REDUCING THE SEARCH SPACE FOR HYPERPARAMETER OPTIMIZATION USING GROUP SPARSITY

Minsu Cho and Chinmay Hegde

Iowa State University
ECE Department
Ames, IA, USA 50011

ABSTRACT

We propose a new algorithm for hyperparameter selection in machine learning algorithms. The algorithm is a novel modification of Harmonica, a spectral hyperparameter selection approach using sparse recovery methods. In particular, we show that a special encoding of hyperparameter space enables a natural group-sparse recovery formulation, which when coupled with HyperBand (a multi-armed bandit strategy) leads to improvement over existing hyperparameter optimization methods such as Successive Halving and Random Search. Experimental results on image datasets such as CIFAR-10 confirm the benefits of our approach.

Index Terms— Hyperparameter optimization, sparse recovery, deep learning.

1. INTRODUCTION

1.1. Setup

Machine learning (ML) models have been developed successfully to perform complex prediction tasks in recent years. However, most ML algorithms, especially in deep learning, require manual selection of several hyperparameters such as learning rate, regularization penalty constants, dropout ratio, and model architecture. The quality of the model depends on how the designer of the ML model has carefully chosen the hyperparameters; however, the complexity and variety of ML models magnifies the practical difficulties of selecting appropriate combinations of parameters to maximize performance. The area of hyperparameter optimization (HPO) addresses the problem of searching for the optimal choices in hyperparameter space.

Formally, let X denote the space of hyperparameters (whether numerical and categorical), and let f be the function mapping from X to the test loss obtained by training a given ML algorithm with a particular set of hyperparameters. The goal of HPO is to approximate a set of hyperparameters "close enough" to the global optimum

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

as efficiently as possible.

1.2. Prior Work

Traditionally, ML practitioners have solved the HPO problem via brute-force techniques such as grid search over X. This strategy quickly runs into exponentially increasing computation costs with each additional dimension in hyperparameter space. As a solution, Bayesian Optimization (BO) techniques have been proposed. These assume a certain prior distribution over the cost function $f(x)$ and updates the posterior distribution with each new "observation" (or measurement of training loss) at a given set of hyperparameters [1, 2, 3, 4, 5, 6, 7]. Subsequently, an acquisition function samples the posterior to form a new set of hyperparameters, and the process iterates.

Despite the popularity of BO techniques, they often provide unstable performance, particularly in high-dimensional hyperparameter space. An alternative technique to BO is Random Search (RS), which not only provides computational efficiency compared to grid search, but also strong "anytime" performance with easy parallel implementation [8].

Multi-armed bandit (MAB) approaches adapt the random search strategy to allocate the different resources to the randomly chosen candidate points to speed up the convergence to the optimum instead of spending full resources as random search and the BO. Successive Halving (SH) and Hyperband adapt the multi-armed bandit approach to random search, picking more candidates than random search with the same amount of budget by pruning poorly-performing hyperparameters in the early state [9, 10, 11]. In contrast with BO techniques (which are hard to parallelize), the integration of BO and Hyperband achieve both advantages of guided selection and parallelization [12, 13, 14].

Gradient descent methods [15, 16, 17, 18, 19] (or more broadly, meta-learning approaches) have also been applied to solve the HPO problem, but these are only suitable to optimize continuous hyperparameters. Since this is a very vast area of current research, we do not compare our approach with these techniques.

While BO dominates the model-based approach, a recent technique called *Harmonica* proposed a *spectral* approach, applying ideas from *sparse recovery* on a Boolean version of the objective function. Using this approach, Harmonica provides the unique benefit of reducing the dimensionality of hyperparameter space by quickly finding highly influential hyperparameters, following which other standard (search or optimization) techniques can be used [20].

1.3. Our Contributions

Our main contribution is an extension to the Harmonica algorithm. While it successfully demonstrates finding important categorical features, we focus on finding the numerical features by proposing a new representation on numerical hyperparameter values. The representation not only reduces the dimension of hyperparameter space, but also groups the hyperparameters based on knowledge of its structure to achieve improved accuracy and stability.

Email: {chomd90,chinmay}@iastate.edu. This work is supported in part by grants from NSF CCF-1750920, a Faculty Fellowship from the Black and Veatch Foundation, and an equipment donation from NVIDIA Corporation.

To supplement our algorithm, we validate our numerical expression with hyperparameters grouping to examine its guidance accurately. We visually show that this algorithm closely approximates the global minimum in hyperparameter space by plotting the loss surface with two hyperparameters. We also show the robustness of our proposed algorithm combining the guidance to the decision-theoretic methods with measurable improvements in test loss using a CNN architecture trained on the CIFAR-10 image classification dataset.

1.4. Techniques

Following [13], we observe three desiderata to be satisfied with any HPO method: parallelizability, scalability, and strong final performance. The first criterion is parallelizability of the algorithm since HPO requires expensive computations. We use Hyperband, which is the current state-of-art in multi-armed bandit approaches, as the base algorithm to satisfy the first qualification.

To achieve the second and third criteria, we use the Harmonica trick [20]: we first binarize the hyperparameter space, and decompose the Fourier expansion of the (Boolean) function f. Finding the influential hyperparameters from a small number of (sampled) training loss observations reduces to solving a group-sparse recovery problem from compressive measurements. This leads us to better overall test error for a given computational budget.

2. MATHEMATICAL MODEL AND ALGORITHM

We now present our HPO algorithm; we restrict our attention to discrete domains (and assume that continuous hyperparameters have been appropriately binned). Let $f : \{-1, 1\}^n \mapsto \mathbb{R}$ be the loss function to be optimized. Let there be k different types of hyperparameters. In other words, we allocate n_i bits to the i^{th} hyperparameter category such that $\sum_{i=1}^{k} n_i = n$. The task of HPO involves searching the approximate hyperparameters close to the global minimizer

$$x^* = \underset{x \in \{-1,1\}^n}{\arg\min}\ f(x). \tag{2.1}$$

2.1. PGSR-HB

We propose *Polynomial Group-Sparse Recovery within Hyperband* (PGSR-HB), a new HPO search algorithm which enables considerable reduction of the hyperparameter space. We combine Hyperband, the multi-armed bandit method that balances exploration and exploitation from uniformly random sampled hyperparameter configurations, with a group sparse version of Polynomial Sparse Recovery, which is the main component of the spectral decomposition-based Harmonica method of HPO. Algorithm 1 shows the pseudo code of PGSR-HB.

PGSR-HB adopts the decision-theoretic approach of Hyperband, but with the additional features of tracking the history of all loss values from different resources. Hyperband contains the subroutine algorithm, Successive Halving (abbreviated as SH, see Lines 7-14), following the assumption that the performance of different hyperparameter choices in the process of training indicates which configurations are worth investing further resources, and which ones are fit to discard.

Algorithm 1 PGSR-HB

```
 1: Inputs: Resource $R$, scaling factor $\eta$, total cycle $c$
 2: Initialization: $s_{max} = \lfloor \log_\eta(R) \rfloor$, $B = (s_{max} + 1)R$, input
    history $H_{input} = \emptyset$, output history $H_{output} = \emptyset$
 3: for $round = 1 : c$ do
 4:     for $s \in \{s_{max}, s_{max} - 1, \dots, 0\}$ do
 5:         $n = \lceil \frac{B}{R} \frac{\eta^s}{(s+1)} \rceil$, $r = R\eta^{-s}$
 6:         T = PGSR_Sampling(n)
 7:         for $i \in \{0, \dots, s\}$ do
 8:             $n_i = \lfloor n\eta^{-i} \rfloor$
 9:             $r_i = r\eta^i$
10:             $L = \{f(t, r_i) : t \in T\}$
11:             $H_{input,r_i} \leftarrow H_{input,r_i} \bigcup T$
12:             $H_{output,r_i} \leftarrow H_{output,r_i} \bigcup L$
13:             $T = sort(T, L, \lfloor \frac{n_i}{\eta} \rfloor)$
14:         end for
15:     end for
16: end for
17: return Configuration with the smallest loss
```

Sub-algorithm - PGSR Sampling

```
18: Input: $H_{input}$, $H_{output}$, sparsity $s$, polynomial degree $d$, mini-
    mum observations $T$, randomness ratio $\rho$
19: if every $|H_{output,r}| < T$ then return random sample from
    original domain of $f$.
20: end if
21: Pick $H_{input,r}$ and $H_{output,r}$ with largest $r$: $|H_{output,r}| \geq T$.
22: Group Fourier basis based on hyperparameter structure.
23: Solve
    $$x^* = \arg\min_{\alpha} \frac{1}{2} \|y - \sum_{l=1}^{m} \Psi^l \alpha^l\|_2^2 + \lambda \sum_{l=1}^{m} \sqrt{p_l} \|\alpha^l\|_2$$
24: Let $S_1, \dots S_s$ be the indices of the largest coefficient of $\alpha$. Then,
    $g(x) = \sum_{i \in [s]} \alpha_{S_i} \chi_{S_i}(x)$ and $J = \bigcup_{i=1}^{s} S_i$
25: With probability $\rho$, return random sample from original domain
    of $f$; else return random sample from reduced domain of $f_{J,x^*}$.
```

Let R denote the (units of computational) resource to be invested in one round to observe the final performance of the model; η denote a scaling factor; and c the total number of rounds. Defining $s_{max} = \log_\eta R$, the total budget spent from SH is $B = (s_{max} + 1)R$. The algorithm samples n configurations with a sub-routine (which we call *PGSR-Sampling*, and explain further in the next section). Here, n is given by:

$$n = \lceil \frac{B}{R} \frac{\eta^s}{(s+1)} \rceil \tag{2.2}$$

and calculate the test loss with

$$r = R\eta^{-s} \tag{2.3}$$

epochs of training. The function $f(t, r_i)$ in Algorithm 1 (Line 10) returns the intermediate test loss of a hyperparameter configuration t with r_i of training epochs. Since the test loss is the metric to measure the performance of the model, the algorithm keeps only the top $\frac{1}{\eta}$ configurations (Line 13) and repeats the process by increasing the training epochs by the factor of η until r reaches to resource R. While SH introduces the new hyperparameter s, SH aggressively explores the hyperparameter space as s close to s_{max} while SH with s equal to zero is equivalent to random search (aggressive exploitation). The algorithm with one cycle contains $(s_{max} + 1)$ subroutines of SH attempting different levels of exploration and exploitation with all possible s values (Line 4).

2.2. PGSR Sampling

As PGSR-HB collects the outputs of the function f, the PGSR-Sampling sub-routine recovers Fourier basis coefficients of the Boolean function f using techniques from sparse recovery to reduce the hyperparameter space. Before we discuss about how PGSR Sampling works and compare differences with Polynomial Sparse Recovery in the Harmonica method of [20], we first establish some standard concepts in Fourier analysis of Boolean functions [21]. Consider a function f defined from $\{-1,1\}^n$ to $\mathbb{R}$. The Fourier basis corresponding to any subset of indices S (such that $S \subseteq [n]$) is defined as

$$\chi_S(x) = \prod_{i \in S} x_i \tag{2.4}$$

where x_i is the i^{th} element of the input vector. Then, the function f can uniquely expressed as the series of a real multilinear polynomial basis (or Fourier basis) given by:

$$f(x) = \sum_{S \subseteq [n]} \hat{f}(S)\chi_S(x) \tag{2.5}$$

where

$$\hat{f}(S) = \mathbb{E}_{x \in \{-1,1\}^n}[f(x)\chi_S(x)] \tag{2.6}$$

where the expectation is taken with respect to the uniform distribution over the nodes of the n-dimensional hypercube. The *restriction* [21] of the Boolean function f by a restriction pair (J, z) where $J \subseteq [n]$ and $z \in \{-1,1\}^J$ is denoted by the function $f_{J,z}$ over $n - |J|$ variables by fixing the variables in J to z.

While Harmonica does not explicitly address how to discretize continuous hyperparameters, we introduce a simple mathematical expression that efficiently induces additional sparsity in the Fourier representation of f. Let x be the m-digit binary number mapping to the set of integers with cardinality 2^m by function g, and y be the n-digits binary number mapping to the set of numbers with cardinality 2^n which are evenly spaced in (0,1] by function h. Then we express the i^{th} numerical hyperparameter value hp_i, for all k categories ($i = 1, \ldots, k$), in a log-linear manner as follows:

$$hp_i = 10^{g(x)} \cdot h(y) \tag{2.7}$$

Our experimental results section shows how this simple nonlinear binning representation induces sparsity on function g, which captures the value's order of magnitude. As PGSR returns the features regard to the function g, the new representation efficiently reduces the hyperparameter space. While PSR in Harmonica recovers the Boolean function with Lasso [22], the intuitive extension (arising from the above log-linear representation) is to replace sparse recovery with Group Lasso [23]; this is used in Algorithm 1 (Line 23) as we group them based on the g and h based on hyperparameter categories. Let $y \in \mathbb{R}^m$ be the observation vector; let the hyperparameters be divided into $m + n$ groups (corresponding to functions g and h) and let Ψ^l is the submatrix of $\Psi \in \mathbb{R}^{m \times \binom{m}{d}}$ where its columns match the l^{th} group. Similarly, α^l is a weight vector corresponding to the submatrix Ψ^l and p_l be the length of vector α^l. In order to construct the submatrices which are the collection of Fourier basis on its columns by the hyperparameter structure, let there exist a set of groups $G = \{g_1, \ldots, g_m, h_1, \ldots, h_n\}$ as defined above. If there are k possible combinations of groups from G such that a d-degree Fourier basis exists, we derive the k submatrices $\Psi^1, \ldots, \Psi^k$ using Eq. (2.4). Then the problem becomes equivalent to a convex optimization problem known as the Group Lasso, represented by the equation:

$$\min_{\alpha} \frac{1}{2}\|y - \sum_{l=1}^{m} \Psi^l \alpha^l\|_2^2 + \lambda \sum_{l=1}^{m} \sqrt{p_l}\|\alpha^l\|_2 \tag{2.8}$$

Lastly, the algorithm requires the input ρ which represents a reset probability parameter that produces random samples from the original reduced hyperparameter space. This parameter prevents gathering the biased observations in different PGSR stages, since the measurements with substantial resources mostly arise from the later stages of Successive Halving.

2.3. Differences between PGSR-HB and Harmonica

The standard Harmonica method samples the measurements under a uniform distribution before starting the search algorithm to recover the function f with PSR (the sparse recovery through l_1 penalty, or standard Lasso). Harmonica requires ML designers to choose the number of randomly sampled measurements and its resources (training epochs) before starting the search algorithm. The reliability of measurements, especially in the deep learning literature, hugely depends on the number of resources used on each sampled point. Investing enormous resources in recovering Fourier coefficients guarantees that the Lasso regression performs reliably, but this is inefficient with respect to total budget; however, collecting the measurements with small resources would make PSR fail to provide the correct guidance for the outer search algorithm. We have experimented with other penalties than the standard L1-penalty: for example, Tikhonov regularization prevents model overfitting particularly in deep architectures. However, the regularized regression tends to learn slower than the model without a regularization, consequently misleading the search algorithm with the worst performance. Since PGSR-HB gathers all the function outputs – from cheap resources to the most expensive resources – PGSR-HB eliminates the need to set an explicit number of samples and training epochs as in Harmonica.

The experimental results in [20] shows significant promise in finding the influential categorical hyperparameters such as presence/absence of the Batch-normalization layer, or determining the descent algorithm (stochastic gradient descent vs. Adam) [both of which can be represented using binary variables], but limitations in optimizing the numerical hyperparameters such as learning rate, weight decay l_2 penalty, and batch size. PGSR-HB overcomes this limitation of Harmonica with the log-linear representation capturing both order-of-magnitude and details in (2.7) and Group Lasso (2.8).

3. EXPERIMENTAL RESULTS

We verify the robustness of PGSR-HB by generating a test loss surface picking two hyperparameter categories as shown in Figure 1. We calculate the test loss by training 120 epochs with the standard benchmark image classification dataset, CIFAR-10. We used the convolutional neural network architecture from the cuda-convnet-82% model that has been used in previous work ([9] and [10]). We specifically choose the range of learning rate and the weight-decay penalty on the first convolutional layer to be from 10^{-6} to 10^2. We keep the log scale with base ten on both horizontal and vertical axis to visualize the loss surface with more natural interpretation and dynamic range on the test loss.

Table 1 compares the performance of PGSR and PSR with (2.7), and PSR with evenly spaced hyperparameter values in log scale. The

Table 1: Guidance Comparison on Learning Rate and Conv1 L2

Method	λ	Learn Rate	Conv1 Penalty
PGSR	0.5	$[\mathbf{10^{-3}}, \mathbf{10^{-2}}]$	$[\mathbf{10^{-5}}, \mathbf{10^{-4}}]$
PGSR	1.0	$[\mathbf{10^{-3}}, \mathbf{10^{-2}}]$	$[\mathbf{10^{-5}}, \mathbf{10^{-4}}]$
PGSR	2.0	$[\mathbf{10^{-3}}, \mathbf{10^{-2}}]$	$[\mathbf{10^{-5}}, \mathbf{10^{-4}}]$
PSR	0.5	$[\mathbf{10^{-3}}, \mathbf{10^{-2}}]$	$[10^{-6}, 10^{2}]$
PSR	1.0	$[10^{-4}, 10^{-3}]$	$[10^{-3}, 10^{-2}]$
PSR	2.0	$[10^{0}, 10^{2}]$	$[10^{-3}, 10^{-2}]$
PSR w/o (2.7)	0.5	$[10^{-4}, 10^{-2}]$	$[10^{-6}, 10^{-3}]$
PSR w/o (2.7)	1.0	$[10^{-4}, 10^{-2}]$	$[10^{-6}, 10^{-3}]$
PSR w/o (2.7)	2.0	$[10^{-4}, 10^{-2}]$	$[10^{-6}, 10^{-4}]$

third and fourth columns in Table 1 list the reduced hyperparameter space for learning rate and first convolution layer l2 penalty by each algorithm. The experiment result shows that (2.7) induces improved sparsity to reduce the space further than the conventional method. Giving extra information of the hyperparameter structure with grouping not only helped PGSR to return the correct guidance, but also provided the stability on the lasso coefficient λ as shown in the test loss surfaces (Figure 2 and Figure 3) with PGSR results in Table 1. More results of PGSR guidance with loss surfaces are in https://chomd90.github.io/.

Table 2: CNN Test Loss and Accuracy on CIFAR-10

Algorithm	RS 2x	SH	HB	PGSR-HB
Loss (I)	0.7118	0.7001	0.7150	$\mathbf{0.6455}$
Acc (I)	81.17%	79.69%	78.74%	$\mathbf{82.79\%}$
Loss (II)	0.6988	0.7179	0.6921	$\mathbf{0.6764}$
Acc (II)	79.51%	79.30%	81.67%	$\mathbf{83.00\%}$
Loss (III)	0.6850	0.6747	0.6960	$\mathbf{0.6467}$
Acc (III)	79.02%	79.80%	$\mathbf{81.47\%}$	80.39%
Loss (IV)	0.7293	$\mathbf{0.6499}$	0.7215	0.6619
Acc (IV)	77.70%	80.68%	80.81%	$\mathbf{81.64\%}$

Next, we optimize the five categories of hyperparameters including the learning rate, three convolution layers' and a fully connected dense layer's Tikhonov regularization constants using the same architecture and dataset used in the previous section. We trained the network using the stochastic gradient descent without a momentum and included the learning rate decay by a factor 0.1 every 100 epochs of training. We compare SH, Hyperband, Random Search with doubled budgets and PGSR-HB based on test loss and accuracy. We set the resource $R = 243$ and the discard ratio input $\eta = 3$ and allocated the equivalent total budget between the algorithm based on the training epochs except for Random Search 2x. Setting the total budget of four cycles of Hyperband and PGSR-HB as the baseline, Random Search 2x evaluates 288 randomly sampled hyperparameter configurations with the resource R and SH cycles 24 times as one Hyperband contains six subroutine SH. Since the randomness involves in these hyperparameter optimization algorithms, we compare four different trials of each algorithms as shown in Table 2. The experiment result verifies the effectiveness of reducing the hyperparameter space through PGSR as the new algorithm returns better performance for most of the trials. Moreover, PGSR-HB found the optimal hyperparameters returning 83% test accuracy which outperforms the other algorithms from all trials.

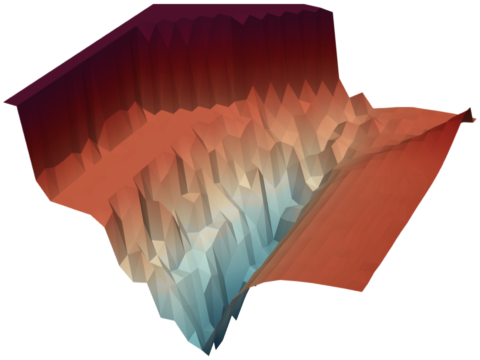

Fig. 1: *Test loss surface with two hyperparameters. Learning rate vs conv1 l2 penalty.*

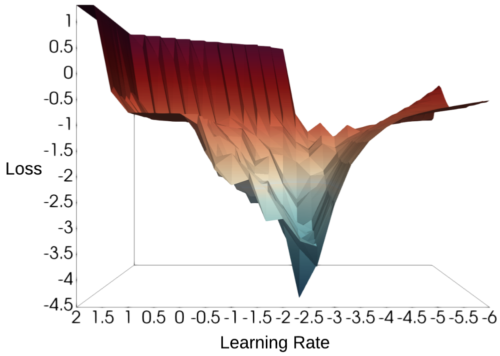

Fig. 2: *The view from learning rate axis.*

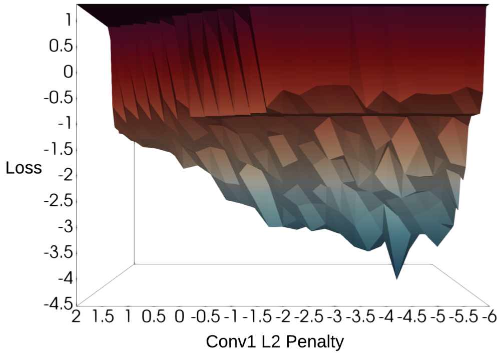

Fig. 3: *The view from conv1 l2 penalty axis.*

4. CONCLUSION

We proposed a new HPO algorithm which learns the most influential hyperparameters by carefully tracking loss function (measurement) history in a Hyperband framework. Our new algorithm is based on a key modification of polynomial sparse recovery (PSR) that induces further improvement via a group-sparsity constraint. Future directions include performing a multi-stage Group Lasso to reduce hyperparameter space further as we obtain new observations. While the goal of the HPO problem is to approximate the global minimizer of the loss over hyperparameter space, HPO methods themselves require tuning, so a fully automatic ML training method is still of great interest.

5. REFERENCES

[1] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl, “Algorithms for hyper-parameter optimization,” in *Advances in neural information processing systems*, 2011, pp. 2546–2554.

[2] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown, “Sequential model-based optimization for general algorithm configuration,” in *International Conference on Learning and Intelligent Optimization*. Springer, 2011, pp. 507–523.

[3] Jasper Snoek, Hugo Larochelle, and Ryan P Adams, “Practical bayesian optimization of machine learning algorithms,” in *Advances in neural information processing systems*, 2012, pp. 2951–2959.

[4] Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown, “Auto-weka: Combined selection and hyperparameter optimization of classification algorithms,” in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2013, pp. 847–855.

[5] Katharina Eggensperger, Matthias Feurer, Frank Hutter, James Bergstra, Jasper Snoek, Holger Hoos, and Kevin Leyton-Brown, “Towards an empirical foundation for assessing bayesian optimization of hyperparameters,” in *NIPS workshop on Bayesian Optimization in Theory and Practice*, 2013, vol. 10, p. 3.

[6] Jasper Snoek, Kevin Swersky, Rich Zemel, and Ryan Adams, “Input warping for bayesian optimization of non-stationary functions,” in *International Conference on Machine Learning*, 2014, pp. 1674–1682.

[7] Ilija Ilievski, Taimoor Akhtar, Jiashi Feng, and Christine Annette Shoemaker, “Efficient hyperparameter optimization for deep learning algorithms using deterministic rbf surrogates.,” in *AAAI*, 2017, pp. 822–829.

[8] James Bergstra and Yoshua Bengio, “Random search for hyper-parameter optimization,” *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.

[9] Kevin Jamieson and Ameet Talwalkar, “Non-stochastic best arm identification and hyperparameter optimization,” in *Artificial Intelligence and Statistics*, 2016, pp. 240–248.

[10] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar, “Hyperband: A novel bandit-based approach to hyperparameter optimization,” *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6765–6816, 2017.

[11] Manoj Kumar, George E Dahl, Vijay Vasudevan, and Mohammad Norouzi, “Parallel architecture and hyperparameter search via successive halving and classification,” *arXiv preprint arXiv:1805.10255*, 2018.

[12] Jiazhuo Wang, Jason Xu, and Xuejun Wang, “Combination of hyperband and bayesian optimization for hyperparameter optimization in deep learning,” *arXiv preprint arXiv:1801.01596*, 2018.

[13] Stefan Falkner, Aaron Klein, and Frank Hutter, “Bohb: Robust and efficient hyperparameter optimization at scale,” *arXiv preprint arXiv:1807.01774*, 2018.

[14] Hadrien Bertrand, Roberto Ardon, Matthieu Perrot, and Isabelle Bloch, “Hyperparameter optimization of deep neural networks: Combining hyperband with bayesian model selection,” .

[15] Yoshua Bengio, “Gradient-based optimization of hyperparameters,” *Neural computation*, vol. 12, no. 8, pp. 1889–1900, 2000.

[16] Dougal Maclaurin, David Duvenaud, and Ryan Adams, “Gradient-based hyperparameter optimization through reversible learning,” in *International Conference on Machine Learning*, 2015, pp. 2113–2122.

[17] Jelena Luketina, Mathias Berglund, Klaus Greff, and Tapani Raiko, “Scalable gradient-based tuning of continuous regularization hyperparameters,” *arXiv preprint arXiv:1511.06727*, 2015.

[18] Jie Fu, Hongyin Luo, Jiashi Feng, Kian Hsiang Low, and Tat-Seng Chua, “Drmad: Distilling reverse-mode automatic differentiation for optimizing hyperparameters of deep neural networks,” *arXiv preprint arXiv:1601.00917*, 2016.

[19] Luca Franceschi, Michele Donini, Paolo Frasconi, and Massimiliano Pontil, “Forward and reverse gradient-based hyperparameter optimization,” *arXiv preprint arXiv:1703.01785*, 2017.

[20] Elad Hazan, Adam Klivans, and Yang Yuan, “Hyperparameter optimization: a spectral approach,” *arXiv preprint arXiv:1706.00764*, 2017.

[21] Ryan O’Donnell, *Analysis of boolean functions*, Cambridge University Press, 2014.

[22] Robert Tibshirani, “Regression shrinkage and selection via the lasso,” *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[23] Ming Yuan and Yi Lin, “Model selection and estimation in regression with grouped variables,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, no. 1, pp. 49–67, 2006.