on the distribution. Anchor Data Augmentation [42] takes a more domain-specific approach where it clusters data points and modifies original points either towards or away from the cluster centroids for the augmentation. Finally, R-Mixup [22] specializes in improving model performance on biological networks. In comparison, RC-Mixup uses the domain-agnostic C-Mixup and makes it robust against noise.

Robust Training. The goal of robust training is to train accurate models against noisy or adversarial data, and we cover the representative works. While data can be problematic in various places [49], most techniques assume that the labels are noisy [45] and mitigate with the following strategies: (1) developing model architectures that are less affected by the noise [3, 8, 15, 21], (2) applying loss regularization techniques to reduce overfitting [13, 17, 34, 39, 46], (3) correcting the loss function to account for the noise [2, 4, 33, 36], and (4) proposing sample selection techniques [18, 43, 44] for selecting clean samples from noisy data. In comparison, RC-Mixup is mainly designed to work with the sample selection techniques, but can also complement other approaches as long as it can access intermediate clean data within rounds.

8 CONCLUSION

We proposed RC-Mixup, an effective and practical data augmentation strategy against noisy data for regression. The state-of-the-art data augmentation technique C-Mixup is not designed to handle noise, and RC-Mixup is the first to tightly integrates it with robust training for a synergistic effect. C-Mixup benefits from the intermediate clean data information identified by robust training and performs different mixing depending on whether noisy data is being used, while robust training cleans its data better with C-Mixup's data augmentation. We also proposed dynamic tuning techniques for C-Mixup's bandwidth. Our extensive experiments showed how RC-Mixup significantly outperforms C-Mixup and robust training baselines on noisy data benchmarks and is compatible with various robust training methods.

REFERENCES

[1] Magne Aldrin. 2004. CMU StatLib Dataset. http://lib.stat.cmu.edu/datasets/. Accessed: 2022-08-15.
[2] Eric Arazo, Diego Ortego, Paul Albert, Noel O'Connor, and Kevin McGuinness. 2019. Unsupervised label noise modeling and loss correction. In *ICML*. 312–321.
[3] Alan Joseph Bekker and Jacob Goldberger. 2016. Training deep neural-networks based on unreliable labels. In *IEEE ICASSP*. 2682–2686.
[4] Haw-Shiuan Chang, Erik Learned-Miller, and Andrew McCallum. 2017. Active Bias: Training More Accurate Neural Networks by Emphasizing High Variance Samples. In *NeurIPS*.
[5] Olivier Chapelle, Jason Weston, Léon Bottou, and Vladimir Vapnik. 2000. Vicinal Risk Minimization. In *NIPS*. 416–422.
[6] Pengfei Chen, Benben Liao, Guangyong Chen, and Shengyu Zhang. 2019. Understanding and Utilizing Deep Neural Networks Trained with Noisy Labels. In *ICML*.
[7] Pengfei Chen, Ben Ben Liao, Guangyong Chen, and Shengyu Zhang. 2019. Understanding and utilizing deep neural networks trained with noisy labels. In *ICML*. 1062–1070.
[8] Xinlei Chen and Abhinav Gupta. 2015. Webly supervised learning of convolutional networks. In *ICCV*. 1431–1439.
[9] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. 2019. AutoAugment: Learning Augmentation Strategies From Data. In *CVPR*. 113–123.
[10] DACON Co., Ltd. 2020. Challenge on Semiconductor Thin Film Thickness Analysis. https://dacon.io/competitions/official/235554/data. Accessed: 2023-07-12.
[11] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml
[12] Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*. PMLR, 1050–1059.
[13] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *ICLR*.
[14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *NIPS*. 2672–2680.
[15] Bo Han, Jiangchao Yao, Niu Gang, Mingyuan Zhou, Ivor Tsang, Ya Zhang, and Masashi Sugiyama. 2018. Masking: A new perspective of noisy supervision. In *NeurIPS*. 5839–5849.
[16] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *NeurIPS*. 8535–8545.
[17] Dan Hendrycks, Kimin Lee, and Mantas Mazeika. 2019. Using pre-training can improve model robustness and uncertainty. In *ICML*. 2712–2721.
[18] Jinchi Huang, Lie Qu, Rongfei Jia, and Binqiang Zhao. 2019. O2U-Net: A Simple Noisy Label Detection Approach for Deep Neural Networks. In *ICCV*. 3325–3333.
[19] Seong-Hyeon Hwang and Steven Euijong Whang. 2022. RegMix: Data Mixing Augmentation for Regression. arXiv:2106.03374 [cs.LG]
[20] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. 2018. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *ICML*. 2304–2313.
[21] Ishan Jindal, Matthew Nokleby, and Xuewen Chen. 2016. Learning deep networks from noisy labels with dropout regularization. In *IEEE ICDM*. 967–972.
[22] Xuan Kan, Zimu Li, Hejie Cui, Yue Yu, Ran Xu, Shaojun Yu, Zilong Zhang, Ying Guo, and Carl Yang. 2023. R-Mixup: Riemannian Mixup for Biological Networks. In *KDD*.
[23] Pilsung Kang, Dongil Kim, and Sungzoon Cho. 2016. Semi-supervised support vector regression based on self-training with label uncertainty: An application to virtual metrology in semiconductor manufacturing. *Expert Syst. Appl.* 51 (2016), 85–106.
[24] JangHyun Kim, Wonho Choo, Hosan Jeong, and Hyun Oh Song. 2021. Co-Mixup: Saliency Guided Joint Mixup with Supermodular Diversity. In *ICLR*.
[25] Jang-Hyun Kim, Wonho Choo, and Hyun Oh Song. 2020. Puzzle mix: Exploiting saliency and local statistics for optimal mixup. In *ICML*. PMLR, 5275–5285.
[26] Sung Wook Kim, Young Gon Lee, Bayu Adhi Tama, and Seungchul Lee. 2020. Reliability-Enhanced Camera Lens Module Classification Using Semi-Supervised Regression Method. *Applied Sciences* 10, 11 (2020).
[27] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *ICLR*.
[28] Georgios Kostopoulos, Stamatis Karlos, Sotiris Kotsiantis, and Omiros Ragos. 2018. Semi-supervised regression: A recent review. *J. Intell. Fuzzy Syst.* 35, 2 (2018), 1483–1500.
[29] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. 2018. Modeling long-and short-term temporal patterns with deep neural networks. In *SIGIR*.

95–104.
[30] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. *NeurIPS* 30 (2017).
[31] Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2019. DARTS: Differentiable Architecture Search. *ICLR* (2019).
[32] Tongliang Liu and Dacheng Tao. 2015. Classification with noisy labels by importance reweighting. *IEEE TPAMI* 38, 3 (2015), 447–461.
[33] Xingjun Ma, Yisen Wang, Michael E Houle, Shuo Zhou, Sarah Erfani, Shutao Xia, Sudanthi Wijewickrema, and James Bailey. 2018. Dimensionality-driven learning with noisy labels. In *ICML*. 3355–3364.
[34] Aditya Krishna Menon, Ankit Singh Rawat, Sashank J. Reddi, and Sanjiv Kumar. 2020. Can gradient clipping mitigate label noise?. In *ICLR*.
[35] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic Differentiation in PyTorch. In *NIPS Autodiff Workshop*.
[36] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. 2017. Making deep neural networks robust to label noise: A loss correction approach. In *CVPR*. 1944–1952.
[37] Andrea Paudice, Luis Muñoz-González, and Emil C. Lupu. 2018. Label Sanitization Against Label Flipping Poisoning Attacks. In *ECML PKDD*. 5–15.
[38] Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. 2021. Deep learning on a data diet: Finding important examples early in training. *NeurIPS* 34 (2021), 20596–20607.
[39] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. (2017).
[40] Yuji Roh, Kangwook Lee, Steven Whang, and Changho Suh. 2021. Sample selection for fair and robust training. *NeurIPS* 34 (2021), 815–827.
[41] Peter J. Rousseeuw. 1984. Least Median of Squares Regression. *J. Amer. Statist. Assoc.* 79, 388 (1984), 871–880.
[42] Nora Schneider, Shirin Goshtasbpour, and Fernando Perez-Cruz. 2023. Anchor Data Augmentation. In *NeurIPS*.
[43] Yanyao Shen and Sujay Sanghavi. 2019. Learning with Bad Training Data via Iterative Trimmed Loss Minimization. In *ICML*, Vol. 97. PMLR, 5739–5748.
[44] Hwanjun Song, Minseok Kim, and Jae-Gil Lee. 2019. SELFIE: Refurbishing Unclean Samples for Robust Deep Learning. In *ICML*, Vol. 97. PMLR, 5907–5915.
[45] Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. 2022. Learning From Noisy Labels With Deep Neural Networks: A Survey. *IEEE Transactions on Neural Networks and Learning Systems* (2022), 1–19.
[46] Ryutaro Tanno, Ardavan Saeedi, Swami Sankaranarayanan, Daniel C Alexander, and Nathan Silberman. 2019. Learning from noisy labels by regularized estimation of annotator confusion. In *CVPR*. 11244–11253.
[47] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. 2019. Manifold mixup: Better representations by interpolating hidden states. In *ICML*.
[48] Sen Wu, Hongyang R. Zhang, Gregory Valiant, and Christopher Ré. 2020. On the Generalization Effects of Linear Transformations in Data Augmentation. In *ICML*. 10410–10420.
[49] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. 2015. Learning from massive noisy labeled data for image classification. In *CVPR*. 2691–2699.
[50] Huaxiu Yao, Yiping Wang, Linjun Zhang, James Y. Zou, and Chelsea Finn. 2022. C-Mixup: Improving Generalization in Regression. In *NeurIPS*.
[51] Xiyu Yu, Tongliang Liu, Mingming Gong, Kayhan Batmanghelich, and Dacheng Tao. 2018. An efficient and provable approach for mixture proportion estimation using linear independence assumption. In *CVPR*. 4480–4489.
[52] Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Seong Joon Oh, Youngjoon Yoo, and Junsuk Choe. 2019. CutMix: Regularization Strategy to Train Strong Classifiers With Localizable Features. In *ICCV*. 6022–6031.
[53] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. 2018. mixup: Beyond Empirical Risk Minimization. In *ICLR*.
[54] Zhi-Hua Zhou and Ming Li. 2005. Semi-Supervised Regression with Co-Training.. In *IJCAI*. 908–913.

A APPENDIX – EXPERIMENTS

A.1 Other Experimental Settings

We explain more detailed experimental setups.

Hyperparameters. We summarize the hyperparameters for all the datasets in Table 8. For each dataset, we either use the traditional notion of Mixup or the more recent Manifold Mixup [47] (ManiMix), whichever performs better. We use a 3-layer fully connected neural network with one hidden layer with 128 nodes (FCN3) for the Spectrum, NO2, and Airfoil datasets. For the Exchange-Rate dataset, we build an LST-Attn model [29] with a horizon value of 12.

Table 8: Detailed hyperparameters for each dataset used in the experiments.

Dataset	Spectrum	NO2	Airfoil	Exchange-Rate
Mixup type	Mixup	Mixup	ManiMix	Mixup
Batch size	128	32	16	128
Learning rate	1e-2	1e-2	1e-2	1e-3
Maximum epochs	1,000	100	200	100
B	{5, 10, 15, 20}	{1e-3, 5e-3, 1e-2, 5e-2, 1e-1, 5e-1, 1}	{1e-3, 1e-2, 1e-1, 1, 10}	{1e-3, 1e-2, 5e-2, 1e-1}
Default bandwidth	20	2	10	2
L	500	100	200	50
N	500	100	200	50
α for Mixup	2.0	2.0	0.5	2.0
Noise magnitude m	2	4	2	5
Model architecture	FCN3	FCN3	FCN3	LST-Attn
Optimizer	Adam	Adam	Adam	Adam

A.2 Noisy Versus Hard-to-train Data

While noisy data tends to have high loss values, so does hard-to-train data. However, noisy data typically has higher losses as shown in Paul et al. [38]. Here hard data is identified using the EL2N score, which is similar to loss. Paul et al. [38] analyze the challenge of distinguishing hard data from noisy data and shows that data with noisy labels tends to have higher EL2N scores. If there is a label noise, data with large loss values are commonly considered as noise [18, 43, 44].