NoisyMix: Boosting Model Robustness to Common Corruptions

N. Benjamin Erichson
ICSI, LBNL

Soon Hoe Lim
Nordita, KTH, Stockholm University

Winnie Xu
University of Toronto

Francisco Utera
ICSI

Ziang Cao
Stanford University

Michael W. Mahoney
ICSI, UC Berkeley, LBNL

Abstract

The robustness of neural networks has become increasingly important in real-world applications where stable and reliable performance is valued over simply achieving high predictive accuracy. To address this, data augmentation techniques have been shown to improve robustness against input perturbations and domain shifts. In this paper, we propose a new training scheme called *NoisyMix* that leverages noisy augmentations in both input and feature space to improve model robustness and in-domain accuracy. We demonstrate the effectiveness of *NoisyMix* on several benchmark datasets, including ImageNet-C, ImageNet-R, and ImageNet-P. Additionally, we provide theoretical analysis to better understand the implicit regularization and robustness properties of *NoisyMix*.

1 INTRODUCTION

While deep learning has achieved remarkable results in computer vision and natural language processing, much of the success is driven by improving test accuracy. However, it is well-known that deep learning models are typically brittle and sensitive in noisy and adversarial environments. This limits their application in many real-world settings which require, at a minimum, deep learning methods produce stable statistical predictions. One can distinguish between structural stability (i.e., how sensitive is a model's prediction is to small perturbations of the weights, or model parameters) and input stability (i.e., how sensitive a model's prediction is to small perturbations of the input data). Structural stability is particularly important when compression techniques (e.g., quantization) or different hardware systems introduce small errors to the model's weights. Input stability is important when models (e.g., a vision model in a self-driving car) should operate reliably in noisy environments. Overall, robustness is an important topic within machine learning and deep learning, but the many facets of robustness prohibit the use of a single metric for measuring it (Hendrycks et al., 2021).

In this work, we are specifically interested in studying input stability (robustness) with respect to common data corruptions and domain shifts that naturally occur in many real-world settings. To do so, we use datasets such as ImageNet-C (Hendrycks and Dietterich, 2018) and ImageNet-R (Hendrycks et al., 2021). ImageNet-C provides examples to test the robustness of a model with respect to common corruptions, including noise

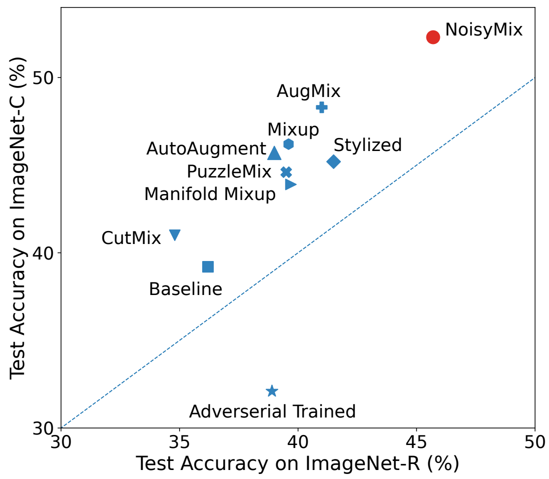


Figure 1: ResNet-50 models trained with data augmentations. *NoisyMix* improves the test accuracy on ImageNet-C and ImageNet-R, indicating improved robustness to common corruptions and domain shifts.

Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238. Copyright 2024 by the author(s).

sources such as white noise, weather variations such as snow, and digital distortions such as image compression. ImageNet-R provides examples to test robustness to naturally occurring domain shifts in the form of abstract visual renditions, including graffiti, origami, or cartoons. State-of-the-art models often fail when facing such common corruptions and domain shifts. For instance, the predictive accuracy of a ResNet-50 (He et al., 2016b) trained on ImageNet drops by over 35%, when evaluated on ImageNet-C.

Data augmentation methods such as Mixup (Zhang et al., 2018), AutoAugment (Cubuk et al., 2019), training on stylized ImageNet (Geirhos et al., 2018), and AugMix (Hendrycks et al., 2020) have the ability to improve the resilience to various data perturbations and domain shifts. Figure 1 shows the test accuracies on ImageNet-C, and ImageNet-R for ResNet-50 models trained with different augmentation schemes on ImageNet (see Sec. 4). Clearly, data augmentation helps to achieve better test accuracies on these corrupted and domain-shifted test sets.

Among these, our proposed method, *NoisyMix*, is the most effective in improving robustness. In contrast, adversarial training (Shafahi et al., 2019a) decreases robustness to common corruptions. The reason for this is that while adversarial training helps to improve robustness to perturbations that are manifested in the high-frequency domain, it reduces robustness to common corruptions that are often manifested in the low-frequency domain (Yin et al., 2019).

Contributions. Our main contributions in this paper are summarized as follows.

- We formulate and study a novel, theoretically justified training scheme, *NoisyMix*, that substantially outperforms standard stability training and data augmentation schemes in improving robustness to common corruptions and in-domain test accuracy.
- We study *NoisyMix* via the lens of implicit regularization and provide theoretical results that identify the effects (see Theorem 1-2 in App. B). Importantly, we show that, under assumptions similar to those made in Lamb et al. (2019); Lim et al. (2022), minimizing the *NoisyMix* loss corresponds to minimizing an upper bound on the sum of an adversarial loss, a stability objective, and data-dependent regularizers (see Theorem 3 in App. C). This suggests that *NoisyMix* provides an inductive bias towards improving model robustness with respect to common corruptions, when compared to standard training.
- We provide empirical results to show that *NoisyMix* improves the robust accuracy on ImageNet-C and ImageNet-R by over 4% compared to AugMix, while achieving comparable accuracy on the clean ImageNet validation set. Moreover, *NoisyMix* has comparable training costs to AugMix. We also show that *NoisyMix* achieves state-of-the-art performance on CIFAR-10-C and CIFAR-100-C, using preactivated ResNet-18 and Wide-ResNet-28x2 architectures. Our empirical results further suggest that the improved effects of *NoisyMix* come from implicit regularization. To understand this better, we illustrate these effects via toy experiments (see App. A)

Notation. $[K] := \{1, \ldots, K\}$, $\odot$ denotes Hadamard product, $\mathbb{1}$ is the vector with all elements equal one. For a vector v, $\|v\|_p$ denotes its l_p norm for $p > 0$. $M_\lambda(a, b) := \lambda a + (1 - \lambda)b$, for random variables a, b, λ.

2 RELATED WORK

The stability of modeling methods for real-world phenomena has a long been studied and foundational to statistics, numerical analysis, and dynamical systems (Yu, 2013). The field of robust statistics is concerned with perturbations to model parameters (Huber, 2004), while other branches of statistics are concerned with the perturbation at the input level (Efron, 1992). The importance of developing robust methods for scientific discovery is at hand, and thus it is no surprise that robustness is an emerging research area within the deep learning community as well. In this work, we are mainly concerned with improving input stability of deep neural networks for computer vision tasks. In particular, there are two types of input perturbations that have recently led to a surge of interest: (i) adversarial perturbations, and (ii) common corruptions. The class of adversarial perturbations consists of artificially crafted perturbations specifically designed to fool neural networks while being imperceptible for the human eye (Szegedy et al., 2013; Goodfellow et al., 2014). There has been a tremendous effort on designing attack and defense methods (see Chakraborty et al. (2018) for a recent overview), and an interesting result of this line of work is that adversarially-trained models (Madry et al., 2018) learn features that are more aligned with human perception (Ilyas et al., 2019). However, it can be argued that adversarial perturbation are a hypothetical thread in real-world applications. Moreover, adversarial robust models turn out to be less accurate on clean data as well as on common courruptions (Yin et al., 2019; Rusak et al., 2020). Common corruptions are a class of perturbations that naturally occur in real-world applications (Hendrycks and Dietterich, 2018), and are often more relevant in real-world science and engineering applications than adversarial perturbations (Rusak et al., 2020).

Early work by Dodge and Karam (2016) has revealed how sensitive neural networks (trained on high quality images) are to slight degradation of image quality and common corruptions. Hendrycks and Dietterich (2018) introduced a more extensive benchmark dataset for testing robustness on a comprehensive set of common corruptions with varying intensity levels. Subsequently, it has been shown that data augmentation and noise injection schemes can help to improve robustness (Yin et al., 2019) to common corruptions. In contrast, Taori et al. (2020) criticize that most work on robustness focuses on synthetic image perturbations and they instead suggest that it is required to train models on larger and more diverse datasets in order to improve robustness to distribution shifts that are arising from natural variations in datasets. As an alternative, Hendrycks et al. (2019) and Xie et al. (2020) showed that pretraining on larger and more diverse datasets (e.g., JFT-300M (Hinton et al., 2015)) can help to improve robustness, but Hendrycks et al. (2021) showed that these strategies do not consistently do better if a broader range of distribution shifts is considered. The disadvantages of pretraining on larger datasets are the increased computational costs and the limited availability of massive computer vision datasets. Hence, we limit our discussion in this paper to data augmentation and noise injection strategies that can be applied to standard architectures trained on common computer vision benchmark datasets.

Most relevant for our work are data augmentation methods that aim to reduce the generalization error and improve robustness by introducing implicit regularization (Kukačka et al., 2017) into the training process. By data augmentation, we refer to schemes that do not introduce new data but train a model with virtual training examples (e.g., proper transformations of the original data) in the vicinity of a given training dataset. Basic examples are random cropping and horizontal flipping of input images (Krizhevsky et al., 2012).

Mixup (Zhang et al., 2018) is a popular data augmentation method, which creates new training examples by forming linear interpolations between random pairs of examples and their corresponding labels. It is a simple but extremely efficient scheme that can improve the generalization and robustness of models with minimal computational overhead. There are added advantages in not introducing additional overhead during training or requiring modifications on the network architecture. Various other proposed methods have since been inspired by Mixup. Manifold Mixup (Verma et al., 2019) extends the idea of interpolating between data points in input space to hidden representations, leading to models that have smoother decision boundaries. However, our experiments show that the advantages compared to Mixup are mixed. Noisy Feature Mixup (NFM) (Lim et al., 2022) generalizes the idea of both Mixup and Manifold Mixup by taking linear combinations of pairs of perturbed input and hidden representations. Another approach involves zeroing out parts of an input image (DeVries and Taylor, 2017) to prevent a model from focusing on a limited image region. CutMix (Yun et al., 2019) extends this idea by replacing the removed parts with a patch from another image. This improves the test accuracy, but it does not help much to improve the robustness to common corruptions. PuzzleMix (Kim et al., 2020) is a mixup method that further improves CutMix by leveraging saliency information and local statistics of the original input data to decide how to mix two data points.

Geirhos et al. (2018) proposed to train models on a stylized version of ImageNet, which helps to increase shape bias and in turn improves robustness to common corruptions. To do so, they use a style transfer network that applies artwork styles to images that are then used to train the model. Similar good results are achieved by AutoAugment (Cubuk et al., 2019), a method that automatically searches for improved data augmentation policies. AugMix (Hendrycks et al., 2020) is designed to improve robustness to common corruptions, which is achieved by leveraging a diverse set of basic data augmentations (e.g., translate, posterize, solarize) and applying consistency regularization (using the Jensen-Shannon divergence loss). This approach improves robustness when compared to the previously discussed data augmentation methods. Recently, Wang et al. (2021) proposed AugMax, a strong data augmentation scheme, that learns an adversarial mixture of several sampled augmentation operators.

Noise injections are another form of data augmentations, which introduce regularization into the training process (Bishop, 1995). Most commonly, the models are trained with noise perturbed inputs (An, 1996). Noise can also be injected into the activation functions (Gulcehre et al., 2016), or the hidden layers of a neural network (Camuto et al., 2020; Lim et al., 2022). Recently, Rusak et al. (2020) demonstrated that training a model with various noise types helps to improve robustness to common corruptions.

3 METHOD

At a high level, *NoisyMix* implements feature mixup and noise injections (Lim et al., 2022) on top of a stability training scheme (Zheng et al., 2016) on an augmented data set (Hendrycks et al., 2020) in a single setup to boost both accuracy and robustness. While components underlying *NoisyMix* exist in the literature, the devil is in the details to get the method right, so we motivate and describe the procedures of *NoisyMix* for multi-class classification in the following.

First, we augment the clean data with appropriately transformed data, obtained by applying the operations used in AugMix to the clean data. The transformed data are diverse because the operations are sampled stochastically and then layered on the same input image. Such diverse transformations are crucial for enabling model robustness by discouraging the model from memorizing fixed augmentations (Hendrycks et al., 2020).

To improve the robustness further, we judiciously combine the stability training scheme with mixup and noise injections at the level of input and hidden layers. The latter approach, NFM (Lim et al., 2022), generalizes input mixup (Zhang et al., 2018) and Manifold Mixup (Verma et al., 2019) in the sense that noise is injected when taking convex combinations of pairs of input and hidden layer features. It has been shown that this makes the decision boundary smoother and helps improving robustness when predicting on out-of-distribution data (Lim et al., 2022).

Applying the procedures of NFM on top of AugMix, and using stability training, leads to our proposed method, *NoisyMix*, which we describe in detail next.

3.1 Detailed Formulation of *NoisyMix*

We consider multi-class classification with K labels. Denote the input space by $\mathcal{X} \subset \mathbb{R}^d$ and the output space by $\mathcal{Y} := \Delta^{K-1}$, where Δ^{K-1} is the probability simplex, i.e., a categorical distribution over K classes. The classifier, g, is constructed from a learnable deep neural network map $p : \mathcal{X} \to \Delta^{K-1}$, mapping an input x to its label, $g(x) = \arg\max_k p^k(x) \in [K]$. We are given a training set, $\mathcal{Z}_n := \{(x_i, y_i)\}_{i\in[n]} \cup \{(A(x_i), y_i)\}_{i\in[n]}$, consisting of $2n$ pairs of input and one-hot label, with each training pair $z_i := (x_i, y_i), z_i' := (A(x_i), y_i) \in \mathcal{X} \times \mathcal{Y}$. The training pairs (x_i, y_i) are drawn independently from a ground-truth distribution $\mathcal{D} := \mathcal{D}_x \times \mathcal{D}_y$. Here, A denotes the random perturbation of an input, i.e., the *AugmentAndMix* function in Hendrycks et al. (2020). In the sequel, we denote $\mathcal{A}(x)$ as the underlying conditional (on x) distribution from which $A(x)$ is sampled.

More precisely, for an (original) input x, we construct a virtual data point x_{am} as

$$x_{am} = A(x) := mx + (1-m)\sum_{i=i}^{3} w_i C(x) \sim \mathcal{A}(x), \quad (1)$$

where $m \sim Beta(\alpha, \alpha)$, the $w_i \sim Dirichlet(\alpha, \ldots, \alpha)$ (with $\alpha := 1$), and C uniformly drawn from a set of operations whose elements including compositions of transformations such as rotating, translating, autocontrasting, equalizing, posterizing, solarizing, and shearing. They can be viewed as randomly perturbed input data, i.e., $A(x) = x + \delta x$, where the random perturbation $\delta x := (1-m)\left(\sum_{i=1}^{3} w_i C(x) - x\right)$ is additive and data-dependent. The transformed data are highly diverse because the operations are sampled stochastically and then layered on the same input image. Such diverse transformations are crucial for enabling model robustness by discouraging the model from memorizing fixed augmentations (Hendrycks et al., 2020).

Within this setting, we propose the *NoisyMix* loss:

$$L^{NoisyMix} = L^{NFM} + \gamma L^{stability}, \quad (2)$$

where $\gamma \geq 0$. This loss is a sum of two components: the expected NFM loss (Lim et al., 2022) and an expected stability loss (Zheng et al., 2016). As in (Lim et al., 2022), the expected NFM loss is given by:

$$L^{NFM} = \mathbb{E}_{(x,y),(x',y')\sim\mathcal{D}}\mathbb{E}_{\lambda\sim Beta(\alpha,\beta)} \mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{Q}}[l(p(M_{\lambda,\boldsymbol{\xi}}(x,x')), M_\lambda(y,y'))],$$

for some loss function (e.g., cross-entropy) $l : \Delta^{K-1} \times \Delta^{K-1} \to [0,\infty)$ (note the dependence of both l and p on the learnable parameter θ is suppressed in the notation), $\boldsymbol{\xi} := (\xi^{add}, \xi^{mult})$ are drawn from some probability distribution $\mathcal{Q}$ with finite first two moments, and

$$M_{\lambda,\boldsymbol{\xi}}(x,x') = (\mathbb{1} + \sigma_1\xi^{mult}) \odot M_\lambda(x,x') + \sigma_2\xi^{add},$$

where σ_1 and σ_2 are tuning parameters. In other words, this approach combines stability training with mixup and noise injections at the level of input and hidden layers. In App. A we illustrate that the combination of these techniques introduce implicit regularization that makes the decision boundary smoother. In turn, this helps improving robustness when predicting on out-of-distribution data.

We choose the distance measure D in the expected stability loss to be the Jensen-Shannon divergence (JSD) and minimize the deviation between the distribution on clean input data and that on the transformed input data, i.e., we minimize $L^{stability} := L^{JSD}$, given by:

$$\mathbb{E}_{x,x'\sim\mathcal{D}_x}\mathbb{E}_{x_{am}\sim\mathcal{A}(x),x'_{am}\sim\mathcal{A}(x')}\mathbb{E}_{\lambda\sim Beta(\alpha,\beta)} \mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{Q}} JS_\pi(p(M_{\lambda,\boldsymbol{\xi}}(x,x')), p(M_{\lambda,\boldsymbol{\xi}}(x_{am},x'_{am}))),$$

where JS_π denotes Jensen-Shannon divergence with the weights π, defined as follows: for $p_1, p_2 \in \Delta^{K-1}$ with the corresponding weights $\pi := (\pi_1, \pi_2) \in \Delta$, the JSD (Lin, 1991) between p_1 and p_2 is

$$JS_\pi(p_1,p_2) := H(\pi_1 p_1 + \pi_2 p_2) - \pi_1 H(p_1) - \pi_2 H(p_2),$$

with H the Shannon entropy, defined as $H(p) := -\sum_{k=1}^{K} p_k \ln(p_k)$ for $p \in \Delta^{K-1}$. Alternatively, $JS_\pi(p_1,p_2) = \pi_1 KL(p_1 \| \pi_1 p_1 + \pi_2 p_2) +$

$\pi_2 KL(p_2 \| \pi_1 p_1 + \pi_2 p_2)$, where KL denotes the Kullback–Leibler divergence.

JSD measures the similarity between two probability distributions, and its square root is a true metric between distributions. Unlike the KL divergence and cross-entropy, it is symmetric, bounded, and does not require absolute continuity of the involved probability distributions. Similar to the KL divergence, $JS_\pi(p_1, p_2) \geq 0$, with equality if and only if $p_1 = p_2$. This follows from applying Jensen's inequality to the concave Shannon entropy. Moreover, JSD can interpolate between cross-entropy and mean absolute error for $\pi_1 \in (0,1)$ (see Proposition 1 in Englesson and Azizpour (2021)). In GANs, JSD is used as a measure to quantify the similarity between the generative data distribution and the real data distribution (Goodfellow et al., 2020; Weng, 2019). We refer to Lin (1991); Englesson and Azizpour (2021) for more properties of JSD. Note that, unlike L^{NFM}, L^{JSD} is computed without the use of the labels.

A Note on Stability Training. We consider a stability training scheme on the augmented training data (Zheng et al., 2016). Stability training stabilizes the output of a model against small perturbations to input data, thereby enforcing robustness. Since the classifiers are probabilistic in nature, we consider a probabilistic notion of robustness, which amounts to making them locally Lipschitz with respect to some distance on the input and output space. This ensures that a small perturbation in the input will not lead to large changes (as measured by some statistical distance) in the output.

We now formalize a notion of robustness. Let $p > 0$. We say that a model $f : \mathcal{X} \to \mathcal{Y}$ is *α_p-robust* if for any $(x, y) \sim \mathcal{D}$ such that $f(x) = y$, one has, for any data perturbation $\delta x \in \mathcal{X}$,

$$\|\delta x\|_p \leq \alpha_p \implies f(x) = f(x + \delta x).$$

An analogous definition can be given for distribution-valued outputs, which leads to a notion of probabilistic robustness. Let D be a metric or divergence between two probability distributions. We say that a model $F : \mathcal{X} \to \mathcal{P}(\mathcal{Y})$, i.e., the space of probability distributions on $\mathcal{Y}$, is *(α_p, ϵ)-robust* with respect to D if, for any $x, \delta x \in \mathcal{X}$,

$$\|\delta x\|_p \leq \alpha_p \implies D(F(x), F(x + \delta x)) \leq \epsilon.$$

In practice, stability training involves formulating a loss that aims to flatten the model output in a small neighborhood of any input data, forcing the output to be similar between the original and perturbed copy. For instance, AugMix incorporates such stability training in the following manner. In contrast to the standard training scheme, one does not simply train with the cross-entropy loss, but instead train with a sum of the cross-entropy (on the clean data) and a suitable stability objective. This encourages model prediction to be more constant around the input data while mitigating underfitting. This is done by combining the cross-entropy loss with a suitable stability objective, encouraging model prediction to be more constant around the input data while mitigating underfitting. However, stability training increases the computational costs when compared to training with simply the standard cross-entropy loss.

3.2 Theoretical Results

NoisyMix seeks to minimize a stochastic approximation of $L^{NoisyMix}$ by sampling a finite number of $\lambda, \boldsymbol{\xi}, x_{am}, x'_{am}$ values and using minibatch gradient descent to minimize this loss approximation. The empirical loss to be minimized during the training is

$$L_n^{NoisyMix} = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \mathbb{E}_{\lambda \sim Beta(\alpha,\beta)} \mathbb{E}_{\boldsymbol{\xi} \sim \mathcal{Q}} [a_{i,j} + \gamma b_{i,j}],$$

where

$$\begin{aligned} a_{i,j} &:= l(p(M_{\lambda,\boldsymbol{\xi}}(x_i, x_j)), M_\lambda(y_i, y_j)), \\ b_{i,j} &:= JS_\pi(p(M_{\lambda,\boldsymbol{\xi}}(x_i, x_j)), p(M_{\lambda,\boldsymbol{\xi}}(A(x_i), A(x_j)))). \end{aligned}$$

In App. A, we illustrate the effectiveness of *NoisyMix* in smoothing the decision boundary for a binary classification task. Overall, we observe that *NoisyMix* leads to the smoothest decision boundary when compared to other methods (see Fig. 3), and thus more robust models, since the predicted label stays the same even if the data are perturbed.

To understand these regularizing effects of *NoisyMix* better, we provide some mathematical analysis via the lens of implicit regularization (Mahoney, 2012) in App. B (see Theorem 1 and Theorem 2). Such lens was adopted to understand stochastic optimization (Ali et al., 2020; Smith et al., 2021) and regularizing effects of noise injections (Camuto et al., 2020; Lim et al., 2021; Gong et al., 2021). One common approach to study implicit regularization is to approximate it by an appropriate explicit regularizer, which we follow here for the regime of small mixing coefficient and noise levels. In this regime, we shall show that *NoisyMix* can lead to models with enhanced robustness and stability.

Following Lamb et al. (2019); Lim et al. (2022), we consider the binary cross-entropy loss, setting $h(z) = \log(1 + e^z)$, with the labels y taking value in $\{0, 1\}$ and the classifier model $f : \mathbb{R}^d \to \mathbb{R}$. In the following, we assume that the model parameter $\theta \in \Theta := \{\theta : y_i f(x_i) + (y_i - 1) f(x_i) \geq 0$ for all $i \in [n]\}$. We remark that this set contains the set of all parameters with correct classifications of training samples (before applying *NoisyMix*), since $\{\theta : 1_{\{f(x_i)\geq 0\}} = y_i \text{ for all } i \in [n]\} \subset \Theta$. Therefore, the condition of $\theta \in \Theta$ is fulfilled when the model classifies all labels correctly for the training data before applying *NoisyMix*. Since the training error often becomes zero in finite time in practice, we shall study the effect of *NoisyMix* on model robustness in the regime of $\theta \in \Theta$.

Working in the data-dependent parameter space Θ, we obtain the following result, which is an informal version of our main theoretical result (see Theorem 3 in App. C for the precise statements).

Theorem (Informal). Let $\epsilon > 0$ be a small parameter, and rescale $1 - \lambda \mapsto \epsilon(1-\lambda)$, $\sigma_{add} \mapsto \epsilon\sigma_{add}$, $\sigma_{mult} \mapsto \epsilon\sigma_{mult}$. Then, under additional reasonable assumptions,

$$L_n^{NoisyMix} \geq \frac{1}{n}\sum_{i=1}^{n} \max_{\|\delta_i\|_2 \leq \epsilon_i^{mix}} l(f(x_i + \delta_i), y_i) + \gamma \tilde{L}_n^{JSD} + \epsilon^2 L_n^{reg} + \epsilon^2 \phi(\epsilon), \tag{3}$$

for some data-dependent radii $\epsilon_i^{mix} > 0$, stability objective $\tilde{L}_n^{JSD}$, data-dependent regularizer L_n^{reg}, and function ϕ such that $\lim_{\epsilon\to 0}\phi(\epsilon) = 0$.

This theorem implies that minimizing the *NoisyMix* loss results in a small *regularized* adversarial loss and a stable model. Therefore, training with *NoisyMix* not only enhances robustness with respect to input perturbations, but it also imposes additional smoothness.

Lastly, we remark that one could generalize the JSD (designed for only two distributions) to provide a similarity measure for any finite number of distributions. This leads to the following definition (Lin, 1991):

$$JS_\pi(p_1, \ldots, p_K) := \sum_{k=1}^{K} \pi_k KL\left(p_k \middle\| \sum_{j=1}^{K} \pi_j p_j\right),$$

where $p_1, \ldots, p_K$ are K probability distributions with weights $\pi := (\pi_1, \ldots, \pi_K)$. We note that such a measure, with $K := 3$ and the $\pi_i := 1/3$, will be used in our experiments and was also used in Hendrycks et al. (2020) for obtaining optimal results.

4 EXPERIMENTAL RESULTS

In this section, we compare *NoisyMix* to other data augmentation schemes, studying in particular the robustness with respect to various common corruptions and data shifts. We will first discuss the experimental details, and then present results for ImageNet, followed by results for smaller architectures trained on CIFAR-10 and CIFAR-100. Finally, we will present a detailed ablation study, and investigate whether robustness transfers to downstream tasks.

Research code is provided on our GitHub repository: https://github.com/erichson/NoisyMix.

4.1 Experiment Details

Datasets. We center our experiments around several datasets to benchmark the robustness of neural networks to common corruptions and perturbations.

The ImageNet-C, CIFAR-10-C, and CIFAR-100-C datasets (Hendrycks and Dietterich, 2018) provide corrupted versions of the original ImageNet (Deng et al., 2009) and CIFAR (Krizhevsky, 2009) test sets. Specifically, each dataset emulates noise, blur, weather, and digital distortions. In total, there are 15 different perturbation and corruption types, and for each type, there are sets with 5 severity levels which allows us to study the performance with respect to increasing data shifts. Figure 5 in the App. illustrates the different corruption types. The average accuracy across all corruption types and severity levels provides a score for robustness.

We also consider the ImageNet-R dataset (Hendrycks et al., 2021), which is designed to measure the robustness of models to various abstract visual renditions (e.g., graffiti, origami, cartoons). This dataset provides 30,000 image renditions for a subset of 200 ImageNet object classes. The collected renditions have textures and local image statistics that are distinct from the standard ImageNet examples. This dataset provides a test for robustness to naturally occurring domain shifts, and it is complementary to ImageNet-C, which is used to test robustness to synthetic domain shifts.

In addition, we consider the ImageNet-P dataset (Hendrycks and Dietterich, 2018), which provides 10 different perturbed sequences for each ImageNet validation image. Each sequence contains 30 or more frames, where each subsequent frame is a perturbed version of the previous frame. The applied perturbations are relatively subtle, ensuring that the frames do not go too far out-of-distribution. This dataset is specifically designed to evaluate the prediction stability of a model f, using the flip probability as a metric to evaluate stability. Specifically, the flip probability is computed as $FB_p = \frac{1}{m(n-1)}\sum_{i=1}^{m}\sum_{j=2}^{n} I(f(x_j^{(i)}) \neq f(x_{j-1}^{(i)}))$, for given m sequences $\{(x_i^{(i)}, \ldots, x_n^{(i)})\}_{i=1}^{m}$ of length n for a corruption type p. Here I denotes an indicator function. Unstable models are characterized by a high flip probability, i.e., the prediction behavior across consecutive frames is erratic, whereas consistent predictions indicate stable statistical predictions. For sequences that are constructed with noise

Table 1: Clean test accuracy of ResNet-50 models on ImageNet, and average robust accuracy on ImageNet-C and ImageNet-R (higher values are better). In addition, we show the mean flip rate for ImageNet-P (lower values are better) and the robustness to adversarial examples constructed with FGSM (higher values are better). The values in parenthesis are the average robust accuracy for ImageNet-C and mean flip probability for ImageNet-P excluding noise perturbations.

	ImageNet (↑%)	ImageNet-C (↑%)	ImageNet-R (↑%)	ImageNet-P (↓%)	FGSM (↑%)
Baseline	76.1	39.2 (42.3)	36.2	58.0 (57.8)	6.6
Adversarial Trained	63.9	32.1 (35.4)	38.9	33.3 (33.4)	**43.1**
Stylized ImageNet	74.9	45.2 (46.6)	41.5	54.4 (55.2)	7.8
AutoAugment	77.6	45.7 (47.3)	39.0	56.5 (57.7)	9.9
Mixup	77.5	46.2 (48.4)	39.6	56.4 (58.7)	23.5
Manifold Mixup	76.7	43.9 (46.5)	39.7	56.0 (58.2)	27.7
CutMix	78.6	41.0 (43.1)	34.8	58.6 (59.9)	33.7
Puzzle Mix	**78.7**	44.6 (46.4)	39.5	55.5 (57.0)	28.4
AugMix	77.5	48.3 (50.5)	41.0	37.6 (37.2)	9.9
NoisyMix (ours)	77.6	**52.3 (52.4)**	**45.7**	**28.5 (29.7)**	29.6

perturbations (i.e., white, shot and impulse noise), the following modified flip probability is computed: $FB_p = \frac{1}{m(n-1)} \sum_{i=1}^{m} \sum_{j=2}^{n} I(f(x_j^{(i)}) \neq f(x_i^{(i)}))$, which accounts for the fact that these sequences are not temporally related. We obtain a flip rate by dividing the flip probability of a given model with the flip probability of AlexNet (Hendrycks and Dietterich, 2018). Finally, the flip rate is averaged across all corruption types to provide an overall score, denoted as mean flip rate (mFP).

Baselines and Training Details. We consider several data augmentation schemes as baselines, including style transfer (Geirhos et al., 2018), AutoAugment (Cubuk et al., 2019), Mixup (Zhang et al., 2018), Manifold Mixup pVerma et al. (2019), CutMix (Yun et al., 2019), PuzzleMix (Kim et al., 2020) and AugMix (Hendrycks et al., 2020). In our ImageNet experiments, we consider the standard ResNet-50 (He et al., 2016a) architecture as a backbone, trained on ImageNet-1k. We use publicly available models, and for a fair comparison we do not consider models that have been pretrained on larger datasets or that have been trained with any extra data to achieve better performance. In our CIFAR-10 and CIFAR-100 experiments, we use preactivated ResNet-18 (He et al., 2016b) and Wide-ResNet-28x2 (Zagoruyko and Komodakis, 2016). We train all models from scratch for 200 epochs, using the same basic tuning parameters. For the augmentation schemes, we use the prescribed parameters in the corresponding papers.

4.2 ImageNet Results

Table 1 summarizes the results for ImageNet models, trained with different data augmentation schemes. Our *NoisyMix* scheme leads to a model that is substantially more robust, and it also improves the test accuracy on clean data as compared to the baseline model. However, it is less accurate than models trained with augmentation schemes such as PuzzleMix, and CutMix. In contrast, adversarial training and style transfer reduce the in-distribution test accuracy.

On ImageNet-C, *NoisyMix* gains about 13% compared to the baseline model and about 4% compared to AugMix. Since *NoisyMix* uses white noise during training, we also show the average robust accuracy in parentheses, excluding any noise perturbations (i.e., excluding white, shot and impulse noise). Table 6 in the App. provides results for each perturbation type. Further, Figure 2 shows the robust accuracy as a function of the severity level of the corruptions, and it can be seen that the advantage of our model is more pronounced with increasing severity levels.

The advantage of *NoisyMix* is also pronounced when

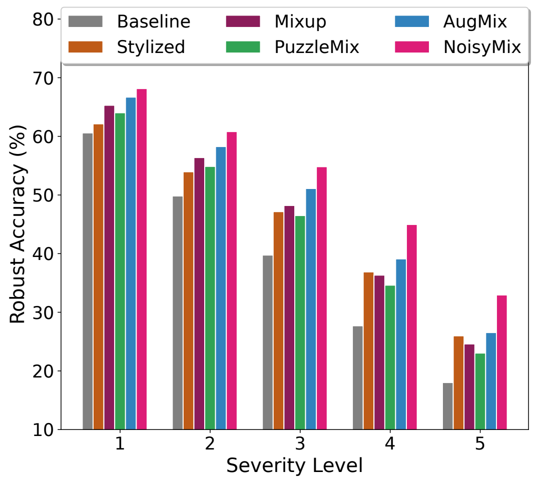


Figure 2: Robust accuracy on ImageNet-C for different severity levels. *NoisyMix* shows the best performance.

Table 2: Results for ResNet-18 models trained on CIFAR-10/100 and evaluated on CIFAR-10/100-C. We train all models with 5 different seeds and show the mean and (minimum/maximum) test accuracy. (Note, the results for AugMax, indicated by a *, are adapted from Wang et al. (2021).)

	CIFAR-10 (↑%)	CIFAR-10-C (↑%)	CIFAR-100 (↑%)	CIFAR-100-C (↑%)
Baseline	95.3 (95.2 / 95.4)	73.8 (73.5 / 74.2)	77.7 (77.5 / 78.1)	48.4 (48.0 / 48.8)
Mixup	95.8 (95.3 / 96.1)	80.4 (79.7 / 80.9)	79.7 (79.5 / 79.9)	54.2 (53.9 / 54.8)
Manifold Mixup	95.9 (95.7 / 96.1)	77.7 (77.3 / 78.2)	**80.3** (**79.8** / **80.7**)	53.4 (52.9 / 53.6)
CutMix	**96.5** (**96.4** / **96.6**)	73.0 (71.8 / 73.9)	79.3 (78.8 / 80.0)	47.9 (47.6 / 48.3)
Puzzle Mix	**96.5** (96.3 / **96.6**)	84.1 (83.7 / 84.3)	79.4 (79.1 / 79.6)	57.4 (56.9 / 57.8)
NFM	95.4 (95.2 / 95.6)	83.3 (82.7 / 83.9)	79.4 (78.9 / 79.8)	59.7 (59.2 / 60.2)
AugMix	95.5 (95.0 / 95.8)	88.8 (88.1 / 89.3)	77.0 (76.6 / 77.3)	64.6 (64.4 / 64.8)
AugMax*	-	90.36	-	65.75
NoisyMix (ours)	95.3 (95.2 / 95.3)	**91.2** (**91.1** / **91.3**)	79.7 (79.6 / 79.9)	**71.0** (**70.3** / **71.3**)

Table 3: Results for Wide-ResNet-28x2 trained on CIFAR-10/100 and evaluated on CIFAR-10/100-C. We train all models with 5 different seeds and show the mean and (minimum/maximum) test accuracy.

	CIFAR-10 (↑%)	CIFAR-10-C (↑%)	CIFAR-100 (↑%)	CIFAR-100-C (↑%)
Baseline	96.0 (95.9 / 96.1)	74.9 (74.5 / 75.4)	79.0 (78.8 / 79.1)	47.4 (47.0 / 48.0)
Mixup	96.7 (96.6 / 96.8)	80.8 (80.7 / 81.0)	80.9 (80.8 / 81.0)	53.9 (53.1 / 54.3)
Manifold Mixup	96.6 (96.5 / 96.7)	77.7 (76.3 / 78.4)	**81.8** (**81.5** / **82.0**)	53.2 (52.4 / 53.7)
CutMix	96.7 (96.6 / 96.8)	71.6 (71.1 / 71.9)	80.3 (80.0 / 80.6)	48.5 (48.1 / 48.9)
Puzzle Mix	**97.0** (**96.9** / **97.2**)	82.6 (82.1 / 83.0)	81.2 (81.0 / 81.6)	56.3 (55.8 / 56.7)
NFM	96.2 (96.1 / 96.3)	82.3 (81.8 / 82.8)	80.2 (79.6 / 80.6)	58.7 (58.2 / 59.2)
AugMix	96.5 (96.4 / 96.6)	90.2 (89.7 / 90.7)	80.5 (79.9 / 80.8)	66.7 (65.9 / 67.2)
NoisyMix (ours)	96.3 (96.2 / 96.4)	**91.8** (**91.2** / **92.7**)	81.3 (81.0 / 81.6)	**71.9** (**71.7** / **72.3**)

evaluated on real-world examples as provided by the ImageNet-R dataset. This shows that *NoisyMix* not only improves the robustness with respect to synthetic distribution shifts but also to natural distribution shifts. It can also be seen that the stylized ImageNet model performs well on this task. We also use the ImageNet-R dataset to show that *NoisyMix* yields well-calibrated predictions, see Table 11 in the Appendix.

NoisyMix also achieves state-of-the-art result on ImageNet-P, reducing the mFP from 58% down to 28.5%. Again, we also show the mFP computed for a subset that excludes noise perturbations (i.e., white and shot noise). Table 6 in the App. shows detailed results for all perturbation types.

To provide another metric for robustness, we evaluate the resilience of ImageNet models with respect to adversarial examples crafted by the simple fast gradient sign method (FGSM) (Goodfellow et al., 2014). *NoisyMix* helps to improve adversarial robustness with respect to these weak adversarial examples crafted with FGSM ($\epsilon = 4/255$). In contrast, AugMix is less resilient.

4.3 CIFAR-10 and CIFAR-100 Results

We train ResNet-18 and Wide-ResNet-28x2 models (with 5 seeds) on both CIFAR-10 and CIFAR-100, using different data augmentation schemes. Table 2 and 3 provide the results, showing the average, minimum, and maximum test accuracy for the various test settings. Similar to the previous results, PuzzleMix achieves the best test accuracy on clean test sets. Further, we can see that the advantage of *NoisyMix* is pronounced on CIFAR-100-C, where we improve the robust accuracy by more than 5% as compared to AugMix.

4.4 Ablation Study

At a high-level, we can decompose *NoisyMix* into 3 building blocks: (i) noisy feature mixup, (ii) stability training, and (iii) data augmentations. Further, noisy feature mixup can be decomposed into feature mixup and noise injections. We provide a detailed ablation to show that all components are required to achieve state-of-the-art performance.

We train models on CIFAR-100 and evaluate the in-domain test accuracy and robust accuracy on CIFAR-100-C. Table 4 shows the results, and it can be seen that data augmentations have a significant impact on robustness, i.e., robust accuracy is improved by 17%. The combination of feature mixup and noise injections helps to improve the in-domain test accuracy, while the robust accuracy is improved by about 11%. The combination of both considerably helps to improve in-domain test accuracy and robustness. The performance is further improved when stability training is

Table 4: Ablation study using a Wide-ResNet-28x2 trained on CIFAR-100. The combination of feature mixing and noise injections with stability training, using augmented data, helps to boost clean and robust accuracy.

Augmentations	Feature Mixing	Noise Injections	JSD Loss	CIFAR-100 (↑%)	CIFAR-100-C (↑%)	Robustness Gain (%)
✗	✗	✗	✗	79.0	47.4	-
✗	✗	✓	✗	79.1	49.5	+2.1
✗	✓	✗	✗	81.8	53.2	+5.8
✗	✓	✓	✗	80.2	58.7	+11.3
✓	✗	✗	✗	79.1	64.0	+16.6
✓	✗	✓	✓	80.9	66.7	+19.3
✓	✗	✗	✓	80.5	66.9	+19.5
✓	✓	✓	✗	80.4	68.6	+21.2
✓	✓	✗	✓	**82.1**	69.3	+21.9
✓	✓	✓	✓	81.3	**71.9**	+**24.4**

Table 5: Transfer learning results for ResNet-50 models trained on ImageNet and transfered to CIFAR-10/100. Then robustness is evaluated on CIFAR-10-C/100-C. We show the average robust accuracy across all common corruptions / across noise corruptions / across whether, blur and digital corruptions. (Higher values are better.)

Source Model	CIFAR-10 (↑%)	CIFAR-10-C (↑%)	CIFAR-100 (↑%)	CIFAR-100-C (↑%)
Baseline	97.0	77.1 / 48.8 / 86.5	84.3	57.1 / 28.0 / 66.8
PuzzleMix	97.1	77.3 / 47.7 / 87.2	84.5	57.2 / 26.1 / 67.6
Mixup	97.1	78.6 / 51.7 / 87.6	84.5	57.3 / 26.4 / 67.6
AutoAugment	97.2	78.7 / 54.6 / 86.7	84.9	57.6 / 27.3 / 67.6
AugMix	**97.7**	79.9 / 53.9 / 88.6	**86.5**	58.8 / 24.2 / 66.8
NoisyMix (ours)	**97.7**	**81.8** / **60.3** / **88.9**	85.9	**60.3** / **34.5** / **68.8**

used in addition, yielding our *NoisyMix* scheme. Indeed, *NoisyMix* (i.e., the favorable combination of all components) is outperforming all other variants.

4.5 Does Robustness Transfer?

Motivated by Shafahi et al. (2019b), who showed that adversarial robustness transfers, we study whether robustness to common corruptions transfers from a source model to a target task. To do so, we fine-tune ImageNet models on CIFAR-10 and CIFAR-100, and then use the corrupted test datasets to evaluate robustness.

Table 5 shows that target models inherit some robustness from the source models. More robust ImageNet models, i.e., models trained with *NoisyMix*, lead to more robust target models. In particular, robustness to noise perturbations is transferred. For example, the parent model trained with *NoisyMix* yields target models that are about 12% more robust to noise perturbations on CIFAR-10-C, and about 6% more robust on CIFAR-100-C. Moreover, a robust parent model can help to improve the in-domain accuracy on the target task, which is in alignment with the results by Utrera et al. (2020); Salman et al. (2020).

5 CONCLUSION

Desirable properties of deep classifiers for computer vision are (i) in-domain accuracy, (ii) robustness to domain shifts, and (iii) well-calibrated estimation of class membership probabilities. Motivated by the challenge to achieve these, we propose *NoisyMix*, a novel and theoretically justified training scheme. Our empirical results demonstrate the advantage, i.e., improvement of (i), (ii), and (iii), when compared to standard training and other data augmentation methods. These findings are supported by theoretical results that show that *NoisyMix* can improve model robustness.

A limitation of *NoisyMix* is that it is tailored towards computer vision tasks and not directly applicable to natural language processing tasks, or time series tasks. Thus, future work will explore how *NoisyMix* can be extended to a broader range of tasks. In particular, we anticipate that *NoisyMix* will be effective in improving the robustness of models used in Scientific Machine Learning (e.g., models used for climate predictions). Since this paper studies a new training scheme to improve model robustness there are no potential negative societal impacts of our work.

Acknowledgements

We would like to thank all the anonymous reviewers for providing helpful feedback for improving this paper. N. B. Erichson would like to acknowledge support from NSF (DMS-2319621), and NERSC (DE-AC02-05CH11231). S. H. Lim would like to acknowledge support from the WINQ Fellowship, the SNIC AI/ML grant (2018-05973), and the Swedish Research Council (VR/2021-03648). Our conclusions do not necessarily reflect the position or the policy of our sponsors, and no official endorsement should be inferred.

References

Alnur Ali, Edgar Dobriban, and Ryan Tibshirani. The implicit regularization of stochastic gradient flow for least squares. In *Proceedings of the International Conference on Machine Learning*, pages 233–244. PMLR, 2020.

Guozhong An. The effects of adding noise during backpropagation training on a generalization performance. *Neural Computation*, 8(3):643–674, 1996.

Chris M Bishop. Training with noise is equivalent to Tikhonov regularization. *Neural Computation*, 7(1): 108–116, 1995.

Alexander Camuto, Matthew Willetts, Umut Şimşekli, Stephen Roberts, and Chris Holmes. Explicit regularisation in Gaussian noise injections. *arXiv preprint arXiv:2007.07368*, 2020.

Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069*, 2018.

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

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

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

Samuel Dodge and Lina Karam. Understanding how image quality affects deep neural networks. In *Proceedings of the International Conference on Quality of Multimedia Experience*, pages 1–6, 2016.

Bradley Efron. Bootstrap methods: another look at the jackknife. In *Breakthroughs in Statistics*, pages 569–593. Springer, 1992.

Erik Englesson and Hossein Azizpour. Generalized Jensen-Shannon divergence loss for learning with noisy labels. *Advances in Neural Information Processing Systems*, 34:30284–30297, 2021.

Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *Proceedings of the International Conference on Learning Representations*, 2018.

Chengyue Gong, Tongzheng Ren, Mao Ye, and Qiang Liu. MaxUp: Lightweight adversarial training with data augmentation improves neural network training. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 2474–2483, 2021.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Caglar Gulcehre, Marcin Moczulski, Misha Denil, and Yoshua Bengio. Noisy activation functions. In *Proceedings of the International Conference on Machine Learning*, pages 3059–3068. PMLR, 2016.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *Proceedings of the International Conference on Machine Learning*, pages 1321–1330, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Proceedings of the European Conference on Computer Vision*, pages 630–645, 2016b.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *Proceedings of the International Conference on Learning Representations*, 2018.

Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty. In *Proceedings of the International Conference on International Conference on Machine Learning*, pages 2712–2721, 2019.

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. *Proceedings of the*

International Conference on Learning Representations, 2020.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the International Conference on Computer Vision*, 2021.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Peter J Huber. *Robust Statistics*, volume 523. John Wiley & Sons, 2004.

Andrew Ilyas, Shibani Santurkar, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. *Proceedings of the International Conference on Neural Information Processing Systems*, 32, 2019.

Jang-Hyun Kim, Wonho Choo, and Hyun Oh Song. Puzzle Mix: Exploiting saliency and local statistics for optimal mixup. In *Proceedings of the International Conference on Machine Learning*, 2020.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. *Proceedings of the International Conference on Neural Information Processing Systems*, 25:1097–1105, 2012.

J. Kukačka, V. Golkov, and D. Cremers. Regularization for deep learning: A taxonomy. Technical Report Preprint: arXiv:1710.10686, 2017.

Alex Lamb, Vikas Verma, Juho Kannala, and Yoshua Bengio. Interpolated adversarial training: Achieving robust neural networks without sacrificing too much accuracy. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, pages 95–103, 2019.

Soon Hoe Lim, N Benjamin Erichson, Liam Hodgkinson, and Michael W Mahoney. Noisy recurrent neural networks. In *Proceedings of the International Conference on Neural Information Processing Systems*, 2021.

Soon Hoe Lim, N. Benjamin Erichson, Francisco Utrera, Winnie Xu, and Michael W. Mahoney. Noisy feature mixup. In *International Conference on Learning Representations*, 2022.

Jianhua Lin. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151, 1991.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *Proceedings of the International Conference on Learning Representations*, 2018.

M. W. Mahoney. Approximate computation and implicit regularization for very large-scale data analysis. In *Proceedings of the 31st ACM Symposium on Principles of Database Systems*, pages 143–154, 2012.

Sylvestre-Alvise Rebuffi, Sven Gowal, Dan A Calian, Florian Stimberg, Olivia Wiles, and Timothy Mann. Fixing data augmentation to improve adversarial robustness. *arXiv preprint arXiv:2103.01946*, 2021.

Leslie Rice, Eric Wong, and Zico Kolter. Overfitting in adversarially robust deep learning. In *Proceedings of the International Conference on Machine Learning*, pages 8093–8104. PMLR, 2020.

Evgenia Rusak, Lukas Schott, Roland S Zimmermann, Julian Bitterwolf, Oliver Bringmann, Matthias Bethge, and Wieland Brendel. A simple way to make neural networks robust against diverse image corruptions. In *Proceedings of the European Conference on Computer Vision*, pages 53–69, 2020.

Hadi Salman, Andrew Ilyas, Logan Engstrom, Ashish Kapoor, and Aleksander Madry. Do adversarially robust ImageNet models transfer better? In *Proceedings of the International Conference on Neural Information Processing Systems*, volume 33, 2020.

Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In *Proceedings of the International Conference on Neural Information Processing Systems*, 2019a.

Ali Shafahi, Parsa Saadatpanah, Chen Zhu, Amin Ghiasi, Christoph Studer, David Jacobs, and Tom Goldstein. Adversarially robust transfer learning. In *Proceedings of the International Conference on Learning Representations*, 2019b.

Samuel L Smith, Benoit Dherin, David GT Barrett, and Soham De. On the origin of implicit regularization in stochastic gradient descent. *arXiv preprint arXiv:2101.12176*, 2021.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Matthew Tancik, Pratul P Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *arXiv preprint arXiv:2006.10739*, 2020.

Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. Measuring robustness to natural distribution shifts in image classification. *arXiv preprint arXiv:2007.00644*, 2020.

Francisco Utrera, Evan Kravitz, N Benjamin Erichson, Rajiv Khanna, and Michael W Mahoney. Adversarially-trained deep nets transfer better: Illustration on image classification. In *Proceedings of the International Conference on Learning Representations*, 2020.

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 *Proceedings of the International Conference on Learning Representations*, 2019.

Haotao Wang, Chaowei Xiao, Jean Kossaifi, Zhiding Yu, Anima Anandkumar, and Zhangyang Wang. AugMax: Adversarial composition of random augmentations for robust training. *Advances in Neural Information Processing Systems*, 34, 2021.

Lilian Weng. From GAN to WGAN. *arXiv preprint arXiv:1904.08994*, 2019.

Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves ImageNet classification. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 10687–10698, 2020.

Dong Yin, Raphael Gontijo Lopes, Jon Shlens, Ekin Dogus Cubuk, and Justin Gilmer. A Fourier perspective on model robustness in computer vision. *Proceedings of the International Conference on Neural Information Processing Systems*, 32:13276–13286, 2019.

Bin Yu. Stability. *Bernoulli*, 19(4):1484–1500, 2013.

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 International Conference on Computer Vision*, 2019.

Bianca Zadrozny and Charles Elkan. Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. In *Proceedings of the International Conference on Machine Learning*, volume 1, pages 609–616, 2001.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference 2016*, 2016.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *Proceedings of the International Conference on Learning Representations*, 2018.

Linjun Zhang, Zhun Deng, Kenji Kawaguchi, Amirata Ghorbani, and James Zou. How does mixup help with robustness and generalization? *arXiv preprint arXiv:2010.04819*, 2020.

Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. Improving the robustness of deep neural networks via stability training. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 4480–4488, 2016.

Checklist

1. For all models and algorithms presented, check if you include:
    (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
    (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [No]
    (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]
2. For any theoretical claim, check if you include:
    (a) Statements of the full set of assumptions of all theoretical results. [Yes]
    (b) Complete proofs of all theoretical results. [Yes]
    (c) Clear explanations of any assumptions. [Yes]
3. For all figures and tables that present empirical results, check if you include:
    (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
    (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
    (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
    (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [No]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
    (a) Citations of the creator If your work uses existing assets. [Yes]
    (b) The license information of the assets, if applicable. [Not Applicable]
    (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
    (d) Information about consent from data providers/curators. [Not Applicable]
    (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]
5. If you used crowdsourcing or conducted research with human subjects, check if you include:
    (a) The full text of instructions given to participants and screenshots. [Not Applicable]
    (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
    (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

APPENDIX FOR "*NoisyMix*"

In this **Appendix**, we provide additional materials to support our results in the main paper. In particular, we provide mathematical analysis of *NoisyMix* to shed light on its regularizing and robustness properties, as well as additional empirical results and the details.

Organizational Details. This **Appendix** is organized as follows.

- In Section A, we illustrate the regularizing effects of stability training with the JSD, Manifold Mixup, and noise injection on a toy dataset.
- In Section B, we study *NoisyMix* via the lens of implicit regularization. Our contributions there are Theorem 1 and Theorem 2, which show that minimizing the *NoisyMix* loss function is approximately equivalent to minimizing a sum of the original loss and data-dependent regularizers, amplifying the regularizing effects of AugMix according to the mixing and noise levels.
- In Section C, we show that *NoisyMix* training can improve model robustness when compared to standard training. Our main contribution there is Theorem 3, which shows that, under appropriate assumptions, *NoisyMix* training approximately minimizes an upper bound on the sum of an adversarial loss, a stability objective and data-dependent regularizers.
- In Section D, we provide additional experimental results and their details.

We recall the notation that we use in the main paper as well as this Appendix.

Notation. I denotes identity matrix, $[K] := \{1, \dots, K\}$, the superscript ^{T} denotes transposition, $\circ$ denotes composition, $\odot$ denotes Hadamard product, $\mathbb{1}$ denotes the vector with all components equal one. For a vector v, v^i denotes its ith component and $\|v\|_p$ denotes its l_p norm for $p > 0$. $M_\lambda(a, b) := \lambda a + (1 - \lambda)b$, for random variables a, b, λ. For $\alpha, \beta > 0$, $\tilde{\mathcal{D}}_\lambda := \frac{\alpha}{\alpha+\beta} Beta(\alpha + 1, \beta) + \frac{\beta}{\alpha+\beta} Beta(\beta + 1, \alpha)$ is a uniform mixture of two Beta distributions. For the two vectors a, b, $\cos(a, b) := \langle a, b\rangle / \|a\|_2 \|b\|_2$ denotes their cosine similarity.

To simplify mathematical analysis and enable clearer intuition, we shall restrict to the special case when the mixing is done only at the input level. Moreover, we consider the setting described in the main paper for the case of $K = 2$ and $\pi_0 = \pi_1 = 1/2$ for our analysis. The results for the general case can be derived analogously using our techniques.

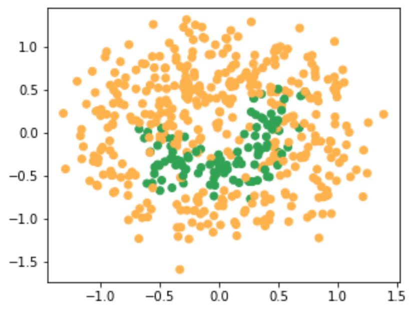
A Illustration of the Regularizing Effects of Stability Training, Manifold Mixup and Noise Injection on a Toy Dataset

We consider a binary classification task for the noise corrupted 2D dataset whose points are separated by a concave polygon-shaped buffer band. Inner and outer points that disjoint the band directly correspond to two label classes, while those falling onto the band are randomly assigned to one of the classes. We generate 500 samples by setting the scale factor to be 0.5 and adding Gaussian noise with zero mean and standard deviation of 0.2 to the shape. Fig. 3 shows the data points.

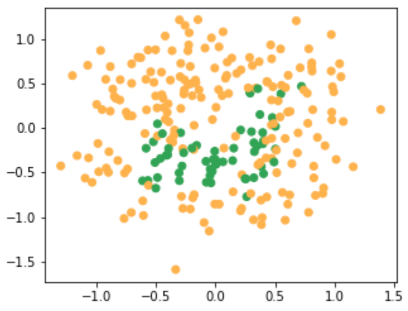
We train a fully connected feedforward shallow neural network (with 32 neurons) with the ReLU activation function on these data, using 250 points for training and 250 for testing. All models are trained with Adam, learning rate of 0.1, and batch size of 50 for 100 epochs. The seed is fixed across all experiments.

We focus on the effects of stability training (with the JSD), Manifold Mixup, noise injection, and their combinations on the test performance. Note that we do not explore AugMix's data augmentation here, since the operations used there are not meaningful for the toy dataset here.

Fig. 4 illustrates how different training schemes affect the decision boundaries of the neural network classifier. First, we can see that *NoisyMix* (which combines feature mixup, noise injections, and stability training) is most effective at smoothing the decision boundary of the trained classifiers, imposing the strongest smoothness on the dataset. Here it also yields the best test accuracy among all considered training schemes. We can also see that stability training helps to boost the effectiveness of using Manifold Mixup. In contrast, NFM without stability training performs better than the baseline and Manifold Mixup, but not as well as *NoisyMix* and Manifold Mixup with stability training.

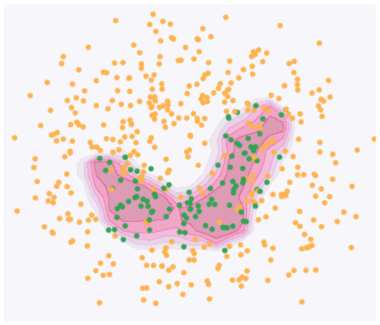


(a) All data points.

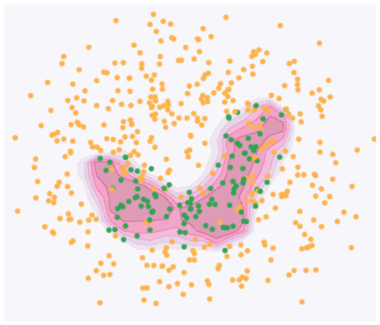


(b) Data points used for testing.

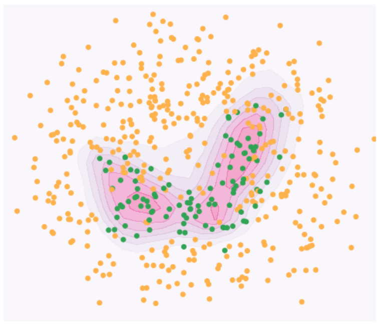
Figure 3: The dataset in $\mathbb{R}^2$ that we use for binary classification.



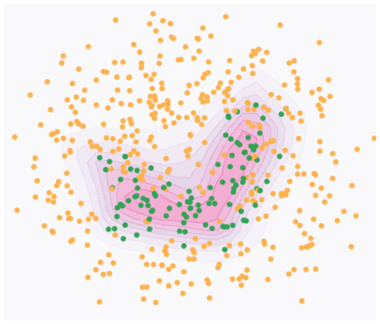
Baseline (86.8%).



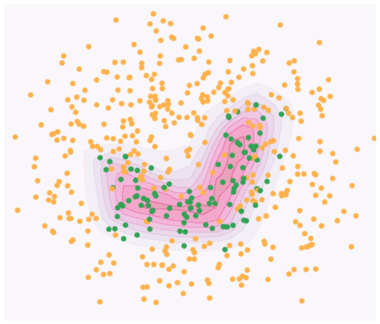
Noise injection (86.8%).



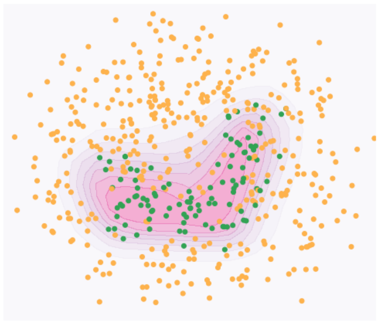
Manifold Mixup (87.4%).



NFM (87.6%).



Manifold Mixup + JSD (88.0 %).



NoisyMix (88.8%).

Figure 4: Decision boundaries and test accuracy (in parenthesis) for different training schemes on a 2D binary classification problem. *NoisyMix* is most effective at smoothing the decision boundary, and it also yields the best test accuracy among all considered training schemes.

B Implicit Regularization of NoisyMix

In this section, we are going to identify the implicit regularization effects of *NoisyMix* by analyzing the NFM loss and the stability loss.

We emphasize that while the analysis for the NFM loss can be adapted from that in Lim et al. (2022) in a straightforward manner, the analysis and results for the stability loss (see Theorem 2) are novel and needs to be derived independently. Importantly, Theorem 2 is a crucial ingredient for the proof of our main theoretical result, Theorem 3 (stated informally in the main paper). The analysis for the stability loss is rather tedious, and we are going to provide the details of the derivation for completeness (see Subsection B.2).

B.1 Implicit Regularization for the NFM Loss

Recall that

$$L^{NFM} = \mathbb{E}_{(x,y),(x',y')\sim\mathcal{D}}\mathbb{E}_{\lambda\sim Beta(\alpha,\beta)}\mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{Q}}l(p(M_{\lambda,\boldsymbol{\xi}}(x,x')), M_\lambda(y,y')). \tag{4}$$

We consider loss functions of the form $l(p(x),y) := h(p(x)) - yp(x)$, which includes standard choices such as the logistic loss and the cross-entropy loss. Denote $L_n^{std} := \frac{1}{n}\sum_{i=1}^n l(p(x_i),y_i)$ and let $\mathcal{D}_x$ be the empirical distribution of training samples $\{x_i\}_{i\in[n]}$. We shall show that NFM exhibits a natural form of implicit regularization, i.e., regularization imposed implicitly by the stochastic learning strategy or approximation algorithm, without explicitly modifying the loss.

Recall from the main paper that the NFM loss function to be minimized is

$$L_n^{NFM} = \frac{1}{n^2}\sum_{i=1}^n\sum_{j=1}^n \mathbb{E}_{\lambda\sim Beta(\alpha,\beta)}\mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{Q}}l(p(M_{\lambda,\boldsymbol{\xi}}(x_i,x_j)), M_\lambda(y_i,y_j)), \tag{5}$$

where l is a loss function, $\boldsymbol{\xi} := (\xi^{add}, \xi^{mult})$ are drawn from some probability distribution $\mathcal{Q}$ with finite first two moments (with zero mean), and

$$M_{\lambda,\boldsymbol{\xi}}(x,x') := (\mathbb{1} + \sigma_{mult}\xi^{mult}) \odot M_\lambda(x,x') + \sigma_{add}\xi^{add}. \tag{6}$$

We emphasize that this loss is computed using only the clean data.

Let $\epsilon > 0$ be a small parameter. In the sequel, we assume that l is a loss function of the form $l(p(x),y) = h(p(x)) - yp(x)$, and rescale $1-\lambda \mapsto \epsilon(1-\lambda)$, $\sigma_{add} \mapsto \epsilon\sigma_{add}$, $\sigma_{mult} \mapsto \epsilon\sigma_{mult}$. We recall Lemma 2 in Lim et al. (2022) (adapted to our setting), which will be useful later, in the following.

Lemma 1. The NFM loss (5) can be equivalently written as

$$L_n^{NFM} = \frac{1}{n}\sum_{i=1}^n \mathbb{E}_{\lambda\sim\tilde{\mathcal{D}}_\lambda}\mathbb{E}_{x_r\sim\mathcal{D}_x}\mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{Q}}[h(p(x_i+\epsilon e_i^{NFM})) - y_i p(x_i + \epsilon e_i^{NFM})], \tag{7}$$

with

$$e_i^{NFM} = (\mathbb{1} + \epsilon\sigma_{mult}\xi^{mult}) \odot e_i^{mixup} + e_i^{noise}. \tag{8}$$

Here $e_i^{mixup} = (1-\lambda)(x_r - x_i)$ and $e_i^{noise} = \sigma_{mult}\xi^{mult}\odot x_i + \sigma_{add}\xi^{add}$, with $x_i, x_r \in \mathcal{X}$ and $\lambda \sim Beta(\alpha,\beta)$.

Note that the explicit mixing of the labels is absent in the formulation of Lemma 1. This gives a more natural interpretation of (5), in the sense that the inputs are sampled from the vicinal distribution induced by certain (data-dependent) random perturbations and the labels are kept the same.

Denote ∇p and $\nabla^2 p$ as the first and second directional derivative of p with respect to x respectively. By working in the small parameter regime, we can relate the NFM empirical loss L_n^{NFM} to the original loss L_n^{std} and identify the regularizing effects of NFM. More precisely, we perform a second-order Taylor expansion of the terms in bracket in Eq. (7) and arrive at the following result.

Theorem 1. Let $\epsilon > 0$ be a small parameter, and assume that h and p are twice differentiable. Then,

$$L_n^{NFM} = L_n^{std} + \epsilon R_1 + \epsilon^2\tilde{R}_2 + \epsilon^2\tilde{R}_3 + \epsilon^2\phi(\epsilon), \tag{9}$$

with

$$\tilde{R}_2 = R_2 + \sigma_{add}^2 R_2^{add} + \sigma_{mult}^2 R_2^{mult}, \tag{10}$$

$$\tilde{R}_3 = R_3 + \sigma_{add}^2 R_3^{add} + \sigma_{mult}^2 R_3^{mult}, \tag{11}$$

where

$$R_1 = \frac{\mathbb{E}_{\lambda\sim\tilde{\mathcal{D}}_\lambda}[1-\lambda]}{n}\sum_{i=1}^{n} h'(p(x_i)-y_i)\nabla p(x_i)^T \mathbb{E}_{x_r\sim\mathcal{D}_x}[x_r - x_i], \tag{12}$$

$$R_2 = \frac{\mathbb{E}_{\lambda\sim\tilde{\mathcal{D}}_\lambda}[(1-\lambda)^2]}{2n}\sum_{i=1}^{n} h''(p(x_i))\nabla p(x_i)^T \times \mathbb{E}_{x_r\sim\mathcal{D}_x}[(x_r - x_i)(x_r-x_i)^T]\nabla p(x_i), \tag{13}$$

$$R_3 = \frac{\mathbb{E}_{\lambda\sim\tilde{\mathcal{D}}_\lambda}[(1-\lambda)^2]}{2n}\sum_{i=1}^{n} (h'(p(x_i))-y_i) \times \mathbb{E}_{x_r\sim\mathcal{D}_x}[(x_r - x_i)^T\nabla^2 p(x_i)(x_r-x_i)], \tag{14}$$

$$R_2^{add} = \frac{1}{2n}\sum_{i=1}^{n} h''(p(x_i))\nabla p(x_i)^T \mathbb{E}_{\boldsymbol{\xi}}[\xi^{add}(\xi^{add})^T]\nabla p(x_i), \tag{15}$$

$$R_2^{mult} = \frac{1}{2n}\sum_{i=1}^{n} h''(p(x_i))\nabla p(x_i)^T (\mathbb{E}_{\boldsymbol{\xi}}[\xi^{mult}(\xi^{mult})^T]\odot x_i x_i^T)\nabla p(x_i), \tag{16}$$

$$R_3^{add} = \frac{1}{2n}\sum_{i=1}^{n} (h'(p(x_i))-y_i)\mathbb{E}_{\boldsymbol{\xi}}[(\xi^{add})^T\nabla^2 p(x_i)\xi^{add}], \tag{17}$$

$$R_3^{mult} = \frac{1}{2n}\sum_{i=1}^{n} (h'(p(x_i))-y_i)\mathbb{E}_{\boldsymbol{\xi}}[(\xi^{mult}\odot x_i)^T\nabla^2 p(x_i)(\xi^{mult}\odot x_i)], \tag{18}$$

and $\phi(\epsilon) = \mathbb{E}_{\lambda\sim\tilde{\mathcal{D}}_\lambda}\mathbb{E}_{x_r\sim\mathcal{D}_x}\mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{Q}}[\varphi(\epsilon)]$, with φ some function such that $\lim_{\epsilon\to 0}\varphi(\epsilon) = 0$.

Theorem 1 implies that, when compared to Manifold Mixup, NFM introduces additional smoothness, regularizing the directional derivatives, $\nabla p(x_i)$ and $\nabla^2 p(x_i)$, with respect to x_i, according to the noise levels σ_{add} and σ_{mult}, and thus amplifying the regularizing effects of Manifold Mixup and noise injection. In particular, making $\nabla^2 p(x_i)$ small can lead to smooth decision boundaries (at the input level), while reducing the confidence of model predictions. On the other hand, making the $\nabla p(x_i)$ small can lead to improvement in model robustness.

B.2 Implicit Regularization for the Stability Objective

Recall that L^{JSD} is given by:

$$\mathbb{E}_{x,x'\sim\mathcal{D}_x}\mathbb{E}_{x_{am}\sim\mathcal{A}(x),x'_{am}\sim\mathcal{A}(x')}\mathbb{E}_{\lambda\sim Beta(\alpha,\beta)}\mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{Q}}JS_\pi(p(M_{\lambda,\boldsymbol{\xi}}(x,x')), p(M_{\lambda,\boldsymbol{\xi}}(x_{am}, x'_{am}))). \tag{19}$$

The stability objective to be minimized is

$$L_n^{JSD} = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\mathbb{E}_{\lambda\sim Beta(\alpha,\beta)}\mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{Q}}JS_\pi(p(M_{\lambda,\boldsymbol{\xi}}(x_i,x_j)), p(M_{\lambda,\boldsymbol{\xi}}(A(x_i),A(x_j)))), \tag{20}$$

where $\boldsymbol{\xi} := (\xi^{add}, \xi^{mult})$ are drawn from some probability distribution $\mathcal{Q}$ with finite first two moments (with zero mean), and

$$M_{\lambda,\boldsymbol{\xi}}(x,x') := (\mathbb{1} + \sigma_{mult}\xi^{mult})\odot M_\lambda(x,x') + \sigma_{add}\xi^{add}. \tag{21}$$

Note that this loss is computed using both the clean data and the diversely transformed data.

Denote

$$\tilde{L}_n^{std} := \frac{1}{n}\sum_{i=1}^{n} JS_\pi(p(x_i), p(A(x_i))), \tag{22}$$

and let $\mathcal{D}_x$ be the empirical distribution of training samples $\{x_i\}_{i\in[n]}$. We shall show that the stability objective exhibits a natural form of implicit regularization.

Let $\epsilon > 0$ be a small parameter, and rescale $1-\lambda \mapsto \epsilon(1-\lambda)$, $\sigma_{add}\mapsto\epsilon\sigma_{add}$, $\sigma_{mult}\mapsto\epsilon\sigma_{mult}$ as before. For this stability objective, we have the following result, derived analogously to that in Lemma 1.

Lemma 2. The JSD loss (20) can be equivalently written as

$$L_n^{JSD} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{\lambda\sim\tilde{\mathcal{D}}_\lambda}\mathbb{E}_{x_r\sim\mathcal{D}_x}\mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{Q}}[JS_\pi(p(x_i+\epsilon e_i^{NFM}), p(A(x_i)+\epsilon e_i^{JSD}))], \tag{23}$$

with

$$e_i^{NFM} = (\mathbb{1}+\epsilon\sigma_{mult}\xi^{mult})\odot e_i^{mixup} + e_i^{noise}, \tag{24}$$

and

$$e_i^{JSD} = (\mathbb{1}+\epsilon\sigma_{mult}\xi^{mult})\odot \tilde{e}_i^{mixup} + \tilde{e}_i^{noise}. \tag{25}$$

Here $e_i^{mixup} = (1-\lambda)(x_r - x_i)$, $e_i^{noise} = \sigma_{mult}\xi^{mult}\odot x_i + \sigma_{add}\xi^{add}$, $\tilde{e}_i^{mixup} = (1-\lambda)(A(x_r)-A(x_i))$ and $\tilde{e}_i^{noise} = \sigma_{mult}\xi^{mult}\odot A(x_i) + \sigma_{add}\xi^{add}$, with $x_i, x_r \in \mathcal{X}$, $A\sim\mathcal{A}$ and $\lambda \sim Beta(\alpha,\beta)$.

The following proposition will be useful later.

Proposition 1. Let $\epsilon > 0$ be a small parameter. For $i = 0, 1, \dots, M$, denote $\tilde{p}_i := p(z_i + R_i(\epsilon))$, $p_i := p(z_i)$, $\nabla p_i := \nabla p(z_i)$ and $\nabla^2 p_i := \nabla^2 p(z_i)$, with the $z_i, \delta z_i \in \mathcal{X}$, and R_i some twice differentiable function such that $R_i(0) = 0$, $R_i'(0) = \delta z_i$, and $R_i''(0) = \delta^2 z_i$. Suppose that $p : \mathcal{X} \to \Delta^{K-1}$ is twice differentiable. Then, we have,

$$\begin{aligned} &JS_\pi(\tilde{p}_0,\dots,\tilde{p}_M) \\ &= JS_\pi(p_0,\dots,p_M) + \epsilon\sum_{k=1}^{K}\sum_{l=0}^{M}\pi_l\left((\nabla p_l^k)^T\delta z_l \cdot \ln\left(\frac{p_l^k}{\sum_{j=0}^M \pi_j p_j^k}\right)\right) \\ &\quad + \frac{\epsilon^2}{2}\sum_{k=1}^{K}\sum_{l=0}^{M}\pi_l\left(((\delta z_l)^T(\nabla^2 p_l^k)\delta z_l + (\nabla p_l^k)^T\delta^2 z_l)\cdot\ln\left(\frac{p_l^k}{\sum_{j=0}^M \pi_j p_j^k}\right) + \frac{((\nabla p_l^k)^T\delta z_l)^2}{p_l^k}\right) \\ &\quad - \frac{\epsilon^2}{2}\sum_{k=1}^{K}\frac{(\sum_{l=0}^M \pi_l(\nabla p_l^k)^T\delta z_l)^2}{\sum_{l=0}^M \pi_l p_l^k} + o(\epsilon^3), \end{aligned} \tag{26}$$

as $\epsilon \to 0$.

Proof. By definition,

$$JS_\pi(\tilde{p}_0,\dots,\tilde{p}_M) := H\left(\sum_{l=0}^M \pi_l\tilde{p}_l\right) - \sum_{l=0}^M \pi_l H(\tilde{p}_l) \tag{27}$$

$$= -\sum_{k=1}^K\left(\sum_{l=0}^M \pi_l\tilde{p}_l^k\right)\ln\left(\sum_{l=0}^M \pi_l\tilde{p}_l^k\right) + \sum_{l=0}^M \pi_l\sum_{k=1}^K \tilde{p}_l^k\ln(\tilde{p}_l^k) \tag{28}$$

$$\begin{aligned} &= -\sum_{k=1}^K\left(\sum_{l=0}^M \pi_l p^k(z_l+R_l(\epsilon))\right)\ln\left(\sum_{l=0}^M \pi_l p^k(z_l+R_l(\epsilon))\right) \\ &\quad + \sum_{l=0}^M \pi_l\sum_{k=1}^K p^k(z_l+R_l(\epsilon))\ln(p^k(z_l+R_l(\epsilon))) \end{aligned} \tag{29}$$

$$= \sum_{l=0}^M\sum_{k=1}^K \pi_l A_l^k(\epsilon), \tag{30}$$

where

$$A_l^k(\epsilon) := p^k(z_l+R_l(\epsilon))\ln\left(\frac{p^k(z_l+R_l(\epsilon))}{\sum_j \pi_j p^k(z_j+R_j(\epsilon))}\right). \tag{31}$$

Then, using twice-differentiability of p, we perform a second-order Taylor expansion on (31) in the small parameter ϵ:

$$A_l^k(\epsilon) = A_l^k(0) + \epsilon(A_l^k)'(0) + \frac{1}{2}\epsilon^2(A_l^k)''(0) + o(\epsilon^2), \tag{32}$$

as $\epsilon \to 0$.

It remains to compute $(A_l^k)'(0)$ and $(A_l^k)''(0)$.

Using chain rule, we obtain:

$$\begin{aligned}(A_l^k)'(\epsilon) &= (\nabla p^k(z_l + R_l(\epsilon)))^T R_l'(\epsilon)\left(1 + \ln\left(\frac{p^k(z_l + R_l(\epsilon))}{\sum_j \pi_j p^k(z_j + R_j(\epsilon))}\right)\right) \\ &\quad - p^k(z_l + R_l(\epsilon))\frac{\sum_j \pi_j (\nabla p^k(z_j + R_j(\epsilon)))^T R_j'(\epsilon)}{\sum_l \pi_l p^k(z_l + R_l(\epsilon))}. \end{aligned} \tag{33}$$

Therefore, using the fact that $R_i(0) = 0$ and $R_i'(0) = \delta z_i$, we have:

$$(A_l^k)'(0) = (\nabla p^k(z_l))^T \delta z_l \left(1 + \ln\left(\frac{p^k(z_l)}{\sum_j \pi_j p^k(z_j)}\right)\right) - p^k(z_l)\frac{\sum_j \pi_j (\nabla p^k(z_j))^T \delta z_j}{\sum_l \pi_l p^k(z_l)}. \tag{34}$$

Similarly, applying chain rule, we obtain:

$$\begin{aligned}&(A_l^k)''(\epsilon) \\ &= [R_l'(\epsilon)^T \nabla^2 p^k(z_l + R_l(\epsilon)) R_l'(\epsilon) \\ &\quad + (\nabla p^k(z_l + R_l(\epsilon)))^T R_l''(\epsilon)]\left(1 + \ln\left(\frac{p^k(z_l + R_l(\epsilon))}{\sum_j \pi_j p^k(z_j + R_j(\epsilon))}\right)\right) \\ &\quad + \frac{(\nabla p^k(z_l + R_l(\epsilon))^T R_l'(\epsilon))^2}{p^k(z_l + R_l(\epsilon))} - \frac{2\nabla p^k(z_l + R_l(\epsilon))^T R_l'(\epsilon)\left(\sum_j \pi_j \nabla p^k(z_j + R_j(\epsilon))^T R_j'(\epsilon)\right)}{\sum_j \pi_j p^k(z_j + R_j(\epsilon))} \\ &\quad - p^k(z_l + R_l(\epsilon))\frac{\sum_j \pi_j (R_j'(\epsilon)^T \nabla^2 p^k(z_j + R_j(\epsilon)) R_j'(\epsilon) + \nabla p^k(z_j + R_j(\epsilon))^T R_j''(\epsilon))}{\sum_l \pi_l p^k(z_l + R_l(\epsilon))} \\ &\quad + p^k(z_l + R_l(\epsilon))\left(\frac{\sum_l \pi_l \nabla p^k(z_l + R_l(\epsilon))^T R_l'(\epsilon)}{\sum_j \pi_j p^k(z_j + R_j(\epsilon))}\right)^2. \end{aligned} \tag{35}$$

Therefore, using the fact that $R_i(0) = 0$, $R_i'(0) = \delta z_i$, and $R_i''(0) = \delta^2 z_i$, we have:

$$\begin{aligned}(A_l^k)''(0) &= [(\delta z_l)^T \nabla^2 p^k(z_l)\delta z_l + (\nabla p^k(z_l))^T \delta^2 z_l]\left(1 + \ln\left(\frac{p^k(z_l)}{\sum_j \pi_j p^k(z_j)}\right)\right) \\ &\quad + \frac{(\nabla p^k(z_l)^T \delta z_l)^2}{p^k(z_l)} - \frac{2\nabla p^k(z_l)^T \delta z_l \left(\sum_j \pi_j \nabla p^k(z_j)^T \delta z_j\right)}{\sum_j \pi_j p^k(z_j)} \\ &\quad - p^k(z_l)\frac{\sum_j \pi_j ((\delta z_j)^T \nabla^2 p^k(z_j)\delta z_j + \nabla p^k(z_j)^T \delta^2 z_j)}{\sum_l \pi_l p^k(z_l)} \\ &\quad + p^k(z_l)\left(\frac{\sum_l \pi_l \nabla p^k(z_l)^T \delta z_j}{\sum_j \pi_j p^k(z_j)}\right)^2. \end{aligned} \tag{36}$$

Combining the above results with the Taylor expansion (32) and simplifying the resulting expression lead to (26).

□

Specializing Proposition 1 to our setting, we arrive at the following result.

Theorem 2. Let $\epsilon > 0$ be a small parameter, and assume that p is twice differentiable. Define, for $k \in [K]$,

$$q^k(x, y) := \nabla^2 p^k(x) \ln\left(\frac{2p^k(x)}{p^k(x) + p^k(y)}\right) + \nabla p^k(x)(\nabla p^k(x))^T \frac{p^k(y)}{p^k(x)(p^k(x) + p^k(y))}. \tag{37}$$

Then,

$$L_n^{JSD} = \tilde{L}_n^{std} + \epsilon S_1 + \epsilon^2 \tilde{S}_2 + \epsilon^2 \phi(\epsilon), \tag{38}$$

with

$$\tilde{S}_2 = S_2 + \sigma_{add}^2 S_2^{add} + \sigma_{mult}^2 S_2^{mult}, \tag{39}$$

where

$$\begin{aligned} S_1 = \frac{\mathbb{E}_\lambda[1-\lambda]}{2n} \sum_{i=1}^n \sum_{k=1}^K \Bigg((\nabla p^k(x_i))^T (\mathbb{E}_{x_r}[x_r] - x_i) \cdot \ln\left(\frac{2p^k(x_i)}{p^k(x_i) + p^k(A(x_i))} \right) \\ + (\nabla p^k(A(x_i)))^T (\mathbb{E}_{x_r}[A(x_r)] - A(x_i)) \cdot \ln\left(\frac{2p^k(A(x_i))}{p^k(x_i) + p^k(A(x_i))} \right) \Bigg) \end{aligned} \tag{40}$$

$$\begin{aligned} S_2 = \frac{\mathbb{E}_\lambda[(1-\lambda)^2]}{4n} \sum_{i=1}^n \sum_{k=1}^K \Bigg(\mathbb{E}_{x_r}[(x_r - x_i)^T q^k(x_i, A(x_i))(x_r - x_i)] \\ + \mathbb{E}_{x_r}[(A(x_r) - A(x_i))^T q^k(A(x_i), x_i)(A(x_r) - A(x_i))] \\ - \frac{2(\nabla p^k(x_i))^T \mathbb{E}_{x_r}[(x_r - x_i)(A(x_r) - A(x_i))^T] \nabla p^k(A(x_i))}{p^k(x_i) + p^k(A(x_i))} \Bigg), \end{aligned} \tag{41}$$

$$\begin{aligned} S_2^{add} = \frac{1}{4n} \sum_{i=1}^n \sum_{k=1}^K \Bigg(\mathbb{E}_{\boldsymbol{\xi}}[(\xi^{add})^T q^k(x_i, A(x_i)) \xi^{add}] + \mathbb{E}_{\boldsymbol{\xi}}[(\xi^{add})^T q^k(A(x_i), x_i) \xi^{add}] \\ - \frac{2(\nabla p^k(x_i))^T \mathbb{E}_{\xi}[\xi^{add} (\xi^{add})^T] \nabla p^k(A(x_i))}{p^k(x_i) + p^k(A(x_i))} \Bigg), \end{aligned} \tag{42}$$

$$\begin{aligned} S_2^{mult} = \frac{1}{4n} \sum_{i=1}^n \sum_{k=1}^K \Bigg(\mathbb{E}_{\boldsymbol{\xi}}[(\xi^{mult} \odot x_i)^T q^k(x_i, A(x_i))(\xi^{mult} \odot x_i)] \\ + \mathbb{E}_{\boldsymbol{\xi}}[(\xi^{mult} \odot A(x_i))^T q^k(A(x_i), x_i)(\xi^{mult} \odot A(x_i))] \\ - \frac{2(\nabla p^k(x_i))^T \mathbb{E}_{\xi}[(\xi^{mult} \odot x_i)(\xi^{mult} \odot A(x_i))^T] \nabla p^k(A(x_i))}{p^k(x_i) + p^k(A(x_i))} \Bigg), \end{aligned} \tag{43}$$

and $\phi(\epsilon) = \mathbb{E}_{\lambda \sim \tilde{\mathcal{D}}_\lambda} \mathbb{E}_{x_r \sim \mathcal{D}_x} \mathbb{E}_{\boldsymbol{\xi} \sim \mathcal{Q}}[\varphi(\epsilon)]$, with φ some function such that $\lim_{\epsilon \to 0} \varphi(\epsilon) = 0$.

Proof. By assumption, p is twice differentiable, so we apply Proposition 1 (with $M = 1$, $K = 2$, $\pi_0 = \pi_1 = 1/2$, $z_0 = x_i$, $z_1 = A(x_i)$, $R_0(\epsilon) = \epsilon(e_i^{mixup} + e_i^{noise}) + \epsilon^2 \sigma_{mult} \xi^{mult} \odot e_i^{mixup}$, and $R_1(\epsilon) = \epsilon(\tilde{e}_i^{mixup} + \tilde{e}_i^{noise}) + \epsilon^2 \sigma_{mult} \xi^{mult} \odot \tilde{e}_i^{mixup}$) to Eq. (7). Simplifying (using linearity of expectation, and the expression for the e_i^{mixup}, $\tilde{e}_i^{mixup}$, e_i^{noise} and $\tilde{e}_i^{noise}$given in Lemma 2) and rearranging the resulting expression then lead to the expression given in the theorem. □

Theorem 2 implies that, when compared to the stability training in AugMix, the stability training in *NoisyMix* introduces additional smoothness, regularizing the directional derivatives, $\nabla p(x_i)$, $\nabla p(A(x_i))$, $\nabla^2 p(x_i)$ and $\nabla^2 p(A(x_i))$, with respect to both the clean data x_i and augmented data $A(x_i)$, according to the mixing coefficient and the noise levels.

Combining this with the analogous interpretation of Theorem 1 and recalling that $L_n^{NoisyMix} = L_n^{NFM} + \gamma L_n^{stability}$, we see that *NoisyMix* amplifies the regularizing effects of AugMix, and can lead to smooth decision boundaries and improvement in model robustness. Since *NoisyMix* implicitly reduces the $\nabla p(x_i)$ and $\nabla p(A(x_i))$, following the argument in Section C of Lim et al. (2022), we see that *NoisyMix* implicitly increases the classification margin, thereby making the model more robust to input perturbations. We shall study another lens in which *NoisyMix* can improve robustness in the next section.

C Robustness of NoisyMix

We now demonstrate how *NoisyMix* training can lead to a model that is both adversarially robust and stable.

We consider the binary cross-entropy loss, setting $h(z) = \log(1 + e^z)$, with the labels y taking value in $\{0, 1\}$ and the classifier model $f : \mathbb{R}^d \to \mathbb{R}$. In the following, we assume that the model parameter $\theta \in \Theta := \{\theta : y_i f(x_i) + (y_i - 1) f(x_i) \geq 0 \text{ for all } i \in [n]\}$. We remark that this set contains the set of all parameters with correct classifications of training samples (before applying *NoisyMix*), since $\{\theta : 1_{\{f(x_i) \geq 0\}} = y_i \text{ for all } i \in [n]\} \subset \Theta$. Therefore, the condition of $\theta \in \Theta$ is fulfilled when the model classifies all labels correctly for the training data before applying *NoisyMix*. Since the training error often becomes zero in finite time in practice, we shall study the effect of *NoisyMix* on model robustness in the regime of $\theta \in \Theta$.

Working in the data-dependent parameter space Θ, we obtain the following result.

Theorem 3. Let $\theta \in \Theta := \{\theta : y_i f(x_i) + (y_i - 1) f(x_i) \geq 0 \text{ for all } i \in [n]\}$ be a point such that $\nabla f(x_i)$, $\nabla f(A(x_i))$, $\nabla^2 f(x_i)$ and $\nabla^2 f(A(x_i))$ exist for all $i \in [n]$. Assume that $f(x_i) = \nabla f(x_i)^T x_i$, $\nabla^2 f(x_i) = 0$ for all $i \in [n]$. In addition, suppose that $\mathbb{E}_{r \sim \mathcal{D}_x}[r] = \mathbb{E}_{r \sim \mathcal{D}_x}[A(r)] = 0$ and $\|x_i\|_2 \geq c_x \sqrt{d}$ for all $i \in [n]$. Then,

$$L_n^{NoisyMix} \geq \frac{1}{n} \sum_{i=1}^{n} \max_{\|\delta_i\|_2 \leq \epsilon_i^{mix}} l(f(x_i + \delta_i), y_i) + \gamma \tilde{L}_n^{JSD} + \epsilon^2 L_n^{reg} + \epsilon^2 \phi(\epsilon), \tag{44}$$

where

$$\epsilon_i^{mix} = \epsilon \mathbb{E}_{\lambda \sim \tilde{\mathcal{D}}_\lambda}[1 - \lambda] \cdot c_x \sqrt{d}\, |\cos(\nabla f(x_i), x_i)|, \tag{45}$$

$$\tilde{L}_n^{JSD} = \frac{1 - \epsilon \mathbb{E}_{\lambda \sim \tilde{\mathcal{D}}_\lambda}[1 - \lambda]}{n} \sum_{i=1}^{n} JS_\pi(f(x_i), f(A(x_i))), \tag{46}$$

$$L_n^{reg} = \frac{1}{2n} \sum_{i=1}^{n} |h''(f(x_i))| (\epsilon_i^{reg})^2 + \gamma \tilde{S}_2, \tag{47}$$

with $\tilde{S}_2$ given in (39) and

$$\begin{aligned}(\epsilon_i^{reg})^2 = \|\nabla f(x_i)\|_2^2 \Big(& \mathbb{E}_\lambda[(1 - \lambda)]^2 \mathbb{E}_{x_r}[\|x_r\|_2^2 \cos(\nabla f(x_i), x_r)^2] \\ & + \sigma_{add}^2 \mathbb{E}_{\boldsymbol{\xi}}[\|\xi_{add}\|_2^2 \cos(\nabla f(x_i), \xi_{add})^2] \\ & + \sigma_{mult}^2 \mathbb{E}_{\boldsymbol{\xi}}[\|\xi_{mult} \odot x_i\|_2^2 \cos(\nabla f(x_i), \xi_{mult} \odot x_i)^2] \Big),\end{aligned} \tag{48}$$

and ϕ is some function such that $\lim_{\epsilon \to 0} \phi(\epsilon) = 0$.

Proof of Theorem 3. The inequality in the theorem follows upon applying Theorem 2 in Lim et al. (2022) together with Theorem 2. □

We remark that the assumption made in Theorem 3 is similar to the one made in Lamb et al. (2019); Zhang et al. (2020), and is satisfied by linear models and deep neural networks with ReLU activation function and max-pooling (for a proof of this, we refer to Section B.2 in Zhang et al. (2020)).

Theorem 3 says that $L_n^{NoisyMix}$ is approximately an upper bound of sum of an adversarial loss with l_2-attack of size $\epsilon^{mix} = \min_i \epsilon_i^{mix}$, a stability objective, and a data-dependent regularizer. Therefore, *minimizing the NoisyMix loss would result in a small regularized adversarial loss, while promoting stability for the model.* This not only amplifies the robustness benefits of Manifold Mixup, but also imposes additional smoothness, due to the noise injection in NFM and stability training on the noise-perturbed transformed data. The latter can also help reduce robust overfitting and improve test performance (Rice et al., 2020; Rebuffi et al., 2021).

D Additional Experiments and Details

In this section we show additional results to support and complement the findings in Section 4.

D.1 Additional ImageNet-C, CIFAR-10-C and CIFAR-100-C Results

ImageNet-C, CIFAR-10-C, and CIFAR-100-C consist of 15 different corruption types that are used to evaluate the robustness of a given model. These corruption types are illustrated in Figure 5. Here we show examples that corresponds to severity level 5, i.e., the most severe perturbation level. The considered corruptions cover a wide verity of perturbations that can potentially occur in real-world situations (i.e., a self-driving car might need to navigate in a snow storm).

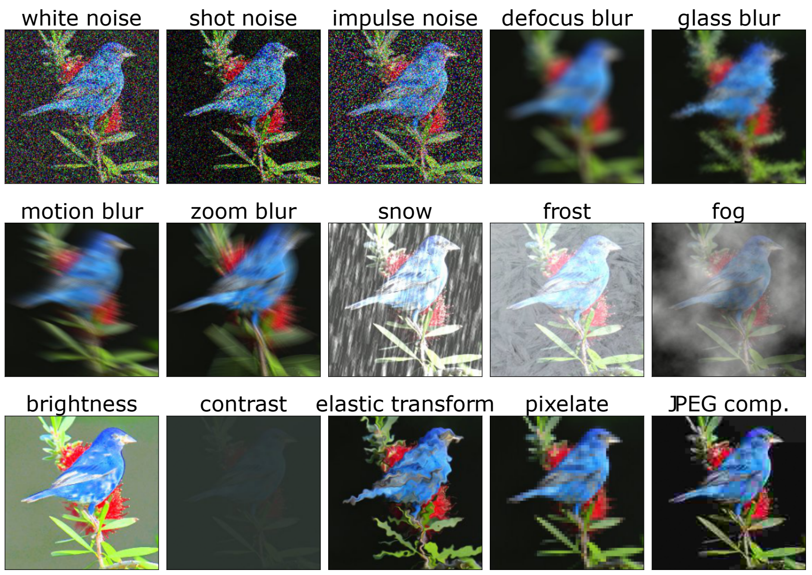


Figure 5: The ImageNet-C example shows the 15 different corruption types at severity level 5.

ImageNet-C Results. Table 6 shows the robust accuracy with respect to each perturbation type for the different ResNet-50 models trained on ImageNet-1k. *NoisyMix* performs best on 9 out of the 15 corruption types. The advantage is particularly pronounced for noise perturbations and weather corruptions. To illustrate this further, we compute the average robustness accuracy for the 4 meta categories (i.e., noise, weather, blur, and digital corruptions). The results are shown in Figure 6, where it can be seen that *NoisyMix* is the most robust model with respect to noise, weather, and digital corruptions, while AugMix is slightly more robust to blur corruptions.

CIFAR-C Results. Table 7 shows the detailed robustness properties of ResNet-18 models trained with different data augmentation schemes and evaluated on CIFAR-10-C. Table 8 shows similar results for ResNet-18 models that are evaluated on CIFAR-100-C. Overall, *NoisyMix* performs best across most of the different input perturbations and data corruptions as compared to various baselines in the data augmentation space. In general, the combination of a noisy training scheme paired with stability training appears to prescribe our model a bias towards stochastic noise due to the nature of fitting a reconstruction over a stochastic representation. This can be seen by the good robust accuracy on the noise, weather and digital perturbations being higher for *NoisyMix* and lower in the other methods that place less encompassing emphasis on stochastic perturbations. Overall, the advantage of training models with *NoisyMix* is more pronounced for the CIFAR-100 task, where *NoisyMix* dominates AugMix.

The limitations of certain neural network architectures in terms of their abilities to learn high-frequency features in low-dimensional domains (Tancik et al., 2020) could also imply that our model will tend towards learning features similar in frequency to the in-domain task, and hence perform most accurately in specific out-of-domain perturbations on the same dataset containing similar frequencies. We leave this to future work.

Table 6: Detailed results for ResNet-50s trained with various data augmentation schemes and evaluated on ImageNet-C. All values indicate the robust accuracy (in %) as a function of the considered 15 perturbation types (higher values are better).

	Noise			Blur				Weather				Digital			
	White	Shot	Impulse	Defocus	Glass	Motion	Zoom	Snow	Frost	Fog	Bright	Contrast	Elastic	Pixel	JPEG
Baseline	29.3	27.0	23.8	38.8	26.8	38.7	36.2	32.5	38.1	45.8	68.0	39.1	45.3	44.8	53.4
Adversarial Trained	22.4	21.0	13.1	24.6	34.8	33.7	35.2	30.6	29.6	6.2	54.4	8.5	49.4	57.3	60.5
Stylized ImageNet	41.3	40.3	37.2	42.9	32.3	45.5	35.8	41.0	41.6	47.0	67.4	43.3	49.5	55.6	57.7
Fast AutoAugment	40.9	40.3	36.3	42.0	35.1	41.2	36.3	40.2	43.0	52.9	**72.0**	51.7	47.5	48.5	57.7
Mixup	40.5	36.9	34.2	41.9	29.1	43.9	41.1	42.0	45.5	**57.5**	71.4	51.0	47.1	51.8	58.6
Manifold Mixup	36.3	34.2	31.0	39.7	27.6	42.0	40.2	38.6	49.0	54.6	69.1	51.3	45.5	45.7	54.2
CutMix	36.0	34.1	28.4	40.2	25.4	40.6	36.9	34.1	39.6	50.3	70.3	46.8	45.2	36.4	51.2
Puzzle Mix	22.4	21.0	13.1	24.6	34.8	33.7	35.2	30.6	29.6	6.2	54.4	8.5	49.4	**57.3**	60.5
AugMix	40.6	41.1	37.7	**47.7**	34.9	**53.5**	**49.0**	39.9	43.8	47.1	69.5	51.1	52.0	57.0	60.3
NoisyMix (ours)	**53.1**	**52.5**	**51.7**	47.1	**37.5**	52.2	47.2	**45.0**	**52.6**	52.9	71.1	**52.6**	**52.6**	53.2	**63.8**

Table 7: Detailed results for ResNet-18s trained with various data augmentation schemes and evaluated on CIFAR-10-C. All values indicate the robust accuracy (in %) as a function of the considered 15 perturbation types (higher values are better).

	Noise			Blur				Weather				Digital			
	White	Shot	Impulse	Defocus	Glass	Motion	Zoom	Snow	Frost	Fog	Bright	Contrast	Elastic	Pixel	JPEG
Baseline	46.4	59.2	52.6	82.6	53.1	78.0	77.5	82.6	78.3	88.4	93.8	76.7	84.4	74.5	79.3
Mixup	58.6	68.3	53.9	88.2	69.2	84.0	84.0	88.5	87.9	91.2	94.4	88.5	87.6	79.8	82.1
Manifold Mixup	55.5	66.4	48.2	85.7	60.5	82.3	82.5	87.3	84.9	90.7	94.5	85.1	86.2	75.3	80.9
CutMix	29.5	42.2	50.2	83.7	58.7	81.0	78.4	87.7	80.9	90.7	94.8	83.5	86.4	75.3	71.5
Puzzle Mix	76.5	81.7	69.0	86.4	77.5	83.3	82.0	90.2	90.1	91.6	**95.3**	87.2	88.8	85.3	77.0
NFM	78.7	83.5	73.4	85.4	72.0	80.3	83.2	89.0	89.5	87.1	94.2	76.2	86.2	82.5	87.9
AugMix	79.8	85.2	85.5	**94.3**	79.6	**92.4**	**93.3**	89.7	89.4	**92.0**	94.6	**90.7**	90.6	87.7	87.7
NoisyMix (ours)	**89.9**	**91.8**	**94.6**	93.6	**85.9**	91.6	92.7	**91.1**	**91.7**	91.1	94.5	88.1	**90.9**	**89.3**	**90.7**

Table 8: Detailed results for ResNet-18s trained with various data augmentation schemes and evaluated on CIFAR-100-C. All values indicate the robust accuracy (in %) as a function of the considered 15 perturbation types (higher values are better).

	Noise			Blur				Weather				Digital			
	White	Shot	Impulse	Defocus	Glass	Motion	Zoom	Snow	Frost	Fog	Bright	Contrast	Elastic	Pixel	JPEG
Baseline	22.1	30.3	24.6	60.2	21.3	54.5	53.8	54.8	48.9	64.5	73.4	55.1	60.6	51.8	50.6
Mixup	27.2	36.0	21.7	65.9	25.8	62.3	60.7	63.7	57.4	70.2	75.9	67.9	64.8	58.7	55.5
Manifold Mixup	24.7	33.6	27.0	65.1	23.1	60.8	60.3	63.8	57.3	69.2	**76.7**	62.8	64.6	57.2	54.2
CutMix	13.1	20.6	29.8	62.4	22.3	56.0	56.5	59.8	49.6	65.4	74.2	56.5	61.6	45.4	45.7
Puzzle Mix	47.3	54.3	40.1	64.1	33.1	59.6	59.0	64.5	59.4	68.0	75.7	60.7	64.4	56.6	54.4
NFM	50.2	57.5	42.1	64.8	34.9	59.8	61.0	66.0	64.7	65.8	76.0	57.1	65.3	64.7	64.8
AugMix	46.8	54.9	61.5	74.9	50.9	72.2	73.0	65.9	63.7	67.1	74.2	67.8	68.3	66.3	61.5
NoisyMix (ours)	**65.3**	**69.2**	**77.8**	**76.8**	**58.6**	**73.7**	**75.4**	**71.1**	**69.7**	**71.0**	**76.7**	**68.0**	**72.2**	**70.5**	**68.9**

Table 9: Detailed results for Wide-ResNet models trained with data augmentation schemes and evaluated on CIFAR-10-C. All values indicate the robust accuracy (in %) as a function of the considered 15 perturbation types (higher values are better).

	Noise			Blur				Weather				Digital			
	White	Shot	Impulse	Defocus	Glass	Motion	Zoom	Snow	Frost	Fog	Bright	Contrast	Elastic	Pixel	JPEG
Baseline	46.5	59.6	50.7	83.0	57.3	79.0	78.4	85.1	81.0	89.2	94.6	77.4	85.4	76.5	79.4
Mixup	56.5	67.9	53.4	88.7	69.8	84.9	85.4	90.4	90.2	91.9	95.3	87.9	88.3	78.8	83.3
Manifold Mixup	48.0	61.1	56.4	85.7	60.1	81.6	82.3	89.0	85.9	91.4	95.4	83.3	86.4	76.3	81.7
CutMix	44.2	36.3	46.6	83.2	57.0	80.7	76.4	87.7	81.3	91.6	95.1	85.5	85.3	72.6	70.4
Puzzle Mix	72.6	79.1	58.6	85.9	72.5	83.7	80.8	91.2	89.8	92.8	**95.8**	89.5	88.2	82.8	75.8
NFM	74.8	81.2	71.4	84.7	69.6	78.9	81.7	89.7	89.5	88.2	95.0	73.9	86.8	81.9	87.9
AugMix	81.0	86.3	87.9	**95.5**	80.9	**93.9**	**94.5**	91.4	91.0	93.3	95.7	**92.1**	91.8	89.4	88.4
NoisyMix (ours)	**87.0**	**90.2**	**95.7**	95.3	**84.2**	93.4	94.4	**92.3**	**92.2**	**93.4**	**95.8**	89.8	**92.5**	**89.5**	**90.8**

Table 10: Detailed results for Wide-ResNet models trained with data augmentation schemes and evaluated on CIFAR-100-C. All values indicate the robust accuracy (in %) as a function of the considered 15 perturbation types (higher values are better).

	Noise			Blur				Weather				Digital			
	White	Shot	Impulse	Defocus	Glass	Motion	Zoom	Snow	Frost	Fog	Bright	Contrast	Elastic	Pixel	JPEG
Baseline	18.4	27.0	19.1	60.6	18.4	54.0	54.5	55.7	48.1	64.4	74.9	54.0	60.4	51.9	50.2
Mixup	25.3	34.3	21.0	66.5	23.6	61.8	61.3	64.5	56.1	71.1	77.1	68.2	65.4	56.3	55.8
Manifold Mixup	20.5	29.7	24.2	65.7	23.9	60.9	60.9	64.8	56.8	70.6	**78.2**	63.3	65.1	57.2	55.3
CutMix	13.1	20.8	31.2	62.2	22.6	57.0	55.7	61.2	52.4	67.8	75.4	59.2	61.2	43.7	44.1
Puzzle Mix	38.6	47.7	33.0	64.4	29.4	60.5	58.7	66.3	61.2	71.2	77.5	65.0	64.9	54.1	51.9
NFM	49.4	57.2	39.4	62.9	35.8	57.3	58.3	66.5	65.5	66.0	76.7	55.4	64.3	62.2	63.8
AugMix	46.6	55.5	65.2	**78.2**	49.3	**74.8**	**76.1**	69.0	66.1	71.5	77.6	**69.9**	71.2	66.9	63.2
NoisyMix (ours)	**68.4**	**71.9**	**80.0**	77.2	**59.5**	73.8	75.6	**71.7**	**70.7**	**72.2**	77.7	67.1	**72.7**	**70.8**	**69.3**

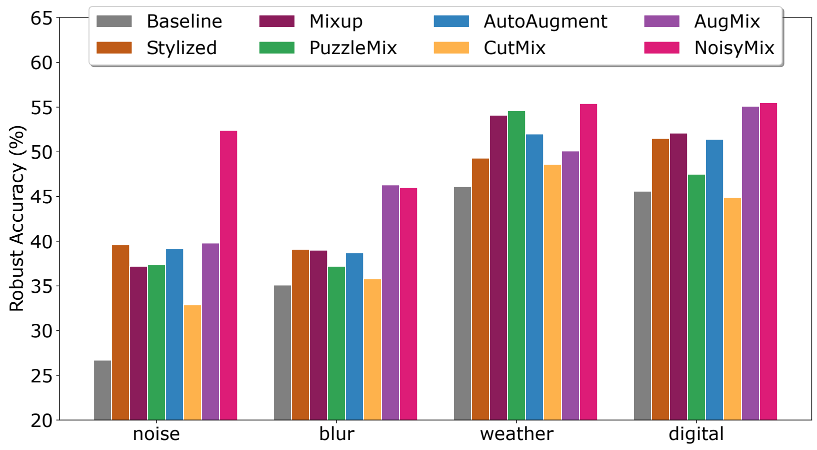


Figure 6: Average robust accuracy of ResNet-50 models for the 4 meta categories noise, blur, weather, and digital distortions. *NoisyMix* dominates in 2 out of the 4 categories, while the robustness to digital corruptions is nearly on par with AugMix.

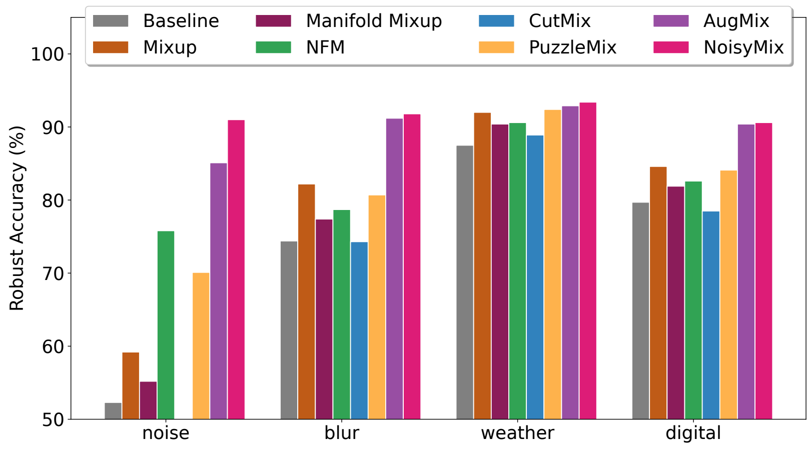


Figure 7: Average robust accuracy of Wide-ResNet-28x2 models, trained on CIFAR-10, for the 4 categories noise, blur, weather, and digital distortions. *NoisyMix* dominates in 1 out of the 4 categories, while the robustness to blur, weather, and digital corruptions is nearly on par with AugMix.

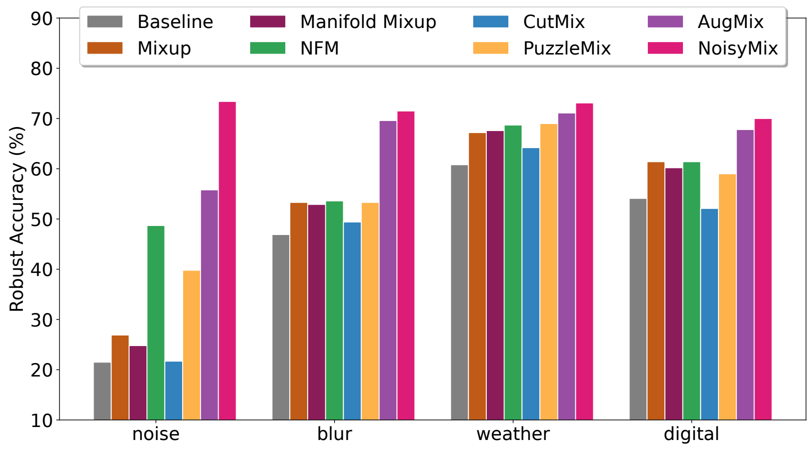


Figure 8: Average robust accuracy of Wide-ResNet-28x2 models, trained on CIFAR-100, for the 4 categories noise, blur, weather, and digital distortions. *NoisyMix* dominates in 4 out of the 4 categories.

Table 9 shows the detailed robustness properties of Wide-ResNet-28x2 models evaluated on CIFAR-10-C, and

Table 10 results for CIFAR-100-C, respectively. *NoisyMix* benefits from the wide architecture and can improve the robustness as compared to the ResNet-18 models. Again, it can be seen that *NoisyMix* significantly improves the robustness to noise perturbations as well as to weather corruptions. Figure 7, and Figure 8 show the average robustness accuracy for the 4 meta categories (i.e., noise, weather, blur, and digital corruptions) for CIFAR-10-C and CIFAR-100-C, respectively.

D.2 Calibration Results for ImageNet-R

State-of-the-art neural networks tend to be biased towards high accuracy prediction, while being poorly calibrated. In turn, these models tend to be less reliable and often also show to lack fairness. This can lead to poor decision making in mission critical applications, e.g., medical imaging. In such applications it might be not sufficient to just compute a score $s(x)$ that can then be used to rank the input from the most probable member to the least probable member of a class c. Rather it is desirable to obtain an reliable estimate for the class membership probability that can be assigned an example-dependent misclassifications cost (Zadrozny and Elkan, 2001). Typically, these probability estimates are obtained by squashing the activations of the output layer through a normalized exponential function that is known as the softmax function.

One hypothesis that explains why DNNs (such as ResNets) are poorly calibrated is their high capacity, i.e., these models have the ability to learn highly-specific features that overfit to the source dataset. This phenomenom is shown by Guo et al. (2017), where calibration error increases as the number of filters per layer increases. We show that data augmentation can mitigate this effect and produce better calibrated models. Table 11 shows the RMS calibration error and the area under the response rate accuracy curve (AURRA), evaluated on IamgeNet-R, as metrics to measure how well a model is calibrated. *NoisyMix* considerably improves the calibration of the ImageNet model, as compared to standard training or training with other data augmentation methods. We believe that this is explained by the fact that *NoisyMix* "wash-out" features that are highly task specific.

D.3 Additional ImageNet-P and CIFAR-10-P Results

ImageNet-C Results. Table 12 shows the flip rates with respect to each perturbation type for the different ResNet-50 models trained on ImageNet-1k. *NoisyMix* dominates on 6 out of the 10 corruption types, resulting in the lowest mean flip rate overall. While *NoisyMix* shows a clear advantage compared to the model trained with AugMix, the adversarial trained model is performing nearly as good as *NoisyMix* on this task. This is surprising, since the adversarial trained model shows a poor performance on the ImageNet-C task. However, this shows the many different facets of robustness and supports the claim that there is no single metric for measuring robustness.

Table 13 and 14 show results for ResNet-18 and Wide-ResNet-28x2 models trained on CIFAR-10 and evaluated on CIFAR-10-P. Here, the models trained with *NoisyMix* clearly dominate. Note, that we do not normalize the flip rates here.

Table 11: Calibration errors measured by the RMS calibration error (lower values are better), and the area under the response rate accuracy curve (higher values are better).

	RMS Calibration Error (↓%)	AURRA (↑%)
Baseline ResNet-50	19.7	64.6
Adversarial Trained (Salman et al., 2020)	8.9	68.8
Stylized ImageNet (Geirhos et al., 2018)	16.2	69.7
AutoAugment (Cubuk et al., 2019)	19.9	67.6
Mixup (Zhang et al., 2018)	17.5	68.6
Manifold Mixup (Verma et al., 2019)	5.0	68.7
CutMix (Yun et al., 2019)	13.6	63.2
Puzzle Mix (Kim et al., 2020)	15.8	68.8
AugMix (Hendrycks et al., 2020)	14.5	70.1
NoisyMix (ours)	**3.9**	**74.8**

Table 12: Detailed results for ResNet-50s trained with various data augmentation schemes and evaluated on ImageNet-P. The values indicate the flipping rates as a function of the 10 perturbation types (lower values are better). The mean flip rate (mFR) summarizes the results and shows that *NoisyMix* improves perturbation stability by approximately 5%.

	Noise		Blur		Weather		Digital				
	White	Shot	Motion	Zoom	Snow	Bright	Translate	Rotate	Tilt	Scale	mFR
Baseline	59.4	57.8	64.5	72.1	63.2	61.9	44.2	51.9	56.9	48.1	58.0
Adversarial Trained	25.6	40.1	**26.9**	**33.1**	**18.3**	60.6	25.7	31.3	30.5	40.4	33.3
Stylized ImageNet	31.1	28.8	41.8	58.1	43.1	50.3	36.7	41.2	42.9	42.1	54.4
Fast AutoAugment	53.7	49.9	63.8	79.3	66.5	58.2	43.2	49.0	57.7	43.6	56.6
Mixup	47.6	47.2	60.8	73.9	58.9	62.1	51.2	55.0	61.1	46.3	56.4
Manifold Mixup	49.1	46.2	66.1	72.4	62.6	59.9	46.5	53.7	58.8	45.2	56.0
CutMix	54.7	52.7	69.4	77.4	73.1	59.2	43.2	51.6	59.1	46.1	58.6
Puzzle Mix	50.1	48.3	64.6	72.5	63.3	56.7	44.3	50.4	57.9	46.6	55.5
AugMix	40.9	37.8	30.0	54.8	37.3	46.1	26.9	32.3	36.4	33.8	37.6
NoisyMix (ours)	**25.1**	**22.8**	30.2	41.5	32.4	**34.1**	**20.5**	**24.5**	**27.7**	**26.3**	**28.5**

Table 13: Detailed results for ResNet-18s trained with various data augmentation schemes and evaluated on CIFAR-10-P. The values indicate the flipping rates as a function of the 10 perturbation types (lower values are better). The mean flip rate (mFR) summarizes the results and shows that *NoisyMix* improves perturbation stability by approximately 1%.

	Noise		Blur		Weather		Digital				
	White	Shot	Motion	Zoom	Snow	Bright	Translate	Rotate	Tilt	Scale	mFR
Baseline	44.6	27.6	9.2	0.4	2.5	0.5	2.1	2.8	0.8	3.1	9.4
Mixup	26.3	17.2	6.1	0.4	1.8	0.6	2.2	2.4	0.8	2.5	6.0
Manifold Mixup	33.4	21.6	7.8	0.4	2.1	0.6	2.2	2.6	0.8	2.9	7.4
NFM	11.5	7.5	6.8	0.3	1.5	**0.4**	1.8	2.0	0.6	2.2	3.5
CutMix	72.3	50.8	9.1	0.5	1.9	0.6	1.9	2.3	0.8	3.1	14.3
Puzzle Mix	16.1	11.2	6.3	0.3	1.4	**0.4**	2.3	1.8	0.6	2.5	4.3
AugMix	11.4	7.1	2.4	0.2	1.0	**0.4**	1.9	1.2	0.5	1.8	2.8
NoisyMix (ours)	**6.6**	**4.5**	**2.3**	**0.1**	**0.9**	**0.4**	**1.6**	**1.0**	**0.4**	**1.4**	**1.9**

Table 14: Detailed results for Wide-ResNet-28x2 trained with various data augmentation schemes and evaluated on CIFAR-10-P. The values indicate the flipping rates as a function of the 10 perturbation types (lower values are better). The mean flip rate (mFR) summarizes the results and shows that *NoisyMix* improves perturbation stability by approximately 5%.

	Noise		Blur		Weather		Digital				
	White	Shot	Motion	Zoom	Snow	Bright	Translate	Rotate	Tilt	Scale	mFR
Baseline	42.7	27.6	8.2	0.3	2.1	0.4	1.4	2.4	0.6	2.6	8.8
Mixup	30.9	18.5	5.7	0.3	1.5	0.5	1.6	1.8	0.7	2.1	6.4
Manifold Mixup	43.8	26.6	7.0	0.3	1.6	0.4	1.3	1.9	0.6	2.2	8.6
NFM	13.1	7.8	8.0	0.3	1.4	0.4	1.4	2.0	0.6	2.1	3.7
CutMix	78.0	60.1	9.6	0.5	1.9	0.6	1.5	2.4	0.8	3.2	15.9
Puzzle Mix	14.6	9.5	7.1	0.4	1.3	0.5	1.8	1.9	0.7	2.5	4.0
AugMix	11.2	6.6	2.1	0.2	0.9	**0.3**	1.3	1.0	0.4	1.3	2.5
NoisyMix (ours)	**6.3**	**4.2**	**2.0**	**0.1**	**0.7**	**0.3**	**1.0**	**0.8**	**0.3**	**1.1**	**1.7**