Addressing the Loss-Metric Mismatch with Adaptive Loss Alignment

Chen Huang [1] Shuangfei Zhai [1] Walter Talbott [1] Miguel Angel Bautista [1] Shih-Yu Sun [1] Carlos Guestrin [1]
Josh Susskind [1]

Abstract

In most machine learning training paradigms a fixed, often handcrafted, loss function is assumed to be a good proxy for an underlying evaluation metric. In this work we assess this assumption by meta-learning an adaptive loss function to directly optimize the evaluation metric. We propose a sample efficient reinforcement learning approach for adapting the loss dynamically during training. We empirically show how this formulation improves performance by simultaneously optimizing the evaluation metric and smoothing the loss landscape. We verify our method in metric learning and classification scenarios, showing considerable improvements over the state-of-the-art on a diverse set of tasks. Importantly, our method is applicable to a wide range of loss functions and evaluation metrics. Furthermore, the learned policies are transferable across tasks and data, demonstrating the versatility of the method.

1. Introduction

Machine learning models, including deep neural networks, are difficult to optimize, particularly for real world performance. One critical reason is that default loss functions are not always good approximations to evaluation metrics, a phenomenon we term the *loss-metric mismatch* (see examples in Figure 1). In fact, loss functions are often designed to be differentiable, and preferably convex and smooth, whereas many evaluation metrics are not. This problem is particularly evident in tasks like metric learning where the evaluation metrics of interest include area under the ROC curve (AUC) or precision/recall rate. These evaluation metrics are non-decomposable (involving statistics of a set of examples), nonlinear and non-continuous w.r.t. the sample-wise predictions and labels. One intuitive remedy is to experiment with many losses and identify one that correlates most to the evaluation metric, but this is inefficient both in terms of computation and human effort to design new losses.

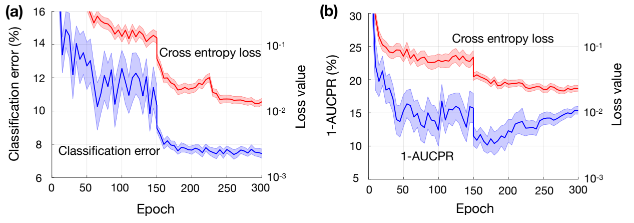

Figure 1. Loss-metric mismatch on CIFAR-10: (a) cross-entropy loss on the training set vs. classification error on the test set, (b) cross-entropy loss on the training set vs. area under the precision recall curve (AUCPR) on the test set. All curves are averaged over 10 runs initialized with different random seeds, with shaded areas showing the standard deviations. Note how training loss and evaluation metric curves differ in shape and noise characteristics, which is more apparent in the AUCPR case.

The ideal choice of a loss function should tightly approximate the metric across a wide range of parameter values. During model training, the distribution of predictions tend to be different early in training compared to later when close to convergence. For instance, the softmax outputs of a classification model usually change gradually from a near uniform distribution to a sharp multinomial distribution. This poses yet another challenge to the ability of the default loss function to approximate the metric, and indicates the potential benefit of choosing an adaptive loss function, where its parameters are dynamically adjusted to serve as a better surrogate to the evaluation metric.

In addition to the loss-metric mismatch, another difficulty lies in the potential difference between the distribution of the training set and that of the test set, which can be due to different levels of data noise or sampling biases (Ren et al., 2018). Optimizing with respect to the training set thus can exhibit different characteristics from the test set, which will lead to a *generalization gap*.

We address the loss-metric mismatch by introducing Adaptive Loss Alignment (ALA), a technique that automatically adjusts loss function parameters to directly optimize the evaluation metric on a validation set. We also find this results in an empirical reduction of the generalization gap. ALA learns to adjust the loss function using Reinforcement

[1]Apple Inc., Cupertino, United States. Correspondence to: Chen Huang <chen-huang@apple.com>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

Learning (RL) at the same time as the model weights are being learned by gradient descent. This helps to align the loss function to the evaluation metric cumulatively over successive training iterations.

We experiment with a variety of classification and metric learning problems. Results demonstrate significant gains of our approach over fixed loss functions and other meta-learning methods. Furthermore, ALA is generic and applies to a wide range of loss formulations and evaluation metrics. The learned loss control policy can be viewed as a data-driven curriculum that enjoys good transferability across tasks and data. In summary, the contributions of this work are as follows:

- We introduce ALA, a sample efficient RL approach to address the loss-metric mismatch directly by continuously adapting the loss.
- We provide RL formulations for metric learning and classification, with state of the art results relative to fixed loss and other meta-learning baselines.
- We show the versatility of ALA for a wide range of loss functions and evaluation metrics, and demonstrate its transferability across tasks and data.
- We empirically show through ablation studies that ALA improves optimization as well as generalization.

2. Related work

Improving model training through efficient experimentation is an active area of research. This includes classic hyperparameter optimization (Bergstra & Bengio, 2012; Snoek et al., 2012) and more recent gradient-based approaches (Maclaurin et al., 2015; Franceschi et al., 2017). These techniques overlap with architecture meta learning, such as activation functions (Ramachandran et al., 2017) and neural architecture search, *e.g.*, (Xie et al., 2019). In addition, certain design choices can aid in generalization such as the use of batch normalization (Ioffe & Szegedy, 2015; Santurkar et al., 2018). ALA focuses on the mechanics of the loss function, and can be used independently or in conjunction with other methods to improve performance.

A different strategy is to focus on the optimization process itself and the associated dynamics of learning. Curriculum learning aims to improve optimization by gradually increasing the difficulty of training (Bengio et al., 2009), which has been extended from the pre-defined case, *e.g.*, focal loss (Lin et al., 2017), to learning a data-driven curriculum (Jiang et al., 2018) or a data reweighting scheme (Ren et al., 2018), by optimizing a proxy loss. Similarly, optimizers can be learned in a data driven fashion using gradient descent, *e.g.*, (Andrychowicz et al., 2016; Wichrowska et al., 2017). In contrast, ALA is a more general method since it adapts the loss function, which allows it to find better optimization strategies to address the loss-metric mismatch.

Other methods focus on the loss function itself. These include approximate methods that bridge the gap between loss functions and evaluation metrics for special cases, *e.g.*, for ranking losses/area under the curve (Eban et al., 2017; Kar et al., 2014). Closest to our work are learning to teach methods that adapt loss functions to optimize target metrics. In (Xu et al., 2019), reinforcement learning is used to learn a discrete optimization schedule that alternates between different loss functions at different time-points, which is suited to multi-objective problems, but does not address more general non-discrete cases. In (Wu et al., 2018) and (Jenni & Favaro, 2018), gradient-based techniques are used to optimize differentiable proxies for the evaluation metric, whereas our method optimizes the metric directly.

ALA differs from the above techniques by optimizing the evaluation metric directly via a sample efficient RL policy that iteratively adjusts the loss function. Our work also extends beyond classification to the metric learning case.

3. Adaptive Loss Alignment (ALA)

We start by formally defining our learning problem, where we would like to improve an evaluation metric $\mathcal{M}(f_w, D_{val})$ on validation set D_{val}, for a parametric model $f_w : \mathcal{X} \rightarrow \mathcal{Y}$. The evaluation metric $\mathcal{M}$ between the ground-truth y and model prediction $f_w(x)$ given input x, can be either decomposable over samples (*e.g.*, classification error) or non-decomposable like area under the precision recall curve (AUCPR) and Recall@k. We learn to optimize for the validation metric and expect it to be a good indicator of the model performance $\mathcal{M}(f_w, D_{test})$ on test set D_{test}.

Optimizing directly for evaluation metrics is a challenging task. This is because the model weights w are actually obtained by optimizing a loss function l on training set D_{train}, *i.e.*, by solving $\min_w \sum_{(x,y)\in D_{train}} l(f_w(x), y)$. However, in many cases the loss l is only a surrogate of the evaluation metric $\mathcal{M}$, which can be non-differentiable w.r.t. w. Moreover, the loss l is optimized on the training set D_{train} instead of D_{val} or D_{test}.

To address the above *loss-metric mismatch* issue, we propose to learn an adaptive loss function $l_\Phi(f_w(x), y)$ with loss parameters $\Phi \in \mathbb{R}^u$. The goal is to align the adaptive loss with evaluation metric $\mathcal{M}$, on a held-out dataset D_{val}. This leads to an alternate direction optimization problem — (1) find metric-minimizing loss parameters Φ and (2) update the model weights w_Φ under the resultant loss l_Φ by, *e.g.*, Stochastic Gradient Descent (SGD). We have:

$$\begin{aligned} &\min_{\Phi} \quad \mathcal{M}(f_{w_\Phi}, D_{val}), \\ &s.t.\ w_\Phi = \arg\min_w \sum_{(x,y)\in D_{train}} l_\Phi(f_w(x), y), \end{aligned} \tag{1}$$

where in practice both the outer loop and inner loop are

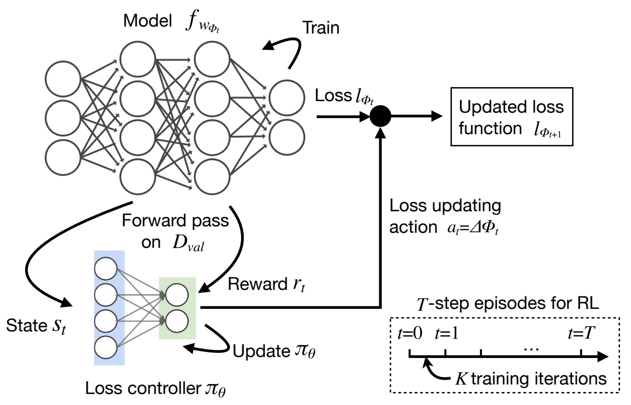

Figure 2. Reinforcement Learning (RL) of Adaptive Loss Alignment (ALA) controller. Model $f_{w_{\Phi_t}}$ is trained under loss function l_{Φ_t} with time-varying parameters Φ_t. Given the model state s_t at time t, the ALA controller proposes an action $a_t = \Delta\Phi_t$ to update the loss for follow-up training. The improvement in some evaluation metric (on validation set D_{val}) is used as reward r_t to update the policy π_θ. Our ALA controller is trained with one-step episodes (*i.e.*, $T = 1$) and is sample efficient.

approximated by a few steps of iterative optimization (*e.g.*, SGD updates). Hence, we denote Φ_t as the loss function parameters at time step t and w_{Φ_t} as the corresponding model parameters. The key here is to bridge the gap between evaluation metric $\mathcal{M}$ and loss function l_{Φ_t} over time, conditioned on the found local optimum w_{Φ_t}.

3.1. Reinforcement Learning of ALA

To capture the conditional relations between loss and evaluation metric, we formulate a reinforcement learning problem. The task is to predict the best change $\Delta\Phi_t$ to loss parameters Φ_t such that optimizing the adjusted loss aligns better with the evaluation metric $\mathcal{M}(f_{w_{\Phi_t}}, D_{val})$. In other words, taking an action that adjusts the loss function should produce a reward that reflects how much the metric $\mathcal{M}$ will improve on held-out data D_{val}. This is analogous to teaching the model how to better optimize on seen data D_{train} and to better generalize (in terms of $\mathcal{M}$) on unseen data D_{val}.

The underlying model behind Reinforcement Learning (RL) is a Markov Decision Process (MDP) defined by states $s \in \mathcal{S}$ and actions $a \in \mathcal{A}$ at discrete time steps t within an episode. In our case, an episode is naturally defined as a set of T consecutive time-steps, each consisting of K training iterations. In other words, the RL policy collects episodes at a slower timescale than the training of f_w. Figure 2 illustrates the schematic of our RL framework. Our state s_t records training progress information (*e.g.*, w_{Φ_t}-induced loss value), and the loss-updating action $a_t = \Delta\Phi_t$ is sampled from a stochastic policy $\pi_\theta(a_t|s_t)$. We implement the policy as a neural network parameterized by θ. Training f_w under the updated loss $l_{\Phi_{t+1}}$ will transition to a new state s_{t+1} and produce a reward signal $r_t = r(s_t, a_t)$. We define the reward by the improvement in evaluation metric $\mathcal{M}(f_{w_{\Phi_t}}, D_{val})$.

We optimize the loss-controlling policy with a policy gradient approach, similar to REINFORCE (Williams, 1992). The objective is to maximize the expected total return

$$J(\theta) = \mathbb{E}_\tau \left[R(\tau)\right], \tag{2}$$

where $R(\tau)$ is the total return $R(\tau) = \sum_{t=0}^{T} r_t$ of an episode $\tau = \{s_t, a_t | t \in [0, T]\}$. The updates to the policy parameters θ are given by the gradient

$$\nabla_\theta J(\theta) = \mathbb{E}_\tau \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \sum_{k=t}^{T} (r_k - b_k)\right], \tag{3}$$

where b_k is a variance-reducing baseline implemented as the exponential moving average of previous rewards.

Local policy learning. As shown in Figure 2, an episode for RL is a set of T consecutive time steps. Choosing one extreme, T could cover the entire length of a training run. However, waiting for the model to fully train and repeating this process enough times for a policy to converge requires a great deal of computational resources. We choose the other extreme and use episodes of a single step, *i.e.*, $T = 1$ (still composed of K training iterations). One advantage of doing so is that a single training run of f_w can contribute many episodes to the training of π_θ, making it sample efficient. Although this ignores longer-term effects of chosen actions, we will show empirically that these one-step episodes are sufficient to learn competent policies, and increasing T does not convey much benefit in our experiments (see Supplementary Materials). The one-step RL setting is similar to a contextual bandit formulation, although actions affect future states. Thus, unlike bandit formulations, the ALA controller can learn state-transition dynamics.

3.2. Learning Algorithm

In this section, we will describe the concrete RL algorithm for simultaneous learning of the ALA policy parameters θ and model weights w.

Rewards: Our reward function r_t measures the relative reduction in validation metric $\mathcal{M}(f_{w_{\Phi_t}}, D_{val})$, after K gradient descent iterations with an updated loss function $l_{\Phi_{t+1}}$. To represent the cumulative performance between model updates, we define:

$$\mathcal{M}_{t+1} = \sum_{j=1}^{K} \gamma^{K-j} \mathcal{M}(f_{w^j}, D_{val}), \tag{4}$$

where γ is a discount factor that weighs more on the recent metric. The main model weights w^j are updated for K

iterations. Then we quantize the reward r_t to ± 1 as follows:

$$r_t = \text{sign}(\mathcal{M}_t - \mathcal{M}_{t+1}), \tag{5}$$

which encourages continuous error metric decreases regardless of magnitude (but until the maximum training iteration).

Action space: For every element $\Phi_t(i)$ of the loss parameters Φ_t, we sample action $a_t(i)$ from the discretized space $\mathcal{A} = \{-\beta, 0, \beta\}$, with β being a predefined step-size. Actions will update the loss parameters $\Phi_t(i) = \Phi_{t-1}(i) + a_t(i)$ at each time step. Our policy network π_θ has $|\mathcal{A}|$ output neurons for each loss parameter, and a_t is sampled from a softmax distribution over these neurons.

State space: Our policy network state $s_t \in \mathcal{S}$ consists of four components:

1. Some task dependent validation statistics $S(f_{w_{\Phi_t}}, D_{val})$, *e.g.*, the log probabilities of different classes, observed at multiple time-steps $\{t, t-1, \dots\}$.
2. The relative change $\Delta S(f_{w_{\Phi_t}}, D_{val})$ of validation statistics from their moving average.
3. The current loss parameters Φ_t.
4. The current iteration number normalized by the total iteration number of the full training run of f_w.

Here we use the validation statistics, among others, to capture model training states. Recall our goal is to find rewarding loss-updating actions $a_t = \Delta\Phi_t$ to improve the evaluation metric on validation set. A successful loss control policy would be able to model the implicit relation between the validation statistics, which is the state of the RL problem, and validation metric, which is the reward. In other words, ALA should learn to mimic the loss optimization process for decreasing the validation metric cumulatively. We choose to use validation statistics instead of training statistics in s_t because the former is a natural proxy of the validation metric. Note the validation statistics are normalized in our state representation. This allows for generic policy learning which is independent of the actual model predictions from different tasks or loss formulations.

Algorithm: Algorithm 1 shows how our RL algorithm alternates between updating the loss control policy π_θ and updating model weights w. Model weights are updated via mini-batch SGD on D_{train}, while policy π_θ is updated every K SGD iterations on D_{val}. We enhance policy learning by training in parallel multiple main networks, which we refer to as child models. Each child model is initialized with random weights w but sharing the loss controller. At each time-step t for policy update, we collect the episodes using π_θ from all the child models. This independent set of episodes, together with replay memory $\mathcal{D}$ that adds randomness to the learning trajectories, help to alleviate the non-stationarity in online policy learning. As a result, more robust policies are learned, as verified in our experiments.

Algorithm 1 Reinforcement Learning for ALA

```
Initialize each child model with random weights $w$
Initialize loss controller $\pi_\theta$ with random weights $\theta$
Initialize loss parameters $\Phi_0$ properly for a given task
Initialize replay memory $\mathcal{D}$
while not converged do
  for each state $s_t$ do
    Sample action $a_t \sim \pi_\theta(a_t|s_t)$
    Take action $a_t$ to update loss function to $l_{\Phi_{t+1}}$
    Update $w$ by $K$ SGD iterations with $l_{\Phi_{t+1}}$
    Collect reward $r_t$ (Equation 5) and new state $s_{t+1}$
    Store $\langle s_t, a_t, r_t, s_{t+1} \rangle$ from all child models in $\mathcal{D}$
    Sample random experiences from $\mathcal{D}$
    Update $\theta$ to maximize reward via Equation 3
  end for
end while
```

It is worth noting that the initial loss parameters Φ_0 are important for efficient policy learning. Proper initialization of Φ_0 must ensure default loss function properties that depend on the particular form of loss parameterization for a given task, *e.g.*, identity class correlation matrix in classification (see Section 4).

4. Instantiation in Typical Learning Problems

Classification: We learn to adapt the parametric classification loss function introduced in (Wu et al., 2018):

$$l_{\Phi_t}(f_w(x), y) = -\sigma(y^{\mathrm{T}} \Phi_t \log f_w(y|x)), \tag{6}$$

where $\sigma(\cdot)$ is the sigmoid function, and $\Phi_t \in \mathbb{R}^{|\mathcal{Y}| \times |\mathcal{Y}|}$ denotes the loss function parameters with $|\mathcal{Y}|$ being the number of classes. $y \in \{0,1\}^{|\mathcal{Y}|}$ denotes the one-hot representation of class labels, and $f_w(y|x) \in \mathbb{R}^{|\mathcal{Y}|}$ denotes the multinomial model output. This adaptive loss function is a generalization of the cross-entropy loss $l_{ce}(f_w(x), y) = -y^{\mathrm{T}} \log f_w(y|x)$ where Φ_t is fixed as the identity matrix.

The matrix Φ_t encodes time-varying class correlations. A positive value of $\Phi_t(i,j)$ encourages the model to increase the prediction probability of class j given ground-truth class i. A negative value of $\Phi_t(i,j)$, on the other hand, penalizes the confusion between class i and j. Thus when Φ_t changes as learning progresses, it is possible to implement a *hierarchical curriculum* for classification, where similar classes are grouped as a super class earlier in training, and discriminated later as training goes further along. To learn the curriculum automatically, we initialize Φ_0 as an identity matrix (reduced to the standard cross-entropy loss in this case), and update Φ_t over time by the ALA policy π_θ.

To learn to update Φ_t, we first construct a confusion matrix

$C(f_{w_{\Phi_t}}, D_{val}) \in \mathbb{R}^{|\mathcal{Y}|\times|\mathcal{Y}|}$ of model prediction on validation set D_{val}, and define

$$C_{i,j} = \frac{\sum_{d=1}^{|D_{val}|} -I(y_d, i) \log f^j_{w_{\Phi_t}}(x_d)}{\sum_{d=1}^{|D_{val}|} I(y_d, i)}, \qquad (7)$$

where $I(y_d, i)$ is an indicator function outputing 1 when y_d equals class i and 0 otherwise.

We take a parameter efficient approach to update each loss parameter $\Phi_t(i, j)$ based on the observed class confusions $[C_{i,j}, C_{j,i}]$. In other words, the ALA loss controller collects the validation statistics $S(f_{w_{\Phi_t}}, D_{val})$ only for class pairs at each time step t, in order to construct state s_t for updating the corresponding loss parameter $\Phi_t(i, j)$ (see Figure S2 in Supplementary Materials). Different class pairs share the same controller, and we update $\Phi_t(i, j)$ and $\Phi_t(j, i)$ to the same value (normalized between $[-1, 1]$) to ensure class symmetry. This implementation is much more parameter efficient than learning to update the whole matrix Φ_t based on C. Furthermore, it does not depend on the number of classes for a given task, thus enabling us to transfer the learned policy to another classification task with an arbitrary number of classes (see Section 6.3).

Metric Learning: A metric learning problem learns a distance metric to encode semantic similarity. Typically, the resulting distance metric cannot directly cater to different, and sometimes contradicting performance metrics of interest (*e.g.*, verification vs. identification rate, or precision vs. recall). This gap is more pronounced in the presence of common techniques like hard mining that only have indirect effects on final performance. Therefore, metric learning can serve as a strong testbed for learning methods that directly optimize evaluation metrics.

The standard triplet loss (Schroff et al., 2015) for metric learning can be formulated as:

$$l_{tri}(f_w(x_{i,i^+,i^-})) = \max\left(0, F(d^+) - F(d^-) + \eta\right), \quad (8)$$

where $F(d) = d^2$ is the squared distance function over both d^+ (distance between anchor instance $f_w(x_i)$ and positive instance $f_w(x_{i^+})$) and d^- (distance between $f_w(x_i)$ and negative instance $f_w(x_{i^-})$), while η is a margin parameter.

As (Wu et al., 2017) pointed out, the shape of distance function matters. The concave shape of $-d^2$ for negative distance d^- will lead to diminishing gradients when d^- approaches zero. Here we propose to reshape the distance function adaptively with two types of loss parameterizations.

For the first parametric loss, called **Distance mixture**, we adopt 5 different-shaped distance functions for both d^+ and d^-, and learn an linear combination of them via Φ_t:

$$l_{\Phi_t} = \sum_{i=1}^{5} \Phi_t(i) F_i^+(d^+) + \sum_{i=1}^{5} \Phi_t(i+5) F_i^-(d^-), \quad (9)$$

where $F_i^+(\cdot)$ and $F_i^-(\cdot)$ correspond to the increasing and decreasing distance functions to penalize large d^+ and small d^- respectively. In this case $\Phi_t = \{\Phi_t(i)\} \in [0, 1]^{10}$, and is initialized as a binary vector such that Φ_0 selects the default distance functions d^2 and $0.5d^{-1}$ for d^+ and d^-. For RL, the validation statistics in state s_t are simply represented by the computed distance $F_i^+(\cdot)$ or $F_i^-(\cdot)$, and our ALA controller updates $\Phi_t(i)$ accordingly. Supplementary Materials specify the forms of $F_i^+(\cdot)$ and $F_i^-(\cdot)$ and show that final performance is not very sensitive to their design choices. It is more important to learn their dynamic weightings.

Similar to the focal loss (Lin et al., 2017), we also introduce a **Focal weighting**-based loss formulation as follows:

$$l_{\Phi_t} = \frac{1}{\Phi_t(1)} \log\left[1 + \sum_{i+} \exp\left(\Phi_t(1) \cdot (d^+_{i^+} - \alpha)\right)\right] + \frac{1}{\Phi_t(2)} \log\left[1 + \sum_{i-} \exp\left(-\Phi_t(2) \cdot (d^-_{i^-} - \alpha)\right)\right], \quad (10)$$

where $\Phi_t \in \mathbb{R}^2$, and $d^+_{i^+}$ and $d^-_{i^-}$ denote the distance between anchor instance and the positive i^+ and negative i^- instances in the batch. We use these distances as validation statistics for ALA controller to update Φ_t. While α here is the distance offset.

5. Implementation Details

The main model architecture and maximum number of training iterations are task-dependent. In the parallel training scenario for our experiments, 10 child models were trained sharing the same ALA controller. The controller is instantiated as an MLP consisting of 2 hidden layers each with 32 ReLU units. Our state s_t includes a sequence of validation statistics observed from past 10 time-steps. Table S4 in Supplementary Materials quantifies these choices.

The ALA controller is learned using the REINFORCE policy gradient method (which worked better than Q-learning variants in early experiments). We use a learning rate of 0.001 for policy learning. Training episodes are collected from all child networks every $K = 200$ gradient descent iterations. We set the discount factor $\gamma = 0.9$ (Equation 4), loss parameter updating step $\beta = 0.1$ and distance offset $\alpha = 1$ (Equation 10), but found that performance is robust to variations of these hyperparameters.

6. Results

We evaluate the proposed ALA method on classification and metric learning tasks. Our ALA controller is learned by multi-model training for both tasks unless otherwise stated.

6.1. Classification

We train and evaluate ALA using two different evaluation metrics to demonstrate generality: (1) classification error, and (2) area under the precision recall curve (AUCPR). We experiment on CIFAR-10 (Krizhevsky, 2009) with 50k images for training and 10k images for testing. For training a loss controller, we divide the training set randomly into a new training set of 40k images and a validation set of 10k images. We use Momentum-SGD for training. The compared methods below use the full 50k training images and their optimal hyper-parameters.

Optimizing classification error: Table 1 reports classification errors using the popular network architectures: ResNet-32 (He et al., 2016), Wide-ResNet (WRN) (Zagoruyko & Komodakis, 2016) and DenseNet (Huang et al., 2017). The self-paced method (Kumar et al., 2010) is a predefined curriculum learning scheme based on example hardness. L-Softmax (Liu et al., 2016) is a strong hand-designed loss function. The recent state-of-the-art L2T-DLF (Wu et al., 2018) method adapts a loss function under the same formulation of Equation 6. However, the objective of L2T-DLF is to minimize an evaluation metric surrogate that is gradient-friendly. ALA outperforms L2T-DLF using single network training, and improves further with multi-network training (default). The competitive performance of our single network RL training validates its sample and time efficiency. Other ALA baselines train with either random loss parameters Φ_t (identity matrix perturbed by 10% noise), or use the prediction confusion matrix (Equation 7) as Φ_t. These baselines do not perform well, while ALA can *learn* meaningful loss parameters to align with evaluation metrics. The L2T method (Fan et al., 2018) similarly uses RL to optimize evaluation metrics, but requires 50 episodes of full training. In contrast, our ALA controller is trained while the main model is being trained, thereby allowing for over 50$\times$ faster learning with even stronger error reduction.

Optimizing AUCPR: Unlike classification error, AUCPR is a highly-structured and non-decomposable evaluation metric. To demonstrate our ability to optimize different metrics, we change the reward for ALA to AUCPR as defined in (Eban et al., 2017). Table 2 shows our method achieves the AUCPR metric of 94.9% (10-run average) on CIFAR-10, outperforming the SGD-based optimization method (Eban et al., 2017). Our advantages are also evident over methods that do not optimize the metric of interest, *e.g.*, via the pairwise AUCROC surrogate (Rakotomamonjy, 2004) and cross-entropy loss that is a proxy for classification accuracy.

6.2. Metric Learning

To validate ALA on metric learning tasks, we perform image retrieval experiments on the Stanford Online Products (SOP) dataset (Song et al., 2016), and face recognition (FR) experiments on the LFW dataset (Huang et al., 2007). The SOP dataset contains 120,053 images of 22,634 categories. The first 10,000 and 1,318 categories are used for training and validation, and the remaining are used for testing. We optimize for the evaluation metric of average Recall@k on SOP. For LFW, we train for the verification accuracy and test on the LFW verification benchmark containing 6,000 verification pairs. Note both the average Recall@k and verification accuracy are non-decomposable evaluation metrics.

Table 1. Classification error (%) on CIFAR-10 dataset. 10-run average and standard deviation are reported for ALA.

Method	ResNet-32	WRN	DenseNet
cross-entropy	7.51	3.80	3.54
Self-paced (Kumar et al., 2010)	7.47	3.84	3.50
L-Softmax (Liu et al., 2016)	7.01	3.69	3.37
L2T (Fan et al., 2018)	7.10	-	-
L2T-DLF (Wu et al., 2018)	6.95	3.42	3.08
ALA (random matrix Φ_t)	8.23$\pm$0.41	4.69$\pm$0.28	4.15$\pm$0.33
ALA (confusion matrix Φ_t)	7.42$\pm$0.04	3.74$\pm$0.02	3.55$\pm$0.02
ALA (single-network)	6.85$\pm$0.09	3.39$\pm$0.04	3.03$\pm$0.04
ALA (multi-network)	**6.79$\pm$0.07**	**3.34$\pm$0.04**	**3.01$\pm$0.02**

Table 2. AUCPR (%) on CIFAR-10 dataset. All methods use the same model architecture as adopted in (Eban et al., 2017). 10-run average and standard deviation are reported as ALA result.

Method	AUCPR
cross-entropy loss for optimizing accuracy	84.6
Pairwise AUCROC loss (Rakotomamonjy, 2004)	94.2
AUCPR loss (Eban et al., 2017)	94.2
ALA	**94.9$\pm$0.14**

Table 3 (top two cells) compares ALA to recent methods on the SOP dataset. ALA is applied to two representative metric learning frameworks—Triplet (Schroff et al., 2015) and Margin (Wu et al., 2017), using the respective network architectures. The two works differ in the loss formulation and data sampling strategies. We can see that ALA leads to consistently significant gains at all recall levels for both frameworks. Margin+ALA outperforms recent embedding ensembles based on boosting (BIER) and attention (ABE-8) mechanisms, as well as HTL using a hand-designed hierarchical class-tree. This indicates the advantages of our adaptive loss function and direct metric optimization. On LFW, ALA improves the triplet framework with distance mixtures, and achieves state-of-the-art verification accuracy 99.57% under the small training data protocol. Please refer to the Supplementary Materials for comparisons to recent strong FR methods on LFW.

6.3. Transfer Learning

One potential disadvantage of hand-designed loss functions compared to learned ones is that the former may be specific to task and/or data, whereas the latter are usually generic

Table 3. Recall(%)@k on Stanford Online Products dataset.

k	1	10	100	1000
Triplet (Schroff et al., 2015)	66.7	82.4	91.9	-
Margin (Wu et al., 2017)	72.7	86.2	93.8	98.0
BIER (Opitz et al., 2017)	72.7	86.5	94.0	98.0
HTL (Ge et al., 2018)	74.8	88.3	94.8	98.4
ABE-8 (Kim et al., 2018)	76.3	88.4	94.8	98.2
Triplet + ALA (Distance mixture)	75.7	89.4	95.3	98.6
Margin + ALA (Distance mixture)	**78.9**	**90.7**	**96.5**	**98.9**
Margin + ALA (Focal weighting)	77.9	90.1	95.8	98.7
Margin + ALA (FR policy transfer)	75.2	89.2	94.9	98.4

Table 4. Policy transfer from CIFAR-10 to ImageNet. Top-1 and Top-5 accuracy rates (%) are reported on ImageNet.

Method	Top-1	Top-5
RMSProp	73.5	91.5
PowerSign-cd (Bello et al., 2017)	73.9	91.9
RMSProp + ALA (CIFAR policy transfer)	74.3	92.1
RMSProp + ALA	**74.6**	**92.6**

and widely applicable. Here we test the degree to which ALA extracts general knowledge about how to adapt the loss. We do so by conducting policy transfer experiments on classification and metric learning tasks in the following.

For classification, we transfer the learned policy that updates pairwise class correlations based on their prediction confusions. Specifically, we train an ImageNet (Deng et al., 2009) classifier with RMSProp optimizer, but using the fixed ALA loss policy learned from CIFAR-10 (with DenseNet). Despite the difference between number of classes and input distribution, the policy transfer is straightforward thanks to the weight sharing design of the policy network, We compare with PowerSign-cd (Bello et al., 2017) that transfers a learned policy *for optimization* from CIFAR-10 to ImageNet. Table 4 compares all methods using identical NASNet-A network architectures. It shows that the transferred ALA policy outperforms the baselines without careful tuning, and is comparable to the ImageNet-tuned RMSProp+ALA. These results show that ALA is able to learn a policy on small, quick-to-train datasets in order to guide large-scale training, which is desirable for fast and efficient learning. Notably, we use a different optimizer—SGD—for training the ALA policy on CIFAR-10, which shows that ALA is robust to the optimizer paired with the loss policy.

We also show that policy transferability holds in the metric learning case. Here we transfer the ALA policy learned from FR experiments to guide the training on SOP for image retrieval. The bottom row of Table 3 shows that the transferred policy is competitive with specialized learning on the target data domain. This demonstrates the superior generalization ability of ALA.

Table 5. Training and testing error (%) of ResNet-32 on CIFAR-10. We report 10-run average and standard deviation results for ALA.

Method	Training	Testing
cross-entropy	0.32	7.51
ALA	**0.12±0.02**	**6.79±0.07**
ALA-2nd run (using policy from 1st run)	0.14±0.01	6.72±0.04
ALA-2nd run (policy finetuning)	**0.08±0.01**	**6.72±0.02**

6.4. Analysis

Ablation studies: (i) Training vs. testing metric: Table 5 (top cell) compares ALA with fixed cross-entropy loss in both training and testing classification errors. Results suggest that through dynamic loss-metric alignment, ALA improves both optimization on the training data and generalization on test data. **(ii) Time efficiency:** Table 5 (bottom cell) compares our default online policy learning against continuing to train with ALA for a second run (thus halving the learning efficiency). The compared baselines either reuse the learned policy in the 1st run or finetune the policy during the 2nd run. These baselines lead to marginal gains due to more learning episodes. On the other hand, online ALA suffices to learn competent policies. **(iii) Robustness to different loss parameterizations:** Table 3 (penultimate row) shows that ALA works similarly well when using a focal weighted loss function for the metric learning task. We conjecture that ALA can work with minimal tuning across a variety of parameterized loss formulations.

Insights on optimization vs. generalization: Figure 3 analyzes the effects of ALA policies on optimization and generalization behavior by switching the reward signal in a 2-factor design: training vs. validation data and loss vs. evaluation metric on CIFAR-10. When the training loss (*i.e.*, the default cross-entropy loss) or metric is used as reward, we compute them on the whole training set D_{train}. For this and the following studies, we train with the ResNet-32 architecture.

Figure 3(a) isolates the optimization effects by comparing ALA, with the cross-entropy loss as the reward, and minimizing cross-entropy loss directly. ALA is shown to encourage better optimization with consistently lower loss values. We note that this is a remarkable result as it shows that ALA is facilitating optimization even when the objective is fully differentiable. Figure 3(b) examines both optimization and generalization when ALA uses validation loss as a reward where we monitor training and test (generalization) errors for ALA and fixed cross-entropy loss. We observe that the generalization error on test set is indeed improved, and the optimization in training error also sees small improvements. The optimization and generalization gains are larger when the validation metric is used as reward, which further addresses the loss-metric mismatch. By comparison, the training metric-based reward yields faster training error

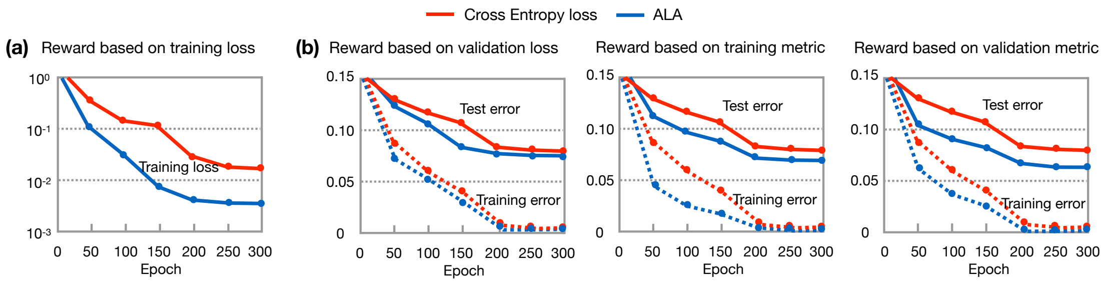

Figure 3. Analysis of optimization vs. generalization on CIFAR-10. (a) Comparing optimization performance in terms of the raw cross-entropy loss outputs on training data: Here ALA is rewarded by the training loss, and we observe that the measured training loss is consistently lower compared to the fixed cross-entropy loss, indicating improved optimization. (b) Comparing ALA policies trained with different rewards: the validation loss-based reward improves both optimization (training error) and generalization (test error), and the gains are larger when using the validation metric-based reward. In contrast, using the training metric as the reward yields smaller gains in test error, potentially due to the diminishing reward (error approaching zero) in training data.

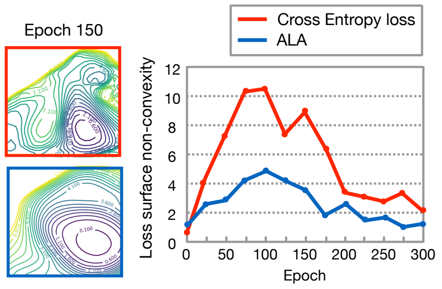

Figure 4. Analysis of the loss surface non-convexity on CIFAR-10: for both the fixed cross entropy loss and ALA, we compute the loss surface along filter-normalized 2D directions as in (Li et al., 2018) at different training epochs. Non-convexity is measured as the average Gaussian curvature for all points in the 2D loss surface.

decrease, but smaller gains in test error potentially due to the diminishing error/reward in training data.

Reasoning behind the improved optimization by ALA: First, we speculate that ALA improves optimization by dynamically smoothing the loss landscape. We verify this hypothesis by taking measurements of the loss surface convexity, calculating the Gaussian curvature of the loss surface around each model checkpoint following (Li et al., 2018). Figure 4 shows a smoother loss surface from the model trained with ALA. This confirms that ALA is learning to manipulate the loss surface in a way that improves convergence of SGD based optimization processes, in agreement with findings in (Li et al., 2018).

Second, we study how ALA improves performance by addressing the loss-metric mismatch. Figure 5 shows CIFAR-10 training with ALA using (a) classification error and (b) AUCPR metrics. We use the validation metric-based reward for ALA by default, and follow the aforementioned training settings for each metric. We can see that the fixed cross-entropy loss, without explicit loss-metric alignment, suffers from an undesirable mismatch between the monitored validation loss and test metric in both cases (see the different

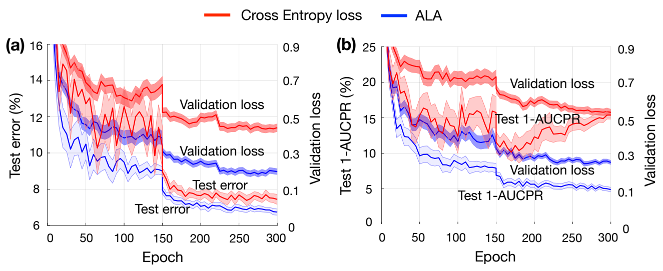

Figure 5. Validation loss vs. test metric of (a) classification error and (b) AUCPR on CIFAR-10. Curves are means over 10 runs initialized with different random seeds, and shaded areas show standard deviations. ALA uses the default reward based on the validation error metric, and improves both validation loss and test metric by addressing the loss-metric mismatch (both before and after the loss/metric drop due to learning rate change).

curve shapes and variances). ALA reduces the loss-metric mismatch by sequentially aligning the loss to the evaluation metric, even though classification error and AUCPR cases exhibit different patterns. In doing so, ALA not only lowers the validation loss but also the generalization test error.

7. Conclusion

We introduced ALA to make it easier to improve model performance on task-specific evaluation metrics. Performance is often hindered by the loss-metric mismatch, and we showed that ALA overcomes this by sequentially aligning the loss to the metric. We demonstrated significant gains over existing methods on classification and metric learning, using an efficient method that is simple to tailor, which makes ALA useful for automated machine learning. Intriguingly, ALA improves optimization and generalization simultaneously, in contrast to methods that focus on one or the other. We leave theoretical understanding of the effectiveness of loss function adaptation to future work.

Acknowledgements

The authors want to thank (in alphabetical order) Leon Gatys, Kelsey Ho, Qi Shan, Feng Tang, Karla Vega, Russ Webb and many others at Apple for helpful discussions during the course of this project. In addition, we are grateful to Harry Guo, Myra Haggerty, Jerremy Holland and John Giannandrea for supporting the research effort. We also thank the ICML reviewers for providing useful feedback.

References

Andrychowicz, M., Denil, M., Gómez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and de Freitas, N. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems 29*, pp. 3981–3989, 2016.

Bello, I., Zoph, B., Vasudevan, V., and Le, Q. V. Neural optimizer search with reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 459–468, 2017.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *Proceedings of the 26th International Conference on Machine Learning*, pp. 41–48, 2009.

Bergstra, J. and Bengio, Y. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition*, 2009.

Eban, E., Schain, M., Mackey, A., Gordon, A., Rifkin, R., and Elidan, G. Scalable Learning of Non-Decomposable Objectives. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pp. 832–840, 2017.

Fan, Y., Tian, F., Qin, T., Li, X.-Y., and Liu, T.-Y. Learning to teach. In *International Conference on Learning Representations*, 2018.

Franceschi, L., Donini, M., Frasconi, P., and Pontil, M. Forward and reverse gradient-based hyperparameter optimization. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 1165–1173, 2017.

Ge, W., Huang, W., Dong, D., and Scott, M. R. Deep metric learning with hierarchical triplet loss. In *The European Conference on Computer Vision (ECCV)*, 2018.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Huang, G. B., Ramesh, M., Berg, T., and Learned-Miller, E. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, University of Massachusetts, Amherst, 2007.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 448–456, 2015.

Jenni, S. and Favaro, P. Deep bilevel learning. In *The European Conference on Computer Vision (ECCV)*, 2018.

Jiang, L., Zhou, Z., Leung, T., Li, L., and Fei-Fei, L. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 2309–2318, 2018.

Kar, P., Narasimhan, H., and Jain, P. Online and stochastic gradient methods for non-decomposable loss functions. In *Advances in Neural Information Processing Systems 27*, pp. 694–702, 2014.

Kim, W., Goyal, B., Chawla, K., Lee, J., and Kwon, K. Attention-based ensemble for deep metric learning. In *The European Conference on Computer Vision (ECCV)*, pp. 760–777, 2018.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Kumar, M. P., Packer, B., and Koller, D. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems 23*, pp. 1189–1197. 2010.

Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. Visualizing the loss landscape of neural nets. In *Neural Information Processing Systems*, 2018.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. Focal Loss for Dense Object Detection. In *International Conference on Computer Vision (ICCV)*, 2017.

Liu, W., Wen, Y., Yu, Z., and Yang, M. Large-margin softmax loss for convolutional neural networks. In *Proceedings of The 33rd International Conference on Machine Learning*, pp. 507–516, 2016.

Maclaurin, D., Duvenaud, D., and Adams, R. P. Gradient-based hyperparameter optimization through reversible learning. In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 2113–2122, 2015.

Opitz, M., Waltner, G., Possegger, H., and Bischof, H. BIER boosting independent embeddings robustly. In *International Conference on Computer Vision (ICCV)*, 2017.

Rakotomamonjy, A. Optimizing area under roc curve with SVMs. In *ROCAI*, 2004.

Ramachandran, P., Zoph, B., and Le, Q. V. Searching for activation functions. *CoRR*, abs/1710.05941, 2017.

Ren, M., Zeng, W., Yang, B., and Urtasun, R. Learning to reweight examples for robust deep learning. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 4331–4340, 2018.

Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A. How does batch normalization help optimization? In *Advances in Neural Information Processing Systems 31*, pp. 2488–2498, 2018.

Schroff, F., Kalenichenko, D., and Philbin, J. Facenet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 815–823, 2015.

Snoek, J., Larochelle, H., and Adams, R. P. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pp. 2951–2959, 2012.

Song, H. O., Xiang, Y., Jegelka, S., and Savarese, S. Deep metric learning via lifted structured feature embedding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4004–4012, 2016.

Wichrowska, O., Maheswaranathan, N., Hoffman, M. W., Colmenarejo, S. G., Denil, M., de Freitas, N., and Sohl-Dickstein, J. Learned optimizers that scale and generalize. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 3751–3760, 2017.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

Wu, C.-Y., Manmatha, R., Smola, A. J., and Krähenbühl, P. Sampling matters in deep embedding learning. In *International Conference on Computer Vision*, 2017.

Wu, L., Tian, F., Xia, Y., Fan, Y., Qin, T., Jian-Huang, L., and Liu, T.-Y. Learning to teach with dynamic loss functions. In *Advances in Neural Information Processing Systems 31*, pp. 6467–6478, 2018.

Xie, S., Zheng, H., Liu, C., and Lin, L. SNAS: stochastic neural architecture search. In *International Conference on Learning Representations*, 2019.

Xu, H., Zhang, H., Hu, Z., Liang, X., Salakhutdinov, R., and Xing, E. Autoloss: Learning discrete schedule for alternate optimization. In *International Conference on Learning Representations*, 2019.

Zagoruyko, S. and Komodakis, N. Wide residual networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2016.