

AUGMENT ON MANIFOLD: MIXUP REGULARIZATION WITH UMAP

Yousef El-Laham*

Elizabeth Fons*

Dillon Daudert[†]

Svitlana Vyetrenko*

J.P. Morgan AI Research*

Stony Brook University[†]

ABSTRACT

Data augmentation techniques play an important role in enhancing the performance of deep learning models. Despite their proven benefits in computer vision tasks, their application in the other domains remains limited. This paper proposes a Mixup regularization scheme, referred to as UMAP Mixup, designed for “on-manifold” automated data augmentation for deep learning predictive models. The proposed approach ensures that the Mixup operations result in synthesized samples that lie on the data manifold of the features and labels by utilizing a dimensionality reduction technique known as uniform manifold approximation and projection. Evaluations across diverse regression tasks show that UMAP Mixup is competitive with or outperforms other Mixup variants, show promise for its potential as an effective tool for enhancing the generalization performance of deep learning models.

Index Terms— data augmentation, Mixup regularization, dimensionality reduction, UMAP

1. INTRODUCTION

Data augmentation (DA) has been extensively explored for deep learning (DL) models in computer vision (see [1] for a review). Mainly, DA plays a pivotal role in improving the generalization of DL models, especially in contexts where available training data is scarce or imbalanced, whereby DL models are more susceptible to overfitting. Fundamentally, DA helps to implicitly regularize a DL model to capture underlying invariances in the data, which is an essential aspect of model generalization. As an example, the label of an image remains invariant across a variety of transformations, such as rotations, cropping, grayscaling, translation, and flipping. These transformations effectively expand the diversity of the training data, which in turn boosts the model’s ability to generalize and improves its performance on out-of-sample data. Despite these advances, the application of DA for DL models in other data modalities, such as tabular data or time series data, remains less explored. As an example, financial time series data poses unique challenges that standard data augmentation techniques might not adequately address. While there have been exploratory studies focusing on data augmentation for financial time series [2], a generalized framework that can affirm the benefits of synthetic data augmentation across diverse financial datasets is yet to be established. This challenge derives from the fact that invariance properties of financial data are not as obvious as those in computer vision.

In recent years, a technique known as Mixup has gained traction as a domain-agnostic form of data augmentation [3]. Mixup works by generating synthetic samples as convex combinations of two random samples during the training process. Notably, Mixup has shown a significant boost in performance comparison to standard training, especially on larger computer vision datasets. At its core, Mixup can be seen as a vicinal risk minimization (VRM) technique that aims to

regularize the learning algorithm by encouraging it to behave linearly in-between observed samples [4]. This stands in contrast to empirical risk minimization (ERM), a principle of statistical learning theory that focuses on minimizing the loss over the empirical distribution of the training data [5]. Over the past several years, a variety of Mixup variants have been proposed, each addressing specific challenges or application domains. These include Manifold Mixup [6], Remix [7], local Mixup [8] and C-Mixup [9], among others. For instance, Manifold Mixup extends the concept of Mixup to the hidden layers of a neural network, applying Mixup in an embedding obtained at some intermediate layer of the network, thereby promoting smoother behavior in the overall model. Another variant, C-Mixup, was introduced for regression tasks, and demonstrates some theoretical guarantees regarding its predictive performance over the standard Mixup technique for certain and simple classes of predictive models. Despite these advances, Mixup and its variants face several challenges. It remains difficult to derive a comprehensive theoretical framework that guarantees that Mixup will consistently improve the performance on out-of-sample, especially in contexts where the data may be subject to distribution shifts. Furthermore, for each Mixup technique, there is no way to guarantee that the mixing procedure of the features and labels will yield a synthesized sample that lies on the data manifold [10]. While the Mixup technique has been touted as a domain-agnostic method, its empirical evaluation and validation, particularly in the context of data modalities other than computer vision, remain sparse.

This work presents a novel Mixup regularization scheme for deep learning models that addresses some of these challenges. Specifically, we introduce a variant of Manifold Mixup that aims to guarantee that the Mixup operation results in a synthesized sample that lies on the data manifold. This is accomplished by training an intermediate layer of the predictor to learn an embedding that exhibits this property, using a technique called uniform manifold approximation and projection (UMAP) [11]. UMAP is a nonlinear dimensionality reduction method that optimizes embeddings to preserve the topological structure of the data. The proposed approach, which we refer to as UMAP Mixup, incorporates an additional UMAP-based regularizer during training to optimize the learned embedding used for constructing the Mixup augmentations. The UMAP-based regularizer encourages the model to assign similar latent embeddings to the data points that are near each other in feature space according to a pre-defined distance metric, thereby promoting better on-manifold data augmentation. Empirically, we demonstrate that UMAP Mixup obtains competitive or better performance than ERM and a variety of Mixup variants on various regression tasks.

2. PROBLEM FORMULATION

This work studies DA in supervised learning problems from the perspective of statistical learning theory. In supervised learning, we are interested in finding a function $f \in \mathcal{F}$ that describes the relationship

between a feature vector $x \in \mathbb{R}^{d_x}$ and an output $y \in \mathbb{R}^{d_y}$. Suppose that x and y have a joint cumulative distribution function $F(x, y)$. Our goal is in finding a function $f \in \mathcal{F}$ that minimizes the risk:

$$R[f] = \mathbb{E}[\ell(f(x), y)] = \int \ell(f(x), y) dF(x, y), \quad (1)$$

where ℓ denotes a loss function used to measure the discrepancy between function output and the true output. In practice, it is common to assume f belongs to some parametric family of functions \mathcal{F}_θ defined by parameters $\theta \in \Theta$. In this context, we would like to solve the following optimization problem:

$$\hat{\theta} = \arg \min_{\theta \in \Theta} R(\theta), \quad (2)$$

where $R(\theta) = \mathbb{E}[\ell(f_\theta(x), y)]$. Unfortunately, $F(x, y)$ is unknown and we must resort to using an approximations of the risk in order to solve (2). Typically, the ERM principle is utilized, whereby parameters are learned by minimizing an approximation of the risk using the empirical distribution of an observed dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$:

$$\hat{\theta}_{\text{ERM}} = \arg \min_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^N \ell(f_\theta(x_i), y_i). \quad (3)$$

The ERM principle results in a good parameter estimate if \mathcal{D} accurately captures the true distribution $F(x, y)$. This is not typically true in practice, and so regularization techniques are often required to help predictive models generalize better to out-of-sample data.

3. BACKGROUND

3.1. Mixup Regularization

Mixup is a VRM technique introduced to improve the generalization performance of deep learning models. The central idea of Mixup is to train with random convex combinations of training samples, rather than the original samples themselves. Mixup augmentations are constructed as follows:

$$\tilde{x} = \lambda x_i + (1 - \lambda) x_j \quad (4)$$

$$\tilde{y} = \lambda y_i + (1 - \lambda) y_j, \quad (5)$$

where (x_i, y_i) and (x_j, y_j) are independently sampled from the empirical distribution of the observed dataset, $\lambda \sim \text{Beta}(\alpha, \alpha)$, and α is a hyperparameter controlling the mixing of the samples. Empirically, Mixup has been shown to improve the test performance of deep learning models. Recent theoretical works attribute this improvement in performance to the fact Mixup is a form of implicit regularization [12]. An important extension in Mixup, called Manifold Mixup, performs the Mixup operation in an intermediate layer of the deep learning model and in some cases can be shown to improve on the performance of the original Mixup algorithm. Unfortunately, Manifold Mixup does not guarantee that augmentations are synthesized on a manifold with nice properties (e.g, preservation of both local and global structure).

3.2. UMAP

UMAP is a nonlinear dimensionality reduction technique that aims to learn a lower-dimensional embedding of the data that preserves its topological structure. A strong advantage of UMAP versus other dimensionality reduction techniques, such as T-SNE [13], is its ability to capture both local and global structure in the learned embeddings.

UMAP makes three assumptions of the data: the data is assumed to be uniformly distributed on a Riemannian manifold; the Riemannian metric is assumed to be locally constant; and the manifold is assumed to be locally connected. Based on these assumptions, one can optimize a set of embeddings $\{z_i \in \mathbb{R}^{d_z}\}_{i=1}^N$, where each z_i corresponds to the embedding of the data point x_i . The basic procedure has two steps: First, a topological representation of the data is constructed using fuzzy simplicial sets, and second, the embeddings are optimized so that their topological representation has the closest fuzzy topological structure to that of the representation in data space. Computationally, this amounts to minimizing the cross entropy between two graph representations: P (data graph) and Q (embedding graph). Nonparametric UMAP optimizes the embeddings directly, whereas parametric UMAP optimizes the parameters of some function approximator (e.g., a neural network) that outputs the embeddings [14]. Since, this work focuses on embeddings obtained from neural networks, for brevity, we only highlight the parametric UMAP procedure.

The data graph P , whose elements are denoted by $p_{i,j}$, is constructed by computing directional probabilities between each data-point and its K -nearest neighbors given by:

$$p_{j|i} = \exp\left(\frac{-(d(x_i, x_j) - \rho_i)}{\sigma_i}\right), \quad x_j \in \mathcal{N}(x_i) \quad (6)$$

where $d(x_i, x_j)$ is a distance function, ρ_i and σ_i are local connectivity parameters, and $\mathcal{N}(x_i)$ is the local neighborhood of x_i . The local connectivity parameters and the local neighborhood are determined by the K number of neighbors considered, which is a hyperparameter in the UMAP algorithm. Once the directional probabilities $p_{j|i}$ are determined, the probabilities are symmetrized

$$p_{i,j} = (p_{j|i} + p_{i|j}) - p_{j|i} p_{i|j} \quad (7)$$

Let $z_i = h_\phi(x_i)$ denote the embedding of data point $x_i \in \mathbb{R}^{d_x}$ obtain from some parametric function $h_\phi: \mathbb{R}^{d_x} \rightarrow \mathbb{R}^{d_z}$ with $\phi \in \Phi$. The embedding graph Q_ϕ , whose elements edges are denoted by $q_{i,j}$, is constructed by computing:

$$q_{i,j} = (1 + a \|z_i - z_j\|^{2b})^{-1}, \quad (8)$$

where a and b are hyperparameters that impact the minimum distance between the embeddings. Given the data graph P , the UMAP algorithm optimizes the embeddings such that the corresponding embedding graph Q_ϕ is the one that minimizes the cross-entropy loss between P and Q_ϕ , which is defined as

$$C(P, Q_\phi) = \sum_{i \neq j} p_{i,j} \log\left(\frac{p_{i,j}}{q_{i,j}}\right) + (1 - p_{i,j}) \log\left(\frac{1 - p_{i,j}}{1 - q_{i,j}}\right) \quad (9)$$

In parametric UMAP, optimization of (9) amounts to solving:

$$\hat{\phi} = \arg \min_{\phi \in \Phi} C(P, Q_\phi), \quad (10)$$

which can be accomplished using stochastic optimization.

4. METHODOLOGY

In this work, we propose a novel variant of Mixup that joins manifold Mixup with parametric UMAP, which we call *UMAP Mixup*. The idea is as follows: we apply the Mixup operation in an intermediate layer of the classifier that is optimized to have similar topological

structure to the original data using the UMAP loss as a regularizer.

4.1. Supervised Parametric UMAP

We consider predictive models of the following form:

$$y = f_\theta(x) = g_{\theta_2}(h_{\theta_1}(x)), \quad (11)$$

where $h_{\theta_1} : \mathbb{R}^{d_x} \rightarrow \mathbb{R}^{d_z}$ is a neural network mapping from features to a d_z -dimensional embedding and $g_{\theta_2} : \mathbb{R}^{d_z} \rightarrow \mathcal{Y}$ is another neural network mapping the embeddings to predictions. The parameters $\theta = \{\theta_1, \theta_2\}$ are learned in a semi-supervised fashion using a loss $\ell(f_\theta(x), y)$ (supervised loss) and the UMAP loss as defined in (9) (unsupervised loss). This is done by adding a regularizer to the risk:

$$R_{\text{reg}}(\theta; \gamma) = \mathbb{E}[\ell(f_\theta(x), y)] + \gamma C(P, Q_{\theta_1}). \quad (12)$$

where Q_{θ_1} is the parametric UMAP embedding graph as determined by the neural network h_{θ_1} and $\gamma \geq 0$ is a regularization parameter that controls the influence of the UMAP loss. The first term can be approximated using the empirical distribution of the data, just as it is done in ERM, while the second term can be approximated as by sampling edges of the data graph P . Specifically, for a given dataset \mathcal{D} , the *supervised parametric UMAP loss function* is given by:

$$\widehat{R}_{\text{reg}}(\theta; \gamma) = \left(\frac{1}{N} \sum_{i=1}^N \ell_\theta(f_\theta(x_i), y_i) \right) + \gamma C(P, Q_{\theta_1}), \quad (13)$$

In practice, the parameters θ_1 and θ_2 can be learned jointly using stochastic optimization algorithms, such as stochastic gradient descent or Adam. In particular, each term in the loss in (13) is approximated by mini-batching, where the supervised component of the loss is approximated by mini-batching over the dataset \mathcal{D} and the UMAP regularizer is approximated by mini-batching over the edges of the data graph P , as described in [11] and [14]. Please see Subsection 4.3 for more details on how we efficiently mini-batch in this work.

4.2. UMAP Mixup

Our proposed method UMAP Mixup is an extension of supervised parametric UMAP that constructs augmentations by applying Mixup to a parametric UMAP embedding. During training, a single forward pass of our algorithm can be summarized in five steps:

1. Generate a sample $(x, y) \sim \mathcal{D}$, where \mathcal{D} denotes the empirical distribution of the data.
2. Generate a neighboring sample (x', y') based on the weighted data graph P . Note: the first two steps can be summarized into a single step by simply sampling a random edge from P .
3. Generate a mixing ratio $\lambda \sim \text{Beta}(\alpha, \alpha)$.
4. Set $z = h_{\theta_1}(x)$ and $z' = h_{\theta_1}(x')$. Generate an interpolated embedding using the samples (x, y) and (x', y') :

$$\tilde{z} = \lambda z + (1 - \lambda)z'. \quad (14)$$

5. Obtain the interpolated prediction as $\tilde{y} = g_{\theta_2}(\tilde{z})$.

The loss for the generated interpolated predictions is evaluated against an interpolation of the ground truth labels y and y' :

$$\ell_\theta^{\text{mix}}(x, x', y, y') = \ell_\theta(\tilde{y}, \lambda y + (1 - \lambda)y') \quad (15)$$

In UMAP Mixup, the expected value of (15), augmented with the UMAP loss $C(P, Q_{\theta_1})$ is optimized in order to learn the model parameters:

$$\widehat{\theta}_{\text{UMAP}}^{\text{mix}} = \arg \min_{\theta \in \Theta} \mathbb{E}[\ell_\theta^{\text{mix}}(x, x', y, y')] + \gamma C(P, Q_{\theta_1}), \quad (16)$$

where α and γ are hyperparameters. In practice, these can be selected using cross-validation, where the choices vary from dataset to dataset. Empirically, we have found that choices of $\alpha = 2$ and $\gamma = 0.1$ lead to favorable results in practice for most datasets.

4.3. Implementation Details

Typical practice for creating batches consists of randomly shuffling the training dataset each epoch and iterating over subsets, effectively sampling without replacement. However, the cross entropy criterion in UMAP is defined over the set of edges of P . Our approach consists of creating mini-batches over the positive ($p_{i,j} > 0$) and negative ($p_{i,j} = 0$) edges, with the vertices that make up the batched positive edges then comprising the minibatch for the supervised loss.

Let $E^+ = \{e_{i,j} = (i, j)\}$ be the set of positive edges in a training epoch, where each edge $e_{i,j}$ is included in E^+ with probability $p_{i,j}$. For each positive edge $e_{i,j}$, we also associate M negative edges $\{e_{i,j_1}, \dots, e_{i,j_M}\}$ by sampling j_1, \dots, j_M uniformly from the dataset, giving the set of negative edges in a training epoch $E^- = \{e_{i,j_1}, \dots, e_{i,j_M} : e_{i,j} \in E^+\}$. Each training epoch of UMAP Mixup therefore consists of iterating over subsets $E_b = E_b^+ \cup E_b^-$, where $E_b^+ \subset E^+$, $E_b^- \subset E^-$. This gives a batch UMAP loss of

$$C(P, Q_{\theta_1}) \approx \widehat{C}(P, Q_{\theta_1}, E_b) \quad (17)$$

$$= \frac{1}{|E_b|} \left(\sum_{e_{i,j} \in E_b^+} \log \left(\frac{p_{i,j}}{q_{i,j}} \right) + \sum_{e_{k,l} \in E_b^-} \log \left(\frac{1 - p_{k,l}}{1 - q_{k,l}} \right) \right) \quad (18)$$

The mini-batched supervised loss is then defined over the set of data points that occur as vertices of the positive edges in the batch:

$$\mathbb{E}[\ell_\theta^{\text{mix}}(x, x', y, y')] \approx \widehat{\mathbb{E}}[\ell_\theta^{\text{mix}}(x, x', y, y')] \quad (19)$$

$$= \frac{1}{|E_b^+|} \sum_{e_{i,j} \in E_b^+} \ell_\theta^{\text{mix}}(x_i, x_j, y_i, y_j). \quad (20)$$

5. EXPERIMENTS

We validate UMAP Mixup on regression tasks covering two different data modalities: tabular data and time series data. All results are summarized in Table 1, which measure performance using root-mean squared error (RMSE).

5.1. Tabular Data

We evaluate the performance of UMAP Mixup on a set of UCI regression benchmark datasets: [15] (a) Boston Housing dataset¹, (b) Concrete compressive strength dataset² [16], and (c) Yacht hydrodynamics dataset³. We use the experimental setup used in [17], with

¹<https://www.kaggle.com/datasets/schirmerchad/bostonhousingm1nd>

²<https://archive.ics.uci.edu/ml/datasets/concrete+compressive+strength>

³<https://archive.ics.uci.edu/ml/datasets/Yacht+Hydrodynamics>

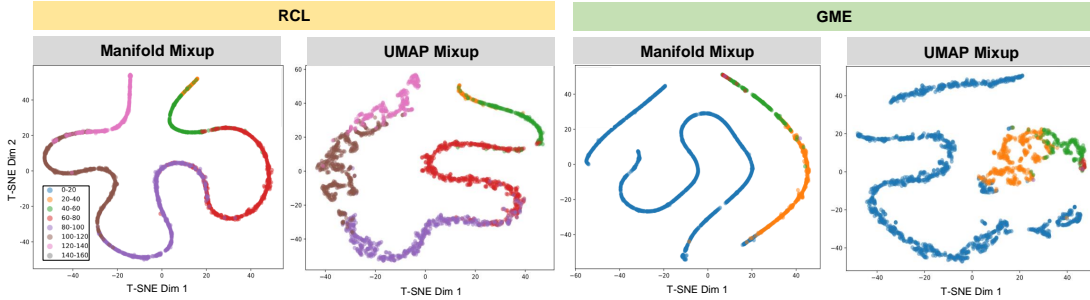


Fig. 1. A visual comparison of resulting embeddings from both Manifold Mixup and UMAP Mixup regularizations on the RCL and GME datasets. Visualizations are obtained by applying T-SNE to the extracted features just before the output layer of each neural network.

Table 1. Experiment results.

Dataset	ERM	Mixup	Manifold Mixup	UMAP Mixup
<i>Tabular (Regression)</i>				
Boston Housing	3.14 ± 0.67	3.01 ± 0.71	3.10 ± 0.76	3.27 ± 0.66
Concrete	5.11 ± 0.59	5.92 ± 0.55	5.08 ± 0.62	4.83 ± 0.79
Yacht	0.91 ± 0.34	4.19 ± 0.63	0.80 ± 0.24	0.71 ± 0.21
<i>Time series (Regression)</i>				
GOOG	2.47 ± 0.05	2.47 ± 0.03	2.50 ± 0.03	2.43 ± 0.04
RCL	4.74 ± 0.69	4.07 ± 0.43	4.30 ± 0.60	3.13 ± 0.61
GME	3.66 ± 0.33	2.77 ± 0.49	3.83 ± 0.47	2.73 ± 0.37

each dataset split into 20 train-test folds. For each method, we utilize a (100, 50) 2-layer feedforward neural network. We train each method using the Adam optimizer, where we select learning rates and batch sizes according to values utilized in [18]. For each baseline, we use cross-validation to select all other hyperparameters. The results in Table 1 indicate that for the Concrete and Yacht datasets, UMAP Mixup is able to get significant reduction in RMSE as compared to the baselines.

5.2. Time Series Data

For the time series data, we focus on the task of one-step-ahead forecasting for financial time series. We use historical daily price data from Yahoo finance and follow a similar experimental set up as [17], using a long short-term memory (LSTM) network [19]. The input to the LSTM corresponds to the closing price of a particular stock over the past 60 trading days and the target output is the next trading day’s closing price. We evaluate each method on three different datasets:

- **GOOG - stable market regime:** We use training data from the Google (GOOG) stock from the period of Jan 2015 - July 2022 and test on GOOG stock data from the period of August 2022 - September 2023.
- **RCL - market shock regime:** We use training data from the Royal Caribbean (RCL) stock from the period of Jan 2015 - June 2020 and test on RCL stock data from the period of July 2020 - September 2023. This choice of dataset is motivated by the COVID shock that heavily affected the Royal Caribbean stock when the global pandemic caused widespread closures in the spring of 2020,
- **GME - high volatility regime:** We use the training data from the Gamestop (GME) stock to include the short squeeze period of January 2015 - Jan 2022 and test on GME stock data following that period.

The results are shown in Table 1, where UMAP Mixup outperforms all baselines. In particular, Manifold Mixup shows worst performance than the other methods, indicating that interpolation in the embedding space without further regularization does not necessarily improve generalization. Figure 1 shows a qualitative comparison of the embeddings of Manifold Mixup and our method, Manifold UMAP for the RCL and GME datasets by using T-SNE to project the extracted features on the last hidden layer of each neural network. We can observe that Manifold Mixup shows a very compact embedding whilst the UMAP Mixup embeddings show more variability and therefore, interpolation between them would yield more diverse samples. This is particularly relevant in these datasets that suffer from distributional shifts such as a market shock regime in the case of RCL due to COVID, and the high volatility regime of GME during the “bubble” period in early 2021.

6. CONCLUSION

This work proposes a novel variant of Mixup regularization called UMAP Mixup. UMAP Mixup hybridizes a nonlinear dimensionality reduction technique called UMAP with Mixup regularization. Unlike other variants of Mixup, such as Manifold Mixup, UMAP Mixup synthesizes augmentations in a hidden layer of neural network that is regularized to have nice topological properties: mainly, that the optimized manifold in UMAP Mixup better preserves global and local structure as compared to its counterparts. Our results on empirical evaluations across three different data modalities (tabular, image, and time series data) show that UMAP Mixup performs well in comparison to multiple state-of-the-art baselines.

7. ACKNOWLEDGEMENTS

This paper was prepared for informational purposes by the Artificial Intelligence Research group of JPMorgan Chase & Co. and its affiliates (“JP Morgan”), and is not a product of the Research Department of JP Morgan. JP Morgan makes no representation and warranty whatsoever and disclaims all liability, for the completeness, accuracy or reliability of the information contained herein. This document is not intended as investment research or investment advice, or a recommendation, offer or solicitation for the purchase or sale of any security, financial instrument, financial product or service, or to be used in any way for evaluating the merits of participating in any transaction, and shall not constitute a solicitation under any jurisdiction or to any person, if such solicitation under such jurisdiction or to such person would be unlawful.

8. REFERENCES

- [1] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019.
- [2] E. Fons, P. Dawson, X.-j. Zeng, J. Keane, and A. Iosifidis, "Evaluating data augmentation for financial time series classification," *arXiv preprint arXiv:2010.15111*, 2020.
- [3] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.
- [4] O. Chapelle, J. Weston, L. Bottou, and V. Vapnik, "Vicinal risk minimization," *Advances in neural information processing systems*, vol. 13, 2000.
- [5] V. N. Vapnik, "An overview of statistical learning theory," *IEEE transactions on neural networks*, vol. 10, no. 5, pp. 988–999, 1999.
- [6] V. Verma, A. Lamb, C. Beckham, A. Najafi, I. Mitliagkas, D. Lopez-Paz, and Y. Bengio, "Manifold mixup: Better representations by interpolating hidden states," in *International conference on machine learning*. PMLR, 2019, pp. 6438–6447.
- [7] H.-P. Chou, S.-C. Chang, J.-Y. Pan, W. Wei, and D.-C. Juan, "Remix: rebalanced mixup," in *Computer Vision—ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*. Springer, 2020, pp. 95–110.
- [8] R. Baena, L. Drumetz, and V. Gripon, "Preventing manifold intrusion with locality: Local mixup," *arXiv preprint arXiv:2201.04368*, 2022.
- [9] H. Yao, Y. Wang, L. Zhang, J. Y. Zou, and C. Finn, "C-mixup: Improving generalization in regression," *Advances in Neural Information Processing Systems*, vol. 35, pp. 3361–3376, 2022.
- [10] H. Guo, Y. Mao, and R. Zhang, "Mixup as locally linear out-of-manifold regularization," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 3714–3722.
- [11] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.
- [12] Y. Zou, V. Verma, S. Mittal, W. H. Tang, H. Pham, J. Kannala, Y. Bengio, A. Solin, and K. Kawaguchi, "Mixupe: Understanding and improving mixup from directional derivative perspective," in *Uncertainty in Artificial Intelligence*. PMLR, 2023, pp. 2597–2607.
- [13] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.
- [14] T. Sainburg, L. McInnes, and T. Q. Gentner, "Parametric umap embeddings for representation and semisupervised learning," *Neural Computation*, vol. 33, no. 11, pp. 2881–2907, 2021.
- [15] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: <http://archive.ics.uci.edu/ml>
- [16] I.-C. Yeh, "Modeling of strength of high-performance concrete using artificial neural networks," *Cement and Concrete research*, vol. 28, no. 12, pp. 1797–1808, 1998.
- [17] Y. El-Laham, N. Dalmaso, E. Fons, and S. Vyetrenko, "Deep gaussian mixture ensembles," in *The 39th Conference on Uncertainty in Artificial Intelligence*, 2023. [Online]. Available: https://openreview.net/forum?id=_Qwp7ATla6P
- [18] U. Sarawgi, W. Zulfikar, R. Khincha, and P. Maes, "Why have a unified predictive uncertainty? disentangling it using deep split ensembles," *arXiv e-prints*, pp. arXiv–2009, 2020.
- [19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.