Bayesian Weight Decay on Bounded Approximation for Deep Convolutional Neural Networks

Jung-Guk Park and Sungho Jo, *Member, IEEE*

***Abstract*— This paper determines the weight decay parameter value of a deep convolutional neural network (CNN) that yields a good generalization. To obtain such a CNN in practice, numerical trials with different weight decay values are needed. However, the larger the CNN architecture is, the higher is the computational cost of the trials. To address this problem, this paper formulates an analytical solution for the decay parameter through a proposed objective function in conjunction with Bayesian probability distributions. For computational efficiency, a novel method to approximate this solution is suggested. This method uses a small amount of information in the Hessian matrix. Theoretically, the approximate solution is guaranteed by a provable bound and is obtained by a proposed algorithm, where its time complexity is linear in terms of both the depth and width of the CNN. The bound provides a consistent result for the proposed learning scheme. By reducing the computational cost of determining the decay value, the approximation allows for the fast investigation of a deep CNN (DCNN) which yields a small generalization error. Experimental results show that our assumption verified with different DCNNs is suitable for real-world image data sets. In addition, the proposed method significantly reduces the time cost of learning with setting the weight decay parameter while achieving good classification performances.**

***Index Terms*— Bayesian method, convolutional neural networks (CNNs), inverse Hessian matrix, regularization, weight decay.**

I. Introduction

BASED on deep learning techniques [1]–[3] that have gained considerable attention, convolutional neural networks (CNNs) get to include deep layers of a large number of trainable weights. Due to their powerful representation of deep layers, CNNs are an essential part of applications, such as handwritten digit recognition [4], object classification [5] and detection [6], and data analysis [7], including intelligent systems [8]–[10]. This is because CNNs possess the property of neural networks (NNs); the training capacity is simply extended by stacking layers or adding more hidden nodes, resulting in a fitting to any input–output paired data [11], [12]. This property provides the reason why recent research [13]–[16] has widely employed NNs. However, this inevitably reduces the generalization ability of NN (i.e., it yields a large generalization error [17]), which can result from either a large number of model parameters [18] or the size of the network weights [19]. In prior NN research, Kroogh and Hertz [20], Hassibi and Stork [21], and MacKay [22], [23] considered similar problems and suggested efficient solutions in a shallow feedforward network. However, their implementations become practically impossible for an NN with a large number of weights. For example, Hassibi and Stork [21] and MacKay [22] used the Hessian matrix that requires $O(W^2)$ computer memory, where W is the number of trainable NN weights. This leads to a significant problem: as the depth of NN increases, the Hessian matrix cannot fit in the memory. Mackay [23] determined the weight decay parameter with the evidence framework in a shallow feedforward network. However, this work cannot be directly applied to a large-scale NN because of the problem of computing the Hessian matrix derived from the Gaussian approximation in the evidence framework. Although the Hessian matrix can provide the networks with good information during learning, its computational problem is hard to solve and should be properly addressed. Therefore, this paper proposes an approximation to information of the giga-scale or tera-scale Hessian matrix without computing all elements in the matrix. This approximation can assist with training deep NNs in order to yield a good performance.

In NNs, a practical way to improve the generalization is to add a weight decay parameter to an objective function (i.e., the regularization of the network weights). Generally, a procedure of determining the decay parameter is based on numerical trials, which requires a high computational cost when the number of trials increases. Moreover, this approach is inefficient for deep NNs that require a longer time for learning. Hence, the cost of the numerical trials cannot be ignored. However, we cannot skip this procedure, since the network's performance on a test data set depends on a properly chosen value of the decay parameter. As an application example, suppose that a deep CNN (DCNN) has been designed and is expected to yield a good test error, where an architecture (number of layers or number of hidden nodes) of the DCNN should be selected. The general approach is a grid search method to determine both the weight decay parameter and CNN architectural parameters. The depth (number of layers) or width (number of hidden nodes) of the CNN can be an architectural parameter of the

Manuscript received February 16, 2018; revised July 25, 2018 and October 29, 2018; accepted December 2, 2018. Date of publication January 16, 2019; date of current version August 21, 2019. This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2017-0-00432). *(Corresponding author: Sungho Jo.)*

The authors are with the School of Computing, Korea Advanced Institute of Science and Technology, Daejeon 34141, South Korea (e-mail: jgparknn@kaist.ac.kr; shjo@kaist.ac.kr).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2018.2886995

network. Let a set of the architectural parameters be $P = \{p_i : p_i \text{ is CNN architectural parameter(s) of interest}\}$. For example, $p_1 = 100$ and $p_2 = 200$ represent the depths 100 and 200 of a CNN; $p_1 = (100, 10)$ and $p_2 = (100, 20)$ denote the widths 10 and 20 with the depth 100, respectively. The time complexity of the grid search procedure for determining values of the decay parameter and $p_i \in P$ can be considered as $O(\sum_{i=1}^{M} CW_i)$, where W_i is the number of weights of the CNN with respect to p_i. C and M represent the number of the decay parameter values and the number of settings of the architectural parameters, $M = |P|$, respectively. This procedure selects the CNN that produces the minimum test error over M CNNs with different settings of architectural parameters. Examples of an empirical study that requires this procedure include recent articles [24], [25], which showed that CNNs with many layers commonly achieve a small test error.

However, the problem of the above-mentioned procedure is the time cost of training such a DCNN with determining a good value of the weight decay parameter, because the cost of feedforward and backpropagation computation is proportional to the number of CNN weights. In addition, the time to select the best one among M CNNs depends on two factors: M and the time cost of determining the weight decay parameter. Because the two combined factors result in a large computational cost, efficiently computing a good weight decay value of the deep NN becomes more important. Thus, we aim to reduce the computational cost of determining the decay parameter so that the cost of the CNN selection problem is $O(\sum_{i=1}^{M} W_i)$. This complexity is valid for training M different CNNs (e.g., $M = 3$ for LeNet [4], AlexNet [5] and residual network (ResNet) [24]) to obtain the best test error among those of the networks with a data set of interest.

As an alternative approach to addressing the generalization problem in deep learning, Srivastava *et al.* [26] and Wan *et al.* [27] reported that learning with randomly removing nodes or connections successfully prevents overfitting and reduces generalization errors in deep NNs. However, these methods should determine their own parameter (i.e., drop probability) in conjunction with selecting an NN layer on which they operate, requiring a high computational cost. Other recent works addressed new regularization methods for the generalization ability of deep NNs [28], [29]. However, the methods mentioned above employed the weight decay whose value should be determined.

Therefore, having considered the importance of the decay parameter, we establish an efficient determining process of the decay parameter that is based on theoretical foundations [22], [30]. Since estimating the parameter is known to be a challenging problem [31], related works are topics of interest in the NN community [32], [33]. To the best of our knowledge, an analytical and practical solution of the decay parameter of DCNN has not yet been introduced. This paper suggests the solution, handling both a nonconvex objective function and a large amount of NN weights which make it difficult to obtain the solution for the decay parameter, especially for deep networks. To overcome this nonconvexity problem, we propose a method that iteratively operates to determine the decay parameter via an analytic Bayesian solution. To address the problem caused by a large amount of NN weights, a novel approximation is presented.

The proposed approach is motivated by earlier works of the structural risk minimization (SRM) principle [18], [34] and is related to the evidence framework [23] in which we propose the parameterized cross-entropy objective function. Extreme learning machines [16], [35] are related to our work because they reduce the computational complexity of learning with a selection procedure. Bayesian NNs include our model. However, in contrast to traditional Bayesian NNs that are related to Gaussian processes [36]–[38], we derive a Bayesian probability function to determine the weight decay value.

The contributions of this paper are as follows. First, we establish a relationship between the decay parameter and a novel Bayesian probability distribution to derive an objective function called the intrinsic integer model (IIM) so that the relationship results in a trained CNN which yields a small generalization error. This automatically determines a proper value of the weight decay parameter, which differs from other recent deep learning techniques that require the procedure of decay parameter setting [26]–[29]. Second, we introduce an approximation for the IIM's computational efficiency called the method of the trace approximation (META) of the inverse Hessian matrix with a provable bound, which guarantees that the time complexity is linear in both the depth and width of the CNN so that it overcomes the problem of the computing the Hessian matrix [21]–[23]. In addition, an analysis of the SRM principle can be conducted when a CNN trained by our method has finite Vapnik–Chervonenkis (VC) dimension [39].

Because the computational cost of numerical trials for determining the decay parameter is expensive for DCNNs, the analytical solution by the proposed method significantly reduces this cost and facilitates an extensive investigation of DCNNs. This is a notable advantage of our work. In this paper, the proposed method is applied to three different CNN models. Their results are obtained using public benchmark image data sets and show that our Bayesian assumption is suitable for real-world image data. The remainder of this paper is organized as follows. Section II briefly introduces the motivation of our work. Section III presents in detail the proposed scheme and algorithm. Section IV shows the experimental results. Finally, Section V concludes this paper.

II. Structural Risk Minimization

A. Theoretical Motivation

Statistical learning theory [18] addresses the problem of function approximation with a class of functions $F = \{f(\mathbf{x}, \mathbf{w}), \mathbf{w} \in \mathbb{R}^W\}$ that are parameterized by a weight vector $\mathbf{w}$ of length W. The vector $\mathbf{w}$ is obtained by a training set of samples $D = \{(\mathbf{x}^{(i)}, t^{(i)})\}_{i=1}^{n}$, where $\mathbf{x}^{(i)}$ and $t^{(i)}$ denote the ith input data and its label, respectively. Assuming samples of the training set are drawn from an unknown joint probability distribution $P(\mathbf{x}, t)$, statistical learning seeks the best function in F which fits well to the training set. The quantity of the fitting is measured with a loss $\mathbb{L}(t, f(\mathbf{x}, \mathbf{w}))$ and is defined by

$$R(\mathbf{w}) = \int \mathbb{L}(t, f(\mathbf{x}, \mathbf{w}))dP(\mathbf{x}, t) \tag{1}$$

which is called the generalization error. Because the joint distribution is unknown, the training set is used to minimize (1) with empirical risk $R_e = (1/n)\sum_{i=1}^{n} \mathbb{L}(t^{(i)}, f(\mathbf{x}^{(i)}, \mathbf{w}))$. However, one of the issues with the empirical risk minimization is that it does not always guarantee the minimum of (1). SRM [40, p. 94] is an alternative approach to minimizing (1) by evaluating both an empirical risk and penalty function, which ensures an upper bound of the risk. SRM introduces a nested structure of subsets S_p of F, and we consider $S_p = \{f(\mathbf{x}, \mathbf{w}), \|\mathbf{w}\|_2^2 \leq C_p\}$. f is an NN and C_p is the structural penalty of subset S_p, $(1 \leq p < \infty)$, which results in $S_1 \subset S_2 \subset \ldots S_p \ldots$ of the nested set. In this paper, we briefly introduce a formula for SRM, which is the basis for our learning method. Using the Lagrange multiplier for the penalty function $\|\mathbf{w}\|_2^2$, SRM constrains the fitting capacity of a function f in F

$$R_s(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{L}(t^{(i)}, f(\mathbf{x}^{(i)}, \mathbf{w})) + \lambda_p \|\mathbf{w}\|_2^2 \tag{2}$$

where $\|\cdot\|_2$ denotes the L_2-norm and λ_p is the weight decay parameter of S_p. The goal is to select the best subset $S^* \subset F$ which contains a function f that produces the minimal penalized risk $R_s^* = \inf_{f \in S^*} R_s(\mathbf{w})$. Since the SRM principle applied to a learning model requires every hypothesis of the nested sets to have a finite VC dimension [39, p. 421], this paper uses the proof of [41, Th. 3] to support that a CNN with a piecewise linear activation function has a finite VC dimension. Consequently, our work can be analyzed using the SRM principle. To select an appropriate CNN that yields an approximation of R_s^*, an implementation and algorithmic way for both determining λ_p and minimizing (2) are presented in the following and described in Section III, respectively.

B. Implementation

For our implementation, a class of CNNs with a fixed architecture (i.e., a fixed number of layers and hidden nodes) is represented by a set of functions, $F = \{f(\mathbf{x}, \mathbf{w}),\ \mathbf{w} \in \mathbb{R}^W\}$, where f is a CNN, $\mathbf{x}$ is the input data, and $\mathbf{w}$ is a CNN weight vector of length W. The CNN consists of cascade functions, and its output is obtained by the feedforward step as

$$f(\mathbf{x}, \mathbf{w}) = \left(f_{\mathbf{m}}^{L} \circ f_{\mathbf{s}}^{L-1} \circ \ldots \circ f_{\mathbf{s}}^{(1)}\right)(\mathbf{x}) \tag{3}$$

where $f_{\mathbf{m}}^{L}$ is the final layer and $f_{\mathbf{s}}^{(l)} = f_{\mathbf{p}}^{(l)} \circ f_{\mathbf{a}}^{(l)} \circ f_{\mathbf{v}}^{(l)}$, where $f_{\mathbf{v}}^{(l)}$ is the lth convolution layer, and $f_{\mathbf{a}}^{(l)}$ denotes an activation function of the network with finite VC dimension (e.g., the rectified linear unit activation function ReLU, $a(x) = \max\{0, x\}$). $f_{\mathbf{p}}^{(l)}$ is the pooling layer and the output activation in the final layer $f_{\mathbf{m}}^{L}$ is a c-dimensional vector. Thus, $f(\mathbf{x}, \mathbf{w}) = (f_1(\mathbf{x}, \mathbf{w}), f_2(\mathbf{x}, \mathbf{w}), \ldots, f_c(\mathbf{x}, \mathbf{w}))$ and each element is obtained through the softmax function as follows:

$$f_k(\mathbf{x}, \mathbf{w}) = \frac{\exp\left(\text{net}_k^{(L)}\right)}{\sum_{z=1}^{c} \exp\left(\text{net}_z^{(L)}\right)} \tag{4}$$

for $1 \leq k \leq c$, where $\text{net}_k^{(L)} = \mathbf{w}_k^{(L)\mathbb{T}} \mathbf{a}^{(L-1)}$. Let $\mathbf{w}_k^{(L)}$ and $\mathbf{a}^{(L-1)}$ denote the column vector of CNN weights that are connected to the kth output node in the Lth layer (final layer) and the column vector of the hidden activation in the $L-1$th layer, respectively. The lth convolutional layer $f_{\mathbf{v}}^{(l)}$ has the weights $\mathbf{w}^{(l)} = (\mathbf{w}_1^{(l)\mathbb{T}}, \mathbf{w}_2^{(l)\mathbb{T}}, \ldots, \mathbf{w}_{k^{(l)}}^{(l)\mathbb{T}})^{\mathbb{T}}$ of $k^{(l)}$ CNN filters of a column vector. Finally, the total CNN weights are represented by a column vector $\mathbf{w} = (\mathbf{w}^{(1)\mathbb{T}}, \mathbf{w}^{(2)\mathbb{T}}, \ldots, \mathbf{w}^{(L)\mathbb{T}})^{\mathbb{T}}$.

A set of samples for training the CNN is denoted by $D = \{(\mathbf{x}^{(i)}, \mathbf{t}^{(i)})\}_{i=1}^{n}$, where $\mathbf{x}^{(i)}$ and $\mathbf{t}^{(i)}$ are an input image data in the ith sample and its target label, respectively. In this paper, $\mathbf{t}^{(i)} = (t_1^{(i)}, t_2^{(i)}, \ldots, t_c^{(i)})$ is represented by one-hot encoding for a class label that an input image $\mathbf{x}^{(i)}$ belongs to.

III. Method

This section follows (2) and begins with a Bayesian framework. The CNN (3) is generally trained by minimizing the objective function

$$R_c(\mathbf{w}) = -\sum_{i=1}^{n}\sum_{k=1}^{c} \mathbb{1}\left(t_k^{(i)} = 1\right) \log f_k(\mathbf{x}^{(i)}, \mathbf{w}) + \lambda \|\mathbf{w}\|_2^2 \tag{5}$$

where $\mathbb{1}(\cdot)$ and f_k denote the indicator function and the kth CNN output in (4), respectively. λ is the parameter that controls the fitting ability of the CNN and constrains the size of CNN weights, where we should choose its appropriate value to minimize (5).

A. Proposed Objective Function

For training the CNN (3) with determining the parameter λ value in (5) simultaneously, we propose a multiclass classification method with two parameters, η and ζ, that are associated with probability distributions. They are related to the evidence framework which is successfully applied to NNs [23] and support vector machines [42], [43]. We model the two parameters in order that one gains weighted learning information from an objective function and the other obtains information about the CNN weights. We assume that CNN weights follow the probability function parameterized by $\eta \in \mathbf{R}_+$ as follows:

$$p(\mathbf{w}|\eta) = \frac{\exp\left(-\eta \sum_j w_j^2\right)}{I_{\mathbf{w}}(\eta)} \tag{6}$$

where $I_{\mathbf{w}}(\eta) = \int \exp(-\eta \sum_j w_j^2) d\mathbf{w}$ and w_j is an element of $\mathbf{w}$. The probability function for the first term on the right-hand side in (5) is the likelihood that is detailed in the following. For the multiclass classification problem, a generalized Bernoulli distribution [44] (i.e., categorical distribution) is commonly assumed in probabilistic machine learning. Considering the intrinsic relationship of the decay parameter, we propose the likelihood of generalized Bernoulli distribution by the CNN output that is parameterized by $\zeta \in \mathbf{Z}_+$ as follows:

$$p(t_k = 1|\mathbf{x}, \mathbf{w}, \zeta) = \prod_{k=1}^{c} u_k(\mathbf{x}, \mathbf{w}, \zeta)^{\mathbb{1}(t_k=1)} \tag{7}$$

where $t_k \in \{0, 1\}$, $(1 \leq k \leq c)$, and $u_k(\mathbf{x}, \mathbf{w}, \zeta) = f_k(\mathbf{x}, \mathbf{w})^{\zeta}$. Thus, the likelihood term as a function of the training set $D = \{(\mathbf{x}^{(i)}, \mathbf{t}^{(i)})\}_{i=1}^{n}$ is evaluated as follows:

$$p(D|\mathbf{w}, \zeta) = \prod_{i=1}^{n} \left\{ \prod_{k=1}^{c} u_k(\mathbf{x}^{(i)}, \mathbf{w}, \zeta)^{\mathbb{1}(t_k^{(i)}=1)} \right\}. \tag{8}$$

For training the CNN, the posterior distribution of $\mathbf{w}$ given both the set D and parameters η, ζ follows the Bayesian formula as:

$$p(\mathbf{w}|D,\zeta,\eta) = \frac{p(D|\mathbf{w},\zeta)p(\mathbf{w}|\eta)}{\int p(D|\mathbf{w},\zeta)p(\mathbf{w}|\eta)d\mathbf{w}}. \tag{9}$$

In (9), the joint probability function $p(D|\mathbf{w},\zeta)p(\mathbf{w}|\eta)$ is evaluated as

$$\left(\frac{\eta}{\pi}\right)^{\frac{W}{2}} \exp\left(\sum_{i=1}^{n}\left(\zeta\sum_{k=1}^{c} Q_{i,k}(\mathbf{w})\right) - \eta\|\mathbf{w}\|_2^2\right) \tag{10}$$

where $Q_{i,k}(\mathbf{w}) = \mathbb{1}(t_k^{(i)} = 1)\log f_k(\mathbf{x}^{(i)}, \mathbf{w})$, W denotes the number of CNN weights, and the negative logarithm of (10) is

$$-\sum_{i=1}^{n}\left(\zeta\sum_{k=1}^{c} Q_{i,k}(\mathbf{w})\right) + \eta\|\mathbf{w}\|_2^2 - \frac{W}{2}\log\left(\frac{\eta}{\pi}\right). \tag{11}$$

Note that minimizing (11) with respect to $\mathbf{w}$ is equivalent to minimizing (5). It is assumed that the posterior on $\mathbf{w}$ is very sharp at its maximum $\eta_{\mathbf{p}}$ and $\zeta_{\mathbf{p}}$, as in [23]. The probability function is then marginalized as

$$\begin{aligned} p(\mathbf{w}|D) &= \int\sum_{\zeta} p(\mathbf{w},\eta,\zeta|D)d\eta \\ &\approx p(\mathbf{w}|D,\eta_{\mathbf{p}},\zeta_{\mathbf{p}})\int\sum_{\zeta} p(\eta,\zeta|D)d\eta \end{aligned} \tag{12}$$

where the probability function $p(\mathbf{w}|D)$ is almost governed by $\eta_{\mathbf{p}}$ and $\zeta_{\mathbf{p}}$. Using the Bayes rule, the posterior distribution of parameters is derived as

$$p(\eta,\zeta|D) = \frac{p(D|\eta,\zeta)p(\eta,\zeta)}{\int\sum_{\zeta} p(D|\eta,\zeta)p(\eta,\zeta)d\eta} \tag{13}$$

where maximizing the posterior $p(\eta,\zeta|D)$ depends on $p(D|\eta,\zeta)$ which is marginalized by

$$p(D|\eta,\zeta) = \int p(D|\mathbf{w},\zeta)p(\mathbf{w}|\eta)d\mathbf{w}. \tag{14}$$

The probability function $p(D|\eta,\zeta)$ is evaluated using the Gaussian approximation with Taylor expansion around a local point $\mathbf{w}^*$, which derives $p(D|\eta,\zeta)$ as follows:

$$\frac{2^{W/2}\exp\left(\zeta\sum_{i=1}^{n}\sum_{k=1}^{c} Q_{i,k}(\mathbf{w}^*) - \eta\|\mathbf{w}^*\|_2^2\right)}{\eta^{-W/2}\det(\mathbf{H})^{\frac{1}{2}}} \tag{15}$$

where $\mathbf{H}$ is the Hessian matrix of (11) and det(.) denotes the matrix determinant. This approximation becomes exact as the gradient of (11) is small. The detailed derivation of (15) is shown in the Appendix.

A value of the parameter η with respect to the extremum of (15) is obtained from

$$\frac{\partial\log p(D|\eta,\zeta)}{\partial\eta} = 0 \tag{16}$$

which provides the analytical solution for η as follows:

$$\eta = \frac{\gamma}{2\|\mathbf{w}^*\|_2^2} \tag{17}$$

with $\gamma = \sum_{j=1}^{W}(\lambda_j(\mathbf{H}) - 2\eta)/\lambda_j(\mathbf{H})$ where $\lambda_j(\mathbf{H})$ and W denote the eigenvalue of the Hessian matrix of (11) and the number of CNN weights, respectively. That is, $\gamma = W - 2\eta\,\mathrm{tr}(\mathbf{H}^{-1})$ where tr(.) is the matrix trace. For an analytical solution of ζ, the density function parameterized by $\zeta_v \in \mathbf{R}_+$ which takes a value of ζ is introduced, resulting in $p(D|\eta,\zeta_v) = p(D|\eta,\zeta)$. A value of the parameter ζ_v is obtained by

$$\frac{\partial\log p(D|\eta,\zeta_v)}{\partial\zeta_v} = \zeta_0 \tag{18}$$

where ζ_0 has a value of $-(n/2\zeta)$, and a value of the parameter ζ is chosen by

$$\zeta = \left\lceil\frac{\gamma_v - n}{2\sum_{i=1}^{n}\sum_{k=1}^{c} Q_{i,k}(\mathbf{w}^*)}\right\rceil \tag{19}$$

where $\lceil\cdot\rceil$ is the ceiling operator, $\gamma_v = \sum_{j=1}^{W}(\lambda_j(\mathbf{H}_v) - 2\eta)/\lambda_j(\mathbf{H}_v)$ and $\gamma_v = W - 2\eta\,\mathrm{tr}(\mathbf{H}_v^{-1})$, and $\lambda_j(\mathbf{H}_v)$ denotes the eigenvalue of the Hessian matrix of

$$-\sum_{i=1}^{n}\left(\zeta_v\sum_{k=1}^{c} Q_{i,k}(\mathbf{w})\right) + \eta\|\mathbf{w}\|_2^2. \tag{20}$$

Note that a value of γ_v is equal to that of γ. Each derivation of (17) and (19) is straightforward using the partial derivative of the log determinant presented in [45].

With the two solutions of η and ζ, the proposed objective function IIM, the negative log-likelihood of (10), is formulated as

$$R_c(\mathbf{w}) = -\sum_{i=1}^{n}\sum_{k=1}^{c}\zeta\mathbb{1}\left(t_k^{(i)} = 1\right)\log f_k(\mathbf{x}^{(i)},\mathbf{w}) + \eta\|\mathbf{w}\|_2^2 \tag{21}$$

where $(W/2)\log(\eta/\pi)$ in (10) is omitted, because it is irrelevant to $\mathbf{w}$. Because the CNN with a deep layer has a larger number of trainable weights, computing the trace of the Hessian matrix in (17) and (19) is practically infeasible. Thus, we devise a method to address the problem presented in the following.

B. Approximation to the Trace of the Inverse Hessian Matrix

Computing the trace of the inverse Hessian matrix in (17) and (19) requires a cost in terms of memory that is proportional to the number of CNN weights squared. For instance, if the CNN has 100 000 weights ($W = 100\,000$), then the implementation requires large amounts of memory $O(10^{10})$. Thus, this section introduces a method to approximate the trace of the inverse Hessian matrix. Suppose that a matrix $\hat{\mathbf{H}}$ is an approximation[1] of the exact Hessian in (17) and (19) defined by

$$\hat{\mathbf{H}} = \frac{1}{2}(\tilde{\mathbf{H}} + \tilde{\mathbf{H}}^{\mathbb{T}}) + \tilde{\delta} + 2\lambda\mathbf{I} \tag{22}$$

with $\lambda = \eta/\zeta$. $\tilde{\mathbf{H}}$ is the W-dimensional Hessian matrix with element $\tilde{\mathbf{H}}(m,n) = -\sum_{i=1}^{n}(\nabla_m Q_i(\mathbf{w}^* + \epsilon_n) - \nabla_m Q_i(\mathbf{w}^*))/\epsilon$, ∇_m denotes the mth element of the gradient with respect to

[1]This form of the matrix can be applied to any objective function that is twice differentiable.

$\mathbf{w}$ and $Q_i(\mathbf{w}^* + \epsilon_n) = \sum_{k=1}^{c} \mathbb{1}(t_k^{(i)} = 1) \log f(\mathbf{x}^{(i)}, \mathbf{w}^* + \epsilon_n)$ where f is the CNN output obtained from (4) and $\mathbf{w}^*$ is the CNN weights. ϵ_n is a W-dimensional point that consists of all zeros except for the nth element such as $[0, \ldots 0, \epsilon, 0, \ldots 0]^{\mathbb{T}}$, and the value of ϵ is set to the constant 10^{-5}. m and n are the indices of CNN weights and $\mathbf{I}$ is the identity matrix. The symmetric matrix $(\tilde{\mathbf{H}}+\tilde{\mathbf{H}}^{\mathbb{T}})/2$ has a small approximation error, because the error between the exact Hessian matrix and its finite-difference approximation is $O(\epsilon)$ [46]. Since the Hessian $(\tilde{\mathbf{H}} + \tilde{\mathbf{H}}^{\mathbb{T}})/2$ may not be positive definite, a diagonal matrix $\tilde{\delta}$ is employed to make the matrix positive definite.

Defining an s-weight set $\mathbf{s}_w$, we suggest the trace approximation META. This method is fast and feasible while having a provable bound of the trace of (22). We first construct the block-diagonal matrix principle for the trace approximation.

Proposition 1: Let the parameter $\lambda \in (0, 1]$ and $\hat{\mathbf{H}}^{\mathbf{d}}$ be any block-diagonal matrix composed of $L \times L$ blocks of $\hat{\mathbf{H}}$ defined in (22). Let $\tilde{\delta}$ be any diagonal matrix which makes $[\hat{\mathbf{H}}^{\mathbf{d}} - 2\lambda\mathbf{I}]$ positive definite. Then, $\mathrm{tr}(\hat{\mathbf{H}}^{-1}) > \mathrm{tr}([\hat{\mathbf{H}}^{\mathbf{d}}]^{-1})$.

Proof: Let $\hat{\mathbf{H}}_{\mathbf{N}} = (1/2)(\tilde{\mathbf{H}} + \tilde{\mathbf{H}}^{\mathbb{T}}) + \tilde{\delta}$ composed of $L \times L$ blocks and $\hat{\mathbf{H}}_{\mathbf{N}}$ be partitioned into four $(L-1) \times (L-1)$, $(L-1) \times 1$, $1 \times (L-1)$, and 1×1 blocks. Let $\hat{\mathbf{H}}_{\mathbf{N}}^{\mathbf{d}}$ be the block-diagonal matrix of $\hat{\mathbf{H}}_{\mathbf{N}}$ corresponding to the partition and $\hat{\mathbf{H}}_{\mathbf{N}}^{\mathbf{off}}$ be the matrix of off-diagonal blocks of $\hat{\mathbf{H}}_{\mathbf{N}}$ such that $\hat{\mathbf{H}}_{\mathbf{N}} = \hat{\mathbf{H}}_{\mathbf{N}}^{\mathbf{d}} + \hat{\mathbf{H}}_{\mathbf{N}}^{\mathbf{off}}$. With the inverse of Schur's complement, $\mathrm{tr}(\hat{\mathbf{H}}_{\mathbf{N}}^{-1}) > \mathrm{tr}([\hat{\mathbf{H}}_{\mathbf{N}}^{\mathbf{d}}]^{-1})$ holds. More generally, this inequality is valid when the partitioning applied to the largest block among four blocks is repeated $L-1$ times so that $\hat{\mathbf{H}}_{\mathbf{N}}^{\mathbf{d}}$ is the block-diagonal matrix composed of $L \times L$ blocks. Since $\hat{\mathbf{H}}_{\mathbf{N}}$, $\hat{\mathbf{H}}_{\mathbf{N}}^{\mathbf{d}}$, and $2\lambda\mathbf{I}$ are symmertic positive definite, $\mathrm{tr}([\hat{\mathbf{H}}_{\mathbf{N}} + 2\lambda\mathbf{I}]^{-1}) > \mathrm{tr}([\hat{\mathbf{H}}_{\mathbf{N}}^{\mathbf{d}} + 2\lambda\mathbf{I}]^{-1})$. By defining $\hat{\mathbf{H}} = \hat{\mathbf{H}}_{\mathbf{N}} + 2\lambda\mathbf{I}$ and $\hat{\mathbf{H}}^{\mathbf{d}} = \hat{\mathbf{H}}_{\mathbf{N}}^{\mathbf{d}} + 2\lambda\mathbf{I}$, $\mathrm{tr}(\hat{\mathbf{H}}^{-1}) > \mathrm{tr}([\hat{\mathbf{H}}^{\mathbf{d}}]^{-1})$. □

To apply Proposition 1 to the DCNN, we define s-weight set $\mathbf{s}_w$ as follows.

Definition 2: An s-weight set $\mathbf{s}_w$ is a subset of indices which are assigned to CNN weights, where each element of $\mathbf{s}_w$ is the row or column index of a block-diagonal matrix of the Hessian matrix $\hat{\mathbf{H}}$ in (22).

Proposition 3: Let $\mathbf{s}_w^{(l)}$ be an s-weight set that contains its elements assigned to weights in the lth layer of CNN and $\mathbf{s}_w = \{\mathbf{s}_w^{(l)} : l \in \{1, 2, \ldots, L\}\}$. Let $\hat{\mathbf{H}}_{\mathbf{N}}$ be defined in Proposition 1 and $\hat{\mathbf{H}}_{\mathbf{s}}^{\mathbf{d}}$ be the block-diagonal matrix of $\hat{\mathbf{H}}_{\mathbf{N}}$, where $\hat{\mathbf{H}}_{\mathbf{s}}^{\mathbf{d}}$ is evaluated with both $\mathbf{s}_w$ and some $\tilde{\delta}$ that makes $\hat{\mathbf{H}}_{\mathbf{s}}^{\mathbf{d}}$ positive definite. Let $\hat{\mathbf{H}}_{\mathbf{s}} = \hat{\mathbf{H}}_{\mathbf{s}}^{\mathbf{d}} + 2\lambda\mathbf{I}$, $\lambda \in (0, 1]$, and $\hat{\mathbf{H}}$ be the Hessian matrix in (22) obtained with the same values of $\tilde{\delta}$ and λ in $\hat{\mathbf{H}}_{\mathbf{s}}$. Then, $\mathrm{tr}(\hat{\mathbf{H}}^{-1}) > \mathrm{tr}([\hat{\mathbf{H}}_{\mathbf{s}}]^{-1})$.

The proof of Proposition 3 immediately follows Proposition 1 with the definition of s-weight set which contributes to the block-diagonal matrix of (22). Proposition 1 shows that the approximate trace is always smaller than its true value, provided that (22) is a reliable approximation of the exact Hessian. Proposition 3 shows that the trace bound holds with the s-weight set $\mathbf{s}_w$ which reflects a layerwise structure of the block-diagonal matrix. These guarantee a consistent learning result, despite using a small number of partial elements in the Hessian matrix. In addition, the META generalizes to the width of CNN, which is shown in Fig. 1 (bottom).

$$\hat{\mathbf{H}}_{\mathbf{N}}^{\mathbf{d}} = \begin{array}{c} \text{LAYER} \quad 1 \quad\quad 2 \quad\quad \ldots \quad\quad L \\ \begin{pmatrix} H_{11} & 0 & \ldots & 0 \\ 0 & H_{22} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & H_{LL} \end{pmatrix} \end{array}$$

$$\hat{\mathbf{H}}_{\mathbf{N}}^{\mathbf{off}} = \begin{pmatrix} 0 & H_{12} & \ldots & H_{1L} \\ H_{21} & 0 & \ldots & H_{2L} \\ \vdots & \vdots & \ddots & \vdots \\ H_{L1} & H_{L2} & \ldots & 0 \end{pmatrix}$$

$$H_{ll} = \begin{array}{c} \text{FILTER} \quad 1 \quad\quad\quad 2 \quad\quad\quad \ldots \quad\quad\quad k^{(l)} \\ \begin{pmatrix} \mathbf{h}_{1,1}^{(l)} & \mathbf{h}_{1,2}^{(l)} & \ldots & \mathbf{h}_{1,k^{(l)}}^{(l)} \\ \mathbf{h}_{2,1}^{(l)} & \mathbf{h}_{2,2}^{(l)} & \ldots & \mathbf{h}_{2,k^{(l)}}^{(l)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{h}_{k^{(l)},1}^{(l)} & \mathbf{h}_{k^{(l)},2}^{(l)} & \ldots & \mathbf{h}_{k^{(l)},k^{(l)}}^{(l)} \end{pmatrix} \end{array}$$

Fig. 1. Decomposed Hessian matrix. Block-diagonal matrix $\hat{\mathbf{H}}_{\mathbf{N}}^{\mathbf{d}}$ and off-diagonal matrix $\hat{\mathbf{H}}_{\mathbf{N}}^{\mathbf{off}}$ in Proposition 1 (top and middle). $k^{(l)}$ and $\mathbf{h}_{u,v}^{(l)}$ denote the number of filters and a block of the Hessian matrix with respect to the uth and vth CNN filters in the lth layer ($1 \le l \le L$), respectively (bottom). The META generalizes to the width of the CNN by decomposing H_{ll} in a filterwise indexing manner.

C. Learning Algorithm

Once the parameter λ is obtained by the IIM&META, the proposed objective function (21) can be minimized by a backpropagation process or its variants. As general learning methods iteratively optimize a CNN objective function, the proposed method is capable of iteratively minimizing (21) with simultaneously obtaining solutions of the parameters ζ and η. The implementation of our learning method is straightforward via a family of stochastic gradient descent (SGD) algorithms. Based on the bound (inequality) on the trace, we design the practical learning algorithm IIM&META with minibatch SGD, which is detailed in Algorithm 1. The approximation of the trace and determining process of λ include lines 16 and 19, respectively. On account of Propositions 1 and 3, the compensation is employed for the approximate trace with the minibatch SGD method, which is presented in line 18 and lines 21–24. The momentum or other methods with the SGD can be applied to Algorithm 1. The time complexity of line 16 is linear in terms of the depth and width. The complexity of other lines is equivalent to that of the standard SGD.

IV. Experimental Results

The experimental section shows classification results of benchmark image data sets evaluated by the most widely known LeNet [4], the popular DCNN [5] (a.k.a. AlexNet), and the cutting-edge ResNet [24]. These networks verify the extensibility of our method using public benchmark data sets, MNIST [47], NORB [48], and CIFAR-10/100 [49]. These CNNs are learned by four minibatch SGD-based methods: no weight decay (NWD), grid-λ (grid search for determining the

TABLE I

TIME COMPLEXITY OF EACH LEARNING METHOD FOR CNN WITH DETERMINING λ. C: THE NUMBER OF TRIALS WITH DIFFERENT SETTINGS OF λ. M: THE NUMBER OF SETTINGS OF ARCHITECTURAL PARAMETERS ($M = |P|$) WHERE $P = \{p_i : p_i$ IS CNN ARCHITECTURAL PARAMETER(S) OF INTEREST$\}$, E.G., $p_1 = 164$ AND $p_2 = 227$ DENOTE THE DEPTHS 164 AND 227, RESPECTIVELY. W_i: THE NUMBER OF CNN WEIGHTS WITH RESPECT TO p_i (WHEN $M = 1$, i IS OMITTED)

	NWD	GRID-λ	DROPOUT	IIM&META
A FIXED SETTING OF ARCHITECTURAL PARAMETERS (M=1)	$O(W)$	$O(CW)$	$O(CW)$	$O(W)$
VARIED SETTINGS OF ARCHITECTURAL PARAMETERS (M>1)	$O(\sum_{i=1}^{M} W_i)$	$O(C\sum_{i=1}^{M} W_i)$	$O(C\sum_{i=1}^{M} W_i)$	$O(\sum_{i=1}^{M} W_i)$

Algorithm 1: IIM&META With Minibatch SGD

Input : training set D, maximum training epoch T, minibatch size n_b, learning rate α, CNN $f(\mathbf{x}, \mathbf{w})$ with initial weights $\mathbf{w}^{(0)}$, compensation factor υ, small constant ϵ, s-weight set $\mathbf{s}_w$

Output: CNN weights $\mathbf{w}^*$

$\zeta = 1, \eta = 0, t = 0, \mathbf{w}^* = \mathbf{w}^{(0)}$;
randomly choose index d_0 among $1, 2, \ldots, \frac{|D|}{n_b}$;
while $t < T$ **do**
  $D_d = \{(\mathbf{x}^{(i)}, \mathbf{t}^{(i)})\}_{i \in I_d}$ with $I_d \subset \{1, 2, ..., |D|\}$;
  $|I_d| = n_b$, d is minibatch index, $1 \le d \le \frac{|D|}{n_b}$;
  $R_c(\mathbf{w}, D) = -\zeta \sum_{i=1}^{|D|} \sum_{k=1}^{c} Q_{i,k}(\mathbf{w}) + \eta \|\mathbf{w}\|_2^2$;
  proposed objective function (21);
  $R_c(\mathbf{w}, D) \propto -\sum_{i=1}^{|D|} \sum_{k=1}^{c} Q_{i,k}(\mathbf{w}) + \lambda \|\mathbf{w}\|_2^2$;
  equivalently minimizing (5);
  where $Q_{i,k}(\mathbf{w})$ is defined in (10) and $\lambda = \frac{\eta}{\zeta}$;
  for $1 \le d \le \frac{|D|}{n_b}$ **do**
    $\mathbf{g}^{(d)} = \frac{\partial R_c(\mathbf{w}, D_d)}{\partial \mathbf{w}}$;
    $\mathbf{w}^{(t)} = \mathbf{w}^{(t)} - \frac{\alpha}{n_b \zeta} \mathbf{g}^{(d)}$;
  end
  $\mathbf{w}^* = \mathbf{w}^{(t)}$;
  compute $\hat{\mathbf{H}}_{\mathbf{s}}$ in Proposition 3 with D_{d_0} and $\mathbf{s}_w$;
  compute γ in (17) and (19) with $\zeta \hat{\mathbf{H}}_{\mathbf{s}}$;
  $\gamma = \upsilon \gamma$;
  update η and ζ by (17) and (19) with D_{d_0} and n_b;
  $q = \sum_{i=1}^{n_b} \sum_{k=1}^{c} Q_{i,k}(\mathbf{w}^*)$ with D_{d_0};
  if $\frac{\gamma}{\gamma - n_b} < \epsilon \frac{\|\mathbf{w}^*\|_2^2}{q}$ **then**
    $\gamma = \min\{2\gamma, n_b - 1\}$;
    go to line 19;
  end
  $t = t + 1$;
end

weight decay parameter), dropout [26], and the IIM&META. NWD shows the importance of the weight decay parameter. Grid-λ determines a value of the decay parameter among its candidates using the time cost which is linear in terms of the number of the candidates. Dropout is a popular method for deep learning. Table I summarizes the time complexity of NWD, grid-λ, dropout, and the IIM&META. The time complexity considers that dropout operates on an arbitrary layer, where the value of the dropout parameter is set to 0.5 (as suggested in [26]) and a value of the weight decay on

Fig. 2. Examples of the benchmark data sets (top row: MNIST, middle: NORB, and bottom: CIFAR-10/100).

dropout is found using the grid search method. The setting of IIM&META is as follows. ϵ in Algorithm 1 is set to 10^{-4} that may depend on arithmetic precision, and a subset of the s-weight set $\mathbf{s}_w$ contains three CNN weight indices for each weight layer of interest. The indices are randomly chosen during CNN weight initialization. υ is set to the constant 0.05. The elements of $\tilde{\delta}$, in (22) and Proposition 3, with respect to indices in $\mathbf{s}_w$ are set by a small value $1/(10^7 W)$ multiplied by the factor 10 in ascending order at every epoch until $\hat{\mathbf{H}}_{\mathbf{s}}^{\mathbf{d}}$ is positive definite ($\hat{\mathbf{H}}_{\mathbf{s}}^{\mathbf{d}}$ is defined in Proposition 3). The remaining elements of $\tilde{\delta}$ with respect to $\mathbf{s}_w^c$, the complement of $\mathbf{s}_w$, are set in order that $\sum_{j \in \mathbf{s}_w^c} (\lambda_j(\hat{\mathbf{H}}) - 2\eta)/\lambda_j(\hat{\mathbf{H}})$ is equal to the constant $10^{-10}|\mathbf{s}_w^c|/\upsilon$. We employ the validation data set that selects an epoch corresponding to the lowest estimated generalization error of a CNN during learning (called holdout). Furthermore, grid-λ and dropout use the validation data set to determine a weight decay value which produces the lowest one of validation errors with different settings of λ. All experiments are conducted using Matconvnet [50] on Intel processor-based GPU workstations. LeNet and DCNN (AlexNet) are learned by minibatch SGD of 100 samples. For ResNet, the size of the minibatch is 128. Each learning method runs five times with random CNN weight initialization.

A. MNIST Data Set

MNIST, handwritten digit images for classification [47], is composed of gray-scale 28×28 pixel images of digits (0∼9). The data set is split into 54 000 training samples (5400 for each digit), 6000 validation samples (600 for each digit) for the holdout, and 10 000 test samples (1000 for each digit). For the four learning methods (NWD, grid-λ, dropout, and IIM&META), the learning rate and the momentum parameter are set to 0.01 and 0.9, respectively. The maximum learning epoch is 20. Grid-λ and dropout select a value of the weight decay from $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ by choosing the minimum validation error corresponding to one of the values. Dropout works on the final layer with a probability of 0.5. Since texture information in this data set is simple (shown in Fig. 2), LeNet [4] is thoroughly sufficient for use. In this experiment, the network denoted by MNIST_L consists of

TABLE II

TRAINING CLASSIFICATION ERROR (%) OF THE CNNS ON BENCHMARK IMAGE DATA SETS. A FIXED SETTING OF ARCHITECTURAL PARAMETERS (I.E., FIXED ARCHITECTURE)

	NWD	GRID-λ	DROPOUT	IIM&META
MNIST_L	1.697±0.03	0.0 ±0.003	0.182±0.037	0.004 ±0.004
NORB_L	5.327±0.558	0.414±0.168	0.692±0.225	0.234±0.049
NORB_D	0.010 ±0.016	0.0±0.001	0.004±0.008	0.0±0.0
CIFAR-10_D	0.038±0.021	0.0 ± 0.0	0.0 ± 0.0	0.002 ± 0.034
CIFAR-10_R	0.0±0.0	0.008±0.005	0.001±0.0	0.033±0.015
$\text{CIFAR-100}_\text{R}$	0.001±0.0	0.79±0.022	0.776±0.005	0.065± 0.038

TABLE III

TEST CLASSIFICATION ERROR (%) OF THE CNNS ON BENCHMARK IMAGE DATA SETS. A FIXED SETTING OF ARCHITECTURAL PARAMETERS (I.E., FIXED ARCHITECTURE)

	NWD	GRID-λ	DROPOUT	IIM&META
MNIST_L	1.454± 0.102	0.839 ± 0.041	0.781 ± 0.045	0.771±0.001
NORB_L	8.082 ±0.832	3.778 ± 0.445	4.076 ± 1.274	3.774 ±0.633
NORB_D	3.726±0.657	3.22±0.424	3.093±0.162	2.98±0.209
CIFAR-10_D	21.768±0.73	15.05 ± 2.61	14.128 ± 0.306	13.828 ± 0.438
CIFAR-10_R	7.232 ±0.085	5.536±0.076	5.334 ± 0.093	5.339 ± 0.142
$\text{CIFAR-100}_\text{R}$	27.676 ± 0.0537	23.59 ± 0.344	23.09±0.047	23.21±0.073

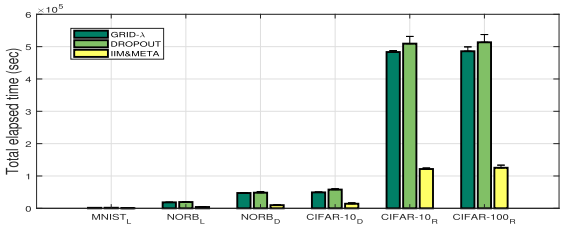

Fig. 3. Time of CNN learning with determining λ. The total elapsed time is proportional to the number of CNN weights. The difference in time between the IIM&META and other methods significantly increases as the size of the CNN grows.

the two layers of 16–256 filters and one fully connected layer of 256 hidden nodes. The average and standard deviation of errors trained by the four methods are compared in Tables II and III. In Table II, the CNNs converge to almost zero of training error except for NWD, explaining that the weight decay is related to a better training error. In Table III, it is also shown that the standard deviation of the test error by IIM&META is less than that of grid-λ. Fig. 3 illustrates the total elapsed time of each learning method with determining the decay parameter. The proposed method yields a good test error compared to other learning methods while demonstrating the low time cost. The weight decay parameter determined by the IIM&META is shown in Fig. 4(a).

B. NORB Data Set

This experiment uses the publicly available NORB data set [48] which consists of gray-scale images of 50 different miniatures of the five categories: airplanes, trucks, four-legged animals, cars, and human figures. The NORB data set is useful for the classification of object shape. These images were captured by a stereo camera with 162 different views from 30°-70° at every 5°. The 50 miniatures are split into a training set, validation set, and test set. We use the small NORB data set with images of a single object, where they are sized at the center of images with uniform backgrounds.

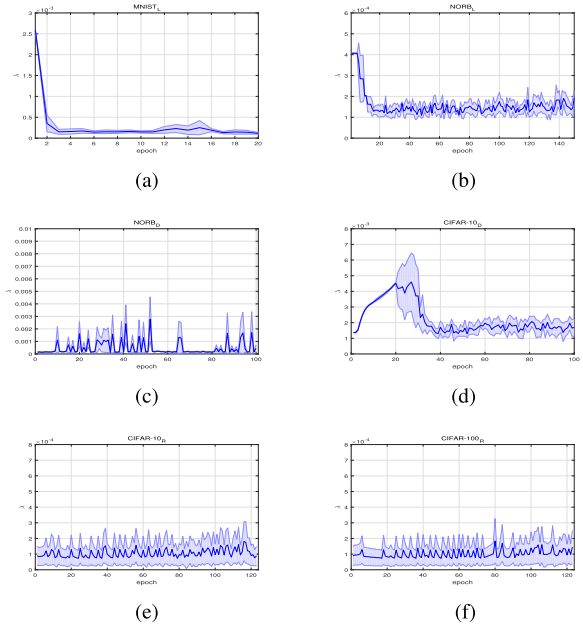

Fig. 4. Plot of λ determined by the IIM&META. Line and shaded area: average and standard deviation, respectively. That is adaptively fitted to each CNN trained with the different data sets. (a) LeNet (MNIST). (b) LeNet (NORB). (c) DCNN (NORB). (d) DCNN (CIFAR-10). (e) ResNet (CIFAR-10). (f) ResNet (CIFAR-100).

A total of 46 600 images in the data set are composed as follows. 26 600 stereo image pairs belong to the training set, 10 000 stereo images comprise the validation set, and the remaining 10 000 images are used for the test set. Grid-λ and dropout take a value of λ in the same manner as in the MNIST experiment. All data presented in the network are of size 96×96 pixels. Because each image has a clean background and simple foreground texture (see Fig. 2), LeNet [4] is used, where the network consists of the 256-16-256 layered convolutional filters and 64 hidden nodes in the fully connected layer. Dropout is applied to the middle CNN filter layer with a probability of 0.5. The learning rate and the momentum parameter are set to 0.001 and 0.9, respectively.

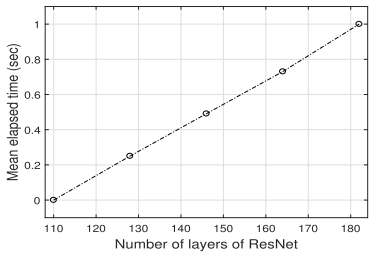

Fig. 5. Mean elapsed time of the IIM&META applied to ResNet. The time axis is normalized for intuitive illustration. The time increases almost linearly with the depth (number of layers).

This CNN, denoted by NORB_L, learns during 150 epochs. Next, the DCNN (a.k.a. AlexNet), denoted by NORB_D, is used where the settings including its architecture and the learning rate follow the implementation detailed in [50]. Fig. 3 shows the elapsed time of each learning method. The determined weight decay parameter by the IIM&META with respect to LeNet and AlexNet is shown in Fig. 4(b) and (c), respectively. The average and standard deviation of training and test errors of each learning method are summarized in Tables II and III. In those, NWD yields both the worst training and test errors, demonstrating that the weight decay affects both the test error and the training error. In contrast to grid-λ and dropout, the proposed method reduces the total elapsed time while preserving the proper decay setting and yielding a small test error shown in Table III.

C. CIFAR-10/100 Data Set

This data set is composed of 32×32-pixel RGB-colored image used in [5]. Cropped from nature scenes, each image belongs to one of 10 classes in CIFAR-10 and is classified as one of 100 labels in CIFAR-100. The data set consists of 50 000 training images and 10 000 test images. Because each image in this data set contains a colored object with a messy background, which leads the network requiring a deep layer to obtain a good classification accuracy for such a data set (see Fig. 2), two different large-scale networks are employed. The network architecture in the first setting is DCNN (AlexNet) [5] denoted by CIFAR-10_D.

The architecture and learning rate follow the implementation detailed in [5]. The momentum value is set to 0.9. The images that are input to AlexNet are preprocessed via the whitening method [51] after the pixels are subtracted with each mean and normalized by each standard deviation. Dropout operates on the final layer. In this experiment, 10 000 samples in a training set are used for both holdout purposes and the grid search to determine λ. The statistics of errors of each learning method are given in Tables II and III. On the second network setting, ResNet with 164 layers is employed for CIFAR-10 and CIFAR-100 denoted by CIFAR-10_R and $\text{CIFAR-100}_\text{R}$, respectively. The establishment of the learning rate, momentum, and experimental condition is similar to those of the original work [24]. In addition, the number of training and test data sets follows the original work. Hence, the validation data set is identical to the test data set. The dropout method operates in the final layer of ResNet.

TABLE IV
CLASSIFICATION ERROR (%) OF RESNET TRAINED BY IIM&META WITH DIFFERENT DEPTHS ON CIFAR-100. (VARIED SETTINGS OF ARCHITECTURAL PARAMETERS)

TRAINING ERROR	TEST ERROR	DEPTH
0.065± 0.038	23.21±0.073	164
0.083 ±0.186	23.08 ±0.132	182
0.0±0.0	22.59± 0.076	227

Fig. 5 shows the mean elapsed time of IIM&META applied to ResNet trained five times. The time scale is normalized for intuitive illustration. The elapsed time increases almost linearly as the depth increases. The classification accuracies of ResNet trained by the four learning methods are presented in Tables II and III. In Table II, all training errors of ResNet on CIFAR-10 and CIFAR-100 data sets are almost zero, showing that ResNet is thoroughly capable of learning the data set. In Table III, the test error of ResNet is lower than that of AlexNet, showing that the deeper CNN architecture affects the better error. Without the tuning the weight decay value as in dropout and grid-λ, our test errors are comparable to those of dropout and grid-λ. Fig. 4(e) and (f) shows the weight decay parameter determined by the IIM&META applied to DCNN (AlexNet) and ResNet. The elapsed times of learning with determining the decay parameter with respect to grid-λ, dropout, and the IIM&META are summarized in Fig. 3, demonstrating that the proposed method requires the shortest time.

D. Deep Layer With IIM&META

Using the cost-effective setting of λ by the IIM&META, a ResNet architecture can be studied to obtain a better classification result on CIFAR-100 which is the most challenging data set in this paper. The depth of ResNet is adjusted rather than the width since this paper considers that the depth of CNN affects the good classification performance. The experimental condition is identical to the original work [24]. The depth of the network is set between 164 and 227. The 227-layered ResNet yields the minimum test error. Its training and test errors are summarized in Table IV, which shows a better test error than those of the 164-layered ResNets trained by grid-λ, dropout, and the IIM&META.

E. Learning Multiple NNs With IIM&META

In applications, there is an experiment that is usually conducted with different CNN models to obtain the best one among the test errors of those models (e.g., LeNet, AlexNet, and ResNet) for a specific image data set of interest. For such an experiment, the time complexity of CNN learning with determining the weight decay can be considered as follows. Given M different CNN models, the best test error is obtained with $O(C\sum_{i=1}^{M} W_i)$ using grid-λ or dropout, where C and W_i denote the number of weight decay values and the number of the ith CNN weights, respectively. By using the proposed method, the best error can be obtained with $O(\sum_{i=1}^{M} W_i)$. In the NORB experiment with LeNet and AlexNet ($M = 2$), our method reduces the total elapsed time by nearly 70%–80%, resulting in the test error that is slightly better than those of

grid-λ and dropout. The best test error on NORB is obtained by AlexNet trained with the IIM&META. In contrast, our method has a tradeoff between the computational efficiency and exactness in terms of setting λ. In the CIFAR-100 results of ResNet with a fixed architectural parameter, the test error of our method is slightly worse than that of dropout, as shown in CIFAR-100_{R} in Table III. Nevertheless, the discrepancy in those test errors is quite small, and the computational efficiency of our method is considered high, especially for training large-scale NNs. In addition, this drawback could be overcome by the deep layer search, which is shown in Table IV.

V. Conclusion

A method to efficiently determine the weight decay parameter value was established through the proposed analytic solutions with the novel objective function IIM for multiclass classification, derived by the Bayesian method. Based on the concerns regarding the computational cost due to a CNN of several hundreds of layers, the approximation method META was suggested for its efficient computation. The IIM&META is suitable for the real-world data sets, verified with different CNNs. It has a provable bound that guarantees the consistent result. In experiments, the proposed method IIM&META significantly reduced the time of CNN learning with the decay parameter setting while achieving classification accuracies that were comparable with or better than those of the grid search approach and dropout. It is worthy of consideration that there exists a tradeoff between the time efficiency and exactness in terms of determining λ. Our approach may be worse than the grid search approach with a large amount of computations, especially for a specified architecture (e.g., a fixed depth and width) of CNN. In contrast, our method allows researchers or practitioners to rapidly explore or investigate a variety of deeper CNNs that produce a small generalization error. This is a notable advantage of our work for deep networks.

For future work, an interesting direction is to generalize our approach to other deep learning models with a rigorous analysis, since our method can be applied to an NN trained by the cross-entropy loss which requires the weight decay parameter setting. Because dropout and its variants use the weight decay, our method would be applied to those, where some improvement may be required.

We believe that our result of the use of partial Hessian information will provide better insight into research on large-scale NNs.

Appendix
Derivation of (15)

All notations of this derivation follow Section III-A. We use the Gaussian approximation (i.e., Laplace method) to evaluate the following marginal:

$$p(D|\eta,\zeta)=\int p(D|\mathbf{w},\zeta)p(\mathbf{w}|\eta)d\mathbf{w}. \tag{23}$$

Let $Q_{i,k}(\mathbf{w})=\mathbb{1}(t_k^{(i)}=1)\log f_k(\mathbf{x}^{(i)},\mathbf{w})$ as defined in (10), $A(\mathbf{w})=-\zeta\sum_{i=1}^{n}\sum_{k=1}^{c}Q_{i,k}(\mathbf{w})+\eta\|\mathbf{w}\|_2^2$, and $\mathbf{w}^*$ be a local optimum point. Then, the Taylor expansion approximates $A(\mathbf{w})$ to

$$\bar{A}(\mathbf{w})=A(\mathbf{w}^*)+\frac{1}{2}(\mathbf{w}-\mathbf{w}^*)^{\mathbb{T}}\mathbf{H}(\mathbf{w}-\mathbf{w}^*) \tag{24}$$

where $\mathbf{H}$ is the Hessian matrix of $A(\mathbf{w})$ and the approximation of the high-order term is omitted. Therefore, the marginal is given by $p(D|\eta,\zeta)=\int p(D|\mathbf{w},\zeta)p(\mathbf{w}|\eta)d\mathbf{w}$, where $p(D|\mathbf{w},\zeta)p(\mathbf{w}|\eta)\approx\exp(-\bar{A})/(\pi/\eta)^{(W/2)}$. With the integral term $\int\exp(-\bar{A})d\mathbf{w}=(2\pi)^{(W/2)}\det(\mathbf{H})^{(-1/2)}A(\mathbf{w}^*)$, (15) is evaluated as follows:

$$p(D|\eta,\zeta)=\frac{2^{W/2}\exp\left(\zeta\sum_{i=1}^{n}\sum_{k=1}^{c}Q_{i,k}(\mathbf{w}^*)-\eta\|\mathbf{w}^*\|_2^2\right)}{\eta^{-W/2}\det(\mathbf{H})^{\frac{1}{2}}}. \tag{25}$$

Acknowledgment

The authors would like to thank the anonymous reviewers for their constructive comments and insightful suggestions.

References

[1] 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, doi: 10.1109/TPAMI.2013.50.

[2] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Proc. 19th Int. Conf. Neural Inf. Process. Syst. (NIPS)*. Cambridge, MA, USA: MIT Press, 2006, pp. 153–160. [Online]. Available: http://dl.acm.org/citation.cfm?id=2976456.2976476

[3] C. Zhang, K. C. Tan, H. Li, and G. S. Hong, "A cost-sensitive deep belief network for imbalanced classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 1, pp. 109–122, Jan. 2019.

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Lake Tahoe, NV, USA, 2012, pp. 1106–1114.

[6] G. Li and Y. Yu, "Contrast-oriented deep neural networks for salient object detection," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 12, pp. 6038–6051, Dec. 2018.

[7] M. Mahmud, M. S. Kaiser, A. Hussain, and S. Vassanelli, "Applications of deep learning and reinforcement learning to biological data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2063–2079, Jun. 2018.

[8] C. Yan, H. Xie, D. Yang, J. Yin, Y. Zhang, and Q. Dai, "Supervised hash coding with deep neural network for environment perception of intelligent vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 1, pp. 284–295, Jan. 2018.

[9] J. Li, X. Mei, D. Prokhorov, and D. Tao, "Deep neural network for structural prediction and lane detection in traffic scene," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 3, pp. 690–703, Mar. 2017.

[10] B. H. Kim and S. Jo, "Deep physiological affect network for the recognition of human emotions," *IEEE Trans. Affective Comput.*, to be published, doi: 10.1109/TAFFC.2018.2790939.

[11] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, 1989.

[12] J. Park and I. W. Sandberg, "Universal approximation using radial-basis-function networks," *Neural Comput.*, vol. 3, no. 2, pp. 246–257, Mar. 1991.

[13] H. Gao, W. He, C. Zhou, and C. Sun, "Neural network control of a two-link flexible robotic manipulator using assumed mode method," *IEEE Trans. Ind. Informat.*, to be published, doi: 10.1109/TII.2018.2818120.

[14] W. He, Z. Yan, C. Sun, and Y. Chen, "Adaptive neural network control of a flapping wing micro aerial vehicle with disturbance observer," *IEEE Trans. Cybern.*, vol. 47, no. 10, pp. 3452–3465, Oct. 2017.

[15] J. Tang, C. Deng, and G.-B. Huang, "Extreme learning machine for multilayer perceptron," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 4, pp. 809–821, Apr. 2016.
[16] H. Li, H. Zhao, and H. Li, "Neural-response-based extreme learning machine for image classification," *IEEE Trans. Neural Netw. Learn. Syst.*, doi: 10.1109/TNNLS.2018.2845857.
[17] C. Nadeau and Y. Bengio, "Inference for the generalization error," *Mach. Learn.*, vol. 52, no. 3, pp. 239–281, Sep. 2003, doi: 10.1023/A:1024068626366.
[18] V. N. Vapnik, *Statistical Learning Theory*. Hoboken, NJ, USA: Wiley, 1998.
[19] P. L. Bartlett, "For valid generalization the size of the weights is more important than the size of the network," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, Denver, CO, USA, Dec. 1996, pp. 134–140.
[20] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *Advances in Neural Information Processing Systems*. Cambridge, MA, USA: MIT Press, 1993, pp. 164–171.
[21] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *Advances in Neural Information Processing Systems*. Cambridge, MA, USA: MIT Press, 1993, pp. 164–171.
[22] D. J. C. MacKay, "Bayesian interpolation," *Neural Comput.*, vol. 4, no. 3, pp. 415–447, 1992.
[23] D. J. C. MacKay, "The evidence framework applied to classification networks," *Neural Comput.*, vol. 4, no. 5, pp. 720–736, Sep. 1992.
[24] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2016, pp. 630–645.
[25] R. K. Srivastava, K. Greff, and J. Schmidhuber. (2015). "Highway networks." [Online]. Available: http://arxiv.org/abs/1505.00387
[26] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 1, pp. 1929–1958, Jan. 2014.
[27] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proc. 30th Int. Conf. Mach. Learn. (ICML)*, vol. 28, May 2013, pp. 1058–1066.
[28] S. Scardapane, D. Comminiello, A. Hussain, and A. Uncini, "Group sparse regularization for deep neural networks," *Neurocomputing*, vol. 241, pp. 81–89, Jun. 2017.
[29] G. Kang, J. Li, and D. Tao, "Shakeout: A new approach to regularized deep neural network training," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 5, pp. 1245–1258, May 2018.
[30] Z. Chen and S. Haykin, "On different facets of regularization theory," *Neural Comput.*, vol. 14, no. 12, pp. 2791–2846, 2002.
[31] O. Bousquet, S. Boucheron, and G. Lugosi, "Introduction to statistical learning theory," in *Advanced Lectures on Machine Learning*, O. Bousquet, Ed. Berlin, Germany: Springer, 2003, pp. 169–207.
[32] P. Guo, M. R. Lyu, and C. L. P. Chen, "Regularization parameter estimation for feedforward neural networks," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 33, no. 1, pp. 35–44, Feb. 2003.
[33] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural networks architectures," *Neural Comput.*, vol. 7, no. 2, pp. 219–269, Mar. 1995.
[34] V. Vapnik, "Principles of risk minimization for learning theory," in *Advances in Neural Information Processing Systems*, J. E. Moody, S. J. Hanson, and R. P. Lippmann, Eds. San Mateo, CA, USA: Morgan Kaufmann, 1992, pp. 831–838. [Online]. Available: http://papers.nips.cc/paper/506-principles-of-risk-minimization-for-learning-theory.pdf
[35] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, nos. 1–3, pp. 489–501, 2006.
[36] R. M. Neal, *Bayesian Learning for Neural Networks*. Secaucus, NJ, USA: Springer-Verlag, 1996.
[37] D. Wu and J. Ma, "A two-layer mixture model of Gaussian process functional regressions and its MCMC EM algorithm," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 10, pp. 4894–4904, Oct. 2018.
[38] Z. Ghahramani, "Probabilistic machine learning and artificial intelligence," *Nature*, vol. 521, no. 7553, pp. 452–459, May 2015, doi: 10.1038/nature14541.
[39] V. Vapnik, *Estimation of Dependences Based on Empirical Data: Springer Series in Statistics* (Springer Series in Statistics). Secaucus, NJ, USA: Springer-Verlag, 1982.
[40] V. Vapnik, *The Nature of Statistical Learning Theory*. New York, NY, USA: Springer-Verlag, 1995.
[41] N. Harvey, C. Liaw, and A. Mehrabian, "Nearly-tight VC-dimension bounds for piecewise linear neural networks," in *Proc. Conf. Learn. Theory* (Proceedings of Machine Learning Research), vol. 65, S. Kale and O. Shamir, Eds. Amsterdam, The Netherlands: PMLR, Jul. 2017, pp. 1064–1068. [Online]. Available: http://proceedings.mlr.press/v65/harvey17a.html
[42] T. V. Gestel *et al.*, "Financial time series prediction using least squares support vector machines within the evidence framework," *IEEE Trans. Neural Netw.*, vol. 12, no. 4, pp. 809–821, Jul. 2001. [Online]. Available: http://dx.doi.org/10.1109/72.935093
[43] J. T.-Y. Kwok, "The evidence framework applied to support vector machines," *IEEE Trans. Neural Netw.*, vol. 11, no. 5, pp. 1162–1173, Sep. 2000.
[44] K. P. Murphy and F. Bach, *Machine Learning—A Probabilistic Perspective*. Cambridge, MA, USA: MIT Press, 2012.
[45] C. M. Bishop and G. Hinton, *Neural Networks for Pattern Recognition*. New York, NY, USA: Oxford Univ. Press, 1995, pp. 409–411.
[46] A. J. Shepherd, *Second-Order Methods for Neural Networks*, 1st ed. Berlin, Germany: Springer-Verlag, 1997, pp. 60–62.
[47] Y. LeCun, C. Cortes, and C. J. Burges, "MNIST handwritten digit database," AT&T Labs, Florham Park, NJ, USA, Tech. Rep., 2010, vol. 2.
[48] Y. LeCun, F. J. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 97–104.
[49] A. Krizhevsky, "Learning multiple layers of features from tiny images," M.S. thesis, 2009. [Online]. Available: http://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf
[50] A. Vedaldi and K. Lenc, "Matconvnet: Convolutional neural networks for matlab," in *Proc. 23rd ACM Int. Conf. Multimedia*, New York, NY, USA, 2015, pp. 689–692.
[51] A. Coates, A. Y. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proc. AISTATS*, 2011, pp. 215–223.

Jung-Guk Park received the B.S. and M.S. degrees in computer science and engineering from Sejong University, Seoul, South Korea, in 2009 and 2012, respectively. He is currently pursuing the Ph.D. degree with the School of Computing, Korea Advanced Institute of Science and Technology, Daejeon, South Korea.

His current research interests include complexity and generalization of computational learning models, and learning algorithms for neural networks, based on statistical learning theory, empirical processes, and approximation theory.

Sungho Jo (M'09) received the B.S. degree from the School of Mechanical and Aerospace Engineering, Seoul National University, Seoul, South Korea, in 1999, and the M.S. degree in mechanical engineering and the Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, in 2001 and 2006, respectively.

He was with the Computer Science and Artificial Intelligence Laboratory and the Laboratory for Information Decision and Systems, MIT. From 2006 to 2007, he was a Post-Doctoral Researcher with the MIT Media Lab, Cambridge, MA, USA. Since 2007, he has been with the School of Computing, Korea Advanced Institute of Science and Technology, Daejeon, South Korea, where he is currently an Associate Professor. His current research interests include neuromachine learning systems, neuro-inspired intelligence, and machine-augmented intelligence.