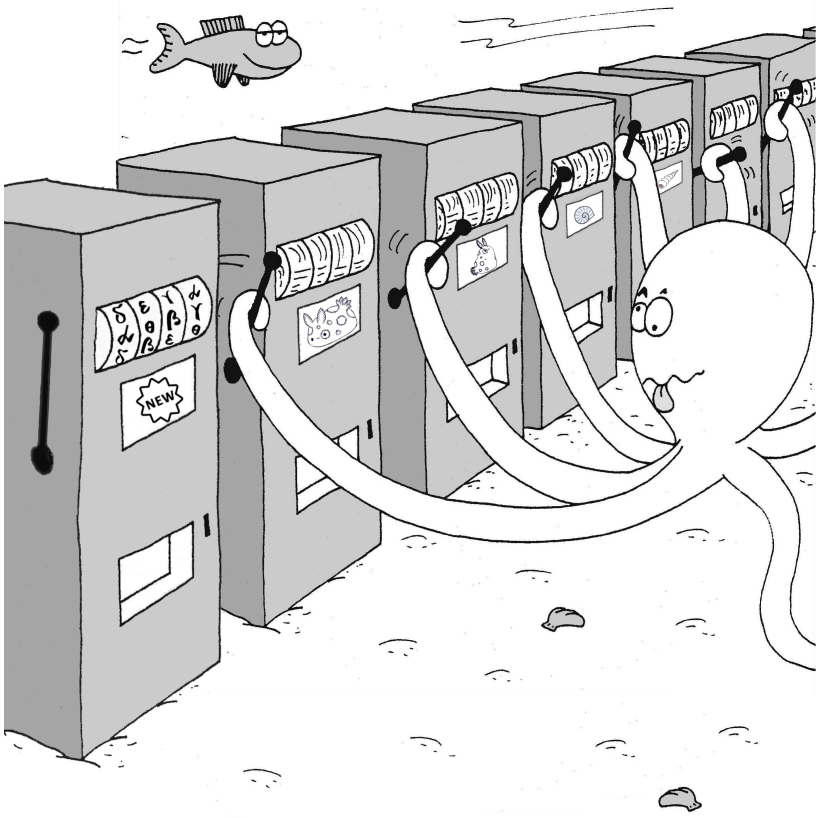
CHALMERS
UNIVERSITY OF TECHNOLOGY | UNIVERSITY OF GOTHENBURG



An Empirical Survey of Bandits in an Industrial Recommender System Setting

Contextual Multi-Armed Bandits and the Importance of Domain-Specific Features

Master's thesis in computer science and engineering

Johan Brandby
Tobias Schwarz

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2023

Master's thesis 2023

An Empirical Survey of Bandits in an Industrial Recommender System Setting

Contextual Multi-Armed Bandits and the Importance of Domain-Specific Features

Johan Brandby
Tobias Schwarz

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2023

An Empirical Survey of Bandits in an Industrial Recommender System Setting
Contextual Multi-Armed Bandits and the Importance of Domain-Specific Features
Johan Brandby Tobias Schwarz

Supervisor: Emilio Jorge, Computer Science and Engineering
Advisor: Johan Gustafsson, `YouPic AB`
Examiner: Devdatt Dubhashi, Computer Science and Engineering

Master's Thesis 2023
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

The cover image uses the idea of multiple slot machines, also known as one-armed bandits, and has an octopus—representing the MAB algorithm—playing on the machines. A "new" machine in the foreground symbolizes the influx of new images. Each machine has its own image with which the algorithm can find connections. The image is purchased[1] with an academic license and adapted from Hagen, 2001, "Compulsive gambling."

Typeset in LaTeX
Gothenburg, Sweden 2023

[1] `https://www.cartoonstock.com/directory/o/one_armed_bandit.asp`

An Empirical Survey of Bandits in an Industrial Recommender System Setting
Contextual Multi-Armed Bandits and the Importance of Domain-Specific Features
Johan Brandby & Tobias Schwarz
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

In this thesis, the effects of incorporating unstructured data—images in the wild—in contextual multi-armed bandits are investigated, when used within a recommender system setting, which focuses on picture-based content suggestion. The idea is to employ image features, extracted by a pre-trained convolutional neural network, and study the resulting bandit behaviors when including respective excluding this information in the typical context creation, which normally relies on structured data sources—such as metadata. The evaluation is made both online, through A/B-testing enabled by the industrial partner `YouPic AB`, and offline, effectuated by a simulation pipeline that models the online counterpart. The results are compiled as a survey, covering a selection of contextual bandit algorithms, highlighting the differences brought by the unstructured data. The offline result points towards that if the contextual bandit utilizes a joint or hybrid action-value function, with respect to the parameterization, the addition of the image vectors can significantly outperform the instances without it; however, if a disjoint model is instead employed, no noticeable change is observed. In comparison, those from the online trials can be interpreted as supporting the inclusion of convolutional features, but due to meager and unbalanced sample sizes, the outcomes are deemed inconclusive. To summarize, though there is support for incorporating unstructured data, given that the action-value function is joint or hybrid, the online experiments gave too little evidence for any trustworthy findings; in other words, the question is still partially open.

Keywords: computer science, industrial application, machine learning, reinforcement learning, multi-armed bandits, MAB, contextual multi-armed bandits, survey, batch learning.

Acknowledgements

We would like to express gratitude to our academic supervisor, Emilio Jorge, who has guided us through this thesis; his enthusiasm for the project, guidance, and numerous discussions has enabled us to carry on when the road was dark. In addition, we give thanks to our industrial supervisor, Johan Gustafsson, for his assistance with the integration of the systems and knowledge of the target industry. Without the support of these two, this work would have never been finished.

Furthermore, we extend special thanks to our examiner, Devdatt Dubhashi, for his patience and time, together with our thesis opponents, Sanjeev Madhavan and Harish Ravi, for their peer review and feedback. Appreciation is also directed towards our industry partner, `YouPic AB`, whose platform and data have enabled this survey. Finally, we would like to give a big thanks to our parents, as well as our siblings, for their support and love.

Johan Brandby & Tobias Schwarz, Gothenburg, January 2023

Contents

List of Figures

List of Tables

1

Introduction

In today's digital age, connecting people to content can be of vital importance for a business venture to succeed; whether it may be a digital marketplace, which seeks to maximize sales, or a social media platform that pursues user retention[1] (i.e., keeping the clients' patronage for a prolonged period.) Regardless of the business model, recommender systems (RS) have shown to be a reliable choice for personalized content, which can outperform standard strategies such as suggesting best sellers or trending posts [1].

Historically, these knowledge systems have primarily followed one of two modeling paradigms: collaborative or content-based filtering [2, 3]. In short, the former utilizes logged feedback from other consumers to impute the user ratings on novel items, while the latter predicts the preferences based on the items' descriptive attributes and their respective past interactions. However, modern engines usually employ a hybrid of the aforementioned models, and potentially other methodologies, as to compound their different modeling strengths.

According to the meta-study of Silva *et al.* [4], over the last two decades, the topic of multi-armed bandits (MAB) in recommender systems has been gaining interest in both academia and industry, owing to its promise of balancing the exploration-exploitation trade-off while in deployment. Traditionally, the bandit problem is defined as a sequential decision process between an agent and its environment, where the former interacts with the latter, through a predefined set of options, and is reciprocated with a (possibly random) feedback signal in a round-based game [5]. The long-term objective, for the agent, is to maximize the total accumulated reward through the interplay with the environment. When used in the RS setting, the action set is the available items to suggest, the environment is the user being served and the response to a recommended item is the feedback signal [4, 6].

Additionally, Silva *et al.* [4] also found that almost half of all published bandit articles employ contextual MABs (CMAB), which utilizes auxiliary information from the users and items to condition its selection policy, in comparison to the standard variant. In contemporary CMABs, these informational contexts tend to be constructed as fixed-sized vectors and are created from structured data sources, such as metadata of users and items [6].

This key point, from the previous paragraph, also holds for the traditional recommen-

[1] `https://en.wikipedia.org/wiki/Customer_retention`

dation practices, such as content-based or hybrid-collaborative filtering; but within these fields, there have been attempts towards assimilating unstructured sources of information—images in the wild—by preprocessing them with pre-trained neural networks (NN) [7, 8, 9]. Put briefly, by adopting an acclaimed convolutional NN (CNN), for instance, AlexNet [10] or the Caffe reference model [11], and interpreting it as a feature extractor, this model can embed images into a reduced, latent space while retaining some of their semantic information [12]. These encoded representations have been shown to be generic enough for usage outside the networks' intended problem domain, thus substantiating their application as a data transformation method.

Now, according to this paper's, albeit small-scale, literature study, there is little research on the integration of CNN features in RS using contextual bandits. This thesis aims to investigate how the performances of well-known CMABs are impacted by the addition of such quantities and also if this affects their convergence speed; hence, in essence, the goal is to produce a survey of the effects CNN features have on CMAB in RS. Moreover, this study is done in collaboration with the social-media company `YouPic AB`, which will provide both datasets for offline evaluation and an opportunity for online confirmation.

1.1 Outline

Before moving on, this paragraph provides a quick outline of this thesis' content. The ensuing Chapter 2 presents the problem setting and needed theory for this work, which delves deeper into the theoretical definitions of RSs, MABs, etc. In Chapter 3 this report's methodology is presented and the found results are displayed in Chapter 4. Lastly, the thesis is concluded with a discussion of the adopted methods, their implications, and the results' potential interpretations, before ending with a conclusion, all of which is in Chapter 5.

2

Background

In this section, the main theoretical fields of the thesis and their keywords are presented. First, the traditional approaches of recommender engines are shown, along with their limitations. Second, the domain of multi-armed bandits (MAB) is briefly explored, including the contextual sub-domain. Then, the fundamentals of neural networks as feature extractors and their usage as reward models, in offline policy evaluation, are covered. Whereafter the background continues onto the methodology chapter, but before that, the problem setting is described.

2.1 Recommendation Systems

Users interact daily with online platforms (e.g., a retail shop) browsing a wide variety of products, which often are too numerous to list. In this scenario, the online platform usually takes the initiative and recommends products believed to be in their interests. Aggarwal [2] and Sharda *et al.* [3] describe these recommender—or recommendation—systems as decision processes that attempt to infer the users' responses to novel items. Furthermore, for such a system, the pool of available items includes more than the classical definition of merchandise; for instance, movies, music, travel destinations, social connections, and job offers are all examples of items that can be suggested.

As mentioned in the introduction, historically there have been two dominant approaches to tackle this problem: collaborative and content-based filtering. The former, i.e., collaborative, pools the past interactions of users to impute the missing values using content-free techniques, such as non-negative matrix factorization. However, its main challenge is that the needed user-image interactions generally are sparse in quantity, hence commonly making the problem ill-conditioned. Whilst the latter, the content-based, focuses on relating the targeted outcome with the item attributes through explicit modeling, often utilizing parameterized functions. This approach's key benefit is that any novel item's relevance can be estimated given its attributes; although, this comes at the cost of introducing modeling bias.

Regardless of what recommender system paradigm is employed, they all are liable to drawbacks brought forth by their respective assumptions, some of which are shared across definitions. A common limitation is the cold-start problem, which comes into play when there is not enough history or attributes to be able to infer any suggestions; the remedy is usually dependent on the deployed system, but tends to include a

short survey for newly created users. Another shared issue is with the scalability of the service; as recommendations are typically presented as interactive and responsive elements, their generation needs to be speedy enough for online deployment while advanced enough to capture the subtleties in the user-item interactions, which are often in conflict as the latter necessitates computational power. Nevertheless, their advantages usually outweigh their disadvantages, and, for this thesis, the integration of multi-armed bandits is explored, whose definition is addressed in the next section.

2.2 Multi-Armed Bandits

The typical stochastic bandit problem entails a finite sequential game between an algorithmic player and an unfamiliar environment, realized over $T \in \mathbb{N}$ rounds of interaction [5]. During each such time step $t \in [T]$, where $[T] \stackrel{\text{def}}{=} \{1, \ldots, T\}$, the learning agent selects an arm—action—$A_t \in \mathcal{A}$ out of $K = |\mathcal{A}| \geq 2$ possible choices; whereafter, the environment reciprocates by sampling a reward signal $R_t \in \mathcal{D}_{A_t}$, where $\mathcal{D}_a \subset \mathbb{R}$ is the reward distribution corresponding to action $a \in \mathcal{A}$ [13]. The reward R_t for each arm $a \in \mathcal{A}$ is assumed to be an independent and identically distributed (i.i.d.) random variable from $\mathcal{D}_a$ and the conventional goal is to have the learning agent maximize the total cumulative sum of the these signals

$$S_T = \sum_{t=1}^{T} R_t. \tag{2.1}$$

The only information available to the learner, at the start of the game, is that the environment is a member of some predefined set $\mathcal{E}$, called environment class [5]. With the dynamics of its setting unknown, the agent has to utilize its experienced history $H_{t-1} = \{(A_\tau, R_\tau)\}_{\tau=1}^{t-1}$ when making a decision at time step t [5, 13]. It is a policy function π that maps such a history to an action, which is formalized as

$$\pi : (\mathcal{A} \times \mathbb{R})^t \to \mathcal{A}, \tag{2.2}$$

where the function output can be stochastic or deterministic over the arm domain. Notice that, when trying to maximize S_T, any such function π will try to capitalize on the arm $a \in \mathcal{A}$ that yields the highest mean reward

$$\mu^\star = \max_{a \in \mathcal{A}} \mu(a), \tag{2.3}$$

where $\mu(a) = \mathbb{E}[\mathcal{D}_a]$ and with ties broken arbitrarily. Although, which subset this corresponds to is hidden, and the set $\mathcal{A}$ needs to be searched.

The preceding paragraph describes the crux of online learning, having to strike a balance between exploiting known rewarding actions, and exploring the under-sampled options to find potentially better alternatives [5, 13]. This equipoise is commonly referred to as the exploration-exploitation trade-off and is what contemporary research attempts to address, by investigating differing learning strategies. Noteworthy examples of MAB algorithms are `LinUCB` and `Thompson Sampling`, proposed by Li *et*

al. [6] respective Thompson [14, 15]; however, observe that the latter was popularized by Scott [16] and Chapelle and Li [17].

The accepted, theoretical practice for evaluating a policy π is the T-trial regret; it is the lost expected reward accumulated up to time step T, relative to the best-performing policy $\pi^\star$ of some set of policies Π, known as the competitor class [5]. The benchmark $\pi^\star$ is usually taken to be the best-arm strategy, which always selects the action yielding the highest mean reward $\mu^\star$, for each round $t \in [T]$; thus allowing it to have a total cumulative reward of $T\mu^\star$ [13]. Mathematically, the regret is formalized as

$$R_T(\pi) = \sum_{t=1}^{T} \Big[\mu^\star - \mu(a_t)\Big] = T\mu^\star - \sum_{t=1}^{T} \mu(a_t), \tag{2.4}$$

where the terms $\{\mu^\star - \mu(a_t)\}_{t=1}^{T}$ are known as the reward gaps. When a closed-form solution is possible, this tends to be asymptotic and expressed using big-$\mathcal{O}$ notation; with the variables of interest being the number of arms and the time horizon, that is K and T, respectively. However, this metric is usually intractable in practice, as the best-known policy is unknown, and one often settles on studying the cumulative feedback rate instead.

2.2.1 Contextual Bandits

The contextual bandit problem has several alternative definitions and, for this work, that described by Li *et al.* [6] is employed due to its similar setting. Paraphrasing their article, for every $t \in [T]$, the algorithm observes a user $u_t \in \mathbb{N}$ and a set of available arms $\mathcal{A}_t \subset \mathbb{N}$. Each option $a \in \mathcal{A}_t$ has an accompanying feature vector $\boldsymbol{x}_{t,a} \in \mathbb{R}^d$ where $\|\boldsymbol{x}_{t,a}\|_2 = 1$ and $d \in \mathbb{N}$, referred to as the context, which incorporates the information from both the user u_t and the action a jointly. The bandit algorithm then selects an arm $a_t \in \mathcal{A}_t$ and receives a reward signal r_t from a bandit-feedback distribution $\mathcal{D}_{u_t,a_t}$ whose mean is dependent on the user u_t and action a_t pair. Thereafter, with the new history record—alternatively, outcome tuple—$(\boldsymbol{x}_{t,a_t}, a_t, r_t)$, the algorithm updates its selection strategy for better performance in future rounds.

For stochastic contextual bandits utilizing the definition above, the feedback signal R_t is assumed to depend on the context $\boldsymbol{x}_{t,a_t}$, for action $a_t \in \mathcal{A}_t$ and round $t \in [T]$, through

$$R_t = f(\boldsymbol{x}_{t,a_t}, a_t) + \epsilon_t, \tag{2.5}$$

where f is an unknown function, with $0 \leq f(\boldsymbol{x}_{t,a}, a) \leq 1$, and ϵ_t is a zero-mean ν-sub-Gaussian random variable, which are standard assumptions in this field of stochastic bandits [18]. Classical examples of f are linear relationships such as $f(\boldsymbol{x}_{t,a}, a) = \boldsymbol{x}_{t,a}^\intercal \boldsymbol{\theta}$ and $f(\boldsymbol{x}_{t,a}, a) = \boldsymbol{x}_{t,a}^\intercal \boldsymbol{\theta}_a$, which are corresponds to a joint respective disjoint linear model [6, 13].

2.2.2 Batch Bandits

Multi-armed bandits are, by their definition, sequential decision processes, but there are instances where the feedback arrives in batches, each after some delay. Examples of this include clinical trials [19], where a group of patients receive treatment and

only after all outcomes have become available the next round of treatment can start, or virus replication experiments [20], where a virus is replicated over b steps, representing all possible treatment options. The common factor between these cases is that the feedback can only be observed after a certain number of rounds or a specific period of time. In recent years, batch learning has gained more attention and has been successfully applied to other domains, such as crowd-sourcing, marketing, and simulations [21].

The number of selects that are batched into a group tends to be dependent on the problem setting; for instance, it can assume a predefined, fixed threshold value or be a dynamic stopping criterion that monitors the bandit's convergence [20]. Moreover, if viewing the batched selection as the fundamental sampling unit, using this learning over the continuous counterpart has no significant drawbacks. In fact, Jun *et al.* [20] showed that, by employing batched sampling, the sample complexity to identify the top arm(s) is not significantly affected. At the same time, the convergence of the arms to the optimal arms is accurate and quick as well as provides a close to min-max optimal regret [22, 19, 21].

2.3 Neural Networks and Deep Learning

Artificial neural networks, commonly rephrased as neural networks (NN), refer to a machine learning framework that utilizes computational graphs to approximate mappings between input and output distributions, utilizing representation learning [23]. Through interconnected layers of neurons—tensor operations processed by element-wise non-linearities—these systems have proven to be powerful tools capable of function approximation, text summarization, etc.; all by using labeled samples, many layers, and self-supervised learning techniques—i.e., *deep learning*.

2.3.1 Convolutional Networks

An important category of neural networks is the convolutional variant; which exploits the grid-like topology of the input features to learn patch-wise patterns correlating with the response [23]. In short, they utilize convolution instead of pure tensor (matrix) operations and a collection of these kernels corresponds to a layer. One of their most famous application is object recognition, whereby stacking these layers on top of each other, it can create higher-level abstractions of the input and thus allow for human-level object recognition.

2.3.2 Feature Extraction

Feature extraction can be described as a pre-processing step that transforms high-dimensional data into a reduced representation while preserving some subset of the original information [24]. Historically, these features are constructed by hand for their particular domain and problem setting—e.g., the scale-invariant feature transform (`SIFT`), proposed by Lowe, within the field of computer vision for key-point detection and description [25]. However, with the rise of machine learning, it became

possible to jointly learn a locally optimal representation and the corresponding mapping between a target and source domain [23].

One example of this is transfer learning, which repurposes a learned feature representation, from a source task, to improve the learning process in a new target setting [26, 27]. Within deep learning, one strategy is to reuse the sub-network, corresponding to the found feature mapping, by incorporating it in a new model through parameter sharing [23, 27]—either via fine-tuning or freezing them to updates. Razavian *et al.* [12] showed that a frozen convolutional network can produce features that are general enough for problem settings not originally intended; e.g., using the coefficients from an object-detection network for attribute-detection and image retrieval.

2.4 Bandit Evaluation

In this section, the applied evaluation strategies are presented. It starts with a brief rundown of the offline setting, where special focus is put on the direct method of reward modeling. Thereafter, a short description of online assessment is provided, along with the particularities of recommender systems.

2.4.1 Offline

Offline evaluation, or offline policy evaluation (OPE) in reinforcement learning, entails estimating the performance of an evaluation policy π_e, based on the logged interactions of a behavior policy π_b [28, 29]. Three common approaches, for inferring the would-have state-/action-values, are the direct method (DM), importance sampling (IS), and double robust (DR) strategy [30, 31, 32]. DR techniques utilize the statistical doubly robust property, as to improve the chances of generating unbiased inference by combining the DM and IS estimators [33]; thus leaving the two first aforementioned approaches for further explanation. In the following paragraphs, assume that $c_t \in \mathcal{C}$ and $a_t \in \mathcal{A}_t$ are the observed context and action at time step $t \in [T]$, respectively. Furthermore, let $r_{a_t} \in \mathbb{R}$ be a reward realization for an arbitrary action.

DM assumes that r_{a_t} can be approximated by an estimate $\hat{r} : \mathcal{C} \times \mathcal{A}_t \mapsto \mathbb{R}$, which is then used as a substitute for the true reward [30, 31, 32, 33]. Depending on the mapping of choice, this method can ensure low variance but could introduce significant modeling bias—which can be difficult to quantify depending on the function class. Moreover, usually $\hat{r}$ does not incorporate any knowledge of π_e. As a consequence, it might put undue weight on approximating domain areas irrelevant to the evaluation policy and give poor predictions where it matters.

IS methods—alternatively inverse propensity score (IPS)—requires knowledge of the behavior policy's history (as defined above) and the probabilities associated with each action [30, 33, 31, 32]. These are used to correct for the differences between the behavior and evaluation policies' distributions, through re-weighting the logged rewards—i.e., importance sampling. Under mild assumptions, this metric is unbiased but can exhibit large variance. In fact, Gilotte *et al.* [34] analyzed the correlation between offline and online A/B-tests for recommendation systems, and reports that

frequently-used, IPS-based estimators can appear uncorrelated with their actual online deployment.

Moving on, recent empirical findings of Voloshin *et al.* [31, 32] show that there is no clear-cut, best class of methods for OPE, and which to use is problem-dependent. However, for contextual bandits, Wang *et al.* [35] shows that DM can sometimes significantly outperform DR techniques. Additionally, in reinforcement learning, recent works have demonstrated how neural networks can be successfully utilized in planning through state- and action-value prediction [36, 37]. Thus unveiling how neural networks can be used as reward models in a DM fashion.

2.4.2 Online

For recommender systems, online evaluation corresponds to controlled randomized experiments, or A/B-testing, which appraises a set of alternative solutions by querying live users [2]. In detail, it entails deploying each system independently and then distributing the incoming users to the resulting service buckets, where the emphasis is put on minimizing any systematic bias between each instance. During their deployment, the evaluation metrics are constantly monitored and, when the experiment comes to a close after a predefined length of time, the score of each system can be derived.

Now, by definition, a multi-armed bandit algorithm is to operate with live data, such that its behavior can impact the effective sampling strategy generating its observations [28]. The online A/B-testing approach allows for evaluating multiple bandits concurrently in the real setting, thus having less bias than other evaluation methods, such as offline evaluation and user studies [2].

2.5 Setting

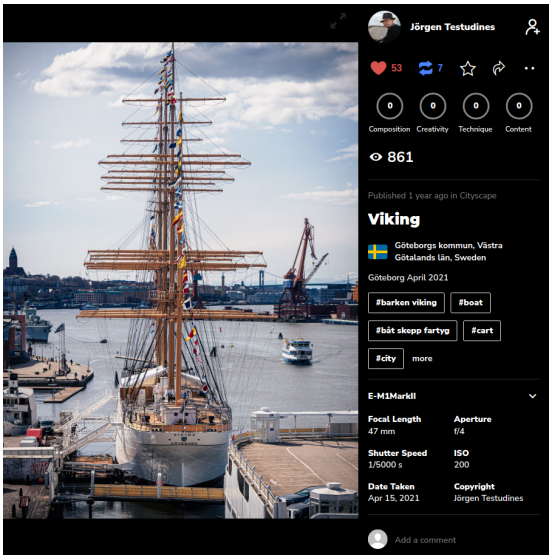
In this chapter, this project's setting is presented, together with the problem definition as given by the industry partner, `YouPic`. This includes the available data—historic and real-time—along with the technical and ethical considerations. However, first, a short description of `YouPic` and its platform are in order.

2.5.1 `YouPic AB`

As mentioned before, the industrial partner of this project is `YouPic AB`, a Gothenburg-based start-up. They host a social-media platform that targets professional and hobby photographers alike, encouraging them to share their work for recognition and networking, while also offering room to learn and find inspiration. It is accessible through both the web (`youpic.com`) and mobile apps, which see a fair bit of traffic. For example, in the first four months of 2022, around 600-thousand images have been uploaded and around 1.5 million users visited the platform, generating 186 million views in total [38, 39].

At the time of writing this, their website consists of four different feeds where

(a) `Hot`-flow



(b) Image post

Figure 2.1: Examples from the `YouPic`-platform on how the `Hot`-flow and the image posts can look like. Subfigure (a) shows an example of how the `Hot`-flow can be structured, while (b) depicts an image posted by Jörgen Testudines, 2022, available at `youpic.com/image/18797958` [Accessed 2022-05-06].

pictures can be shown, each specializing on a specific attribute or category. Their main stream is the `Inspiration`-flow, whose content is provided by human curators that have sifted through the newest posts. The three remaining mediums are called `Hot`, `Newest`, and `Models`; the first generates personalized recommendations using collaborative filtering, the second is as the name implies and the last shows the latest photos with human models in. Every such feed can be further filtered by supplying additional tags, such as `#cityscape`, `#water`, or `#reflection`.

For this thesis, the `Hot`-feed will serve as the vehicle for the bandit algorithms' suggestions, as its purpose is already that of targeted proposals. The deployed bandits will only see a proportion of its traffic, but given a long enough time frame a sufficient amount of data should have been generated. An example of the flow's appearance can be found in Figure 2.1.

2.5.2 Data

As previously stated, all data are provided by the project's industry partner, `YouPic AB`, and entails both the action logs and the historic states of users, and items—alternatively image posts. The former can be seen as feedback signals and

encompasses the interactions between patrons and pictures, which is the topic of the next section. On the other hand, the latter is composed of metadata from the customers' profiles, and the items' attributes, which corresponds to the structured data of this project, i.e., a part of the feature contexts, although unprocessed. Since these records already exist in their database, in some shape or form, the same data sources can be used in both the offline and the online evaluation.

The metadata, of users and items, contains quantities such as their time of inception, a description, camera model, etc., while also including fields unique to their type. For instance, consumers incorporate values like their region of origin, while the image posts hold semantic tags specifying the picture's content. An example of the latter's visual rendering can be found in Figure 2.1, which depicts an image post's pop-out when browsing the `Hot`-flow. On a side note, for this work, observe that the items' images are referred to as unstructured data.

2.5.3 Metrics

There are many metrics that are tracked on the `YouPic`-platform, which could function as feedback signals with minimal modifications, and are categorized into two types: generic and platform-specific. Examples of the former include views, likes, shares, and comments, which can be found on any content-based platform, while the latter contains scores that focus on photography, such as composition, creativity, technique, and content.

The metric chosen for this project is that corresponding to likes, with the motivation of limiting the thesis' scope, as it is accessible already in the flow itself. Whenever a user requests a batch of items while browsing the `Hot`-flow, the selected score is assumed to be generated with respect to the suggested image in question, whose entire batch of responses is referred to as an event.

3

Methods

This chapter details the methodology used in this thesis and begins with presenting the data handling, including how the contexts are constructed and from where they are sourced. Thereafter, the online and the offline evaluation strategies are addressed, elaborating on the shared and individual parts of their implementations. Then, the architecture and training of the employed neural networks are outlined, which are utilized as plugin modules in the aforementioned designs. Lastly, the chapter concludes by listing the menagerie of bandit algorithms tested in this survey.

3.1 Data Handling

Reiterating the definition in Section 2.2.1, the contexts $\{\boldsymbol{x}_{t,a}\}_{a\in\mathcal{A}_t}$ are constructed from the pairwise combination of a user, indexed $u_t \in \mathbb{N}$, and the coupled items $\mathcal{A}_t$, for round $t \in [T]$. Denote the user and the item sub-contexts as $\boldsymbol{u}_t \in \mathbb{R}^{p_u}$ and $\boldsymbol{i}_{t,a} \in \mathbb{R}^{p_i}$, respectively, with $p_u, p_i \in \mathbb{N}$; then, for this thesis' implementation, the main vectors are constructed from their concatenation. Mathematically, this becomes

$$\boldsymbol{x}_{t,a} = \begin{bmatrix} \boldsymbol{u}_t^\intercal & \boldsymbol{i}_{t,a}^\intercal \end{bmatrix}^\intercal, \tag{3.1}$$

where $\boldsymbol{x}_{t,a} \in \mathbb{R}^p$, $\|\boldsymbol{x}_{t,a}\|_2 = 1$ and $p = p_u + p_i$.

The following subsection addresses how these sub-contexts are fabricated, delving into the utilized strategies for processing the structured and unstructured sources independently. Thereafter, a segment is dedicated to the scraping of said feature vectors in both the offline and online evaluation.

3.1.1 Feature Construction

As motivated by this work's aim, the context features $\{\boldsymbol{x}_{t,a}\}_{a\in\mathcal{A}_t}$ are fashioned from both structured and unstructured sources; the former is selected to be the metadata from users and items, while the latter as the images associated with each item. To enable the bandits to utilize this auxiliary information, their fields' data types need to be mapped to the numerical domain, e.g., convert categorical variables to one-hot vectors and encode timestamps to a set of unit circles—to preserve the cyclic nature of time. Thereby, given that this pre-processing is immutable and done with complete fidelity, the returned set of numeric variables can be combined to make a

single feature vector of unchanging dimensionality. The following proposed feature construction is used in both the online and the offline settings.

The two following subsections describe the employed transformation procedures for the structured respective unstructured sources in depth. By design, the user context $\boldsymbol{u}_t$ is entirely made from the metadata recorded in its profile description, which includes country of residence, interaction statistics, etc. Contrarily, the items utilize both sources of information in their context creations; this is formalized as

$$\boldsymbol{i}_{t,a} = \left[\begin{array}{cc} \boldsymbol{i}_{t,a}^{(\mathtt{s})\intercal} & \boldsymbol{i}_{t,a}^{(\mathtt{u})\intercal} \end{array} \right]^{\intercal}, \tag{3.2}$$

where $\boldsymbol{i}_{t,a}^{(\mathtt{s})} \in \mathbb{R}^{p_i^{(\mathtt{s})}}$ and $\boldsymbol{i}_{t,a}^{(\mathtt{u})} \in \mathbb{R}^{p_i^{(\mathtt{u})}}$ are the sub-vectors from the item's structured ($\mathtt{s}$) and unstructured ($\mathtt{u}$) sources, respectively, and $p_i = p_i^{(\mathtt{s})} + p_i^{(\mathtt{u})}$ for $p_i^{(\mathtt{s})}, p_i^{(\mathtt{u})} \in \mathbb{N}$. Summarizing, $\boldsymbol{u}_t$ and $\boldsymbol{i}_{t,a}^{(\mathtt{s})}$ contains the numerical features extracted from their metadata, while $\boldsymbol{i}_{t,a}^{(\mathtt{u})}$ contains the embedded image vectors; that is, $\boldsymbol{i}_{t,a}$ holds both sources of information.

Moreover, as MAB algorithms are intended to function in online environments, they will all require low-latency inference capabilities, which are accentuated further by their usage in recommender systems. To enable this, each of the core sub-contexts—$\boldsymbol{u}_t$, $\boldsymbol{i}_{t,a}^{(\mathtt{s})}$ and $\boldsymbol{i}_{t,a}^{(\mathtt{u})}$—have their dimensionality reduced by a respective encoder network, whose architectures and training regimes are described in Section 3.3. This procedure also serves to combat the superfluously sized context dimensions emerging from the one-hot encodings, which comes as an added benefit.

3.1.1.1 Structured

For this report, structured data is synonymous with the metadata from the user profiles and the item descriptions, where the aim is to interpret and transform their fields into a real-valued representation, such as scalars or arrays of such, concatenating these to make the vectors $\boldsymbol{u}_t$ and $\boldsymbol{i}_{t,a}^{(\mathtt{s})}$, respectively. As the feature construction is dependent on data-type, and not its source, the procedures to construct either of the aforementioned contexts overlap to a large extent. This section aims to describe how these different data fields are mapped to real-valued numerals or vectors. Furthermore, due to space limitations, only a portion of the key transformations are reported here.

Now, two of the methods have already been briefly mentioned, namely those for categorical variables and timestamps. Examples of the former are the users' country of residence and camera types, which both are encoded as one-hot vectors. While for the latter, there are the items' publication dates and the timestamps resulting from user requests, all of which follow the UTC standard and are encoded as a set of unit circles. Elaborating, as time is a cyclic measure, every such instance is converted into three parts: the percentage of the elapsed month, week, and day. As every such value spans between zero and unity, it is trivial to map each of them to their own unit circle without loss of information, while still representing their periodic nature. The progression into the year is intentionally left out, due to including it would make the encoder networks extrapolate their outputs when dates outside of their training range are encountered and thus introduce unwarranted numerical instability.

Moving on, every item can be attributed multiple descriptive tags in natural language, as to make the image identifiable and searchable on the platform, which allows for them to be reformatted as dummy variables—Boolean indicators. Since there is no limit to the number of unique text combinations that can exist on the platform, a countable subset has to be established and a natural choice is the top-k prominent labels. In addition, by removing white spaces and hyphens, any trivial variations to a tag's text can be pooled together and their shared word stem used as a group representative, enabling more robust estimates of their ranking; although, the resulting surrogate does not account for misspelling or alternative wordings, thus providing a limited but effective improvement. For this survey, the subset is set to the size of $k = 250$.

Values that can be viewed as numeric are left unaltered, save for bijective data transformations to coerce them to downplay outliers. However, for instances that exhibit highly multi-modal distributions and are littered with unsalvageable extremities, such as the image pixel ranges and aspect ratios, these are coarse-coded using discretization; with the motivation being to separate the modes while allowing for the distributional tails to be represented without having their magnitudes affect the encodings.

When all the selected fields, for a user or an item, have been converted into the sought real-valued representations, they are pooled into a shared vector through concatenation. Thus producing a joint context—$\boldsymbol{u}_t$ or $\boldsymbol{i}^{(\mathtt{s})}_{t,a}$—that is then encoded into a latent space using the corresponding network.

3.1.1.2 Unstructured

On the `YouPic`-platform, there is virtually no restriction on what dimensions an image can take; therefore, the neural network responsible for the image vectorization should be size and shape invariant, necessitating either integrating a rescaling step or deploying a fully convolutional neural network. For this thesis, the latter approach is chosen and the model selected is the `EfficientNet-B7` by Tan and Le [40], as it reduces the outputted convolutional features with an adaptive average, before finishing with applying a logistic regression. By discarding the last layer, i.e., the classifier, the pre-trained network can embed any image into a single 4096-sized vector, which is then further reduced by the aforementioned custom encoder network to produce $\boldsymbol{i}^{(\mathtt{u})}_{t,a}$.

However, the `EfficientNet-B7` network is trained on the ImageNet dataset [41], which has an average image shape of 469×387. To ensure that the images feed to this vectorizer model are on a similar scale, the medium-sized, pre-rendered thumbnails for each image are used as representatives, as these are guaranteed to have their largest dimension capped at a maximum of 600 pixels.

3.1.2 Data Scraping

With the assistance of `YouPic`, all data are gathered from the `Newest`-flow on their platform, for two spurts during separate time frames; the first spans the entirety of

January 2022, while the other encompasses the whole of April, the same year; each of which is intended for a different purpose and, therefore, also includes additional auxiliary data crucial for their settings. Regardless, they both include the patrons' browsing histories on the flow, which entails records of their requests in form of the user ID, timestamp, and snapshots of their profile description—otherwise known as the metadata.

The first dataset, i.e., that of January, is designed for the offline evaluation of the bandit algorithms, which is actuated through a custom-built simulator. Although this pseudo-flow is described in Section 3.2, it can be summarized as mimicking the `Newest`-feed by replaying the logged user requests and have the bandits reciprocate with recommendations from an item pool, whose responses then are predicted using a neural click-model. To achieve the highest degree of veracity, this pool is constructed after that in the online architecture, presented in the aforementioned section, and contains all images uploaded from the last seven days. The items' are assumed to be characterized by their metadata and image vectors, which are then converted to the contexts $\{\boldsymbol{i}_{t,a}\}_{a\in\mathcal{A}_t}$, for any round $t \in [T]$.

In comparison, the second dataset—from April—is for the neural encoders and the click-model to jointly train on, and in doing so learn the latent spaces and users' behaviors, respectively. To enable this, each of the user requests is amended with their logged batch of recommended items and their patron's responses to each of them, all sampled from the `Newest`-flow. The item contexts are constructed as described in the previous paragraph, but here there is no pool of the most recent image additions, only the interaction history between users and items; thus producing a time series of the user-image pairs' contexts and their resulting feedback, i.e., $\boldsymbol{x}_{t,a}$ respective $r_{t,a}$.

When scraping the underlying data from the `YouPic`-servers, no logs of the historic user or item states exist; only the contemporary versions (at the time of the query) and their interplay are available. This advocates that the pre-staged contexts $\{\boldsymbol{x}_{t,a}\}_{a\in\mathcal{A}_t}$ are to be rolled back before use. However, due to time limitations, only the images' metadata could be reconstructed, while those of the users' had to be integrated as is and thus introduce a possible point of error.

3.2 Evaluation

The primary research objective is to evaluate the selection of bandits in both an online and an offline setting, studying the changes in performance resulting from incorporating the encoded image vectors. However, as pushing uninitiated algorithms to the `YouPic`-platform could present issues with the quality of their service, owing to the cold-start problem, all bandits will need to be given a warm-start from the offline evaluation. Furthermore, to ensure a smooth transition between the two settings, the online and offline system designs has to be equivalent in their software pipelines, in order to not introduce any implications when switching.

Towards this end, the shared system architecture is inspired by the Decision Service protocol presented by Slivkins [13], which provides a framework for deployment of

large-scale, CMAB-based applications. This is done by partitioning the service into a front- and a back-end process, also referred to as the selection respective the update routines in this survey. The former's purpose is to enable an installed bandit to interact with the users and supervise the data collection of their interplay. Meanwhile, the latter is responsible for updating the bandits when new samples become available, synchronizing the revised model with the front-end at regular intervals.

In the following two sub-sections, the implementation of the front- and the back-end are presented in depth. However, a visualization of the entire software pipeline, constructed using the Unified Modeling Language (UML) framework, can be found in Figure 3.1 and exists as a visual aid for the subsequent renderings.

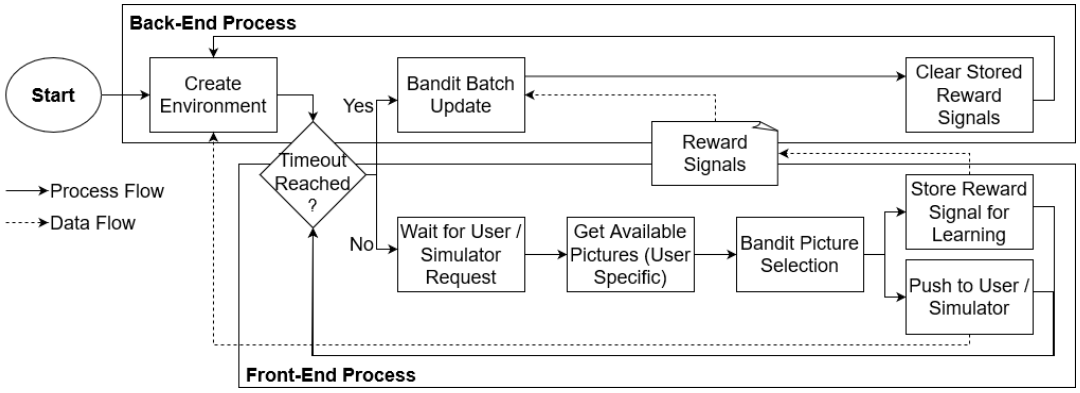


Figure 3.1: Depiction of final bandit pipeline, using the Unified Modeling Language (UML), where the front- and back-end processes corresponds to the select and update routines, respectively.

3.2.1 Front-End

The front-end system reformats the user requests as select queries for the deployed bandit, hence prompting it to recommend content from a provided set of items. In a similar spirit to the work of Li *et al.* [6], for every such incoming inquiry, a subset is uniformly sampled from a hidden pool, corresponding to the images uploaded in the last seven days, and is then presented to the decision process as that round's available choices. This is done to minimize the execution time of the algorithm when evaluating its options, as the service needs to be responsive in an online environment. For this thesis, the action set's size is set to 1024 unique instances.

When a patron makes an inquiry, indexed as round $t \in [T]$, its context $\boldsymbol{u}_t$ is immediately constructed, whereafter a random item pool $\mathcal{A}_t$ and their corresponding feature vectors $\{\boldsymbol{i}_{t,a}\}_{a\in\mathcal{A}_t}$ are fabricated, where $|\mathcal{A}_t| = 1024$. As mentioned earlier in Section 3.1.1, it is their pairwise concatenations that create the action pool's contexts $\{\boldsymbol{x}_{t,a}\}_{a\in\mathcal{A}_t}$, which are then processed by the installed bandit to generate that round's recommendations. However, as the `YouPic`-platform stipulates that each flow takes a batch of 14 items for each request, for efficient content loading, the top-k selection policy is utilized; this entails picking the top-14 items $\{a_{t,\ell}\}_{\ell=1}^{14} \subset \mathcal{A}_t$ according to its action-value estimates and policy, which are then returned instead of single best option—as in the standard setting.

Moving on, notice that no learning happens while inside the selection process, but is instead left to the discretion of the back-end. What is more, the front-end only assists through logging the bandit's resulting decision tuples $\{(\boldsymbol{x}_{t,a_{t,\ell}}, a_{t,\ell})\}_{\ell=1}^{14}$, i.e., the selected actions and their respective contexts, and transfer them to the update procedure when available. Afterward, it continues to monitor the served user for feedback, where each such interaction will trigger a supplementary data transmission that carries the recorded feedback signal $r_{t,\ell}$. Any recommended item that is graced by a response is temporarily removed from the background pool for a predefined set of time, in this case, an hour, to prevent spamming recent successful suggestions.

Whenever the back-end has an updated bandit instance, this has to be synchronized with the front and is managed by a fixed schedule that initiates the handover every hour on every hour, where the interval between two such events is referred to as a session. Once started, the updated algorithmic state is copied over to the selection process and immediately pushed to the live users, at which both systems return to their previous tasks. Furthermore, observe how this matches up with the batched bandit setting when viewing the fundamental sampling unit to be the hour-long interval, thus providing support for the approach's validity.

Before delving into the specifics of the two front-ends' implementations, a quick overview of their shared data management is in order. To speed up the computations, at the start of every session, the systems cache the contexts of the items in the background pool, thus reducing the number of fetch requests to once every hour. Furthermore, the features of any user, which is browsing the flow, are saved at their first encounter and are kept until the next synchronization event, hence similarly minimizing the overhead for constructing the contexts.

The front-ends' (partially) differing implementations stem primarily from their sources of user requests; the online accesses a live feed from the `Hot`-flow, while the offline simulates the activity using logs from the `Newest`. This enables targeted optimizations based on their respective settings, whose particularities are detailed in the following subsections. On a side note, regarding how the bandits are deployed, the reader is directed to Section 3.4.

3.2.1.1 Offline

The offline implementation mimics that of the online counterpart, which is the topic of the next section, except for its support for multiple bandit instances, but allows instead for reproducible simulations by replaying the user requests from the first dataset, as is presented in Section 3.1.2. Each inquiry is constructed as presented above, but the user's responses $\{r_{t,\ell}\}_{\ell=1}^{14}$ are sampled from a probabilistic click model; it is a custom-built feed-forward neural network that predicts, given the features of a specific user-item pair $\boldsymbol{x}_{t,a}$, the probability $p_{t,a} \in [0, 1]$ of a resulting click. For every round $t \in [T]$, the feedback signals are drawn from a set of Bernoulli random variables using the probabilities $\{p_{t,\ell}\}_{\ell=1}^{14}$ as outputted by the model, thus replicating the feedback in the online setting.

Having the primary evaluation of the bandits offline enables accelerated hyper-parameter tuning of the algorithms, without the risk of exposing real users to

possibly suboptimal solutions; however, this is the topic of Section 3.4. For instance, the simulation can be speedup by pre-computing all of the user and item contexts beforehand, which then circumvents the need for the feature construction step in the front-end system, thus saving computations and execution time.

3.2.1.2 Online

The integration of the online front-end is supplied by `YouPic AB`, as to ensure their quality of service, but follows the principles laid out in the previous parent section. To minimize any evaluation bias stemming from differing installment times, the online system supports multiple deployed algorithms, which the offline counterpart does not—that is, A/B-testing. For this thesis, all contextual algorithms are launched to live users for the same period, save for those that can not be given a warm-start by the offline simulator.

These bandits are assigned a certain fraction of all patrons on the `Hot`-flow, the specific value is intentionally withheld due to security concerns of the industrial partner, which are further partitioned into separate service buckets—one for each deployed decision process. Every such endpoint is allocated individuals using their ID numbers' congruence class representatives as the bucket index specifiers, who are congruent modulo the number of deployed instances. Moreover, as the identifiers make a monotonic increasing sequence of integers, by definition of `YouPic AB`, this dichotomization should yield a fairly uniform distribution of designations.

3.2.2 Back-End

The back-end system is virtually the same in both the offline and the online settings. Whenever a decision is made, by the bandit in the front-end, the update process is piped the consequent decision tuples $\{(\boldsymbol{x}_{t,a_{t,\ell}}, a_{t,\ell})\}_{\ell=1}^{14}$, i.e., the selected actions and their corresponding contexts, where it is then retained in a temporary database, called the Join Server, for a predefined length of time. Therefrom, the service performs one of two procedures: the timer runs out and the feedback is assumed to be zero, or the relevant user reciprocates with a like and is assigned a positive signal. Whatever the conclusion, the feedback signal $r_{t,\ell}$ will be appended to its matching decision tuple, thus constituting an outcome tuple $(\boldsymbol{x}_{t,a_{t,\ell}}, a_{t,\ell}, r_{t,\ell})$ and makes an observation to update the internal bandit with.

3.3 Modeling

As previously mentioned, this thesis employs neural networks as modules in the bandit pipeline, i.e., the context encoders and the click model. This section aims to present their architectural designs and display how they are jointly trained for higher sample efficiency and coherence.

3.3.1 Architecture

All network instances are of the feed-forward class, as no grid-like topology is present in the input features. A hidden unit is defined as a composition of a summation, a batch normalization, a leaky rectified linear unit (ReLU) activation, and a dropout operation; a group of which constitutes a layer and is the core building block in the coming neural architectures. In other words, any layer mentioned is assumed to follow the aforementioned description, unless specified otherwise. Observe that the number of hidden units and the dropout proportion are considered hyper-parameters, and as such are subject to optimization—the details of which can be found in Section 3.3.2.

3.3.1.1 Feature Encoders

Adopting pre-trained neural networks as feature maps is not a novel concept; in fact, it is promoted in the book of Lattimore and Szepesvári [5]—see page 239 for reference. In their work, by training a feed-forward regression network to predict the feedback signal, one can discard the output layer and the remaining neural model can encode the input features such that they are linear with respect to the target signal. As long as the data used to train the feature encoder does not overlap with that used to evaluate the bandits, this approach presents an opportunity for robust distilling of the original features' information and representation.

For this thesis, three different context encoders are employed, as to allow independent embedding of the three user and item contexts, and are responsible for producing the finalized vectors $\boldsymbol{u}_t$, $\boldsymbol{i}_{t,a}^{(\mathtt{s})}$ and $\boldsymbol{i}_{t,a}^{(\mathtt{u})}$. In the interest of efficiency and time, every feature map will be comprised of a single-layer network, whose optimizations are discussed in Section 3.3.2.

3.3.1.2 Click Model

To interpret the model's output as the probability of a click, given a user-item pair, the prediction needs to assume values in the range between zero and unity. This is accomplished by replacing the output layer's activation with the logistic function, which modifies the original regression problem into one of binary classification. The model is a feed-forward neural network with a single hidden layer, as to allow non-linear interactions between the patron and the image, whose number of hidden units is also found through a grid-search.

3.3.2 Training

The aforementioned neural modules are collectively trained using a super model, which is outlined in Figure 3.2. This shared optimization problem allows for coordinated learning of the parameters and higher data efficiency than training the individual parts separately. However, as the main output is nonlinear, due to it being a classifier, there is no guarantee that the encoded features are linear with respect to the feedback signal. Therefore, an auxiliary objective is introduced, whose sole purpose is to ensure that encoded features contain an affine relationship to the response.

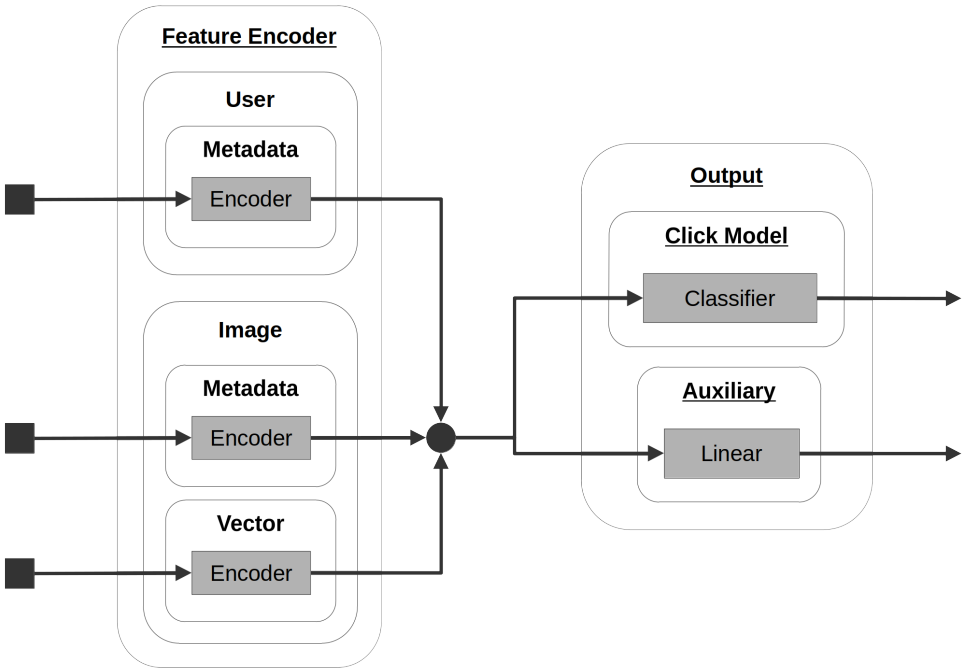


Figure 3.2: Sketch of the employed super model for training the feature encoders and the click model jointly. The shaded, sharp-edged, labeled boxes define the feed-forward modules of the neural network, whose in- and outputs are defined by their respective arrows' directions. The circular node denotes the concatenation operation, while the right-side cubes signify the input vectors to be encoded and correspond to the $\boldsymbol{u}_t$, $\boldsymbol{i}_{t,a}^{(\mathrm{s})}$ and $\boldsymbol{i}_{t,a}^{(\mathrm{u})}$ variables, respectively, when enumerating them in descending order.

The super network is optimized using the second dataset from Section 3.1.2, where its' examples are randomly partitioned into three sets: training, validation, and testing—using the respective target proportions of 80%, 10%, and 10%. The main output function is controlled using a binary cross-entropy loss function, while the auxiliary is by a mean square error; both of which are weighted using the inverse category proportions to combat class imbalance. Thereafter, this multi-objective is conjoined through their average, resulting in a pooled loss function that is used during training, which stretches for 64 epochs.

3.3.2.1 Hyper-Parameter Tuning

To prevent overfitting and the identification of sufficiently sized latent spaces for neural modules, an exhaustive grid-search is performed to optimize the number of hidden units and their dropout rates. The context encoders (i.e., for the user metadata, the item metadata, and the image vector) assume values in the set $\{8, 16, 32\}$, which corresponds to the same set of values that the click model can adopt. In addition, the dropout probability is evaluated at the points $\{0.0, 0.5\}$. This means that there are 54 unique permutations to be tested, where the constellation that achieves the lowest, average validation loss will see its neural modules integrated into the methodology. A summary of the results can be found in Appendix A.

3.4 Bandit Algorithms

This survey's selection of bandit algorithms includes both contextual and context-free representatives, where the latter are included for reference and only used in the offline simulation. In this section, their definitions are briefly presented, along with some of their theoretical aspects.

As mentioned earlier, the offline evaluation allows for hyper-parameter tuning of the bandits' control variables, whose results' validity depends on the quality of the click model. In similarity with the estimation of the neural modules' architectures, the optimization for every bandit is done through a grid-search; where every tested permutation is run for five trials, to get the average performance.

In the following definitions, the same notation as presented in Chapter 2 and Section 3.1 is employed; however, for ease of reading, this paragraph provides a brief reminder. Recall that $\boldsymbol{x}_{t,a} \in \mathbb{R}^p$ is the feature context for arm $a \in \mathcal{A}_t$ and round $t \in [T]$, which is the concatenation of the user's $\boldsymbol{u}_t \in \mathbb{R}^{p_u}$ and image $\boldsymbol{i}_{t,a} \in \mathbb{R}^{p_i}$ features, where $p = p_u + p_i$ and $p, p_u, p_i \in \mathbb{N}$. Moreover, $\boldsymbol{i}_{t,a}$ is partitioned into two vectors that contain the image's metadata $\boldsymbol{i}^{(\mathtt{s})}_{t,a}$ and convolutional features $\boldsymbol{i}^{(\mathtt{u})}_{t,a}$, which corresponds to the structured and unstructured data, respectively. When specifying that bandits evaluated without the image vector, this entails that the latter is withheld from the contextual bandits, hence implying $\boldsymbol{i}_{t,a} = \boldsymbol{i}^{(\mathtt{s})}_{t,a}$.

3.4.1 ε-Greedy

Being one of the most well-known reinforcement learning policies, for $\varepsilon \in [0, 1]$, any bandit adopting the ε-Greedy will follow the greedy strategy with probability $1-\varepsilon$, where greedy equates to selecting the option that has the largest action-value estimate, otherwise, it selects an arm uniformly at random [5]. The action-value function is what decides whether the algorithm is contextual or not, and for this thesis one of each is tested, which is detailed in the following subsections. Moreover, as both instances incorporates the hyper-parameter ε, they both evaluate the points $\{10^{-3}, 10^{-2}, 10^{-1}\}$ when performing the hyper-parameter optimization.

3.4.1.1 Tabular

A tabular value function implies that every arm's expected return is modeled by its sample average, as identified by the action's index. Mathematically, assuming $t \in [T]$, this is formulated as

$$\hat{\mu}_{n(a)+1}(a) = \hat{\mu}_{n(a)}(a) + \frac{1}{n(a)+1}\Big(R_t(a) - \hat{\mu}_{n(a)}(a)\Big), \tag{3.3}$$

where $\hat{\mu}_n(a)$ is the n^{th} sample average of action a, $R_t(a)$ is the reward from the arm a (at round t) and $n(a)$ counts the times arm a has been selected [28].

The tunable parameters are the initial action-value estimate $\hat{\mu}_0(a)$ and the probability ε of a random selection. Regarding the hyper-parameter optimization, the grid-search

covers the points $\{0, 0.25, 0.5, 0.75, 1\}$ for the former variable, while the latter's have already been presented; the result of which is available in Appendix B.1.

3.4.1.2 Linear

A natural extension of standard ε-`Greedy` bandit is to curate a joint linear model instead of a tabular action-value function, which entails making the algorithm contextual by assuming

$$\hat{\mu}(a) = \boldsymbol{x}_{t,a}^{\intercal}\boldsymbol{\theta}, \tag{3.4}$$

where $\boldsymbol{\theta}$ is the unknown coefficient vector, as shown in Section 2.2.1. As the resulting algorithm utilizes a linear model as its action-value function, the only tunable parameter is the probability of a random selection ε; whose grid-search points can be seen in the parent section. The complete results are presented in Appendix B.2.

3.4.2 Thompson Sampling

Paraphrasing the book of Slivkins [13], without delving into the theory, the key property of the basic `Thompson Sampling` algorithm is its inclusion of the Bayesian paradigm. By modeling each arm's feedback distribution by a parametric prior, these can be queried for a random sample of the action-values and used in conjunction with the greedy policy to select an arm; where the selected action's distribution is then updated using the resulting feedback signal, and the posterior can then be used in the next round.

For binary feedback signals, every arm's feedback signal is assumed to be a Bernoulli random variable with an unknown mean signal, making the Beta distribution a suitable prior. Its density is parameterized by two concentration parameters $\alpha > 0$ and $\beta > 0$, alternatively known as the pseudo-counts.

These variables are treated as hyper-parameters and both concentration spaces are evaluated at the points $\{1, 10, 100\}$, and the result can be found in Appendix B.3. Perhaps this algorithm is a bit out of scope for this thesis, as it is a Bayesian bandit, but it is included for reference.

3.4.3 UCB

In essence, the upper confidence bound (UCB) method explicitly models the uncertainty of each action-value estimate and additively combines these into an optimistic upper bound for each arm's expected return, which is then fed to the standard greedy policy for selection [5]. Mathematically, these quantities can be defined as

$$\hat{v}(a) = \hat{\mu}(a) + \delta \cdot \hat{\sigma}(a), \tag{3.5}$$

where $\hat{\mu}(a)$ is the sample average of the arm a, $\hat{\sigma}(a)$ is the upper confidence bound and δ is the exploration factor—i.e., it determines how much the confidence bound influences the selection.

For the hyper-parameter tuning, all context-dependent algorithms have the exploration parameter δ as a free instance; while for the context-free case, it assumes a

fixed constant. When applicable, each instance will test the points $\{10^{-1}, 10^{0}, 10^{1}\}$, which is then added to the grid-search's permutations when other hyper-parameters are present.

3.4.3.1 UCB1

This context-free implementation, of the UCB paradigm, tends to be one of the first textbook examples students get acquainted with when beginning to explore the topic of multi-armed bandits [5, 13]. It follows the formula given by Equation (3.5), where action-value estimate is as defined by Equation (3.3) and the optimistic uncertainty bound as

$$\hat{\sigma}(a) = \sqrt{\frac{2 \log N}{n(a)}}, \tag{3.6}$$

where N is the total number of rounds so far and $n(a)$ is a counter for the action a. When $n(a) = 0$, i.e., a novel arm, $\hat{\sigma}(a)$ is assumed to be infinite, which effectively makes it always explore the new options. The exploration factor δ can be included, but it is usually assumed to be unity.

Observe that this bandit algorithm has effectively no hyper-parameters to tune, except the initial action-value $\bar{\mu}_0$ that tests the same points as given in Section 3.4.1.1. Nevertheless, the five-trial offline simulation results can be found in Appendix B.4.

3.4.3.2 Linear

In similarity with the `Linear` ε`-Greedy` bandit in Section 3.4.1.2, one can employ a joint linear model as the action-value function in Equation (3.5); i.e., the expected average $\hat{\mu}(a)$ is estimated by Equation (3.4). The algorithms imposing linear restraints in this fashion have the collective name `LinUCB` [6], which is the label used for the bandit of this section.

Now, using matrix notation, the optimistic upper bound is formalized as

$$\hat{\sigma}(a) = \sqrt{\boldsymbol{x}_{t,a}^{\intercal} A^{-1} \boldsymbol{x}_{t,a}} \qquad \text{where} \qquad A = X_{t-1}^{\intercal} X_{t-1} + \lambda I, \tag{3.7}$$

with $X_{t-1} \in \mathbb{R}^{(t-1)\times p}$ is the design matrix, with $p \in \mathbb{N}$ features, and $\lambda > 0$ is a regularization constant. The hyper-parameter λ is evaluated the points $\{10^{-1}, 10^{0}, 10^{1}\}$ and its complete search result can be found in Appendix B.5.

3.4.3.3 Disjoint Linear

One of the most famous contextual bandit algorithms, whose intended environment is that of recommender systems—an ever-changing action set—is the `Disjoint LinUCB` by Li *et al.* [6]. It stipulates that every arm's mean feedback signal is given by an unknown linear function, i.e.,

$$\hat{\mu}(a) = \boldsymbol{x}_{t,a}^{\intercal} \boldsymbol{\theta}_a, \tag{3.8}$$

where $\boldsymbol{\theta}_a$ is the coefficients for the arm a; in other words, every action has its own regression model. The optimistic bounds are constructed as

$$\hat{\sigma}(a) = \sqrt{\boldsymbol{x}_{t,a}^{\intercal} A_a^{-1} \boldsymbol{x}_{t,a}} \qquad \text{where} \qquad A_a = X_{t-1,a}^{\intercal} X_{t-1,a} + \lambda I, \tag{3.9}$$

$X_{t-1,a} \in \mathbb{R}^{(t-1)\times p}$ is the design matrix for arm a and $\lambda > 0$ is a regularization constant.

The benefit of this algorithm is its intentional design for usage inside a recommender system and allows for each arm to specialize on a specific user segment, as given by the $\boldsymbol{u}_t$, which the joint counterpart can not do. As all of the hyper-parameters are the same as in the previous section, their tuning follows the same strategy. The result of the search is available in Appendix B.6.

3.4.3.4 Hybrid Linear

In the same work of `Disjoint LinUCB`, Li *et al.* [6] also introduced an extension called `Hybrid LinUCB`, which combines the joint and disjoint approaches. This is realized by assuming that

$$\hat{\mu}(a) = \boldsymbol{u}_t^\intercal \boldsymbol{\theta}_a + \boldsymbol{z}_{t,a}^\intercal \boldsymbol{\theta}, \tag{3.10}$$

where the coefficient vector $\boldsymbol{\theta}$ is shared between all actions, while $\{\boldsymbol{\theta}_a\}_a$ are all dedicated to their respective arm a, cf. 3.4.3.2 and 3.4.3.3, respectively. Furthermore, to compute the optimistic upper bound $\hat{\sigma}$, Li *et al.* [6] relies quite heavily on block matrix inversion techniques, whose resulting derivation and formula are too complex to include here; however, the interested are directed to their original report.

Observe that in Equation (3.10) the context is not denoted by $\boldsymbol{x}_{t,a}$, but by $\boldsymbol{u}_t$ and $\boldsymbol{z}_{t,a}$, which is to enable the parametric partitioning. The former, $\boldsymbol{u}_t$, is the user context from before, whilst $\boldsymbol{z}_{t,a}$ symbolizes the second-order pair-wise interactions between the user and the item contexts $\{\boldsymbol{i}_{t,a}\}_a$, using the terminology from statistical linear models.

The hyper-parameters of this algorithm coincide with those of the two preceding algorithms, which are the exploration parameter δ and the regularization parameter λ; both of which are tuned according to the same scheme as in the others. The results can be found in Appendix B.7.

3.4.3.5 Neural

Quite recently, Zhou *et al.* [18] proposed the `NeuralUCB` algorithm, which conjoins the UCB framework with that of neural networks. Following the definition in Section 2.2.1 and using the notation from Equation (3.5), their achievement entails successfully modeling $\hat{\mu}(\boldsymbol{x})$ as a feed-forward network and proving this integration theoretically. This allows them to tap into the function approximation ability of neural networks, resulting in the bandit not imposing any restrictions on the feedback signal.

Though a complete description of this algorithm is a bit out of scope for this thesis, a brief description of it is in order; curious readers are directed to the original article of Zhou *et al.* [18] for more information. The internal network is of the regression class with variable depth and `ReLU` activations, but due to theoretical aspects, all hidden layers have the same number of neurons; for this thesis, these values are kept at the authors' default, which corresponds to a depth of two with 20 hidden units each. For this survey, the model is trained at a fixed schedule that coincides with the end of each session—as defined in Section 3.2—at which it runs for 512 epochs

using the gradient descent algorithm together with the mean squared error (MSE) loss function.

Other than an explore factor δ and a regularization parameter λ, whose explanations and grid-search points follow from their respective descriptions above, the learning rate γ is also treated as a hyper-parameter and is evaluated at the points $\{10^{-3}, 10^{-2}, 10^{-1}\}$. The grid-search results can be found in Appendix B.8.

4

Results

This chapter presents the results of this thesis. Starting with the offline evaluation, first, the results from the hyper-parameter tuning are presented, followed by their representatives' performances measured by the estimated feedback rate and total regret. Thereafter, the contextual algorithms' online deployment is addressed.

4.1 Offline

As mentioned by this chapter's opening, this section contains the offline results of the hyper-parameter tuning and the respective bandits' performances. In Section 4.1.1 the former's outcome can be seen, while the latter is presented in Section 4.1.2.

4.1.1 Hyper-Parameter Tuning

Table 4.1 presents the outcomes of the hyper-parameter tuning, as described in Section 3.4. The values selected corresponds to those with the highest click through rate, but further details about the algorithms' grid-searches can be found in Appendix B.

Table 4.1: The found hyper-parameters for the contextual and context-free bandits when using the image vector $\boldsymbol{i}_{t,a}^{(\mathrm{u})}$, if applicable. The definition of the parameters can be found in Section 3.4.

	Bandit	Parameters						
		ε	$\hat{\mu}_0$	α	β	δ	λ	γ
Contextual	`Linear ε-Greedy`	0.001	-	-	-	-	0.1	-
	`LinUCB`	-	-	-	-	0.1	1.0	-
	`Disjoint LinUCB`	-	-	-	-	0.1	0.1	-
	`Hybrid LinUCB`	-	-	-	-	0.1	10.0	-
	`NeuralUCB`	-	-	-	-	0.1	1.0	0.01
Context Free	`Tabular ε-Greedy`	0.001	0.25	-	-	-	-	-
	`Thompson Sampling`	-	-	1	100	-	-	-
	`UCB1`	-	1.00	-	-	-	-	-

4.1.2 Performance

In Table 4.2, the aggregated results of each best-performing bandit algorithm from their respective hyper-parameter optimization are presented, whose method is described in Section 3.2. Furthermore, the metrics of consideration are the cumulative feedback rate and regret, where the complete findings of the grid-search can be found in Appendix B.

Table 4.2: Results of the bandit algorithms that achieved the highest average click rate $\bar{\mu}$, where (a) and (b) tabulates the statistics for the contextual and context-free bandits, respectively; furthermore, their corresponding parameterization can be found in Table 4.1. The vector $\boldsymbol{i}_{t,a}^{(\mathrm{u})}$ represents the whether the optional convolutional features are utilized (✓) or not (✗), while $\bar{\sigma}$ is the standard error in regards to $\bar{\mu}$. Notice that these estimates are from five independent trials using the offline simulator.

(a) Contextual

	Feedback (%)				Regret (10^3)			
$\boldsymbol{i}_{t,a}^{(\mathrm{u})}$	✗		✓		✗		✓	
Bandit	$\bar{\mu}$	$\bar{\sigma}$	$\bar{\mu}$	$\bar{\sigma}$	$\bar{\mu}$	$\bar{\sigma}$	$\bar{\mu}$	$\bar{\sigma}$
`Linear` ε`-Greedy`	19.55	0.07	26.52	0.05	212.57	0.46	145.75	0.37
`LinUCB`	19.52	0.05	26.49	0.04	212.73	0.76	146.11	0.25
`Disjoint LinUCB`	22.24	0.11	22.24	0.09	186.95	1.03	187.06	0.80
`Hybrid LinUCB`	27.13	0.03	31.25	0.12	136.81	0.71	91.25	1.65
`NeuralUCB`	23.60	0.14	32.84	0.14	172.60	1.08	73.52	1.94

(b) Context-Free

	Feedback (%)		Regret (10^3)	
Bandit	$\bar{\mu}$	$\bar{\sigma}$	$\bar{\mu}$	$\bar{\sigma}$
`Tabular` ε`-Greedy`	22.05	0.10	188.85	0.95
`Thompson Sampling`	22.03	0.11	192.88	1.41
`UCB1`	17.51	0.03	234.18	0.42

To be able to analyze and compare the learning rates of the algorithmic selection, their cumulative average feedback rates and regrets are traced, and the resulting profiles can be found in Figure 4.1 and Figure 4.2. The bandits are sorted into one of the three canvases, one for the instances including the image vector $\boldsymbol{i}_{t,a}^{(\mathrm{u})}$, one for those who exclude it and one for the context-free alternatives. For the curious, the profiles of all tested hyper-parameter configurations can be found in Appendix B.

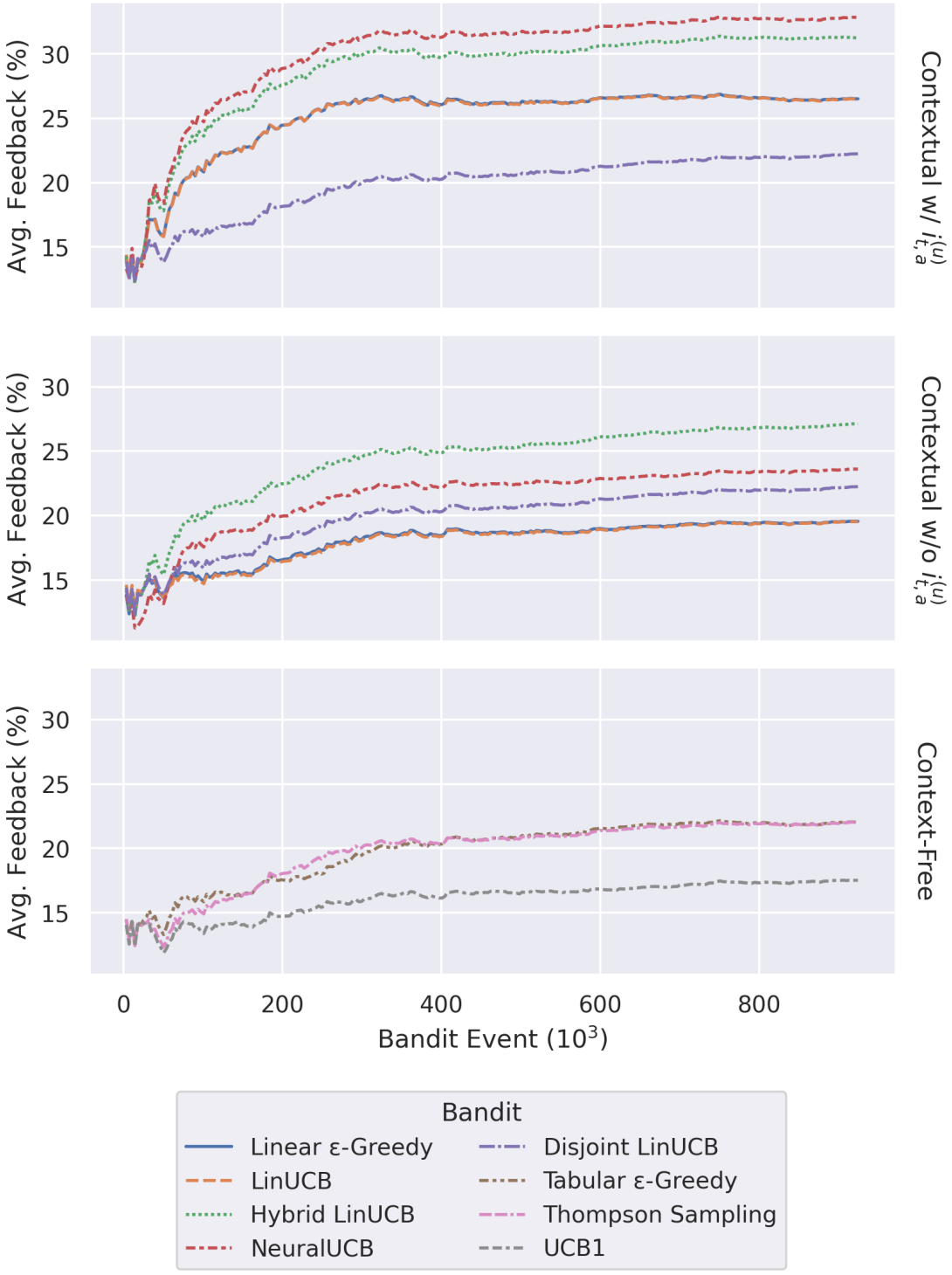


Figure 4.1: The cumulative average feedback profiles of the best-performing bandit algorithms from each hyper-parameter search, categorized after if they exclude or include the image vector $\boldsymbol{i}_{t,a}^{(\mathrm{u})}$, along with one group for the context-free instances. On a side note, observe how given the correct hyper-parameterization, some of the context-free algorithms outperform two of the contextual when excluding the image features.

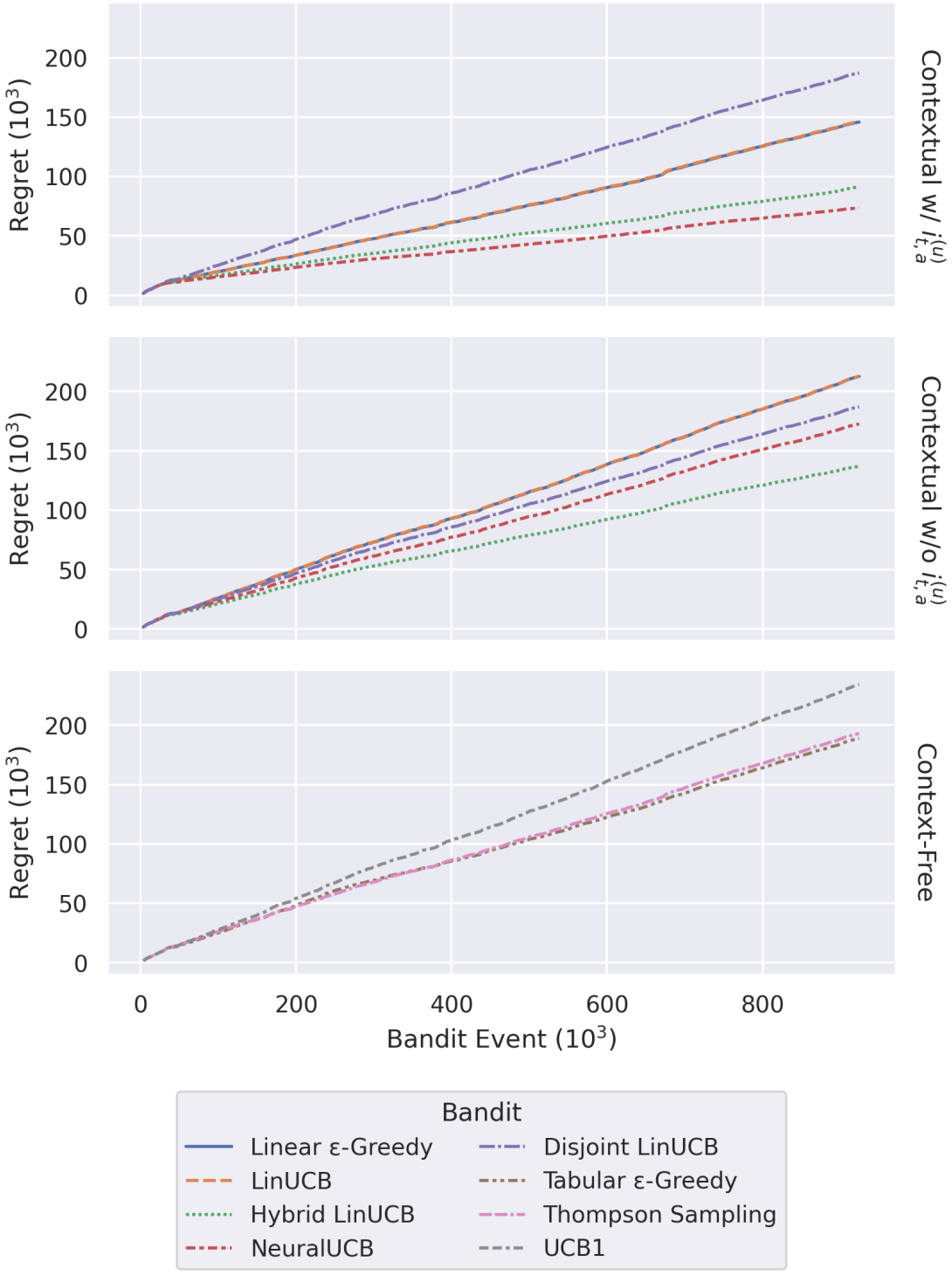


Figure 4.2: The regret profiles of the best performing-bandit algorithms from each hyper-parameter search, categorized after if they exclude or include the image vector $\boldsymbol{i}_{t,a}^{(\mathrm{u})}$, along with one group for the context-free instances. Notice that, in this recommender system setting, there will be a constant influx of regret as new arms are constantly added, which explains the relatively linear profiles of all the tested algorithms.

4.2 Online

As mentioned in Section 3.2, all contextual bandit algorithms are deployed online for evaluation, using the bandits from the offline evaluation as a warm-start. However, note that the `Disjoint LinUCB` is not deployed and none of the context-free alternatives are likewise, as they fail to generalize across arms and capture their setting by definition, respectively.

In similarity with Section 4.1, a table with the estimated feedback rates and a visualization of their respective cumulative traces are presented in the following section. For comparison, the feedback rate for the second dataset—that of April 2022—is estimated to be 9.27%.

4.2.1 Performance

In Table 4.3, the estimated feedback rates for the deployed algorithms are presented, with and without the image vector $\boldsymbol{i}_{t,a}^{(\mathrm{u})}$. The scores' underlying samples have been captured as described in Section 3.2, where the deployment period started at the 30^{th} of December 2022 and ended at the 16^{th} of January 2023.

Table 4.3: In this table, the online feedback rates for all the contextual algorithms, except for the `Disjoint LinUCB`, whose shared deployment period spans the 30^{th} of December 2022 to the 16^{th} of January 2023, can be seen. All bandit have separate instances including (✓) respective excluding (✗) the image vector $\boldsymbol{i}_{t,a}^{(\mathrm{u})}$.

	Feedback (%)		User (#)	
$\boldsymbol{i}_{t,a}^{(\mathrm{u})}$	✗	✓	✗	✓
Bandit				
`Linear` ε`-Greedy`	1.60	2.41	78	73
`LinUCB`	2.19	0.83	252	88
`Hybrid LinUCB`	0.66	4.95	76	80
`NeuralUCB`	2.68	2.26	235	79

As to enable visual comparison of the learning agents, their cumulative feedback rates are drawn and can be seen in Figure 4.3. The different lengths of the figure's profiles are due to their uniquely assigned users and their behaviors; furthermore, as the `Hot`-flow generally sees the lowest activity of all feeds on the platform, any specific patron's actions can strongly influence the average's trajectory.

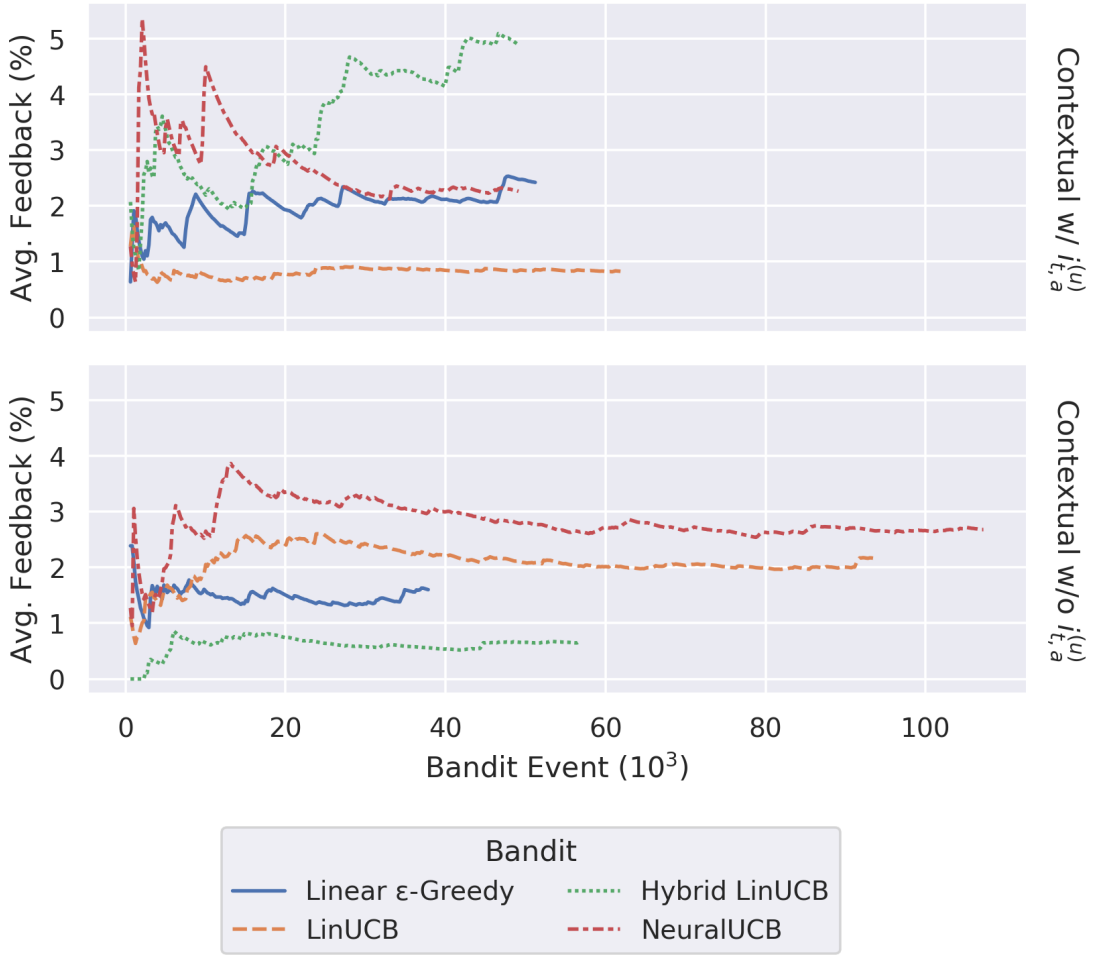


Figure 4.3: This figure depicts the feedback rate traces of online deployed bandit instances, which includes data from the 30^{th} of December 2022 to the 16^{th} of January 2023. Observe the different amount of activity the algorithms sees, which originates from that their assigned users are random, but unique, and their behaviors varies. In addition, as the `Hot`-flow has the least amount of activity on the platform, any specific consumer's conduct can significantly alter the profiles' trajectories.

5

Conclusion

This chapter discusses the results in Chapter 4 and their importance in relation to the research question—whether or not unstructured data provide any benefit to contextual multi-armed bandits in a recommender system setting. First, the results of the offline and the online evaluations are reviewed and compared, before debating if these sorts of algorithms are applicable for web-based personalization. Then, a review of the work's limitations, ethical considerations, and directions for future research are considered, after which the thesis comes to a close with a final conclusion.

5.1 Discussion

This survey aims to answer the question of whether features extracted by pre-trained neural networks can provide a competitive edge for contextual multi-armed bandits when employed within a recommender system. In collaboration with the industrial partner `YouPic AB`, both an offline and an online evaluation have been conducted, which was enabled by data from their online social-media platform and production system.

The offline experiments support that the unstructured data—image features extracted by `EfficientNet-B7`—can provide a competitive edge. This can be seen in Table 4.1, which shows that all bandits' accumulated clicks increase by incorporating the convolutional features, except the `Disjoint LinUCB` which will be discussed later. In addition, the results show that the contextual algorithms outperform their context-free alternatives when utilizing this additional information; although, the gap between the best-performing context-free MAB and that of the worst contextual instance is quite small. Moreover, when excluding the additional features, the offline simulation shows that the context-free variants can achieve similar scores as their top contextual contenders.

In comparison, online evaluation exhibits vague indications of validating the observed trend from the offline environment; however, the live trials encountered issues with insufficient and disproportionate sample sizes, which makes their results potentially inconclusive. As a sanity check, the algorithms' reported feedback rates are contrasted with the average from the `Newest`-flow, which is approximately two times that of the A/B-tests' highest recorded value; although, that is probably due to their differing predispositions that arise from their disjoint source flows and associated clienteles.

In other words, based on these outcomes alone, there are clues seemingly affirming the thesis' hypothesis, but whether or not convolutional features are beneficial to multi-armed bandits in recommender system settings is an open question.

Now, the discussion shortly continues onto the results of the offline and the online evaluations. Thereafter, a quick review of the implementation details of this project and its implications are given, before moving on to debating the integration of bandits in recommender systems. Then some of this work's limitations, ethical considerations, and future work are brought forth.

5.1.1 Results

This section addresses the results of the offline and online evaluations. Their order of presentation follows that used in Chapter 4, but do note that only the contextual MABs are pushed to live users, save for the `Disjoint LinUCB`.

5.1.1.1 Offline

The offline evaluation aims to survey the performance of bandit algorithms when placed in a recommender system setting and study the effects brought by introducing convolutional features in its input. At the same time, it also serves for identifying each of their best-performing hyper-parameterization, which is selected as their representative and used to provide a warm-start—if designated for the online deployment. Through this, the algorithms' strengths and weaknesses can be uncovered, along with their resulting quirks to the recommender system setting, which some were not designed for.

In the following paragraphs, each of the bandit algorithms' selected representatives is discussed, their key findings interpreted and, if they are contextual, the effects brought by the image vector are reviewed. Additionally, their presentation order follows that in Section 3.4 and, towards the end, any systematic findings, which all the bandits share, are examined.

The parameterization of the `Tabular` ε`-Greedy` representative could be seen as counter-intuitive, as the lowest available value of ε is adopted, resulting in little to no random exploration, and its default action-value $\hat{\mu}_0$ is found to be optimal at 0.25. This effectively makes the algorithm explore through optimistic initialization; i.e., setting a positive initial value that is then corrected as the arm is sampled, instead of the intended uniformly stochastic selects. In the setting of recommender systems, where the action set is constantly changing, this ensures that its exploration can target new alternatives instead of accidentally sampling well-known items.

Then there is its contextual variant, the `Linear` ε`-Greedy` process, whose offline performance is one of the worst in the entire survey when excluding the convolutional features; however, when including it, the bandit sees its feedback rate increase with approximately seven percentage points and its regret reduced by roughly 30%. Its hyper-parameter ε assumes the same value as that of the tabular counterpart, which is the lowest possible and the regularization factor λ follows suit. Observe that the contextual algorithm's definition does not allow for optimistic initialization,

but instead models the feedback signal through a joint linear regression problem, hence making it unclear how it explores. The answer appears to be that before the coefficients converge, it effectively samples at random for the first session; whereafter, when enough data has been collected, it can estimate the linear system's solution and can thus exploit the learned patterns in the contexts.

The `Thompson Sampling` algorithm has its Beta concentrations settle on $\alpha = 1$ and $\beta = 100$, hence shifting the initial prior's density towards zero, with a quite high certainty. In other words, the offline simulation seems to have favored informed priors over any clueless alternative, which for example occurs when both α and β are closer toward unity. This ensures that new arms' do not dominate the selection when introduced, as the real feedback rate is at the lower end of the feedback rate spectrum and too much uncertainty in the prior would make the bandit less likely to exploit—due to new arms constantly arriving.

Moving on, the `UCB1` algorithm has both the lowest feedback rate and the highest total regret observed out of all tested bandit representatives, making it the worst-performing multi-armed bandit tested in the simulated recommender setting. If one consults the complete records in Appendix B.4, the reported profiles and estimates there are virtually the same for every tested value of $\bar{\mu}_0$ in the grid-search. This implies that the representative's hyper-parameter $\bar{\mu}_0 = 1.0$ is not necessarily chosen due to better average performance, but due to noise introduced by the offline simulator. However, this behavior is to be expected of the `UCB1` algorithm, as its definition of the optimistic upper bound stipulates that unexplored arms have infinite potential, as no samples have been drawn yet. Put simply, it is forced to test each action at least once, making it effectively adopt an exploration-first policy. Moreover, when coupling that with a recommender system setting, the bandit will seldom get the chance to exploit, as there is a constant influx of novel options.

For the bandit labeled `LinUCB`, it achieves a mean score equal to that of the `Linear` ε`-Greedy`, even with similar standard errors, while employing an exploration factor $\delta = 0.1$ and a regularization parameter $\lambda = 1.0$. With such a low value of δ, it downplays the importance of exploration and favors its linear model's prediction for exploitation, which is very similar to the situation of the ε-greedy relative.

This is also observed for the disjoint variant, `Disjoint LinUCB`, where its optimal parameterization is found to be when both δ and λ equals 0.1. Moreover, when studying the algorithm's results, notice how the presence of the convolutional features does not appear to affect its performance, as the scores are virtually the same for both categories. This could be explained by the disjoint action-value estimator—a separate regression problem for each arm—and would imply that this extra information becomes redundant if the algorithm can not generalize across the arms. In other words, the content of an image becomes less relevant if its composition can not be compared to other instances. On a side note, notice how the algorithm settles on the lowest regularization value possible, i.e., $\lambda = 0.1$; this should have been expected, as this bandit is less sample-efficient than the others in the survey, due to every sample is prescribed to only one arm and its linear model, hence prompting it to maximize every observation's utility and influence.

The amalgamation of the two preceding bandits is the `Hybrid LinUCB` algorithm, which has both local and global parameters in its linear action-value model. It attains the next best average feedback rate and total regret of all algorithms, when including the image vector $\boldsymbol{i}_{t,a}^{(\mathrm{u})}$, and comes out on top if excluding it. Its exploration and regularization factors assume the values 0.1 and 10.0, respectively, making it also dismiss the exploration, but this version fancies regularization, which results in it being less susceptible to small noise in its observations. Furthermore, notice that it appears to get a boost from the image vector, hence implying that joint and hybrid action-value models are benefited from its inclusion.

The last bandit, of this survey, is the `NeuralUCB` algorithm, which has the best performance of all decision processes when including the convolutional features; however, when excluding the image vector $\boldsymbol{i}_{t,a}^{(\mathrm{u})}$, it falls to second place—beaten by `Hybrid LinUCB`. Its hyper-parameters were found to be optimal with $\delta = 0.1$, $\lambda = 1.0$, and $\gamma = 0.01$, where the last hyper-parameter is the learning rate, used in training the action-value neural network.

Before moving on to the online discussion, notice the trend among the stochastic bandit representatives where they adopt the respective hyper-parameterization corresponding to the least explorative behavior. For example, all contextual `UCB`-based algorithms settled on employing the lowest allowed exploration factor, $\delta = 0.1$, while those of the ε`-Greedy` type selected the smallest available probability, $\varepsilon = 0.001$. Even the `Thompson Sampling` algorithm, which is a Bayesian multi-armed bandit, picks a representative whose solution has a narrow spread in its prior density—i.e., relatively low uncertainty. The only exception is the `UCB1` bandit, whose shortcomings have already been discussed and is therefore an expected outlier.

In other words, all functioning bandit algorithms have downplayed the importance of exploration, in favor of exploitation, but for what reason? The culprit would appear to be the item pre-selection introduced by the recommender system setting, which uniformly samples a fixed-sized subset of items for the decision processes to evaluate, every round $t \in [T]$. As this probabilistic nomination is unguided, it negates the need for an explicit probing of novel or under-explored arms, hence explaining the observed tendency of the representatives. This should perhaps have been predicted, but it shows that multi-armed bandit needs special care when installed as recommender engines to function properly. On a related note, notice how even the `Disjoint` and `Hybrid LinUCB` algorithms exhibit this characteristic, despite the fact that they were designed for the same setting by Li *et al.* [6].

5.1.1.2 Online

Studying the deployed algorithms' scores in Table 4.3, none of these rates reaches the same levels as their respective offline counterparts, all of which have ratios an order of magnitude higher in value. This discrepancy can probably be attributed to a distributional shift in the environment or an ingrained modeling bias from the offline setting, which makes themselves apparent when migrating to the online analog. For instance, any error in the click-model's feedback signal could become embedded in the agents' respective action-value functions, thus potentially necessitating an initial

acclimation phase when installed—given that their contrast is too significant.

Moreover, even the largest of these feedback rates is only half of that independently measured on the `Newest`-flow—estimated using the second dataset of Section 3.1.2—and reaches a score of 9.27%. However, according to `YouPic AB`, this particular feed attracts a comparably unique clientele than the rest of the platform, including the `Hot`-stream employed by the online evaluation. The majority of the users, found browsing this relatively unfiltered medium, are proactively searching for networking opportunities, where potential relationships are usually probed by being quite liberal with positive feedback. In contrast, the flow used by the online experiments is the platform's least visited page, which provides some clarity on why its feedback averages are smaller.

Moving on, the estimated feedback rates in Table 4.3 have no apparent pattern emerging from the reported values, and the previously clear advantage of including, over excluding, the image features $\boldsymbol{i}_{t,a}^{(\mathrm{u})}$ has seemingly disappeared. What is striking, however, is the relatively few unique users each bandit instance has been assigned, over the deployment period—observe how their median is still below 100. This would imply that their estimated averages might not have completely converged yet and could even be biased due to the strong individual influence each user has on their respective signal's trajectory, both of which are caused by insufficient population sizes. A sign of this can be seen in Figure 4.3, where the profiles' early volatility could be resultant of such a sparse feedback signal; however, this variability is probably also due to additional factors, such as the aforementioned adaptation stage.

On the topic of browser counts, observe how the traces of `LinUCB` and `NeuralUCB`, without the convolutional vector, have almost two-to-three times as many bandit events than their peers, which allows them to visibly stabilize their rate estimates before the others. By cross-referencing this with the associated result table, these algorithmic instances appear to have been allocated around thrice as many patrons as their fellow agents, hence disproving the assumption of a uniform distribution over the service buckets needed for the A/B-testing, as detailed in Section 3.2.1.2. This implies that these specific scores are inferred with a higher degree of certainty, making comparisons to their respective counterparts—who exclude the additional features—potentially inconclusive.

Returning to Table 4.3, notice that the remaining two bandit algorithms—`Linear` ε`-Greedy` and `Hybrid LinUCB`—have the expected uniform partitioning of users and preserves the ordering found in the offline evaluation—i.e., including over excluding the image vector—albeit on a smaller scale. This leads one to speculate that the reverse trend, observed for the `LinUCB` and the `NeuralUCB`, is due to the disproportionate sample sizes; which can probably be corrected if given a longer deployment period or a larger subset of the `Hot`-flow's activity; however, both are not feasible because of time and safety restrictions, respectively. Therefore, with respect to all the aforesaid discoveries, though there is evidence that supports the findings from the offline evaluation, the question of whether those results can be translated to the online setting is still open.

5.1.2 Implementation

There are existing frameworks that can simplify the development of MAB evaluation pipelines, such as Google's AutoML [42], Open Bandit Pipeline [43], and Vowpal Wabbit [6]. However, Google's AutoML is a cloud service and therefore would require to upload `YouPic`'s data to their servers, which introduces security and privacy concerns. In comparison, the Open Bandit Pipeline (OBP) is an open-source project and free for industrial use; nevertheless, it does not support the recommender system setting unique for `YouPic`, making this implementation not fit the bill either and Vowpal Wabbit falls short for similar reasons.

To summarize, the integration of an existing framework would necessitate a lot of familiarization with their code and possibly many workarounds for it to fit the mold of the `YouPic`'s platform. Therefore it was decided that a custom offline simulator would be created, such that control over all aspects of the offline evaluation could be guaranteed and that it adheres to the online problem setting.

5.1.3 Applicability of Bandits

This work has studied stochastic multi-armed bandits in a recommender system setting and the effects of including convolutional features in their contexts; however, what is not analyzed, but is still of importance, is whether these algorithms are applicable for content suggestion. Historic alternatives, the likes of which are presented in Section 2.1, have primarily been of the offline learning class, such as matrix factorization or regression problems, while multi-armed bandits, who are an example of online agents, have only seen recent interest [4]. They promise to balance the exploration-exploitation trade-off while in deployment, but are their adaptive behavior worth the extra complexity and computations?

A clear example of this sought property is when a bandit algorithm adapts its data sampling strategy to investigate under-explored regions of the context space, which is enabled by its malleable policy. In other words, online learning can actively search for new trends in the input vectors, while not forgetting about exploiting known items. However, this assumes that the feature space has under-sampled areas, which raises the question that, given enough data such that the entire context manifold is approximated, would not an offline algorithm perform on par with fewer calculations, as there would be no need for exploration.

Perhaps online learning, and by inclusion multi-armed bandits, can be of benefit to recommender systems if no historic interactions are available; whereas, if enough observations are available, offline learning is probably a more relevant option. Although, where the transition threshold between the two options lies is unknown, but probably problem dependent.

5.1.4 Limitations & Future Work

Given the available resources and time constraints, some limitations have to be acknowledged for this research. However, such restrictions are always chances for

improvement and, in this section, both types are jointly listed.

The collected data, which has been used to design the offline evaluation (January 2022) and neural module training (April 2022), was reconstructed from logged data that was originally not designed for this purpose. For example, only user interactions have been recorded while no information is available if the user stayed after a certain recommendation. Simultaneously, due to constraints in research time and computation resources, the limited samples for training and evaluation might contain unusual seasons or user behaviors which bias the trained weights.

The influence of seasons and other trends could be investigated further as the online evaluation was conducted from December 2022 to January 2023. Additionally, a long-term study might unveil the effect of MAB recommendations on the patrons and if long-term retention can be improved.

Working with human behavior is interesting as there is seldom a clear answer and the type of user might have unknown effects on the recommendations. For example, while some patrons give only purposeful picture likes, others are quite generous in their feedback and consume a large number of images. The only reward signal used is whether a patron like or not, which leaves room for the meaning of other measures such as picture sharing, dwell time and depth, or interactions such as picture ratings and comments.

To provide image recommendations, the MABs are prompted to return a batch of 14 items, which is enabled by the top-k selection policy. Observe that this does not take the order of the suggested items into account, but returns them as a ranking based on the algorithm's action-value predictions, which could be sub-optimal. The class of combinatorial bandits addresses this challenge and tries to find the best combination of items that maximize their return. This becomes especially interesting and challenging, considering the different web layouts the `YouPic`-platform is currently providing.

5.1.5 Ethics

It is important to consider ethical implications when designing any system, especially when dealing with automated user interactions and protocols. Even if a moral dilemma has not been stipulated in law, violating it can create significant harm to the consumer and, in addition, the offender's reputation. As this thesis is a part of an existing and operating platform, the focus is put on ethical aspects concerning machine learning, some of which are presented below. Similarly, problems relating to personal data, picture content, manipulation, and ownership are not considered.

- Creating a model of users' preferences can unknowingly unveil personal, social, political, or sexual behaviors.
- Emphasizing adult content to underaged individuals.
- Creation of an information bubble, thereby isolating the users and promoting extremism.

5.2 Conclusion

This research surveys the effects of multi-armed bandits in a recommender system setting and analyzes the effects that incorporating unstructured data—images in the wild—may imply. The investigations are evaluated through both offline and online evaluation, where the former is enabled by historic logs and the latter by random experiments on live users, both of which are granted by the industrial partner `YouPic AB`.

From offline evaluation, it appears as contextual bandit algorithms that utilize a joint or hybrid action-value model, for all arms, allow for exploitation of the image features, which are generated by a pre-trained convolutional neural network; while those employing a disjoint function perform about the same with the additional information. This can be interpreted as, given that the action-value estimator allows for informational crossover between the arms in the function, it can draw upon the additional features to achieve a higher simulated click-through rate, thus allowing cross-arm generalization.

When deploying all the contextual bandit algorithms online, except the `Disjoint LinUCB` due to quality of service concerns from `YouPic AB`, the previously clear advantage of employing the image features has seemingly been dampened. Nevertheless, some indications support the conclusions drawn from the offline simulations, but these results should not be taken at face value, as the live trials are plagued by insufficient and disproportionate user samples.

Though the experimental results should be taken with a grain of salt, due to uncertainty and bias stemming from the evaluations, this can be a hint of the importance of domain-specific features in contextual bandits integrated into recommender systems. In other words, the results are promising, but further research is required to prove the conclusions with certainty.

Bibliography

[1] D. Agarwal and B. Chen, *Statistical Methods for Recommender Systems*, ser. Statistical Methods for Recommender Systems. Cambridge University Press, 2016, ISBN: 9781107036079. [Online]. Available: https://books.google.se/books?id=k2VoCwAAQBAJ.

[2] C. Aggarwal, *Recommender Systems: The Textbook*. Springer International Publishing, 2016, ISBN: 9783319296579. [Online]. Available: https://books.google.se/books?id=Wpf3jgEACAAJ.

[3] R. Sharda, D. Delen, and E. Turban, *Analytics Data Science & Artificial Intelligence: Systems for Decision Support*. Pearson Education Limited, 2021, ISBN: 9781292341606.

[4] N. Silva, H. Werneck, T. Silva, A. Pereira, and L. Rocha, "Multi-armed bandits in recommendation systems: A survey of the state-of-the-art and future directions," *Expert Syst. Appl.*, vol. 197, 2022, ISSN: 0957-4174. [Online]. Available: https://doi.org/10.1016/j.eswa.2022.116669.

[5] T. Lattimore and C. Szepesvári, *Bandit Algorithms*. Cambridge University Press, 2020. DOI: 10.1017/9781108571401.

[6] L. Li, W. Chu, J. Langford, and R. Schapire, "A contextual-bandit approach to personalized news article recommendation," *Computing Research Repository - CORR*, Feb. 2010. DOI: 10.1145/1772690.1772758.

[7] X. Geng, H. Zhang, J. Bian, and T. Chua, "Learning image and user features for recommendation in social networks," *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 4274–4282, 2015.

[8] J. McAuley, C. Targett, Q. Shi, and A. van den Hengel, "Image-based recommendations on styles and substitutes," *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2015.

[9] R. He and J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," *Proceedings of the 25th International Conference on World Wide Web*, 2016.

[10] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, pp. 84–90, 2012.

[11] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," *Proceedings of the 22nd ACM international conference on Multimedia*, 2014.

[12] A. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: An astounding baseline for recognition," *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 512–519, 2014.

[13] A. Slivkins, *Introduction to multi-armed bandits*, 2019. DOI: 10.48550/ARXIV.1904.07272. [Online]. Available: https://arxiv.org/abs/1904.07272.

[14] W. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, pp. 285–294, 1933.

[15] ——, "On the theory of apportionment," *American Journal of Mathematics*, vol. 57, p. 450, 1935.

[16] S. Scott, "A modern bayesian look at the multi-armed bandit," *Applied Stochastic Models in Business and Industry*, vol. 26, pp. 639–658, 2010.

[17] O. Chapelle and L. Li, "An empirical evaluation of thompson sampling," in *NIPS*, 2011.

[18] D. Zhou, L. Li, and Q. Gu, "Neural contextual bandits with UCB-based exploration," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 119, PMLR, 2020, pp. 11 492–11 502. [Online]. Available: %5Curl%7Bhttps://proceedings.mlr.press/v119/zhou20a.html%7D.

[19] V. Perchet, P. Rigollet, S. Chassang, and E. Snowberg, "Batched bandit problems," *The Annals of Statistics*, vol. 44, no. 2, pp. 660–681, 2016, ISSN: 00905364. [Online]. Available: http://www.jstor.org/stable/43818624 (visited on 08/09/2022).

[20] K. Jun, K. Jamieson, R. Nowak, and X. Zhu, "Top arm identification in multi-armed bandits with batch arm pulls," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, A. Gretton and C. C. Robert, Eds., ser. Proceedings of Machine Learning Research, vol. 51, Cadiz, Spain: PMLR, May 2016, pp. 139–148. [Online]. Available: https://proceedings.mlr.press/v51/jun16.html.

[21] Z. Gao, Y. Han, Z. Ren, and Z. Zhou, *Batched multi-armed bandits problem*, 2019. DOI: 10.48550/ARXIV.1904.01763. [Online]. Available: https://arxiv.org/abs/1904.01763.

[22] Y. Mao, M. Chen, A. Wagle, J. Pan, M. Natkovich, and D. Matheson, *A batched multi-armed bandit approach to news headline testing*, 2019. DOI: 10.48550/ARXIV.1908.06256. [Online]. Available: https://arxiv.org/abs/1908.06256.

[23] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[24] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data mining, Inference, and Prediction*, ser. Springer Series in Statistics. Springer, 2009, ISBN: 9780387848570. [Online]. Available: https://doi.org/10.1007/978-0-387-84858-7.

[25] D. Lowe, "Object recognition from local scale-invariant features," *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, 1150–1157 vol.2, 1999.

[26] S. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, pp. 1345–1359, 2010.

[27] F. Zhuang *et al.*, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, pp. 43–76, 2021.

[28] R. Sutton and A. Barto, *Reinforcement learning, An Introduction*, ser. Adaptive computation and machine learning. The MIT Press, 2018, ISBN: 9780262039246.

[29] Y. Wan, M. H. Zaheer, A. White, M. White, and R. Sutton, "Planning with expectation models," in *IJCAI*, 2019.

[30] M. Farajtabar, Y. Chow, and M. Ghavamzadeh, "More robust doubly robust off-policy evaluation," in *ICML*, 2018.

[31] C. Voloshin, H. Le, and Y. Yue, "Empirical analysis of off-policy policy evaluation for reinforcement learning," 2019.

[32] C. Voloshin, H. Le, Y. Yue, and N. Jiang, "Empirical study of off-policy policy evaluation for reinforcement learning," *ArXiv*, vol. abs/1911.06854, 2019.

[33] M. Dudík, J. Langford, and L. Li, "Doubly robust policy evaluation and learning," in *ICML*, 2011.

[34] A. Gilotte, C. Calauzènes, T. Nedelec, A. Abraham, and S. Dollé, "Offline a/b testing for recommender systems," *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 2018.

[35] Y. Wang, A. Agarwal, and M. Dudík, "Optimal and adaptive off-policy evaluation in contextual bandits," *ArXiv*, vol. abs/1612.01205, 2017.

[36] J. Oh, S. Singh, and H. Lee, "Value prediction network," in *NIPS*, 2017.

[37] S. Chiappa, S. Racanière, D. Wierstra, and S. Mohamed, "Recurrent environment simulators," *ArXiv*, vol. abs/1704.02254, 2017.

[38] `YouPic AB`, *Bring YouPic with you wherever you go!* `https://youpic.com/mobile`, [Online; accessed 2022-05-05], 2021.

[39] ——, *Google Play Store: YouPic App*, `https://play.google.com/store/apps/details?id=com.youpic.youpicandroid&hl=en&gl=US`, [Online; accessed 2022-05-05], 2021.

[40] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *ArXiv*, vol. abs/1905.11946, 2019.

[41] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015. DOI: `10.1007/s11263-015-0816-y`.

[42] P. Dutta, J. Cheuk, J. Kim, and M. Mascaro, *Automl for contextual bandits*, 2019. DOI: `10.48550/ARXIV.1909.03212`. [Online]. Available: `https://arxiv.org/abs/1909.03212`.

[43] Y. Saito, S. Aihara, M. Matsutani, and Y. Narita, *Open bandit dataset and pipeline: Towards realistic and reproducible off-policy evaluation*, 2020. DOI: `10.48550/ARXIV.2008.07146`. [Online]. Available: `https://arxiv.org/abs/2008.07146`.

A

Modeling

This chapter presents the performance of the best-performing super model, as given by the validation loss, whose definition can be found in Section 3.3. The hyper-parameters, for the selected model, have the user feature space set to 32, the image's structured and unstructured vectors to 16 and 8, respectively, while the click model's hidden state set to 32 and no dropout was employed. In the Figure A.3, this network's training and validation loss profiles are shown, together with the four closest contenders as measured by the lowest validation; in other words, it is a top-five visualization. Regarding the notation used in the legend, the letters represent the initials of user metadata (`u`), image metadata (`i`), image vector (`v`), main output (`o`) or dropout probability (`p`), where a dash (`-`) is the delimiter; for example, the top performing model is represented as `u32-i16-v08-o32-p00`.

To verify that the model mimics the user behavior, the confusion matrices for the training, validation, and testing datasets are constructed for the main output module (i.e., click model), which can be seen in Figure A.1. Note that the underlying data is unfiltered, e.g. there are users clicking everything, but this predictor appears to be a fairly good candidate for estimating the user-image interactions.

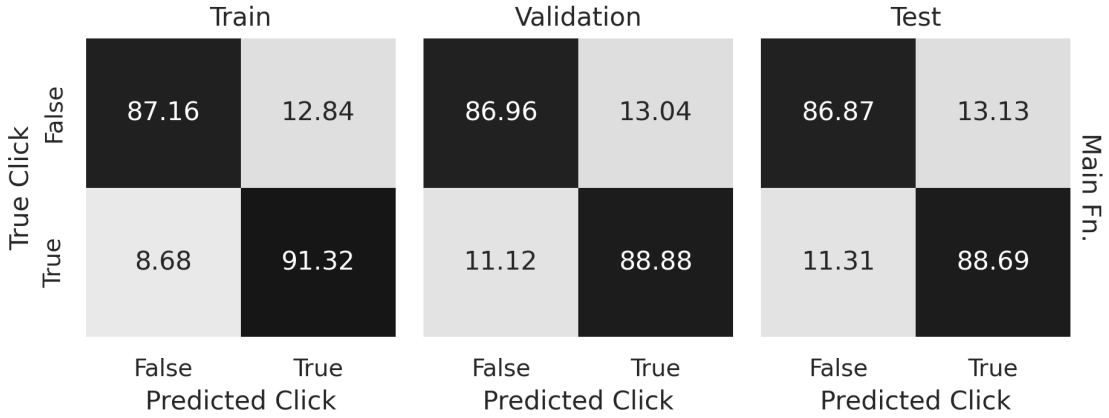


Figure A.1: This illustration displays the confusion matrices of the main output (i.e., click model) for the training, validation, and testing datasets. As the data has not been cleaned from unusual online behavior, this model appears to be fairly good at predicting a user's click.

To understand the output of the super network a bit better, the output densities, of both the main and the auxiliary estimators, are visualized for the same three sets and can be found in Figure A.2. The main function, or the click model, appears to have been able to separate the two categories of the Boolean target variable—the `click` feedback. In comparison, the linear auxiliary function appears to have the `True` outcome under control, but the `False` outcome appears to span the whole range; this behavior is probably due to the model being linear and it is weighted to fight class imbalance. Given these results, it appears as the two outputs are working mostly as intended.

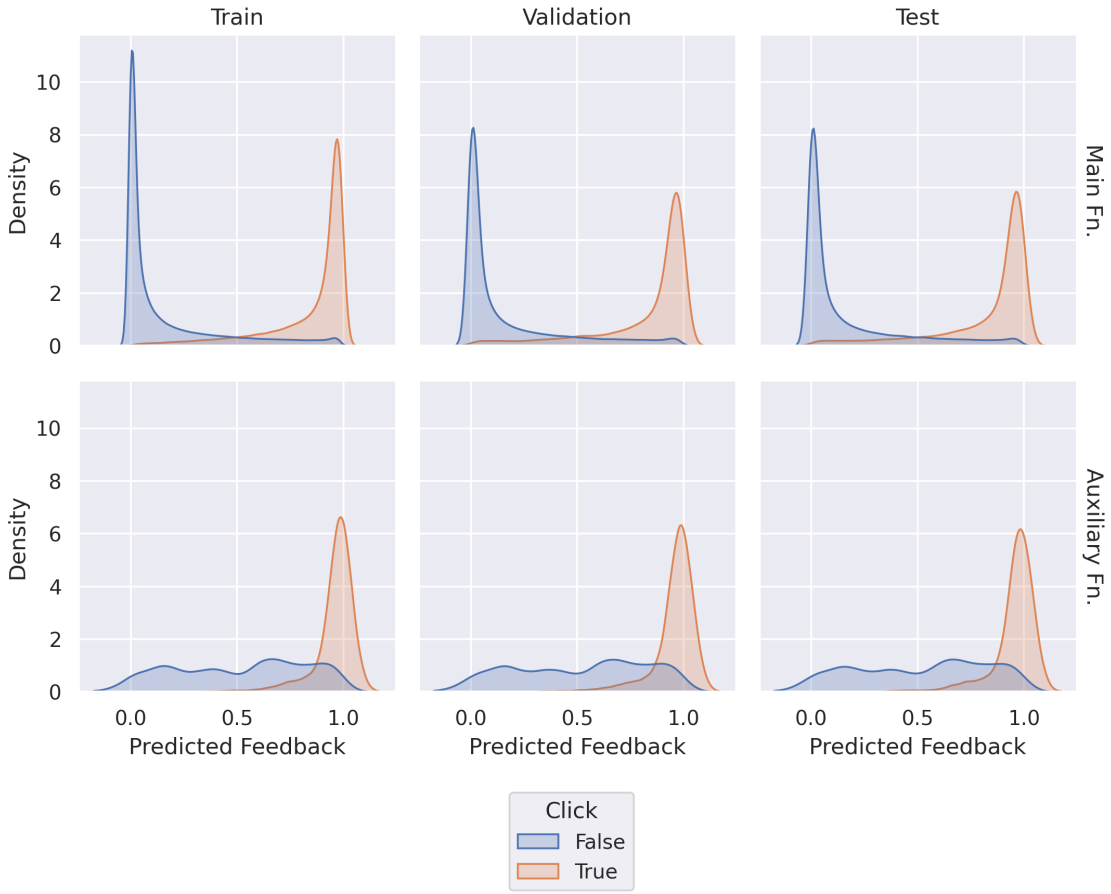


Figure A.2: This visualization depicts the output densities of the main (click model) and the auxiliary output modules for the training, validation, and testing sets. As the main function is a feed-forward, binary classifier, its outputs appear good enough as a click predictor. On the other hand, the auxiliary model appears to be able to linearly predict the positive outcome, but struggles to have a clear-cut prediction for the negative; which is to be expected, given that it is linear and is re-weighted to combat the class imbalance.

Lastly, Figure A.3 displays the training and validation loss profiles for the super models achieving the lost validation value during its training, which was mentioned in this appendix chapter's opening paragraph. Notice that even though the training continues up to the limit of 64 epochs, the instance corresponding to the epoch

with the lowest observed loss is taken; for the model `u32-i16-v08-o32-p00`, this corresponds to its 16^{th} epoch.

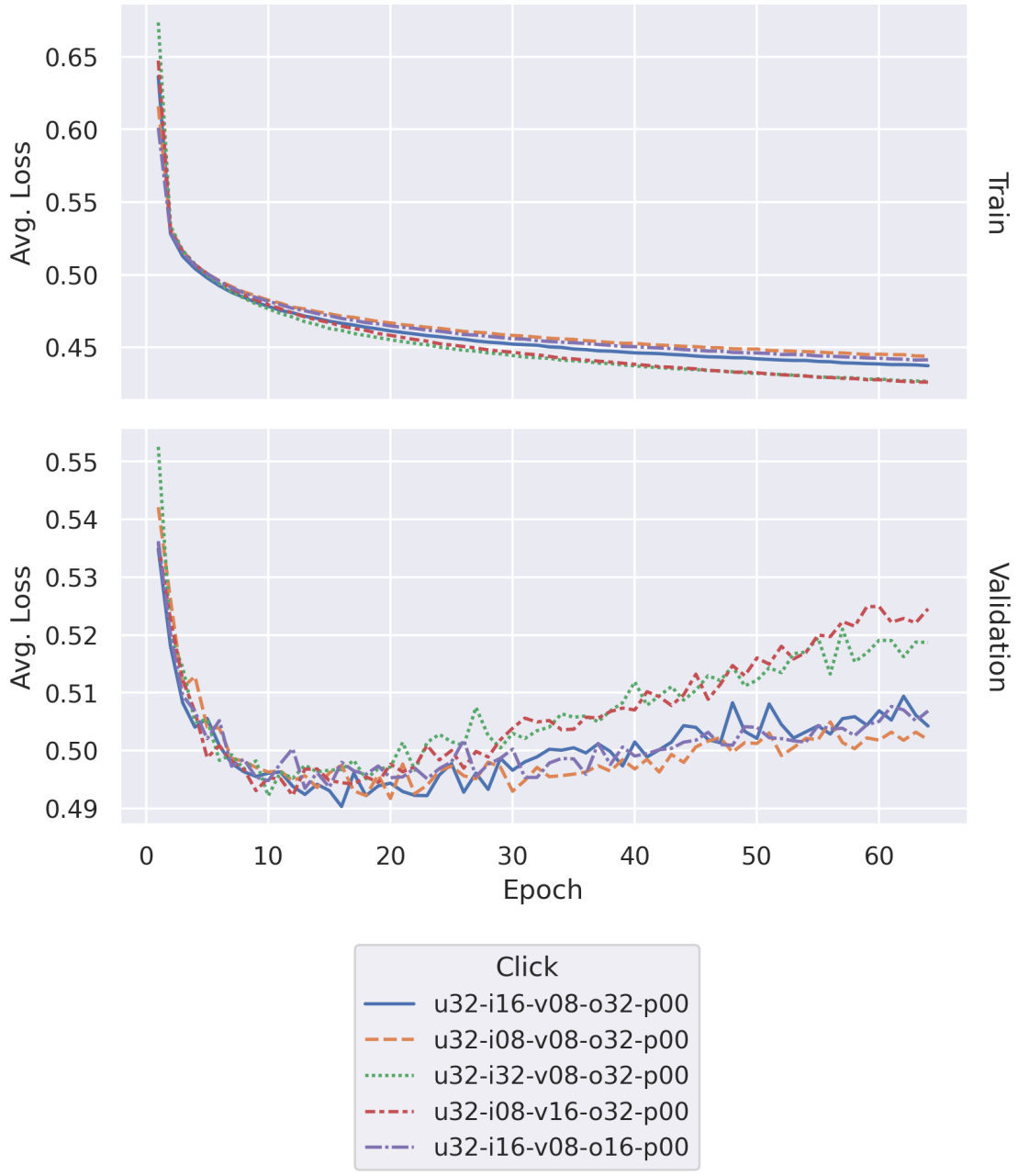


Figure A.3: A diagram showing the loss profiles of the top-five performing super neural networks from the grid-search, when going by the lowest observed validation loss for the whole training session. The legend entries represent the hyper-parameters, used to initialize the model, where `u` stands for the user's metadata latent space, `i` for the image's metadata, `v` for the image vector, `o` for the output module's hidden space and `p` for the dropout probability. Notice that no instance with the dropout set to 50% made the cut.

B

Hyper-Parameter Tuning

This chapter provides the detailed results of the hyper-parameter tuning as described in Section 3.4 and whose results are briefly presented in Section 4.1.1. All reported values are aggregates from the five independent trials using the offline simulator, if not specified otherwise, and are the metrics employed when comparing different initialization configurations when performing the grid-search. Due to time limitations, one such trial could only be run for the first 24 hours of the offline simulation, not the entire dataset covering the month of January 2022 as planned.

Each algorithm presented in Section 3.4 has its dedicated section in this appendix chapter. In these, every bandit has a figure depicting their cumulative feedback rate and their cumulative regret, visualized by their mean profile from the five trials. Furthermore, a table with their final feedback rates and total regrets is also present, which also contains their respective standard errors, estimated using the trials' scores, and can serve as a quantity of each estimate's precision.

B.1 `Tabular` ε`-Greedy`

In this section, the hyper-parameter grid-search results for the `Tabular` ε`-Greedy` algorithm are presented, which are summarized in Figure B.1 and Table B.1 below. The configuration that is selected as their representative is the bandit with $\varepsilon = 0.001$ and $\hat{\mu}_0 = 0.25$, i.e., the probability of a random action and the default initialization value for the average feedback estimate, respectively. Notice how there is a tendency to prefer smaller values of ε, while having a positive value of $\hat{\mu}_0$; this could mean that the algorithm rather employs optimistic initialization to explore, than achieving this through random actions. This should have been expected, as only the former allows for targeting novel arms for exploration, which becomes an edge while in a recommender system setting, where new options are continuously introduced.

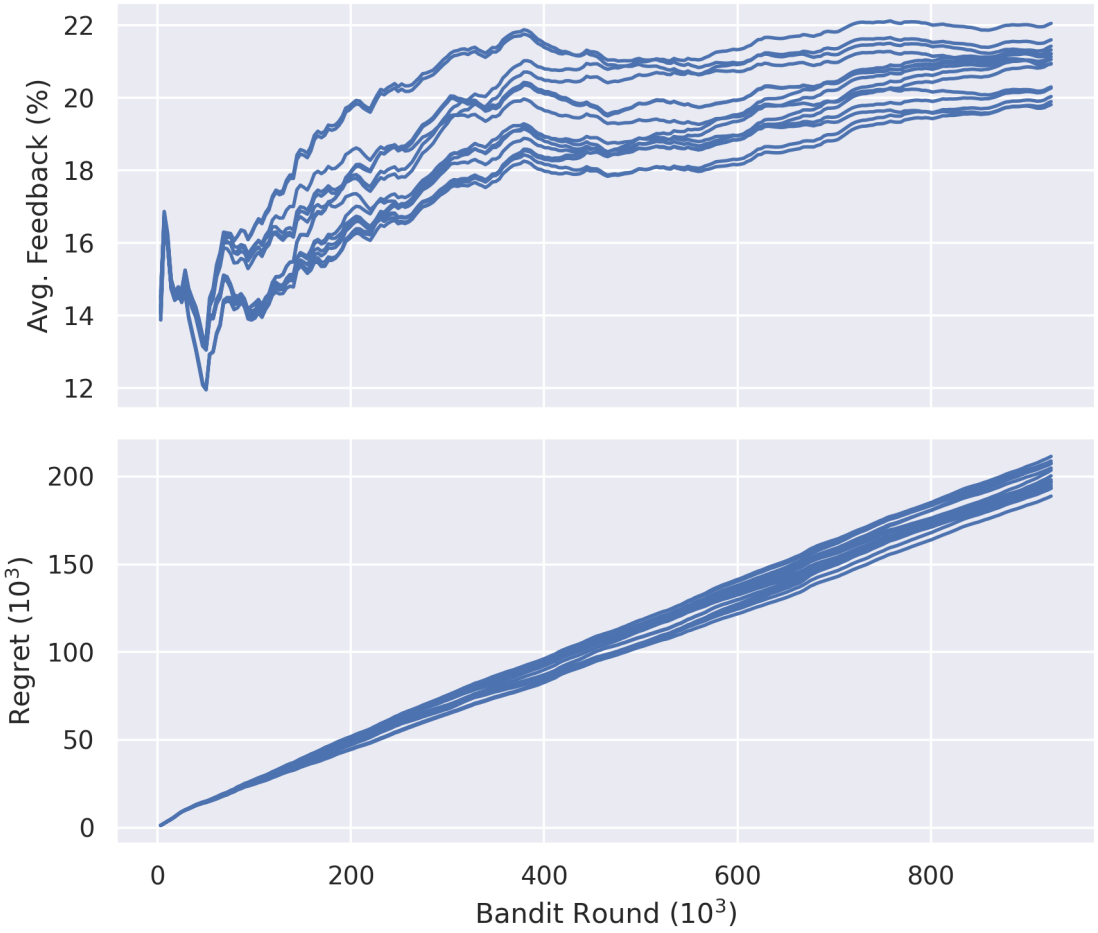


Figure B.1: The `Tabular` ε`-Greedy`'s profiles for the cumulative average feedback and regret, corresponding to the upper and lower panes, respectively; being a context-free algorithm, there are only solid blue lines present and each denotes the average of the five independent trials. Studying the upper graph, notice how all traces appear to start from approximately the same point, whereafter their curves diverge and show how their hyper-parameter initialization determines their predisposition, in terms of performance; although, it would appear as their cumulative regret have a similar rate of growth, but not equal as they seem to fork.

Table B.1: Results of the `Tabular` ε`-Greedy`'s hyper-parameter tuning, which encompasses the average feedback and regret, both of which are presented as their five-trial sample mean $\bar{\mu}$ and associated standard error $\bar{\sigma}$. Notice that there appears to exist a trend that favors smaller exploration probabilities ε, given that all their initial guesses for $\hat{\mu}_0$ are non-zero. This pattern seems to point towards that the algorithm prefers optimistic initialization for exploration, i.e., defaulting to a higher-than-expected initial value of $\hat{\mu}_0$ and then correcting it with sampling, instead of the random exploration through a larger ε. In fact, this is to be expected, as it allows for guided exploration of new arms, which becomes a powerful ability when the action set is changing constantly.

	Parameters		Feedback (%)		Regret (10^3)	
Rank	ε	$\hat{\mu}_0$	$\bar{\mu}$	$\bar{\sigma}$	$\bar{\mu}$	$\bar{\sigma}$
1	0.001	0.25	22.05	0.10	188.85	0.95
2	0.01	0.25	21.60	0.11	193.27	1.26
3	0.001	0.50	21.42	0.08	193.39	0.87
4	0.01	0.50	21.32	0.04	194.36	0.78
5	0.001	0.75	21.21	0.09	195.38	0.90
6	0.01	0.00	21.21	0.13	200.45	1.41
7	0.001	1.00	21.13	0.03	196.70	0.51
8	0.01	0.75	21.06	0.06	197.11	0.56
9	0.01	1.00	20.95	0.04	198.14	0.46
10	0.001	0.00	20.93	0.09	203.65	0.66
11	0.1	0.00	20.30	0.07	207.65	0.51
12	0.1	0.50	20.27	0.06	204.87	0.40
13	0.1	0.75	20.04	0.05	207.36	0.44
14	0.1	1.00	19.90	0.07	208.86	0.78
15	0.1	0.25	19.81	0.05	211.51	0.53

B.2 `Linear` ε-`Greedy`

The representative of the `Linear` ε-`Greedy` algorithms appears to be the instance with an exploration probability $\varepsilon = 0.001$ and regularization parameter $\lambda = 0.1$, when consulting results in Table B.2. Observe that an interesting trend is visualized in Figure B.2, which shows that bandits using the image vector $\boldsymbol{i}_{t,a}^{(\mathrm{u})}$, and have $\varepsilon = 0.1$, displays a noticeable reduction in their estimated sample average, thus potentially highlighting that random exploration is sub-optimal for bandit algorithms in the recommendation setting. This is also the case for the instances not using the image vector, only on a smaller scale.

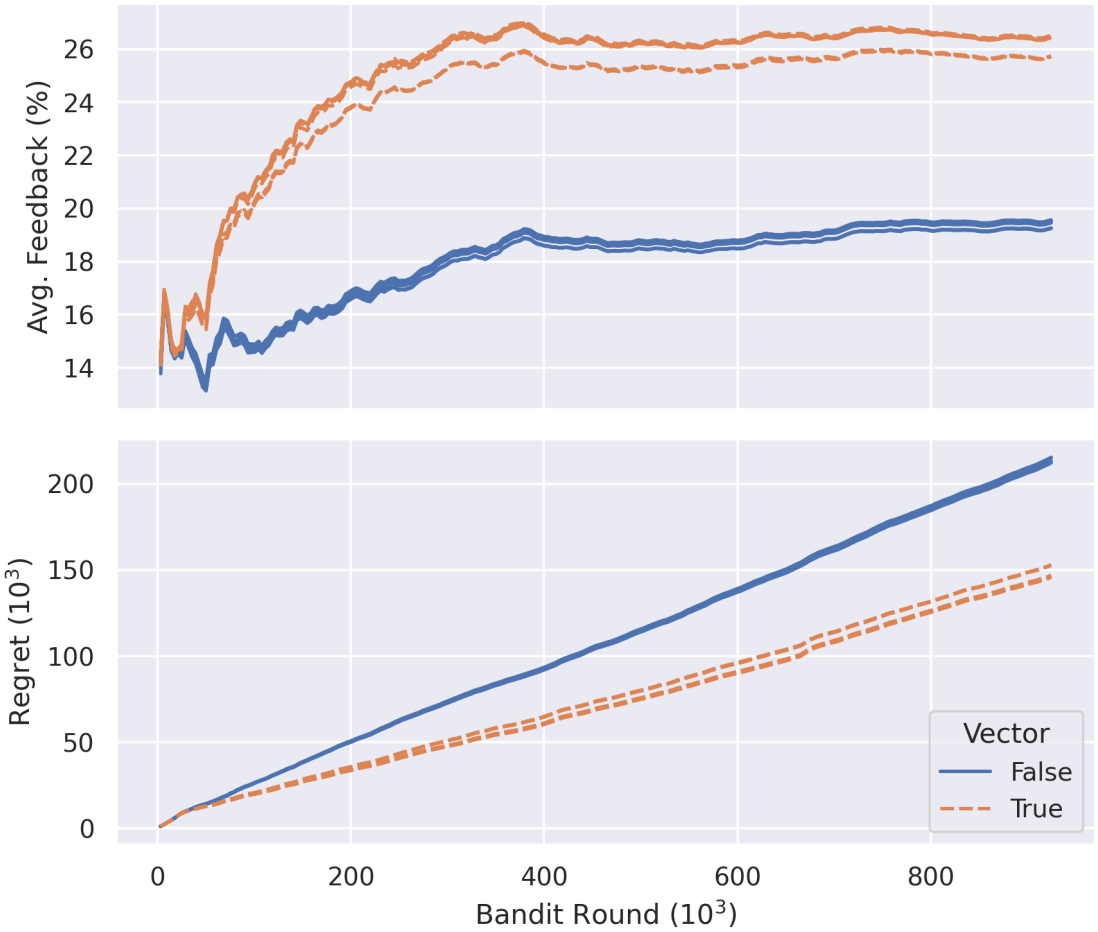


Figure B.2: The cumulative average feedback and regret profiles of the `Linear` ε-`Greedy` algorithm, every line is the pooled average corresponding to a hyper-parameter configuration tested in the grid-search and are estimated using the traces from the five independent trials. Furthermore, by being a contextual algorithm, there are two types of lines present, where the solid blue and dashed orange denote the instances excluding and including the image vector $\boldsymbol{i}_{t,a}^{(\mathrm{u})}$, respectively. Notice how a subset of the curves, who includes the convolutions features, all have systematically worse performance than the rest; cross-referencing with Table B.2, it appears to be an effect of having a larger probability for random exploration $\varepsilon > 0$. This also seems to be true when excluding $\boldsymbol{i}_{t,a}^{(\mathrm{u})}$, though here this trend is less pronounced.

Table B.2: The feedback rates and regrets of the `Linear` ε-`Greedy`'s hyper-parameter search; both are presented by their sample average $\bar{\mu}$ and their associated standard error $\bar{\sigma}$, which are estimated from the optimization's five independent trials. Interestingly, it appears as including the convolutional features $\boldsymbol{i}_{t,a}^{(\mathrm{u})}$ gives a competitive edge over excluding it, as the worst feedback rate of the former is approximately six percentage points higher than the best performing one of the latter, and a similar story is found for the total regret.

	Parameters			Feedback (%)		Regret (10^3)	
Rank	ε	λ	$\boldsymbol{i}_{t,a}^{(\mathrm{u})}$	$\bar{\mu}$	$\bar{\sigma}$	$\bar{\mu}$	$\bar{\sigma}$
1	0.001	0.1	✓	26.52	0.05	145.75	0.37
2	0.001	1.0	✓	26.47	0.06	146.08	0.58
3	0.01	0.1	✓	26.45	0.02	146.43	0.57
4	0.001	10.0	✓	26.45	0.03	146.51	0.54
5	0.01	10.0	✓	26.43	0.02	146.93	0.41
6	0.01	1.0	✓	26.43	0.02	146.70	0.35
7	0.10	1.0	✓	25.74	0.03	152.73	0.31
8	0.10	0.1	✓	25.72	0.04	153.02	0.39
9	0.10	10.0	✓	25.71	0.04	152.94	0.54
10	0.001	10.0	✗	19.57	0.08	212.21	0.73
11	0.001	0.1	✗	19.55	0.07	212.57	0.46
12	0.01	10.0	✗	19.50	0.04	212.70	0.45
13	0.01	1.0	✗	19.49	0.05	212.65	0.28
14	0.001	1.0	✗	19.48	0.03	212.66	0.35
15	0.01	0.1	✗	19.45	0.07	213.31	0.67
16	0.1	0.1	✗	19.27	0.03	215.13	0.23
17	0.1	10.0	✗	19.25	0.05	215.31	0.65
18	0.1	1.0	✗	19.24	0.11	215.38	0.96

B.3 `Thompson Sampling`

From the `Thompson Sampling` algorithm's grid-search, see Figure B.3 and Table B.3, the representative is selected to be that corresponding to the concentrations $\alpha = 1$ and $\beta = 100$, as to that setup have both the highest average feedback rate and the lowest total regret. However, notice that the profile of the second best bandit appears to close in on the selected representative; cross-checking with this section's table, it seems as if initializing $\alpha = 10$ could be better in the long-run, but the offline simulation cut off too early for any definitive proof.

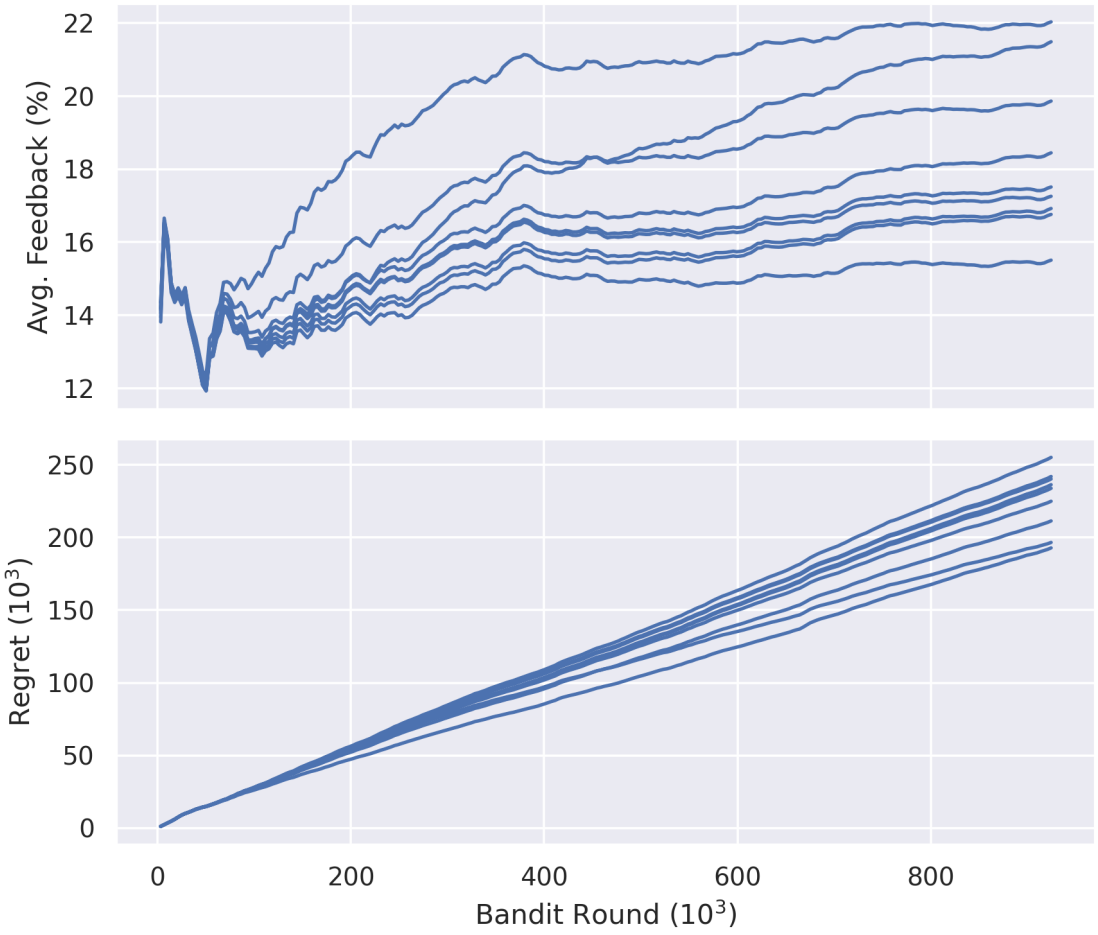


Figure B.3: In this figure, the profiles of the `Thompson Sampling` algorithm's cumulative feedback rate and regret can be seen, which corresponds to the upper and lower panes, respectively. In addition, being a context-free algorithm, there are only solid blue lines present and each denotes the average from five independent trials. Observe that the prior distribution's initial concentrations have a huge impact on their traces' development, over the course of the simulation. In particular, notice how the next-best hyper-parameter configuration, in terms of both the feedback rate and total regret, appears to close in on the trace of the highest scorer, i.e., the selected representative, which corresponds to setting $\alpha = 10$ respective 1, and $\beta = 100$ for both cases.

Table B.3: The estimated feedback rates and regrets resulting from the `Thompson Sampling` algorithm's hyper-parameter search, which are both rendered as the sample average $\bar{\mu}$ and its affiliated standard error $\bar{\sigma}$ from five independent trials. Analyzing the table, it would appear as `Thompson Sampling` favors Beta distributions that initialize their concentrations such that the majority of the respective densities are shifted towards zero, i.e., $\beta > \alpha$. This is to be expected, as the average click-through rate is believed to be approximately 0.1, where the parameterization that seems to emphasize this region the most has $\alpha = 1$ and $\beta = 100$. On a side note, observe that the concentrations that put the density mass towards unity (i.e., $\alpha > \beta$) perform much worse than those doing the opposite, thus highlighting the fact that prior knowledge can be paramount when dealing with Bayesian problems.

Rank	Parameters		Feedback (%)		Regret (10^3)	
	α	β	$\bar{\mu}$	$\bar{\sigma}$	$\bar{\mu}$	$\bar{\sigma}$
1	1	100	22.03	0.11	192.88	1.41
2	10	100	21.49	0.15	196.63	0.92
3	1	10	19.86	0.05	211.44	0.64
4	1	1	18.45	0.04	224.99	0.21
5	10	1	17.51	0.01	233.90	0.11
6	100	1	17.26	0.03	236.41	0.08
7	10	10	16.92	0.03	240.32	0.19
8	100	10	16.76	0.03	242.02	0.09
9	100	100	15.51	0.02	255.19	0.21

B.4 UCB1

In this section, the result from the UCB1 algorithm's grid-search is presented, see Figure B.4 and Table B.4. Observe that there are little to no differences, in terms of performance, between the tested initial values of $\hat{\mu}_0$; this shows that there is no conclusive evidence promoting any specific prior value. This can be explained by the UCB1's explore-first policy, which stems from assigning unexplored arms an infinite opportunistic upper bound. Anyhow, the representative selected is that with $\hat{\mu}_0 = 1$; though, any of the candidates' rankings are likely due to noise from the simulation.

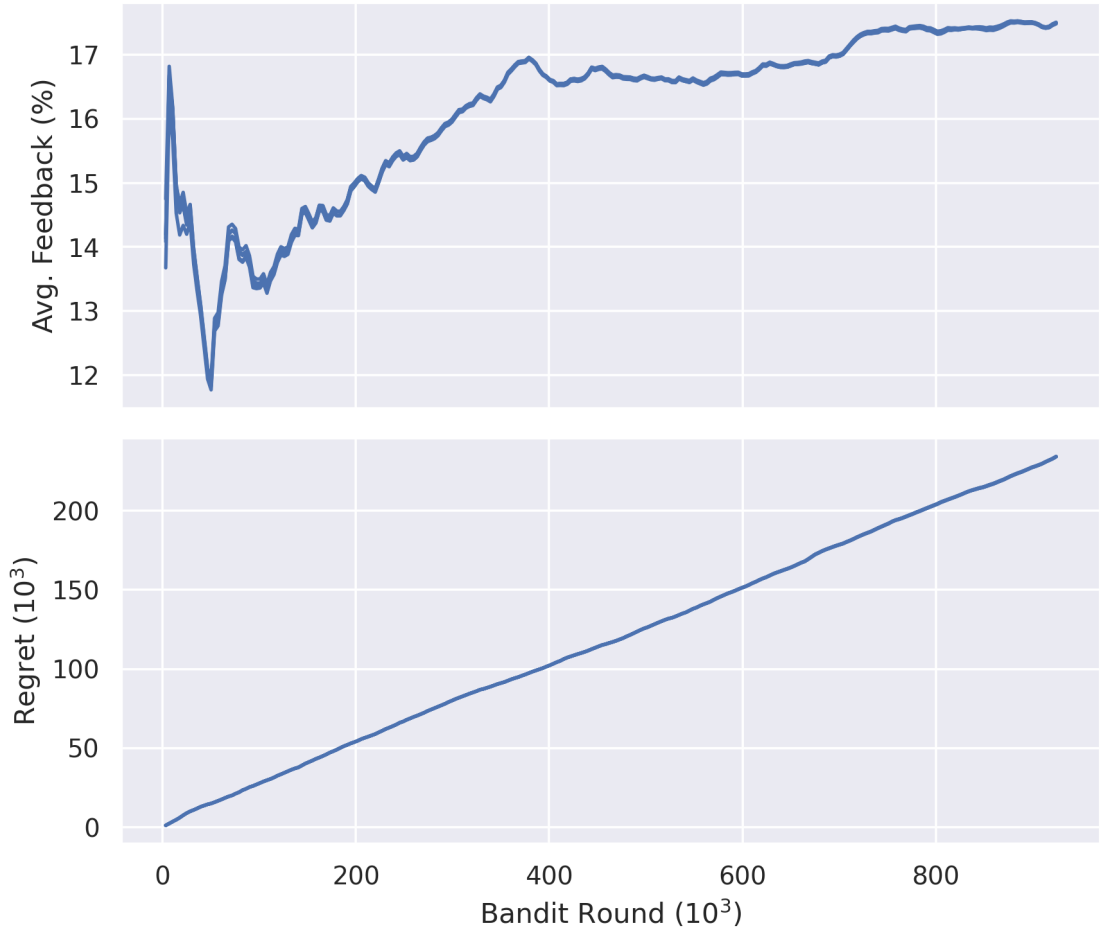


Figure B.4: This visualization depicts the cumulative feedback rate and regret profiles, from the hyper-parameter search of the UCB1 algorithm, which corresponds to the upper and lower panes, respectively. Furthermore, by being a context-free algorithm, there are only solid blue lines present in this visualization, where the curves are the pooled averages from five independent trials created by the offline simulation. Notice how there is virtually no difference between all the curves in either the top or the bottom visualizations, which illustrates how the initial action-value $\hat{\mu}_0$ seemingly has little or no effect on the performance of bandits. This can be explained by the UCB1's explore-first property, which emphasizes exploring novel arms, and, with an ever-growing pool of images, the algorithm will not have many chances to exploit rewarding options.

Table B.4: The feedback rates and total regrets generated from the hyper-parameter grid-search of the `UCB1` algorithm, presented as the average $\bar{\mu}$ and their associated standard error $\bar{\sigma}$ estimated from five independent trials. Interestingly, there appears as none of the tested values of the initial click rate $\hat{\mu}_0$ provides any advantage over the others, as all achieve practically the same score, and can be attributed to the definition of `UCB1`; remember that unexplored arms are given an expected return of infinity, which forces the algorithm to always explore novel options and, for the case where the action pool is always changing and growing, as is the case of the recommender system setting, this can be detrimental to its performance.

Rank	Parameters $\hat{\mu}_0$	Feedback (%) $\bar{\mu}$	$\bar{\sigma}$	Regret (10^3) $\bar{\mu}$	$\bar{\sigma}$
1	1.00	17.51	0.03	234.18	0.42
2	0.25	17.50	0.06	234.18	0.53
3	0.75	17.49	0.02	234.45	0.14
4	0.00	17.49	0.05	234.30	0.20
5	0.50	17.48	0.02	234.22	0.23

B.5 `LinUCB`

Studying the results from the hyper-parameter search, see Figure B.5 and Table B.5, the selected representative for the `LinUCB` algorithm has an exploration factor $\delta = 0.1$ and a regularization $\lambda = 1.0$. However, notice that there are configurations that are consistently worse than their peers, in terms of including or excluding the image vector $\boldsymbol{i}_{t,a}^{(\mathrm{u})}$; this happens occurs when $\delta = 10.0$ and indicates that emphasizing the exploration too much early on can hamper the bandit's learning, at least for the initial phase of deployment.

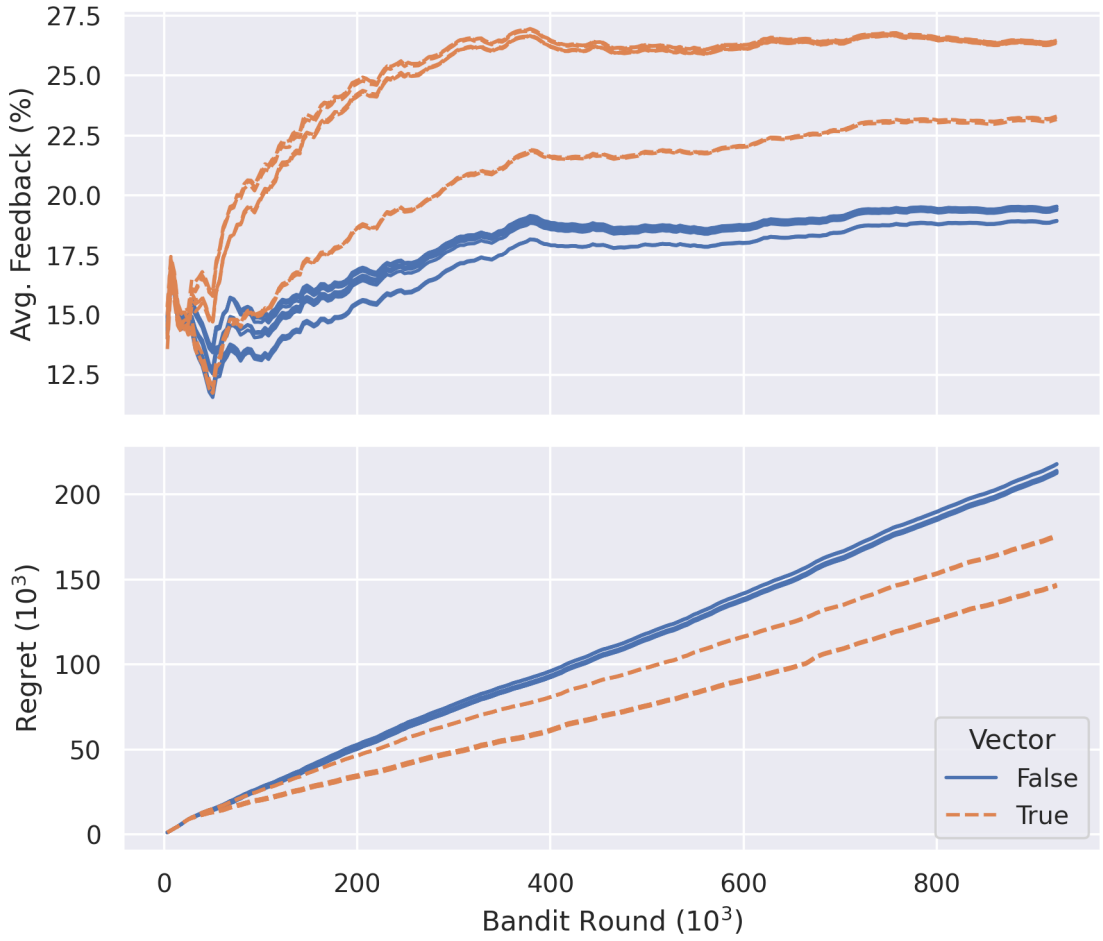


Figure B.5: The cumulative feedback rate and regret curves of the `LinUCB` algorithm, which were estimated from five independent trials using the offline simulator when running the hyper-parameter tuning. In addition, by being a contextual algorithm, there are two types of lines present, where the solid blue and dashed orange denote the instances excluding and including the image vector $\boldsymbol{i}_{t,a}^{(\mathrm{u})}$, respectively. Observe that including the convolutional features provides a competitive edge in this artificial recommender system setting, save for three traces that lags behind with approximately three percentage points, which include $\boldsymbol{i}_{t,a}^{(\mathrm{u})}$. Though the trials ended before these subsets of curves could converge, this finding indicates that emphasizing exploration too much early on can significantly decrease the algorithm's performance.

Table B.5: The `LinUCB` algorithm's final feedback rate and regret estimates for all configurations tested in the hyper-parameter search, presented by their five-trial sample average $\bar{\mu}$ and corresponding standard $\bar{\sigma}$. It appears as the average performance $\bar{\mu}$ is sorted by whether or not the convolutional features $\boldsymbol{i}_{t,a}^{(\mathrm{u})}$ are included, or not, and by the exploration factor δ, in ascending respective descending order. This shows that emphasizing the exploration too much can be detrimental in a recommender system setting.

	Parameters			Feedback (%)		Regret (10^3)	
Rank	δ	λ	$\boldsymbol{i}_{t,a}^{(\mathrm{u})}$	$\bar{\mu}$	$\bar{\sigma}$	$\bar{\mu}$	$\bar{\sigma}$
1	0.1	1.0	✓	26.49	0.04	146.11	0.25
2	0.1	0.1	✓	26.47	0.06	146.31	0.21
3	0.1	10.0	✓	26.46	0.03	146.63	0.12
4	1.0	0.1	✓	26.39	0.04	146.91	0.39
5	1.0	10.0	✓	26.38	0.04	146.95	0.43
6	1.0	1.0	✓	26.36	0.06	147.12	0.43
7	10.0	0.1	✓	23.30	0.13	175.14	1.18
8	10.0	1.0	✓	23.26	0.09	175.65	0.71
9	10.0	10.0	✓	23.17	0.10	176.27	0.94
10	0.1	0.1	✗	19.54	0.06	212.51	0.71
11	0.1	1.0	✗	19.52	0.05	212.73	0.76
12	0.1	10.0	✗	19.44	0.27	213.47	2.72
13	1.0	10.0	✗	19.44	0.03	213.19	0.24
14	1.0	1.0	✗	19.43	0.03	213.60	0.40
15	1.0	0.1	✗	19.37	0.09	214.08	0.76
16	10.0	10.0	✗	18.94	0.07	217.87	0.70
17	10.0	1.0	✗	18.93	0.06	217.73	0.45
18	10.0	0.1	✗	18.91	0.04	218.05	0.34

B.6 `Disjoint LinUCB`

Based on the results below, Figure B.6 and Table B.6, the `Disjoint LinUCB` algorithm's representative is selected to be that with an exploration factor $\delta = 0.1$ and a regularization parameter $\lambda = 1.0$. The reason is that this corresponds to the highest scoring configuration while using the image vector $\boldsymbol{i}_{t,a}^{(\mathrm{u})}$, which is the selection criterion for this thesis. On a side note, observe how the presence of the convolutional features does not seemingly provide any visible advantages over disregarding them; this can be attributed to its disjoint action-value function, which prevents cross-arm pattern generalization.

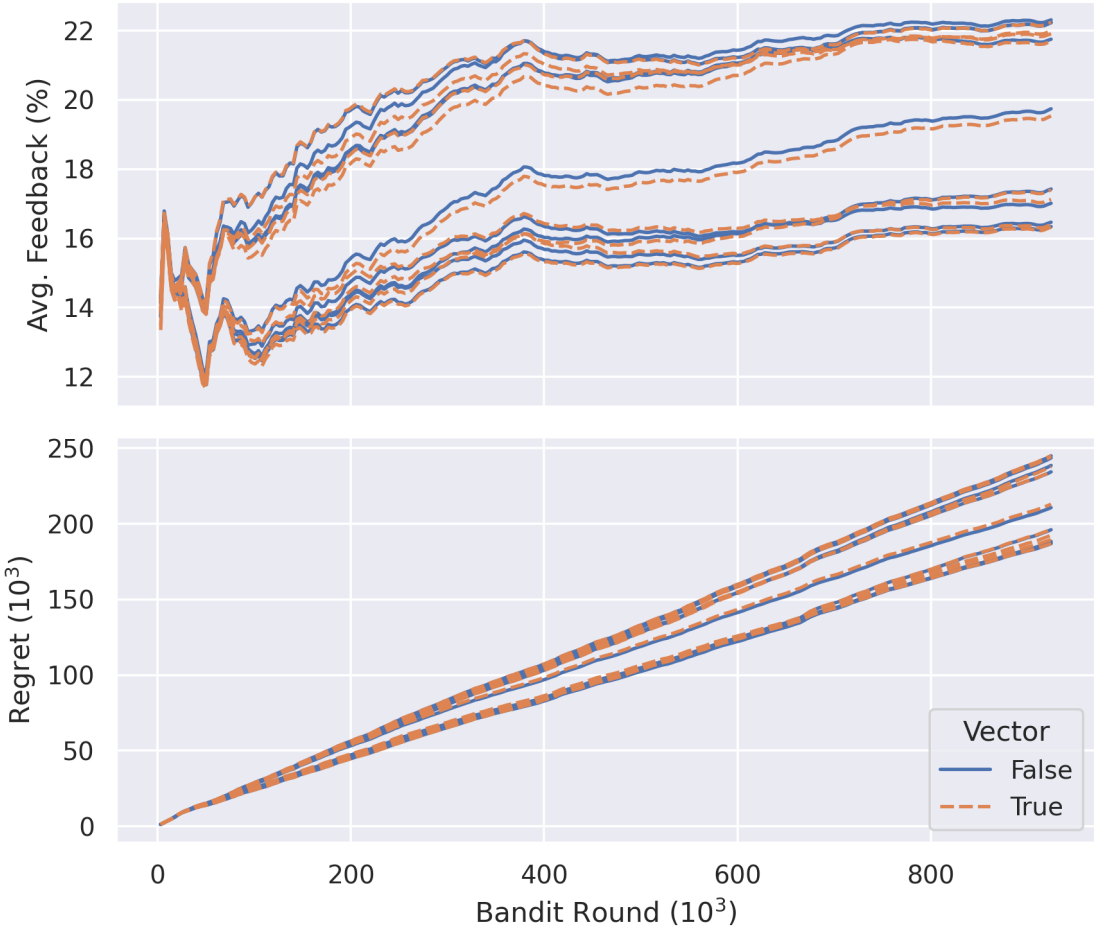


Figure B.6: The displayed `Disjoint LinUCB` algorithm's cumulative average feedback and regret profiles are estimated from five independent trials using the offline simulator, where every line denotes a hyper-parameter configuration tested in the grid-search. Moreover, as it is a contextual algorithm, there are two types of lines present, where the solid blue and dashed orange denote the instances excluding and including the image vector $\boldsymbol{i}_{t,a}^{(\mathrm{u})}$, respectively. Notice how there is no trend differentiating the instances including and excluding the convolutional features, respectively; the reason should be its disjoint action-value function, which prevents it from generalizing across arms for higher sample efficiency and coordinated pattern recognition.

Table B.6: Result of the hyper-parameter grid-search's produced feedback rates and total regrets for the `Disjoint LinUCB` algorithm, each staged through their average $\bar{\mu}$ and standard error $\bar{\sigma}$ from five independent trials. Fascinatingly, the results appear to be, in principle, sorted by the exploration factor δ in ascending order. Furthermore, it appears as the additional information provided by the image vector $\boldsymbol{i}_{t,a}^{(\mathrm{u})}$ does not provide any significant advantage, since pairs employing the same hyper-parameters, except for the mentioned convolutional features, achieve effectively the same simulated feedback rates and regrets.

	Parameters			Feedback (%)		Regret (10^3)	
Rank	δ	λ	$\boldsymbol{i}_{t,a}^{(\mathrm{u})}$	$\bar{\mu}$	$\bar{\sigma}$	$\bar{\mu}$	$\bar{\sigma}$
1	0.1	1.0	✗	22.31	0.15	188.45	1.63
2	0.1	0.1	✓	22.24	0.09	187.06	0.80
3	0.1	0.1	✗	22.24	0.11	186.95	1.03
4	1.0	10.0	✗	22.23	0.09	187.65	1.07
5	1.0	10.0	✓	22.00	0.03	189.72	0.66
6	0.1	1.0	✓	21.91	0.06	192.59	1.00
7	0.1	10.0	✗	21.75	0.10	196.15	1.24
8	0.1	10.0	✓	21.69	0.23	196.45	2.27
9	1.0	1.0	✗	19.74	0.05	210.75	0.42
10	1.0	1.0	✓	19.53	0.04	213.01	0.29
11	1.0	0.1	✗	17.43	0.02	234.42	0.08
12	1.0	0.1	✓	17.40	0.02	234.52	0.12
13	10.0	10.0	✓	17.12	0.06	237.95	0.16
14	10.0	10.0	✗	17.01	0.03	238.67	0.15
15	10.0	1.0	✗	16.47	0.03	243.64	0.11
16	10.0	1.0	✓	16.39	0.03	244.40	0.06
17	10.0	0.1	✗	16.34	0.03	245.01	0.08
18	10.0	0.1	✓	16.32	0.03	245.22	0.11

B.7 `Hybrid LinUCB`

The `HybridLinUCB` algorithm's representative is selected to have the parameterization $\delta = 0.1$ and $\lambda = 10.0$, which denotes the exploration multiplier and regularization factor, respectively. Studying Figure B.7 and Table B.7 below, observe how this bandit has one hyper-parameter configuration that levels out with a feedback rate above 30%, while also having its total regret stay under the 100,000-mark. Furthermore, observe how intermingled the traces of the instances using respective ignoring the image vector $\boldsymbol{i}_{t,a}^{(\mathrm{u})}$, when compared to the other tested algorithms, which could be an indication of dependence on the hyper-parameter initialization.

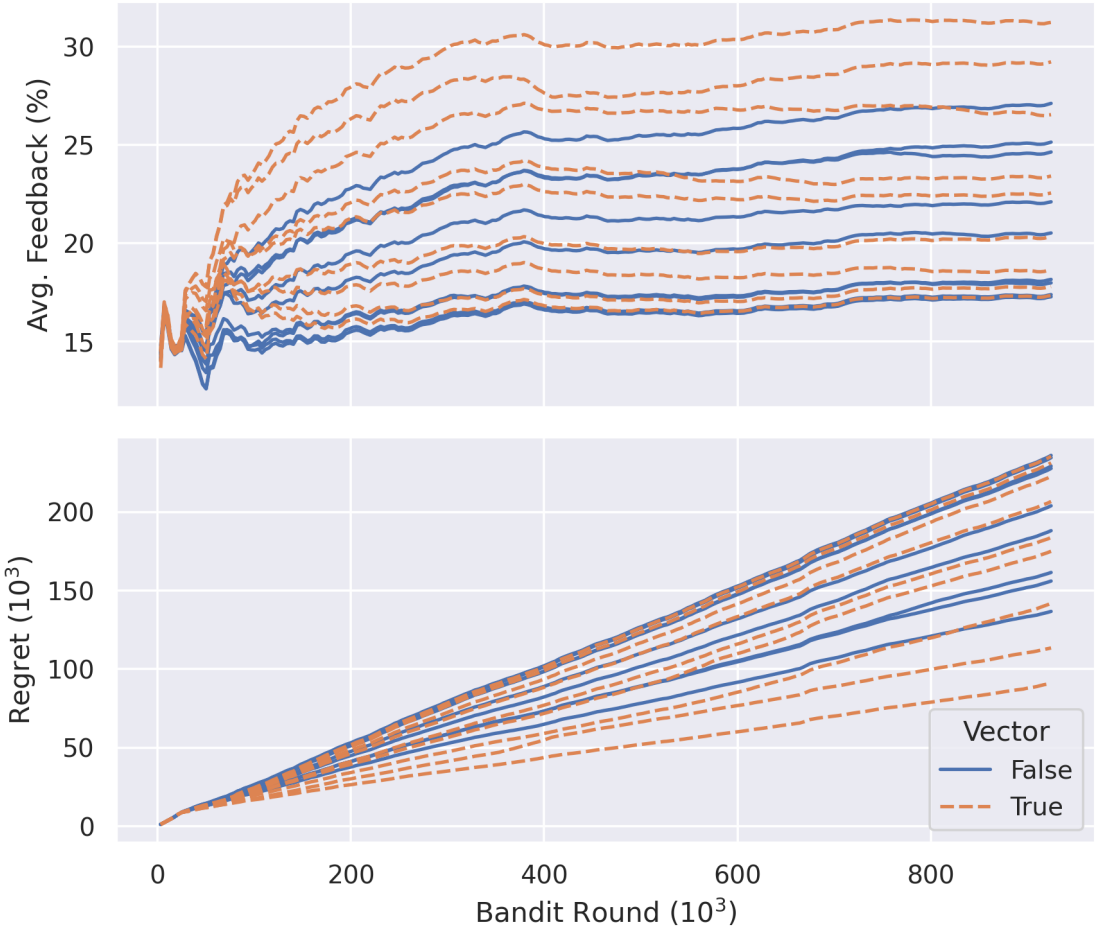


Figure B.7: This figure shows the cumulative feedback rate and regret profiles from the hyper-parameter tuning of the `Hybrid LinUCB` algorithm; every line represents the average of five independent trials, employing the offline simulation, where the solid blue and dashed orange traces denotes the instances excluding respective including the convolutional features $\boldsymbol{i}_{t,a}^{(\mathrm{u})}$, due to it being a contextual bandit. Notice how intermingled the two types of curves are, hence potentially indicating sensitivity to the initialization of its parameterization; furthermore, observe how one of the profiles breaches the threshold of a 30% feedback rate, all while having its total regret stay below the 100,000 mark.

Table B.7: The estimated feedback rates and total regrets of the `Hybrid LinUCB` algorithms' tested hyper-parameter configurations, generated from the grid-search. For each such parameterization, the reported results are the five-trial sample average $\bar{\mu}$ and standard error $\bar{\sigma}$, where their ranking is based on the estimated feedback rate in descending order. Interestingly, though utilizing the image features $\boldsymbol{i}_{t,a}^{(\mathrm{u})}$ has the bandit produce one of the highest feedback rates and the lowest regrets in this survey, the majority of the bandit arrangements does not have a clear separation between including and excluding the convolutional features, but displays a more interwoven structure of the two groups compared to some of the aforementioned bandits. In other words, this algorithm is comparatively more sensitive to its hyper-parameter initialization than the others in this survey.

	Parameters			Feedback (%)		Regret (10^3)	
	δ	λ	$\boldsymbol{i}_{t,a}^{(\mathrm{u})}$	$\bar{\mu}$	$\bar{\sigma}$	$\bar{\mu}$	$\bar{\sigma}$
Rank							
1	0.1	10.0	✓	31.25	0.12	91.25	1.65
2	1.0	10.0	✓	29.24	0.31	113.56	3.47
3	0.1	10.0	✗	27.13	0.03	136.81	0.71
4	0.1	1.0	✓	26.55	0.63	141.97	6.21
5	1.0	10.0	✗	25.15	0.03	156.14	0.82
6	0.1	1.0	✗	24.65	0.61	161.61	5.69
7	0.1	0.1	✓	23.43	0.71	175.13	7.10
8	1.0	1.0	✓	22.57	0.41	183.83	3.88
9	0.1	0.1	✗	22.11	0.34	188.23	3.05
10	1.0	1.0	✗	20.53	0.19	204.05	2.25
11	10.0	10.0	✓	20.34	0.05	206.82	0.39
12	1.0	0.1	✓	18.64	0.18	222.88	1.87
13	10.0	10.0	✗	18.16	0.08	227.84	0.67
14	1.0	0.1	✗	17.99	0.09	229.34	0.67
15	10.0	1.0	✓	17.80	0.00	231.41	0.13
16	10.0	1.0	✗	17.39	0.06	234.66	0.47
17	10.0	0.1	✓	17.35	0.03	235.58	0.12
18	10.0	0.1	✗	17.28	0.02	236.15	0.22

B.8 `NeuralUCB`

The selected representative of the `NeuralUCB` algorithm has the parameterization $\delta = 0.1$, $\lambda = 1.0$ and $\gamma = 0.01$, which corresponds to the exploration multiplier, regularization factor respective learning rate—used when training the internal neural network. Observe the large span of converged feedback rates and regrets, some of which have the highest averages estimated and lowest simulated reward gap summations observed in this survey.

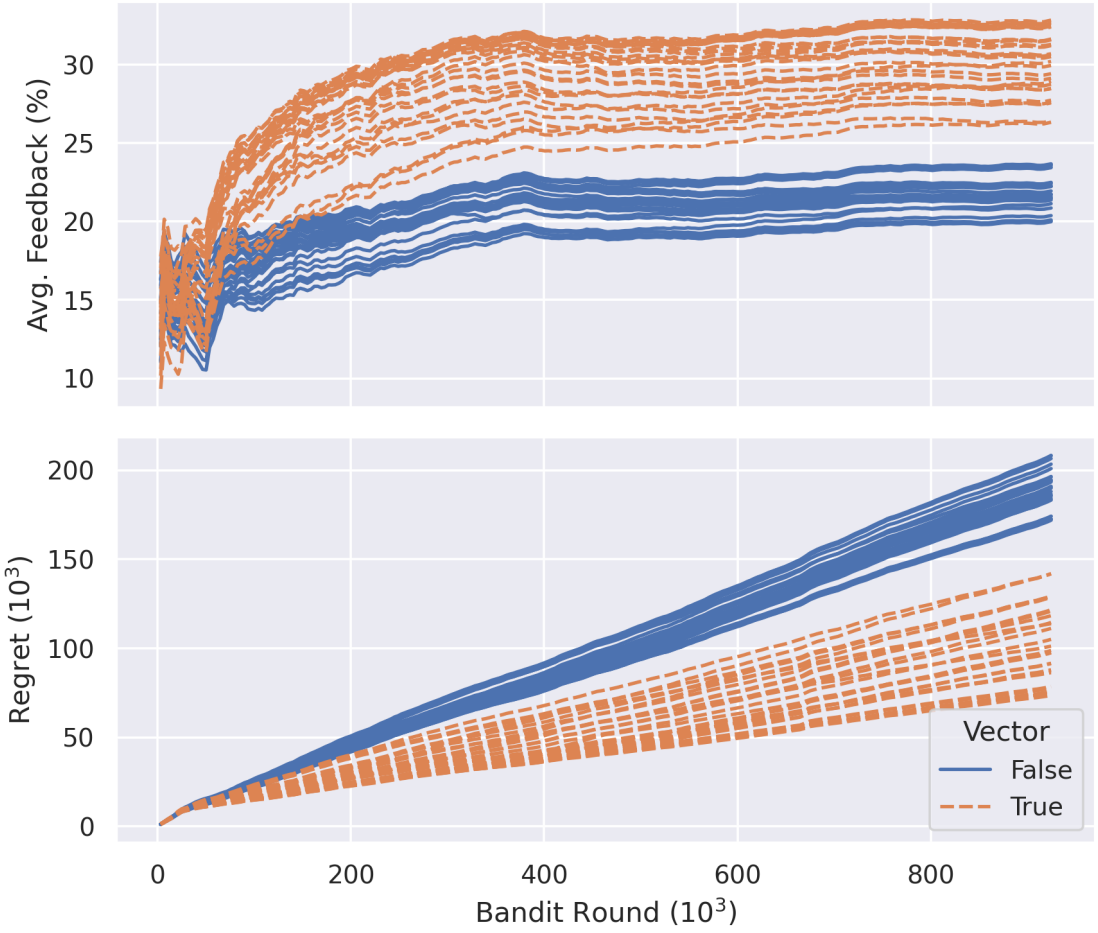


Figure B.8: The cumulative feedback rate and regret profiles from the `NeuralUCB` algorithm's hyper-parameter tuning, where the trace estimates are based on a single offline simulation trial due to time limitations. Furthermore, by being a contextual bandit, there are two types of lines in the figure panes, namely solid blue and dashed orange lines that correspond to excluding and including the convolutional features $\boldsymbol{i}_{t,a}^{(\mathrm{u})}$. One interesting observation is the spread of the feedback rates early on in the simulation, which seems to originate from a single point for the other algorithms, but can probably be credited to the approximation noise resulting from the internal neural network. Moreover, notice how every trace that includes the convolutional features $\boldsymbol{i}_{t,a}^{(\mathrm{u})}$ consistently outperforms their counterparts who exclude it, which can be seen by their distinct groupings characterized by their line style and hue.

Table B.8: This table contains the `NeuralUCB` algorithm's feedback rates and total regrets, which are generated from the hyper-parameter tuning, estimated through its five-trial sample average $\bar{\mu}$ and associated standard error $\bar{\sigma}$. It would appear that utilizing the image vector $\boldsymbol{i}_{t,a}^{(\mathrm{u})}$ provides a distinct advantage over not employing it, which can be seen by the natural partitioning of the table, where the first and second halves correspond to including and excluding the convolutional features, respectively. Furthermore, this bandit is one of the few that scores feedback rates above 30% and total regrets below the 100,000 point, while still having some of its worst runs producing estimates similar to those of the context-free algorithms' best.

	Parameters				Feedback (%)		Regret (10^3)	
Rank	δ	λ	γ	$\boldsymbol{i}_{t,a}^{(\mathrm{u})}$	$\bar{\mu}$	$\bar{\sigma}$	$\bar{\mu}$	$\bar{\sigma}$
1	0.1	1.0	0.01	✓	32.84	0.14	73.52	1.94
2	0.1	10.0	0.01	✓	32.73	0.15	74.90	2.01
3	1.0	0.1	0.01	✓	32.63	0.05	75.13	1.26
4	1.0	10.0	0.01	✓	32.60	0.44	75.80	5.22
5	1.0	1.0	0.01	✓	32.46	0.13	77.59	1.17
6	0.1	0.1	0.01	✓	32.40	0.25	78.55	3.33
7	0.1	1.0	0.001	✓	31.63	0.38	87.40	4.43
8	0.1	10.0	0.001	✓	31.61	0.43	87.96	4.26
9	10.0	10.0	0.01	✓	31.59	0.23	86.36	2.99
10	0.1	0.1	0.001	✓	31.29	0.69	91.46	6.85
11	1.0	10.0	0.001	✓	31.22	0.56	91.61	5.86
12	1.0	1.0	0.001	✓	30.69	0.66	97.52	7.08
13	1.0	0.1	0.001	✓	30.64	0.31	97.66	3.44
14	1.0	10.0	0.1	✓	30.51	0.94	98.83	10.05
15	10.0	1.0	0.01	✓	30.21	0.11	101.21	0.98
16	0.1	10.0	0.1	✓	29.95	1.79	105.11	19.77
17	10.0	10.0	0.001	✓	29.42	0.48	111.24	4.00
18	0.1	0.1	0.1	✓	29.13	1.69	114.58	19.17
19	10.0	0.1	0.01	✓	28.94	0.21	113.87	1.86
20	1.0	0.1	0.1	✓	28.75	0.99	118.37	11.00
21	1.0	1.0	0.1	✓	28.48	2.22	120.93	25.30
22	0.1	1.0	0.1	✓	28.48	2.24	121.70	24.86
23	10.0	10.0	0.1	✓	27.73	1.50	128.73	15.32
24	10.0	0.1	0.1	✓	27.62	1.70	129.17	17.87
25	10.0	1.0	0.001	✓	27.56	0.53	129.28	6.01
26	10.0	1.0	0.1	✓	26.41	0.08	141.92	0.58
27	10.0	0.1	0.001	✓	26.33	0.20	141.92	2.71
28	1.0	10.0	0.01	✗	23.68	0.21	171.99	2.03
29	0.1	0.1	0.01	✗	23.63	0.06	172.56	0.26
30	0.1	1.0	0.01	✗	23.60	0.14	172.60	1.08
31	1.0	1.0	0.01	✗	23.60	0.16	172.63	1.63
32	0.1	10.0	0.01	✗	23.49	0.19	174.16	1.78

Continued on next page

Rank	Parameters				Feedback (%)		Regret (10^3)	
	δ	λ	γ	$\boldsymbol{i}_{t,a}^{(\mathrm{u})}$	$\bar{\mu}$	$\bar{\sigma}$	$\bar{\mu}$	$\bar{\sigma}$
33	1.0	0.1	0.01	✗	23.43	0.22	174.27	1.74
34	10.0	10.0	0.01	✗	22.49	0.37	183.30	3.72
35	0.1	10.0	0.001	✗	22.46	0.42	183.60	3.72
36	0.1	0.1	0.001	✗	22.39	0.38	184.46	3.61
37	1.0	10.0	0.001	✗	22.26	0.12	185.73	0.88
38	0.1	1.0	0.001	✗	22.25	0.13	185.77	1.11
39	1.0	1.0	0.1	✗	22.23	0.57	186.64	5.86
40	1.0	0.1	0.001	✗	21.95	0.34	188.14	3.44
41	1.0	1.0	0.001	✗	21.74	0.32	190.01	3.05
42	10.0	1.0	0.01	✗	21.73	0.20	190.51	2.16
43	10.0	10.0	0.001	✗	21.70	0.25	191.09	2.27
44	0.1	10.0	0.1	✗	21.59	0.79	193.49	7.93
45	0.1	1.0	0.1	✗	21.53	1.28	194.11	13.36
46	1.0	0.1	0.1	✗	21.47	1.04	194.71	10.27
47	10.0	10.0	0.1	✗	21.46	1.05	194.32	10.47
48	1.0	10.0	0.1	✗	21.37	1.59	196.18	15.79
49	10.0	0.1	0.01	✗	21.14	0.32	196.57	3.00
50	0.1	0.1	0.1	✗	20.86	1.52	201.02	15.42
51	10.0	1.0	0.001	✗	20.38	0.23	203.52	2.25
52	10.0	0.1	0.001	✗	20.10	0.13	206.65	1.10
53	10.0	1.0	0.1	✗	20.04	0.19	208.18	1.73
54	10.0	0.1	0.1	✗	19.99	0.19	208.44	1.80