LOW: Training Deep Neural Networks by Learning Optimal Sample Weights

Carlos Santiago[a], Catarina Barata[a], Michele Sasdelli[b], Gustavo Carneiro[b], Jacinto C. Nascimento[a]

[a]*Institute for Systems and Robotics, Instituto Superior Técnico, Lisboa, Portugal*
[b]*School of Computer Science, Australian Institute for Machine Learning, University of Adelaide, Australia*

Abstract

The performance of deep learning (DL) models is highly dependent on the quality and size of the training data, whose annotations are often expensive and hard to obtain. This work proposes a new strategy to train DL models by Learning Optimal samples Weights (LOW), making better use of the available data. LOW determines how much each sample in a batch should contribute to the training process, by automatically estimating its weight in the loss function. This effectively forces the model to focus on more relevant samples. Consequently, the models exhibit a faster convergence and better generalization, specially on imbalanced data sets where class distribution is long-tailed. LOW can be easily integrated to train any DL model and can be combined with any loss function, while adding marginal computational burden to the training process. Additionally, the analysis of how sample weights change during training provides insights on what the model is learning and which samples or classes are more challenging. Results on popular computer vision benchmarks and on medical data sets show that DL models trained with LOW perform better than with other state-of-the-art strategies[1].

Keywords: Deep learning, sample weighting, imbalanced data sets

Email address: carlos.santiago@tecnico.ulisboa.pt (Carlos Santiago)

[1]Code is available at https://github.com/cajosantiago/LOW

1. Introduction

The performance of machine learning methods has improved dramatically in the last few years due to deep neural networks (DNNs) [1]. These models learn complex tasks by looking at many training examples, a computationally demanding task and that usually requires huge amounts of data.

In many real world problems (*e.g.*, in medical image analysis), obtaining data is often a challenge, particularly because annotations are expensive. Consequently, many of the available data sets lack the necessary variability and representativeness to train DNNs that generalize well. Additionally, these data sets are often long-tailed, meaning that the distribution of samples across different classes is not uniform (class imbalance) and some classes are under-represented. This typically prevents DNNs from learning effectively and often leads to biased systems. The goal of this work is to provide a learning strategy that is robust under the above conditions, and ensure the models make the most of the available data.

There has been an increasing effort to develop more efficient learning strategies for DNNs, such as using quantization approaches [2] or curriculum learning [3]. This paper focuses on the latter, which is inspired by the idea that training samples are not equally relevant during training [4]. Recent works have relied on importance sampling [5, 6], which aims to increasing convergence speed by selecting samples according to their gradients. However, importance sampling requires the normalization of the gradients across the entire training set, which is expensive to compute. Therefore, approximations based on information from previous epochs are usually made, preventing the combination of these strategies with online data augmentation approaches, which are popular and crucial to prevent overfitting [7, 8].

Alternative strategies rely on weighting schemes to obtain improved models. For instance, some works rely on weighted combinations of different DNNs (ensemble strategies) to achieve better performances [9, 10], while others resort to weighting to perform spatial or depth-based feature selection [11, 12]. These

works incorporate attention modules in their DNNs to actively select the most discriminative features. Other approaches adopt sample weighting strategies, where non-uniform weights are assigned to the training samples. For instance, the balanced cross-entropy loss [13] has been extensively used to deal with long-tailed data sets, particularly in medical applications [14]. The idea is to assign each sample a weight that captures the distribution of classes in the training set: less represented classes should receive higher weights in the loss function to ensure they are not ignored. A direct consequence of this strategy is that all samples from the same class are assumed to be equally relevant, which may not hold, due to the variability of samples within classes and their distribution. Sample weighting has also been adopted in boosting [15], where incorrectly classified samples receive higher weights when training the following weak classifier. However, sequentially training multiple DNNs is inefficient and often computationally infeasible. The focal loss function [16] has been proposed to address the above issues, optimizing the parameters of a single DNN using a weighted loss function that reduces the importance of well-classified samples. This method has already been applied to medical images [17], but requires a careful tuning of hyper parameters, and has been shown to not always improve the accuracy of the network [18]. Alternatively, the approach proposed in [19] optimizes the weights assigned to samples so that they minimize the loss in a validation set. The limitation of this approach is the need to build a balanced validation set that is sufficiently representative to allow a proper estimation of the gradient correction.

More complex strategies that use teacher-student models [20, 21] have also been proposed to automatically learn the weight of each sample (teacher), based on the performance of the DNN (student). Despite their promising results, these methods are computationally expensive, since they require the training of an additional DNN to act as the teacher model and the careful fine-tuning of the additional hyper-parameters.

In this work, we propose a new sample weighting strategy that overcomes many of the limitations of the aforementioned methods. This new approach,

based on **L**earning **O**ptimal sample **W**eights (LOW), aims to provide a greater decrease in the loss function at each gradient descent step. Unlike with the balanced cross-entropy loss, the weights in LOW are sample specific, simultaneously addressing class imbalance and intra-class variability issues. Moreover, the computation of its weights adds marginal burden to the training process and does not increase the number of model parameters being learned as teacher-student approaches. Furthermore, LOW can be easily incorporated into the training of any state-of-the-art DNN architecture, and can be combined with different loss functions, such as cross-entropy and focal loss, to improve their performances.

We apply LOW to both popular computer vision benchmarks (MNIST, CIFAR 10, and CIFAR 100) and real world problems (ISIC 2017 and 2018 data sets for the diagnosis of skin cancer). Our results demonstrate that LOW is particularly suited for problems with imbalanced data sets, by forcing the network to learn to classify under-represented examples. With LOW, we are able to outperform conventional weighting strategies and improve the accuracy of the model on all data sets. LOW also provides insights on which samples contribute the most during the training process, making it easier to interpret and analyze the model results. Therefore, LOW is a valuable contribution to the topic of explainable deep learning for efficient and robust pattern recognition.

2. Literature Review

In this section, we discuss the most relevant learning strategies proposed in the literature, which are either based on sampling, weighting, or teacher-student strategies.

2.1. Sampling Strategies

This type of strategies differs from the ones described in the previous section by focusing on smartly selecting the training samples used in the learning process. Importance sampling [22] is one of the most popular approaches, which

aims to select samples that make the learning process more efficient [5, 6, 23, 24, 25]. The idea is to reduce the variance of (stochastic) gradient estimates by selecting samples with an adaptive sampling distribution, instead of the traditional uniform sampling [22, 23]. However, importance sampling requires knowing the loss gradient with respect to each of the network's parameters and for each training sample before every single gradient descent step. In the context of deep neural networks, this is computationally infeasible. To overcome this issue, some methods have relaxed the problem by using either easy-to-compute approximations of the true gradients [5], or outdated versions of the gradients from previous epochs [24]. The underlying drawback is that these approaches may not be compatible with online data augmentation strategies that randomly modify the training samples at each step.

Alternatively to importance sampling, other types of sample selection strategies have also been devised. These are based on ranking samples according to the corresponding loss [26], or using more complex metrics that combine classifier uncertainty, class balance, and sample representativeness [27].

Compared to the above sampling-based methods, our proposed learning strategy has several advantages. Since it focuses only on sample weighting (and not on sample selection), the network processes all samples every epoch, thus guaranteeing that we always know how the network is performing on each sample. Furthermore, the sample weights are computed directly from the output of the forward pass, making it compatible with online data augmentation strategies.

2.2. Weighting Strategies

Curriculum learning was one of the first sample weighting strategies [3]. It relies on a predefined curriculum, typically designed based on human judgment of sample difficulty. The premise is that a network (the student) learns faster/better when the teaching material (curriculum) is chosen wisely, an analogy to how humans are educated. However, defining a good curriculum depends heavily on human expertise. To address this limitation, various authors intro-

duced the concept of self-paced learning [28, 29], which does not require human input. Instead, sample weights are determined by how well the student model performs on those samples. These weights are forced to be sparse, leading to a framework that autonomously filters out samples that the student model cannot fit at the early stages of the learning process. As the network starts to get better at the target task, the filter gradually begins accepting harder samples, leading to an adaptive curriculum learning strategy, which contrasts with the previous pre-defined curriculum. However, forcing the weights to be sparse may overfit the network to easy samples on the early stages of training. A weighting strategy has been proposed for data sets with noisy labels in [19]. They propose optimizing the weights based on a clean validation set, so that samples with incorrect labels do not interfere with the learning process. However, validation sets are often much smaller than the training set and may lack the necessary variability to compute the optimal sample weights for more challenging problems. Weighting samples has also been frequently used in problems with imbalanced data sets, namely using class-specific weights [30], to enforce good predictions on all classes. However, within each class, the relative importance of different training samples should also be taken into account.

Our proposed approach falls under this type of strategies. However, it does not required a pre-defined curriculum and can be combined with any model without requiring expert knowledge or adaptation. It also avoids overfitting easy samples by not constraining the weights to be sparse, thus samples are never disregarded during the learning process. Instead, our approach forces the network to focus more on specific samples. Finally, it also takes into account the intra-class variability, since the weights are specific for each sample.

2.3. Teacher-Student Strategies

In the last few years, several papers have proposed new learning strategies that rely on an additional model to act as the teacher [20, 21, 31, 32], whose task has been referred to as *learning to teach* [33]. For instance, Fan et al. [34, 33] used deep reinforcement learning to decide if a sample should be selected for

training (policy) based on the past performance of the student network (reward). Another example is MentorNet [21], which uses a teacher network to assign weights to the samples based on the student network performance. In this case, samples that are too hard for the student network to fit will be filtered out (i.e., their weight is reduced to zero), as in self-paced learning [28, 29], which makes this framework more robust to data sets with noisy or corrupted labels.

Although these type of approaches have a similar goal to our proposed strategy, they require additional DNNs and training to adapt to new student models or new data sets.

3. Learning with Optimal Sample Weights

In classification problems, DNNs are trained using a set of N training samples, $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, with $\mathbf{x} \in \mathbb{R}^D$ and label $y \in \mathcal{Y}$. The network takes as input a training sample $\mathbf{x}_i$ and tries to estimate the corresponding label y_i. Let $\tilde{y}_i = \psi(\mathbf{x}_i|\theta)$ be the output of a DNN parameterized by θ. The learning procedure consists of minimizing the empirical loss that relates $\psi(\mathbf{x}_i|\theta)$ with the true label y_i of the corresponding sample, such that

$$\theta^\star = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} \ell\left(\psi(\mathbf{x}_i|\theta), y_i\right), \tag{1}$$

where ℓ represents the loss function. This optimization is typically solved with (some variant of) the stochastic gradient descent (SGD) method with mini-batches of size $M \ll N$, leading to the update equation

$$\theta_{t+1} = \theta_t - \eta \frac{1}{M} \sum_{j=1}^{M} \nabla_{\theta_t} \ell\left(\psi(\mathbf{x}_j|\theta_t), y_j\right), \tag{2}$$

where η represents the learning rate, t denotes the step number, and the samples $j = 1, \ldots, M$ in the mini-batch are chosen randomly from the training set.

3.1. Gradient Descent with Weighting Strategies

In the update equation (2), all the samples contribute equally to the optimization of the network parameters, despite the fact that some samples may be more relevant than others for the learning process [4].

Weighting strategies enforce the network to pay attention to more relevant samples by assigning them higher weights. These weights, denoted by $w_j^t \in \mathbb{R}^+$, are incorporated in the update equation by multiplying the sample loss by a specific weight as follows

$$\theta_{t+1} = \theta_t - \eta \frac{1}{M} \sum_{j=1}^{M} w_j^t \nabla_{\theta_t} \ell\left(\psi(\mathbf{x}_j|\theta_t), y_j\right). \tag{3}$$

The main challenge is thus to choose the best sample weights, w_j^t.

3.2. Optimal Sample Weight

In this work, we assume that the optimal weights are those that lead to a greater decrease in the loss function at each gradient descent step. Formally, at each step t and for each sample j, we want to compute sample weight that maximizes

$$\ell\left(\psi(\mathbf{x}_j|\theta_t), y_j\right) - \ell\left(\psi(\mathbf{x}_j|\theta_{t+1}), y_j\right). \tag{4}$$

Using the linear approximation, we can rewrite the term $\ell\left(\psi(\mathbf{x}_j|\theta_{t+1}), y_j\right)$ as

$$\ell\left(\psi(\mathbf{x}_j|\theta_{t+1}), y_j\right) \approx \ell\left(\psi(\mathbf{x}_j|\theta_t), y_j\right) + \left(\nabla_{\theta_t} \ell\left(\psi(\mathbf{x}_j|\theta_t), y_j\right)\right)^\top \left(\theta_{t+1} - \theta_t\right) \tag{5}$$

Replacing this approximation in (4) and using the update equation for θ_{t+1} (based on (3) when $M = 1$) leads to

$$\begin{aligned}
&\ell(\psi(\mathbf{x}_j|\theta_t), y_j) - \ell\left(\psi(\mathbf{x}_j|\theta_{t+1}), y_j\right) = \\
&\quad = \ell\left(\psi(\mathbf{x}_j|\theta_t), y_j\right) - \ell\left(\psi(\mathbf{x}_j|\theta_t), y_j\right) - \left(\nabla_{\theta_t} \ell\left(\psi(\mathbf{x}_j|\theta_t), y_j\right)\right)^\top \left(\theta_{t+1} - \theta_t\right) \\
&\quad = -\left(\nabla_{\theta_t} \ell\left(\psi(\mathbf{x}_j|\theta_t), y_j\right)\right)^\top \left(\theta_t - \eta w_j^t \nabla_{\theta_t} \ell\left(\psi(\mathbf{x}_j|\theta_t), y_j\right) - \theta_t\right) \\
&\quad = \eta w_j^t (\nabla_{\theta_t} \ell(\psi(\mathbf{x}_j|\theta_t), y_j))^\top (\nabla_{\theta_t} \ell(\psi(\mathbf{x}_j|\theta_t), y_j)) \\
&\quad = \eta w_j^t \|\nabla_{\theta_t} \ell(\psi(\mathbf{x}_j|\theta_t), y_j)\|^2
\end{aligned} \tag{6}$$

Therefore, the computation of the weights can be expressed by the following optimization problem

$$\begin{aligned} \arg\max_{\mathbf{w}} \quad & \mathbf{w}^\top \boldsymbol{\nabla}_{\theta_t} \\ \text{subject to} \quad & w_j \geq 0,\ j = 1, \ldots, M \\ & \sum_{j=1}^{M} w_j = M \end{aligned} \tag{7}$$

where

$$\mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_M \end{bmatrix} \qquad \boldsymbol{\nabla}_{\theta_t} = \begin{bmatrix} \left\| \nabla_{\theta_t} \ell\left(\psi(\mathbf{x}_1|\theta_t), y_1\right) \right\|^2 \\ \vdots \\ \left\| \nabla_{\theta_t} \ell\left(\psi(\mathbf{x}_M|\theta_t), y_M\right) \right\|^2 \end{bmatrix}. \tag{8}$$

The first constraint in (7) ensures all sample weights are non-negative and the second constraint forces the average weights in the batch to be equal to one, so that the effective learning rate does not change. This optimization problem has a trivial solution, which makes all weights go to zero except for the sample with greater gradient norm. Under this scenario, the network would effectively use only one sample per batch to update its parameters, severely hampering its learning process. Therefore, an additional regularization term is required, penalizing large deviations from a standard weight ($w_j = 1$ for all samples $j = 1, \ldots, M$ in the batch, as in SGD), leading to

$$\begin{aligned} \arg\max_{\mathbf{w}} \quad & \mathbf{w}^\top \boldsymbol{\nabla}_{\theta_t} - \lambda \left\| \mathbf{w} - \mathbf{1} \right\|^2 \\ \text{subject to} \quad & w_j \geq 0,\ j = 1, \ldots, M \\ & \sum_{j=1}^{M} w_j = M \end{aligned} \tag{9}$$

This is a quadratic program that can be efficiently solved using appropriate optimization algorithms, where the regularization term is controlled by parameter $\lambda > 0$. As the value of λ decreases, the distribution of the sample weights will be less uniform, and will eventually make some components of $\mathbf{w}$ (associated to samples with smaller gradient norm) go to zero, while others will increase. On

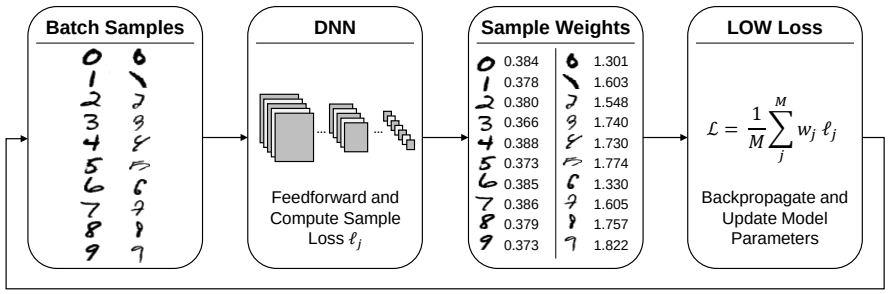

Figure 1: Overview of the training process with LOW.

the other hand, as λ increases, all sample weights will become closer to one, *i.e.*, all samples will contribute equally to the training process.

The computational complexity of the optimization problem in (9) is mostly dependent on the pre-computation of the gradient norms $\boldsymbol{\nabla}_{\theta_t}$. However, it has been previously shown that these gradient norms can be approximated by the gradient of the loss function with respect to the pre-activation outputs of the last layer in the DL model [5]. This approximation significantly simplifies the computation of $\boldsymbol{\nabla}_{\theta_t}$ and allows the sample weights to be obtained without backpropagating through the entire DNN. The computation of sample weights using this approach leads to a new learning strategy that we denote as Learning Optimal sample Weights (LOW).

An overview of the training process with LOW is shown in Fig. 1. Given a batch of samples, the DNN performs the forward pass and the sample loss (denoted by ℓ_j in the figure) is computed based on its output. Then, the optimization problem in (9) is solved based on each sample loss, from which the sample weights are obtained with marginal computational overhead. More challenging samples, such as the ones illustrated by the right column of the batch, will receive higher weights, increasing their importance during training. Once the LOW loss is computed, the model parameters are updated accordingly. Notice that this learning strategy can be applied to different DNNs and different sample loss functions.

In Section 5, we show that this approach can be particularly effective to

train DNNs when the training set is imbalanced, because the weights assigned to under-represented and more challenging classes will become higher as the model begins overfitting the most common classes. In practice, this learning strategy acts as a regularizer that prevents samples from being disregarded during the training process.

3.3. Comparison with Other Formulations

Other weighting strategies in the literature consider the problem of jointly optimizing the DNN parameters, θ, and the sample weights, $\mathbf{w}$ [28, 29, 21]. This alternative formulation leads to

$$\arg\min_{\mathbf{w},\theta} \quad \frac{1}{M}\sum_{j=1}^{M} w_j \ell\left(\psi(\mathbf{x}_j|\theta), y_j\right) + G(\mathbf{w};\lambda) + R(\theta), \tag{10}$$

where $R(\theta)$ encodes the regularization on the network parameters (such as weight decay), and $G(\mathbf{w};\lambda)$ defines the learning curriculum (*i.e.*, the importance of each sample), based on a hyper-parameter λ.

The above optimization problem is typically solved using an alternating minimization strategy, where $\mathbf{w}$ and θ are alternatively minimized while the other is assumed to be fixed. When $\mathbf{w}$ is fixed, the optimization over the network parameters, θ, leads to the formulation described in Subsection 3.1. On the other hand, the choice of function $G(\mathbf{w};\lambda)$ dictates the weighting strategy used in the optimization. Examples include self-paced learning [28], which defines this function as $G(\mathbf{w};\lambda) = -\lambda\|\mathbf{w}\|_1$, forcing the sample weights to be either zeros (for more challenging samples) or one (for easier samples), or MentorNet [21], which relies on a teacher model to provide the desired sample weights. In other words, their function $G(\mathbf{w};\lambda)$ is an additional neural network that is trained to either provide a pre-determined curriculum or learned it from a data-driven approach for each specific data set and student model.

In the case of the proposed LOW strategy, the function $G(\mathbf{w};\lambda)$ can be obtained from (9), leading to

$$G(\mathbf{w};\lambda) = -\mathbf{w}^\top \boldsymbol{\nabla}_{\theta_t} + \lambda \left\|\mathbf{w} - \mathbf{1}\right\|^2. \tag{11}$$

Notice that the signs of both terms in (11) are now switched because $G(\mathbf{w}; \lambda)$ is used in a minimization problem, whereas (9) was a maximization problem.

4. Experimental Setup

This section describes the data sets, architectures, and experiments used to evaluate the proposed learning strategy.

4.1. Data Sets

4.1.1. Benchmark Data Sets

The proposed learning strategy is evaluated on three standard benchmarks for image classification with DNNs: MNIST, CIFAR 10, and CIFAR 100.

MNIST consist of large data sets of 60,000 training samples and 10,000 test samples. Each sample is a 28×28 image with one hand-written digit, from 0 to 9 (total of 10 classes). This simple classification problem is used as a testbed in the evaluation of LOW and its impact in the training process. The robustness of LOW is further demonstrated using imbalanced versions of MNIST. To accomplish this, we randomly selected samples from each classes to a maximum of $\mathrm{NC}/2^k$ samples, where NC is the maximum number of samples per class and $k \in \{0, 1, \ldots, 9\}$. Each k is randomly assigned to one class. To exemplify, the total number of samples per class in one of the created MNIST data sets could be: $[47, 23, 3000, 12, 1500, 94, 750, 6000, 375, 188]$, which corresponds to a total of 11,989 training samples. As the example shows, some classes have a significantly higher number of samples than others, representing approximately 0.1% to 50% of the total number of training samples. The network architecture used in these tests is Lenet-5 [35].

CIFAR 10 and 100 are also large data sets of 50,000 training samples and 10,000 test samples each. Each sample is a 32×32 image from 1 out of 10 or 100 classes, respectively. For these data sets, we use two state-of-the-art networks: 1) Densenet with bottleneck and compression (Densenet-BC), without dropout and with growth rate set to 12 [36]; and 2) Wide residual network [37] with depth 28 and widening factor 2 (WRN-28-2), which follows the setup used in [5].

Table 1: Class distribution for each ISIC data set.

2017		2018	
Classes	**# Training Samples**	**Classes**	**# Training Samples**
Benign	1372	Nevus	6741
Melanoma	374	Melanoma	1119
Keratosis	254	DF	116
		Actinic	331
		Keratosis	1101
		BCC	517
		Vascular	143

The standard SGD update equation (2) is used as the baseline in all the experiments (referred to as the "Normal" learning strategy in Section 5), in which all training samples are assigned a weight equal to one. A comparison is performed with various weighting strategies, including: 1) the focal loss (FL) [16]; 2) self-paced learning (SPL) [28]; 3) the importance sampling strategy proposed in [5] (IS); and 4) the teacher-student based approach MentorNet [21].

4.1.2. Medical Application

The proposed approach is evaluated on two data sets of dermoscopy images: ISIC 2017 and 2018, which have been released as part of conference challenges. Dermoscopy image analysis is rapidly becoming a very active research field [38], mainly due to the yearly release of public data sets by the International Skin Imaging Collaboration (ISIC). These sets are increasingly larger in size and complexity, with the 2018 set comprising more than 10,000 images of seven different classes. Most recent papers in this field use DNNs to achieve an automatic diagnosis [39]. Unfortunately, dermoscopy data sets are highly imbalanced, with significantly more examples of non-malignant lesions, such as nevus or keratosis. This imbalance is clearly shown in Table 1, which summarizes the distribution of samples per class in the two data sets. Since training DNNs under these conditions is challenging, this is a good scenario to evaluate LOW.

The ISIC 2017 set is divided into two subsets: training and test with 2,000 and 600 images, respectively. Ground truth labels of the images on both subsets

are available. The ISIC 2018 data set is divided into three subsets (including a validation set), but the ground truth labels are only available for the training set. Thus, to evaluate the classification performance, the training set of 10,015 images was randomly partitioned into training (80% of the subset) and test (remaining 20%).

The DNN used in these data sets is Densenet-161 [36] pre-trained on ImageNet [40], with the last layer modified to the corresponding number of classes (three for ISIC 2017 and seven for ISIC 2018). The performance of the proposed LOW strategy is evaluated using the following metrics: 1) the accuracy (Acc); 2) the balanced accuracy (BAcc); 3) the mean area under the curve (mAUC) across all classes; and 4) the mean F1-score across all classes (mF1). These metrics are computed as follows:

- Acc: Given the total number of correctly classified samples, S^+, and the total number of samples, S, then the accuracy is given by

$$\text{Acc} = \frac{S^+}{S} \tag{12}$$

- BAcc: Denoting S_c^+ and S_c as the total number of correctly classified samples for class c and the total number of samples in class c, respectively, with $c \in \{1, \ldots, C\}$, the balanced accuracy is computed as

$$\text{BAcc} = \frac{1}{C} \sum_{c=1}^{C} \frac{S_c^+}{S_c} \tag{13}$$

- mAUC: The mAUC metric is based on the area under the receiver operating characteristics (ROC) curve of each class. The ROC curve measures the true positive rate (sensitivity) as a function of the false positive rate (specificity), obtained using different thresholds on the predicted classification confidence (see [41] for details). Let AUC_c denote the area under the receiver operating characteristics (ROC) curve for class c. Then, the mean area under the curve is given by

$$\text{mAUC} = \frac{1}{C} \sum_{c=1}^{C} \text{AUC}_c \tag{14}$$

- mF1: The F1-score is a measure of the accuracy in binary classification problems. Denoting $\mathrm{TP}_c, \mathrm{TN}_c, \mathrm{FP}_c$ and FN_c as the number of true positives, true negatives, false positives, and false negatives for class c, then the mean F1-score is given by

$$\frac{1}{C}\sum_{c=1}^{C}\frac{2\mathrm{TP}_c}{2\mathrm{TP}_c+\mathrm{FN}_c+\mathrm{FP}_c} \tag{15}$$

For comparison purposes, the proposed LOW is combined with three popular classification loss functions: 1) the traditional cross-entropy loss (CE) [42]; 2) the focal loss (FL) [16]; and 3) the class balanced cross entropy (bCE) [30], commonly used on the imbalanced data sets, in which sample weights are inversely proportional to its class distribution. This means that the computation of $\boldsymbol{\nabla}_{\theta_t}$ in (9) is based on each of the above loss functions, effectively combining them with LOW.

4.2. Implementation Details

All models mentioned in the previous subsection were trained with an SGD optimizer, using Nesterov accelerated gradient with momentum of 0.9. The Lenet-5 model, used on MNIST, was trained for 100 epochs, with a learning rate of $\eta = 0.001$ and a batch size of 256. In the case of Densenet-BC on CIFAR 10 and 100, the model was trained for 300 epochs, batch size 64, and an initial learning rate of $\eta = 0.1$ with decay to 0.01 and 0.001 after 150 and 225 epochs, respectively. Conventional online data augmentation strategies (random cropping and horizontal flipping) were used when training Densenet. These training conditions were based on [36]. The WideResnet-28-2 model used on CIFAR 10 and 100 was trained for 100 epochs, using a batch size of 128 and an initial learning rate of $\eta = 0.1$ with a decay to 0.02 and 0.004 after 30 and 60 epochs, similarly to [5]. As before, random cropping and horizontal flipping were used for data augmentation. Finally, Densenet-161 was trained for 150 epochs on the medical data sets, with batch size of 32, and an initial learning rate of $\eta = 10^{-4}$ and 10^{-5} and 10^{-6} after epochs 75 and 112, respectively.

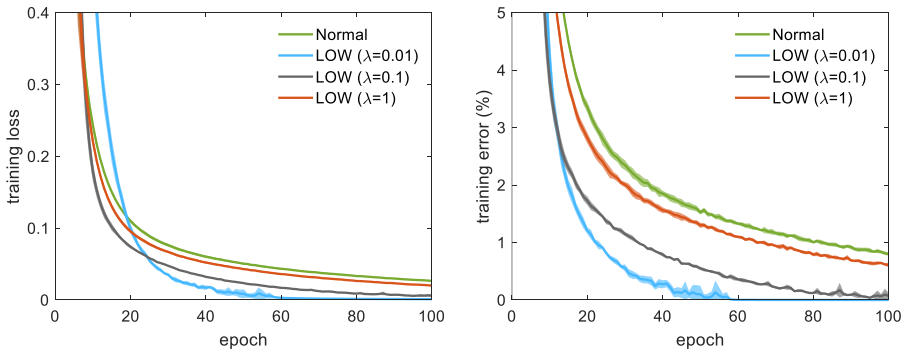

Figure 2: Training loss and classification error on MNIST over 5 runs using a normal training strategy and using LOW with $\lambda \in \{0.01, 0.1, 1\}$

.

The images were pre-processed with a color normalization method [43] and, for training, both random horizontal and vertical flipping were used, as well as a random crop from 300×300 to 224×224 without padding.

The results were based on a Pytorch [44] implementation[2], using an NVIDIA Titan Xp. The quadratic program in (9) is solved using CVXOPT [45], and, unless otherwise stated, the reported results for LOW were based on the best results of a grid search for regularization term $\lambda \in \{1, 0.5, 0.1, 0.01\}$. For a fair comparison, the reported results are based on 5 different runs and, in each run, all the approaches use the same initialization and the same seed to ensure identical sample selections.

5. Results

This section divides the evaluation of the proposed learning strategy in two parts: 1) benchmark data sets; and 2) real world application: the classification of dermoscopy image for skin cancer diagnosis.

[2]Code is available at `https://github.com/cajosantiago/LOW`

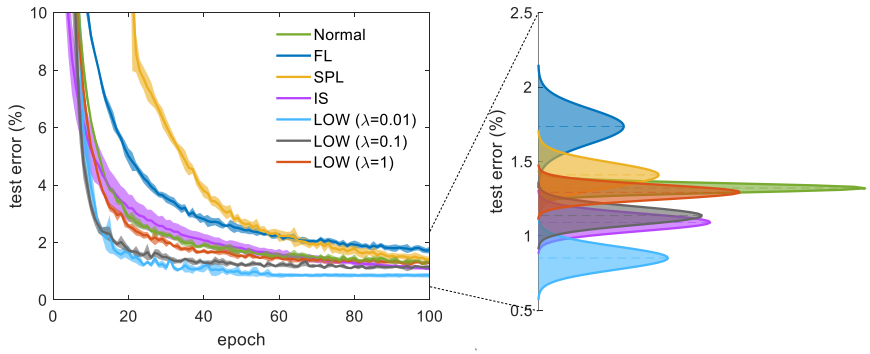

Figure 3: Classification error on MNIST test set over 5 runs and comparison with focal loss (FL) [16], self-paced learning (SPL) [28], and importance sampling (IS) [5].

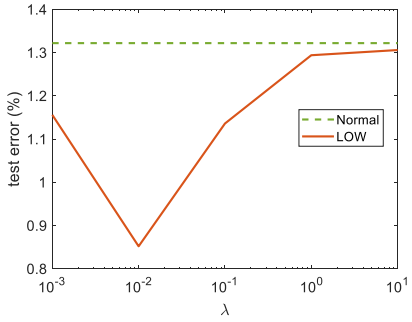

Figure 4: Influence of λ in LOW on the test error on MNIST.

5.1. Benchmark Data Sets

5.1.1. MNIST

The first experiment is based on the training of a Lenet-5 for the classification of the MNIST data set. In this experiment, we use the full training set (60,000 images) and evaluate the performance of the network in the test set (10,000 images), both of which are balanced (all classes have approximately the same number of samples). The average convergence curves can be seen in Figures 2 and 3. Both figures show the curves obtained using different values of $\lambda \in \{0.01, 0.1, 1\}$. In these figures, the standard deviation across different runs is also represented by the shaded areas along the convergence curves, and also by

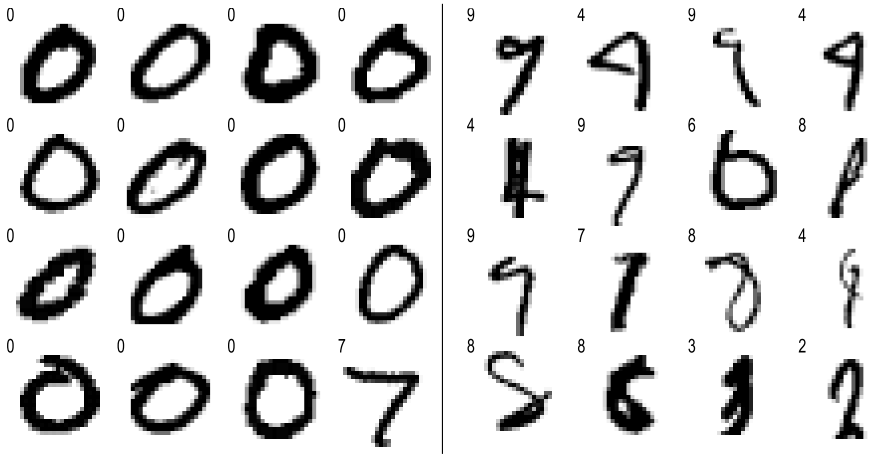

Figure 5: Examples of samples that received lower weights (left) and higher weights (right) on average across the epochs. The true label is represented in the top left corner of each sample.

the width of the distributions in Fig. 3 (right).

Fig. 2 shows that the proposed learning strategy LOW converges to a lower loss value and classification error, compared to a normal training (i.e., when all samples have a weight equal to one). For a λ value of 1 and 0.1, the loss function decreases faster than the normal approach, while for $\lambda = 0.01$, the loss is higher in the first 20 epochs, but then converges to a lower value. A similar behavior occurs in the training error, with lower value of λ leading to lower final error rates.

Fig. 3 also shows a comparison with other state-of-the-art approaches. It is possible to see that LOW outperforms focal loss (FL) [16] and self-paced learning (SPL) [28] for all the values of λ. LOW also achieves a lower generalization error than importance sampling (IS) [5] for $\lambda = 0.01$, as shown in Fig. 3 (right).

A more detailed study on the effect of λ is performed in Fig. 4. Recall that λ defines the weight of the regularization term in (9), meaning that higher values force the sample weights to be closer to the normal SGD. This is clearly visible by the LOW curve convergence to the normal SGD baseline as λ increases. On the other hand, very small λ values will make the weight of some samples to go to zero, which means the DNN will disregard them during training. This

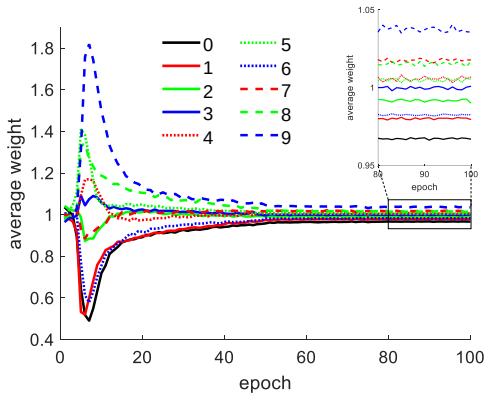

Figure 6: Average sample weight per class across the epochs on MNIST.

explains the increase in the error rate for $\lambda = 10^{-3}$.

By analyzing all the 60,000 sample weights across the 100 epochs, it is possible to determine which samples contributed the least and the most to the learning process. To evaluate how much each sample contributes, we computed their overall weights by averaging over the 100 epochs. Examples of samples that contributed the least are shown on the left side of Fig. 5, with the majority belonging to class "0", while the samples that contributed the most belong to several classes but clearly constitute harder-to-classify examples (the true label in provided at the top left corner of each sample).

It was also possible to determine the average contribution of each class throughout the epochs, shown in Fig. 6. The following conclusions can be drawn from this figure. In the beginning of the training, all samples have approximately the same weight. This makes sense, as the network initially finds all samples equally challenging, making them all important. Then, the network starts to learn to classify some samples and their weights become more diverse. In particular, the classes "0", "1", and "6" receive, on average, lower weights, while "9", "8", and "5" receive higher weights. However, as the performance of the network starts to converge (approximately after epoch 40, as shown in Fig. 2), all class weights tend to 1, corresponding to the phase of the training process in which the model has already learned to classify most samples with

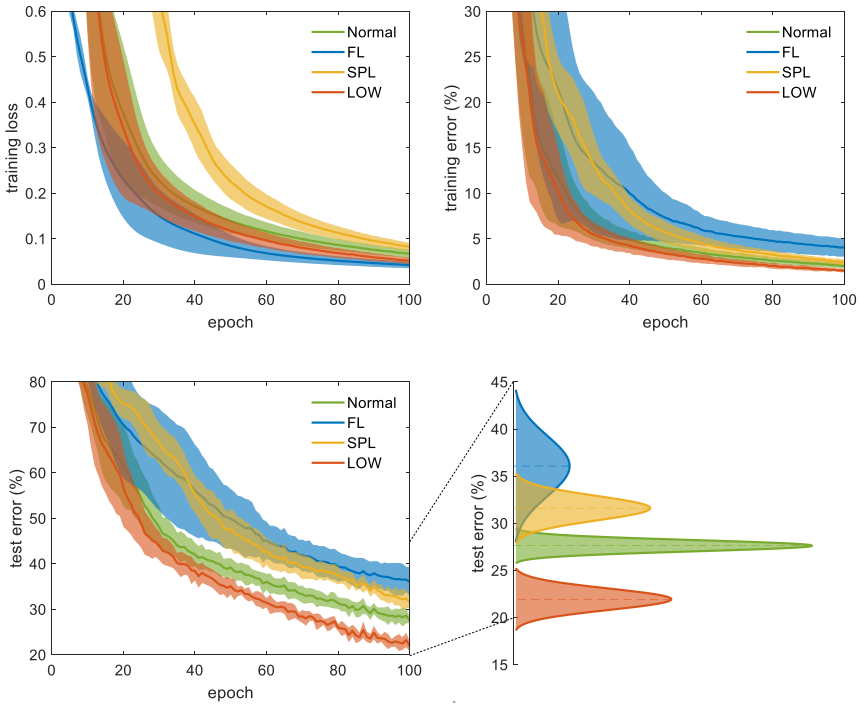

Figure 7: Performance of Lenet on imbalanced MNIST over 5 runs.

high accuracy and no class is considered more important than the others.

5.1.2. Imbalanced MNIST

In the second experiment, we created imbalanced versions of MNIST by selecting a non-uniform number of samples per classes, as explained in the Section 4. The convergence curves are plotted in Fig. 7, and show that all learning strategies achieve a worse classification performance than using the full data set. However, training the network with LOW leads to a significantly lower classification error. Compared to Fig. 2, it is possible to see that the standard deviation is much higher now. This is expected since the distribution of the number of samples per class is randomly chosen at each run and, as shown in the previous section, not all classes are equally challenging to the network.

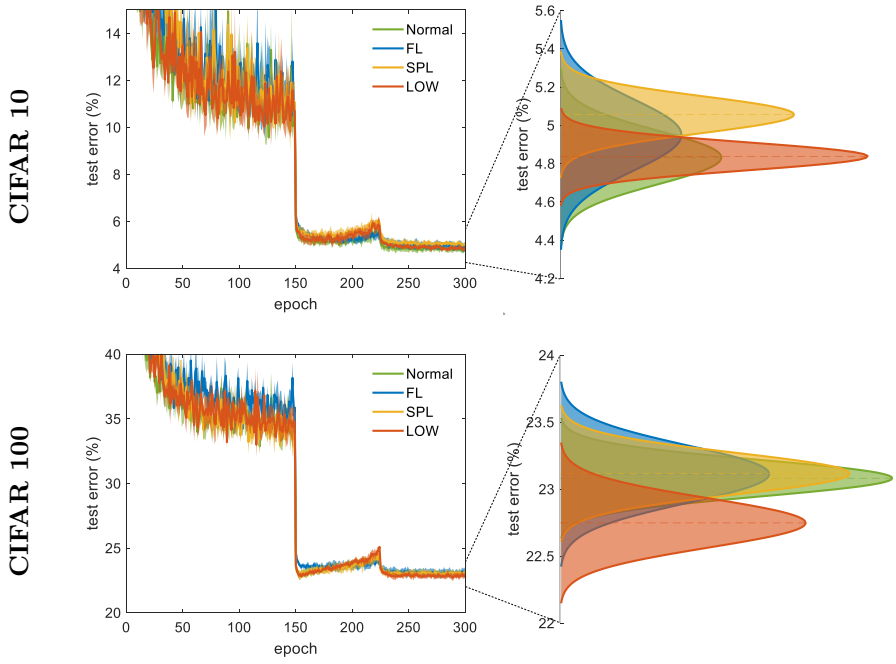

Figure 8: Test classification error on CIFAR 10 and 100 over 5 runs using Densenet.

5.1.3. CIFAR 10 and CIFAR 100

To demonstrate that LOW works with more complex models and data sets, we evaluated its performance on CIFAR 10 and 100. For each of them, we trained two models (Densenet and WideResnet) on 5 different runs.

Fig. 8 shows the convergence curves of the test error for the Densenet model. The difference between these curves is not as clear as in the MNIST case, but the final classification error, shown on the right, demonstrates that, for CIFAR 10, LOW is able to achieve the same performance as the normal strategy but with a smaller variance (*i.e.*, the results are more consistent across different runs), while for CIFAR 100 it clearly achieves a better performance than all the remaining strategies.

Similar conclusions can be drawn for the WideResnet model shown in Table 2. Here, we compare our results with FL [16], SPL [28], IS [5], and also MentorNet [21], in which we fine-tuned a MentorNet pre-trained on a 20% noise CIFAR

Table 2: Average classification error on CIFAR 10 and CIFAR 100 test set.

	Densenet-BC		**WideResnet-28-2**	
Method	CIFAR 10	CIFAR 100	CIFAR 10	CIFAR 100
Normal	**4.8**	23.1	7.2	28.0
FL [16]	5.0	23.1	7.6	28.6
SPL [28]	5.1	23.1	7.7	28.2
IS [5]	15.7	43.2	7.9	32.0
MentorNet [21]	13.1	38.9	8.5	29.1
LOW	**4.8**	**22.8**	**6.8**	**27.7**

10 with WideResnet as the student network, as proposed by the authors. Comparing the results, we can see that LOW outperforms all the other approaches in both data sets and for both models, while both FL and SPL achieve similar performances to the normal training strategy. The results obtained with MentorNet are slightly worse than the other approaches. However, it is important to recall that MentorNet was proposed in the context of noisy data sets (where some samples were assigned the wrong label), which is not the scenario evaluated in this work. Nonetheless, the table also shows that this strategy is sensitive to the DNN used as the student network – using a MentorNet pre-trained on a different student model seems to be a poor strategy. Regarding the IS strategy, the authors mention that the estimation of the sampling probabilities can be affected by batch normalization layers, which are used in the Densenet model, thus explaining the poor performance of this strategy for this case.

5.2. Medical Data Sets

In this section, we evaluated the performance of LOW in a real world application: diagnosis of skin cancer in dermoscopy images. As previously mentioned, two data sets were used (ISIC 2017 and 2018), and both are significantly imbalanced (recall Table 1), as is common is most medical data sets. In this scenario, conventional learning strategies will focus on the most represented class, and perform poorly on classes with fewer samples. The proposed learning strategy

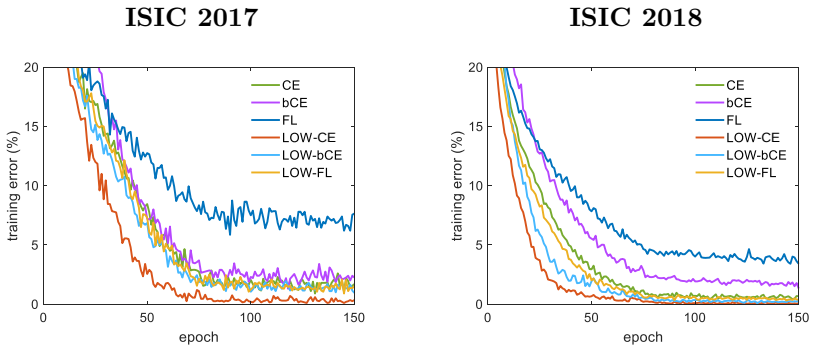

Figure 9: Classification error on the ISIC 2018 training set obtained with three loss functions: 1) the traditional cross-entropy (CE), 2) the balanced CE (bCE), and the focal loss (FL); and comparison with the corresponding combination with LOW.

Table 3: Results of skin cancer classification on ISIC 2017 test set.

	Loss	Acc	BAcc	mAUC	mF1
Normal	**CE**	75.7	65.4	86.0	65.4
	BCE	75.0	**70.4**	**87.2**	67.8
	FL	74.7	65.3	85.6	64.5
LOW	**CE**	**78.0**	68.2	**87.2**	**68.9**
	BCE	75.5	67.2	86.2	66.9
	FL	75.5	66.2	85.1	66.1

prevents this by forcing the network to address the most challenging samples through the corresponding sample weights. Furthermore, we demonstrate the advantage of combining LOW with different loss functions, namely: 1) the traditional cross-entropy (CE) [42]; 2) the balanced cross-entropy (bCE) [30]; and 3) the focal loss (FL) [16].

Fig. 9 shows the convergence of the classification error on the ISIC 2017 and 2018 training set. In both cases, training with LOW leads to a faster convergence and to a better training error on all the losses. The generalization of the network evaluated on ISIC 2017 and 2018 test sets is shown in Tables 3 and 4. Results show that the combination of LOW with CE loss outperforms

Table 4: Results of skin cancer classification on ISIC 2018 test set.

	Loss	Acc	BAcc	mAUC	mF1
Normal	**CE**	86.6	73.1	96.8	76.4
	BCE	88.8	**78.9**	97.2	79.7
	FL	85.5	74.5	96.3	74.9
LOW	**CE**	**89.5**	77.7	**97.8**	80.4
	BCE	89.4	77.4	**97.8**	**81.1**
	FL	87.7	75.5	96.8	77.5

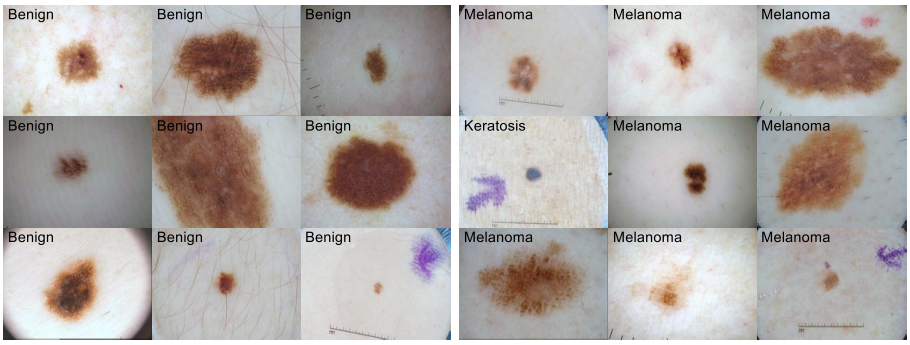

Figure 10: Examples from ISIC 2017 of samples that received lower weights (left) and higher weights (right). The true label is shown in the top left corner of each sample.

all other strategies on most metrics and that, for all losses, LOW outperforms its "normal" counterpart on most of the metrics. This means that it is always beneficial to use LOW even when using the other losses.

As in the MNIST case, we also analyzed the weights assigned to each sample in the ISIC 2017 data set during training, in order to determine which samples received higher weights throughout the learning process. Fig. 10 shows a few samples from each case, and it is possible to see that all of the samples with lower weights belong to the "Benign" class. These samples share similar appearance to the ones on the right side of the figure, which makes the model unable to correctly classify both groups. Melanoma often mimics other kinds of lesions, in particular benign nevi as the ones shown on the left side of Fig. 10, which

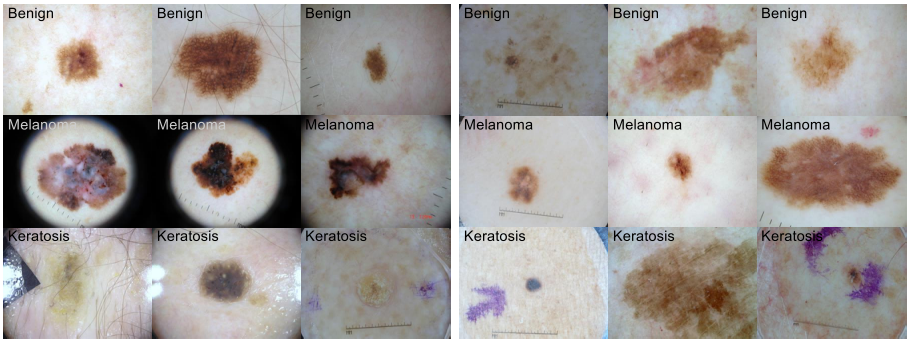

Figure 11: Examples from ISIC 2017 of samples from each class (one specific class in each row) that received lower weights (left) and higher weights (right). The true label is represented in the top left corner of each sample.

makes the medical diagnosis challenging and subjective [39]. For comparison, Fig. 11 shows, for each class, samples that received lower (left) and higher (right) weights. On the left side are examples of melanoma that are clearly distinct from the other classes. These lesions exhibit multiple colors and irregular textures, features that are usually associated with the "Melanoma" class, making them less challenging for the model. This helps explain why LOW assigns them lower weights. Overall, Fig. 11 suggests that samples that are easier to classify, due to their distinctive features, are assigned lower weights in the learning process, while samples whose appearance is misleading are assigned higher weights, which means the model is struggling to find discriminative features.

The average sample weight for each class, shown in Fig. 12, also suggests that the "Benign" class contributes the least to train the network, while "Keratosis" contributes more in the the first epochs and is then surpassed by "Melanoma" as the most contributing class. This is an interesting observation because "Keratosis" is the less common class, but the model rapidly learns to classify it. The "Melanoma" class is clearly more challenging, as the average weights remain above one for a longer period. This supports the results from Fig. 10, where almost all of challenging examples were "Melanoma". Nonetheless, as in the MNIST case, in the later stages of the learning process, all classes are

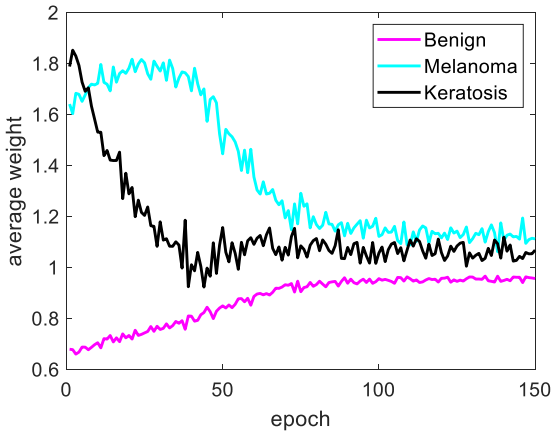

Figure 12: Average sample weight per class on ISIC 2017.

considered equally important and their average sample weights tend to one.

A similar analysis can be performed for ISIC 2018 data set, whose average weights per class are shown in Fig. 13. In this case, the classes that receive higher weights in the early stages of the training process are the least represented ones. However, it is interesting to note that the average weight is not directly related to the class proportions in the dataset. For instance, "Melanoma" has the highest average weight between epochs 25 and 50, but it is actually the class with second higher number of samples. On the other hand, "Vascular" is one of the classes with fewer samples, but it is assigned a low average weight early on. This justifies why applying a fixed class weight as in bCE may not be the best approach.

6. Conclusions

This paper proposed a new learning strategy denoted as learning with optimal sample weights (LOW). This weighting-based approach automatically determines the contribution of each training samples in each step of the gradient descent. The sample weights are computed through the optimization of a quadratic program that aims to maximize the decrease in the loss function.

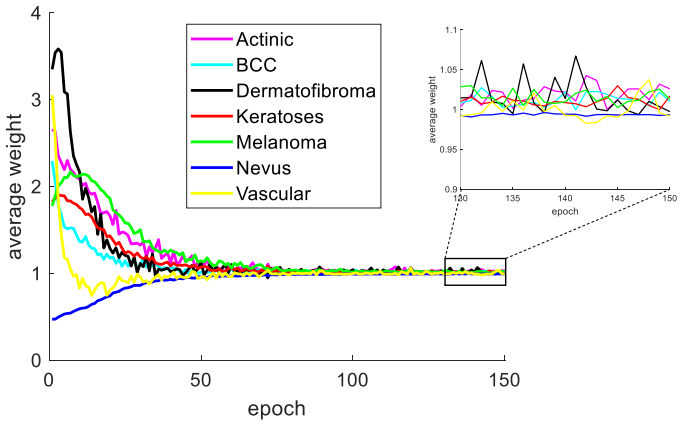

Figure 13: Average sample weight per class on ISIC 2018.

This approach ensures that the DNNs focus on different samples throughout the learning process, and prevents the models from overfitting predominant classes. The proposed strategy is evaluated on both computer vision benchmark data sets (MNIST, CIFAR 10, and CIFAR 100) as well as medical data sets for skin cancer diagnosis (ISIC 2017 and 2018). The results show that LOW leads to better generalization than the normal learning strategy on all data sets, particularly when the data sets are imbalanced. Furthermore, it outperforms other popular weighting schemes, including sampling based and teacher-student based approaches. LOW also provides insights on which samples contribute the most throughout the learning process, improving its explainability and the interpretability of the learning process.

LOW assumes that all samples are beneficial to the training process, although some more than others. This assumption is common in most learning scenarios, but does not hold when the data set labels are noisy. In fact, since LOW forces the model to focus on less represented or more challenging samples, it may lead to an exaggerated attention to outliers when used to train models with noisy labels, eventually causing the model to overfit to this data. However, such limitation can be exploited in future work. Specifically, persistently high

weights during training can be indicative of potential outliers, in which case LOW could be used preemptively to identify these errors.

Acknowledgments

This work was supported by Portuguese FCT projects UIDB/50009/2020, UID/EEA/50009/2019, and CEECIND/00326/2017, and by Australian Research Council through grants DP180103232, CE140100016, and FT190100525. G.C. acknowledges the support by the Alexander von Humboldt-Stiftung for the renewed research stay sponsorship. The Titan Xp used for this research was donated by the NVIDIA Corporation.

References

[1] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, nature 521 (7553) (2015) 436.

[2] H. Qin, R. Gong, X. Liu, X. Bai, J. Song, N. Sebe, Binary neural networks: A survey, Pattern Recognition (2020) 107281.

[3] Y. Bengio, J. Louradour, R. Collobert, J. Weston, Curriculum learning, in: Proceedings of the 26th Annual International Conference on Machine Learning, ACM, 2009, pp. 41–48.

[4] A. Lapedriza, H. Pirsiavash, Z. Bylinskii, A. Torralba, Are all training examples equally valuable?, arXiv e-prints (2013) arXiv:1311.6510`arXiv:1311.6510`.

[5] A. Katharopoulos, F. Fleuret, Not all samples are created equal: Deep learning with importance sampling, in: International Conference on Machine Learning, 2018, pp. 2530–2539.

[6] T. B. Johnson, C. Guestrin, Training deep models faster with robust, approximate importance sampling, in: Advances in Neural Information Processing Systems, 2018, pp. 7276–7286.

[7] D. Shen, G. Wu, H.-I. Suk, Deep learning in medical image analysis, Annual review of biomedical engineering 19 (2017) 221–248.

[8] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. van der Laak, B. van Ginneken, C. I. Sánchez, A survey on deep learning in medical image analysis, Medical image analysis 42 (2017) 60–88.

[9] L. Gao, X. Li, J. Song, H. T. Shen, Hierarchical lstms with adaptive attention for visual captioning, IEEE Transactions on Pattern Analysis and Machine Intelligence (2019) 1–1.

[10] W.-J. Yu, Z.-D. Chen, X. Luo, W. Liu, X.-S. Xu, Delta: A deep dual-stream network for multi-label image classification, Pattern Recognition 91 (2019) 322–331.

[11] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, Y. Bengio, Show, attend and tell: Neural image caption generation with visual attention, in: International conference on machine learning, 2015, pp. 2048–2057.

[12] L. Chen, H. Zhang, J. Xiao, L. Nie, J. Shao, W. Liu, T. S. Chua, Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 5659–5667.

[13] S. Xie, Z. Tu, Holistically-nested edge detection, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 1395–1403.

[14] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: International Conference on Medical image computing and computer-assisted intervention, Springer, 2015, pp. 234–241.

[15] Y. Freund, R. E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, Journal of computer and system sciences 55 (1) (1997) 119–139.

[16] T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollár, Focal loss for dense object detection, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 2980–2988.

[17] W. Zhu, Y. Huang, L. Zeng, X. Chen, Y. Liu, Z. Qian, N. Du, W. Fan, X. Xie, Anatomynet: Deep learning for fast and fully automated whole-volume segmentation of head and neck anatomy, Medical physics 46 (2) (2019) 576–589.

[18] J. Redmon, A. Farhadi, YOLOv3: An Incremental Improvement, arXiv e-prints (2018) arXiv:1804.02767arXiv:1804.02767.

[19] M. Ren, W. Zeng, B. Yang, R. Urtasun, Learning to reweight examples for robust deep learning, in: International Conference on Machine Learning, 2018, pp. 4334–4343.

[20] C. Finn, P. Abbeel, S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org, 2017, pp. 1126–1135.

[21] L. Jiang, Z. Zhou, T. Leung, L.-J. Li, L. Fei-Fei, Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels, in: International Conference on Machine Learning, 2018, pp. 2309–2318.

[22] D. Needell, R. Ward, N. Srebro, Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm, in: Advances in Neural Information Processing Systems, 2014, pp. 1017–1025.

[23] P. Zhao, T. Zhang, Stochastic optimization with importance sampling for regularized loss minimization, in: International Conference on Machine Learning, 2015, pp. 1–9.

[24] G. Alain, A. Lamb, C. Sankar, A. Courville, Y. Bengio, Variance Reduction in SGD by Distributed Importance Sampling, arXiv e-prints (2015) arXiv:1511.06481arXiv:1511.06481.

[25] D. Needell, R. Ward, Batched stochastic gradient descent with weighted sampling, in: International Conference Approximation Theory, Springer, 2016, pp. 279–306.

[26] I. Loshchilov, F. Hutter, Online batch selection for faster training of neural networks, in: International Conference on Machine Learning (ICML) Workshop, 2015, pp. 1–1.

[27] M. Kabkab, A. Alavi, R. Chellappa, DCNNs on a Diet: Sampling Strategies for Reducing the Training Set Size, arXiv e-prints (2016) arXiv:1606.04232arXiv:1606.04232.

[28] M. P. Kumar, B. Packer, D. Koller, Self-paced learning for latent variable models, in: Advances in Neural Information Processing Systems, 2010, pp. 1189–1197.

[29] L. Jiang, D. Meng, Q. Zhao, S. Shan, A. G. Hauptmann, Self-paced curriculum learning., in: AAAI, Vol. 2, 2015, p. 6.

[30] F. Provost, Machine learning from imbalanced data sets 101, in: Proceedings of the AAAI2000 workshop on imbalanced data sets, 2000, pp. 1–3.

[31] A. Graves, M. G. Bellemare, J. Menick, R. Munos, K. Kavukcuoglu, Automated curriculum learning for neural networks, in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org, 2017, pp. 1311–1320.

[32] T. Matiisen, A. Oliver, T. Cohen, J. Schulman, Teacher-student curriculum learning, IEEE Transactions on Neural Networks and Learning Systems (2019) 1–9doi:10.1109/TNNLS.2019.2934906.

[33] Y. Fan, F. Tian, T. Qin, X.-Y. Li, T.-Y. Liu, Learning to teach, in: International Conference on Learning Representations (ICLR), 2018, pp. 1–1.

[34] Y. Fan, F. Tian, T. Qin, J. Bian, T.-Y. Liu, Learning What Data to Learn, arXiv e-prints (2017) arXiv:1702.08635arXiv:1702.08635.

[35] Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, et al., Comparison of learning algorithms for handwritten digit recognition, in: International conference on artificial neural networks, Vol. 60, Perth, Australia, 1995, pp. 53–60.

[36] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4700–4708.

[37] S. Zagoruyko, N. Komodakis, Wide residual networks, arXiv preprint arXiv:1605.07146.

[38] M. E. Celebi, N. Codella, A. Halpern, Dermoscopy image analysis: overview and future directions, IEEE journal of biomedical and health informatics 23 (2) (2019) 474–478.

[39] C. Barata, M. E. Celebi, J. S. Marques, A survey of feature extraction in dermoscopy image analysis of skin cancer, IEEE journal of biomedical and health informatics 23 (3) (2018) 1096–1109.

[40] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE conference on computer vision and pattern recognition, Ieee, 2009, pp. 248–255.

[41] A. P. Bradley, The use of the area under the roc curve in the evaluation of machine learning algorithms, Pattern recognition 30 (7) (1997) 1145–1159.

[42] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016, http://www.deeplearningbook.org.

[43] C. Barata, M. E. Celebi, J. S. Marques, Improving dermoscopy image classification using color constancy, IEEE journal of biomedical and health informatics 19 (3) (2014) 1146–1152.

[44] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., Pytorch: An imperative style, high-performance deep learning library, in: Advances in Neural Information Processing Systems, 2019, pp. 8024–8035.

[45] M. S. Andersen, J. Dahl, L. Vandenberghe, CVXOPT: A Python package for convex optimization, `http://cvxopt.org`, accessed: 2019-07-10.