When Sparse Neural Network Meets Label Noise Learning: A Multi-Stage Learning Framework

Runqing Jiang, Yan Yan, *Member, IEEE,* Jing-Hao Xue, Biao Wang, and Hanzi Wang, *Senior Member, IEEE*

***Abstract*—Recent methods in network pruning have indicated that a dense neural network involves a sparse sub-network (called a winning ticket), which can achieve similar test accuracy to its dense counterpart with much fewer network parameters. Generally, these methods search for the winning tickets on well-labeled data. Unfortunately, in many real-world applications, the training data are unavoidably contaminated with noisy labels, thereby leading to performance deterioration of these methods. To address the above problem, we propose a novel Two-Stream Sample Selection Network (TS^3-Net), which consists of a sparse sub-network and a dense sub-network, to effectively identify the winning ticket with noisy labels. The training of TS^3-Net contains an iterative procedure that switches between training both sub-networks and pruning the smallest-magnitude weights of the sparse sub-network. In particular, we develop a multi-stage learning framework including a warm-up stage, a semi-supervised alternate learning stage, and a label refinement stage, to progressively train the two sub-networks. In this way, the classification capability of the sparse sub-network can be gradually improved at a high sparsity level. Extensive experimental results on both synthetic and real-world noisy datasets (including MNIST, CIFAR-10, CIFAR-100, ANIMAL-10N, Clothing1M, and WebVision) demonstrate that our proposed method achieves state-of-the-art performance with very small memory consumption for label noise learning. Code is available at https://github.com/Runqing-forMost/TS3-Net/tree/master.**

***Index Terms*—Deep Learning, network pruning, label noise learning, image classification.**

I. Introduction

RECENT advances in deep learning have demonstrated the great potential of Deep Neural Networks (DNN) in a variety of computer vision tasks, such as image classification [1], [2], object detection [3], [4], and so on. Nevertheless, many of them depend heavily on complex DNN models, which usually involve large memory consumption and high computational cost. The heavy resource burden of these methods severely inhibits their adoption in real-world applications, especially embedding systems and devices demanding small memory usage and low inference latency.

In order to reduce both memory storage and computational cost, considerable efforts [5]–[9] have been made to compress and accelerate DNN models without greatly sacrificing the final performance. Among these efforts, the network pruning technique [10], [11], which can significantly reduce the model size without greatly compromising accuracy, has currently attracted increasing attention from both academia and industry. Recent works [12], [13] have indicated that pruning DNN models at initialization can lead similar benefits to both training and inference. For example, the Lottery Ticket Hypothesis (LTH) [12] reveals that a randomly-initialized dense neural network contains a sparse sub-network whose architecture plays an important role in the accuracy and training efficiency of the original network. In particular, when the sparse sub-network is trained in isolation, it can achieve competitive performance to the original dense neural network. Note that conventional network pruning methods are often trained on well-labeled data to obtain pruned DNN models (i.e., sparse DNN models).

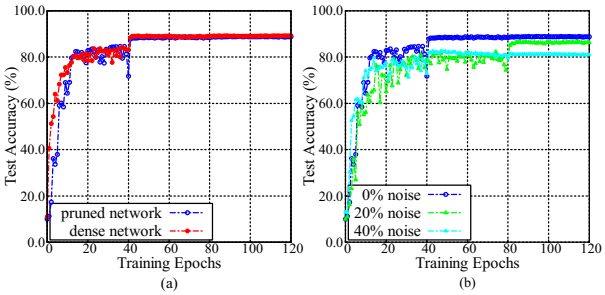

Fig. 1: (a) Test accuracy curves of the dense network vs. the pruned network with 78.4% sparsity on the CIFAR-10 dataset. (b) Test accuracy curves of the pruned network under different levels of symmetric label noise. All the networks are based on an eight-layer CNN models. LTH is used for network pruning.

It is commonly acknowledged that the availability of large-scale well-labeled data is of great importance to ensure the good performance of DNN models. Unfortunately, in many real-world scenarios, collecting large-scale data with clean labels is practically challenging, which unavoidably results in noisy labels [14], [15]. Some studies [16], [17] have shown that traditional DNN models are very sensitive to label noise and easily suffer from overfitting, leading to poor generalization. To handle the above problem, Label Noise Learning (LNL) methods have been proposed to learn robust DNN models with excellent generalization capability from noisy labeled data. Some methods develop robust loss functions (such as Symmetric Cross Entropy (SCE) [18]) to alleviate the overfitting problem, while other methods (such as Co-teaching [19] and Co-teaching+ [20]) explicitly remove potentially noisy labeled samples to achieve robustness.

The aforementioned LNL methods are often based on dense

R. Jiang, Y. Yan, H. Wang are with the Fujian Key Laboratory of Sensing and Computing for Smart City, School of Informatics, Xiamen University, Xiamen 361005, China (e-mail: jiangrunqing@stu.xmu.edu.cn; yanyan@xmu.edu.cn; hanzi.wang@xmu.edu.cn).

J.-H. Xue is with the Department of Statistical Science, University College London, London WC1E 6BT, UK (e-mail: jinghao.xue@ucl.ac.uk).

B. Wang is with the Zhejiang Lab, Hanzhou 311100, China (e-mail: wangbiao@zhejianglab.com).

DNN models (such as ResNet [21] and VGG-Net [22]). Few of them are concerned with robust learning on sparse DNN models. As shown in Fig. 1(a), we can see that the performances of the dense network and the pruned network are similar when they are trained on the clean dataset. However, as shown in Fig. 1(b), the performance of the pruned network significantly drops when it is trained on noisy labeled data. In other words, simply training network pruning methods with label noise cannot guarantee to obtain a robust sparse DNN model. Therefore, how to effectively combine label noise learning with network pruning merits further investigation.

In this paper, we propose a novel robust learning method called Two-Stream Sample Selection Network (TS^3-Net), consisting of a sparse sub-network and a dense sub-network, to search for a winning ticket given noisy labeled data. Generally, the whole training process of TS^3-Net involves an iterative procedure that alternates between training both sub-networks and pruning the smallest-magnitude weights of the sparse sub-network. In particular, we develop a multi-stage learning framework to train both sub-networks in a progressive way. Our framework contains three closely related stages: a warm-up stage, a semi-supervised alternate learning stage, and a label refinement stage. Therefore, we are able to fully exploit the whole training data to improve the classification capability of the sparse sub-network under the label noise condition.

In summary, our main contributions are given as follows:

- We propose a novel TS^3-Net to effectively identify the winning ticket in the case of label noise. Thus, a sparse sub-network with small memory consumption can be obtained when trained on noisy labeled data. To the best of our knowledge, we are the first to learn a robust sparse DNN model from the data corrupted with label noise.
- We develop a multi-stage learning framework to train the sparse sub-network and the dense sub-network progressively. By tightly combining the three stages in a unified framework, the sparse sub-network can gradually improve its classification capability at a high sparsity level during label noise learning.
- We evaluate our proposed method on both synthetic and real-world noisy datasets (including MNIST, CIFAR-10, CIFAR-100, ANIMAL-10N, Clothing1M, and WebVision) with different network architectures and different levels of sparsity. Without any whistles and bells, our method performs favorably against several state-of-the-art LNL methods with low memory usage for the image classification task. This clearly demonstrates that our method is able to achieve great generalization capability when dealing with label noise.

The rest of this paper is organized as follows. First, we briefly review the related work in Sec. II. Then, we present the details of our proposed method in Sec. III. Next, we evaluate the performance of our proposed method and compare it with several state-of-the-art LNL methods in Sec. IV. Finally, we conclude our work and discuss the future work in Sec. V.

II. Related Work

In this section, we briefly review the related work, including network pruning and label noise learning.

A. Network Pruning

Network pruning, which performs model compression by removing redundant parameters and structures from DNN models, has been widely studied in recent years. A variety of network pruning methods [10], [23], [24] have been developed to reduce a large number of network parameters without greatly influencing the performance. Generally, the gain on the inference efficiency of these methods depends on the special hardware support.

Recently, Frankle and Carbin [12] develop the Lottery Ticket Hypothesis (LTH). LTH states that a randomly-initialized and over-parameterized (dense) neural network contains a sparse sub-network (called a winning ticket), which can reach the similar test accuracy to the original network with almost the same number of iterations when trained in isolation. Hence, the time-consuming training process of a dense neural network can be avoided to a large extent since one just needs to find a good sparse sub-network and then trains it separately. Although it is not a trivial task to obtain such a sub-network, training a sparse sub-network is much easier than training a dense neural network with millions of parameters. Later, Frankle *et al.* [13] extend the application of LTH to a larger and deeper neural network by introducing the concept of "late resetting". Malach *et al.* [25] give theoretical evidence to prove the strong LTH. Girish *et al.* [26] incorporate LTH into various object recognition tasks. Chen *et al.* [27] study LTH for supervised and self-supervised pre-training in different computer vision models.

The above methods identify the winning tickets on large-scale training data with clean labels. Unfortunately, in many real-world applications, the training data are often contaminated with noisy labels since labeling large-scale data is labor-intensive and time-consuming. In this paper, we are concerned with the problem of searching for the winning tickets from data involving label noise.

B. Label Noise Learning

According to [17], state-of-the-art LNL methods can be roughly divided into: label transition matrix estimation, robust losses, sample weighting, sample selection, meta-learning, and combined methods. In this paper, we mainly review robust losses and sample selection.

Robust Losses. Several efforts have been dedicated to designing noise-robust loss functions, whose goal is to minimize the classification risk for unseen clean data when learning with label noise. Wang *et al.* [18] propose a Symmetric Cross Entropy (SCE) loss to alleviate the influence of noisy labeled samples by adding a noise-tolerant term to the cross-entropy loss. Zhou *et al.* [28] develop a class of asymmetric loss functions to combat label noise. Ma *et al.* [29] point out that existing robust loss functions easily suffer from the underfitting problem. Hence, they further propose a mutual boosted framework called Active Passive Loss (APL), which can create new loss functions with theoretically guaranteed robustness.

Sample Selection. To alleviate the negative influence of noisy labeled samples on model training, many methods [14], [19],

[20], [30] leverage sample selection that involves selecting clean samples from a noisy training dataset. Jiang *et al.* [30] propose MentorNet which employs a pre-trained teacher network to supervise the training of a student network. Han *et al.* [19] propose the Co-teaching method to simultaneously train two networks, where each network selects the small-loss samples and feeds them into its peer network for updating the parameters. Later, they extend Co-teaching to Co-teaching+ [20], which updates the networks by only using the small-loss samples from disagreement data (i.e., the samples giving different predictions from two networks). Different from Co-teaching+, Wei *et al.* [14] propose a robust learning paradigm called JoCoR to minimize the diversity of two networks. Jo-SRC [31] introduces the Jensen-Shannon divergence, which measures the likelihood of a sample being clean, to select clean samples globally.

Traditional sample selection methods explicitly reduce the risk of error accumulation from noisy labeled samples by removing these samples. Although the labels of noisy labeled samples are not reliable, these samples themselves are still informative and can be helpful for training. More recent works (such as DivideMix [32] and SELF [33]) make use of semi-supervised learning by considering the selected clean samples as labeled data and other samples as unlabeled data.

Most existing sample selection methods work on the dense DNN models. The high complexity of dense DNN models limits their applications demanding real-time response or interaction. In this paper, we design an effective sample selection method, which is able to learn a sparse DNN model from noisy training data.

III. Methodology

In this section, we first give an overview of the proposed method in Sec. III-A. Then, we illustrate our developed multi-stage learning framework in detail in Sec. III-B. Finally, we summarize the overall training of our method, and present some discussions in Sec. III-C and Sec. III-D, respectively.

A. Overview

In this paper, we develop a novel Two-Stream Sample Selection Network (TS^3-Net) to identify the winning ticket under label noise. The network architecture of TS^3-Net consists of a sparse sub-network (denoted as $\mathcal{S}$) and a dense sub-network (denoted as $\mathcal{D}$).

We follow the principle of the Iterative Magnitude Pruning (IMP) strategy [12] to train TS^3-Net, where we sparsify $\mathcal{S}$ at discrete time intervals with an increasing sparsity level during training. Generally, the training process involves the following iterative steps. First, $\mathcal{S}$ and $\mathcal{D}$ are trained to obtain network parameters. Second, a fixed fraction of network parameters of $\mathcal{S}$ is pruned with the smallest-magnitude weights. Third, the remaining parameters of $\mathcal{S}$ and $\mathcal{D}$ are reset to their original initializations. After the above iterative steps, a sparse sub-network (i.e., a winning ticket) is finally obtained and thus can be easily retrained for robust classification.

In particular, in the first step, we develop a multi-stage learning framework to train both sub-networks in a progressive way. More specifically, this framework contains three stages: (1) a warm-up stage, (2) a semi-supervised alternate learning stage, and (3) a label refinement stage. An illustration of the developed learning framework is given in Fig. 2.

In the first stage of warm-up, both $\mathcal{S}$ and $\mathcal{D}$ learn some preliminary knowledge from the whole training data without overfitting to noisy labeled samples. In the second stage, each sub-network teaches its peer sub-network by semi-supervised alternate learning. Given a batch of samples, each sub-network identifies the small-loss samples and noisy labeled ones (which are viewed as labeled samples and unlabeled ones, respectively). Based on this, each sub-network is updated by leveraging the samples from its peer sub-network in a semi-supervised learning manner, where a cross-entropy loss is optimized over labeled samples and an entropy loss is minimized over unlabeled samples. In the third stage of label refinement, some noisy labeled samples, whose maximum prediction results are larger than a given threshold, are assigned with pseudo-labels by relabeling. These relabeled samples and small-loss ones are then jointly used to update $\mathcal{S}$, where $\mathcal{D}$ is fixed. In this way, the classification performance of $\mathcal{S}$ is further boosted. In the end, with these three stages, our proposed learning framework enables $\mathcal{S}$ to effectively learn from the data corrupted with label noise by gradually enhancing its classification capability.

B. Multi-Stage Learning Framework

Suppose that we have a training dataset $\mathcal{T} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$ for a C-class classification problem, where $\boldsymbol{x}_i$ denotes the i-th training image, $\boldsymbol{y}_i \in \{0,1\}^C$ represents its corresponding label vector (an one-hot vector), and N is the total number of training samples. In real-world applications, the image labels are often corrupted by noise.

Generally, training the sparse sub-network by only using the standard cross-entropy loss easily suffers from the overfitting problem under label noise, and thus leads to poor generalization [19]. Meanwhile, traditional sample selection methods [14], [19], [20], [30] often consider the small-loss samples, and thus neglect noisy labeled samples which can also provide valuable information for classification. Moreover, the classification ability of $\mathcal{S}$ is weakened at a higher sparsity level. To address the above problems, we propose a multi-stage learning framework to progressively train the two sub-networks, thus guiding our model to gradually learn the knowledge from noisy labeled training samples.

Warm-Up Stage. Recent studies on the memorization effects [34] show that the DNN model is prone to first memorize the training data with easy and clean samples, and then focus on noisy labeled samples. Motivated by this observation, similar to [35], we first adopt a warm-up stage, which precedes the more sophisticated stages, to perform fully-supervised learning on the whole noisy dataset at the beginning of training.

Mathematically, given a batch of samples $\mathcal{B}$, we use the cross-entropy loss as the supervision to minimize the distribution gap between prediction results and ground-truth labels for both $\mathcal{S}$ and $\mathcal{D}$. The cross-entropy loss can be defined as

$$\mathcal{L}_{ce}(f_{\Theta_k}, \mathcal{B}) = -\frac{1}{n}\sum_{i=1}^{n} \boldsymbol{y}_i^{\mathrm{T}} \log(\boldsymbol{f}_{\Theta_k}(\boldsymbol{x}_i)), k \in \{\mathcal{S}, \mathcal{D}\} \quad (1)$$

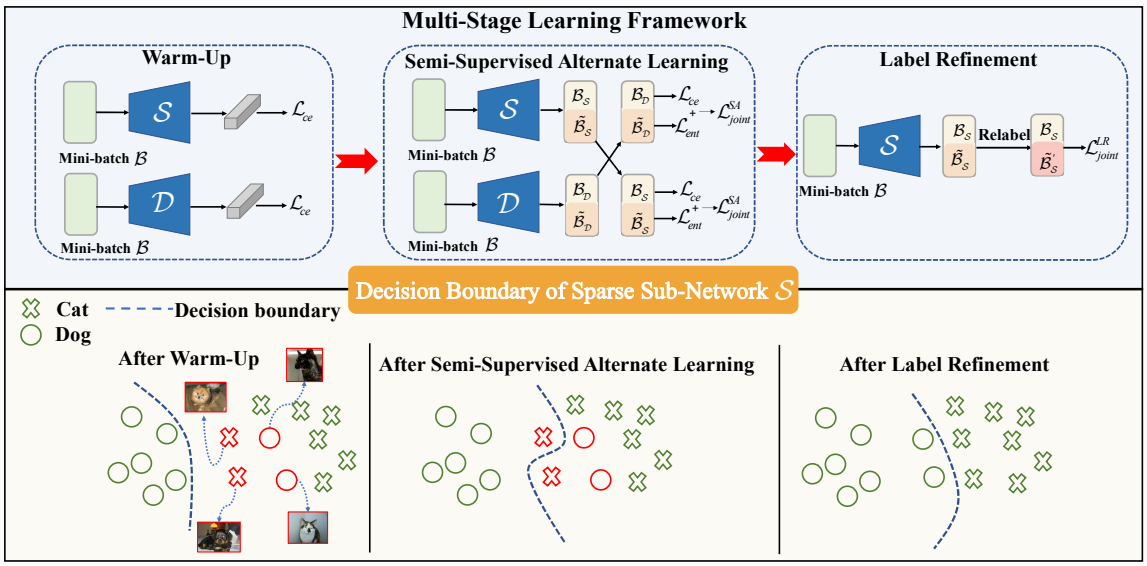

Fig. 2: Overview of the proposed multi-stage learning framework. In the upper panel, we show the training process of three stages. $\mathcal{S}$ and $\mathcal{D}$ denote the sparse sub-network and the dense sub-network, respectively. $\mathcal{B}_{\mathcal{S}}$ and $\mathcal{B}_{\mathcal{D}}$ represent the labeled samples selected by $\mathcal{S}$ and $\mathcal{D}$, respectively. $\tilde{\mathcal{B}}_{\mathcal{S}}$ and $\tilde{\mathcal{B}}_{\mathcal{D}}$ represent the unlabeled samples selected by $\mathcal{S}$ and $\mathcal{D}$, respectively. $\tilde{\mathcal{B}}'_{\mathcal{S}}$ denotes the samples which are relabeled by $\mathcal{S}$. In the lower panel, we show the decision boundary of $\mathcal{S}$ obtained from three stages. We take the two-class classification (cat and dog) as an example, where the crosses and circles in green represent samples with correct labels, and those in red represent samples with noisy labels. The classification ability of $\mathcal{S}$ is gradually enhanced, thereby refining the decision boundary after each stage.

where n denotes the number of samples in a batch; k indicates the sub-network ($\mathcal{S}$ or $\mathcal{D}$) to be trained; $\boldsymbol{y}_i$ is the ground-truth label of the sample $\boldsymbol{x}_i \in \mathcal{B}$; $\boldsymbol{f}_{\Theta_k}(\boldsymbol{x}_i)$ represents the output of one sub-network with the network parameters Θ_k given the input $\boldsymbol{x}_i$.

At this stage, the classification loss of easy and clean samples tends to decrease more than that of noisy labeled ones as iterations proceed [36], [37]. Therefore, both $\mathcal{S}$ and $\mathcal{D}$ can learn simple knowledge from easy and clean samples due to the memorization effects, and thus both sub-networks has the basic classification ability after this stage.

Semi-Supervised Alternate Learning Stage. Memorization effects show that the cross-entropy loss gradually memorizes all the samples, including clean and noisy labeled samples. This usually results in overfitting of the model to noisy labeled samples and seriously affects the generalization ability of the trained model. Hence, we adopt the warm-up stage as the initial training stage, and further develop a semi-supervised alternate learning stage.

During this stage, each sub-network views its small-loss samples and noisy labeled ones from a batch as labeled training samples and unlabeled ones, respectively, and then teaches its peer sub-network with these samples to update the network parameters in an alternate way. The two sub-networks capture the information from different views, alleviating the confirmation bias and benefiting sample selection. By taking advantage of semi-supervised alternate learning to train each sub-network, we are able to not only exploit the entire batch data for training, but also address the negative influence caused by bad "memorization" of noisy labels.

Semi-supervised learning is a powerful paradigm that leverages both labeled and unlabeled data for training. It often jointly optimizes two objective functions: a supervised loss over labeled data and an unsupervised loss over unlabeled data (or both labeled and unlabeled data). For the supervised loss, the classical cross-entropy can be used. For the unsupervised loss, to train the model with good generalization ability, a common assumption is that the decision boundary of a classifier should avoid intersecting high-density regions of the marginal data distribution [38]. To achieve this, one straightforward way is to encourage the classifier to give highly confident predictions (i.e., low-entropy) on unlabeled data. This can be explicitly done by minimizing the entropy loss of the model over unlabeled data [39].

More specifically, we first divide $\mathcal{B}$ into a clean sample set $\mathcal{B}_k$ ($k \in \{\mathcal{S}, \mathcal{D}\}$) and a noisy labeled sample set $\tilde{\mathcal{B}}_k$ ($k \in \{\mathcal{S}, \mathcal{D}\}$). In this paper, the small-loss criterion [19] is used to select clean samples (also called small-loss samples) from each sub-network. Mathematically, $\mathcal{B}_k$ is formulated as

$$\mathcal{B}_k = \arg\min_{\mathcal{B}':|\mathcal{B}'|\geq(1-\epsilon)|\mathcal{B}|} \mathcal{L}_{ce}(f_{\Theta_k}, \mathcal{B}'), k \in \{\mathcal{S}, \mathcal{D}\}, \quad (2)$$

where ϵ represents the noise rate in the training set; $|\mathcal{B}'|$ and $|\mathcal{B}|$ denote the sizes of $\mathcal{B}'$ and $\mathcal{B}$, respectively. Accordingly, $\tilde{\mathcal{B}}_k$ is defined by set subtraction as $\tilde{\mathcal{B}}_k = \mathcal{B} - \mathcal{B}_k$.

On the one hand, to overcome the problem of accumulated errors caused by the confirmation bias, we select the labeled samples provided by the peer sub-network, as done in [14], [19], [20]. Formally, $\mathcal{B}_k$ (viewed as labeled training samples) selected by each sub-network is employed to compute the classification loss of its peer sub-network. Thus, the classification

loss of each sub-network can be defined as

$$\mathcal{L}_{ce}(f_{\Theta_k}, \mathcal{B}_{k'}) = -\frac{1}{|\mathcal{B}_{k'}|}\sum_{i=1}^{|\mathcal{B}_{k'}|} \boldsymbol{y}_i^{l\mathrm{T}} \log(\boldsymbol{f}_{\Theta_k}(\boldsymbol{x}_i^l)), k \in \{\mathcal{S}, \mathcal{D}\}, \tag{3}$$

where k' indicates the peer sub-network of k, that is, $k' = \mathcal{D}$ (or $\mathcal{S}$), if $k = \mathcal{S}$ (or $\mathcal{D}$); $\mathcal{B}_{k'}$ denotes the clean sample set selected by k'; $|\mathcal{B}_{k'}|$ is the size of batch $\mathcal{B}_{k'}$; $\boldsymbol{y}_i^l$ indicates the ground-truth label of $\boldsymbol{x}_i^l \in \mathcal{B}_{k'}$.

On the other hand, $\tilde{\mathcal{B}}_k$ (viewed as unlabeled training samples) selected by each sub-network is used to calculate the entropy loss of its peer sub-network. The entropy loss is defined as

$$\mathcal{L}_{ent}(f_{\Theta_k}, \tilde{\mathcal{B}}_{k'}) = -\frac{1}{|\tilde{\mathcal{B}}_{k'}|}\sum_{i=1}^{|\tilde{\mathcal{B}}_{k'}|} \boldsymbol{f}_{\Theta_k}(\boldsymbol{x}_i^u)^{\mathrm{T}} \log(\boldsymbol{f}_{\Theta_k}(\boldsymbol{x}_i^u)), \quad k \in \{\mathcal{S}, \mathcal{D}\}, \tag{4}$$

where $|\tilde{\mathcal{B}}_{k'}|$ is the size of batch $\tilde{\mathcal{B}}_{k'}$ and $\boldsymbol{x}_i^u \in \tilde{\mathcal{B}}_{k'}$ denotes the i-th sample in $\tilde{\mathcal{B}}_{k'}$.

By minimizing the entropy loss, we are able to reduce the uncertainty of model prediction and force the decision boundary to pass through the sparse regions of the marginal data distribution. It is worth noting that the entropy loss only uses the outputs of the model (without relying on the label information), which can effectively alleviate the risk of overfitting to noisy labeled samples.

Therefore, for the batch $\mathcal{B}$, the joint loss to train each sub-network in this stage is formulated as follows:

$$\mathcal{L}_{joint}^{SA}(f_{\Theta_k}, \mathcal{B}) = \mathcal{L}_{ce}(f_{\Theta_k}, \mathcal{B}_{k'}) + \lambda \mathcal{L}_{ent}(f_{\Theta_k}, \tilde{\mathcal{B}}_{k'}), \quad k \in \{\mathcal{S}, \mathcal{D}\} \tag{5}$$

where $\lambda \in [0, 1]$ is a parameter to balance the two losses.

Based on the above joint loss, the semi-supervised alternate learning stage involves the following steps. First, we fix $\mathcal{D}$ and only update $\mathcal{S}$ according to $\mathcal{L}_{joint}^{SA}(f_{\Theta_\mathcal{S}}, \mathcal{B})$. Then, we fix $\mathcal{S}$ and only update $\mathcal{D}$ according to $\mathcal{L}_{joint}^{SA}(f_{\Theta_\mathcal{D}}, \mathcal{B})$. Next, $\mathcal{D}$ and $\mathcal{S}$ are alternately trained several times. During this stage, each sub-network treats all the batch samples as useful knowledge, and teaches its peer network with these samples in a semi-supervised learning manner. Consequently, the learning capability of $\mathcal{S}$ and $\mathcal{D}$ is further improved. In this way, we are able to relabel some noisy labeled samples with high confidence according to the output of $\mathcal{S}$, thereby facilitating the training in the next stage.

It is worth pointing that both TS^3-Net and DivideMix [32] leverage semi-supervised learning on the full training data. However, they are significantly different. First, TS^3-Net applies entropy minimization for unlabeled samples, while DivideMix adopts the MixMatch method for unlabeled samples. That is, TS^3-Net and DivideMix work on different semi-supervised learning techniques. Second, TS^3-Net performs alternate learning between a sparse sub-network and a dense sub-network, and thus offers two distinguished views for addressing noisy labels during the semi-supervised alternate learning stage. On the contrary, DivideMix trains two sub-networks having the same network architectures by a joint learning framework. Third, TS^3-Net involves a multi-stage learning framework. In particular, the semi-supervised alternate learning stage makes the model more confident for unlabeled samples. Such a manner greatly benefits the training of the subsequent label refinement stage. In contrast, DivideMix directly assigns soft pseudo-labels to noisy labeled samples by using model predictions during the whole training process.

Label Refinement Stage. During the semi-supervised alternate learning stage, both dense and sparse sub-networks learn and communicate the knowledge from each other from a different perspective, enhancing the robustness of both sub-networks against label noise. As a result, the sparse sub-network $\mathcal{S}$ is able to predict more reliable results. However, the noisy labeled samples are treated as unlabeled samples in this stage, which may still limit the classification ability of $\mathcal{S}$. Therefore, we adopt a label refinement stage, where we fix $\mathcal{D}$ and only update $\mathcal{S}$. Especially, we select noisy labeled samples given by $\mathcal{S}$ and relabel some of them according to prediction confidence. Then, the clean and relabeled samples are jointly used to train $\mathcal{S}$, further improving its classification ability.

More specifically, the batch $\mathcal{B}$ is first divided into $\mathcal{B}_\mathcal{S}$ and $\tilde{\mathcal{B}}_\mathcal{S}$ according to the small-loss criterion by $\mathcal{S}$ (as defined in Eq. (2)). To fully leverage the information in unlabeled samples, the output predicted by $\mathcal{S}$ is used to update the labels of training samples in $\tilde{\mathcal{B}}_S$ by

$$\tilde{\boldsymbol{y}}_i^r(j) = \Pi[j = \arg\max_c \boldsymbol{f}_{\Theta_\mathcal{S}}^c(\boldsymbol{x}_i^r)] \quad \text{if} \quad \max_c \boldsymbol{f}_{\Theta_\mathcal{S}}^c(\boldsymbol{x}_i^r) \geq \tau, \tag{6}$$

where $\tilde{\boldsymbol{y}}_i^r(j)$ denotes the j-th element in $\tilde{\boldsymbol{y}}_i^r$ that represents the pseudo-label vector of $\boldsymbol{x}_i^r$ after relabeling and $\boldsymbol{f}_{\Theta_\mathcal{S}}^c(\boldsymbol{x}_i^r)$ denotes the c-th element in $\boldsymbol{f}_{\Theta_\mathcal{S}}(\boldsymbol{x}_i^r)$; the indicator function $\Pi\{\cdot\}$ takes on the value 1 if its argument is true, and 0 otherwise; the threshold τ is a hyperparameter to determine whether a sample should be relabeled. By relabeling, more samples with less label noise can be obtained. Therefore, the labels of the whole training set are refined gradually, thus enhancing the classification capability of $\mathcal{S}$.

Finally, we employ both clean samples and relabeled ones to calculate the classification loss. For convenience, we use $\tilde{\mathcal{B}}'_\mathcal{S}$ to represent the relabeled sample set, where the maximum predicted output of each sample is greater than τ. The joint loss in this stage is formulated as

$$\mathcal{L}_{joint}^{LR}(f_{\Theta_\mathcal{S}}, \mathcal{B}) = \mathcal{L}_{ce}(f_{\Theta_\mathcal{S}}, \mathcal{B}_\mathcal{S}) + \gamma \mathcal{L}_{ce}(f_{\Theta_\mathcal{S}}, \tilde{\mathcal{B}}'_\mathcal{S}), \tag{7}$$

where $\gamma \in [0, 1]$ is a parameter to balance the two losses.

Assigning pseudo-labels to unlabeled samples is a widely-used technique in semi-supervised/unsupervised learning methods (such as NoisyStudent [40], LUDA [41], PL [42], and FixMatch [43]). However, these methods are intrinsically different from ours. First, NoisyStudent and LUDA directly assign pseudo-labels to unlabeled samples via model predictions without considering the sample confidence. In contrast, FixMatch and our method refine the labels only for the samples with high confidence by using a threshold. Moreover, NoisyStudent, LUDA, PL, and FixMatch do not work on label noise learning, while our method is specifically designed

Algorithm 1: Multi-Stage Learning Framework

Input: A training dataset $\mathcal{T}$; a sparse sub-network $\mathcal{S}$; a dense sub-network $\mathcal{D}$; total training epochs E; training epochs for the first stage E_1 and the second stage E_2.

for $loop \leftarrow 1$ *to* E **do**
  Shuffle $\mathcal{T}$ into M mini-batches;
  if $loop \leq E_1$ **then**
    //warm-up stage
    for $b \leftarrow 1$ *to* M **do**
      Fetch the b-th mini-batch $\mathcal{B}$ from $\mathcal{T}$;
      Calculate $\mathcal{L}_{ce}(f_{\Theta_\mathcal{S}}, \mathcal{B})$ and $\mathcal{L}_{ce}(f_{\Theta_\mathcal{D}}, \mathcal{B})$ according to Eq. (1);
      Update $\Theta_\mathcal{S}$ and $\Theta_\mathcal{D}$ by Stochastic Gradient Descent (SGD);
    end
  end
  else if $E_1 + 1 \leq loop \leq E_1 + E_2$ **then**
    //semi-supervised alternate learning stage
    for $b \leftarrow 1$ *to* M **do**
      Fetch the b-th mini-batch $\mathcal{B}$ from $\mathcal{T}$;
      Fix $\mathcal{D}$ and **Calculate** $\mathcal{L}^{SA}_{joint}(f_{\Theta_\mathcal{S}}, \mathcal{B})$ according to Eq. (5);
      Update $\Theta_\mathcal{S}$ by SGD;
    end
    for $b \leftarrow 1$ *to* M **do**
      Fetch the b-th mini-batch $\mathcal{B}$ from $\mathcal{T}$;
      Fix $\mathcal{S}$ and **Calculate** $\mathcal{L}^{SA}_{joint}(f_{\Theta_\mathcal{D}}, \mathcal{B})$ according to Eq. (5);
      Update $\Theta_\mathcal{D}$ by SGD;
    end
  end
  else
    //label refinement stage
    for $b \leftarrow 1$ *to* M **do**
      Fetch the b-th mini-batch $\mathcal{B}$ from $\mathcal{T}$;
      Get $\mathcal{B}_\mathcal{S}$ according to Eq. (2);
      Get $\tilde{\mathcal{B}}'_\mathcal{S}$ according to Eq. (6);
      Calculate $\mathcal{L}^{LR}_{joint}(f_{\Theta_\mathcal{S}}, \mathcal{B})$ according to Eq. (7);
      Update $\Theta_\mathcal{S}$ by SGD;
    end
  end
end

Output: A sparse sub-network $\mathcal{S}$ and a dense sub-network $\mathcal{D}$.

for addressing label noise. Finally, NoisyStudent, LUDA, and PL rely on the model predictions by self-training. On the contrary, in our method, by minimizing the entropy in the semi-supervised alternate learning stage, the model prediction is prone to give a peaked probability distribution (the peak corresponds to a certain class with a high probability). This can improve the confidence about unlabeled samples, helpful to the training in the label refinement stage.

The training process of our multi-stage learning framework is given in Algorithm 1. Although the warm-up stage and the label refinement stage share some similarities to existing methods [35], [40]–[42], we would like to highlight that the three stages are tightly combined and jointly optimized in our integrated network. The warm-up stage enables the sparse and dense sub-networks to learn the basic classification ability. The semi-supervised alternate learning stage further enhances the learning capacity of both sub-networks, which is advantageous to label refinement in the third stage. Hence, a sparse DNN model can be effectively trained in a progressive manner.

Algorithm 2: Overall Training of TS^3-Net

Input: A pruning factor p_f; the number of pruning times R.

Randomly initialize the two sub-networks $\mathcal{S}$ and $\mathcal{D}$;
for $r \leftarrow 1$ *to* R **do**
  Train $\mathcal{S}$ and $\mathcal{D}$ by Algorithm 1;
  Prune the p_f of parameters of $\mathcal{S}$ with a mask according to smallest-magnitude weights;
  Reset the weights of the remaining portions of $\mathcal{S}$ and $\mathcal{D}$ to their initial values;
end
Retrain $\mathcal{S}$ and $\mathcal{D}$ by Algorithm 1 to obtain the final sparse model;

Output: A sparse sub-network $\mathcal{S}$.

C. Overall Training

The overall training of TS^3-Net is summarized in Algorithm 2. First, $\mathcal{S}$ and $\mathcal{D}$, which have the same network architectures, are initialized with different network parameters. Second, $\mathcal{S}$ and $\mathcal{D}$ are trained based on the proposed multi-stage learning framework. Third, $\mathcal{S}$ is pruned according to its smallest-magnitude weights. Fourth, the remaining network parameters of $\mathcal{S}$ and $\mathcal{D}$ are reset to their initial values. The above second to fourth steps are iteratively performed for several rounds to identify the winning ticket. Finally, the winning ticket is retrained to obtain a robust sparse sub-network.

D. Differences from Conventional LNL Methods with Two Sub-Networks

Conventional LNL methods (such as Co-teaching [19], Co-teaching+ [20], JoCoR [14], and DivideMix [32]) also train two sub-networks. However, the differences between these methods and our proposed method are significant.

First, Co-teaching, Co-teaching+, JoCoR, and DivideMix do not work on model compression and acceleration. Unlike the above methods, our method is able to obtain a very lightweight network (i.e., the winning ticket) from noisy labeled training data. Second, Co-teaching, Co-teaching+, and JoCoR do not leverage the noisy labeled sample set during training. Although the labels of noisy labeled samples are noisy, these samples involve useful information for training. Therefore, different from these methods, DivideMix and our method view the samples in the noisy labeled sample set as unlabeled ones and train each sub-network in a semi-supervised learning setting. Third, the two sub-networks used in Co-teaching, Co-teaching+, JoCoR, and DivideMix have the same network architectures during training. In other words, they have similar learning capability. Hence, the two sub-networks are often simultaneously and jointly learned to improve both performances. However, in our setting of combining network pruning and label noise learning, the learning capability of $\mathcal{S}$ is generally weaker than that of $\mathcal{D}$ due to the network pruning. The joint learning of $\mathcal{S}$ and $\mathcal{D}$ may increase the training difficulty. To alleviate this problem, we leverage alternate learning between $\mathcal{S}$ and $\mathcal{D}$ in the second stage. The weak learner (i.e., $\mathcal{S}$) learns from the strong learner (i.e, $\mathcal{D}$) first and then the strong learner learns from the weak learner in turn. Thus, the performances of $\mathcal{S}$ and $\mathcal{D}$ can be gradually improved.

IV. Experiments

In this section, we perform extensive experiments to evaluate the superiority of our proposed method on both synthetic and real-world noisy datasets. First, we introduce several popular datasets and implementation details in Sec. IV-A. Then, we compare our proposed method with several state-of-the-art LNL methods in Sec. IV-B. Finally, we give further evaluation and discussions in Sec. IV-C.

A. Datasets and Implementation Details

1) Datasets: To validate the effectiveness of our method, we perform the image classification task on six benchmark datasets: MNIST [44], CIFAR-10 [45], CIFAR-100 [45], ANIMAL-10N [46], Clothing1M [47], and WebVision [48].

MNIST is a handwritten digit dataset which contains 10 classes with 50,000 training images and 10,000 test images. The size of each image in MNIST is 28×28. CIFAR-10 and CIFAR-100 are comprised of RGB images with the size of 32×32, corresponding to 10 classes and 100 classes, respectively. Both of them contain $50,000$ images for training and $10,000$ images for testing. ANIMAL-10N is a real-world noisy dataset consisting of human-labeled online images for 10 confusing animals, where the noisy labels are naturally generated by human labeling errors. It provides $50,000$ images for training and $5,000$ images for testing, where the size of each image is 64×64. Clothing1M and WebVision are two large-scale real-world noisy datasets. Clothing1M consists of around 1 million training data and about 10,000 clean test data, which are collected from online shopping websites. WebVision includes about 2.4 million training samples obtained from the web using 1,000 concepts in ImageNet ILSVRC12. Following [32], we adopt the first 50 classes of the Google image subset for training and testing.

Our goal is to identify the winning ticket under label noise. Since MNIST, CIFAR-10, and CIFAR-100 are clean, we add the synthetic label noise to the dataset according to a transition matrix $T = \{T_{ij}\}$, where T_{ij} denotes the probability of flipping a ground-truth label i to a corrupted label j. Following the settings in Co-teaching [19], we use two representative noise structures. (1) Symmetric noise, where each sample is independently assigned to a uniformly random label instead of its true label in the training set. In this case, $T_{ij} = \frac{\epsilon}{C-1}$ for $j \neq i$, where C is the number of classes and ϵ denotes the noise rate. (2) Asymmetric noise, where samples in one class are assigned to the label of a similar class in the training set. In this case, $T_{ij} = \epsilon$ for a particular class j ($j \neq i$) and $T_{ij} = 0$ otherwise. To evaluate our method, we select the noise rate $\epsilon \in \{20\%, 40\%\}$ for symmetric noise and $\epsilon = 20\%$ for asymmetric noise. For ANIMAL-10N, Clothing1M, and WebVision, we set $\epsilon = 8\%$, 38.5%, and 20.0%, respectively, since the noise rates in these datasets are around 8%, 38.5%, and 20.0%, as suggested by [16].

2) Implementation Details: We take $\mathcal{S}$ and $\mathcal{D}$ with the same architectures but different initializations at the beginning of the training process. Note that the scales of different datasets vary. Following [18], [19], we use simple backbones for small-scale datasets (such as MNIST and CIFAR-10), while we adopt more sophisticated backbones for large-scale datasets (such as Clothing1M and WebVision). Concretely, we use a three-layer Multi-Layer Perception (MLP) [12] for MNIST. We use an eight-layer Convolutional Neural Network (CNN) [18] for CIFAR-10. We adopt VGG-16 [22] for CIFAR-100. We employ VGG-19 [22] for ANIMAL-10N. Besides, we adopt ResNet-18 [21] for Clothing1M. Following [32], Inception-ResNet-v2 [49] is used for WebVision.

We implement all the methods via PyTorch and conduct all the experiments on NVIDIA RTX 3080 GPU. We use the SGD optimizer with momentum 0.9 for all the datasets. The total number of training epochs E is set to 120 for CIFAR-10, CIFAR-100, ANIMAL-10N, 40 for MNIST, and 100 for Clothing1M and WebVision. The batch size is set to 256 for all the datasets except for Clothing1M and WebVision. For Clothing1M and WebVision, the batch size is set to 32. For all the datasets except for MNIST, the numbers of training epochs E_1 and E_2 for the warm-up stage and the semi-supervised alternate learning stage are set to 20 and 60, respectively. For MNIST, E_1 and E_2 are set to 10 and 20, respectively.

For MNIST, we use a constant learning rate (e.g., 1.0). For CIFAR-10, CIFAR-100, and ANIMAL-10N, the initial learning rate is set to 0.10 and decayed to 0.01 after 40 epochs. For Clothing1M and WebVision, the initial learning rate is set to 0.02 and decayed to 0.002 after 60 epochs. The parameter λ in Eq. (5), the threshold τ in Eq. (6) and the parameter γ in Eq. (7) are empirically set to 0.50, 0.70 and 0.50, respectively, for all the datasets. For network pruning, we use IMP to prune the sparse sub-network with a pruning factor $p_f \in \{0.3, 0.4, 0.5, 0.6\}$. The number of pruning times R is set to 1, 2, or 3. The sparsity level of the network is computed as $1 - (1 - p_f)^R$. Therefore, 12 levels of sparsity (i.e., 30%, 40%, 50%, 51%, 60%, 64%, 65.7%, 75%, 78.4%, 84%, 87.5%, and 93.6%) are obtained.

To measure the performance, we use the test accuracy as well as the number of parameters (Params) to evaluate the accuracy and memory consumption, respectively.

B. Comparisons with State-of-the-Art LNL Methods

In this subsection, we compare our proposed method with several state-of-the-art LNL methods on both synthetic and real-world noisy datasets.

Competing Methods. We compare our proposed TS^3-Net method with several state-of-the-art LNL methods (including SCE [18], MAE [50], APL [29], ITLM [51], Co-teaching [19], JoCoR [14], SELFIE [46], DivideMix [32], MentorNet [30], F-correction [52], D2L [53], Decoupling [54], and a baseline method) under different types of label noise. For MentorNet, F-correction, D2L, and Decoupling, we directly cite the results reported in [55]. For SCE, MAE, APL, ITLM, Co-teaching, JoCoR and DivideMix, we report their results by running the source codes provided by their respective authors to train the models. We report the results obtained by APL with a combination of NCE and MAE, and those obtained by DivideMix without model ensemble. When evaluating on a dataset, we use the same backbones for all the competing methods. The baseline method refers to the method that trains

TABLE I: Test Accuracy (%) and the number of parameters obtained by different methods on MNIST.

Method	Symmetric-20%	Symmetric-40%	Asymmetric-20%	Params (M)
Baseline	90.16	84.82	92.21	≈ 0.28
SCE (2019)	97.54	96.77	97.05	≈ 0.28
APL (2020)	96.88	95.76	96.90	≈ 0.28
MAE (2016)	91.95	88.93	92.48	≈ 0.28
ITLM (2019)	95.60	94.91	95.68	≈ 0.28
Co-teaching (2018)	96.37	95.49	97.00	≈ 0.28
JoCoR (2020)	**98.06**	96.81	96.99	≈ 0.28
DivideMix (2020)	96.21	95.37	96.07	≈ 0.28
TS^3-Net (dense)	97.84	97.02	**97.54**	≈ 0.28
TS^3-Net	97.30	**97.11**	97.28	≈ 0.04

TABLE II: Test Accuracy (%) and the number of parameters obtained by different methods on CIFAR-10.

Method	Symmetric-20%	Symmetric-40%	Asymmetric-20%	Params (M)
Baseline	83.58	78.67	84.32	≈ 0.52
SCE (2019)	87.17	84.31	85.05	≈ 0.52
APL (2020)	86.77	84.13	85.90	≈ 0.52
MAE (2016)	86.13	82.21	84.77	≈ 0.52
ITLM (2019)	85.58	84.49	87.12	≈ 0.52
Co-teaching (2018)	86.85	84.95	86.35	≈ 0.52
JoCoR (2020)	85.73	82.21	82.11	≈ 0.52
DivideMix (2020)	88.41	86.11	88.66	≈ 0.52
TS^3-Net (dense)	**88.52**	85.52	**88.79**	≈ 0.52
TS^3-Net	88.28	**86.54**	88.67	≈ 0.07

TABLE III: Test Accuracy (%) and the number of parameters obtained by different methods on CIFAR-100.

Method	Symmetric-20%	Symmetric-40%	Asymmetric-20%	Params (M)
Baseline	56.76	43.13	55.18	≈ 138.36
SCE (2019)	57.25	43.88	57.50	≈ 138.36
APL (2020)	53.47	44.62	56.67	≈ 138.36
ITLM (2019)	59.27	57.37	61.99	≈ 138.36
Co-teaching (2018)	63.17	59.24	63.22	≈ 138.36
DivideMix (2020)	65.28	60.91	65.37	≈ 138.36
TS^3-Net (dense)	65.60	**61.09**	64.02	≈ 138.36
TS^3-Net	**66.87**	60.99	**65.96**	≈ 17.30

the neural network by using the standard cross-entropy loss. Moreover, we also evaluate the performance of the trained dense sub-network $\mathcal{D}$ (denoted as TS^3-Net (dense)) in our method. In this subsection, we set $p_f = 0.5$ and $R = 3$ (the sparsity level is 87.5%) for MNIST, CIFAR-10, CIFAR-100, and ANIMAL-10N, while we set $p_f = 0.4$ and $R = 3$ (the sparsity level is 78.4%) for Clothing1M and WebVision.

Results on MNIST We report the test accuracy obtained by all the competing methods on MNIST, as shown in Table I. We can see that all the LNL methods perform better than the baseline method. This shows the effectiveness of label noise learning. For the Symmetric-20% case, SCE, JoCoR, TS^3-Net (dense) and TS^3-Net obtain better test accuracy than the other methods. SCE leverages a robust term to learn with label noise, while JoCoR selects the small-loss samples with a joint loss to update the two networks. TS^3-Net takes advantage of a multi-stage learning framework to update $\mathcal{S}$ and $\mathcal{D}$. In this case, although JoCoR performs slightly better than TS^3-Net, the network parameters of TS^3-Net (about 0.04M) are much fewer than those of JoCoR (about 0.28M). For the Asymmetric-20% and Symmetric-40% cases, our proposed TS^3-Net (dense) and TS^3-Net obtain better performance than the other competing methods. Compared with TS^3-Net (dense), TS^3-Net achieves similar performance with greatly fewer network parameters.

Results on CIFAR-10 Table II shows the test accuracy on CIFAR-10. We use an eight-layer CNN model which contains about 0.52M parameters as the backbone. For the Symmetric-20% case, TS^3-Net (dense) performs better than the other competing methods. For the Symmetric-40% case, DivideMix performs better than TS^3-Net (dense) while TS^3-Net gives the best performance among all the competing methods. The sparse model is beneficial to alleviate the overfitting problem due to label noise by removing redundant parameters, leading to better generalization capability. Note that SCE, APL, and MAE cannot effectively deal with severe label noise and they achieve low test accuracy for the Symmetric-40% case. JoCoR, Co-teaching, and ITLM are representative sample selection based methods, which only use the small-loss samples to update model parameters. However, such a way may limit the generalization capability of these methods. Hence, their performance is inferior to that of TS^3-Net.

Results on CIFAR-100 The comparison results on CIFAR-100 are shown in Table III. We use VGG-16 as the backbone which contains about 138.86M parameters. As shown in Table III, the performance of APL declines, which may be mainly due to the bad memorization of noisy samples. The

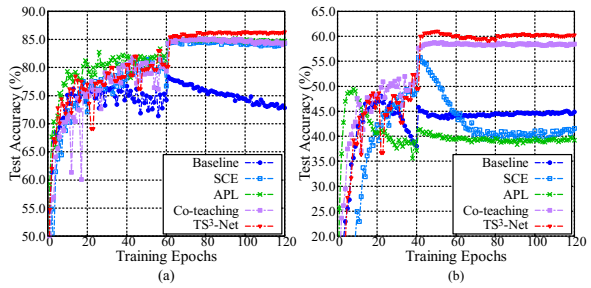

Fig. 3: Test accuracy vs. training epochs under Symmetric-40% label noise on (a) CIFAR-10 and (b) CIFAR-100.

TABLE IV: Test Accuracy (%) and the number of parameters obtained by different methods on ANIMAL-10N.

Method	Accuracy	Params (M)
Baseline	78.20	$\approx$ 143.67
SCE (2019)	76.82	$\approx$ 143.67
APL (2020)	76.94	$\approx$ 143.67
ITLM (2019)	78.80	$\approx$ 143.67
Co-teaching (2018)	79.10	$\approx$ 143.67
DivideMix (2020)	79.98	$\approx$ 143.67
SELFIE (2019)	**81.80**	$\approx$ 143.67
TS^3-Net (dense)	80.38	$\approx$ 143.67
TS^3-Net	81.36	$\approx$ 17.96

TABLE V: Test Accuracy (%) and the number of parameters obtained by different methods on Clothing1M. "Best" and "Last" represent the trained model at the epoch (when the validation accuracy is optimal) and at the end of training epochs, respectively.

Method	Best	Last	Params (M)
Baseline	69.20	66.49	$\approx$ 11.69
DivideMix (2020)	71.88	71.61	$\approx$ 11.69
JoCoR (2020)	70.30	69.79	$\approx$ 11.69
Co-teaching (2018)	69.21	68.51	$\approx$ 11.69
TS^3-Net (dense)	71.15	71.07	$\approx$ 11.69
TS^3-Net	**72.20**	**72.09**	$\approx$ 2.53

performance of SCE is affected when the number of classes is large, as validated in [16]. For the hard Symmetric-40% case, ITLM performs poorly since it may not effectively select clean samples and thus overfits some noisy labeled samples during training. For all the types of label noise, TS^3-Net outperforms Co-teaching and ITLM by a moderate margin. Compared with TS^3-Net (dense), TS^3-Net gives similar/better results. This demonstrates the effectiveness of our method.

We also plot the curves of test accuracy vs. the number of training epochs obtained by our method and several competing methods in Fig. 3. Our method outperforms the other competing methods, which shows the superiority of our method.

Results on ANIMAL-10N In this section, we compare our proposed TS^3-Net with several methods on a real-world noisy dataset ANIMAL-10N, as shown in Table IV. All the competing methods are based on VGG-19, which contains about 143.67M parameters. The baseline method outperforms SCE and APL on this dataset because the noise rate is small (about 8%). SELFIE obtains the best test accuracy (81.80%) among all the methods. However, it is based on a dense network and

TABLE VI: The top-1, top-5 accuracy (%), and the number of parameters obtained by different methods on WebVision (mini). The accuracy is reported on both the WebVision validation set and the ImageNet ILSVRC12 validation set.

Test dataset	WebVision		ILSVRC12		-
Accuracy (%)	top-1	top-5	top-1	top-5	Params (M)
F-correction (2017)	61.12	82.68	57.36	82.36	54.38
Decoupling (2017)	62.54	84.74	58.26	82.26	54.38
D2L (2018)	62.68	84.00	57.80	81.36	54.38
MentorNet (2018)	63.00	81.40	57.80	79.92	54.38
Co-teaching (2018)	63.58	85.20	61.48	84.70	54.38
Iterative-CV (2019)	65.24	85.34	61.60	84.98	54.38
DivideMix (2020)	**75.70**	90.01	**73.18**	89.11	54.38
TS^3-Net (dense)	74.60	90.68	71.08	90.31	54.38
TS^3-Net	73.48	**90.92**	70.27	**90.46**	11.75

TABLE VII: The details of seven variants in TS^3-Net

Method	warm-up	alternate	refine	entropy
TS^3-Net (warm)	✓			
TS^3-Net (semi)		✓		✓
TS^3-Net (warm+semi)	✓	✓		✓
TS^3-Net (warm+refine)	✓		✓	
TS^3-Net (semi+refine)		✓	✓	✓
TS^3-Net (warm+alter)	✓	✓		
TS^3-Net	✓	✓	✓	✓

"warm-up", "alternate", and "refine" represent the warm-up stage, the semi-supervised alternate learning stage without using the entropy loss and the label refinement stage. "entropy" represents that the entropy loss is used.

uses multiple restart operations, which are time-consuming. Compared with SELFIE, TS^3-Net achieves comparable performance (81.36%) with only about 1/8 of network parameters. TS^3-Net outperforms TS^3-Net (dense) by about 1%, which indicates that the winning ticket identified by TS^3-Net has good generalization capability on the real-world noisy dataset.

Results on Clothing1M We compare our TS^3-Net with several state-of-the-art LNL methods on Clothing1M, as shown in Table V. All the competing methods are based on ResNet-18 which contains about 11.69M parameters. We can observe that TS^3-Net consistently outperforms the other competing methods. Compared with TS^3-Net (dense), TS^3-Net achieves higher accuracy with around only 1/5 of network parameters. This further indicates the effectiveness of TS^3-Net on the large-scale real-world dataset with a high noise rate (38.5%).

Results on WebVision We show the results on WebVision in Table VI. For the top-1 accuracy, TS^3-Net consistently outperforms other methods except for DivideMix. But TS^3-Net requires only about 1/5 of network parameters compared with DivideMix. Moreover, TS^3-Net obtains the best top-5 accuracy. This is because an effective multi-stage learning framework is designed by explicitly considering the differences between the sparse sub-network and the dense sub-network.

C. Further Evaluation and Discussions

1) Ablation Studies: In TS^3-Net, a multi-stage learning framework, consisting of a warm-up stage, a semi-supervised alternate learning stage, and a label refinement stage, is developed to train $\mathcal{S}$ and $\mathcal{D}$. To evaluate the effectiveness of each stage, we conduct ablation studies on MNIST under

TABLE VIII: Test accuracy (%) obtained by different variants at different levels of sparsity on MNIST and CIFAR-100.

Dataset	Method	Sparsity (%)												
		0	30	40	50	51	60	64	65.7	75	78.4	84	87.5	93.6
MNIST	TS^3-Net (warm)	84.82	77.07	80.15	81.57	76.67	86.19	87.65	82.10	86.92	86.44	86.36	92.29	92.09
	TS^3-Net (semi)	95.97	96.10	95.58	96.23	95.99	95.86	95.90	96.29	95.61	95.60	95.93	95.81	96.20
	TS^3-Net (warm+semi)	96.77	96.37	96.10	96.42	96.61	96.33	96.18	96.41	96.47	96.01	96.38	96.87	96.55
	TS^3-Net (warm+refine)	96.60	96.72	96.53	96.51	96.82	96.58	96.50	96.83	96.48	96.64	96.49	96.27	96.25
	TS^3-Net (semi+refine)	96.56	96.53	96.48	96.41	96.39	96.38	96.43	96.38	96.67	96.41	96.58	96.58	96.38
	TS^3-Net (warm+alter)	96.20	96.23	96.19	96.48	95.59	96.36	95.99	96.45	95.63	96.07	96.18	96.97	96.31
	TS^3-Net	**97.02**	**96.76**	**97.20**	**96.84**	**97.01**	**97.08**	**97.00**	**96.95**	**97.14**	**97.13**	**97.01**	**97.11**	**96.77**
CIFAR-100	TS^3-Net (warm)	56.76	57.07	56.88	56.46	56.21	59.98	57.04	56.91	56.54	56.13	56.31	56.32	54.18
	TS^3-Net (semi)	61.31	62.13	61.37	61.34	63.09	62.03	63.01	64.07	62.17	63.50	63.88	61.92	62.70
	TS^3-Net (warm+semi)	64.47	64.85	64.34	64.66	64.73	64.77	64.90	65.39	65.57	64.85	64.90	65.87	65.00
	TS^3-Net (warm+refine)	63.62	63.41	63.71	62.90	63.33	63.48	64.00	64.47	64.25	64.11	63.99	65.01	65.00
	TS^3-Net (semi+refine)	61.98	59.90	61.01	62.17	61.29	62.84	63.67	64.43	62.86	64.25	64.06	61.91	63.69
	TS^3-Net (warm+alter)	63.69	63.97	64.12	63.77	63.64	63.56	63.89	63.78	64.47	64.16	64.90	63.77	64.09
	TS^3-Net	**65.60**	**65.78**	**65.98**	**65.10**	**65.34**	**65.41**	**65.49**	**66.25**	**66.56**	**65.99**	**66.17**	**66.87**	**66.78**

TABLE IX: Accuracy (%) for the different values of λ, τ, γ, E_1, E_2 under C_1: CIFAR-10 under Symmetric-40% label noise and C_2: CIFAR-100 under Symmetric-40% label noise.

(a) Influence of λ.

λ	C_1	C_2
0.00	84.96	58.67
0.50	**86.54**	**60.99**
0.70	86.21	60.88
1.00	85.42	59.94

(b) Influence of τ.

τ	C_1	C_2
0.00	85.69	60.27
0.30	86.33	60.65
0.50	86.38	60.71
0.70	**86.54**	**60.99**

(c) Influence of γ.

γ	C_1	C_2
0.00	85.45	60.18
0.10	86.44	60.91
0.50	**86.54**	**60.99**
1.00	85.79	60.78

(d) Influence of E_1.

E_1	C_1	C_2
10	86.11	60.87
15	86.32	60.63
20	**86.54**	**60.99**
30	85.68	60.27

(e) Influence of E_2.

E_2	C_1	C_2
30	85.78	60.38
45	86.49	61.01
60	**86.54**	60.99
80	86.32	**61.35**

Symmetric-40% label noise and CIFAR-100 under Symmetric-20% label noise.

Specifically, we evaluate several variants of TS^3-Net, including TS^3-Net based on only the warm-up stage (denoted as TS^3-Net (warm)), TS^3-Net based on only the semi-supervised alternate learning stage (denoted as TS^3-Net (semi)), TS^3-Net based on the warm-up and semi-supervised alternate learning stages (denoted as TS^3-Net (warm+semi)), TS^3-Net based on the warm-up and label refinement stages (denoted as TS^3-Net (warm+refine)), TS^3-Net based on the semi-supervised alternate learning and label refinement stages (denoted as TS^3-Net (semi+refine)), and TS^3-Net based on the warm-up and alternate learning stages (denoted as TS^3-Net (warm+alter), where the entropy loss is not employed). The details of these variants are shown in Table VII. Moreover, TS^3-Net based on the three stages is also evaluated. In all ablation studies, we select 13 levels of sparsity (including 0.00% sparsity, where the pruning factor is set to 0.0) for evaluation. The comparison results are given in Table VIII.

From Table VIII, we can observe the following patterns. First, among all the variants, TS^3-Net (warm) achieves the worst performance at different levels of sparsity on two datasets. This is due to the fact that TS^3-Net (warm) gradually overfits noisy labels during the training. Meanwhile, compared with TS^3-Net (warm+semi), TS^3-Net (semi) gives worse performance. TS^3-Net (semi+refine), which does not employ the warm-up stage, achieves lower accuracy than TS^3-Net. These results verify the necessity of the warm-up stage. Second, TS^3-Net (warm+semi) achieves better performance than TS^3-Net (warm) and TS^3-Net gives higher test accuracy than TS^3-Net (warm+refine). The above results show the importance of the semi-supervised alternate learning stage. Third, TS^3-Net (warm+semi) obtains higher accuracy than TS^3-Net (warm+alter). Therefore, the semi-supervised learning manner is helpful to improve the performance by exploiting all the batch samples during training. Finally, TS^3-Net gives better results than TS^3-Net (warm+semi). This clearly shows the effectiveness of the label refinement stage. In summary, from the ablation studies on two datasets, the effectiveness of each stage in the multi-stage learning framework is validated.

2) Parameter Sensitivity Analysis: In this subsection, we illustrate the influence of several key parameters (including the parameters λ, τ, and γ in Eq. (5), Eq. (6), and Eq. (7), respectively; the training epochs E_1 and E_2 for the first stage and the second stage, respectively; the noise rate ϵ) of TS^3-Net on the final performance. We set $p_f = 0.5$ and $R = 3$. Thus, the sparsity level is 87.5%. CIFAR-10 under Symmetric-40% label noise and CIFAR-100 under Symmetric-40% label noise are used. The results are given in Table IX.

Influence of λ, τ, and γ. We first fix $\tau = 0.70$ and $\gamma = 0.50$, and vary the value of λ from 0.00 to 1.00. The results are listed in Table IX(a). From Table IX(a), we can see that, when λ is set to 0.00, which indicates that TS^3-Net is trained without using the unlabeled samples in the semi-supervised alternate learning stage, TS^3-Net gives the worst performance. TS^3-Net gives the top accuracy when $\lambda = 0.50$. Next, we fix $\lambda = 0.50$ and $\gamma = 0.50$ and vary the value of τ from 0.00 to 0.70. As shown in Table IX(b), TS^3-Net achieves the best performance when $\tau = 0.70$ and achieves the worst performance when $\tau = 0.00$. Then, we fix $\lambda = 0.50$ and $\tau = 0.70$ to investigate the influence of γ. From Table IX(c), we observe that the performance gap between $\gamma = 0.10$ and $\gamma = 0.50$ is marginal and TS^3-Net obtains the worst performance when $\gamma = 0.00$ (indicating that all unlabeled samples are used in the label

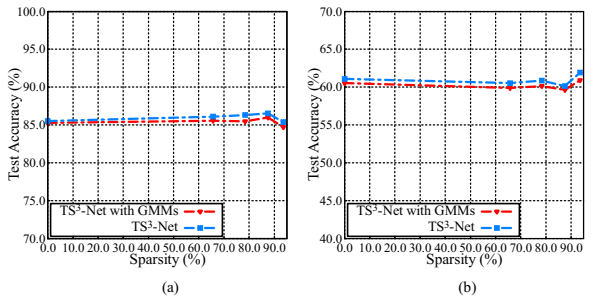

Fig. 4: Test accuracy vs. sparsity levels under Symmetric-40% label noise on (a) CIFAR-10 and (b) CIFAR-100.

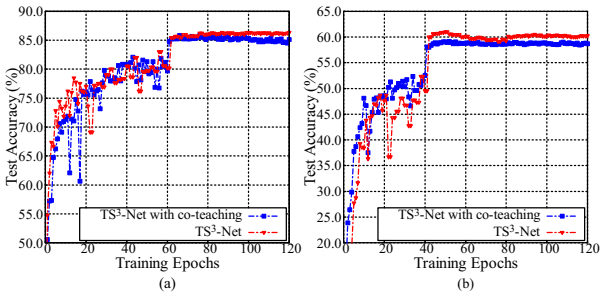

Fig. 5: Test accuracy vs. training epochs under Symmetric-40% label noise on (a) CIFAR-10 and (b) CIFAR-100.

refinement stage). For all our experiments, we fix $\lambda = 0.50$, $\tau = 0.70$, and $\gamma = 0.50$, respectively.

Influence of E_1 and E_2. The influences of E_1 and E_2 are shown in Table IX(d) and Table IX(e), respectively. Specifically, we first fix $E_2 = 60$ and vary the value of E_1 from 10 to 30. From Table IX(d), we can observe that TS^3-Net obtains the best performance when $E_1 = 20$ and the worst performance when $E_1 = 30$. This can be due to the overfitting caused by a large number of warm-up epochs. Then, we fix the $E_1 = 20$ and vary the value of E_2 from 30 to 80. From Table IX(e), we see that $E_2 = 60$ achieves the best results for CIFAR-10 while $E_2 = 80$ obtains the highest accuracy for CIFAR-100. It can be ascribed to the fact that CIFAR-100 contains more classes than CIFAR-10 and thus our method requires more epochs to learn for the semi-supervised alternate learning stage before label refinement.

Influence of unknown ϵ. In this paper, we empirically set the value of the noise rate ϵ for each dataset. Unfortunately, ϵ is unknown in some real-world applications. Thus, we also validate the performance of TS^3-Net without giving the accurate noise rate. Specifically, as done in [32], we select clean samples from noisy data by fitting Gaussian Mixture Models (GMMs) to the loss of training examples, where the clean posterior probability is set to 0.50. We perform experiments on CIFAR-10 under Symmetric-40% label noise and CIFAR-100 under Symmetric-40% label noise at five levels of sparsity.

As shown in Fig. 4, we evaluate the performance obtained by TS^3-Net with the accurate noise rate and TS^3-Net with GMMs. Compared with TS^3-Net with the accurate noise rate, the performance of TS^3-Net with GMMs slightly decreases. This demonstrates that TS^3-Net works well when GMMs are employed to select clean and noisy labeled samples.

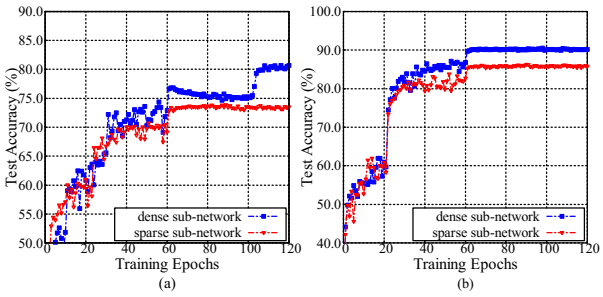

Fig. 6: Test accuracy vs. training epochs under (a) Symmetric-40% label noise and (b) clean data on CIFAR-10.

3) Effectiveness of Two-Stream Architecture: Our TS^3-Net involves a sparse sub-network and a dense sub-network. In TS^3-Net, we gradually prune the dense sub-network to obtain the sparse sub-network. During semi-supervised alternate learning, the dense sub-network and the sparse sub-network are updated alternately with the guidance of each other. To show the effectiveness of our two-stream architecture, we further evaluate some variants of our method: 1) our TS^3-Net with one sparse sub-network (denote as sparse). That is, the dense sub-network is first trained by the multi-stage learning framework (alternate learning is not used since only one sub-network is considered) and then a sparse sub-network is obtained from the trained dense sub-network by IMP; 2) our TS^3-Net with two sparse sub-networks initialized differently during the training (denoted as sparse+sparse); 3) our TS^3-Net with co-teaching, where alternate learning is replaced with co-teaching (denoted as dense+sparse (co-teaching)); and 4) our TS^3-Net. We evaluate our method with the different orders of alternate learning of $\mathcal{S}$ and $\mathcal{D}$. CIFAR-10 under Symmetric-40% label noise and CIFAR-100 under Symmetric-40% label noise are used. The results are given in Table X.

We have the following observations. First, in comparison to the methods with the two-stream architecture, the sparse method achieves worse performance. This shows the advantage of maintaining the two-stream architecture, which helps to alleviate the confirmation bias of sample selection. Second, our method with the dense and sparse sub-networks gives better performance than that with two sparse sub-networks. This can be ascribed to the superior learning capability of the dense sub-network to select clean samples in the second stage. Third, our methods under the different orders of alternate learning achieve basically the same accuracy. The order of alternate learning is not critical since $\mathcal{S}$ and $\mathcal{D}$ are trained in iterations. Fourth, our method with alternate learning outperforms that with co-teaching. That is, the joint learning of $\mathcal{S}$ and $\mathcal{D}$ decreases the model accuracy.

Fig. 5 further illustrates the test accuracy vs. training epochs, respectively obtained by TS^3-Net with co-teaching and TS^3-Net. As training proceeds, TS^3-Net obtains higher test accuracy than TS^3-Net with co-teaching. This may be ascribed to less error accumulation introduced by alternating learning.

Moreover, we compare the learning capability of sparse sub-network with that of dense sub-network, as shown in Fig. 6. Specifically, we apply the small-loss criterion (or the standard cross-entropy loss) to separately train the sparse sub-network

TABLE X: Test Accuracy (%) of different architecture designs.

Architecture Design	CIFAR-10	CIFAR-100
sparse	85.21	58.01
sparse+sparse	85.89	59.76
dense+sparse (co-teaching)	84.39	58.68
TS^3-Net^1	86.54	**60.61**
TS^3-Net^2	**86.71**	60.28

^{1}: $\mathcal{S}$ is trained before $\mathcal{D}$ during alternate learning; ^{2}: $\mathcal{D}$ is trained before $\mathcal{S}$ during alternate learning.

and the dense sub-network with noisy (or clean) supervision. The sparsity level is set to 93.6%. Clearly, the dense sub-network gives higher test accuracy than the sparse sub-network under both label noise and clean conditions. This validates the stronger learning capability of the dense sub-network.

4) Effectiveness of Multi-stage Learning Framework: Now we evaluate the effectiveness of multi-stage learning framework on synthetic and real-world noisy datasets. In particular, we replace the multi-stage learning framework (denoted as MSL) in TS^3-Net with five state-of-the-art LNL methods (including SCE [18], MAE [50], APL [29], ITLM [51], and Co-teaching [19]) and evaluate their corresponding performance. We re-implement these methods according to the source codes provided by their respective authors. Co-teaching and MSL train both $\mathcal{S}$ and $\mathcal{D}$ while the other methods only train $\mathcal{S}$. MNIST, CIFAR-10, CIFAR-100, and ANIMAL-10N are used for evaluation.

Results on MNIST Fig. 7 shows the test accuracy obtained by all the competing methods using a three-layer MLP. We can see that MSL outperforms the other state-of-the-art methods in most cases. Among all the methods, MAE is a robust loss based method, which easily suffers from underfitting. The performance of MAE significantly drops with the increasing levels of sparsity. This can be ascribed to the inferior learning capability of $\mathcal{S}$ when only MAE is used for training, leading to poor generalization. Compared with MAE, APL and SCE achieve better performance. This is because they use a convex combination consisting of an underfitting term and an overfitting term to balance the overfitting and underfitting. For the hard Symmetric-40% case, ITLM performs poorly since it unavoidably overfits some noisy labeled samples. Compared with Co-teaching, our MSL achieves better results due to the fact that MSL progressively enhances the learning capability of $\mathcal{S}$ and $\mathcal{D}$.

Results on CIFAR-10 The comparison results on CIFAR-10 are reported in Fig. 8. For the Asymmetric-20% case, ITLM gives better performance than robust loss based methods (such as SCE and MAE). This is because that ITLM leverages clean samples to calculate the cross-entropy loss, which can alleviate the influence of noisy labels during the training of $\mathcal{S}$. For the Symmetric-40% case, Co-teaching gives the best performance among all the competing methods, except for MSL that selects more reliable samples by using two sub-networks.

Results on CIFAR-100 The comparison results on CIFAR-100 are shown in Fig. 9 for the Symmetric-20%, Symmetric-40%, and Asymmetric-20% cases, respectively. The CIFAR-100 dataset contains 100 classes and is more challenging than CIFAR-10. Hence, the test accuracy of most competing methods is less than 70%. For the Symmetric-20% case, MSL

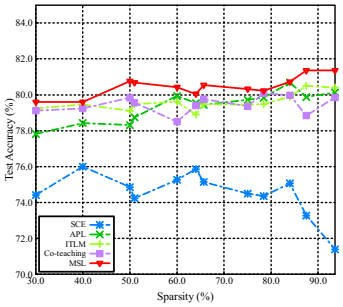

Fig. 10: Test accuracy obtained by different methods at different levels of sparsity on ANIMAL-10N.

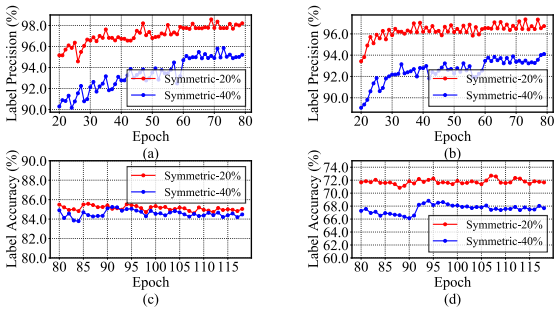

Fig. 11: The label precision during the semi-supervised alternate learning stage on (a) CIFAR-10 and (b) CIFAR-100, respectively. The pseudo-label accuracy during the label refinement stage on (c) CIFAR-10 and (d) CIFAR-100, respectively.

obtains significantly better test accuracy than the ITLM and Co-teaching methods (about 6% and 2% improvements in average). For the Asymmetric-20% case, MSL still outperforms the other competing methods. The performance of both SCE and APL for the Symmetric-40% case is significantly worse (about 15% and 10% decreases in average) than that for the Symmetric-20% case. This is because that the overfitting term in these methods dominates the overall training under severe noise conditions. Such a way is determinant to alleviate the influence of label noise. Among all the methods, MSL achieves the best results under different label noise conditions and different levels of sparsity.

Results on ANIMAL-10N The comparison results on the ANIMAL-10N dataset are given in Fig. 10. We can see that APL consistently outperforms SCE at all the levels of sparsity. Moreover, our proposed MSL outperforms the other competing methods at different levels of sparsity. ITLM, Co-teaching, and MSL achieve better results than APL. This shows the importance of sample selection in the real-world noisy dataset.

5) Visualization: In this subsection, we first visualize the label precision obtained by the sparse sub-network $\mathcal{S}$ during the semi-supervised alternate learning stage and the pseudo-label accuracy obtained by $\mathcal{S}$ during the label refinement stage in Fig. 11. CIFAR-10 under Symmetric-20% and Symmetric-40% label noise, and CIFAR-100 under Symmetric-20% and Symmetric-40% label noise are used for evaluation.

As shown in Figs. 11(a) and 11(b), the label precision obtained by $\mathcal{S}$ is constantly improved during the semi-supervised

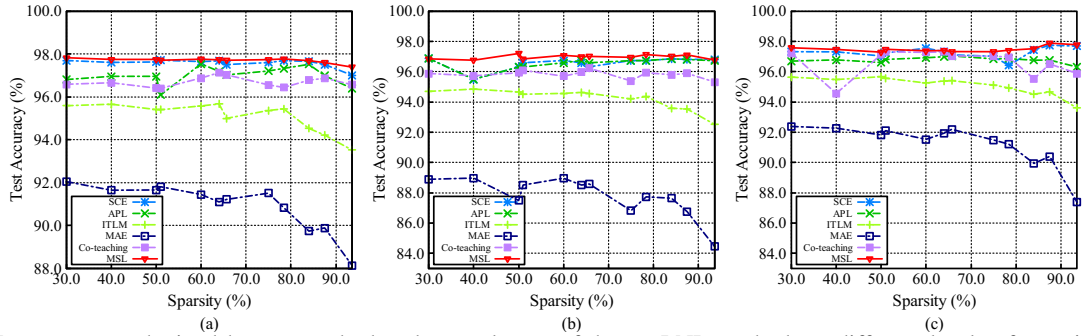

Fig. 7: Test accuracy obtained by our method and several state-of-the-art LNL methods at different levels of sparsity under three label noise conditions, including (a) Symmetric-20%, (b) Symmetric-40% and (c) Asymmetric-20%, on MINIST.

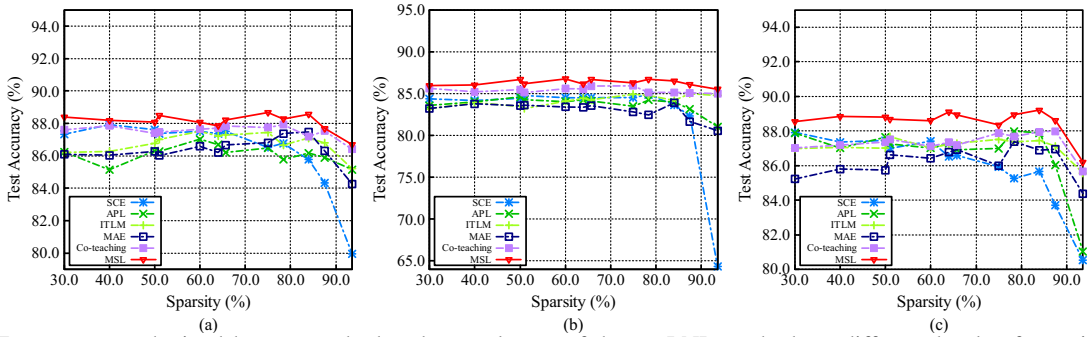

Fig. 8: Test accuracy obtained by our method and several state-of-the-art LNL methods at different levels of sparsity under three label noise conditions, including (a) Symmetric-20%, (b) Symmetric-40% and (c) Asymmetric-20%, on CIFAR-10.

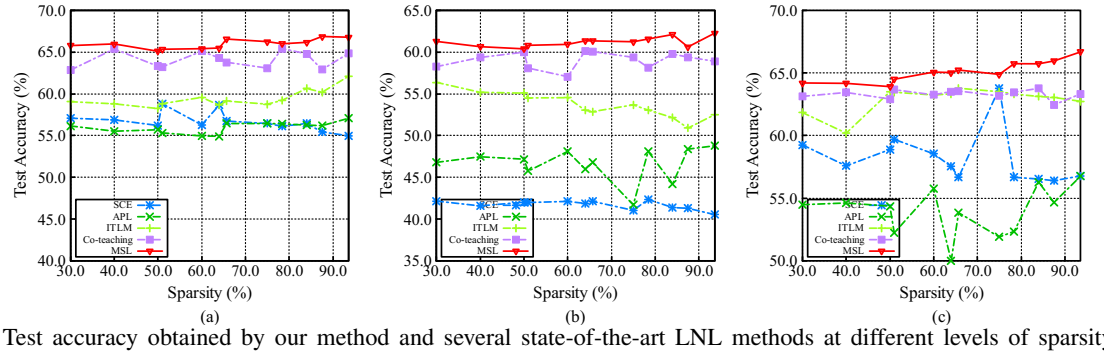

Fig. 9: Test accuracy obtained by our method and several state-of-the-art LNL methods at different levels of sparsity under three label noise conditions, including (a) Symmetric-20%, (b) Symmetric-40% and (c) Asymmetric-20%, on CIFAR-100.

alternate learning stage. The improved label quality can facilitate the training of $\mathcal{S}$, resulting in the enhanced classification capability of $\mathcal{S}$. As shown in Figs. 11(c) and 11(d), the pseudo-label accuracy obtained by $\mathcal{S}$ keeps stable during the label refinement stage. This shows the effectiveness of our relabeling strategy.

6) Robustness against Different Network Architectures: Here, we report the performance of our proposed TS^3-Net based on different network architectures (i.e., eight-layer CNN, ResNet-18, and VGG-16) as backbones. We evaluate TS^3-Net under the Symmetric-40% noise condition on CIFAR-10. The results are shown in Table XI. The TS^3-Net (warm) method is also given for a comparison.

TS^3-Net based on VGG-16 gives better results than that based on the other two network architectures, even at a very high sparsity level (i.e., 93.6%). Compared with the TS^3-Net (warm), TS^3-Net achieves much higher accuracy on three network architectures. Among these networks, VGG-16 has the largest number of parameters while the eight-layer CNN has the smallest number of parameters. However, TS^3-Net based on the eight-layer CNN still obtains comparable performance to that based on VGG-16. Overall, the above experimental results demonstrate the robustness of the proposed TS^3-Net against different network architectures.

7) Winning Ticket Visualization: To further investigate the superiority of TS^3-Net, we visualize the distribution of ini-

TABLE XI: Accuracy (%) with three network architectures at different levels of sparsity under Symmetric-40% label noise on CIFAR-10. The architecture of the dense sub-network and its corresponding number of parameters are also given.

Network & Params (M)	Method	Sparsity%											
		30	40	50	51	60	64	65.7	75	78.4	84	87.5	93.6
eight-layer CNN (≈ 0.52)	TS^3-Net (warm)	78.01	77.32	78.58	78.75	77.34	78.66	79.39	79.56	79.49	78.21	80.52	80.02
	TS^3-Net	**85.96**	**86.03**	**86.68**	**86.18**	**86.75**	**86.14**	**86.68**	**86.28**	**86.71**	**86.51**	**86.08**	**85.51**
ResNet-18 (≈ 11.69)	TS^3-Net (warm)	78.91	79.50	80.76	80.57	81.08	81.35	81.55	81.04	80.20	82.35	81.56	82.2
	TS^3-Net	**85.78**	**85.99**	**86.04**	**86.67**	**86.43**	**86.55**	**85.90**	**85.78**	85.67	**85.65**	**85.33**	**84.78**
VGG-16 (≈ 138.36)	TS^3-Net (warm)	60.05	64.47	62.72	61.75	62.25	59.72	59.51	61.38	61.87	62.26	63.13	59.62
	TS^3-Net	**87.62**	**86.90**	**87.11**	**87.51**	**87.23**	**87.52**	**88.08**	**87.73**	**88.09**	**88.12**	**88.62**	**88.65**

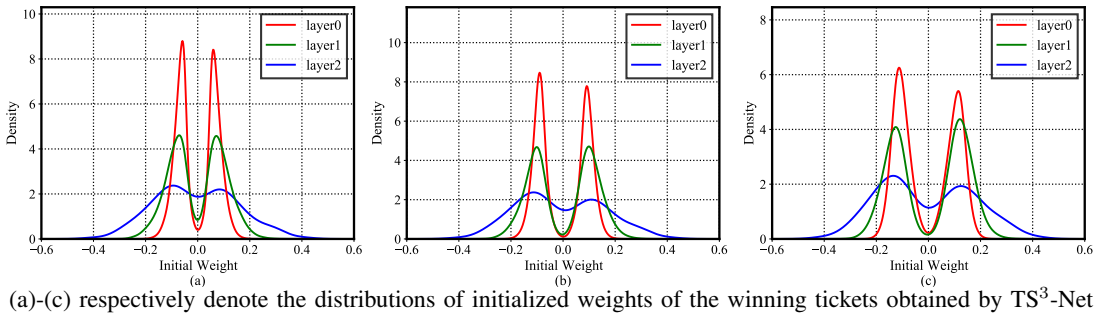

Fig. 12: (a)-(c) respectively denote the distributions of initialized weights of the winning tickets obtained by TS^3-Net at three levels of sparsity (i.e., 60%, 84%, and 93.6%) under Symmetric-20% noise on MNIST. The red, green and blue lines show the distributions for the first hidden layer, second hidden layer, and output layer of the three-layer MLP architecture, respectively.

tialized weights of the winning ticket obtained by TS^3-Net on MNIST. Fig. 12 shows the distributions at three different levels of sparsity (i.e., three columns) in the case of Symmetric-20% label noise. We can see that the bimodal distributions are present across all layers for initialized weights of the winning ticket, as validated in [12]. Moreover, the distributions are similar at each sparsity level. Hence, TS^3-Net can effectively search for the winning tickets under the label noise condition.

V. Conclusion and Future Work

In this paper, we have proposed a novel TS^3-Net, consisting of a sparse sub-network and a dense sub-network, to effectively identify the winning ticket under label noise. Based on our developed multi-stage learning framework, TS^3-Net is able to learn a sparse sub-network that has extremely low memory usage and good classification capability, when it is trained on noisy labeled data. Extensive experimental results on synthetic and real-world noisy benchmark datasets have shown the effectiveness of our method in comparison with several state-of-the-art LNL methods.

Currently, our method works in the closed-set settings, where only in-distribution label noise is considered. In the future, we will extend our method to the more challenging out-of-distribution label noise.

References

[1] Y. Wang, Z.-P. Bian, J. Hou, and L.-P. Chau, "Convolutional neural networks with dynamic regularization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 5, pp. 2299–2304, 2020.

[2] P. Tang, X. Wang, B. Shi, X. Bai, W. Liu, and Z. Tu, "Deep fishernet for image classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 7, pp. 2244–2250, 2018.

[3] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2961–2969.

[4] Y. Chen, Y. Cao, H. Hu, and L. Wang, "Memory enhanced global-local aggregation for video object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10 337–10 346.

[5] X. Ding, G. Ding, J. Han, and S. Tang, "Auto-balanced filter pruning for efficient convolutional neural networks," in *Proc. AAAI Conf. Art. Intell.*, vol. 32, no. 1, 2018, pp. 6797–6804.

[6] S. Srinivas and R. V. Babu, "Data-free parameter pruning for deep neural networks," *2015, arXiv:1507.06149. [Online]. Available: https://arxiv.org/abs/1507.06149*.

[7] M. Zhu and S. Gupta, "To prune, or not to prune: Exploring the efficacy of pruning for model compression," *2017, arXiv:1710.01878. [Online]. Available: https://arxiv.org/abs/1710.01878*.

[8] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, "Quantized convolutional neural networks for mobile devices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 4820–4828.

[9] Z. Yang, M. Moczulski, M. Denil, N. De Freitas, A. Smola, L. Song, and Z. Wang, "Deep fried convnets," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1476–1483.

[10] S.-K. Yeom, P. Seegerer, S. Lapuschkin, A. Binder, S. Wiedemann, K.-R. Müller, and W. Samek, "Pruning by explaining: A novel criterion for deep neural network pruning," *Pattern Recognit.*, vol. 115, pp. 1–14, 2021.

[11] C. Kaplan and A. Bulbul, "Goal driven network pruning for object recognition," *Pattern Recognit.*, vol. 110, pp. 1–11, 2021.

[12] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *2018, arXiv:1803.03635. [Online]. Available: https://arxiv.org/abs/1803.03635*.

[13] J. Frankle, G. K. Dziugaite, D. M. Roy, and M. Carbin, "Stabilizing the lottery ticket hypothesis," *2019, arXiv:1903.01611. [Online]. Available: https://arxiv.org/abs/1903.01611*.

[14] H. Wei, L. Feng, X. Chen, and B. An, "Combating noisy labels by agreement: A joint training method with co-regularization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 13 726–13 735.

[15] B. Frénay and M. Verleysen, "Classification in the presence of label noise: a survey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 5, pp. 845–869, 2013.

[16] H. Song, M. Kim, D. Park, Y. Shin, and J.-G. Lee, "Learning from noisy labels with deep neural networks: A survey," *2020, arXiv:2007.08199. [Online]. Available: https://arxiv.org/abs/2007.08199*.

[17] F. R. Cordeiro and G. Carneiro, "A survey on deep learning with noisy labels: How to train your model when you cannot trust on the annotations?" in *SIBGRAPI Conf. Graph., Patterns Images*, 2020, pp. 9–16.

[18] Y. Wang, X. Ma, Z. Chen, Y. Luo, J. Yi, and J. Bailey, "Symmetric cross entropy for robust learning with noisy labels," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 322–330.

[19] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. Tsang, and M. Sugiyama, "Co-teaching: Robust training of deep neural networks with extremely noisy labels," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 8536–8546.

[20] X. Yu, B. Han, J. Yao, G. Niu, I. Tsang, and M. Sugiyama, "How does disagreement help generalization against label corruption?" in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7164–7173.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[22] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *2014, arXiv:1409.1556. [Online]. Available: https://arxiv.org/abs/1409.1556*.

[23] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Proc. Adv. Neural Inf. Process. Syst.*, 1990, pp. 598–605.

[24] B. Hassibi, D. G. Stork, and G. J. Wolff, "Optimal brain surgeon and general network pruning," in *Proc. IEEE. Int. Conf. Neural. Networks.*, 1993, pp. 293–299.

[25] E. Malach, G. Yehudai, S. Shalev-Schwartz, and O. Shamir, "Proving the lottery ticket hypothesis: Pruning is all you need," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 6682–6691.

[26] S. Girish, S. R. Maiya, K. Gupta, H. Chen, L. Davis, and A. Shrivastava, "The lottery ticket hypothesis for object recognition," *2020, arXiv:2012.04643. [Online]. Available: https://arxiv.org/abs/2012.04643*, 2020.

[27] T. Chen, J. Frankle, S. Chang, S. Liu, Y. Zhang, M. Carbin, and Z. Wang, "The lottery tickets hypothesis for supervised and self-supervised pre-training in computer vision models," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 16 306–16 316.

[28] X. Zhou, X. Liu, J. Jiang, X. Gao, and X. Ji, "Asymmetric loss functions for learning with noisy labels," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 12846–12856.

[29] X. Ma, H. Huang, Y. Wang, S. Romano, S. Erfani, and J. Bailey, "Normalized loss functions for deep learning with noisy labels," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 6543–6553.

[30] 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 *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 2304–2313.

[31] Y. Yao, Z. Sun, C. Zhang, F. Shen, Q. Wu, J. Zhang, and Z. Tang, "Jo-SRC: A contrastive approach for combating noisy labels," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 5192–5201.

[32] J. Li, R. Socher, and S. C. Hoi, "DivideMix: Learning with noisy labels as semi-supervised learning," in *Int. Conf. Learn. Represent.*, 2019.

[33] T. Nguyen, C. Mummadi, T. Ngo, L. Beggel, and T. Brox, "SELF: learning to filter noisy labels with self-ensembling," in *Int. Conf. Learn. Represent.*, 2020.

[34] D. Arpit, S. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. C. Courville, Y. Bengio, and S. Lacoste-Julien, "A closer look at memorization in deep networks," in *Proc. Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 233–242.

[35] Y. Wang, R. Huang, G. Huang, S. Song, and C. Wu, "Collaborative learning with corrupted labels," *Neural Netw.*, vol. 125, pp. 205–213, 2020.

[36] S. Cicek, A. Fawzi, and S. Soatto, "SaaS: Speed as a supervisor for semi-supervised learning," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 149–163.

[37] S. Liu, J. Niles-Weed, N. Razavian, and C. Fernandez-Granda, "Early-learning regularization prevents memorization of noisy labels," *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 1–26, 2020.

[38] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. Raffel, "MixMatch: A holistic approach to semi-supervised learning," *2019, arXiv:1905.02249. [Online]. Available: https://arxiv.org/abs/1905.02249*.

[39] Y. Grandvalet, Y. Bengio *et al.*, "Semi-supervised learning by entropy minimization." *CAP*, vol. 367, pp. 281–296, 2005.

[40] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, "Self-training with noisy student improves imagenet classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit*, 2020, pp. 10 687–10 698.

[41] Y. Wang, J. Hou, X. Hou, and L.-P. Chau, "A self-training approach for point-supervised object detection and counting in crowds," *IEEE Trans. Image Process.*, vol. 30, pp. 2876–2887, 2021.

[42] D.-H. Lee *et al.*, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Workshop on challenges in representation learning, ICML*, vol. 3, no. 2, 2013, pp. 2–7.

[43] K. Sohn, D. Berthelot, C.-L. Li, Z. Zhang, N. Carlini, E. D. Cubuk, A. Kurakin, H. Zhang, and C. Raffel, "FixMatch: Simplifying semi-supervised learning with consistency and confidence," *2020, arXiv:2001.07685. [Online]. Available: https://arxiv.org/abs/2001.07685*.

[44] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[45] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," pp. 1–60, 2009.

[46] H. Song, M. Kim, and J.-G. Lee, "SELFIE: Refurbishing unclean samples for robust deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 5907–5915.

[47] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang, "Learning from massive noisy labeled data for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 2691–2699.

[48] W. Li, L. Wang, W. Li, E. Agustsson, and L. Van Gool, "Webvision database: Visual learning and understanding from web data," *arXiv preprint arXiv:1708.02862. [Online]. Available: https://arxiv.org/abs/1708.02862*, 2017.

[49] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proc. AAAI Conf. Art. Intell*, 2017.

[50] A. Ghosh, H. Kumar, and P. Sastry, "Robust loss functions under label noise for deep neural networks," in *Proc. AAAI Conf. Art. Intell.*, vol. 31, no. 1, pp. 1919–1925.

[51] Y. Shen and S. Sanghavi, "Learning with bad training data via iterative trimmed loss minimization," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 5739–5748.

[52] G. Patrini, A. Rozza, A. Krishna Menon, R. Nock, and L. Qu, "Making deep neural networks robust to label noise: A loss correction approach," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1944–1952.

[53] X. Ma, Y. Wang, M. E. Houle, S. Zhou, S. Erfani, S. Xia, S. Wijewickrema, and J. Bailey, "Dimensionality-driven learning with noisy labels," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 3355–3364.

[54] E. Malach and S. Shalev-Shwartz, "Decoupling" when to update" from" how to update"," *arXiv preprint arXiv:1706.02613. [Online]. Available: https://arxiv.org/abs/1706.02613*, 2017.

[55] P. Chen, B. B. Liao, G. Chen, and S. Zhang, "Understanding and utilizing deep neural networks trained with noisy labels," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 1062–1070.

[56] L. Van der Maaten and G. Hinton, "Visualizing data using T-SNE." *J. Machine Learning Res*, vol. 9, no. 11, 2008.