What Are Effective Labels for Augmented Data? Improving Calibration and Robustness with AutoLabel

1st Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

***Abstract*—A wide breadth of research has devised data augmentation approaches that can improve both accuracy and generalization performance for neural networks. However, augmented data can end up being far from the clean training data and what is the appropriate label is less clear. Despite this, most existing work simply uses one-hot labels for augmented data. In this paper, we show re-using one-hot labels for highly distorted data might run the risk of adding noise and degrading accuracy and calibration. To mitigate this, we propose a *generic* method `AutoLabel` to automatically learn the confidence in the labels for augmented data, based on the transformation distance between the clean distribution and augmented distribution. `AutoLabel` is built on label smoothing and is guided by the calibration-performance over a hold-out validation set. We successfully apply `AutoLabel` to three different data augmentation techniques: the state-of-the-art RandAug, AugMix, and adversarial training. Experiments on CIFAR-10, CIFAR-100 and ImageNet show that `AutoLabel` significantly improves existing data augmentation techniques over models' calibration and accuracy, especially under distributional shift. Additionally, `AutoLabel` improves adversarial training by bridging the gap between clean accuracy and adversarial robustness.**

***Index Terms*—data augmentation, calibration, distributional shift, adversarial robustness**

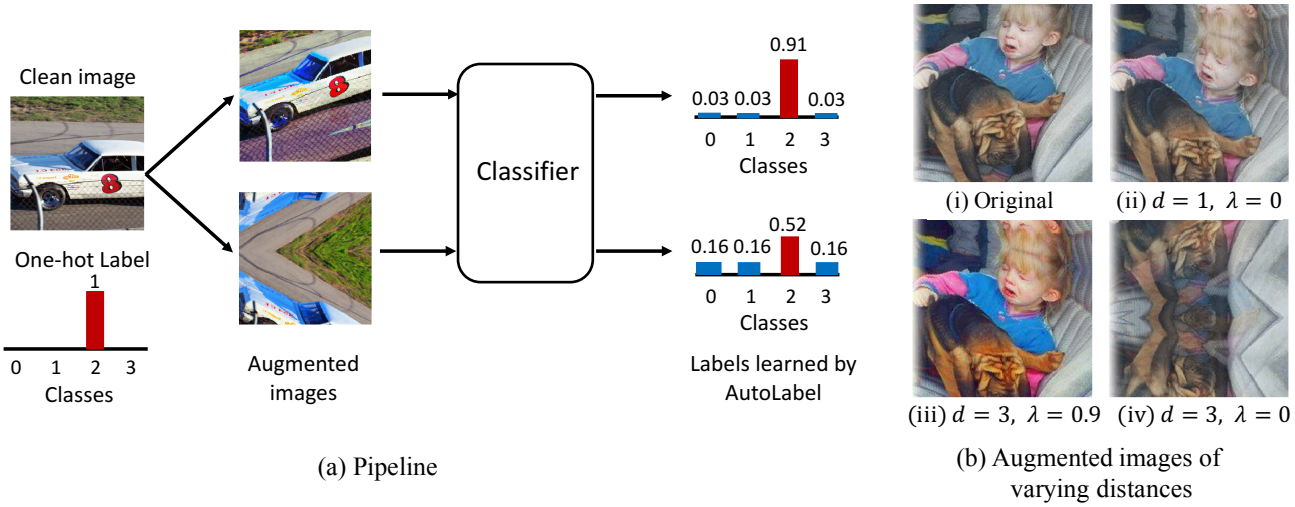
I. Introduction

Deep neural networks are increasingly being used in high-stakes applications such as healthcare and autonomous driving. For safe deployment, we not only want models to be accurate on independent and identically distributed (i.i.d.) test cases, but we also want models to be robust to distribution shift [1] and to not be vulnerable to adversarial attacks [2]–[5]. Recent work has shown that the accuracy of state-of-the-art models drops significantly when tested on corrupted data [6]. Furthermore, these models do not just perform worse on these unexpected examples, but are also over-confident – [7] showed that calibration of models degrades under shift. Calibration measures the gap between a model's own estimate of correctness (i.e., confidence) versus the empirical accuracy, which measures the actual probability of correctness. When a model is not well calibrated, particularly on unexpected examples, it undermines our ability to trust its predictions. Building models that are accurate *and* robust, i.e. can be *trusted* under unexpected inputs from both distributional shift and adversarial attacks, is a challenging but important research problem.

While numerous approaches have been explored for improving both calibration under distribution shift and adversarial robustness, one of the fundamental building blocks is *data augmentation*: generating synthetic examples, typically by modifying existing training examples, that provide additional training data outside the empirical training distribution. A wide breadth of literature has explored what are effective ways to modify training examples, such as making use of domain knowledge through label-preserving transformations [8] or adding adversarially generated, imperceptible noise [4], [9]. Approaches like these have been shown to improve the robustness and calibration of overparametrized neural networks as they alleviate the issue of neural networks overfitting to spurious features that do not generalize beyond the i.i.d. test set.

In the broad amount of research on data augmentation, most of it attempts to apply transformations that do not change the true label such that the label of the original example can also be assumed to be the label of the transformed example, without expensive manual review. While there has been a significant amount of work in how to construct such pseudo-examples in *input* space, there has been relatively little attention on whether this assumption of label-preservation holds in practice and what *label* should be assigned to such augmented inputs. For instance, many popular methods assign one-hot targets to both training data as well as augmented inputs that can be quite far away from the training data where even human raters may not be 100% sure of the label. This runs the risk of adding noise to the training process and degrading accuracy and calibration, as the model may learn to assign high confidence predictions to inputs far away from training data.

With this observation, in this paper we investigate the confidence assigned to target labels for augmented inputs and propose `AutoLabel` , a method that automatically adapts the confidence assigned to augmented labels, assigning high confidence to inputs close to the training data and lowering the confidence as we move farther away from the training data.



(a) Pipeline

(b) Augmented images of varying distances

Fig. 1: (a): An example showing `AutoLabel` assigning different labels to augmented images (e.g., by AugMix [8]) based on their transformation distances to the clean image. The label for the true class is automatically learned based on the calibration performance on validation set. (b): Examples of images augmented by AugMix with different distances to the original image.

Figure 1 (left) gives a high-level overview of our proposed `AutoLabel` along with examples of augmented images of varying distances generating by AugMix [8] on the right.

In summary, our key contributions are:

- We propose `AutoLabel` , a *generic* approach that can automatically learn the confidence in labels for augmented data based on their transformation distance.
- We show that `AutoLabel` is complementary to methods which focus on generating augmented inputs by combining it with RandAug [10] (which includes 10 different augmentation types), the state-of-the-art method AugMix [8], as well as adversarial training [4].
- We perform experiments on CIFAR-10, CIFAR-100 and ImageNet demonstrating that `AutoLabel` significantly improves the calibration of models on both clean and corrupted data. In addition, `AutoLabel` improves adversarial training via bridging the gap between accuracy and adversarial robustness.

II. Motivation: Why do we need AutoLabel?

Before we present our method, we investigate a key question:

How does the confidence assigned to labels of highly distorted augmented data affect model performance?

Using highly distorted augmented data is typically beneficial for improving robustness of the model and improves accuracy under covariate shift. However, using one-hot labels for highly distorted augmented data might decrease the clean accuracy as we assign 100% confidence to both clean training data and highly distorted training data. We hypothesize that tuning the confidence assigned to distorted augmented data using `AutoLabel` avoids this trade-off and increases both clean accuracy and accuracy under distribution shift.

To test this hypothesis, we first train a Wide ResNet-28-10 [11] on CIFAR-100 [12] and then report the accuracy on the test set with each transformation of different magnitudes applied. We use five different transformations: rotation, posterize, solarize, shear X and shear Y, whose distortion degree increases monotonically with the magnitude of the transformation, as used in [13]. We use the test accuracy of a transformation at a specific magnitude as an approximation of the distortion degree of the transformed images: lower test accuracy usually indicates higher degree of distortion on the transformed images. Figure 2 shows that test accuracy monotonically drops with increasing distortion magnitude; when such highly distorted data are used during training with one-hot labels, we may add a lot of noisy inputs which may lead to overfitting and potentially hurt clean accuracy.

Next, we train two networks using the above five transformations to augment the training data. The first network is trained with one-hot labels, another is trained with `AutoLabel` (which we will introduce in Section IV), which automatically adjusts the labels for each magnitude of each transformation. All the other training details are kept the same for these two networks. During training, we randomly sample a transformation with a magnitude ranging from 1 to the max magnitude and then apply this transformation to the training image for data augmentation. We report test accuracy and expected calibration error (ECE) [14], which measures how well aligned the average accuracy and the average predicted confidence, on CIFAR-100 as well as CIFAR-100-C [6], which includes 17 different corruption types. Note that there is *no overlap* between the 5 transformations used for data augmentation and the 17 corruption types in CIFAR-100-C.

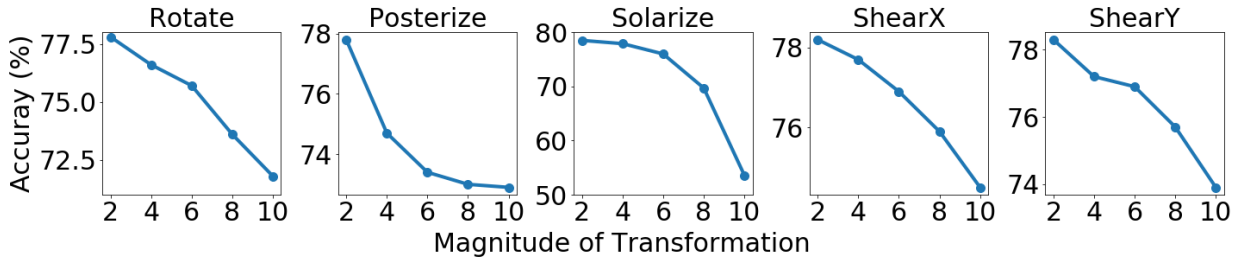


Fig. 2: Test accuracy on CIFAR-100 test dataset to which each transformation of a specific magnitude is applied. Lower accuracy indicates higher distortion on the transformed images.

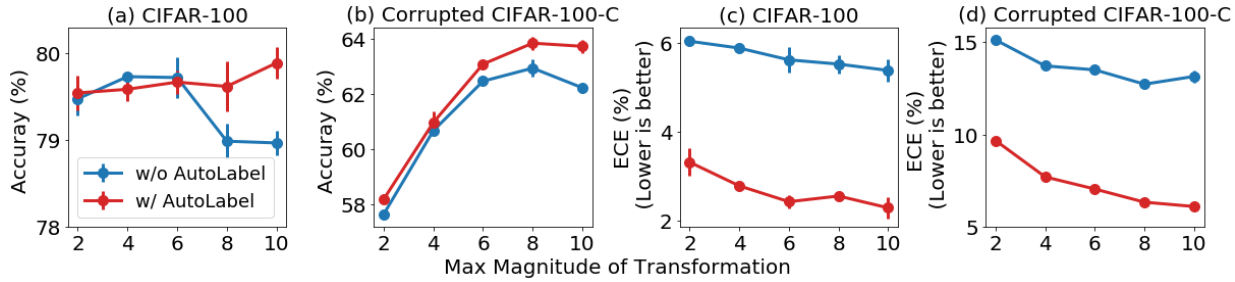


Fig. 3: **Large distortions limit or hurt accuracy and calibration:** Test accuracy (higher is better) and expected calibration error (ECE, lower is better) on CIFAR-100 test dataset and CIFAR-100-C corruption dataset. We randomly sample a transformation from rotation, posterzie, solarize, shear X and shear Y with a magnitude ranging from 1 to the max magnitude and apply this transformation to the training image for data augmentation. As the x axis increases, we include transformed images with higher distortion during training. The results are based on 5 independent runs.

In Figure 3, we report the accuracy (higher is better) and calibration (lower ECE is better) as the max magnitude of transformation increases (more highly distorted augmented data are incorporated into training). For networks trained without `AutoLabel` i.e., using one-hot labels (shown in blue), as we can see from the first graph in Figure 3, the clean accuracy drops significantly when transformed images with higher distortion are added. On the other hand, the networks trained with `AutoLabel` (shown in red) can leverage highly distorted augmented data without incurring such a drop in clean accuracy. In addition, the model trained with `AutoLabel` consistently has a higher accuracy and a smaller calibration error on both CIFAR-100 and CIFAR-100-C (more significantly), and the gap widens as the max magnitude increases. This motivates the need for `AutoLabel` when using highly distorted augmented training data to improve model accuracy and calibration, especially under *distributional shift*.

III. Related Work

A. Data Augmentation

Recent work has shown that introducing additional training examples can further improve a model's accuracy and generalization [13], [15]–[19]. For example, AugMix [8] utilizes stochasticity and diverse augmentations, together with a consistency loss over the augmentations, to achieve state-of-the-art corruption robustness. Mixup [20], on the other hand, trains a neural network over convex combinations of pairs of examples and shows improved generalization of neural networks. Furthermore, adversarial training [2], [4], [9] can also be thought as a special data augmentation technique aiming for improving model's adversarial robustness. In this paper, we investigate the choice of the target labels for augmented inputs and show how to apply `AutoLabel` to these existing data augmentation techniques to further improve model's robustness.

B. Calibration and Uncertainty Estimates

A variety of methods have been developed for improving a model's calibration, e.g., post-hoc calibration by temperature scaling [14] and multiclasss Dirichlet calibration [21]. Model's predictive uncertainty can also be quantified using Bayesian neural networks and approximate Bayesian approaches, e.g., variational inference [22], [23], MCMC sampling based on stochastic gradients [24], and dropout-based variational inference [25], [26]. In addition to calibration over in-distribution data, more recently, [7] show that model calibration can further degrade under unseen data shifts, where ensemble of deep neural networks [27] is shown to be most robust to dataset shift. On the other hand, several data augmentation methods have also been shown to improve model's calibration under data shifts. For example, AugMix is shown to improve uncertainty measures on corrupted image classification benchmarks [8]. [28] demonstrate that neural networks trained with mixup are significantly better calibrated under dataset shift, and are less prone to over-confident predictions on out-of-distribution data.

C. Label Smoothing

Label smoothing, initially proposed in [29], is used to prevent a model from being too over-confident in its predictions, thus improving its generalization ability. It has been shown by [28], [30] that label smoothing can also effectively improve the quality of a model's uncertainty estimates. Our work is most closely related to the adaptive label smoothing algorithm in [31]. [31] observe the connection between adversarial robustness and uncertainty, and propose an algorithm for adaptively updating the amount of label smoothing based on the adversarial vulnerability of *clean data* to improve model's calibration. In contrast, we propose to adaptively smooth the labels for *augmented data* based on the distance to the clean training data, and show it can further improve a model's accuracy, calibration and adversarial robustness.

IV. AutoLabel: A Generic Framework for Setting Labels on Augmented Data

A. Notation

Given a clean dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1,\cdots,m}$, where $y \in \{1, ..., K\}$, the one-hot encoding of the label is denoted as $\hat{y} \in \{0, 1\}^K$, where the label for the true class $\hat{y}_{k=y} = 1$ and $\hat{y}_{k\neq y} = 0$ for others. In addition to the training data $\mathcal{D}$, we also have a clean validation set $\mathcal{D}_V$ drawn i.i.d. from the same distribution.

B. The `AutoLabel` Algorithm

Our key insight is that the ***confidence*** in the labels associated with the augmented data likely depends on how distorted the transformation is. To make use of this insight, we would like to assign different confidence values in labels for the augmented training data based on their transformation distances. Many data augmentation approaches have hyperparameters that reflect how large the transformation should be. As examples, which we will discuss in-depth below, this can take the form of the number of transformations in AugMix [8] or the norm of the adversarial perturbation in [4] as shown in Table I. With aware of the transformation distance, then the problem becomes:

How should we set the confidence value in the labels for augmented data?

To address this problem, we build `AutoLabel` upon the hypothesis that effective labels for augmented training data can lead to well-calibrated predictions on the **samely** augmented validation set. Thus, the training labels for augmented data can be automatically updated according to the calibration performance on the samely augmented validation data. A schematic diagram for updating confidence of training labels in `AutoLabel` is shown in Figure 4. Specifically, if a model is over-confident on the augmented validation set, then the confidence in the training labels should be decreased accordingly; otherwise the confidence should be increased. Note that computing calibration on the augmented validation set does not use the confidence in the labels of validation data.

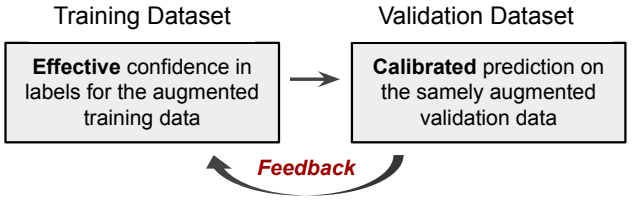


Fig. 4: A schematic diagram for updating confidence of training labels in `AutoLabel` algorithm.

Taking these together, our proposed `AutoLabel` is mainly composed of two components:

1) a measure of the transformation distance for the augmented data,
2) a subroutine for updating labels of the augmented data during training.

Specifically, given a data augmentation technique `Aug` that takes in an image x and outputs an augmented image $\texttt{Aug}(x, s)$ that transformed by a distance $s \in \mathrm{R}$, `AutoLabel` updates its label based on the calibration performance on the samely augmented validation data. Since calibration can not be computed over a single data point, we must obtain an augmented validation set that is transformed by the same distance s. To this end, we discretize the transformation distance s into N buckets $\{S_1, \cdots, S_N\}$ where each S_n is a range, and we can generate augmented data for bucket S_n by sampling a distance uniformly in that range $s \sim \mathcal{U}(S_n)$ to generate $\texttt{Aug}(x, s)$. In this way, we can generate the augmented validation set $\mathcal{Q}(S_n) = \{(\texttt{Aug}(x_i, s), y_i) | (x_i, y_i) \in \mathcal{D}_V, s \sim \mathcal{U}(S_n)\}$, which is used to learn the labels for any training data transformed by a distance $s \in S_n$.

With the augmented validation set $\mathcal{Q}(S_n)$, `AutoLabel` updates the confidence of the true class $\tilde{y}_{k=y}(S_n)$ after each training epoch t according to:

$$\tilde{y}^{t+1}_{k=y}(S_n) = \tilde{y}^{t}_{k=y}(S_n) - \alpha \cdot \mathrm{ECE}^t(\mathcal{Q}(S_n)) \cdot \mathrm{sign}(\mathrm{Conf}^t(\mathcal{Q}(S_n)) - \mathrm{Acc}^t(\mathcal{Q}(S_n))) \quad (1)$$

where $\mathrm{ECE}(\mathcal{Q}(S_n))$, $\mathrm{Acc}(Q(S_n))$ and $\mathrm{Conf}(\mathcal{Q}(S_n))$ are respectively the expected calibration error, accuracy and confidence on the augmented validation set. The sign of ($\mathrm{Conf}(\mathcal{Q})$ - $\mathrm{Acc}(\mathcal{Q})$) indicates if the model is overall over-confident (> 0) or under-confident (< 0). Intuitively, if the model is over-confident on the validation set, we should reduce the confidence given to the true class $\tilde{y}_{k=y}$, otherwise we should increase $\tilde{y}_{k=y}$. The expected calibration error on the augmented validation set $\mathrm{ECE}(\mathcal{Q}) \geq 0$ suggests to what extent we should adjust the labels as the optimal result is $\mathrm{ECE}(\mathcal{Q}) = 0$ when the training converges. The hyperparameter α controls the step size of updating the labels. Since $\tilde{y}^{t+1}_{k=y}$ stands for the probability of the true class, we clip the value to be within $[\mathrm{Acc}^t(\mathcal{Q}), 1]$ after each update. $\mathrm{Acc}^t(\mathcal{Q})$ is used as the minimum clipping value to prevent $\tilde{y}^{t+1}_{k=y}$ from being too small as $\mathrm{Acc}^t(\mathcal{Q}) \to \frac{1}{K}$ when the classifier is a random guesser.

TABLE I: Transformation distance used by AutoLabel for each data augmentation method.

Augmentation Method	Transformation distance determined by
RandAug	transformation type, magnitude of transformation m
AugMix	depth of augmentation chain d, mixing parameter λ
Adversarial Training	max ℓ_∞ norm ϵ

Given the updated label for the true class $\tilde{y}^{t+1}_{k=y}(S_n)$, AutoLabel takes a label smoothing approach to uniformly distribute the remaining probability to other classes:

$$\tilde{y}^{t+1}_{k\neq y}(S_n) = (1 - \tilde{y}^{t+1}_{k=y}(S_n)) \cdot \frac{1}{K-1}, \tag{2}$$

where K is the number of classes in the dataset and $\sum_{k=1}^{K} \tilde{y}_k = 1$. Finally, AutoLabel trains the model using $\tilde{y}(S_n)$ as the target for the cross-entropy loss across the augmented data. A complete pseudocode for AutoLabel is presented in Algorithm 1.

V. AutoLabel + Data Augmentations

To demonstrate AutoLabel can easily slot into existing data augmentation methods, we show how to apply AutoLabel to automatically adjust the confidence in labels over different data augmentation methods:

- RandAug [10]: including 10 different types of simple transformations used by AutoAugment [13]. Note that these transformations **do not overlap** with corruption types in the test corrupted datasets.
- AugMix [8]: Mixing diverse simple augmentations in convex combinations.
- Adversarial Training [4]: A special case of data augmentation to improve adversarial robustness via training on constructed adversarial examples.

These data augmentation methods originally use one-hot labels for augmented data and we discuss in details how to apply AutoLabel for each of them.[1] Table I shows an overview of how AutoLabel differentiates the augmented data based on the transformation distance under each data augmentation method.

Below we first give an overview of each data augmentation method and introduce how to differentiate the augmented data based on their distance to the clean distribution, then we show how to apply AutoLabel to learn more appropriate labels.

A. AutoLabel for RandAug

RandAug includes 10 different types of transformations from AutoAugment [13] and RandAugment [10]: color, rotation, autocontrast, equalize, posterize, solarize, shear X, shear Y, translate X and translate Y. During training, RandAug randomly applies one transformation to generate the augmented data. Unlike RandAugment [10] that optimizes a single distortion magnitude for all the transformations, RandAug randomly samples a distortion magnitude $m \in \{1, \cdots, m_{max}\}$, where m_{max} is the maximum distortion magnitude. Finally, the model is trained on the augmented data with one-hot labels.

Instead of using one-hot labels, we propose AutoLabel to automatically learn the confidence in labels for the augmented data that are transformed by different transformation distances. In RandAug, the transformation distance is determined by two factors: (1) the type of sampled transformation, in total we have 10 different transformations, and (2) the distortion magnitude m.

To learn the labels for the augmented data transformed by a specific operation with a distortion magnitude at m, AutoLabel applies the same transformation distorted by the same magnitude m to construct the augmented validation set. Then AutoLabel updates the confidence of labels according to Eqn (1) & (2) and trains the model with these updated labels.

B. AutoLabel for AugMix

AugMix [8] is a data augmentation technique that achieves state-of-the-art robustness and uncertainty estimates under data shift. Specifically, AugMix augments the input data via feeding the input x into an augmentation chain[2] which consists of $d \in \{1, 2, 3\}$ transformations randomly sampled from 10 different operations used in RandAug with a fixed distortion magnitude. Then a convex combination is performed to mix the augmented image x_{aug} with the original image x: $\texttt{Aug}_{augmix}(x) = \lambda \cdot x + (1 - \lambda) \cdot x_{\text{aug}}$, where the mixing parameter $\lambda \in [0, 1]$ is randomly sampled from a uniform distribution.

In AugMix, the transformation distance is mainly controlled by two parameters[3]: (1) the depth of the augmentation chain d, which decides how many augmentation operations are applied to the original image; (2) the mixing parameter λ, which controls the ratio of the augmented image x_{aug} and the original image x. In Figure 1(b) (ii) and (iv), a deeper augmentation chain causes the image to quickly degrade and drift off the data manifold. In addition, when comparing the augmented images with different mixing parameter λ, shown in Figure 1(b) (iii) and (iv), we can see that as $\lambda \to 0$, the augmented image is further away from the clean image. As a result, we can define the distance bucket $S_{d,n}$ for the augmented data as:

[1] We also apply AutoLabel to mixup [20], which is a data augmentation technique that uses soft-labels, and observe an improvement of AutoLabel over calibration. We refer interested readers to our supplementary material for more details and results.

[2] The original AugMix [8] uses 3 augmentation chains. However, we consistently observe an accuracy increase when we use one augmentation chain.

[3] Transformation types could provide us more precise transformation distance but is not our main focus in AugMix.

Algorithm 1 Pseudocode of `AutoLabel`

Input: A training dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1,\cdots,m}$, a validation dataset $\mathcal{D}_\mathcal{V}$ drawn i.i.d. from the same distribution, an augmentation method `Aug`. Number of classes K, number of training epochs T, number of distance buckets N and the hyperparameter α.
We perform `Aug` to obtain the augmented training data $\texttt{Aug}(x, s)$, where the transformation distance s is determined by the hyperparameters in the `Aug`. We discretize the transformation distance s into N buckets $\{S_1, \cdots, S_N\}$, where each S_n is a range.
For each distance bucket S_n, we initialize $\tilde{y}^0(S_n)$ as the one-hot label.
for $t = 0$ **to** $T - 1$ **do**
  Minimize cross-entropy loss over the augmented training data with smoothed labels $\tilde{y}^t(S_n)$.
  for $n = 1$ **to** N **do**
    Generate an augmented validation set: $\mathcal{Q}(S_n) = \{(\texttt{Aug}(x_i, s), y_i) | (x_i, y_i) \in \mathcal{D}_V, s \sim \mathcal{U}(S_n)\}$.
    Update the label for the true class $\tilde{y}^{t+1}_{k=y}(S_n)$:
    $\tilde{y}^{t+1}_{k=y}(S_n) = \tilde{y}^t_{k=y}(S_n) - \alpha \cdot \text{ECE}^t(\mathcal{Q}(S_n)) \cdot \text{sign}(\text{Conf}^t(\mathcal{Q}(S_n)) - \text{Acc}^t(\mathcal{Q}(S_n)))$ ▷ *according to Eqn (1)*
    Clip $\tilde{y}^{t+1}_{k=y}(S_n)$ to be within $[\text{Acc}^t(\mathcal{Q}(S_n)), 1]$
    Update the label for other classes $\tilde{y}^{t+1}_{k\neq y}(S_n)$: $\tilde{y}^{t+1}_{k\neq y}(S_n) = (1 - \tilde{y}^{t+1}_{k=y}(S_n)) \cdot \frac{1}{K-1}$ ▷ *according to Eqn (2)*
  end for
end for

$S_{d,n} = S_{d,\lceil \lambda N \rceil}$,[4] where N is the total number of buckets at a given depth d.

Next, to learn the labels for augmented training data within a distance bucket $S_{d,n}$, `AutoLabel` constructs an augmented validation set $\mathcal{Q}(S_{d,n})$ by feeding the validation images into an augmentation chain with the depth d and then randomly sample a mixing parameter λ' from a uniformly distribution: $\lambda' \sim \mathcal{U}(\frac{n}{N}, \frac{n+1}{N})$ to mix the original image and the augmented image. Finally, `AutoLabel` updates the labels $\tilde{y}(S_{d,n})$ according to Eqn (1) & (2) and trains the model using these updated labels.

C. AutoLabel for Adversarial Training

Adversarial training [2] can be formulated as solving the min-max problem:

$$\min_w \mathop{\mathbb{E}}_{||\delta||_\infty \leq \epsilon} [\max \mathcal{L}(f(x + \delta; w), y)], \tag{3}$$

where δ denotes the adversarial perturbation, ϵ denotes the maximum ℓ_∞ norm of adversarial perturbation and a one-hot encoding of the label y is used as the target for the cross-entropy loss $\mathcal{L}$. In [4], the inner maximization problem is approximately solved by generating projected gradient descent (PGD) attacks. Therefore, standard adversarial training can be considered as a specific data augmentation that aims for improving model's adversarial robustness.

Instead of training a model with one-hot labels, `AutoLabel` differentiates the adversarial examples according to the distance between the adversarial examples and clean data, which is approximately captured by the ℓ_∞ norm of the adversarial perturbation ϵ. Unlike [4] using a fixed ϵ to construct PGD adversarial attacks, we randomly sample ϵ from a uniform distribution $\epsilon \sim \mathcal{U}(0, \epsilon_{max})$ to construct PGD adversarial attacks with different distances to the original data.

[4]In the special case where $\lambda = 0$, we merge it into bucket S_1 to avoid creating an additional bucket, similarly for ϵ in adversarial training.

If the ℓ_∞ norm of the adversarial perturbation is bounded by ϵ, then the constructed adversarial example falls into the distance bucket $S_n = S_{\lceil \epsilon \cdot \frac{N}{\epsilon_{max}} \rceil}$, where N is the total number of distance buckets.

In order to learn the labels for adversarial examples within a distance bucket S_n, `AutoLabel` constructs adversarial examples for the validation images with the ℓ_∞ norm of the adversarial perturbation bounded by ϵ', where ϵ' is randomly sampled from a uniform distribution: $\epsilon' \sim \mathcal{U}(\frac{n \cdot \epsilon_{max}}{N}, \frac{(n+1) \cdot \epsilon_{max}}{N})$. Finally the training labels for adversarial examples are updated following Eqn (1) & (2).

VI. AutoLabel improves calibration

In this section, we mainly show how `AutoLabel` can help improve standard data augmentation techniques in terms of the calibration performance by assigning effective confidence in labels.

A. Baselines

1) Baselines for RandAug and AugMix: In order to demonstrate the effectiveness of `AutoLabel` while applying to existing data augmentation techniques, we use the state-of-the-art RandAug [10] and AugMix [8] trained with one-hot labels as baselines.

Further, as `AutoLabel` is built upon label smoothing (LS) [29], we also report the performance when label smoothing is applied to each of these data augmentations.

2) Baselines for adversarial training: In order to show `AutoLabel` can help adversarial training benefit calibration under distributional shift, we compare it with other 5 different models. They are

- A vanilla model trained with one-hot labels.
- Adversarial training (AT) [4], which is trained on projected gradient descent (PGD) attacks with one-hot labels.

- Adversarial training with label smoothing (AT + LS).
- Adversarial training with PGD attacks generated with multiple ℓ_∞ norm bounds (AT + multiple ϵ). Unlike standard adversarial training in [4] using a fixed ϵ to construct PGD attacks, this model randomly samples an $\epsilon \in (0, \epsilon_{max}]$ during training. The one-hot labels are used for the generated adversarial attacks.
- Confidence-calibrated adversarial training (CCAT) [32], which smooths the labels for adversarial examples according to $\tilde{y} = g(\delta)\hat{y} + (1 - g(\delta)\frac{1}{K})$, where the balancing parameter $g(\delta)$ follows a "power transition": $g(\delta) := (1 - \min(1, \frac{||\delta||_\infty}{\epsilon}))^\rho$, ρ is set to 10 as [32] to ensure a uniform distribution is used as the labels for adversarial examples when $||\delta||_\infty \geq \epsilon$. Similarly, ϵ is randomly sampled from (0, ϵ_{max}] during training.

B. Datasets

For RandAug and AugMix related experiments, we report the performance on CIFAR-100 [12] and ImageNet [33]. As adversarial training is hard to scale on ImageNet, we mainly perform experiments for adversarial training on CIFAR-10 and CIFAR-100 [12]. We randomly sample 5000 images from 50000 training data to serve as the hold-out validation set for both CIFAR datasets. For ImageNet, we evenly split 50000 test images into a validation set with 25000 images and test the models' performance on the hold-out test set with 25000 images.

In addition, we test models' robustness on the corrupted datasets: CIFAR-10-C, CIFAR-100-C, ImageNet-C [6], which include different corruptions types (17 types for CIFAR-10/100 and 15 types for ImageNet) that are frequently encountered in natural images. Each type of corruption has five corruption severities. Note that the corruption type in the corrupted dataset ***do not overlap*** with transformations used for data augmentations.

C. Evaluation Metrics

We report the classification accuracy and expected calibration error on the clean datasets as **Accuracy** (higher is better) and **ECE** (lower is better) respectively. Specifically, given a classifier $f(\cdot)$ for a K-class classification problem, let $f_k(x)$ denote the predicted probability for the k-th class. We use $f(x) := \text{argmax}_k f_k(x)$ to represent the predicted class and $c(x) := \max_k f_k(x)$ as model's confidence of the predicted class. The expected calibration error (ECE) [7], [14] is defined as

$$\text{ECE} = \sum_r^R \frac{|B_r|}{m} |\text{acc}(B_r) - \text{conf}(B_r)|,$$

where the input data is divided into R buckets, B_r indexes the r-th confidence bucket and m denotes the data size, the accuracy and the confidence of B_r are defined as:

$$\text{acc}(B_r) = \frac{1}{|B_r|} \sum_{i \in B_r} \mathbf{1}(f(x_i) = y_i)$$

$$\text{conf}(B_r) = \frac{1}{|B_r|} \sum_{i \in B_r} c(x_i)$$

Expected calibration error measures how well aligned the average accuracy and the average predicted confidence are.

In addition, we also report the accuracy and expected calibration error on the corrupted datasets, denoted as **cAccuracy** and **cECE**, which are computed as an average over all the corruption types across 5 corruption severities.

D. Network Architectures and Hyperparameters

We use a Wide ResNet-28-10 [11] for both CIFAR-10 and CIFAR-100 datasets, and a ResNet-50 [34] for ImageNet as our basic model architectures. We use the open-sourced code at https://github.com/google/uncertainty-baselines/tree/master/baselines to train all the models with the same training hyperparameters for fair comparison.

Below we display all other hyperparameters involved for each method as well as `AutoLabel` .

1) Label smoothing: When applying label smoothing to RandAug, AugMix and adversarial training, we sweep the hyperparameter ρ which decides the smoothing degree in a range [0, 0.1] with a step size 0.01 and find the best $\rho = 0.02$ for CIFAR10 and CIFAR100 and $\rho = 0.01$ for ImageNet.

2) Adversarial training: Similar to recent work [35], we observe that training models with adversarial examples bounded with a smaller ℓ_∞ norm, e.g., $\epsilon_{max} = 0.01$, can benefit more to the corrupted accuracy with a small accuracy drop on the clean data. Therefore, we train all models with PGD attacks bounded by $\epsilon_{max} = 0.01$ updated with 10 iterations. The step size is set to be $\epsilon/4$.

3) `AutoLabel` for RandAug: When `AutoLabel` is applied to RandAug, the hyperparameter α in Eqn (1) is swept in [0, 0.1]. We choose the best $\alpha = 0.01$ for CIFAR-100 and $\alpha = 0.02$ for ImageNet based on the holdout validation set. The number of distance bucket is set to be $N = 10$.

4) `AutoLabel` for AugMix: Following original AugMix [8], the max depth of the augmentation chain is set to be $d_{max} = 3$ and the number of distance bucket is set to be $d_{max} \cdot N = 3 \cdot 5 = 15$. We use the best $\alpha = 0.02$ in Eqn (1) for CIFAR-100 and ImageNet.

5) `AutoLabel` for adversarial training: The number of distance buckets is set to be $N = 10$ and the hyperparameter α in Eqn (1) is set to be $\alpha = 0.5$ for CIFAR-10 and $\alpha = 0.005$ for CIFAR-100.

E. Improvement over Calibration

1) `AutoLabel` improves calibration of RandAug and AugMix: In this section, we apply `AutoLabel` to RandAug and AugMix to investigate if `AutoLabel` can make data augmentation approaches more effective in improving calibration, especially under distributional shifts. We see in Table II a clear picture: `AutoLabel` consistently helps RandAug and AugMix improve both accuracy and calibration across CIFAR100 and ImageNet, and greatly outperforms label smoothing. In addition, we can see that `AutoLabel` has a much more significant improvement on models' calibration on the corrupted datasets. As shown in Figure 5 we analyze how calibration performance changes with the severity of the corruption being

TABLE II: **Improvements of `AutoLabel` over RandAug and AugMix.** The accuracy and ECE are reported on both in-distribution test datasets (CIFAR-100 and ImageNet) as well as the corresponding corrupted datasets (CIFAR-100-C and ImageNet-C). All numbers reported in the table are in %, an average of 4 independent runs on CIFAR-100 and 2 independent runs on ImageNet. The arrow indicates better direction. Best results are highlighted in **Bold**.

Method	Accuracy / cAccuracy (↑)		ECE / cECE (↓)	
	CIFAR-100	ImageNet	CIFAR-100-C	ImageNet-C
RandAug [10]	82.0 / 63.0	76.9 / 43.4	4.1 / 13.0	2.0 / 6.4
+ Label Smoothing	82.2 / 63.6	76.9 / 43.5	2.4 / 8.2	1.2 / 5.5
+ `AutoLabel`	**82.7 / 64.8**	**76.9 / 43.9**	**1.9 / 5.6**	**1.0 / 5.1**
AugMix [8]	81.1 / 64.3	75.9 / 46.1	4.5 / 10.9	1.5 / 4.9
+ Label Smoothing	81.3 / 64.6	76.0 / 46.3	2.6 / 6.5	1.5 / 4.6
+ `AutoLabel`	**81.9 / 65.5**	**76.4 / 46.5**	**1.8 / 4.2**	**1.4 / 4.2**

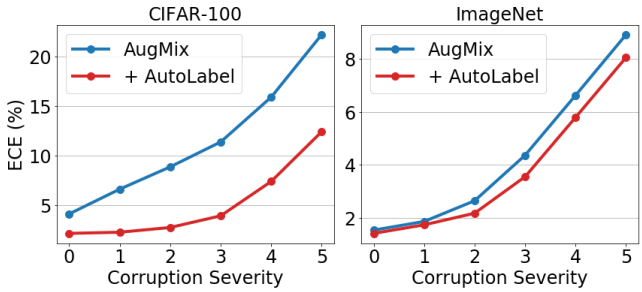


Fig. 5: Expected calibration error (ECE) of AugMix trained with one-hot labels and `AutoLabel` across corruption severities on CIFAR100 and ImageNet. Severity 0 denotes clean data. As the corruption severity increases, the improvement of `AutoLabel` increases.

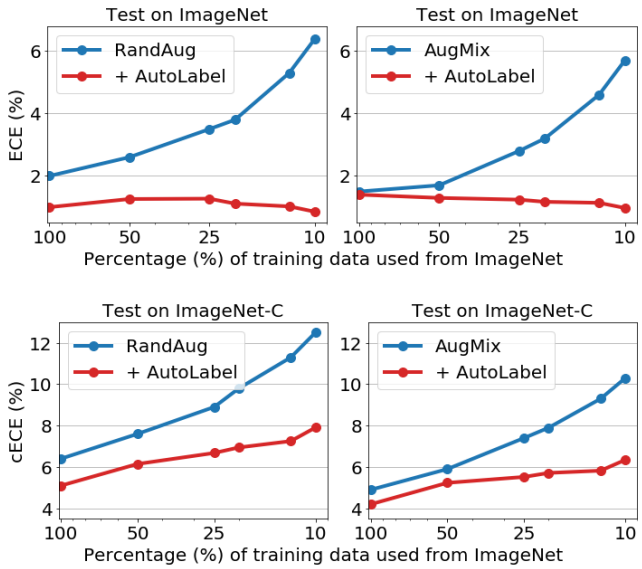


Fig. 6: **`AutoLabel` improves calibration on ImageNet with a reduced size of training data.** When training a model with limited training data, the improvement of `AutoLabel` over calibration is significantly increased. $\alpha = 0.02$ is used for `AutoLabel` for all experiments here without further tuning.

tested against, comparing AugMix trained with one-hot labels and with `AutoLabel` . We see that the baseline AugMix is increasingly worse calibrated as the corruption increases, but `AutoLabel` dampens that trend effectively. Similar patterns are observed when `AutoLabel` is applied to RandAug.

Looking more closely, we find that the improvement of `AutoLabel` over calibration on ImageNet is relatively smaller compared to CIFAR-100. We conjecture that this is mainly due to the much larger training data on ImageNet, leading to a greater generalization performance and the headroom for improvement is relatively limited. To validate this, we train the same networks with a reduced size of training images, e.g., we randomly sample 50% training images from the whole training dataset (∼1.2M training images). Note that we use the same hyperparameter $\alpha = 0.02$ in Eqn (1) without further tuning for each training data size. From Figure 6 we can see that as the percentage of training data is reduced, the calibration performance of both RandAug and AugMix trained with one-hot labels becomes significantly worse. In contrast, `AutoLabel` enables models to keep a low calibration error on the clean test set and a much smaller calibration error on the corrupted dataset. This indicates that `AutoLabel` could be especially beneficial to models' calibration performance when we have limited training data.

2) `AutoLabel` improves calibration of adversarial training: Similar to human-designed data augmentation, we also believe adversarial training runs the risk of attacks not being truly label-preserving [36] and thus should benefit from setting labels carefully. Therefore, we would like to see if `AutoLabel` can improve adversarial training with adversarial robustness.

To this end, we apply `AutoLabel` to adversarial training and report the calibration error on the clean and corrupted datasets in Table III. When comparing standard adversarial training (AT) with a vanilla model, we can see that adversarial training slightly improves the calibration on the corrupted dataset but at the cost of a larger calibration error on the clean data. All other adversarial training based models using one-hot labels suffer from this trade-off between clean and corrupted calibration. In contrast, `AutoLabel` nicely addresses this

TABLE III: **`AutoLabel` improves calibration over adversarial training.** Expected calibration error (ECE) on the clean CIFAR-10 and CIFAR-100 and on the corrupted CIFAR-10-C and CIFAR-100-C (measured by cECE). Note adversarial examples during training are bounded with a small ℓ_∞ norm ($\epsilon_{max} = 0.01$) for better in-distribution performance as [35]. All the numbers reported are an average over 2 independents runs and in %. Best result is highlighted in **bold**.

Method	Accuracy / cAccuracy (↑)		ECE / cECE (↓)	
	CIFAR-10	CIFAR-100	CIFAR-10-C	CIFAR-100-C
Vanilla	95.6 / 76.0	79.5 / 52.0	2.6 / 15.8	6.1 / 17.6
AT [4]	93.6 / **83.9**	71.5 / 58.1	3.7 / 10.5	8.0 / 13.5
CCAT [32]	93.2 / 68.9	74.8 / 49.8	2.4 / 9.9	7.9 / 16.1
AT + LS	93.1 / 83.5	71.7 / 57.9	2.1 / 7.9	4.4 / 6.7
AT + multiple ϵ	94.3 / 84.5	74.6 / 59.6	3.4 / 10.0	7.0 / 12.9
AT + `AutoLabel`	**94.6** / 83.6	**75.8 / 59.9**	**2.0 / 6.5**	**3.7 / 6.2**

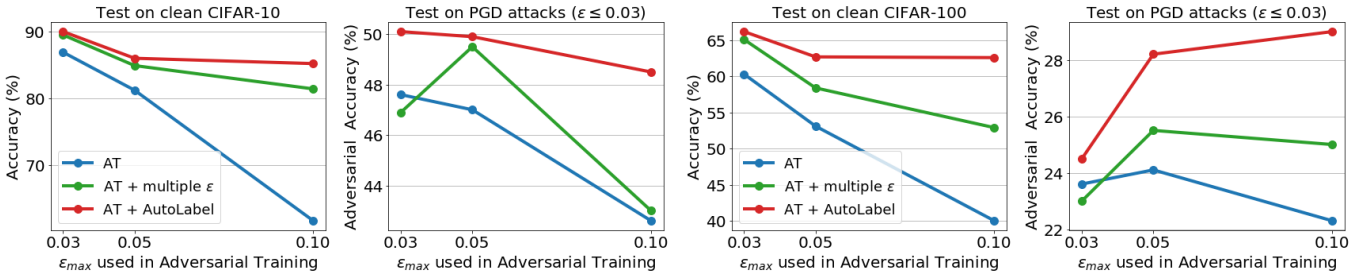


Fig. 7: **`AutoLabel` improves trade-off between clean and adversarial accuracy.** Each model is tested on the clean data (Left) and against white-box PGD attacks bounded by $\epsilon_{max} = 0.03$ (right).

problem: it improves calibration on both clean and corrupted datasets compared to vanilla model and adversarial training. Note that a pre-defined function to smooth the labels, e.g, the power transition used in CCAT [32], does not significantly help model's calibration. Although adversarial training together with `AutoLabel` still suffers from a small sacrifice on in-distribution accuracy, the improvements over both accuracy and calibration on the corrupted datasets are much more significant, e.g., the improvement of `AutoLabel` over calibration error on the corrupted data is around 60% compared to vanilla model on CIFAR-10/100-C.

Taking all together, we can naturally arrive at the conclusion: `AutoLabel` can again effectively help adversarial training improve models' calibration by adaptively setting appropriate labels on the augmented (adversarial) examples.

VII. AutoLabel Improves Adversarial Robustness

In this section, we further investigate if `AutoLabel` can also help bridge trade-off between clean accuracy & adversarial robustness, especially when highly distorted adversarial examples are incorporated into training.

A. Baselines

We use the same baselines introduced in Section VI-A2. The "Vanilla" model is trained without any adversarial examples. For other adversarial training based methods, we construct ℓ_∞ norm based PGD attacks with 10 iterations and the step size is set to be $\epsilon/4$. For each method, we train three models with the ℓ_∞ norm of adversarial perturbation ϵ_{max} chosen from $\{0.03, 0.05, 0.1\}$, where the image scale is [0, 1].

B. Evaluation Metrics

We report both clean accuracy and adversarial accuracy of each test model. The adversarial accuracy is computed over white-box PGD attacks bounded by $\epsilon_{max} = 0.03$ with image scale belongs to [0, 1]. These test PGD attacks are generated by 50 iterations and 3 random restarts.

In addition, we also propose a new evaluation metric called "Accuracy Difference Δ" to better measure the effectiveness of any given method in terms of clean accuracy as well as adversarial robustness. "Accuracy Difference Δ" is to compute the clean and adversarial accuracy difference between any given method as well as a baseline model. In our case, the "Vanilla" model trained with one-hot labels is used as the baseline. Denote any given method as Ω and the "Vanilla" model as Γ, the "Accuracy Difference Δ" between Ω and Γ can be computed as

$$\begin{aligned}\Delta(\Omega, \Gamma) = &\{\text{Accuracy}(\Omega) + \text{Adversarial. Accuracy}(\Omega)\} \\ &- \{\text{Accuracy}(\Gamma) + \text{Adversarial. Accuracy}(\Gamma)\}\end{aligned} \tag{4}$$

where the accuracy and adversarial accuracy are computed on the clean test dataset as well as white-box PGD attacks respectively. While computing Δ, clean accuracy and adversarial accuracy are assigned with equal importance. If $\Delta(\Omega, \Gamma)$ is larger than 0, this indicating that this method Ω is overall better than the vanilla model Γ in terms of clean accuracy as well as adversarial robustness.

TABLE IV: **`AutoLabel` helps balance the tradeoff between clean accuracy and adversarial accuracy.** "Adversarial Accuracy" denotes the model is tested on white-box PGD attacks bounded by $\epsilon_{max} = 0.03$ and generated by 50 iterations and 3 random restarts. Larger Accuracy Difference Δ is better (the arrow indicates higher is better). The best model with the highest Δ is highlighted in **bold**.

Method	ϵ_{max} in Adversarial Training	CIFAR-10			CIFAR-100		
		Clean Accuracy	Adversarial Accuracy	Δ (↑)	Clean Accuracy	Adversarial Accuracy	Δ (↑)
Vanilla	-	95.6	0	-	79.5	0	-
AT	0.03	86.9	47.6	+ 38.9	60.3	23.6	+ 4.4
	0.05	81.2	47.0	+ 32.6	53.1	24.1	- 2.3
	0.1	61.6	42.6	+ 8.6	40.0	22.3	-17.2
CCAT	0.03	92.9	0	- 2.7	73.3	0	- 6.2
	0.05	92.4	1.9	- 1.3	72.3	0.8	- 6.4
	0.1	92.1	4	+ 0.5	71.2	2.8	- 5.5
AT + LS	0.03	86.7	52.3	+ 43.4	61.2	27.4	+ 9.1
	0.05	81.9	46.6	+ 32.9	53.7	26.7	+ 0.9
	0.1	61.0	36.2	+ 1.6	39.6	23.8	-16.1
AT + multiple ϵ	0.03	89.5	46.9	+ 40.8	65.1	23.0	+ 8.6
	0.05	84.9	49.5	+ 38.8	58.4	25.5	+ 4.4
	0.1	81.4	43.0	+ 28.8	52.9	25.0	- 1.6
AT + `AutoLabel`	0.03	90.0	50.1	**+ 45.5**	66.2	24.5	**+ 11.2**
	0.05	86.0	49.9	**+ 40.3**	62.7	28.2	**+ 11.4**
	0.1	85.2	48.5	**+ 38.1**	62.6	29.0	**+ 12.1**

C. Network Architectures and Hyperparameters

We use the same Wide ResNet-28-10 [11] for both CIFAR-10 and CIFAR-100 datasets introduced in Section VI-D.

For adversarial training together with label smoothing (AT+ LS), we sweep the smoothing parameter ρ from $\{0.1, 0.2, 0.3\}$ and choose the best parameter for each ϵ_{max} on CIFAR-10 and CIFAR-100. Similarly, for adversarial training with `AutoLabel` , we set the number of distance buckets to be $N = 10$ and sweep the hyperparameter α in Eqn. (1) within a range [0.005, 1] and choose the best based on the performance on the holdout validation set.

D. Improvements over Adversarial Robustness

To validate if `AutoLabel` can help balance clean accuracy and adversarial accuracy when highly distorted adversarial examples are incorporated into training, we compare `AutoLabel` with standard adversarial training (AT) [4] as well as adversarial training with multiple ϵ (AT + multiple ϵ), which are trained with one-hot labels. We see a clear picture in Figure 7: as the ϵ_{max} used in adversarial training increases, that is we incorporate more corrupted adversarial examples into training, there is a significant clean accuracy drop for the models trained with one-hot labels. In contrast, `AutoLabel` effectively helps the model maintain a relatively high clean accuracy as well as adversarial accuracy even when $\epsilon_{max} = 0.1$.

This picture is even more clear when we look at the Accuracy Difference Δ in Table IV: AT + `AutoLabel` keeps Δ relatively stable and always larger than 0 when we increase ϵ_{max} used in adversarial training, e.g., $\epsilon_{max} = 0.1$, whereas other models suffer from a significant performance drop. This further validates that `AutoLabel` is especially useful when highly corrupted adversarial examples are involved into training. Note this is perfectly aligned with the observation in standard data augmentation techniques shown in Figure 3: assigning one-hot labels to highly-distorted augmented data can hurt accuracy and `AutoLabel` can better balance the accuracy trade-off by assigning more appropriate labels for augmented data.

In addition, since confidence-calibrated adversarial training (CCAT) [32] is originally proposed to detect adversarial examples with a confidence-thresholding, it hardly helps adversarial accuracy compared to vanilla model, as shown in Table IV. This is mainly because CCAT uses the uniform distribution for adversarial examples during training, resulting in a loss of class information.

VIII. Conclusion

In this paper, we show that simply re-using one-hot labels for augmented data, as commonly used in existing data augmentation works, runs the risk of adding noise and degrading accuracy and calibration. To mitigate this, we propose `AutoLabel` to automatically learn the confidence in labels for augmented data based on the transformation distance between the augmented data and the clean data. We demonstrate the effectiveness of `AutoLabel` by applying it to RandAug, AugMix and adversarial training. We see that `AutoLabel` greatly improves the models' calibration, especially on corrupted data, and also helps adversarial training with a better trade-off between clean accuracy and adversarial robustness. More generally, we believe that more nuanced approaches to setting labels for augmented data, beyond

assuming label-preserving transformations, will lead to more effective data augmentation techniques.

REFERENCES

[1] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in AI safety," *arXiv preprint arXiv:1606.06565*, 2016.

[2] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, 2014.

[3] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 39–57.

[4] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations*, 2017.

[5] Y. Qin, N. Frosst, S. Sabour, C. Raffel, G. Cottrell, and G. Hinton, "Detecting and diagnosing adversarial images with class-conditional capsule reconstructions," *International Conference on Learning Representations*, 2020.

[6] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," in *Proceedings of the International Conference on Learning Representations*, 2019.

[7] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek, "Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift," in *Advances in Neural Information Processing Systems*, 2019, pp. 13 991–14 002.

[8] D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan, "AugMix: A Simple Data Processing Method to Improve Robustness and Uncertainty," in *International Conference on Learning Representations*, 2020.

[9] H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. E. Ghaoui, and M. I. Jordan, "Theoretically principled trade-off between robustness and accuracy," *International Conference on International Conference on Machine Learning*, 2019.

[10] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "RandAugment: Practical automated data augmentation with a reduced search space," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 702–703.

[11] S. Zagoruyko and N. Komodakis, "Wide residual networks," *ArXiv*, vol. abs/1605.07146, 2016.

[12] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009.

[13] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation strategies from data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 113–123.

[14] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *International Conference on Machine Learning*, 2017.

[15] T. Devries and G. W. Taylor, "Improved regularization of convolutional neural networks with CutOut," *arXiv preprint arXiv:1708.04552*, 2017.

[16] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," *ICCV*, 2019.

[17] R. Takahashi, T. Matsubara, and K. Uehara, "Data augmentation using random image cropping and patching for deep cnns," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. PP, pp. 1–1, 08 2019.

[18] R. G. Lopes, D. Yin, B. Poole, J. Gilmer, and E. D. Cubuk, "Improving robustness without sacrificing accuracy with patch Gaussian augmentation," *arXiv preprint arXiv:1906.02611*, 2019.

[19] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," in *AAAI*, 2020.

[20] H. Zhang, M. Cissé, Y. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," in *International Conference on Learning Representation*, 2018.

[21] M. Kull, M. Perello Nieto, M. Kängsepp, T. Silva Filho, H. Song, and P. Flach, "Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with dirichlet calibration," in *Advances in Neural Information Processing Systems*, 2019, pp. 12 316–12 326.

[22] A. Graves, "Practical variational inference for neural networks," in *Advances in Neural Information Processing Systems 24*, 2011, pp. 2348–2356.

[23] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," in *ICML*, vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 1613–1622.

[24] M. Welling and Y. W. Teh, "Bayesian learning via stochastic gradient Langevin dynamics," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ser. ICML'11. Madison, WI, USA: Omnipress, 2011, p. 681–688.

[25] D. P. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," in *Advances in Neural Information Processing Systems*, 2015, pp. 2575–2583.

[26] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *International Conference on Machine Learning*, 2016, pp. 1050–1059.

[27] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Advances in Neural Information Processing Systems*, 2017, p. 6405–6416.

[28] S. Thulasidasan, G. Chennupati, J. A. Bilmes, T. Bhattacharya, and S. E. Michalak, "On mixup training: Improved calibration and predictive uncertainty for deep neural networks," in *Advances in Neural Information Processing Systems*, 2019.

[29] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *2016 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.

[30] R. Müller, S. Kornblith, and G. E. Hinton, "When does label smoothing help?" in *Advances in Neural Information Processing Systems*, 2019, pp. 4694–4703.

[31] Y. Qin, X. Wang, A. Beutel, and E. H. Chi, "Improving calibration through the relationship with adversarial robustness," in *Advances in Neural Information Processing Systems*, 2021.

[32] D. Stutz, M. Hein, and B. Schiele, "Confidence-calibrated adversarial training: Generalizing to unseen attacks," in *Proc. of the International Conference on Machine Learning (ICML)*, 2020.

[33] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F.-F. Li, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, pp. 211–252, 2015.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[35] C. Xie, M. Tan, B. Gong, J. Wang, A. L. Yuille, and Q. V. Le, "Adversarial examples improve image recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 819–828.

[36] Y. Qin, N. Frosst, C. Raffel, G. Cottrell, and G. Hinton, "Deflecting adversarial attacks," *arXiv preprint arXiv:2002.07405*, 2020.

APPENDIX

A. AutoLabel for mixup

Other than data augmentation techniques that using one-hot labels, we also apply `AutoLabel` to mixup [20], which mixes the input data as well as their correspondingly labels. Recently, mixup has also been shown to be able to help with calibration in [28]. Specifically, the input data as well as their labels are augmented by

$$\begin{aligned} \texttt{Aug}_{mixup}(x_i, x_j) &= \gamma x_i + (1-\gamma) x_j \\ \texttt{Aug}_{mixup}(y_i, y_j) &= \gamma \hat{y}_i + (1-\gamma) \hat{y}_j \end{aligned} \tag{5}$$

where x_i and x_j are two randomly sampled input data and $\hat{y}_i$ and $\hat{y}_j$ are their associated one-hot labels. The model is trained with the standard cross-entropy loss $\mathcal{L}(f(x_{mixup}), y_{mixup})$ and the mixing parameter $\gamma \in [0, 1]$ that determines the mixing ratio is randomly sampled from a Beta distribution Beta(β, β) at each training iteration. Rather than using the same mixing parameter γ to combine the labels, we show

how to apply AutoLabel to automatically learn its labels $\texttt{Aug}_{mixup}(y_i, y_j)$ based on the validation calibration.

a) Transformation Distance: The transformation distance in mixup is determined by the mixing parameter γ in Eqn (5). When $\gamma \to 0.5$, combining two images equally, the augmented image $\texttt{Aug}_{mixup}(x_i, x_j)$ is the most far away from the clean distribution. On the other hand, when $\gamma \to 0$, the augmented image is close to the original image. Hence, the distance bucket S_n for each augmented example $\texttt{Aug}_{mixup}(x_i, x_j)$ can be defined as: $S_n = S_{\lceil 2N \cdot (\min(\gamma, 1-\gamma))\rceil}$.

b) Update Labels: To learn the labels for the augmented training image within the distance bucket S_n, AutoLabel constructs an augmented validation set $\mathcal{Q}(S_n)$ by randomly mixing two images from validation data with a mixing parameter γ' that is sampled from a uniform distribution: $\gamma' \sim \mathcal{U}\left(\frac{n}{2N}, \frac{n+1}{2N}\right)$ and $\gamma' \in [0, 0.5]$. Unlike AugMix, there are two classes y_i and y_j existing in the augmented image $\texttt{Aug}_{mixup}(x_i, x_j)$. Due to $\gamma' \in [0, 0.5]$, the class in the image x_j plays a dominant role in determining the main class in $\texttt{Aug}_{mixup}(x_i, x_j)$. Therefore, we follow Eqn (1) in the main text to update the label $\tilde{y}_{k=y_j}$ for the class $k = y_j$. Unlike Eqn (2) in the main text that uniformly distributes the probability $1 - \tilde{y}_{k=y_j}$ to all other classes, we update the label for the class $k = y_i$ as $\tilde{y}_{k=y_i} = \min(1 - \tilde{y}_{k=y_j}, \frac{\gamma'}{1-\gamma'}\tilde{y}_{k=y_j})$ and then distribute the probability $1 - \tilde{y}_{k=y_i} - \tilde{y}_{k=y_j}$ to all other $K - 2$ classes. Finally, the model is trained by minimizing the cross-entropy loss with the new labels $\tilde{y}$ as the target.

B. Experiments on mixup

1) Setup: When applying AutoLabel to mixup, we set the number of distance buckets to be $N = 5$. The hyperparameter α in Eqn (1) is sweep in a set and we choose the best $\alpha = 0.005$ for CIFAR10 and $\alpha = 0.008$ for CIFAR100.

2) Results analysis: To test how well AutoLabel improves mixup [20] is more nuanced because mixup's baseline effectiveness is sensitive to its hyperparameters. In particular, the mixing parameter γ in Eqn (5) is sampled from a beta distribution Beta(β, β). When $\beta \to 0$, most sampled γ are close to 0 or 1; when $\beta = 1$, γ is randomly sampled from a uniform distribution. We observe that mixup suffers from a trade-off between accuracy and calibration on the clean data on CIFAR10 and CIFAR100, shown in Table V. When the hyperparameter β is large, e.g., $\beta = 1$, the model is trained on more diverse augmented data compared to a smaller β, e.g., $\beta = 0.2$. This results in a higher accuracy but leads the model to be too under-confident and a much larger calibration error on the clean data. This trade-off between clean accuracy and calibration of mixup is also observed in other datasets and networks in Figure 2(j) in [28]. After applying AutoLabel to mixup to automatically adjust the labels for augmented images, we find that the trade-off between accuracy and calibration is well addressed: high accuracy and low calibration error are achieved on the clean data, shown in Table V. However, this trade-off between accuracy and calibration does not exist in the large scale datasets, e.g., ImageNet, where $\beta = 0.2$ consistently has the best accuracy and calibration and we did not observe a significant improvement when applying AutoLabel to mixup on ImageNet.

TABLE V: Effects of AutoLabel for mixup on CIFAR10 and CIFAR100. All numbers reported are averaged over 4 independent runs and in %. Best highlighted in **bold**.

Method	CIFAR10		CIFAR100	
	Accuracy	ECE	Accuracy	ECE
Vanilla	95.6	2.6	79.5	6.1
mixup (β = 0.2)	96.2	0.8	80.8	1.8
mixup (β = 1)	96.5	5.3	80.9	5.5
+ AutoLabel (β = 1)	**96.7**	**0.6**	**81.1**	**1.2**

TABLE VI: Effects of AutoLabel for mixup on the corrupted datasets: CIFAR10-C and CIFAR100-C. All numbers reported are averaged over 4 independent runs and in %. Best highlighted in **bold**.

Method	CIFAR10-C		CIFAR100-C	
	Accuracy	ECE	Accuracy	ECE
mixup	81.1	8.8	55.6	11.2
+ AutoLabel	**81.3**	**8.5**	**56.9**	**10.0**

In addition, we also find that AutoLabel can improve accuracy and calibration of mixup when applying AutoLabel to mixup with $\beta = 1$ on the corrupted datasets, as shown Table VI. As we can see that the mean accuracy on CIFAR100-C over mixup with from 55.6% to 56.9% and reduce the mean calibration error from 11.2% to 10.0%.

C. Updates based on previous reviews

1) Previous Meta Reviews: The paper introduces a simple and interesting method that adaptively smoothes the labels of augmented data based on a distance to the "clean" training data. The reviewers have raised concerns about limited novelty, minor improvement over baselines, and insufficient experiments. The author's response was not sufficient to eliminate these concerns. The AC agrees with the reviewers that the paper does not pass the acceptance bar of ICLR.

2) Updates: There is a fundamental change of the new submission compared to the previous version in terms of methodology as well as empirical experiments. We apply AutoLabel to more types of data augmentation to discover its effectiveness for models' calibration and adversarial robustness, especially when highly corrupted augmented data are incorporated into training.