AFA: Auto-tuning Filters for Ads

Joobin Gharibshah
eBay Inc.
San Jose, California, USA
jgharibshah@ebay.com

Mahmuda Rahman
eBay Inc.
San Jose, California, USA
mahrahman@ebay.com

Abraham Bagherjeiran
eBay Inc.
San Jose, California, USA
abagherjeiran@ebay.com

ABSTRACT

Tuning filters to refine Ads eligibility to surface in search results emerges as a pivotal problem. It often necessitates a nuanced approach to cater to diverse requirements from the customers. Adjusting these filters must judiciously balance the preferences of both advertisers and users in the online marketplace. Hence, it requires a multi-objective optimization which often turns out to be hard due to the conflicting nature of the objectives from these customers. In this paper we present AFA: **A**uto-tuning **F**ilters for **A**ds - a novel application of Bayesian Optimization for auto-tuning these filters. We specifically develop AFA to employ a probabilistic model to navigate the intricate trade-offs between multiple objectives. It iterates over a feasible solution space and quickly converges to an operating point which ensures showing well performing ads while increasing their scale. This offers a substantial advancement in the automation for digital advertising campaigns. Our approach significantly reduces the reliance on manual adjustments and expensive A/B testing, as demonstrated by empirical results from a large-scale e-commerce platform.

CCS CONCEPTS

• **Information systems** → **E-commerce infrastructure**; **Computational advertising**; • **Applied computing** → **Online auctions**.

KEYWORDS

Bayesian Optimization, Surrogate Model, Acquisition Function, Automated Pipeline

ACM Reference Format:

Joobin Gharibshah, Mahmuda Rahman, and Abraham Bagherjeiran. 2024. AFA: **A**uto-tuning **F**ilters for **A**ds . In *Proceedings of August 26, 2024 (AdKDD '24).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Sponsored search is an advertising model used by search engines to display paid advertisements alongside organic search results. When a user enters a query, the search engine runs an auction among advertisers who have bid on keywords relevant to the user's search terms. The winners of this auction have their ads displayed in prominent positions on the search results page, typically marked as "Sponsored" or "Ad" to distinguish them from other content.

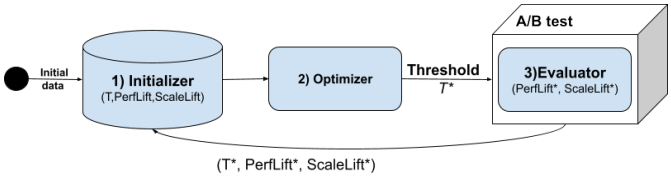


Figure 1: A visual overview of AFA pipeline to auto tune the quality filter: (1) Observed data points consist of thresholds and metrics lift (set of values for each objective function on that given threshold) denoted as $< T, PerformanceLift(T), ScaleLift(T) >$ respectively, initial data points collected based on random thresholds from a feasible solution space. (2) Optimizer fits surrogate function on the observed data points and produce T^* as the next threshold (T*) to be evaluated (3) Evaluator measures the impact of the new threshold T^* on the objective value and produce another triple $< T^*, PerformanceLift(T^*), ScaleLift(T^*) >$ to be appended to the stored observed data points so that we can explore next point based on that

While these advertisements (a.k.a. ads) benefit advertisers by increasing the visibility of their products to potential users, maintaining user satisfaction is also important for the e-commerce platform to monetize the search from showing ads. Thereby, it is crucial to ensure that the participating ads are of high quality to comply with user's desirability. Generally, ad platform employs a variety of quality filters controlled by hyperparameters [1]. Its core purpose is to efficiently navigate through extensive datasets, find content that aligns closely with the user's search intent and personal preferences by providing more relevant outcomes.

The paper presents a innovative approach for automatically tuning quality filters in the search monetization domain using Bayesian Optimization. The approach aims to optimize the quality threshold to balance scale improvement and ad performance. An automated pipeline was developed to minimize human effort, time, and errors in this process.

1.1 Challenges

E-commerce platforms face the challenge of setting the right quality threshold for ads while ensuring sufficient ad exposure to increase scale. This issue arises from the differing needs of two main stakeholders: advertiser and user

1.1.1 Advertiser's preferences. Advertisers enlist their items on e-commerce marketplace to sell and expect greater visibility of their

[1] https://support.google.com/google-ads/answer/6167130?hl=en accessed on 07/11/2024

AdKDD '24, Barcelona, Spain,

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

product in exchange for paying more to the platform. As they often prioritize visibility over the quality of their product, the platform needs to maintain a quality filter that strikes a balance between their need for exposure for the product (measured by scale) and its relevance to the user's preferences.

1.1.2 User's preferences. The e-commerce platform is also committed to meeting the essential needs of user who uses this platform to find high quality product based on their search queries as input. It is the platform's responsibility to provide them with relevant and high quality search results for their queries. To achieve this, an appropriate quality filter is required to surface products in the search results that are most relevant to the user's query. User's preference is calculated as the ratio of the click over number of impressions they have (to measure filter's performance).

This dilemma between the preferences of users and advertisers presents a challenging problem in e-commerce platforms. On one hand, they need to increase the quality threshold to meet users' expectations of obtaining more relevant items. On the other hand, they must keep the quality filter to a degree where the platform can accommodate advertisers' appetite for increased visibility. Therefore they aim to enhance ad scales without compromising the performance.

1.2 Motivation

The dynamic nature of the marketplace, characterized by continuous growth, frequent updates, and new feature roll outs, further complicates the task by rendering the static thresholds obsolete in short order. Manual tuning of filters is not only time-intensive but also lacks efficiency, underscoring the need for an automated, scalable solution. Main inspirations for this work are:

1.2.1 Need for a Formal Definition. Capturing and quantifying the intricate correlations between two different objectives (Scale and Performance), which often exhibit complex interactions, is a hard task. The absence of a formal function to encompass both the objectives, exacerbates this issue and hinders the systematic exploration of the trade-offs inherent to the objectives. Consequently, without a clear mathematical framework to navigate the multi-dimensional objective space, manual optimization becomes not only cumbersome but also prone to sub-optimal decision-making, as it relies on intuition rather than analytical precision. This emphasize the necessity for a structured approach that can effectively balance these competing goals and facilitate the discovery of an optimal solution that results in a desired outcomes.

1.2.2 Opportunity Cost. As a common practice, multiple experiments covering various operating points and parameters via grid or random search [1] are utilized to find a proper thresholds for filters. This manual process is expensive and time-consuming which often needs to be repeated whenever there is a change in the environment. On an e-commerce platform, the vast number of advertisements and transactions necessitates that modifications are made with accuracy and speed to stay attuned in the market. As the customer tastes and industry tendencies shift quickly, it leaves only a brief time window to detect and react to these developments successfully. Delayed tuning can lead to outdated search results, diminishing user experience and potentially leading to a loss in sales, customer trust and platform's reputation.

1.3 System Overview

Our proposed system employs a probabilistic model that captures the complex relationship between the quality threshold and the multi-objective function, which includes both performance and scale lift. By utilizing Bayesian optimization, we iteratively update the quality filter settings, efficiently navigating the search space to identify the optimal threshold that satisfies the dual objectives as demonstrated in Figure 1. This approach not only accounts for the inherent uncertainty in user behavior but also reduces the need for extensive manual tuning and A/B testing.

To this end, we introduce AFA as an auto filter tuning pipeline for ads to tune the quality filter. AFA consists of three major steps that we illustrated in the figure 1 and we introduce them here.

1.3.1 Initializer is a process to collect and store data points observed over the course of running our system. We store a triplet $(\theta, Obj_1(\theta), Obj_2(\theta))$ for each filter threshold θ that we explore. This triplet has a threshold along with two corresponding objective values, i.e., scale lift and performance lift. The pipeline starts with some initial data points which are collected via running A/B tests over various thresholds within the exploration range to give the process a warm start.

1.3.2 Optimizer utilizes a Bayesian optimizer to solve multiple objectives consisting of scale and performance. This optimizer will read initial data points i.e $(\theta, Obj_1(\theta), Obj_2(\theta))$ and build a surrogate model based on those data points. Then it utilizes an acquisition function [7] to suggest a new threshold θ^* which is the next optimal point to be evaluated by *Evaluator*.

1.3.3 Evaluator is the component which runs an A/B test to evaluate the objective function with respect to the suggested threshold. Then, we compute a new triple as $(\theta^*, Obj_1(\theta^*), Obj_2(\theta^*))$ which consists of the suggested threshold θ^* by the optimizer and corresponding objectives values evaluated by the evaluator. This newly explored data point is then feedback to the initilizer.

Thus, the pipeline constantly updates the quality threshold to respond to new changes, ensuring that search results remain relevant.

1.4 Contributions

This paper presents several key contributions:

- We formalize the process of general filter tuning problem by leveraging a Bayesian optimization method in AFA, considering the need of multiple competing stakeholders.
- AFA provides a fast-converging approach that significantly reduces human effort and time for tuning quality threshold.
- AFA has been successfully deployed within an e-commerce platform, demonstrating its effectiveness in refining quality filters for the search engine and attesting to its scalability in large-scale industry settings.

The structure of the paper is as follows: Section 2 describes implementation details in the optimization process and evaluation metrics. Section 3 analyzes the results of AFA. Section 4 reviews

related literature. Section 5 concludes with a summary and future research directions.

2 IMPLEMENTATION DETAILS

In the pipeline presented in Figure 1, there are three components as we explained in section 1.3. In this section, we focus on the *Optimizer* and provide more details regarding its implementation and evaluation.

2.1 Optimization Process

In this section, we describe our objective function and its formulation.

Objective function In AFA, we aim to find a threshold value for quality filter that satisfies two objectives related to advertisers and users needs: maintaining performance lift and a positive scale lift in ads. The lift amounts calculated for this purpose are all relative differences between the suggested value and the current value in the system as AFA updates the threshold in each iteration. (more detail in section 3 and equation 14).

We formulate these objectives as follows:

Increasing scale: AFA looks for quality thresholds which increase the ad scale in search result by:

$$\mathbb{C} \in \underset{t \in \text{Thresholds}}{OA6{<}OG} \; X_{s(t)} \tag{1}$$

where $X_{s(t)}$ represents ad scale changes.

Maintaining performance: AFA also looks for that quality thresholds to minimize performance lift via:

$$\mathbb{C} \in \underset{t \in \text{Thresholds}}{OA6{<}OG} \; -|X_{p(t)}| \tag{2}$$

subject to the constraint $X_{p(t)}$ $\ddot{Y}$. where $X_{p(t)}$ represents performance changes.

We combine these two objective in one as follows:

$$\mathbb{C} = \underset{t \in \text{Thresholds}}{\text{argmax}} \; X_{s(t)} - X_{p(t)} \tag{3}$$

subject to the constraint $X_{p(t)}$ $\ddot{Y}$. In this function, we apply the same weight to both objectives, although these weights may vary in different scenarios.

To impose the constraint on the objective function we employ a penalty and reward mechanism. We penalize and reward the objective function when the explored thresholds fails and succeeds to maintain the constraint respectively. Thereby, we formulate the optimization problem as follows:

$$5(\mathbb{C}) = \mathbb{C} = \underset{t \in \text{Thresholds}}{OA6{<}OG} \; X_{s(t)} - X_{p^*(t)} \tag{4}$$

where the modified performance will be defined as:

$$X_{p^*(t)} = \begin{cases} '(\mathbb{C}) * X_{p(t)} & - \leq X_{p(t)} \; \ddot{Y} \\ \%(\mathbb{C}) * X_{p(t)} & >\mathbb{C} \; 4AF8B4 \end{cases} \tag{5}$$

Here, $\%(\mathbb{C})$ and $'(\mathbb{C})$ are the amount of penalty and reward we apply on the %4A5>A<O=24!85C.

This generic formulation can accommodate additional objectives and constraints, as well as custom-defined penalty and reward values, according to business requirements. Although we have defined the constraint in closed-form, we have not devised a closed-form definition for the objective components that would allow us to use the Lagrange multiplier method. However, we discuss related work and alternative approaches in Section 4.

In our pipeline, at each iteration, the *Optimizer* solves for the objective function defined in Equation 4 by using a Bayesian optimization approach, as we describe below.

Bayesian Optimization employs a probabilistic model to represent the uncertainty about the objective function's behavior, and it updates this model iteratively using Bayes' Theorem as new data points are observed.

Bayes' Theorem is formulated as follows:

$$\%(\backslash|\text{data}) = \frac{\%(\text{data}|\backslash)\%(\backslash)}{\%(\text{data})} \tag{6}$$

where: $\%(\backslash)$ is the prior probability of the hypothesis before seeing the data. $\%(\text{data}|\backslash)$ is the likelihood of the data under the hypothesis. $\%(\text{data})$ is the marginal likelihood or evidence, the probability of the data under all possible hypotheses. $\%(\backslash|\text{data})$ is the posterior probability of the hypothesis after seeing the data. In our case, $\backslash$ is representing the threshold

Based on this Bayes' Theorem, we build a surrogate model which will rely on the observed data points (thresholds) and make a prediction for unobserved data points. The surrogate model is as follows:

a) Surrogate model is a probabilistic model used to approximate the unknown objective function $5(G)$ that we wish to optimize. The surrogate model, denoted as $\hat{5}(G)$, is used to predict the output of $5(G)$ given new inputs G, and to estimate the uncertainty of that prediction.

The surrogate model we used is a Gaussian Process (GP), which is defined by a mean function $`(G)$ and a covariance function (kernel) $:(G{\bullet}G')$. The GP surrogate model for any input point G is:

$$\hat{5}(G) \sim \mathcal{GP}(`(G){\bullet}:(G{\bullet}G')) \tag{7}$$

This formulation allows the Bayesian optimization algorithm to not only predict the function value at unobserved points but also quantify the prediction uncertainty, which is crucial for balancing exploration and exploitation during the optimization process. In our GP, we used a White Noise as a kernel function in Bayesian optimization[11], This kernel function is defined to represent the idea that observations have some amount of uncorrelated noise.

The White Noise kernel is defined as:

$$:(G{\bullet}G') = f_n^2 \Delta(G{\bullet}G') \tag{8}$$

where f_n^2 is the noise variance, a hyper-parameter that represents the variance of the noise in the observations. $\Delta(G{\bullet}G')$ is the Kronecker delta function, which equals 1 if (x = x') (i.e., the points are identical) and 0 otherwise [11]. The White Noise kernel is just one possible choice among many kernels for GPs

b) Acquisition function is a function that guides the optimization process by determining where to sample next. The acquisition function balances exploration of the search space (sampling where the model is uncertain) with exploitation (sampling where the model predicts high performance).

Two common acquisition functions which we used in AFA are:

1) Expected Improvement (EI): This function measures the expected amount of improvement over the current best observation

$5(G^+)$ at a new point G.

$$(G) = \mathbb{E}\ \max(5(G) - 5(G^+)\cdot 0) \tag{9}$$

where $5(G)$ is the objective function and $5(G^+)$ is the best observed value so far.

2) Probability of Improvement (PI): This function measures the probability that sampling at a new point G will lead to an improvement over the current best observation $5(G^+)$.

$$\%\ (G) = \%(5(G)\ ¡\ 5(G^+)) \tag{10}$$

The selected acquisition function will generate a new threshold, which will be passed to the *evaluator* for testing in the next step. To ensure fast convergence we adopted *PI*. We explain the evaluation process in Section 2.2.

2.2 Evaluation Metrics

We need to evaluate the optimizer's efficiency using the *Evaluator* through an A/B test. We execute this phase of the pipeline at the conclusion of each iteration, which, in our case, spans a period of six days. We set the recommended threshold by AFA as a treatment in an A/B test and get the real-time impact on both the scale and performance since metrics like performance are dependent to user behaviour and is not possible to be computed offline precisely. The objectives calculated by AFA are:

Scale is calculated as the ratio of the total number of times ads are displayed to the number of qualified queries (i.e. buyer searches) that trigger the ads, expressed by the formula:

$$(20;4 = \frac{\#D{<}14A{>}5\ C8{<}4B\quad 3\quad 8B?;O{\sim}43}{\#D{<}14A{>}5\ \&DO;85843\ \&D4A84B}\text{"} \tag{11}$$

Performance is measured as the proportion of clicks an advertisement receives relative to the number of times it is shown (impressions), represented by the formula:

$$\%4A5{>}A{<}O{=}24 = \frac{)\,{>}CO;\quad ;82:B}{)\,{>}CO;\quad {<}?A4BB8{>}{=}B}\text{"} \tag{12}$$

We introduce quality and revenue here as additional output metrics in order to track business impacts.

Revenue is the income earned from displaying ads on a platform. It can be calculated as:

$$'4E4{=}D4 = \text{Buyers' action (i.e clicks)} \times \text{Revenue per action} \tag{13}$$

Quality is the indicator of the relevance of the ads with respect to query.

The amount of lift in an A/B test is calculated by comparing the evaluation metric (e.g., scale) between the treatment group (T), $"4CA82_T$, and the control group (C), $"4CA82_C$, using the formula

$$!85C = \frac{("4CA82_T - "4CA82_C)}{"4CA82_C} \times 100\% \tag{14}$$

The amount of lifts will be used by the Optimizer and also stored in the table introduced at Section 2 for the ongoing exploration.

3 EXPERIMENT RESULTS

Analyzing the outcomes of filter tuning to meet multiple objectives simultaneously, is not a trivial task. It requires a deep and precise understanding of the interplay between different objectives. However, in our algorithm, we have formalized this relationship in a robust manner, allowing measurable and accurate improvements. In this section, we discuss our experimental results.

3.1 Execution Efficiency

Our experiments demonstrate that AFA could achieve the defined objectives within three iterations through our tuning pipeline. Figure 2 illustrates how the model evolved over multiple iterations.

One of the standout results of using AFA for automatic filter tuning is its marked efficiency in both time and human effort. In a comparative analysis between manual tuning and AFA-assisted tuning, we observed a significant reduction in the number of iterations required to optimize the quality filter. Manual efforts necessitated 12 iterations to achieve satisfactory results, whereas AFA achieves comparable improvements in the ad scale metric and maintained performance within just 3 iterations.

Moreover, the total number of data points needed for AFA was halved, with only 6 data points (including 3 initial data points) compared to the 12 required for manual tuning. This reduction in data points translates directly into savings extensive AB testing spanning for multiple weeks as well as post test analysis efforts. AFA autonomously computes the next threshold to test, minimizing the need for manual evaluation.

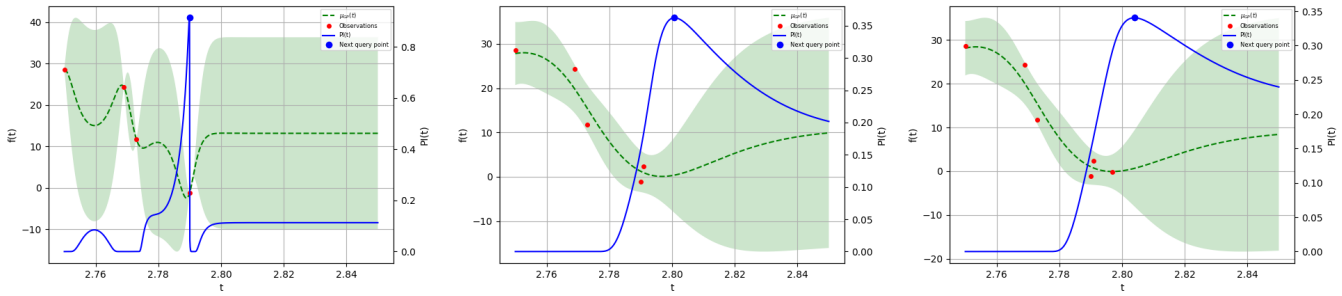
Time efficiency was also significantly improved. Each data point in the AFA corresponded to outcome of one week of experimentation using our A/B testing platform, leading to an optimization timeline of just 4 weeks. This consisted of a week of initial data collection followed by 3 weeks of iterations. Conversely, the manual approach spanned over 12 weeks, with each iteration taking one week - yet resulting in a sub-optimal solution for the problem.

In summary, the deployment of AFA for quality filter tuning within our experimental framework yielded a 4X increase in efficiency in terms of both time and human effort. This demonstrates AFA's potential to accelerate the tuning process, ultimately reducing the laborious and time-intensive nature of manual filter tuning.

3.2 Business Targets

In this section, we present the amount of improvement achieved by employing AFA in our real world production platform, targeting a large online user population over a two-week experiment period.

We compared two thresholds: the recommended threshold from AFA, which converged after three iterations via an automated pipeline, and the best threshold suggested by manual tuning, which required human adjustment over twelve iterations. As indicated in Table 1, both methods improved the scale, but AFA showed a higher impact on business metrics. AFA not only improves the quality score 80% but also we achieved a 0.72% higher performance compared to manual tuning, resulting in a mere 0.18% performance loss when using AFA, as opposed to a 0.66% performance loss with the manual approach. Considering that performance was our constraint during the optimization phase, a smaller loss signifies a better operating point. Moreover, we observed a 44% improvement in ad revenue with AFA; manual tuning resulted in a 0.22% loss in revenue, while AFA managed a 0.01% gain. We also observed a startling gain of 80% quality as well. In summary, our approach enabled the display of more ads without sacrificing performance. Furthermore, it improved quality and revenue which are key business metrics.



(a) Surrogate model and acquisition function after feeding initial data points (b) Surrogate model and acquisition function after second iteration (c) Surrogate model and acquisition function after third iteration

Figure 2: Changes in the surrogate model and acquisition function over three iterations. The dashed green line represents the surrogate model fitted to the red dot observations. The green shade indicates the uncertainty for each threshold based on the surrogate function. The blue line represents the acquisition function, and the blue dot marks the next point suggested for testing, based on the maximum value of the acquisition function that optimizes the defined objective.

Table 1: Comparing business metrics lift results between manual filter tuning vs AFA tuning approa ch

Method	Scale	Performance	Quality	Revenue
Manual Tuning	**0.79%**	-0.66%	0.05%	-0.22%
AFA	**0.88%**	**-0.18%**	**0.09%**	**0.01%**
AFA vs Manual Tuning	**11%**	**72%**	**80%**	**44.0%**

In Figure 3, we provide an empirical example illustrating the search results page for a buyer query "leather jacket". Using the threshold recommended by AFA, we were able to display more sponsored ads compared to the threshold determined by manual tuning. Specifically, AFA enabled the presentation of four sponsored ads, whereas the manual tuning approach yielded only two. This empirical evidence supports our claim that AFA fine-tunes the threshold more precisely than manual methods, thereby improving ad scale. Additionally, we observed that all sponsored items were relevant and of high quality, which correlates with the improved performance and ad revenue as detailed in Table 1.

4 RELATED WORK

There is limited research on automatic filter tuning in ads industry, and even fewer studies on pipelines to facilitate the process. Prior works can be categorized in the following categories.

Black box optimization (BBO) methods are essential for optimizing functions without closed-form expressions, which are often encountered in real-world scenarios involving complex systems. Evolutionary algorithms (EAs) and genetic algorithms (GAs), such as those described by [5], have been fundamental in exploring search spaces in a gradient-free manner. While effective, these methods can require a prohibitively large number of evaluations to converge, which is not always practical [6].

To overcome the limitations of traditional EAs and GAs, surrogate-based optimization (SBO) techniques have been developed. Bayesian optimization was introduced by [7] as Efficient Global Optimization (EGO), utilizing surrogate models to approximate objective functions. Bayesian Optimization (BO), a subset of SBO, has gained traction for its sample efficiency and effectiveness in noisy evaluations, as highlighted by [2].

The incorporation of Bayesian optimization into multi-objective optimization (MOO) has been an area of active research. [3] presented a framework for efficiently optimizing black-box functions with multiple objectives. Bayesian optimization and MOO used to improve their recommendation models in feeds and notification [9].

Multi-objective optimization (MOO) addresses complex problems where multiple, often conflicting, objectives must be optimized simultaneously. Researchers like [4] have significantly contributed to this field with algorithms such as NSGA-II, which efficiently guide the search towards Pareto-optimal solutions under constraints such as limited evaluations. The extension of BBO principles to MOO has enabled the application of these techniques in various domains, including e-commerce, where balancing trade-offs is crucial.

Recent work in MOO has focused on improving the efficiency and scalability of these algorithms. For instance, [12] offers a robust approach for handling many-objective problems by introducing NSGA-III. Additionally, the integration of machine learning models, as seen in work on the SPEA2 algorithm, has improved the handling of complex objective landscapes [13]. However, most of these works focus on the closed form of objective functions.

Hyper-parameter tuning: Hyper-parameter tuning is a critical step in machine learning that involves selecting the optimal set of hyper-parameter for a learning algorithm to maximize its performance. Hyper-parameter are the configuration settings used to structure the learning process, as opposed to model parameters that are learned from the data. There are several work focusing on search mechanism like grid and random search which are the simplest and most commonly used approaches [1]. There are other efforts on Gradient-Based Optimization which uses gradient information to guide the search for optimal hyper-parameters [10]. There are bandit based approaches which dynamically allocates resources to a set of hyper-parameter configurations and rapidly eliminates poor-performing options [8].