Title

Exploring the Use of Experimental Design Techniques for Hyperparameter Optimization in Convolutional Neural Networks

Author

Chiu, Ashley

Publication Date

2021

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

Exploring the Use of Experimental Design Techniques for Hyperparameter Optimization in Convolutional Neural Networks

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Statistics

by

Ashley Kathleen Chiu

2021

ABSTRACT OF THE THESIS

Exploring the Use of Experimental Design Techniques for
Hyperparameter Optimization in Convolutional Neural Networks

by

Ashley Kathleen Chiu
Master of Science in Statistics
University of California, Los Angeles, 2021
Professor Hongquan Xu, Chair

Deep learning techniques have become commonplace tools for complex prediction, classification, and recognition tasks. Yet, the performance of such learning techniques is highly influenced by user-set hyperparameters. As a result, efficient hyperparameter tuning and optimization is an increasingly important area of study. Traditional model-free tuning methods are often computationally inefficient and may miss optimal settings, while model-based approaches rely on parametric models and cannot easily be parallelized. In this thesis, we propose the use of experimental design techniques, a Design of Experiments (DOE) Approach, to more efficiently and intuitively optimize hyperparameters. We use fractional factorial designs, nearly orthogonal arrays, sliced Latin hypercube designs, and composite variations of these three small-run designs to identify relationships between continuous, discrete and categorical hyperparameters and test accuracy in convolutional neural networks using beta regression. We find that our proposed methodology successfully identifies optimal hyperparameter settings for convolutional neural networks trained on the MNIST dataset.

The thesis of Ashley Kathleen Chiu is approved.

Jingyi Jessica Li

Ying Nian Wu

Hongquan Xu, Committee Chair

University of California, Los Angeles

2021

To everyone who was a part of this journey,

especially my family...

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Between the COVID-19 pandemic, four quarters of remote learning, and courses that tested the limits of my intelligence, earning this degree has been one of the most challenging experiences of my life. I am immensely grateful to everyone who helped me achieve this goal.

In particular, I would like to extend my deepest gratitude to my thesis committee, consisting of Professors Hongquan Xu, Jingyi Jessica Li, and Ying Nian Wu. Their knowledge, guidance, and enlightening perspectives on their respective areas of expertise made this thesis possible... and made me a better statistician. A special thanks to my Committee Chair, Professor Xu for his patience and consistently helpful answers to my many questions.

Beyond my committee, I must also thank Professors Vivian Lew, Nicolas Christou, Rob Gould, and Mahtash Esfandiari for their encouragement and support dating back to my undergraduate days in the department nearly a decade ago. Other members of the UCLA Statistics community that I have had the pleasure of working alongside and would like to acknowledge include Kekona Sorenson and Dave Zes.

My success in the program would also not be possible without the amazing classmates, turned friends, that I have studied and empathized with over the past two years. This list includes Diana Zhang, Ritvik Kharkar, Ian McGovern, and Adam Rohde—a special thanks to Adam whose collaboration on one of our first projects in graduate school inspired eventual work on this thesis. I'd also like to recognize the UCLA Society of Women in Statistics for their partnership and friendship, especially Melody Huang.

Last, but certainly not least, thank you to my family and close friends for their unwavering love and support throughout this adventure – you know who you are! Special thanks to my younger, but much smarter, brother, Alec, for his championship, advice, IT support, and hugs, and my longtime friend Marina Heskel for proofreading this thesis on her vacation.

Thank you all for believing in me when I did not have the courage to believe in myself!

CHAPTER 1

Introduction

In the modern computer age of big data, deep learning has become an intense area of interest and study. The pace of development in deep learning techniques, including deep neural networks, recurrent neural networks, and convolutional neural networks has evolved so rapidly that these computationally intensive learning techniques are even commonly utilized in industry. Researchers and industry practitioners alike are continuously striving to develop new algorithms and architectures to achieve state of the art performance on increasingly complex and challenging datasets.

The performance of these deep learning algorithms, however, is largely controlled by hyperparameters – parameters that are predetermined by the user prior to training the deep learning model (i.e. parameters that are not learned by the model itself). These hyperparameters, of which there are several that can and should be tuned, effectively control and guide the training the process. Poorly set hyperparameters can drastically alter the performance of the model [FH19]. Further, the possible hyperparameters themselves not only vary algorithm to algorithm, architecture to architecture, and dataset to dataset, but also the set of most important hyperparameters can be unique across these different domains [BB12, RH18].

Traditionally, hyperparameter optimization, which is often considered an exercise in trial and error, has been conducted according to two different frameworks: model-free and model based. Model-free hyperparameter optimization, which includes manual search, grid search, and random search, does not depend on any parametric assumptions. On the other hand, model-based methods, such as Bayesian Optimization, rely on fitting surrogate models for the

desired performance metric in order to identify the optimal settings [BBB11]. Performance, typically measured in terms of test accuracy, evaluates the model's ability to generalize to unseen data. Explicitly, hyperparameter optimization aims to find the set of hyperparameter settings that produces a generalizable model and maximizes test accuracy. In this sense, hyperparameter optimization is equivalent to a black-box optimization problem:

$$x^* = \arg\max_{x \in \mathcal{X}} f(x)$$

where x^* is the optimal set of hyperparameters, $f(x)$ represents test accuracy, and $\mathcal{X}$ is the search space.

In this thesis, we approach the computationally expensive black-box optimization problem from an experimental design or Design of Experiments (DoE) and Response Surface Methodology (RSM) perspective [BW51]. In efficient DoE, we aim to isolate and estimate factor (i.e. hyperparameter) effects, using as few design points as possible; RSM has the overall goal of understanding and identifying the optimal factor settings that will yield the highest response (i.e. test accuracy).

In taking this DoE approach, we propose using a variety of small-run designs and their composites to efficiently and strategically test several combinations of hyperparameter settings. The small-run designs constructed include a 2-level fractional factorial design, nearly orthogonal array (NOA) with mixed levels, and a maximin sliced Latin hypercube design (SLHD). These designs are specifically chosen for their small run sizes, orthogonality and space-filling properties, and their abilities to handle categorical, discrete, and continuous hyperparameters. Further, the NOA and SLHD allow us to estimate higher order quadratic and cubic effects. We apply these designs in the context of optimizing hyperparameters in convolutional neural networks trained and tested on the MNIST dataset. Performance is measured in terms of test accuracy, which is bounded on the interval (0,1)[1]. As a result,

[1]Note that while technically accuracy can take values in the closed interval [0,1], we consider the open interval (0,1), as 0% and 1% accuracy are exceedingly rare and improbable values for experimental data of our type. The experimental data collected does not indicate a need for zero or one-inflated beta regression (refer to Section 5).

we use beta regression for modeling rates and proportions to fit and analyze the response surface [FC04].

Overall, our experimentation and analysis (1) confirms the idea that model performance is largely influenced by hyperparameter settings (2) proposes the use of several existing parallelizable designs and new composite, hybrid designs to efficiently explore the parameter grid-space with less runs than would be required with a traditional grid search and (3) successfully identifies optimal conditions by analyzing experimental data with beta regression.

The remainder of this thesis provides helpful background on RSM and Convolutional Neural Networks (CNNs) in Chapter 2 and briefly discusses related works in Chapter 3. Next, we detail our proposed DoE approach and review the three individual types of experimental designs constructed for this study in Chapter 4. Chapter 5 applies the proposed DoE approach for optimizing a CNN to classify hand drawn digits in the MNIST dataset. Chapter 6 considers the advantages and limitations of our proposed approach. Lastly, Chapter 7 concludes and suggests future directions for research.

CHAPTER 2

Background

Before continuing, we provide some brief background on convolutional neural networks and Response Surface Methodology, the two underlying motivations for our study.

2.1 Convolutional Neural Networks

The use of artificial neural networks (neural networks), a learning technique modeled after the structure and processing abilities of the human brain, has become increasingly popular in statistics, data science and machine learning. This general class of algorithms, in its supervised forms, uses pre-labeled example data (i.e. images) to learn and recognize patterns or identifying characteristics that are consistently correlated with given labels. Subsequently, the developed knowledge is used to properly classify or label new observations (e.g. images).

More specifically, one type of deep neural network, the convolutional neural network (CNN) has proven particularly effective in image recognition [LBB98]. As the name implies, such networks contain at least one convolutional layer, which reduce overall complexity, yielding simpler, more efficient, and easier to train models [KSH12]. Convolutional layers derive their name from the convolution operation in mathematics and can be thought of as a many-to-one type mapping. The use of convolutional layers, thus, allows each layer to consider local information (i.e. dependencies such as neighboring pixels in an image). Explicitly, these models contain fewer connections and parameters because of the addition of these convolution filters.

While several CNN architectures have been developed on the widely used training databases

ImageNet and MNIST, one seminal CNN architecture worth discussing briefly is AlexNet, a deep CNN developed at the ImageNet LSVRC-2012 contest [KSH12]. This particular CNN was found to significantly outperform competitor models in classifying 1.2 million high-resolution images in the ImageNet database into 1000 different classes. Figure 2.1 summarizes the architecture of this novel CNN and shows dimensions of each of the eight layers in the network[1]. In brief, AlexNet consists of five convolutional layers and three fully connected layers. The creators also introduced a variety of unprecedented features to achieve state-of-the-art performance and reduce training time. One such novel feature, that has been widely adopted following its debut in AlexNet, was the use of Rectified Linear Units (ReLUs) as activation functions between network layers.

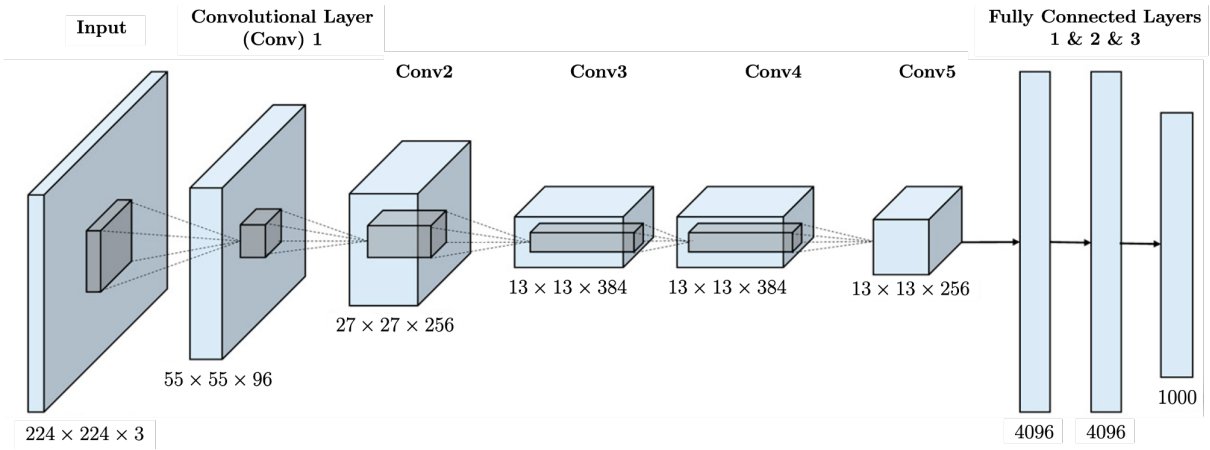

Figure 2.1: Summarized CNN Architecture used in AlexNet

Hyperparameters The activation function or introduction of non-linearity, ReLU, used in AlexNet is an example of a hyperparameter. As mentioned in Chapter 1, in neural networks, hyperparameters refer to pre-set (i.e. before training) variables which determine the network structure and/or the way a network is trained. By this definition, in addition to activation functions, neural networks require the determination and tuning of a number of different hyperparameters including, but not limited to, learning rate, batch size, number

[1]The original AlexNet architecture utilized two GPUs. This is a condensed representation adapted to a single GPU.

of layers, dropout, number of epochs, number of training iterations, normalization, pooling, momentum and weight initialization. Optimizing this large list of hyperparameters is a substantial task due to the sheer volume of possibilities alone. Even more, hyperparameter settings typically work in concert to drastically affect the success and accuracy of the network. As such, researchers employ a number of different techniques, such as random or grid searches, discussed in Chapter 3, to find an optimal combination of hyperparameter settings with the ultimate goal of creating stable, reproducible neural networks that are less prone to overfitting. This search and optimization process, clearly, is what motivates this thesis.

2.2 Response Surface Methodology

Originally introduced by Box and Wilson in 1951 [BW51], Response Surface Methodology (RSM) refers to a class of sequential experimentation and analysis techniques that can be used to model and analyze a response that is influenced by several variables or factors. Through these sequential experiments, the practitioner is able to identify relationships between the various factors and response, continuously improving with each experiment. The response can then be modeled as a response surface characterized by either a first-order (2.1) or second-order model (2.2), typically fit by ordinary least squares (OLS). The overall objective of RSM is to identify the optimal settings for a response surface (e.g. yield, accuracy). These techniques have been heavily used for process and product improvement, particularly in the fields of industrial engineering and chemistry [Mon13]. Moreover, we can see that the ideas can be adapted/extended to the process of hyperparameter optimization.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k + \epsilon \tag{2.1}$$

$$y = \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \sum_{i=1}^{k} \beta_{ii} x_i^2 + \sum \sum_{i<j} \beta_{ij} x_i x_j + \epsilon \tag{2.2}$$

More specifically, traditionally RSM is conducted in three stages. The first stage consists of conducting an experiment for factor screening, which helps identify the most important

factors from a larger pool of potential factors. This is typically done using a small-run two-level complete factorial or fractional factorial design. The second stage consists of a series of sequential experiments that helps determine the optimum region, as depicted in Figure 2.2. Once the optimum region is identified in the second stage, a polynomial model is fit in the region during the final stage. The final stage often uses some second-order response surface designs, such as a Central Composite Design (CCD) [BW51] or Box-Benkhen Design (BBD) [BB60], which allow for the estimation of quadratic effects.

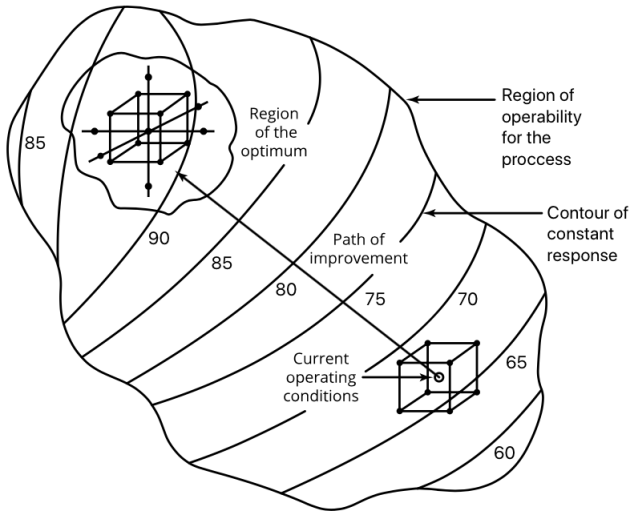

Figure 2.2: The sequential nature of RSM as depicted in [Mon13]

We can see that one pitfall of traditional RSM is the need for sequential experiments in the second stage. Further, separate experiments are often used for each of the three stages. In the context of hyperparameter optimization, this becomes incredibly inefficient as each model takes a non-trivial amount of time to train. Cheng and Wu recognized this disadvantage and proposed methodology that allows factor screening and response surface exploration (first and final stage) simultaneously [CW01] [XPW09]. The two stages of their analysis strategy are (1) to perform factor screening and identify important factors and (2) to fit a second-order model for the factors identified in Stage 1. This effectively bypasses and eliminates the need for sequential experimentation (Stage 2 in traditional RSM). RSM can be made even more efficient by utilizing smaller response surface designs proposed in [DL90]

or an Orthogonal Array Composite Design proposed by Xu et al. [XJD14]. We discuss the use of OACDs in our DoE approach to hyperparameter optimization in this thesis (refer to Chapters 4 and 5).

Lastly, we mentioned that the parameters of the first and second-order models in RSM are estimated through OLS [BW51] [Mon13]. However, in the case of hyperparameter optimization, our response is test accuracy, which is the proportion of correctly classified images divided by the total number of images. This proportion is bounded between (0,1). As a result, we must use alternate analysis techniques (i.e. beta regression [FC04]) to model the bounded response.

The adjustments we make to traditional RSM and the designs utilized for our DoE approach are discussed further in Chapter 4.

CHAPTER 3

Related Work

In this chapter, we briefly review traditionally used methods of hyperparameter optimization, present the reader with recently proposed examples of (experimental) design-based approaches, and consider areas for improvement.

As discussed in Chapter 1, hyperparameter optimization generally falls under two categories: model-free and model-based [FH19]. Model-free methods include manual search, grid search and random search, as well as a handful of very recently proposed design-based approaches. Our method takes inspiration from and expands upon these novel design-based approaches. While parallelizable, these methods are often computationally expensive and exhaustive. We also find that the existing design-based approaches lack necessary flexibility in assessing categorical, discrete, and continuous hyperparameters.

Model-based methods, which have gained popularity, use parametric surrogate models to search the parameter grid space. Hyperparameter combinations (design points) are typically chosen sequentially based on prior knowledge acquired in the search. Thus, these methods are not usually suitable for parallel processing. Bayesian Optimization/Sequential Model-Based Optimization, discussed below in Section 3.2, is one of the earliest and broadest forms of model-based hyperparameter optimization. Model-based methods have become increasingly complex, with some researchers developing online hyperparameter optimization methods [ISC21] or by viewing hyperparameter optimization as a reinforcement-learning problem in itself [WCL20]. These advanced developments are discussed herein.

3.1 Model-Free Methods

Manual Search: Manual search is the most primitive type of hyperparameter optimization. As the name implies, it involves manually adjusting hyperparameters in order to identify the optimal combination of settings. This is usually done using domain knowledge or prior expertise. It is easy to see that this method is not necessarily efficient and is highly dependent on the practitioner's skill level. There is no guarantee that manual search can identify optimal settings, as it is largely an exercise in trial and error. In addition, it may not be possible to reproduce results [BB12] – this is an important aspect of claiming state-of-the-art performance. One advantage to manual search, though, is that the practitioner intuitively gains an understanding of the response surface (i.e. how each hyperparameter affects performance) through these repeated trials.

Grid Search: Perhaps the most commonly used model-free method, grid search naively explores all possible combinations of hyperparameter settings. The combination that yields the best (average) performance is then selected as the optimal hyperparameter combination.

We can see that from a design perspective, a grid search is equivalent to running a complete factorial experiment. For example, if we wish to test 6 hyperparameters at 3 levels each, we can view this as a factorial experiment with $3^6 = 729$ runs. As the number of hyperparameters and levels for each of those hyperparameter grows, the number of runs increases exponentially. Evidently, while this search is exhaustive, it can be incredibly inefficient as not all hyperparameters will have a significant effect on performance. It is also highly possible to miss optimal settings if the range of each experimented hyperparameter is too large and/or the intervals tested are too widely spaced. Some advantages to this method, though, are that each combination is independent and uninformed, and thus, can be computed in parallel. It also requires limited input and analysis from the experimenter.

Random Search: Random search, developed by Bergstra and Bengio in 2012 [BB12], suggests a more efficient alternative to the completely exhaustive grid search. In random

search, design points are independently and randomly drawn from a uniform distribution covering the same space that would be spanned by a regular grid search. The design points being independent and identically distributed, thus allows practitioners to add new runs and increase the granularity of the experiment, which cannot occur in a grid search. Further, random search addresses the idea that not all hyperparameters are as sensitive or important for the response surface (i.e. the response surface has low effective dimensionality): randomly selecting points more evenly explores not only the total multi-dimensional grid space, but also each individual subspace/dimension. This idea is illustrated for the 2-d case in Figure 3.1, which clearly shows that there are less areas in the grid that are completely unexplored under random search.

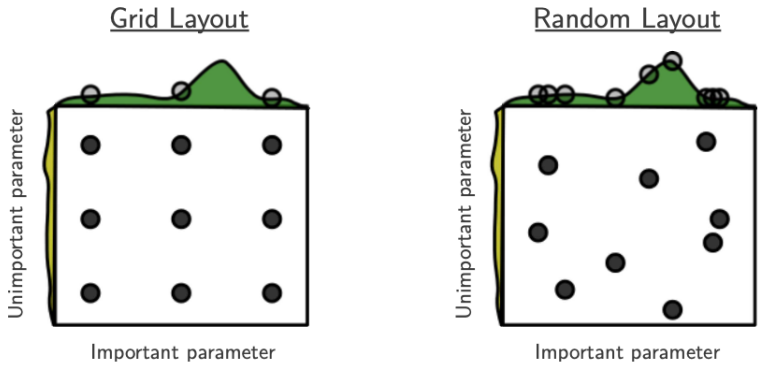

Figure 3.1: An example of the advantages of Random Search vs. Grid Search from [BB12]. The budget size $n = 9$ is the same in both layouts.

Specifically, in Figure 3.1, the random layout on the right (1) more effectively covers the 2-d projection space and (2) as a result, is able to provide more useful information for the labeled important hyperparameter. The grid layout, on the left, leaves portions of the grid-space completely unexplored, as all design points are evenly spaced at the vertices of the grid-space; we miss a large region/potential granularity for the important hyperparameter under this method. Thus, random search has the same efficiency in the relevant subspace as if it had been used to search only the relevant dimensions. Effectively, there are less wasted runs, which is one of the salient issues with grid search. Empirically, [BB12] finds that neural

networks tuned via random search can achieve equal or superior performance to those tuned by traditional grid search, but do so with less computational effort and time.

Other advantages of random search are that it suitable for parallel processing. Additionally, it is budget-flexible, as we can fix the number of design points n to be within the confines of our computational limitations. However, like grid search, it again carries no guarantee of identifying the global optimum.

3.2 Model-Based Methods

Clearly, model-free methods like grid search and random search can suffer from wasted runs – each combination of hyperparameters is selected naively, without incorporating any prior information, so it is possible to spend a significant amount of time evaluating poor hyperparameter combinations. One advantage of manual search is that a practitioner is able to use prior information to adjust the hyperparameter settings, ideally continuing to improve with each experiment run. This idea of using prior knowledge to efficiently explore the parameter space in a smarter manner is highlighted in model-based hyperparameter, specifically sequential model-based hyperparameter optimization (SMBO) [HHL11].

There are several recent SMBO methods in use, which rely on Bayesian Optimization (BO) [MTZ78, SLA12], a sequential design strategy for global optimization of computationally expensive black-box functions (i.e. hyperparameter response surfaces) [FH19]. SMBO/BO methods use an iterative algorithm to (1) improve with each trial by learning a cheaper, probabilistic surrogate model $\mathcal{M}$ of the true response surface modeled by $f(x)$ and (2) balance the trade-off between exploration and exploitation when selecting additional design points to evaluate in the parameter space. The posterior surrogate model $\mathcal{M}_{t+1}$ is updated with each carefully selected, additional design point. Gaussian process (GPs) are common choices for the surrogate function, but others like the Tree Parzen Estimator have been studied [BBB11]. Once the algorithm has identified a set of good or best hyperparameters on the surrogate function, this set of hyperparameters is tried on the true model, and

the process of updating the surrogate model can be continued until there is no additional improvement. Overall, BO methods have been shown to find better hyperparameters settings than random search in fewer iterations. However, they provide limited interpretability.

We leave the discussion of SMBO/BO methods brief, as the design-based approach we will be proposing is primarily model-free and not sequential in nature. Nevertheless, a brief introduction to the topic is helpful in thinking about future directions for our proposed DoE approach.

3.3 Other Model-Free Design-Based Approaches

The traditional approaches to hyperparameter optimization allude to the core strategy of experimental design: executing efficient experiments under a strict computational budget, using a minimal number of runs, that will provide enough information to analyze the hyperparameter effects and improve yield. As such, Design of Experiments (DoE) approaches have recently been attempted. Some examples include using design of experiments and response surface methodology to tune hyperparameters in random forests [LHR18], using Orthogonal (Taguchi) Arrays to tune artificial neural networks using ANOVA techniques [KA16], and Orthogonal Array Tuning Method (OATM) for recurrent neural networks and convolutional neural networks [ZCY19]. Most recently, Modular Factorial Design for Hyperparameter Optimization (MOFA) was proposed, which uses Orthogonal Latin Hypercube Designs and Orthogonal Arrays to sequentially iterate towards the optimal hyperparameter settings.

While these proposed methods incorporate the idea of efficient runs, they implement the designs without robust modeling, inference, or error estimation. They also only test a handful of hyperparameters in their implementation – in most cases only 3 continuous hyperparameters. A majority of these methods also do not consider any interaction effects between hyperparameters. Further, their methods were not flexible enough to handle a mixture of categorical and quantitative hyperparameters. These are issues we aim to address in our proposed approach, detailed in Chapter 4.

3.4 Considerations

Before proposing our approach, we take a moment to summarize some of the key points in these related works that are helpful in developing our own DoE approach.

First, even though manual search is the most dependent on a given individual's expertise, it allows the practitioner to gain some degree of insight and intuition into the response surface and how each hyperparameter affects the response. This is an important idea to incorporate into our proposed method, especially as researchers continue to develop more advanced learning algorithms. While some intuition can be gained in the recently proposed design-based methods like OATM and MOFA, these methods both only consider main effects and ignore the potential interactions between hyperparameters. The interactions may not have a material effect on response, but it is helpful to at least assess that relationship, for example, with some regression model.

Second, new methods should incorporate some degree of adjustability: the ability to adjust the experiment or increase granularity at certain points throughout the process, based on the insight gained on the relationships between the performance response surface and each factor. This was one of the advantages of random search. Closely related, these subsequent additional design points should explore the grid space thoroughly and as exhaustively as possible within the confines of some computational budget. Additionally, easily parallelizable methods that are simple to implement and adjust are preferred.

Third, to improve upon existing design-based approaches, a new approach should be able to incorporate a mixture of categorical, discrete, and continuous hyperparameters.

Lastly, as mentioned in [BB12], in certain hyperparameter optimization schemes, it is difficult to reproduce or verify results. Improved DoE-based methods can incorporate some degree of cross validation through composite designs. In this context, cross validation is the ability to fit models to different parts of the data and compare the resulting estimates across such models. Consistent estimates indicate high data quality and allude to the overall reproducibility of experimental data and analysis results [XJD14].

CHAPTER 4

A New Proposed DoE Approach

To address the considerations set forth in Section 3.4, we approach hyperparameter optimization as an optimization problem that can be solved through efficient DoE and analysis. We propose a DoE-based method that uses a series of small-run increasingly "granular" designs to estimate the response surface that is accuracy. That is, with each new design, we increase the number of levels tested to continue to explore the hyperparameter grid space and allow for the estimation of higher order effects impacting the response surface.

We start with a 2-level fractional factorial design, which can be considered a screening experiment. Next, we add either an orthogonal array or a nearly orthogonal array with three, or possibly with mixed, levels. Lastly, we construct a maximin sliced Latin hypercube design. These small-run designs are strategically chosen not only for their efficient run sizes, orthogonality, and space-filling properties, but also for their flexibility in handling different types of hyperparameters.

In the analysis stage, these three sub-designs can be combined together to create a series of composite designs. The idea to use composite designs stems from previous research in Response Surface Methodology, wherein these types of designs can achieve the benefits of RSM, but with fewer sequential experiments. The response surface here is modeled through beta regression, as the recorded metric is test accuracy (0,1). Since effects are estimated individually for each design, we can also cross validate our results across experiments.

In the final stage, we use a proposed hybrid composite design, consisting of all evaluated design points, to fit a beta regression model. This model indicates which settings are optimal.

Each component of this approach is discussed in more detail in the following subsections.

In Chapter 5 we successfully apply this series of experiments and analysis to the MNIST dataset. Overall, this proposed methodology addresses many of the considerations previously deliberated in Section 3.4.

4.1 Two-Level (Fractional) Factorial Designs

Two-level factorial designs are commonly used in the preliminary stages of experimentation. In a 2^k complete factorial design, k factors are tested at two levels (low and high), and each combination of levels is tested once (2^k total runs). It is easy to see that this is equivalent to a grid search where all hyperparameters can take on exactly two possible values. In these two-level designs , as k grows, the number of runs grows exponentially. An easy alternative is a 2^{k-p}_r two-level fractional factorial design, in which k factors are tested at two levels each, but now with only 2^{k-p} runs. In such a design, the number of runs are reduced by a fraction of $1/2^p$ and p, consequently, is the number of independent generators for the design. The subscript, r indicates the design's resolution. The resolution r controls the amount of aliasing (i.e. confounding) that exists between these effects: if factors are aliased, their effects are indistinguishable from each other. More generally, for a design with resolution r, a p-factor effect will only be aliased other factors that contain at least $r-p$ factors. Two-level factorial designs have a relatively simple aliasing structure.

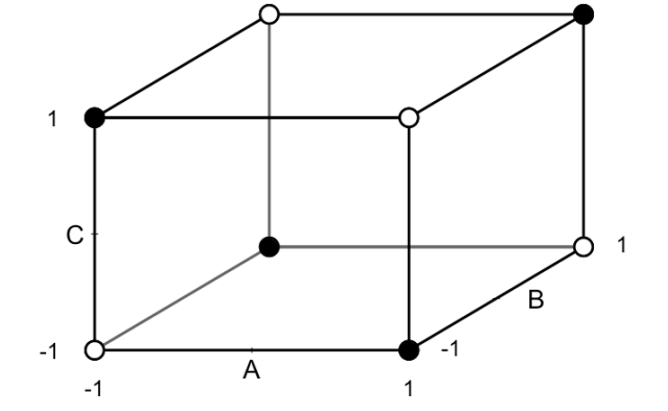

Figure 4.1: Example of a 2^{3-1}_{III} design with $n = 4$ runs

A toy example of a 2^{3-1}_{III} design with factors A, B and C is shown at Figure 4.1, where $\bullet$ represent the four evaluated design points/runs in the fractional factorial experiment. This design has, $k = 3$ factors at two levels each, -1 and 1, and $p = 1$ design generators, $C = AB$. In resolution $r = III$ designs, no main effects are aliased with any other main effects, but main effects are aliased with other bilinear (two-factor) interaction effects. Additionally, two-factor interaction effects may be aliased with each other.

In fractional factorial designs, there are clearly insufficient degrees of freedom to estimate all possible effects (main effects, two and three-way interactions). Moreover, there is an inverse relationship between the resolution r and the amount of the aliasing. As a result, we favor fractional factorial designs with a higher resolution. Specifically, we suggest using a 2^{k-p}_{IV} minimum aberration design. In an $r = IV$ design, all main effects are estimable and not aliased with any two-way interactions (i.e. bilinear effects). Some two-way interactions will be aliased with each other, but aliasing is minimized by using a minimum aberration design [Mon13]. Even so, this type of design can help identify whether two-factor interactions affect the response surface at all. In the context of hyperparameter optimization, it has been shown that interaction effects are likely significant, as there can be trade-offs between, for example, features introduced to combat overfitting (e.g. batch normalization, dropout, architecture complexity) [MSM16, RH18].

Additionally, 2^k and 2^{k-p}_r designs often possess the property of orthogonality. This is an important property that allows estimates to be independent of each other. Explicitly, there is no covariance/correlation between effect estimates, which means main effect and interaction can be estimated independently.

We specifically choose the two-level factorial design as our initial design because of their flexibility and emphasis on three important characteristics, as outlined in [Mon13]. In terms of flexibility, these designs (1) can handle quantitative and qualitative (categorical) factors and (2) have adaptable run sizes, as the practitioner can opt between a complete or fractional design. When fractional designs are selected, they can be balanced with resolution, based on the practitioner's domain knowledge of whether interaction effects are suspected to be

present. Further, the low/high levels can easily be adjusted in a follow-up experiment, if necessary.

The three important characteristics emphasized by this design are the sparsity of effects principle, projection property, and use in sequential experimentation [Mon13]. Briefly, the sparsity of effects principle suggests that the response surface/system is dominated by main effects over lower-order interactions. Second, the projection property of these designs allows us to harness additional strength from the design, should certain factors be unimportant. Explicitly, a fractional factorial design can become a complete factorial design when projected to lower dimensions. This is shown at the toy example provided at Figure 4.1, where the given design is a complete 2^2 factorial design for any $\binom{3}{2}$ factors. Third, we can easily add additional runs or designs to help de-alias effects of interest – this is reflected subsequently in our approach. It should be noted that the projection property is also discussed in the context of random search (see Section 3.1) [BB12].

4.2 Nearly Orthogonal Array Designs

Two-level designs serve as good initial experiments, but do not have the ability to estimate higher-order (e.g. quadratic) effects, which is often necessary and important for modeling a response surface. A common way to augment a two-level design is by constructing a Central Composite Design (CCD) [Mon13, BW51]; however, CCDs are not ideal for situations where there are categorical variables (i.e. situation with possible mixed levels). Instead, we can consider adding on an Orthogonal (Taguchi) Array [KLF91].

An orthogonal array, $OA(N, s_1 \times s_2 \times \ldots \times s_k, t)$ is an $N \times k$ matrix where the i^{th} column contains s_i levels. Further, in any $N \times t$ sub-matrix all combinations of the levels appear equally often as row vectors. Namely, t refers to the strength of the orthogonal array. As is common in DoE, we discuss OA's with $t = 2$. Under this construction, it can be seen that any of the $i = 1, \ldots, k$ factors can have distinct s_i levels. As such, it is possible to have an OA with mixed levels. We can take advantage of this in our setting hyperparameter

optimization in order to test categorical factors that may have at most two levels. Referring back to Section 4.1, OA's, evidently, possess the property of orthogonality, which would allow us to estimate effects independently. Specifically, an OA of strength $t = 2m$ can be used estimate all the main effects and interaction contrasts involving up to m factors independently (i.e. estimates are uncorrelated).

There are many pre-studied, standard OA's readily available for use [Gro18]. However, depending on the combinations of factors and factor levels, the required OA may have too large a number of runs. Instead, we can loosen the orthogonality condition and reduce run-size by utilizing a nearly orthogonal array (NOA) as our second design [WW92]. We define a nearly orthogonal array $OA'(s_1 \times s_2 \times \ldots \times s_k)$[1] as array where the orthogonality requirement is nearly satisfied. The definition for nearly is ill-defined [WW92] and varies amongst NOAs. Depending on the relationships of interest, NOAs can be constructed to meet orthogonality conditions for certain sets of factors/estimates. For example, orthogonality between two-level factors may be more important than orthogonality between two-level factor and three-level factor. [WW92] and [Xu02] provide algorithms for constructing such designs. Explicitly, for this DoE approach, all factors that can be expanded to take a third level (between the low and high settings set forth in the two-level design) are expanded as such. All (categorical) factors where the number of levels cannot be increased remain at two levels, thus creating a mixed-level design. The addition of a third level for applicable factors allows us to estimate quadratic effects.

While the aliasing structure of NOAs can be complex, as interactions can be partially aliased with several main effects, (1) the smaller run-size and computationally efficiency makes it a well worth compromise and (2) this design is not only to be added to an existing two-level (fractional) factorial design, but also augmented by the addition of a sliced Latin hypercube design. Further, the NOA can handle factors of mixed levels and allows us to continue incorporating categorical factors.

[1]All OAs and NOAs discussed henceforth are of strength $t = 2$, the case where all main effects are orthogonal

4.3 Sliced Latin Hypercube Designs

The last individual design to construct is a sliced Latin hypercube design. Before discussing the sliced variant of the Latin hypercube, we briefly review the traditional Latin hypercube design.

Traditional Latin hypercube designs (LHD) [MBC79, Ste87] are often used in computer experiments because of their space-filling properties and ability to capture higher-order effects. We define an LHD to be an $n \times k$ matrix, where each column consists of n equally-spaced levels. Explicitly, for a given factor, we divide a range $[a, b]$ into n equally spaced levels. In our DoE approach, a and b would correspond to the low and high settings from the two-level design discussed in Section 4.1. The n levels within each column are then independently, randomly permuted to yield the final design matrix. By this construction, we can see that the number of runs is equal to to the number of levels, n. Further each level for a given factor is tested only once (no replication). We can see, however, that these designs can well spread points over the design region, one of the desirable considerations outlined in Section 3.4. When the n design points are projected onto any one dimension, exactly one point is in each of the n intervals for that dimension (i.e. factor). It is this space-filling property that makes Latin hypercube designs popular for computer experiments, where it may be necessary to capture higher-order effects. Other designs, like the two-level (fractional) factorial design or nearly orthogonal array place more design points at the corners of the design region (i.e. grid space).

If no categorical hyperparameters need to be optimized through our DoE approach, an LHD can be used. Of course, however, the traditional configuration of the LHD cannot handle categorical factors (i.e. categorical factors cannot be divided into n levels). Instead, we suggest using a sliced Latin hypercube Design (SLHD) [Qia12], a special type of Latin hypercube that can be partitioned into slices and is useful for computer experiments with qualitative and quantitative factors. In this special LHD, each slice is a smaller Latin hypercube design and can correspond to a specific level for a categorical factor. For example, one

categorical factor may have three levels, where each level can assigned to one slice. These designs have two key features, as outlined in [Qia12]: (1) each slice of the design achieves maximum uniformity in any one-dimensional projection, and (2) when collapsed over all the slices, the whole design possess maximum stratification in any one dimension (i.e. we have one larger Latin hypercube). Formally, continuing with the notation from the LHD above, we define an SLHD(q, v, k) with $n = qv$ total runs , where k is the number of (quantitative) factors, v is the number of slices, and q is the number of levels within each slice. This is equivalent to an LHD of n runs with k factors that can be divided into v smaller slices, each with q equally spaced levels. To more clearly illustrate, we provide, from [Qia12], a visual illustration of an example SLHD, projected in two-dimensions at Figure 4.2 and the corresponding design (in transpose) at Figure 4.3.

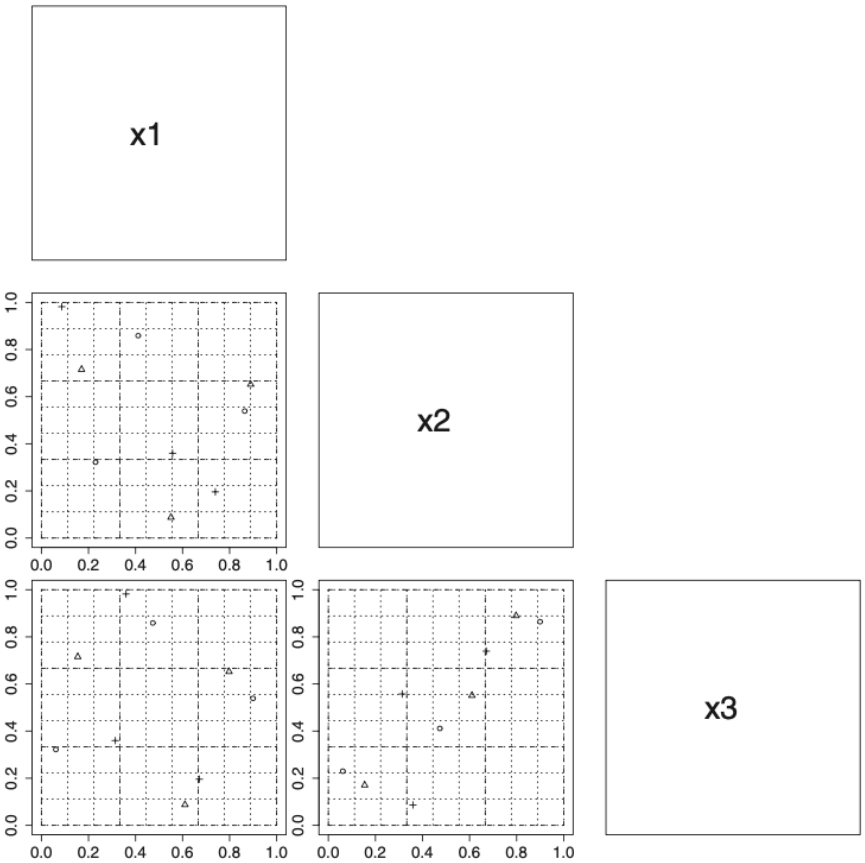

Figure 4.2: Two-Dimensional Projections of a 9×3 SLHD with 3 Slices

$$\left(\begin{array}{ccc|ccc|ccc} 8 & 5 & 3 & 7 & 1 & 6 & 4 & 2 & 9 \\ 4 & 8 & 3 & 2 & 5 & 9 & 6 & 7 & 1 \\ 5 & 9 & 1 & 2 & 6 & 8 & 3 & 7 & 4 \end{array}\right)$$

Figure 4.3: Example of a 9×3 SLHD with 3 Slices (in transpose)

The example SLHD shows a 9×3 LHD with $v = 3$ slices. Within each slice there are $q = 3$ levels. Explicity $n = qv = 3 \times 3 = 9$ and $k = 3$ factors, x_1, x_2, x_3. Returning to Figure 4.2, we can see the 2-d projection of the SLHD, where the three distinct symbols $(\mathrm{o}, +, \Delta)$, correspond to each of the $v = 3$ slices.

[Qia12] details algorithms for constructing SLHDs. It should be noted that when dealing with LHD/SLHDs, it is possible to collapse levels. For example, a factor that should contain 9 levels, can be mapped/collapsed into 3 levels instead. In doing so, we may lose some of the optimal sampling properties of the LHD/SLHD in their classical constructions, but this sacrifice is worth the flexibility of incorporating different types of factors (e.g. discrete factors that cannot be collapsed into the same range). We will see examples of such collapsing in our application at Section 5.

4.4 Composite Designs

Once a two-level design, NOA (or OA), and SLHD (or LHD) have been constructed, we can utilize combinations of these designs to form different composites. We draw inspiration from Xu et al's introduction of the Orthogonal Array Composite Design (OACD) [XJD14], a new class of composite design that combines a two-level factorial or fractional factorial array with a three-level orthogonal array. The proposed OACD has many benefits including the ability to incorporate higher resolution two-level fractional factorial designs with less aliasing with three-level arrays that allow us to estimate quadratic effects. In addition, multiple analyses can be conducted with different parts of the data. That is, estimates can be calculated and cross-validated by computing them for each sub-design and then again together from the

full OACD. This cross validation helps evaluate data quality and consistency. We adapt this idea and propose combining our two-level design and NOA to form our first composite design, the nearly orthogonal array composite design (NOACD).

Mirroring the ideas and advantages of the OACD/NAOCD, we next propose combining the two-level design and sliced Latin hypercube design to create a composite sliced Latin hypercube (CSLHD). This CSLHD echos many of the same advantages as the OACD/NOACD, but allows us to estimate higher-order effects depending on the number of levels chosen for the LHD. In particular, combining the two-level design and SLHD can be extremely powerful, as the two-level design explores more of the corners of the design region than a SLHD normally would.

The last possible paired sub-design consists of the NOA and SLHD (NOA-SLHD). Similar to the CLSHD, adding the NOA to the SLHD introduces more corner points. However, this design now also ensures that the center of the design-region is more completely covered.

Lastly, we propose combining all three designs together. We introduce this combined design as the *hybrid composite design*. Using the Hybrid Composite Design we are able to, again, maintain many of the benefits of the previous designs (e.g. cross validation, corner points), but now have expansive coverage over the entire design region (corner points, center points, "space-filling" points). Ideally, the hybrid composite design has also accumulated enough design points to de-alias any interaction effects that were previously aliased or unable to be estimated in any of the previous sub-designs or their composites. A practical example follows in Section 5.

4.5 Beta Regression

Once the designs and composites have been constructed, we must analyze the results to fit the accuracy response surface. While OLS has traditionally been used to estimate the effects for data generated by these designs, OLS also assumes that response variables are continuous and unbounded. In the case of hyperparameter optimization, our response variable, accuracy,

is proportion bounded $(0, 1)$. In addition, proportions tend to be heteroskedastic. That is, there is typically more variability around the mean response, compared to the bounds $(0, 1)$. If were to use standard OLS, we would like experiences issues with this and violate the assumptions of regression needed for inference. Beta regression, developed by Ferrari et al., is a more suitable approach for modeling rates and proportions [FC04].

Beta regression, assumes that the response variable is beta-distributed. As such, it is an incredibly flexible model: the underlying beta distribution can take on many different shapes depending on its parameterization. As a result, this regression method naturally incorporates heteroskedasticity into the model. Figure 4.4 provides some examples. In this thesis, we deal with the version of beta regression that predicts the mean of the response variable. We are not as concerned with the dispersion model.

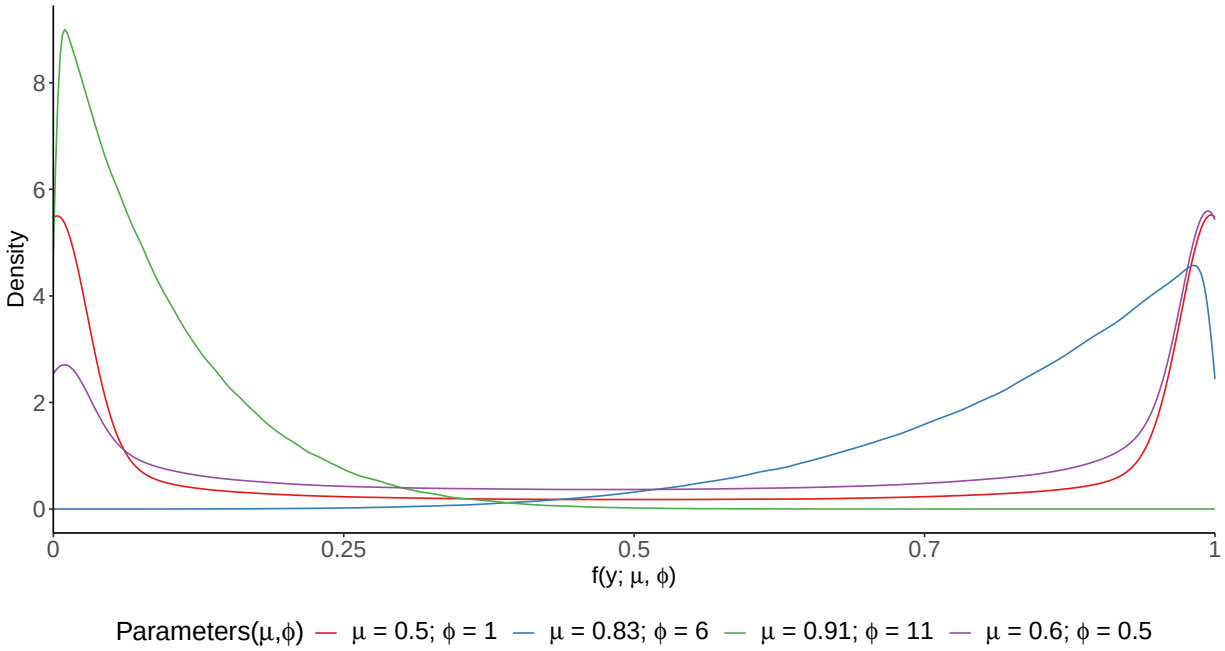

Figure 4.4: Various Beta Distributions with parameters μ and ϕ

In beta regression models, the beta distribution uses an alternative parameterization with μ and ϕ, such that the density is expressed by:

$$f(y; \mu, \phi) = \frac{\Gamma(\phi)}{\Gamma(\mu\phi)\Gamma((1-\mu)\phi)} y^{\mu\phi-1}(1-y)^{(1-\mu)\phi-1}, \quad 0 < y < 1$$

with $0 < \mu < 1$ and $\phi > 0$. Thus, $y \sim \mathcal{B}(\mu, \phi)$ and

$$\mathrm{E}(y) = \mu \tag{4.1}$$

$$var(y) = \frac{\mu(1-\mu)}{1+\phi} \tag{4.2}$$

Under this parameterization ϕ is referred to as the precision parameter, and its inverse, ϕ^{-1} is the dispersion parameter.

Similar to the set up in Generalized Linear Models (GLMs), we assume response vector $\boldsymbol{y} = (y_i, \ldots, y_n)$, where $y_i \sim \mathcal{B}(\mu_i, \phi_i)$ and the regression model is defined as

$$g(\mu_i) = \boldsymbol{x}_i^T \boldsymbol{\beta} = \eta_i \tag{4.3}$$

where $\boldsymbol{x}_i^T$ is the i^{th} row in our an $n \times k$ data/design matrix $\mathbf{X}$, $\boldsymbol{\beta}$ is a $k \times 1$ parameter vector, and $g(\cdot)$ is a link function which maps (0,1) to $\mathbb{R}$. The choice of link function can vary, as with GLMs. In this thesis, we use the traditional logit link function:

$$g(\mu) = \log\left(\frac{\mu}{1-\mu}\right) \tag{4.4}$$

Based on Equation 4.3,

$$\mu_i = g^{-1}\left(\boldsymbol{x}_i^T \boldsymbol{\beta}\right) = \frac{e^{\boldsymbol{x}_i^T \boldsymbol{\beta}}}{1 + e^{\boldsymbol{x}_i^T \boldsymbol{\beta}}}$$

where parameters $\boldsymbol{\beta}$ and ϕ are estimated by maximum likelihood estimation with log-likelihood function

$$\begin{aligned}
\ell(\boldsymbol{\beta}, \phi) &= \sum_{i=1}^{n} \ell_i(\mu_i, \phi) \\
&= \sum_{i=1}^{n} \left[\log \Gamma(\phi) - \log \Gamma\left(\mu_i \phi\right) - \log \Gamma\left(\left(1-\mu_i\right)\phi\right) + \left(\mu_i \phi - 1\right) \log y_i \right. \\
&\quad \left. + \left\{\left(1-\mu_i\right)\phi - 1\right\} \log\left(1 - y_i\right)\right]
\end{aligned}$$

Diagnostic analysis and goodness-of-fit for this model, similar to other linear models, are assessed by residual plots and pseudo R^2, which is defined as the correlation between $\hat{\eta}$ and $g(y)$ and is a value in [0,1] [FC04]. Like OLS, residual plots should show no obvious patterns.

In addition to beta regression being well suited to model accuracy, when using the logit link function shown in Equation 4.4, the coefficients for beta regression can be interpreted similarly to in logistic regression in terms of log-odds.

4.6 Methodology Summary

To summarize, our proposed DoE approach constructs three independent designs, namely (1) a two-level (fractional) factorial design, (2) a (nearly) orthogonal Array, and (3) a (sliced) Latin hypercube design. From these three independent designs, we generate four additional composite designs, (4) a (nearly) orthogonal composite design, (5) a composite (sliced) Latin hypercube design, (6) a (nearly) orthogonal array - (sliced) Latin hypercube composite (NOA-SLHD), and (7) a hybrid composite design. That is, there are seven possible overlapping data-sets or sub-designs that can be analyzed. This approach allows for an incredibly comprehensive, in-depth analysis of the data and cross-validation using beta regression. In addition, we strategically use increasingly granular designs (i.e. designs with an increasing amount of levels) to ensure adequate coverage across the design region. In doing so, we will be able to better and more accurately estimate high-order effects in the data. We can see that using this approach, we have properly addressed many of the considerations outlined in Section 3.4.

CHAPTER 5

Application to MNIST

In Chapter 2, we introduced CNNs and AlexNet, a seminal architecture, that significantly influenced developments in deep learning by introducing additional features and hyperparameters like dropout, ReLU, and normalization. As a result, to demonstrate the proposed methodology outlined in Section 4, we tune hyperparameters for a CNN loosely based on AlexNet [KSH12]. We use Krizhevsky et al.'s commentary on which features and hyperparameter decisions they felt significantly improved performance as a guide for which hyperparameters we can tune on a simpler dataset, MNIST. The dataset, factors (hyperparameters to tune), design matrices, data collection procedure, and analysis of results are described in the following sections.

5.1 MNIST

While newer CNN architectures are typically deployed on the ImageNet dataset used in the original AlexNet implementation, we choose to apply our DoE approach to the more manageable MNIST dataset due to computational limitations.

The MNIST, Modified National Institute of Standards and Technology, database of handwritten digits is a subset of a larger dataset present at NIST, the National Institute of Standards and Technology. It was first developed and released by Yann LeCunn et al. in 1999 [LCB10]. Since its release it has served as a canonical training dataset for many learning techniques and pattern recognition methods.

The dataset contains 70,000 grayscale images of the 10 digits that are 28×28 pixels in

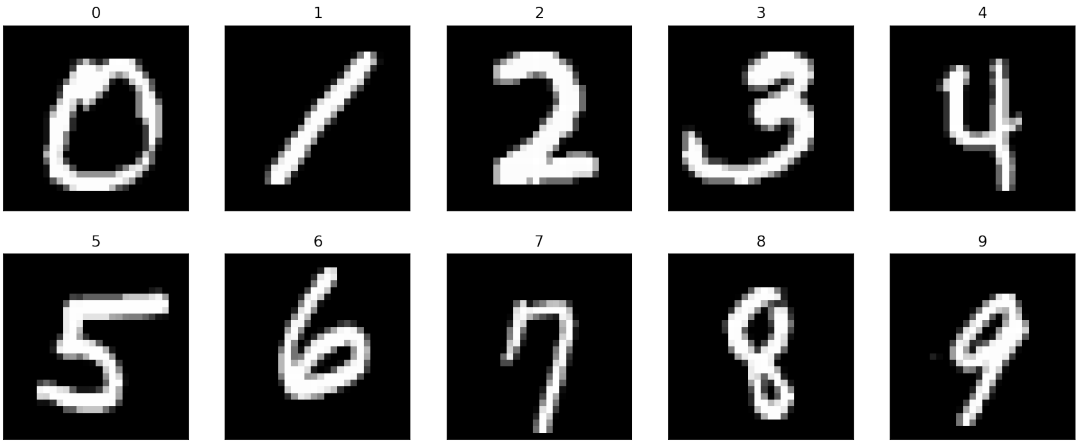

Figure 5.1: Examples from MNIST Handwritten Digit Database

width and height. Figure 5.1 shows some examples. In grayscale, pixel values range from 0 to 255[1]. The 70,000 total images are split into a training set of 60,000 examples, and a test set of 10,000 examples. The images in this dataset come with labels, and such are used for supervised learning tasks.

In our experiments, we further split the 60,000 images in the MNIST training set into a training subset set of 58,000 images and a validation set of 2,000 images (See Figure 5.2). The test set of 10,000 images remains unchanged.

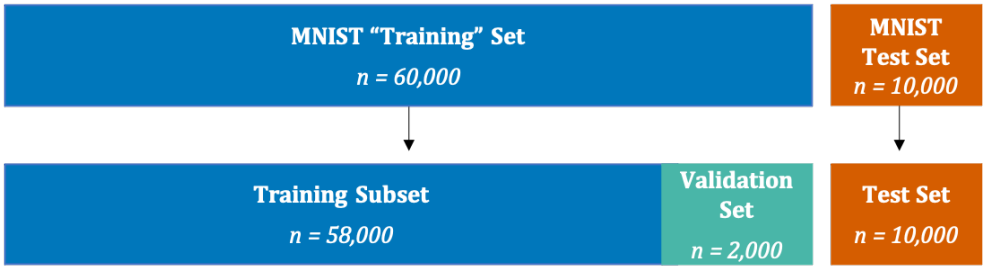

Figure 5.2: MNIST Dataset Split

[1]Higher pixel values correspond to dark; lower pixel values correspond to light

5.2 Experiment Factors: Hyperparameters to Tune

As mentioned, the original AlexNet authors incorporated "new and unusual features" [KSH12] to improve model accuracy, including the ReLU activation function, normalization, dropout, and additional convolutional layers (i.e. a deeper network architecture). We test and tune these unique hyperparameters, along with three other "universal" hyperparameters. There are a total of seven factors tested in our application.

Learning Rate (A): Refers to step size for the underlying gradient descent algorithm. It is often cited as the most important tuning parameters in a neural network, as without a properly tuned learning rate, neural networks can often fail to converge [Ben12]. AlexNet initialized their learning rate at 0.01. A common heuristic used in learning rate tuning is a $\log_{10}$ scale (i.e. dividing by 10). As such, *we set our factor range to [0.0001, 0.01]. This is a continuous factor.*

Number of Epochs/Iterations (B): Refer to the number of times the entire training dataset is shown to the network while training. This is a common universal hyperparameter to tune. Intuitively, number of epochs are generally increased/high, as it allows the network more opportunities to learn. However, too large a number can result in overfitting and drastically increases training time. AlexNet trained on ImageNet with 90 epochs, but MNIST is much less complex. *We test a range of [2,8] epochs. This is a discrete factor.*

Batch Size (C): Batch Size is closely related to number of epochs. It refers to the number of training examples given to the network at each pass (epoch) before network parameter are updated. This is another universal hyperparameter. Batch size can improve model accuracy, as this determines the number of examples the network can learn from in any given pass. Batch size usually increased in powers of 2. *We test batch size range of [16,128]. This is a discrete factor.*

Dropout Rate (D): Dropout rate forces a more robust model and refers to the fraction of neurons to drop throughout the learning process. This is a new feature added in AlexNet, which adds a regularization effect and helps with overfitting. *We test dropout rates of [0.3, 0.7] in each of the fully connected layers. This is a continuous factor.*

Activation Function (E): AlexNet was the first CNN to use the ReLU function to introduce nonlinearity. Until this introduction, the standard activation function was tanh. The authors applied ReLU activation after every convolutional and fully-connected layer. It has also recently been shown that ReLU significantly improves convergence over tanh. As such, *we test ReLU versus tanh. This is clearly a categorical factor of 2 levels.*

Number of Convolutional Layers (F): The number of convolutional layers controls the depth of the network. Generally, deeper networks perform better, but can also lead to overfitting. For AlexNet specifically, the creators were adamant that removing any 1 of their 5 convolutional layers resulted in worse performance. To adapt this architecture to MNIST, *we test architectures with a range of [2, 5] convolutional layers. This is a discrete factor.*

Normalization (G): Lastly, normalization rescales data by normalizing the output of each activation layer (subtracts batch mean and divides by batch standard deviation). AlexNet uses local response normalization, but this has fallen out of favor to batch normalization. [KSH12] claims that normalization reduced their error rates by 1-2% on ImageNet. For our application, *we test the model without and with batch normalization in the first 2 convolutional layers. This is a categorical factor of 2 levels.*

The ranges for each hyperparameter are intentionally large and are carefully chosen based on the types of factors we test. For example, our factor set includes 2 categorical variables with 2 levels each. In certain designs used, we may need to merge these 2 categorical variables into 1 categorical variable with 4 levels. The discrete factor ranges are chosen to follow suit – we ensure that there are at least 4 possible levels between the high and low boundaries of

the range. The seven factors for our experiment are summarized in Table 5.1.

Table 5.1: Summary of Factors

Factor	**Type**	**Range [Low, High]**	**Total Levels Tested**
Learning Rate (A)	Continuous	[0.0001, 0.01]	16
Number of Epochs (B)	Discrete	[2, 8]	5
Batch Size (C)	Discrete	[16, 128]	16
Dropout Rate (D)	Continuous	[0.3, 0.7]	16
Activation Function (E)	Categorical	[ReLU, Tanh]	2
Number of Convolutional Layers (F)	Discrete	[2, 5]	4
Batch Normalization (G)	Categorical	[Off, On]	2

5.3 Designs

The total composite design (hybrid composite design) for our application, which consists of three different small-run designs joined together, contains of 50 design points. Each of the components is described briefly.

2-level Fractional Factorial Design: For our 2-level design, we choose 2_{IV}^{7-3} design. The 2 levels used correspond to the low and high range summarized in Table 5.1. As discussed in Section 4, this serves as an initial screening design. We choose this specific minimum aberration, orthogonal design because of its limited run size and high resolution. The full aliasing structure is shown in Appendix A.

Nearly Orthogonal Array with Mixed Levels: For our expanded 3-level design, we needed to be mindful of the 2-level categorical factors that cannot be expanded to three levels. The additional level is necessary for all quantitative factors in order to estimate

higher-order effects. There are no existing OAs with an acceptable run-size that contain the appropriate combination of factors and levels for our experiment, as the smallest OA that can accommodate two 2-level factors and five 3-level factors requires 36 runs. As such, we constructed a nearly orthogonal array with mixed levels using the algorithm in [Xu02]. In our $OA'(18, 2^2, 3^5)$, the 3-level factors are orthogonal to each other and to the 2-level factors. However, the 2-level factors are not orthogonal to each other. This sacrifice is beneficial as the run size of $n = 18$ is 1/2 of the smallest possible OA.

Sliced Latin Hypercube Design: We construct a maximin sliced Latin hypercube design (SLHD) with $n = 16$ runs/levels to increase the spatial coverage of our quantitative factors. In this design, we combine our two 2-level categorical variables into 1 categorical factor of 4 levels. Explicitly, the possible combinations of Activation Function (E) $\times$ Batch Normalization (G) are treated as 4 levels of one factor. This factor creates the 4 slices for our SLHD summarized in Table 5.2. The slices were randomly assigned to these level combinations. It should also be noted that for two discrete factors, we collapse the 16 levels into 4 levels. The SLHD in terms of levels is shown at Table 5.3. In Table 5.3, it can be seen that the two discrete factors Number of Epochs (B) and Number of Convolutional Layers (F) contain only 4 distinct levels. Table 5.3 also shows that the levels for Activation Function (E) and Batch Normalization (G) remain the same within each slice. For example, in slice 1, $E = 2$ and $G = 1$; in slice 2, $E = 2$ and $G = 2$. As a result, within each slice, we have a smaller Latin hypercube design, as shown in Figure 5.3. Lastly, Figure 5.4 shows the projection of all $n = 16$ generated design points in 2-dimensions.

Table 5.2: Combined Categorical Factors for SLHD

Activation Function (E) x Batch Normalization (G)	**Slice**
Tanh × No Batch Normalization	1
Tanh × Batch Normalization	2
ReLU × Batch Normalization	3
ReLU × No Batch Normalization	4

Table 5.3: Maximin Sliced Latin Hybercube Design, $n = 16$

Slice	**A**	**B**	**C**	**D**	**E**	**F**	**G**
1	1	3	4	13	2	3	1
1	7	1	12	9	2	1	1
1	9	2	8	1	2	4	1
1	16	4	13	7	2	2	1
2	2	2	15	8	2	3	2
2	5	3	6	2	2	1	2
2	12	4	3	14	2	2	2
2	14	1	9	12	2	4	2
3	3	1	7	10	1	4	2
3	6	4	10	3	1	3	2
3	11	3	14	16	1	2	2
3	15	2	2	5	1	1	2
4	4	4	11	11	1	1	1
4	8	1	5	15	1	2	1
4	10	3	1	6	1	4	1
4	13	2	16	4	1	3	1

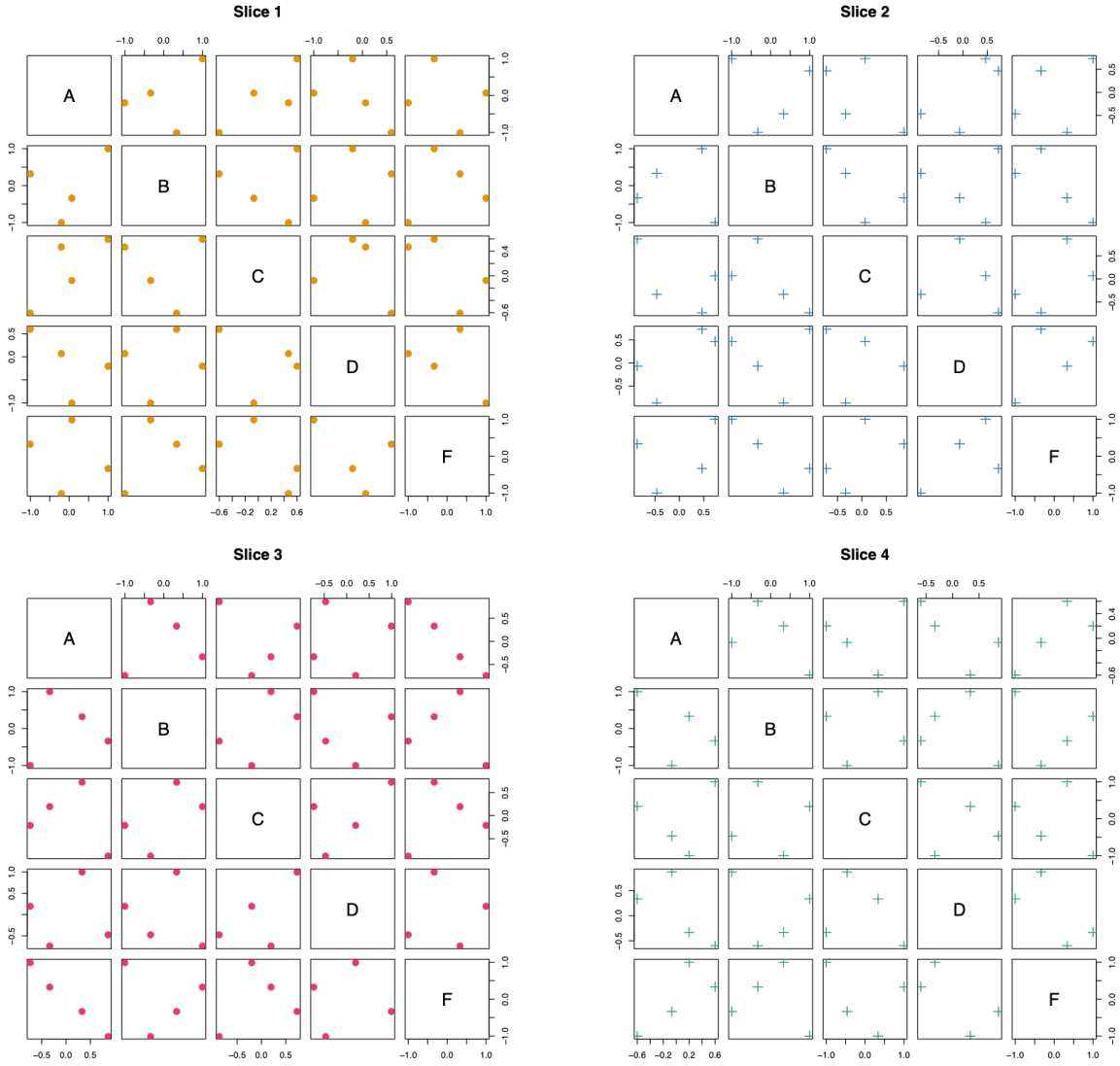

Figure 5.3: Latin Hypercube Design Points Projected in 2-d by Slice

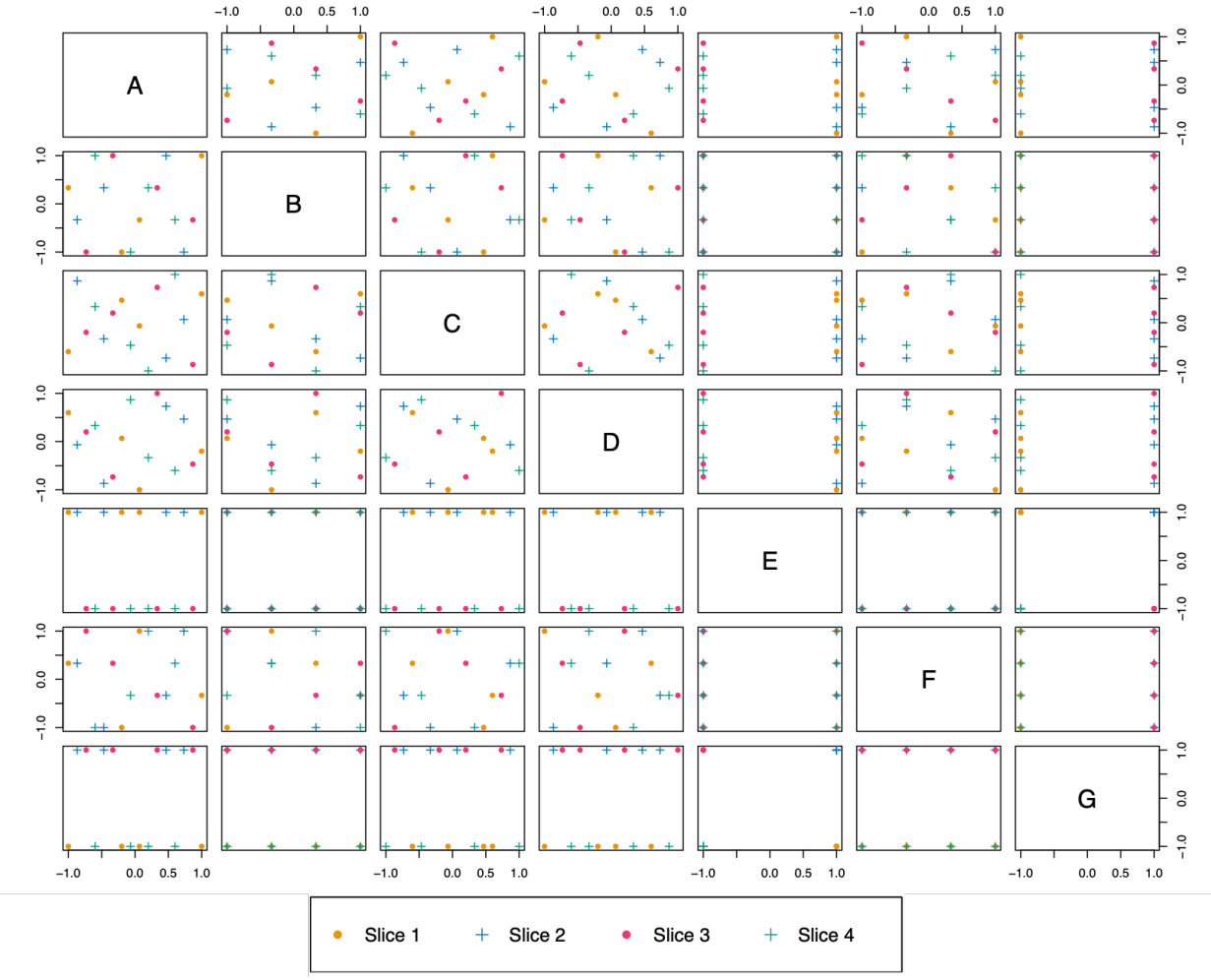

Figure 5.4: Sliced Latin Hypercube Design Points Projected in 2-d

Hybrid Composite Design: Finally, the total hybrid composite design and data for our experiments are shown in Table 5.4. The run order was randomized within each sub-design. Factor levels are coded [-1,1] according to the ranges listed in Section 5.2 and summarized in Table 5.1. For each factor, -1 corresponds to the low end of the provided range and 1 corresponds to the high end of the range. The observed data, accuracy and training time, were the percentage of correctly classified images in the test data set and total training time for the CNN characterized by factors A-G in seconds, respectively. Runs 1-16 represent the 2-level fractional factorial design, runs 17-34 correspond to the nearly orthogonal array with mixed levels, and runs 35-50 correspond to the sliced Latin hypercube design with 16 levels for all continuous factors. We use consistent coded factor levels, so coefficients and analysis across each component of this hybrid composite design are comparable (i.e. cross validation). Figure 5.5 shows all design points projected in two dimensions.

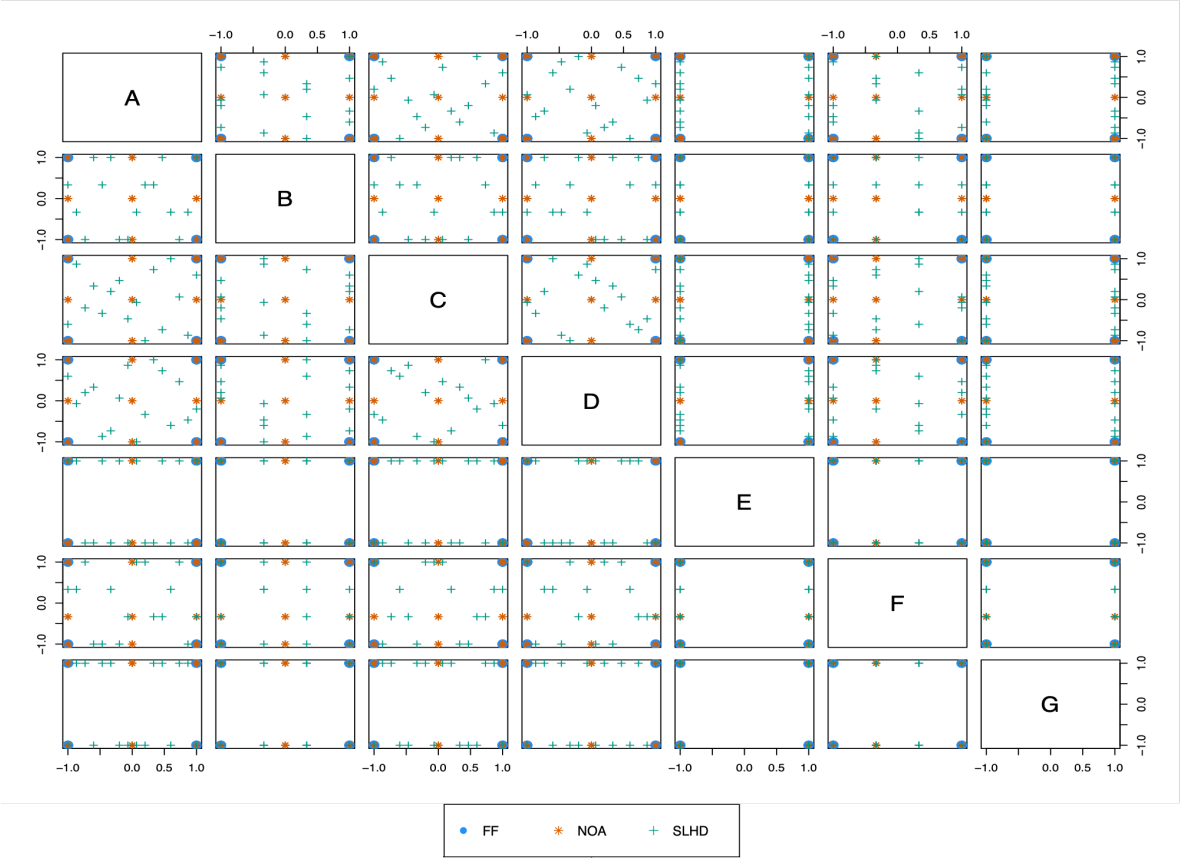

Figure 5.5: 50 Total Design Points Projected in 2-d

Table 5.4: Full Design Matrix with Coded Factors

Run	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**Accuracy**	**Training Time**
1	1	1	1	1	1	1	1	0.0982	5836.2
2	-1	1	-1	-1	1	1	-1	0.9852	7528.6
3	-1	1	1	-1	-1	-1	1	0.9837	5448.7
4	-1	-1	1	1	1	-1	-1	0.9813	1115.0
5	-1	-1	-1	-1	-1	-1	-1	0.9856	1993.7
6	-1	-1	-1	1	-1	1	1	0.9758	2408.3
7	1	-1	-1	1	1	1	-1	0.0892	1898.2
8	-1	-1	1	-1	1	1	1	0.9652	1465.5
9	1	-1	1	-1	-1	1	-1	0.1135	1181.5
10	-1	1	-1	1	1	-1	1	0.9791	8780.7
11	1	1	-1	1	-1	-1	-1	0.1010	6942.9
12	1	-1	1	1	-1	-1	1	0.1029	1179.1
13	1	-1	-1	-1	1	-1	1	0.0980	2079.5
14	-1	1	1	1	-1	1	-1	0.9857	5106.3
15	1	1	-1	-1	-1	1	1	0.1010	8447.8
16	1	1	1	-1	1	-1	-1	0.0982	3642.4
17	-1	-1	-1	-1	-1	-1	1	0.9765	2089.8
18	0	0	0	0	-1	-0.333	1	0.6629	2988.8
19	1	1	1	1	-1	1	1	0.1135	5812.8
20	-1	-1	0	0	-1	1	-1	0.9770	11934.0
21	0	0	1	1	-1	-1	-1	0.9844	2325.7
22	1	1	-1	-1	-1	-0.333	-1	0.1135	6900.8
23	-1	0	-1	1	-1	-0.333	-1	0.9860	5373.0
24	0	1	0	-1	-1	1	-1	0.9808	4782.9
25	1	-1	1	0	-1	-1	-1	0.1135	915.7

Table 5.4: Full Design Matrix with Coded Factors

Run	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**Accuracy**	**Training Time**
26	-1	1	1	0	1	-0.333	-1	0.9842	3716.7
27	0	-1	-1	1	1	1	-1	0.8955	1962.3
28	1	0	0	-1	1	-1	-1	0.1032	2554.3
29	-1	0	1	-1	1	1	1	0.9571	3715.4
30	0	1	-1	0	1	-1	1	0.1009	8553.0
31	1	-1	0	1	1	-0.333	1	0.1028	1214.4
32	-1	1	0	1	1	-1	1	0.9764	4739.0
33	0	-1	1	-1	1	-0.333	1	0.9397	1193.7
34	1	0	-1	0	1	1	1	0.1032	5564.9
35	-1	0.333	-0.6	0.6	1	0.333	-1	0.9795	4754.3
36	-0.2	-1	0.467	0.067	1	-1	-1	0.9684	1056.6
37	0.067	-0.333	-0.067	-1	1	1	-1	0.9354	2446.5
38	1	1	0.6	-0.2	1	-0.333	-1	0.1028	3735.3
39	-0.867	-0.333	0.867	-0.067	1	0.333	1	0.9688	3214.0
40	-0.467	0.333	-0.333	-0.867	1	-1	1	0.9833	3890.8
41	0.467	1	-0.733	0.733	1	-0.333	1	0.0892	5921.9
42	0.733	-1	0.067	0.467	1	1	1	0.1010	1422.6
43	-0.733	-1	-0.2	0.2	-1	1	1	0.9634	1470.0
44	-0.333	1	0.2	-0.733	-1	0.333	1	0.9862	4757.4
45	0.333	0.333	0.733	1	-1	-0.333	1	0.2098	3619.4
46	0.867	-0.333	-0.867	-0.467	-1	-1	1	0.1028	3711.2
47	-0.6	1	0.333	0.333	-1	-1	-1	0.9841	3733.8
48	-0.067	-1	-0.467	0.867	-1	-0.333	-1	0.9810	1035.3
49	0.2	0.333	-1	-0.333	-1	1	-1	0.9765	6976.0
50	0.6	-0.333	1	-0.6	-1	0.333	-1	0.9663	2045.9

It is important to note that when creating levels for certain discrete factors, it was necessary to use $\lfloor \cdot \rfloor$ or $\lceil \cdot \rceil$. We see this reflected in the coded levels at Table 5.4. For example, looking at Number of Convolutional Layers (F), a discrete variable, we see coded levels of -1, -0.333, 0.333, and 1. There is no level 0 in order to keep consistency across the 3 sub-designs. These coded levels correspond to 2, 3, 4, and 5 convolutional layers in natural units. A similar pattern is used for the other discrete variable, Number of Epochs (B). This is easily seen in a 2-d projection of design points shown in Figure 5.5. The total number of levels tested for each factor is summarized at Table 5.1, and the design matrix in terms of natural units is shown in Appendix B for reference.

5.4 Data Collection

All CNN implementations are built in TensorFlow[2]. Images are shuffled for each model trained, so there is a stochastic component to the achieved accuracy. After training is complete the network assesses test accuracy on the test dataset of 10,000 images. Accuracy and total training time are collected, as shown in Table 5.4[3]. We summarize the total training time and maximum accuracy from each of the component designs in Table 5.5.

Table 5.5: Summary of Experiment Data

Design	Runs	Max Accuracy	Total Training Time (hours)
2-Level Fractional Factorial	16	0.9857	18.1
Nearly Orthogonal Array	18	0.9860	18.2
Sliced Latin Hypercube	16	0.9862	14.9

[2] `https://www.tensorflow.org`

[3] Cross-entropy loss was also recorded, but is excluded from this thesis. Interested parties can find the full dataset available on GitHub. Refer to Appendix B.

A design point from the SLHD objectively achieved the highest accuracy. Overall, this design also took the least amount of time to train with $n = 16$ runs. For completeness, the distribution of accuracy and training time across all design points is shown in Figures 5.6 and 5.7. Of note, certain runs, for example, runs 9, 11, and 12, achieve low accuracy rates of $\pm 10\%$. From Figure 5.6, we can see that this is largely a function of the Learning Rate, set at $A = 1$. Other factors contributing to accuracy are evaluated in Section 5.5.

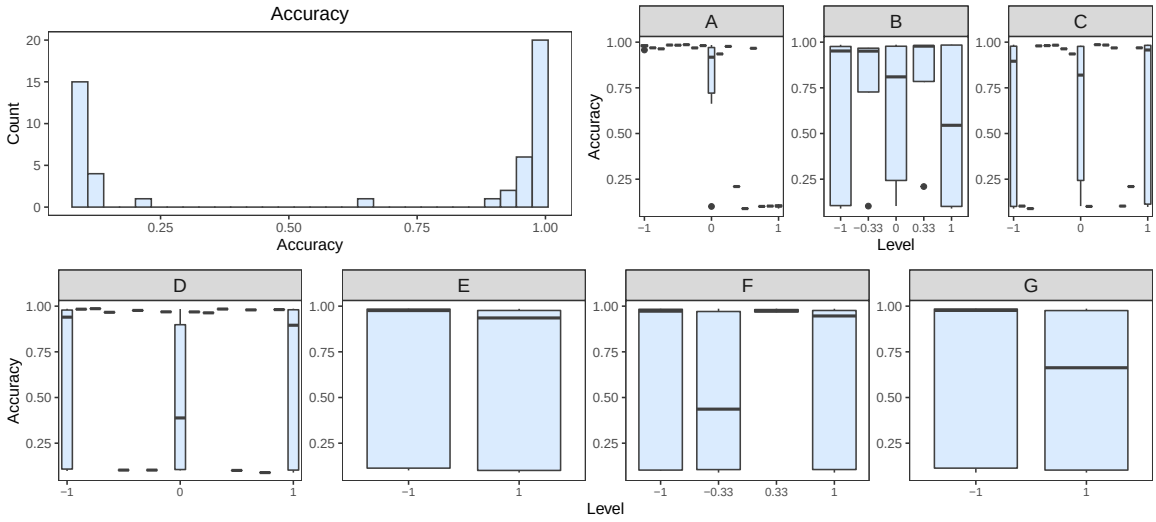

Figure 5.6: Accuracy Results (Hybrid Composite Design)

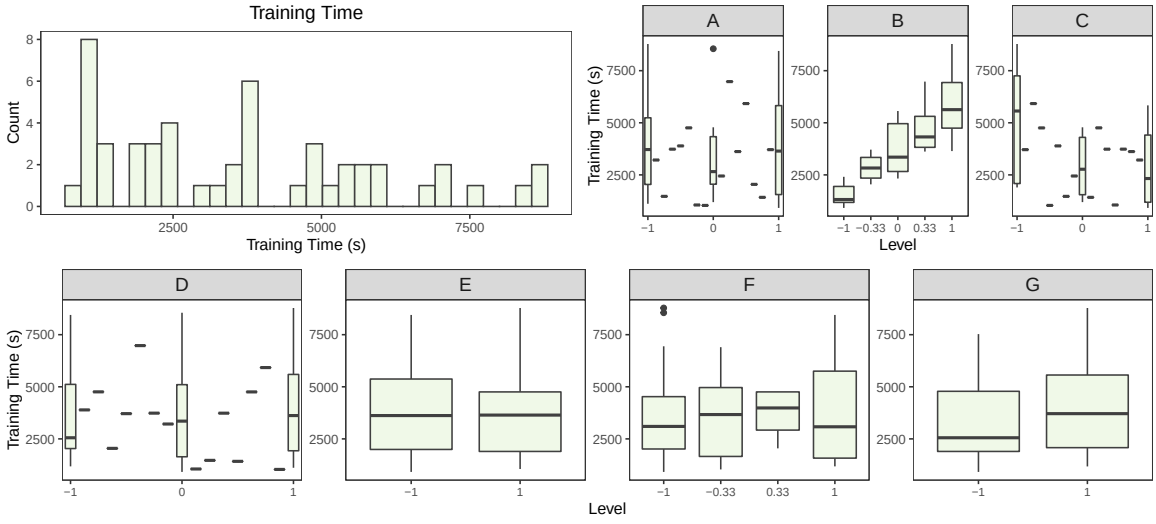

Figure 5.7: Training Time Results (Hybrid Composite Design)

5.5 Analysis using Beta Regression

5.5.1 Cross Validation

One of the considerations in Section 3.4 was the ability for cross validation, which allows us to reproduce or verify results. As such, our analysis strategy for the experimental results on MNIST consists of separating the 50 total design points into seven distinct datasets/sub-designs. As discussed in Section 4.6, these seven datasets[4] represent all available combinations of the three small-run designs: (1) the 2-level fractional factorial design (runs 1-16), (2) the nearly orthogonal array (runs 17-34), (3) the SLHD (runs 35-50), (4) the NOACD (runs 1-34), (5) the composite sliced Latin hypercube Design or CSLHD (runs 1-16, 35-50), (6) the NOA-SLHD (runs 17-50), and (7) the hybrid composite design (runs 1-50). We fit beta regression models to each of these sub-designs to help cross-validate our estimates. The model building process and estimates are discussed in Sections 5.5.2 and 5.5.3, respectively.

Briefly, we note that each of these sub-designs has its own benefits and limitations: each design possesses a different number of runs, levels, and orthogonality/space-filling properties. This poses a unique challenge in that each design is only capable of estimating certain effects. In addition, including all estimable effects for a given sub-design may result in a fully saturated model with limited remaining degrees of freedom. Based on this, we limit the effects we estimate/include for each sub-design (refer to Section 5.5.2). Still, we will see that several effects are estimated multiple times, providing the ability to cross-validate our results. Table 5.6 summarizes the seven sub-designs and estimates considered under each sub-design.

5.5.2 Model Building

In order to maintain the ability to cross-validate results and prevent overly saturated models, we use forward forward selection [DS66] with the Akaike Information Criterion (AIC) [Aka73]

[4]The seven combinations are from $\binom{3}{1} + \binom{3}{2} + \binom{3}{3}$

Table 5.6: Summary of Effects Available for Forward Selection by Sub-design

		Effects			
Design	**Runs**	**Linear**	**Bilinear**	**Quadratic**	**Cubic**
FF	16	✓	-	-	-
NOA	18	✓	-	✓	-
SLHD	16	✓	-	✓	✓
NOACD	34	✓	✓	✓	-
CSLHD	32	✓	✓	✓	-
NOA-SLHD	34	✓	✓	✓	-
Hybrid Composite Design	50	✓	✓	✓	✓

when building our beta regression models. AIC is used as the selection criterion, as we assume the true model is not amongst the candidate models. Models selected under AIC are shown to identify the model that gives the best prediction, assuming the true model is unknown [Boz87]. Forward selection is used due to the relatively small sample size in relation to the number of possible parameters that can be included in the model. For example, there are many possible bilinear interactions to consider, but a limited number of runs in each sub-design. This modeling approach allows us to not only identify the most important interactions effects, but also enforce the most parsimonious model [HTT20]. Of note, this model-building strategy can lead to different models under each sub-design.

As a baseline, we start by fitting a model that includes all linear main effects using the 2-level fractional factorial (FF) design. Forward selection is not applied to this design. While this design is technically capable of estimating a few bilinear interaction effects, estimating all such effects would result in an overly saturated model. A summary of estimates by sub-design at Table 5.6 shows that a beta regression model consisting of only main effects for these 16 runs already results in a pseudo R^2 of 99.7%. For consistency and cross-validation, all linear effects are included in the models subsequently fit to the remaining sub-designs. This is also in accordance with the sparsity of effects principle.

For the remaining six sub-designs, we employ forward selection, while still including all main effects in the models. As previously mentioned, not every sub-design is capable of estimating all interested effects, so we must limit the candidate pool of effects available for forward selection. For example, the NOA consists of only 18 runs with a maximum of three levels. As a result, only linear effects and pure quadratic effects are made available for forward selection into the model fit on this sub-design. Table 5.6 summarizes the effects available for inclusion by forward selection for each of the remaining sub-designs. We can see that bilinear effects can be selected across four sub-designs; quadratic effects can be included up to six times; and cubic effects can be selected up to two times. The decision for which effects to make available also considers the sparsity of effects principle discussed in Section 4.1.

Altogether, this approach provides an extensive and comprehensive view of our experimental data, but clearly addresses one of the considerations of cross-validation we set forth in Section 3.4. In addition, we have addressed other considerations such as incorporating interaction effects, as well as gaining an understanding of how each hyperparameter affects the response. The resulting estimates for each of the seven selected models are shown in Table 5.7. Note that we did not require any transformations on the response variable, as the usual assumptions on the error and residuals are reasonable.

5.5.3 Model Discussion

All six models generally have a high pseudo R^2 and the diagnostic plots are reasonable, which indicates goodness-of-fit [FC04]. The model fit to the FF design yields the highest pseudo R^2, and we can see that this model also has the highest value for ϕ, the precision parameter, indicating a smaller variance for clustered data near the bounds 0 and 1 [SWM13]. To gain some intuition for this, we show the marginal distribution of y (accuracy) from each sub-design at Figure 5.8 . For fixed mean μ, the larger ϕ, the smaller the variance of the response [FC04] based on Equation 4.2.

Table 5.7: Estimates of parameters for MNIST Experimental Data

	Design						
	FF	**NOA**	**SLHD**	**NOACD**	**CSLHD**	**NOA-SLHD**	**Hybrid Composite**
Int.	0.86***	0.96***	3.06***	1.32***	3.57***	2.21***	3.17***
A	-3.06***	-3.17***	-3.34***	-3.15***	-3.27***	-3.21***	-4.37***
B	0.02	-0.32*	-0.59**	0.03	0.08	-0.45***	-0.003
C	0.02	0.77***	-0.01	0.01	0.04	0.51***	-0.06
D	-0.02	0.06	-1.81***	-0.04	-0.13	-1.5***	-0.06
E	-0.06*	-0.35**	-0.28*	-0.20***	-0.20*	-0.38***	-0.21**
F	-0.01	0.11	0.24	-0.02	0.01	0.17	4.74***
G	-0.04	-0.61***	-1.01***	-0.24***	-0.44***	-0.74***	-0.57***
A^2	-	-1.17***	-2.11***	-0.76***	-0.26	-1.08***	-1.07***
B^2	-	-	-	-0.38**	-1.94**	-0.34	-0.36*
C^2	-	-	-	-1.26***	-1.41***	-	-0.70***
D^2	-	1.81***	-	1.64***	-	0.41**	0.78***
F^2	-	-0.58*	-0.66	0.37*	1.05***	-	-0.81**
A^3	-	-	-	-	-	-	1.23*
F^3	-	-	-	-	-	-	-4.61***
AE	-	-	-	-0.72***	-	-0.25	-0.39**
AF	-	-	-	0.29**	-	0.26	0.53***
AG	-	-	-	0.17*	-1.56***	0.59***	-0.02
BC	-	-	-	0.55***	-	1.91***	0.56***
BD	-	-	-	0.79***	-1.17***	-	-
BF	-	-	-	0.04	1.88***	0.35*	1.05***
BG	-	-	-	-0.42***	-	-	-0.63***
CD	-	-	-	-	-	-	-0.65***

Table 5.7: Estimates of parameters for MNIST Experimental Data

	Design						
	FF	**NOA**	**SLHD**	**NOACD**	**CSLHD**	**NOA-SLHD**	**Hybrid Composite**
CF	-	-	-	-1.41***	-	-0.86***	-0.47***
DE	-	-	-	0.16	-	0.62***	-
DF	-	-	-	0.29**	-	-0.34*	-
EF	-	-	-	-	-	-0.84***	-0.10
EG	-	-	-	0.65***	1.22***	0.82***	0.49***
ϕ	1285.5	59.9	106.2	547.9	105.1	119.0	86.36
Pseudo R^2	0.997	0.91	0.93	0.97	0.92	0.89	0.93
df	7	6	6	9	15	12	24

Note: Significance levels are coded as 0 (***) 0.001 (**) 0.01 (*) 0.05

It should be noted that pseudo R^2 is a basic measure of model-fit and is not be used for model selection. In the case of the FF design, the pseudo R^2 is noticeably high due to the limited number of design points.

Overall, we see that Learning Rate (A), Activation Function (E) and Batch Normalization (G) are consistently important factors in CNN accuracy for this dataset, detected as significant across almost all designs. Quickly in the first model fit to the FF design, we can see that Learning Rate (A) and Activation Function (E) emerge as the most important hyperparameters. This demonstrates the FF's ability to serve as an initial screening experiment. Unequivocally, lower settings for Learning Rate (A), Activation Function (E), and Batch Normalization (G) produce the best results. The quadratic effects for many factors are also consistently selected and significant across the models. In particular, Learning Rate (A), Number of Epochs (B), Dropout Rate (D), and Number of Convolutional Layers (F) emerge as a regularly selected and significant higher-order term. Additionally, for the factors

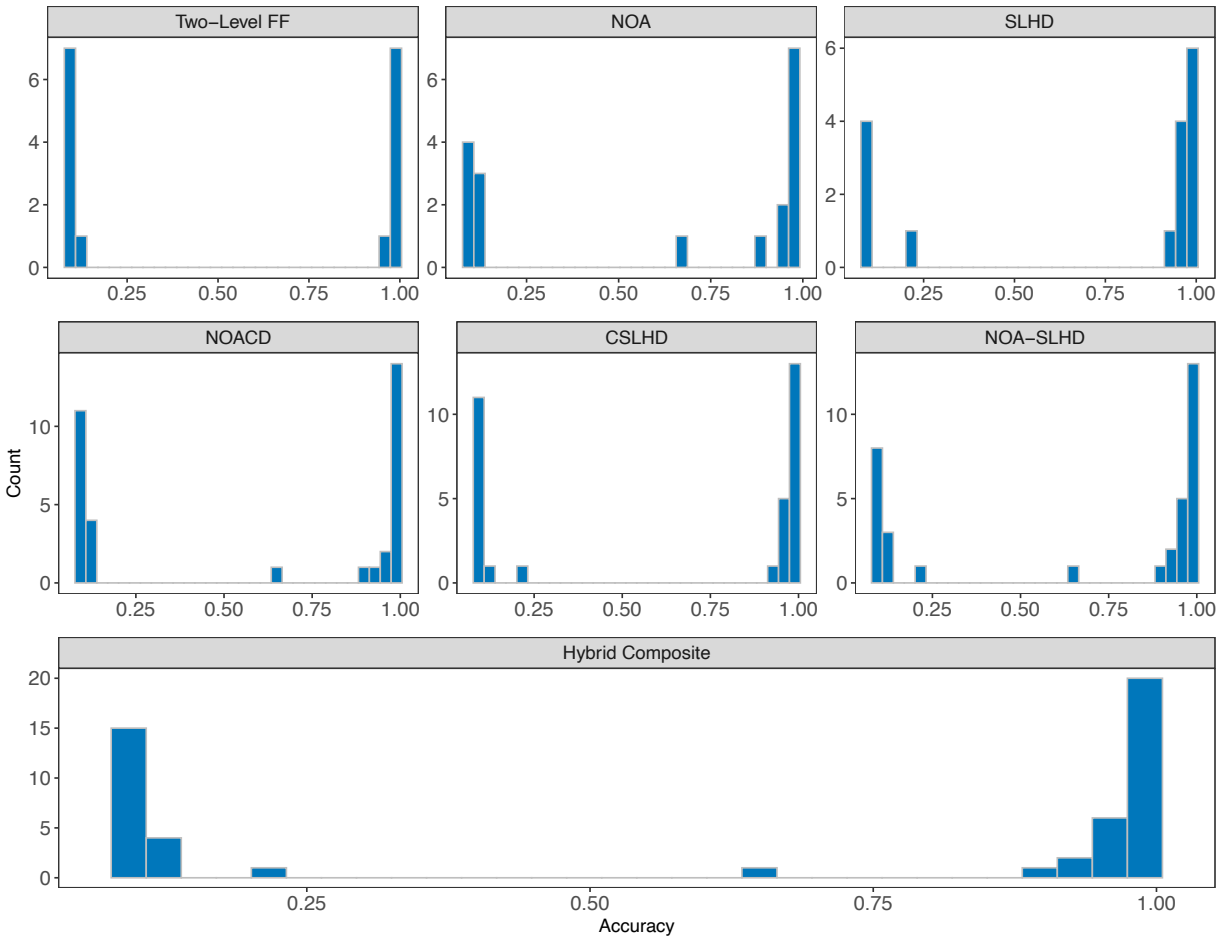

Figure 5.8: Histogram of Accuracy by Design

highlighted here, we note that the signs and estimates for the coefficients are largely similar across all 7 sub-designs. Several bilinear effects are also consistently selected, including AG, BF, and EG.

We do find some discrepancies across the sub-designs, especially amongst the bilinear and cubic effects selected. One explanation for this discrepancy is similar to that given in [XJD14]: different data being used to fit different models with distinct aliasing. Recall that our NOACD has a complex aliasing structure, because we sacrificed orthogonality for a smaller run-size. With the hybrid composite design, we are able to de-alias certain effects. We find that certain effects that had been aliased or excluded in forward selection for previous sub-designs are now included and significant. Two such examples are bilinear effects CD

and EF. We also see that harnessing the power of all 50 design points, certain cubic effects are now selected into the model, namely A^3 and F^3. Additionally, signs may flip between models due to the introduction of other significant effects. For example, we see the signs for certain main effects flip when quadratic effects are selected into the model. Likewise, signs for quadratic effects may flip when significant cubic effects are added into the model. Lastly, we notice differences in the coefficients selected between models that can estimate higher order effects, namely the NOA vs. SLHD designs and their corresponding composite designs. This is likely a benefit of the space-filling property of the SLHD, as these designs tend to be better at identifying nonlinear effects even compared to the NOA or NOACD. In addition, certain designs may be more apt to identify important bilinear effects. For example, the models based on the three composite designs that utilize NOA runs all include bilinear terms AE, AF, BC, and CF, where as the CSLHD fails to select these interaction effects that end up being selected in the hybrid composite model.

Residual analysis of these models showed that runs 9, 25, and 30 had large Cook's distances (i.e. large degree of influence) [Coo77] and residuals. When the hybrid composite model was re-fit without these data points, pseudo R^2 improved to 0.9561 and ϕ increased to 115.07. Additionally, quadratic effects for B^2, C^2, and F^2 were no longer found to be significant at the 0.05 level.

Generally, in examining the selected models for each sub-design, we can see that each of the three small-run designs contributes meaningful information. By using the data from all 50 design points, we are able to build a stronger model for an ostensibly complicated response surface. Explicitly, the FF design, being orthogonal, provides a strong basis for estimating the linear effects; the NOA provides an ability to estimate early quadratic effects; and the SLHD provides an ability to estimate cubic effects. In addition, the composites of these designs help identify the more important bilinear effects, based on their coverage of the design space.

5.6 Results

We take the model selected from the hybrid composite design as final model, which includes linear terms for A,B,C,E,E, F, and G; pure quadratic terms for A, B, C, D, and F; and pure cubic terms for A and F, and bilinear interaction effects for AE, AF, AG, BC, BF, BG, CD, CF, EF, and EG. The traditional logit link function shown at Equation 4.4 was used, and the resulting model is

$$\begin{aligned} g(\mu_i) = \; & 3.18 - 4.37A - 1.07A^2 + 1.23A^3 - 0.003B - 0.36B^2 - 0.06C - 0.7C^2 - 0.06D \\ & + 0.78D^2 - 0.21E + 4.74F - 0.82F^2 - 4.60F^3 - 0.57G - 0.39AE + 0.53AF - 0.02AG \\ & + 0.56BC + 1.05BF - 0.63BG - 0.65CD - 0.47CF - 0.10EF + 0.49EG \end{aligned} \quad (5.1)$$

The model produces a pseudo-R^2 of 0.9294 with $\phi = 86.36$.

Given Equation 5.1, the optimal settings based on all combinations of levels tested are shown in Table 5.8:

Table 5.8: Optimal Hyperparameter Settings Identified

Factor	**Coded Setting**	**Natural Setting**
Learning Rate (A)	-0.733	0.000184785
Number of Epochs (B)	1	8
Batch Size (C)	0.733	113
Dropout Rate (D)	-1	0.3
Activation Function (E)	-1	reLU
Number of Convolutional Layers (F)	0.333	4
Batch Normalization (G)	-1	Off

We performed five confirmation runs using these settings, which yielded mean accuracy of 0.98828, with a standard deviation of 0.00158. The lowest achieved accuracy in these runs was 0.9865, which is higher than any of the results in our initial 50 runs (refer to Table 5.5). The maximum achieved accuracy was 0.9900. Additionally, we can see that the optimal

factor settings identified are not limited to the combinations explicitly tested in the original experiments. This combination was entirely uniquely identified based on the fitted beta regression model.

We performed a similar analysis for the models selected under the other six sub-designs and noted that the optimal settings found under these models failed to achieve an improved accuracy rate. The model fit on the hybrid composite design without runs 9, 25, and 30 achieved similar performance to final model shown above. These results further indicates the necessity to utilize the hybrid composite design for the best model fitting.

CHAPTER 6

Discussion

Our results in Chapter 5 indicate the success of our proposed method in tuning a CNN to classify digits from the MNIST dataset. Here we briefly discuss a few of the strengths and limitations of our proposed methodology.

Model-Based, but not sequential: Our DoE approach utilizes beta regression to identify and model the important relationships between the hyperparameters and test accuracy. As such, it is not an entirely model-free approach. However, compared to other model-based approaches that rely on Bayesian Optimization [SLA12, MTZ78, BBB11] or Hyperband [LJD18], our proposed method is not sequential in nature. Each design point is pre-determined and selected. Further, our modeling process uses beta regression, which is relatively simple.

Parallelizable: Each of the design points in the 2-level fractional factorial design, nearly orthogonal array, and sliced Latin hypercube is predetermined and independent of all other design points. As such, each design point can be run individually, in parallel.

Sample Efficiency and Improved Computational Time: Each of the experimental designs utilized in this approach is carefully selected in order to maximize resolution (2-level fractional factorial design), and best preserve orthogonality (NOA) and space-filling properties (SLHD), while limiting the number of runs required. Our proposed approach, in total, uses only 50 design points. By contrast, a full grid search at minimum would have required 128 runs, which would be equivalent to a 2^7 complete factorial design. Testing more

than 2 levels for each factor would have required additional runs.

Diversity of hyperparameters: Previously proposed DoE-based hyperparameter optimization methods have focused on tuning quantitative continuous or discrete integer-valued hyperparameters. Our proposed approach considers and incorporates categorical hyperparameters, as demonstrated by our tuning of activation function and batch normalization. This can be extended to other important categorical hyperparameters such as optimizer.

Interpretability: Unlike traditionally used methods of grid search, random search, and Bayesian Optimization, our proposed DoE approach provides the ability to interpret and understand the effects of the hyperparameters by the use of beta regression. This is an important step towards more interpretable machine learning and future research.

Flexibility: In addition to the diversity of hyperparameters that can be incorporated into this framework, the sub-designs we have chosen are incredibly flexible. There are several 2-level designs of varying resolutions that can be chosen. This first experiment can also be re-run if the initial level ranges are too far apart. In the second experiment, a 3-level OA can be replaced with a nearly orthogonal array to reduce run-size. Lastly, for the SLHD, the practitioner can set the number of runs/levels and collapse factors as needed to fit within this fixed budget.

Cross-Validation: As shown in Table 5.6, the total hybrid composite design is comprised of several sub-designs. Each of which has the ability to estimate certain effects. Most effects, therefore, are estimated at least twice in this analysis, providing the ability to cross-validate data quality and results. Given an adequate model and reliable data, we expect estimates to be consistent across different models. Our experiment shows mild discrepancies explained by the introduction of new interaction terms and higher-order effects.

Limitations: As this method is not sequential, each of the three designs utilized is independent of the others. We do not utilize prior data to improve at each step. As a result, we may continue to test hyperparameters that have no significant effect on performance, which can unnecessarily consume precious computational resources. Additionally, in order to minimize run-size and test factors with a mixed number of levels, we sacrificed orthogonality in using a nearly orthogonal array. Lastly, we were required to collapse levels for certain discrete factors when constructing our SLHD.

CHAPTER 7

Conclusion and Future Work

In this thesis, we explored the use of a DoE approach to hyperparameter optimization by considering test accuracy as a response surface that can be modeled through beta regression. The studied approach borrows ideas from both model-free and model-based hyperparameter optimization methods. The three individual designs selected for study, a 2-level fractional factorial design, nearly orthogonal array with mixed levels, and a sliced Latin hypercube design maximize desirable properties such as resolution, orthogonality and space-fillingness, respectively, while minimizing the number of required runs for effect estimation. We also explore composites of these designs. In an application of this approach on the MNIST dataset, we are successfully able to identify a set of settings that achieves higher test accuracy than any of the original 50 design points.

There are several additional directions that this research can take. We can categorize these directions into: design, modeling, and algorithmic improvements. First, we have used three relatively common small-run designs in this thesis, we can potentially explore other space-filling designs, optimal designs for mixed categorical, discrete, and continuous factors, such as marginally coupled designs [DHL15], or optimal designs for beta regression. From a modeling perspective, we can explore the use Kriging/Gaussian Process models [QWW08] as opposed to beta regression. Additionally, the data collected for this experiment also includes cross-entropy loss and total training time. These additional response variables can be analyzed either using beta regression or via new modeling techniques. Analyzing these response variables may show that cross-entropy loss, which is closely related to accuracy, produces a less complex response surface. Likewise, modeling and analysis on training time

may help practitioners and researchers achieve an optimal balance between performance and computational limitations. From an algorithmic perspective, this work can be extended to develop an adaptive, sequential, or active learning approaches, following the principles of response surface methodology [BW51] that progressively improves the important hyperparameters and reduces, or eliminates unimportant factors in order to converge to a guaranteed global optimum.

Overall, this thesis takes a tactical, applied approach to investigating the use of various designs in hyperparameter optimization and limits the analysis to a study in Convolutional Neural Networks. Future research may benefit from more in-depth bench-marking against other DoE approaches (or other model-free methods) and thorough analysis of the theoretical properties of our design and analysis or any future derivatives. This theoretical analysis can also be a step in understanding how well our proposed DoE-based approach may translate to other deep learning algorithms.

APPENDIX A

Alias Relationship for 2^{7-3}_{IV} Fractional Factorial Design

1/8 fraction 7 factors in 16 runs

Resolution IV

Design Generators

E = ABC F = BCD G = ACD

Defining Relation: I = ABCE = BCDF = ADEF = ACDG = BDEG = ABFG = CEFG

Aliases

A = BCE = DEF = CDG = BEF	AB = CE = FG
B = ACE = CDF = DEG = AFG	AC = BE = DG
C = ABE = BDF = ADG = EFG	AD = EF = CG
D = BCF = AEF = ACG = BEG	AE = BC = DF
E = ABC = ADF = BDG = CFG	AF = DE = BG
F = BCD = ADE = ABG = CEG	AG = CD = BF
G = ACD = BDE = ABF = CEF	BD = CF = EG

ABD = CDE = ACF = BEF = BCG = AEG = DFG

APPENDIX B

Full Design Matrix in Natural Units

Table B.1: Full Design Matrix (Natural Units)

Run	A	B	C	D	E	F	G	Accuracy	Training Time
1	0.0100	8	128	0.7	tanh	5	on	0.0982	5836.2
2	0.0001	8	16	0.3	tanh	5	off	0.9852	7528.6
3	0.0001	8	128	0.3	relu	2	on	0.9837	5448.7
4	0.0001	2	128	0.7	tanh	2	off	0.9813	1115.0
5	0.0001	2	16	0.3	relu	2	off	0.9856	1993.7
6	0.0001	2	16	0.7	relu	5	on	0.9758	2408.3
7	0.0100	2	16	0.7	tanh	5	off	0.0892	1898.2
8	0.0001	2	128	0.3	tanh	5	on	0.9652	1465.5
9	0.0100	2	128	0.3	relu	5	off	0.1135	1181.5
10	0.0001	8	16	0.7	tanh	2	on	0.9791	8780.7
11	0.0100	8	16	0.7	relu	2	off	0.1010	6942.9
12	0.0100	2	128	0.7	relu	2	on	0.1029	1179.1
13	0.0100	2	16	0.3	tanh	2	on	0.0980	2079.5
14	0.0001	8	128	0.7	relu	5	off	0.9857	5106.3
15	0.0100	8	16	0.3	relu	5	on	0.1010	8447.8
16	0.0100	8	128	0.3	tanh	2	off	0.0982	3642.4
17	0.0001	2	16	0.3	relu	2	on	0.9765	2089.8
18	0.0010	5	72	0.5	relu	3	on	0.6629	2988.8

19	0.0100	8	128	0.7	relu	5	on	0.1135	5812.8
20	0.0001	2	72	0.5	relu	5	off	0.9770	1194.0
21	0.0010	5	128	0.7	relu	2	off	0.9844	2325.7
22	0.0100	8	16	0.3	relu	3	off	0.1135	6900.8
23	0.0001	5	16	0.7	relu	3	off	0.9860	5373.0
24	0.0010	8	72	0.3	relu	5	off	0.9808	4782.9
25	0.0100	2	128	0.5	relu	2	off	0.1135	915.7
26	0.0001	8	128	0.5	tanh	3	off	0.9842	3716.7
27	0.0010	2	16	0.7	tanh	5	off	0.8955	1962.3
28	0.0100	5	72	0.3	tanh	2	off	0.1032	2554.3
29	0.0001	5	128	0.3	tanh	5	on	0.9571	3715.4
30	0.0010	8	16	0.5	tanh	2	on	0.1009	8553.0
31	0.0100	2	72	0.7	tanh	3	on	0.1028	1214.4
32	0.0001	8	72	0.7	tanh	2	on	0.9764	4739.0
33	0.0010	2	128	0.3	tanh	3	on	0.9397	1193.7
34	0.0100	5	16	0.5	tanh	5	on	0.1032	5564.9
35	0.0001	6	38	0.62	tanh	4	off	0.9795	4754.3
36	0.0006	2	98	0.5133	tanh	2	off	0.9684	1056.6
37	0.0012	4	68	0.3	tanh	5	off	0.9354	2446.5
38	0.0100	8	105	0.46	tanh	3	off	0.1028	3735.3
39	0.0001	4	120	0.4867	tanh	4	on	0.9688	3214.0
40	0.0003	6	53	0.3267	tanh	2	on	0.9833	3890.8
41	0.0029	8	30	0.6467	tanh	3	on	0.0892	5921.9
42	0.0054	2	75	0.5933	tanh	5	on	0.1010	1422.6
43	0.0002	2	60	0.54	relu	5	on	0.9634	1470.0
44	0.0005	8	83	0.3533	relu	4	on	0.9862	4757.4
45	0.0022	6	113	0.7	relu	3	on	0.2098	3619.4

46	0.0074	4	23	0.4067	relu	2	on	0.1028	3711.3
47	0.0003	8	90	0.5667	relu	2	off	0.9841	3733.8
48	0.0009	2	45	0.6733	relu	3	off	0.9810	1035.3
49	0.0016	6	16	0.4333	relu	5	off	0.9765	6976.0
50	0.0040	4	128	0.3800	relu	4	off	0.9663	2045.9

The full dataset including the recorded response of cross-entropy loss is available on GitHub.

REFERENCES

[Aka73] Hirotogu Akaike. "Information Theory and an Extension of the Maximum Likelihood Principle." In B.N. Petrov and F. Csaki, editors, *International Symposium on Information Theory*, pp. 267–281, 1973.

[BB60] G. E. P. Box and D. W. Behnken. "Some New Three Level Designs for the Study of Quantitative Variables." *Technometrics*, **2**:455–475, 1960.

[BB12] James Bergstra and Yoshua Bengio. "Random Search for Hyper-Parameter Optimization." *Journal of Machine Learning Research*, **13**(10):281–305, 2012.

[BBB11] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. "Algorithms for hyper-parameter optimization." In *25th annual conference on neural information processing systems (NIPS 2011)*, volume 24. Neural Information Processing Systems Foundation, 2011.

[Ben12] Yoshua Bengio. "Practical recommendations for gradient-based training of deep architectures." *CoRR*, **abs/1206.5533**, 2012.

[Boz87] Hamparsum Bozdogan. "Model selection and Akaike's Information Criterion (AIC): The general theory and its analytical extensions." *Psychometrika*, **52**(3):345–370, September 1987.

[BW51] G. E. P. Box and K. B. Wilson. "On the Experimental Attainment of Optimum Conditions." *Journal of the Royal Statistical Society. Series B (Methodological)*, **13**(1):1–45, 1951.

[Coo77] R. Dennis Cook. "Detection of Influential Observation in Linear Regression." *Technometrics*, **19**(1):15–18, 1977.

[CW01] Shao-Wei Cheng and C. F. J. Wu. "Factor Screening and Response Surface Exploration." *Statistica Sinica*, **11**(3):553–580, 2001.

[DHL15] Xinwei Deng, Ying Hung, and C. Devon Lin. "Design for Computer Experiments with Qualitative and Quantitative Factors." *Statistica Sinica*, **25**(4):1567–1581, 2015.

[DL90] Norman R. Draper and Dennis K. J. Lin. "Small Response-Surface Designs." *Technometrics*, **32**(2):187–194, 1990.

[DS66] N. Draper and H. Smith. *Applied Regression Analysis*. Wiley, New York, 1966.

[FC04] Silvia Ferrari and Francisco Cribari-Neto. "Beta Regression for Modelling Rates and Proportions." *Journal of Applied Statistics*, **31**(7):799–815, 2004.

[FH19] Matthias Feurer and Frank Hutter. "Hyperparameter optimization." In *Automated Machine Learning*, pp. 3–33. Springer, Cham, 2019.

[Gro18] Ulrike Grömping. "R Package DoE.base for Factorial Experiments." *Journal of Statistical Software*, **85**(5):1–41, 2018.

[HHL11] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. "Sequential Model-Based Optimization for General Algorithm Configuration." In Carlos A. Coello Coello, editor, *Learning and Intelligent Optimization*, pp. 507–523, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[HTT20] Trevor Hastie, Robert Tibshirani, and Ryan Tibshirani. "Best Subset, Forward Stepwise or Lasso? Analysis and Recommendations Based on Extensive Comparisons." *Statistical Science*, **35**(4):579 – 592, 2020.

[ISC21] D. Im, Cristina Savin, and Kyunghyun Cho. "Online hyperparameter optimization by real-time recurrent learning." *ArXiv*, **abs/2102.07813**, 2021.

[KA16] Vahab Khoshdel and A. Akbarzadeh. "Application of statistical techniques and artificial neural network to estimate force from sEMG signals." *Journal of AI and Data Mining*, **4**:135–141, 2016.

[KLF91] R. Kacker, E. Lagergren, and J. Filliben. "Taguchi's Orthogonal Arrays Are Classical Designs of Experiments." *Journal of Research of the National Institute of Standards and Technology*, **96**:577 – 591, 1991.

[KSH12] A. Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet classification with deep convolutional neural networks." *Communications of the ACM*, **60**:84 – 90, 2012.

[LBB98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-Based Learning Applied to Document Recognition." In *Proceedings of the IEEE*, volume 86, pp. 2278–2324, 1998.

[LCB10] Yann LeCun, Corinna Cortes, and CJ Burges. "MNIST handwritten digit database." ATT Labs [Online], 2010. `http://yann.lecun.com/exdb/mnist`.

[LHR18] Gustavo A. Lujan-Moreno, Phillip R. Howard, Omar G. Rojas, and Douglas C. Montgomery. "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, 2018.

[LJD18] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. "Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization." *Journal of Machine Learning Research*, **18**(185):1–52, 2018.

[MBC79] M. D. McKay, R. J. Beckman, and W. J. Conover. "Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code." *Technometrics*, **21**(2):239–245, 1979.

[Mon13] Douglas C. Montgomery. *Design and Analysis of Experiments*. John Wiley & Sons, Inc., Hoboken, NJ, USA, 2013.

[MSM16] Dmytro Mishkin, Nikolay Sergievskiy, and Jiri Matas. "Systematic evaluation of CNN advances on the ImageNet." *CoRR*, **abs/1606.02228**, 2016.

[MTZ78] Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. "The Application of Bayesian Methods for Seeking the Extremum." *Towards Global Optimization*, **2**:117–129, 1978.

[Qia12] Peter Z. G. Qian. "Sliced Latin Hypercube Designs." *Journal of the American Statistical Association*, **107**(497):393–399, 2012.

[QWW08] Peter Z. G Qian, Huaiqing Wu, and C. F. Jeff Wu. "Gaussian Process Models for Computer Experiments With Qualitative and Quantitative Factors." *Technometrics*, **50**(3):383–396, 2008.

[RH18] Jan N. van Rijn and Frank Hutter. "Hyperparameter Importance Across Datasets." *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, Jul 2018.

[SLA12] Jasper Snoek, Hugo Larochelle, and Ryan Adams. "Practical Bayesian Optimization of Machine Learning Algorithms." *Advances in Neural Information Processing Systems*, **4**, 2012.

[Ste87] M. L. Stein. "Large sample properties of simulations using latin hypercube sampling." *Technometrics*, **29**:143–151, 1987.

[SWM13] M. Schmid, Florian Wickler, K. Maloney, R. Mitchell, Nora Fenske, and A. Mayr. "Boosted Beta Regression." *PLoS ONE*, **8**, 2013.

[WCL20] Jia Wu, SenPeng Chen, and XiYuan Liu. "Efficient Hyperparameter Optimization Through Model-based Reinforcement Learning." *Neurocomputing*, **409**:381–292, 2020.

[WW92] J. C. Wang and C. F. J. Wu. "Nearly Orthogonal Arrays With Mixed Levels and Small Runs." *Technometrics*, **34**(4):409–422, 1992.

[XJD14] Hongquan Xu, Jessica Jaynes, and Xianting Ding. "Combining two-level and three-level orthogonal arrays for factor screening and response surface exploration." *Statistica Sinica*, **24**:269–289, 2014.

[XPW09] Hongquan Xu, Frederick Kin Hing Phoa, and Weng Kee Wong. "Recent Developments in Nonregular Fractional Factorial Designs." *Statistics Surveys*, **3**:18–46, 2009.

[Xu02] Hongquan Xu. "An Algorithm for Constructing Orthogonal and Nearly-Orthogonal Arrays With Mixed Levels and Small Runs." *Technometrics*, **44**(4):356–368, 2002.

[ZCY19] X. Zhang, Xiaocong Chen, L. Yao, Chang Ge, and Manqing Dong. "Deep Neural Network Hyperparameter Optimization with Orthogonal Array Tuning." *ArXiv*, **abs/1907.13359**, 2019.