mixReg: A Simple Way to Improve Generalization in Regression for Deep Neural Networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

Improving the generalization of deep networks is an important open challenge, particularly in domains without plentiful data. The mixup algorithm improves generalization by linearly interpolating the input features of a pair of examples and their corresponding labels. These interpolated examples augment the original training dataset. It has shown promising results in various classification tasks, but systematic analysis of mixup in regression remains underexplored. Using mixup directly on regression labels could result in arbitrarily wrong labels since the linearity assumption behind mixup may not hold. In this paper, we propose a simple yet powerful algorithm, mixReg, to improve generalization on regression tasks. In contrast with vanilla mixup, which uniformly samples mixing pairs from the training set, mixReg adjusts the sampling probability based on the similarity of labels. Our theoretical analysis further confirms that mixReg with label similarity obtains a smaller mean square error than vanilla mixup and using feature similarity. Another benefit of mixReg is that it can improve out-of-distribution robustness, where the test distribution is different from the training distribution. By selectively interpolating examples with similar labels, it mitigates the effects of domain-associated information and pushes invariant representations. We evaluate mixReg on eleven datasets, ranging from tabular to video data. Compared to other methods, mixReg achieves 6.56%, 4.76%, 5.14% improvements in in-distribution generalization, task generalization, and out-of-distribution robustness, respectively.

1 Introduction

Training deep neural networks faces the challenge of mitigating overfitting and improving generalization. In order to accomplish this, a number of techniques, including data augmentation [2, 7, 9, 67, 68] and explicit regularization [11, 32, 49] have been proposed. Representatively, mixup [68] densifies the data distribution and regularizes the model by linearly interpolating the features between randomly sampled pairs of original examples and applying the same interpolation on the corresponding labels. Despite mixup having demonstrated promising results in improving generalization in classification problems, it has rarely been studied in the context of regression with continuous responses, on which we focus in this paper.

In contrast to classification, the goal of regression is to predict a continuous response from each input. Directly applying mixup to input features and labels to regression tasks may yield arbitrarily incorrect labels. For example, as shown in Figure 1(a), ShapeNet1D pose prediction [14] aims to predict the current orientation of the object relative to its canonical orientation. We randomly select three mixing pairs and show the mixed images and labels in Figure 1(b), where only pair 1 exhibits reasonable mixing results. Thus, sampling mixing pairs uniformly from the dataset introduces a number of noisy pairs. Alternatively, Manifold Mixup [56] proposes to linearly interpolate hidden representations rather than input features. Although it is possible to improve the original mixup in regression to some

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

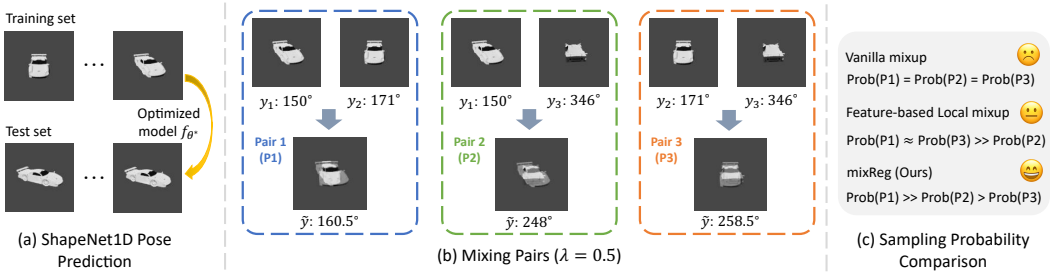


Figure 1: Illustration of mixReg on ShapeNet1D pose prediction. λ represents the interpolation ratio. (a) ShapeNet1D pose prediction task, aiming to predict the current orientation of the object relative to its canonical orientation. (b) Three mixing pairs are randomly picked, where the interpolated images are visualized and $\tilde{y}$ represents interpolated labels. (c) Illustration of a rough comparison of sampling probabilities among three mixing pairs in (b). The Euclidean distance measures input feature distance and the corresponding results between examples in pairs 1, 2, 3 are 1.51×10^5, 1.82×10^5, 1.50×10^5, respectively. Hence, pairs 1 and 3 have similar results, leading to similar sampling probabilities. mixReg is able to assign higher sampling probability for more reasonable mixing pairs.

extent, empirical evidence suggests that it cannot achieve consistent improvements over a wide range of scenarios [23]. Instead of applying mixup or Manifold Mixup to all examples, MixRL [23] selects the mixing pairs in regression by using a well-trained policy network. The training of a complex policy network, however, requires considerable computing resources, which are not suitable for high-dimensional real-world data.

Instead of training an external policy network, we aim to directly adjust the sampling probability of mixing pairs according to the similarity of examples, resulting in a simple training technique named as **mixReg** (**mix**up for **Reg**ression). Specifically, we employ a Gaussian kernel to calculate the sampling probability of drawing another example for mixing, where closer examples are more likely to be sampled. Here, the core question is: *how to measure the similarity between two examples?* The most straightforward solution is to compute input feature similarity. Yet, using input similarity has two major downsides when dealing with high-dimensional data such as images or time-series: substantial computational costs and lack of good distance metrics. Specifically, first, it takes considerable time to compute pairwise similarities across all samples. Second, directly applying classical distance metrics (e.g., Euclidean distance, cosine distance) does not reflect the high-level relation between input features. In the ShapeNet1D rotation prediction example (Figure 1(a)), pair 1 and 3 have close input similarities, which is not expected since only pair 1 shows the reasonable scenario after mixing.

To overcome these drawbacks, mixReg instead uses label similarity, which is easy to compute and reduces the computation cost since the label space is usually low dimensional. In addition to the obvious computational advantages, mixReg benefits three kinds of regression problems. *First*, it empirically improves in-distribution generalization in supervised learning compared with the vanilla mixup and using input feature similarity. *Second*, we extend mixReg to gradient-based meta-learning and particularly incorporated it with MetaMix, a mixup-based task augmentation method [61]. Compared with vanilla MetaMix, mixReg further improves task generalization; *Third*, mixReg is well-suited for improving out-of-distribution robustness, particularly to covariate shift (see the corresponding example of ShapeNet1D rotation prediction in Appendix A.1). Concretely, we aim to learn a regression model that is robust to the covariate shift between training and test distributions. By performing mixup on examples with close continuous labels, mixReg encourages the regression model to rely on the invariant features to make prediction and ignore unrelated or spurious correlations, making the model more robust to covariate shift.

The primary contribution of this paper is: (1) We systematically investigate how to adapt mixup in regression, leading to a simple and scalable algorithm – mixReg; (2) We theoretically analyze the advantages of mixReg. Compared with vanilla mixup and using feature similarity, mixReg has the lower mean squared error and improves in-distribution generalization, task generalization and robustness to covariate shift; (3) Empirically, we conduct comprehensive experiments to evaluate mixReg on eleven datasets, including many large-scale real-world applications like drug-target interaction prediction [22], ejection fraction estimation with annotated echocardiogram videos [41], poverty estimation with satellite imagery [64]. Compared to the best baseline, the results demonstrate the promise of mixReg with 6.56%, 4.76%, 5.14% improvements in in-distribution generalization, task generalization, and out-of-distribution robustness, respectively.

2 Preliminaries

In this section, we define notation and describe the background of ERM and mixup in the supervised learning setting, and MetaMix in the meta-learning setting for task generalization.

ERM. Assume a machine learning model f with parameter space Θ. In this paper, we consider the setting where one predicts the continuous label $y \in \mathcal{Y}$ according to the input feature $x \in \mathcal{X}$. Given a loss function ℓ, we train a model f_θ under the empirical training distribution P^{tr} with the following objective, and get the optimized parameter $\theta^* \in \Theta$:

$$\theta^* \leftarrow \arg\min_{\theta \in \Theta} \mathbb{E}_{(x,y)\sim P^{tr}}[\ell(f_\theta(x), y)]. \tag{1}$$

Typically, we expect the model to perform well on unseen examples drawn from the test distribution P^{ts}. We are interested in both *in-distribution* ($P^{tr} = P^{ts}$) and *out-of-distribution* ($P^{tr} \neq P^{ts}$) settings.

Mixup. The mixup algorithm samples a pair of instances (x_i, y_i) and (x_j, y_j), sampled uniformly at random from the training dataset, and generates new examples by performing linear interpolation on the input features and corresponding labels as:

$$\tilde{x} = \lambda \cdot x_i + (1-\lambda) \cdot x_j, \ \tilde{y} = \lambda \cdot y_i + (1-\lambda) \cdot y_j, \tag{2}$$

where the interpolation ratio $\lambda \in [0, 1]$ is drawn from a Beta distribution, i.e., $\lambda \sim \mathrm{Beta}(\alpha, \alpha)$. The interpolated examples are then used to optimize the model as follows:

$$\theta^* \leftarrow \arg\min_{\theta \in \Theta} \mathbb{E}_{(x_i,y_i),(x_j,y_j)\sim P^{tr}}[\ell(f_\theta(\tilde{x}), \tilde{y})]. \tag{3}$$

Task Generalization and MetaMix. In this paper, we also investigate few-shot *task generalization* under the gradient-based meta-regression setting. Given a task distribution $p(\mathcal{T})$, we assume each task $\mathcal{T}_m$ is sampled from $p(\mathcal{T})$ and is associated with a dataset $\mathcal{D}_m$. A support set $\mathcal{D}^s_m = \{(X^s_m, Y^s_m)\} = \{(x^s_{m,i}, y^s_{m,i})\}_{i=1}^{N^s}$ and a query set $\mathcal{D}^q_m = \{(X^s_m, Y^s_m)\} = \{(x^q_{m,j}, y^q_{m,j})\}_{i=1}^{N^q}$ are sampled from $\mathcal{D}_m$. Representatively, in model-agnostic meta-learning (MAML) [10], given a predictive model f with parameter θ, it aims to learn an initialization θ^* from meta-training tasks $\{\mathcal{T}_m\}_{m=1}^{|M|}$. Specifically, at the meta-training phase, MAML obtained the task-specific parameter ϕ_m for each task $\mathcal{T}_m$ by performing a few gradient steps starting from θ. Then, the corresponding query set $\mathcal{D}^q_m$ is used to evaluate the performance of the task-specific model and optimize the model initialization as:

$$\theta^* := \arg\min_\theta \frac{1}{|M|} \sum_{i=1}^{|M|} \mathcal{L}(f_{\phi_m}; \mathcal{D}^q_m), \ where \ \ \phi_m = \theta - \alpha \nabla_\theta \mathcal{L}(f_\theta; \mathcal{D}^s_m) \tag{4}$$

At the meta-testing phase, for each meta-testing task $\mathcal{T}_t$, MAML fine-tunes the learned initialization θ^* on the support set $\mathcal{D}^s_t$ and evaluates the performance on the corresponding query set $\mathcal{D}^q_t$.

To improve task generalization, MetaMix [61] adapts mixup (Eqn. (3)) to meta-learning, which linearly interpolates the support set and query set and uses the interpolated set to replace the original query set $\mathcal{D}^q_m$ in Eqn. (4). Specifically, the interpolated query set is formulated as:

$$\tilde{X}^q_m = \lambda X^s_m + (1-\lambda) X^q_m, \ \tilde{Y}^q_m = \lambda Y^s_m + (1-\lambda) Y^q_m, \tag{5}$$

where $\lambda \sim \mathrm{Beta}(\alpha, \alpha)$.

3 Mixup for Regression (mixReg)

This section first introduces the main idea of mixReg. Then, we theoretically explain how mixReg benefits in-distribution generalization, task generalization, and out-of-distribution robustness.

3.1 mixReg

For continuous labels, the example in Figure 1(b) illustrates that applying vanilla mixup to the entire distribution is likely to produce arbitrary labels. To resolve this issue, mixReg proposes to sample closer pairs of examples with higher probability. Specifically, given an example (x_i, y_i), mixReg introduces a symmetric Gaussian kernel in order to calculate the sampling probability $P((x_j, y_j)|(x_i, y_i))$ for another (x_j, y_j) example to be mixed as follows:

$$P((x_j, y_j)|(x_i, y_i)) \propto \exp\left(-\frac{d(i,j)}{2\sigma^2}\right) \tag{6}$$

where $d(i, j)$ represents the distance between the examples (x_i, y_i) and (x_j, y_j), and σ describes the bandwidth. In the case of example (x_i, y_i), the set $\{P((x_j, y_j)|(x_i, y_i))|\forall j\}$ is then normalized to a probability mass function that sum up to one.

The most straightforward way to compute the distance is using the input feature x, i.e., $d(i,j) = d(x_i, x_j)$. However, when dealing with the high-dimensional data such as images or videos, we lack good distance metrics to capture structured feature information and the distances can be easily influenced by feature noise. Additionally, computing feature distances for high-dimensional data is time-consuming. Instead, mixReg leverages the labels with $d(i,j) = d(y_i, y_j) = \|y_i - y_j\|_2^2$, where y_i and y_j are vectors with continuous values. The dimension of label is typically much smaller than that of the input feature, therefore reducing computational costs (see more discussions about compuatational efficiency in Appendix A.3). The overall algorithm of mixReg is described in Alg. 1 and we detail the difference between mixReg and mixup in Appendix A.4. In addition to its computational benefits, mixReg improves generalization on three distinct kinds of regression problems, which is theoretically shown in the following sections.

Algorithm 1 Training with mixReg

Require: Learning rates η; Shape parameter α
Require: Training data $\mathcal{D} := \{(x_i, y_i)\}_{i=1}^N$
Randomly initialize model parameters θ
Calculate pairwise distance matrix P via Eqn. (6).
while not converge **do**
  Sample a batch of examples $\mathcal{B} \sim \mathcal{D}$
  for each example $(x_i, y_i) \in \mathcal{B}$ **do**
    Sample example (x_j, y_j) according to the probability $P(\cdot \mid (x_i, y_i))$
    Linearly interpolate $(x_i,\ y_i)$ and $(x_j,\ y_j)$ to get $(\tilde{x}, \tilde{y})$
  Use interpolated examples to update the model via Eqn. (3).

3.2 mixReg for Improving In-distribution Generalization

In this section, we show that mixReg provably improves in-distribution generalization when the features are observed with noise, and the response depends on a small fraction of the features in a monotonic way. Specifically, we consider the following single index model with measurement error,

$$y = g(\theta^\top z) + \epsilon, \tag{7}$$

where $\theta \in \mathbb{R}^p$ and ϵ is a sub-Gaussian random variable and g is a monotonic transformation. Since images are often inaccurately observed in practice, we assume the feature z is observed or measured with noise, and denote the observed value by x: $x = z + \xi$ with ξ being a random vector with mean 0 and covariance matrix $\sigma_\xi^2 I$. We assume g to be monotonic to model the nearly one-to-one correspondence between causal features (e.g., the car pose in Figure 1(a)) and labels (rotation) in the in-distribution setting. The out-of-distribution setting will be discussed in Section 3.4. We would like to also comment that the single index model has been commonly used in econometrics, statistics, and deep learning theory [15, 21, 38, 43, 60].

Suppose we have $\{(x_i, y_i)\}_{i=1}^N$ i.i.d. drawn from the above model. We first follow the single index model literature (e.g. [60]) and estimate θ by minimizing the square error $\sum_{i=1}^n (\tilde{y}_i - \tilde{x}_i^\top \theta)^2$, where the $(\tilde{x}_i, \tilde{y}_i)'s$ are the augmented data by either vanilla mixup, mixup with input feature similarity, and mixReg. We denote the solution by θ^*_{mixup}, θ^*_{feat}, and θ^*_{mixReg} respectively. Given an estimate θ^*, we estimate g by $\hat{g}$ via the standard nonparametric kernel estimator [52] (we specify this in detail in Appendix B.1 for completeness) using the augmented data. We consider the mean square error metric as $\mathrm{MSE}(\theta) = \mathbb{E}[(y - \hat{g}(\theta^\top x))^2]$, and then have the following theorem (proof: Appendix B.1):

Theorem 1. *Suppose $\theta \in \mathbb{R}^p$ is sparse with sparsity $s = o(\min\{p, \sigma_\xi^2\})$, $p = o(N)$ and g is smooth with $c_0 < g' < c_1$, $c_2 < g'' < c_3$ for some universal constants $c_0, c_1, c_2, c_3 > 0$. There exists a distribution on x with a kernel function, such that when the sample size N is sufficiently large, with probability $1 - o(1)$,*

$$\mathrm{MSE}(\theta^*_{mixReg}) < \min(\mathrm{MSE}(\theta^*_{feat}), \mathrm{MSE}(\theta^*_{mixup})). \tag{8}$$

The high-level intuition of why mixReg helps is that the vanilla mixup imposes linearity regularization on the relationship between the feature and response. When the relationship is strongly nonlinear and one-to-one, such a regularization hurts the generalization, but could be mitigated by mixReg.

3.3 mixReg for Improving Task Generalization

The second benefit of mixReg is improving task generalization in meta-learning when the data from each task follows the model discussed in the last section. Concretely, we apply mixReg to MetaMix [61]. For each query example, the support example with a more similar label will have a higher probability of being mixed. The algorithm of mixReg on MetaMix is summarized in Appendix A.2.

Similar to in-distribution generalization analysis, we consider the following data generative model: for the m-th task ($m \in [M]$), we have $(x^{(m)}, y^{(m)}) \sim \mathcal{T}_m$ with

$$y^{(m)} = g_m(\theta^\top z^{(m)}) + \epsilon \quad \text{and} \quad x^{(m)} = z^{(m)} + \epsilon^{(m)}. \tag{9}$$

Here, θ denotes the globally-shared representation, and g_m's are the task-specific transformations. Note that, the formulation is close to [44] and wildly applied to theoretical analysis of meta-learning [61, 63]. Following similar spirit of [50] and last section, we obtain the estimation of θ by $\theta^* = \frac{1}{M}\sum_{m=1}^{M}(\mathbb{E}_{(\tilde{x}^{(m)},\tilde{y}^{(m)})\in\hat{\mathcal{D}}_m}[\tilde{y}^{(m)} - \theta^\top \tilde{x}^{(m)}])$. Here, $\hat{\mathcal{D}}_m$ denotes the generic dataset augmented by different approaches, including the vanilla MetaMix, MetaMix with input feature similarity, and mixReg. We denote these approaches by $\theta^*_{MetaMix}$, $\theta^*_{Meta-feat}$, and $\theta^*_{Meta-mixReg}$ respectively. For a new task $\mathcal{T}_t$, we again use the standard nonparametric kernel estimator to estimate g_t via the augmented target data. We then consider the following error metric $\mathrm{MSE}_{\mathrm{Target}}(\theta^*) = \mathbb{E}_{(x,y)\sim\mathcal{T}_t}[(y - \hat{g}_t({\theta^*}^\top x))^2]$.

Based on this metric, we get the following theorem to show the promise of mixReg in improving task generalization (see Appendix B.2 for detailed proof). Here, mixReg achieves smaller $\mathrm{MSE}_{\mathrm{Target}}$ compared to vanilla MetaMix and MetaMix with input feature similarity.

Theorem 2. *Let $N = \sum_{m=1}^{M} N_m$ and N_m is the number of examples of $\mathcal{T}_m$. Suppose θ_k is sparse with sparsity $s = o(\min\{d, \sigma_\xi^2\})$, $p = o(N)$ and g_m's are smooth with $0 < g'_m < c_1$, $c_2 < g''_m < c_3$ for some universal constants $c_1, c_2, c_3 > 0$ and $m \in [M] \cup \{t\}$. There exists a distribution on x with a kernel function, such that when the sample size N is sufficiently large, with probability $1 - o(1)$,*

$$\mathrm{MSE}_{\mathrm{Target}}(\theta^*_{Meta-mixReg}) < \min(\mathrm{MSE}_{\mathrm{Target}}(\theta^*_{Meta-feat}), \mathrm{MSE}_{\mathrm{Target}}(\theta^*_{MetaMix})). \tag{10}$$

3.4 mixReg for Improving Out-of-distribution Robustness

Finally, we show that mixReg improves OOD robustness in the covariate shift setting where the some unrelated features vary across different domains. In this setting, we regard the entire data distribution consisting of $\mathcal{E} = \{1, \ldots, E\}$ domains, where each domain is associated with a data distribution P_e for $e \in \mathcal{E}$. Given a set of training domains $\mathcal{E}^{tr} \subseteq \mathcal{E}$, we aim to make the trained model generalize well to an unseen test domain $\mathcal{E}^{ts}$ that is not necessarily in $\mathcal{E}^{tr}$. Here, we focus on covariate shift, i.e., the change of P_e among domains is only caused by the change of marginal distribution $P_e(X)$, while the conditional distribution $P_e(Y|X)$ is fixed across different domains.

To overcome covariate shift, mixing examples with close labels without considering domain information can effectively average out domain-changeable correlations and make the predicted values rely on the invariant causal features. To further understand how mixReg improves the robustness to covariate shift, we provide the following theoretical analysis.

We assume the training data $(x_i, y_i)_{i=1}^n$ follows $x_i = (z_i; a_i) \in \mathbb{R}^{p_1+p_2}$ and $y_i = \theta^\top x_i + \epsilon_i$, where $z_i \in \mathbb{R}^{p_1}$ and $a_i \in \mathbb{R}^{p_2}$ are regarded as invariant and domain-changeable unrelated features, respectively, and the last p_2 coordinates of $\theta \in \mathbb{R}^{p_1+p_2}$ are 0. Now we consider the case where the training data consists of a pair of domains with almost identical invariant features and opposite domain-changeable features, i.e., $x_i = (z_i, a_i), x'_i = (z'_i, a'_i)$, where $z_i \sim \mathcal{N}_{p_1}(0, \sigma_x^2 I_{p_1})$, $z'_i = z_i + \epsilon'_i$, $a_i \sim \mathcal{N}_{p_2}(0, \sigma_a^2 I_{p_2})$, $a'_i = -a_i + \epsilon''_i$. $\epsilon_i, \epsilon'_i, \epsilon_i''$ are noise terms with mean 0 and sub-Gaussian norm bounded by σ_ϵ. We use ridge estimator $\theta^*(k) = \arg\min_\theta(\sum_i \|y_i - \theta^\top x_i\|^2 + k\|\theta\|^2)$ to reflect the implicit regularization effect of deep neural networks [39]. The covariant shift happens when the test domains have different a_i distributions compared with training domains. We then have the following theorem to show that mixReg can improve robustness to covariant shift (proof: Appendix B.3):

Theorem 3. *Supposed for some $\max(\exp(-n^{1-o(1)}), \exp(-\frac{p_1^2}{2n})) < \delta \ll 1$, we have variance constraints: $\sigma_a = c_1\sigma_x$, $\sigma_x \geq c_2 \max(\frac{n^{5/2}}{\|\theta\|^\delta}\sigma_\epsilon, \frac{\sqrt{p_2}\|\theta\|}{\sqrt{n}p_1})$ and $\sigma_\epsilon^2 \leq \frac{c_3}{pn^{3/2}}$. Then for any penalty k satisfies $c_4\sqrt{\frac{p_2}{p_1}}n^{1/4+o(1)} < k < c_5 \min(\frac{\sigma_x}{\|\theta\|}\sqrt{p_1 n^{1-o(1)}}, n)$ and bandwidth h satisfies $0 < h \leq c_6 \frac{l}{\sqrt{\log(n^2/p_1)}}$ in mixReg, when n is sufficiently large, with probability at least $1 - o(1)$, we have*

$$\mathrm{MSE}(\theta^*_{mixReg}) < \min(\mathrm{MSE}(\theta^*_{feat}), \mathrm{MSE}(\theta^*_{mixup})), \tag{11}$$

where $c_1 \geq 1$, c_2, c_3, c_4, c_5, c_6 > 0 are universal constants, $l = \min_{i\neq j}|y_i - y'_j|$ and $p_1 \ll n < p_1^2$.

4 Experiments

In this section, we evaluate the performance of mixReg, aiming to answer the following questions: **Q1:** Compared to prior approaches, can mixReg improve the in-distribution, task generalization, and

Table 1: Results for in-distribution generalization. We report the averaged RMSE and MAPE of three seeds. Full results with standard deviation are reported in Appendix C.4. The best results and second best results are **bold** and underlined, respectively.

	Airfoil		NO2		Exchange-Rate		Electricity		Echo	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
ERM	2.901	1.753%	0.537	13.615%	0.0236	2.423%	0.0581	13.861%	5.402	8.700%
mixup	3.730	2.327%	0.528	13.534%	0.0239	2.441%	0.0585	14.306%	5.393	8.838%
Mani mixup	3.063	1.842%	0.522	13.382%	0.0242	2.475%	0.0583	14.556%	5.482	8.955%
k-mixup	2.938	1.769%	0.519	13.173%	0.0236	2.403%	0.0575	14.134%	5.518	9.206%
Local Mixup	3.703	2.290%	0.517	13.202%	0.0236	2.341%	0.0582	14.245%	5.652	9.313%
MixRL	3.614	2.163%	0.527	13.298%	0.0238	2.397%	0.0585	14.417%	5.618	9.165%
mixReg	**2.717**	**1.610%**	**0.509**	**12.998%**	**0.0203**	**2.041%**	**0.0570**	**13.372%**	**5.177**	**8.435%**

out-of-distribution robustness on regression? **Q2:** How does mixReg perform compared to using other distance metrics? **Q3:** Is mixReg sensitive to the choice of bandwidth σ in Eqn. (6)? In our experiments, we apply cross-validation to tune all hyperparameters with grid search.

4.1 In-Distribution Generalization

Datasets. The following five datasets are used to evaluate the performance of in-distribution generalization (see Appendix C.1 for detailed data statistics). **(1)&(2) Airfoil Self-Noise (Airfoil) and NO2 [29]** are both are tabular datasets, where airfoil contains aerodynamic and acoustic test results of airfoil blade sections and NO2 aims to predict the mount of air pollution at a particular location. **(3)&(4): Exchange-Rate, and Electricity [34]** are two time-series datasets, where Exchange-Rate reports the collection of the daily exchange rates and Electricity is used to predict the hourly electricity consumption. **(5) Echocardiogram Videos (Echo): [41]** is a ejection fraction prediction dataset, which consists of a series of videos illustrating the heart from different aspects.

Comparisons and Experimental Setups. We first compare mixReg with mixup and its variants (Manifold mixup [56], k-Mixup [16] and Local Mixup [4]) that can be easily to adapted to regression tasks. Second, MixRL, a recent reinforcement learning framework to select mixup pairs in regression, is also used for comparison. Note that, for k-Mixup, Local Mixup, MixRL, and mixReg, we apply them to both mixup and Manifold Mixup and report the best-performing combination.

For Airfoil and NO2, we employ a three-layer fully connected network as the backbone model. We use LST-Attn [34] for the Exchange-Rate and Electricity, and EchoNet-Dynamic [41] for predicting the ejection fraction. We use Root Mean Square Error (RMSE) and Mean Averaged Percentage Error (MAPE) as evaluation metrics. Detailed experimental setups are in Appendix C.2.

Results. We report the results in Table 1 and have the following observations. *First*, vanilla mixup and manifold mixup are typically less performant than ERM when they are applied directly to regression tasks. These results support our hypothesis that random selection of example pairs for mixing may produce arbitrary inaccurate virtual labels. *Second*, while limiting the scope of interpolation (e.g., k-mixup, manifold k-mixup, Local Mixup) helps to improve generalization in most cases, the performance of these approaches is inconsistent across different datasets. As an example, Local Mixup outperforms ERM in NO2 and Exchange-Rate, but fails to benefit results in Electricity. *Third*, even in more complicated datasets, such as Electricity and Echo, MixRL performs worse than Mixup and ERM, indicating that it is non-trivial to train a policy network that works well. *Finally*, mixReg consistently outperforms mixup and its variants, ERM, and MixRL, demonstrating its capability to improve in-distribution generalization on regression problems.

Analysis of Overfitting. In Figure 2, we visualize the test loss (RMSE) and the generalization gap between training loss and test loss of ERM, Manifold Mixup, and mixReg with respect to the training epoch for Exchange-Rate. More results are illustrated in Appendix C.3. Compared with ERM and mixReg, the better test performance and smaller generalization gap of mixReg demonstrates its ability to improve in-distribution generalization and mitigate overfitting.

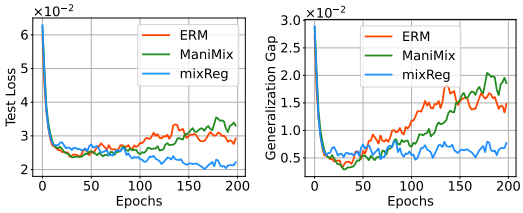


Figure 2: Overfitting Analysis on Exchange-Rate

Table 3: Results for out-of-distribution robustness. We report the averaged and worst-domain performance here and the full results are listed in Appendix E.3. Higher R or lower RMSE represent better performance. For PovertyMap, most results are copied from WILDS benchmark [28] and Worst R is the primary metric. We tune the hyperparameters of DTI and achieves better performance compared with the original paper [22]. We bold the best results and underline the second best results.

	PovertyMap (R)		Crime (RMSE)		SkillCraft (RMSE)		DTI (R)	
	Avg. ↑	Worst ↑	Avg. ↓	Worst ↓	Avg. ↓	Worst ↓	Avg. ↑	Worst ↑
ERM	0.80	<u>0.50</u>	0.134	0.173	5.887	10.182	0.464	0.429
IRM	0.77	0.43	<u>0.127</u>	0.155	5.937	7.849	0.478	0.432
IB-IRM	0.78	0.40	0.127	0.153	6.055	7.650	0.479	0.435
V-REx	**0.83**	0.48	0.129	0.157	6.059	<u>7.444</u>	<u>0.485</u>	0.435
CORAL	0.78	0.44	0.133	0.166	6.353	8.272	0.483	0.432
GroupDRO	0.75	0.39	0.138	0.168	6.155	8.131	0.442	0.407
Fish	0.80	0.30	0.128	<u>0.152</u>	6.356	8.676	0.470	<u>0.443</u>
mixup	<u>0.81</u>	0.46	0.128	0.154	<u>5.764</u>	9.206	0.465	0.437
mixReg (Ours)	**0.83**	**0.51**	**0.123**	**0.146**	**5.201**	**7.362**	**0.498**	**0.458**

4.2 Task Generalization

Dataset and Experimental Setups. To evaluate task generalization in meta-learning settings, we use two rotation prediction datasets named as ShapeNet1D [14] and Pascal1D [58, 65], aiming to predict an object's rotation relative to the canonical orientation. Each task is rotation regression for one object, where the model takes a 128×128 grey-scale image as the input, and the output is an azimuth angle normalized between $[0, 10]$ [65]. We detail the description in Appendix D.1.

Since MetaMix outperforms most other methods in PASCAL3D in Yao et al. [61], we only select two other representative approaches – Meta-Aug [45] and MR-MAML [65] for comparison. We further apply k-mixup and Local Mixup to MetaMix, which are called as k-MetaMix and Local MetaMix, respectively. Following Yin et al. [65], the base model consists of an encoder with three convolutional blocks and a decoder with four convolutional blocks. All baselines use the same base model. Hyperparameters are listed in Appendix D.2.

Results. Table 2 shows the average MSE with a 95% confidence interval over 6000 meta-testing tasks. Our results corroborate the findings of Yao et al. [61] that MetaMix improves performance compared with non-mixup approaches (i.e., MAML, MR-MAML, Meta-Aug). Similar to our previous in-distribution generalization findings, the better performance of Local MetaMix over MetaMix demonstrates the effectiveness of interpolating nearby examples. By mixing query examples with support examples that have similar labels, mixReg outperforms all of the other approaches on both datasets, verifying its effectiveness of improving task generalization.

Table 2: Meta-regression performance (MSE ± 95% confidence interval) on 6000 meta-test tasks.

Model	ShapeNet1D ↓	PASCAL3D ↓
MAML	4.698 ± 0.079	2.370 ± 0.072
MR-MAML	4.433 ± 0.083	2.276 ± 0.075
Meta-Aug	4.312 ± 0.086	2.298 ± 0.071
MetaMix	4.275 ± 0.082	2.135 ± 0.070
k-MetaMix	4.268 ± 0.078	<u>2.091 ± 0.069</u>
Local MetaMix	<u>4.201 ± 0.087</u>	2.107 ± 0.078
mixReg (Ours)	$\mathbf{4.024 \pm 0.081}$	$\mathbf{1.995 \pm 0.067}$

4.3 Out-of-Distribution Robustness

Datasets. In the following, we briefly discuss the datasets used to evaluate out-of-distribution robustness; the detailed data descriptions in Appendix E.1. **(1) PovertyMap [28]** is a satellite image regression dataset, aiming to estimate asset wealth in countries that are not shown in the training set. **(2) Communities and Crime (Crime)** [46] is a tabular dataset, where the problem is to predict total number of violent crimes per 100K population and we aim to generalize the model to unseen states. **(3) SkillCraft1 Master Table (SkillCraft)** [5] is a tabular dataset, aiming to predict the mean latency from the onset of a perception action cycles to their first action in milliseconds. Here, "LeagueIndex" is treated as domain information. **(4) Drug-target Interactions (DTI)** [22] is aiming to predict out-of-distribution drug-target interactions in 2019-2020 after training on 2013-2018.

Comparisons and Experimental Setups. To evaluate the out-of-distribution robustness of mixReg, we compare it with nine invariant learning approaches that can be adapted to regression tasks, including ERM, IRM [3], IB-IRM [1], V-REx [33], CORAL [35], DRNN [12], GroupDRO [47],

Fish [48], and mixup. All approaches use the same backbone model, where we adopt ResNet-50, three-layer full connected network, and DeepDTA [42] for Poverty, Crime and SkillCraft, and DTI, respectively. Following the original papers of PovertyMap [64] and DTI [22], we use R-value to evaluate the performance. For Crime and SkillCraft, we use RMSE as the evaluation metric. Detailed setups and hyperparameters are listed in Appendix E.2

Results. We report both average performance and worst-domain performance in Table 3. According to the results, we can see that the performance of prior invariant learning approaches is not stable across datasets. For example, IRM and CORAL outperform ERM on DTI, but fail to improve the performance on PovertyMap. mixReg instead consistently shows the best performance regardless of the data types, indicating its effectiveness in improving robustness to covariate shift.

Analysis of Learned Invariance. Following Yao et al. [62], we analyze the domain invariance of mixReg. In regression, we measure domain invariance as pairwise divergence of the last hidden representation. Since the labels are continuous, we first evenly split examples into $\mathcal{C} = \{1, \dots, C\}$ bins, where $C = 10$. After splitting, we perform kernel density estimation to estimate the probability density function $P(h_e^c)$ of the last hidden representation. We calculate the pairwise divergence as $\frac{1}{|\mathcal{C}||\mathcal{E}|^2} \sum_{c \in \mathcal{C}} \sum_{e', e \in \mathcal{E}} \mathrm{KL}(P(g_E^c \mid E = e) | P(g_E^c \mid E = e'))$. The results on Crime and SkillCraft are reported in Table 4, indicating that mixReg learns better invariant representation than prior invariant learning methods.

Table 4: Results ($\times 10^4$) of invariance analysis. Smaller pairwise divergence value represents stronger invariance.

Model	Crime	SkillCraft
ERM	2.010	8.156
mixup	1.439	1.071
IRM	1.728	1.208
CORAL	1.821	0.943
mixReg (Ours)	**1.199**	**0.913**

4.4 Analysis of mixReg

In this section, we conduct three analyses including the compatibility of mixReg, alternative distance metrics, and sensitivity of bandwidth. In Appendix F.3.2 and F.4, we analyze the sensitivity of hyperparameter α in Beta distribution and the robustness of mixReg to label noise, respectively.

I. Compatibility of mixReg. mixReg is a complementary approach to vanilla mixup and its variants, where it changes the probabilities of sampling mixing pairs instead of changing the way to mixing. We further conduct compatibility analysis of mixReg by integrating it to three representative mixup variants – PuzzleMix [26], CutMix [67], AutoMix [72]. We evaluate the performance on three real-world datasets (Exchange-Rate, Echo, PovertyMap). The reported results in Table 5 indicate the compatibility and efficacy of mixReg in regression.

Table 5: Compatibility analysis. See Appendix F.1 for full results.

Model		Ex.-Rate	Echo	PovertyMap
		RMSE ↓	RMSE ↓	Worst R ↑
CutMix		0.0264	5.405	0.49
	+mixReg	**0.0240**	**5.161**	**0.52**
PuzzleMix		0.0254	5.368	0.47
	+mixReg	**0.0233**	**5.206**	**0.50**
AutoMix		0.0242	5.525	0.50
	+mixReg	**0.0228**	**5.239**	**0.53**

II. Analysis of Distance Metrics. Besides the theoretical analysis, here we empirically analyze the effectiveness of using label distance. Here, we use $d(a, b)$ to denote the distance between objects a and b, e.g., $d(y_i, y_j)$ in our case. We consider four substitute distance metrics, including: (1) *feature distance*: $d(x_i, x_j)$; (2) *feature and label distance*: we concatenate the input feature x and label y to compute the distance, i.e., $d(x_i \oplus y_i, x_j \oplus y_j)$; (3) *representation distance*: since using feature distance may fail to capture high-level feature relations, we propose to use the model's hidden representations to measure the distance $d(h_i, h_j)$, detailed in Appendix F.2. (4) *representation and label distance*: $d(h_i \oplus y_i, h_j \oplus y_j)$.

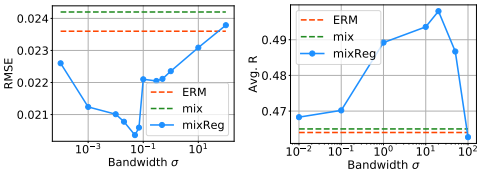
Table 6: Performance of different distance metrics. x, y, h represent input feature, label, and hidden representation, respectively. The full results are listed in Appendix F.2.

Model	Ex.-Rate	Shape1D	DTI
	RMSE ↓	MSE ↓	Avg. R ↑
ERM/MAML	0.0236	4.698	0.464
mixup/MetaMix	0.0239	4.275	0.465
$d(x_i, x_j)$	0.0212	4.539	0.478
$d(x_i \oplus y_i, x_j \oplus y_j)$	0.0212	4.395	0.484
$d(h_i, h_j)$	0.0213	4.202	0.483
$d(h_i \oplus y_i, h_j \oplus y_j)$	0.0208	4.176	0.487
$d(y_i, y_j)$ **(mixReg)**	**0.0203**	**4.024**	**0.498**

The results are reported in Table 6. We find that (1) feature distance $d(x_i, x_j)$ does benefit performance compared with ERM and vanilla mixup in all cases since it selects more reasonable example pairs for mixing; (2) representation distance $d(h_i, h_j)$ outperforms feature distance $d(x_i, x_j)$ since it captures high-level features; (3) though involving label in representation distance is likely to overwhelm the effect of labels, the better performance of $d(h_i \oplus y_i, h_j \oplus y_j)$ over $d(h_i, h_j)$ indicates the effectiveness

of regression labels in measuring example distance, which is further verified by the superiority of mixReg. (4) Our discussion of computational efficiency in Appendix A.3 indicates that mixReg is much more computationally efficient than other distance measurements.

III. How does the choice of bandwidth affects the performance? Finally, we analyze the effects of the bandwidth σ in Eqn. (6). The performance with respect to bandwidth of Exchange-Rate and DTI are visualized in Figure 3, respectively. We point out that if the bandwidth is too large, the results are close to vanilla mixup or Manifold Mixup, depending on which one is used. Otherwise, the results are close to ERM if the bandwidth is too small. According to the results, we see that the improvements of mixReg is somewhat stable over different bandwidths. Most importantly, mixReg yields a good model for a wide range of bandwidths, which reduces the efforts to tune the bandwidth for every specific dataset. Additionally, we provide empirical guidance about how to pick suitable bandwidth in Appendix F.3.1.



(a) : Exchange-Rate (b) : DTI

Figure 3: Robustness Analysis of Bandwidth. mix: better mixing approach between mixup and Manifold Mixup.

5 Related Work

Data Augmentation and mixup. Various data augmentation strategies have been proposed to improve the generalization of deep neural networks, including directly augmenting images with manually designed strategies (e.g., whitening, cropping) [31], generating more examples with generative models [2, 6, 73, 70], and automatically finding augmentation strategies [7, 8, 36]. Mixup [68] and its variants [9, 16, 19, 20, 27, 26, 54–56, 67, 72] propose to improve generalization by linear interpolating input features of a pair of examples and their corresponding labels. Though mixup and its variants have demonstrated their power in classification [68], sequence labeling [69], and reinforcement learning [57], systematic analysis of mixup on different regression tasks is still underexplored. The recent MixRL [23] learns a policy network to select nearby example pairs for mixing, which requires substantial computational resources and is not suitable to high-dimensional real-world data. Unlike this method, mixReg instead adjusts the sampling probability of mixing pairs based on label similarity, which makes it much more efficient. Furthermore, mixReg is an complementary method over mixup and its representative variants (see more discussion in Appendix A.4). Empirically, our experiments show the effectiveness and compatibility of mixReg on multiple regression tasks in Section 4.1.

Task Generalization. Our experiments extended mixReg to gradient-based meta-learning, aiming to improve task generalization. In the literature, there are two lines of related works. The first line of research directly imposes regularization on meta-learning algorithms [17, 24, 51, 65]. The second line of approaches introduces task augmentation to produce more tasks for meta-training, including imposing label noise [45], mixing support and query sets in the outer-loop optimization [40, 61], and directly interpolating tasks to densify the entire task distribution [63]. mixReg is complimentary to the latter mixup-based task augmentation methods. Furthermore, Section 4.2 indicates that mixReg empirically outperforms multiple representative prior approaches [65, 45].

Out-of-distribution Robustness. Many recent methods aim to build machine learning models that are robust to distribution shift, including learning invariant representations with domain alignment [13, 35, 37, 53, 66, 71, 59] or using explicit regularizers to finding a invariant predictors that performs well over all domains [1, 3, 18, 25, 30, 33]. Recently, LISA [62] cancels out domain-associated correlations and learns invariant predictors by mixing examples either with the same label but different domains or with the same domain but different labels. While related, mixReg can be considered as a more general version of LISA that can be used for regression tasks. Besides, unlike LISA, mixReg do not use domain annotations, which are often expensive to obtain.

6 Conclusion

In this paper, we propose mixReg, a simple yet effective variant of mixup that is well-suited to regression tasks in deep neural networks. Specifically, mixReg adjusts the sampling probability of mixing example pairs by assigning higher probability to pairs with closer label values. Both theoretical and empirical results demonstrate the promise of mixReg in improving in-distribution generalization, task generalization, and out-of-distribution robustness.

References

[1] Kartik Ahuja, Ethan Caballero, Dinghuai Zhang, Yoshua Bengio, Ioannis Mitliagkas, and Irina Rish. Invariance principle meets information bottleneck for out-of-distribution generalization. 2021.

[2] Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2017.

[3] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

[4] Raphael Baena, Lucas Drumetz, and Vincent Gripon. Preventing manifold intrusion with locality: Local mixup. *arXiv preprint arXiv:2201.04368*, 2022.

[5] Mark Blair, Joe Thompson, Andrew Henrey, and Bill Chen. SkillCraft1 Master Table Dataset. UCI Machine Learning Repository, 2013.

[6] Christopher Bowles, Liang Chen, Ricardo Guerrero, Paul Bentley, Roger Gunn, Alexander Hammers, David Alexander Dickie, Maria Valdés Hernández, Joanna Wardlaw, and Daniel Rueckert. Gan augmentation: Augmenting training data using generative adversarial networks. *arXiv preprint arXiv:1810.10863*, 2018.

[7] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 113–123, 2019.

[8] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020.

[9] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[10] Chelsea Finn and Sergey Levine. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. *arXiv preprint arXiv:1710.11622*, 2017.

[11] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, pages 1050–1059, 2016.

[12] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR, 2015.

[13] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

[14] Ning Gao, Hanna Ziesche, Ngo Anh Vien, Michael Volpp, and Gerhard Neumann. What matters for meta-learning vision regression tasks? 2022.

[15] Rong Ge, Rohith Kuditipudi, Zhize Li, and Xiang Wang. Learning two-layer neural networks with symmetric inputs. In *International Conference on Learning Representations*, 2019.

[16] Kristjan Greenewald, Anming Gu, Mikhail Yurochkin, Justin Solomon, and Edward Chien. k-mixup regularization for deep learning via optimal transport. *arXiv preprint arXiv:2106.02933*, 2021.

[17] Simon Guiroy, Vikas Verma, and Christopher Pal. Towards understanding generalization in gradient-based meta-learning. *arXiv preprint arXiv:1907.07287*, 2019.

[18] Ruocheng Guo, Pengchuan Zhang, Hao Liu, and Emre Kiciman. Out-of-distribution prediction with invariant risk minimization: The limitation and an effective fix. *arXiv preprint arXiv:2101.07732*, 2021.

[19] Dan Hendrycks, Norman Mu, Ekin D Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. *arXiv preprint arXiv:1912.02781*, 2019.

[20] Minui Hong, Jinwoo Choi, and Gunhee Kim. Stylemix: Separating content and style for enhanced data augmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14862–14870, 2021.

[21] Joel L Horowitz. *Semiparametric and nonparametric methods in econometrics*, volume 12. Springer, 2009.

[22] Kexin Huang, Tianfan Fu, Wenhao Gao, Yue Zhao, Yusuf Roohani, Jure Leskovec, Connor W Coley, Cao Xiao, Jimeng Sun, and Marinka Zitnik. Therapeutics data commons: Machine learning datasets and tasks for therapeutics. *arXiv preprint arXiv:2102.09548*, 2021.

[23] Seong-Hyeon Hwang and Steven Euijong Whang. Mixrl: Data mixing augmentation for regression using reinforcement learning. *arXiv preprint arXiv:2106.03374*, 2021.

[24] Muhammad Abdullah Jamal and Guo-Jun Qi. Task agnostic meta-learning for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11719–11727, 2019.

[25] Kia Khezeli, Arno Blaas, Frank Soboczenski, Nicholas Chia, and John Kalantari. On invariance penalties for risk minimization. *arXiv preprint arXiv:2106.09777*, 2021.

[26] Jang-Hyun Kim, Wonho Choo, and Hyun Oh Song. Puzzle mix: Exploiting saliency and local statistics for optimal mixup. In *International Conference on Machine Learning*, pages 5275–5285. PMLR, 2020.

[27] Jang-Hyun Kim, Wonho Choo, Hosan Jeong, and Hyun Oh Song. Co-mixup: Saliency guided joint mixup with supermodular diversity. *arXiv preprint arXiv:2102.03065*, 2021.

[28] Pang Wei Koh, Shiori Sagawa, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, pages 5637–5664. PMLR, 2021.

[29] Charles Kooperberg. Statlib: an archive for statistical software, datasets, and information. *The American Statistician*, 51(1):98, 1997.

[30] Masanori Koyama and Shoichiro Yamaguchi. Out-of-distribution generalization with maximal invariant predictor. *arXiv preprint arXiv:2008.01883*, 2020.

[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Annual Conference on Neural Information Processing Systems*, pages 1097–1105, 2012.

[32] Anders Krogh and John A Hertz. A simple weight decay can improve generalization. In *NeurIPS*, pages 950–957, 1992.

[33] David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Dinghuai Zhang, Remi Le Priol, and Aaron Courville. Out-of-distribution generalization via risk extrapolation (rex). In *International Conference on Machine Learning*, pages 5815–5826. PMLR, 2021.

[34] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 95–104, 2018.

[35] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot. Domain generalization with adversarial feature learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5400–5409, 2018.

[36] Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast autoaugment. *Advances in Neural Information Processing Systems*, 32, 2019.

[37] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pages 97–105. PMLR, 2015.

[38] Kaifeng Lyu and Jian Li. Gradient descent maximizes the margin of homogeneous neural networks. In *International Conference on Learning Representations*, 2019.

[39] Behnam Neyshabur. Implicit regularization in deep learning. *arXiv preprint arXiv:1709.01953*, 2017.

[40] Renkun Ni, Micah Goldblum, Amr Sharaf, Kezhi Kong, and Tom Goldstein. Data augmentation for meta-learning. *ICML*, 2021.

[41] David Ouyang, Bryan He, Amirata Ghorbani, Neal Yuan, Joseph Ebinger, Curtis P Langlotz, Paul A Heidenreich, Robert A Harrington, David H Liang, Euan A Ashley, et al. Video-based ai for beat-to-beat assessment of cardiac function. *Nature*, 580(7802):252–256, 2020.

[42] Hakime Öztürk, Arzucan Özgür, and Elif Ozkirimli. Deepdta: deep drug–target binding affinity prediction. *Bioinformatics*, 34(17):i821–i829, 2018.

[43] James L Powell, James H Stock, and Thomas M Stoker. Semiparametric estimation of index coefficients. *Econometrica: Journal of the Econometric Society*, pages 1403–1430, 1989.

[44] Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. In *International Conference on Learning Representations*, 2020.

[45] Janarthanan Rajendran, Alex Irpan, and Eric Jang. Meta-learning requires meta-augmentation. In *International Conference on Neural Information Processing Systems*, 2020.

[46] Michael Redmond. Communities and Crime. UCI Machine Learning Repository, 2009.

[47] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. In *ICLR*, 2020.

[48] Yuge Shi, Jeffrey Seely, Philip HS Torr, N Siddharth, Awni Hannun, Nicolas Usunier, and Gabriel Synnaeve. Gradient matching for domain generalization. *arXiv preprint arXiv:2104.09937*, 2021.

[49] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958, 2014.

[50] Nilesh Tripuraneni, Chi Jin, and Michael Jordan. Provable meta-learning of linear representations. In *International Conference on Machine Learning*, pages 10434–10443. PMLR, 2021.

[51] Hung-Yu Tseng, Yi-Wen Chen, Yi-Hsuan Tsai, Sifei Liu, Yen-Yu Lin, and Ming-Hsuan Yang. Regularizing meta-learning via gradient dropout. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

[52] Alexandre B Tsybakov. Introduction to nonparametric estimation, 2009. *URL https://doi.org/10.1007/b13794. Revised and extended from the*, 9(10), 2004.

[53] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[54] AFM Uddin, Mst Monira, Wheemyung Shin, TaeChoong Chung, Sung-Ho Bae, et al. Saliencymix: A saliency guided data augmentation strategy for better regularization. *arXiv preprint arXiv:2006.01791*, 2020.

[55] Shashanka Venkataramanan, Ewa Kijak, Laurent Amsaleg, and Yannis Avrithis. Alignmixup: Improving representations by interpolating aligned features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19174–19183, 2022.

[56] 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 *International Conference on Machine Learning*, pages 6438–6447. PMLR, 2019.

[57] Kaixin Wang, Bingyi Kang, Jie Shao, and Jiashi Feng. Improving generalization in reinforcement learning with mixture regularization. *Advances in Neural Information Processing Systems*, 33:7968–7978, 2020.

[58] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *WACV*, pages 75–82. IEEE, 2014.

[59] Minghao Xu, Jian Zhang, Bingbing Ni, Teng Li, Chengjie Wang, Qi Tian, and Wenjun Zhang. Adversarial domain adaptation with domain mixup. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6502–6509, 2020.

[60] Zhuoran Yang, Krishnakumar Balasubramanian, and Han Liu. High-dimensional non-gaussian single index models via thresholded score function estimation. In *International conference on machine learning*, pages 3851–3860. PMLR, 2017.

[61] Huaxiu Yao, Longkai Huang, Linjun Zhang, Ying Wei, Li Tian, James Zou, Junzhou Huang, and Zhenhui Li. Improving generalization in meta-learning via task augmentation. In *Proceeding of the Thirty-eighth International Conference on Machine Learning*, 2021.

[62] Huaxiu Yao, Yu Wang, Sai Li, Linjun Zhang, Weixin Liang, James Zou, and Chelsea Finn. Improving out-of-distribution robustness via selective augmentation. *arXiv preprint arXiv:2201.00299*, 2022.

[63] Huaxiu Yao, Linjun Zhang, and Chelsea Finn. Meta-learning with fewer tasks through task interpolation. In *Proceeding of the 10th International Conference on Learning Representations*, 2022.

[64] Christopher Yeh, Anthony Perez, Anne Driscoll, George Azzari, Zhongyi Tang, David Lobell, Stefano Ermon, and Marshall Burke. Using publicly available satellite imagery and deep learning to understand economic well-being in africa. *Nature Communications*, 2020.

[65] Mingzhang Yin, George Tucker, Mingyuan Zhou, Sergey Levine, and Chelsea Finn. Meta-learning without memorization. In *International Conference on Learning Representations*, 2020.

[66] Xiangyu Yue, Yang Zhang, Sicheng Zhao, Alberto Sangiovanni-Vincentelli, Kurt Keutzer, and Boqing Gong. Domain randomization and pyramid consistency: Simulation-to-real generalization without accessing target domain data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2100–2110, 2019.

[67] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6023–6032, 2019.

[68] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. 2018.

[69] Rongzhi Zhang, Yue Yu, and Chao Zhang. Seqmix: Augmenting active sequence labeling via sequence mixup. *arXiv preprint arXiv:2010.02322*, 2020.

[70] Allan Zhou, Fahim Tajwar, Alexander Robey, Tom Knowles, George J Pappas, Hamed Hassani, and Chelsea Finn. Do deep networks transfer invariances across classes? *arXiv preprint arXiv:2203.09739*, 2022.

[71] Kaiyang Zhou, Yongxin Yang, Timothy Hospedales, and Tao Xiang. Deep domain-adversarial image generation for domain generalisation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13025–13032, 2020.

[72] Jianchao Zhu, Liangliang Shi, Junchi Yan, and Hongyuan Zha. Automix: Mixup networks for sample interpolation via cooperative barycenter learning. In *European Conference on Computer Vision*, pages 633–649. Springer, 2020.

[73] Barret Zoph, Ekin D Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens, and Quoc V Le. Learning data augmentation strategies for object detection. In *European conference on computer vision*, pages 566–583. Springer, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Appendix G
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] mixReg is a general machine learning method and we believe that it has no potential negative societal impacts.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 3 and Appendix B
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix B
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] Our code will be released once the paper is accepted.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix C.2, D.2, and E.2.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Appendix C, D, E, F.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix C.2, D.2, and E.2.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]