Searching for Robustness: Loss Learning for Noisy Classification Tasks

Boyan Gao[1] Henry Gouk[1] Timothy M. Hospedales[1, 2]
[1]School of Informatics, University of Edinburgh, United Kingdom
[2]Samsung AI Centre, Cambridge, United Kingdom
{boyan.gao,henry.gouk,t.hospedales}@ed.ac.uk

Abstract

We present a "learning to learn" approach for discovering white-box classification loss functions that are robust to label noise in the training data. We parameterise a flexible family of loss functions using Taylor polynomials, and apply evolutionary strategies to search for noise-robust losses in this space. To learn re-usable loss functions that can apply to new tasks, our fitness function scores their performance in aggregate across a range of training datasets and architectures. The resulting white-box loss provides a simple and fast "plug-and-play" module that enables effective label-noise-robust learning in diverse downstream tasks, without requiring a special training procedure or network architecture. The efficacy of our loss is demonstrated on a variety of datasets with both synthetic and real label noise, where we compare favourably to prior work.

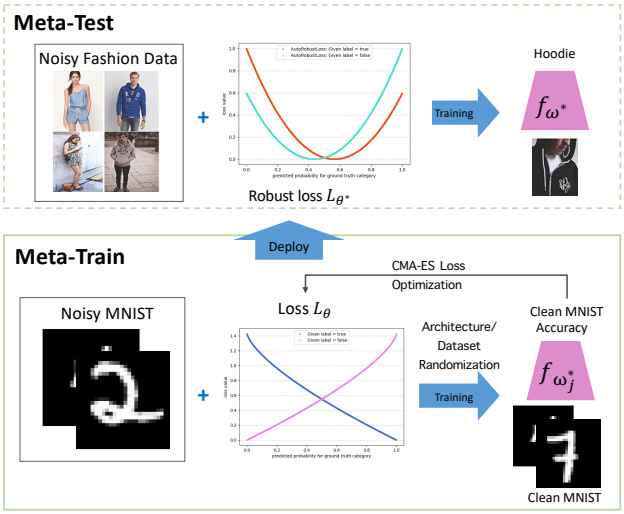

Figure 1. Schematic of our robust loss search framework. (1) We train a robust loss function so as to optimise validation performance of a CNN trained with synthetic label noise using this loss. (2) Thanks to dataset and architecture randomisation, our AutoRobust-Loss (ARL) is reusable and can be deployed to new tasks, including those without clean validation set to drive robust learning.

1. Introduction

The success of modern deep learning is predicated on large amounts of accurately labelled training data. However, training with large quantities of gold-standard labelled data is often not achievable. This is because professional annotation is often too costly to achieve at scale and so machine learning practitioners resort to less reliable crowd-sourcing, web-crawled incidental annotations [6], or imperfect machine annotation [27]; while in other situations the data is hard to classify reliably even by human experts, and thus label-noise is inevitable. These considerations have led to a large body of work focusing on developing noise-robust learning approaches [38, 13]. Diverse solutions have been studied including those that modify the training algorithm through teacher-student [23, 13] learning, or identify and down-weight noisy instances [38]. Much simpler, and therefore more widely applicable, are attempts to define noise-robust loss functions that provide drop-in replacements for standard losses such as cross-entropy [45, 54, 10]. These studies hand engineer robust losses, motivated by different considerations including risk minimisation [10] and information theory [51]. In this paper we explore an alternative data-driven AutoML [21] approach to loss design, and search for a simple white-box function that provides a general-purpose noise-robust drop-in loss. While AutoML approaches have been widely [36, 9] and successfully [43] applied to general purpose neural architecture search (NAS), their application to discovery of reusable losses is much less widely studied.

We perform evolutionary search on a space of loss functions parameterised as Taylor polynomials. Every function in this space is smooth and differentiable, and thus provides a valid loss that can be easily plugged into existing deep learning frameworks. Meanwhile, this search space provides a good trade-off between the flexibility to represent non-trivial losses, and a low-dimensional white-box parameterisation that is efficient to search and reusable across tasks without overfitting. To score a given loss during our search, we use it to train neural networks on noisy data, and then evaluate the

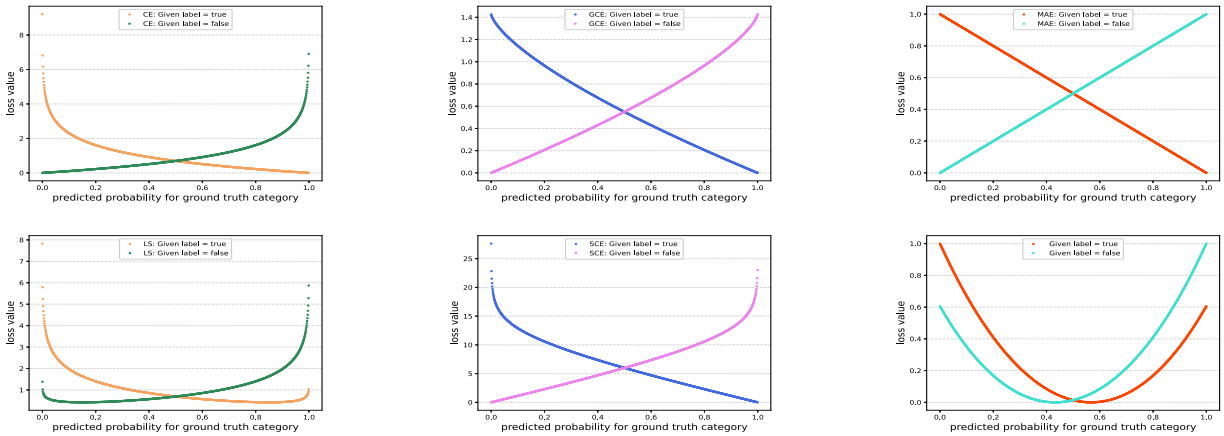

Figure 2. Existing hand-designed robust losses and our meta-learned robust loss. Top left: Conventional Cross-Entropy (CE); Top middle: Generalised Cross Entropy (GCE) [54]; Top right: Mean Absolute Error (MAE) [10]; Bottom left: label-smoothing [34]. Bottom middle: Symmetric Cross Entropy (SCE) [45]. Bottom right: Our learned ARL.

clean validation performance of the trained model. To learn a general purpose loss, rather than one that is specific to a given architecture or dataset, we explore domain randomisation [44] in the space of architectures and datasets. Scoring losses according to their validation performance in diverse conditions leads to reusable functions that can be applied to new datasets and architectures, as illustrated in Figure 1.

We apply our learned ARL to train various MLP and CNN architectures on several benchmarks including MNIST, FashionMNIST, USPS, CIFAR-10, and CIFAR-100 with different types of simulated label noise. We also test our loss on a large real-world noisy label dataset, Clothing1M. The results verify the re-usability of ARL and its efficacy compared to state-of-the-art in a variety of settings. This means that, analogously to CNNs discovered by NAS [36, 43], readers are free to use our loss on new noisy problems with no further complicated or expensive AutoML required. This is an important distinction and major advantage of our approach compared to previous work that uses AutoML or meta-learning techniques to perform noise-robust learning [38, 39]. These methods often require (i) expensive meta-learning on a per-problem basis, and (ii) a clean (i.e., noiseless) validation dataset to use as a meta-supervision signal, which may not be available in real applications. In contrast, our ultimate contribution is a general-purpose loss (Figure 2, bottom right) that provides a simple and fast drop-in replacement for conventional losses (such as cross-entropy) in a standard learning pipeline; and furthermore no clean validation set is required to use it.

2. Related Work

Learning with Label Noise Learning with label noise is now a large research area due to its practical importance. Song *et al*. [41] present a detailed survey explaining the variety of existing approaches including designing noise robust neural network architectures [6], regularisers such as label-smoothing [42, 34], sample selection methods that attempt to filter out noisy samples – often by co-teaching or student teacher learning with multiple neural networks [23, 13, 46, 29], various meta-learning approaches that often aim to down-weight noisy samples using meta-gradients from a validation set [38, 39, 52], and robust loss design. Among these families of approaches, we are motivated to focus on robust loss design due to simplicity and general applicability – we wish to provide a loss that can be widely used together with standard architectures and standard learning algorithms.

Major existing robust losses include: mean absolute error (MAE), shown to be theoretically robust in [10], but hard to train in [54]; generalised cross-entropy (GCE) which attempts to be robust yet easy to train [54]; bi-temper [1], two-temperature [2], and Huber [15] motivated by heavy tailed outlier robustness; symmetric cross-entropy [45] motivated by reducing overfitting; and active-passive loss (APL) [31] which aims to balance over-and-underfitting for robust losses. These losses are all hand-designed based on various good motivations, but (as we will see in our evaluation) none provide reliably high performance empirically. Instead we take a data-driven AutoML approach and search for a loss function that is empirically robust across various benchmark and neural architectures. This draws upon meta-learning techniques but, differently from existing meta-robustness work, focuses on discovering a general white box loss that can be re-used in any downstream problem (Figure 1), unlike others [39, 38, 29, 52] that require expensive per-problem meta-learning. Incidentally, we note that our final loss covers all six desiderata for noise-robust learning outlined in [41].

Meta-learning, AutoML and Loss Learning Meta-learning, aka learning to learn, and AutoML have been applied for a wide variety of purposes as summarised

in [17, 21]. Of particular relevance is meta-learning of loss functions, which has been studied for various purposes including providing differentiable surrogates of non-differentiable objectives [19], optimising efficiency and asymptotic performance of learning [22, 4, 18, 48, 11, 12], and improving robustness to train/test domain-shift [3, 30]. We are interested in learning *white-box* losses – i.e., those that can be expressed a short human-readable parametric equation – for efficiency and improved task-transferability compared to neural network alternatives [4, 18, 3, 30], which tend to be less interpretable and need to be learned task-specifically. Meta-learning of white-box model components has been demonstrated for optimisers [47], activation functions [35], neural architectures [43] and losses for accelerating conventional supervised learning [11, 12]. We are the first to demonstrate the value of automatic loss function discovery for general purpose label-noise robust learning.

3. Method

We aim to learn a loss function for multi-class classification that is robust to noisy labels in the training set.

Overview Our workflow has two phases (Figure 1). **Meta-train:** Given a set of auxiliary dataset(s) and network architecture(s), we meta learn a label-noise robust loss $\mathcal{L}_\theta$. The auxiliary datasets are assumed to either be clean (in which case we simulate label noise during meta-training), or noisy but come with clean validation sets. The loss $\mathcal{L}_\theta$ should produce models with high performance on the clean validation set(s) after learning on noisy training sets. **Meta-test:** Given the learned robust loss $\mathcal{L}_\theta$ from the previous step, denoted AutoRobustLoss (ARL), we can deploy it to learn any target noisy-label learning problem without requiring a validation set. The target dataset and neural architecture need not overlap with the source dataset/architecture from the meta-train step. The loss provides a drop-in replacement for standard cross-entropy in a conventional learning pipeline.

3.1. Meta-Training Procedure

We formalise loss function learning as a bilevel optimisation with an upper/outer loop problem defined as optimising the parameters of an adaptive loss function $\mathcal{L}_\theta$, and a lower/inner loop problem of training neural networks f_ω using the loss function $\mathcal{L}_\theta$. The upper level optimisation problem uses as a supervision signal the clean validation performance of models trained with the prospective loss function, averaged across a variety of domains. The lower level optimisation problem consists of learning a collection of neural networks f_ω on noisy-label datasets using the prospective loss function $\mathcal{L}_\theta$. The prospective loss functions are represented by their parameters, θ, which correspond to the coefficients of an n-th order polynomial. These polynomials can be viewed as a Taylor expansion of the ideal loss function. The bilevel optimisation problem is given by

Algorithm 1 Robust Loss Function Search

Input: $\mathcal{D}, F, \mu^{(0)}, \Sigma^{(0)}$
Output: $p(\theta; \mu^*, \Sigma^*)$
$t = 0$
while not converged or reached max steps **do**
  $\Theta = \{\theta_1, \theta_2, ..., \theta_n\} \sim p(\theta; \mu^{(t)}, \Sigma^{(t)})$ # Sample losses for exploration
  $G = F \times \mathcal{D} \times \Theta$ # Assign datasets and architectures to losses
  $\mathbf{s} =$ zeros $\in \mathbb{R}^n$
  for all $(f^{(k)}, D_j, \theta_i) \in G$ **do**
    $(D_j^{train}, D_j^{val}) =$ split(D_j) # Train/val splits
    $\omega^* = \arg\min_\omega \mathcal{L}_{\theta_i}(f_\omega^{(k)}, D_j^{train})$ # Train the net
    $\mathbf{s}_i = \mathbf{s}_i + \frac{1}{|F||D|}\mathcal{M}(f_{\omega^*}^{(k)}, D_j^{val})$ # Evaluate on validation data
  end for
  $(\mu^{(t+1)}, \Sigma^{(t+1)}) =$ CMA-ES$(\mu^{(t)}, \Sigma^{(t)}, \Theta, \mathbf{s})$ # Update μ and Σ according to CMA-ES
  $t = t + 1$
end while

$$\max_\theta \mathbb{E}_{D,f}[\mathcal{M}(f_{\omega_D^*}, D^{val})] \quad (1)$$
$$s.t. \quad \omega_D^* = \arg\min_\omega \mathcal{L}_\theta(f_\omega, D^{train}),$$

where $\mathcal{M}(\cdot,\cdot)$ is a fitness function measuring network performance, D is a random variable representing a domain, with noisy training D^{train} and clean validation D^{val} splits, and f is a neural network parameterised by ω. The performance of $f_{\omega_D^*}$, as measured by $\mathcal{M}$, reflects the quality of robust supervision provided by the candidate loss $\mathcal{L}_\theta$ on dataset D. We use the Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) [14] to solve the upper layer problem, and standard stochastic gradient-based optimisation approaches to solve the lower level problems. Algorithm 1 summarises our algorithm for solving the optimisation in Equation 1.

CMA-ES for Loss Function Learning We use CMA-ES to solve the upper optimisation problem, and any variant of stochastic gradient descent for the lower problem. CMA-ES finds a Gaussian distribution defined over the search space of θ that places most of its mass on high quality solutions to the optimisation problem. A benefit of using CMA-ES is that it does not require the performance measurement $\mathcal{M}$ to be differentiable, which means the learned loss function can be evaluated using informative metrics, such as accuracy. Each generation consists of a set, Θ, of loss functions obtained by sampling multiple individuals from the parameter distribution, $p(\theta; \mu, \Sigma) = \mathcal{N}(\mu, \Sigma)$. Each of the individuals,

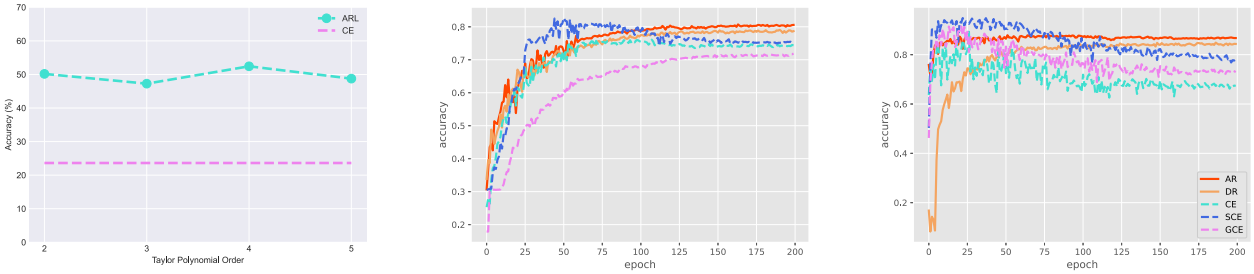

Figure 3. Left: A preliminary experiment on hyperparameter selection. The performance of a linear model trained by the ARL loss function with different orders vs training with cross-entropy (CE). Middle/Right: Example learning curves of test accuracy vs iterations when using different robust losses. Middle: USPS/VGG-11/80% symmetric noise. Right: USPS/ResNet-18/40% asymmetric noise.

$\theta_i \in \Theta$, is evaluated according to

$$\mathbb{E}_{D,f}[\mathcal{M}(f_{\omega_D^*}, D^{val})] \approx \frac{1}{N}\sum_{j=1}^{N}\mathcal{M}(f_{\omega_j}^{(j)}, D_j^{val}) \quad (2)$$
$$\text{s.t.} \quad \omega_j = \arg\min_{\omega} \mathcal{L}_{\theta_i}(f_{\omega}^{(j)}, D_j^{train}),$$

where $f_{\omega}^{(j)}$ and D_j are different network architectures and datasets respectively, as discussed later. We apply a scale normalisation, detailed in Appendix A.3, to the final loss function.

Taylor Polynomial Representation The space of potential loss functions in which CMA-ES searches is a crucial design parameter. For search efficiency, we should consider a space parameterised by a small number of values. This must be balanced with the ability to represent a wide enough variety of functions such that a good solution can be found. By selecting a low-dimensional space with well-understood nonlinear form, it should be possible to re-use the learned loss on diverse problems. The function space that we choose is the Taylor series approximations of all β-times differentiable functions [12], $g : \mathbb{R}^m \to \mathbb{R}$,

$$g(\mathbf{x}) = \sum_{n=0}^{\beta}\frac{1}{n!}\nabla^n g(\mathbf{x}_0)^T(\mathbf{x}-\mathbf{x}_0)^n. \quad (3)$$

where each $\nabla^n g(\mathbf{x}_0)$ is the n-th order gradient of g evaluated at a fixed point, $\mathbf{x}_0$. We make the simplifying assumption that the loss function should be class-wise separable. That is, each potential class is considered in isolation, and we learn a loss function that measures the divergence between a noisy binary label and the probability predicted by the network. To compute the loss on vectors we sum over the C possible classes,

$$\mathcal{L}_\theta(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{C}\sum_{i=1}^{C}\ell_\theta(\hat{\mathbf{y}}_i, \mathbf{y}_i), \quad (4)$$

where $\hat{\mathbf{y}}$ and $\mathbf{y}$ are the vectors of predicted probabilities and (possibly noisy) ground-truth labels, respectively. The result of performing this simplification is that the loss function can be used in a variety of settings with different numbers of classes. We found that $\beta = 4$ is a good trade-off between modelling capacity and meta-training efficiency. In a Taylor expansion the polynomial coefficients are given by the fixed point around which the Taylor expansion is being evaluated and the gradients of the function at this fixed point. Hence, for learning a bi-variate function we say that (θ_0, θ_1) give the location of the fixed point, and $(\theta_2, ..., \theta_{11})$ encode the values of the gradients of the optimal loss function when it is evaluated at (θ_0, θ_1). The resulting loss has the form

$$\begin{aligned}\ell_\theta(\hat{\mathbf{y}}_i, \mathbf{y}_i) &= \theta_2(\hat{\mathbf{y}}_i - \theta_0) + \frac{1}{2}\theta_3(\hat{\mathbf{y}}_i - \theta_0)^2 \quad (5)\\ &+ \frac{1}{6}\theta_4(\hat{\mathbf{y}}_i - \theta_0)^3 + \frac{1}{24}\theta_5(\hat{\mathbf{y}}_i - \theta_0)^4\\ &+ \theta_6(\hat{\mathbf{y}}_i - \theta_0)(\mathbf{y}_i - \theta_1)\\ &+ \frac{1}{2}\theta_7(\hat{\mathbf{y}}_i - \theta_0)(\mathbf{y}_i - \theta_1)^2 + \frac{1}{2}\theta_8(\hat{\mathbf{y}}_i - \theta_0)^2(\mathbf{y}_i - \theta_1)\\ &+ \frac{1}{6}\theta_9(\hat{\mathbf{y}}_i - \theta_0)^3(\mathbf{y}_i - \theta_1) + \frac{1}{6}\theta_{10}(\hat{\mathbf{y}}_i - \theta_0)(\mathbf{y}_i - \theta_1)^3\\ &+ \frac{1}{4}\theta_{11}(\hat{\mathbf{y}}_i - \theta_0)^2(\mathbf{y}_i - \theta_1)^2.\end{aligned}$$

Note that we have omitted terms where $\hat{\mathbf{y}}$ does not appear, as these do not impact the solution of the optimisation problem. In total there are only 12 parameters to fit, which is considerably smaller than the number of parameters found in a typical neural network paramaterised loss function [30, 4, 25].

Generalisation Across Architectures To enable achieve good generalisation to novel architectures in deployment (meta-testing), we apply the domain randomisation [44] strategy to evaluate the expected performance across a range of architectures during meta-training. Specifically, we use a set of architectures, F, containing a variety of common neural network designs. The total population for evolutionary optimisation is then given by the Cartesian product $F \times \Theta$. The fitness function can then be computed as shown in Equation 2, where a mean is taken over all different architectures trained with the same loss.

Generalisation Across Datasets The learned loss should also generalise to novel datasets in deployment (meta-testing). To this end we investigate exposing it to several datasets during training, so as to ensure it is maximally agnostic to specific training dataset. Sampled loss functions are used to train several models with the same architecture and initial weights, but on different datasets. Similarly to architecture generalisation, we use a set of datasets, $\mathcal{D}$, and take the Cartesian product, $\mathcal{D} \times \Theta$, to generate a population to be evaluated. The performance of the loss functions is evaluated by the mean performance of all the networks on their corresponding datasets. In principle, one can perform dataset and architecture randomisation simultaneously. However, due to the implied three-way Cartesian product, we found this computationally infeasible.

4. Experiments

In this section we evaluate ARL on various noisy label learning tasks. In particular, we aim to answer three questions: (Q1) Does our AutoRobustLoss (ARL) generalise across different datasets and architectures? (Q2) How well does ARL generalise across different noise levels? (Q3) Can ARL scale to larger real-world noisy-label tasks?

Datasets We experiment on seven datasets: MNIST [28], CIFAR-10, CIFAR-100 [26], KMNIST [7], USPS [20], FashionMNIST [49] and Clothing1M [50]. Clothing1M is a dataset containing 1 million clothing images in 14 classes (T-shirt, Shirt, Knitwear, Chiffon, Sweater, Hoodie, Windbreaker, Jacket, Down Coat, Suit, Shawl, Dress, Vest, Underwear). The images are collected from shopping websites and the labels are generated from the text surrounding images, thus providing a realistic noisy label setting. MNIST, and optionally KMNIST and CIFAR-10, are used for learning the loss function (meta-training), and the others are completely held out for experimental evaluation (meta-testing).

Noise types For loss learning, we simulate both symmetric and asymmetric (pair-flip) noise types. Symmetric noisy labels are generated by uniformly flipping from the positive label to a negative one, while asymmetric noisy labels are produced to simulate the more realistic scenario where particular pairs of categories are more easily confused than others by annotators. For example, among digits label noise could manifest in such a way that a 7 is more likely to mislabelled as a 1 than as a 6; or a 3 mislabelled as an 8 rather than a 4.

Architectures We train and evaluate ARL with a range of neural networks including shallow (2-layer MLP, 3-layer MLP, and 4-layer CNN) and deep (VGG-11 [40] and ResNet-18 [16]). We also use the medium-size architecture in [46], which we term JoCoR-Net (see supplemental for details). For a fair comparison, we train 2-layer MLP, 3-layer MLP, and 4-layer CNN with SGD optimiser, learning rate 0.01 and momentum 0.9. For JoCor-Net, we apply Adam [24] with learning rate 0.001. For ResNet-18 and VGG-11, we follow the training protocol in [53].

Taylor Polynomial Order Selection We perform a preliminary experiment to select the order of ARL. We train a linear classifier in the inner loop of the dataset randomisation algorithm (on MNIST, KMNIST, and CIFAR-10), and evaluate performance for polynomial orders 2, 3, 4 and 5. From the results in Figure 3(left), we can see that the impact of the specific polynomial order is small compared to the impact of loss learning overall. Nevertheless, we pick order 4 for the subsequent experiments, as this was the hyperparameter that achieved the best performance.

Competitors We compare our ARL with the standard cross-entropy (CE) baseline, as well as several strong alternative losses hand-designed for label-noise robustness: **MAE:** Mean Absolute Error was theoretically shown to be robust in [10]. **GCE:** [54] analysed MAE as hard to train, and proposed generalised cross-entropy to provide the best of CE and MAE; **FW:** [33] iteratively estimates the label noise transfer matrix, and trains the model corrected by the label noise estimate; **SCE:** [45] argued that symmetrising cross-entropy by adding reverse cross-entropy (RCE) improves label-noise robustness; **Bootstrap:** A classic method of replacing the noisy labels in training by the convex combination of the prediction and the given labels [37]. **LSR:** Label-smoothing is an effective general purpose regulariser [34, 42, 32] whose properties in promoting noise robustness have been studied [45]. **Huber** [15] and **Bi-Temper** [1]: Classic and recent approaches based on robust heavy-tailed loss functions. **Active Passive**: The best out of a selection of normalised losses proposed in [31].

Early Stopping and Hyper-parameter Tuning While conventional supervised learning can use early stopping, the lack of a clean validation set during meta-testing makes this impossible. Therefore our main experiments follow the majority of work [23, 46] in this area by reporting performance at convergence. Similarly, lack of a clean validation set prevents automated hyperparameter tuning, so we reuse a single set of hyperparameters chosen on the meta-training sets.

4.1. Training a general-purpose robust loss function

Meta-Training Setup We consider two domain randomisation strategies for training a general purpose loss function, namely architecture (AR) and dataset (DR) randomisation. In AR, we build a pool of training architectures including 2-layer MLP, 3-layer MLP, and 4-layer CNN and solely use MNIST as the training set. In DR, we solely use the 4-layer CNN as the architecture build a dataset pool of MNIST, KMNIST, and CIFAR-10. We train models for 80% symmetric, and 40% asymmetric noise conditions. More details are given in the Appendix.

Meta-Testing (Deployment) Setup Given our ARL learned in meta-training, we evaluate it by deploying on a

Table 1. Accuracy (%) of robust losses, 80% symmetric noise condition. Our loss trained under architecture randomisation (AR) and dataset randomisation (DR) conditions has the best average rank. Grey cols: datasets seen during DR training. White cols: totally novel datasets.

Architecture type dataset	VGG11 Cifar10	VGG11 Cifar100	VGG11 FashionMNIST	VGG11 USPS	ResNet18 Cifar10	ResNet18 Cifar100	ResNet18 FashionMNIST	ResNet18 USPS	Avg.Rank
CE	18.38 ± 0.21	4.25 ± 0.28	20.55 ± 0.93	51.42 ± 0.94	18.44 ± 0.34	8.86 ± 0.10	21.92 ± 0.74	57.05 ± 0.42	6.87
GCE [54]	16.56 ± 0.54	1.04 ± 0.47	25.10 ± 0.68	63.45 ± 0.86	31.69 ± 0.36	11.98 ± 0.18	42.62 ± 0.89	**79.52 ± 0.63**	5.63
SCE [45]	28.61 ± 0.64	2.31 ± 0.80	36.64 ± 0.59	63.68 ± 0.56	**45.34 ± 0.40**	8.16 ± 0.07	59.93 ± 0.75	58.35 ± 0.76	4.63
FW [33]	16.97 ± 0.44	1.41 ± 0.07	22.57 ± 0.76	53.66 ± 0.40	10.15 ± 0.68	1.16 ± 0.04	13.18 ± 0.35	42.80 ± 0.77	9.38
Bootstrap [37]	17.58 ± 0.82	4.18 ± 0.72	20.40 ± 0.31	64.58 ± 0.21	12.10 ± 0.32	8.67 ± 0.61	22.36 ± 1.76	72.17 ± 1.24	6.38
MAE [10]	14.20 ± 0.42	1.01 ± 0.11	63.40 ± 0.16	30.94 ± 0.35	22.95 ± 1.25	0.82 ± 0.17	68.20 ± 1.87	37.17 ± 0.93	8.13
Label-smooth [34]	17.74 ± 0.46	4.47 ± 0.12	21.19 ± 0.39	54.26 ± 0.19	17.67 ± 0.35	7.66 ± 1.52	20.99 ± 0.83	59.94 ± 0.54	6.86
Huber [15]	10.28 ± 0.68	1.30 ± 0.57	19.66 ± 0.67	23.92 ± 1.34	13.56 ± 0.75	1.14 ± 1.11	17.59 ± 0.91	24.61 ± 0.31	11.00
NCE+MAE [31]	40.47 ± 0.93	2.06 ± 0.44	48.40 ± 1.01	70.75 ± 0.71	33.57 ± 1.17	5.72 ± 0.92	48.65 ± 0.96	71.25 ± 1.24	4.50
NFL+MAE [31]	41.91 ± 0.98	2.54 ± 0.63	45.06 ± 1.06	69.36 ± 0.84	37.66 ± 0.64	6.03 ± 0.91	54.43 ± 1.15	72.25 ± 1.60	3.50
Bi-Temper [1]	10.44 ± 0.96	3.23 ± 0.11	15.00 ± 0.46	17.67 ± 0.56	40.41 ± 1.33	9.35 ± 0.52	30.06 ± 0.72	26.91 ± 0.64	7.75
ARL-AR	**41.36 ± 0.47**	**5.63 ± 0.24**	**70.16 ± 0.87**	**78.71 ± 0.90**	29.50 ± 0.30	**14.94 ± 0.26**	71.96 ± 0.89	68.80 ± 0.92	**1.63**
ARL-DR	31.12 ± 0.23	5.04 ± 0.14	67.29 ± 1.01	77.34 ± 1.34	35.23 ± 0.23	13.36 ± 0.63	**71.97 ± 0.87**	70.17 ± 0.64	1.75

Table 2. Accuracy (%) of robust losses. 40% asymmetric noise condition. Our loss trained under architecture (AR) and dataset (DR) randomisation conditions has the best average rank. Grey cols: datasets seen during DR training. White cols: totally novel datasets.

Architecture type dataset	VGG11 Cifar10	VGG11 Cifar100	VGG11 FashionMNIST	VGG11 USPS	ResNet18 Cifar10	ResNet18 Cifar100	ResNet18 FashionMNIST	ResNet18 USPS	Avg.Rank
CE	56.43±0.12	30.20±0.18	50.34±1.23	77.74±0.74	58.69±0.43	44.14±0.15	58.68±0.63	73.84±0.85	6.25
GCE [54]	56.42±0.54	22.39±0.35	53.57±0.47	78.72±0.72	57.90±0.31	40.76±0.24	58.51±0.70	80.77±0.35	6.25
SCE [45]	78.23±0.55	25.33±0.73	64.47±0.97	85.50±0.43	63.22±0.22	40.90±0.37	59.63±0.96	81.57±0.17	3.63
FW [33]	54.42±0.79	5.21±0.39	45.18±0.84	76.41±0.81	48.40±0.08	3.83±0.23	49.46±0.73	46.04±0.18	10.13
Bootstrap [37]	57.69±0.11	**31.07±1.09**	53.23±1.53	77.81±0.61	57.69±0.76	**45.78±0.15**	54.60±0.85	75.67±0.56	5.75
MAE [10]	49.06±0.22	0.96±0.10	49.02±0.27	62.38±0.89	55.67±3.05	1.02±0.14	56.31±1.21	70.05±0.35	10.50
Label-smooth [34]	57.76±0.37	20.64±0.18	51.12±1.03	77.49±0.11	59.69±0.36	39.92±0.49	57.53±0.73	78.97±0.46	6.88
Huber [15]	38.28±0.80	5.18±0.72	75.57±0.93	73.44±2.70	56.11±0.41	4.14±0.37	77.50±1.64	79.37±1.55	7.38
NCE+MAE [31]	66.22±0.64	2.06±0.36	69.83±0.73	87.05±1.32	60.51±0.96	45.00±0.87	63.00±1.97	81.81±1.31	4.00
NFL+MAE [31]	65.55±1.76	2.59±0.17	69.51±1.59	89.24±1.92	62.51±0.83	44.84±1.76	58.55±1.04	82.96±2.82	4.25
Bi-Temper [1]	10.12±0.17	34.22±1.23	18.02±0.87	17.89±0.82	17.74±0.73	45.36±0.43	19.44±1.32	27.85±0.94	9.13
ARL-AR	74.30±0.20	22.50±0.33	**87.23±1.22**	**90.67±1.21**	**86.70±0.12**	44.47±0.48	**89.24±0.25**	**91.17±0.25**	**1.25**
ARL-DR	**79.09±0.51**	18.30±0.27	81.18±0.80	89.78±0.46	68.88±0.41	31.47±0.65	88.22±0.97	89.59±1.05	2.63

fresh suite of evaluation datasets and architectures including those unseen during training. We report results in terms of accuracy at convergence, and summarise via the average ranks of each loss across different datasets and architectures [8].

Benchmark Results The results for symmetric and asymmetric noise are shown in Table 1 and 2 respectively. From the results, we can see that our ARL performs favourably compared to hand-designed alternatives across a variety of benchmarks, with a higher average rank than competitors in both experiments. However, there is no clear winner between architecture (AR) and dataset (DR) randomization for meta-learning. We expect that best performance would be obtained by performing these simultaneously during meta-training, but as this experiment is computationally costly, we leave this to future work. Note that during deployment, all methods have a similar computational cost, except for FW which requires training the network twice for noise estimation.

Analysis of Learning Curves The plots in Figure 3(right) compare the learning curves of test accuracy for USPS/VGG-11 and USPS/ResNet-18 with 80% symmetric and 40% asymmetric noise respectively. We can see that while some alternative losses have early peaks, they all overfit after continued training. As discussed earlier, the asymptotic performance is the relevant and standard [23, 46] metric in this area due to lack of a clean validation set to cherry pick a good iteration; and on this metric our losses are clear winners.

Real-world Clothing1M results The previous experiment reported performance of the learned model after training on manually corrupted labels. In this section, we follow the ResNet-18 setting described in [46] to apply our learned loss to the real-world Clothing1M noisy-label benchmark. Note that neither Clothing1M, nor ResNet-18 were seen during loss discovery meta-learning, above. We train with Adam using learning rate 8×10^{-4}, 5×10^{-4}, 5×10^{-5} for 5 epochs each. We report the mean accuracy of each model after ten trials in Table 4. Among the competitors, JoCoR is the state art method in the broader range of noise robust learners. It uses a complex co-distillation scheme with multiple network branches, while the other listed competitors are simple plug-in robust losses applied to vanilla ResNet training. Nevertheless, ARL obtains the top performance.

4.2. Additional Analysis

Noisy Validation Very recently, the established protocol for noisy-label experiments used in the previous section was challenged in [5], who claim that the metric of validation

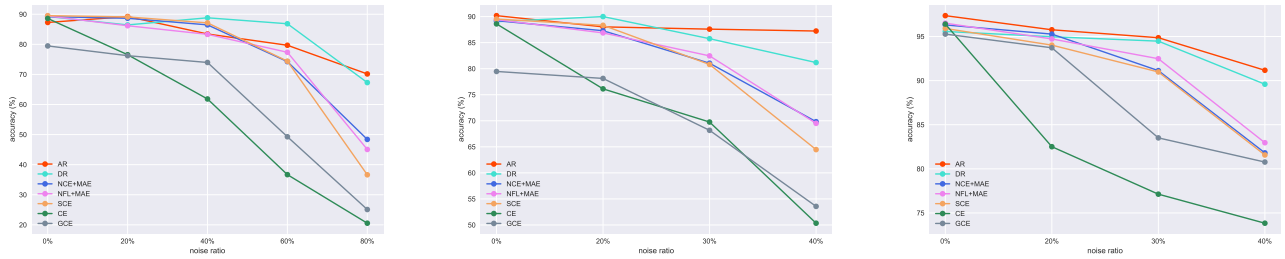

Figure 4. Generalisation of learned ARL loss to varying noise-levels. Left: VGG11-FashionMNIST (Symmetric noise), Middle:VGG11-FashionMNIST (Asymmetric noise), Right: ResNet18-USPS (Asymmetric noise).

Table 3. Accuracy (%) and average rank of different robust losses using noisy-validation based early stopping and hyper-parameter tuning.

	Architecture type dataset	VGG11 Cifar10	VGG11 Cifar100	VGG11 FashionMNIST	VGG11 USPS	ResNet18 Cifar10	ResNet18 Cifar100	ResNet18 FashionMNIST	ResNet18 USPS	Avg.Rank
Symmetric 80%	CE	41.94	12.12	76.03	75.19	39.80	18.18	72.18	75.88	2.88
	GCE [54]	43.94	5.25	74.88	77.68	40.80	15.16	72.50	75.88	3.00
	SCE [45]	**48.48**	7.33	72.17	71.95	**45.81**	16.99	75.36	77.91	2.38
	NCE+MAE [31]	38.96	2.28	75.63	74.39	20.47	10.33	74.33	**82.21**	3.86
	NFL+MAE [31]	43.35	2.61	73.08	73.44	42.92	2.69	70.45	76.83	4.00
	ARL-AR	42.52	**13.71**	71.47	**79.37**	35.86	**20.87**	**77.23**	79.77	**1.88**
	ARL-DR	42.76	7.17	**77.58**	77.58	34.77	17.75	73.28	74.93	3.00
Asymmetric 40%	CE	79.47	29.89	84.07	92.33	82.15	40.36	87.91	89.74	2.86
	GCE [54]	77.13	24.30	85.16	88.09	78.02	43.73	87.57	91.63	3.86
	SCE [45]	78.39	**29.90**	82.95	91.68	78.98	43.06	87.20	93.47	2.86
	NCE+MAE [31]	74.87	7.49	86.81	90.87	80.05	40.89	82.23	90.63	4.38
	NFL+MAE [31]	76.61	7.78	86.36	89.79	76.39	36.79	88.42	92.48	4.25
	ARL-AR	76.13	25.37	**88.81**	**94.27**	**87.03**	45.41	89.44	**94.57**	**1.25**
	ARL-DR	**80.82**	29.21	87.14	91.78	76.62	**47.19**	**89.79**	93.17	1.50

Table 4. Test accuracy (%) of robust learners on Clothing1M with ResNet18. *JoCoR is a multi-network co-distillation training framework. The others are simple plug-in robust losses.

Method	CE	Bootstrap [37]	GCE [54]	FW [33]	SCE [45]	Huber [15]	JoCoR* [46]
Accuracy	66.88	67.28	66.63	68.33	67.63	10.83	69.79
Method	NCE+MAE [31]	NFL+MAE [31]	Bi-Temper [1]	ARL (AR-A40)	ARL (DR-A40)	ARL (AR-S80)	ARL (DR-S80)
Accuracy	66.15	65.97	9.46	69.14	**70.09**	68.85	69.34

set *accuracy* provides a valid model-selection criterion, even when the validation set itself contains label noise. Therefore we select the top performing losses from the previous experiment and report their performance under a new deployment condition using both early stopping and hyper-parameter tuning according to this proxy metric.

From the results in Table 3, we can see that: (i) Early stopping allows CE to reduce overfitting to noise and hence improve in rank compared to the asymptotic results in Tables 1-2, but it is still not best; (ii) Most accuracies have increased compared to the previous condition (e.g., FashionMNIST), but our losses have increased less by comparison, suggesting that AR and DR rely less on careful parameter tuning and checkpoint selection compared to alternatives. (iii) Overall both AR and DR learned losses perform strongly, with AR performing best overall in both noise conditions.

Generalisation across noise-levels We trained our losses on high levels of label noise (80%-symmetric, 40%-asymmetric) as detailed previously, conjecturing that training on a difficult task would be sufficient for generalisation to other tasks with diverse noise conditions, as shown on Clothing1M. To evaluate this more systematically, we next apply our losses on problems with a range of noise levels. From the results in Figure 4 we can see that our loss does provide strong performance across a range of operating points. Notably, the leftmost point on each plot corresponds to the clean data (0% noise) condition. Here our ARL losses provides comparable performance to the standard (i.e., cross-entropy) approach, thus confirming that they are safe to use in cases where it is unknown whether label noise is present or not.

Qualitative analysis of representations We visualise the feature distributions learned by the losses when applied to CIFAR-10 under 40% symmetric label noise in Figure 5. We can see that conventional CE applied on noisy labels leads to a very mixed distribution of instances, while our loss leads to quite cleanly separable clusters despite the label noise.

Table 5. Accuracy (%) of different robust learners. JoCoR net CNN used throughout. ARL is trained for each target problem.

	Noise Type	CE (Reproduced)	CE (JoCoR)	GCE [54]	SCE [45]	FW [33]	Bootstrap [37]	JoCoR [46]	Our ARL
MNIST	Sym-20%	81.21±0.53	79.56±0.44	97.64±0.65	89.50±0.44	96.85±0.67	76.18±0.98	**98.06±0.04**	97.90±0.12
	Sym-50%	59.51±0.70	52.66±0.43	94.14±1.32	67.38±0.53	94.25±0.43	51.53±1.56	96.64±0.12	**96.71±0.21**
	Sym-80%	22.43±1.21	23.43±0.31	40.57±0.72	31.23±0.89	54.01±1.82	23.46±0.46	84.89±4.55	**89.88±0.34**
	Asym-40%	78.73±1.16	79.00±0.28	81.94±1.22	79.87±0.78	90.14±0.67	78.31±2.34	95.24±0.10	**97.38±0.17**
CIFAR-100	Sym-20%	39.19±0.58	35.14±0.44	34.66±0.76	35.09±0.50	38.18±0.76	3.53±0.18	**53.01±0.04**	51.34±0.10
	Sym-50%	19.50±0.43	16.97±0.40	10.29±0.53	18.54±0.29	3.25±0.15	18.36±0.63	**43.49±0.46**	42.18±0.27
	Sym-80%	5.56±0.24	4.41±0.14	2.03±0.36	5.75±0.39	6.12±0.27	2.33±0.13	15.49±0.98	**20.20±0.42**
	Asym-40%	30.16±0.44	27.29±0.25	1.32±0.23	27.07±0.42	4.23±0.51	31.72±0.74	32.70±0.35	**36.01±0.39**
	Avg.Rank	5.25	6.13	5.62	5.00	4.38	6.63	1.63	**1.38**

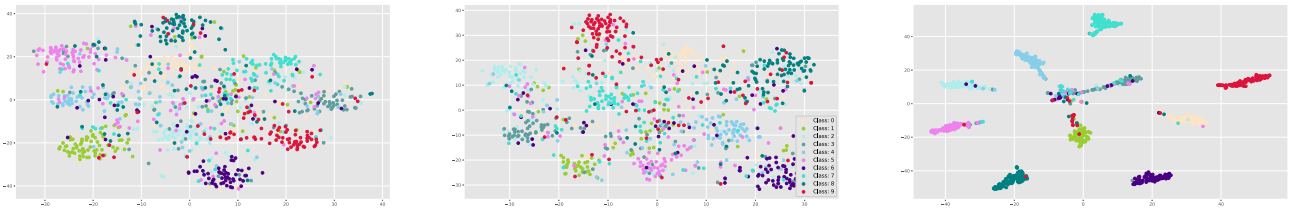

Figure 5. t-SNE visualisation of penultimate layer ResNet-18 features after learning on CIFAR-10 with 40% symmetric label noise. Left: CE training. Middle: Bootstrap training. Right: Our ARL training.

Dataset-specific loss learning Our main goal in this paper has been to learn a general purpose robust loss. In this section we examine an alternative use case of applying our framework to train a *dataset-specific* robust loss, in which case better performance could be achieved by customising the loss for the target problem. To achieve this, we now additionally assume a clean subset of data for the target problem is available (unlike the previous experiments, but similarly to several alternative methods in this area [46]) in order to drive loss learning. For this experiment we focus on comparison with JoCoR [46], since this is the current state-of-the-art model. We use the same medium sized CNN architecture as JoCoR for fair comparison, and train our loss to optimise the validation performance. From the results in Table 5, we can see that our ARL provides comparable or better performance than state of the art competitor JoCoR. However, this is now at significantly greater cost since the cost of data-specific loss training is not amortisable over multiple tasks as before.

Qualitative Analysis and Intuition of Learned Loss To gain some intuition about our loss' efficacy, we compare popular standard and robust losses in Figure 2. Comparing our ARL loss against alternatives, we conjecture that there are two properties that account for our label-noise robustness in practice: Feedback in response to perceived major prediction errors by the network, and the location of the minima where network predictions maximally satisfy the loss. In the case of a noisy labelled example that the network actually classifies correctly, (e.g., $y_{true} = 1$, $y_{label} = 0$, $y_{pred} \approx 1$), conventional CE aggressively "corrects" the network by reporting exponentially large loss. This aggressive feedback leads to fast training on clean data, but overfitting in noisy data [54]. Existing robust alternatives MAE [10] and GCE [54] are explicitly motivated by softening this aggressive "correction" compared to CE. Although not explicitly motivated by this, SCE also softens the feedback as shown in the figure. Meanwhile in terms of the minima that best satisfies the loss, conventional CE, as well as SCE, GCE and MAE lead to maximally confident predictions (minima at 0 or 1); which, if applied to a noisy label, leads to overfitting. In contrast, label smoothing [34, 45] improves robustness by inducing softer minima at $[0 + \epsilon, 1 - \epsilon]$ compared to the others' $[0, 1]$. However, LS issues the same aggressive correction of large errors as CE, and thus suffers from this accordingly. Only our ARL has learned to exploit both these strategies of less aggressive "corrections" and softer targets.

5. Conclusion

In this work, we took an AutoML perspective on the problem of noise robust loss function design. Our results reveal a new loss function that combines low-penalty and soft minima features to produce a noise-robust loss function. ARL provides a simple re-usable loss that can be plugged into diverse benchmarks and model architectures to learn robust features and classifiers in the presence of label noise, all without requiring a clean validation set or expensive meta-learning or distillation procedures.

References

[1] Ehsan Amid, Manfred K Warmuth, Rohan Anil, and Tomer Koren. Robust bi-tempered logistic loss based on bregman divergences. In *NeurIPS*, 2019.

[2] Ehsan Amid, Manfred K Warmuth, and Sriram Srinivasan. Two-temperature logistic regression based on the tsallis divergence. In *AISTATS*, 2019.

[3] Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. Metareg: Towards domain generalization using meta-regularization. In *NeurIPS*, 2018.

[4] Sarah Bechtle, Artem Molchanov, Yevgen Chebotar, Edward Grefenstette, Ludovic Righetti, Gaurav Sukhatme, and Franziska Meier. Meta-learning via learned loss. *ICPR*, 2020.

[5] Pengfei Chen, Junjie Ye, Guangyong Chen, Jingwei Zhao, and Pheng-Ann Heng. Robustness of accuracy metric and its inspirations in learning with noisy labels. In *AAAI*, 2021.

[6] Xinlei Chen and Abhinav Gupta. Webly supervised learning of convolutional networks. In *ICCV*, 2015.

[7] Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep learning for classical japanese literature. In *NeurIPS (Workshop)*, 2018.

[8] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(Jan):1–30, 2006.

[9] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019.

[10] Aritra Ghosh, Himanshu Kumar, and PS Sastry. Robust loss functions under label noise for deep neural networks. In *AAAI*, 2017.

[11] Santiago Gonzalez and Risto Miikkulainen. Improved training speed, accuracy, and data utilization through loss function optimization. In *CEC*, 2020.

[12] Santiago Gonzalez and Risto Miikkulainen. Optimizing loss functions through multi-variate taylor polynomial parameterization. In *GECCO*, 2021.

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

[14] Nikolaus Hansen and Andreas Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *CEC*, 1996.

[15] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[17] Timothy M Hospedales, Antreas Antoniou, Paul Micaelli, and Amos J. Storkey. Meta-Learning in Neural Networks: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[18] Rein Houthooft, Yuhua Chen, Phillip Isola, Bradly Stadie, Filip Wolski, OpenAI Jonathan Ho, and Pieter Abbeel. Evolved policy gradients. In *NeurIPS*, 2018.

[19] Chen Huang, Shuangfei Zhai, Walter Talbott, Miguel Angel Bautista, Shih-Yu Sun, Carlos Guestrin, and Josh Susskind. Addressing the loss-metric mismatch with adaptive loss alignment. In *ICML*, 2019.

[20] Jonathan J. Hull. A database for handwritten text recognition research. *IEEE Transactions on pattern analysis and machine intelligence*, 16(5):550–554, 1994.

[21] Frank Hutter, Lars Kotthoff, and J. Vanschoren, editors. *Automatic machine learning: methods, systems, challenges*. Challenges in Machine Learning. Springer, 2019.

[22] Simon Jenni and Paolo Favaro. Deep bilevel learning. In *ECCV*, 2018.

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

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[25] Louis Kirsch, Sjoerd van Steenkiste, and Juergen Schmidhuber. Improving generalization in meta reinforcement learning using learned objectives. In *ICLR*, 2020.

[26] Alex Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, 2009.

[27] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, et al. The open images dataset v4. *International Journal of Computer Vision*, 128(7):1956–1981, 2020.

[28] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[29] Junnan Li, Yongkang Wong, Qi Zhao, and Mohan S Kankanhalli. Learning to learn from noisy labeled data. In *CVPR*, 2019.

[30] Yiying Li, Yongxin Yang, Wei Zhou, and Timothy M Hospedales. Feature-critic networks for heterogeneous domain generalization. In *ICML*, 2019.

[31] Xingjun Ma, Hanxun Huang, Yisen Wang, Simone Romano, Sarah Erfani, and James Bailey. Normalized loss functions for deep learning with noisy labels. In *ICML*, 2020.

[32] Rafael Müller, Simon Kornblith, and Geoffrey E. Hinton. When does label smoothing help? In *NeurIPS*, 2019.

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

[34] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. In *ICLR*, 2017.

[35] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. In *ICLR (Workshop)*, 2018.

[36] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *AAAI*, 2019.

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

[38] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *ICML*, 2018.

[39] Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. Meta-weight-net: Learning an explicit mapping for sample weighting. In *NeurIPS*, 2019.

[40] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[41] Hwanjun Song, Minseok Kim, Dongmin Park, and Jae-Gil Lee. Learning from noisy labels with deep neural networks: A survey. *arXiv preprint arXiv:2007.08199*, 2020.

[42] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

[43] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019.

[44] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IROS*, 2017.

[45] Yisen Wang, Xingjun Ma, Zaiyi Chen, Yuan Luo, Jinfeng Yi, and James Bailey. Symmetric cross entropy for robust learning with noisy labels. In *CVPR*, 2019.

[46] Hongxin Wei, Lei Feng, Xiangyu Chen, and Bo An. Combating noisy labels by agreement: A joint training method with co-regularization. In *CVPR*, 2020.

[47] Olga Wichrowska, Niru Maheswaranathan, Matthew W Hoffman, Sergio Gomez Colmenarejo, Misha Denil, Nando de Freitas, and Jascha Sohl-Dickstein. Learned optimizers that scale and generalize. In *ICML*, 2017.

[48] Lijun Wu, Fei Tian, Yingce Xia, Yang Fan, Tao Qin, Lai Jian-Huang, and Tie-Yan Liu. Learning to teach with dynamic loss functions. In *NeurIPS*, 2018.

[49] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[50] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *CVPR*, 2015.

[51] Yilun Xu, Peng Cao, Yuqing Kong, and Yizhou Wang. L_dmi: A novel information-theoretic loss function for training deep nets robust to label noise. In *NeurIPS*, 2019.

[52] Quanming Yao, Hansi Yang, Bo Han, Gang Niu, and J Kwok. Searching to exploit memorization effect in learning with noisy labels. In *ICML*, 2020.

[53] Michael Zhang, James Lucas, Jimmy Ba, and Geoffrey E Hinton. Lookahead optimizer: k steps forward, 1 step back. In *NeurIPS*, 2019.

[54] Zhilu Zhang and Mert Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *NeurIPS*, 2018.