SEARCHING FOR ROBUSTNESS: LOSS LEARNING FOR NOISY CLASSIFICATION TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

We present a "learning to learn" approach for automatically constructing white-box classification loss functions that are robust to label noise in the training data. We paramaterize 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 dataset and architecture combinations. The resulting white-box loss provides a simple and fast "plug-and-play" module that enables effective noise-robust learning in diverse downstream tasks, without requiring a special training procedure or network architecture.

1 INTRODUCTION

The success of modern deep learning is largely 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 often because professional annotation is too costly to achieve at scale and users resort to less reliable crowd-sourcing, web-crawled incidental annotations (Chen & Gupta, 2015), or imperfect machine annotation (Kuznetsova et al., 2020); while in other situations the data is hard to classify reliably even by human experts, and label-noise is inevitable. These considerations have led to a large and rapidly progressing body of work focusing on developing noise-robust learning approaches (Ren et al., 2018; Han et al., 2018). Diverse solutions have been studied including those that modify the training algorithm through teacher-student (Jiang et al., 2018; Han et al., 2018) learning, or identify and down-weight noisy instances (Ren et al., 2018). Much simpler, and therefore more widely applicable, are attempts (Wang et al., 2019; Zhang & Sabuncu, 2018; Ghosh et al., 2017) to define noise-robust loss functions that provide drop-in replacements for standard losses such as cross-entropy. These studies hand engineer robust losses, motivated by different considerations including risk minimisation (Ghosh et al., 2017) and information theory (Xu et al., 2019). In this paper we explore an alternative data-driven approach (Hutter et al., 2019) to loss design, and search for a simple white-box function that provides a general-purpose noise-robust drop-in loss.

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 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 (Tobin et al., 2017) 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.

We apply our learned loss function to train various MLP and CNN architectures on several benchmarks including MNIST, 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 our learned loss and its efficacy compared to state-of-the-art in a variety of settings.

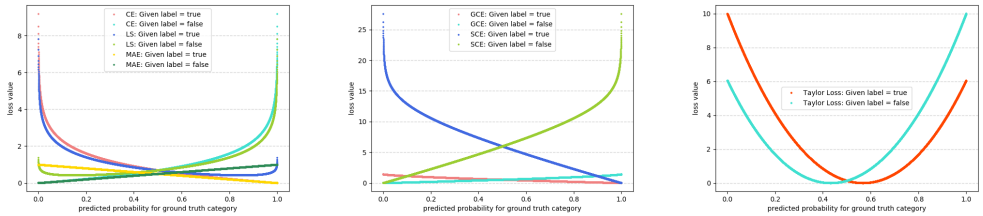

Figure 1: Existing hand-designed robust losses and our meta-learned robust loss. Left: Conventional Cross-Entropy (CE), Mean Absolute Error (MAE) (Ghosh et al., 2017), and label-smoothing (Pereyra et al., 2017). Middle: Generalised Cross Entropy (GCE) (Zhang & Sabuncu, 2018), Symmetric Cross Entropy (Wang et al., 2019). Right: Our learned loss.

2 METHOD

We aim to learn a loss function for multi-class classification problem that is robust to noisy labels in the training set. We consider the task of learning a loss function as a bilevel optimisation, where solutions generated by the upper objectives (in the outer loop) are conditioned on the response of the lower objectives (in the inner loop). In our loss function learning setting, the upper and lower objectives are defined, respectively, as optimising the parameters of an adaptive loss function and training neural networks, f_ω, with the learned loss function. The loss function parameters, θ, correspond to the coefficients of an n-th order polynomial, which can be viewed as a Taylor expansion of the ideal loss function. The bilevel optimisation problem is given by

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

where $\mathcal{M}(\cdot,\cdot)$ is a fitness function measuring network performance, and D is a random variable representing a domain, with D^{val} and D^{train} representing the validation and training sets respectively, and f is a neural network architecture. The performance of $f_{\omega^*_D}$, as measured by $\mathcal{M}$, reflects the quality of the supervision provided by the candidate loss function $\mathcal{L}_\theta$ on dataset D. During meta-learning, the training set and validation set are not identically distributed: the validation set contains clean labels, while the training set is assumed to have some form of label corruption. We use the Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) (Hansen & Ostermeier, 1996) to solve the upper layer problem, and standard stochastic gradient-based optimisation approaches to solve the lower level problems. A general overview of our algorithm for solving the optimisation problem in Equations 1 is described in Algorithm 1.

2.1 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 that places most of its mass on high quality solutions to the optimisation problem. One of the benefits of using CMA-ES is that this algorithm does not require the performance measurement 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, $\theta_i \in \Theta$, is evaluated according to

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

where $f^{(j)}_\omega$ and D_j are different network architectures and datasets, respectively.

Algorithm 1 Offline Taylor CMA-ES

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

2.2 Taylor Polynomial Representation

The space of potential loss functions in which CMA-ES searches is a crucial design parameter. From a practical point of view, we must limit ourselves to a space that can be parameterised by a small number of values. However, this must be balanced with the ability to represent a wide enough variety of functions that a good solution can be found. Moreover, by selecting a small space with a small number of free parameters and well-understood nonlinear form, it becomes possible to transfer the learned loss to new problems without having to retrain the network. The function space that we choose is the Taylor series approximations of all β-smooth functions, $\mathcal{L} : \mathbb{R}^m \to \mathbb{R}$,

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

where each $\nabla^n \mathcal{L}(\mathbf{x}_0)$ is the n-th order gradient of $\mathcal{L}$ 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. We then sum over the different possible classes,

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

where $\hat{\mathbf{y}}$ and $\mathbf{y}$ are the vectors of predicted probabilities and (possibly noisy) ground-truth labels, respectively. The result of performing the simplification is that the loss function can be used in a variety of settings with different numbers of classes, and we can fix $m = 2$. We found that $\beta = 4$ is a good trade-off between modelling capacity and meta-training efficiency. Note that $\nabla^n \mathcal{L}(\mathbf{x}_0)$ does not depend on $\mathbf{x}$, meaning these values can be computed during a meta-training period before regular training commences. As such, we can reinterpret the task of meta-learning $\mathcal{L}$ as inferring $\theta = \{\nabla^n \mathcal{L}(\mathbf{x}_0)\}_{n=0}^{\beta}$. The resulting loss function is represented as

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

The fixed point where the gradients are evaluated is also left as a learned parameter, (θ_0, θ_1). 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 paramaterized loss function (Li et al., 2019; Bechtle et al., 2019; Kirsch et al., 2020).

2.3 GENERALISATION ACROSS ARCHITECTURES

To enable the learned loss function to generalise to different architectures, we extend the strategy domain randomisation Tobin et al. (2017) to introduce the idea of evaluating the expected performance across a range of architectures during meta-learning. Specifically, we use a set, F, of m architectures 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 of the different architectures trained with the same loss.

2.4 GENERALISATION ACROSS DATASETS

We also explore another method for improving the generality of the loss function. To enable a loss function to be applied to an unseen dataset, the loss function should be exposed to several datasets during training so as not to overfit to the prediction distributions encountered for a specific machine learning problem. In our method a loss function sampled from the current target distribution is deployed 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 could perform both dataset and architecture randomisation simultaneously. However, due to the implied three-way Cartesian product, we found this computationally infeasible.

2.5 NORMALISATION

We make use of a normalisation approach to prevent the learned loss functions from exhibiting an arbitrary output range,

$$\hat{f} = \eta \frac{f - f_{min}}{f_{max} - f_{min}} \tag{5}$$

where f_{min} and f_{max} denotes the minimum and maximum and η is a hyperparameter deciding the dynamic range of the loss function. Both f_{min} and f_{max} are easily approximated by sampling random points satisfying $\{(\hat{\mathbf{y}}, \mathbf{y}) | \hat{\mathbf{y}}_i \geq 0, \sum_i \hat{\mathbf{y}}_i = 1; \mathbf{y}_i \in \{0, 1\}, \sum_i \mathbf{y} = 1\}$, which defines the domain of the loss function.

3 EXPERIMENTS

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

Datasets We use five datasets in our experiments: MNIST (LeCun & Cortes, 2010), CIFAR-10, CIFAR-100 (Krizhevsky, 2009), KMNIST (Clanuwat et al., 2018), and Clothing1M (Xiao et al., 2015). 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, and 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.

Noise types For loss learning, we consider simulating two types of noise, symmetric noise and asymmetric noise (pair-flip noise). 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.

Architectures We train and evaluate our learned loss with a range of neural networks from very shallow ones, including 2-layer MLP, 3-layer MLP, and 4-layer CNN, to deeper ones, such as VGG-11 (Simonyan & Zisserman, 2015) and Resnet-18 (He et al., 2016). We also use the medium-size architecture considered in (Wei et al., 2020), which we term JoCoR-Net (see Appendix A.3 for

Table 1: Accuracy (%) of different robust learning strategies. 80 % symmetric noise condition. Our loss trained under architecture randomization (AR) and dataset randomization (DR) conditions.

Architecture type dataset	2layer MLP MNIST	4layer CNN MNIST	vgg11 Cifar10	vgg11 Cifar100	Resnet18 Cifar10	Resnet18 Cifar100	Avg.Rank
CE	22.10±0.68	28.48±0.35	18.38±0.21	4.25±0.28	18.44±0.34	8.86±0.10	5.67
GCE	40.57±0.43	9.80±0.58	16.56±0.54	1.04±0.47	31.69±0.36	11.98±0.18	5.83
SCE	31.23±0.70	28.53±1.02	28.61±0.64	2.31±0.80	45.34±0.40	8.16±0.07	4.66
FW	54.01±0.89	80.34±1.21	16.97±0.44	1.41±0.07	10.15±0.68	1.16±0.04	5.83
Bootstrap	23.46±1.31	28.78±1.03	17.58±0.82	4.18±0.72	12.10±0.32	8.67±0.61	6.17
MAE	**85.40±3.39**	78.70±11.49	14.20±0.42	1.01±0.11	22.95±1.25	0.82±0.17	6.00
Label-smooth	24.31±1.25	27.02±0.48	17.74±0.46	4.47±0.12	17.67±0.35	7.66±1.52	6.17
Taylor Loss (AR)	34.27±0.34	37.08±0.44	**41.36±0.47**	5.63±0.24	29.50±0.30	**14.94±0.26**	2.66
Taylor Loss (DR)	48.57±0.11	**85.31±0.12**	31.12±0.23	5.04±0.14	**35.23±0.23**	13.36±0.63	**2.00**

Table 2: Accuracy (%) of different robust learning strategies. 40 % asymmetric noise condition. Our loss trained under architecture randomization (AR) and dataset randomization (DR) conditions.

Architecture type dataset	2layer MLP MNIST	4layer CNN MNIST	vgg11 Cifar10	vgg11 Cifar100	Resnet18 Cifar10	Resnet18 Cifar100	Avg.Rank
CE	78.73±1.16	84.01±0.34	56.43±0.12	30.20±0.18	58.69±0.43	44.14±0.15	4.16
GCE	81.94±1.22	9.80±0.10	56.42±0.54	22.39±0.35	57.90±0.31	40.76±0.24	6.00
SCE	79.87±0.78	84.09±0.62	78.23±0.55	25.33±0.73	63.22±0.22	40.90±0.37	3.33
FW	90.14±0.67	69.98±0.49	54.42±0.79	5.21±0.39	48.40±0.08	3.83±0.23	6.83
Bootstrap	78.31±2.34	83.68±1.27	57.69±0.11	**31.07±1.09**	57.69±0.76	**45.78±0.15**	4.33
MAE	71.61±4.50	69.91±0.49	49.06±0.22	0.96±0.10	55.67±3.05	1.02±0.14	8.33
Label-smooth	59.66±1.16	68.14±0.61	57.76±0.37	20.64±0.18	59.69±0.36	39.92±0.49	6.17
Taylor Loss (AR)	**97.16±0.20**	**96.88±0.66**	74.30±0.20	22.50±0.33	**86.70±0.12**	44.47±0.48	**2.00**
Taylor Loss (DR)	85.77±0.33	93.47±0.28	**79.09±0.51**	18.30±0.27	68.88±0.41	31.47±0.65	3.67

details). For a fair comparison, we train 2-layer MLP, 3-layer MLP, and 4-layer CNN with SGD optimiser and set the learning rate to 0.01 and momentum to 0.9. For the training of JoCor-Net, we apply the Adam optimiser (Kingma & Ba, 2015) and the learning rate is set to 0.001. When training Resnet-18 and VGG-11, we follow the training protocol in (Zhang et al., 2019).

Competitors We compare our learned loss functions 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 Ghosh et al. (2017). **GCE:** Zhang & Sabuncu (2018) analysed MAE as hard to train, and proposed generalised cross-entropy to provide the best of CE and MAE; **FW:** (Patrini et al., 2017) iteratively estimates the label noise transfer matrix, and trains the model corrected by the label noise estimate; **SCE:** Wang et al. (2019) argued that symmetrizing 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 (Reed et al., 2015). **LSR:** Label-smoothing is an effective general purpose regularizer (Pereyra et al., 2017; Szegedy et al., 2016; Müller et al., 2019) whose properties in promoting noise robustness have been studied (Wang et al., 2019).

3.1 Training a general-purpose robust loss function

Experimental setup We consider two domain generalisation protocols for training a general purpose loss function, namely architecture and dataset generalisation. In architecture generalisation, 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 dataset generalisation, we solely use the 4-layer CNN as the backbone build a dataset pool from MNIST, KMNIST, and CIFAR-10. We also consider two label-noise conditions during meta-learning: symmetric label noise with 80% noise, and asymmetric label noise with 40% noise. Losses are trained under each domain generalisation protocol, and each noise distribution, using the normalisation trick introduced in Section 2.5. After loss function learning we deploy the losses to train fresh models from scratch on a suite of evaluation tasks. For evaluation, we compare the accuracy at convergence, and summarise via the average ranks of each methods across different datasets and architectures (Demšar, 2006).

Table 3: 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.

CE	Bootstrap	GCE	FW	SCE	JoCoR*	Talyor (AR-A40)	Taylor (DR-A40)	Taylor (AR-S80)	Taylor (DR-S80)
66.88	67.28	66.63	68.33	67.63	69.79	69.14	**70.09**	68.85	69.34

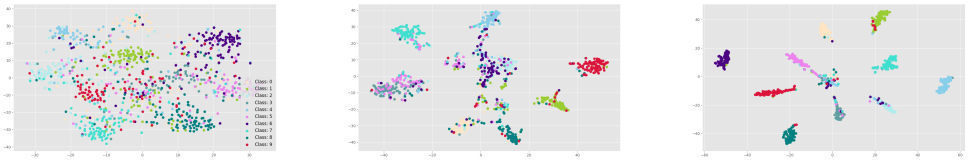

Figure 2: t-SNE visualisation of penultimate layer Resnet18 features after learning on CIFAR-10 with 40% symmetric label noise. Left: model trained by CE. Middle: model trained by FW. Right: model trained by Our learned loss.

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 learned losses perform favourably compared to hand-designed alternatives across a variety of benchmarks, with our learned loss providing a higher average rank than competitors in both experiments. However, there is no clear winner between architecture (AR) and dataset (DR) condition for meta-learning. We conjecture that best performance would be obtained by performing these simultaneously, 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.

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 setting for Resent-18 described in Wei et al. (2020) to apply our learned loss to the real-world Clothing1M noisy-label benchmark. As a real-world noisy-label problem we apply our model from the asymmetric-40% condition above. Note that neither Clothing1M, nor ResNet-18 were seen during meta-learning, above. We train with Adam optimiser with learning rate 8×10^{-4}, 5×10^{-4}, 5×10^{-4} for 5 epochs each in a sequence. We report the mean accuracy of each model after ten trials in Table 3. 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 and ours are simple plug-in robust losses applied to vanilla ResNet training. Nevertheless, our method obtains the highest performance.

3.2 Additional Analysis

Generalisation across noise-levels We trained our main 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 apply our 80%-symmetric loss on problems with a range of noise levels. From the results in Figure 3 we can see that our loss does tend to provide competitive performance across a range of operating points.

Qualitative analysis of representations We visualise the feature representation learned by our loss when applied to CIFAR-10 under 40% symmetric label noise in Figure 2. 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 intense degree of label noise.

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 (Wei et al., 2020)) in order to drive loss learning. For this experiment we focus

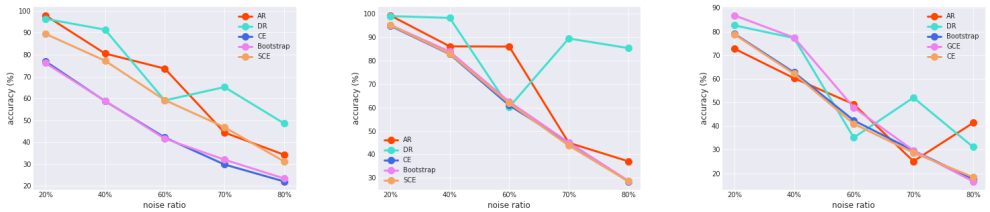

Figure 3: Generalisation of learned loss to varying noise-levels. Left: 2MLP-MNIST, Middle: 4CNN-MNIST, Right: VGG11-CIFAR10.

Table 4: Accuracy (%) of different robust learners. JoCoR net medium-sized CNN used throughout. Taylor Loss is trained specifically for the target problem.

	Noise Type	CE (ours)	CE (JoCoR)	GCE	SCE	FW	Bootstrap	JoCoR	Ours
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**
	Asy-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	Symm-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
	Symm-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
	Symm-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**

on comparison with JoCoR (Wei et al., 2020), since this is the current state-of-the-art model, and in their experiments medium sides networks are applied. We use the same medium sized CNN architecture as JoCoR for fair comparison, and train our loss to optimize the validation performance. From the results in Table 4, we can see that our method 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 amortizable over multiple tasks as before.

Qualitative Analysis and Intuition of Learned Loss To gain some intuition about our loss functions' efficacy, we compare popular standard and robust losses in Figure 1. Comparing our robust loss, and comparison with the alternatives, we conjecture that there are two properties that impact 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 can lead to fast training on clean data, but overfitting in noisy data (Zhang & Sabuncu, 2018). Existing robust alternatives MAE (Ghosh et al., 2017) and GCE (Zhang & Sabuncu, 2018) 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 (Pereyra et al., 2017; Wang et al., 2019) 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 Taylor loss has learned to exploit both these strategies of less aggressive "corrections" and softer targets.

4 Related Work

Label Noise Learning with label noise is now a large research area due to its practical importance. Song et al. (2020) present a detailed survey explaining the variety of approaches previously studied including designing noise robust neural network architectures (Chen & Gupta, 2015), regularisers such as label-smoothing (Szegedy et al., 2016; Pereyra et al., 2017), sample selection methods that attempt to filter out noisy samples – often by co-teaching or student teacher learning with multiple neural networks (Jiang et al., 2018; Han et al., 2018; Wei et al., 2020), various meta-learning

approaches that often aim to down-weight noisy samples using meta-gradients from a validation set (Ren et al., 2018; Shu et al., 2019), 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 use standard architectures and standard learning algorithms. Major robust losses include MAE, shown to be theoretically robust in Ghosh et al. (2017), but hard to train in Zhang & Sabuncu (2018); GCE which attempts to be robust yet easy to train (Zhang & Sabuncu, 2018), and symmetric cross-entropy (Wang et al., 2019). These losses are all hand-designed based on various motivations. Instead we take a data-driven AutoML approach and search for the a robust loss function. This draws upon meta-learning techniques but, differently from existing meta-robustness work, focuses on general-purpose white box loss discovery. Incidentally, we note that our resulting Taylor loss covers all six desiderata for noise-robust learning outlined in Song et al. (2020).

Meta-learning and Loss Learning Meta-learning, also known as learning to learn, has been applied for a wide variety of purposes as summarized in Hospedales et al. (2020). Of particular relevance is meta-learning of loss functions, which has been studied for various purposes including providing differentiable surrogates of non-differentiable objectives (Huang et al., 2019), optimizing efficiency and asymptotic performance of learning (Jenni & Favaro, 2018; Bechtle et al., 2019; Houthooft et al., 2018; Wu et al., 2018; Gonzalez & Miikkulainen, 2019; 2020), and improving robustness to train/test domain-shift (Balaji et al., 2018; Li et al., 2019). We are particularly interested in learning *white-box* losses for efficiency and improved task-transferability compared to neural network alternatives (Bechtle et al., 2019; Houthooft et al., 2018; Balaji et al., 2018; Li et al., 2019). Meta-learning of white-box learner components has been demonstrated for optimizers (Wichrowska et al., 2017), activation functions (Ramachandran et al., 2018) and losses for accelerating conventional supervised learning Gonzalez & Miikkulainen (2019; 2020). We are the first to demonstrate the value of automatic loss function discovery for general purpose label-noise robust learning.

5 Conclusion

In this work, we leverage CMA-ES to discover novel loss functions in the defined Taylor series function space. A framework is developed and enriched with two variations to enable the transferability and the generality of the loss function. In order to demonstrate the efficiency of the learned loss function, we deploy them on to a variance of tasks where the dataset and architecture are different to where it is trained and compete with recent works show the strength of our method empirically. In addition, we show that the proposed method is also able to produce well-behaved models trained with noisy data and these models outperform state-of-the-art models on a range of tasks.

REFERENCES

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

Sarah Bechtle, Artem Molchanov, Yevgen Chebotar, Edward Grefenstette, Ludovic Righetti, Gaurav Sukhatme, and Franziska Meier. Meta-learning via learned loss. *arXiv preprint arXiv:1906.05374*, 2019.

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

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

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.

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

Santiago Gonzalez and Risto Miikkulainen. Improved training speed, accuracy, and data utilization through loss function optimization. *arXiv preprint arXiv:1905.11528*, 2019.

Santiago Gonzalez and Risto Miikkulainen. Optimizing loss functions through multivariate taylor polynomial parameterization. *arXiv preprint arXiv:2002.00059*, 2020.

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.

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

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

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey, 2020.

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

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.

Frank Hutter, Lars Kotthoff, and J. Vanschoren (eds.). *Automatic machine learning: methods, systems, challenges*. Challenges in Machine Learning. Springer, 2019.

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

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.

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

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

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

Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Tom Duerig, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *IJCV*, 2020.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

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

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

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.

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

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

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.

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

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.

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

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.

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

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.

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.

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

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.

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.

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

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.

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

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

A APPENDIX

A.1 EXPERIMENT DATASET

Table 5: The datasets used in the experiments.

	number of training	number of test	number of class	image size
MNIST	60,000	10,000	10	28×28
KMNIST	60,000	10,000	10	28×28
CIFAR-10	50,000	10,000	10	32×32
CIFAR-100	50,000	10,000	100	32×32
Clothing1M	1,000,000	10,000	14	224×224

A.2 TAYLOR POLYNOMIAL

The 4^{th} order taylor series:

$$\begin{aligned}\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) =& \theta_2(\hat{\mathbf{y}}_i - \theta_0) + \frac{1}{2}\theta_3(\hat{\mathbf{y}}_i - \theta_0)^2 + \frac{1}{6}\theta_4(\hat{\mathbf{y}}_i - \theta_0)^3 + \frac{1}{24}\theta_5(\hat{\mathbf{y}}_i - \theta_0)^4 \\ &+ \theta_6(\hat{\mathbf{y}}_i - \theta_0)(\mathbf{y}_i - \theta_1) + \frac{1}{2}\theta_7(\hat{\mathbf{y}}_i - \theta_0)(\mathbf{y}_i - \theta_1)^2 + \frac{1}{2}\theta_8(\hat{\mathbf{y}}_i - \theta_0)^2(\mathbf{y}_i - \theta_1) \\ &+ \frac{1}{6}\theta_9(\hat{\mathbf{y}}_i - \theta_0)^3(\mathbf{y}_i - \theta_1) + \frac{1}{6}\theta_{10}(\hat{\mathbf{y}}_i - \theta_0)(\mathbf{y}_i - \theta_1)^3 + \frac{1}{4}\theta_{11}(\hat{\mathbf{y}}_i - \theta_0)^2(\mathbf{y}_i - \theta_1)^2 \\ &+ \theta_{12}(\mathbf{y}_i - \theta_1) + \frac{1}{2}\theta_{13}(\mathbf{y}_i - \theta_1)^2 + \frac{1}{6}\theta_{14}(\mathbf{y}_i - \theta_1)^3 + \frac{1}{24}\theta_{15}(\mathbf{y}_i - \theta_1)^4\end{aligned}$$

A.3 ARCHITECTURE OF NEURAL NETWORKS

The network architectures of the MLP and CNN models used in the experiments part.

Table 6: The architectures used in the experiments

2-Layer MLP	3-Layer MLP	4-Layer CNN	JoCoR-Net
28×28 Gray Image	28×28 Gray Image	32×32 RGB Image	32×32 RGB Image
FC $28 \times 28 \longrightarrow 256$, ReLU	FC $28 \times 28 \longrightarrow 256$, ReLU FC$256 \longrightarrow 256$, ReLU	5×5, 32, ReLU 5×5, 64, ReLU FC $1024 \longrightarrow 1024$, ReLU	3×3, 64 BN, ReLU 3×3, 64 BN, ReLU 2×2 Max-pool 3×3, 128 BN, ReLU 3×3, 128 BN, ReLU 2×2 Max-pool 3×3, 196 BN, ReLU 3×3, 196 BN, ReLU 8×8 Avg-pool
FC $256 \longrightarrow 10$	FC $256 \longrightarrow 10$	FC $1024 \longrightarrow 10$	FC $196 \longrightarrow 100$