Identifying Training Stop Point with Noisy Labeled Data

Sree Ram Kamabattula and Venkat Devarajan
Department of Electrical Engineering
The University of Texas at Arlington, USA
sreeram.kamabattula@mavs.uta.edu
venkat@uta.edu

Babak Namazi and Ganesh Sankaranarayanan
Baylor Scott and White Health, Dallas, USA
babak.namazi@bswhealth.org
ganesh.sankaranarayanan@bswhealth.org

Abstract—Training deep neural networks (DNNs) with noisy labels is a challenging problem due to over-parameterization. DNNs tend to essentially fit on clean samples at a higher rate in the initial stages, and later fit on the noisy samples at a relatively lower rate. Thus, with a noisy dataset, the test accuracy increases initially and drops in the later stages. To find an early stopping point at the maximum obtainable test accuracy (MOTA), recent studies assume either that i) a clean validation set is available or ii) the noise ratio is known, or, both. However, often a clean validation set is unavailable, and the noise estimation can be inaccurate. To overcome these issues, we provide a novel training solution, free of these conditions. We analyze the rate of change of the training accuracy for different noise ratios under different conditions to identify a training stop region. We further develop a heuristic algorithm based on a small-learning assumption to find a training stop point (TSP) at or close to MOTA. To the best of our knowledge, our method is the first to rely solely on the training behavior, while utilizing the entire training set, to automatically find a TSP. We validated the robustness of our algorithm (AutoTSP) through several experiments on CIFAR-10, CIFAR-100, and a real-world noisy dataset for different noise ratios, noise types and architectures.

Index Terms—Memorization stages, TSP, MOTA.

I. INTRODUCTION

DNNs have achieved remarkable performance in several computer vision domains due to the availability of large datasets and advanced deep learning techniques. However, in areas such as medical imaging, vast amount of data is still unlabeled. Labeling by experts is time-consuming and expensive, and is often performed by crowd sourcing [28], online queries [3], etc. Such datasets introduce a large number of noisy labels, caused by errors during the labeling process. This work focuses on label noise. Therefore, noise and label noise are used interchangeably throughout the paper.

Training DNNs robustly with noisy datasets has become challenging and prominent work [1], [27]. While DNNs are robust to a certain amount of noise [20], they have the high capacity to fit on the noisy samples [2], [30]. However, the performance of the DNNs declines as they begin to learn on a significant number of noisy samples.

One notable approach to deal with this problem is to select and train on only clean samples from the noisy training data. In [4], authors check for inconsistency in predictions. O2UNet [9] uses cyclic learning rate and gathers loss statistics to select clean samples. SELF [16] takes ensemble of predictions at different epochs.

Few methods utilize two networks to select clean samples. MentorNet [10] pre-trains an extra network to supervise a StudentNet. Decoupling [15] trains two networks simultaneously, and the network parameters are updated only on the examples with different predictions. Co-teaching [7] updates one network’s parameters based on the peer network’s small-loss samples. Co-teaching is further improved in co-teaching+[29].

Small-loss observation suggests that clean samples have smaller loss compared to noisy samples in the early stages of training. However, in the later stages, it is harder to distinguish clean and noisy samples based on the loss distribution. Therefore, it is desirable to stop the training in the initial stages. In [13], authors theoretically show that the DNNs can be robust to noise despite over-parameterization with early-stopping, when trained with a first order derivative-based optimization method.

However, identifying the point, where the network starts to fit on noisy samples, is a challenging problem. Most previous methods assume that either i) the noise ratio is known or ii) the noise type is known. For instance, when the noise ratio is unknown, in [5], authors propose a cross-validation approach to estimate the noise based on the validation accuracy, which becomes inaccurate for higher number of classes and harder datasets. In [23], the authors assume that a clean validation set is available to find the early stopping point. However, often, a clean validation set is not available. In any case, [23] suggests that the overall performance of the network is sensitive to the validation set quality. Therefore, a clean or noisy validation set is not reliable to identify the best stop point.

On the other hand, we attempt to stop the training before the network learns on a significant amount of noisy samples without any of the above assumptions. We propose a novel heuristic approach called AutoTSP to identify a TSP by monitoring the training behavior of the DNN. Briefly, our main contributions are:

- We define different memorization stages using clean and
We analyze the positive and negative rates of change of training accuracy to define a training stop region.

We further develop a heuristic algorithm to identify the TSP that does not require a clean validation set, and is independent of noise estimation and noise type.

We show that AutoTSP can be incorporated in other clean sample selection methods [5], [7] to further improve the performance.

We validate AutoTSP on two clean benchmark datasets: CIFAR-10 and CIFAR-100. Since CIFAR-10 and CIFAR-100 are clean, we synthetically add noise by changing the true label $y$ to a different label $y'$ with a noise transition matrix $N_{ij}$. We employ two types of noise to evaluate our robustness: 1) Symmetric (Sym): The true label is changed to any other class symmetrically with rate $\tau$. ($N_{ij} = \frac{\tau}{n-1}$ for $i \neq j$). 2) Asymmetric (Asym): Noise is distributed with just one class, we change the true label to the label of the following class with rate $\tau$. Asymmetric datasets are more realistic, as the labeler is likely to confuse one class with another class. We also validate our algorithm on a real world noisy dataset ANIMAL-10N.

We perform several experiments with different architectures: 9-layer CNN (CNN9), ResNet32 and ResNet110 [8] to show our algorithm on a real world noisy dataset ANIMAL-10N.

Several related methods assign higher weights to clean samples for effective training [19]. Few other works develop noise-robust loss functions [25], [31], [26]. Few methods attempt to correct the noisy labels. Bootstrap-related work training but requires human supervision.

CleanNet [11] produces weights for each sample during network training but requires human supervision.

We validate AutoTSP on two clean benchmark datasets: CIFAR-10 and CIFAR-100. Since CIFAR-10 and CIFAR-100 are clean, we synthetically add noise by changing the true label $y$ to a different label $y'$ with a noise transition matrix $N_{ij}$. We employ two types of noise to evaluate our robustness: 1) Symmetric (Sym): The true label is changed to any other class symmetrically with rate $\tau$. ($N_{ij} = \frac{\tau}{n-1}$ for $i \neq j$). 2) Asymmetric (Asym): Noise is distributed with just one class, we change the true label to the label of the following class with rate $\tau$. Asymmetric datasets are more realistic, as the labeler is likely to confuse one class with another class. We also validate our algorithm on a real world noisy dataset ANIMAL-10N.

We perform several experiments with different architectures: 9-layer CNN (CNN9), ResNet32 and ResNet110 [8] to show our algorithm on a real world noisy dataset ANIMAL-10N.

Several related methods assign higher weights to clean samples for effective training [19]. Few other works develop noise-robust loss functions [25], [31], [26]. Few methods attempt to correct the noisy labels. Bootstrap-related work training but requires human supervision.

CleanNet [11] produces weights for each sample during network training but requires human supervision.

We validate AutoTSP on two clean benchmark datasets: CIFAR-10 and CIFAR-100. Since CIFAR-10 and CIFAR-100 are clean, we synthetically add noise by changing the true label $y$ to a different label $y'$ with a noise transition matrix $N_{ij}$. We employ two types of noise to evaluate our robustness: 1) Symmetric (Sym): The true label is changed to any other class symmetrically with rate $\tau$. ($N_{ij} = \frac{\tau}{n-1}$ for $i \neq j$). 2) Asymmetric (Asym): Noise is distributed with just one class, we change the true label to the label of the following class with rate $\tau$. Asymmetric datasets are more realistic, as the labeler is likely to confuse one class with another class. We also validate our algorithm on a real world noisy dataset ANIMAL-10N.

We perform several experiments with different architectures: 9-layer CNN (CNN9), ResNet32 and ResNet110 [8] to show our algorithm on a real world noisy dataset ANIMAL-10N.
The $e_{mem}$ separates the PM and MM regions by calculating the argmax of the ratio of label recall, i.e. the highest gap between the blue and the orange line as shown in the top plots of Fig[1] The $e_{sm}$ separates MM and SM regions by monitoring the point at which the ratio drops progressively. In some cases, with a high capacity network, the $e_{mem}$ and $e_{sm}$ might coincide, because the network can fit noisy examples at a high rate as shown in the right plot of Fig[3].

In the PM stage, before $e_{mem}$, the unshaded region in Fig[1] the learning is essentially on clean samples and thus the increase in the training accuracy is proportional to the test accuracy. In particular, the $LR_{clean}$ increases at a higher rate, whereas the $LR_{noisy}$ remains minimal. In the MM stage (between $e_{mem}$ and $e_{sm}$), the rate at which $LR_{clean}$ increases drops and learning on noisy labels $LR_{noisy}$ increases gradually. Thus, the proportionality between training and test accuracy begins to attenuate. Later, in the SM region, as $LR_{noisy}$ keeps increasing, the proportionality completely fails, as the learning is mostly on noisy labels. In other words, the test accuracy keeps dropping as the training accuracy increases. This can be seen in the bottom plots of Fig[1]. The network learns on higher number of noisy samples in the SM stage, and it might not learn significant number of clean samples in the PM stage. Therefore, the desired training stopping point should be between $e_{mem}$ and $e_{sm}$.

However, $e_{mem}$ and $e_{sm}$ are obtained with the ground truth. Therefore, we utilize the learning behavior, specifically, the rate of change of training accuracy at each memorization stage to find the desired TSP without the ground truth.

**IV. IDEAL TIME TO STOP TRAINING**

We monitor the rate of change of training accuracy $y_{f}^{(e)}$ at each epoch $e$, which can be either positive or negative. We represent the magnitude of rate of change at epoch $e$ by $pr^{(e)}$ if positive, or $nr^{(e)}$ if negative. We separate all the $pr^{(e)}$ and $nr^{(e)}$ into vectors $PROCE$ and $NROCE$ respectively. Now, we observe the variance of magnitudes of the $PROCE$ and $NROCE$ in different memorization stages.

In the PM stage, both $pr^{(e)}$ and $nr^{(e)}$ have high magnitude consistently. Thus, we can observe higher spikes in the unshaded region in Fig[1]. Therefore, we assume there would be very few epochs with small-magnitude initially. We refer to these epochs with small-magnitude as small learning epochs (SLEs). As the network begins to fit on the noisy samples, in the MM region, the magnitudes of both $pr^{(e)}$ and $nr^{(e)}$ decrease, so there would be higher number of SLEs. We would also observe fewer $nr^{(e)}$ epochs, resulting in a smoothness in the training accuracy curve near the SM region. As the network severely memorizes the noisy labels in the SM region, there would still be higher number of SLEs. But, towards the end, as the network fits on all the training examples, the rate tends to increase slightly, causing the oscillation towards the end in Fig[1]. The magnitude in the SM region varies with different settings, but, the drop in the magnitude near the SM region (smoothness) is consistent across different architectures, datasets, noise types and noise ratios as shown in later figures.

From our analysis in the previous section, the desired training stop point would be between $e_{mem}$ and $e_{sm}$. Therefore, we identify this smoothness i.e. find the initial reduction in the $PROCE$ and $NROCE$ to choose a training stop region.

We divide the $PROCE$ and $NROCE$ into intervals of fixed length $\beta_{1}$ and calculate the cumulative sum of magnitudes in each interval. Then, we standardize (zero mean and unit variance) these sums and identify the interval, where the standardized value becomes less than zero. Two different reduction points will be obtained for PROCE and NROCE (labeled as posline and negline in figures). We define this region between the two points as the training stop region.

We plot the NROCE, PROCE along with the test accuracy in Fig[2]. The blue asterisk points in the figure are located at the epoch numbers in NROCE and PROCE, spaced apart by length $\beta_{1}$. The height of the blue asterisk points represents the cumulative sum at each interval. The red vertical dotted line is plotted at the point, where the reduction happens.

The small-learning belief can be seen in Fig[2] i.e. the asterisk points are higher in the beginning and drop in the later stages. It can be observed that the red dotted line on both the figures are close to the MOTA. Therefore, this supports our training stop region consideration.

Fig[3] shows plots for the higher noise ratio with a different architecture. The PROCE asterisk points are higher initially, but subsequently vary in the SM stage as discussed earlier. However, as the smoothness is consistent, the red dotted lines of both PROCE and NROCE are still close to MOTA.

Since, our belief depends on the rate of change of training accuracy, we check the sensitivity of our belief on different learning rate schedules. The top plot in Fig[4] are of 0.001 initial rate and multiplied by 0.5, 0.25, 0.1 at 80, 120, 160 epochs respectively (LR2). The bottom plot is of constant learning rate 0.001 (LR3). The reduction of sums in the later stages can still be observed in both cases. This further verifies the robustness of our training stop region.
We define an epoch as a small-learning epoch (SLE), if the samples at SLEs, which results in a drop in test accuracy. In this stage, we assume that the network learns on the noisy samples in the later stages. Since, the test accuracy decreases in the later stages. Initially, there would be very few SLEs as discussed earlier, there would be higher number of SLEs in MT initially. In contrast, there would be higher number of SLEs in the later stages. Since, the test accuracy decreases in the later stages, we assume that the network learns on the noisy samples at SLEs, which results in a drop in test accuracy. We define an epoch as a small-learning epoch (SLE), if the magnitude value $MT^{(e)}$ is less than the threshold value $\theta_1$. We negate the values of these SLEs in $MT$ to indicate that the test accuracy is dropped at these epochs. To identify the TSP, we determine whether the epochs following the SLEs compensate the drop in the test accuracy at SLEs, and achieve higher test accuracy. Therefore, we define a vector $C$ to calculate the cumulative sum of the SLEs and the following epochs until the sum becomes positive, which we believe indicates the increase in test accuracy. We store the cumulative sum at epoch $e$ in $C(e)$. We set the cumulative sum to zero, if the sum at epoch $e$ is negative, to indicate that the test accuracy is not increasing. Now, we calculate the sums of consecutive values in $C$ within the training stop region and store them in vector $V$. We consider the highest cumulative sum point (argmax) in $V$ as the training stop point $\epsilon_{tsp}$.

V. Experiments

We use the Adam optimizer and a momentum of 0.9. The batch size is set to 128 and the network is trained for 200 epochs. The initial learning rate is set to 0.001 and multiplied by 0.5, 0.25, 0.1 at 20, 30 and 40 epochs respectively for all experiments, otherwise mentioned. We set the algorithm hyper-parameters $\beta_1$ to 5, 6 and 7 denoted by $\{5,6,7\}$, and $\theta_1$, $\theta_2$ to 0.5 based on experimental observations.
A. Performance Comparison of AutoTSP:

In this section, we compare the AutoTSP when all else are equal: use of entire training data, assumption that a clean validation set is not available. Thus, we compare our method only with 1) MOTA, point where the maximum test accuracy is obtained, as represented in the figures by the green vertical line. 2) Standard training, i.e., test accuracy at the end of the epochs without early stopping (labeled as Standard in the tables). 3) Additionally, we also compare with Noise Heuristic Accuracy (NHA) [22], which assumes that the noise ratio $\tau$ is known, an assumption, which we believe should be avoided. NHA suggests that during training on a $\tau$ percent noisy dataset, the best point to stop is when the training accuracy reaches $(1 - \tau)$ percent.

We measure the performance of AutoTSP to the above-mentioned methods with test accuracy (acc), label precision (LP) and label recall (LR) of the training data $(LP = \frac{S_{\text{clean}}^{(\text{step})}}{S_{\text{etsp}}^{(\text{step})}}$, $LR = \frac{S_{\text{clean}}^{(\text{step})}}{D_{\text{clean}}}$).

Table I reports the results on ResNet110 architecture. It can be seen that the AutoTSP is either exact or close to MOTA across different noise ratios, datasets and noise types. It can also be observed that AutoTSP outperforms NHA in most of the cases. The results also confirm that the standard training performs poorly when training with noisy labels.

On CIFAR-10 dataset, with the easier 0.2 Sym noise, it can be seen that the AutoTSP is able to exactly identify the MOTA point. High LP indicates that the learning on noisy samples is minimal, and high LR represents that the network learned significant number of clean samples at the TSP. The significance of AutoTSP can be observed for the harder cases. The AutoTSP still finds the exact stop point at MOTA, when the noise ratio is increased to 0.5, as shown in the top left plot of Fig. 5. It performs similarly for both asymmetric cases as shown in the bottom left plot. On the other hand, NHA and standard achieve higher LR than the MOTA, because the network stops the training in the later stages, where the number of clean samples learned is higher. But, lower LP indicates that they learn on higher number of noisy samples due to the stop point in the SM region, which is not desired.

Results on the harder dataset CIFAR-100 are similar to CIFAR-10 for Sym as shown in the top righ plot of Fig. 5.
where the network does not fit all the training samples. Results are presented in Table III. The networks learning drops significantly when fitting on noisy samples and thus the PROCE is close to MOTA in Fig. 4. Similar to CNN9 case, AutoTSP identifies the beginning of memorization region as shown in Fig. 7. However, we can still observe that the AutoTSP works fairly well on both CIFAR-10 and CIFAR-100, and outperforms NHA on both Sym and Asym cases.

**ANIMAL-10N:** We also conducted experiments on a real world noisy ANIMAL-10N dataset with 50000 training and 5000 testing samples. The noise is caused by human labeling error on 5 pairs of confusing classes and the approximate noise rate is 8%. It can be seen that AutoTSP still exactly identifies MOTA with CNN9 in the left plot of Fig 8. With ResNet32 in the right plot, the AutoTSP stops a little earlier than the MOTA, where the accuracy at MOTA was observed to be 79.6 and AutoTSP accuracy was 78.4. It can also be observed that the NHA stops the training deep in the SM region. The promising results on this real world noisy dataset validate the robustness of AutoTSP.

**Clean dataset:** We further validated our method on two benchmark clean datasets CIFAR-10 and CIFAR-100 with CNN9 architecture. In general, we utilize a validation dataset and stop the training when the validation accuracy does not improve for a few epochs. Since, utilizing a validation set is not reliable [23], as an alternative, we use the AutoTSP to find the TSP on a clean dataset. It can be observed from Fig. 9 that, the AutoTSP is able to stop the training when the test accuracy no longer improves.

**LR2 and LR3:** Additionally, we trained CIFAR-10 0.5 Sym with ResNet32 using LR2 and LR3 discussed with Fig. 4. It can be observed that the AutoTSP performs well on both cases. AutoTSP finds the beginning of memorization region with the LR2, as can be seen in the left plot if Fig 10. The accuracy at MOTA is 77.65 and the accuracy of AutoTSP is 75.53. Similarly, with LR3, it can be observed that the AutoTSP is very close to MOTA in the right plot. These results continue to validate our claim that the rate of change of training accuracy can be utilized to avoid learning on noisy labels.

### 2. Comparison of Test Accuracy with Other Baseline Methods:

For additional validation, we concatenate AutoTSP with the INCV and co-teaching method (referred as AutoTSP+INCV in Table IV) to obtain higher test accuracy. We compare the test accuracy with the following baselines: F-correction, Decoupling, Coteaching, MentorNet, D2L, INCV and the standard training method. Additionally, we also compare with the test accuracy obtained when trained only on the clean set of the noisy training data (Labeled as Train on clean in the table). The results are shown in Table IV.

The objective of AutoTSP is to find the best training stop point close to the MOTA. Thus, it is not fair to compare AutoTSP directly with the other baseline methods, whose objective is to improve the test accuracy by re-labeling the noisy samples or, selecting higher number of samples iteratively, etc. We utilize the INCV discard measure to remove a percentage of high-loss values at $\epsilon_{\text{tsp}}$ which are believed to be noisy samples. Then, we retrain on the remaining samples, find the $\epsilon_{\text{tsp}}$ and discard few more high-loss samples. So, the new discarded dataset will be less noisy than the given training dataset. Now, we train the network using co-teaching method.

### Table II: Comparison of Results with CNN9

|                  | CIFAR 10 | CIFAR 100 |
|------------------|----------|-----------|
|                  | Symmetry |          | Symmetry |          |
|                  | 0.2      | 0.5      | 0.2      | 0.5      |
| MOTA (acc)       | 89.63    | 70.97    | 97.31    | 51.18    |
| AutoTSP (acc)    | 89.63    | 70.97    | 97.31    | 51.18    |
| NHA (acc)        | 80.52    | 69.16    | 82.17    | 79.15    |
| Standard (acc)   | 72.49    | 64.52    | 77.29    | 56.88    |
| MOTA (LP)        | 97.78    | 95.48    | 96.93    | 93.35    |
| AutoTSP (LP)     | 97.78    | 95.48    | 96.93    | 93.35    |
| NHA (LP)         | 96.95    | 93.92    | 94.87    | 93.35    |
| Standard (LP)    | 94.35    | 80.06    | 92.03    | 89.29    |

### Table III: Comparison of Results with ResNet32

|                  | CIFAR 10 | CIFAR 100 |
|------------------|----------|-----------|
|                  | Symmetry |          | Symmetry |          |
|                  | 0.2      | 0.5      | 0.2      | 0.5      |
| MOTA (acc)       | 88.72    | 78.38    | 87.22    | 80.74    |
| AutoTSP (acc)    | 88.72    | 78.38    | 87.22    | 80.74    |
| NHA (acc)        | 80.42    | 77.82    | 86.21    | 77.09    |
| Standard (acc)   | 78.65    | 79.09    | 78.87    | 56.21    |
| MOTA (LP)        | 90.07    | 93.72    | 95.01    | 85.91    |
| AutoTSP (LP)     | 90.07    | 93.72    | 95.01    | 85.91    |
| NHA (LP)         | 94.87    | 93.72    | 95.01    | 85.91    |
| Standard (LP)    | 93.33    | 88.66    | 93.33    | 63.55    |

### Table IV: Comparison of Results with CIFAR-10 and CIFAR-100

|                  | Clean | Noise |
|------------------|-------|-------|
|                  | Symmetry |          | Symmetry |          |
|                  | 0.2      | 0.5      | 0.2      | 0.5      |
| MOTA (acc)       | 88.72    | 78.38    | 87.22    | 80.74    |
| AutoTSP (acc)    | 88.72    | 78.38    | 87.22    | 80.74    |
| NHA (acc)        | 80.42    | 77.82    | 86.21    | 77.09    |
| Standard (acc)   | 78.65    | 79.09    | 78.87    | 56.21    |

**Fig. 7.** AutoTSP comparison on CIFAR-10 with ResNet32: Left 0.2 Sym, Right 0.5 Sym

**Fig. 8.** AutoTSP comparison on ANIMAL-10N with: Left CNN9, Right ResNet32
noisy labels are memorized as discussed earlier. Thus, high LP is because, the AutoTSP would stop the training before several LR of AutoTSP+INCV is much less compared to INCV. This toTSP+INCV is higher. On CIFAR-100 it can be seen that the Asym 0.4 on CIFAR-10, as the LP and LR of AutoTSP+INCV is much less compared to INCV. This is because, the INCV discard measure.

The claims in Table [V] are supported by Table [V] For CIFAR-10 Sym, AutoTSP+INCV and INCV LP and LR are very similar. The significance of AutoTSP can be observed for CIFAR-100 as our accuracy is higher than INCV. This is because, INCV depends largely on the noise estimation and the noise estimation becomes inaccurate for higher number of classes. On the other hand, we find the TSP without using the noise information to adopt the INCV discard measure.

Fig. 10. AutoTSP comparison on CIFAR-10 dataset 0.5 Sym on ResNet32: Left LR2, Right LR3.

The significance of AutoTSP can be observed for CIFAR-100 as our AutoTSP + INCV and INCV accuracy are very close. The claims in Table IV are supported by Table V. For CIFAR-10, we can see that our AutoTSP + INCV and INCV accuracy are very close. The significance of AutoTSP can be observed for CIFAR-100 as our accuracy is higher than INCV. This is because, INCV depends largely on the noise estimation and the noise estimation becomes inaccurate for higher number of classes. On the other hand, we find the TSP without using the noise information to adopt the INCV discard measure.

The claims in Table IV are supported by Table V. For CIFAR-10 Sym, AutoTSP+INCV and INCV LP and LR are very similar. The significance of AutoTSP can be observed with Asym 0.4 on CIFAR-10, as the LP and LR of AutoTSP+INCV is higher. On CIFAR-100 it can be seen that the LR of AutoTSP+INCV is much less compared to INCV. This is because, the AutoTSP would stop the training before several noisy labels are memorized as discussed earlier. Thus, high LP is achieved for both symmetric noise cases on CIFAR-100.

VI. CONCLUSION

In this work, we have shown that the rate of change of training accuracy can be essential in understanding the behavior of learning on noisy labels. Our key idea is to monitor the smoothness in the training accuracy to identify the training stop region. We further proposed a heuristic algorithm to automatically identify a training stop point based on the small-learning assumption. Our algorithm does not require a clean validation set and does not depend on noise estimation or the type of noise. We conducted several experiments to validate the robustness of our algorithm. One drawback of early stopping, in general, is that the network does not train on the harder-clean samples learned in the later stages, which could improve the networks generalization performance. Therefore, one possible future work is to discard noisy samples effectively without depending on the noise estimation.

REFERENCES

[1] G. Algan and I. Ulusoy, “Image Classification with Deep Learning in the Presence of Noisy Labels: A Survey,” Arxiv Preprint arXiv:1912.05170, 2019. [Online]. Available: http://arxiv.org/abs/1912.05170
[2] D. Arpit, S. Jastrzębski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, and S. Lacoste-Julien, “A Closer Look at Memorization in Deep Networks,” in Proceedings of the 34th International Conference on Machine Learning (ICML), 2017.
[3] A. Blum, A. Kalai, and H. Wasserman, “Noise-tolerant learning, the parity problem, and the statistical query model,” Journal of the ACM, vol. 50, no. 4, pp. 506–519, 2003.
[4] H.-S. Chang, E. Learned-Miller, and A. McCallum, “Active Bias: Training More Accurate Neural Networks by Emphasizing High Variance Samples,” in Advances in Neural Information Processing Systems 30 (NIPS 2017), 2017, pp. 1002–1012.
[5] P. Chen, B. Liao, G. Chen, and S. Zhang, “Understanding and Utilizing Deep Neural Networks Trained with Noisy Labels,” in 36th International Conference on Machine Learning, ICML. International Machine Learning Society (IMLS), 2019.
[6] J. Goldberger and E. Ben-Reuven, “TRAINING DEEP NEURAL-NETWORKS USING A NOISE ADAPTATION LAYER,” in International Conference on Learning Representations, 2017.
[7] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. W. Tsang, and M. Sugiyama, “Co-teaching: Robust training of deep neural networks with extremely noisy labels,” in Advances in Neural Information Processing Systems, 2018.
[8] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778. [Online]. Available: http://image-net.org/challenges/LSVRC/2015/
[9] J. Huang, L. Qu, R. Jia, and B. Zhao, “O2U-Net: A simple noisy label detection approach for deep neural networks,” Proceedings of the IEEE International Conference on Computer Vision, vol. 2019-Octob, no. Iccv, pp. 3325–3333, 2019.
[10] L. Jiang, Z. Zhou, T. Leung, L.-J. Li, and L. Fei-Fei, “MentorNet: Learning Data-Driven Curriculum for Very Deep Neural Networks on Corrupted Labels,” in 35th International Conference on Machine Learning, ICML, vol. 5, 2018, pp. 3601–3620.

[11] K.-H. Lee, X. He, L. Zhang, L. Yang, and J. D. A. I. Research, “CleanNet: Transfer Learning for Scalable Image Classifier Training with Label Noise,” in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 5447–5456.

[12] J. Li, R. Socher, and S. C. H. Hoi, “DIVIDEMIX: LEARNING WITH NOISY LABELS AS SEMI-SUPERVISED LEARNING,” in International Conference on Learning Representations (ICLR), 2020.

[13] M. Li, M. Soltanolkotabi, and S. Oymak, “Gradient Descent with Early Stopping is Provably Robust to Label Noise for Overparameterized Neural Networks,” arXiv preprint arXiv:1903.11680, 2019. [Online]. Available: http://arxiv.org/abs/1903.11680

[14] X. Ma, Y. Wang, M. E. Houle, S. Zhou, S. M. Erfani, S.-T. Xia, S. Wijewickrema, and J. Bailey, “Dimensionality-Driven Learning with Noisy Labels,” in 35th International Conference on Machine Learning, ICML 2018, vol. 8. International Machine Learning Society (IMLS), 2018, pp. 5332–5341.

[15] E. Malach and S. Shalev-Shwartz, “Decoupling "when to update" from "how to update,"” in Advances in Neural Information Processing Systems 30 (NIPS 2017), 2017, pp. 960–970.

[16] D. T. Nguyen, C. K. Mummadi, T. P. N. Ngo, T. H. P. Nguyen, L. Beggel, and T. Brox, “Self: Learning to filter noisy labels with self-ensembling,” arXiv, pp. 1–15, 2019.

[17] G. Patrini, A. Rozza, A. K. Menon, R. Nock, and L. Qu, “Making Deep Neural Networks Robust to Label Noise: a Loss Correction Approach,” in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 1944–1952.

[18] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich, “Training Deep Neural Networks on Noisy Labels with Bootstrapping,” in 3rd International Conference on Learning Representations, ICLR 2015 – Workshop Track Proceedings. International Conference on Learning Representations, ICLR, 2015.

[19] M. Ren, W. Zeng, B. Yang, and R. Urtasun, “Learning to R eweight Examples for Robust Deep Learning,” in 35th International Conference on Machine Learning, ICML 2018, vol. 10. International Machine Learning Society (IMLS), 2018.

[20] D. Rolnick, A. Veit, S. Belongie, and N. Shavit, “Deep Learning is Robust to Massive Label Noise,” Arxiv Preprint arXiv:1705.10694, 2017.

[21] H. Song, M. Kim, and J.-G. Lee, “SELFIE: Refurbishing Unclean Samples for Robust Deep Learning,” in the 36th International Conference on Machine Learning (ICML), 2019.

[22] H. Song, M. Kim, D. Park, and J.-G. Lee, “Prestopping: How Does Early Stopping Help Generalization against Label Noise?,” arXiv:1911.08059, 2019. [Online]. Available: http://arxiv.org/abs/1911.08059

[23] Y. Sun, Y. Tian, Y. Xu, and J. Li, “Limited Gradient Descent: Learning with Noisy Labels,” IEEE Access, vol. 7, pp. 168 296–168 306, 2019.

[24] D. Tanaka, D. Ikami, T. Yamasaki, and K. Aizawa, “Joint Optimization Framework for Learning with Noisy Labels,” in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 5552–5560.

[25] X. Wang, Y. Hua, E. Kodirov, and N. M. Robertson, “IMAE for noise-robust learning: Mean absolute error does not treat examples equally and gradient magnitude’s variance matters,” in arXiv, 2019.

[26] Y. Wang, X. Ma, Z. Chen, Y. Luo, J. Yi, and J. Bailey, “Symmetric cross entropy for robust learning with noisy labels,” arXiv, pp. 322–330, 2019.

[27] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang, “Learning from Massive Noisy Labeled Data for Image Classification,” in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 2691–2699.

[28] Y. Yan, R. Rosales, G. Fung, R. Subramanian, and J. Dy, “Learning from multiple annotators with varying expertise,” Machine Learning, vol. 95, no. 3, pp. 291–327, 2014.

[29] X. Yu, B. Han, J. Yao, G. Niu, I. W. Tsang, and M. Sugiyama, “How does Disagreement Help Generalization against Label Corruption?” in 36th International Conference on Machine Learning, ICML. International Machine Learning Society (IMLS), 2019, pp. 7167–7173.

[30] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, “Understanding deep learning requires rethinking generalization,” 5th International Conference on Learning Representations, ICLR 2017 – Conference Track Proceedings, 2017.
VII. SUPPLEMENTARY MATERIAL

A. Choice of $\beta_1$:

To find the training stop region, we monitor the magnitude of PROCE and NROCE over different intervals and identify the point at which the magnitude decreases. Since, we are looking to identify the reduction of the rate of change at different intervals in the training process, a few smaller magnitude epochs in the same interval could result in finding a region too early. Thus, to ignore such an effect, we choose a large interval size to identify the consistent drop in magnitude. We also consider three different interval sizes for redundancy and compare the reduction point among the three intervals to choose the training stop region. We choose $\beta_1$ to be 3, 4, and 5 for our experiments and denote this case as $\{3, 4, 5\}$. Now, we consider the region between the $\min(\min(\text{NROCE}\{3, 4, 5\}), \max(\text{PROCE}\{3, 4, 5\}))$ and $\max(\max(\text{NROCE}\{3, 4, 5\}), \max(\text{PROCE}\{3, 4, 5\}))$ as the training stop region.

Table VI shows the accuracy results when we set $\beta_1$ to $\{3, 5, 6, 7\}$. It can be observed that the accuracy is similar in both cases. This shows that the algorithm is not very sensitive to different interval sizes. This result makes qualitative sense, because, the reduction happens at the same point in the training irrespective of the different interval sizes.

| TABLE VI | COMPARISON OF TEST ACCURACY AT THE TSP WITH DIFFERENT INTERVAL SIZES ON ResNet110 |
|------------------|--------------------------------|------------------------------------------------------------------------------------------------|
| CIFAR 10         | Symmetric | Asymmetric | CIFAR 10 | Symmetric | Asymmetric | CIFAR 100 | Symmetric | Asymmetric | CIFAR 100 | Symmetric | Asymmetric |
| AutoTSP ($\beta_1 = 0.2 \text{ acc}$) | 83.86 | 74.76 | 84.35 | 76.36 | 57.33 | 46.65 | 58.03 | 42.25 |
| AutoTSP ($\beta_1 = 0.5 \text{ acc}$) | 85.01 | 75.86 | 86.28 | 76.31 | 57.33 | 46.65 | 58.03 | 42.25 |
| AutoTSP ($\beta_1 = 0.8 \text{ acc}$) | 83.7 | 75.86 | 86.28 | 76.31 | 58.04 | 46.65 | 58.03 | 42.25 |
| AutoTSP ($\beta_1 = 0.2 \text{ LR}$) | 98.11 | 93.57 | 98.42 | 88.1 | 94.5 | 96.45 | 87.24 | 63.1 |
| AutoTSP ($\beta_1 = 0.5 \text{ LR}$) | 97.6 | 95.2 | 97.54 | 87.54 | 94.5 | 96.45 | 84.25 | 63.1 |
| AutoTSP ($\beta_1 = 0.8 \text{ LR}$) | 98.27 | 95.2 | 98.42 | 87.54 | 95.61 | 97.16 | 84.25 | 63.1 |
| AutoTSP ($\beta_1 = 0.2 \text{ LP}$) | 92.47 | 82.72 | 91.12 | 83.56 | 88.9 | 61.48 | 89.46 | 88.21 |
| AutoTSP ($\beta_1 = 0.5 \text{ LP}$) | 94.23 | 80.23 | 96.96 | 82.66 | 88.9 | 61.48 | 93.83 | 88.21 |
| AutoTSP ($\beta_1 = 0.8 \text{ LP}$) | 91.42 | 80.23 | 91.12 | 82.66 | 88.08 | 60.9 | 93.83 | 88.21 |

B. Small-learning assumption:

The small-learning assumption negates the magnitude when $MT(x)$ is less than the threshold $\theta_1$. Since, in the initial stages, there would be only a few SLEs, the following epochs would be able to compensate for the drop and achieve positive sum, meaning achieving higher test accuracy. But, in the later stages, there would be higher number of SLEs and fewer epochs with high magnitude. So, these fewer high magnitude epochs cannot overcome the large negative cumulative sum. Therefore, this results in several zeros in the later stages. Thus, to avoid these false positives, we only apply the small-learning assumption within the training stop region, assuming the training stop region truncates this SM region as shown in Fig 1.

C. Choice of $\theta_1$ and $\theta_2$:

To show the impact of the thresholds, we set $\theta_1$ to 0.2, 0.5, 0.8 and $\theta_2$ to 0, 0.5. Note that, if $\theta_1$ is set to zero the small-learning assumption is ignored.

Table VII shows accuracy results for different $\theta_1$ values. It can be observed that the results are similar for the three cases, showing that the algorithm is not very sensitive to the thresholds chosen. However, if the threshold is set even higher, it found a stop point too early in the training. So, the threshold size should increase with the epoch number, as, in the later stages, even the higher magnitude epochs result in learning on noisy labels. But, finding that relation is harder without knowing the noise rate. Both INCV [5] and co-teaching [7] utilize a threshold to select a percentage of loss values as small or high loss values. However, the two methods utilize the noise ratio in their thresholds. But our algorithm assumes that the noise rate is unknown. Thus, we observed the rate of change across different datasets, architectures and hyper-parameters to choose the threshold. Based on the observation, we set the ideal $\theta_1$ to be 0.5.

The results are similar for both $\theta_2$ values 0 and 0.5. $\theta_2$ helps when the training stop region finds an upper bound in the later region by further penalizing the small-magnitude epochs. This, $\theta_2$ can be ignored if threshold value is not kept constant and increased with the epoch number.

| TABLE VII | COMPARISON RESULTS OF TEST ACCURACY, LP AND LR WITH ResNet110 FOR DIFFERENT $\theta_1$ VALUES WHEN $\theta_2$ IS SET TO 0.5 |
|------------------|--------------------------------|------------------------------------------------------------------------------------------------|
| CIFAR 10         | Symmetric | Asymmetric | CIFAR 10 | Symmetric | Asymmetric | CIFAR 10 | Symmetric | Asymmetric |
| AutoTSP ($\theta_1 = 0.2 \text{ acc}$) | 83.86 | 74.76 | 84.35 | 76.36 | 57.33 | 46.65 | 58.03 | 42.25 |
| AutoTSP ($\theta_1 = 0.5 \text{ acc}$) | 85.01 | 75.86 | 86.28 | 76.31 | 57.33 | 46.65 | 58.03 | 42.25 |
| AutoTSP ($\theta_1 = 0.8 \text{ acc}$) | 83.7 | 75.86 | 86.28 | 76.31 | 58.04 | 46.65 | 58.03 | 42.25 |
| AutoTSP ($\theta_1 = 0.2 \text{ LP}$) | 97.6 | 95.2 | 97.54 | 87.54 | 94.5 | 96.45 | 84.25 | 63.1 |
| AutoTSP ($\theta_1 = 0.5 \text{ LP}$) | 98.27 | 95.2 | 98.42 | 87.54 | 95.61 | 97.16 | 84.25 | 63.1 |
| AutoTSP ($\theta_1 = 0.8 \text{ LP}$) | 91.42 | 80.23 | 91.12 | 82.66 | 88.08 | 60.9 | 93.83 | 88.21 |

D. Choosing a different training stop point:

Table VIII [XI] and [X] show the results for different architectures when a different training stop point is chosen within the training stop region. We implemented the last non-zero element within the training stop region as the training stop point instead of the maximum cumulative sum (argmax). However, we observed that argmax performs slightly better than the last non-zero element.

| TABLE VIII | COMPARISON RESULTS OF TEST ACCURACY, LP AND LR WITH ResNet110 FOR DIFFERENT TRAINING STOP POINTS |
|------------------|--------------------------------|------------------------------------------------------------------------------------------------|
| CIFAR 10         | Symmetric | Asymmetric | CIFAR 10 | Symmetric | Asymmetric |
| AutoTSP (last non-zero acc) | 83.86 | 74.76 | 84.35 | 76.36 | 57.33 | 46.65 | 58.03 | 42.25 |
| AutoTSP (last non-zero LP) | 97.6 | 95.2 | 97.54 | 87.54 | 94.5 | 96.45 | 84.25 | 63.1 |
| AutoTSP (last non-zero LR) | 98.27 | 95.2 | 98.42 | 87.54 | 95.61 | 97.16 | 84.25 | 63.1 |

E. Architecture description:

The 9-layer CNN architecture we used is described below in Table [XI].
### TABLE IX
Comparison results of test accuracy, LP and LR with CNN9 for different training stop points

|                  | CIFAR 10 | CIFAR 100 |
|------------------|----------|-----------|
|                  | Symmetric | Asymmetric | Symmetric | Asymmetric |
| AutoTSP (max acc) | 83.26     | 79.19     | 81.82     | 57.06      |
| AutoTSP (last non-zero acc) | 83.26     | 77.64     | 87.58     | 56.91      |
| AutoTSP (max LP)  | 98.99     | 95.48     | 97.72     | 94.41      |
| AutoTSP (last non-zero LP) | 98.99     | 93.95     | 98.93     | 98.62      |
| AutoTSP (max LR)  | 88.12     | 87.17     | 90.32     | 75.5       |
| AutoTSP (last non-zero LR) | 88.12     | 87.17     | 90.32     | 75.5       |

### TABLE X
Comparison results of test accuracy, LP and LR with ResNet32 for different training stop points

|                  | CIFAR 10 | CIFAR 100 |
|------------------|----------|-----------|
|                  | Sym | Asym | Sym | Asym |
| AutoTSP (max acc) | 80.42 | 77.82 | 86.21 | 77.09 |
| AutoTSP (acc)     | 80.42 | 77.39 | 86.21 | 77.09 |
| AutoTSP (max LP)  | 99.07 | 95.64 | 98.87 | 90.74 |
| AutoTSP (LP)      | 99.07 | 94.97 | 98.87 | 90.74 |
| AutoTSP (max LR)  | 84.3  | 81.58 | 91.79 | 80.9 |
| AutoTSP (LR)      | 84.3  | 84.56 | 91.79 | 80.9 |

### TABLE XI
9 Layer CNN Architecture

| CNN-9 |
|------------------|
| 3x3 conv,128 ReLu Batchnorm |
| 3x3 conv,128 ReLu Batchnorm |
| 3x3 conv,128 ReLu Batchnorm |
| 3x3 conv,256 ReLu Batchnorm |
| 3x3 conv,256 ReLu Batchnorm |
| 3x3 conv,256 ReLu Batchnorm |
| 3x3 conv,512 ReLu Batchnorm |
| 3x3 conv,512 ReLu Batchnorm |
|平均pool |
| Fully connected layer |