Let the Noise Flow Away: Combating Noisy Labels using Normalizing Flows

Kuan-An Su[1], YiHao Su[1], Yun-Hsuan Lien[1*], Yu-Shuen Wang[1]

[1*]Department of Computer Science, National Yang Ming Chiao Tung University, Engineering Bldg 3, 1001 University Road, Hsinchu, 300, Taiwan.

*Corresponding author(s). E-mail(s): sophia.yh.lien@gmail.com; Contributing authors: a07458666@gmail.com; suyihao1999@gmail.com; yushuen@cs.nctu.edu.tw;

Abstract

We introduce NoiseFlow, a generative network that addresses the issue of noisy labels in classification problems by modeling the entire label distribution based on the input data/image. Unlike previous methods, which assign each input to only one specific class, NoiseFlow generates different labels by considering the image and a random noise drawn from a standard normal distribution. This approach improves generalization performance since it does not require extensive parameter adjustments to fit the unknown data noise. To model the label distribution, we use conditional normalizing flows, which are effective at avoiding mode collapse and ensuring the presence of the correct label in the distribution for accurate classification. Moreover, NoiseFlow can be combined with other training strategies, such as mixup interpolation and contrastive learning, to achieve even better performance. We compared NoiseFlow with baseline methods on several synthetic and real-world datasets, and the experiment results demonstrate its effectiveness.

Keywords: Noisy Labels, Uncertainty, Generalization, Normalizing Flows

1 Introduction

Classification is the process of assigning input samples to specific labels. However, if the data used for training contains errors, networks trained on such a dataset will

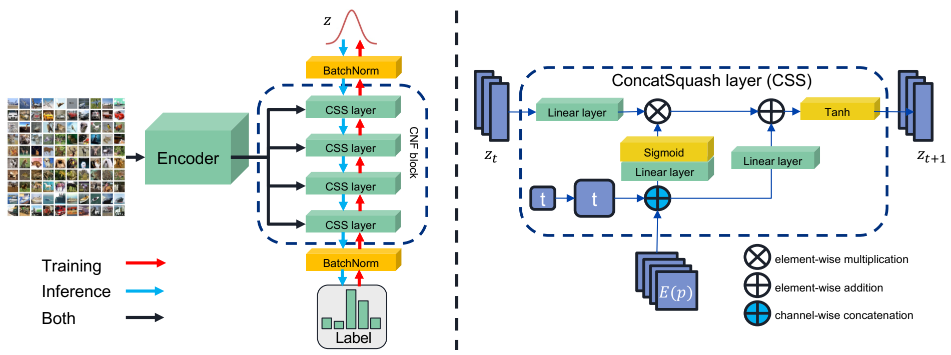

Fig. 1: The network architecture of NoiseFlow. (Left) During training, the conditional normalizing flow (CNF) block seeks to convert the encoded representations and label vectors into a standard normal distribution. During inference, the block employs noise vectors to generate the label distribution based on the representation. (Right) The structure of a ConcatSquash layer.

attempt to accommodate the incorrect labels, resulting in poor generalization performance (Li et al, 2020b). To mitigate this, recent methods correct the labels (Ma et al, 2018; Yi and Wu, 2019; Tanaka et al, 2018) or remove the noisy samples (Malach and Shalev-Shwartz, 2017; Jiang et al, 2018; Han et al, 2018; Yu et al, 2019; Wei et al, 2020) before the network has a chance to memorize them. There are also methods that interpolate the samples and their respective labels (Zhang et al, 2018; Ortego et al, 2021), making it more challenging for the network to memorize and thereby enhancing its generalization.

Feedforward networks are limited to assigning just one label to each input, and they require extensive parameter adjustments to learn from data with noise (Li et al, 2020b). To overcome this, we present NoiseFlow, a network that can associate an input with multiple labels. Our solution involves training conditional normalizing flows (CNFs) (Chen et al, 2018; Grathwohl et al, 2019) to perform classification tasks. CNFs are a type of generative model that uses a random noise vector and an input image (i.e., condition) to produce a label distribution that corresponds to the image. As illustrated in Figure 1 left, NoiseFlow consists of an encoder that extracts features from the input and a CNF block that combines the feature with noise vectors to generate labels. By allowing the CNF to model the entire label distribution, it reduces the impact caused by incorrect labels. The generated label distributions will be more precise if the dataset is clean, and more uncertain if the dataset contains noise. Additionally, if the dataset does not contain purely noise, the density of the correct label will still be higher than that of the incorrect labels.

Our NoiseFlow models the entire label distribution to address the noisy label problem. It can be combined with existing methods to achieve even better performance. First, we use mixup interpolation (Zhang et al, 2018) to create virtual examples for the network to train on. This helps to resolve the challenge of training a network to generate a sharp/discrete distribution of one-hot vectors. Second, we use contrastive learning (Oord et al, 2018; Chen et al, 2020) to enable the network to learn robust representations, as this method has been demonstrated to be effective in enhancing generalization. Third, although the presence of noisy samples in the dataset can harm network training, simply removing them based on loss or disagreement with ground-truth labels may overlook clean but difficult examples. Given that samples between classes may have varying levels of difficulty, we select the same ratios of clean samples from each class, ensuring that we don't miss out on any challenging but useful examples (Karim et al, 2022). Finally, we leverage the estimated noisy samples as unlabeled data and use semi-supervised learning (Sohn et al, 2020) to train the network with pseudo-labels. One inherent risk is that the network, while generating these pseudo-labels and subsequently training on them, could end up producing homogenous pseudo-labels (Sohn et al, 2020). To mitigate this, we incorporate sharpening (Sohn et al, 2020) and centering (Caron et al, 2021) techniques, ensuring these pseudo-labels are fine-tuned to truly represent their respective classes.

We tackle the issue of noisy labels using a generative network that models the entire label distribution for each input, further enhanced with noise vectors to minimize the effect of these labels. Our evaluations, benchmarked against standard methodologies, underscore the efficacy of our approach. Specifically, our results suggest that NoiseFlow is particularly useful for datasets with high noise ratios. NoiseFlow registered an accuracy of 60.4% on Cifar-100 with a 90% symmetric noise ratio, substantially outperforming baseline methods that yielded a 44.8% accuracy. On real-world datasets, NoiseFlow recorded accuracies of 67.45% and 78.44% for Clothing1M and WebVision, respectively, surpassing the state-of-the-art method, which posted accuracies of 61.01% and 77.6% on these datasets.

2 Related Works

When manually annotating data for network training, it is common to encounter incorrect labels. Since memorizing corrupted data would negatively impact the network's generalization performance, several methods have been proposed to address this issue (Song et al, 2022). Previous techniques can generally be classified into two categories: (1) label correction and (2) separation of clean samples.

Methods in the first category gradually adjust the training loss (Goldberger and Ben-Reuven, 2017; Patrini et al, 2017) or the label (Ma et al, 2018; Yi and Wu, 2019; Tanaka et al, 2018) during training. They estimate the noise rates (Patrini et al, 2017; Tanaka et al, 2018; Northcutt et al, 2017) and the noise transition matrix (Goldberger and Ben-Reuven, 2017; Patrini et al, 2017) to correct the labels. Some methods relax the assumption that a small subset of data is trusted and utilize the trusted data to improve performance (Hendrycks et al, 2018). However, estimating the noise rate and transition matrix is challenging, and the performance of these methods can suffer significantly when the noise rate or the number of classes is high.

In the second category, methods focus on selecting clean samples for network training. To avoid bias, these methods typically rely on two networks (Nguyen et al, 2020), with one used to select clean samples for the other. Strategies for sample selection may vary, including using the variance of prediction during training, disagreement of the outputs (Malach and Shalev-Shwartz, 2017), and magnitude of loss values (Jiang et al, 2018; Han et al, 2018). The strategies can also be combined to further enhance network performance (Yu et al, 2019; Wei et al, 2020). Moreover, recent methods apply semi-supervised learning to utilize the corrupted data by training the network on both the estimated clean and noisy (i.e., unlabeled) samples. Specifically, clean data is trained using the cross-entropy loss, while corrupted data is trained with consistency regularization (Ding et al, 2018; Yao et al, 2021) and contrastive learning. While separating clean and corrupted samples may require a hyperparameter, Li et al (2020a) and Nishi et al (2021) leveraged the Gaussian mixture model to estimate the two distributions based on the predicted confidences. In addition, to address the different difficulties of classes, such as distinguishing between dogs and cats, Karim et al (2022) assumed that the numbers of clean samples in each class are similar. Although this assumption could be strong, the applied priors result in significant improvements in network generalization.

Besides label correction and separation of clean samples, Zhang et al (2018) and Ortego et al (2021) proposed using sample interpolation to address the noisy label problem, as learning the interpolations of correct samples is easier than memorizing the interpolations involving random labels (Zhang et al, 2018). Our proposed method, which uses normalizing flows, also does not fall into the two mentioned categories, as it learns the label distribution instead of a single label for each input. By modeling both correct and incorrect labels, our method can reduce the memorizing effect when fitting noise, and can work together with existing training strategies to further improve generalization performance.

3 Background

3.1 Problem Definition

Given a dataset $D = \{(\mathbf{p}_0, \mathbf{q}_0), (\mathbf{p}_1, \mathbf{q}_1), ..., (\mathbf{p}_n, \mathbf{q}_n)\}$, where $\mathbf{p}$ and $\mathbf{q}$ denote the input sample and its corresponding label, and n is the size of D. By minimizing the cross-entropy loss, a deep neural network that includes an encoder E_θ and a classification layer G_ϕ learns to map each input $\mathbf{p}$ to its corresponding label $\mathbf{q}$. The cross-entropy loss can be mathematically expressed as follows:

$$\mathcal{L}_{CE} = -\frac{1}{n}\sum_{i=1}^{n} \mathbf{q}_i^T \log \hat{\mathbf{q}}_i, \tag{1}$$

where $\hat{\mathbf{q}}_i = \mathrm{softmax}(\mathrm{G}_\phi(\mathrm{E}_\theta(\mathbf{p}_\mathrm{i})))$ is the estimated label probability of $\mathbf{p}_i$. When dealing with noisy label training, the training set D_{train} contains corrupted labels. For example, an image with a car may be labeled as a person. On the other hand, the testing set $D_{test} = D \setminus D_{train}$ is considered to be clean and used for evaluation.

3.2 Normalizing Flows

Our approach is based on utilizing normalizing flows (NFs) (Dinh et al, 2017; Kingma and Dhariwal, 2018; Dinh et al, 2014; Chen et al, 2018; Grathwohl et al, 2019), which is a type of generative model. The primary idea behind NFs is to use a series of invertible transformations to map samples from a standard normal distribution $p_z(z)$ to the target distribution $p_x(x)$. Let $F : R_d \rightarrow R_d$ represent an NF, and let x and z be samples drawn from $p_x(x)$ and $p_z(z)$, respectively. The relationship between x and z can be expressed as $z = F_\eta^{-1}(x)$, $x = F_\eta(z)$, where η denotes the parameters of F.

NFs models can be classified into two categories: discrete normalizing flows (Dinh et al, 2017; Kingma and Dhariwal, 2018; Dinh et al, 2014) and continuous normalizing flows (Chen et al, 2018; Grathwohl et al, 2019), depending on the specific designs of the invertible layers. We employ continuous NFs in this paper to model the distributions of noisy labels because they are more expressive than discrete NFs.

Conditional Continuous Normalizing Flows

A continuous NF uses a continuous-time transformation to map a simple probability distribution to a complex target distribution. This transformation is determined by a system of ordinary differential equations (ODEs), which describe how the variables change over time (Chen et al, 2018; Grathwohl et al, 2019):

$$\frac{d\mathbf{z}_t}{dt} = f(t, \mathbf{y}, \mathbf{z}_t), \tag{2}$$

where $\mathbf{z}_t$ is the state of $\mathbf{z}$ at time t, $\mathbf{z}_0 \sim \mathcal{N}(\mathbf{0}, \mathcal{I})$ is a random vector, and $\mathbf{y}$ is the condition. The function f parameterizes how a data point $\mathbf{z}_{t_0}(= \mathbf{z})$ evolves to $\mathbf{z}_{t_1}(= \mathbf{x})$ at each time t. It also satisfies uniform Lipschitz continuity at each $\mathbf{z}_t$, which allows for the invertibility of f. The relationship between $\mathbf{x}$ and $\mathbf{z}$ is therefore expressed as follows:

$$\mathbf{x} = F(\mathbf{z}_{t_0}) = \mathbf{z}_{t_0} + \int_{t_0}^{t_1} f(t, \mathbf{y}, \mathbf{z}_t)dt. \tag{3}$$

In this context, one can train the network F^{-1} by minimizing the negative log-likelihood (NLL) of the conditioned distribution:

$$\mathcal{L} = -\log p_{x|y}(\mathbf{x}|\mathbf{y}) = -\log p_z(\mathbf{z}_{t_0}) + \int_{t_0}^{t_1} tr\left(\frac{\partial f}{\partial \mathbf{z}_t}\right) dt. \tag{4}$$

The ODEs are typically solved using numerical integration methods such as Runge-Kutta or the adjoint method. We apply the adjoint method (Pontryagin et al, 1962) to compute the gradients for updating network parameters. We refer readers to (Chen et al, 2018; Grathwohl et al, 2019) for the equations and optimization details.

4 Noisy Label Training

Typically, classification methods employ an encoder E_θ to transform an input sample/image $\mathbf{p}$ into a compact representation, which is then fed into a classification layer G_ϕ to estimate the probability of $\mathbf{p}$ belonging to a particular class. For samples that belong to the same class, particularly those that are visually similar, E_θ tends to encode $\mathbf{p}$ into similar representations, enabling the classification layer G_ϕ to classify them into the same class accurately. However, in cases where visually similar samples are mislabeled and need to be classified into different classes, the encoder E_θ must amplify the differences between their representations, or the classification layer G_ϕ must exhibit high nonlinearity to minimize the cross-entropy loss (Li et al, 2020b). Unfortunately, these solutions can lead to poor generalization of the network.

4.1 NoiseFlow: Label Distribution Learning

Our approach involves training a generative network to produce a label distribution (i.e., a distribution of label probabilities), which includes both correct and incorrect labels, conditioned on the input. Suppose the majority label is correct, and the generator can accurately model the label distribution. In that case, the density of the correct label in the generated distribution will be higher than that of the incorrect label. To accomplish this, we utilize conditional NFs F_η to learn the label distribution. Since NFs have a lower level of expressiveness due to their traceability and invertibility limitations, we first extract representations from input samples using an encoder E_θ and then utilize these representations as a condition of the NF. The network architecture is shown in Figure 1. Note that an NF is a bidirectional neural network, with training and inference proceeding in opposite directions. During the training stage, the NF transforms the label distribution into a standard normal distribution, and its parameters are updated by minimizing the negative log-likelihood, which can be represented as follows:

$$\mathcal{L}_{n\ell\ell} = -\log p_{q|E(\mathbf{p})}(\mathbf{q}|E(\mathbf{p})) = -\log p_z(\mathbf{z}_{t_0}) + \int_{t_0}^{t_1} tr\left(\frac{\partial f}{\partial \mathbf{z}_t}\right) dt. \tag{5}$$

In this equation, $\mathbf{p}$ is the input, E is the encoder, and $\mathbf{q}$ is a one-hot vector that denotes the label of $\mathbf{p}$. Once training is complete, the NF can be utilized to generate a label distribution based on the representation $E(\mathbf{p})$ and random noise. Let $\hat{\mathbf{q}}$ be a generated label. The process is formally written as:

$$\hat{\mathbf{q}} = \mathbf{z}_{t_0} + \int_{t_0}^{t_1} f(t, E_\theta(\mathbf{p}), \mathbf{z}_t)dt. \tag{6}$$

As previously mentioned, $\mathbf{z}_{t_0}$ is a random vector drawn from the normal distribution, which enables an NF to generate multiple labels even when conditioned on the same input. This characteristic is beneficial in mitigating the impact of incorrect labels. In practice, when inferring the label of an input, we set $\mathbf{z}_{t_0} = \mathbf{0}$ because it represents the center of the standard normal distribution with the highest probability.

We utilize NFs instead of adversarial generative networks (Goodfellow et al, 2014) to learn the label distribution since NFs do not suffer from mode collapse (Winkler et al, 2019), which occurs when the model fails to produce data that is as diverse as the real-world data distribution. This is problematic for noisy label training since the label distribution may not include the correct label if mode collapse occurs, resulting in reduced network performance. Since the NF is a bi-directional network, if the network can convert the label distribution to a normal distribution during training, it must be able to replicate the label distribution during inference.

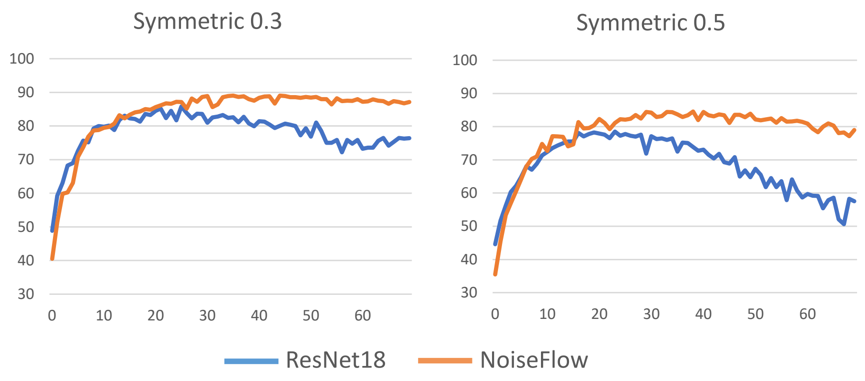

Fig. 2: We conducted a preliminary experiment on the Cifar-10 dataset. In each line chart, x and y axes indicate the training epoch and the measured testing accuracy, respectively. The results suggest that NFs can mitigate the effects of noisy data. In this experiment, we did not use any training techniques to address the issue of noisy labels.

Preliminary Experiment

We conducted a preliminary experiment to investigate the notion that learning the entire label distribution can help mitigate the impact of erroneous labels. In this study, we adopted the standard configuration used in (Han et al, 2018) and evaluated the performance of ResNet18 and NoiseFlow on the Cifar-10 dataset. While both networks used the same encoder, the classification layer of NoiseFlow was replaced with an NF block. Notably, our experiment was solely focused on assessing the effectiveness of learning the complete label distribution, and we did not utilize any noisy data training techniques.

Results from the experiment, as depicted in Figure 2, show that NoiseFlow outperformed ResNet18 in terms of testing accuracy when faced with incorrect labels. Furthermore, when compared with ResNet18, NoiseFlow's accuracy demonstrated a more gradual decline in datasets with symmetric noise rates of 0.3 and 0.5 (i.e., 30% and 50% of labels in the dataset are randomly reassigned). These results imply that enabling the network to map a single condition to various labels can mitigate the effects of incorrect labels. The slower degradation also allows the network to recognize clean samples and rectify the noisy labels when incorporating techniques for training on noisy data.

4.2 Combining Existing Training Strategies

The presented NoiseFlow can be combined with existing training methods to tackle the problem of noisy labels because they address noisy label problems from different perspectives. The conditional normalizing flow enables the model to map an input to multiple labels, thereby reducing the negative impact of noisy labels. Concurrently, the existing training strategies focus on selecting clean samples for network training, which aids in modeling sharper label distributions through conditional normalizing flows.

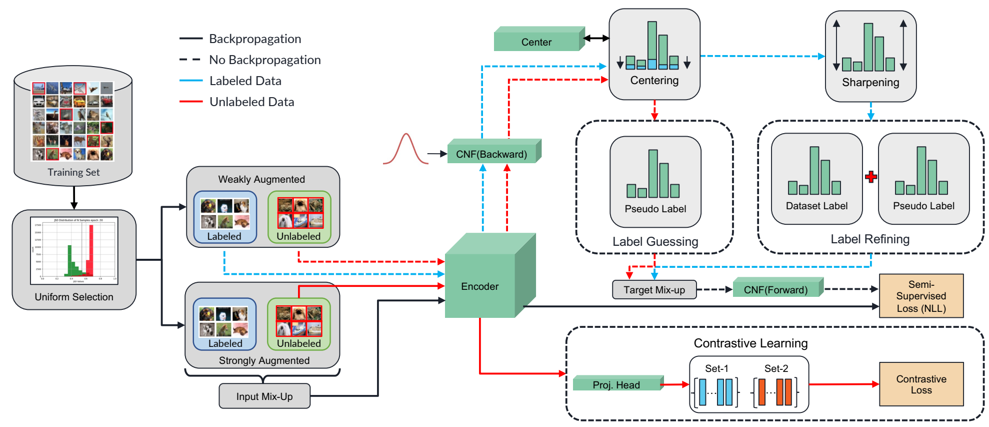

Fig. 3: The training of NoiseFlow involves four strategies: (1) mixup interpolation (Zhang et al, 2018), (2) contrastive learning (Chen et al, 2020), (3) semi-supervised training (Sohn et al, 2020), and (4) uniform selection (Karim et al, 2022).

Since a portion of incorrect labels is removed, sharper generated label distributions result in more confident and accurate predictions. Together, these strategies enhance network performance. We illustrate the training process in Figure 3 and describe the details in the following paragraphs.

Mixup Interpolation

During training, NoiseFlow aims to map a label distribution to a standard normal distribution. As mentioned earlier, labels are represented using one-hot vectors. Although NoiseFlow can still transform these vectors to fit a normal distribution, the resulting transformed vectors would cluster around a few positions, leaving many gaps in the fitted distribution. This can lead to challenges when estimating labels based on a noise vector that is randomly drawn from the normal distribution. While adding a small amount of random noise to each one-hot vector may improve the performance of NoiseFlow, a more effective approach to addressing the problem is mixup interpolation (Zhang et al, 2018). The method is beneficial because it not only relaxes the discrete label distribution but also helps to mitigate the impact of noisy labels (Li et al, 2020a). Specifically, mixup interpolation linearly interpolates a pair of samples and their corresponding labels to generate virtual examples for the network to learn from. Let $(\mathbf{p}_i, \mathbf{q}_i)$ and $(\mathbf{p}_j, \mathbf{q}_j)$ be two arbitrary data pairs, where $\mathbf{p}$ and $\mathbf{q}$ represent the input and corresponding label (i.e., a one-hot vector), and α be a random interpolation coefficient that is sampled from a Beta distribution. The virtual example $(\mathbf{p}_v, \mathbf{q}_v)$ can be represented as:

$$\mathbf{p}_v = \alpha\mathbf{p}_i + (1-\alpha)\mathbf{p}_j, \quad \mathbf{q}_v = \alpha\mathbf{q}_i + (1-\alpha)\mathbf{q}_j. \tag{7}$$

Contrastive Learning

Contrastive learning (Chen et al, 2020) is a widely used strategy that helps neural networks learn robust representations by exposing them to diverse perspectives that are distinct from the primary task. In practice, the strategy trains a neural network to distinguish whether a pair of images is positive or negative. A positive pair $(\mathbf{p}_{i,1}, \mathbf{p}_{i,2})$ is created by randomly augmenting the same image, and a negative pair is created by selecting two different images $(\mathbf{p}_i, \mathbf{p}_j)$. To integrate contrastive learning with our NoiseFlow model, we include a projection head $\mathbf{h}_\psi$ with parameters ψ to obtain the projected representations $\mathbf{r}_s = \mathbf{h}_\psi(E_\theta(\mathbf{p}_{i,1}))$ and $\mathbf{r}_t = \mathbf{h}_\psi(E_\theta(\mathbf{p}_{i,2}))$. We then minimize the InfoNCE loss (Oord et al, 2018; Chen et al, 2020) for each positive pair $(\mathbf{p}_{i,1}, \mathbf{p}_{i,2})$, which is expressed as:

$$\mathcal{L}_{c\ell}(s,t) = -log\frac{\exp(\text{sim}(\mathbf{r}_\text{s}, \mathbf{r}_\text{t})/\kappa)}{\sum_{b=1}^{2n}\mathbb{I}_{[b\neq s]}\exp(\text{sim}(\mathbf{r}_\text{s}, \mathbf{r}_\text{b})/\kappa)}, \tag{8}$$

where $\mathbb{I}_{[b\neq s]}$ is an indicator function that returns 1 if $b \neq s$ and 0 otherwise, κ is a user-defined temperature, n is the batch size, and $\text{sim}(\mathbf{r}_\text{s}, \mathbf{r}_\text{t})$ measures the cosine similarity of $\mathbf{r}_s$ and $\mathbf{r}_t$. The denominator in this equation is the summation of the cosine similarity scores of all negative pairs.

Uniform Selection of Clean Samples in each Class

We extract clean samples from the noisy dataset to prevent the networks from overfitting to the noisy data. Since different classes within a dataset may present varying levels of difficulty, we avoid overlooking challenging yet accurate examples by assuming that incorrect labels are evenly distributed across all classes (Karim et al, 2022). Consequently, an equal number of clean samples is selected from each class for further training. To achieve this, we utilize the Jensen-Shannon divergence to measure the disagreement between the predicted labels and the actual ground-truth labels. A low level of disagreement between the labels indicates that they are more likely to be correct, while a high level of disagreement suggests the opposite. In our approach, we establish a divergence threshold for each class to differentiate between clean and noisy samples. For more details, please refer to the appendix section and the study in (Karim et al, 2022).

Semi-Supervised Learning

While noisy samples are unreliable, they can still be leveraged as unlabeled data during network training (Sohn et al, 2020; Li et al, 2020a). To achieve this, we estimate the pseudo-label of each noisy sample using previous versions of NoiseFlow, followed by sharpening the label probability (Sohn et al, 2020; Karim et al, 2022). Specifically, each label probability $\hat{\mathbf{q}}_i$ undergoes the following equation:

$$\mathbf{q}'_i = \frac{\hat{\mathbf{q}}_i^{1/T}}{\sum_{j=1}^{L}\hat{\mathbf{q}}_j^{1/T}}, \tag{9}$$

where T is a hyperparameter that controls the degree of sharpness, and L is the number of classes. The sample and its corresponding pseudo-label are then utilized to train the network (Equation 5). However, since the network estimates the pseudo-labels and employs them to train itself, this approach can lead to the risk of the network consistently producing the same label, regardless of the input (Caron et al, 2021). To prevent the problem, we adopt the centering strategy to refine the pseudo-labels, which is based on the assumption that clean labels in a random batch should be balanced. Specifically, we subtract each label probability by the mean probability in a batch:

$$\mathbf{q}'' = \mathbf{q}' - \bar{\mathbf{q}}, \qquad \text{where} \quad \bar{\mathbf{q}} \leftarrow \mathrm{m}\bar{\mathbf{q}} + (1-\mathrm{m})\frac{1}{\mathrm{B}}\sum_{\mathrm{i}=1}^{\mathrm{B}} \mathbf{q}', \tag{10}$$

is iteratively updated, m is a hyperparameter and B is the batch size. The probability of each label in $\mathbf{q}''$ is then added by $1/n$, where n is the number of classes, and then normalized to maintain the sum of probabilities equal to 1.

During training, we also refine the labels of clean samples as data separation may not be perfect. We use linear interpolation to reduce the memorization effect when there is a mismatch between the given and estimated labels. The interpolated label is then sharpened and normalized.

4.3 Network architectur

The NoiseFlow architecture consists of an encoder and an NF block. The encoder can be any type of backbone that transforms an input into a compact representation. Meanwhile, the NF block is constructed using four concatsquash layers (Grathwohl et al, 2019; Yang et al, 2019; Abdal et al, 2021) and two batch normalizing layers (Yang et al, 2019), as depicted on the right-hand side of Figure 1. The concatsquash layer is defined as:

$$\begin{gathered} CCS(t, \mathbf{c}, \mathbf{u}) = \sigma_1((W_u \mathbf{u} + b_u) \times gate + bias), \\ \text{where} \quad \mathrm{gate} = \sigma_2(\mathrm{W_{tt}t} + \mathrm{W_{tc}}\mathbf{c} + \mathrm{b_t}), \quad \mathrm{bias} = (\mathrm{W_{bt}t} + \mathrm{W_{bc}}\mathbf{c} + \mathrm{b_bt}), \end{gathered} \tag{11}$$

W_u, W_{tt}, W_{tc}, W_{bt}, W_{bc}, b_u, b_t, and b_b are learnable parameters, and σ_1 and σ_2 denote the *tanh* and *sigmoid* activation functions, respectively.

5 Evaluations and Experiments

To evaluate the effectiveness of NoiseFlow, we compared it with several baseline methods. Detailed information about the experiment settings, training procedures, and results are provided below. We also provide pseudo-codes, training parameters, and all implementation details in the appendix for readers to replicate our experiment results.

	Cifar-10								Cifar-100							
Method		Symmetry				Asymmetry				Symmetry				Asymmetry		
	0%	20%	50%	80%	90%	10%	30%	40%	0%	20%	50%	80%	90%	10%	30%	40%
CE	93.9	86.8	79.4	62.9	42.7	88.8	81.7	76.1	74.3	62.0	46.7	19.9	10.1	68.1	53.3	44.5
LDMI	-	88.3	81.2	43.7	36.9	91.1	91.2	84.0	-	58.8	51.8	27.9	13.7	68.1	54.1	46.2
M-Up	96.0	95.6	87.1	71.6	52.2	93.3	83.3	77.7	77.9	67.8	57.3	30.8	14.6	72.4	57.6	48.1
PCIL	93.9	92.4	89.1	77.5	58.9	94.2	92.5	90.7	77.8	69.4	57.5	31.1	15.3	72.0	68.1	59.5
JPL	-	93.5	90.2	35.7	23.4	93.1	92.9	91.6	-	70.9	67.7	17.8	12.8	76.0	59.3	48.3
MOIT	95.7	94.1	91.1	75.8	70.1	93.8	92.5	91.7	77.1	75.9	70.1	51.4	24.5	71.6	69.5	55.1
DMix	94.3	96.1	94.6	92.9	76.0	95.4	94.7	93.0	67.4	77.3	74.6	60.2	31.5	77.3	74.6	73.2
ELR	95.5	95.8	94.8	93.3	78.7	94.2	94.1	93.2	78.0	77.6	73.6	60.8	33.4	77.4	75.1	74.0
AUG	-	96.3	95.4	93.8	91.9	-	-	94.6	-	79.5	77.2	66.4	41.2	-	-	-
ILL	-	**96.8**	**96.6**	94.3	-	-	-	**94.8**	-	77.5	75.5	66.5	-	-	-	**75.8**
UNICON	93.4	96.0	95.6	93.9	90.8	95.3	94.8	94.1	77.5	78.9	77.6	65.2	44.8	78.2	75.6	74.8
NoiseFlow	**96.3**	**96.8**	96.1	**95.1**	**92.9**	**96.7**	**95.1**	93.8	**79.1**	**79.1**	**77.9**	**72.9**	**60.4**	**79.2**	**77.9**	75.5

Table 1: The comparison of NoiseFlow with the baseline methods, LDMI (Xu et al, 2019), M-Up (Zhang et al, 2018), PCIL (Yi and Wu, 2019), JPL (Kim et al, 2021), MOIT (Ortego et al, 2021), DMix (Li et al, 2020a), ELR (Liu et al, 2020), AUG (Nishi et al, 2021), ILL (Chen et al, 2023), and UNICON (Karim et al, 2022), on the Cifar-10 and Cifar-100 datasets. We consider both symmetric and asymmetric label noise and various noise ratios. For the baseline methods, we obtained the results from UNICON and MOIT papers. The numbers showed that our NoiseFlow model outperformed the methods in most experimental conditions.

5.1 Experiment Settings and Results

Evaluation on Synthetic datasets

We compared NoiseFlow with baseline methods on synthetic datasets because it allowed us to precisely control the noise ratios. The backbone architecture used in the experiments was ResNet18. Specifically, we randomly reassigned the labels in Cifar-10, Cifar-100, and TinyImageNet to generate the synthetic datasets. Note that the labels in the testing set were not modified during evaluation. We followed the experimental settings in prior studies (Li et al, 2020a; Liu et al, 2020; Karim et al, 2022) and applied both symmetric and asymmetric noise to the datasets. In the symmetric setting, we randomly selected a subset of the images and reassigned their labels to other classes in an evenly distributed manner. In the asymmetric setting, we aimed to simulate structural errors by reassigning the sample to the next class. For instance, an airplane would be reassigned to an automobile, and an automobile would be reassigned to a bird, and so on.

Initially, we trained the network using all training samples for the warmup phase. The warmup process took 10, 30, and 15 epochs on Cifar-10, Cifar-100, and TinyImageNet, respectively. Following the warmup phase, we trained the networks for 350 epochs on Cifar-10 and Cifar-100, and 200 epochs on TinyImageNet. We reported the highest accuracy values on the testing set. For TinyImageNet, we additionally calculated the average results of the last 10 epochs to determine whether the methods overfit the noise.

Table 1 presents the experimental results for Cifar-10 and Cifar-100, showing that NoiseFlow performed better than the baseline methods in most settings, with the exception of Cifar-10 with a 40% asymmetric noise ratio. Remarkably, NoiseFlow was particularly robust to symmetric noise, achieving an accuracy of 60.4% on Cifar-100

	0 %		20 %		50 %	
Method	Best	Avg.	Best	Avg.	Best	Avg.
CE	57.4	56.7	35.8	35.6	19.8	19.6
Decoupling (Malach and Shalev-Shwartz, 2017)	-	-	37.0	36.3	22.8	22.6
F-correction (Patrini et al, 2017)	-	-	44.5	44.4	33.1	32.8
MentorNet (Jiang et al, 2018)	-	-	45.7	45.5	35.8	35.5
Co-teaching+ (Yu et al, 2019)	52.4	52.1	48.2	47.7	41.8	41.2
M-correction (Arazo et al, 2019)	57.7	57.2	57.2	56.6	51.6	51.3
NCT (Sarfraz et al, 2021)	62.4	61.5	58.0	57.2	47.8	47.4
UNICON (Karim et al, 2022)	63.1	62.7	59.2	58.4	52.7	52.4
NoiseFlow	**63.9**	**63.7**	**63.3**	**63.0**	**59.2**	**58.9**

Method		Clothing	WebVision
NoiseFlow	Best	**67.45**	**78.44**
	Last	**67.45**	77.68
UniCON	Best	61.01	77.60
	Last	60.44	-

Table 2: (Top) NoiseFlow outperformed baseline methods on TinyImage under simulated symmetric noise ratios (i.e., 0%, 20%, and 50%). Both the highest accuracy and the average accuracy of the last 10 epochs on the testing set are reported. (Down) We compared NoiseFlow with the state-of-the-art, UNICON (Karim et al, 2022), on real-world datasets.

Arch.	Clean		sym 80%	
	Best	Last	Best	Last
UNICON	77.49	75.61	65.19	64.33
UNICON+	77.08	75.16	65.48	64.61
NoiseFlow	**79.09**	**79.06**	**72.91**	**72.74**

Table 3: Since a CNF block is typically larger than a classification layer, we compared our NoiseFlow to an extended version of UNICON, called UNICON+, by adding an extra non-linear projection. Our evaluations were carried out on the Cifar-100 dataset under conditions of 0% and 80% symmetric noise rates. As can be seen, UNICON and UNICON+ performed similarly and revealed that a larger network does not inherently guarantee superior results.

with a 90% noise ratio, which is a significant improvement over the current state-of-the-art method that achieved only 44.8% accuracy. Table 2 left displays the results for TinyImageNet, which also demonstrates the effectiveness of NoiseFlow, as it outperformed baseline methods by a substantial margin on the dataset with zero, small, and medium noise ratios. For instance, under the 50% noise ratio, NoiseFlow increased the best accuracy from 52.7% to 59.2% and the average accuracy of the last 10 epochs from 52.4% to 58.9%. It's worth noting that NoiseFlow and most of the baseline methods serve as a superior classifier in general, as evidenced by the results on Cifar-10, Cifar-100, and TinyImageNet with 0% noise. This outcome is expected since the labels of these datasets are manually annotated and are not entirely noise-free. We refer readers to the work of (Northcutt et al, 2021) for further discussions on label errors.

	sym 20%	sym 90%	asym 40%
NoiseFlow	**79.13**	**60.37**	**75.50**
w/o contrastive	78.95	58.51	73.95
w/o centering	77.44	36.99	72.27
w/o sharpening	78.55	42.82	75.10
w/o Mixup	76.43	47.04	70.90

Table 4: We conducted ablation experiments to compare the performance of NoiseFlow without contrastive learning, centering, sharpening, and mixup.

Evaluation on real-world datasets

We compared NoiseFlow and UNICON (Karim et al, 2022) on the WebVision and Clothing1M datasets. Results from UNICON on the WebVision dataset were directly obtained from their original paper. For the Clothing1M dataset, we replicated the experiment by initializing both networks with random parameters and training them for 200 epochs, as UNICON's results relied on a pre-trained network. We set UNICON's hyperparameters based on their official Github page's recommendations. The best and final test set accuracies are documented in Table 2 right, which clearly highlight NoiseFlow's superior performance on both datasets.

Discussions on Network Size and Effeciency

Studies (Zagoruyko and Komodakis, 2016; He et al, 2016) have demonstrated that increasing the depth and width of the network can enhance learning performance. While a CNF block is larger than a classification layer, which contains 391457 and 51200 parameters, respectively, when classifying the Cifar-100 dataset, we added an extra layer to ResNet18 (i.e., the backbone of UNICON) and compared its performance with NoiseFlow. Specifically, we utilized two fully-connected layers in UNICON+ (i.e., 512×512 and 512×100) to map the extracted representations to their corresponding classes. In the comparison, the encoders of UNICON, UNICON+, and NoiseFlow were the same, and all strategies for noisy label training were applied. The results in Table 3 indicated that UNICON and UNICON+ performed similarly in the datasets with zero and high noise ratios. The results suggest that the reason why NoiseFlow outperformed UNICON is due to the modeling of the entire label distribution rather than a large network.

In addition to network size, the use of CNF block leads to higher computational costs. Since we replaced the fully connected layer in ResNet18 with a CNF block, we mainly compared the training and inference time of NoiseFlow and UNICON. When training networks for 250 epochs on the Cifar10 dataset using a RTX 4090 GPU, NoiseFlow took 13 hours, while UNICON required 8 hours. Both methods consume approximately 10GB of memory during training. The training duration of NoiseFlow is longer because the CNF block is optimized using numerical integration (i.e., adjoint method (Pontryagin et al, 1962)). During inference, NoiseFlow takes approximately 0.03 seconds on average to classify an image. Although NoiseFlow's runtime performance is slower than UNICON, which solves an ODE in the CNF block, it can be used to correct labels in a dataset for the second pass training. In this pass, the CNF block can be changed back to the traditional fully connected layers.

Ablation study

We conducted an ablation study focused on training strategies using the CIFAR-100 dataset. The findings, as shown in Table 4, suggest that every strategy offers benefits, especially when dealing with datasets subjected to high noise levels. Notably, in the absence of centering and sharpening, the performance drop was evident under conditions of 90% symmetrical noise.

Pseudo-label determination

While NoiseFlow can generate different label probabilities for an input by incorporating noise vectors, in practice, we use only the mean of the normal distribution to estimate the label as the mean has the highest density. We have experimented with averaging the label probabilities generated using noise vectors sampled from the normal distribution $\mathcal{N}(\mathbf{0}, 0.2)$. However, this approach resulted in slightly worse performance, and the performance deteriorated further as the variance increased.

5.2 Limitations

While NoiseFlow can mitigate the impact of noisy labels, our experiments indicate it is less robust against asymmetric noise. This could be due to the concentrated nature of such noise. We aim to address this challenge in future research. Additionally, even though NoiseFlow exhibits superior performance, it is larger than the baseline models due to our replacement of the fully connected layer with an NF block. Although experiments presented in Table 3 show that NoiseFlow's improved performance is not due to increased parameters, it does require more computational power.

6 Conclusion

We have developed a framework to tackle the issue of noisy samples when training neural networks. Our approach involves modeling the complete label distribution in the dataset, which includes both clean and noisy samples, conditioned on the inputs. By utilizing noise vectors to generate different labels for a specific sample, the network avoids the need to separate the representations of similar data or define highly nonlinear boundaries for classifying data with incorrect labels. This results in improved generalization performance. Furthermore, our framework can be combined with existing methods such as clean label separation, sample interpolation, contrastive learning, and semi-supervised learning to further enhance the network's performance. The experimental results verify the effectiveness of our method.

Acknowledgements. We thank the reviewers for their constructive comments. This study is based upon work partially supported by the National Science and Technology Council (NSTC), Taiwan, under Contract No. 111-2221-E-A49 -129 -MY3, Contract No. 113-2425-H-A49 -001 - and Contract No. 113-2221-E-A49 -149 -MY3.

Appendix A

A.1 Training Details

We simultaneously train two NoiseFlow models to avoid the selection bias issue when separating clean samples. The training process is outlined in Figure 3 of the main manuscript and Algorithm 1. Initially, we warm up the two models by training them using the NLL loss (Equation 5, Lines 5-7). Subsequently, the two models separate the dataset into clean/labeled and noisy/unlabeled sets, utilizing the uniform selection strategy (Karim et al, 2022). For the clean samples, we refine their labels by considering the given and estimated labels (Lines 14-16). For the noisy samples, we determine their labels using the outcomes generated by the two NoiseFlow models (Lines 17-19). As one-hot label vectors collapse at only a few points, we employ mixup interpolation to relax the distribution fitting (Line 20). Finally, we adopt contrastive learning and integrate the NLL and InfoNCE losses to iteratively update the two models (Lines 21-22).

Algorithm 1 Pseudo Codes for Training NoiseFlow

```
Require: $\mathcal{D}$: dataset
 1: $E_1$, $F_1$, $H_1$: encoder, flow, and projector of network 1
 2: $E_2$, $F_2$, $H_2$: encoder, flow, and projector of network 2
 3: $\lambda$: weight of the contrastive loss
Ensure:
 4: for epoch i=1 to m do
 5:     SGD-update($\mathcal{L}_{n\ell\ell}$, ($E_1$, $F_1$, $E_1$, $F_2$))          ▷ Warm up
 6: end for
 7: for epoch i=1 to n do
 8:     for j=1 to 2 do
 9:         $\mathcal{D}_{labeled}$, $\mathcal{D}_{unlabeled}$
10:               = Select($\mathcal{D}$, ($E_1$, $F_1$), ($E_2$, $F_2$))
11:         for ($\mathbf{P}_l$, $\mathbf{Q}_l$) $\sim \mathcal{D}_{labeled}$, ($\mathbf{P}_u$, $\mathbf{Q}_u$) $\sim \mathcal{D}_{unlabeled}$
12:         do
13:             $\hat{\mathbf{Q}}_{rf}$ = NF($E_j$, $F_j$, $\mathbf{P}_l$)          ▷ Eq. 6
14:             $\hat{\mathbf{Q}}_{rf} = w \times \mathbf{Q}_l + (1-w) \times \hat{\mathbf{Q}}_{rf}$
15:             $\hat{\mathbf{Q}}'_{rf}$ = Sharp($\hat{\mathbf{Q}}_{rf}$)          ▷ Eq. 9
16:             $\hat{\mathbf{Q}}_{pseu}$ = NF($E_1$, $F_1$, $E_2$, $F_2$, $\mathbf{P}_u$)          ▷ Eq. 6
17:             $\mathbf{Q}'_{pseu}$ = Sharp($\hat{\mathbf{Q}}_{pseu}$)          ▷ Eq. 9
18:             $\mathbf{Q}''_{pseu}$ = $\mathbf{Q}'_{pseu}$ - $\bar{\mathbf{Q}}'_{pseu}$          ▷ Eq. 10
19:             $\mathbf{P}_{mix}$, $\mathbf{Q}_{mix}$ = Mix($\mathbf{P}_l$, $\mathbf{Q}'_{rf}$, $\mathbf{P}_u$, $\mathbf{Q}''_{pseu}$)
20:             $\mathcal{L} = \mathcal{L}_{cl}(\mathbf{P}_l) + \lambda\, \mathcal{L}_{n\ell\ell}(E_j, F_j, \mathbf{P}_{mix}, \mathbf{Q}_{mix})$
21:             SGD-update($\mathcal{L}$, ($E_j$, $F_j$, $H_j$))
22:         end for
23:     end for
24: end for
```

A.2 Parameters

The hyperparameters used for NoiseFlow in the experiments are presented in Table A1. As can be seen, most of the parameters are consistent across the datasets, which demonstrates the widespread applicability of our NoiseFlow model.

	Cifar-10	Cifar-100	TinyImageNet
# Classes	10	100	200
# samples	50,000	50,000	100,000
Flow modules	8-8-8-8	64-64-64-64	160-160-160-160
# Warm Up Epochs	10	10	15
# Training Epochs	350	350	200
Learning Rate (E_θ)	2E-02	2E-02	4E-02
Learning Rate (F_η)	4E-05	5E-05	4E-05
Batch Size	256	256	256
Weight Decay	5E-04	5E-04	5E-04
Dimension of $E_\theta(\mathbf{p})$	512	512	512
α, β (Beta Dist, Eq. 6)	4, 4	4, 4	0.5, 0.5
κ (Eq. 7)	0.025	0.025	0.025
T (Eq. 9)	0.5	0.5	0.2
m (Eq. 10)	0.8	0.8	0.8

Table A1: This table displays the hyperparameters employed in the experimental setups. The majority of these parameters remain consistent across the datasets, demonstrating the broad adaptability of our NoiseFlow model.

A.3 Single vs. Twin Networks

To prevent bias in the separation of clean samples, all noisy label training methods employ a dual network training strategy. In this approach, one network selects clean samples for the other network to learn in the next iteration. NoiseFlow, which is capable of generating multiple labels for a single input, prompted us to explore its adaptability and sustained performance amidst noise exposure. In contrast to the dual network setup, we experimented with a variant called *Single*, which uses only one network. This network estimates the label probability of an input using noise vectors from a standard normal distribution $\mathcal{N}(\mathbf{0}, 0.2)$.

Our experiments (Table A2) demonstrated that *Single* performed effectively when the noise ratio was low, but its performance deteriorated as the ratio increased, particularly on the Cifar-100 dataset. We noticed that the network consistently predicted the same label in the later stages of training since it relied on the generated pseudo-labels. Consequently, the mixup interpolation failed to alleviate the sharp label distribution, and the NFs struggled to map an identical label vector to fit the normal distribution. The significant gradient prevented the minimization of NLL loss.

To address this problem, we trained a second version of NoiseFlow called *EMA*. *EMA* had a teacher and a student network, and the teacher network was updated by the exponential moving average of the student network's parameters. Moreover, the

Method	Symmetry								Asymmetry					
	Cifar-10				**Cifar-100**				**Cifar-10**			**Cifar-100**		
	20%	50%	80%	90%	20%	50%	80%	90%	10%	30%	40%	10%	30%	40%
Single	96.0	95.1	93.9	91.0	71.1	UNF	UNF	UNF	95.8	95.0	92.9	76.5	75.2	71.3
EMA	96.3	95.7	94.6	91.6	77.1	75	67.5	51.3	96.5	95.0	93.3	77.3	75.9	70.5
NoiseFlow	**96.8**	**96.1**	**95.1**	**92.9**	**79.1**	**77.9**	**72.9**	**60.4**	**96.7**	**95.1**	**93.8**	**79.2**	**77.9**	**75.5**

Table A2: NoiseFlow comprises two networks that can identify clean samples, allowing the peer network to learn more accurately. To assess if generating different labels can avoid selection bias, we analyzed two versions of NoiseFlow: *Single* and *EMA*. The *Single* employs only one network, while the *EMA* adopts a teacher-student framework, wherein the teacher network is updated by iteratively averaging the student network's parameters. In this table, UNF is the abbreviation of underflow, which appears when NFs fail to minimize the NLL loss.

teacher network selected clean samples and assigned pseudo-labels for the student network to learn. The results presented in Table A2 demonstrated that *EMA* improved the performance of NoiseFlow and prevented the label collapse issue. However, the results also suggested that training two networks simultaneously is the most effective approach since personal bias in sample selection cannot correct errors when the network does not learn effectively (Han et al, 2018). This phenomenon is particularly noticeable when the dataset contains a high level of noise.

References

Abdal R, Zhu P, Mitra NJ, et al (2021) Styleflow: Attribute-conditioned exploration of stylegan-generated images using conditional continuous normalizing flows. ACM Transactions on Graphics 40(3):1–21

Arazo E, Ortego D, Albert P, et al (2019) Unsupervised label noise modeling and loss correction. In: International conference on machine learning, PMLR, pp 312–321

Caron M, Touvron H, Misra I, et al (2021) Emerging properties in self-supervised vision transformers. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 9650–9660

Chen H, Shah A, Wang J, et al (2023) Imprecise label learning: A unified framework for learning with various imprecise label configurations. arXiv preprint arXiv:230512715

Chen RT, Rubanova Y, Bettencourt J, et al (2018) Neural ordinary differential equations. Advances in Neural Information Processing Systems 31

Chen T, Kornblith S, Norouzi M, et al (2020) A simple framework for contrastive learning of visual representations. In: International conference on machine learning, pp 1597–1607

Ding Y, Wang L, Fan D, et al (2018) A semi-supervised two-stage approach to learning from noisy labels. In: 2018 IEEE Winter conference on applications of computer vision (WACV), IEEE, pp 1215–1224

Dinh L, Krueger D, Bengio Y (2014) Nice: Non-linear independent components estimation. arXiv preprint arXiv:14108516

Dinh L, Sohl-Dickstein J, Bengio S (2017) Density estimation using real nvp. International Conference on Learning Representations

Goldberger J, Ben-Reuven E (2017) Training deep neural-networks using a noise adaptation layer. In: International conference on learning representations

Goodfellow I, Pouget-Abadie J, Mirza M, et al (2014) Generative adversarial nets. Advances in neural information processing systems 27

Grathwohl W, Chen RT, Bettencourt J, et al (2019) Ffjord: Free-form continuous dynamics for scalable reversible generative models. International Conference on Learning Representations

Han B, Yao Q, Yu X, et al (2018) Co-teaching: Robust training of deep neural networks with extremely noisy labels. Advances in neural information processing systems 31

He K, Zhang X, Ren S, et al (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778

Hendrycks D, Mazeika M, Wilson D, et al (2018) Using trusted data to train deep networks on labels corrupted by severe noise. Advances in neural information processing systems 31

Jiang L, Zhou Z, Leung T, et al (2018) Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In: International conference on machine learning, PMLR, pp 2304–2313

Karim N, Rizve MN, Rahnavard N, et al (2022) Unicon: Combating label noise through uniform selection and contrastive learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 9676–9686

Kim Y, Yun J, Shon H, et al (2021) Joint negative and positive learning for noisy labels. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 9442–9451

Kingma DP, Dhariwal P (2018) Glow: Generative flow with invertible 1x1 convolutions. Advances in Neural Information Processing Systems 31

Li J, Socher R, Hoi SC (2020a) Dividemix: Learning with noisy labels as semi-supervised learning. International Conference on Learning Representations

Li M, Soltanolkotabi M, Oymak S (2020b) Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks. In: International conference on artificial intelligence and statistics, PMLR, pp 4313–4324

Liu S, Niles-Weed J, Razavian N, et al (2020) Early-learning regularization prevents memorization of noisy labels. Advances in neural information processing systems 33:20331–20342

Ma X, Wang Y, Houle ME, et al (2018) Dimensionality-driven learning with noisy labels. In: International Conference on Machine Learning, PMLR, pp 3355–3364

Malach E, Shalev-Shwartz S (2017) Decoupling" when to update" from" how to update". Advances in neural information processing systems 30

Nguyen DT, Mummadi CK, Ngo TPN, et al (2020) Self: Learning to filter noisy labels with self-ensembling. International conference on learning representations

Nishi K, Ding Y, Rich A, et al (2021) Augmentation strategies for learning with noisy labels. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 8022–8031

Northcutt CG, Wu T, Chuang IL (2017) Learning with confident examples: Rank pruning for robust classification with noisy labels. In Conference on Uncertainty in Artificial Intelligence

Northcutt CG, Athalye A, Mueller J (2021) Pervasive label errors in test sets destabilize machine learning benchmarks. arXiv preprint arXiv:210314749

Oord Avd, Li Y, Vinyals O (2018) Representation learning with contrastive predictive coding. arXiv preprint arXiv:180703748

Ortego D, Arazo E, Albert P, et al (2021) Multi-objective interpolation training for robustness to label noise. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 6606–6615

Patrini G, Rozza A, Krishna Menon A, et al (2017) Making deep neural networks robust to label noise: A loss correction approach. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1944–1952

Pontryagin LS, Mishchenko E, Boltyanskii V, et al (1962) The mathematical theory of optimal processes

Sarfraz F, Arani E, Zonooz B (2021) Noisy concurrent training for efficient learning under label noise. In: Proceedings of the IEEE/CVF Winter Conference on applications of computer vision, pp 3159–3168

Sohn K, Berthelot D, Carlini N, et al (2020) Fixmatch: Simplifying semi-supervised learning with consistency and confidence. Advances in neural information processing systems 33:596–608

Song H, Kim M, Park D, et al (2022) Learning from noisy labels with deep neural networks: A survey. IEEE Transactions on Neural Networks and Learning Systems

Tanaka D, Ikami D, Yamasaki T, et al (2018) Joint optimization framework for learning with noisy labels. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 5552–5560

Wei H, Feng L, Chen X, et al (2020) Combating noisy labels by agreement: A joint training method with co-regularization. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 13726–13735

Winkler C, Worrall D, Hoogeboom E, et al (2019) Learning likelihoods with conditional normalizing flows. arXiv preprint arXiv:191200042

Xu Y, Cao P, Kong Y, et al (2019) L_dmi: A novel information-theoretic loss function for training deep nets robust to label noise. Advances in neural information processing systems 32

Yang G, Huang X, Hao Z, et al (2019) Pointflow: 3d point cloud generation with continuous normalizing flows. In: IEEE/CVF International Conference on Computer Vision, pp 4541–4550

Yao Y, Sun Z, Zhang C, et al (2021) Jo-src: A contrastive approach for combating noisy labels. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 5192–5201

Yi K, Wu J (2019) Probabilistic end-to-end noise correction for learning with noisy labels. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 7017–7025

Yu X, Han B, Yao J, et al (2019) How does disagreement help generalization against label corruption? In: International Conference on Machine Learning, PMLR, pp 7164–7173

Zagoruyko S, Komodakis N (2016) Wide residual networks. arXiv preprint arXiv:160507146

Zhang H, Cisse M, Dauphin YN, et al (2018) mixup: Beyond empirical risk minimization. International conference on learning representations