Ozaki *et al. IPSJ Transactions on Computer Vision and Applications* (2017) 9:20
DOI 10.1186/s41074-017-0030-7

IPSJ Transactions on Computer Vision and Applications

RESEARCH PAPER Open Access

Effective hyperparameter optimization using Nelder-Mead method in deep learning

Yoshihiko Ozaki[1,2], Masaki Yano[1,2] and Masaki Onishi[1,2*]

Abstract

In deep learning, deep neural network (DNN) hyperparameters can severely affect network performance. Currently, such hyperparameters are frequently optimized by several methods, such as Bayesian optimization and the covariance matrix adaptation evolution strategy. However, it is difficult for non-experts to employ these methods. In this paper, we adapted the simpler coordinate-search and Nelder-Mead methods to optimize hyperparameters. Several hyperparameter optimization methods were compared by configuring DNNs for character recognition and age/gender classification. Numerical results demonstrated that the Nelder-Mead method outperforms the other methods and achieves state-of-the-art accuracy for age/gender classification.

Keywords: Hyperparameter optimization, Nelder-Mead method, Deep learning, Convolutional neural network

1 Introduction

The evolution of deep neural networks (DNNs) has dramatically improved the accuracy of character recognition [1], object recognition [2, 3], and other tasks. However, the their increasing complexity increases the number of hyperparameters, which makes tuning of hyperparameters an intractable task.

Traditionally, DNN hyperparameters are adjusted using manual search, grid search, or random search [4]. However, search space expands exponentially relative to the number of hyperparameters; thus, such naive methods no longer work well. Therefore, more sophisticated hyperparameter optimization methods are required.

In deep learning, a hyperparameter optimization problem can be formulated as a stochastic black box optimization problem to minimize a noisy black box objective function $f(\mathbf{x})$:

$$\text{Minimize } f(\mathbf{x}) \quad (\mathbf{x} \in \chi). \tag{1}$$

Here, using all information available about the objective function, we can obtain its value at point $\mathbf{x}$ with noise ϵ as follows:

$$y = f(\mathbf{x}) + \epsilon, \quad \epsilon \overset{iid}{\sim} \mathcal{N}\left(0, \sigma^2_{\text{noise}}\right). \tag{2}$$

This means that no analytical properties of the objective function, e.g., its derivatives, can be optimized. In addition, a loss function of the target DNN is typically chosen as $f(\mathbf{x})$, and its evaluation cost is so expensive that training and testing of the DNN is required. The search space χ comprises combinations of multiple conditions such as real numbers, integers, and categories.

Currently, Bayesian optimization [5] and the covariance matrix adaptation evolution strategy (CMA-ES) [6] are considered the most promising methods for DNN hyperparameter optimization, and their optimization ability has been proven experimentally [7–10]. However, Bayesian optimization has some hyperparameters that significantly affect its optimization performance, e.g., choices of its kernel and acquisition function. Moreover, maximizing a non-convex acquisition function is required for each iteration of the optimization process. On the other hand, CMA-ES requires several populations and generations for sufficient performance. Although such calculations can be parallelized easily, significant computing resources are required.

It is evident that simple classical manual search, grid search, and random search remain common; thus, we consider that most people are unwilling to adjust the hyperparameters of a difficult optimization method or implement the method and do not have sufficient computing resources to optimize DNN hyperparameters.

*Correspondence: onishi@ni.aist.go.jp
[1]Artificial Intelligence Research Center, AIST, 1-1-1 Umezono, Tsukuba, Ibaraki, Japan
[2]University of Tsukuba, 1-1-1 Tennodai, Tsukuba, Ibaraki, Japan

In this paper, we describe simple substitutional methods, i.e., the coordinate-search and Nelder-Mead methods, for hyperparameter optimization in deep learning. To the best of our knowledge, no report has examined the application of these methods to hyperparameter optimization.

Our numerical results indicate that these methods are more efficient than other well-known methods. In particular, the Nelder-Mead method is the most effective for deep learning.

2 Related work

2.1 Random search

Random search is one of the simplest ways to optimize DNN hyperparameters. This method iteratively generates hyperparameter settings and evaluates the objective function. Random search has excellent parallelization and can handle integer and categorical hyperparameters naturally. Bergstra and Bengio demonstrated that random search outperforms a manual search by a human expert and grid search [4].

2.2 Bayesian optimization

Bayesian optimization is one of the most remarkable hyperparameter optimization methods in recent years. Its base concept was proposed in the 1970s; however, it has been significantly improved since then due to the attention paid to DNN hyperparameter optimization.

There are several variations of Bayesian optimization, e.g., Gaussian process (GP)-based variation [11], Tree-structured Parzen Estimator (TPE) [7], and Deep Networks for Global Optimization (DNGO) [12]. The most standard one is the GP-based variation.

GP-based Bayesian optimization is shown in Algorithm 1. In this method, we assume that an objective

Algorithm 1: Bayesian optimization [11]

for $t = 1, 2, \ldots$ **do**
  Find $\mathbf{x}_t$ by optimizing the acquisition function over GP: $\mathbf{x}_t = \mathrm{argmax}_{\mathbf{x}} u(\mathbf{x}|\mathcal{D}_{1:t-1})$;
  Sample the objective function: $y_t = f(\mathbf{x}_t) + \epsilon_t$;
  Augment the data $D_{1:t} = \{D_{1:t-1}, (\mathbf{x}_t, y_t)\}$ and update the GP;
end

function follows the GP specified by its mean function m and kernel k:

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')). \tag{3}$$

For simplicity, we assume $m(\mathbf{x}) = 0$. Then, we must consider the kernel $k(\mathbf{x}, \mathbf{x}')$. For the kernel, an automatic relevance determination (ARD) squared exponential (SE) kernel

$$\begin{aligned} &K_{\mathrm{SE}}(\mathbf{x}, \mathbf{x}') = \theta_0 \exp\left(-\frac{1}{2} r^2(\mathbf{x}, \mathbf{x}')\right), \\ &\text{where } r^2(\mathbf{x}, \mathbf{x}') = \sum_{d=1}^{D} \left(x_d - x'_d\right)^2 / \theta_d^2, \end{aligned} \tag{4}$$

or ARD Matérn 5/2 kernel

$$\begin{aligned} &K_{\mathrm{M52}}(\mathbf{x}, \mathbf{x}') \\ &= \theta_0 \left(1 + \sqrt{5r^2(\mathbf{x}, \mathbf{x}') + \frac{5}{3} r^2(\mathbf{x}, \mathbf{x}')}\right) \exp\left(-\sqrt{5r^2(\mathbf{x}, \mathbf{x}')}\right), \end{aligned} \tag{5}$$

is commonly used in Bayesian optimization [8]. Here, θ_0, $\theta_1, \ldots,$ and θ_D are the kernel's hyperparameters.

Once $k(\mathbf{x}, \mathbf{x}')$ is determined, we can predict information about a new sample point $\mathbf{x}_{t+1}$ from previous observations $\mathcal{D}_{1:t} = \{\mathbf{x}_{1:t}, y_{1:t}\}$:

$$P(y_{t+1}|\mathcal{D}_{1:t}, \mathbf{x}_{t+1}) = \mathcal{N}\left(\mu_t(\mathbf{x}_{t+1}), \sigma_t^2(\mathbf{x}_{t+1}) + \sigma_{\mathrm{noise}}^2\right), \tag{6}$$

$$\mu_t(\mathbf{x}_{t+1}) = \mathbf{k}^T \left[\mathbf{K} + \sigma_{\mathrm{noise}}^2 \mathbf{I}\right]_{y_{1:t}}^{-1}, \tag{7}$$

$$\sigma_t^2(\mathbf{x}_{t+1}) = k(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}) - \mathbf{k}^T \left[\mathbf{K} + \sigma_{\mathrm{noise}}^2 \mathbf{I}\right]^{-1} \mathbf{k} \tag{8}$$

$$\text{where } \mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_t) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_t, \mathbf{x}_1) & \cdots & k(\mathbf{x}_t, \mathbf{x}_t) \end{bmatrix} + \sigma_{\mathrm{noise}}^2 \mathbf{I},$$

$$\mathbf{k} = \left[k(\mathbf{x}_{t+1}, \mathbf{x}_1)\ k(\mathbf{x}_{t+1}, \mathbf{x}_2)\ \cdots\ k(\mathbf{x}_{t+1}, \mathbf{x}_t)\right].$$

The remaining problem is how to determine new sample points iteratively. To determine new candidates for a sample, we generally employ an acquisition function. Here, it is necessary to select an acquisition function that achieves a good balance between exploration and exploitation based on past observations. One well-known acquisition function is expected improvement (EI):

$$\mathrm{EI}(\mathbf{x}) = \begin{cases} (\mu(\mathbf{x}) - f(\mathbf{x}^+))\Phi(Z) + \sigma(\mathbf{x})\Phi(Z) & (\sigma(\mathbf{x}) > 0) \\ 0 & (\sigma(\mathbf{x}) = 0) \end{cases} \tag{9}$$

$$\text{where } Z = \frac{\mu(\mathbf{x}) - f(\mathbf{x}^+)}{\sigma(\mathbf{x})},\ \mathbf{x}^+ = \mathrm{argmax}_{\mathbf{x}_i \in \mathbf{x}_{1:t}} f(\mathbf{x}_i).$$

The point that maximizes the acquisition function becomes a new sample point. Although maximizing the non-convex acquisition function is difficult, the evaluation cost of the function is considerably less than that of the original objective function. Therefore, it is easier to handle than the original problem.

Practically, Bayesian optimization is combined with random search to collect initial observation data.

Bergstra et al. and Snoek et al. performed several computational experiments. The results demonstrated that Bayesian optimization outperforms manual search by a human expert and random search [7, 8].

2.3 Covariance matrix adaptation evolution strategy

While Bayesian optimization has been developed in the machine learning community, CMA-ES has been developed in the optimization community. CMA-ES is a type of evolutionary computation that demonstrates outstanding performance in benchmarks as a state-of-the-art black box optimization method [13].

The (μ_{W}, λ)-CMA-ES [6] is shown in Algorithm 2. It conducts a weighted recombination from the μ best out of λ individuals. The procedure is explained as follows.

Algorithm 2: The (μ_{W}, λ)-CMA-ES [6]

Initialize the mean $\langle \mathbf{x} \rangle_{\mathrm{w}}^{(0)}$, and the standard deviation $\sigma^{(0)}$;
$\mathbf{p}_{\mathrm{c}}^{(0)} = \mathbf{0}$; $\mathbf{p}_{\sigma}^{(0)} = \mathbf{0}$; $\mathbf{C}^{(0)} = \mathbf{I}$;
for $g = 0, 1, \ldots$ **do**
  Generate $g+1$ generation λ individuals;
  Select μ best individuals;
  Update the evolution path and the covariance matrix;
  Update the evolution path and the step size;
end

(i) Initialize the mean $\langle \mathbf{x} \rangle_{\mathrm{w}}^{(0)}$ and the standard deviation $\sigma^{(0)}$ of individuals. Set the evolution path $\mathbf{p}_{\mathrm{c}}^{(0)} = \mathbf{p}_{\sigma}^{(0)} = \mathbf{0}$ and the covariance matrix $\mathbf{C}^{(0)} = \mathbf{I}$.

(ii) Generate $g+1$ generation individuals $\mathbf{x}_k^{(g+1)}$ $(k = 1, \ldots, \lambda)$:

$$\mathbf{x}_k^{(g+1)} = \langle \mathbf{x} \rangle_{\mathrm{w}}^{(g)} + \sigma^{(g)} \mathbf{B}^{(g)} \mathbf{D}^{(g)} \mathbf{z}_k^{(g+1)}, \quad (10)$$

where

$$\langle \mathbf{x} \rangle_{\mathrm{w}}^{(g)} := \frac{1}{\sum_{i=1}^{\mu} w_i} \sum_{i=1}^{\mu} w_i \mathbf{x}_{i:\lambda}^{(g)},$$

$$\mathbf{B}^{(g)} \mathbf{D}^{(g)} \left(\mathbf{B}^{(g)} \mathbf{D}^{(g)} \right)^T = \mathbf{C}^{(g)},$$

$$\mathbf{z}_k^{(g+1)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).$$

Here, $w_1, \ldots, w_\mu$ are weights, and $i{:}\lambda$ denotes the ith best individual.

(iii) Update the evolution path $\mathbf{p}_{\mathrm{c}}^{(g)}$ and the covariance matrix $\mathbf{C}^{(g)}$:

$$\mathbf{p}_{\mathrm{c}}^{(g+1)} = (1 - c_{\mathrm{c}}) \mathbf{p}_{\mathrm{c}}^{(g)} + c_{\mathrm{c}}^{\mathrm{u}} c_{\mathrm{w}} \mathbf{B}^{(g)} \mathbf{D}^{(g)} \langle \mathbf{z} \rangle_{\mathrm{w}}^{(g+1)}, \quad (11)$$

$$\mathbf{C}^{(g+1)} = (1 - c_{\mathrm{cov}}) \mathbf{C}^{(g)} + c_{\mathrm{cov}} \mathbf{p}_{\mathrm{c}}^{(g+1)} \left(\mathbf{p}_{\mathrm{c}}^{(g+1)} \right)^T, \quad (12)$$

where

$$c_{\mathrm{c}}^{\mathrm{u}} := \sqrt{c_{\mathrm{c}}(2 - c_{\mathrm{c}})},$$

$$c_{\mathrm{w}} := \frac{\sum_{i=1}^{\mu} w_i}{\sqrt{\sum_{i=1}^{\mu} w_i^2}},$$

$$\langle \mathbf{z} \rangle_{\mathrm{w}}^{(g+1)} := \frac{1}{\sum_{i=1}^{\mu} w_i} \sum_{i=1}^{\mu} w_i \mathbf{z}_{i:\lambda}^{(g+1)}.$$

Here, c_{c} and c_{cov} are hyperparameters.

(iv) Update the evolution path $\mathbf{p}_{\sigma}^{(g)}$ and the step size $\sigma^{(g)}$:

$$\mathbf{p}_{\sigma}^{(g+1)} = (1 - c_{\sigma}) \mathbf{p}_{\sigma}^{(g)} + c_{\sigma}^{\mathrm{u}} c_{\mathrm{w}} \mathbf{B}^{(g)} \langle \mathbf{z} \rangle_{\mathrm{w}}^{(g+1)}, \quad (13)$$

$$\sigma^{(g+1)} = \sigma^{(g)} \exp \left(\frac{1}{d_{\sigma}} \frac{\| \mathbf{p}_{\sigma}^{(g+1)} - \hat{\chi}_{\mathrm{n}} \|}{\hat{\chi}_{\mathrm{n}}} \right), \quad (14)$$

where

$$c_{\sigma}^{\mathrm{u}} := \sqrt{c_{\sigma}(2 - c_{\sigma})},$$

$$\hat{\chi}_{\mathrm{n}} = \mathbb{E}[\, \| \mathcal{N}(\mathbf{0}, \mathbf{I}) \| \,].$$

Here, c_{σ} and d_{σ} are hyperparameters.

Details about the hyperparameters used to update CMA-ES parameters are provided in the literature [6].

Since evaluations of each individual for each generation can be calculated simultaneously, CMA-ES can be parallelized easily. Watanabe and Le Roux and Loshchilov et al. demonstrated that CMA-ES outperforms manual search by a human expert and Bayesian optimization in certain cases [9, 10].

3 Coordinate-search and Nelder-Mead methods

In the previous section, we introduced the random search, Bayesian optimization, and CMA-ES methods. Note that the achievements of these methods have already been proven experimentally, and the results indicate that the latter two methods are very promising and considered superior to random search. However, both Bayesian optimization and CMA-ES have many hyperparameters related to their optimization performance. To set these hyperparameters appropriately, it is necessary to have sufficient knowledge about the given method. In addition, Bayesian optimization must maximize its non-convex acquisition function, and CMA-ES requires significant computing resources to exploit its advantages. These factors make it difficult for non-experts to utilize these methods.

In this section, we introduce two optimization methods, coordinate-search and Nelder-Mead, that are easy to implement. These methods have fewer hyperparameters to adjust and are practically usable with fewer computing resources.

3.1 Mathematical concepts

Before introducing the methods, we define some required mathematical concepts here.

Definition 1 *The positive span of a set of vectors* $[\, \mathbf{v}_1 \cdots \mathbf{v}_r] \in \mathbb{R}^n$ *is the convex cone:*

$$\{ \mathbf{v} \in \mathbb{R}^n : \mathbf{v} = \alpha_1 \mathbf{v}_1 + \cdots + \alpha_r \mathbf{v}_r,\ \alpha_i \geq 0,\ i = 1, \ldots, r \}.$$

Definition 2 *A positive spanning set in* $\mathbb{R}^n$ *is a set of vectors whose positive span is* $\mathbb{R}^n$.

Definition 3 *The set* $[\mathbf{v}_1 \cdots \mathbf{v}_r] \in \mathbb{R}^n$ *is considered positively dependent if one of the vectors is in the positive span of the remaining vectors; otherwise, the set is considered positively independent.*

Definition 4 *A positive basis for* $\mathbb{R}^n$ *is a positively independent set whose positive span is* $\mathbb{R}^n$*. A positive basis for* $\mathbb{R}^n$ *that has* $n+1$ *vectors is considered a minimal positive basis and a positive basis that has* $2n$ *vectors is considered a maximal positive basis (Fig.* 1*). Here, a maximal positive basis is denoted as* $\mathbf{D}_\oplus$*.*

Definition 5 *A simplex of dimension* m *is a convex hull of an affinely independent set of points* $\mathbf{Y} = \{\mathbf{y}^0, \mathbf{y}^1, \ldots, \mathbf{y}^m\}$*.*

3.2 Coordinate-search method

The coordinate-search method [14] (Algorithm 3, Fig. 2) is one of the simplest direct search methods. It minimizes its objective function iteratively using the maximal positive basis $\mathbf{D}_\oplus = [\mathbf{I} \ -\mathbf{I}] = [\mathbf{e}_1 \ \cdots \ \mathbf{e}_n \ -\mathbf{e}_1 \ \cdots \ -\mathbf{e}_n]$.

Algorithm 3: Coordinate-search method [14]

Initialization: Choose $\mathbf{x}_0$ and $\alpha_0 > 0$;
for $k = 0, 1, \ldots$ **do**
  Poll step;
  Parameter update;
end

This method performs a poll step iteratively to search a better solution and updates parameters to adjust its learning rate.

(i) **Poll step:** Order the poll set $\mathbf{P}_k = \{\mathbf{x}_k + \alpha_k \mathbf{d} : \mathbf{d} \in \mathbf{D}_\oplus\}$. Evaluate f at the poll points in order. If a poll point $\mathbf{x}_k + \alpha_k \mathbf{d}_k$ that satisfies the condition $f(\mathbf{x}_k + \alpha_k \mathbf{d}_k) < f(\mathbf{x}_k)$ is found, then stop polling, set $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$, and declare the iteration and poll step successful.

(ii) **Parameter update:** If iteration k succeeds, then set $\alpha_{k+1} = \alpha_k$ (or $\alpha_{k+1} = 2\alpha_k$). Otherwise, set $\alpha_{k+1} = \alpha_k/2$.

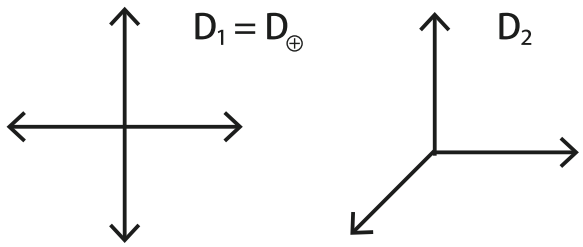

Fig. 1 Maximal positive basis ($\mathbf{D}_1$) and minimal positive basis ($\mathbf{D}_2$) in $\mathbb{R}^2$

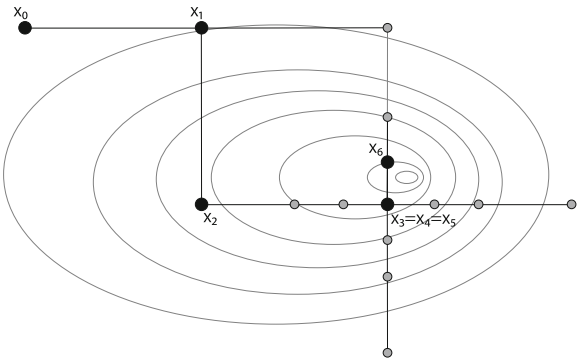

Fig. 2 Coordinate-search method

When the step size becomes sufficiently small, the search is terminated. Note that the evaluation of functions in the poll step can be parallelized.

This method deteriorates the performance for search ranges with different scales; thus, in this study, we normalize parameters to [0, 1] in our computational experiments. In addition, we adopt the updating rule $\alpha_{k+1} = 2\alpha_k$ on iteration success and order the vectors of the poll set randomly for each iteration.

3.3 Nelder-Mead method

The Nelder-Mead method [14, 15] (Algorithm 4, Fig. 3) is an optimization method that uses a simplex proposed by Nelder and Mead. Gilles et al. applied this method for the hyperparameter tuning problem in support vector machine modeling. They demonstrated that the method can find very good hyperparameter settings reliably for support vector machines [16]. Currently, the Nelder-Mead method is not considered in DNN research; however, it has a long history and many achievements in other research areas [14]. Thus, we think it is worth considering

Algorithm 4: Nelder-Mead method [15]

Initialization: Choose an initial simplex of vertices $\mathbf{Y}_0 = \{\mathbf{y}_0^0, \mathbf{y}_0^1, \ldots, \mathbf{y}_0^n\}$. Evaluate f at the points in $\mathbf{Y}_0$. Choose constants:

$$0 < \gamma^s < 1, \quad -1 < \delta^{ic} < 0 < \delta^{oc} < \delta^r < \delta^e.$$

for $k = 0, 1, \ldots$ **do**
  Set $\mathbf{Y} = \mathbf{Y}_k$;
  Order;
  Reflect;
  Expand;
  Contract;
  Shrink;
end

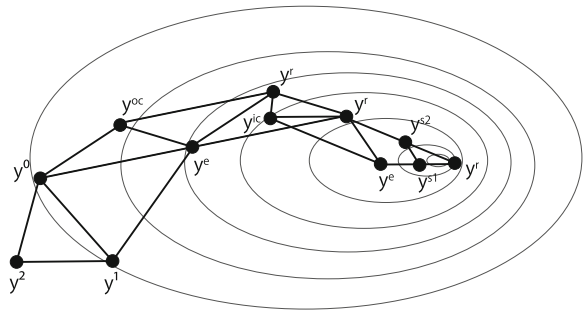

Fig. 3 Nelder-Mead method

the Nelder-Mead method for DNN hyperparameter optimization. In the study by Gilles et al., their SVM has only two hyperparameters. On the other hand, DNNs often have more than 10 times number of hyperparameters. So, our task is more challenging.

The Nelder-Mead method minimizes the objective function by repeating its evaluation at each vertex of the simplex and by replacing points according to the following procedure (Figs. 4 and 5).

(i) **Order:** Order the $n+1$ vertices $\mathbf{Y} = \{\mathbf{y}^0, \mathbf{y}^1, \ldots, \mathbf{y}^n\}$ as follows:

$$f^0 = f(\mathbf{y}^0) \leq f^1 = f(\mathbf{y}^1) \leq \cdots \leq f^n = f(\mathbf{y}^n).$$

(ii) **Reflect:** Reflect the worst vertex $\mathbf{y}^n$ over the centroid $\mathbf{y}^c = \sum_{i=0}^{n-1} \mathbf{y}^i/n$ of the remaining n vertices:

$$\mathbf{y}^r = \mathbf{y}^c + \delta^r(\mathbf{y}^c - \mathbf{y}^n).$$

Evaluate $f^r = f(\mathbf{y}^r)$. If $f^0 \leq f^r < f^{n-1}$, then replace $\mathbf{y}^n$ with the reflected point $\mathbf{y}^r$ and terminate iteration k: $\mathbf{Y}_{k+1} = \{\mathbf{y}^0, \mathbf{y}^1, \ldots, \mathbf{y}^{n-1}, \mathbf{y}^r\}$.

(iii) **Expand:** If $f^r < f^0$, calculate:

$$\mathbf{y}^e = \mathbf{y}^c + \delta^e(\mathbf{y}^c - \mathbf{y}^n)$$

and evaluate $f^e = f(\mathbf{y}^e)$. If $f^e \leq f^r$, then replace $\mathbf{y}^n$ with the expansion point $\mathbf{y}^e$ and terminate iteration k: $\mathbf{Y}_{k+1} = \{\mathbf{y}^0, \mathbf{y}^1, \ldots, \mathbf{y}^{n-1}, \mathbf{y}^e\}$. Otherwise, replace

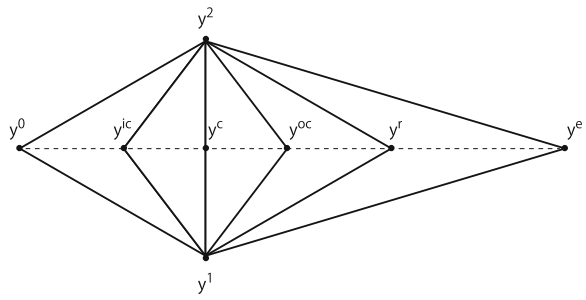

Fig. 4 Reflection, expansion, outside contraction, and inside contraction of a simplex by the Nelder-Mead method

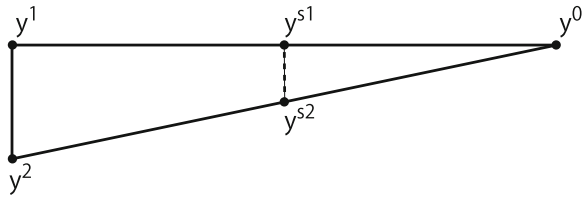

Fig. 5 Shrinking a simplex by the Nelder-Mead method. $\mathbf{y}^1$ and $\mathbf{y}^2$ are shrunk to $\mathbf{y}^{s1}$ and $\mathbf{y}^{s2}$, respectively

$\mathbf{y}^n$ with the reflected point $\mathbf{y}^r$ and terminate iteration k: $\mathbf{Y}_{k+1} = \{\mathbf{y}^0, \mathbf{y}^1, \ldots, \mathbf{y}^{n-1}, \mathbf{y}^r\}$.

(iv) **Contract:** If $f^r \geq f^{n-1}$, then a contraction is performed between the best of $\mathbf{y}^r$ and $\mathbf{y}^n$.

(a) **Outside contraction:** If $f^r < f^n$, perform an outside contraction:

$$\mathbf{y}^{oc} = \mathbf{y}^c + \delta^{oc}(\mathbf{y}^c - \mathbf{y}^n)$$

and evaluate $f^{oc} = f(\mathbf{y}^{oc})$. If $f^{oc} \leq f^r$, then replace $\mathbf{y}^n$ with the outside contraction point $\mathbf{y}_k^{oc}$ and terminate iteration k: $\mathbf{Y}_{k+1} = \{\mathbf{y}^0, \mathbf{y}^1, \ldots, \mathbf{y}^{n-1}, \mathbf{y}^{oc}\}$. Otherwise, perform a shrink.

(b) **Inside contraction:** If $f^r \geq f^n$, perform an inside contraction:

$$\mathbf{y}^{ic} = \mathbf{y}^c + \delta^{ic}(\mathbf{y}^c - \mathbf{y}^n)$$

and evaluate $f^{ic} = f(\mathbf{y}^{ic})$. If $f^{ic} < f^n$, then replace $\mathbf{y}^n$ with the inside contraction point $\mathbf{y}^{ic}$ and terminate iteration k: $\mathbf{Y}_{k+1} = \{\mathbf{y}^0, \mathbf{y}^1, \ldots, \mathbf{y}^{n-1}, \mathbf{y}^{ic}\}$. Otherwise, perform a shrink.

(v) **Shrink:** Evaluate f at the n points $\mathbf{y}^0 + \gamma^s(\mathbf{y}^i - \mathbf{y}^0)$, *where* $i = 1, \ldots, n$, replace $\mathbf{y}^1, \ldots, \mathbf{y}^n$ with these points, and terminate iteration k: $\mathbf{Y}_{k+1} = \{\mathbf{y}^0 + \gamma^s(\mathbf{y}^i - \mathbf{y}^0),\ i = 0, \ldots, n\}$.

Here, γ^s, δ^{ic}, δ^{oc}, δ^r, and δ^e are constant hyperparameters usually taking the following values:

$$\gamma^s = \frac{1}{2}, \delta^{ic} = -\frac{1}{2}, \delta^{oc} = \frac{1}{2}, \delta^r = 1 \text{ and } \delta^e = 2. \quad (15)$$

Note that each step of an iteration, e.g., initialization and shrink operations, can be parallelized easily.

4 Poor hyperparameter setting detection

DNNs are very sensitive to hyperparameter settings. As a result, training can fail simply because some hyperparameters, e.g., the learning rate, are slightly inappropriate.

When appropriate hyperparameter values are given, training loss is reduced in each iteration (Fig. 6, top graph); otherwise, regardless of how many iterations have been executed, training loss is not reduced (Fig. 6, bottom graph).

The advantage of human experts is that they can detect training failures and terminate them at an early stage. Domhan et al. proposed a method that accelerates hyperparameter optimization methods by detecting and terminating such training failures using learning curve prediction [17]. In addition, Klein et al. proposed a specialized Bayesian neural network to model DNN learning curves [18, 19]. We apply Algorithm 5 to detect training failures at an early stage.

Algorithm 5: Poor hyperparameter setting detection

Initialize n and t. Calculate initial loss l_0;
After n learning iterations, calculate current loss l_n;
if $t < l_n/l_0$ **then**
  Terminate the training;
end

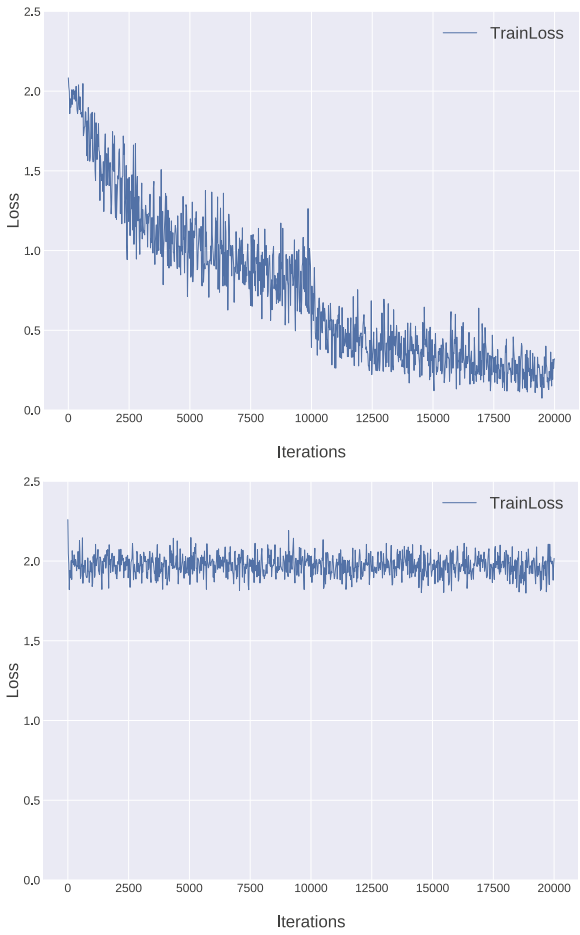

Fig. 6 Learning curve when training is successful (top) and unsuccessful (bottom)

Note that this method does not optimize hyperparameters directly, but accelerates a hyperparameter optimization method. If a large number of training iterations with poor hyperparameter settings appear in the optimization process, this detection process improves the execution time of the optimization method.

In our experiments, we apply this method to all hyperparameter optimization methods with n equaling 10% of the training iterations and t equaling 0.8. These values are chosen based on experience. As can be seen in Fig. 6, the learning curve of poor hyperparameter settings is distinctive and easy to detect; thus, there is no need to be too careful to decide n and t.

5 Numerical results

We perform computational experiments to optimize real and integer hyperparameters in combination with various datasets, tasks, and convolutional neural networks (CNNs) to compare the performance of the random search, Bayesian optimization, CMA-ES, coordinate-search, and Nelder-Mead methods.

The experimental settings for each method are given in Table 1. We use the first 100 random search evaluations to initialize the Bayesian optimization and coordinate-search methods. The number of evaluations and initialization parameters of CMA-ES and Bayesian optimization are determined with reference to the literature [10]. We implement CMA-ES using Distributed Evolutionary Algorithms in Python (DEAP) [20], which is an evolutionary computation framework. In addition, for optimization methods that cannot handle integer values directly, integer hyperparameters are handled as continuous values and rounding is performed when evaluating the objective function.

Table 1 Experimental setting for each method

Method	Detail
Random search	Perform 600 random evaluations.
Bayesian optimization	Initialize the observation data with the first 100 evaluations of the random search, then perform the optimization with exactly 500 evaluations. The kernel is the ARD Matérn 5/2 and the acquisition function is the EI [8, 10].
CMA-ES	Perform 600 evaluations with 20 generations where each generation consists of 30 individuals. $\langle \mathbf{x} \rangle_w^{(0)} = 0.5, \sigma^{(0)} = 0.2$. All variables are scaled to [0, 1] [10].
Coordinate-search method	Initialize $\mathbf{x}_0$ as the best point of the first 100 random search evaluations, then perform optimization for up to 500 evaluations. $\alpha = 0.5$. All variables are scaled to [0, 1].
Nelder-Mead method	Generate an initial simplex randomly, then perform optimization for up to 600 evaluations (including initialization). $\gamma^s = \frac{1}{2}, \delta^{ic} = -\frac{1}{2}, \delta^{oc} = \frac{1}{2}, \delta^r = 1$ and $\delta^e = 2$.

Fig. 7 MNIST database of handwritten digits [21]

5.1 MNIST

The LeNet [1] hyperparameters are optimized by the five methods to measure their performance. This network performs a 10-class classification of the MNIST handwritten digit database [21] (Fig. 7). Here, we use Caffe's tutorial implementation [22, 23]. This implementation uses a rectified linear unit [24] as its activation function rather than the sigmoid used in the original LeNet.

These methods are also applied to the optimization of hyperparameters of the Batch-Normalized Maxout Network in Network proposed by Chang et al. [25]. Note that this network is deeper and has many more hyperparameters to optimize than LeNet.

Tables 2, 3, 4, 5, 6, and 7 show the details of each network, the fixed hyperparameters, the optimized hyperparameters, and the search ranges. Note that preprocessing and augmentation of the training data are not performed.

Table 2 LeNet network architecture [1]

Conv 1	Kernel size: 2, stride: 1, pad: 0
Pool 1 (MAX pooling)	Kernel size: 2, stride: 2, pad: 2
Conv 2	Kernel size: 5, stride: 1, pad: 0
Pool 2 (MAX pooling)	Kernel size: 2, stride: 2, pad: 0
FC 1	

Table 3 LeNet fixed parameters

Name	Description
Iteration	10,000
Batch size	64
Learning rate decay policy	inv (gamma = 0.01, power = 0.75) [29]

Table 4 LeNet hyperparameters

Name	Description	Range
x_1	Learning rate ($= 0.1^{x_1}$)	[1, 4]
x_2	Momentum ($= 1 - 0.1^{x_2}$)	[0.5, 2]
x_3	L2 weight decay	[0.001, 0.01]
x_4^*	FC1 units	[256, 1024]

Integer parameters are marked with *

Table 5 Network architecture of Batch-Normalized Maxout Network in Network [25]

Conv 1	Kernel size: 5, stride: 1, pad: 2, BN
MMLP 1-1	Kernel size: 1, stride: 1, pad: 0, k = 5, BN
MMLP 1-2	Kernel size: 1, stride: 1, pad: 0, k = 5, BN
Pool 1 (AVE pooling)	Kernel size: 3, stride: 2, pad: 0, dropout
Conv 2	Kernel size: 5, stride: 1, pad: 2, BN
MMLP 2-1	Kernel size: 1, stride: 1, pad: 0, k = 5, BN
MMLP 2-2	Kernel size: 1, stride: 1, pad: 0, k = 5, BN
Pool 2 (AVE pooling)	Kernel size: 3, stride: 2, pad: 0, dropout
Conv 3	Kernel size: 3, stride: 1, pad: 1, BN
MMLP 3-1	Kernel size: 1, stride: 1, pad: 0, k = 5, BN
MMLP 3-2	Kernel size: 1, stride: 1, pad: 0, k = 5, BN
Pool 3 (AVE pooling)	

5.2 Age and gender classification

Gil and Tal proposed a CNN for age/gender classification [26]. In these experiments, the hyperparameters of this CNN are optimized by the five methods. This network consists of three convolution layers and two fully connected layers, receives an image, and outputs a gender label or an age group label. We use the implementation available on the project's web page [27]. We test DNNs using the Adience DB [28] for the age/gender classification benchmark used in the literature [26] (Fig. 8). We divide the dataset into five sets, train the network with four sets, and test it with one set. Note that these processes require significant calculation time; thus, in the optimization process, cross validations are not performed. We perform cross validation for only the optimal solution among the optimal solutions of all methods and calculate the cross-validated accuracy for comparison with results in the literature [26].

Tables 8, 9, and 10 show the details of each network, the fixed hyperparameters, the optimized hyperparameters, and the search ranges. Note that, for this experiment, data augmentation is conducted in a single-crop manner [26].

5.3 Results

The experiments are executed for one month using 32 modern GPUs. The experimental results are given in

Table 6 Fixed parameters of Batch-Normalized Maxout Network in Network

Name	Description
Iteration	20,000
Batch size	100
Learning rate decay policy	Multistep (gamma = 0.1, step value = {15, 000, 18, 000}) [29]

Table 7 Batch-Normalized Maxout Network in Network hyperparameters

Name	Description	Range
x_1	Learning rate ($= 0.1^{x_1}$)	[0.5, 2]
x_2	Momentum ($= 1 - 0.1^{x_2}$)	[0.5, 2]
x_3	L2 weight decay	[0.001, 0.01]
x_4	Dropout 1	[0.4, 0.6]
x_5	Dropout 2	[0.4, 0.6]
x_6	Conv 1 initialization deviation	[0.01, 0.05]
x_7	Conv 2 initialization deviation	[0.01, 0.05]
x_8	Conv 3 initialization deviation	[0.01, 0.05]
x_9	MMLP 1-1 initialization deviation	[0.01, 0.05]
x_{10}	MMLP 1-2 initialization deviation	[0.01, 0.05]
x_{11}	MMLP 2-1 initialization deviation	[0.01, 0.05]
x_{12}	MMLP 2-2 initialization deviation	[0.01, 0.05]
x_{13}	MMLP 3-1 initialization deviation	[0.01, 0.05]
x_{14}	MMLP 3-2 initialization deviation	[0.01, 0.05]

Fig. 8 Faces from the Adience benchmark for age/gender classification [28]

Tables 11, 12, 13, and 14. In all experiments, the Nelder-Mead method achieves both minimal loss and variance. The small variance suggests that the initial values of the method do not significantly affect the results. Furthermore, the accuracy of the cross validation with the best solution found by the Nelder-Mead method in gender classification is 87.20% (±1.328024) and that for age classification is 51.25% (±5.461970). These values are higher than previous state-of-the-art results (86.8% (±1.4) and 50.7% (±5.1)) reported in the literature [26]. The stability and search performance of this method are magnificent.

The coordinate-search method also achieves good results with LeNet and Batch-Normalized Maxout Network in Network. However, the coordinate-search method searches points using each vector of the positive basis; thus, convergence speed is reduced as the number of dimensions increases. This appears to be the reason why the coordinate-search method does not work for the age/gender classification CNN, which has more

Table 8 Network architecture of the age/gender classification CNN [26]

Conv 1	Kernel size: 7, stride: 4, pad: 0
Pool 1 (MAX pooling)	Kernel size: 3, stride: 2, pad: 0
Conv 2	Kernel size: 5, stride: 1, pad: 2
Pool 2 (MAX pooling)	Kernel size: 3, stride: 2, pad: 0
Conv 3	Kernel size: 3, stride: 1, pad: 1
Pool 3 (MAX pooling)	Kernel size: 3, stride: 2, pad: 0
FC 1	Dropout
FC 2	Dropout
FC 3	

Table 9 Fixed parameters of the age/gender classification CNN

Name	Description
Iteration	20,000
Batch size	50
Learning rate decay policy	Step (gamma = 0.1, step size = 10000) [29]

Table 10 Hyperparameters of the age/gender classification CNN

Name	Description	Range
x_1	Learning rate ($= 0.1^{x_1}$)	[1, 4]
x_2	Momentum ($= 1 - 0.1^{x_2}$)	[0.5, 2]
x_3	L2 weight decay	[0.001, 0.01]
x_4	Dropout 1	[0.4, 0.6]
x_5	Dropout 2	[0.4, 0.6]
x_6^*	FC 1 units	[512, 1024]
x_7^*	FC 2 units	[256, 512]
x_8	Conv 1 initialization deviation	[0.01, 0.05]
x_9	Conv 2 initialization deviation	[0.01, 0.05]
x_{10}	Conv 3 initialization deviation	[0.01, 0.05]
x_{11}	FC 1 initialization deviation	[0.001, 0.01]
x_{12}	FC 2 initialization deviation	[0.001, 0.01]
x_{13}	FC 3 initialization deviation	[0.001, 0.01]
x_{14}	Conv 1 bias	[0, 1]
x_{15}	Conv 2 bias	[0, 1]
x_{16}	Conv 3 bias	[0, 1]
x_{17}	FC 1 bias	[0, 1]
x_{18}	FC 2 bias	[0, 1]
x_{19}^*	Normalization 1 localsize ($= 2x_{19} + 3$)	[0, 2]
x_{20}^*	Normalization 2 localsize ($= 2x_{20} + 3$)	[0, 2]
x_{21}	Normalization 1 alpha	[0.0001, 0.0002]
x_{22}	Normalization 2 alpha	[0.0001, 0.0002]
x_{23}	Normalization 1 beta	[0.5, 0.95]
x_{24}	Normalization 2 beta	[0.5, 0.95]

Integer parameters are marked with *

Table 11 MNIST results (LeNet)

Method	Mean loss	Min loss
Random search	0.005411 (±0.001413)	0.002781
Bayesian optimization	0.004217 (±0.002242)	0.000089
CMA-ES	0.000926 (±0.001420)	0.000047
Coordinate-search method	0.000052 (±0.000094)	**0.000002**
Nelder-Mead method	**0.000029 (±0.000029)**	0.000004
Method	Mean accuracy (%)	Accuracy with min loss (%)
Random search	98.98 (±0.08)	99.06
Bayesian optimization	99.07 (±0.02)	99.25
CMA-ES	99.20 (±0.08)	99.30
The coordinate-search method	99.26 (±0.05)	99.35
The Nelder-Mead method	99.24 (±0.04)	99.28

The smallest loss for each experiment is indicated by bold-faced font

Table 13 Gender classification results

Method	Mean loss	Min loss
Random search	0.001732 (±0.000540)	0.000984
Bayesian optimization	0.00183 (±0.000547)	0.001097
CMA-ES	0.001804 (±0.000480)	0.001249
Coordinate-search method	0.002240 (±0.001448)	0.000378
Nelder-Mead method	**0.000395 (±0.000129)**	**0.000245**
Method	Mean accuracy (%)	Accuracy with min loss (%)
Random search	87.93 (±0.24)	88.21
Bayesian optimization	88.07 (±0.27)	87.85
CMA-ES	88.20 (±0.38)	88.55
Coordinate-search method	87.04 (±0.52)	87.72
Nelder-Mead method	88.38 (±0.47)	88.83

The smallest loss for each experiment is indicated by bold-faced font

hyperparameters. Thus, we should use the Nelder-Mead method rather than the coordinate-search method. As demonstrated in the literature [9], CMA-ES is superior to random search because it finds better parameters earlier. Despite using the same hyperparameters for Bayesian optimization, the method works well for age estimation but not for other tasks. This indicates that, for Bayesian optimization, hyperparameters should be adjusted carefully depending on the given task.

The mean loss graphs (Figs. 9, 10, 11 and 12) show that the Nelder-Mead method rapidly finds a good solution and converges faster than the other methods. We anticipate that the objective function for hyperparameter optimization is multimodal, and many local optima that achieve similar results exist. We confirmed this property via additional experiments that optimized the hyperparameters of the gender classification CNN, the network which has the largest search space, using the Nelder-Mead method. The optimized hyperparameter settings after 600 evaluations of each experiment are shown using the parallel coordinates plot in Fig. 13. In the figure, points in the search space are represented as polylines with vertices on parallel axes. The position of the vertex on the ith axis corresponds to the value of the hyperparameter x_i. The polylines exhibiting small losses are shown in dark colors.

Experimental results showed that the Nelder-Mead method converged to different points every time and the objective function was almost multimodal. Different hyperparameters settings achieved similar losses. From Table 13 and Fig. 13, we deduce that many local optima that achieve similar results exist. In such cases,

Table 12 MNIST Results (Batch-Normalized Maxout Network in Network)

Method	Mean loss	Min loss
Random search	0.045438 (±0.002142)	0.042694
Bayesian optimization	0.045636 (±0.001197)	0.044447
CMA-ES	0.045248 (±0.002537)	**0.042250**
Coordinate-search method	0.045131 (±0.001088)	0.043639
Nelder-Mead method	**0.044549 (±0.001079)**	0.043238
Method	Mean accuracy (%)	Accuracy with min loss (%)
Random search	99.56 (±0.02)	99.58
Bayesian optimization	99.47 (±0.05)	99.59
CMA-ES	99.49 (±0.14)	99.59
Coordinate-search method	99.48 (±0.04)	99.53
Nelder-Mead method	99.53 (±0.00)	99.54

The smallest loss for each experiment is indicated by bold-faced font

Table 14 Age classification results

Method	Mean loss	Min loss
Random search	0.035694 (±0.006958)	0.026563
Bayesian optimization	0.024792 (±0.003076)	0.020466
CMA-ES	0.031244 (±0.010834)	0.016952
Coordinate-search method	0.032244 (±0.006109)	0.024637
Nelder-Mead method	**0.015492 (±0.002276)**	**0.013556**
Method	Mean accuracy (%)	Accuracy with min loss (%)
Random search	57.18 (±0.96)	57.90
Bayesian optimization	56.28 (±1.68)	57.19
CMA-ES	57.17 (±0.80)	58.19
Coordinate-search method	55.06 (±2.31)	56.98
Nelder-Mead method	56.72 (±0.50)	57.42

The smallest loss for each experiment is indicated by bold-faced font

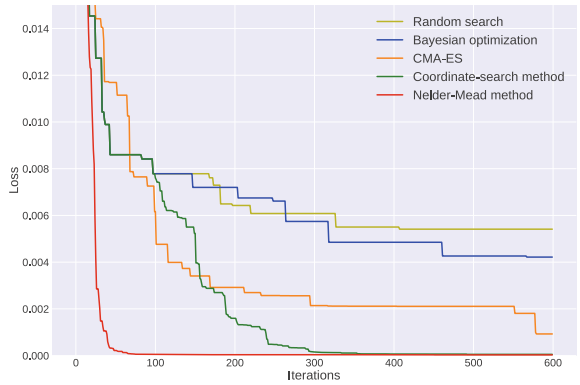

Fig. 9 Mean loss of all executions for each method per iteration (LeNet)

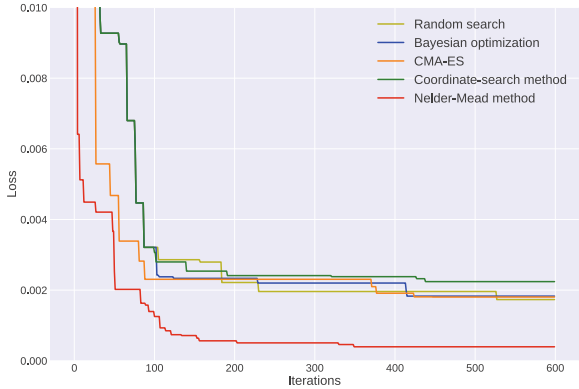

Fig. 11 Mean loss of all executions for each method per iteration (gender classification CNN)

the Nelder-Mead method tends to directly converge to a close local optimum without being influenced by the objective function values of distant points. In contrast, other methods perform a global search, e.g., Bayesian optimization and CMA-ES try to find potential candidates of global optima and require more iterations to find a local optimum in comparison to the Nelder-Mead method.

According to the poor hyperparameter setting detection rates (Tables 15, 16, 17, and 18), on average, approximately 8, 1, 33, and 26% of executions in each experiment are detected as having poor hyperparameter settings and optimization is accelerated in proportion to the detection rate. In particular, the CNN for age/gender classification tends to be very sensitive to hyperparameter settings.

Note that the Nelder-Mead method rarely generates poor hyperparameter settings because of its strategy, e.g., reflection moves the simplex in a direction away from the point of poor hyperparameter settings.

From the above results, we conclude that the Nelder-Mead method is the best choice for DNN hyperparameter optimization.

6 Conclusions

In this study, we tested methods for DNN hyperparameter optimization. We showed that the Nelder-Mead method achieved good results in all experiments. Moreover, we achieved state-of-the-art accuracy with age/gender classification using the Adience DB by optimizing the CNN hyperparameters proposed in [26].

Complicated hyperparameter optimization methods are difficult to implement and have sensitive hyperparameters, which affects their performance. Therefore, it is difficult for non-experts to use these methods. In contrast, the Nelder-Mead method is easy to use and outperforms such complicated methods in many cases.

In our experiments, we optimized the hyperparameters of DNNs for character recognition and age/gender classification. These tasks are important and have been well

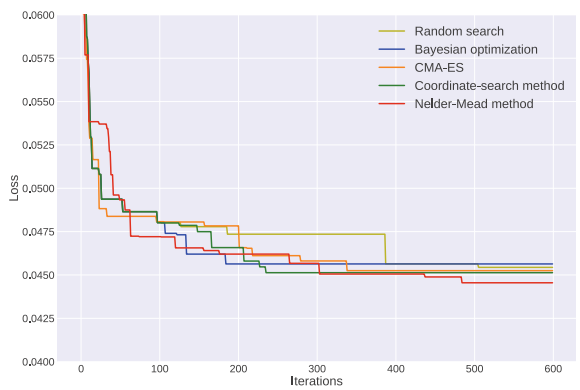

Fig. 10 Mean loss of all executions for each method per iteration (Batch-Normalized Maxout Network in Network)

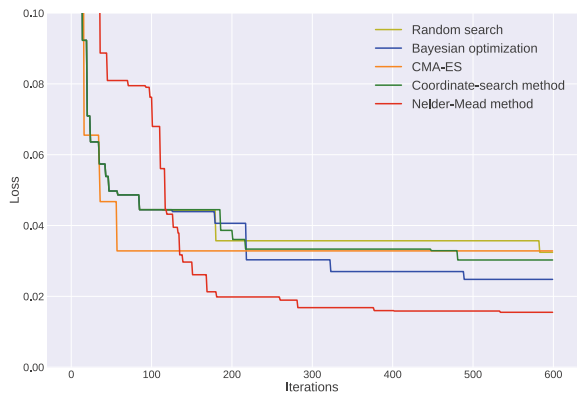

Fig. 12 Mean loss of all executions for each method per iteration (age classification CNN)

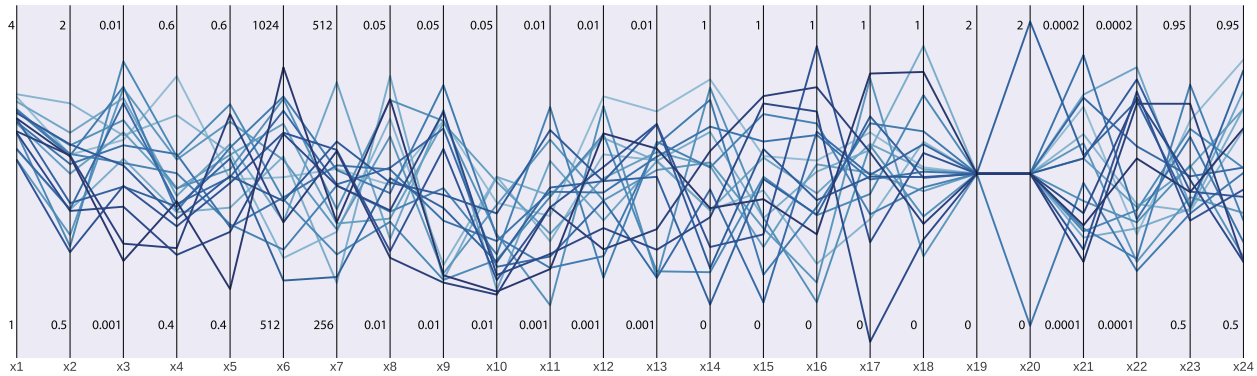

Fig. 13 Parallel coordinates plot of the optimized hyperparameters of the gender classification CNN

Table 15 Poor hyperparameter setting detection rate for each method (LeNet)

Method	Detection rate
Random search	0.144722 (±0.010690)
Bayesian optimization	0.066041 (±0.013354)
Coordinate-search method	0.080674 (±0.041299)
CMA-ES	0.111304 (±0.098500)
Nelder-Mead method	0.003151 (±0.003056)

Table 16 Poor hyperparameter setting detection rate for each method (Batch-Normalized Maxout Network in Network)

Method	Detection rate
Random search	0 (±0)
Bayesian optimization	0.004943 (±0.001991)
Coordinate-search method	0.048941 (±0.054884)
CMA-ES	0 (±0)
Nelder-Mead method	0 (±0)

known for a long time. However, it is desirable to evaluate the proposed method using the generic object recognition data set. Therefore, in future, we plan to evaluate the proposed method using other data sets. A detailed analysis of the dependency on initial parameters and the optimization of categorical variables will be also the focus of future work.

Table 17 Poor hyperparameter setting detection rate for each method (gender classification CNN)

Method	Detection rate
Random search	0.506177 (±0.015931)
Bayesian optimization	0.445244 (±0.005658)
Coordinate-search method	0.274671 (±0.198506)
CMA-ES	0.360734 (±0.091648)
Nelder-Mead method	0.051413 (±0.006113)

Table 18 Poor hyperparameter setting detection rate for each method (age classification CNN)

Method	Detection rate
Random search	0.444667 (±0.039355)
Bayesian optimization	0.355933 (±0.008577)
Coordinate-search method	0.147418 (±0.019866)
CMA-ES	0.317533 (±0.207479)
Nelder-Mead method	0.040334 (±0.004082)

Acknowledgements
This paper is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

Authors' contributions
YO implemented the hyperparameter optimization methods for DNNs, performed the experiments, and drafted the manuscript. MY implemented the hyperparameter optimization methods for DNNs and helped perform the experiments and draft the manuscript. MO guided the work, supervised the experimental design, and helped draft the manuscript. All authors have read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Publisher's Note

Received: 9 January 2017 Accepted: 4 September 2017
Published online: 10 November 2017

References

1. LeCun Y, Bottou L, Bengio Y, Patrick H (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
2. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. Adv Neural Inf Process Syst (NIPS) 25:1097–1105
3. Christian S, Wei L, Yangqing J, Pierre S, Scott R, Dragomir A, Dumitru E, Vincent V, Andrew R (2015) Going deeper with convolutions. Comput Vis Pattern Recognit (CVPR):1–9. http://ieeexplore.ieee.org/document/7298594/

4. Bergstra J, Bengio Y (2012) Random search for hyper-parameter optimization. J Mach Learn Res 13:281–305
5. Mockus J (1974) On Bayesian Methods for Seeking the Extremum. In: Optimization Techniques IFIP Technical Conference. pp 400–404
6. Hansen N, Ostermeier A (2001) Completely derandomized self-adaptation in evolution strategies. Evol Comput 9:159–195
7. Bergstra J, Bardenet R, Bengio Y, Kégl B (2011) Algorithms for hyper-parameter optimization. Adv Neural Inf Process Syst (NIPS) 24:2546–2554
8. Snoek J, Larochelle H, Adams RP (2012) Practical Bayesian optimization of machine learning algorithms. Adv Neural Inf Process Syst (NIPS) 25:2951–2959
9. Watanabe S, Le Roux J (2014) A black box optimization for automatic speech recognition. In: International Conference on Acoustics, Speech, and Signal Processing. (ICASSP). pp 3256–3260
10. Loshchilov I, Hutter F (2016) CMA-ES for hyperparameter optimization of deep neural networks. https://arxiv.org/abs/1604.07269. Accessed 20 Sept 2017
11. Brochu E, Cora VM, De Freitas N (2010) A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. https://arxiv.org/abs/1012.2599. Accessed 20 Sept 2017
12. Snoek J, Rippel O, Swersky K, Kiros R, Satish N, Sundaram N, Patwary M, Prabhat M, Adams R (2015) Scalable Bayesian optimization using deep neural networks. In: International Conference on Machine Learning. (ICML). pp 2171–2180
13. Hansen N, Auger A, Ros R, Finck S, Pošík P (2010) Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009. In: Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation. pp 1689–1696
14. Conn AR, Scheinberg K, Vicente LN (2009) Introduction to derivative-free optimization. MPS-SIAM Ser Optim. http://epubs.siam.org/doi/book/10.1137/1.9780898718768
15. Nelder JA, Mead RA (1965) Simplex method for function minimization. Comput J 7:308–313
16. Gilles C, Patrick R, Mélanie H (2005) Model selection for support vector classifiers via direct simplex search. The Florida Artificial Intelligence Research Society (FLAIRS) Conference:431–435
17. Domhan T, Springenberg JT, Hutter F (2015) Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In: Proceedings of the 24th International Joint Conference on Artificial Intelligence. (IJCAI). pp 3460–3468
18. Klein A, Falkner S, Springenberg JT, Hutter F (2016) Bayesian neural networks for predicting learning curves. Workshop on Bayesian Deep Learning, NIPS. http://bayesiandeeplearning.org/2016/papers/BDL_38.pdf
19. Klein A, Falkner S, Springenberg JT, Hutter F (2017) Learning curve prediction with bayesian neural networks. International Conference on Learning Representations (ICLR). http://www.iclr.cc/doku.php?id=iclr2017:conference_posters#tuesday_morning
20. De Rainville FM, Fortin FA, Gardner MA, Parizeau M, Gagné C (2012) "DEAP: A Python Framework for Evolutionary Algorithms". In: EvoSoft Workshop, Companion proc. of the Genetic and Evolutionary Computation Conference (GECCO). https://dl.acm.org/citation.cfm?id=2330799
21. LeCun Y, Cortes C (2010) MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/. Accessed 20 Sept 2017
22. Yangqing J, Evan S, Jeff D, Sergey K, Jonathan L, Ross G, Sergio G, Trevor D (2014) Caffe: Convolutional architecture for fast feature embedding. https://arxiv.org/abs/1408.5093. Accessed 20 Sept 2017
23. Evan S (2014) Training LeNet on MNIST with Caffe. http://caffe.berkeleyvision.org/gathered/examples/mnist.html. Accessed 20 Sept 2017
24. Nair V, Hinton GE (2010) Rectified linear units improve restricted Boltzmann machines. International Conference on Machine Learning. https://dl.acm.org/citation.cfm?id=3104425
25. Chang JR, Chen YS (2015) Batch-Normalized Maxout Network in Network. In: Proceedings of the 33rd International Conference on Machine Learning. https://arxiv.org/abs/1511.02583. Accessed 20 Sept 2017
26. Gil L, Tal H (2015) Age and gender classification using convolutional neural networks. Computer Vision and Pattern Recognition Workshops (CVPRW). http://ieeexplore.ieee.org/document/7301352/
27. Gil L, Tal H (2015) Age and gender classification using convolutional neural networks. http://www.openu.ac.il/home/hassner/projects/cnn_agegender. Accessed 20 Sept 2017
28. Eran E, Roee E, Tal E (2014) Age and gender estimation of unfiltered faces. IEEE Trans Inf Forensic Secur 9(12):2170–2179
29. Yangjin J (2013) The learning rate decay policy. https://github.com/BVLC/caffe/blob/master/src/caffe/proto/caffe.proto. Accessed 20 Sept 2017