# Warm Starting Method for CMA-ES

**Masahiro Nomura** [*] [†] [‡]
nomura-masahiro@aist.go.jp

**Shuhei Watanabe** [*] [†] [§]
shuhei.watanabe@aist.go.jp

**Yoshihiko Ozaki** [†] [¶]
ozaki-y@aist.go.jp

**Masaki Onishi** [†]
onishi@ni.aist.go.jp

## Abstract

The covariance matrix adaptation evolution strategy (CMA-ES) is one of the most promising black-box optimization methods. However, one issue with the CMA-ES is to require substantial amount of time to fit the internal parameters to sample good solutions even if previous results on similar tasks are available. To alleviate the problem, we propose the warm starting method for the CMA-ES. The method determines the initial internal parameters of the CMA-ES through minimization of KL divergence between the initial multivariate Gaussian distribution of the CMA-ES and Gaussian mixture models of good solutions on a previous similar task. The results show that the proposed method converges faster than the vanilla CMA-ES, Bayesian optimization, and multi-task Bayesian optimization.

## 1 Introduction

Black-box optimization (BBO) is a problem optimizing an objective function that cannot be described using an algebraic representation. BBO methods are widely used to solve the automated machine learning problems such as hyperparameter optimization (HPO) [2] and neural architecture search [1].

The covariance matrix adaptation evolution strategy (CMA-ES) [8, 7] is one of the most powerful methods for BBO, and it has shown the best performance out of over 100 optimization methods for a variety of BBO problems [14]. The CMA-ES facilitates optimization by updating the parameters of the multivariate Gaussian distribution (MGD) according to sampled solutions. There are several application examples using the CMA-ES in a wide range of fields, including applications to the HPO of machine learning algorithms [13, 4, 23]. Loshchilov and Hutter have reported that the CMA-ES achieves better results than Bayesian optimization (BO) in HPO when we can afford a large amount of evaluation budget [13]. However, it is practically difficult to evaluate many solutions due to expensive objective functions. For this reason, the CMA-ES has not been studied widely in the HPO field.

In the real world, practitioners often apply machine learning to similar tasks and tune hyperparameters from scratch even in the case that they have already trained the same machine learning algorithm on similar tasks. To address the problem, there have been several studies to accelerate the optimization by transferring the results on previous tasks [22, 3]. This approach is called “warm starting” and has a great success in HPO with BO. One advantage of the CMA-ES over BO is that it can produce solutions quickly in each iteration even when the amount of data is substantial. Due to this property, the CMA-ES can be benefited from a massive amount of data. However, to our best knowledge, there is no study in the context of warm starting methods for the CMA-ES.

[*] The first two authors contributed equally.
[†] Artificial Intelligence Research Center, AIST
[‡] CyberAgent, Inc.
[§] The University of Tokyo
[¶] GREE, Inc.

In this paper, we propose a warm starting method for the CMA-ES. The method transfers previous knowledge through the minimization of KL divergence between the initial MGD of the CMA-ES and Gaussian mixture models (GMM) of good solutions on previous similar tasks. To evaluate the performance of the CMA-ES with warm starting (CMA-ES-WS), we compare CMA-ES-WS with the vanilla CMA-ES and multi-task BO (MTBO) [22] in HPO problems. The experimental results demonstrate that CMA-ES-WS achieves better performance.

## 2 Backgrounds

### 2.1 Covariance Matrix Adaptation Evolution Strategy (CMA-ES)

The CMA-ES is a BBO method using multivariate Gaussian distribution (MGD) $\mathcal{N}(\mathbf{m}, \sigma^2\mathbf{C})$ where $\mathbf{m} \in \mathbb{R}^d$, $\mathbf{C} \in \mathbb{R}^{d \times d}$ is a positive definite symmetric matrix, $d$ is the dimension of a solution vector, and $\sigma$ is the arbitrary positive number. In each iteration, this algorithm generates $\lambda$ solutions from MGD and evaluates each solution. According to the ranking of the generated solutions, $\mathbf{m}$, $\mathbf{C}$, and $\sigma$ are updated and the CMA-ES learns to sample solutions from a promising region. The CMA-ES tutorial [7, 6] provides more details of the algorithm.

To sample the best solution by the CMA-ES, it is desirable to approximate the inverse Hessian matrix of the objective function by the covariance matrix $\mathbf{C}$; however, it requires an enormous evaluation budget. Loshchilov and Hutter [13] reported that the CMA-ES outperforms BO when an abundant budget is available; however, the CMA-ES does not achieve better results on a small budget.

### 2.2 Warm Starting Methods for Hyperparameter Optimization

Warm starting methods for BO that learn across tasks have been investigated by others before. Multi-task Bayesian optimization (MTBO) [22] extended BO to the multi-task settings. This method measures the correlation between tasks using a Gaussian process and transfers previously optimized tasks by jointly minimizing the average error across multiple tasks. Springenberg et al. [20] applied a Bayesian neural network instead of the Gaussian process to embed the features of tasks. Feurer et al. measures the similarity between tasks using statistical information of dataset and makes sure to evaluate hyperparameter settings that were superior on previous tasks [3]. One issue with MTBO is the elapsed time which increases cubically as the number of evaluations accumulates. To address the problem, Perrone et al. used adaptive Bayesian linear regression whose complexity is linear in terms of the number of evaluations [18].

## 3 Warm Starting Method for CMA-ES

In this section, we propose a warm starting method to determine the initial parameters of the MGD $\mathcal{N}(\mathbf{m}, \sigma^2\mathbf{C})$ used for the CMA-ES by using data optimized on a similar task. As mentioned in the previous section, the performance of the CMA-ES deteriorates when overall budgets that users can afford are small. We alleviate this problem by initializing the distribution using the results on a similar task.

The proposed method transfers the knowledge by three processes. First, the top $\gamma \times 100\%$ solutions are chosen from the set of solutions on a similar task. Second, a Gaussian mixture model (GMM) is constructed by the solutions chosen above as a likelihood belonging to the promising region. Finally, the parameters of the MGD are determined to fit the GMM, and optimized parameters are used as the initial parameters of the CMA-ES. We describe the further details of the two operations below.

### 3.1 Estimating a Promising Region on a Previous Task

In the proposed method, a promising region is estimated using the top $\gamma \times 100$ % solutions. Note that we measure the ranking of solutions based on the evaluation value of solutions. We denote the set of the top $\gamma \times 100$ % solutions on a similar task as $U'$. To approximate the promising region, we construct a probability density function $p(\mathbf{x})$ using GMM. $p(\mathbf{x})$ is formulated as follows:

$$p(\mathbf{x}) = \frac{1}{|U'|} \sum_{\mathbf{x}' \in U'} \mathcal{N}(\mathbf{x} \mid \mathbf{x}', \alpha^2 \mathbf{I}), \tag{1}$$

where $\alpha(> 0)$ is a parameter that controls the extent of a promising region and $\mathbf{I}$ is the identity matrix.

## 3.2 Transferring the Previous Knowledge to the CMA-ES

We showed how to obtain $p(\mathbf{x})$ which measures the likelihood belonging to the promising region in Section 3.1. In this section, we approximate the initial parameters of MGD used for the CMA-ES so that the MGD fits $p(\mathbf{x})$. Specifically, the initial parameter of MGD is determined through minimization of Kullback-Leibler (KL) divergence [11] between $p(\mathbf{x})$ and $q(\mathbf{x}) = \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$.

The gradient of the KL divergence is computed by the following equation and approximated via a Monte Carlo estimation. The derivation of the gradient is shown in Appendix A.

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{\mu}} \mathrm{KL}(p||q) &= -\int \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})p(\mathbf{x})d\mathbf{x} \\ &\approx -\frac{1}{n}\sum_{i=1}^{n} \boldsymbol{\Sigma}^{-1}(\mathbf{x}_i-\boldsymbol{\mu}),\end{aligned} \tag{2}$$

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{\Sigma}} \mathrm{KL}(p||q) &= -\frac{1}{2}\int \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} p(\mathbf{x})d\mathbf{x} + \frac{1}{2}\boldsymbol{\Sigma}^{-1} \\ &\approx -\frac{1}{2n}\sum_{i=1}^{n} \boldsymbol{\Sigma}^{-1}(\mathbf{x}_i-\boldsymbol{\mu})(\mathbf{x}_i-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} + \frac{1}{2}\boldsymbol{\Sigma}^{-1},\end{aligned} \tag{3}$$

where samples $\mathbf{x}_i$ follows the distribution $p(\mathbf{x})$.

As a result of the minimization of KL divergence, we obtain $\boldsymbol{\mu}, \boldsymbol{\Sigma}$. At the initialization step of the CMA-ES, we substitute $\boldsymbol{\mu}$ for $\boldsymbol{m}$, $\boldsymbol{\Sigma}/\sqrt[d]{\det(\boldsymbol{\Sigma})}$ for $\mathbf{C}$ and $\sqrt[2d]{\det(\boldsymbol{\Sigma})}$ for $\sigma$. Note that we choose the initial parameters to be $\det(\mathbf{C}) = 1$ as the vanilla CMA-ES usually choose.

## 4 Results and Discussion

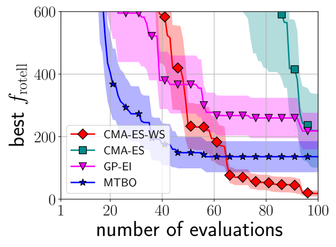
To evaluate the proposed method, we conducted experiments on a benchmark function and the HPO of machine learning algorithms. We compared the proposed method with (1) the vanilla CMA-ES, (2) BO using a Gaussian process as a surrogate model and the expected improvement [9] as an acquisition function (GP-EI) [19], and (3) multi-task BO (MTBO) [22] as the baseline methods. Note that we evaluated 100 input parameter settings as previous data and those input parameter settings were sampled via random sampling to allow MTBO and the proposed method to transfer the same data fairly. Details of the experimental settings are shown in Appendix B.

In the experiment of the optimization of a benchmark function, we optimized the Rotated Ellipsoid function $f_{\mathrm{rotell}}(\mathbf{x}) = f_{\mathrm{ell}}(\mathbf{R}\mathbf{x})$, where $f_{\mathrm{ell}}(\mathbf{x}) = (x_1-3)^2 + 10^6(x_2-3)^2$, and $\mathbf{R}$ is $2 \times 2$ rotation matrix rotating $\pi/6$ around the origin. As previously optimized data, we chose the shifted Rotated Ellipsoid function $f^{\mathrm{shift}}_{\mathrm{rotell}}(\mathbf{x}) = f^{\mathrm{shift}}_{\mathrm{ell}}(\mathbf{R}\mathbf{x})$, where $f^{\mathrm{shift}}_{\mathrm{ell}}(\mathbf{x}) = (x_1-2)^2 + 10^6(x_2-2)^2$. The optimizations were run 30 times.

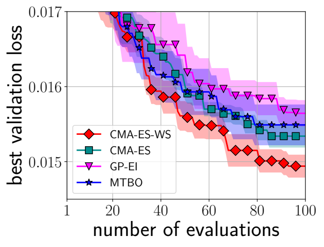
In addition to the benchmark function, we conducted the HPO of multi-layer perceptrons (MLPs). MLPs were trained on MNIST handwritten digits dataset [12] and Fashion-MNIST clothing articles dataset [25]. We optimized the four hyperparameters, (1) learning rate, (2) momentum, (3) the number of hidden nodes and (4) dropout rate. First, we trained MLPs on a subset of each dataset. Note that $2\%$ of all the training data were used as a subset of each dataset. After that, we optimized the MLPs trained on all the data by the proposed method and MTBO accelerated by transferring the previous optimization knowledge. The optimizations were run 10 times for each. Further details of the HPO we carried out are shown in Appendix C.

Finally, we conducted the HPO of convolutional neural networks (CNNs). We applied CNNs to the street view house numbers (SVHN) dataset [16]. SVHN has the same input dimension as CIFAR-10 [10]; therefore, we trained CNNs on a CIFAR-10 to transfer. In addition to CIFAR-10, we used $10\%$ of SVHN dataset to warm start the HPO. The optimizations were run 5 times for each. The details of the HPO we carried out are shown in Appendix C.

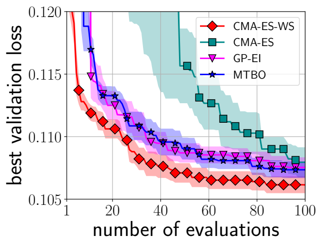
The results of these experiments are shown in Figure 1(a) to 1(e). In Figure 1(a), CMA-ES-WS shows much better optimization performance than the vanilla CMA-ES because CMA-ES-WS can correctly learn the variable dependency and the ill-conditioned nature in advance. BO was advantageous at an early stage as stated in [13]; however, since the CMA-ES-WS takes advantage of previous knowledge, it surpassed the BO at the end. It demonstrates that if there are an abundant evaluation budget or



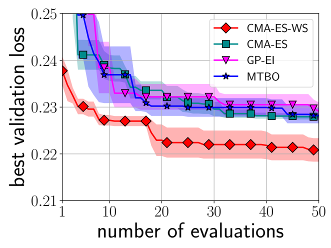
(a) Rotated Ellipsoid function



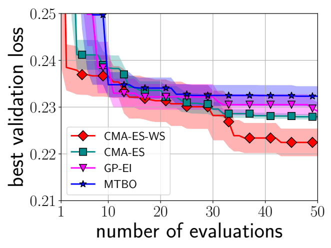
(b) MLP on MNIST



(c) MLP on Fashion-MNIST



(d) CNN on SVHN. Transferring from CIFAR-10



(e) CNN on SVHN. Transferring from a subset of SVHN

Figure 1: Results of optimization. The solid line is the mean and the shadow is the standard error of the mean of the evaluation value.

data optimized on a similar task, the CMA-ES can outstrip the BO. Similarly, in Figure 1(b) to 1(e), CMA-ES-WS showed better performance than the other methods. Furthermore, when using a subset of some dataset, the proposed method is economically efficient. For example, according to Figure 1(c), the proposed method identified better solutions than the vanilla CMA-ES in about $40$ evaluations. To warm start the CMA-ES, we trained $100$ of MLPs on $2\%$ size of training data used by MLPs optimized in the experiment. Therefore, the evaluation time to obtain knowledge on the subset is nearly equal to the time to train $2$ MLPs on all the training data. In other words, the proposed method could achieve a better result with $60\%$ fewer evaluations than the vanilla CMA-ES and it is effective even if we collect the previous data for the sake of transfer learning.

## 5 Conclusion

In this paper, we proposed a warm staring method for the CMA-ES, which transfers data optimized on a similar task. The proposed method estimates a promising region on previous data by a GMM and determines the initial parameters of multivariate Gaussian distribution (MGD) used for the CMA-ES by fitting to the GMM. Due to the proposed method, the CMA-ES can utilize the accumulated observations on similar tasks; therefore, it can identify a better solution even when only a small evaluation budget is available. In the series of experiments, we demonstrated that the proposed method allows the CMA-ES to search efficiently with a small budget. Additionally, since the proposed method embeds previous knowledge in MGD, it can also accelerate other methods using MGD such as the natural evolution strategies [24, 21, 5, 17].

On the other hand, one major problem with the CMA-ES is that it tends to fail to optimize difficult multimodal functions [15]. The proposed method does not address this major issue. Based on the fact that CMA-ES has shown promising results in HPO [13] and our experiments, it might not be a big problem at present in the HPO field. However, it would be important to be able to handle this problem when the architecture of neural networks becomes complex more and more and the relation between hyperparameter settings and the performance is similar to difficult multimodal functions. One solution is to construct a GMM that includes an estimation of the number of clusters. In the case of difficult multimodality, the top solutions can be divided into multiple clusters. Therefore, estimating the GMM including the number of clusters and learning the initial parameters so that the proposed method fits the best cluster can be one solution and we do such a study in the future.

## Acknowledgement

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

## References

[1] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural Architecture Search. In *Automated Machine Learning*, pages 63–77. 2019.

[2] Matthias Feurer and Frank Hutter. Hyperparameter Optimization. In *Automated Machine Learning*, pages 3–33. 2019.

[3] Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. Initializing Bayesian Hyperparameter Optimization via Meta-learning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[4] Frauke Friedrichs and Christian Igel. Evolutionary Tuning of Multiple SVM Parameters. *Neurocomputing*, 64:107–117, 2005.

[5] Tobias Glasmachers, Tom Schaul, Sun Yi, Daan Wierstra, and Jürgen Schmidhuber. Exponential natural evolution strategies. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 393–400, 2010.

[6] Nikolaus Hansen. The CMA Evolution Strategy: A Comparing Review. In *Towards a new evolutionary computation*, pages 75–102. 2006.

[7] Nikolaus Hansen. The CMA Evolution Strategy: A Tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

[8] Nikolaus Hansen and Andreas Ostermeier. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary computation*, 9(2):159–195, 2001.

[9] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global optimization*, 13(4):455–492, 1998.

[10] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. *Master's thesis, University of Tront*, 2009.

[11] Solomon Kullback and Richard A Leibler. On Information and Sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[12] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[13] Ilya Loshchilov and Frank Hutter. CMA-ES for Hyperparameter Optimization of Deep Neural Networks. In *ICLR Workshop*, 2016.

[14] Ilya Loshchilov, Marc Schoenauer, and Michèle Sebag. Bi-population CMA-ES Algorithms with Surrogate Models and Line Searches. In *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation*, pages 1177–1184, 2013.

[15] Monte Lunacek, Darrell Whitley, and Andrew Sutton. The Impact of Global Structure on Search. In *International Conference on Parallel Problem Solving from Nature*, pages 498–507, 2008.

[16] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading Digits in Natural Images with Unsupervised Feature Learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

[17] Yann Ollivier, Ludovic Arnold, Anne Auger, and Nikolaus Hansen. Information-geometric optimization algorithms: A unifying picture via invariance principles. *The Journal of Machine Learning Research*, 18(1):564–628, 2017.

[18] Valerio Perrone, Rodolphe Jenatton, Matthias W Seeger, and Cédric Archambeau. Scalable Hyperparameter Transfer Learning. In *Advances in Neural Information Processing Systems*, pages 6845–6855, 2018.

[19] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian Optimization of Machine Learning Algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[20] Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. Bayesian Optimization with Robust Bayesian Neural Networks. In *Advances in Neural Information Processing Systems*, pages 4134–4142, 2016.

[21] Yi Sun, Daan Wierstra, Tom Schaul, and Juergen Schmidhuber. Efficient natural evolution strategies. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 539–546, 2009.

[22] Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-Task Bayesian Optimization. In *Advances in neural information processing systems*, pages 2004–2012, 2013.

[23] Shinji Watanabe and Jonathan Le Roux. Black box optimization for automatic speech recognition. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3256–3260, 2014.

[24] Daan Wierstra, Tom Schaul, Jan Peters, and Juergen Schmidhuber. Natural Evolution Strategies. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 3381–3387, 2008.

[25] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

## A Derivation of Gradients of KL Divergence

The KL divergence between a probability density function $p(\mathbf{x})$ and $q(\mathbf{x})$ is defined such that:

$$\mathrm{KL}(p||q) = \int p(\mathbf{x}) \ln \frac{p(\mathbf{x})}{q(\mathbf{x})} d\mathbf{x}. \tag{4}$$

Here the KL divergence between a general probability density function $p(\mathbf{x})$ and $q(\mathbf{x}) = \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is calculated as follows:

$$\begin{aligned}\mathrm{KL}(p||q) = &\int \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})p(\mathbf{x})d\mathbf{x} + \frac{1}{2}\log\det(\boldsymbol{\Sigma}) \\ &+ \frac{d}{2}\log 2\pi + \int p(\mathbf{x})\log p(\mathbf{x})d\mathbf{x}.\end{aligned} \tag{5}$$

First, the gradient $\frac{\partial}{\partial \boldsymbol{\mu}}\mathrm{KL}(p||q)$ is calculated.

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{\mu}}\mathrm{KL}(p||q) &= \frac{\partial}{\partial \boldsymbol{\mu}}\int \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})p(\mathbf{x})d\mathbf{x} \\ &= \int \frac{1}{2}\frac{\partial}{\partial \boldsymbol{\mu}}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})p(\mathbf{x})d\mathbf{x} \\ &= -\int \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})p(\mathbf{x})d\mathbf{x}.\end{aligned} \tag{6}$$

Next, the gradient $\frac{\partial}{\partial s_{ij}}\mathrm{KL}(p||q)$ is calculated. Let $s_{ij}$ be the $(i,j)$ element of $\boldsymbol{\Sigma}$, $s'_{ij}$ be the $(i,j)$ element of $\boldsymbol{\Sigma}^{-1}$ and $\mathbf{e}_i$ be the $d$-tuple with all components equal to 0, except the $i$-th, which is 1.

$$\begin{aligned}\frac{\partial}{\partial s_{ij}}\mathrm{KL}(p||q) &= \frac{\partial}{\partial s_{ij}}\left(\int \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})p(\mathbf{x})d\mathbf{x} + \frac{1}{2}\log\det(\boldsymbol{\Sigma})\right) \\ &= -\frac{1}{2}\int (\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}\mathbf{e}_i\mathbf{e}_j^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})p(\mathbf{x})d\mathbf{x} + \frac{1}{2}s'_{ji} \\ &= -\frac{1}{2}\int (\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}))^\top \mathbf{e}_i\mathbf{e}_j^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})p(\mathbf{x})d\mathbf{x} + \frac{1}{2}s'_{ji} \\ &= -\frac{1}{2}\int (\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}))_i(\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}))_j p(\mathbf{x})d\mathbf{x} + \frac{1}{2}s'_{ij} \ \ (\because s'_{ij} = s'_{ji}).\end{aligned} \tag{7}$$

We used the formulas $\frac{\partial \log\det(A)}{\partial a_{ij}} = A^{-1}_{ji}$ and $\frac{\partial A^{-1}}{\partial a_{ij}} = -A^{-1}\mathbf{e}_i\mathbf{e}_j^\top A^{-1}$ to transform line 1 to line 2 and obtain the gradient $\frac{\partial}{\partial \boldsymbol{\Sigma}}\mathrm{KL}(p||q)$ such that:

$$\frac{\partial}{\partial \boldsymbol{\Sigma}}\mathrm{KL}(p||q) = -\frac{1}{2}\int \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}p(\mathbf{x})d\mathbf{x} + \frac{1}{2}\boldsymbol{\Sigma}^{-1}. \tag{8}$$

## B Details of Experimental Setup

We used pycma[6] and GPyOpt[7] libraries to obtain the results for the vanilla CMA-ES and GP-EI.

The CMA-ES has two control parameters, i.e. the sample size $\lambda$ and the initial sampling distribution. Since the recommended value of $\lambda$ suggested by [7] is $4 + \lfloor 3\log d \rfloor$ and the dimension of the input space was $d = 2, 4, 9$ in the experiments, we took the middle and set $\lambda$ to 8. We chose $\mathcal{N}(0.5, 0.2^2\mathbf{I})$ as the initial distribution because we scaled all the variables to $[0, 1]$ and this setting is the same in [13]. The other control parameters of GP-EI and the CMA-ES followed the default settings of each library. In MTBO, the initial random sample size was set to 10. In the proposed method, $\alpha$ in equation (1) and $\gamma$ were set to $0.05$ and $0.1$ for each. Note that $\alpha$ necessarily need not be redefined when the definition of the search space changes because GMM in Section 3.1 is performed in the scaled space $[0, 1]^d$.

[6]https://github.com/CMA-ES/pycma
[7]https://github.com/SheffieldML/GPyOpt

## C Details of Hyperparameter Optimization

Table 1, 2 show the hyperparameters of MLP and CNN and their respective search spaces. We treat integer-valued hyperparameters as a continuous variable and rounded off before evaluations. The MLP consists of two fully-connected layers with softmax at the end. We set the maximum number of epochs during training to 20, and the mini-batch size to 128 in the experiments on MLPs. In the experiments on CNNs, we set the maximum number of epochs during training to 160. The architecture of CNNs is presented in Table 3.

In the experiments on MLPs, the MNIST and Fashion-MNIST were used and we used $50,000$ images as training data and $10,000$ images as validation data. As previous data, we trained MLPs on $1,000$ training images and evaluated the performance on $10,000$ validation data for each dataset. In the experiments on CNNs, SVHN was used and we used $73,257$ images as training data and $26,032$ images as validation data. As previous data, we trained CNNs on a subset of SVHN and CIFAR-10. When using CIFAR-10, we used $50,000$ images as training data and $10,000$ images as validation data. When using a subset of SVHN, we used $7,352$ images as training data and $1,832$ images as validation data.

Table 1: Details of hyperparameters of MLP on MNIST and Fashion-MNIST dataset.

| Hyperparameters | Type | Scale | Range |
|---|---|---|---|
| Learning Rate | float | linear | $[1.0 \times 10^{-3}, 0.2]$ |
| Momentum | float | linear | $[0.8, 0.99]$ |
| # of Hidden Nodes | int | linear | $[50, 500]$ |
| Dropout Rate | float | linear | $[0, 0.8]$ |

Table 2: Details of hyperparameters of CNN on SVHN dataset.

| Hyperparameters | Type | Scale | Range |
|---|---|---|---|
| Batch Size | int | log | $[32, 256]$ |
| Learning Rate | float | log | $[5.0 \times 10^{-3}, 0.5]$ |
| Momentum | float | linear | $[0.8, 1.0]$ |
| Weight Decay | float | log | $[5.0 \times 10^{-6}, 5.0 \times 10^{-2}]$ |
| # of Feature Maps in Conv $C_1, C_2, C_3$ | int | log | $[16, 128]$ |
| # of Feature Maps in FC $C_4$ | int | log | $[16, 128]$ |
| Dropout Rate | float | linear | $[0, 1.0]$ |

Table 3: Details of architecture of CNN on SVHN dataset.

| Components | Output Size | Layer Type |
|---|---|---|
| Convolution 1 | $32 \times 32 \times \mathrm{C}_1$ | $5 \times 5$ conv $\times$ $\mathrm{C}_1$, Padding 2 |
| Max Pooling | $15 \times 15 \times \mathrm{C}_1$ | $3 \times 3$, Stride 2 |
| Convolution 2 | $15 \times 15 \times \mathrm{C}_2$ | $5 \times 5$ conv $\times$ $\mathrm{C}_2$, Padding 2 |
| Average Pooling | $7 \times 7 \times \mathrm{C}_2$ | $3 \times 3$, Stride 2 |
| Convolution 3 | $7 \times 7 \times \mathrm{C}_3$ | $5 \times 5$ conv $\times$ $\mathrm{C}_3$, Padding 2 |
| Average Pooling | $3 \times 3 \times \mathrm{C}_3$ | $3 \times 3$, Stride 2 |
| Fully-connected Layer | $\mathrm{C}_4 \times 1$ | $\mathrm{C}_4$ dimensional fully-connected |
| Classification Layer | $100 \times 1$ | 100 dimensional fully-connected, softmax |