The New England Journal of Statistics in Data Science Volume 1, 334–341 (2023)
DOI: https://doi.org/10.51387/23-NEJSDS26

Evaluating Designs for Hyperparameter Tuning in Deep Neural Networks

Chenlu SHI*, Ashley Kathleen CHIU, and Hongquan XU

Abstract

The performance of a learning technique relies heavily on hyperparameter settings. It calls for hyperparameter tuning for a deep learning technique, which may be too computationally expensive for sophisticated learning techniques. As such, expeditiously exploring the relationship between hyperparameters and the performance of a learning technique controlled by these hyperparameters is desired, and thus it entails the consideration of design strategies to collect informative data efficiently to do so. Various designs can be considered for this purpose. The question as to which design to use then naturally arises. In this paper, we examine the use of different types of designs in efficiently collecting informative data to study the surface of test accuracy, a measure of the performance of a learning technique, over hyperparameters. Under the settings we considered, we find that the strong orthogonal array outperforms all other comparable designs.

keywords and phrases: Big data analysis, Factorial design, Kriging model, Machine learning, MNIST dataset, Space-filling design.

1. INTRODUCTION

In the modern computer age, deep learning has been a powerful tool for big data analysis, which utilizes various sophisticated computer models to learn hidden patterns or intricate relationships among a large number of variables from monumental amounts of complex and challenging data. The performance of a deep learning technique, however, depends heavily on hyperparameters – parameters that control the learning process whose values are predetermined by the user in advance of the process of training the learning model. An inappropriate hyperparameter setting can result in poor performance of the learning process [13]. Not only do possible hyperparameters vary algorithm to algorithm, architecture to architecture, and dataset to dataset, but also those most important hyperparameters can be unique across different domains [3, 43]. These call for investigations on hyperparameter optimization.

Generally, work in the literature on hyperparameter optimization has been discussed under both model-free and model-based frameworks. Model-based hyperparameter optimization targets tuning hyperparameters by finding the best possible approximation to the true learning algorithm, while model-free methods consider this optimization problem without making any parametric assumptions. Relevant work along the line of model-based approaches can be found in [4, 9, 19, 26, 30, 34, 39, 46, 51]. There has been extensive work under the model-free framework – see, for example, manual search, grid search, random search [3] and orthogonal array tuning method [50].

*Corresponding author.

For more sophisticated learning algorithms, however, the above hyperparameter optimization approaches, either model-free or model-based methods, may be too computationally expensive to afford. This draws our attention to expeditiously explore the relationship between hyperparameters and the response containing measures of the performances of models learned by an algorithm with different hyperparameter combinations using a set of data collected through testing a minimal number of hyperparameter combinations. It alludes to a primary goal of design and analysis of experiments – executing efficient experiments to collect informative data for studying the relationship between multiple input variables and an output variable. As such, it is natural to consider design strategies to efficiently collect data for exploring the surface of the performances of a hyperparameter-controlled learning algorithm, which gives rise to the question of which design to use for this purpose, as diverse designs are available in the literature on experimental design.

In this paper, we investigate the above question by comparing different types of designs in terms of their ability to efficiently collect informative data to study how hyperparameters influence the performances of a learning algorithm controlled by them. In order to break the bottleneck of computational limitations, we use a deep neural network model with three hidden layers and apply this comparison to a popular dataset, the MNIST dataset. Five hyperparameters and various factorial and space-filling designs for selecting hyperparameter combinations are considered. We choose the Kriging model to describe the complex relationship between the hyperparameters of a learning algorithm and the test accuracy that measures the performance of such a learning algorithm in the study. The results show that the performance of the 32-run strong orthogonal array surpasses the performances of all other comparable designs.

This paper is organized as follows. In Section 2, we provide preparation for the comparison, reviewing (deep) neural networks, hyperparameters, experimental designs and the Kriging model. Section 3 compares various designs by applying them to the MNIST dataset. We end this paper with some discussion in Section 4.

2. PRELIMINARIES

Suppose that $\mathcal{A}$ is a machine learning algorithm with k hyperparameters. Let Λ_i be the domain of the i-th hyperparameter and $\mathbf{\Lambda} = \Lambda_1 \times \cdots \times \Lambda_k$. We write values of a combination of k hyperparameters into a vector x. For any $x \in \mathbf{\Lambda}$, a learning algorithm based on the hyperparameter setting x is denoted by $\mathcal{A}_x$. Given a set of data D, we denote a measure of the performance of a model learned by algorithm $\mathcal{A}_x$ on data D by y. One commonly used measure of the performance of a learning algorithm model is test accuracy which assesses how accurate the prediction of test data from the model based on the training dataset is compared to the observed value. The higher the test accuracy, the better the performance of the algorithm.

In order to find an appropriate hyperparameter combination, it is desirable to explore the relationship between hyperparameter combination x and test accuracy y. The complexity of the algorithm $\mathcal{A}_x$ results in obtaining the accuracy y being computationally expensive. Therefore, a design strategy is needed to efficiently collect informative data for building a statistical model to specify the relationship between hyperparameter combination x and test accuracy y. Our goal here is to compare different types of experimental designs for doing this job.

Neural Networks and Deep Neural Networks

Neural networks [31], also known as artificial neural networks, are one of the most well-known algorithms used to recognize patterns and solve common problems in statistics, data science and machine learning. A neural network is comprised of an input layer, one or more hidden layers and an output layer, with several units called neurons in each layer. Neurons in the input layer bring the initial data into the network for further processing, while the result from this network is produced by neurons in the output layer. Each neuron in hidden layers connects to another via receiving input produced by neurons in the preceding layer and producing the output to neurons in the next layer. Neural network models, in general, are very flexible, which, on the other hand, brings challenges in terms of constructing the best network structures. See [29] for more details on neural networks.

In this day and age, problems, such as image classification, object detection, or natural language processing task, become increasingly complex. This calls for deep neural networks. A deep neural network refers to an artificial neural network with more than one hidden layer [2, 37]. As an example, consider a deep neural network with three hidden layers, which is shown in Figure 1. The number of neurons in the hidden layers is one of the common hyperparameters of interest. The number of neurons in the input layer is equal to the number of features in the data, while the number of neurons in the output layer depends on the type of problems we solve. For example, very often, there is one neuron in the output layer for a regression problem. If a multi-class classification problem is considered, one common choice is to use one neuron per a class.

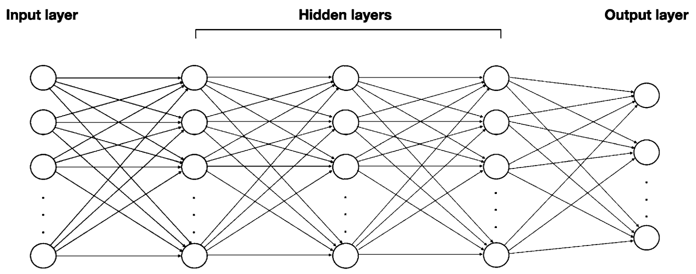

Figure 1: The neural network model with three hidden layers.

Hyperparameters

Hyperparameters for neural networks are those variables whose values determine the network structure and/or the way a network is learned. The number of neurons in the hidden layers of Figure 1 is an example of a hyperparameter. Typically, a neural network requires us to determine values for a set of different hyperparameters including, but not limited to, learning rate, batch size, number of layers, dropout rate, number of epochs, number of training iterations, normalization, pooling, momentum and weight initialization. Those hyperparameter settings drastically affect the success and accuracy of the network. As a consequence, finding an appropriate hyperparameter combination is increasingly important.

Design of Experiments

Design of experiments is a systematic and efficient method that allows scientists and engineers to explore the relationship between multiple input variables and output variables. Traditional physical experiments favor various factorial designs for studying systems or processes [45]. But they are no longer proper when the systems or processes become complex, which calls for computer experiments and space-filling designs, a family of designs appropriate for computer experiments [10, 36].

Factorial Designs At the early stage of an investigation, due to the lack of enough prior knowledge, the experimenter conducts a screening experiment involving as many variables as possible and the primary goal is to identify some important factors from them using a first-order model. Two-level factorial designs [7, 33], thanks to their simple structure and nice statistical properties, are commonly used for this purpose. The experimenter often proceeds to the next stage by capturing the curvature in the response surface, which requires the consideration of three-level factorial designs or composite designs such as orthogonal array composite designs (OACDs) [48].

Space-Filling Designs A space-filling design refers to a design that scatters its design points uniformly over the whole design region. It is tremendously complicated to find a space-filling design that provides good coverage of the entire input space, especially in the high-dimensional input space. Instead, it is natural and reasonable to consider a design that enjoys the space-filling property in low-dimensional projections. This idea dates back to Latin hypercube designs (LHDs) proposed by [32]. Such designs are orthogonal arrays of strength one and enjoy the maximum space-filling property in all univariate projections – there is exactly one observation in any interval formed through dividing the range of an input variable into the same number of equally spaced intervals as the run size. The space-filling property of an LHD may be further evaluated by other optimality criteria such as orthogonality [5], a distance criterion [20] and a discrepancy criterion [11]. There has been extensive work along this line – see, for example, [12, 21, 27].

Strong orthogonal arrays (SOAs) introduced and studied by [17] are the other class of space-filling designs in the literature with a focus on low-dimensional projections. An SOA of strength t achieves the space-filling properties in all $g \leq t$ dimensions. Such an array does well as a comparable orthogonal array of strength t in all t dimensions but is more space-filling in all $1 \leq g \leq t-1$ dimensions than the latter. It can also be made to be an LHD through level expansion to achieve the maximum space-filling property in all one-dimensional projections. As a result, this SOA-based LHD enjoys the benefits of better space-filling properties than the comparable ordinary LHD in all $2 \leq g \leq t$ dimensions. More developments on SOAs can be found in [15, 18, 28, 38, 42].

Kriging Model

The Kriging model originated in the areas of geosciences [23] and is now popular in computer experiments for building a surrogate model for complicated computer models. In a Kriging model, the response is referred to as a realization of a Gaussian process, and the predicted response at a target point can be represented by a weighted average of the responses at observed points. For more details on Kriging models, we refer to [8, 14, 22, 35, 44].

As most learning algorithms are stochastic, instead of using the Universal Kriging model for the deterministic case in computer experiments, we consider the Universal Kriging model with a noise term

$$y(x) = \sum_{i=1}^{m} \beta_i f_i(x) + Z(x) + \epsilon, \tag{2.1}$$

where $\sum_{i=1}^{m} \beta_i f_i(x)$ is the mean function with f_i being the ith basis function and β_i being the ith coefficient, $Z(x)$ is a second-order stationary random process with constant mean 0 and the covariance matrix given by $\mathrm{Cov}[Z(u), Z(v)] = \sigma^2 \prod_{i=1}^{k} R(|u_i - v_i|)$, where $u = (u_1, \ldots, u_k)$ and $v = (v_1, \ldots, v_k)$ are two points (runs), $R(\cdot)$ is a spatial correlation function and σ^2 is the variance of the random process, and $\epsilon \sim N(0, \tau^2)$ is independent of $Z(x)$. In this study, we adopt the Ordinary Kriging model with a noise term, where the mean function in equation (2.1) is a constant. We choose the Matern correlation function with parameter $\nu = 5/2$ in the study based on the conclusion drawn by [47] that the Matern correlation function provides a good balance between differentiability and smoothness. The explicit form is given by

$$R(h_i; \theta_i) = \left(1 + \frac{\sqrt{5} h_i}{\theta_i} + \frac{5 h_i^2}{3 \theta_i^2}\right) \exp\left(-\frac{\sqrt{5} h_i}{\theta_i}\right),$$

where $h_i = |u_i - v_i|$ and θ_i is the unknown hyperparameter which can be estimated using the maximum likelihood estimation method.

3. COMPARISON STUDY BASED ON MNIST DATASET

Due to computational limitations, to compare the performance of various designs, we explore the test accuracy surface on hyperparameters by considering a deep neural network model with three hidden layers, as shown in Figure 1, for a manageable dataset, MNIST.

3.1 Setups

MNIST Dataset The MNIST dataset (Modified National Institute of Standards and Technology database) is a subset of a larger dataset available at NIST, the National Institute of Standards and Technology, and has served as a canonical training dataset for many learning techniques and pattern recognition methods. It is a dataset of handwritten digits, first developed and released by [25], containing 70000 grayscale images of the 10 digits that are 28×28 pixels in width and height. Some examples are demonstrated in Figure 2. These images come with labels and thus are used for supervised learning tasks. MNIST is user-friendly, as images in this set have been split into a training set of 60,000 images, and a test set of 10,000 images. In this paper, we further consider a subset of the training set as a validation set. Each time, we randomly take out 15% of the full training set to serve as a validation set.

Figure 2: Examples from MNIST Handwritten Digit Database.

Hyperparameters We consider five common hyperparameters: learning rate, number of epochs, batch size, dropout rate and number of neurons in a hidden layer. These five hyperparameters are numerical factors: learning rate and dropout rate are continuous factors and number of epochs, batch size and number of neurons are discrete factors. The domain of each hyperparameter is given in Table 1.

Designs Various factorial designs and space-filling designs of 5 columns corresponding to five hyperparameters are examined. More specifically, we use three factorials: the 2^5 and 3^5 full factorials and an OACD of 34 runs combining the 2_V^{5-1} factorial with a three-level 18-run orthogonal array. Two full factorials are generated using the R package DoE.base [16]. The 34-run OACD can be constructed using Table 2 in [48] and is given in the supplementary material. In addition to three factorials, this study includes a 243-run random LHD generated using the R package lhs [6] and several 32-run space-filling designs – an SOA of strength three, its corresponding LHD and four other types of LHDs: a random LHD, a maximin LHD, a maximum projection LHD and a uniform LHD, which are generated using R packages lhs, SLHD [1], MaxPro [21], and UniDOE [49], respectively. There are 32 levels in each column of these LHDs while the SOA of strength three has 8 levels and is listed in the supplementary material. An LHD based on this SOA can be obtained by expanding 8 levels to 32 levels following [17]. The two largest designs, 3^5 factorial and 243-run random LHD, serve as the benchmarks in comparison with designs of small run sizes.

The design matrices of the above designs in terms of natural units are provided in the supplementary material for reference, where the lowest and highest levels are given in Table 1 for each factor. We linearly interpolate other levels within the range. For discrete variables, we further round the levels to the nearest integers.

Data Collection All neural network implementations are built in TensorFlow: https://www.tensorflow.org. There is a stochastic component to the achieved accuracy, as images are shuffled for each model trained. The network based on the training dataset is further assessed on the test dataset of 10,000 images using test accuracy. The test accuracy is appended in the tables of design matrices with natural units in the supplementary material. We also record the cross-entropy loss of each hyperparameter combination in the column right before the test accuracy column in these tables for those interested parties.

3.2 Analysis and Results

We evaluate designs by considering their ability to collect informative data for building a statistical model that specifies the relationship between hyperparameters and test accuracy. In our comparison, the Kriging model is an appropriate consideration, as it can compensate for the effects of data clustering and give a better estimation of prediction error. More specifically, we use the Ordinary Kriging model with a noise term. As test accuracy has values with $(0, 1)$, we consider the Arcsine Transformation for test accuracy when building the model.

Essentially, the model is built using a set of data including hyperparameter combinations selected by a design we consider and their corresponding accuracy, which is then expected to be evaluated on the data comprising as many diverse hyperparameter combinations as possible over the whole hyperparameter region and their accuracy. This, however, is infeasible especially for the continuous hyperparameters. In this paper, we consider two extreme cases – generating test data from the 3^5 factorial and 243-run random LHD. The random LHD enjoys the maximum space-filling property in all one dimensions, while the 3^5 factorial covers the entire 5-dimensional input space in a uniform fashion. The root mean square error (RMSE) values for predicting the mean test accuracy on these two designs are listed in Table 2. Table 2 also includes results on the hybrid design combining the 3^5 factorial with 243-run random LHD.

When tested on the 3^5 factorial, from Table 2, it can be seen that a three-level design, 34-run OACD, outperforms all other small designs. This design performs as well as we expected because this 34-run OACD is a design comprising 34 runs taken from the 3^5 factorial. In other words, a subset of the test data is used to train the Kriging model in this case. Surprisingly, although the SOA is tested on a different type of design, the 3^5 factorial, it dramatically outclasses

Table 1. Domains of hyperparameters.

Learning Rate	Number of Epochs	Batch Size	Dropout Rate	Number of units
[0.0001, 0.01]	[1, 32]	[16, 128]	[0.5, 0.8]	[32, 256]

Table 2. Test RMSE values on 3^5 factorial, 243-run random LHD and their hybrid, respectively, where Maxpro LHD represents Maximum Projection LHD.

Training Designs	**Testing Designs**		
	3^5 Factorial	243-run Random LHD	3^5 Factorial + 243-run Random LHD
2^5 factorial	0.131	0.223	0.183
34-run OACD	0.092	0.112	0.102
3^5 Factorial	0.035	0.090	0.068
32-run SOA	0.118	0.068	0.097
32-run SOA LHD	0.164	0.083	0.130
32-run Maximin LHD	0.175	0.067	0.133
32-run Maxpro LHD	0.148	0.072	0.116
32-run Uniform LHD	0.200	0.076	0.151
32-run Random LHD	0.160	0.097	0.132
243-run Random LHD	0.144	0.027	0.104

all other small designs except the OACD. Notably, both the OACD and SOA have smaller RMSE than the 243-run LHD when tested on the 3^5 factorial. Moreover, the model on the 3^5 factorial produces the smallest test RMSE, because the test RMSE is indeed the training RMSE.

Considering the test data generated from the 243-run random LHD, the results in Table 2 clearly show that all factorials are worse than those space-filling designs, and that, when the Kriging model is built on the 243-run random LHD, the test RMSE is the training RMSE and thus is the smallest. Though the maximin LHD produces as the smallest test RMSEs as the SOA in this case, its performance when tested on the 3^5 factorial is not as well as that of the SOA – the SOA performs exceedingly well compared to the other small space-filling designs.

Both the OACD and SOA perform excessively well when tested on the same type of design, the 3^5 factorial and 243-run random LHD, respectively. But their performances tested on the different types of designs considerably differ. The SOA is still very competitive on the 3^5 factorial, as it produces the smallest test RMSE among all comparable designs except the OACD. On the 243-run random LHD, however, the performance of OACD is the worst except for the performance of the 2^5 factorial. The SOA is a clear winner among all small designs we considered. The numerical results in the last column of Table 2 further support the above conclusion. In other words, for the setting we considered, this SOA allows us to efficiently collect the most informative data for building a Kriging model that specifies the relationship between hyperparameters and test accuracy. The SOA of strength three we use is indeed optimal in some sense – Theorem 4 of [40] implies that this SOA is optimal under the uniform projection criterion. Moreover, it enjoys better space-filling properties in all two dimensions than an ordinary SOA of strength three. Notably, the model built on the 2^5 design produces an extremely large test RMSE value when tested on the hybrid design, which implies that this two-level design is inadequate for building a Kriging model to capture the true surface of test accuracy.

Figure 3 and Figure 4 also sustain that the performance of this SOA is superb. Figure 3 displays the density plots of the observed test accuracy values for all designs. Each density plot in the first two rows of Figure 3 corresponds to a type of design that is used to generate the training data. The density curve of observed test accuracy values for the hybrid design combining the 3^5 factorial and 243-run random LHD is present in the last plot of Figure 3. It is bimodal with a higher peak within $[0.8, 1]$ and a lower peak of no more than 0.4. Clearly, only the SOA correctly captures this feature while all others fail to do so. We also provide the density curves of observed test accuracy values for the 3^5 factorial and 243-run random LHD, respectively, in the last row for reference. Figure 4 gives the histograms of the test errors which are the differences from predictions of test accuracy

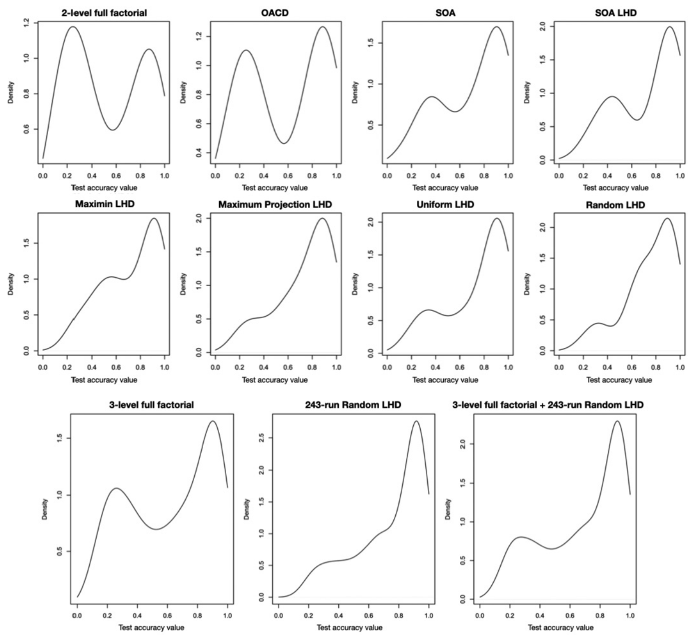

Figure 3: Density plots of test accuracy values.

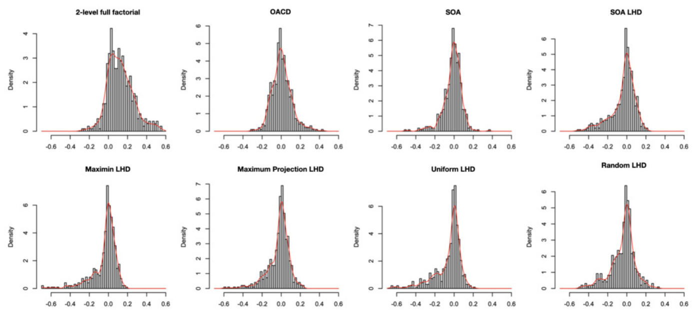

Figure 4: Histograms of test errors, differences between the observed test accuracy and the predicted test accuracy.

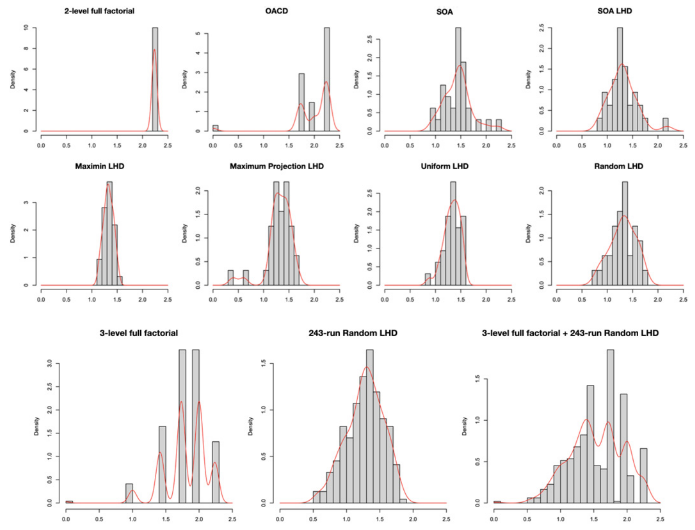

Figure 5: Histograms of distances from design points to design centers.

on the hybrid design to the observed values. The histograms of those LHDs are skewed to the left, which implies that the models based on these designs tend to overpredict the test accuracy on the 3^5 factorial, while models from the 2^5 factorial and OACD tend to underpredict the test accuracy on the 243-run LHD, as their histograms are skewed to the right. Only the SOA is superior because its histogram depicts an approximately normal distribution that is symmetric around 0.

The significant advantage in building a Kriging model of the SOA brings us to take a closer look at these small designs themselves. For all designs, each column is rescaled to $[-1, 1]$, and we then calculate the Euclidean distance from each design point to the center of the design and make a histogram with a density curve for the distances of each design. These plots are given in Figure 5. Histograms of the distances for the 3^5 factorial, 243-run random LHD and their hybrid design provided in the last row of Figure 5 serve as benchmarks for comparison. Only the SOA captures the feature of the hybrid design that the density plot has a light lower tail but a heavy upper tail, while all other designs fail to do so. Remarkably, though a space-filling design is expected to scatter its design points uniformly over the design region, these space-filling designs we considered seem not as space-filling as expected because Figure 5 shows that they do not provide points that are close to the centers or at corners of the regions.

In addition to using the Kriging model to determine the relationship between hyperparameters and test accuracy, we also consider the second-order model. Although the findings from this study are similar to the above, the second-order model is incapable of adequately capturing the test accuracy surface, as the test RMSE values are consistently larger than those obtained using the Kriging model for all cases listed in Table 2.

4. CONCLUSION AND DISCUSSION

This paper compared small designs in exploring the relationship between hyperparameters and test accuracy. We considered five numerical hyperparameters: learning rate, number of epochs, batch size, dropout rate and number of neurons in a hidden layer. Various factorials and space-filling designs for selecting combinations of these hyperparameters were examined. We evaluated the performance of a design via building a Kriging model that describes the relationship between hyperparameters and test accuracy. The comparison is made based on the MNIST dataset, and a deep neural network model with three hidden layers was our learning algorithm. Under the settings we considered, the comparison demonstrates that the 32-run SOA is the best choice for exploring the test accuracy surface based on the hyperparameters we used.

To further explore the usefulness of SOAs in hyperparameter optimization, more investigations are needed. For example, we may set up simulations for evaluating the performance of various designs, similar to the simulation in [42], use other datasets for the implementation, like the CIFAR-10 dataset [24], or consider broader comparison settings such as more designs, e.g., uniform projection designs [41], more hyperparameters, e.g., the number of training iterations, normalization and weight initialization, and so on. The present paper centers on examining the effect of various designs on the performance of hyperparameter tuning, while the primary goal of optimizing hyperparameters is to find an optimal hyperparameter combination that maximizes the overall performance of a learning algorithm. One may wish to consider a model-based method using a carefully selected design in the initial step of the hyperparameter tuning process in the follow-up work. Moreover, in this study, the SOA that outperforms all other comparable designs has 8 levels, while all other space-filling designs have 32 levels and factorials have no more than 3 levels. It would be interesting to consider a problem as to how many levels of a design are appropriate for use in the investigation of the test accuracy surface over hyperparameters. We leave all these to future research.

SUPPLEMENTARY MATERIAL

The supplementary material includes all design matrices in terms of the natural units we used.

Accepted 14 February 2023

REFERENCES

[1] Ba, S., Myers, W. R. and Brenneman, W. A. (2015). Optimal sliced Latin hypercube designs. *Technometrics* **57** 479–487. https://doi.org/10.1080/00401706.2014.957867. MR3425485

[2] Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning* **2** 1–127.

[3] Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research* **13** 281–305. MR2913701

[4] Bergstra, J., Bardenet, R., Bengio, Y. and Kégl, B. (2011). Algorithms for hyper-parameter optimization. *Advances in Neural Information Processing Systems* **24** 2546–2554.

[5] Bingham, D., Sitter, R. R. and Tang, B. (2009). Orthogonal and nearly orthogonal designs for computer experiments. *Biometrika* **96** 51–65. https://doi.org/10.1093/biomet/asn057. MR2482134

[6] Carnell, R. (2022). lhs: Latin hypercube samples. R package version 1.1.5. https://cran.r-project.org/web/packages/lhs/index.html.

[7] Cheng, C. S. (2014) *Theory of factorial design: single- and multi-stratum experiments.* CRC Press.

[8] Cressie, N. (2015) *Statistics for spatial data.* John Wiley & Sons. MR3559472

[9] Falkner, S., Klein, A. and Hutter, F. (2018). BOHB: robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning* **80** 1437–1446. PMLR.

[10] Fang, K. T., Li, R. and Sudjianto, A. (2006) *Design and modeling for computer experiments.* CRC Press. MR2510302

[11] Fang, K. T., Lin, D. K., Winker, P. and Zhang, Y. (2000). Uniform design: theory and application. *Technometrics* **42** 237–248. https://doi.org/10.2307/1271079. MR1801031

[12] Fang, K. T., Liu, M. Q., Qin, H. and Zhou, Y. (2018) *Theory and application of uniform experimental designs.* Springer. https://doi.org/10.1007/978-981-13-2041-5. MR3837569

[13] Feurer, M. and Hutter, F. (2019). Hyperparameter optimization. In *Automated Machine Learning* 3–33 Springer.

[14] Ginsbourger, D., Dupuy, D., Badea, A., Carraro, L. and Roustant, O. (2009). A note on the choice and the estimation of kriging models for the analysis of deterministic computer experiments. *Applied Stochastic Models in Business and Industry* **25** 115–131. https://doi.org/10.1002/asmb.741. MR2510851

[15] Groemping, U. and Carnell, R. (2022). SOAs: creation of stratum orthogonal arrays. R package version 1.3. https://cran.r-project.org/web/packages/SOAs/index.html.

[16] Groemping, U., Amarov, B. and Xu, H. (2022). DoE.base: full factorials, orthogonal arrays and base utilities for DoE packages. R package version 1.2-1. https://cran.r-project.org/web/packages/DoE.base/index.html.

[17] He, Y. and Tang, B. (2013). Strong orthogonal arrays and associated Latin hypercubes for computer experiments. *Biometrika* **100** 254–260. https://doi.org/10.1093/biomet/ass065. MR3034340

[18] He, Y., Cheng, C. -S. and Tang, B. (2018). Strong orthogonal arrays of strength two plus. *The Annals of Statistics* **46** 457–468. https://doi.org/10.1214/17-AOS1555. MR3782373

[19] Hutter, F., Hoos, H. H. and Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In *International Conference on Learning and Intelligent Optimization* 507–523. Springer.

[20] Johnson, M. E., Moore, L. M. and Ylvisaker, D. (1990). Minimax and maximin distance designs. *Journal of Statistical Planning and Inference* **26** 131–148. https://doi.org/10.1016/0378-3758(90)90122-B. MR1079258

[21] Joseph, V. R., Gul, E. and Ba, S. (2015). Maximum projection designs for computer experiments. *Biometrika* **102** 371–380. https://doi.org/10.1093/biomet/asv002. MR3371010

[22] Kleijnen, J. P. (2009). Kriging metamodeling in simulation: a review. *European Journal of Operational Research* **192** 707–716. https://doi.org/10.1016/j.ejor.2007.10.013. MR2457613

[23] Krige, D. G. (1951). A statistical approach to some basic mine valuation problems on the Witwatersrand. *Journal of the Southern African Institute of Mining and Metallurgy* **52** 119–139.

[24] Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical Report, University of Toronto. http://www.cs.utoronto.ca/~kriz/learning-features-2009-TR.pdf.

[25] Lecun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE* **86** 2278–2324.

[26] Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A. and Talwalkar, A. (2017). Hyperband: a novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research* **18** 6765–6816. MR3827073

[27] Lin, C. D., Mukerjee, R. and Tang, B. (2009). Construction of orthogonal and nearly orthogonal Latin hypercubes. *Biometrika* **96** 243–247. https://doi.org/10.1093/biomet/asn064. MR2482150

[28] Liu, H. and Liu, M. Q. (2015). Column-orthogonal strong orthogonal arrays and sliced strong orthogonal arrays. *Statistica Sinica* 1713–1734. MR3409089

[29] Livingstone, D. J. (2008) *Artificial neural networks: methods and applications.* Springer.

[30] Lujan-Moreno, G. A., Howard, P. R., Rojas, O. G. and Montgomery, D. C. (2018). Design of experiments and response surface methodology to tune machine learning hyperparameters, with a random forest case-study. *Expert Systems with Applications* **109** 195–205.

[31] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics* **5** 115–133. https://doi.org/10.1007/bf02478259. MR0010388

[32] McKay, M. D., Beckman, R. J. and Conover, W. J. (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics* **21** 239–245. https://doi.org/10.2307/1268522. MR0533252

[33] Mee, R. (2009) *A comprehensive guide to factorial two-level experimentation.* Springer Science & Business Media.

[34] Mockus, J., Tiesis, V. and Zilinskas, A. (1978). The application of Bayesian methods for seeking the extremum. *Towards Global Optimization* **2** 117–129. MR0471305

[35] Sacks, J., Welch, W. J., Mitchell, T. J. and Wynn, H. P. (1989). Design and analysis of computer experiments. *Statistical Science* **4** 409–423. MR1041765

[36] Santner, T. J., Williams, B. J. and Notz, W. I. (2003) *The design and analysis of computer experiments.* Springer. https://doi.org/10.1007/978-1-4757-3799-8. MR2160708

[37] Schmidhuber, J. (2015). Deep learning in neural networks: an overview. *Neural Networks* **61** 85–117.

[38] Shi, C. and Tang, B. (2020). Construction results for strong orthogonal arrays of strength three. *Bernoulli* **26** 418–431. https://doi.org/10.3150/19-BEJ1130. MR4036039

[39] Snoek, J., Larochelle, H. and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems* **25**.

[40] Sun, C. and Tang, B. (2021). Uniform projection designs and strong orthogonal arrays. *Journal of the American Statistical Association* **0** 1–15. https://doi.org/10.1080/01621459.2021.1935268.

[41] Sun, F., Wang, Y. and Xu, H. (2019). Uniform projection designs. *The Annals of Statistics* **47** 641–661. https://doi.org/10.1214/18-AOS1705. MR3909945

[42] Tian, Y. and Xu, H. (2022). A minimum aberration-type criterion for selecting space-filling designs. *Biometrika* **109** 489–501. https://doi.org/10.1093/biomet/asab021. MR4430970

[43] Van Rijn, J. N. and Hutter, F. (2018). Hyperparameter importance across datasets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* 2367–2376.

[44] Wackernagel, H. (2003) *Multivariate geostatistics: an introduction with applications*. Springer Science & Business Media.

[45] Wu, C. F. J. and Hamada, M. S. (2009) *Experiments: planning, analysis, and optimization*. John Wiley & Sons. MR2583259

[46] Wu, J., Chen, S. and Liu, X. (2020). Efficient hyperparameter optimization through model-based reinforcement learning. *Neurocomputing* **409** 381–393.

[47] Xiao, Q., Wang, L. and Xu, H. (2019). Application of Kriging models for a drug combination experiment on lung cancer. *Statistics in Medicine* **38** 236–246. https://doi.org/10.1002/sim.7971. MR3892817

[48] Xu, H., Jaynes, J. and Ding, X. (2014). Combining two-level and three-level orthogonal arrays for factor screening and response surface exploration. *Statistica Sinica* **24** 269–289. MR3183684

[49] Zhang, A., Li, H., Quan, S. and Yang, Z. (2018). UniDOE: uniform design of experiments. R package version 1.0.2. http://rmirror.lau.edu.lb/web/packages/UniDOE/index.html.

[50] Zhang, X., Chen, X., Yao, L., Ge, C. and Dong, M. (2019). Deep neural network hyperparameter optimization with orthogonal array tuning. In *International Conference on Neural Information Processing* 287–295. Springer.

[51] Zoph, B. and Le, Q. V. (2016). Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.

Chenlu Shi. Department of Statistics, Colorado State University, USA.
E-mail address: chenlu.shi@colostate.edu

Ashley Kathleen Chiu. Department of Statistics, University of California, Los Angeles, USA.
E-mail address: ashleychiu@ucla.edu

Hongquan Xu. Department of Statistics, University of California, Los Angeles, USA.
E-mail address: hqxu@stat.ucla.edu