Augmentation Strategies for Learning with Noisy Labels

Kento Nishi [*†] Yi Ding[*‡] Alex Rich [‡] Tobias Höllerer [‡]

[†]Lynbrook High School, San Jose CA, USA
[‡]University of California Santa Barbara, Santa Barbara CA, USA
kento24gs@outlook.com, yding@cs.ucsb.edu, anrich@cs.ucsb.edu, holl@cs.ucsb.edu

Abstract

Imperfect labels are ubiquitous in real-world datasets. Several recent successful methods for training deep neural networks (DNNs) robust to label noise have used two primary techniques: filtering samples based on loss during a warm-up phase to curate an initial set of cleanly labeled samples, and using the output of a network as a pseudo-label for subsequent loss calculations. In this paper, we evaluate different augmentation strategies for algorithms tackling the "learning with noisy labels" problem. We propose and examine multiple augmentation strategies and evaluate them using synthetic datasets based on CIFAR-10 and CIFAR-100, as well as on the real-world dataset Clothing1M. Due to several commonalities in these algorithms, we find that using one set of augmentations for loss modeling tasks and another set for learning is the most effective, improving results on the state-of-the-art and other previous methods. Furthermore, we find that applying augmentation during the warm-up period can negatively impact the loss convergence behavior of correctly versus incorrectly labeled samples. We introduce this augmentation strategy to the state-of-the-art technique and demonstrate that we can improve performance across all evaluated noise levels. In particular, we improve accuracy on the CIFAR-10 benchmark at 90% symmetric noise by more than 15% in absolute accuracy, and we also improve performance on the Clothing1M dataset.

1. Introduction

Data augmentation is a common method used to expand datasets and has been applied successfully in many computer vision problems such as image classification [32] and object detection [28], among many others. In particular, there has been much success using learned augmentations such as AutoAugment [6] and RandAugment [7] which do not require an expert who knows the dataset to curate augmentation policies. It has been shown that incorporating augmentation policies during training can improve generalization and robustness [12, 8]. However, few works have explored their efficacy for the domain of learning with noisy labels (LNL) [21].

Many techniques which tackle the LNL problem make use of the network memorization effect, where correctly labeled data fit before incorrectly labeled data as discovered by Arpit et al. [2]. This phenomenon was successfully explored in Deep Neural Networks (DNNs) through modeling the loss function and the training process, leading to the development of approaches such as loss correction [29] and sample selection [10]. Recently, the incorporation of MixUp augmentation [35] has dramatically improved the ability for algorithms to tolerate higher noise levels [1, 14].

While many existing works use the common random flip and crop image augmentation which we refer to as *weak augmentation*, to the best of our knowledge, no work at the time of writing has explored using more aggressive augmentation from learned policies such as AutoAugment during training for LNL algorithms. These stronger augmentation policies include transformations such as rotate, invert, sheer, etc. We propose to incorporate these stronger augmentation policies into existing architectures in a strategic way to improve performance. Our intuition is that for any augmentation technique to succeed, they must *(1)* improve the generalization of the dataset and *(2)* not negatively impact the loss modeling and loss convergence behavior that LNL techniques rely on.

With this in mind, we propose an augmentation strategy we call Augmented Descent (AUGDESC) to benefit from data augmentation without negatively impacting the network memorization effect. Our idea for AUGDESC is to use two different augmentations: a weak augmentation for any loss modeling and pseudo-labeling task, and a strong augmentation for the back-propagation step to improve gen-

[*]Equal contribution

Source code is available at https://github.com/KentoNishi/Augmentation-for-LNL.

eralization.

In this paper, we propose and examine how we can incorporate stronger augmentation into existing LNL algorithms to yield improved results. We provide some answers to this problem through the following contributions:

- We propose an augmentation strategy, Augmented Descent, which demonstrates state-of-the-art performance on synthetic and real-world datasets under noisy label scenarios. We show empirically that this can increase performance across all evaluated noise levels (Section 4.4). In particular, we improve accuracy on the CIFAR-10 benchmark at 90% symmetric noise by more than 15% in absolute accuracy, and we also improve performance on the real-world dataset Clothing1M (Section 4.5).
- We show that there is a large effect on performance depending on how augmentation is incorporated into the training process (Section 4.2). We empirically determine that it is best to use weaker augmentation during earlier epochs followed by stronger augmentations to not adversely affect the memorization effect. We analyze the behavior of loss distribution to yield insight to guide effective incorporation of augmentation in future work (Section 4.3).
- We evaluate the effectiveness of our augmentation methodology by performing generalization studies on existing techniques (Section 4.7). Without tuning any hyperparameters, we were able to improve existing techniques with only the addition of our proposed augmentation strategy by up to 5% in absolute accuracy.

2. Related Work

Learning with Noisy Labels The most recent advances in training with noisy labels use varying strategies of *(1)* selecting or heavily weighting a subset of clean labels during training [20, 13, 10, 5], or *(2)* using the output predictions of the DNN or an additional network to correct the loss [25, 22, 9, 29, 19].

Many methods use varying strategies of training two networks, using the output of one or both networks to guide selection of inputs with clean labels. Decoupling [20] maintains two networks during training, updating their parameters using only inputs which the two networks disagree on. MentorNet [13] pre-trains an extra network and uses the pre-trained network to apply weights to cleanly labeled inputs more heavily during training of a student network. Co-teaching [10] maintains two networks, and feeds the low-loss inputs of each network to its peer for parameter updating. The low-loss inputs are expected to be clean, following the finding that DNNs fit to the underlying clean distribution before overfitting to noisy labels [2]. INCV [5] trains two networks on mutually exclusive partitions of the training dataset, then uses cross-validation to select clean inputs. INCV uses the Co-teaching architecture for its networks. The main drawback of these strategies is they only utilize a subset of the information available for training.

The second category of techniques attempts to use the model's output prediction to correct the loss at training time. One such common method is to estimate the noise transition matrix and use it to correct the loss, as in forward and backward correction [22] and S-Model [9]. Another common method is to linearly combine the output of the network and the noisy label for calculating loss. Bootstrap [25] replaces labels with a combination of the label and the prediction from the DNN. Joint Optimization [29] uses a similar approach to the work in [25], but adds a term to the loss to optimize the correction of noisy labels. D2L [19] monitors the dimensionality of subspaces during training, using it to guide weighting of a linear combination of output prediction and noisy label during loss calculation.

Optimized Augmentation Augmentation of training data is a widely used method for improving generalization of machine learning models. Recent works such as AutoAugment [6] and RandAugment [7] have focused on studying which augmentation policies are optimal. AutoAugment uses reinforcement learning to determine the selection and ordering of a set of augmentation functions in order to optimize validation loss. To remove the search phase of AutoAugment and therefore reduce training complexity, RandAugment drastically reduces the search space for optimal augmentations and uses grid search to determine the optimal set. Both techniques are widely used in semi-supervised settings.

In semi-supervised learning settings, augmentation has been successfully applied to consistency regularization [26, 31, 3, 27]. In consistency regularization, a loss is applied to minimize the difference in network prediction between two versions of the same input during training. [26] uses a mixture of augmentation, random dropout, and random max-pooling to produce these two versions. More recently, unsupervised data augmentation [31] and ReMixMatch [3] minimize the network predictions between a strongly augmented and weakly augmented version of the input. All of these findings motivate us to incorporate strong augmentation within the realm of LNL to improve performance.

The semi-supervised learning problem itself is similar to the LNL problem with the subtle difference that some labels are unknown rather than corrupt. As techniques in semi-supervised learning have been able to make predictions on a larger dataset from a smaller clean dataset, it would be logical that LNL techniques would benefit from the generalization effects of augmentation. In fact, the recent semi-supervised techniques MixUp [35], and Luo et al. [18] all exhibit strong robustness to label noise.

Most recently, FixMatch [27] successfully combines strong vs. weak augmentation in consistency regularization with pseudo-labeling to achieve state-of-the-art results in semi-supervised classification tasks. While we similarly employ two separate pools of augmentation functions for use in downstream tasks, there are key important differences. Most notably, our key idea is separating augmentations used during loss analysis from augmentations used during back-propagation, rather than focusing on pseudo-labeling and consistency regularization. Additionally, we apply this idea to LNL, a separate domain with different considerations. We experimentally show improvements for a wide variety of LNL algorithms and demonstrate improvements on both synthetic and real-world datasets.

3. Method

We first describe how various algorithms operate within the context of the network memorization effect [2]. We then propose the Augmented Descent strategy for filtering and generating pseudo-labels for high confidence samples based on one set of augmentations, then performing gradient descent on a different set of augmentations. Lastly, we provide an example for how to retrofit existing techniques.

3.1. Loss Modeling Under Noisy Label Scenarios

For some training data $D = (x_i, y_i)_{i=1}^{N}$, a classifier can be trained to make predictions using the cross entropy loss:

$$l(\theta) = -\sum_{x,y \in D} y^T \log(h_\theta(x)),$$

where h_θ is the function approximated by a neural network. Fundamentally, many algorithms are exploiting the behavior outlined in Arpit et al. [2] which finds that correctly labeled data tends to converge before incorrectly label data when training neural networks.

Many existing algorithms are then employing some degree of "pseudo-labeling", where the network is using its own guesses to approximate the labels for the remainder of the dataset. This is done by encouraging the learning of high confidence (or lower initial loss) samples via filtering or modifications to the loss function.

For example, in the sample selection technique Co-teaching [10], this is accomplished by feeding low-loss samples to a sister network, training the networks on data which it believes is correct. Abstractly, this would create two datasets from the input for each training epoch of what is believed to be correctly labeled $C = arg\,min_{D:|D| \geq R(T)|D|} l(f, D)$, where $R(T)$ is a threshold for the number of samples to place into the clean set determined empirically by the loss behavior, and incorrectly labeled $I = D \setminus C$. Using these sets, we obtain the loss:

$$l(\theta) = -\sum_{x,y \in C} y^T \log(h_\theta(x)) - 0 * \sum_{x,y \in I} y^T \log(h_\theta(x)).$$

Here, the learning process is ignoring samples which are believed to be incorrectly labeled as the training progresses. This is represented by the 0 term multiplied into what the model believes to be incorrect samples.

By contrast, Arazo et al. [1] accomplishes noise tolerance by incorporating the network's own prediction into its loss as a weighted sum based on the confidence determined by a mixture model fit to the previous epoch's losses, enabling a softer incorporation of the labels:

$$l(\theta) = -\sum_{x,y \in D, w \in W} (1-w) y^T \log(h_\theta(x)) - \sum_{x \in D, w \in W} w z^T \log(h_\theta(x)),$$

where W is a set of weights learned using a beta mixture model and z is the model's prediction for input x. More recently, DivideMix [14] combines these ideas and assigns weights to inputs to incorporate network guesses, separates the input into two sets, and trains with the resulting data in a semi-supervised manner using MixMatch [4].

With this understanding, we propose Augmented Descent (AUGDESC) for LNL techniques that employ loss modeling to separate correctly labeled from incorrectly labeled data. We propose to use one augmentation of the input for sample loss modeling and categorization to create the hypothetical sets C and I or to determine the pseudo label z, while utilizing another different augmentation as input to the network h_θ for purposes of back-propagation. This would require twice the number of forward passes during training for each input. The goal of this is so that we do not adversely affect any loss modeling but also be able to inject more generalization during the learning process. We provide an example in section 3.4 for how we can incorporate AUGDESC into DivideMix.

3.2. Augmentation Strategies

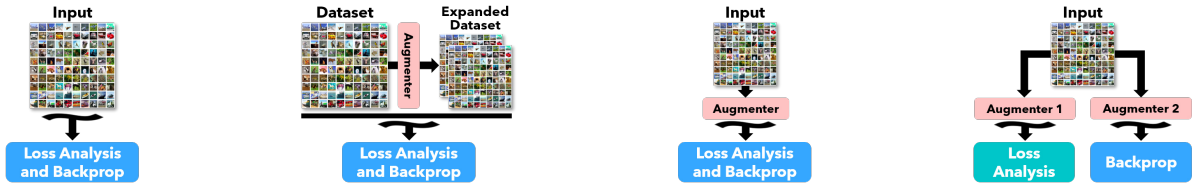
We examine the following strategies for incorporating augmentation into existing algorithms. Figure 1 presents a conceptual representation for incorporating our augmentation strategy into existing techniques.

Raw: Original image is used without any modifications.

Dataset Expansion: A dataset is created that is twice the original size of the dataset. This is then fed directly into the model without further augmentation.

Runtime Augmentation: Images are transformed before being fed into network at runtime.

Augmented Descent (AUGDESC): Two sets of augmented images are created. One set is used for any loss



(a) Raw (b) Dataset Expansion (c) Runtime (d) Augmented Descent

Figure 1: Visualization of training methods when incorporating different augmentation strategies. Raw takes the input directly and feeds it into the model for loss analysis and back-propagation. Dataset expansion first creates an expanded dataset which is then sampled by batches and fed into the network. Runtime Augmentation applies a random augmentation policy during runtime for each sampled batch. Augmented Descent produces two sets of random augmentations at the batch level: one is used for all loss analysis tasks, and the other is used for gradient descent.

Algorithm 1: Batch level training modifications to DivideMix for Augmented Descent. Full implementation provided in the supplemental.

Input: θ^1, θ^2, training batch possibly labeled x, possibly unlabeled u, dataset labels y, gmm probabilities w, number of augmentations M, augmentation policies Augment_1 and Augment_2

$x^{desc} = \text{Augment}_2(x)$

$u^{desc} = \text{Augment}_2(u)$

for m = 1 to M

$x = \text{Augment}_1(x)$

$u = \text{Augment}_1(u)$

end // co-guessing and sharpening

$p = \frac{1}{M} \sum_m p_{model}(x; \theta^{(k)})$

$\bar{y} = wy + (1-w)p$

$\hat{y} = Sharpen(y, T)$

$\bar{q} = \frac{1}{2M} \sum_m (p_{model}(\hat{u}; \theta^{(1)}) + p_{model}(\hat{u}; \theta^{(2)}))$

$\hat{q} = Sharpen(\bar{q}, T)$

// train using a different augmentation

$\hat{\mathcal{X}} = \{(x, y) | x \in x^{desc}, y \in \hat{y}\}$

$\hat{\mathcal{U}} = \{(u, q) | u \in u^{desc}, q \in \hat{q}\}$

$\mathcal{L}_x, \mathcal{L}_u$ = MixMatch($\hat{\mathcal{X}}, \hat{\mathcal{U}}$)

$\mathcal{L} = \mathcal{L}_x + \lambda_u \mathcal{L}_u + \lambda_r \mathcal{L}_{reg}$

$\theta^{(k)}$ =SGD($\mathcal{L}, \theta^{(k)}$)

analysis tasks, while the other is used for gradient descent. The motivation is that we can learn a better representation for each image while not compromising the sample filtering and pseudo-labeling process.

3.3. Augmentation Policy

We evaluate three different augmentation policies, classified into "weak" and "strong". Many algorithms make use of the standard random crop and flip for augmentation [16]. We call this process *weak augmentation*. We experiment with *strong augmentations* using automatically learned policies from AutoAugment [6] and RandAugment [7]. AutoAugment and RandAugment both provide a way to apply augmentations without hand-tuning the particular policy. Our strong augmentation policy first applies a random crop and flip, followed by an AutoAugment or RandAugment transformation, and lastly normalization. For dataset expansion and runtime augmentation, we experiment with both weak and strong augmentations.

We examine three variants of Augmented Descent. AUGDESC-WW means we perform loss analysis using a weakly-augmented input, then use this label to train a different weakly augmented version of the same input. Similarly, AUGDESC-SS represents strongly-augmented loss analysis, coupled with strongly augmented gradient descent. Finally, AUGDESC-WS corresponds to weakly-augmented loss analysis with strongly augmented optimization.

Because AutoAugment is learned on a small subset of the actual data, it is easy to incorporate into existing architectures. We further perform an ablation study using RandAugment to show that our augmentation strategy is agnostic to augmentation policy, as well as the fact that no dataset-specific or pre-trained augmentations are necessary. We use AutoAugment for most of our experiments as it prescribes a pre-trained set of policies, while RandAugment requires tuning that can depend on the networks used as well as the training set size.

3.4. Application to State of the Art

While many techniques beyond those above have similar characteristics that we can analyze in a similar manner, we examine this augmentation strategy within the context of the current state-of-the-art DivideMix [14] in this paper. We then extend our augmentation strategy to other techniques and report results in the experiments section.

DivideMix incorporates aspects of warm-up, co-training[13, 10], and MixUp [35]. The original DivideMix algorithm works by first warming up using normal cross-entropy loss with a penalty for confident predictions by adding a negative cross entropy term from Pereyra et al.

[23]. Afterwards, for each training epoch, the algorithm first uses a GMM to model the per-sample loss with each of the two networks. Using this and a clean probability threshold, the network then categorizes samples into a labeled set x and an unlabeled set u. Batches are pulled from from each of these two sets and are first augmented. Predictions using the augmented samples are made and a sharpening function is applied to the output [4] to reduce the entropy of the label distribution. This produces sharpened guesses for the labeled and unlabeled inputs which is used for optimization.

We outline the application of our augmentation strategy in Algorithm 1. We require two different sets of augmentations: one for the original DivideMix pipeline, and one to augment the original input for training with MixMatch losses. Additional examples of implementation in previous techniques are included in the supplemental.

4. Experiments

We first perform evaluations on synthetically generated noise to determine an effective augmentation strategy. We then conduct generalization experiments on real-world datasets, apply our strategies to previous techniques, and experiment with alternative augmentation policies.

4.1. Experimental Setup

We perform extensive validation of each augmentation technique on CIFAR-10 and CIFAR-100, two well-known synthetic image classification datasets frequently used for this task. CIFAR-10 contains 10 categories of images and CIFAR-100 contains 100 categories for classification. Each dataset has 50K color images for training and 10K test images of size 32x32. Symmetric and asymmetric noise injection methods [29, 15] are evaluated. We perform most of the ablation studies within the DivideMix framework as this is the state-of-the-art technique. We then extend the augmentation strategies we found to other techniques.

We use an 18-layer PreAct Resnet [11] as the network backbone and train it using SGD with a batch size of 128. Some experiments are conducted using a batch size of 64 due to hardware constraints but consistency is maintained in the comparisons. We conduct the experiments using the method outlined in [14] with all the same hyperparameters: a momentum of 0.9, weight decay of 0.0005, and trained for roughly 300 epochs depending on the speed of convergence. The initial learning rate is set to 0.02 and reduced by a factor of 10 after roughly 150 epochs. Warm-up periods where applicable are set to 10 epochs for CIFAR-10 and to 30 epochs for CIFAR-100. We keep the number of augmentations parameter $M = 2$ fixed for a fair comparison.

4.2. Comparison of Augmentation Strategies

We examine the performance of each proposed augmentation strategy outlined in Section 3.2 using DivideMix as our baseline model. We investigate the performance impact on lower label noise (20%) and very high label noise (90%) for some performance bounds. We report results in Table 1.

Method/Noise		CIFAR-10 20%	CIFAR-10 90%	CIFAR-100 20%	CIFAR-100 90%
Raw	Best	85.94	27.58	52.24	7.99
	Last	83.23	23.92	39.18	2.98
Expansion-W	Best	90.86	31.22	57.11	7.30
	Last	89.95	10.00	53.29	2.23
Expansion-S	Best	90.56	35.10	55.15	7.54
	Last	89.51	34.23	54.37	3.24
Runtime-W [14]	Best	96.10	76.00	77.30	31.50
	Last	95.70	75.40	76.90	31.00
Runtime-S	Best	**96.54**	70.47	**79.89**	40.52
	Last	**96.33**	70.22	79.40	40.34
AugDesc-WW	Best	96.27	36.05	78.90	30.33
	Last	96.08	23.50	78.44	29.88
AugDesc-SS	Best	96.47	81.77	79.79	38.85
	Last	96.19	81.54	**79.51**	38.55
AugDesc-WS	Best	96.33	**91.88**	79.50	**41.20**
	Last	96.17	**91.76**	79.22	**40.90**

Table 1: Performance differences for each augmentation strategy. The best performance in each category is highlighted in bold. Removing all augmentation is highly detrimental to performance, while more augmentation seemingly improves performance. However, too much augmentation is also detrimental to performance (AugDesc-SS). Strategically adding augmentation by exploiting the loss properties (AugDesc-WS) yields the best results in general.

As shown in the table, there is a large effect on algorithm performance based on how augmentations are included. While in some aspects this is unsurprising, what is surprising is the huge effect augmentation can have with regards to higher noise datasets. In the best case, we see AUGDESC-WS at 90% noise achieve results on CIFAR-10 close to accuracies reported on augmentation techniques with 20% label noise. For CIFAR-100, we also witness a large effect with higher noise rates but it remains a challenging benchmark for noisy datasets. Overall, we find that AugDesc-WS achieves the strongest result across the board.

It should be noted that a vast number of image-based machine learning algorithms incorporate some level of weak augmentation (flip, crop, and normalization) during training time. For completeness, we retrospectively examine the effect of removing these augmentations to tease out the effect of augmentation, i.e. the raw input method. We see that including some very small amount of augmentation is hugely beneficial, particularly evident when examining the transition from raw to weak augmentation at runtime.

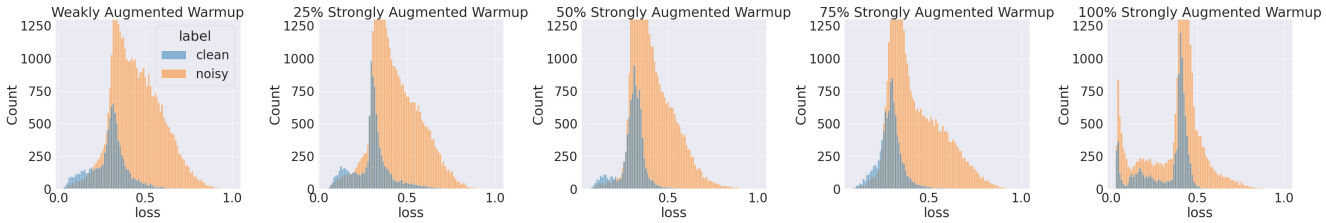


Figure 2: Effect of augmentation strength on the distribution of normalized loss for noisy versus clean segments of the dataset during warm-up for 90% label noise. Too much augmentation can cause samples in the clean dataset to be have higher loss, causing lower loss in samples from the noisy dataset.

		CIFAR-10					CIFAR-100			
Model	Noise	20%	50%	80%	90%	40% Asym	20%	50%	80%	90%
DivideMix (baseline) [14]	Best	96.1	94.6	92.3	76.0	93.4	77.3	74.6	60.2	31.5
	Last	95.7	94.4	92.9	75.4	92.1	76.9	74.2	59.6	31.0
DM-AugDesc-WS-SAW	Best	**96.3**	**95.6**	93.7	35.3	94.4	**79.6**	**77.6**	61.8	17.3
	Last	**96.2**	**95.4**	**93.6**	10.0	94.1	**79.5**	**77.5**	61.6	15.1
DM-AugDesc-WS-WAW	Best	**96.3**	95.4	**93.8**	**91.9**	**94.6**	79.5	77.2	**66.4**	**41.2**
	Last	**96.2**	95.1	**93.6**	**91.8**	**94.3**	79.2	77.0	**66.1**	**40.9**

Table 2: Application of strong versus weak augmentation during the warm-up period of DivideMix, in comparison to the baseline model. WAW signifies weakly augmented warm-up, SAW represents strongly augmented warm-up. Weak warm-up appears to benefit datasets with higher noise while strong warm-up benefits datasets with lower noise.

4.3. Effect of Augmentation During Warm-up

LNL algorithms generally rely on fact that clean samples are fit before noisy ones. To take advantage of such a property, many algorithms create scheduled learning or tune the loss function, explicitly designating warm-up period to exploit the label noise learning property [1, 14, 34]. We test the effect of introducing augmentation before and after this period by comparing the performance of models injected with augmentations from the first epoch and models trained with augmentations after the designated warm-up period.

We report performance metrics in Table 2 for various noise levels. We find that injecting strong augmentations during the warm-up period in low noise datasets benefit performance, but is detrimental when the dataset becomes increasingly noisy. This is particularly evident when examining the 90% noise rate. Conversely, weakly augmented warm-up greatly increases performance at higher noise levels.

To better understand why this is, we perform an experiment by stochastically applying strong augmentation to each batch with increasing chance to observe its distribution at epoch 20. Figure 2 shows the loss distribution for samples in the training set associated with the clean versus the noisy dataset. We find that applying too much augmentation too soon can encourage lower noise data to have too high of a loss and noisy data to have lower loss.

4.4. Synthetic Dataset Summary Results

We report the summary results in Table 3. The results show that augmenting the state-of-the-art algorithm using our best augmentation strategy increases accuracy across all noise levels. In particular, the improvement for extremely noisy datasets (90%) is very large, and approaches the best performance of lower noise datasets and represents an error reduction of 65%. For comparison, we achieve 91% accuracy for 90% symmetric noise on the CIFAR-10 dataset while the previous state of the art achieves 96.1% on only 20% label noise. Furthermore, we achieve an over 15% improvement in accuracy over previous state of the art for CIFAR-10 at 90% label noise.

4.5. Clothing1M Performance

Clothing1M [30] is a large-scale real-world dataset containing 1 million images obtained from online shopping websites. Labels are generated by extracting tags from the surrounding texts and keywords, and are thus very noisy. A ResNet-50 with pre-trained ImageNet weights are used following the work of [15]. We applied the pre-trained ImageNet AutoAugment augmentation policy for this task.

We report results in table 4. Our augmentation strategy obtained state-of-the-art performance when utilizing a strongly augmented warm-up cycle. In addition to obtaining competitive results, this further indicates that the noise level is likely to be below 80% based on our previous experiments, as strong warm-up improves accuracy. This is in

Model	Noise	CIFAR-10 20%	CIFAR-10 50%	CIFAR-10 80%	CIFAR-10 90%	CIFAR-100 20%	CIFAR-100 50%	CIFAR-100 80%	CIFAR-100 90%
Cross-Entropy	Best	86.8	79.4	62.9	42.7	62.0	46.7	19.9	10.1
	Last	82.7	57.9	26.1	16.8	61.8	37.3	8.8	3.5
Reed et. al. [24]	Best	86.8	79.8	63.3	42.9	62.1	46.6	19.9	10.2
	Last	82.9	58.4	26.8	17.0	62.0	37.9	8.9	3.8
Yu et al. [34]	Best	89.5	85.7	67.4	47.9	65.6	51.8	27.9	13.7
	Last	88.2	84.1	45.5	30.1	64.1	45.3	15.5	8.8
Zhang et al. [35]	Best	95.6	87.1	71.6	52.2	67.8	57.3	30.8	14.6
	Last	92.3	77.6	46.7	43.9	66.0	46.6	17.6	8.1
Yi & Wu [33]	Best	92.4	89.1	77.5	58.9	69.4	57.5	31.1	15.3
	Last	92.0	88.7	76.5	58.2	68.1	56.4	20.7	8.8
Li et al. [15]	Best	92.9	89.3	77.4	58.7	68.5	59.2	42.4	19.5
	Last	92.0	88.8	76.1	58.3	67.7	58.0	40.1	14.3
Arazo et al. [1]	Best	94.0	92.0	86.8	69.1	73.9	66.1	48.2	24.3
	Last	93.8	91.9	86.6	68.7	73.4	65.4	47.6	20.5
Li et al. [14]	Best	96.1	94.6	92.9	76.0	77.3	74.6	60.2	31.5
	Last	95.7	94.4	92.3	75.4	76.9	74.2	59.6	31.0
DM-AugDesc-WS-SAW	Best	**96.3**	**95.6**	93.7	35.3	**79.6**	**77.6**	61.8	17.3
	Last	**96.2**	**95.4**	**93.6**	10.0	**79.5**	**77.5**	61.6	15.1
DM-AugDesc-WS-WAW	Best	**96.3**	95.4	**93.8**	**91.9**	79.5	77.2	**66.4**	**41.2**
	Last	**96.2**	95.1	**93.6**	**91.8**	79.2	77.0	**66.1**	**40.9**

Table 3: Performance comparison when incorporating our best augmentation strategy into the current state-of-the-art. Our augmentation strategy improves performance at every noise level. Results for previous techniques were directly copied from their respective papers.

Method	Test Accuracy
Cross Entropy	69.21
M-correction [1]	71.00
Joint Optimization [29]	72.16
MetaCleaner [36]	72.50
MLNT [15]	73.47
PENCIL [33]	73.49
DivideMix [14]	74.76
ELR+ [17]	74.81
DM-AugDesc-WS-WAW (ours)	74.72
DM-AugDesc-WS-SAW (ours)	**75.11**

Table 4: Comparison against state-of-the-art methods for accuracy on the Clothing1M dataset.

Method/Noise		CIFAR-10 20%	CIFAR-10 90%	CIFAR-100 20%	CIFAR-100 90%
Baseline [14]	Best	96.1	76.0	77.3	31.5
	Last	95.7	75.4	76.9	31.0
AutoAugment	Best	**96.3**	**91.9**	**79.5**	**41.2**
	Last	**96.2**	**91.8**	**79.2**	**40.9**
RandAugment	Best	96.1	89.6	78.1	36.8
	Last	96.0	89.4	77.8	36.7

Table 5: Comparison of different automated augmentation policy algorithms. We compare performance of each policy using the AugDesc-WS approach. Adjusting the augmentation policy has minimal effect but still handily outperforms the runtime augmentation used in the baseline. The improved performance is still large with a noise ratio of 90%.

concordance with the estimates of the noise level of Clothing1M, said to be approximately 61.54% [30].

4.6. Automatic Augmentation Policies

In our evaluation benchmarks, we primarily used AutoAugment pre-trained policies. These policies are trained on a small subset of the original dataset with regards to CIFAR-10 and CIFAR-100 (5000 samples). We do this due to the simplistic nature of integrating pre-trained AutoAugment policies. For completeness, we evaluate whether we can achieve similar performance with an untrained set of augmentations, as theoretically we could then tune policies based on validation accuracy. To do this, we examine whether we can achieve performance on-par with AutoAugment using RandAugment [7], which can be tuned by adjusting 2 parameters. For these experiments, we used $N = 1$ and $M = 6$ for RandAugment hyperparameters.

We report results in Table 5. As shown in the table, RandAugment can achieve performance on-par with AutoAugment with minimal tuning and demonstrates the validity of our approach. Furthermore, since we were able to outperform the state-of-the-art on Clothing1M while using a pre-trained ImageNet AutoAugment policy for the task, optimizing an AutoAugment policy on Clothing1M could potentially yield better results.

4.7. Generalization to Previous Techniques

Based on our evaluations, we find that a weakly augmented warm-up period followed by the application of strong augmentation works best. Furthermore, it is beneficial to perform the loss analysis process on a weakly augmented input, then forwarding a strongly augmented input through the network for training. We apply our most effective augmentation strategy to previous techniques to evaluate generalizability of our approach.

We choose to compare to Cross-Entropy, Co-Teaching+[34], M-DYR-H [1], and DivideMix [14] due to the range of techniques these algorithms employ. Co-Teaching+ uses two networks and thresholding to exploit the memorization effect and is an updated work based on the popular Co-Teaching [10] technique. M-DYR-H uses mixture models to fit the loss to previous epochs to weight the models predictions using a single network. DivideMix is the current state-of-the-art which combines these and brings in a semi-supervised learning framework.

All source code for each evaluated technique was available publicly published by the original authors. We follow the hyperparameters and models outlined in the original published paper and apply no tuning of our own. This demonstrates the ease at which augmentations can be incorporated without delicate tuning of hyperparameters, highlighting the generalizability of our approach. We detail the exact algorithm modifications for inserting augmentations in the supplemental of this paper. We perform the evaluation on a lower noise setting (20%) as many previous techniques did not perform well at high noise levels. Table 6 shows the performance of our evaluation.

For vanilla cross-entropy, we used RUNTIME-S since as there is no warm-up period. For other techniques, we applied the AUGDESC-WS-WAW strategy. We evaluated our augmentation strategy on these algorithms as they cover a range of general approaches to learning with label noise. Some differences in performance are larger than expected due to the specific implementation of network architecture and synthetic noise generation techniques. We attempted strongly augmented warm-up for Co-teaching and found that there was a very large detrimental impact to performance. This agrees with our earlier observation that too much augmentation during the warm-up period can be detrimental. In particular, it appears to have a strong impact on the way noisy and clean data converge during the warm-up period, which these algorithms typically rely on.

The AUGDESC-WS-WAW strategy and even augmentation in general benefits performance in multiple categories (Table 6). As the experiments conducted were with no tuning of hyperparameters, we expect that further improvements can be seen when tuning with augmentation in mind due to the ways in which these algorithms exploit the loss distributions. Additionally, we see that across the board, the average performance of the last few epochs with augmentation is better than performance without. This indicates that using our augmentation strategy aids in learning a better distribution.

		CIFAR-10		CIFAR-100	
		Base	**Aug**	**Base**	**Aug**
Cross Entropy	Best	86.8	**89.9**	60.2	**61.2**
	Last	82.7	**85.1**	59.9	**60.4**
Co-Teaching+ [34]	Best	59.3	**60.6**	**26.2**	25.6
	Last	55.9	**57.4**	23.0	**23.7**
M-DYR-H [1]	Best	**94.0**	93.9	68.2	**73.0**
	Last	93.8	**93.9**	67.5	**72.7**
DivideMix	Best	96.1	**96.3**	77.3	**79.5**
	Last	95.7	**96.2**	76.9	**79.2**

Table 6: Performance benefits when applying our augmentation strategy to previous techniques at 20% noise level. Baseline and augmented accuracy scores are reported.

5. Conclusion

In this paper, we propose and examine the effect of various augmentation strategies within the domain of learning with label noise. We find that it is advantageous to add additional augmentation, particularly for higher noise ratios. Furthermore, copious amounts of augmentation during warm-up periods should be avoided if the noise rate is high, as this can have detrimental effects on the property that neural networks fit clean data before noisy data [2]. We performed extensive studies and found that the AUGDESC-WS strategy is capable of producing improvements across all noise levels and in multiple datasets. We further show its generalization capabilities by applying it to previous techniques with demonstrated success. This is additional evidence for how using two separate pools of augmentation operations for two separate tasks in these machine learning algorithms can be beneficial. This idea has previously been demonstrated to be effective in SSL settings [27], and we now show this for LNL settings.

In summary, we examined where it is advantageous to incorporate varying degrees of augmentation, and were able to demonstrate a strategy to advance the state-of-the-art as well as improve the performance of previous techniques. We hope the insights regarding the strength and amount of augmentation will be beneficial for future applications of augmentation when developing LNL algorithms.

6. Acknowledgements

This work was supported in part by ONR awards N00014-19-1-2553 and N00174-19-1-0024, as well as NSF awards IIS-1911230 and IIS-1845587. We would also like to thank Dr. Lina Kim and all those involved with the UCSB RMP program.

References

[1] Eric Arazo, Diego Ortego, Paul Albert, Noel E O'Connor, and Kevin McGuinness. Unsupervised label noise modeling and loss correction. *arXiv preprint arXiv:1904.11238*, 2019.

[2] Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. A closer look at memorization in deep networks. In *ICML*, 2017.

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

[4] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems*, pages 5049–5059, 2019.

[5] Pengfei Chen, Ben Ben Liao, Guangyong Chen, and Shengyu Zhang. Understanding and utilizing deep neural networks trained with noisy labels. In *ICML*, 2019.

[6] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 113–123, 2019.

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

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

[9] Jacob Goldberger and Ehud Ben-Reuven. Training deep neural-networks using a noise adaptation layer. In *ICLR*, 2017.

[10] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *NeurIPS*, pages 8535–8545, 2018.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

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

[13] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *ICML*, 2018.

[14] Junnan Li, Richard Socher, and Steven CH Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. *arXiv preprint arXiv:2002.07394*, 2020.

[15] Junnan Li, Yongkang Wong, Qi Zhao, and Mohan S Kankanhalli. Learning to learn from noisy labeled data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5051–5059, 2019.

[16] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[17] Sheng Liu, Jonathan Niles-Weed, Narges Razavian, and Carlos Fernandez-Granda. Early-learning regularization prevents memorization of noisy labels. *arXiv preprint arXiv:2007.00151*, 2020.

[18] Yucen Luo, Jun Zhu, Mengxi Li, Yong Ren, and Bo Zhang. Smooth neighbors on teacher graphs for semi-supervised learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8896–8905, 2018.

[19] Xingjun Ma, Yisen Wang, Michael E. Houle, Shuo Zhou, Sarah M. Erfani, Shu-Tao Xia, Sudanthi Wijewickrema, and James Bailey. Dimensionality-driven learning with noisy labels. In *ICML*, 2018.

[20] Eran Malach and Shai Shalev-Shwartz. Decoupling "when to update" from "how to update". In *NIPS*, 2017.

[21] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *Advances in neural information processing systems*, pages 1196–1204, 2013.

[22] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: a loss correction approach. In *CVPR*, 2017.

[23] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.

[24] Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*, 2014.

[25] Scott E. Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. In *ICLR*, 2015.

[26] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Advances in Neural Information Processing Systems*, 2016.

[27] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *arXiv preprint arXiv:2001.07685*, 2020.

[28] Kihyuk Sohn, Zizhao Zhang, Chun-Liang Li, Han Zhang, Chen-Yu Lee, and Tomas Pfister. A simple semi-supervised learning framework for object detection. *arXiv preprint arXiv:2005.04757*, 2020.

[29] Daiki Tanaka, Daiki Ikami, Toshihiko Yamasaki, and Kiyoharu Aizawa. Joint optimization framework for learning

with noisy labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5552–5560, 2018.

[30] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2691–2699, 2015.

[31] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le1. Unsupervised data augmentation for consistency training. In *NeurIPS*, 2020.

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

[33] Kun Yi and Jianxin Wu. Probabilistic end-to-end noise correction for learning with noisy labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7017–7025, 2019.

[34] Xingrui Yu, Bo Han, Jiangchao Yao, Gang Niu, Ivor W Tsang, and Masashi Sugiyama. How does disagreement help generalization against label corruption? *arXiv preprint arXiv:1901.04215*, 2019.

[35] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[36] Weihe Zhang, Yali Wang, and Yu Qiao. Metacleaner: Learning to hallucinate clean representations for noisy-labeled visual recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7373–7382, 2019.