Deep Bilevel Learning

Simon Jenni(✉) and Paolo Favaro

University of Bern, Bern, Switzerland
{jenni,favaro}@inf.unibe.ch

Abstract. We present a novel regularization approach to train neural networks that enjoys better generalization and test error than standard stochastic gradient descent. Our approach is based on the principles of cross-validation, where a validation set is used to limit the model overfitting. We formulate such principles as a bilevel optimization problem. This formulation allows us to define the optimization of a cost on the validation set subject to another optimization on the training set. The overfitting is controlled by introducing weights on each mini-batch in the training set and by choosing their values so that they minimize the error on the validation set. In practice, these weights define mini-batch learning rates in a gradient descent update equation that favor gradients with better generalization capabilities. Because of its simplicity, this approach can be integrated with other regularization methods and training schemes. We evaluate extensively our proposed algorithm on several neural network architectures and datasets, and find that it consistently improves the generalization of the model, especially when labels are noisy.

Keywords: Bilevel optimization · Regularization · Generalization
Neural networks · Noisy labels

1 Introduction

A core objective in machine learning is to build models that generalize well, *i.e.*, that have the ability to perform well on new unseen data. A common strategy to achieve generalization is to employ regularization, which is a way to incorporate additional information about the space of suitable models. This, in principle, prevents the estimated model from overfitting the training data. However, recent work [36] shows that current regularization methods applied to neural networks do not work according to conventional wisdom. In fact, it has been shown that neural networks can learn to map data samples to arbitrary labels despite using regularization techniques such as weight decay, dropout, and data augmentation. While the lone model architecture of a neural network seems to have an implicit regularizing effect [33], experiments show that it can overfit on any dataset, given enough training time. This poses a limitation to the performance of any trained neural network, especially when labels are partially noisy.

In this paper we introduce a novel learning framework that reduces overfitting by formulating training as a *bilevel optimization* problem [5,6]. Although

V. Ferrari et al. (Eds.): ECCV 2018, LNCS 11214, pp. 632–648, 2018.
https://doi.org/10.1007/978-3-030-01249-6_38

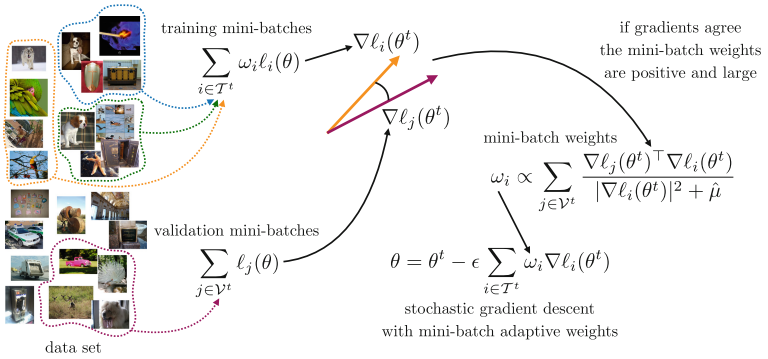

Fig. 1. The training procedure of our bilevel formulation. At each iteration we sample mini-batches from the data set, which we split into a validation and a training set. The validation is used to define the weights of the loss gradient used in the stochastic gradient descent to update the model parameters. If the gradients of the training set and those of the validation set agree, then the weights are large and positive. Vice versa, if they disagree the weights might be zero or negative.

the mathematical formulation of bilevel optimization is often involved, our final algorithm is a quite straightforward modification of the current training methods. Bilevel optimization differs from the conventional one in that one of the constraints is also an optimization problem. The main objective function is called the *upper-level* optimization task and the optimization problem in the set of constraints is called the *lower-level* optimization task. In our formulation, the lower-level problem is a model parameter optimization on samples from the *training set*, while the upper-level problem works as a performance evaluation on samples from a separate *validation set*. The optimal model is thus the one that is trained on one dataset, but performs well on a different one, a property that closely follows the definition of generalization.

In the optimization procedure we introduce a scalar weight for each sample mini-batch. The purpose of these variables is to find the linear combination of a subset of mini-batches from the training set that can best approximate the validation set error. They can also be seen as a way to: (1) discard noisy samples and (2) adjust the parameter optimization path. Finally, these weights can also be interpreted as hyper-parameters. Hence, bilevel optimization can be seen as an integrated way to continuously optimize for both the model parameters and the hyper-parameters as done in cross-validation.

In its general form, bilevel optimization is known to present computational challenges. To address these challenges, we propose to approximate the loss objectives at every iteration with quadratic functions. These approximations result in closed-form solutions that resemble the well-known stochastic gradient descent (SGD) update rules. Essentially, our bilevel optimization computes loss gradients on the training set and then prescribes adjustments to the learning rates of the SGD iteration so that the updated parameters perform well on the

validation set. As we will show later, these adjustments depend on how well the gradients computed on the training set "agree" with the gradients computed on the validation set (see Fig. 1).

Our method can be easily integrated in current training procedures for neural networks and our experiments show that it yields models with better generalization on several network architectures and datasets.

2 Prior Work

We give an overview of prior work relating to three main aspects of the paper: (1) Generalization properties of deep networks and how learning algorithms affect them, (2) memorization of corrupt labels as a special case of overfitting and (3) bilevel optimization in the context of deep learning. Parts of the techniques in our approach can be found also in other work, but with different uses and purposes. Therefore, we do not discuss these cases. For instance, Lopez and Ranzato [20] also use the dot-product between training gradients, but apply it to the context of continual learning with multiple tasks.

Understanding Generalization in Deep Learning. Although convolutional neural networks trained using stochastic gradient descent generalize well in practice, Zhang *et al.* [36] experimentally demonstrate that these models are able to fit random labelings of the training data. This is true even when using common explicit regularization techniques. Several recent works provide possible explanations for the apparent paradox of good generalization despite the high capacity of the models. The work of Kawaguchi *et al.* [16] provides an explanation based on model-selection (*e.g.*, network architecture) via cross-validation. Their theoretical analysis also results in new generalization bounds and regularization strategies. Zhang *et al.* [37] attribute the generalization properties of convolutional neural networks (CNNs) to characteristics of the stochastic gradient descent optimizers. Their results show that SGD favors flat minima, which in turn correspond to large (geometrical) margin classifiers. Smith and Le [29] provide an explanation by evaluating the Bayesian evidence in favor of each model, which penalizes sharp minima. In contrast, we argue that current training schemes for neural networks can avoid overfitting altogether by exploiting cross-validation during the optimization.

Combating Memorization of Noisy Labels. The memorization of corrupted labels is a form of overfitting that is of practical importance since labels are often unreliable. Several works have therefore addressed the problem of learning with noisy labels. Rolnick *et al.* [28] show that neural networks can be robust to even high levels of noise provided good hyper-parameter choices. They specifically demonstrate that larger batch sizes are beneficial in the case of label noise. Patriani *et al.* [25] address label noise with a loss correction approach. Nataranjan *et al.* [22] provide a theoretical study of the binary classification problem under the presence of label noise and provide approaches to modify the loss accordingly. Jindal and Chen [15] use dropout and augment networks with a

softmax layer that models the label noise and is trained jointly with the network. Sukhabar *et al.* [31] introduce an extra noise layer into the network that adapts the network output to match the noisy label distribution. Reed *et al.* [27] tackle the problem by augmenting the classification objective with a notion of consistency given similar percepts. Besides approaches that explicitly model the noise distribution, several regularization techniques have proven effective in this scenario. The recent work of Jiang *et al.* [14] introduce a regularization technique to counter label noise. They train a network (MentorNet) to assign weights to each training example. Another recent regularization technique was introduced by Zhang *et al.* [38]. Their method is a form of data augmentation where two training examples are mixed (both images and labels) in a convex combination. Azadi *et al.* [2] propose a regularization technique based on overlapping group norms. Their regularizer demonstrates good performance, but relies on features trained on correctly labeled data. Our method differs from the above, because we avoid memorization by encouraging only model parameter updates that reduce errors on shared sample patterns, rather than example-specific details.

Bilevel Optimization. Bilevel optimization approaches have been proposed by various authors to solve for hyper-parameters with respect to the performance on a validation set [3,4]. Domke [8] introduced a truncated bilevel optimization method where the lower-level is approximated by running an iterative algorithm for a given number of steps and subsequently computing the gradient on the validation loss via algorithmic differentiation. Our method uses the limiting case of using a single step in the lower-level problem. Ochs *et al.* [24] introduce a similar technique to the case of non-smooth lower-level problems by differentiating the iterations of a primal-dual algorithm. Maclaurin *et al.* [21] address the issue of expensive caching required for this kind of optimization by deriving an algorithm to exactly reverse SGD while storing only a minimal amount of information. Kunish *et al.* [19] apply bilevel optimization to learn parameters of a variational image denoising model. We do not use bilevel optimization to solve for existing hyper-parameters, but rather introduce and solve for new hyper-parameters by assigning weights to stochastic gradient samples at each iteration.

Meta Learning. Our proposed algorithm has some similarity to the meta-learning literature [10,23,34]. Most notably, the MAML algorithm by Finn *et al.* [10] also incorporates gradient information of two datasets, but does so in different ways: Their method uses second order derivatives, whereas we only use first-order derivatives. In general, the purpose and data of our approach is quite different to the meta-learning setting: We have only one task while in meta-learning there are multiple tasks.

3 Learning to Generalize

We are given m sample pairs $(x^{(k)}, y^{(k)})_{k=1,\dots,m}$, where $x^{(k)} \in \mathcal{X}$ represents input data and $y^{(k)} \in \mathcal{Y}$ represents targets/labels. We denote with $\phi_\theta : \mathcal{X} \mapsto \mathcal{Y}$ a

model that depends on parameters $\theta \in \mathbb{R}^d$ for some positive integer d. In all our experiments this model is a neural network and θ collects all its parameters. To measure the performance of the model, we introduce a loss function $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$ per sample. Since we evaluate the loss $\mathcal{L}$ on b mini-batches $\mathcal{B}_i \subset \{1, \dots, m\}$, $i = 1, \dots, b$, where $\mathcal{B}_i \cap \mathcal{B}_j = \emptyset$ for $i \neq j$, we redefine the loss as

$$\ell_i(\theta) \triangleq \sum_{k \in \mathcal{B}_i} \mathcal{L}\left(\phi_\theta\left(x^{(k)}\right), y^{(k)}\right). \tag{1}$$

At every iteration, we collect a subset of the mini-batches $\mathcal{U}^t \subset \{1, \dots, b\}$, which we partition into two separate sets: one for training $\mathcal{T}^t \subset \mathcal{U}^t$ and one for validation $\mathcal{V}^t \subset \mathcal{U}^t$, where $\mathcal{T}^t \cap \mathcal{V}^t = \emptyset$ and $\mathcal{T}^t \cup \mathcal{V}^t = \mathcal{U}^t$. Thus, mini-batches $\mathcal{B}_i$ in the training set have $i \in \mathcal{T}^t$ and those in the validation set have $i \in \mathcal{V}^t$. In all our experiments, the validation set $\mathcal{V}^t$ is always a singleton (one mini-batch).

3.1 Bilevel Learning

At the t-th iteration, Stochastic Gradient Descent (SGD) uses only one mini-batch to update the parameters via

$$\theta^{t+1} = \theta^t - \hat{\epsilon} \nabla \ell_i(\theta^t), \tag{2}$$

where $\hat{\epsilon} > 0$ is the SGD learning rate and $i \in \mathcal{U}^t$. Instead, we consider the subset $\mathcal{T}^t \subset \mathcal{U}^t$ of mini-batches and look for the linear combination of the losses that best approximates the validation error. We introduce an additional coefficient ω_i per mini-batch in $\mathcal{T}^t$, which we estimate during training. Our task is then to find parameters θ of our model by using exclusively mini-batches in the training set $\mathcal{T}^t \subset \mathcal{U}^t$, and to identify coefficients (hyper-parameters) ω_i so that the model performs well on the validation set $\mathcal{V}^t \subset \mathcal{U}^t$. We thus propose to optimize

$$\begin{aligned} \hat{\theta}, \hat{\boldsymbol{\omega}} = \arg\min_{\theta, \boldsymbol{\omega}} \; & \textstyle\sum_{j \in \mathcal{V}^t} \ell_j(\theta(\boldsymbol{\omega})) + \frac{\mu}{2} |\boldsymbol{\omega}|^2 \\ \text{subj. to} \; & \theta(\boldsymbol{\omega}) = \arg\min_{\bar{\theta}} \textstyle\sum_{i \in \mathcal{T}^t} \omega_i \ell_i(\bar{\theta}) \\ & |\boldsymbol{\omega}|_1 = 1, \end{aligned} \tag{3}$$

where $\boldsymbol{\omega}$ is the vector collecting all ω_i, $i \in \mathcal{T}^t$ and $\mu > 0$ is a parameter to regulate the distribution of the weights (large values would encourage a uniform distribution across mini-batches and small values would allow more sparsity). Notice that the solution of the lower-level problem does not change if we multiply all the coefficients ω_i by the same strictly positive constant. Therefore, to fix the magnitude of $\boldsymbol{\omega}$ we introduced the L^1 normalization constraint $|\boldsymbol{\omega}|_1 = 1$.

A classical method to solve the above bilevel problem is to solve a linear system in the second order derivatives of the lower-level problem, the so-called *implicit differentiation* [8]. This step leads to solving a very high-dimensional linear system. To avoid these computational challenges, in the next section we introduce a proximal approximation. Notice that when we compare the bilevel formulation (3) with SGD in the experiments, we equalize computational complexity by using the same number of visits per sample.

3.2 A Proximal Formulation

To simplify the bilevel formulation (3) we propose to solve a sequence of approximated problems. The parameters estimated at the t-th approximated problem are denoted θ^{t+1}. Both the upper-level and the lower-level problems are approximated via a first-order Taylor expansion of the loss function based on the previous parameter estimate θ^t, *i.e.*, we let

$$\ell_i(\theta) \simeq \ell_i(\theta^t) + \nabla \ell_i(\theta^t)^\top (\theta - \theta^t). \tag{4}$$

Since the above Taylor expansion holds only in the proximity of the previous parameter estimates θ^t, we also introduce *proximal quadratic* terms $|\theta - \theta^t|^2$. By plugging the linear approximation (4) and the proximal terms in Problem (3) we obtain the following formulation

$$\begin{aligned} \theta^{t+1}, \hat{\boldsymbol{\omega}} = \arg\min_{\theta, \boldsymbol{\omega}} & \textstyle\sum_{j \in \mathcal{V}^t} \ell_j(\theta^t) + \nabla \ell_j(\theta^t)^\top (\theta(\boldsymbol{\omega}) - \theta^t) + \frac{|\theta(\boldsymbol{\omega}) - \theta^t|^2}{2\lambda} + \frac{\mu}{2} |\boldsymbol{\omega}|^2 \\ \text{s.t.} \quad & \theta(\boldsymbol{\omega}) = \arg\min_{\bar{\theta}} \textstyle\sum_{i \in \mathcal{T}^t} \omega_i \left[\ell_i(\theta^t) + \nabla \ell_i(\theta^t)^\top (\bar{\theta} - \theta^t) \right] + \frac{|\bar{\theta} - \theta^t|^2}{2\epsilon} \\ & |\boldsymbol{\omega}|_1 = 1, \end{aligned} \tag{5}$$

where the coefficients $\lambda, \epsilon > 0$. The lower-level problem is now quadratic and can be solved in closed-form. This yields an update rule identical to the SGD step (2) when $\omega_i = 1$

$$\theta(\boldsymbol{\omega}) = \theta^t - \epsilon \textstyle\sum_{i \in \mathcal{T}^t} \omega_i \nabla \ell_i(\theta^t). \tag{6}$$

Now we can plug this solution in the upper-level problem and obtain

$$\begin{aligned} \hat{\boldsymbol{\omega}} = \arg\min_{\theta, \boldsymbol{\omega}} & \textstyle\sum_{j \in \mathcal{V}^t, i \in \mathcal{T}^t} -\omega_i \nabla \ell_j(\theta^t)^\top \nabla \ell_i(\theta^t) + \frac{\left| \sum_{i \in \mathcal{T}^t} \omega_i \nabla \ell_i(\theta^t) \right|^2}{2\lambda/\epsilon} + \frac{\mu}{2\epsilon} |\boldsymbol{\omega}|^2 \\ \text{s.t.} \quad & |\boldsymbol{\omega}|_1 = 1. \end{aligned} \tag{7}$$

We simplify the notation by introducing $\hat{\lambda} = \lambda/\epsilon$ and $\hat{\mu} = \mu/\epsilon$. To find the optimal coefficients $\boldsymbol{\omega}$ we temporarily ignore the normalization constraint $|\boldsymbol{\omega}|_1 = 1$ and simply solve the unconstrained optimization. Afterwards, we enforce the L^1 normalization to the solution. As a first step, we compute the derivative of the cost functional with respect to w_i and set it to zero, *i.e.*, $\forall i \in \mathcal{T}^t$

$$0 = \textstyle\sum_{j \in \mathcal{V}^t} -\nabla \ell_j(\theta^t)^\top \nabla \ell_i(\theta^t) + \frac{1}{\hat{\lambda}} \sum_{k \in \mathcal{T}^t} \omega_k \nabla \ell_k(\theta^t)^\top \nabla \ell_i(\theta^t) + \hat{\mu} \omega_i. \tag{8}$$

We now approximate the second sum by ignoring all terms such that $k \neq i$, *i.e.*,

$$0 = \textstyle\sum_{j \in \mathcal{V}^t} -\nabla \ell_j(\theta^t)^\top \nabla \ell_i(\theta^t) + \left(\frac{1}{\hat{\lambda}} |\nabla \ell_i(\theta^t)|^2 + \hat{\mu} \right) \omega_i \tag{9}$$

so that we can obtain the weight update rule

$$\forall i \in \mathcal{T}^t, \quad \omega_i \leftarrow \textstyle\sum_{j \in \mathcal{V}^t} \frac{\nabla \ell_j(\theta^t)^\top \nabla \ell_i(\theta^t)}{|\nabla \ell_i(\theta^t)|^2 / \hat{\lambda} + \hat{\mu}}, \qquad \hat{\boldsymbol{\omega}} = \omega / |\omega|_1. \tag{10}$$

Since Eq. (8) describes a linear system, it could be solved exactly via several iterative methods, such as Gauss-Seidel or successive over-relaxations [12]. However,

we found that using this level of accuracy does not give a substantial improvement in the model performance to justify the additional computational cost. We can then combine the update rule (10) with the update (6) of the parameters θ and obtain a new gradient descent step

$$\theta(\boldsymbol{\omega}) = \theta^t - \epsilon \sum_{i \in \mathcal{T}^t} \hat{\omega}_i \nabla \ell_i(\theta^t). \tag{11}$$

Notice that $\epsilon\hat{\omega}_i$ can be seen as a learning rate specific to each mini-batch. The update rule for the weights follows a very intuitive scheme: if the gradients of a mini-batch in the training set $\nabla \ell_i(\theta^t)$ agree with the gradients of a mini-batch in the validation set $\nabla \ell_j(\theta^t)$, then their inner product $\nabla \ell_j(\theta^t)^\top \nabla \ell_i(\theta^t) > 0$ and their corresponding weights are also positive and large. This means that we encourage updates of the parameters that also minimize the upper-level problem. When these two gradients disagree, that is, if they are orthogonal $\nabla \ell_j(\theta^t)^\top \nabla \ell_i(\theta^t) = 0$ or in the opposite directions $\nabla \ell_j(\theta^t)^\top \nabla \ell_i(\theta^t) < 0$, then the corresponding weights are also set to zero or a negative value, respectively (see Fig. 1 for a general overview of the training procedure). Moreover, these inner products are scaled by the gradient magnitude of mini-batches from the training set and division by zero is avoided when $\mu > 0$.

Remark 1. Attention must be paid to the sample composition in each mini-batch, since we aim to approximate the validation error with a linear combination of a few mini-batches. In fact, if samples in a mini-batch of the training set are quite independent from samples in mini-batches of the validation set (for example, they belong to very different categories in a classification problem), then their inner product will tend to be very small on average. This would not allow any progress in the estimation of the parameters θ. At each iteration we ensure that samples in each mini-batch from the training set have overlapping labels with samples in mini-batches from the validation set.

4 Implementation

To implement our method we modify SGD with momentum [26]. First, at each iteration t we sample k mini-batches $\mathcal{B}_i$ in such a way that the distributions of labels across the k mini-batches are identical (in the experiments, we consider $k \in \{2, 4, 8, 16, 32\}$). Next, we compute the gradients $\nabla \ell_i(\theta^t)$ of the loss function on each mini-batch $\mathcal{B}_i$. $\mathcal{V}^t$ contains only the index of one mini-batch and $\mathcal{T}^t$ all the remaining indices. We then use $\nabla \ell_j(\theta^t)$, $j \in \mathcal{V}^t$, as the *single* validation gradient and compute the weights ω_i of $\nabla \ell_i(\theta^t)$, $i \in \mathcal{T}^t$, using Eq. (10). The re-weighted gradient $\sum_{i \in \mathcal{T}^t} \omega_i \nabla \ell_i(\theta^t)$ is then fed to the neural network optimizer.

5 Experiments

We perform extensive experiments on several common datasets used for training image classifiers. Section 5.1 shows ablations to verify several design choices. In

Sects. 5.2 and 5.3 we follow the experimental setup of Zhang *et al.* [36] to demonstrate that our method reduces sample memorization and improves performance on noisy labels at test time. In Sect. 5.4 we show improvements on small datasets. The datasets considered in this section are the following:

CIFAR-10 [17]**:** It contains 50K training and 10K test images of size 32×32 pixels, equally distributed among 10 classes.
CIFAR-100 [17]**:** It contains 50K training and 10K test images of size 32×32 pixels, equally distributed among 100 classes.
Pascal VOC 2007 [9]**:** It contains 5,011 training and 4,952 test images (the `trainval` set) of 20 object classes.
ImageNet [7]**:** It is a large dataset containing 1.28M training images of objects from 1K classes. We test on the validation set, which has 50K images.

We evaluate our method on several network architectures. On Pascal VOC and ImageNet we use AlexNet [18]. Following Zhang *et al.* [36] we use CifarNet (an AlexNet-style network) and a small Inception architecture adapted to the smaller image sizes of CIFAR-10 and CIFAR-100. We refer the reader to [36] for a detailed description of those architectures. We also train variants of the ResNet architecture [13] to compare to other methods.

5.1 Ablations

We perform extensive ablation experiments on CIFAR-10 using the CifarNet and Inception network. The networks are trained on both clean labels and labels with 50% random noise. We report classification accuracy on the training labels (clean or noisy) and the accuracy on the *clean* test labels. The baseline in all the ablation experiments compares 8 mini-batches and uses $\mu = 0.01$ and $\lambda = 1$. Both networks have a single dropout layer and the baseline configuration uses the same dropping in all the compared mini-batches. The networks are trained for 200 epochs on mini-batches of size 128. We do not use data augmentation for CifarNet, but we use standard augmentations for the Inception network (*i.e.*, random cropping and perturbation of brightness and contrast). The case of the Inception network is therefore closer to the common setup for training neural networks and the absence of augmentation in the case of CifarNet makes overfitting more likely. We use SGD with momentum of 0.9 and an initial learning rate of 0.01 in the case of CifarNet and 0.1 for Inception. The learning rate is reduced by a factor of 0.95 after every epoch. Although in our formulation the validation and training sets split the selected mini-batches into two separate sets, after one epoch, mini-batches used in the validation set could be used in the training set and vice versa. We test the case where we manually enforce that no examples (in mini-batches) used in the validation set are ever used for training, and find no benefit. We explore different sizes of the separate validation and training sets. We define as *validation ratio* the fraction of samples from the dataset used for validation only. Figure 2 demonstrates the influence of the validation ratio (top row), the number of compared mini-batches (second row), the size

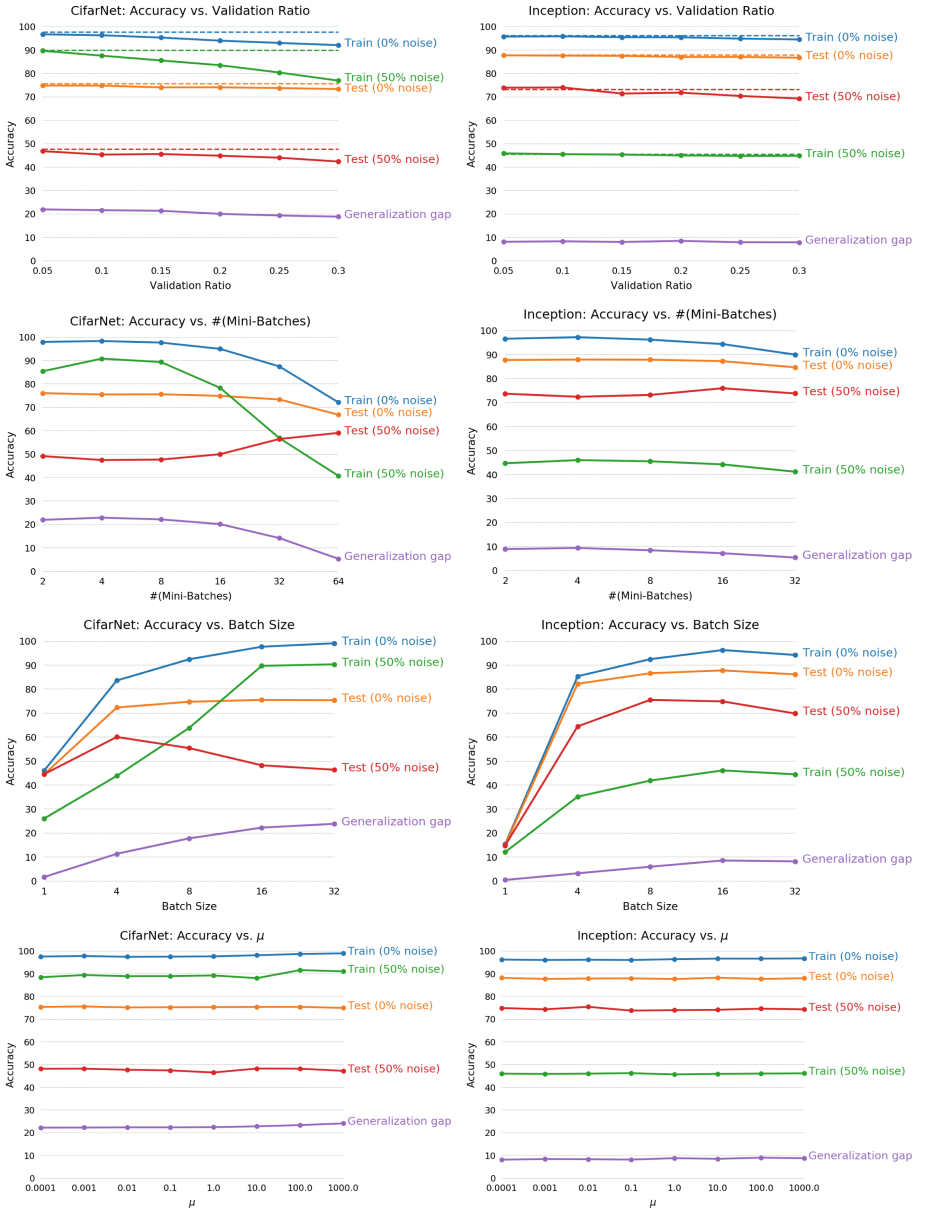

Fig. 2. Ablation experiments on CIFAR-10 with CifarNet (a small AlexNet style network) (*left*) and a small Inception network (*right*). We vary the size of the validation set (*1st row*), the number of mini-batches being compared (*2nd row*), the mini-batch size (*3rd row*) and the hyper-parameter μ (*4th row*). The networks were trained on clean as well as 50% noisy labels. The amount of label noise during training is indicated in parentheses. We show the accuracy on the clean or noisy training data, but always evaluate it on clean data. Note that the baseline of using the full training data as validation set is indicated with dashed lines on the top row.

of the compared mini-batches (third row) and the hyper-parameter μ (bottom row). We can observe that the validation ratio has only a small influence on the performance. We see an overall negative trend in the test accuracy with increasing size of the validation set, probably due to the corresponding reduction of the training set size. The number of mini-batches has a much more pronounced influence on the networks performance, especially in the case of CifarNet, where overfitting is more likely. Note that we keep the number of training steps constant in this experiment. Hence, the case with more mini-batches corresponds to smaller batch sizes. While the performance in case of noisy labels increases with the number of compared mini-batches, we observe a decrease in performance on clean data. We would like to mention that the case of 2 mini-batches is rather interesting, since it amounts to flipping (or not) the sign of the single training gradient based on the dot product with the single validation gradient. To test whether the performance in the case of a growing number of batches is due to the batch sizes, we perform experiments where we vary the batch size while keeping the number of compared batches fixed at 8. Since this modification leads to more iterations we adjust the learning rate schedule accordingly. Notice that all comparisons use the same overall number of times each sample is used. We can observe a behavior similar to the case of the varying number of mini-batches. This suggests that small mini-batch sizes lead to better generalization in the presence of label noise. Notice also the special case where the batch size is 1, which corresponds to per-example weights. Besides inferior performance we found this choice to be computationally inefficient and interfering with batch norm. Interestingly, the parameter μ does not seem to have a significant influence on the performance of both networks. Overall the performance on clean labels is quite robust to hyper-parameter choices except for the size of the mini-batches.

In Table 1, we also summarize the following set of ablation experiments:

(a) **No L^1-Constraint on $\boldsymbol{\omega}$:** We show that using the L^1 constraint $|\boldsymbol{\omega}|_1 = 1$ is beneficial for both clean and noisy labels. We set $\mu = 0.01$ and $\lambda = 1$ for this experiment in order for the magnitude of the weights ω_i to resemble the case with the L^1 constraint. While tuning of μ and λ might lead to an improvement, the use of the L^1 constraint allows plugging our optimization method without adjusting the learning rate schedule of existing models;

(b) **Weights per Layer:** In this experiment we compute a separate $\omega_i^{(l)}$ for the gradients corresponding to each layer l. We then also apply L^1 normalization to the weights $\omega_i^{(l)}$ per layer. While the results on noisy data with CifarNet improve in this case, the performance of CifarNet on clean data and the Inception network on both datasets clearly degrades;

(c) **Mini-Batch sampling:** Here we do not force the distribution of (noisy) labels in the compared mini-batches to be identical. The poor performance in this case highlights the importance of identically distributed labels in the compared mini-batches;

(d) **Dropout:** We remove the restriction of equal dropping in all the compared mini-batches. Somewhat surprisingly, this improves performance in most cases. Note that unequal dropping lowers the influence of gradients in

Table 1. Results of ablation experiments on CIFAR-10 as described in Sect. 5.1. Models were trained on clean labels and labels with 50% random noise. We report classification accuracy on the clean or noisy training labels and clean test labels. The generalization gap (difference between training and test accuracy) on clean data is also included. We also show results of the baseline model and of a model trained with standard SGD.

Experiment	CifarNet					Inception				
	Clean			50% Random		Clean			50% Random	
	Train	Test	Gap	Train	Test	Train	Test	Gap	Train	Test
SGD	99.99	75.68	24.31	96.75	45.15	99.91	88.13	11.78	65.06	47.64
Baseline	97.60	75.52	22.08	89.28	47.62	96.13	87.78	8.35	45.43	73.08
(a) L^1	96.44	74.32	22.12	95.50	45.79	79.46	77.07	2.39	33.86	62.16
(b) ω per layer	97.43	74.36	23.07	81.60	49.62	90.38	85.25	5.13	81.60	49.62
(c) Sampling	72.69	68.19	4.50	16.13	23.93	79.78	78.25	1.53	17.71	27.20
(d) Dropout	95.92	74.76	21.16	82.22	49.23	95.58	87.86	7.72	44.61	75.71

Table 2. Results of the Inception network when trained on data with random pixel permutations (fixed per image). We observe much less overfitting using our method when compared to standard SGD

Model	Train	Test	Gap
SGD	50.0	33.2	16.8
Bilevel	34.8	33.6	1.2

the deep fully-connected layers, therefore giving more weight to gradients of early convolutional layers in the dot-product. Also, dropout essentially amounts to having a different classifier at each iteration. Our method could encourage gradient updates that work well for different classifiers, possibly leading to a more universal representation.

5.2 Fitting Random Pixel Permutations

Zhang *et al.* [36] demonstrated that CNNs are able to fit the training data even when images undergo random permutations of the pixels. Since object patterns are destroyed under such manipulations, learning should be very limited (restricted to simple statistics of pixel colors). We test our method with the Inception network trained for 200 epochs on images undergoing fixed random permutations of the pixels and report a comparison to standard SGD in Table 2. While the test accuracy of both variants is similar, the network trained using our optimization shows a very small generalization gap.

5.3 Memorization of Partially Corrupted Labels

The problem of label noise is of practical importance since the labelling process is in general unreliable and incorrect labels are often introduced in the process.

Table 3. Comparison to state-of-the-art regularization techniques and methods for dealing with label noise on 40% corrupted labels.

Method	Ref.	Network	CIFAR-10	CIFAR-100
Reed *et al.* [27]	[14]	ResNet	62.3%	46.5%
Golderberger *et al.* [11]	[14]	ResNet	69.9%	45.8%
Azadi *et al.* [2]	[2]	AlexNet	75.0%	-
Jilang *et al.* [14]	[14]	ResNet	76.6%	56.9%
Zhang *et al.* [38]	-	PreAct ResNet-18	88.3%	56.4%
Standard SGD	-	PreAct ResNet-18	69.6%	44.9%
Dropout ($p = 0.3$) [30]	-	PreAct ResNet-18	84.5%	50.1%
Label Smoothing (0.1) [32]	-	PreAct ResNet-18	69.3%	46.1%
Bilevel	-	PreAct ResNet-18	87.0%	59.8%
Bilevel + [38]	-	PreAct ResNet-18	89.0%	61.6%

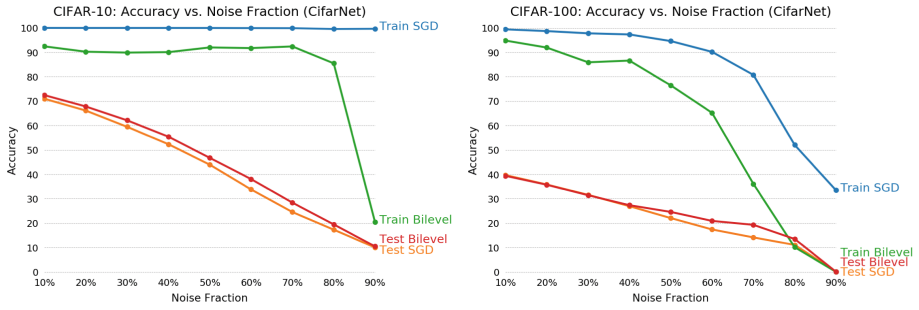

Fig. 3. CifarNet is trained on data from CIFAR-10 and CIFAR-100 with varying amounts of random label noise. We observe that our optimization leads to higher test accuracy and less overfitting in all cases when compared to standard SGD.

Providing methods that are robust to noise in the training labels is therefore of interest. In this section we perform experiments on several datasets (CIFAR-10, CIFAR-100, ImageNet) with different forms and levels of label corruption and using different network architectures. We compare to other state-of-the-art regularization and label-noise methods on CIFAR-10 and CIFAR-100.

Random Label Corruptions on CIFAR-10 and CIFAR-100. We test our method under different levels of synthetic label noise. For a noise level $\pi \in [0, 1]$ and a dataset with c classes, we randomly choose a fraction of π examples per class and uniformly assign labels of the other $c-1$ classes. Note that this leads to a completely random labelling in the case of 90% label noise on CIFAR-10. Networks are trained on datasets with varying amounts of label noise. We train the networks with our bilevel optimizer using 8 mini-batches and using the training set for validation. The networks are trained for 100 epochs on mini-batches of size 64. Learning schedules, initial learning rates and data augmentation are identical to those in Sect. 5.1. The results using CifarNet are summarized in Fig. 3

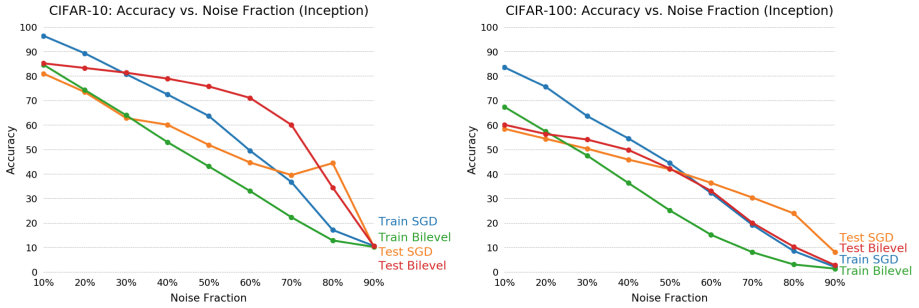

Fig. 4. The Inception network trained on data from CIFAR-10 and CIFAR-100 with varying amounts of random label noise. On CIFAR-10 our optimization leads to substantially higher test accuracy in most cases when compared to standard SGD. Our method also shows more robustness to noise levels up to 50% on CIFAR-100.

Table 4. Experiments with a realistic noise model on ImageNet

Method	44% Noise	Clean
SGD	50.75%	57.4%
Bilevel	52.69%	58.2%

and the results for Inception in Fig. 4. We observe a consistent improvement over standard SGD on CifarNet and significant gains for Inception on CIFAR-10 up to 70% noise. On CIFAR-100 our method leads to better results up to a noise level of 50%. We compare to state-of-the-art regularization methods as well as methods for dealing with label noise in Table 3. The networks used in the comparison are variants of the ResNet architecture [13] as specified in [14] and [38]. An exception is [2], which uses AlexNet, but relies on having a separate large dataset with clean labels for their model. We use the same architecture as the state-of-the-art method by Zhang *et al.* [38] for our results. We also explored the combination of our bilevel optimization with the data augmentation introduced by [38] in the last row. This results in the best performance on both CIFAR-10 and CIFAR-100. We also include results using Dropout [30] with a low keep-probability p as suggested by Arpit *et al.* [1] and results with label-smoothing as suggested by Szegedy *et al.* [32].

Modelling Realistic Label Noise on ImageNet. In order to test the method on more realistic label noise we perform the following experiment: We use the predicted labels of a pre-trained AlexNet to model realistic label noise. Our rationale here is that predictions of a neural network will make similar mistakes as a human annotator would. To obtain a high noise level we leave dropout active when making the predictions on the training set. This results in approximately 44% label noise. We then retrain an AlexNet from scratch on those labels using standard SGD and our bilevel optimizer. The results of this experiment and a comparison on clean data is given in Table 4. The bilevel optimization leads to

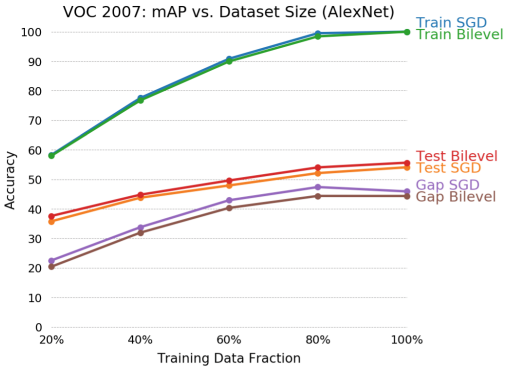

Fig. 5. We train an AlexNet for multi-label classification on varying fractions of the Pascal VOC 2007 `trainval` set and report mAP on the test set as well as the complete `trainval` set. Our optimization technique leads to higher test performance and smaller generalization gap in all cases.

better performance in both cases, improving over standard SGD by nearly 2% in case of noisy labels.

Experiments on Real-World Data with Noisy Labels. We test our method on the Clothing1M dataset introduced by Xiao *et al.* [35]. The dataset consists of fashion images belonging to 14 classes. It contains 1M images with noisy labels and additional smaller sets with clean labels for training (50K), validation (14K) and testing (10K). We follow the same setup as the state-of-the-art by Patrini *et al.* [25] using an ImageNet pre-trained 50-layer ResNet. We achieve 69.9% after training only on the noisy data and 79.9% after fine-tuning on the clean training data. These results are comparable to [25] with 69.8% and 80.4% respectively.

5.4 Generalization on Small Datasets

Small datasets pose a challenge since deep networks will easily overfit in this case. We test our method under this scenario by training an AlexNet on the multi-label classification task of Pascal VOC 2007. Training images are randomly cropped to an area between 30% to 100% of the original and then resized to 227×227. We linearly decay the learning rate from 0.01 to 0 and train for 1K epochs on mini-batches of size 64. We use the bilevel optimization method with 4 mini-batches and without a separate validation set. In Fig. 5 we report the mAP obtained from the average prediction over 10 random crops on varying fractions of the original dataset. We observe a small, but consistent, improvement over the baseline in all cases.

6 Conclusions

Neural networks seem to benefit from additional regularization during training when compared to alternative models in machine learning. However, neural

networks still suffer from overfitting and current regularization methods have a limited impact. We introduce a novel regularization approach that implements the principles of cross-validation as a bilevel optimization problem. This formulation is computationally efficient, can be incorporated with other regularizations and is shown to consistently improve the generalization of several neural network architectures on challenging datasets such as CIFAR10/100, Pascal VOC 2007, and ImageNet. In particular, we show that the proposed method is effective in avoiding overfitting with noisy labels.

Acknowledgements. This work was supported by the Swiss National Science Foundation (SNSF) grant number 200021_169622.

References

1. Arpit, D., et al.: A closer look at memorization in deep networks. arXiv preprint arXiv:1706.05394 (2017)
2. Azadi, S., Feng, J., Jegelka, S., Darrell, T.: Auxiliary image regularization for deep CNNs with noisy labels. In: International Conference on Learning Representations (2016)
3. Baydin, A.G., Pearlmutter, B.A.: Automatic differentiation of algorithms for machine learning. arXiv preprint arXiv:1404.7456 (2014)
4. Bengio, Y.: Gradient-based optimization of hyperparameters. Neural Comput. **12**(8), 1889–1900 (2000)
5. Bracken, J., McGill, J.T.: Mathematical programs with optimization problems in the constraints. Oper. Res. **21**(1), 37–44 (1973). https://doi.org/10.1287/opre.21.1.37
6. Colson, B., Marcotte, P., Savard, G.: An overview of bilevel optimization. Ann. Oper. Res. **153**(1), 235–256 (2007). https://doi.org/10.1007/s10479-007-0176-2
7. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: a large-scale hierarchical image database. In: Computer Vision and Pattern Recognition. pp. 248–255. IEEE (2009)
8. Domke, J.: Generic methods for optimization-based modeling. In: Artificial Intelligence and Statistics, pp. 318–326 (2012)
9. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (VOC) challenge. Int. J. Comput. Vis. **88**(2), 303–338 (2010)
10. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. arXiv preprint arXiv:1703.03400 (2017)
11. Goldberger, J., Ben-Reuven, E.: Training deep neural-networks using a noise adaptation layer. In: International Conference on Learning Representations (2016)
12. Hadjidimos, A.: Successive overrelaxation (SOR) and related methods. J. Comput. Appl. Math. **123**(1–2), 177–199 (2000). https://doi.org/10.1016/S0377-0427(00)00403-9
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Computer Vision and Pattern Recognition, pp. 770–778 (2016)
14. Jiang, L., Zhou, Z., Leung, T., Li, L.J., Fei-Fei, L.: Mentornet: regularizing very deep neural networks on corrupted labels. arXiv preprint arXiv:1712.05055 (2017)
15. Jindal, I., Nokleby, M., Chen, X.: Learning deep networks from noisy labels with dropout regularization. arXiv preprint arXiv:1705.03419 (2017)

16. Kawaguchi, K., Kaelbling, L.P., Bengio, Y.: Generalization in deep learning. arXiv preprint arXiv:1710.05468 (2017)
17. Krizhevsky, A.: Learning multiple layers of features from tiny images (2009)
18. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
19. Kunisch, K., Pock, T.: A bilevel optimization approach for parameter learning in variational models. SIAM J. Imaging Sci. **6**(2), 938–983 (2013)
20. Lopez-Paz, D., et al.: Gradient episodic memory for continual learning. In: Advances in Neural Information Processing Systems, pp. 6470–6479 (2017)
21. Maclaurin, D., Duvenaud, D., Adams, R.: Gradient-based hyperparameter optimization through reversible learning. In: International Conference on Machine Learning, pp. 2113–2122 (2015)
22. Natarajan, N., Dhillon, I.S., Ravikumar, P.K., Tewari, A.: Learning with noisy labels. In: Advances in Neural Information Processing Systems, pp. 1196–1204 (2013)
23. Nichol, A., Schulman, J.: Reptile: a scalable metalearning algorithm. arXiv preprint arXiv:1803.02999 (2018)
24. Ochs, P., Ranftl, R., Brox, T., Pock, T.: Bilevel optimization with nonsmooth lower level problems. In: Aujol, J.-F., Nikolova, M., Papadakis, N. (eds.) SSVM 2015. LNCS, vol. 9087, pp. 654–665. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18461-6_52
25. Patrini, G., Rozza, A., Menon, A., Nock, R., Qu, L.: Making neural networks robust to label noise: a loss correction approach. In: Computer Vision and Pattern Recognition (2017)
26. Qian, N.: On the momentum term in gradient descent learning algorithms. Neural Netw. **12**(1), 145–151 (1999)
27. Reed, S., Lee, H., Anguelov, D., Szegedy, C., Erhan, D., Rabinovich, A.: Training deep neural networks on noisy labels with bootstrapping. arXiv preprint arXiv:1412.6596 (2014)
28. Rolnick, D., Veit, A., Belongie, S., Shavit, N.: Deep learning is robust to massive label noise. arXiv preprint arXiv:1705.10694 (2017)
29. Smith, S.L., et al.: A Bayesian perspective on generalization and stochastic gradient descent. In: International Conference on Learning Representations (2018)
30. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
31. Sukhbaatar, S., Bruna, J., Paluri, M., Bourdev, L., Fergus, R.: Training convolutional networks with noisy labels. arXiv preprint arXiv:1406.2080 (2014)
32. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2818–2826 (2016)
33. Ulyanov, D., Vedaldi, A., Lempitsky, V.: Deep image prior. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018
34. Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al.: Matching networks for one shot learning. In: Advances in Neural Information Processing Systems, pp. 3630–3638 (2016)
35. Xiao, T., Xia, T., Yang, Y., Huang, C., Wang, X.: Learning from massive noisy labeled data for image classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2691–2699 (2015)

36. Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning requires rethinking generalization. In: International Conference on Learning Representations (2017)
37. Zhang, C., et al.: Theory of deep learning III: generalization properties of SGD. Technical report, Center for Brains, Minds and Machines (CBMM) (2017)
38. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: beyond empirical risk minimization. In: International Conference on Learning Representations (2017)