IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 17, 2020, accepted May 19, 2020, date of publication June 10, 2020, date of current version June 26, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3001531

Approximating the Gradient of Cross-Entropy Loss Function

LI LI, (Student Member, IEEE), MILOŠ DOROSLOVAČKI, (Member, IEEE), AND MURRAY H. LOEW, (Life Fellow, IEEE)

Department of Electrical and Computer Engineering, The George Washington University, Washington, DC 20052, USA

Corresponding authors: Li Li (lili1986@gwu.edu) and Miloš Doroslovački (doroslov@gwu.edu)

ABSTRACT A loss function has two crucial roles in training a conventional discriminant deep neural network (DNN): (i) it measures the goodness of classification and (ii) generates the gradients that drive the training of the network. In this paper, we approximate the gradients of cross-entropy loss which is the most often used loss function in the classification DNNs. The proposed approximations are noise-free, which means they depend only on the labels of the training set. They have a fixed length to avoid the vanishing gradient problem of the cross-entropy loss. By skipping the forward pass, the computational complexities of the proposed approximations are reduced to $\mathcal{O}(n)$ where n is the batch size. Two claims are established based on the experiments of training DNNs using the proposed approximations: (i) It is possible to train a discriminant network without explicitly defining a loss function. (ii) The success of training does not imply the convergence of network parameters to fixed values. The experiments show that the proposed gradient approximations achieve comparable classification accuracy to the conventional loss functions and can accelerate the training process on multiple datasets.

INDEX TERMS Deep neural networks, cross-entropy, gradient, loss function.

I. INTRODUCTION

Deep neural networks (DNNs) are hotspots of machine learning research in recent years as they are proven to be successful in a wide range of discriminant applications. As a supervised approach, DNNs need to be trained to obtain a set of parameters that determine the mapping of input data to the representation space. After the mapping, a classifier is employed to predict the label of corresponding input based on the obtained representations. Conventional training of a DNN assumes a loss function that measures the ''goodness'' of the classification by comparing the prediction to the ground truth. Specifically, errors between the predicted and true labels are calculated over the training set. The errors are then combined into a scalar which is called loss. This phase of calculating the loss value from representation points is called forward propagation of the loss function [1]. The training of the network actually occurs in the back propagation phase, in which the parameters of the network are updated proportionally to the gradient of the loss with respect to the parameters. As all the negative gradients are calculated by the chain rule that starts with the partial derivatives of the loss with respect to the representations, the derivatives of the loss function are the starting ''forces'' that drive the training of the network. This forward-backward pass paradigm for loss function is used in most of modern DNNs, whether the loss function is probability-based [2], energy-based [3], or geometry-based [4], [5].

As the training of a DNN is driven by the gradients of the loss function that are generated in the backward pass, a question that naturally arises is whether the forward pass of loss function is necessary? Although the value of a loss function provides partial information about training, e.g., it is helpful for determination of overfitting, the loss *value* itself is not essential for getting iteratively better representations. The gradients provide the direction and strength that nudge the parameters in every iteration and promote the representations that are easier to separate. Now the second question appears: can we approximate the gradient and make calculation of gradients easier without having a noticeable loss in prediction performance? Then, the followed question is that if such approximations exist, can they shorten the training process? In this paper, we give the answers to these questions by approximating the gradient of cross-entropy which is one of

The associate editor coordinating the review of this manuscript and approving it for publication was Mingjun Dai.

the most popular loss functions for training DNNs. We give new explanations of the effectiveness of cross-entropy loss using geometric interpretations and propose two approximations of the loss gradients. These approximations result in very simple functions that could be used for training DNNs and reduce the computational complexity of the last (loss function) layer in a DNN to $\mathcal{O}(n)$. These approximations do not require explicit calculation of the loss and are applied in convolutional neural networks (CNNs) and in fully-connected networks (FC-nets). Experiments on the optical coherence tomography (OCT) and MNIST datasets show the effectiveness and efficiency of our proposed gradient approximations for training DNNs.

The goal of this paper is to discuss the properties of the cross-entropy gradients and approximate the gradients by preserving their important properties. The experiments focus on *training* and are used to show the behavior of the approximations. We show that a network with parameters not converging to fixed values can fit the training dataset and achieve similar test accuracy compared to the conventional training approach; in other words, the success of training does not imply the convergence of network parameters to fixed values. Our discussions focus on the classification problems, as they could be effectively solved by the cross-entropy loss function. For the other problems, e.g., regression, DNNs usually use other loss functions. Consideration of these problems is beyond the scope of this paper.

II. BACKGROUND

The parameters of a DNN are updated by the gradients of a loss function. The parameters form the mapping from data to data representations. Therefore, a loss function plays a key role in training DNNs and determines the forms of representations learned by the DNN. There exist three major categories of loss functions, and they lead to different interpretations for DNNs.

The first category contains probability-based loss functions. The most outstanding one in this category is cross-entropy, which is a generalization of logistic regression to multi-class scenarios and was first proposed by Bridle [2]. Its popularity in the neural networks community gave birth to its variants [6]–[11]. Another often used loss function in this category is earth mover's distance [12], [13]. All of these loss functions have clear probabilistic or information theoretic interpretations. The main idea is to maximize the likelihood of the correct prediction given the ground truth in the training set.

The second category is energy-based [14]–[16], which were mainly contributed by B. Juang and Y. LeCun *et al.*, and summarized in [3]. For these loss functions, the corresponding models are viewed as an energy function which measures the "goodness" of each possible configuration of input data and labels. The loss value can be interpreted as the degree of compatibility between the values of input and labels [3]. The relation between the energy-based to the probability-based approaches can be established by Gibbs distribution [3].

Third, metric-based learning (or similarity learning) comprises a big family of the geometric-based approaches. They include the popular mean squared loss, triplet loss [17], neighborhood-based approaches [18], [19], and principal component analysis/linear discriminant analysis-based approaches [4], [5], [20]. These loss functions rely on a metric of distance or similarity that encodes the correlation and variation of the variables.

Some other research directions consider acceleration of the computation of gradient for DNN optimizations by analyzing numerical aspects of processing. E.g., accelerating gradient method to non-convex optimization problems, reducing the precision of weights and gradients to accelerate the computations in DNNs.

The existing research on the loss functions mainly focuses on the loss-margin, robustness, and specific applications (e.g., many loss functions are designed for facial recognition [6], [9], [11]). And the aforementioned gradient approximation approaches focus on the simplification of the numerical computation of gradient. There are only a few published works that discuss the properties of the loss function gradients, which are the actual drivers of training DNNs. The next two sections of the paper will provide insights into the gradients of cross-entropy loss, and then propose two approximations for the cross-entropy gradients based on the properties of the gradient.

III. GRADIENTS OF CROSS-ENTROPY

A. REVISITING CROSS-ENTROPY LOSS

Cross-entropy is used ubiquitously in state-of-the-art DNNs. To discuss the properties of cross-entropy loss, it is necessary to briefly introduce its mathematical definition.

Suppose a discriminant problem has N classes. A neural network maps the input space to the representation space by $\mathcal{F} : \mathbb{R}^{d_x} \longmapsto \mathbb{R}^N$, where d_x is the dimension of the input space, and N is the dimension of the representation space, which must be the same as the number of classes. Suppose a data sample $\boldsymbol{x}$ is mapped by the network to its representation (scores) $\boldsymbol{S} = \mathcal{F}(\boldsymbol{x} \in \mathcal{X}_L) = [s_1, s_2, \ldots, s_N]^\intercal$, where $\mathcal{X}_L$ is the set of training samples labeled by L. The softmax nonlinearity is used to normalize the output $\boldsymbol{S}$ to a probability distribution $\boldsymbol{O} = [o_1, o_2, \ldots, o_N]^\intercal$, where o_i is defined as [2]:

$$o_i = P(c_p = i | \boldsymbol{x} \in \mathcal{X}_L) = \frac{e^{s_i}}{\sum_{l=1}^{N} e^{s_l}}, \tag{1}$$

where c_p denotes the predicted label. Cross-entropy loss is defined by

$$J = -\log(o_L) = -\log \frac{e^{s_L}}{\sum_{l=1}^{N} e^{s_l}}. \tag{2}$$

J is minimized when $o_L = 1$. It happens that the gradient of the loss with respect to scores has a simple form:

$$\nabla_{\boldsymbol{S}} J(\boldsymbol{S}) = [g_1, g_2, \ldots, g_N]^\intercal = \boldsymbol{O} - \boldsymbol{T}_L, \tag{3}$$

where $\boldsymbol{T}_L = [t_1, t_2, \ldots, t_N]^\intercal$ is a vector with entries $t_i = 0$ for $i \neq L$ and $t_L = 1$.

The success of the cross-entropy loss has been proven by a wide range of applications. It is used under various names, such as negative log-likelihood, softmax loss, mutual information loss, etc. [3]. However, only a few works try to explain the reason of its effectiveness. J.S. Bridle mentioned that the cross-entropy loss uses cross-class information and results in better performance for class discrimination than the usual within-class training method [2]. Y. LeCun *et al.* explained effectiveness from the viewpoint of energy-based learning. They interpret the numerator of (2) as an energy associated with the correct configuration and the denominator as a constructive factor which pushes the energies of the incorrect answers towards zero, i.e., the corresponding loss towards infinity [3]. We will further scrutinize the gradients of cross entropy and try to explain its effectiveness from the geometric point of view.

B. PROPERTIES OF THE GRADIENTS OF CROSS-ENTROPY LOSS

As the gradients of cross-entropy loss are vectors in an $\mathbb{R}^N$ space, we define $(x_1, x_2, \ldots, x_N)$ as the Cartesian coordinate of a point in the space.

Property 1: All gradients of cross-entropy loss are on the hyperplane $\sum_{l=1}^{N} x_l = 0$.

Proof: Since $\sum_{l=1}^{N} o_l = 1$ by (1) and $\sum_{l=1}^{N} t_l = 1$ by (3), thus $\sum_{l=1}^{N} x_l = \sum_{l=1}^{N} o_l - \sum_{l=1}^{N} t_l = 0$. □

A representation is the "worst" when $s_1 = s_2 = \cdots = s_N$, because it implies the prediction is most uncertain, i.e., $\forall i, o_i = P(c_p = i | \boldsymbol{x} \in \mathcal{X}_L) = 1/N$. Then we have the next property:

Property 2: All the gradients of cross-entropy are orthogonal to the most uncertain decision line $\{\eta \mathbf{1} | \eta \in \mathbb{R}\}$*, where* $\mathbf{1}$ *is an all-one vector with N entries.*

Proof: Since $\mathbf{1}^\intercal \cdot \nabla_{\boldsymbol{S}} J(\boldsymbol{S}) = 0$ by Property 1, Property 2 holds. □

Property 2 just rephrases Property 1.

Property 3: Assume that $\mathbb{E}[o_i] = \lambda, \forall i \neq L$ *then*

$$\boldsymbol{V}_L = \mathbb{E}[\nabla_{\boldsymbol{S}} J(\boldsymbol{S})] = \lambda(\mathbf{1} - N\boldsymbol{T}_L), \tag{4}$$

where $\mathbb{E}[\cdot]$ *denotes the expectation.*

Proof:

$$\begin{aligned}
\mathbb{E}[\nabla_{\boldsymbol{S}} J(\boldsymbol{S})] &\triangleq \mathbb{E}[\boldsymbol{O}] - \boldsymbol{T}_L \\
&= [\lambda, \cdots, 1 - (N-1)\lambda, \cdots \lambda]^\intercal \\
&\quad - [0, \cdots, 1, \cdots 0]^\intercal \\
&= \lambda \mathbf{1} - \lambda [0, \cdots, N, \cdots, 0]^\intercal \\
&= \lambda(\mathbf{1} - N\boldsymbol{T}_L).
\end{aligned}$$

□

The assumption that the expected values of probability for the classes not corresponding to the true label are the same is reasonable when the neural network is randomly initialized and on average balances treatment of classes. The representations conditioned on class L are symmetrically distributed around the s_L-axis. At the beginning of training, the prediction of the neural networks is a random guess. In this case $\lambda = 1/N$. At the end of training $o_L \gg o_i, \forall i \neq L$, and $\lambda = (1 - \mathbb{E}[o_L])/(N-1) \approx 0$.

To illustrate the properties of cross-entropy loss gradients, we use a group of synthetic representations generated by zero-mean unit-covariance Gaussian distribution to model the situation at the beginning of training in a three-class scenario. The negative gradients of cross-entropy loss are illustrated in Fig. 1. *They are all orthogonal to and drive the representations from the most-uncertain-decision line if the training is sufficiently long*. The orthogonality comes from Property 2. The increase of distance from line $\{\eta \mathbf{1} | \eta \in \mathbb{R}\}$ can be explained as follows.

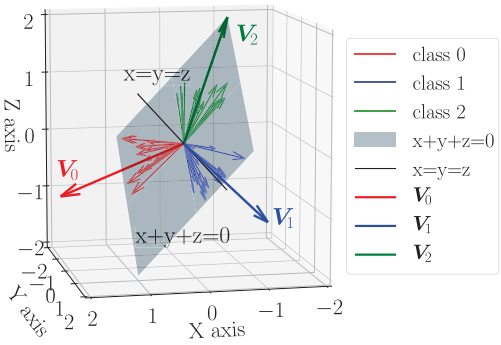


FIGURE 1. The negative gradients of cross-entropy loss, the plane $x + y + z = 0$, and the most-uncertain-decision line $x = y = z$. For the visualization purpose, the length of the cross-entropy gradients are enlarged 40 times.

Let us consider the process of *one-step* parameter updating for a *single* input sample using the gradient descent algorithm. Let $\boldsymbol{S}^{(0)} = [s_1^{(0)}, s_2^{(0)}, \ldots, s_N^{(0)}]^\intercal$ be the current coordinate of a representation for an input $\boldsymbol{x}$, and $\boldsymbol{S}^{(1)}$ be the updated representation for using the same training sample $\boldsymbol{x} \in \mathcal{X}_L$. After updating the network parameters by using the steepest descent method, $\boldsymbol{S}^{(1)} = \boldsymbol{S}^{(0)} - \gamma\, \mathbb{E}[\nabla_{\boldsymbol{S}^{(0)}} J(\boldsymbol{S}^{(0)})]$, where $\gamma > 0$ is the learning rate. Let $d^2(\mathbf{1}, \boldsymbol{S})$ be the squared distance of $\boldsymbol{S}$ from $\{\eta \mathbf{1} | \eta \in \mathbb{R}\}$. Then we have

$$\begin{aligned}
&d^2\left(\mathbf{1}, \boldsymbol{S}^{(1)}\right) - d^2\left(\mathbf{1}, \boldsymbol{S}^{(0)}\right) \\
&= \|\boldsymbol{S}^{(1)} - \mathrm{Proj}_{\mathbf{1}} \boldsymbol{S}^{(1)}\|_2^2 - \|\boldsymbol{S}^{(0)} - \mathrm{Proj}_{\mathbf{1}} \boldsymbol{S}^{(0)}\|_2^2 \\
&= \|\boldsymbol{S}^{(1)} - \frac{1}{N}\mathbf{1}^\intercal \boldsymbol{S}^{(1)} \mathbf{1}\|_2^2 - \|\boldsymbol{S}^{(0)} - \frac{1}{N}\mathbf{1}^\intercal \boldsymbol{S}^{(0)} \mathbf{1}\|_2^2 \\
&= \gamma^2 \lambda^2 N(N-1) - 2\gamma\lambda(\mathbf{1} - N\boldsymbol{T}_L)^\intercal (\boldsymbol{S}^{(0)} - \frac{1}{N}\mathbf{1}^\intercal \boldsymbol{S}^{(0)} \mathbf{1}),
\end{aligned}$$

where, $\mathrm{Proj}_{\boldsymbol{u}}(\boldsymbol{v})$ represents the projection of vector $\boldsymbol{v}$ onto vector $\boldsymbol{u}$. Since

$$\begin{aligned}
&(\mathbf{1} - N\boldsymbol{T}_L)^\intercal \left(\boldsymbol{S}^{(0)} - \frac{1}{N}\mathbf{1}^\intercal \boldsymbol{S}^{(0)} \mathbf{1}\right) \\
&= -N s_L^{(0)} + \left(\sum_{l=1}^{N} s_l^{(0)}\right) \boldsymbol{T}_L^\intercal \mathbf{1} \\
&= -N s_L^{(0)} + \sum_{l=1}^{N} s_l^{(0)},
\end{aligned}$$

we obtain

$$d^2\left(\mathbf{1}, \boldsymbol{S}^{(1)}\right) - d^2\left(\mathbf{1}, \boldsymbol{S}^{(0)}\right) = \gamma^2\lambda^2 N(N-1) - 2\gamma\lambda\left(\sum_{l=1}^{N} s_l^{(0)} - N s_L^{(0)}\right).$$

For $\lambda = 0$ the distance after updating does not change. For $\lambda > 0$ the distance increase if $-2\left(\sum_{l=1}^{N} s_l^{(0)} - N s_L^{(0)}\right) + \gamma\lambda N(N-1) > 0$. A sufficient condition that the updated representation in the next step is farther away from $\{\eta\mathbf{1}|\eta \in \mathbb{R}\}$ is $s_L^{(0)} > (1/(N-1))\sum_{l=1,\, l\neq L}^{N} s_l^{(0)}$, i.e., $s_L^{(0)}$ component should be larger than the arithmetic mean of the other components in $\boldsymbol{S}^{(0)}$. After sufficiently long training this condition will be satisfied since (4) drives s_L towards $+\infty$ and all other components in $\boldsymbol{S}$ towards $-\infty$.

IV. VANISHING GRADIENTS OF CROSS-ENTROPY LOSS

The gradient can be characterized by direction and length (intensity). In addition to the three discussed properties, we investigate the length of cross-entropy gradient next.

The problem of vanishing gradients is well-known in training of deep neural networks, and it is conventionally referred to as the reduction of length of gradient caused by the saturating activation functions and small singular values of the Jacobian matrix associated with the transformation from the features at one level into the features at the next level in backpropagation [21]. It is the one of the key factors that prevented training a very deep net in the early development of artificial neural networks [22].

The history of overcoming the vanishing gradient problem suggests that it is important to retain the intensity of the gradient during training. Based on the chain-rule that is used for obtaining the gradients w.r.t. the network parameters, many existing proposals focus on mitigating the vanishing gradient in the backpropagation process but omit the first term of the chain-rule—the gradient of the loss function, which drives the training. Practically, the choice of a loss function determines its gradient and could cause vanishing gradients. The three properties discussed in last section focus on the direction of the cross-entropy gradient; its intensity will be analyzed next.

Property 4: The length of the expected gradient vector of cross-entropy is $\lambda\sqrt{N(N-1)}$.

Proof: The length of the expected gradients can be expressed by

$$\begin{aligned}\|\mathbb{E}[\nabla_{\boldsymbol{S}} J(\boldsymbol{S})]\|_2 &= \|\lambda(\mathbf{1} - N\boldsymbol{T}_L)\|_2 \\ &= \sqrt{(N-1)\lambda^2 + \lambda^2(1-N)^2} \\ &= \lambda\sqrt{N(N-1)}. \end{aligned} \tag{5}$$

□

Equation (5) indicates that the L_2 norm (length) of the expected gradients of cross-entropy loss has a linear relation to λ. As λ is the expected probability for the classes that are *not* corresponding to the true label, it has the largest value $1/N$ at the beginning of the training assuming an unbiased initialization. In this case $\|\mathbb{E}[\nabla_{\boldsymbol{S}} J(\boldsymbol{S})]\|_2 = \sqrt{1 - 1/N}$. λ approaches zero at the end of the training if the training is effective (i.e., most of the training samples are correctly classified). In this case, $\|\mathbb{E}[\nabla_{\boldsymbol{S}} J(\boldsymbol{S})]\|_2$ approaches zero accordingly.

The analysis above discloses the vanishing gradient caused by cross-entropy—the intensity of the gradient decays linearly as the confidence of the classification grows. In practice, the norm of the expected gradient of cross-entropy is seen to diminish very quickly (even faster than exponential) with the training iterations (see Fig. 2). As the training goes on, the "force" that drives the training approaches zero and the training progress stagnates. The very short length of the cross-entropy gradient after a few iterations can be considered as another kind of vanishing gradient; changing the activation functions or the architectures of the network barely help, because the shortening happens in the first term of the chain rule of backpropagation and only depends on the loss function.

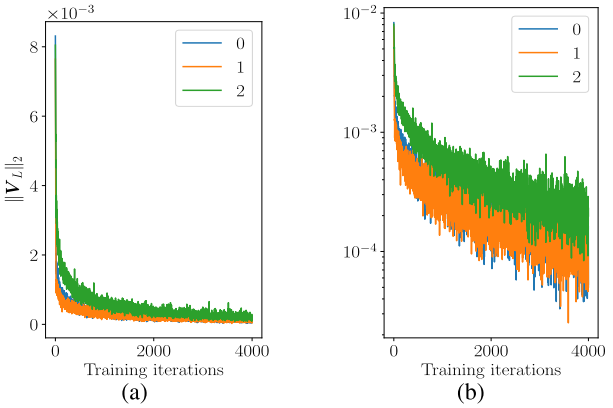


FIGURE 2. The length of expected gradient of cross-entropy in the first 4,000 training batches of MNIST dataset. The values on the curves are L_2 norm of average gradient of cross-entropy loss ($\|\boldsymbol{V}_L\|_2 = \|\mathbb{E}[\nabla_{\boldsymbol{S}} J(\boldsymbol{S})]\|_2$) for $L = 0, 1, 2$ over one batch. The batch size of the experiments is 128. The curves in the figure are averages of 20 repetitive experiments. Subfigure (b) presents the same data as subfigure (a) but in logarithm scale in the vertical axis. The curves for $L = 0$ (blue) and $L = 1$ (orange) are almost completely overlapped. The other classes of MNIST have the similar trend of the length of average gradient as the three classes shown in the figures.

V. THE FUNCTIONS THAT APPROXIMATE GRADIENTS OF CROSS-ENTROPY LOSS

The properties of the cross-entropy gradient motivate us to propose two functions that eliminate the forward pass of the loss function and generate the vectors that replace the gradients of cross-entropy loss. The purpose of the approximations is to avoid the vanishing gradient of the cross-entropy loss and to circumvent the calculation of the exponential and logarithm in (1) and (2), thereby, simplifying the procedure of generating the (gradient) vectors that are used to train the networks.

Approximation 1:

$$\widehat{\boldsymbol{G}}_1 = \frac{\boldsymbol{V}_L}{\|\boldsymbol{V}_L\|}. \tag{6}$$

This approximation simply uses the unit vector in the direction of the expectation of cross-entropy loss gradient. The length of the gradient of cross-entropy loss decays quickly with the increase of the prediction confidences for the network (Fig. 2). In contrast, the length of $\widehat{\boldsymbol{G}}_1$ does not change, and it keeps pushing the representations toward infinity, far from $\{\eta\mathbf{1}|\eta \in \mathbb{R}\}$. This strategy may cause overfitting however.

Approximation 2:

$$\widehat{\boldsymbol{G}}_2 = -\boldsymbol{T}_L. \tag{7}$$

This is a coarser approximation of the gradient of cross-entropy loss. The form of the vector $\widehat{\boldsymbol{G}}_2$ is rather simple—we change the sign of the entry that has value of one in $\boldsymbol{T}_L$, and inject the vector to the backward pass of training neural networks. It reduces the computational complexity drastically, and might be the simplest way of generating the vectors that could drive the training of DNNs.

$\widehat{\boldsymbol{G}}_1$ and $\widehat{\boldsymbol{G}}_2$ have three notable properties: (i) They are "noise-free," because they only depend on the labels of a training set but not on an individual data or its representation explicitly. (ii) The length (intensity) of the vectors they generate for training the networks is unit. (iii) They simply push the representations in proper directions corresponding to the labels. The usefulness of these properties will be shown by the experiments in the next section.

VI. EXPERIMENTS

A. ACCELERATION OF TRAINING USING $\widehat{G}_1$ AND $\widehat{G}_2$

Since $\widehat{\boldsymbol{G}}_1$ and $\widehat{\boldsymbol{G}}_2$ are noise-free if the training labels are reliable, the networks could be trained without smoothing the gradient. To verify this possibility we conduct the following experiments on the CIFAR10 dataset.

Fig. 3 illustrates the training and test errors of CIFAR10 dataset by a Wide-ResNet(28-10)[1] [23] using $\widehat{\boldsymbol{G}}_1$, $\widehat{\boldsymbol{G}}_2$, and cross-entropy with different optimizers and learning strategies. For the purpose of investigating the noise-free effects, we choose stochastic gradient descent (SGD) as the optimizer in the experiments producing Figs. 3(a), (b), (c), and (d) and turn off the smoothing of $\widehat{\boldsymbol{G}}_1$ and $\widehat{\boldsymbol{G}}_2$ (set the momentum coefficient of the optimizer to zero). For comparison, cross-entropy is employed with the parameters suggested in [23], i.e., the momentum coefficient is set as 0.9. By observing Fig. 2, one can conclude that the range of lengths of the gradients for cross-entropy is quite different from $\widehat{\boldsymbol{G}}_1$ and $\widehat{\boldsymbol{G}}_2$ (they have constant length of 1). These contrasting ranges imply that they should have very different learning rates. We set the initial learning rate for $\widehat{\boldsymbol{G}}_1$ and $\widehat{\boldsymbol{G}}_2$ to be the largest number in the set $\left\{1 \times 10^{-q} \,|\, q = 1, 2, \ldots, 10\right\}$ that keeps the training stable, i.e.,1×10^{-7}. The learning rate for cross-entropy is chosen as 0.1, which is also suggested by [23].

To further eliminate the setting difference of algorithms to be compared, we use Adam [24] as the optimizer in the second group of experiments, because Adam could help to stabilize the training and, more importantly, it provides the parameter updating magnitude that is invariant to rescaling of the gradient [24]. Thus, it is possible to fairly compare the training processes using $\widehat{\boldsymbol{G}}_1$, $\widehat{\boldsymbol{G}}_2$, and cross-entropy choosing the same initial learning rate. The initial learning rates for $\widehat{\boldsymbol{G}}_1$, $\widehat{\boldsymbol{G}}_2$, and cross-entropy are all 0.001 in this group of experiments. The comparison results are shown in the second row of Figs. 3 ((e), (f), (g), and (h)).

We also compare the performances of $\widehat{\boldsymbol{G}}_1$, $\widehat{\boldsymbol{G}}_2$, and the cross-entropy gradient using different learning rate decay strategies: (i) The learning rates are kept constant (Figs. 3(a) and (e)). (ii) To make the cross-entropy based algorithms converge faster and deeper, the learning rates are decayed by γ_0/m, where m is the training epoch number (Figs. 3(b) and (f)). (iii) The learning rates are decayed to $(0.2)^q \times \gamma_0$, where $q = 1, 2, 3$ at epochs 60, 120, and 160 respectively (Figs. 3(c) and (g)). This strategy is referred in [23]. (iv) The total number of training epochs for $\widehat{\boldsymbol{G}}_1$ and $\widehat{\boldsymbol{G}}_2$ is reduced by half (100 epochs), and learning rates are decayed to $(0.2)^q \times \gamma_0$, where $q = 1, 2, 3$ at epochs 30, 60, and 80 respectively (Figs. 3(d) and (h)).

One can see that the training errors for $\widehat{\boldsymbol{G}}_1$ and $\widehat{\boldsymbol{G}}_2$ decay much faster than those for the cross-entropy gradient in most of the scenarios, e.g., in Fig. 3(f) $\widehat{\boldsymbol{G}}_1$ and $\widehat{\boldsymbol{G}}_2$ achieve the training error of 2% at about 36^{th} epoch[2] but the cross-entropy gradient achieves the same training error at the 60^{th} epoch. $\widehat{\boldsymbol{G}}_1$ and $\widehat{\boldsymbol{G}}_2$ save about 24 epochs (i.e., one hour two minutes training time on the GPU of our workstation[3]). In Figs. 3(d) and (h), the numbers of training epochs of $\widehat{\boldsymbol{G}}_1$ and $\widehat{\boldsymbol{G}}_2$ are halved compared to the cross-entropy loss case, though they may lose only 1% of the test accuracy[4].

These experiments show the usefulness of the properties (i) and (ii) of $\widehat{\boldsymbol{G}}_1$ and $\widehat{\boldsymbol{G}}_2$ at the end of Section V. The training errors approach zero without need for smoothing the gradient and decrease rapidly when using $\widehat{\boldsymbol{G}}_1$ and $\widehat{\boldsymbol{G}}_2$.

B. VISUALIZATIONS OF THE REPRESENTATIONS AND NETWORK PARAMETERS FOR A FULLY-CONNECTED NETWORK (FC-NET)

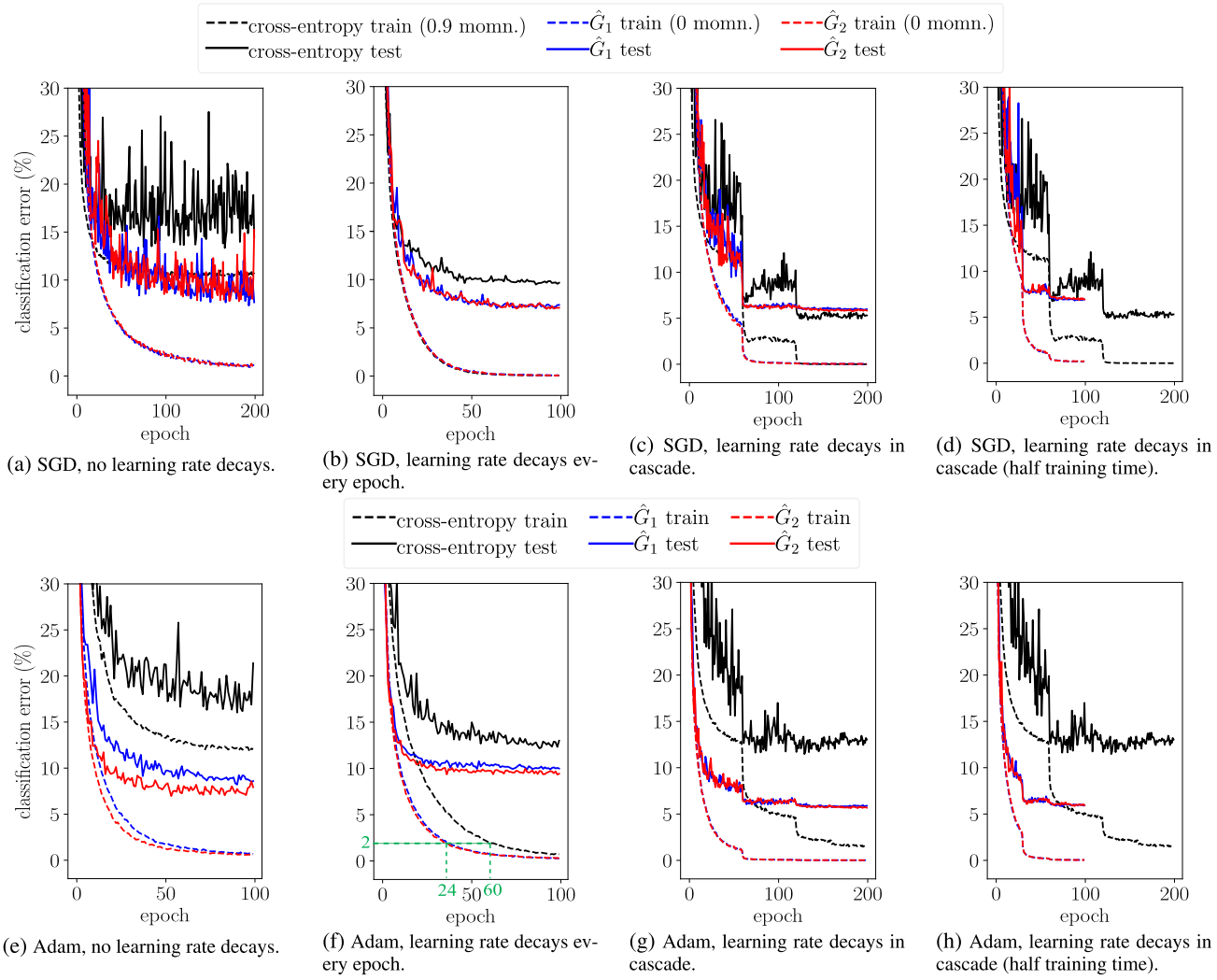
Fig. 4 displays the trajectories of class mean for "0", "1", "2" for the MNIST dataset in the 3-D representation space. Fig. 4(a) shows the ideal trajectories obtained by iterating $\boldsymbol{S}^{(q+1)} = \boldsymbol{S}^{(q)} + \gamma \nabla_{\boldsymbol{S}^{(q)}} J(\boldsymbol{S}^{(q)})$ without training a network, where q is the iteration number. Fig. 4(b) shows the class

[1]We downloaded the source code written by the authors of [23] from *https://github.com/szagoruyko/wide-residual-networks*.

[2]$\widehat{\boldsymbol{G}}_1$ achieves 2% training error at the 34^{th} epoch, and $\widehat{\boldsymbol{G}}_2$ at 38^{th}.

[3]The work station has an Intel Core i7-8700K CPU, 16 GB memory, and a GeForce 1080Ti GPU.

[4]The test accuracies for $\widehat{\boldsymbol{G}}_1$ and $\widehat{\boldsymbol{G}}_2$ are 94.18% and 94.15% respectively in Fig. 3(h). The best test accuracies for $\widehat{\boldsymbol{G}}_1$, $\widehat{\boldsymbol{G}}_2$, and cross-entropy gradient are 94.87%, 94.72%, and 95.17% which was obtained in Fig. 3(d).



(a) SGD, no learning rate decays.
(b) SGD, learning rate decays every epoch.
(c) SGD, learning rate decays in cascade.
(d) SGD, learning rate decays in cascade (half training time).
(e) Adam, no learning rate decays.
(f) Adam, learning rate decays every epoch.
(g) Adam, learning rate decays in cascade.
(h) Adam, learning rate decays in cascade (half training time).

FIGURE 3. **The classification errors obtained by $\widehat{G}_1$, $\widehat{G}_2$, and cross-entropy gradient with different learning rate decay strategies and optimizers, where (a), (b), (c), and (d) use SGD, and (e), (f), (g), and (h) use Adam. The momentum (momn.) for $\widehat{G}_1$ and $\widehat{G}_2$ using SGD is zero; the momentum for cross-entropy gradient using SGD is 0.9. (a) and (e) have no learning rate decay. (b) and (f) decay learning rates as γ_0/m, where m is the number of epoch. The learning rates in (c) and (g) reduce to $(0.2)^q \times \gamma_0$, where $q = 1, 2, 3$ at epochs 60, 120, and 160. In (d) and (h) the learning rates reduce to $(0.2)^q \times \gamma_0$, where $q = 1, 2, 3$ at epochs 30, 60, and 80 when using $\widehat{G}_1$ and $\widehat{G}_2$. Note that the training error curves (dash lines) may overlap each other which make them difficult to distinguish, e.g., the three dash lines in (b) are almost completely overlapped, and in most of the figures the blue and red dash lines overlap. Note that the epoch scales differ.**

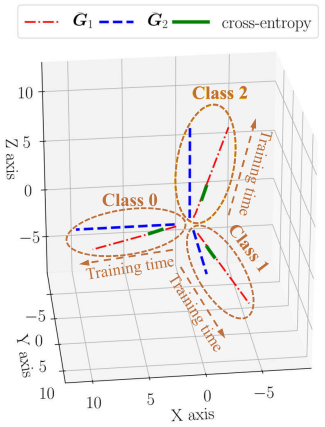
mean trajectories in the representation space when training an FC-net. One can see that the trajectories for cross-entropy loss are shortest (bold green lines). It indicates that λ decays quickly with the prediction confidence increase. The representations, therefore, move little when close to the end of training. By contrast, the class means keep moving farther from $\{\eta\mathbf{1}|\eta \in \mathbb{R}\}$ by using $\widehat{G}_1$ and $\widehat{G}_2$ forming different trajectories because $\widehat{G}_1$ and $\widehat{G}_2$ both have unit length but different directions.

Up to this point, we have shown the successful training of DNNs by using $\widehat{G}_1$ and $\widehat{G}_2$; additionally, we are interested in evolution of the parameters in the network, since the driving force of training have a constant intensity in this case. To demonstrate the parameter updating behavior using $\widehat{G}_1$ and $\widehat{G}_2$, we randomly chose 100 parameters from the last layer of the same FC-net trained by $\widehat{G}_1$, $\widehat{G}_2$, and cross-entropy gradient for the MNIST dataset, and Fig. 5 illustrates the evolution of these 100 parameters. One can see that these parameters trained by $\widehat{G}_1$ and $\widehat{G}_2$ do not converge to fixed values. About half of the parameters drift toward infinity, and the others drift toward minus infinity. By contrast, the parameters trained by cross-entropy gradient converge to fixed values. The test accuracy of the networks trained by $\widehat{G}_1$ and $\widehat{G}_2$ achieves 98.72% and 98.65% in 100 epochs compared to the 98.47% for the cross-entropy case. Therefore, the success in training a network does not imply the convergence of network parameters to fixed values. These experiments justified the property (iii) at the end of Section V.

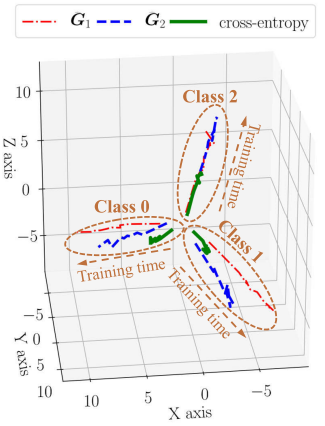
The reason for the success of training a DNN by these unit vectors is that the decision regions for the max-out

TABLE 1. The classification accuracy for different approximations/loss functions on OCT dataset (mean ± std %).

Approx./Loss	$\widehat{G}_1$	$\widehat{G}_2$	cross-entropy	hinge [25]	hinge-square [26]
Test acc.	99.08 ± 0.17	99.14 ± 0.16	99.06 ± 0.15	98.82 ± 0.27	98.96 ± 0.13



(a) Ideal trajectories of class means in a 3-D space.



(b) The class mean trajectories in a FC-net representation space.

FIGURE 4. These figures show the class mean trajectories of "0", "1", and "2" in MNIST dataset in the 3-D representation space. (a) displays the ideal trajectories, and (b) displays the trajectories in a training process.

classification strategy (i.e., the prediction of the network for the label corresponding to an input is based on its representation $\boldsymbol{S}$, and the final prediction is $\hat{L} = \text{argmax}_i\, s_i$, where s_i ($i = 1, 2, \ldots N$) is an element in the representation vector $\boldsymbol{S}$) are open-regions that include infinity. Fig. 6 illustrates the decision boundaries and the proposed driving vectors in a three-class problem. A representation that falls into a region provides a prediction label associated with the region. E.g., a representation at (0, 0, 1) will be predicted as class 2, as well as the representation at (0, 0, z), where $z > 0$ and possibly very large. Accordingly, the parameters that map the input to the representation can be large, too. The similar conclusion can be generalized to high dimensional scenarios. The decision regions for the max-out decision strategy in a high dimensional space are hyperplane bouned and contain exactly one positive coordinate semiaxis and $N - 1$ negative coordinate semiaxes.

C. APPLICATION WITH CNNs–OPTICAL COHERENCE TOMOGRAPHY (OCT) DATASET CASE

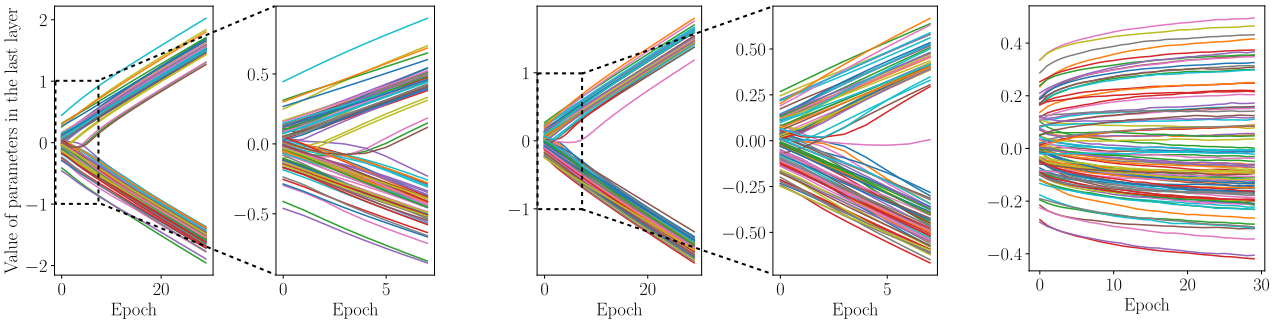
We use an optical coherence tomography dataset and an eighteen layer ResNet (ResNet-18) [27] to demonstrate the effectiveness of the proposed approximations and compare them with other conventional loss functions. The OCT dataset contains 84,485 retinal images, in which 1,000 samples are used as a test set. There are four classes, namely, Choroidal Neovascularization (CNV), Diabetic Macular Edema (DME), Drusen, and Normal [28]. OCT is a current standard tool for the diagnosis of some of the leading causes of blindness worldwide [28].

Table 1 compares the classification accuracy of the proposed approximations and of other frequently used loss functions. The training for every approximation and loss function is repeated five times. The accuracies are illustrated by mean ± standard deviation format.

As one can see in Table 1, the two approximations ($\widehat{\boldsymbol{G}}_1, \widehat{\boldsymbol{G}}_2$) achieved comparable classification accuracies to the frequently used loss functions including cross-entropy loss, hinge loss [25], and hinge-square loss [26]. Moreover, the classification accuracies of our proposed approximations are better than the result (96.6% test accuracy) reported in [28], which published the dataset.

D. COMPUTATIONAL COMPLEXITY OF THE APPROXIMATIONS

We anticipate that the computational complexity using the proposed approximations is lower than for the frequently used cross-entropy loss function because we avoid the forward calculation. $\widehat{\boldsymbol{G}}_1$ and $\widehat{\boldsymbol{G}}_2$ both have the computational complexity of $\mathcal{O}(n)$ (n is the batch size) and circumvent the computation of exponential functions. A group of timing experiments were conducted to test the hypothesis. For a fair comparison, the two approximations and forward-backward pass of cross-entropy loss were implemented by the NumPy module [29] without the precompiling headers of the cross-entropy loss functions. The input of the approximations/loss function based training was a randomly generated mini batch, which had 100 classes and size of 256. The experiments are repeated 1,000 times for each approximation/loss function, and they were done using a workstation that has an Intel Core i7-8700K CPU and 16 GB memory. The timing results are presented in Table 2. It can be seen that the computational time of $\widehat{\boldsymbol{G}}_1$ and $\widehat{\boldsymbol{G}}_2$ are about one-tenth that of cross-entropy. Although the calculation of the loss function



(a) $\widehat{G}_1$ (b) $\widehat{G}_2$ (c) cross-entropy

FIGURE 5. The evolution of 100 parameters which are randomly chosen in the last layer of FC nets trained by (a) $\widehat{G}_1$, (b) $\widehat{G}_2$, and (c) cross-entropy gradient for MNIST dataset (best viewed with color images). One can see that the parameters trained by $\widehat{G}_1$ and $\widehat{G}_2$ do not converge to fixed values, however, the network can be successfully trained.

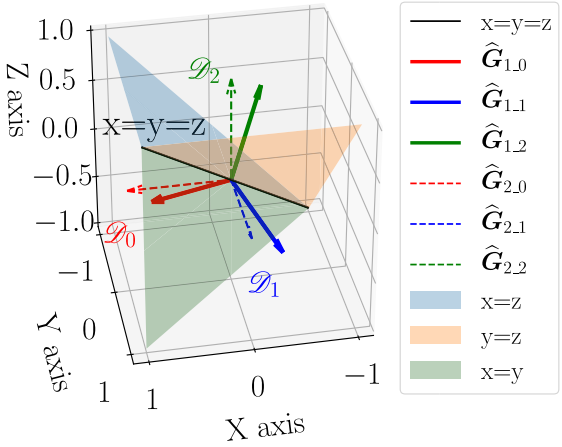


FIGURE 6. The decision boundaries in the representation space for a three-class problem using the max-out decision strategy; they are defined by $x = z$, $y = z$, and $x = y$ in the 3-D space. The corresponding decision regions of class 0, 1, and 2 are $\mathscr{D}_0$, $\mathscr{D}_1$, and $\mathscr{D}_2$, respectively. The figure also show the proposed approximation vectors $\widehat{G}_1$ and $\widehat{G}_2$.

TABLE 2. The comparison of computational time (ms) for the gradient approximations and cross-entropy case. The results are the averages over 1,000 experiments.

Approx./Loss	$\widehat{G}_1$	$\widehat{G}_2$	cross-entropy
Comput. time	0.027	0.012	0.245

is a small proportion in the computation of forward and backward propagation, and the computational complexity of the network is mainly determined by the architecture of the network, $\widehat{G}_1$ and $\widehat{G}_2$ overcome the vanishing gradient of cross-entropy loss and increase the training speed.

VII. MORE GENERAL APPROXIMATIONS AND RELATION TO LABEL-SMOOTHING REGULARIZATION (LSR)

A. GENERALIZATION

Based on the analysis and experiments above, we are proposing vectors that generalizes the family of potential approximations of the cross-entropy gradient. By defining

$$\boldsymbol{T}_0 = -\alpha \boldsymbol{T}_L + \beta \mathbf{1}, \quad \text{where } \alpha, \beta > 0,\ \beta < \alpha, \tag{8}$$

the proposed approximation is

$$\widehat{\boldsymbol{G}}_0 = \frac{\boldsymbol{T}_0}{\|\boldsymbol{T}_0\|}. \tag{9}$$

The constraints in (8) guarantee that the negative gradient is largest in the $\boldsymbol{T}_L$ direction. By comparing $\widehat{\boldsymbol{G}}_1$, $\widehat{\boldsymbol{G}}_2$, and $\widehat{\boldsymbol{G}}_0$, one can conclude that $\widehat{\boldsymbol{G}}_1$ and $\widehat{\boldsymbol{G}}_2$ are the special cases of $\widehat{\boldsymbol{G}}_0$, where $\alpha = \lambda N$, $\beta = \lambda$ for $\widehat{\boldsymbol{G}}_1$, and $\alpha = 1$, $\beta = 0$ for $\widehat{\boldsymbol{G}}_2$.

B. THE RELATION TO LSR

R. Szegedy *et al.* proposed a label-smoothing technique in [7] for regularization of the networks. In simple words, the technique replaces $\boldsymbol{T}_L = [t_1, t_2, \ldots, t_N]^\intercal$ ($t_i = 0$ for $i \neq L$ and $t_L = 1$) with $\boldsymbol{T'}_L = [t'_1, t'_2, \ldots, t'_N]^\intercal$ ($t'_i = \epsilon/N$ for $i \neq L$ and $t'_L = 1 - \epsilon(N-1)/N$). The authors interpreted this technique as reducing the confidence of the prediction, or adding one more term in the loss function which penalizes the deviation of predicted label distribution from a uniform distribution with parameter N [7].

Since we can write $\boldsymbol{T'}_L = (1-\epsilon)\boldsymbol{T}_L + (\epsilon/N)\mathbf{1}$, the expectation of the gradient for LSR can be written as

$$\begin{aligned}
&\mathbb{E}[\nabla_{\boldsymbol{S}} J(\boldsymbol{S})]_{LSR} \\
&= \mathbb{E}[\boldsymbol{O}] - \boldsymbol{T'}_L \\
&= [\lambda, \cdots, 1-(N-1)\lambda, \cdots \lambda]^\intercal - (1-\epsilon)\boldsymbol{T}_L - \frac{\epsilon}{N}\mathbf{1} \\
&= \lambda \mathbf{1} + [0, \cdots, 1-\lambda N, \cdots, 0]^\intercal - (1-\epsilon)\boldsymbol{T}_L - \frac{\epsilon}{N}\mathbf{1} \\
&= -(\lambda N - \epsilon)\boldsymbol{T}_L + \left(\lambda - \frac{\epsilon}{N}\right)\mathbf{1} \\
&= \left(\lambda - \frac{\epsilon}{N}\right)(\mathbf{1} - N\boldsymbol{T}_L).
\end{aligned} \tag{10}$$

One can conclude that (10) is a special case of $\widehat{\boldsymbol{G}}_0$, where $\alpha = \lambda N - \epsilon$ and $\beta = \lambda - \epsilon/N$. By comparing (4) and (10), one can further recognize that the expectation of the gradient generated by LSR has the same direction as $\boldsymbol{V}_L$ and $\widehat{\boldsymbol{G}}_1$ but is modulated by $(\lambda - \epsilon/N)$. This implies that one has to carefully choose the value of ϵ, because if $\epsilon < \lambda N$, the gradient will be zero when λ decreases by training to ϵ/N (gradient vanishes), and if $\epsilon > \lambda N$, the gradient will become zero

when λ increases by training to value ϵ/N. This increase of confidence for incorrect labels is generally undesirable but can be useful to recover from training overfitting. Moreover, LSR still calculates (1)–(3), but our proposed approaches do not need these calculations.

VIII. DISCUSSION AND CONCLUSION

In this paper, we explored the geometric properties of the gradient generated by the cross-entropy loss function, and show their implications to the process of classification. The length of the cross-entropy gradient decays rapidly as the training iteration proceed. Based on the properties of the cross-entropy gradient, two approximations of the gradient of cross-entropy loss were proposed. Obtaining the approximations does not need the calculation of the loss function. The vectors driving the representation training of DNNs are directly generated by knowing only the correct labels. They preserve the properties related to the direction of cross-entropy gradient.

We have shown three properties from the theoretical analysis of the approximations. First, they are ''noise-free'' and depend on the labels of the training samples only. Second, the length (intensity) of the approximations have unit value; thereby, they avoid the vanishing gradient problem. Third, our proposed approaches obtain the representations similar to those obtained when using cross-entropy.

One assumption underlying the training using $\widehat{\boldsymbol{G}}_1$ and $\widehat{\boldsymbol{G}}_2$ and the noise-free claim is that the training labels are reliable. Note that $\widehat{\boldsymbol{G}}_1$ and $\widehat{\boldsymbol{G}}_2$ depend on the label of the training set only. If the labels of the training set are incorrect, they may cause greater negative impacts to the training comparing to the ordinary training based on the cross-entropy loss, because the directions of the gradients for the incorrect labels are wrong. Moreover, as (6) and (7) do not rely on J, the calculation of neither $\boldsymbol{O}$ nor J is necessary.

The experimental results justified the usefulness of the proposed method. The training by the proposed approximations achieved comparable classification accuracy to other conventional loss functions and accelerated the training on some datasets. By observing the behavior of the training using the approximation functions, we argue that it is possible to use the pre-defined vectors to drive the training without defining a loss function explicitly. Furthermore, the success of training does not necessarily imply the convergence of network parameters to fixed values. The timing experiments justify that the proposed approximations save computational time. $\widehat{\boldsymbol{G}}_2$ might be the simplest way to generate the vectors that could train the DNNs. A general approximation is proposed at the end of the paper. It unifies the two proposed approximations and label-smoothing regularization.

One weakness of the proposed approximations might be the capacity of generalization. We focused on training accuracy in this paper, but the success of training does not in general imply a good generalization, since the generalization of DNNs is still a complicated problem [30]. The other potential problem is the adaptation of the values of network parameters to the large intensity of $\widehat{\boldsymbol{G}}_1$ and $\widehat{\boldsymbol{G}}_2$, especially in the last layer. There are two strategies to solve this problem: either using a smaller learning rate or larger standard deviation of the initial values in the last layer. In the experiments on MNIST dataset, we used the standard Xavier initialization and the learning rate of 1.0×10^{-3}. In the experiments on CIFAR10 dataset, the initial learning rate was 1.0×10^{-7}, and the standard deviation of the initial values for the last FC-layer was 10. For the OCT dataset, the initial learning rate was 1.0×10^{-5}, and Xavier initialization was used.

The other potential problem caused by the large length of the approximation vectors is the adaptation of values of network initialization and the large intensity of $\widehat{\boldsymbol{G}}_1$ and $\widehat{\boldsymbol{G}}_2$, especially the values in the last layer.

As the representations obtained by the proposed approximations are similar to but different from ones obtained using cross-entropy loss, they can potentially improve the robustness of the trained DNNs against adversarial test samples, i.e., against test samples that can mislead the DNNs although they are very close to the samples that the DNNs correctly predict. More properties of the proposed approximations need to be explored in the future. Our novel interpretation and analysis can provide further insights to energy- or metric-based loss functions and be helpful to understand the behavior of the DNNs.

REFERENCES

[1] A. Karpathy. (2019). *Convolutional Neural Networks for Visual Recognition*. Accessed: Mar. 16, 2020. [Online]. Available: http://cs231n.github.io/optimization-1/
[2] J. S. Bridle, ''Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition,'' in *Neurocomputing*. Berlin, Germany: Springer, 1990, pp. 227–236.
[3] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang, ''A tutorial on energy-based learning,'' *Predicting Structured Data*, vol. 1, pp. 815–819, Aug. 2006.
[4] L. Li, M. Doroslovacki, and M. H. Loew, ''Loss functions forcing cluster separations for multi-class classification using deep neural networks,'' in *Proc. 53rd Asilomar Conf. Signals, Syst., Comput.*, Nov. 2019, pp. 2106–2110.
[5] L. Li, M. Doroslovacki, and M. H. Loew, ''Discriminant analysis deep neural networks,'' in *Proc. 53rd Annu. Conf. Inf. Sci. Syst. (CISS)*, Mar. 2019, pp. 1–6.
[6] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu, ''CosFace: Large margin cosine loss for deep face recognition,'' in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5265–5274.
[7] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, ''Rethinking the inception architecture for computer vision,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 2818–2826.
[8] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, ''Focal loss for dense object detection,'' in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2980–2988.
[9] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, ''SphereFace: Deep hypersphere embedding for face recognition,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 212–220.
[10] W. Liu, Y. Wen, Z. Yu, and M. Yang, ''Large-margin softmax loss for convolutional neural networks,'' in *Proc. ICML*, vol. 2, p. 7, Jun. 2016.
[11] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, ''A discriminative feature learning approach for deep face recognition,'' in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 499–515. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-46478-7_31#citeas
[12] Y. Rubner, C. Tomasi, and L. J. Guibas, ''The earth mover's distance as a metric for image retrieval,'' *Int. J. Comput. Vis.*, vol. 40, no. 2, pp. 99–121, Nov. 2000.

[13] L. Hou, C.-P. Yu, and D. Samaras, "Squared earth mover's distance-based loss for training deep neural networks," 2016, *arXiv:1611.05916*. [Online]. Available: http://arxiv.org/abs/1611.05916
[14] X. Driancourt, L. Bottou, and P. Gallinari, "Learning vector quantization, multi layer perceptron and dynamic programming: Comparison and cooperation," in *Proc. Seattle Int. Joint Conf. Neural Netw. (IJCNN)*, vol. 2, 1991, pp. 815–819.
[15] Y. LeCun and F. J. Huang, "Loss functions for discriminative training of energy-based models," in *Proc. AIStats*, vol. 6, 2005, p. 34.
[16] B.-H. Juang, W. Hou, and C.-H. Lee, "Minimum classification error rate methods for speech recognition," *IEEE Trans. Speech Audio Process.*, vol. 5, no. 3, pp. 257–265, May 1997.
[17] G. Chechik, V. Sharma, U. Shalit, and S. Bengio, "Large scale online learning of image similarity through ranking," *J. Mach. Learn. Res.*, vol. 11, pp. 1109–1135, Mar. 2010.
[18] J. Goldberger, G. E. Hinton, S. T. Roweis, and R. R. Salakhutdinov, "Neighbourhood components analysis," in *Proc. Adv. Neural Inf. Process. Syst.*, 2005, pp. 513–520.
[19] R. Salakhutdinov and G. Hinton, "Learning a nonlinear embedding by preserving class neighbourhood structure," in *Artificial Intelligence and Statistics*. San Juan, PR, USA: PMLR, 2007, pp. 412–419.
[20] M. Dorfer, R. Kelz, and G. Widmer, "Deep linear discriminant analysis," 2015, *arXiv:1511.04707*. [Online]. Available: http://arxiv.org/abs/1511.04707
[21] J. F. Kolen and S. C. Kremer, "Gradient flow in recurrent nets: The difficulty of learning longterm dependencies," in *A Field Guide to Dynamical Recurrent Networks*. Piscataway, NJ, USA: IEEE Press, 2001, pp. 237–243.
[22] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.
[23] S. Zagoruyko and N. Komodakis, "Wide residual networks," 2016, *arXiv:1605.07146*. [Online]. Available: http://arxiv.org/abs/1605.07146
[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980
[25] Y. Tang, "Deep learning using linear support vector machines," 2013, *arXiv:1306.0239*. [Online]. Available: http://arxiv.org/abs/1306.0239
[26] K. Janocha and W. M. Czarnecki, "On loss functions for deep neural networks in classification," 2017, *arXiv:1702.05659*. [Online]. Available: http://arxiv.org/abs/1702.05659
[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[28] D. S. Kermany *et al.*, "Identifying medical diagnoses and treatable diseases by image-based deep learning," *Cell*, vol. 172, no. 5, pp. 1122–1131, 2018.
[29] T. E. Oliphant, *A guide to NumPy*, vol. 1. New York, NY, USA: Trelgol, 2006.
[30] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," 2016, *arXiv:1611.03530*. [Online]. Available: http://arxiv.org/abs/1611.03530

LI LI (Student Member, IEEE) received the B.Sc. degree in control engineering from Jilin University, Changchun, China, in 2008, and the M.Sc. degree in electrical engineering from The George Washington University, Washington, DC, USA, in 2014, where he is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering. His research interests include image processing, artificial neural networks, and machine learning.

MILOŠ DOROSLOVAČKI (Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from the University of Belgrade, in 1979 and 1984, respectively, and the Ph.D. degree in electrical engineering from the University of Cincinnati, in 1994.

Since 1995, he has been with the Department of Electrical and Computer Engineering, The George Washington University, where he is currently an Associate Professor. His research interests are in the fields of adaptive signal processing, communication signals and systems, and discrete-time signal and system theory.

MURRAY H. LOEW (Life Fellow, IEEE) received the B.S. degree in electrical engineering from the Drexel Institute of Technology, in 1965, and the M.S. and Ph.D. degrees from Purdue University, in 1967 and 1972, respectively.

He has been with the Department of Electrical and Computer Engineering, The George Washington University, and the Department of Biomedical Engineering, The George Washington University, where he is currently a Professor and the Chair of the Department of Biomedical Engineering. His research interests include medical imaging, multi- and hyper-spectral analysis, machine learning, and infrared imaging for early cancer detection. He is a Fellow of SPIE and the American Institute for Medical and Biological Engineering. He was the inaugural recipient of the Fulbright Distinguished Chair in Advanced Science and Technology in Australia, from 2013 to 2014.