Towards a Robust Differentiable Architecture Search under Label Noise

Christian Simon[†,§],   Piotr Koniusz[§,†],   Lars Petersson[§,†],   Yan Han[†,§],   Mehrtash Harandi[♣,§]

[†]The Australian National University   [♣]Monash University   [§]Data61-CSIRO

firstname.lastname@{`anu.edu.au,monash.edu,data61.csiro.au`}

Abstract

Neural Architecture Search (NAS) is the game changer in designing robust neural architectures. Architectures designed by NAS outperform or compete with the best manual network designs in terms of accuracy, size, memory footprint and FLOPs. That said, previous studies focus on developing NAS algorithms for clean high quality data, a restrictive and somewhat unrealistic assumption. In this paper, focusing on the differentiable NAS algorithms, we show that vanilla NAS algorithms suffer from a performance loss if class labels are noisy. To combat this issue, we make use of the principle of information bottleneck as a regularizer. This leads us to develop a noise injecting operation that is included during the learning process, preventing the network from learning from noisy samples. Our empirical evaluations show that the noise injecting operation does not degrade the performance of the NAS algorithm if the data is indeed clean. In contrast, if the data is noisy, the architecture learned by our algorithm comfortably outperforms algorithms specifically equipped with sophisticated mechanisms to learn in the presence of label noise. In contrast to many algorithms designed to work in the presence of noisy labels, prior knowledge about the properties of the noise and its characteristics are not required for our algorithm.

1. Introduction

To avoid exhausting engineering, Neural Architecture Search (NAS) has emerged as a leading mechanism for automatic design and wiring of neural networks. NAS has been successfully moving forward with diverse approaches to achieve a robust automatic architecture search *e.g.*, evolution-based NAS [11, 30, 31, 37], optimization-based NAS [10, 23, 24, 34, 43], and Reinforcement Learning (RL) based NAS [29, 62, 63]. Specifically, a gradient-based method with a continuous architecture space called Differentiable ARchiTecture Search (DARTS) has attracted significant attention in NAS because of a reduced cost and complexity of searching for high performance architectures.

In this paper, we go beyond merely learning with NAS under regular assumptions of clean labels.

Supervised learning with neural networks often leads to a performance degradation due to overfitting, especially in the presence of label noise which often emerges due to data corruption and/or human annotation errors especially prominent in large scale datasets. As a result, neural networks fail to generalize well to previously unseen data and achieve suboptimal classification results. Given the importance of such problems, existing state-of-the-art methods are specifically designed to deal with the data noise by correcting labels [42], employing dedicated loss functions [5, 26, 46, 58], reweighting samples [17, 33], selecting samples [15], and modeling a transition matrix [28, 40, 48]. However, designing robust neural networks that mitigate overfitting due to label noise is still unexplored. To this end, we propose a structural approach, that is a method that requires no explicit changes to neither the loss functions nor the input samples nor the final outputs (predictions) but the neural network structure. Moreover, our approach does not require specific assumptions on the amount or the type of label noise.

Our primary focus[1] is to investigate and advance a robust method to search an architecture in the presence of label noise. To this end, we aim to answer the following questions:

- **Motivational Curiosity.** As handcrafted neural networks (*e.g.*, Inception [38]) overfit to noisy labels [55] (even the pretext labeling in self-supervised UnNAS is imperfect), is the test performance of neural networks constructed by NAS also degraded by the label noise?

- **Research Curiosity.** Can we design an operator[2] that is robust to noisy labels and helps existing NAS methods (*e.g.*, DARTS [23]) to perform well in the presence of label noise?

The answer to the first question is affirmative as shown in Fig. 1. Firstly, we highlight that the performance of vanilla

---

[1]Note that training robust NAS under the label noise is not the same problem as using static network for classification under the label noise.

[2]An operator is an operation connecting two nodes in NAS *e.g.*, the standard NAS uses conv., max- or average-pooling, skip connections, *etc*.
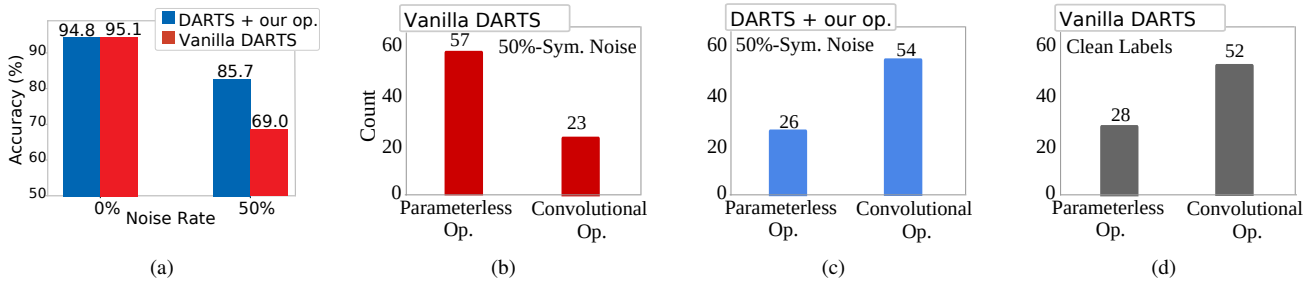
Figure 1: (a) Testing accuracy of vanilla *vs.* our approach (CIFAR-10, clean labels *vs.* 50%-symmetric label noise). The histogram of found operators (five runs) of (b) the normal cell of vanilla DARTS under 50%-symmetric label noise, (c) the normal cell of DARTS with nConv (our noise injecting operator) searched on CIFAR-10 (50%-symmetric noisy labels), and (d) the normal cell of vanilla DARTS with clean labels. The normal cell of vanilla DARTS under label noise is constructed poorly due to the large number of parameterless operations. It is apparent the network thus looses the learning capacity in an attempt to prevent overfitting to the noise by selecting parameterless operators. In contrast, DARTS with our proposed operator mitigates such a poor cell design as highlighted by the larger number of convolutional operators being selected in place of a parameterless operator.

DARTS [23] suffers when subjected to noisy labels as shown in Fig. 1 (a). Furthermore, the architecture searched by vanilla DARTS under label noise results in a bad architecture as shown in Fig. 1 (b) while our approach shown in Fig. 1 (c) produces a better designed cell. To answer the second question, we analyze the task of learning deep neural networks under noisy labels using Information Bottleneck. As a result of this analysis, we introduce a variant of the convolutional operation by injecting noise into the pipeline. Upon learning the parameters of the noise from data, we will empirically show that robustness against label corruption can be attained. In essence, we will show later that the noisy convolution regularizes and limits the gradient of the noisy samples during training. In short, our key contributions are:

i. We show that the search through noisy labels degrades the classification performance of standard NAS.

ii. We provide an information theoretic framework to tackle learning noisy labels. Our proposed noise injection operator performs implicit regularization during learning preventing overfitting to noisy labels.

iii. The proposed operator is included into the NAS search space with the goal of preventing overfitting during training with noisy labels. A noise injection on the input of the operator turns activations of hidden units into the so-called stochastic activations.

iv. We show experimentally that architectures emerging from the NAS search with our proposed noise injecting operator outperform under fewer parameters the state of the art, especially in the case of no prior knowledge given *e.g.*, the lack of the noise type/its rates.

2. Related work

Neural architecture search. NAS is the process of automating the design of a neural network architecture. NAS has been successfully applied to various recognition and image generation problems [9, 12, 62]. NAS algorithms can be computationally demanding *e.g.*, when based on evolutionary algorithms [11, 30, 31, 37] or reinforcement learning [6, 62], which provide flexible schemes to sample and evaluate architectures from a vast pool of architectures (so-called search space). However, searching directly on a very large dataset with a large neural network is computationally expensive. Zoph *et al.* [63] propose a scalable solution by searching a cell structure instead of the entire neural network structure. Due to the above issues, our method follows the formulation of differentiable NAS [23, 53] which is computationally feasible on a broad range of tasks [20, 21, 50] and results in a cell transferrable to other tasks.

Noisy labels. Label noise can harm the performance of neural networks if training is carried out as if data is clean. One of the solutions to tackle label noise is to relabel the samples as proposed in [39, 42, 51]. Loss correction approaches [5, 17, 32, 58] achieve the same purpose of addressing label noise by compensating and modifying the loss functions. Several types of noise (*e.g.* symmetric) can even be modeled as a transition matrix [28, 48] that indicates the probability of clean labels being flipped to noisy labels. Other approaches combat noisy labels by ranking [14], re-weighting [33] and selecting [15] samples.

Information bottleneck. Information bottleneck introduced in [41] provides a trade-off formulation between compression of inputs and prediction of outputs. Maximization of the information bottleneck can be approximated by the variational methods [3]. The variational information bottleneck uses so-called noisy computations for adversarial robustness

and it achieves so-called sufficiency, minimality and invariance criteria described in [2]. Furthermore, Schwartz-Ziv and Tishby [36] propose a visualization technique operating in the so-called *information plane* to analyze the information bottleneck principle in deep learning. The theories and analyses from these prior works inspire our investigations of robust NAS learning in the presence of label noise under principles of the information bottleneck.

Training neural networks with noise. The effect of injecting noise into neural networks has been studied in various contexts. Noise injection as a form of regularization was shown to improve the generalization capability of the model [4, 8, 13]. Noh *et al*. [27] propose a noise injection method by applying gradient updates over multiple feed-forwards of (random) noisy samples and the gradient is weighted based on the importance per sample in each iteration. The above methods differ from ours *e.g.*, they act on the gradient to prevent saturation or consider optimized dropout. In contrast, we propose a parametric noise injector which prevents overfitting to the label noise and is inspired by the information bottleneck principle.

3. Robust Differentiable Architecture Search

A NAS algorithm operates in two stages, namely the **search phase** and the **evaluation phase**. In the search phase, NAS searches the space of cell architectures given a set of operations. During the evaluation phase, a neural network is constructed by stacking multiple cells prior to re-training it from scratch to evaluate the obtained architecture. In the noisy setting, NAS faces incorrect (noisy) labels during both the search and the evaluation phases.

To be more specific, let $\mathcal{D}$ be a dataset with M samples $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^M$ where $\boldsymbol{x}_i \in \mathbb{R}^n$ and $\boldsymbol{y}_i \in \{0,1\}^C$. We define the noisy data as $\tilde{\mathcal{D}} = \{(\boldsymbol{x}_i, \tilde{\boldsymbol{y}}_i)\}$ where $\tilde{\boldsymbol{y}}_i \in \{0,1\}^C$ is a noisy label. There are two types of label noise that we consider in this work, namely symmetric and asymmetric noise. Symmetric noise is constructed by swapping clean labels so that labels of each class are contaminated in a chosen equal proportion by other classes. Asymmetric noise is generated by replacing a chosen percentage of labels of one class with labels of another class *e.g.*, cat $\rightarrow$ tiger. The goal of NAS for noisy labels is to find a robust high-performance architecture that copes well with the label noise during training and testing (prevents overfitting to noisy labels).

3.1. Vanilla DARTS

In what follows, we build on the DARTS algorithm [23] which is the driving force behind several recent developments [20, 21, 50] due to its significant reduction in computational load of constructing high-performance architectures compared to non-differentiable NAS. Prior to introducing our key contributions, we briefly outline DARTS below.

To perform the search over architectures, DARTS formulates the search phase as a bilevel optimization problem. To this end, an inner and an outer objective function $\mathcal{L}_{\text{trn}}$ and $\mathcal{L}_{\text{val}}$ are introduced with the goal of finding the architecture parameterization $\boldsymbol{\alpha} \in \mathcal{A}$ that minimizes $\mathcal{L}_{\text{val}}$ and the network weights $\boldsymbol{\theta}$ that minimize $\mathcal{L}_{\text{trn}}$ as follows:

$$
\begin{aligned}
\min_{\boldsymbol{\alpha}} \ &\mathcal{L}_{\text{val}}(\boldsymbol{\theta}^*(\boldsymbol{\alpha}), \boldsymbol{\alpha}) \\
\text{s.t. } &\boldsymbol{\theta}^*(\boldsymbol{\alpha}) = \arg\min_{\boldsymbol{\theta}} \mathcal{L}_{\text{trn}}(\boldsymbol{\theta}, \boldsymbol{\alpha}),
\end{aligned}
\tag{1}
$$

where $\boldsymbol{\theta}^*(\boldsymbol{\alpha})$ represents the weights of the associated architecture that minimize the inner loss. Optimizing both losses is achieved by gradient descent. Unfortunately, optimization in the inner loop is time consuming. As such, DARTS applies a simple approximation with a single unrolling step to optimize the inner loss $\nabla_{\boldsymbol{\alpha}} \mathcal{L}_{\text{val}}(\boldsymbol{\theta}^*(\boldsymbol{\alpha}), \boldsymbol{\alpha}) \approx \nabla_{\boldsymbol{\alpha}} \mathcal{L}_{\text{val}}(\boldsymbol{\theta}', \boldsymbol{\alpha})$ where $\boldsymbol{\theta}' = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{trn}}(\boldsymbol{\theta}, \boldsymbol{\alpha})$. In the bi-level formulation, we encounter two derivatives (related to the network parameters and the architecture parameters, respectively) for which the chain rule is applied together with the implicit function theorem [7, 19, 35] which yields:

$$
\begin{aligned}
\nabla_{\boldsymbol{\alpha}} \mathcal{L}_{\text{val}}(\boldsymbol{\theta}^*(\boldsymbol{\alpha}), \boldsymbol{\alpha}) \approx \ &\nabla_{\boldsymbol{\alpha}} \mathcal{L}_{\text{val}}(\boldsymbol{\theta}', \boldsymbol{\alpha}) \\
&- \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{val}}(\boldsymbol{\theta}', \boldsymbol{\alpha}) \nabla_{\boldsymbol{\theta}, \boldsymbol{\alpha}}^2 \mathcal{L}_{\text{trn}}(\boldsymbol{\theta}, \boldsymbol{\alpha}).
\end{aligned}
\tag{2}
$$

To reduce the training time and computational complexity of the algorithm, one can disregard the higher-order derivatives in Eq. 2. This simplification leads to a speed-accuracy trade-off as observed in several studies (*e.g.*, [23, 53])

Following [23, 30, 63], we make use of two types of cells to design the network, namely **I.** the normal cell and **II.** the reduction cell. Each cell is represented by a DAG and can realize one operation from a complete set of candidate operations. The final architecture of the network is constructed by stacking the obtained cells together. This enables us to make the composition of the cells both transferable between datasets and independent of the network depth. A cell is a Directed Acyclic Graph (DAG) with an ordered sequence of N nodes and is connected to its preceding two cells. Let $\mathcal{O}$ be the set of candidate operators (*e.g.*, skip connection, 3×3 dilated convolution, *etc*.) defined for a cell. Consider a node i whose output represented by $\boldsymbol{z}^{(i)}$ is obtained by:

$$
\boldsymbol{z}^{(i)} = \sum_{j \rightsquigarrow i} o^{(i,j)}\big(\boldsymbol{z}^{(j)}\big),
\tag{3}
$$

where $\rightsquigarrow$ denotes the edge between nodes i and j with a total weight over possible operations given as:

$$
o^{(i,j)}(\boldsymbol{z}^{(j)}) = \sum_{r=1}^{\nu} \frac{\exp\left(\alpha_r^{(i,j)}\right)}{\sum_{s=1}^{\nu} \exp\left(\alpha_s^{(i,j)}\right)} o_r^{(i,j)}(\boldsymbol{z}^{(j)}),
\tag{4}
$$

where $o_r^{(i,j)}$ are individual operations. In essence, choosing the operation that will connect node j to node

i is formulated by a mixing weight vector $\boldsymbol{\alpha}^{(i,j)} = (\alpha_1^{(i,j)}, \alpha_2^{(i,j)}, \cdots, \alpha_\nu^{(i,j)})$ using a softmax function. The task of architecture search is then to learn $\boldsymbol{\alpha} = \{\alpha^{(i,j)}\}$. At the end of the search phase, a discrete architecture is picked by choosing the most likely operation between the nodes. That is, $o^{(i,j)} = o_{r^*}^{(i,j)}$ where $r^* = \arg\max_r \alpha_r^{(i,j)}$.

3.2. Learning in the presence of label noise via the information bottleneck principle

Differentiable NAS uses the cross-entropy loss to train (search and evaluation) networks. Below, we formulate a loss function which prevents the trained model from overfitting to noisy labels. To this end, we bring the information theoretic view and explain how it helps with training under label noise. We begin by defining several terms in our analysis: the input $\boldsymbol{x} \in X$, the output $\boldsymbol{y} \in Y$, the representation of the input $\boldsymbol{z} \in Z$, entropy $H(X) = \mathbb{E}_{p(\boldsymbol{x})}[-\log p(\boldsymbol{x})]$, cross-entropy $H(p,q) = \mathbb{E}_{p(\boldsymbol{x})}[-\log q(\boldsymbol{x})]$, conditional entropy $H(Y|X) = H(X,Y) - H(X)$, mutual information $I(X;Y) = H(Y) - H(Y|X)$, and Kullback-Leibler (KL) divergence $\mathrm{KL}(p(\boldsymbol{x})\|q(\boldsymbol{x})) = \mathbb{E}_{p(\boldsymbol{x})}[\log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}]$.

Considering our NAS setting, we revisit the Information Bottleneck [41] principle which provides a computational framework that controls the trade-off between the compression of input $\boldsymbol{x}$ and the prediction of $\boldsymbol{y}$ according to:

$$\max_{\boldsymbol{\phi}} \ I(Z;Y,\boldsymbol{\phi}) - \beta I(X;Z,\boldsymbol{\phi}), \tag{5}$$

where $\beta \geq 0$ is a hyper-parameter adjusting the level of relevant information captured by Z while $\boldsymbol{\phi}$ denotes network parameters. For training neural networks under the regular setting (clean labels), one maximizes $I(Z;Y,\boldsymbol{\phi})$ to obtain a model with a high performance. However, increasing the value of $I(Z;Y,\boldsymbol{\phi})$ alone makes the model prone to overfitting [1] *ie.*, the network memorizes the dataset and overfits to wrongly labeled datapoints which results in a performance degradation, as shown in our experiments § 4. Let us consider the clean and noisy labels $\hat{Y}$ and $\widetilde{Y}$ separately by splitting the first term in Eq. 5 which leads to:

$$
\begin{aligned}
I(Z;Y,\boldsymbol{\phi}) &= I(Z;\hat{Y},\boldsymbol{\phi}) + I(Z;\widetilde{Y},\boldsymbol{\phi}) \\
&= \int p(\hat{\boldsymbol{y}},\boldsymbol{z}|\boldsymbol{\phi}) \log \frac{p(\hat{\boldsymbol{y}},\boldsymbol{z}|\boldsymbol{\phi})}{p(\hat{\boldsymbol{y}}|\boldsymbol{\phi})p(\boldsymbol{z}|\boldsymbol{\phi})} \, d\hat{\boldsymbol{y}} \, d\boldsymbol{z} \\
&+ \int p(\widetilde{\boldsymbol{y}},\boldsymbol{z}|\boldsymbol{\phi}) \log \frac{p(\widetilde{\boldsymbol{y}},\boldsymbol{z}|\boldsymbol{\phi})}{p(\widetilde{\boldsymbol{y}}|\boldsymbol{\phi})p(\boldsymbol{z}|\boldsymbol{\phi})} \, d\widetilde{\boldsymbol{y}} \, d\boldsymbol{z}. \tag{6}
\end{aligned}
$$

Substituting $p(\boldsymbol{y},\boldsymbol{z}) = p(\boldsymbol{y}|\boldsymbol{z})p(\boldsymbol{z})$ in Eq. 6, we obtain:

$$
\begin{aligned}
I(Z;\hat{Y},\boldsymbol{\phi}) &= \int p(\hat{\boldsymbol{y}},\boldsymbol{z}|\boldsymbol{\phi}) \log \frac{p(\hat{\boldsymbol{y}},\boldsymbol{z}|\boldsymbol{\phi})}{p(\hat{\boldsymbol{y}}|\boldsymbol{\phi})} \, d\hat{\boldsymbol{y}} \, d\boldsymbol{z} \\
I(Z;\widetilde{Y},\boldsymbol{\phi}) &= \int p(\widetilde{\boldsymbol{y}},\boldsymbol{z}|\boldsymbol{\phi}) \log \frac{p(\widetilde{\boldsymbol{y}}|\boldsymbol{z};\boldsymbol{\phi})}{p(\widetilde{\boldsymbol{y}}|\boldsymbol{\phi})} \, d\widetilde{\boldsymbol{y}} \, d\boldsymbol{z}. \tag{7}
\end{aligned}
$$

As estimating terms $p(\widetilde{\boldsymbol{y}}|\boldsymbol{z};\boldsymbol{\phi})$ and $p(\hat{\boldsymbol{y}}|\boldsymbol{z};\boldsymbol{\phi})$ of the mutual information is often computationally intractable, it can be completed with the variational approximation. We note that:

$$
\begin{aligned}
I(Z;\hat{Y},\boldsymbol{\phi}) &\geq \int p(\hat{\boldsymbol{y}},\boldsymbol{z}|\boldsymbol{\phi}) \log \frac{\bar{p}(\hat{\boldsymbol{y}}|\boldsymbol{z};\boldsymbol{\phi})}{p(\hat{\boldsymbol{y}}|\boldsymbol{\phi})} \, d\hat{\boldsymbol{y}} \, d\boldsymbol{z}, \\
I(Z;\widetilde{Y},\boldsymbol{\phi}) &\geq \int p(\widetilde{\boldsymbol{y}},\boldsymbol{z}|\boldsymbol{\phi}) \log \frac{\bar{p}(\widetilde{\boldsymbol{y}}|\boldsymbol{z};\boldsymbol{\phi})}{p(\widetilde{\boldsymbol{y}}|\boldsymbol{\phi})} \, d\widetilde{\boldsymbol{y}} \, d\boldsymbol{z}.
\end{aligned}
$$

We formulate the lower bound for the information bottleneck with noisy labels as follows:

$$
\begin{aligned}
&I(Z;\hat{Y},\boldsymbol{\phi}) + I(Z;\widetilde{Y},\boldsymbol{\phi}) - \beta I(X;Z,\boldsymbol{\phi}) \geq \\
&\int p(\boldsymbol{x})p(\hat{\boldsymbol{y}}|\boldsymbol{x};\boldsymbol{\phi})p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi}) \log \bar{p}(\hat{\boldsymbol{y}}|\boldsymbol{z};\boldsymbol{\phi}) \, d\boldsymbol{x} \, d\hat{\boldsymbol{y}} \, d\boldsymbol{z} \\
&+ \int p(\boldsymbol{x})p(\widetilde{\boldsymbol{y}}|\boldsymbol{x};\boldsymbol{\phi})p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi}) \log \bar{p}(\widetilde{\boldsymbol{y}}|\boldsymbol{z};\boldsymbol{\phi}) \, d\boldsymbol{x} \, d\widetilde{\boldsymbol{y}} \, d\boldsymbol{z} \\
&- \beta \int p(\boldsymbol{x})p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi}) \log \frac{p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})}{p(\boldsymbol{z}|\boldsymbol{\phi})} \, d\boldsymbol{x} \, d\boldsymbol{z} = \mathcal{L}. \\
&\tag{8}
\end{aligned}
$$

Let $q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})$ and $r(\boldsymbol{z})$ be a variational approximation for $p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})$ and $p(\boldsymbol{z}|\boldsymbol{\phi})$, then we have the lower bound:

$$
\begin{aligned}
\mathcal{L} \approx \frac{1}{|\hat{Y}| + |\widetilde{Y}|} \sum_{n=1}^{|\hat{Y}|+|\widetilde{Y}|} &\mathbb{E}_{\boldsymbol{z} \sim q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})} \big[\log p(\boldsymbol{y}_n|\boldsymbol{z}) \big] \\
&- \beta \mathrm{KL}\big[q(\boldsymbol{z}|\boldsymbol{x}_n;\boldsymbol{\phi}) \| r(\boldsymbol{z}) \big], \tag{9}
\end{aligned}
$$

where $q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})$ denotes our variational approximation using the reparameterization trick [3, 18]. Given the case with noisy labels, we can reduce $I(Z;\widetilde{Y},\boldsymbol{\phi})$ by controlling the first term in Eq. 9 to avoid overfitting in the training phase. In fact, we can control the first term by adjusting the noise variance of the second term of Eq. 9 as follows.

Suppose an encoder is used for $q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})$, $r(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{0}, \mathbf{I})$ and $\boldsymbol{\phi} = \{\phi_1, \phi_2\}$. We can calculate the KL divergence in the closed form as follows:

$$\mathrm{KL}(q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})\|r(\boldsymbol{z})) = -\sum_j^J \log \sigma_j^2 + \frac{\sigma_j^2 + \mu_j^2}{2} - \frac{1}{2}, \tag{10}$$

where $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{x}}, \boldsymbol{\sigma}_{\boldsymbol{x}}^2) = q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})$, $\boldsymbol{\mu}_{\boldsymbol{x}} = f_{\phi_1}(\boldsymbol{x})$, $\boldsymbol{\sigma}_{\boldsymbol{x}} = f_{\phi_2}(\boldsymbol{x})$, and j is the index element of $\boldsymbol{\mu}_{\boldsymbol{x}}$ and $\boldsymbol{\sigma}_{\boldsymbol{x}}$.

3.3. DARTS meets the label noise

The variational information bottleneck provides an inspiration for our noise injecting operators for differentiable NAS. It helps us limit overfitting and memorization to noisy labels. We argue that the designed operator should exhibit the variational approximation in Eq. 9 to adjust $I(Z;\widetilde{Y})$. To this end, a robust operator should inject the right levels
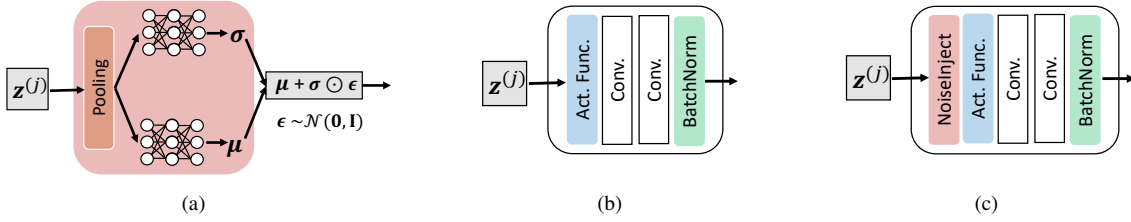
Figure 2: (a) The noise injection module which is incorporated into a convolutional operator. (b) The basic convolutional operator used in DARTS (c) Our convolutional operator with additional noise injection at the input h.

of noise to successfully train NAS in the presence of label noise. Thus, we propose a new operator, the noise injecting Convolution, *nConv* for short. In NAS, the architecture parameters α play an important role in assigning weights of operators. As we train the model under label noise, the larger the weights corresponding to our noise injecting operator are, the stronger the regularization effect becomes.

We define the functionality of our nConv operator on its input z as $z \sim \mathcal{N}(\mu_x, \sigma_x^2)$[3]. By properly scaling the noise parameters, we expect a neural network with nConv to exhibit robustness against noisy labels. To this end, we propose to learn the noise parameters (*ie.*, μ_x and σ_x^2) based on the data distribution. Drawing inspiration from the reparameterization trick [18], this is achieved by learning μ and σ^2 through a network parameterized by ϕ as $z \sim q_z(z|x; \phi)$ (see Fig. 2 (a) for a conceptual diagram). In DARTS, ϕ corresponds to noise parameters of all instantiated noise injecting operators. Let us assume ϕ is contained within θ represents the entire learnable set of parameters of all instantiated operators. As in Eq. 9, the problem of learning the parameters of noise is cast as marginalizing the negative log-likelihood over the data with a KL divergence regularization term as:

$$\mathcal{L}_{\text{NAS}} = \sum_{(\boldsymbol{x}, \tilde{\boldsymbol{y}}) \in \tilde{D}} -\mathbb{E}_{\boldsymbol{z} \sim q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\alpha}; \boldsymbol{\theta})} \big[\log p\big(\tilde{\boldsymbol{y}}|\boldsymbol{x}; \boldsymbol{\theta}; \boldsymbol{\alpha}; \boldsymbol{z}\big) \big]$$
$$\qquad\qquad (11)$$
$$+ \beta \text{KL}\big[q\big(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\alpha}; \boldsymbol{\theta}\big) || \mathcal{N}(\mathbf{0}, \mathbf{I})\big] \,.$$

Note that, minimizing Eq. 11 also applies when we retrain the network in the evaluation phase by using a discrete version of α. Algorithm 1 details the steps of performing DARTS inclusive of our nConv operator.

In vanilla DARTS, two blocks of convolution, namely *Separable Convolution* (SepConv) and *Dilated Convolution* (DilConv) are used to construct the search space. Both blocks share the same architectural design (see Fig. 2 (b)). The variants of these two blocks are constructed by adjusting the kernel size, the dilation size, and the stride size of the underlying convolutions. We extend this basic block by

---

[3]We have modeled the Normal distribution by a diagonal covariance. However, our algorithm is generic and can deal with full covariance matrices. We denote the diagonal elements of the covariance matrix by $\boldsymbol{\sigma^2}$.

Algorithm 1 DARTS + nConv

Operation $\mathcal{O}$, arch. parameters $\boldsymbol{\alpha}$, network parameters $\boldsymbol{\theta}$, noise injection parameters $\boldsymbol{\phi}$, dataset $\tilde{\mathcal{D}}$

1: **Phase 1: Search an architecture**
2: **while** not converged **do**
3:     Update $\boldsymbol{\alpha}$ using Equation 2
4:     Update $\boldsymbol{\theta}, \boldsymbol{\phi}$ using $\nabla_{\boldsymbol{\theta}, \boldsymbol{\phi}} \mathcal{L}_{\text{NAS}}$
5: **Phase 2: Evaluate an architecture**
6: Construct a final architecture from $\boldsymbol{\alpha}$
7: Reinitialize $\boldsymbol{\theta}, \boldsymbol{\phi}$
8: **while** not converged **do**
9:     Update $\boldsymbol{\theta}, \boldsymbol{\phi}$ using $\nabla_{\boldsymbol{\theta}, \boldsymbol{\phi}} \mathcal{L}_{\text{NAS}}$

---

injecting noise at the input $\boldsymbol{z}^{(j)}$ (see Fig. 2 (c)). We can readily extend both blocks with nConv to attain *SepNConv* and *DilNConv* blocks. At the cell level, both normal and reduction cells may include nConv in their structures. In the normal cell, the feature map is generated with the same dimension as the input. For the reduction cell, the input to the noise injection module is applied with a stride of two to shrink the cell output size by a factor of two. Selecting nConv in the cells encourages stochastic activations (the output of hidden units) that help regularize the neural network.

Note that we experimentally observe that multiplication of noise and hidden units (*e.g.* dropout) in nConv does not work well to tackle the label noise problem. Thus, our noise injection function is defined by addition of noise to the hidden units.

Theoretical Interpretation. We wire up a network under noise conditions. To this end, it does not suffice to merely tackle the label noise at the classifier layer. From the information bottleneck, the model with the noise injector emerges $\mathbf{x}_{l+1} = f(\mathbf{x}_l; \boldsymbol{\Theta}_l^{\mu}) + \mathcal{N}(\mathbf{0}, \mathbf{I}) g(\mathbf{x}_l; \boldsymbol{\Theta}_l^{\sigma})$. In effect, $g(\cdot)$ can be understood as modeling the variance by $\boldsymbol{\Theta}_l^{\sigma}$ and hence, the sum $\mathbf{x}_{l+1} = f(\cdot) + \mathcal{N}(\mathbf{0}, \mathbf{I}) g(\cdot)$ pushes $\mathbf{x}_{l+1}$ to contain the information about the parameter uncertainty. The uncertainty flows through the layers as shown for example in Fig. 2. This propagation of uncertainty prevents trivial skip-connections.
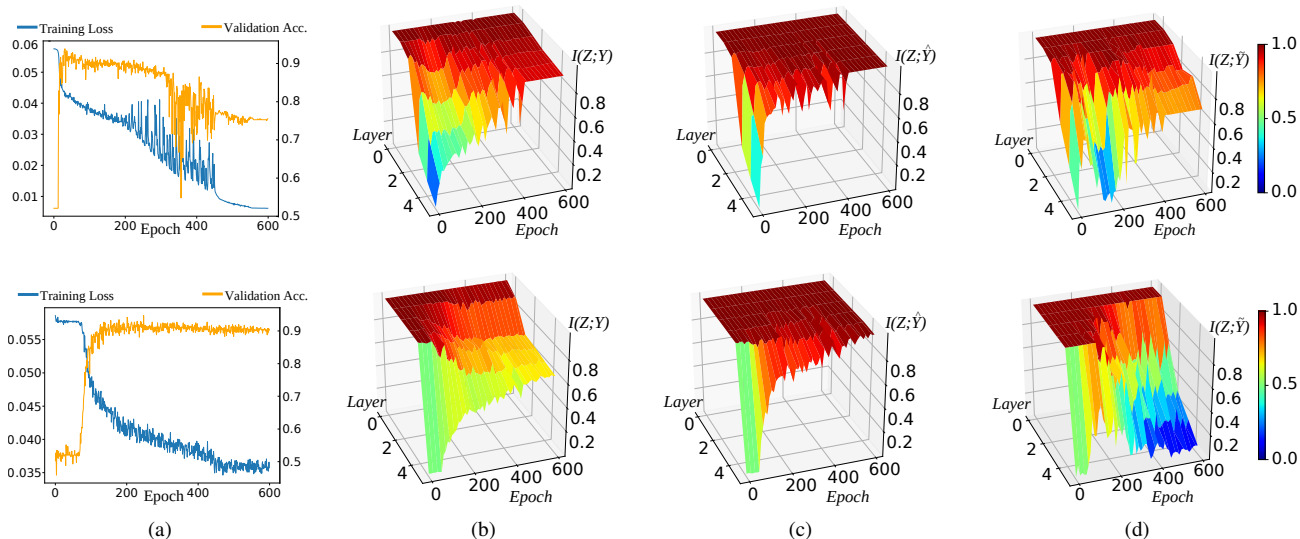
Figure 3: Visualization of mutual information for noisy labels. (Top) Training under 20% label noise without (top) and with our approach (bottom). (a) Training loss and validation accuracy across 600 epochs. (b) Mutual information of all labels and the hidden units ($I(Z;Y)$). (c) Mutual information of clean labels with hidden units. (d) Mutual information of noisy labels with hidden units. The higher the value of $I(Z;Y)$, the more information about Y is preserved by variable Z.

4. Experiments

4.1. The dynamics of training under label noise

In this experiment, we estimate $I(Z;Y)$ to analyze the mutual information between the hidden units of neural networks (with and w/o noise injecting operators) and their respective labels. The settings are adopted from [36] which provides a toy dataset and a technique to visualize the mutual information. The network for this experiment has six layers in total with Tanh activations on intermediate layers and a Sigmoid function for final outputs. We contaminate 20% training data from the toy dataset [36] that has only two categories. Fig. 3 shows that training the network with our approach ($\beta = 1$) under label noise is more stable across 600 epochs. The mutual information is indicative of a correlation between two variables. Fig. 3 (d) shows that a neural network w/o noise injecting operator overfits to noisy labels (top) while our proposed noise injection reduces overfitting (bottom). This experiment confirms our theoretical analysis that the variational information bottleneck, in essence, reduces the mutual information between the hidden units and noisy labels to combat memorization of corrupted labels.

4.2. Searching the architecture

We search an architecture with the following operators $\mathcal{O}$: 3×3 separable convolution, 3×3 dilated convolution, 3×3 separable convolution with noise, 3×3 dilated convolution with noise, 3×3 max pooling, 3×3 average pooling, and the identity. Separable and dilated convolution follows the

operations in [23, 30, 63]. The architecture search is run on CIFAR-10 consisting of training and validation sets. The setup is similar to DARTS in that there are 7 nodes for each cell and 16 initial filters. We use the momentum SGD, we set the learning rate and the momentum to 0.025 and 0.9, resp., and the weight decay to 3×10^{-4}. Moreover, a cosine schedule is employed to anneal the learning rate. A network with 8 cells is trained for 50 epochs with a batch size of 96. The search time is 6 hours on a NVIDIA Titan V GPU.

4.3. Datasets and implementations details

The evaluation phase follows the protocol in DARTS in which the architecture is found via a small task and then the cell structure is transferred to other (bigger) tasks. Normal cells and reduction cells are based on the architecture found on CIFAR-10, then the cells are stacked for evaluation. All experiments for the architecture evaluation are averaged over 3 runs. The model is trained for 300 epochs for all datasets. In the evaluation phase, the cells are stacked to form a group of 16 cells and 16 initial filters. The optimizer parameters are set following the setup during the search phase.

Benchmark datasets for robustness evaluation used in our experiments are CIFAR-10 and CIFAR-100 consisting of 50k samples and 10k samples for training and testing, respectively. The image size of both CIFAR datasets are 32×32 following the standard protocol in past NAS and noisy label works [15, 23]. We investigate both symmetric and asymmetric label noise. Symmetric noise rates on CIFAR-10 and CIFAR-100 are 20% and 50%, and the asymmetric
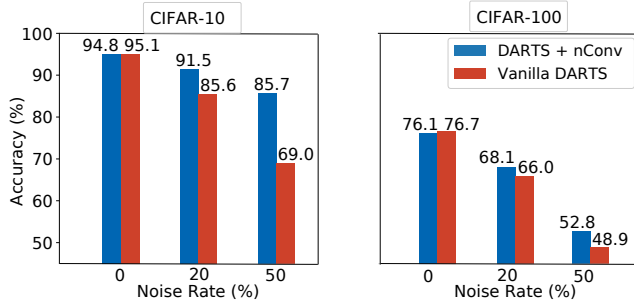
Figure 4: Neural architecture search under noisy labels with DARTS and DARTS + nConv on CIFAR-10 and CIFAR-100. The comparison uses three protocols: clean, 20%, and 50% symmetric label noise.

noise rate is 40%. All results are reported based on the average of the last ten epochs with $\beta = 1$. We also run evaluations on a larger Tiny-ImageNet[4] with 200 classes and 120k images in total. The image size of Tiny-ImageNet is set to 64×64 following the setup in [52]. Finally, we experiment with the Clothing1M [49] which contains real-world noisy labels, 14 categories, and ~1M images (224×224 size). The network is trained for 80 epochs. SGD with momentum is used to optimize the model with the learning rate set to 0.1. Remaining parameters follow the setup for CIFAR.

4.4. Comparison with Vanilla DARTS

Below, we compare the performance of Vanilla DARTS and our DARTS with the nConv operator (DARTS + nConv) to study the improvement of the latter when learning in the presence of label noise. We use DARTS with following operators $\mathcal{O}$: 3×3 separable convolution, 3×3 dilated convolution, 5×5 separable convolution, 5×5 dilated convolution, 3×3 max pooling, 3×3 average pooling, and the identity. We use the same setup (*e.g.* optimizer parameters/batch size) for both techniques to search and evaluate the found architecture. The search/evaluation phases use the same label noise rates (*e.g.*, searching on CIFAR-10 with 50% noise and evaluating the found architecture on CIFAR-100 with 50% noise). Fig. 4 shows that DARTS + nConv achieves superior performance compared to Vanilla DARTS.

4.5. Comparison with the state of the art for training under label noise

We compare our method to multiple baselines that are designed to tackle problems resulting from label noise. The baselines we compare to, use three different backbones: Conv-9 [15], ResNet-18, and ResNet-34 [16]. Table 2 shows the results on CIFAR-10 and CIFAR-100 in the presence of 20% as well as 50% symmetric noise. Our proposed method with nConv outperforms T-Revision [48] on all protocols by

---

[4]https://tiny-imagenet.herokuapp.com/

Method	Backbone	20%-Sym	50%-Sym
F-Correction [28]	Conv-9 (Size: 4.4M)	84.6	59.8
Decoupling [25]		80.4	51.5
MentorNet [17]		80.8	71.1
Co-Teaching [15]		82.3	74.0
JoCoR [47]		85.7	79.41
Cross Entropy	ResNet-18 (Size: 11.2M)	85.6	57.8
F-Correction [28]		83.1	59.4
Decoupling [25]		79.9	52.2
MentorNet [17]		80.5	70.7
Co-Teaching [15]		82.4	72.8
T Revision [48]		89.6	83.4
DARTS + nConv	Auto (Size: 1.2M)	**91.4**	**85.7**

Table 1: CIFAR-10 test acc. (%) using ResNet and Conv-9.

Method	Backbone	20%-Sym	50%-Sym
F-Correction [28]	Conv-9 (Size: 4.4M)	61.9	41.0
Decoupling [25]		44.5	25.8
MentorNet [17]		52.1	39.0
Co-Teaching [15]		54.2	41.4
JoCoR [47]		53.0	43.5
Cross Entropy	ResNet-34 (Size: 20.1M)	61.2	41.2
F-Correction [28]		61.4	37.3
Decoupling [25]		52.1	38.5
MentorNet [17]		80.5	70.7
Co-Teaching [15]		54.2	41.4
T Revision [48]		65.4	50.5
DARTS + nConv	Auto (Size: 1.2M)	**68.1**	**52.8**

Table 2: CIFAR-100 test acc. (%) using ResNet and Conv-9.

Method	Backbone	Accuracy (%)
Cross-entropy	ResNet-18 (Size: 11.2M)	72.3
F-Correction [28]		83.1
DARTS + nConv	Auto (Size: 1.2M)	**89.5**

Table 3: Testing acc. (%) with 40% asymmetric noise on CIFAR-10.

Method	Backbone	20%-Sym	50%-Sym
Decoupling [25]	ResNet-18 (Size: 11.2M)	36.3	22.6
F-Correction [28]		44.4	32.8
MentorNet [17]		45.5	35.5
Co-teaching [15]		45.6	37.1
Co-teaching+ [52]		47.7	41.2
DARTS + nConv	Auto (Size: 1.2M)	**50.6**	**45.7**

Table 4: Testing acc. (%) with 20% and 50% symmetric noise on Tiny-ImageNet.

a substantial margin of 2%. In addition, Table 3 shows that our proposed method consistently performs better than the state of the art *ie.*, joint-optimization [39], PENCIL [51], and F-Correction [28]. Note that our approach does not need any modifications of the loss function to train the model under symmetric or asymmetric noise.

We further evaluate the performance on challenging datasets, namely Tiny-ImageNet and Clothing1M [49]. Note
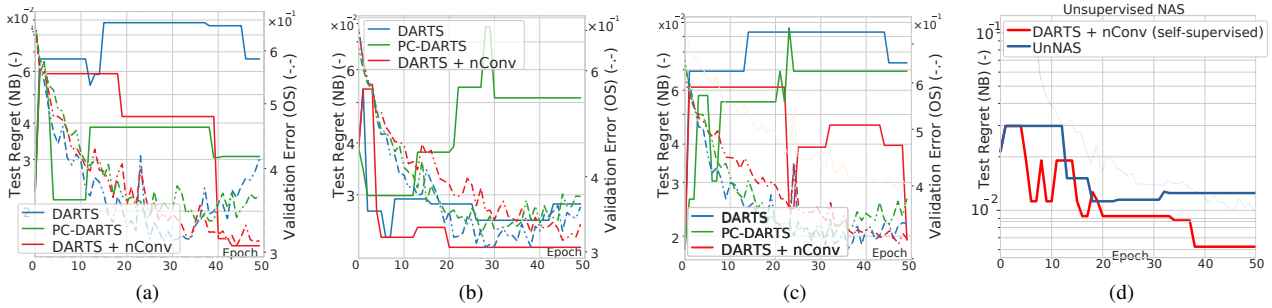
Figure 5: Test regret of NAS benchmark 1shot1 on CIFAR-10 (searching with 50% symmetric noise) using three search spaces: (a) search space 1, (b) search space 2, and (c) search space 3. (d) Test regret of the self-supervised setting using search space 2.

that our evaluation does not use pre-trained neural networks for both datasets. As shown in Table 4, DARTS + nConv also outperforms Co-teaching+ [52], Co-teaching [15], MentorNet [17], and F-Correction [28]. When the noise rate increases from 20% to 50%, the performance gap is widened. As an example, the performance gap increases by 3% when comparing our method with Co-teaching [15]. Moreover, Table 5 shows that our method outperforms state-of-the-art methods on the Clothing1M dataset by 1.3%.

Our proposed method outperforms current state of the art on all datasets irrespective of noise rates while substantially reducing the number of parameters compared to Conv-9, ResNet-18, and ResNet34 backbones. The searched architecture has fewer parameters (size reduced by 10-20×) compared to ResNet backbones. Finally, our approach maintains only one neural network while previous methods (*e.g.*, Decoupling [25] and Co-teaching [15]) train two networks simultaneously.

Dataset	DARTS + nConv	Co-Teaching [15]	Decoupling [25]	F-Correction [28]
Clothing1M	**69.8**	68.5	67.3	65.4

Table 5: Test acc. on the Clothing1M dataset. The prior methods use ResNet-18.

4.6. NAS benchmark evaluation

Below, we examine the capability of DARTS to obtain high performance architectures in the presence of label noise. We compare our method with Vanilla DARTS and PC-DARTS [50] using NAS-benchmark 1Shot1 [54]. There are three different search spaces (1,2, and 3) used in our evaluation which differ by the number of nodes and node parents. The search space 3 in this benchmark is the largest search space (∼360,000 architectures) among NAS benchmarks. CIFAR-10 is used to search the architecture with 50% symmetric noise. Fig. 5 shows that our method does not overfit to the noisy labels and performs better (lower test regret) compared to Vanilla DARTS and PC-DARTS. Fig

5. Discussion

In our experiments, we have observed that reducing the mutual information by adjusting the noise variance (following the ELBO derivations) according to data helps us overcome overfitting to the label noise. The benefit of our structural approach is that we train a neural network in the presence of label noise with fewer hyper-parameters *e.g.*, no noise rate [15] or a bootstrapping parameter [32]. Moreover, a standard training loss (*e.g.*, cross-entropy) can be used directly without any modifications (*e.g.*, multiplying with a certain weight or hinge loss) for both label noise cases.

In the recent NAS work, Liu *et al*. [22] proposes a self-supervised method so-called UnNAS to search an optimal architecture. However, the labels of pretext tasks (*e.g.*, rotation, scaling, permutations [56, 57]) suffer from the noise depending on the initial state of the input. Even unsupervised, contrastive and hallucinating methods [44, 45, 59–61] suffer from the diffusion, negative sampling noise or quantization noise. Yet, NAS using our approach is able to find a better architecture compared to UnNAS [22]. Fig. 5 (d) shows that our method outperforms the result of UnNAS by achieving lower test regret of NAS benchmark 1shot1.

6. Conclusions

We have presented a new operator coined **nConv** to search and train a robust neural network in the presence of label noise. We have shown that label noise harms the performance of Vanilla DARTS while DARTS + nConv performs robustly on problems with noisy labels. The proposed operator has a minimum amount of overhead compared to the existing operators in the search space (*e.g.*, convolution operations in Vanilla DARTS). As changes are applied to the neural network structure, mechanisms such as sample selection, modified loss functions or transition matrices are not needed by our approach. Our empirical results also show that we can design a neural network with fewer parameters compared to ResNet backbones, without the loss of accuracy.

References

[1] Alessandro Achille and Stefano Soatto. Emergence of invariance and disentanglement in deep representations. *The Journal of Machine Learning Research*, 19(1):1947–1980, 2018. 4

[2] Alessandro Achille and Stefano Soatto. Information dropout: Learning optimal representations through noisy computation. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2897–2905, 2018. 3

[3] Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. Deep variational information bottleneck. In *International conference on Learning Representations*, 2017. 2, 4

[4] G. An. The effects of adding noise during backpropagation training on a generalization performance. *Neural Computation*, 8:643–674, 1996. 3

[5] Eric Arazo, Diego Ortego, Paul Albert, Noel O'Connor, and Kevin Mcguinness. Unsupervised label noise modeling and loss correction. In *International Conference on Machine Learning*, pages 312–321, 2019. 1, 2

[6] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. In *International Conference on Learning Representations*, 2016. 2

[7] Yoshua Bengio. Gradient-based optimization of hyperparameters. *Neural computation*, 12(8):1889–1900, 2000. 3

[8] Chris M Bishop. Training with noise is equivalent to tikhonov regularization. *Neural computation*, 7(1):108–116, 1995. 3

[9] Yukang Chen, Tong Yang, Xiangyu Zhang, Gaofeng Meng, Xinyu Xiao, and Jian Sun. Detnas: Backbone search for object detection. In *Advances in Neural Information Processing Systems*, pages 6638–6648, 2019. 2

[10] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 1

[11] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Efficient multi-objective neural architecture search via lamarckian evolution. In *International Conference on Learning Representations*, 2019. 1, 2

[12] Xinyu Gong, Shiyu Chang, Yifan Jiang, and Zhangyang Wang. Autogan: Neural architecture search for generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3224–3234, 2019. 2

[13] Caglar Gulcehre, Marcin Moczulski, Misha Denil, and Yoshua Bengio. Noisy activation functions. In *International conference on machine learning*, pages 3059–3068, 2016. 3

[14] Sheng Guo, Weilin Huang, Haozhi Zhang, Chenfan Zhuang, Dengke Dong, Matthew R Scott, and Dinglong Huang. Curriculumnet: Weakly supervised learning from large-scale web images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 135–150, 2018. 2

[15] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in neural information processing systems*, pages 8527–8537, 2018. 1, 2, 6, 7, 8

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 7

[17] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International Conference on Machine Learning*, pages 2304–2313, 2018. 1, 2, 7, 8

[18] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 4, 5

[19] Jan Larsen, Lars Kai Hansen, Claus Svarer, and M Ohlsson. Design and regularization of neural networks: the optimal use of a validation set. In *Neural Networks for Signal Processing VI. Proceedings of the 1996 IEEE Signal Processing Society Workshop*, pages 62–71, 1996. 3

[20] Dongze Lian, Yin Zheng, Yintao Xu, Yanxiong Lu, Leyu Lin, Peilin Zhao, Junzhou Huang, and Shenghua Gao. Towards fast adaptation of neural architectures with meta learning. In *International Conference on Learning Representations*, 2020. 2, 3

[21] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 82–92, 2019. 2, 3

[22] Chenxi Liu, Piotr Dollár, Kaiming He, Ross Girshick, Alan Yuille, and Saining Xie. Are labels necessary for neural architecture search?, 2020. 8

[23] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019. 1, 2, 3, 6

[24] Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. Neural architecture optimization. In *Advances in neural information processing systems*, pages 7816–7827, 2018. 1

[25] Eran Malach and Shai Shalev-Shwartz. Decoupling" when to update" from" how to update". In *Advances in Neural Information Processing Systems*, pages 960–970, 2017. 7, 8

[26] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *Advances in neural information processing systems*, pages 1196–1204, 2013. 1

[27] Hyeonwoo Noh, Tackgeun You, Jonghwan Mun, and Bohyung Han. Regularizing deep neural networks by noise: Its interpretation and optimization. In *Advances in Neural Information Processing Systems*, pages 5109–5118, 2017. 3

[28] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1944–1952, 2017. 1, 2, 7, 8

[29] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *ICML*, pages 4092–4101, 2018. 1

[30] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture

search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789, 2019. 1, 2, 3, 6

[31] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V. Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, page 2902–2911, 2017. 1, 2

[32] Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. In *International Conference on Learning Representations*, 2015. 2, 8

[33] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *International Conference on Machine Learning*, pages 4334–4343, 2018. 1, 2

[34] Shreyas Saxena and Jakob Verbeek. Convolutional neural fabrics. In *Advances in Neural Information Processing Systems*, pages 4053–4061, 2016. 1

[35] Chuan sheng Foo, Chuong B. Do, and Andrew Y. Ng. Efficient multiple hyperparameter learning for log-linear models. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 377–384, 2008. 3

[36] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017. 3, 6

[37] Kenneth O. Stanley, Jeff Clune, Joel Lehman, and Risto Miikkulainen. Designing neural networks through neuroevolution. pages 24–35, 2019. 1, 2

[38] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 1

[39] Daiki Tanaka, Daiki Ikami, Toshihiko Yamasaki, and Kiyoharu Aizawa. Joint optimization framework for learning with noisy labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5552–5560, 2018. 2, 7

[40] Ryutaro Tanno, Ardavan Saeedi, Swami Sankaranarayanan, Daniel C Alexander, and Nathan Silberman. Learning from noisy labels by regularized estimation of annotator confusion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11244–11253, 2019. 1

[41] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000. 2, 4

[42] Arash Vahdat. Toward robustness against label noise in training deep discriminative neural networks. In *Advances in Neural Information Processing Systems*, pages 5596–5605, 2017. 1, 2

[43] Tom Veniat and Ludovic Denoyer. Learning time/memory-efficient deep architectures with budgeted super networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3492–3500, 2018. 1

[44] Lei Wang and Piotr Koniusz. Hallucinating statistical moment and subspace descriptors for action recognition. *ACM Multimedia*, 2021. 8

[45] Lei Wang, Piotr Koniusz, and Du Q. Huynh. Hallucinating idt descriptors and i3d optical flow features for action recognition with cnns. *International Conference on Computer Vision*, 2019. 8

[46] Yisen Wang, Xingjun Ma, Zaiyi Chen, Yuan Luo, Jinfeng Yi, and James Bailey. Symmetric cross entropy for robust learning with noisy labels. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 1

[47] Hongxin Wei, Lei Feng, Xiangyu Chen, and Bo An. 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*, pages 13726–13735, 2020. 7

[48] Xiaobo Xia et al. Are anchor points really indispensable in label-noise learning? In *Advances in Neural Information Processing Systems*, pages 6835–6846, 2019. 1, 2, 7

[49] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2691–2699, 2015. 7

[50] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. Pc-darts: Partial channel connections for memory-efficient architecture search. In *International Conference on Learning Representations*, 2020. 2, 3, 8

[51] Kun Yi and Jianxin Wu. Probabilistic end-to-end noise correction for learning with noisy labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7017–7025, 2019. 2, 7

[52] Xingrui Yu, Bo Han, Jiangchao Yao, Gang Niu, Ivor Tsang, and Masashi Sugiyama. How does disagreement help generalization against label corruption? In *International Conference on Machine Learning*, pages 7164–7173, 2019. 7, 8

[53] Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. In *International Conference on Learning Representations*, 2020. 2, 3

[54] Arber Zela, Julien Siems, and Frank Hutter. Nas-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search. In *International Conference on Learning Representations*, 2020. 8

[55] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International conference on Learning Representations*, 2017. 1

[56] Hongguang Zhang, Piotr Koniusz, Songlei Jian, Hongdong Li, and Philip H. S. Torr. Rethinking class relations: Absolute-relative supervised and unsupervised few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9432–9441, 2021. 8

[57] H Zhang, L Zhang, X Qi, H Li, PHS Torr, and P Koniusz. Few-shot action recognition with permutation-invariant attention. In *The European Conference on Computer Vision*, 2020. 8

[58] Zhilu Zhang and Mert Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in neural information processing systems*, pages 8778–8788, 2018. 1, 2

[59] Hao Zhu and Piotr Koniusz. REFINE: Random RangE FInder for Network Embedding. In *ACM International Conference on Information and Knowledge Management*, 2021. 8

[60] Hao Zhu and Piotr Koniusz. Simple spectral graph convolution. In *International Conference on Learning Representations*, 2021. 8

[61] Hao Zhu, Ke Sun, and Piotr Koniusz. Contrastive laplacian eigenmaps. In *Conference on Neural Information Processing Systems*, 2021. 8

[62] Barret Zoph and Quoc Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2017. 1, 2

[63] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018. 1, 2, 3, 6