Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

Optimizing contextual bandit hyperparameters: A dynamic transfer learning-based framework

Farshad Seifi[a*] and Seyed Taghi Akhavan Niaki[a]

[a]*Department of Industrial Engineering, Sharif University of Technology, Tehran, Iran*

C H R O N I C L E

Article history:
Received May 2 2024
Received in Revised Format May 30 2024
Accepted June 20 2024
Available online June 20 2024

Keywords:
Hyperparameter Optimization
Contextual Bandit
Transfer Learning
Bayesian optimization

A B S T R A C T

The stochastic contextual bandit problem, recognized for its effectiveness in navigating the classic exploration-exploitation dilemma through ongoing player-environment interactions, has found broad applications across various industries. This utility largely stems from the algorithms' ability to accurately forecast reward functions and maintain an optimal balance between exploration and exploitation, contingent upon the precise selection and calibration of hyperparameters. However, the inherently dynamic and real-time nature of bandit environments significantly complicates hyperparameter tuning, rendering traditional offline methods inadequate. While specialized methods have been developed to overcome these challenges, they often face three primary issues: difficulty in adaptively learning hyperparameters in ever-changing environments, inability to simultaneously optimize multiple hyperparameters for complex models, and inefficiencies in data utilization and knowledge transfer from analogous tasks. To tackle these hurdles, this paper introduces an innovative transfer learning-based approach designed to harness past task knowledge for accelerated optimization and dynamically optimize multiple hyperparameters, making it well-suited for fluctuating environments. The method employs a dual Gaussian meta-model strategy—one for transfer learning and the other for assessing hyperparameters' performance within the current task —enabling it to leverage insights from previous tasks while quickly adapting to new environmental changes. Furthermore, the framework's meta-model-centric architecture enables simultaneous optimization of multiple hyperparameters. Experimental evaluations demonstrate that this approach markedly outperforms competing methods in scenarios with perturbations and exhibits superior performance in 70% of stationary cases while matching performance in the remaining 30%. This superiority in performance, coupled with its computational efficiency on par with existing alternatives, positions it as a superior and practical solution for optimizing hyperparameters in contextual bandit settings.

1. Introduction

The stochastic contextual bandit problem presents a classic exploration-exploitation dilemma within a continuous interaction framework between a player and their environment. In each stage of this iterative process, the player chooses one arm to pull from a set, with each arm characterized by an N-dimensional vector representing contextual features (Ding et al., 2022). The outcome of this choice is a stochastic reward specific to the selected arm, which is the only feedback provided to the player. The overarching objective for the player is to either maximize the total accumulated reward over time or minimize the overall regret experienced. This scenario necessitates a strategic balance by the player: on one hand, exploiting by repeatedly choosing the arm believed to offer the highest reward based on available data, and on the other, exploring by testing other arms that potentially could yield greater rewards, despite the uncertainty surrounding their outcomes. Given the contextual bandit's adeptness at navigating the exploration-exploitation trade-off, it has found extensive applications across diverse fields such as recommender systems (Lihong Li, Chu, Langford, & Schapire, 2010), online advertising (Schwartz, Bradlow, & Fader, 2017), clinical trials (Woodroofe, 1979), and personalized medicine (Bastani & Bayati, 2020) among others. This widespread

* Corresponding author
E-mail farshad.seifi@ie.sharif.edu (F. Seifi)
ISSN 1923-2934 (Online) - ISSN 1923-2926 (Print)
2024 Growing Science Ltd.
doi: 10.5267/j.ijiec.2024.6.003

adoption can be attributed to the bandit algorithms' proficiency in accurately predicting reward functions while effectively managing the exploration-exploitation balance. The efficacy of these algorithms in achieving their objectives is significantly influenced by the selection and tuning of hyperparameters. Such parameters, including the exploration rate α, the regularization parameter λ in ridge regression, and the array of parameters tied to the NeuralUCB algorithm (Zhou, Li, & Gu, 2020) like network width, depth, learning rate, and momentum, are crucial determinants of a bandit model's performance. These hyperparameters directly impact the algorithm's ability to accurately predict rewards and judiciously balance the act of exploring new options against exploiting known ones. Consequently, the quest to optimize these hyperparameters is not just a technical challenge but a fundamental necessity to harness the full potential of contextual bandit models. Optimizing hyperparameters ensures that these models can perform optimally across a range of dynamic environments, thereby maximizing cumulative rewards and minimizing overall regret in a scientifically robust manner.

However, the dynamic and real-time nature of bandit environments poses a significant challenge for hyperparameter tuning, making traditional offline methods like cross-validation less effective. To navigate these challenges, specialized approaches have been developed. Greedy strategies, as explored by Bouneffouf (Bouneffouf, 2016) and Bastani et al. (Bastani, Bayati, & Khosravi, 2021), alongside the MÊLÉE algorithm—a meta-learning approach by Sharaf and Daumé (Sharaf & Daumé III, 2019) —and bandit-over-bandit frameworks such as the OPLINUCB and DOPLINUCB algorithms (Bouneffouf & Claeys, 2020) and the Continuous Dynamic Tuning (CDT) framework (Kang, Hsieh, & Lee, 2023), offer tailored solutions for hyperparameter optimization in these settings. Despite the innovative nature of these solutions, they are not without their limitations. Greedy strategies, for example, may not efficiently utilize available information, potentially requiring additional evaluations and compromising performance. The MÊLÉE algorithm, while promising, may struggle with the adaptive learning of exploration parameters in the constantly changing contexts of live bandit environments. This could render it less effective where adaptability is crucial. Additionally, while the OPLINUCB and DOPLINUCB algorithms excel in optimizing individual hyperparameters, their effectiveness diminishes when multiple hyperparameters need simultaneous optimization. The CDT framework attempts to overcome this limitation but faces challenges in data efficiency, especially when compared to meta-learning approaches like MÊLÉE.

To address the above-mentioned challenges, this paper introduces a novel transfer learning-based approach for contextual bandit hyperparameter optimization. This proposed algorithm capitalizes on the insights derived from similar past contextual bandit tasks, utilizing Bayesian optimization to enhance data efficiency and accelerate the achievement of the optimal solution. Furthermore, by incorporating two Gaussian meta-models that operate in tandem—one focused on transfer learning from past tasks and the other on analyzing historical data from the current task—this method achieves a level of dynamism that allows for rapid adaptation to changes. Notably, the meta-model-based design of the proposed method facilitates the optimization of multiple hyperparameters simultaneously, effectively addressing the limitations encountered by other frameworks, such as bandit-over-bandit strategies.

The structure of this paper is outlined as follows: Section 2 delves into the existing body of research concerning hyperparameter optimization in bandit contexts, providing a literature review. Section 3 introduces the methodology of the proposed hyperparameter optimization approach. The experimental setup, alongside the outcomes and discussion of these results, is detailed in Section 4. The paper concludes with Section 5, where the final remarks and conclusions are presented.

2. Related Work

The effectiveness of multi-armed bandit frameworks in optimization and strategy selection is fundamentally driven by the adept management of the exploration-exploitation trade-off. Efficient exploration stands as the focal point of contextual bandit learning (Agarwal, Dudík, Kale, Langford, & Schapire, 2012; Agarwal et al., 2014; Agrawal & Goyal, 2013; Dudik et al., 2011; Langford & Zhang, 2007; Russo, Van Roy, Kazerouni, Osband, & Wen, 2018). To manage this trade-off, several methods such as stochastic formulation ((Lai & Robbins, 1985; Auer, Cesa-Bianchi, & Fischer, 2002; Bouneffouf, Parthasarathy, Samulowitz, & Wistub, 2019; Lin, Bouneffouf, Cecchi, & Rish, 2018; Bouneffouf & Féraud, 2016; Balakrishnan, Bouneffouf, Mattei, & Rossi, 2019a; Balakrishnan, Bouneffouf, Mattei, & Rossi, 2019b; Bouneffouf, Laroche, Urvoy, Féraud, & Allesiardo, 2014; Noothigattu et al., 2018), Bayesian formulation (Bouneffouf & Rish, 2019; Bouneffouf, Rish, Cecchi, & Féraud, 2017), adversarial formulation (Auer, Cesa-Bianchi, Freund, & Schapire, 2002; Balakrishnan, Bouneffouf, Mattei, & Rossi, 2018), and context-based frameworks (Lihong Li et al., 2010; Bastani et al., 2021; Agrawal & Goyal, 2013; Allesiardo, Féraud, & Bouneffouf, 2014) have been proposed. However, the magnitude of exploration in these algorithms, a critical hyperparameter with substantial effects on performance, must be specified by the user. In more sophisticated multi-armed bandit frameworks such as NeuralUCB (Zhou et al., 2020), additional hyperparameters need to be provided. These hyperparameters influence the model's ability to predict the regret of actions for various contexts, thereby impacting the overall performance of the bandit framework. Given the significant influence of hyperparameters on the performance of multi-armed bandit algorithms, finding their optimal values is imperative.

Several methods have been developed for optimizing machine learning hyperparameters, generally classified into six groups (Seifi & Niaki, 2023): search-based algorithms ((Abreu, 2019; Hutter, Kotthoff, & Vanschoren, 2019; Bergstra & Bengio, 2012; Zöller & Huber, 2021), heuristic algorithms (Di Francescomarino et al., 2018; Lorenzo, Nalepa, Ramos, & Pastor, 2017; Guo, Hu, Wu, Peng, & Wu, 2019), Bayesian algorithm (Injadat, Salo, Nassif, Essex, & Shami, 2018; Hutter, Hoos, & Leyton-Brown, 2011; Bergstra, Bardenet, Bengio, & Kégl, 2011), multi-fidelity algorithms (Swersky, Snoek, & Adams, 2014; Karnin,

Koren, & Somekh, 2013; Lisha Li, Jamieson, DeSalvo, Rostamizadeh, & Talwalkar, 2017; Falkner, Klein, & Hutter, 2018), population-based algorithms (Jaderberg et al., 2017; Parker-Holder, Nguyen, & Roberts, 2020), and reinforcement learning algorithms (Jomaa, Grabocka, & Schmidt-Thieme, 2019). Each method has its own advantages and disadvantages, but due to the real-time nature of multi-armed bandit frameworks, they cannot be directly employed for bandit hyperparameter optimization. Consequently, the literature proposes numerous methodologies customized for bandit hyperparameter optimization.

As an initial step, Bouneffouf (2016) and Bastani et al. (2021) explored greedy strategies to ascertain the optimal exploration rate, albeit with the risk of converging to suboptimal solutions. Similarly, Bietti et al. (2018) reviewed contextual bandit algorithms, emphasizing minimizing the exploration rate. They studied common exploration strategies such as Bootstrap Thompson Sampling, Online Cover, and ϵ-greedy across more than 500 datasets, demonstrating that minimizing exploration is crucial for practical performance. However, this approach is not universally applicable, as in nonstationary situations, it can lead to significant optimality gaps.

Ding et al. (2021) and Jun et al. (2017) employed grid search to optimize hyperparameters, but this approach is infeasible in practice, and manually discretizing the hyperparameter space is unclear. Chu (Chu, 2024) investigated the importance of selecting appropriate hyperparameters for an Explore-Then-Commit (ETC) algorithm using a trial-and-error method. Although this method is neither efficient nor practical, Chu emphasized the effect of hyperparameter selection on multi-armed bandit regret.

Meanwhile, Sharaf and Daumé (Sharaf & Daumé III, 2019) introduced the MÊLÉE algorithm leveraging data from simulated contextual bandit tasks to formulate effective exploration strategies. Despite its innovative approach and the power of meta-learning, this methodology struggles to adapt exploration parameters from live contextual bandit environments, potentially leading to ineffective solutions in dynamic contexts. Importantly, research (Ding et al., 2022) underscores that optimal exploration parameters vary across different scenarios, necessitating the dynamic adjustment of optimization algorithms to accommodate evolving conditions.

To address these challenges and facilitate dynamic learning of the exploration rate, the introduction of the OPLINUCB and DOPLINUCB algorithms (Bouneffouf & Claeys, 2020) marks significant progress. These algorithms, utilizing Thompson Sampling (TS) and Conditional Inference Tree (CTree), respectively, aim to dynamically tune the exploration rate. However, their efficacy is limited as they focus on optimizing a single hyperparameter. In the contextual bandit landscape, the need to tune multiple hyperparameters concurrently, such as the exploration parameter α, the regularization parameter λ in ridge regression, and various parameters associated with NeuralUCB (Zhou, Li, & Gu, 2020) (e.g., network width, depth, learning rate, and momentum), becomes evident. Therefore, methods that efficiently optimize multiple hyperparameters are imperative, and OPLINUCB and DOPLINUCB algorithms (Bouneffouf & Claeys, 2020) face challenges in this context.

To address these challenges and facilitate the dynamic learning of exploration rate, the introduction of the OPLINUCB and DOPLINUCB algorithms (Bouneffouf & Claeys, 2020) marks a significant advancement. These algorithms, utilizing Thompson Sampling (TS) and Conditional Inference Tree (CTree), respectively, aim to dynamically tune the exploration rate. However, their efficacy is tempered by a limitation: they primarily focus on optimizing a singular hyperparameter. In the contextual bandit landscape, the necessity to concurrently tune multiple hyperparameters, such as the exploration parameter α, the regularization parameter λ in ridge regression, and various parameters associated with NeuralUCB (Zhou et al., 2020) (e.g., network width, depth, learning rate, and momentum), becomes evident. Therefore, methods that efficiently optimize multiple hyperparameters are imperative, and OPLINUCB and DOPLINUCB algorithms (Bouneffouf & Claeys, 2020) face challenges in this context.

To overcome this issue, the Syndicated Bandits framework was proposed, addressing the multi-hyperparameter challenge without succumbing to the exponential growth in regret bounds associated with the number of parameters (Ding et al., 2022). Nevertheless, this framework's reliance on a predefined set of hyperparameter candidates and the need to discretize continuous hyperparameters present notable constraints. The Continuous Dynamic Tuning (CDT) framework (Kang et al., 2023) seeks to mitigate these limitations through a novel bandit-over-bandit approach, ensuring adaptability in dynamic and shifting environments. Despite its innovations, the CDT framework does not fully capitalize on the insights from previously optimized bandit tasks, a strategy that could significantly expedite the hyperparameter tuning process (Seifi & Niaki). This gap highlights an opportunity to harness the wealth of existing knowledge within the field to enhance algorithmic efficiency further.

In this paper, we propose a hyperparameter optimization framework for contextual multi-armed bandit problems with the following features:

- **Efficient Multi-Hyperparameter Optimization:** Considering the complexity of modern contextual bandit methods, the proposed framework should be capable of simultaneously optimizing multiple hyperparameters efficiently.
- **Dynamic Adaptation:** The framework should dynamically optimize hyperparameters, swiftly capturing new trends in the data.
- **Knowledge Leveraging:** The proposed algorithm is expected to leverage existing knowledge from similar tasks to expedite the time to reach an optimal solution and enhance the quality of results.

3. Methodology

In the preceding discussions, three primary challenges facing contextual bandit hyperparameter optimization methodologies were identified. The first challenge arises from certain algorithms' inability to function effectively within non-stationary and switching environments. The second challenge is the growing necessity for frameworks capable of concurrently optimizing multiple hyperparameters, a need accentuated by the advancement of bandit algorithms and the introduction of more sophisticated methods like NeuralUCB. The third issue pertains to data inefficiency, a problem that has become more pronounced as the complexity and dimensionality of these problems have expanded.

To address these challenges, this paper introduces a novel method that harnesses the capabilities of transfer learning and Bayesian optimization. This approach leverages the strength of transfer learning to utilize knowledge from similar prior tasks, thereby improving data efficiency and accelerating the optimization process. Additionally, to utilize the current problem's data and avoid issues caused by switching environments or adversarial attacks, two Gaussian meta-models are employed. The first meta-model facilitates knowledge transfer from analogous tasks through meta-features, while the second predicts the performance of various hyperparameter sets based on historical data. Should there be any alterations in the reward function, an increase in the prediction error of the second meta-model triggers the selection of new points via the transfer learning mechanism, ensuring the adaptability of our algorithm. Finally, unlike bandit-over-bandit frameworks, which face exponential complexity with the addition of more hyperparameters, our meta-model-based approach allows for the simultaneous optimization of multiple hyperparameters.

Key to our method is the assessment of task similarity, which is crucial for transferring knowledge effectively. To quantify the similarity among diverse bandit tasks, it is imperative to employ certain indices of similarity. These metrics, or meta-features, are designed to encapsulate all facets of the tasks under consideration comprehensively. In pursuit of this objective, seven meta-features, that encapsulate various dimensions of bandit tasks, have been introduced. Initially, the number of features and actions serve as foundational meta-features, playing a pivotal role in the similarity assessment among tasks. Their significance cannot be overstated, as they profoundly influence the exploration-exploitation balance. To gauge the attractiveness of different actions along with the variation of that among different actions, two additional meta-features have been proposed. These include the normalized likelihood of the most frequently chosen action and the normalized discrepancy between the selection probabilities of the two top-most chosen actions. These metrics elucidate the relative advantage of one action over others and underscore the importance of exploration. However, it is important to acknowledge that these two meta-features, while informative, may be susceptible to biases introduced by hyperparameters such as exploration rate α. To mitigate this vulnerability, two further meta-features that are not influenced by the effects of hyperparameters have been introduced. The first is the normalized difference between the highest predicted value of the reward function μ and the second-highest, and the second quantifies the normalized difference when incorporating the standard deviation σ to these predicted values. Lastly, to assess the accuracy of predictions at each step and determine the ongoing necessity for exploration or exploitation, we have utilized the Mean Absolute Percentage Error (MAPE) of the rewards' prediction. This metric serves as a crucial indicator of prediction quality and the adaptive requirements for exploration-exploitation dynamics. These meta-features are presented in Table 1.

Table 1
Meta-features Used for Transfer Learning

Meta-Feature	Formula
Number of Features	F
Number of Actions	A
Probability Ratio of Top Action	$\frac{\underset{i}{\mathrm{Max}}\, C_i \times A}{T}$
Gap Between Top Two Actions' Probabilities	$\frac{(\underset{i}{\mathrm{Max}}\, C_i - \underset{i}{\mathrm{Secondmax}}\, C_i) \times A}{T}$
Gap Between Top Two Predicted Action Values	$\frac{\underset{i}{\mathrm{Max}}\, \mu_i - \underset{i}{\mathrm{Secondmax}}\, \mu_i}{\underset{i}{\mathrm{Max}}\, \mu_i - \underset{i}{\mathrm{Min}}\, \mu_i}$
Gap With Uncertainty Between Top Two Predicted Action Values	$\frac{\underset{i}{\mathrm{Max}}(\mu_i + \sigma_i) - \underset{i}{\mathrm{Secondmax}}(\mu_i + \sigma_i)}{\underset{i}{\mathrm{Max}}(\mu_i + \sigma_i) - \underset{i}{\mathrm{Min}}(\mu_i + \sigma_i)}$
MAPE	$\left\lvert \frac{\text{Actual Reward} - \text{Predicted Reward}}{\text{Actual Reward}} \right\rvert$

Note. C_i:Number of selections for action *i*. *T*: Number of iterations.

At each step (t), meta-features (f) from previous tasks, along with their corresponding sets of hyperparameters ($\boldsymbol{h}$) and observed regret (R), are used to fit a Gaussian meta-model as equation 1.

$$\mathcal{GP}_t^{Transfer} = \mathcal{GP}\left(\begin{bmatrix} f_{1,t}^1 & \dots & f_{n,t}^1 & \boldsymbol{h}_t^1 \\ \vdots & \ddots & \vdots & \vdots \\ f_{1,t}^m & \dots & f_{n,t}^m & \boldsymbol{h}_t^m \end{bmatrix}, \begin{bmatrix} R_t^1 \\ \vdots \\ R_t^m \end{bmatrix}\right) \tag{1}$$

Then the meta-features of the current task would be extracted as $[f_{1,t}^{target}, f_{2,t}^{target}, \dots, f_{n,t}^{target}]$. Using this vector and Gaussian meta-model presented in Eq. (1), the set of hyperparameters that minimizes the regret given the current task's meta-features would be proposed as $\boldsymbol{h}_t^{*Transfer}$.

$$\boldsymbol{h}_t^{*Transfer} = \underset{\boldsymbol{h}}{\operatorname{argmin}}\, \mathcal{GP}_t^{Trtansfer} \,|[f_{1,t}^{target}, f_{2,t}^{target}, \dots, f_{n,t}^{target}] \tag{2}$$

In parallel, a Gaussian meta-model would be set to selected hyperparameters and observed regrets as Eq. (3).

$$\mathcal{GP}_t^{Surrogate} = \mathcal{GP}\left(\begin{bmatrix} \boldsymbol{h}_0^{target} \\ \vdots \\ \boldsymbol{h}_t^{target} \end{bmatrix}, \begin{bmatrix} R_0^{target} \\ \vdots \\ R_t^{target} \end{bmatrix}\right) \tag{3}$$

Using this model, the optimum hyperparameters that minimize regret based on Eq. (4) would be proposed as $\boldsymbol{h}_t^{*Surrogate}$.

$$\boldsymbol{h}_t^{*Surrogate} = \underset{\boldsymbol{h}}{\operatorname{argmin}}\, \mathcal{GP}_t^{Surrogate} \tag{4}$$

Now we should select between $\boldsymbol{h}_t^{*Transfer}$ or $\boldsymbol{h}_t^{*Surrogate}$ for evaluation in next step. To this aim, the accuracy of $\mathcal{GP}_{t-1}^{Transfer}$ and $\mathcal{GP}_{t-1}^{Surrogate}$ in forecasting the performance of $\boldsymbol{h}_{t-1}$ would be measured and the proposed hyperparameters of those with more accuracy would be selected. In other words, the predicted regret of transfer learning meta-model ($\hat{R}_{t-1}^{Transfer}$) and the predicted regret of the surrogate meta-model ($\hat{R}_{t-1}^{Surrogate}$) would be calculated as Eq. (5) and Eq. (6).

$$\hat{R}_{t-1}^{Transfer} = \mathcal{GP}_{t-1}^{Trtansfer}([f_{1,t-1}^{target}, f_{2,t-1}^{target}, \dots, f_{n,t-1}^{target}, \boldsymbol{h}_{t-1}]) \tag{5}$$

$$\hat{R}_{t-1}^{Surrogate} = \mathcal{GP}_{t-1}^{Surrogate}(\boldsymbol{h}_{t-1}) \tag{6}$$

By these predictions and the realization of R_{t-1} the MAPE of both models can be written as Eq. (7) and Eq. (8).

$$MAPE_{t-1}^{Transfer} = \frac{\left|R_{t-1} - \hat{R}_{t-1}^{Transfer}\right|}{R_{t-1}} \tag{7}$$

$$MAPE_{t-1}^{Surrogate} = \frac{\left|R_{t-1} - \hat{R}_{t-1}^{Surrogate}\right|}{R_{t-1}} \tag{8}$$

Finally, if the $MAPE_{t-1}^{Transfer}$ be less than $MAPE_{t-1}^{Surrogate}$, the $\boldsymbol{h}_t^{*Transfer}$ would be selected as next-step hyperparameters and vice versa. This framework, referred to as the Transfer Learning for Contextual Bandit Hyperparameter Optimization (TLCB-HPO), harnesses the power of transfer learning and Bayesian optimization. It is designed to enhance data efficiency and expedite the optimization process, enabling the simultaneous optimization of multiple hyperparameters and ensuring robust performance in dynamic and adversarial settings, thus offering a comprehensive solution to the challenges of hyperparameter optimization in contextual bandit environments. The proposed algorithm is displayed in Algorithm 1.

Algorithm1: Transfer Learning for Contextual Bandit Hyperparameter Optimization (TLCB-HPO) framework

Initialization: Set the Budget (B), Number of Features (NF), Number of Actions (NA), and Initial HPs $\boldsymbol{h}_0$

Source data: $\{[\boldsymbol{f}^1, \boldsymbol{h}^1, \boldsymbol{R}^1], [\boldsymbol{f}^2, \boldsymbol{h}^2, \boldsymbol{R}^2], \dots, [\boldsymbol{f}^m, \boldsymbol{h}^m, \boldsymbol{R}^m]\}$

For $t = 1, 2, \dots, B$:

1: $a_t = \underset{a}{\operatorname{argmax}}\, g(\boldsymbol{x}_t, a_t)$

2: Update coefficients of g (A, b, $\boldsymbol{\theta}$ for LinUCB or $\boldsymbol{w}$ for neural network)

3: Extract the target task's meta-features $[f_{1,t}^{target}, f_{2,t}^{target}, \dots, f_{n,t}^{target}]$

4: Fit a Gaussian process to the source data:

$$\mathcal{GP}_t^{Transfer} = \mathcal{GP}\left(\begin{bmatrix} f_{1,t}^1 & \dots & f_{n,t}^1 & \boldsymbol{h}_t^1 \\ \vdots & \ddots & \vdots & \vdots \\ f_{1,t}^m & \dots & f_{n,t}^m & \boldsymbol{h}_t^m \end{bmatrix}, \begin{bmatrix} R_t^1 \\ \vdots \\ R_t^m \end{bmatrix}\right)$$

5: Find the optimum value for the hyperparameters based on the fitted Gaussian process and extracted meta-features

$$\boldsymbol{h}_t^{*Transfer} = \underset{\boldsymbol{h}}{\operatorname{argmin}}\, \mathcal{GP}_t^{Trtansfer} \,|[f_{1,t}^{target}, f_{2,t}^{target}, \dots, f_{n,t}^{target}]$$

6: Fit a Gaussian process to the selected hyperparameters and their corresponding regrets:

$$\mathcal{GP}_t^{Surrogate} = \mathcal{GP}\left(\begin{bmatrix} \boldsymbol{h}_0^{target} \\ \vdots \\ \boldsymbol{h}_t^{target} \end{bmatrix}, \begin{bmatrix} R_0^{target} \\ \vdots \\ R_t^{target} \end{bmatrix}\right)$$

7: Find the optimum value for the hyperparameters based on the fitted Gaussian process

$$\boldsymbol{h}_t^{*Surrogate} = \underset{\boldsymbol{h}}{\operatorname{argmin}}\, \mathcal{GP}_t^{Surrogate}$$

If $t \geq 1$:

8: Predict the gained reward using the $\mathcal{GP}_{t-1}^{Trtansfer}$

$$\hat{R}_{t-1}^{Transfer} = \mathcal{GP}_{t-1}^{Trtansfer}([f_{1,t-1}^{target}, f_{2,t-1}^{target}, \ldots, f_{n,t-1}^{target}, \boldsymbol{h}_{t-1}])$$

9: Predict the gained reward using the $\mathcal{GP}_{t-1}^{Surrogate}$

$$\hat{R}_{t-1}^{Surrogate} = \mathcal{GP}_{t-1}^{Surrogate}(\boldsymbol{h}_{t-1})$$

10: Compare the models' prediction accuracy

$$MAPE_{t-1}^{Transfer} = \frac{\left|R_{t-1} - \hat{R}_{t-1}^{Transfer}\right|}{R_{t-1}}$$

$$MAPE_{t-1}^{Surrogate} = \frac{\left|R_{t-1} - \hat{R}_{t-1}^{Surrogate}\right|}{R_{t-1}}$$

If $MAPE_{t-1}^{Transfer} \leq MAPE_{t-1}^{Surrogate}$:

$$\boldsymbol{h}_t = \boldsymbol{h}_t^{*Transfer}$$

Else:

$$\boldsymbol{h}_t = \boldsymbol{h}_t^{*Surrogate}$$

End for

4. Experiments and Results

In this section, the efficacy of the proposed algorithm is evaluated through two distinct case studies designed to encompass a broad range of scenarios. The first case study focuses on assessing the ability of the TLCB framework to optimize the exploration rate α within the LinUCB algorithm. The second case study aims to refine the learning rate and momentum parameters in the NeuralUCB method. To rigorously test the algorithm's adaptability to dynamic environments, each case study is conducted under two markedly different conditions. The initial condition reverses rewards after a predetermined number of steps, whereas the second condition shuffles the reward functions associated with varying arms after a specific interval. These modifications facilitate examining the algorithm's performance across stationary and non-stationary environments. Furthermore, compared to its competitors, the proposed method's effectiveness is systematically analyzed across various actions and features in each case study.

The study examines five competing algorithms alongside the proposed method for a comprehensive comparison. The Bayesian Optimization algorithm is initially included due to its recognized success as the meta-model-based approach in hyperparameter optimization literature. This is complemented by two contemporary bandit-over-bandit frameworks, which represent the cutting edge in bandit hyperparameter optimization. The first of these frameworks employs an Upper Confidence Bound (UCB)-based approach, while the second utilizes a SoftMax-based strategy. Additionally, the analysis incorporates the greedy search algorithms and a gradual decrease algorithm, both acknowledged for their simplicity and practicality in various hyperparameter optimization scenarios. A series of simulations are conducted to construct a knowledge base for transfer learning. Specifically, for the LinUCB case study, simulations explore various configurations, encompassing differing numbers of actions, features, and exploration rates (α). A similar approach is applied to the NeuralUCB scenario, where simulations vary regarding actions, features, learning rate, and momentum parameters. The specific values for these parameters, as they pertain to both LinUCB and NeuralUCB, are detailed in Tables 2 and 3, respectively. Additionally, reward functions are altered at random intervals in specific simulations to simulate changing environments.

Table 2
Parameter Settings for LinUCB Simulations

Variable	Values for Simulation
Number of Action	[5, 7, 10, 12, 15]
Number of Features	[3, 5, 7, 10, 12, 15]
Exploration Rate α	[0.1, 0.3, 0.5, 0.75, 1, 3, 5, 7, 10]

Table 3
Parameter Settings for NeuralUCB Simulations

Variable	Values for Simulation
Number of Action	[5, 7, 10, 12, 15]
Number of Features	[3, 5, 7, 10, 12, 15]
Learning Rate	[0.0001, 0.001, 0.005, 0.01, 0.05, 0.1, 0.2]
Momentum	[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]

To construct a knowledge base for transfer learning, a series of simulations were conducted. Specifically, for the LinUCB case study, simulations explored various configurations, encompassing differing numbers of actions, features, and exploration rates (α). A similar approach was applied to the NeuralUCB scenario, where simulations varied in terms of actions, features, learning rate, and momentum parameters. The specific values for these parameters, as they pertain to both LinUCB and NeuralUCB, are detailed in Table 2 and Table 3, respectively. Additionally, reward functions are altered at random intervals in specific simulations to simulate changing environments.

In each scenario, simulations are conducted over 5,000 steps, with results and meta-features collected and aggregated at every 100-step interval. Similarly, the hyperparameters for both the proposed algorithm and its competitors are optimized at these intervals. Each combination of actions and features is optimized five times for all the methods to ensure thorough comparison. It is important to note that the reward function is consistent across all simulations within each run.

Table 4
Performance Analysis of LinUCB with Inverted Reward Function Post-2500 Steps - Average Regret and Standard Deviation

Actions	Features	Measure	TLCB	Bayesian	UCB-BoB	SoftMax-BoB	GD	GS
7	12	Best Regret Before Perturbation	0.00064 ± 0.0004	0.00062 ± 0.0001	0.00076 ± 0.0006	0.00074 ± 0.0002	**0.00039 ± 0.0001**	0.00071 ± 0.0003
		Average Regret Before Perturbation	0.0060± 0.0015	0.0056± 0.0013	0.0056± 0.0011	0.0068 ± 0.0043	**0.0048 ± 0.0006**	0.0058 ± 0.0009
		Best Regret After Perturbation	**0.2582 ± 0.0409**	0.3221 ± 0.0572	0.3092 ± 0.0574	0.3228 ± 0.0461	0.2970 ± 0.0598	0.3117 ± 0.0557
		Average Regret After Perturbation	**0.3171 ± 0.0298**	0.3734± 0.0616	0.3633± 0.0641	0.3658± 0.0396	0.3420± 0.0531	0.3793 ± 0.0714
10	10	Best Regret Before Perturbation	0.00079 ± 0.0002	0.00087 ± 0.0004	0.00118 ± 0.0006	0.00161 ± 0.0010	**0.00070 ± 0.0003**	0.00091 ± 0.0005
		Average Regret Before Perturbation	0.0095 ± 0.0029	0.0076 ± 0.0018	0.0082± 0.0018	0.0122 ± 0.0047	**0.0063 ± 0.0007**	0.0098 ± 0.0027
		Best Regret After Perturbation	**0.2825 ± 0.0255**	0.3285 ± 0.0420	0.3530 ± 0.0285	0.3379 ± 0.0304	0.3631 ± 0.0496	0.3530 ± 0.0562
		Average Regret After Perturbation	**0.3410 ± 0.0303**	0.3694 ± 0.0403	0.3963 ± 0.0320	0.3973 ± 0.0466	0.4108 ± 0.0448	0.3974 ± 0.0438
12	7	Best Regret Before Perturbation	**0.0011± 0.0008**	0.0016 ± 0.0007	0.0034 ± 0.0022	0.0013 ± 0.0006	0.0023 ± 0.0012	0.0022 ± 0.0015
		Average Regret Before Perturbation	0.0122 ± 0.0016	0.0119 ± 0.0039	0.0132 ± 0.0060	0.0217 ± 0.0059	**0.0117 ± 0.0362**	0.0169 ± 0.0077
		Best Regret After Perturbation	**0.2720 ± 0.0202**	0.3108 ± 0.0210	0.3196 ± 0.0218	0.3237 ± 0.0156	0.3356 ± 0.0282	0.3366 ± 0.0319
		Average Regret After Perturbation	**0.3123 ± 0.0214**	0.3435 ± 0.0305	0.3671 ± 0.0374	0.3580 ± 0.0238	0.3872 ± 0.0393	0.3641 ± 0.0361
12	12	Best Regret Before Perturbation	0.0014 ± 0.0006	0.0019 ± 0.0015	0.0016 ± 0.0011	**0.0011 ± 0.0004**	0.0027 ± 0.0014	0.0019 ± 0.0006
		Average Regret Before Perturbation	0.0115 ± 0.0035	**0.0081 ± 0.0020**	0.0146 ± 0.0045	0.0136 ± 0.0027	0.0159 ± 0.0039	0.0121 ± 0.0033
		Best Regret After Perturbation	**0.2882 ± 0.0116**	0.3567 ± 0.01578	0.3578 ± 0.0350	0.3524 ± 0.0325	0.3679 ± 0.0354	0.3554 ± 0.0163
		Average Regret After Perturbation	**0.3461 ± 0.0277**	0.4000 ± 0.0351	0.3972 ± 0.0435	0.3965 ± 0.0561	0.4434 ± 0.0409	0.4152 ± 0.0196

The outcomes of these simulations, explicitly focusing on regret metrics for both the LinUCB and NeuralUCB models under different conditions, are detailed in Tables 4, 6, 8, and 10. These tables present the average regret and standard deviation for LinUCB and NeuralUCB when rewards are reversed, or the arms' reward functions are shuffled after 2,500 steps. For each set of actions and features within these tables, four key metrics are calculated:

1. Best Regret Before Perturbation: This metric represents the lowest average regret (in each 100 calculation steps) obtained from the method across the initial 2500 steps (before any perturbations), offering insight into the best performance achievable by the algorithm under stable conditions.
2. Average Regret Before Perturbation: This indicates the overall performance level of the method before any disturbances, providing a general assessment of its effectiveness in a stable environment.
3. Best Regret After Perturbation: This shows the lowest average regret (in each 100 calculation steps) following perturbations, demonstrating the algorithm's capacity to adapt and recover its performance in the face of changes.
4. Average Regret After Perturbation: Useful for evaluating the method's long-term stability and adaptability, this metric reflects the overall performance during the final 500 steps, capturing how well the algorithm adjusted post-perturbation.

These metrics serve to assess the algorithms' performances before and after environmental changes, thus evaluating their resilience and adaptability to dynamic conditions. To enhance the comparison between the proposed method and its competitors, a Student's t-test is employed. This statistical test is utilized to quantify the difference between the average improvement achieved by the proposed method over its competitors for each case study. The analysis tests the following hypotheses:

$$\begin{cases} H_0: \mu_{Change} > 0 \\ H_a: \mu_{Change} \leq 0 \end{cases} \tag{9}$$

The outcomes of these statistical tests are presented in Tables 5, 7, 9, and 11. In these tables, the t-value, derived from the formula:

$$t = \frac{\hat{\mu}_{Change}\sqrt{n-1}}{\hat{\sigma}_{Change}} \tag{10}$$

where $change^i$ is as follows:

$$change^i = \frac{\mathrm{TLCB}^{\mathrm{i}}_{\mathrm{performance}} - \mathrm{Competitor}^{\mathrm{i}}_{\mathrm{performance}}}{\mathrm{Competitor}^{\mathrm{i}}_{\mathrm{performance}}} \tag{11}$$

The calculated t-values are then compared against the critical values for 95% confidence. The results in Tables 5, 7, 9, and 11 indicate whether the changes resulted in a statistically significant improvement (noted as an Improvement in the tables) or if the changes are not statistically meaningful (indicated as "No change").

Table 5
The Results of Student's t-Test for LinUCB with Inverted Reward Function Post-2500 Steps

Variable	Metric	Bayesian	UCB-BoB	SoftMax-BoB	GD	GS
Best Regret	**Average Improvement**	76.180%	84.268%	184.997%	97.920%	88.896%
Before	**t-value**	2.0854	2.8343	5.0246	2.0266	2.5216
Perturbation	**Significant level**	No Change	Improvement	Improvement	No Change	Improvement
Average Regret	**Average Improvement**	-11.166%	6.020%	134.623%	0.937%	14.504%
Before	**t-value**	-1.6530	0.7375	12.0778	0.0896	1.4113
Perturbation	**Significant level**	No Change	No Change	Improvement	No Change	No Change
Best Regret	**Average Improvement**	19.794%	22.158%	121.743%	23.748%	23.173%
After	**t-value**	9.7296	6.4154	60.8268	8.0267	10.2766
Perturbation	**Significant level**	Improvement	Improvement	Improvement	Improvement	Improvement
Average Regret	**Average Improvement**	12.957%	16.214%	115.253%	19.614%	18.164%
After	**t-value**	5.5780	4.7705	66.0329	6.0951	8.1687
Perturbation	**Significant level**	Improvement	Improvement	Improvement	Improvement	Improvement

Table 6
Performance Analysis of LinUCB with Shuffled Reward Function Post-2500 Steps - Average Regret and Standard Deviation

Actions	Features	Measure	TLCB	Bayesian	UCB-BoB	SoftMax-BoB	GD	GS
7	12	Best Regret Before Perturbation	**0.00025 ± 0.0002**	0.00054 ± 0.0006	0.00098 ± 0.0005	0.00035 ± 0.0002	0.00046 ± 0.0003	0.00055 ± 0.0002
		Average Regret Before Perturbation	0.0057 ± 0.0008	**0.0040 ± 0.0009**	0.0051 ± 0.0015	0.0058 ± 0.0014	0.0044 ± 0.0011	0.0064 ± 0.0010
		Best Regret After Perturbation	**0.1208± 0.0229**	0.1750± 0.0512	0.1664± 0.0582	0.1768 ± 0.0320	0.1874 ± 0.0551	0.1811 ± 0.0628
		Average Regret After Perturbation	**0.1482 ± 0.0362**	0.2080 ± 0.0538	0.2055 ± 0.0435	0.2093 ± 0.0424	0.2202 ± 0.0522	0.2182 ± 0.0570
10	10	Best Regret Before Perturbation	0.00082 ± 0.0004	**0.00077 ± 0.0003**	0.00168 ± 0.0012	0.00085 ± 0.0002	0.00100 ± 0.0003	0.00137 ± 0.0009
		Average Regret Before Perturbation	0.0094 ± 0.0026	0.0079 ± 0.0021	0.0114 ± 0.0029	0.0114 ± 0.0030	**0.0065 ± 0.0016**	0.0118 ± 0.0032
		Best Regret After Perturbation	**0.1288 ± 0.0278**	0.1840 ± 0.0203	0.1755 ± 0.0139	0.1795 ± 0.0177	0.1815 ± 0.0142	0.1827 ± 0.0221
		Average Regret After Perturbation	**0.1599 ± 0.0261**	0.2058 ± 0.0236	0.2016 ± 0.0177	0.2023 ± 0.0204	0.2044 ± 0.0193	0.2096 ± 0.0233
12	7	Best Regret Before Perturbation	**0.0010± 0.0006**	0.0017 ± 0.0002	0.0032 ± 0.0008	0.0012 ± 0.0004	0.0016 ± 0.0007	0.0019 ± 0.0014
		Average Regret Before Perturbation	**0.0106 ± 0.0014**	0.0129 ± 0.0027	0.0149 ± 0.0022	0.0175 ± 0.0055	0.0113 ± 0.0015	0.0169 ± 0.0059
		Best Regret After Perturbation	**0.1081 ± 0.0168**	0.1543 ± 0.0292	0.1221 ± 0.0222	0.1282 ± 0.0206	0.1388 ± 0.0463	0.1438 ± 0.0176
		Average Regret After Perturbation	**0.1307 ± 0.0138**	0.1756 ± 0.0382	0.1573 ± 0.0203	0.1569 ± 0.0285	0.1710 ± 0.0433	0.1636 ± 0.0219
12	12	Best Regret Before Perturbation	**0.0010 ± 0.0006**	0.0012 ± 0.0008	0.0018 ± 0.0009	0.0012 ± 0.0007	0.0016 ± 0.0007	0.0011 ± 0.0007
		Average Regret Before Perturbation	0.0128 ± 0.0021	**0.0103 ± 0.0053**	0.0182 ± 0.0099	0.0127 ± 0.0029	0.0152 ± 0.0024	0.0118 ± 0.0016
		Best Regret After Perturbation	**0.1254 ± 0.0202**	0.1648 ± 0.0262	0.1599 ± 0.0101	0.1634 ± 0.0152	0.1409 ± 0.0286	0.1513 ± 0.0160
		Average Regret After Perturbation	**0.1518 ± 0.0229**	0.1874 ± 0.0221	0.1815 ± 0.0183	0.1896 ± 0.0204	0.1803 ± 0.0237	0.1773 ± 0.0204

Table 7
The Results of Student's t-Test for LinUCB with Shuffled Reward Function Post-2500 Steps

Variable	Metric	Bayesian	UCB-BoB	SoftMax-BoB	GD	GS
Best Regret	**Average Improvement**	151.281%	312.061%	198.258%	165.888%	213.619%
Before	**t-value**	2.0458	4.0015	4.6566	2.2396	2.0639
Perturbation	**Significant level**	No Change	Improvement	Improvement	Improvement	No Change
Average Regret	**Average Improvement**	-11.512%	22.154%	121.960%	-9.190%	19.807%
Before	**t-value**	-1.4577	1.9328	10.9859	-1.2803	1.8683
Perturbation	**Significant level**	No Change	No Change	Improvement	No Change	No Change
Best Regret	**Average Improvement**	41.379%	29.677%	135.564%	34.953%	37.557%
After	**t-value**	8.6231	5.5875	26.2925	4.9260	6.5405
Perturbation	**Significant level**	Improvement	Improvement	Improvement	Improvement	Improvement
Average Regret	**Average Improvement**	32.973%	27.867%	129.754%	32.777%	31.488%
After	**t-value**	6.0970	5.7827	25.2942	5.4698	6.0321
Perturbation	**Significant level**	Improvement	Improvement	Improvement	Improvement	Improvement

Table 8
Performance Analysis of NeuralUCB with Inverted Reward Function Post-2500 Steps - Average Regret and Standard Deviation

Actions	Features	Measure	TLCB	Bayesian	UCB-BoB	SoftMax-BoB	GD	GS
7	7	Best Regret Before Perturbation	**0.0394 ± 0.0128**	0.0628 ± 0.0098	0.0699 ± 0.0160	0.0698 ± 0.0139	0.0592 ± 0.0168	0.0665 ± 0.0041
		Average Regret Before Perturbation	**0.0733 ± 0.0104**	0.1212 ± 0.0188	0.1124 ± 0.0058	0.1185 ± 0.0179	0.1120 ± 0.0223	0.1088 ± 0.0073
		Best Regret After Perturbation	**0.1308± 0.0233**	0.2704± 0.0922	0.3161± 0.0249	0.2728 ± 0.0717	0.2602 ± 0.0598	0.2928 ± 0.1111
		Average Regret After Perturbation	**0.1618 ± 0.0287**	0.3075 ± 0.1049	0.3621 ± 0.0164	0.3182 ± 0.0833	0.2961 ± 0.0682	0.3274 ± 0.1129
7	12	Best Regret Before Perturbation	**0.0561 ± 0.0328**	0.0833 ± 0.0103	0.0837 ± 0.0184	0.0743 ± 0.0180	0.0885 ± 0.0147	0.0732 ± 0.0136
		Average Regret Before Perturbation	**0.1026 ± 0.0347**	0.1371 ± 0.0047	0.1361 ± 0.0082	0.1315 ± 0.0036	0.1423 ± 0.0132	0.1312 ± 0.0175
		Best Regret After Perturbation	**0.0723 ± 0.0257**	0.1863 ± 0.0310	0.2366 ± 0.0763	0.2011 ± 0.0205	0.1689 ± 0.0227	0.1671 ± 0.0288
		Average Regret After Perturbation	**0.0938 ± 0.0270**	0.2360 ± 0.0330	0.2664 ± 0.0633	0.2474 ± 0.0266	0.2110 ± 0.0371	0.2192 ± 0.0272
10	10	Best Regret Before Perturbation	**0.0744± 0.0194**	0.0998 ± 0.0180	0.0898 ± 0.0095	0.0918 ± 0.0189	0.0812 ± 0.0096	0.0842 ± 0.0093
		Average Regret Before Perturbation	**0.1237 ± 0.0281**	0.1502 ± 0.0126	0.1462 ± 0.0076	0.1402 ± 0.0104	0.1477 ± 0.0122	0.1293 ± 0.0068
		Best Regret After Perturbation	**0.0903 ± 0.0244**	0.2111 ± 0.0228	0.2271 ± 0.0424	0.1920 ± 0.0358	0.1830 ± 0.0214	0.2241 ± 0.0307
		Average Regret After Perturbation	**0.1162 ± 0.0214**	0.2322 ± 0.0255	0.2607 ± 0.0396	0.2281 ± 0.0377	0.2106 ± 0.0216	0.2500 ± 0.0281
12	7	Best Regret Before Perturbation	**0.0821 ± 0.0218**	0.0872 ± 0.0106	0.1035 ± 0.0151	0.0848 ± 0.0137	0.0840 ± 0.0112	0.0845 ± 0.0060
		Average Regret Before Perturbation	0.1313 ± 0.0234	**0.1287 ± 0.0221**	0.1494 ± 0.0089	0.1434 ± 0.0182	0.1387 ± 0.0132	0.1292 ± 0.0078
		Best Regret After Perturbation	**0.1129 ± 0.0410**	0.2213 ± 0.0378	0.1759 ± 0.0289	0.2163 ± 0.0458	0.2024 ± 0.0276	0.1984 ± 0.0308
		Average Regret After Perturbation	**0.1406 ± 0.0404**	0.2521 ± 0.0366	0.2186 ± 0.0292	0.2552 ± 0.0395	0.2414 ± 0.0154	0.2301 ± 0.0324

Table 9
The Results of Student's t-Test for NeuralUCB with Inverted Reward Function Post-2500 Steps

Variable	Metric	Bayesian	UCB-BoB	SoftMax-BoB	GD	GS
Best Regret Before Perturbation	**Average Improvement**	53.784%	57.961%	147.381%	45.297%	42.798%
	t-value	3.4544	4.2490	10.3104	2.9561	2.9989
	Significant level	Improvement	Improvement	Improvement	Improvement	Improvement
Average Regret Before Perturbation	**Average Improvement**	34.376%	34.833%	132.710%	33.721%	23.990%
	t-value	3.9934	4.7787	16.8280	4.4688	3.4521
	Significant level	Improvement	Improvement	Improvement	Improvement	Improvement
Best Regret After Perturbation	**Average Improvement**	139.056%	160.491%	239.596%	122.094%	133.664%
	t-value	7.7432	5.9579	12.4927	6.1508	7.5913
	Significant level	Improvement	Improvement	Improvement	Improvement	Improvement
Average Regret After Perturbation	**Average Improvement**	115.464%	132.784%	220.817%	103.788%	112.680%
	t-value	6.9694	6.5928	13.5838	5.5292	7.7963
	Significant level	Improvement	Improvement	Improvement	Improvement	Improvement

The results, as presented in Tables 5 and 7, demonstrate that in the first case study, which focuses on optimizing the exploration rate (α) within the LinUCB framework, the proposed method significantly outperformed all its competitors following perturbation events. This superior performance can be attributed to the TLCB framework's ability to dynamically adjust its exploration rate in response to perturbations. Specifically, in the event of changes in the reward function, the Mean Absolute Percentage Error (MAPE) of reward prediction (as a measured meta-feature presented in Table 1) increases. By analyzing this value and leveraging knowledge from similar tasks—either their initial stages where MAPE is higher or tasks experiencing perturbations—the TLCB framework proactively increases the exploration rate, facilitating rapid adaptation to changes in the reward function. In contrast, competing methods tend to rely on historical performance data to determine exploration rates, which may hinder their ability to quickly respond to new shifts in reward dynamics.

Before the perturbation, the TLCB framework demonstrated dominant performance over the UCB-based Bandit-over-Bandit (UCB-BoB), SoftMax-based Bandit-over-Bandit (SoftMax-BoB), and Greedy Search (GS) algorithms, as indicated in Table 5, and over UCB-BoB, SoftMax-BoB, and Gradual Decrease (GD) methods, as shown in Table 7. While the TLCB method also exhibited superior performance in other comparisons, these improvements were not statistically significant at the 95% confidence level. Consequently, the TLCB framework is recommended for use not only in dynamic, perturbation-prone environments but also in stationary settings, given its consistent and robust performance across scenarios.

In the second case study, aimed at optimizing both the learning rate and momentum within the NeuralUCB framework, the proposed method outperformed all competitors across all evaluated metrics, both before and after perturbation events. This superiority can be attributed to several aspects of the TLCB framework compared to its competitors. Firstly, unlike the bandit-over-bandit algorithms and the Greedy Search algorithm, which, when faced with multiple hyperparameters, embark on an exhaustive exploration of all possible hyperparameter combinations—thereby prolonging the journey to the optimal solution—the TLCB employs a Gaussian meta-model to predict the performance of different hyperparameter sets. This approach helps the algorithm avoid exhaustive searches, reducing time to the optimal solution and enhancing efficiency. Secondly, compared to the Bayesian optimization algorithm, the advantage of utilizing insights from similar tasks through the TLCB approach becomes apparent. Leveraging knowledge from similar tasks in initial steps has helped the TLCB algorithm combat cold-start problems and reduce time to optimal solutions. Similarly, using transfer learning allows the TLCB framework to capture and respond to abrupt changes in the reward function, leading to its superior performance compared to competitors. Consequently, the proposed algorithm has shown better performance in all metrics at the 95% confidence level in both static and dynamic environments when optimizing multiple hyperparameters simultaneously.

Table 10
Performance Analysis of NeuralUCB with Shuffled Reward Function Post-2500 Steps – Average Regret and Standard Deviation

Actions	Features	Measure	TLCB	Bayesian	UCB-BoB	SoftMax-BoB	GD	GS
7	7	Best Regret Before Perturbation	**0.0365 ± 0.0080**	0.0582 ± 0.0091	0.0736 ± 0.0110	0.0751 ± 0.0257	0.0669 ± 0.0144	0.0708 ± 0.0162
		Average Regret Before Perturbation	**0.0825 ± 0.0171**	0.1129 ± 0.0131	0.1258 ± 0.0142	0.1272 ± 0.0229	0.1141 ± 0.0136	0.1127 ± 0.0136
		Best Regret After Perturbation	**0.0429 ± 0.0076**	0.1047 ± 0.0241	0.0859 ± 0.0192	0.0930 ± 0.0199	0.1069 ± 0.0293	0.0973 ± 0.0319
		Average Regret After Perturbation	**0.0629 ± 0.0117**	0.1347 ± 0.0170	0.1287 ± 0.0296	0.1427 ± 0.0221	0.1317 ± 0.0325	0.1338 ± 0.0335
7	12	Best Regret Before Perturbation	**0.0317 ± 0.0122**	0.0732 ± 0.0090	0.0798 ± 0.0151	0.0787 ± 0.0125	0.0889 ± 0.0162	0.0761 ± 0.0104
		Average Regret Before Perturbation	**0.0869 ± 0.0187**	0.1206 ± 0.0108	0.1333 ± 0.0092	0.1268 ± 0.0141	0.1387 ± 0.0049	0.1257 ± 0.0090
		Best Regret After Perturbation	**0.0559 ± 0.0249**	0.1122 ± 0.0197	0.1097 ± 0.0273	0.1270 ± 0.0410	0.1057 ± 0.0283	0.0921 ± 0.0255
		Average Regret After Perturbation	**0.0744 ± 0.0195**	0.1401 ± 0.0193	0.1460 ± 0.0256	0.1597 ± 0.0457	0.1430 ± 0.0239	0.1244 ± 0.0331
10	10	Best Regret Before Perturbation	**0.0559± 0.0127**	0.0856 ± 0.0163	0.0937 ± 0.0121	0.0795 ± 0.0137	0.0690 ± 0.0156	0.0867 ± 0.0123
		Average Regret Before Perturbation	**0.1114 ± 0.0161**	0.1282 ± 0.0092	0.1433 ± 0.0170	0.1381 ± 0.0126	0.1298 ± 0.0119	0.1266 ± 0.0121
		Best Regret After Perturbation	**0.0680 ± 0.0187**	0.1035 ± 0.0222	0.1144 ± 0.0139	0.1106 ± 0.0179	0.0951 ± 0.0202	0.1040 ± 0.0177
		Average Regret After Perturbation	**0.0825 ± 0.0179**	0.1301 ± 0.0174	0.1447 ± 0.0210	0.1380 ± 0.0177	0.1244 ± 0.0128	0.1368 ± 0.0335
12	7	Best Regret Before Perturbation	**0.0582 ± 0.0140**	0.0871 ± 0.0077	0.0832 ± 0.0084	0.0885 ± 0.0090	0.0890 ± 0.0193	0.0968 ± 0.0112
		Average Regret Before Perturbation	0.1175 ± 0.0173	0.1381 ± 0.0117	0.1418 ± 0.0155	0.1388 ± 0.0057	0.1447 ± 0.0117	0.1439 ± 0.0099
		Best Regret After Perturbation	**0.0762 ± 0.0123**	0.1097 ± 0.0215	0.0973 ± 0.0132	0.1194 ± 0.0269	0.1206 ± 0.0214	0.1086 ± 0.0188
		Average Regret After Perturbation	**0.1000 ± 0.0104**	0.1408 ± 0.0294	0.1286 ± 0.0161	0.1424 ± 0.0211	0.1520 ± 0.0311	0.1397 ± 0.0284

Table 11
The Results of Student's t-Test for NeuralUCB with Shuffled Reward Function Post-2500 Steps

Variable	Metric	Bayesian	UCB-BoB	SoftMax-BoB	GD	GS
Best Regret Before Perturbation	**Average Improvement**	86.762%	107.680%	204.056%	105.305%	102.585%
	t-value	4.6884	4.4987	8.9540	3.4485	5.3431
	Significant level	Improvement	Improvement	Improvement	Improvement	Improvement
Average Regret Before Perturbation	**Average Improvement**	30.796%	43.549%	138.805%	38.832%	32.207%
	t-value	4.4997	4.7342	19.6067	4.7450	5.7299
	Significant level	Improvement	Improvement	Improvement	Improvement	Improvement
Best Regret After Perturbation	**Average Improvement**	92.788%	80.731%	200.564%	91.223%	79.124%
	t-value	6.3844	6.1690	12.0380	5.8925	5.3430
	Significant level	Improvement	Improvement	Improvement	Improvement	Improvement
Average Regret After Perturbation	**Average Improvement**	79.980%	80.670%	193.412%	82.557%	75.157%
	t-value	6.9969	6.7294	13.1673	5.8536	6.0470
	Significant level	Improvement	Improvement	Improvement	Improvement	Improvement

In summary, the TLCB algorithm has demonstrated superior performance across all cases in the presence of perturbations, attributable to its ability to capture and respond to changes by readjusting hyperparameters using transfer learning. Moreover, the TLCB algorithm exhibited significantly better performance compared to its competitors in 70% of scenarios under

stationary conditions. In the remaining 30% of stationary cases, the TLCB algorithm offered better solutions in nearly all instances, though these were not statistically significant improvements. This consistent superiority in performance, both in dynamic and stable environments, underscores the TLCB algorithm's capability to excel over its competitors.

Furthermore, regarding computational efficiency, the complexity of the TLCB framework is denoted as $O(\max(B^3, m^3))$, where B represents the budget for evaluations, and m is the number of similar tasks considered. This indicates that in terms of execution time, the TLCB algorithm matches its competitors, including Bayesian optimization, exhibiting no significant discrepancy in running time. This parallel in computational performance, coupled with the qualitative advantages in solution quality, positions the TLCB framework as a highly competitive choice in the realm of hyperparameter optimization.

5. Conclusion

In this study, we introduced a novel framework for optimizing hyperparameters in contextual bandit environments, utilizing a transfer learning approach. Our methodology employs a dual Gaussian meta-model structure: one model for assimilating knowledge from analogous tasks and another for assessing hyperparameter performance within the current context. The selection of evaluation points is guided by accuracy metrics, employing Bayesian optimization to navigate the decision-making process effectively. This dual-structured approach empowers the algorithm to both draw upon historical task insights and swiftly adapt to environmental shifts. Additionally, the framework's meta-model-centric architecture enables simultaneous optimization of multiple hyperparameters, overcoming challenges that have hampered previous solutions.

In comparative analyses with five distinct competing methods, reflective of various strategic paradigms within the field, our Transfer Learning for Contextual Bandit Hyperparameter Optimization (TLCB) framework demonstrated marked superiority in all cases under conditions of environmental perturbation. In stable conditions, the TLCB framework outperformed its counterparts in 70% of stationary scenarios, while matching performance in the remaining 30%. The algorithm's computational complexity, akin to that of the Bayesian optimization approach, underscores its viability as both a robust and efficient tool for hyperparameter tuning in dynamic contextual bandit environments.

Code Availability: The complete source code used in this study, including scripts for the experiments, data processing, and analysis, is available on GitHub. Interested researchers can access and download the code from github.com/farshad-seifi/Transfer-Learning_for_Contextual_Bandit. This repository aims to facilitate further research and exploration in the field of contextual bandit hyperparameter optimization.

References

Abreu, S. (2019). Automated architecture design for deep neural networks. *arXiv preprint arXiv:1908.10714*.

Agarwal, A., Dudík, M., Kale, S., Langford, J., & Schapire, R. (2012). *Contextual bandit learning with predictable rewards.* Paper presented at the Artificial Intelligence and Statistics.

Agarwal, A., Hsu, D., Kale, S., Langford, J., Li, L., & Schapire, R. (2014). *Taming the monster: A fast and simple algorithm for contextual bandits.* Paper presented at the International Conference on Machine Learning.

Agrawal, S., & Goyal, N. (2013). *Thompson sampling for contextual bandits with linear payoffs.* Paper presented at the International conference on machine learning.

Allesiardo, R., Féraud, R., & Bouneffouf, D. (2014). *A neural networks committee for the contextual bandit problem.* Paper presented at the Neural Information Processing: 21st International Conference, ICONIP 2014, Kuching, Malaysia, November 3-6, 2014. Proceedings, Part I 21.

Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine Learning, 47*, 235-256.

Auer, P., Cesa-Bianchi, N., Freund, Y., & Schapire, R. E. (2002). The nonstochastic multiarmed bandit problem. *SIAM journal on computing, 32*(1), 48-77.

Balakrishnan, A., Bouneffouf, D., Mattei, N., & Rossi, F. (2018). *Using Contextual Bandits with Behavioral Constraints for Constrained Online Movie Recommendation.* Paper presented at the IJCAI.

Balakrishnan, A., Bouneffouf, D., Mattei, N., & Rossi, F. (2019a). *Incorporating behavioral constraints in online AI systems.* Paper presented at the Proceedings of the AAAI Conference on Artificial Intelligence.

Balakrishnan, A., Bouneffouf, D., Mattei, N., & Rossi, F. (2019b). Using multi-armed bandits to learn ethical priorities for online AI systems. *IBM Journal of Research and Development, 63*(4/5), 1: 1-1: 13.

Bastani, H., & Bayati, M. (2020). Online decision making with high-dimensional covariates. *Operations Research, 68*(1), 276-294.

Bastani, H., Bayati, M., & Khosravi, K. (2021). Mostly exploration-free algorithms for contextual bandits. *Management Science, 67*(3), 1329-1349.

Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization. *Advances in neural information processing systems, 24*.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research, 13*(2).

Bietti, A., Agarwal, A., & Langford, J. (2018). Practical evaluation and optimization of contextual bandit algorithms. *Statistics, 1050*, 12.

Bouneffouf, D. (2016). Exponentiated gradient exploration for active learning. *Computers, 5*(1), 1.

Bouneffouf, D., & Claeys, E. (2020). Hyper-parameter tuning for the contextual bandit. *arXiv preprint arXiv:2005.02209*.

Bouneffouf, D., & Féraud, R. (2016). Multi-armed bandit problem with known trend. *Neurocomputing, 205*, 16-21.

Bouneffouf, D., Laroche, R., Urvoy, T., Féraud, R., & Allesiardo, R. (2014). *Contextual bandit for active learning: Active thompson sampling.* Paper presented at the Neural Information Processing: 21st International Conference, ICONIP 2014, Kuching, Malaysia, November 3-6, 2014. Proceedings, Part I 21.

Bouneffouf, D., Parthasarathy, S., Samulowitz, H., & Wistub, M. (2019). Optimal exploitation of clustering and history information in multi-armed bandit. *arXiv preprint arXiv:1906.03979*.

Bouneffouf, D., & Rish, I. (2019). A survey on practical applications of multi-armed and contextual bandits. *arXiv preprint arXiv:1904.10040*.

Bouneffouf, D., Rish, I., Cecchi, G. A., & Féraud, R. (2017). Context attentive bandits: Contextual bandit with restricted context. *arXiv preprint arXiv:1705.03821*.

Chu, J.-C. (2024). *Hyperparameter optimization strategy for Multi-Armed Bandits: Genre recommendation in MovieLens dataset.* Paper presented at the Proceedings of the 2023 International Conference on Machine Learning and Automation.

Di Francescomarino, C., Dumas, M., Federici, M., Ghidini, C., Maggi, F. M., Rizzi, W., & Simonetto, L. (2018). Genetic algorithms for hyperparameter optimization in predictive business process monitoring. *Information Systems, 74*, 67-83.

Ding, Q., Hsieh, C.-J., & Sharpnack, J. (2021). *An efficient algorithm for generalized linear bandit: Online stochastic gradient descent and thompson sampling.* Paper presented at the International Conference on Artificial Intelligence and Statistics.

Ding, Q., Kang, Y., Liu, Y.-W., Lee, T. C. M., Hsieh, C.-J., & Sharpnack, J. (2022). Syndicated bandits: A framework for auto tuning hyper-parameters in contextual bandit algorithms. *Advances in Neural Information Processing Systems, 35*, 1170-1181.

Dudik, M., Hsu, D., Kale, S., Karampatziakis, N., Langford, J., Reyzin, L., & Zhang, T. (2011). Efficient optimal learning for contextual bandits. *arXiv preprint arXiv:1106.2369*.

Falkner, S., Klein, A., & Hutter, F. (2018). *BOHB: Robust and efficient hyperparameter optimization at scale.* Paper presented at the International Conference on Machine Learning.

Guo, B., Hu, J., Wu, W., Peng, Q., & Wu, F. (2019). The Tabu_genetic algorithm: a novel method for hyper-parameter optimization of learning algorithms. *Electronics, 8*(5), 579.

Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2011). *Sequential model-based optimization for general algorithm configuration.* Paper presented at the International conference on learning and intelligent optimization.

Hutter, F., Kotthoff, L., & Vanschoren, J. (2019). *Automated machine learning: methods, systems, challenges*: Springer Nature.

Injadat, M., Salo, F., Nassif, A. B., Essex, A., & Shami, A. (2018). *Bayesian optimization with machine learning algorithms towards anomaly detection.* Paper presented at the 2018 IEEE global communications conference (GLOBECOM).

Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W. M., Donahue, J., Razavi, A., . . . Simonyan, K. (2017). Population based training of neural networks. *arXiv preprint arXiv:1711.09846*.

Jomaa, H. S., Grabocka, J., & Schmidt-Thieme, L. (2019). Hyp-rl: Hyperparameter optimization by reinforcement learning. *arXiv preprint arXiv:1906.11527*.

Jun, K.-S., Bhargava, A., Nowak, R., & Willett, R. (2017). Scalable generalized linear bandits: Online computation and hashing. *Advances in neural information processing systems, 30*.

Kang, Y., Hsieh, C.-J., & Lee, T. (2023). Online continuous hyperparameter optimization for contextual bandits. *arXiv preprint arXiv:2302.09440*.

Karnin, Z., Koren, T., & Somekh, O. (2013). *Almost optimal exploration in multi-armed bandits.* Paper presented at the International Conference on Machine Learning.

Lai, T. L., & Robbins, H. (1985). Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics, 6*(1), 4-22.

Langford, J., & Zhang, T. (2007). The epoch-greedy algorithm for multi-armed bandits with side information. *Advances in neural information processing systems, 20*.

Li, L., Chu, W., Langford, J., & Schapire, R. E. (2010). *A contextual-bandit approach to personalized news article recommendation.* Paper presented at the Proceedings of the 19th international conference on World wide web.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2017). Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research, 18*(1), 6765-6816.

Lin, B., Bouneffouf, D., Cecchi, G. A., & Rish, I. (2018). *Contextual bandit with adaptive feature extraction.* Paper presented at the 2018 IEEE International Conference on Data Mining Workshops (ICDMW).

Lorenzo, P. R., Nalepa, J., Ramos, L. S., & Pastor, J. R. (2017). *Hyper-parameter selection in deep neural networks using parallel particle swarm optimization.* Paper presented at the Proceedings of the Genetic and Evolutionary Computation Conference Companion.

Noothigattu, R., Bouneffouf, D., Mattei, N., Chandra, R., Madan, P., Varshney, K., . . . Rossi, F. (2018). Interpretable multi-objective reinforcement learning through policy orchestration. *arXiv preprint arXiv:1809.08343*.

Parker-Holder, J., Nguyen, V., & Roberts, S. (2020). Provably efficient online hyperparameter optimization with population-based bandits. *arXiv preprint arXiv:2002.02518*.

Russo, D. J., Van Roy, B., Kazerouni, A., Osband, I., & Wen, Z. (2018). A tutorial on thompson sampling. *Foundations and Trends® in Machine Learning, 11*(1), 1-96.

Schwartz, E. M., Bradlow, E. T., & Fader, P. S. (2017). Customer acquisition via display advertising using multi-armed bandit experiments. *Marketing Science, 36*(4), 500-522.

Seifi, F., & Niaki, S. T. A. Dynamic Meta-Learning Acquisition Function Method for Bayesian Optimization with Early Stopping Criteria for Hyperparameter Optimization. *Available at SSRN 4205030*.

Seifi, F., & Niaki, S. T. A. (2023). Extending the hypergradient descent technique to reduce the time of optimal solution achieved in hyperparameter optimization algorithms. *International Journal of Industrial Engineering Computations, 14*(3), 501-510.

Sharaf, A., & Daumé III, H. (2019). Meta-learning for contextual bandit exploration. *arXiv preprint arXiv:1901.08159*.

Swersky, K., Snoek, J., & Adams, R. P. (2014). Freeze-thaw Bayesian optimization. *arXiv preprint arXiv:1406.3896*.

Woodroofe, M. (1979). A one-armed bandit problem with a concomitant variable. *Journal of the American Statistical Association, 74*(368), 799-806.

Zhou, D., Li, L., & Gu, Q. (2020). *Neural contextual bandits with ucb-based exploration.* Paper presented at the International Conference on Machine Learning.

Zöller, M.-A., & Huber, M. F. (2021). Benchmark and survey of automated machine learning frameworks. *Journal of Artificial Intelligence Research, 70*, 409-472.