MIXRL: DATA MIXING AUGMENTATION FOR REGRESSION USING REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Data augmentation is becoming essential for improving regression accuracy in critical applications including manufacturing and finance. Existing techniques for data augmentation largely focus on classification tasks and do not readily apply to regression tasks. In particular, the recent Mixup techniques for classification rely on the key assumption that linearity holds among training examples, which is reasonable if the label space is discrete, but has limitations when the label space is continuous as in regression. We show that mixing examples that either have a large data or label distance may have an increasingly-negative effect on model performance. Hence, we use the stricter assumption that linearity only holds within certain data or label distances for regression where the degree may vary by each example. We then propose MixRL, a data augmentation meta learning framework for regression that learns for each example how many nearest neighbors it should be mixed with for the best model performance using a small validation set. MixRL achieves these objectives using Monte Carlo policy gradient reinforcement learning. Our experiments conducted both on synthetic and real datasets show that MixRL significantly outperforms state-of-the-art data augmentation baselines. MixRL can also be integrated with other classification Mixup techniques for better results.

1 INTRODUCTION

As machine learning becomes widely used in critical applications including manufacturing and finance, data augmentation for regression becomes essential as it provides an opportunity to improve model performance without additional data collection. In comparison to classification tasks like object detection in images, the goal of regression is to predict one or more real numbers.

To emphasize the importance of data augmentation in regression, we provide a case study of semiconductor manufacturing. Here a common quality check is to measure the layer thicknesses of a 3-dimensional semiconductor and see if they are even. However, directly measuring each thickness results in destroying the semiconductor itself, so a recently-common approach is to take an indirect measurement by applying light waves on the semiconductor, measuring the spectrum of wavelengths that bounce back from all the layers, and use machine learning to predict the layer thicknesses from the spectrum data (see Fig. 4 in Sec. 4 for an illustration). With enough spectrum data and thickness information, machine learning models can be trained to accurately predict thicknesses from a spectrum. The main challenge is that there is not enough training data, and the only cost-effective solution is to augment small amounts of data that exist. Even a small improvement in model performance from the data augmentation has significant impact in this industry. In general, any regression task that predicts real values like emissions, stock prices, or even someone's salary can also benefit from data augmentation.

Most data augmentation techniques are designed for image classification. In particular, Mixup (Zhang et al., 2018; Berthelot et al., 2019; Yun et al., 2019) is a popular data augmentation technique that is widely used for classification tasks, but is seldom used for regression because it assumes a distinct label space. The idea of Mixup is to mix pairs of examples using the key assumption that taking a linear interpolation between examples can be used to estimate the label of any examples in between. Mixup is known to effectively regularize the model. More recently, AdaMixUp (Guo et al., 2019) has been proposed to avoid manifold intrusions where the mixed examples are generated

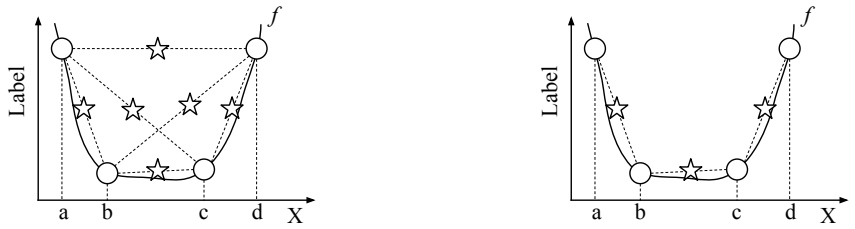


Figure 1: A comparison between the ~~original~~ Mixup (Zhang et al., 2018) ~~for classification~~ and MixRL on a small regression dataset where each example (circle) is associated with a single-dimensional feature X. (a) The ~~original~~ Mixup takes a linear interpolation of all possible example pairs. Unfortunately, the mixed examples (stars) do not properly reflect the function f where some labels are arbitrarily incorrect. (b) MixRL learns for each example how many nearest neighbors it should be mixed with. As a result, each example is only mixed with its 1- or 2-nearest neighbors to better reflect f.

too close to other existing examples, but have different labels. In addition, Manifold Mixup (Verma et al., 2019) has also been suggested to improve the hidden representation and decision boundaries where two examples are mixed at multiple hidden layers of neural networks.

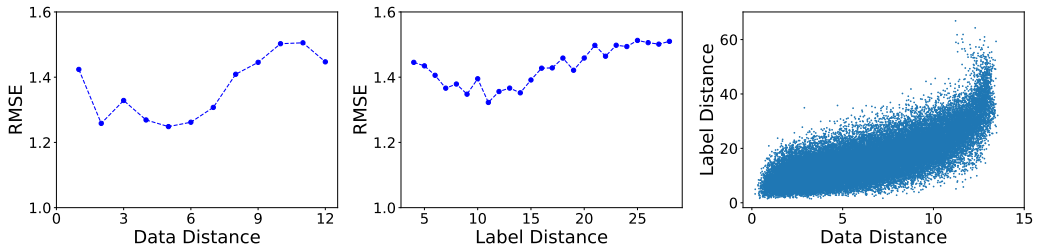
However, the Mixup techniques are not readily applicable to a regression setting because the key linearity assumption does not necessarily hold. Since the label space is continuous, taking a linear interpolation of examples that are very different either data-wise or label-wise may result in arbitrarily-incorrect labels as shown in Fig. 1a. As a result, the linearity assumption only holds to a certain extent, and the degree may vary for each example. Moreover, other data augmentation techniques for classification including image processing (e.g., flipping or rotating) and generative models (e.g., GAN (Goodfellow et al., 2014) and VAE (Kingma & Welling, 2014)) are even less applicable to a regression setting (see Sec. 5).

Our contribution is to propose a data mixing augmentation framework called MixRL, which is the first to tailor Mixup for regression tasks using reinforcement learning. MixRL uses a stricter linearity assumption where it only holds within a certain data or label distance. These distance limits may vary by example, and we formulate the problem of learning for each example how many nearest neighbors it should be mixed with. MixRL employs a meta learning framework that estimates how valuable mixing an example is for reducing the model loss on a small validation set using Monte Carlo policy gradient reinforcement learning. MixRL's framework is inspired by the recent Data Valuation using Reinforcement Learning (DVRL) framework (Yoon et al., 2020), which solves the different problem of measuring how individual examples contribute to ~~accuracy~~ model loss without any mixing involved. Fig. 1b demonstrates why it is important to limit the nearest neighbors to mix instead of mixing with all neighbors as in classification. Since MixRL is simply limiting the examples to mix, it can easily be integrated with other Mixup techniques for classification like AdaMixUp (Guo et al., 2019) or Manifold Mixup (Verma et al., 2019).

Experiments conducted on real and synthetic datasets (NO2 emissions (sta) and 3-d semiconductor manufacturing datasets (DACON Co., Ltd.)) show that MixRL shows greater model accuracy performances relative to baselines. We also comprehensively analyze MixRL and show that it only requires small validation sets, scales to large training sets, and can be integrated with AdaMixUp for possibly-better results.

2 Limited Linearity in Data and Label Space for Regression

We explain why the key linearity assumption used for Mixup in classification has limitations in a regression setting. In classification, the labels are discrete where many examples may have the same label. The original version of Mixup (Zhang et al., 2018) is to take a linear interpolation between any pair of examples x_i and x_j with the labels y_i and y_j to produce the new example $\lambda \mathrm{x}_i + (1-\lambda)\mathrm{x}_j$ with the label $\lambda \mathrm{y}_i + (1-\lambda)\mathrm{y}_j$ where $\lambda \sim Beta(\alpha, \alpha)$. The linearity assumption turns out to be reasonable because the label difference between examples is only 0 or 1 and thus not that sensitive



(a) (b) (c)

Figure 2: Using the Product dataset described in Sec. 4, (a) as the data distances between mixed examples increase, the RMSE (lower is better) of a model trained on the augmented training set initially decreases, but then gradually increases; (b) similar observations are made when increasing the label distances; and (c) the data and label Euclidean distances do not have a linear relationship.

to the data difference. In contrast, the labels in regression are in a continuous space. Although there is still a many-to-one relationship where multiple examples may have the same label, the degree is much smaller than in classification. As a result, when two examples are mixed, the interpolated label can be arbitrarily different than the actual label, e.g., mixing the points a and d in Fig. 1a results in a label nowhere near the actual label. In Sec. 4.1, we also show empirical results where the label error increases for larger data or label distances. Furthermore, mixing examples with larger data or label distances tend to have increasingly-negative effects on the model trained on the augmented training set. Figs. 2a and 2b show the model accuracies using Root Mean Square Error (RMSE) for the Product dataset (described in Sec. 4) when mixing examples with different ranges of distances. Regardless of adjusting the label or data distance, there are diminishing benefits for larger distances.

How do we limit the data and label distances to improve Mixup for regression? One approach is to only limit the data distance, which limits the label distance as well. Suppose that the regression function f is continuous where $\lim_{\mathrm{x}\to\mathrm{c}} f(\mathrm{x}) = f(\mathrm{c})$ for any x and c within the domain of f. We show that a short-enough data distance sufficiently reduces the label distance as well. Given f's domain D and $\lim_{\mathrm{x}\to\mathrm{c}} f(\mathrm{x}) = L$, the following is known to hold: $\forall\epsilon, \exists\delta$ s.t. $\forall \mathrm{x} \in D$, if $|\mathrm{x} - \mathrm{c}| < \delta$, then $|f(\mathrm{x}) - L| < \epsilon$. We can use this result to prove that $\forall\epsilon, \exists\delta$ s.t. $\forall \mathrm{x}_i, \mathrm{x}_j \in D$, if $\alpha\mathrm{x}_i + (1-\alpha)\mathrm{x}_j = \mathrm{c}$, $0 \leq \alpha \leq 1$, and $|\mathrm{x}_i - \mathrm{x}_j| < \delta$ then the absolute difference between the mixed example's x_j value and L is small where $|\alpha f(\mathrm{x}_i) + (1-\alpha) f(\mathrm{x}_j) - L| = |\alpha(f(\mathrm{x}_i) - L) + (1-\alpha)(f(\mathrm{x}_j) - L)| \leq \alpha|f(\mathrm{x}_i) - L| + (1-\alpha)|f(\mathrm{x}_j) - L| < \alpha\epsilon + (1-\alpha)\epsilon = \epsilon$. We cannot do the converse and limit the data distance by limiting label distance because there is a many-to-one mapping from data to labels, which means that two different examples may have identical labels (e.g., Fig. 1a's a and d).

However, limiting the data distance to also limit the label distance may be too restrictive because there is not much correlation between the data and label distances in real data. Fig. 2c shows how the data distance relates to the label distance for the Product dataset. Even for small data distances, the label distance has a large range, which means that the data distance would have to be extremely limited. Hence, our solution is to limit the data and/or label distance as needed instead of just the data distance. This approach turns out to be more practical as we demonstrate in Sec. 4.4.

3 MIXRL

The goal of MixRL is to identify which examples to mix with which nearest neighbors. Instead of finding the actual distance limits themselves, we solve the identical problem of finding the number of nearest neighbors to mix per example for convenience. Since we do not have training data on how mixing each example affects the model performance, we contend that reinforcement learning is the correct approach to learning which examples to mix. MixRL's framework is inspired by DVRL (Yoon et al., 2020), but we solve the different problem of mixing examples and address new issues.

3.1 POLICY OPTIMIZATION REINFORCEMENT LEARNING

Finding the optimal policy involves taking the gradient of the objective function $J(\theta) = \mathbb{E}_{\pi_\theta}[R]$ where π is the policy, θ is its parameters, and R is the reward function. For MixRL, we would like to *minimize* the regression model loss on a small validation set when mixing a batch of examples. ~~Here we cannot analytically compute the differential of the loss with respect to θ.~~ However, the validation loss is computed by the other independent regression model, not θ. Thus, the reward function is non-differentiable with respect to θ. Instead, we use the classical REINFORCE (Williams, 1992) policy gradient algorithm, which is a Monte Carlo method that is also widely used for data valuation (Yoon et al., 2020) and neural architecture searching (Zoph & Le, 2017). It is known that $\nabla_\theta J(\theta)$ can be approximated (cs2) as (see derivation in the supplementary):

$$\nabla_\theta J(\theta) \approx \frac{1}{m} \sum_{i=1}^{m} R(\tau^i) \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \tag{1}$$

where τ^i is the ith state-action trajectory under policy π_θ, m is the number of sample trajectories, T is the number of actions taken in a path, s_t^i is a state at time t, and a_t^i is an action at time t.

In our setting of minimizing loss, REINFORCE would perform gradient descent for each example decreasing θ by $\alpha \nabla_\theta J(\theta)$ where α is a learning rate. Although the estimated gradient is unbiased, it is known to have a high variance, and baseline techniques (Sutton & Barto, 2018) can be used to reduce it.

3.2 FRAMEWORK

We define notations used in MixRL's framework shown in Fig. 3. Let $\mathcal{D} = \{(\mathrm{x}_i, \mathrm{y}_i)\}_{i=1}^{S} \sim P$ be the training set where $\mathrm{x}_i \in X$ is a d-dimensional input example, and $\mathrm{y}_i \in Y$ is an e-dimensional label. Let $\mathcal{D}^v = \{(\mathrm{x}_i^v, \mathrm{y}_i^v)\}_{i=1}^{V} \sim P^t$ be the validation set, where P^t is the distribution of the test set, which is not necessarily the same as P. Let f_ϕ be a regression model, and L the loss function that returns a performance score comparing $f_\phi(\mathrm{x}_i)$ with the true label y_i using Mean Square Error (MSE). We assume a list N of possible data and label nearest neighbors (NNs) that can be mixed with an example. For instance, N could contain the options "1 data NN", "2 data NNs", and "2 label NNs". The more fine-grained the NN options are, the more precisely MixRL can determine the optimal number of NNs to mix per example. We do not add a "0 NN" option because selecting and not selecting it have identical effects, which makes the policy network training unstable because nothing can be learned. Instead, we support this option separately by excluding examples that are not worth mixing as we explain later in this section. The possible NNs can be represented as a one-hot encoding vector of $|N|$ values.

We now define the states and actions, which is an important design choice of MixRL. A state s_t is a batch of examples $\mathcal{D}^b = \{((\mathrm{x}_i, \mathrm{y}_i), \mathrm{k}_i)\}_{i=1}^{B}$ where $\{(\mathrm{x}_i, \mathrm{y}_i)\}_{i=1}^{B} \subseteq \mathcal{D}$ and each k_i is an index of an NN option in N that specifies how many data or label NNs x_i should be mixed with. An action a_t is then choosing $\mathcal{D}^m \subseteq \mathcal{D}^b$ where each $(\mathrm{x}_i, \mathrm{y}_i)$ in $\mathcal{D}^m$ is mixed with its $N[\mathrm{k}_i]$ NNs in $\mathcal{D}$. A policy $\pi_\theta(\mathcal{D}^m | \mathcal{D}^b)$ returns the probability of selecting $\mathcal{D}^m$ at state $\mathcal{D}^b$. For each episode, MixRL selects a batch (state) once and chooses a subset of the batch to obtain a reward (action) once. Since there is only one time step, the transition function does not play a role.

A naïve implementation of the policy network would have an input dimension of $B \times (d + e + |N|)$ and an output dimension of 2^B, which may be too large to train for large batch sizes. For example, in a typical setting of $B = 1000$, $d = 100$, $e = 1$, and $|N| = 10$, the input and output dimensions become $111,000$ and 2^{1000}, respectively. Instead, we can significantly reduce the network size by assuming independence among examples and decomposing the policy network's prediction into two phases. First, a Mixup value network $h_\theta(\mathrm{x}, \mathrm{y}, \mathrm{k})$ is used to estimate the probability of an example being mixed with k NNs. Next, we randomly select which examples to mix according to the probabilities. The value of $\pi_\theta(\mathcal{D}^m | \mathcal{D}^b)$ is thus $\prod_{(\mathrm{x},\mathrm{y},\mathrm{k}) \in \mathcal{D}^m} h_\theta(\mathrm{x}, \mathrm{y}, \mathrm{k}) \prod_{(\mathrm{x},\mathrm{y},\mathrm{k}) \in \mathcal{D}^b \setminus \mathcal{D}^m} [1 - h_\theta(\mathrm{x}, \mathrm{y}, \mathrm{k})]$. Also, h_θ has input and output dimensions of only $d + e + |N|$ and 1, respectively, and is practical.

The objective function $J(\theta)$ is the validation set loss l_θ of f_ϕ trained on $\mathcal{D}^b \cup Mix(\mathcal{D}^m, \mathcal{D}, N)$ (either from scratch or from a pre-trained state) where $Mix(\mathcal{D}^m, \mathcal{D}, N)$ mixes each $(\mathrm{x}_i, \mathrm{y}_i)$ in $\mathcal{D}^m$

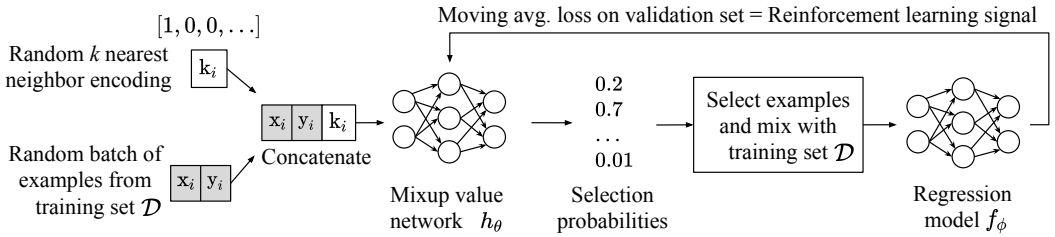


Figure 3: MixRL's framework. Given a random batch of examples from the training set, each example is combined with a one-hot encoding feature containing a random number of nearest neighbors to mix in terms of data or label distance. The Mixup value network h_θ is a neural network that returns probabilities in proportion to how helpful mixing an example with its neighbors is. The examples are then randomly selected according to the probabilities and mixed with their nearest neighbors in the training set. The random batch is augmented with the mixed examples and used to train a regression model f_ϕ. The validation loss is used to update the Mixup value network. We also use reward scaling and employ a baseline function that takes the moving average of previous losses.

Algorithm 1: Pseudo code for Mixup value network training.

Input : Training set $\mathcal{D}$, validation set $\mathcal{D}^v$, nearest neighbor options N, learning rate α, reward scaling constant C, moving average window W
Output: Mixup value network h_θ
Initialize θ, $Base = 0$;
while *until convergence* **do**
  Sample $\mathcal{D}^b = \{(\mathrm{x}_i, \mathrm{y}_i), \mathrm{k}_i\}_1^B \sim P \times Uniform(N)$;
  $\mathcal{D}^m = \emptyset$;
  for $((\mathrm{x}_i, \mathrm{y}_i), \mathrm{k}_i) \in \mathcal{D}^b$ **do**
    Add $(\mathrm{x}_i, \mathrm{y}_i, \mathrm{k}_i)$ to $\mathcal{D}^m$ with probability $h_\theta(\mathrm{x}_i, \mathrm{y}_i, \mathrm{k}_i)$;
  Train regression model f_ϕ with initialized ϕ on $\mathcal{D}^b \cup Mix(\mathcal{D}^m, \mathcal{D}, N)$;
  $Loss = \frac{1}{V}\sum_{i=1}^{V} L(f_\phi(\mathrm{x}_i^v), \mathrm{y}_i^v)$;
  $\theta = \theta - \alpha \cdot C \cdot (Loss - Base) \cdot \nabla_\theta \log(\pi_\theta(\mathcal{D}^m, \mathcal{D}^b))$;
  $Base = \frac{W-1}{W} Base + \frac{1}{W} Loss$;
return return h_θ;

with its $N[\mathrm{k}_i]$ NNs in $\mathcal{D}$:

$$J(\theta) = l(\phi) = \mathop{\mathbb{E}}_{(\mathrm{x}_i^v, \mathrm{y}_i^v) \sim P^t} L(f_\phi(\mathrm{x}_i^v), \mathrm{y}_i^v) \tag{2}$$

Using Eq. 1, the gradient of Eq. 2 can be approximated using the validation set $\mathcal{D}^v$as:

$$\nabla_\theta J(\theta) = \nabla_\theta l(\theta) \approx \frac{1}{V} \sum_{i=1}^{V} L(f_\phi(\mathrm{x}_i^v), \mathrm{y}_i^v) \cdot \nabla_\theta \log(\pi_\theta(\mathcal{D}^m, \mathcal{D}^b)) \tag{3}$$

where $\nabla_\theta \log \pi_\theta(\mathcal{D}^m, \mathcal{D}^b) = \nabla_\phi \sum_{(\mathrm{x},\mathrm{y},\mathrm{k}) \in \mathcal{D}^m} \log h_\theta(\mathrm{x}, \mathrm{y}, \mathrm{k}) + \sum_{(\mathrm{x},\mathrm{y},\mathrm{k}) \in \mathcal{D}^b \setminus \mathcal{D}^m} [1 - h_\theta(\mathrm{x}, \mathrm{y}, \mathrm{k})]$, ~~and $m = T = 1$~~. We also use reward scaling (Henderson et al., 2018) to further improve the training.

Algorithm 1 shows the pseudo code for training h_θ. The computational complexity is not directly related to the training set size, but depends on the number of iterations needed to train h_θ and how long each iteration takes. Another factor is the number of possible NNs $|N|$ where higher numbers may result in slower training. We show in Sec. 4.2 that MixRL scales to large training sets.

Once we train h_θ, we choose the $(\mathrm{x}, \mathrm{y}, \mathrm{k})$'s with the highest $h_\theta(\mathrm{x}, \mathrm{y}, \mathrm{k})$ values of at least a threshold T. We determine T where $h_\theta(\mathrm{x}, \mathrm{y}, \mathrm{k})$ values higher than T show the best validation accuracy. We ignore any $(\mathrm{x}, \mathrm{y}, \mathrm{k})$ whose (x, y) pair has already been chosen with a different k value for the same data or label distance. We apply T in order to exclude $(\mathrm{x}, \mathrm{y}, \mathrm{k})$'s that are not worth mixing (i.e., making them mix with 0 NNs). ~~We also drop any duplicate mixed examples.~~

Although actor-critic methods (Mnih et al., 2016; Schulman et al., 2017) improve on REINFORCE by training a value network in conjunction with a policy network, they may not be practical in our

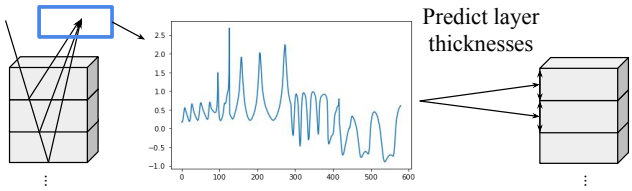


Figure 4: Spectrum generation on a 3-d semiconductor for the Product dataset.

Table 1: Settings for the three datasets.

Dataset	Data dim.	Label dim.	Training set size	Val. set size	Test set size
NO2	7	1	200	200	100
Product	580	20	300	300	234
Synthetic	226	4	500	300	300

setting ~~because the value network has a high input dimensionality of $B \times (d + e + |N|)$, and we cannot decompose its work and use a smaller network as in the policy network because its output values are not probabilities~~. For the policy network, we are able to reduce its dimensions using the fact that the log probabilities can be calculated by multiplying selection probabilities. Unfortunately, we cannot use the same trick for value networks because its output values are not probabilities.

Since MixRL only limits which examples should be mixed, it can be integrated with any existing Mixup technique for classification. In Sec. 4.2, we show how MixRL can integrate with AdaMixUp (Guo et al., 2019), ~~a state-of-the-art Mixup technique for classification.~~

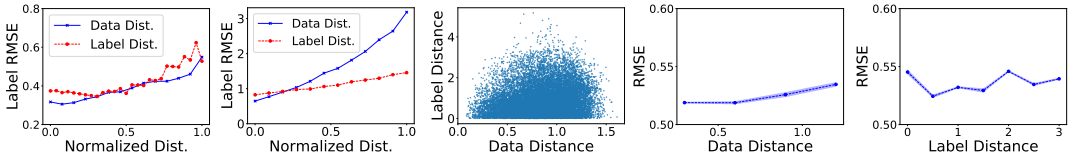
4 Experiments

We provide experimental results for MixRL. We evaluate the regression models trained on the augmented training sets on separate test sets and repeat all model trainings 5 times. We use PyTorch (Paszke et al., 2017), and all experiments are performed using Intel Xeon Silver CPUs and NVidia RTX GPUs. More experimental setting details and all of our source codes are in the supplementary.

Measures We use two accuracy metrics. RMSE = $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$ is the absolute difference between predicted labels and true labels where a lower value is better. $R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$ is the relative improvement compared to when returning the label average as a prediction. The value is in the range [0, 1] where a higher value is better. The two measures complement each other.

Datasets We use two real and one synthetic datasets. The NO2 emissions dataset (sta) contains traffic and meteorological information around roads. The features include cars per hour, wind speed, temperature, and others. The label is the NO2 concentration. The Product dataset obtained through a collaboration with a company contains spectrum data that is generated using the procedure described in Sec. 1 and illustrated in Fig. 4 on 20-layer 3-d semiconductors. The Product dataset is proprietary, so for clarity we also experiment on a public synthetic dataset (Synthetic) provided by the DACON challenge (DACON Co., Ltd.) where a simulator generates spectrum data using the same procedure as Product assuming 4-layer 3-d semiconductors. We use a subset of the full dataset as described in the supplementary. Table 1 compares the three datasets in more detail. Note that Product has much higher data and label dimensions than NO2. Together, the datasets represent various scenarios.

MixRL Settings For the regression model, we use a multi-layer perceptron (MLP) with 2 to 4 hidden layers and set the number of nodes per hidden layer to be [512, 256] for NO2, [2048, 1024, 512, 256] for Product, and [2048, 1024, 512, 256] for Synthetic. We use layer normalization (Ba et al., 2016) for Product and Synthetic and use early stopping to prevent overfitting on the validation set for regularization. We use the Adam optimizer for all MLP trainings. For Mixup value network



(a) NO2 (b) Product (c) NO2 (d) NO2 (e) NO2

Figure 5: (a) For NO2, the label RMSE increases for larger (normalized) data or label distances. (b) Similar label RMSE trends for Product. (c) For NO2, the data and label distances do not have a linear relationship. (d) For NO2, the regression model's RMSE increases for larger data distances. (e) For NO2, the model's RMSE decreases, but then increases for larger label distances.

h_θ, we use reward scaling and baseline techniques that improve the performance and stability of reinforcement learning. When mixing examples, we set the Mixup ratio λ to 0.5 unless specified otherwise. This value empirically results in the best Mixup performance in a regression setting, and we give an explanation in the supplementary. For the possible numbers of NNs for NO2, our default setting is a linear series of integers $\{i\}_{i=1}^{16} = \{1, 2, \ldots, 16\}$ for both data and label distances, i.e., $|N| = 32$. For Product, the default series is $\{10i\}_{i=1}^{27} = \{10, 20, \ldots, 270\}$ and for Synthetic, it is $\{i\}_{i=1}^{32} = \{1, 2, \ldots, 32\}$.

Baselines We employ four baselines. As the simplest approach, we directly train a regression model on the available labeled data without any data augmentation ("No Augmentation"). Next, we faithfully implement the original Mixup algorithm (Zhang et al., 2018) where all example pairs can be mixed ("Original Mixup"). We also implement AdaMixUp (Guo et al., 2019), which is a ~~state-of-the-art~~ improved Mixup technique for classification. ~~AdaMixUp penalizes mixed examples that cause manifold intrusions where their labels conflict with those of existing examples.~~ Finally, we compare with a simplified version of MixRL where we use a single k NN value to mix all examples with their data and label neighbors ("Global kNN"). This k parameter is tuned using Bayesian Optimization (Mockus, 1974).

4.1 Limited Linearity in Regression

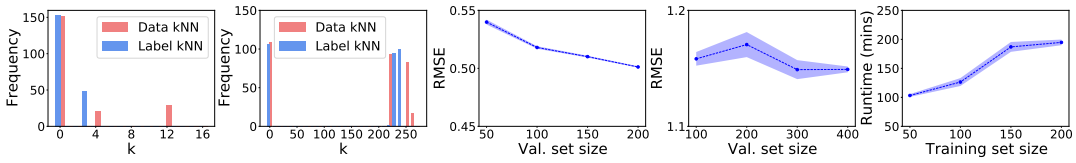
We empirically show that the linearity assumption does not hold from certain data or label distances using the following simple experiment. Given a dataset, we first train a regression model. We then generated mixed examples and compute the RMSE between their interpolated labels and the labels predicted by the regression model. While the regression model is not perfect, the relative trend of the error gives sufficient insight. Figs. 5a and 5b indeed show increasing label errors for larger label and data distances for the NO2 and Product datasets, respectively. Figs. 2c and 5c also show how label and data distances do not necessarily have a linear relationship for the two datasets. We also train regression models on top of augmented training sets where we only mix examples within certain distance ranges. As a result, mixing examples with larger data or label distances has increasingly-negative effects on the model accuracy as shown in Figs. 2a, 2b, 5d, and 5e.

4.2 Accuracy and Runtime Results

We compare MixRL with the four baselines using the settings in Table 1. Table 2 shows that MixRL significantly outperforms all the baselines in terms of regression model accuracy, which demonstrates that it is able to pinpoint which examples should be mixed with how many kNNs. Figs. 6a and 6b show which kNN options are frequently used for NO2 and Product (the kNN options for Synthetic are in the supplementary). Although the runtime is slower than the baselines due to the reinforcement learning, the gained accuracy can be worth the effort in real applications because data augmentation is a one-time cost that can run in batch mode. Another observation is that Original Mixup performs relatively better for Product than other datasets where mixing examples is more often beneficial. This observation shows that MixRL performs better than baselines the shorter the ground truth distances. The ground truth distances are data or label distances in which the limited linearity holds. For example, the NO2 dataset tends to have shorter ground truth distance (k=2), so MixRL clearly outperforms Original Mixup, which mixes all neighbors. On the other hand, if the

Table 2: Accuracy and runtime performances on the three datasets. Four baselines are compared with MixRL: (1) No data augmentation; (2) Original Mixup (Zhang et al., 2018); (3) AdaMixUp (Guo et al., 2019); and (4) Global kNN. For each global kNN, we show the best k value found with Bayesian optimization. Another version of MixRL integrated with AdaMixUp is compared as well.

Dataset	Method	RMSE	R^2	Runtime (mins)
NO2	No Augmentation	$0.5165_{\pm 0.0008}$	$0.4970_{\pm 0.0011}$	0.03
	Original Mixup	$0.5320_{\pm 0.0035}$	$0.4766_{\pm 0.0035}$	0.04
	AdaMixUp	$0.5266_{\pm 0.0031}$	$0.4849_{\pm 0.0034}$	5
	Global kNN (k=2)	$0.5171_{\pm 0.0015}$	$0.5042_{\pm 0.0024}$	5
	MixRL	$\mathbf{0.5012}_{\pm 0.0006}$	$\mathbf{0.5268}_{\pm 0.0007}$	185
	MixRL + AdaMixUp	$0.5014_{\pm 0.0007}$	$0.5256_{\pm 0.0012}$	285
Product	No Augmentation	$1.4528_{\pm 0.0059}$	$0.6700_{\pm 0.0025}$	0.35
	Original Mixup	$1.1664_{\pm 0.0063}$	$0.7830_{\pm 0.0015}$	2.5
	AdaMixUp	$1.1824_{\pm 0.0044}$	$0.7762_{\pm 0.0018}$	5
	Global kNN (k=151)	$1.1536_{\pm 0.0076}$	$0.7863_{\pm 0.0026}$	160
	MixRL	$\mathbf{1.1489}_{\pm 0.0082}$	$\mathbf{0.7898}_{\pm 0.0023}$	830
	MixRL + AdaMixUp	$1.3194_{\pm 0.0073}$	$0.7275_{\pm 0.0023}$	900
Synthetic	No Augmentation	$15.5463_{\pm 0.1873}$	$0.7153_{\pm 0.0071}$	0.1
	Original Mixup	$15.0700_{\pm 0.0624}$	$0.7315_{\pm 0.0026}$	1
	AdaMixUp	$14.8497_{\pm 0.0840}$	$0.7373_{\pm 0.0030}$	5
	Global kNN (k=36)	$13.7732_{\pm 0.0989}$	$0.7735_{\pm 0.0020}$	50
	MixRL	$13.2519_{\pm 0.1211}$	$0.7898_{\pm 0.0034}$	780
	MixRL + AdaMixUp	$\mathbf{13.1288}_{\pm 0.0584}$	$\mathbf{0.7930}_{\pm 0.0020}$	1240



(a) NO2 (b) Product (c) NO2 (d) Product (e) NO2

Figure 6: (a)-(b) kNN option frequency histograms for the real datasets. (c)-(d) Model accuracy vs. validation set size for the real datasets. (e) Runtime of MixRL against the training set size for NO2.

ground truth distances are effectively infinity, MixRL would simply mix all or most neighbors, and its performance falls back to that of Original Mixup, which is not a bad result at all. In our experiments, the Product datasets tend to have large ground truth distances (k=151), so naturally MixRL's performance is closer to that of Original Mixup. We also integrate MixRL with AdaMixUp where examples are mixed with their NNs while also avoiding manifold intrusions (more details are in the supplementary). The results are mixed where there is an improvement compared to MixRL for the Synthetic dataset, but not for the other datasets.

Validation Set and Scalability Figs. 6c and 6d show the validation set size impact on MixRL's accuracy for the real datasets. The results suggest that a small validation set is sufficient for MixRL to be effective. Also MixRL scales well against increasing training set sizes as shown in Fig. 6e.

4.3 K Nearest Neighbor Options

We investigate how changing the kNN options N affects MixRL's performance in Table 3 using the two real datasets. The Synthetic dataset results are similar to those of Product and shown in the supplementary. For each dataset, we compare three linear series of different k ranges (first three rows). As the range increases, MixRL's accuracy improves for both datasets, but only to a certain extent for NO2. We mark the default series used throughout the other experiments. We also compare

Table 3: Comparing kNN options for MixRL on the two real datasets. We mark the default options.

Dataset	Data and Label kNN Options	RMSE	R^2
NO2	$\{i\}_{i=1}^{4}$	$0.5082_{\pm 0.0007}$	$0.5161_{\pm 0.0010}$
	$\{i\}_{i=1}^{16}$ (Default)	$\mathbf{0.5012}_{\pm 0.0006}$	$\mathbf{0.5268}_{\pm 0.0007}$
	$\{i\}_{i=1}^{32}$	$0.5014_{\pm 0.0032}$	$0.5253_{\pm 0.0049}$
	$\{2^i\}_{i=0}^{4}$	$0.5048_{\pm 0.0029}$	$0.5244_{\pm 0.0016}$
Product	$\{10i\}_{i=1}^{9}$	$1.2195_{\pm 0.0101}$	$0.7670_{\pm 0.0024}$
	$\{10i\}_{i=1}^{18}$	$1.2115_{\pm 0.0090}$	$0.7699_{\pm 0.0029}$
	$\{10i\}_{i=1}^{27}$ (Default)	$\mathbf{1.1489}_{\pm 0.0082}$	$\mathbf{0.7898}_{\pm 0.0023}$
	$\{2^i\}_{i=0}^{8}$	$1.1523_{\pm 0.0071}$	$0.7878_{\pm 0.0017}$

Table 4: Ablation study for MixRL on the two real datasets. Three variants are compared: (1) w/o label and data distance limits; (2) w/o label distance limits; and (3) w/o data distance limits.

Dataset	Method	RMSE	R^2
NO2	W/O Label & Data Dist. Limits	$0.5221_{\pm 0.0039}$	$0.4931_{\pm 0.0047}$
	W/O Label Dist. Limits	$0.5117_{\pm 0.0014}$	$0.5067_{\pm 0.0023}$
	W/O Data Dist. Limits	$0.5163_{\pm 0.0013}$	$0.5066_{\pm 0.0015}$
	MixRL	$\mathbf{0.5012}_{\pm 0.0006}$	$\mathbf{0.5268}_{\pm 0.0007}$
Product	W/O Label & Data Dist. Limits	TLE	TLE
	W/O Label Dist. Limits	$1.1617_{\pm 0.0037}$	$0.7813_{\pm 0.0008}$
	W/O Data Dist. Limits	$1.1793_{\pm 0.0091}$	$0.7765_{\pm 0.0036}$
	MixRL	$\mathbf{1.1489}_{\pm 0.0082}$	$\mathbf{0.7898}_{\pm 0.0023}$

the default series with an exponential series that has a similar k range (fourth row). As a result, the exponential series has a worse accuracy because MixRL has fewer k values to choose from.

4.4 Ablation Study

We perform an ablation study to investigate the effect of limiting label and data distances in Table 4 using the two real datasets. The Synthetic dataset results are similar to those of Product and shown in the supplementary. We compare MixRL with three variants: (1) MixRL that does not exploit any distances and selects examples that have the highest h_θ values when fully mixed with all the other examples; (2) MixRL that does not use label distances; and (3) MixRL that does not use data distances. If any method takes longer than a day, we report a Time Limit Exceeded (TLE). We observe that using both distances is necessary for the best accuracy results as we suggested in Sec. 2.

5 Related Work

Data Augmentation for Regression There are largely two branches of work for data augmentation in regression. One is semi-supervised regression (Kostopoulos et al., 2018; Zhou & Li, 2005; Kang et al., 2016; Kim et al., 2020) where the goal is to utilize unlabeled data for training. In comparison, MixRL does not assume the availability of unlabeled data. Another branch is data augmentation when there is no unlabeled data, which is our research focus. To the best of our knowledge, there is no technique tailored to regression, so we instead cover data augmentation techniques for classification that can be extended to regression.

We explore data augmentation techniques for classification and investigate if they can be used in a regression setting. There are largely three approaches: generative models, policies, and Mixup techniques. Generative models including GANs (Goodfellow et al., 2014) and VAEs (Kingma & Welling, 2014) are popular in classification where the idea is to generate realistic data that cannot be distinguished from the real data by a discriminator. However, a major assumption is that the labels of the generated examples are the same, which does not necessarily hold in a regression setting where most examples may have different labels. Another approach is to use policies (Cubuk et al., 2019), which specify fixing rules for transforming the data while maintaining the label value. For example,

image processing policies may flip, rotate, or adjust the brightness of images to generate new valid images. However, in a regression setting, the same transformed examples may now have different unknown labels, making them unsuitable for training. The recently-proposed Mixup (Zhang et al., 2018; Berthelot et al., 2019; Yun et al., 2019; Berthelot et al., 2020; Hendrycks et al., 2020; Verma et al., 2019) takes the alternative approach of generating both data and labels together by mixing existing examples with different labels assuming linearity between training examples (Chapelle et al., 2000; Wu et al., 2020). ~~More recently, AdaMixUp improves Mixup by avoiding manifold intrusions and supporting k-fold mixing among examples.~~ Although the Mixup paper mentions that its techniques can be extended to regression in a straightforward fashion, our key observation is that the linearity assumption does not necessarily hold for large data or label distances, which makes the extension non-trivial. We believe MixRL provides the first solution for adaptively selecting promising examples to mix for the purpose of improving regression accuracy.

Policy Optimization Reinforcement Learning Within reinforcement learning (RL) (Sutton & Barto, 2018; Kaelbling et al., 1996), we are interested in policy searching methods (Neumann & Peters, 2015) where our goal is to find the best policy for mixing examples. In particular, MixRL's framework is inspired by the recent Data Valuation using Reinforcement Learning (DVRL) framework (Yoon et al., 2020), which uses REINFORCE (Williams, 1992) to estimate the value of data in terms of how much it ~~improves the model performance~~ affects the model loss. In comparison, we solve the completely different problem of data augmentation for regression where we extend Mixup using REINFORCE. Some data augmentation techniques use actor-critic methods like A3C (Mnih et al., 2016) and PPO (Schulman et al., 2017) that train value networks along with policy networks. For example, AutoAugment (Cubuk et al., 2019) uses PPO to search for data augmentation policies that dictate how to modify existing examples to generate additional training data for image classification. However, using actor-critic methods is practically challenging in our setting due to the large value networks that need to be trained as we discussed in Sec. 3.2.

6 Conclusion

We proposed MixRL, which is to our knowledge the first Mixup-based data augmentation meta learning framework tailored to regression tasks. We observe that linearity only holds up to a certain data or label distance per example in a regression setting. Hence, MixRL extends Mixup by determining the k nearest neighbors to mix per example using Monte Carlo policy gradient reinforcement learning where the objective is to minimize the regression model's loss on a small validation set. Since MixRL only limits the examples to mix, it can be integrated with any existing Mixup technique for classification. In our experiments, we showed that MixRL outperforms various data augmentation baselines for regression by effectively selecting the k nearest neighbor options.

Societal Impacts and Limitations MixRL demonstrates that data augmentation can significantly enhance the model accuracy for regression tasks. However, if not controlled, data augmentation may also introduce new bias in the data that makes the model more discriminative. Thus an interesting future work is to augment data while ensuring model fairness. MixRL also has limitations where it can only mix existing examples instead of creating new ones and does require hours of runtime.

Acknowledgement

Sung Yoon Ryu, Soo Seok Lee, and Gwang Nae Gil of Samsung Electronics provided the Product dataset and key insights on the 3-d semiconductor application. This work is supported by the Samsung Electronics' University R&D program [Algorithm advancement and reliability-enhancing technique development for spectrum data-based deep learning] and a Kwon Oh-Hyun Assistant Professorship.

References

Policy gradient lecture notes. `http://web.stanford.edu/class/cs234/slides/lecture8.pdf`. Accessed May 28th, 2021.

Cmu statlib dataset. http://lib.stat.cmu.edu/datasets/. Accessed May 28th, 2021.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016.

David Berthelot, Nicholas Carlini, Ian J. Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *NeurIPS*, pp. 5050–5060, 2019.

David Berthelot, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. Remixmatch: Semi-supervised learning with distribution matching and augmentation anchoring. In *ICLR*, 2020.

Olivier Chapelle, Jason Weston, Léon Bottou, and Vladimir Vapnik. Vicinal risk minimization. In *NIPS*, pp. 416–422, 2000.

Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation strategies from data. In *CVPR*, pp. 113–123, 2019.

DACON Co., Ltd. Challenge on semiconductor thin film thickness analysis. https://dacon.io/competitions/official/235554/data/. Accessed May 28th, 2021.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, pp. 2672–2680, 2014.

Hongyu Guo, Yongyi Mao, and Richong Zhang. Mixup as locally linear out-of-manifold regularization. In *AAAI*, pp. 3714–3722, 2019.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *AAAI*, pp. 3207–3214, 2018.

Dan Hendrycks, Norman Mu, Ekin Dogus Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. In *ICLR*, 2020.

Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *J. Artif. Int. Res.*, 4(1):237–285, May 1996.

Pilsung Kang, Dongil Kim, and Sungzoon Cho. Semi-supervised support vector regression based on self-training with label uncertainty: An application to virtual metrology in semiconductor manufacturing. *Expert Syst. Appl.*, 51:85–106, 2016.

Sung Wook Kim, Young Gon Lee, Bayu Adhi Tama, and Seungchul Lee. Reliability-enhanced camera lens module classification using semi-supervised regression method. *Applied Sciences*, 10(11), 2020.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

Georgios Kostopoulos, Stamatis Karlos, Sotiris Kotsiantis, and Omiros Ragos. Semi-supervised regression: A recent review. *J. Intell. Fuzzy Syst.*, 35(2):1483–1500, 2018.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, pp. 1928–1937, 2016.

Jonas Mockus. On bayesian methods for seeking the extremum. In *Proceedings of the IFIP Technical Conference*, pp. 400–404, Berlin, Heidelberg, 1974. Springer-Verlag. ISBN 3540071652.

Gerhard Neumann and Jan Peters. Policy search: Methods and applications. In *ICML*, 2015.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, 2018. ISBN 0262039249.

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *ICML*, 2019.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8:229–256, 1992.

Sen Wu, Hongyang R. Zhang, Gregory Valiant, and Christopher Ré. On the generalization effects of linear transformations in data augmentation. In *ICML*, pp. 10410–10420, 2020.

Jinsung Yoon, Sercan Ömer Arik, and Tomas Pfister. Data valuation using reinforcement learning. In *ICML*, pp. 10842–10851, 2020.

Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Seong Joon Oh, Youngjoon Yoo, and Junsuk Choe. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, pp. 6022–6031, 2019.

Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018.

Zhi-Hua Zhou and Ming Li. Semi-supervised regression with co-training. In *IJCAI*, pp. 908–913, 2005.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017.

A Appendix – Framework

A.1 Gradient of $J(\theta)$

We continue from Sec. 3.1 and derive Eq. 1 (cs2):

$$\nabla_\theta J(\theta) \approx \frac{1}{m} \sum_{i=1}^{m} R(\tau^i) \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \tag{1}$$

where τ_i is a state-action trajectory under policy π_θ, m is the number of sample trajectories, T is the number of actions taken in a path, s_t is a state at time t, and a_t is an action at time t. Suppose that $P(\tau;\theta)$ is the probability of a trajectory τ when using the policy π_θ, and $R(\tau)$ is the reward of τ. Then $J(\theta)$ can be expressed as$\sum_\tau P(\tau;\theta)R(\tau)$. In addition, let us specify s_0 as the initial state, $\mu(s_0)$ as the initial state's distribution, and $P(s_{t+1}|s_t, a_t)$ as the dynamics model that specifies which predicts the next state s_{t+1} when taking action a_t from state s_t. We can then take the gradient of $J(\theta)$ as follows:

$$
\begin{aligned}
\nabla_\theta J(\theta) =& \nabla_\theta \sum_\tau P(\tau;\theta)R(\tau) \\
=& \sum_\tau \nabla_\theta P(\tau;\theta)R(\tau) \\
=& \sum_\tau \frac{P(\tau;\theta)}{P(\tau;\theta)} \nabla_\theta P(\tau;\theta)R(\tau) \\
=& \sum_\tau P(\tau;\theta)R(\tau)\frac{\nabla_\theta P(\tau;\theta)}{P(\tau,\theta)} \\
=& \sum_\tau P(\tau;\theta)R(\tau)\nabla_\theta \log P(\tau;\theta) \\
\approx& \frac{1}{m}\sum_{i=1}^{m} R(\tau^i)\nabla_\theta \log P(\tau^i;\theta) \\
=& \frac{1}{m}\sum_{i=1}^{m} R(\tau^i)\nabla_\theta \log \left[\mu(s_0)\prod_{t=0}^{T-1}\pi_\theta(a_t^i|s_t^i)P(s_{t+1}^i|s_t^i,a_t^i)\right] \\
=& \frac{1}{m}\sum_{i=1}^{m} R(\tau^i)\nabla_\theta \left[\log \mu(s_0^i)\sum_{t=0}^{T-1}\log \pi_\theta(a_t^i|s_t^i) + \log P(s_{t+1}|s_t^i,a_t^i)\right] \\
=& \frac{1}{m}\sum_{i=1}^{m} R(\tau^i)\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)
\end{aligned}
$$

B Appendix – Experiments

B.1 More Experimental Settings

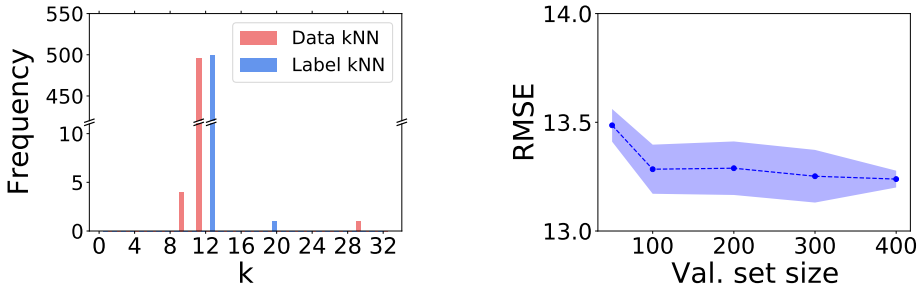
We continue describing our experimental settings from Sec. 4. For the mixup value network h_θ, we use a multi-layer perceptron with 4 hidden layers and set the number of nodes per hidden layer to be [256, 256, 100, 64] for the three datasets. We use a learning rate of $\alpha = 0.001$ for h_θ and a learning rate of 0.0001 for the regression model f_ϕ for the three datasets. When training h_θ, we set the batch size B to be the training data's size. When training f_ϕ, we use a batch size of 32 for NO2, 64 for Product, and 128 for Synthetic. We use a reward scaling (Henderson et al., 2018) constant of $C = 2$ for NO2, $C = 0.5$ for Product, and $C = 0.001$ for Synthetic. We use a moving average window of size $W = 20$ for the three datasets. When setting the threshold T for limiting examples to mix, we choose a value that results in a high model accuracy (low RMSE) on the validation set.

When constructing the Synthetic dataset, we take a subset of the entire DACON challenge dataset (DACON Co., Ltd.). While the the range of thicknesses for the full datasets is [10,300] for all four layers, we only use examples with thicknesses within the range [10,100]. The purpose is to reduce the training time while still making a clear comparison between MixRL and the other methods.

B.2 More Synthetic Dataset Experiments

kNN Options Frequency Histogram We continue from Sec. 4.2 and show which kNN options are frequently used in our default settings for the Synthetic dataset in Figure 7a. Compared to the Product dataset, there are no cases where k = 0 because mixing is more beneficial to the model's accuracy according to the validation set.

Validation Set Size and Accuracy We continue from Sec. 4.2 and show how the validation set size impacts MixRL's accuracy on the Synthetic dataset in Figure 7b. Just like for the other datasets, we observe that a small validation set size is sufficient for MixRL to perform well.



(a) (b)

Figure 7: (a) kNN option frequency histogram for the Synthetic dataset. (b) Model accuracy versus validation set size for the Synthetic dataset.

kNN Options Comparison We continue from Sec. 4.3 and provide the kNN options experiments for the Synthetic dataset experiments in Table 5. The observations are similar to those of the Product dataset in Table 3.

Table 5: Comparing kNN options for MixRL on the Synthetic dataset.

Dataset	Data and Label kNN Options	RMSE	R^2
Synthetic	$\{i\}_{i=1}^{8}$	$13.6199_{\pm 0.0412}$	$0.7766_{\pm 0.0009}$
	$\{i\}_{i=1}^{16}$	$13.3081_{\pm 0.1116}$	$0.7867_{\pm 0.0035}$
	$\{i\}_{i=1}^{32}$ (Default)	$\mathbf{13.2519}_{\pm 0.1211}$	$\mathbf{0.7898}_{\pm 0.0034}$
	$\{2^i\}_{i=0}^{5}$	$13.4783_{\pm 0.0981}$	$0.7825_{\pm 0.0033}$

Ablation Study We continue from Sec. 4.4 and provide the ablation study for the Synthetic dataset in Table 6. The results are similar to those of the Product dataset in Table 4.

Table 6: Ablation study for MixRL on the Synthetic dataset. Three variants are compared: (1) w/o label and data distance limits; (2) w/o label distance limits; and (3) w/o data distance limits.

Dataset	Method	RMSE	R^2
Synthetic	W/O Label & Data Dist. Limits	TLE	TLE
	W/O Label Dist. Limits	$13.5346_{\pm 0.0287}$	$0.7807_{\pm 0.0013}$
	W/O Data Dist. Limits	$13.7705_{\pm 0.0951}$	$0.7725_{\pm 0.0031}$
	MixRL	$\mathbf{13.2519}_{\pm 0.1211}$	$\mathbf{0.7898}_{\pm 0.0034}$

B.3 Justification for Using $\lambda = 0.5$

We continue from Sec. 4 and provide more details on why setting $\lambda = 0.5$ empirically results in the best Mixup performance in a regression setting. Figure 8 shows MixRL's accuracy when we vary λ using the same setting in Table 1 for the NO2 and Product datasets. As a result, the accuracy peaks when $\lambda = 0.5$. We suspect that generating examples in the sparsest areas helps the regularization the most. In a classification setting, however, this strategy may not be ideal especially if the generated examples are too close to a decision boundary that is in the middle of the mixed examples.

B.4 Integrating MixRL with AdaMixUp

We continue from Sec. 4.2 and provide more details on integrating MixRL with AdaMixUp (Guo et al., 2019). The key feature of AdaMixUp is learning a λ value for each example pair in order to avoid manifold intrusions. We use multi-task learning for the regression model and a classification model that predicts whether a mixed example is within the data manifold or not. We first run the AdaMixUp algorithm (2-fold version) and then run MixRL using learned λ values by AdaMixUp.

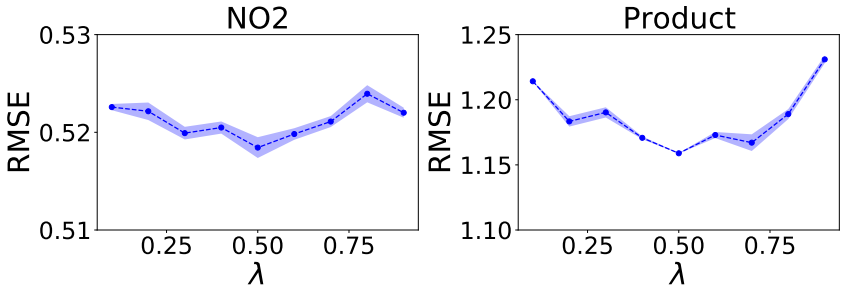


Figure 8: MixRL's accuracy for different λ values on the NO2 and Product datasets.