EE-NET: EXPLOITATION-EXPLORATION NEURAL NETWORKS IN CONTEXTUAL BANDITS

Anonymous authors
Paper under double-blind review

ABSTRACT

Contextual multi-armed bandits have been studied for decades and adapted to various applications such as online advertising and personalized recommendation. To solve the exploitation-exploration tradeoff in bandits, there are three main techniques: epsilon-greedy, Thompson Sampling (TS), and Upper Confidence Bound (UCB). In recent literature, linear contextual bandits have adopted ridge regression to estimate the reward function and combine it with TS or UCB strategies for exploration. However, this line of works explicitly assumes the reward is based on a linear function of arm vectors, which may not be true in real-world datasets. To overcome this challenge, a series of neural-based bandit algorithms have been proposed, where a neural network is assigned to learn the underlying reward function and TS or UCB are adapted for exploration. In this paper, we propose "EE-Net", a neural-based bandit approach with a novel exploration strategy. In addition to utilizing a neural network (Exploitation network) to learn the reward function, EE-Net adopts another neural network (Exploration network) to adaptively learn potential gains compared to currently estimated reward. Then, a decision-maker is constructed to combine the outputs from the Exploitation and Exploration networks. We prove that EE-Net achieves $\mathcal{O}(\sqrt{T \log T})$ regret, which is tighter than existing state-of-the-art neural bandit algorithms ($\mathcal{O}(\sqrt{T} \log T)$ for both UCB-based and TS-based). Through extensive experiments on four real-world datasets, we show that EE-Net outperforms existing linear and neural bandit approaches.

1 INTRODUCTION

The stochastic contextual multi-armed bandit (MAB) (Dani et al., 2008; Lattimore and Szepesvári, 2020) has been studied for decades in machine learning community to solve sequential decision making, with applications in online advertising (Li et al., 2010), personal recommendation (Wu et al., 2016; Ban and He, 2021b), etc. In the standard contextual bandit setting, a set of n arms are presented to a learner in each round, where each arm is represented by a context vector. Then by certain strategy, the learner selects and plays one arm, receiving a reward. The goal of this problem is to maximize the cumulative rewards of T rounds.

MAB algorithms have principled approaches to address the trade-off between Exploitation and Exploration (EE), as the collected data from past rounds should be exploited to get good rewards but also under-explored arms need to be explored with the hope of getting even better rewards. The most widely-used approaches for EE trade-off can be classified into three main techniques: Epsilon-greedy (Langford and Zhang, 2008), Thompson Sampling (TS) (Thompson, 1933), and Upper Confidence Bound (UCB) (Auer, 2002).

Linear contextual bandits (Dani et al., 2008; Li et al., 2010; Abbasi-Yadkori et al., 2011), where the reward is assumed to be a linear function with respect to arm vectors, have been well studied and succeeded both empirically and theoretically. Given an arm, ridge regression is usually adapted to estimate its reward based on collected data from past rounds. UCB-based algorithms (Li et al., 2010; Chu et al., 2011; Wu et al., 2016; Ban and He, 2021b) calculate an upper bound for the confidence ellipsoid of estimated reward and determine the arm according to the sum of estimated reward and UCB. TS-based algorithms (Agrawal and Goyal, 2013; Abeille and Lazaric, 2017) formulate each arm as a posterior distribution where mean is the estimated reward and choose the one with the

Table 1: Selection Criterion Comparison ($\mathbf{x}_t$: selected arm in round t).

Methods	Selection Criterion
Neural Epsilon-greedy	With probability $1-\delta$, $\mathbf{x}_t = \arg\max_{i\in[n]} f_1(\mathbf{x}_{t,i};\boldsymbol{\theta}^1)$; Otherwise, select $\mathbf{x}_t$ randomly.
NeuralTS (Zhang et al., 2020)	For $\mathbf{x}_{t,i}, \forall i \in [n]$, draw $\hat{r}_{t,i}$ from $\mathcal{N}(f_1(\mathbf{x}_{t,i};\boldsymbol{\theta}^1), {\sigma_{t,i}}^2)$. Then, $\mathbf{x}_t = \arg\max_{i\in[n]} \hat{r}_{t,i}$.
NeuralUCB (Zhou et al., 2020)	$\mathbf{x}_t = \arg\max_{i\in[n]} \left(f_1(\mathbf{x}_{t,i};\boldsymbol{\theta}^1) + \text{UCB}_{t,i}\right)$.
EE-Net (Our approach)	$\forall i \in [n]$, compute $f_1(\mathbf{x}_{t,i};\boldsymbol{\theta}^1)$, $f_2\left(\nabla_{\boldsymbol{\theta}^1} f_1(\mathbf{x}_{t,i};\boldsymbol{\theta}^1);\boldsymbol{\theta}^2\right)$ (Exploration Net). Then $\mathbf{x}_t = \arg\max_{i\in[n]} f_3(f_1, f_2;\boldsymbol{\theta}^3)$.

maximal sampled reward. However, the linear assumption regarding the reward may not be true in real-world applications (Valko et al., 2013).

To learn non-linear reward functions, recent works have utilized deep neural networks to learn the underlying reward function, thanks to its powerful representation ability. Considering the past selected arms and received rewards as training samples, a neural network f_1 is built for exploitation. Zhou et al. (2020) computes a gradient-based upper confidence bound with respect to f_1 and uses UCB strategy to select arms. Zhang et al. (2020) formulates each arm as a normal distribution where the mean is f_1 and deviation is calculated based on gradient of f_1, and then uses the TS strategy to choose arms. Both Zhou et al. (2020) and Zhang et al. (2020) achieve the near-optimal regret bound of $O(\sqrt{T}\log T)$.

In this paper, we propose a neural-based bandit algorithm coming with a novel exploration strategy, named "EE-Net". Similar to other neural bandits, EE-Net has an exploitation network f_1 to estimate rewards for each arm. The crucial difference from existing works is that EE-Net has an exploration network f_2 to predict the potential gain for each arm compared to current reward estimate. The input to the exploration network is the gradient of f_1 and the ground-truth is residual between received reward and estimated reward. The strategy is inspired by recent advances in the neural-based UCB (Ban et al., 2021; Zhou et al., 2020). Finally, a decision-maker f_3 is constructed to select arms. f_3 has two modes: linear or nonlinear. In linear mode, f_3 is a linear combination of f_1 and f_2, inspired by the UCB strategy. In the nonlinear mode, f_3 is formulated as a neural network with input (f_1, f_2) and the goal is to learn the probability of being an optimal arm for each arm. Table 1 summarizes the selection criterion difference of EE-Net from other neural bandit algorithms. To sum up, the contributions of this paper can be summarized as follows:

1. We propose a novel exploration strategy, EE-Net, where a neural network is assigned to learn the potential gain compared to the current estimation.
2. Under standard assumptions of over-parameterized neural networks, we prove that EE-Net can achieve the regret upper bound of $\mathcal{O}(\sqrt{T\log T})$, which is tighter than existing state-of-the-art bandit algorithms.
3. We conduct extensive experiments on four real-world datasets, showing that EE-Net outperforms baselines crossing ϵ-greedy, TS, and UCB, and becomes the new state-of-the-art exploration policy.

Next, we will show the standard problem definition and elaborate the proposed EE-Net, before we present our theoretical analysis. In the end, we provide the empirical evaluation and conclusion.

2 Related Work

Constrained Contextual bandits. The common constrain placed on the reward function is the linear assumption, usually calculated by ridge regression (Dani et al., 2008; Li et al., 2010; Abbasi-Yadkori et al., 2011; Valko et al., 2013). The linear UCB-based bandit algorithms (Abbasi-Yadkori et al., 2011; Li et al., 2016) and the linear Thompson Sampling (Agrawal and Goyal, 2013; Abeille and Lazaric, 2017) can achieve successful performance and the near-optimal regret bound of $\tilde{\mathcal{O}}(\sqrt{T})$. To

break the linear assumption, Filippi et al. (2010) generalizes the reward function to a composition of linear and non-linear functions and then adopt a UCB-based algorithm to deal with it; Bubeck et al. (2011) imposes the Lipschitz property on reward metric space and constructs a hierarchical optimistic optimization to make selections; Valko et al. (2013) embeds the reward function into Reproducing Kernel Hilbert Space and proposes the kernelized TS/UCB bandit algorithms.

Neural Bandits. To learn non-linear reward functions, deep neural networks have been adapted to bandits with various variants. Riquelme et al. (2018); Lu and Van Roy (2017) build L-layer DNN to learn the arm embeddings and apply Thompson sampling on the last layer for exploration. Zhou et al. (2020) first introduces a provable neural-based contextual bandit algorithm with a UCB exploration strategy and then Zhang et al. (2020) extends the neural network to Thompson sampling framework. Their regret analysis is built on recent advances on the convergence theory in over-parameterized neural networks(Du et al., 2019; Allen-Zhu et al., 2019) and utilizes Neural Tangent Kernel (Jacot et al., 2018; Arora et al., 2019) to construct connections with linear contextual bandits (Abbasi-Yadkori et al., 2011). Ban and He (2021a) further adopts convolutional neural networks with UCB exploration aiming for visual-aware applications. Xu et al. (2020) performs UCB-based exploration on the last layer of neural networks to reduce the computational cost brought by gradient-based UCB. Different from the above existing works, EE-Net keeps the powerful representation ability of neural networks to learn reward function and first assigns another neural network to determine exploration.

3 Problem definition

We consider the standard contextual multi-armed bandit with the known number of rounds T (Zhou et al., 2020; Zhang et al., 2020). In each round $t \in [T]$, where the sequence $[T] = [1, 2, \dots, T]$, the learner is presented with n arms, in which each arm is represented by a feature vector $\mathbf{x}_{t,i} \in \mathbb{R}^d$ for each $i \in [n]$. After playing one arm $\mathbf{x}_{t,i}$, its reward $r_{t,i}$ is assumed to be generated by the function:

$$r_{t,i} = h(\mathbf{x}_{t,i}) + \eta_{t,i}, \tag{3.1}$$

where the *unknown* reward function $h(\mathbf{x}_{t,i})$ can be either linear or non-linear and the noise $\eta_{t,i}$ is drawn from certain distribution with expectation $\mathbb{E}[\eta_{t,i}] = 0$. Following many existing works (Zhou et al., 2020; Ban et al., 2021; Zhang et al., 2020), we consider bounded rewards, $r_{t,i} \in [a, b]$. For the brevity, we denote the selected arm in round t by $\mathbf{x}_t$ and the reward received in t by r_t. The standard *regret* of T rounds is defined as:

$$\mathbf{R}_T = \mathbb{E}\left[\sum_{t=1}^{T}(r_t^* - r_t)\right] = \sum_{t=1}^{T}(h(\mathbf{x}_t^*) - h(\mathbf{x}_t)), \tag{3.2}$$

where $\mathbf{x}_t^* = \arg\max_{i \in [n]} h(\mathbf{x}_{t,i})$. The goal of this problem is to minimize $\mathbf{R}_T$ by certain selection strategy.

Notation. We denote by $\{\mathbf{x}_i\}_{i=1}^t$ the sequence $(\mathbf{x}_1, \dots, \mathbf{x}_t)$. We use $\|v\|_2$ or $\|v\|$ to denote the Euclidean norm for a vector v, and $\|\mathbf{W}\|_2$ and $\|\mathbf{W}\|_F$ to denote the spectral and Frobenius norm for a matrix $\mathbf{W}$. We use $\langle \cdot, \cdot \rangle$ to denote the standard inner product between two vectors or two matrices.

4 Proposed Method: EE-Net

EE-Net is composed of three independent components, while their input and output are closely correlated. The first component is the *exploitation network*, $f_1(\cdot; \boldsymbol{\theta}^1)$, which is to learn the unknown reward function h based on the data collected in past rounds. The second component is the *exploration network*, $f_2(\cdot; \boldsymbol{\theta}^2)$, which is to measure the exploration efforts we should make in the present round. The third component is the *decision-maker*, f_3, which is to further trade off between exploitation and exploration, and make the selection.

1) Exploitation Net. A neural network model f_1 is provided to learn the mapping from arms to rewards. In round t, denote the network by $f_1(\cdot; \boldsymbol{\theta}_{t-1}^1)$, where the superscript of $\boldsymbol{\theta}_{t-1}^1$ is the index of network and the subscript represents the round where the parameters of f_1 finished the last update. Given an arm $\mathbf{x}_{t,i}, i \in [n]$, $f_1(\mathbf{x}_{t,i}; \boldsymbol{\theta}_{t-1}^1)$ is considered the "exploitation score" for $\mathbf{x}_{t,i}$. By some criterion, after playing arm $\mathbf{x}_t$, we receive a reward r_t. Therefore, we can conduct gradient descent to

Table 2: Structure of EE-Net (Round t).

Input	Network	Label
$\{\mathbf{x}_i\}_{i=1}^t$	$f_1(\cdot;\boldsymbol{\theta}^1)$ (Exploitation)	$\{r_i\}_{i=1}^t$
$\{\triangledown_{\boldsymbol{\theta}} f_1(\mathbf{x}_i;\boldsymbol{\theta}_i^1)\}_{i=1}^t$	$f_2(\cdot;\boldsymbol{\theta}^2)$ (Exploration)	$\left\{\left(r_i - f_1(\mathbf{x}_i;\boldsymbol{\theta}_i^1)\right)\right\}_{i=1}^t$
$\{\left(f_1(\mathbf{x}_i;\boldsymbol{\theta}_i^1), f_2(\triangledown_{\boldsymbol{\theta}} f_1;\boldsymbol{\theta}_i^2)\right)\}_{i=1}^t$	$f_3(\cdot;\boldsymbol{\theta}^3)$ (Decision-maker with non-linear function)	$\{p_i\}_{i=1}^t$

update $\boldsymbol{\theta}^1$ based on the collected training samples $\{\mathbf{x}_i, r_i\}_{i=1}^t$ and denote the updated parameters by $\boldsymbol{\theta}_t^1$.

2) Exploration Net. Our exploration strategy is inspired by existing UCB-based neural bandits (Zhou et al., 2020; Ban et al., 2021). Based on the Lemma 5.2 in (Ban et al., 2021), given an arm $\mathbf{x}_{t,i}$, with probability at least $1-\delta$, we have the following form:

$$|h(\mathbf{x}_{t,i}) - f_1(\mathbf{x}_{t,i};\boldsymbol{\theta}_t^1)| \leq \Psi(\triangledown_{\boldsymbol{\theta}^1} f_1(\mathbf{x}_{t,i};\boldsymbol{\theta}_t^1)), \tag{4.1}$$

where h is defined in Eq. (3.1) and Ψ is the upper confidence bound represented by a function with respect to the gradient $\triangledown_{\boldsymbol{\theta}^1} f_1(\mathbf{x}_{t,i};\boldsymbol{\theta}_t^1)$ (See more details and discussions in **Appendix D**). Then we have the following definition.

Definition 4.1. *Given an arm $\mathbf{x}_{t,i}$, we define $h(\mathbf{x}_{t,i}) - f_1(\mathbf{x}_{t,i};\boldsymbol{\theta}_t^1)$ as the "*expected potential gain*" for $\mathbf{x}_{t,i}$ and $r_{t,i} - f_1(\mathbf{x}_{t,i};\boldsymbol{\theta}_t^1)$ as the "*potential gain*" for $\mathbf{x}_{t,i}$.*

Let $y_{t,i} = r_{t,i} - f_1(\mathbf{x}_{t,i};\boldsymbol{\theta}_t^1)$. When $y_{t,i} > 0$, the arm $\mathbf{x}_{t,i}$ has positive potential gain compared to the estimated reward $f_1(\mathbf{x}_{t,i};\boldsymbol{\theta}_t^1)$. A large positive $y_{t,i}$ makes the arm more suitable for exploration, whereas a small (or negative) $y_{t,i}$ makes the arm unsuitable for exploration. Recall that traditional approaches such as UCB effectively compute such potential gain $y_{t,i}$ using standard tools, e.g., Markov inequality, Hoeffding bounds, etc. from large deviation bounds.

Instead of calculating a large-deviation based statistical form for Ψ, we use a neural network f_2 to learn Ψ, where the input is $\triangledown_{\boldsymbol{\theta}^1} f_1(\mathbf{x}_{t,i};\boldsymbol{\theta}_t^1)$ and the ground truth is $y_{t,i}$. Adopting gradient $\triangledown_{\boldsymbol{\theta}^1} f_1(\mathbf{x}_{t,i};\boldsymbol{\theta}_t^1)$ as the input also is due to the fact that it incorporates two aspects of information: the feature of arm and the discriminative information of f_1.

To sum up, we consider $f_2(\triangledown_{\boldsymbol{\theta}^1} f_1(\mathbf{x}_{t,i};\boldsymbol{\theta}_t^1);\boldsymbol{\theta}_t^2)$ as the "exploration score" of $\mathbf{x}_{t,i}$, because it indicates the potential gain of $\mathbf{x}_{t,i}$ compared to our current exploitation score f_1. Given the selected arm $\mathbf{x}_t$, let $y_t = r_t - f_1(\mathbf{x}_t;\boldsymbol{\theta}_t^1)$. Therefore, in round t, we can use gradient descent to update $\boldsymbol{\theta}^2$ based on collected training samples $\{\triangledown_{\boldsymbol{\theta}^1} f_1(\mathbf{x}_i;\boldsymbol{\theta}_i^1), y_i\}_i^t$. We also provide other two heuristic forms: $y_{t,i} = |r_{t,i} - f_1(\mathbf{x}_{t,i};\boldsymbol{\theta}_t^1)|$ and $y_{t,i} = \text{ReLU}(r_{t,i} - f_1(\mathbf{x}_{t,i};\boldsymbol{\theta}_t^1))$. We compare them in an ablation study in **Appendix A**.

3) Decision-maker. In round t, given an arm $\mathbf{x}_{t,i}, i \in [n]$, with the computed exploitation score $f_1(\mathbf{x}_{t,i};\boldsymbol{\theta}_t^1)$ and exploration score $f_2(\triangledown_{\boldsymbol{\theta}^1} f_1;\boldsymbol{\theta}_t^2)$, we use a function $f_3\left(f_1, f_2;\boldsymbol{\theta}^3\right)$ to trade off between exploitation and exploration and compute the final score for $\mathbf{x}_{t,i}$. The selection criterion is defined as

$$\mathbf{x}_t = \arg\max_{i\in[n]} f_3\left(f_1(\mathbf{x}_{t,i};\boldsymbol{\theta}_{t-1}^1), f_2\left(\triangledown_{\boldsymbol{\theta}^1} f_1(\mathbf{x}_{t,i};\boldsymbol{\theta}_{t-1}^1);\boldsymbol{\theta}_{t-1}^2\right);\boldsymbol{\theta}_{t-1}^3\right).$$

Note that f_3 can be either linear or non-linear functions. We provide the following two forms.

(1) *Linear function*. f_3 can be formulated as a linear function with respect to f_1 and f_2 :

$$f_3(f_1, f_2;\boldsymbol{\theta}^3) = w_1 f_1(\mathbf{x}_{t,i};\boldsymbol{\theta}^1) + w_2 f_2(\triangledown_{\boldsymbol{\theta}^1} f_1;\boldsymbol{\theta}^2)$$

where w_1, w_2 are two weights preset by the learner. When $w_1 = w_2 = 1$, f_3 can be thought of as UCB-type policy, where the estimated reward f_1 and potential gain f_2 are simply added together. In experiments, we report its empirical performance in ablation study (**Appendix A**).

(2) *Non-linear function*. f_3 also can be formulated as a neural network to learn the mapping from (f_1, f_2) to the optimal arm. We transform the bandit problem into a binary classification problem.

Algorithm 1 EE-Net

Input: f_1, f_2, f_3, T (number of rounds), η_1 (learning rate for f_1), η_2 (learning rate for f_2), η_3 (learning rate for f_3), K_1 (number of iterations for f_1), K_2 (number of iterations for f_2) , K_3 (number of iterations for f_3), c (tuning parameter)

Initialize $\boldsymbol{\theta}_0^1, \boldsymbol{\theta}_0^2, \boldsymbol{\theta}_0^3$
for each $t \in [T]$ **do**
  Observe n arms $\{\mathbf{x}_{t,1}, \dots, \mathbf{x}_{t,n}\}$
  for each $i \in [n]$ **do**
    Compute $f_1(\mathbf{x}_i; \boldsymbol{\theta}_{t-1}^1)$, $f_2(\nabla_{\boldsymbol{\theta}_{t-1}^1} f_1(\mathbf{x}_i; \boldsymbol{\theta}_{t-1}^1)/c\sqrt{mL}; \boldsymbol{\theta}_{t-1}^2)$, $f_3((f_1, f_2); \boldsymbol{\theta}_{t-1}^3)$
  end for
  $\mathbf{x}_t = \arg\max_{i \in [n]} f_3\left(f_1(\mathbf{x}_{t,i}; \boldsymbol{\theta}_{t-1}^1), f_2(\nabla_{\boldsymbol{\theta}^1} f_1; \boldsymbol{\theta}_{t-1}^2); \boldsymbol{\theta}_{t-1}^3\right)$
  Play $\mathbf{x}_t$ and observe reward r_t
  $\boldsymbol{\theta}_t^1, \boldsymbol{\theta}_t^2, \boldsymbol{\theta}_t^3$ = GradientDescent($\boldsymbol{\theta}_0^1, \boldsymbol{\theta}_0^2, \boldsymbol{\theta}_0^3$, $\{\mathbf{x}_i\}_{i=1}^t$, $\{r_i\}_{i=1}^t$)
end for

procedure GRADIENTDESCENT($\boldsymbol{\theta}_0$, $\{\mathbf{x}_i\}_{i=1}^t$, $\{r_i\}_{i=1}^t$)
  $\mathcal{L}_1 = \frac{1}{2}\sum_{i=1}^t \left(f_1(\mathbf{x}_i; \boldsymbol{\theta}^1) - r_i\right)^2$
  $\boldsymbol{\theta}^{1,(0)} = \boldsymbol{\theta}_0^1$
  for $i \in \{1, \dots, K_1\}$ **do**
    $\boldsymbol{\theta}^{1,(i)} = \boldsymbol{\theta}^{1,(i-1)} - \eta_1 \nabla_{\boldsymbol{\theta}^{1,(i-1)}} \mathcal{L}_1$
  end for
  $\mathcal{L}_2 = \frac{1}{2}\sum_{i=1}^t \left(f_2(\nabla_{\boldsymbol{\theta}^1} f_1/c\sqrt{mL}; \boldsymbol{\theta}^2) - (r_i - f_1(\mathbf{x}_i; \boldsymbol{\theta}^{1,(K_1)}))\right)^2$
  $\boldsymbol{\theta}^{2,(0)} = \boldsymbol{\theta}_0^2$
  for $i \in \{1, \dots, K_2\}$ **do**
    $\boldsymbol{\theta}^{2,(i)} = \boldsymbol{\theta}^{2,(i-1)} - \eta_2 \nabla_{\boldsymbol{\theta}^{2,(i-1)}} \mathcal{L}_2$
  end for
  if Applicable **then**
    Determine label p_t
    $\mathcal{L}_3 = -\frac{1}{t}\sum_{i=1}^t \left[p_t \log f_3((f_1, f_2); \boldsymbol{\theta}^3) + (1 - p_t)\log(1 - f_3((f_1, f_2); \boldsymbol{\theta}^3))\right]$
    $\boldsymbol{\theta}^{3,(0)} = \boldsymbol{\theta}_0^3$
    for $i \in \{1, \dots, K_3\}$ **do**
      $\boldsymbol{\theta}^{3,(i)} = \boldsymbol{\theta}^{3,(i-1)} - \eta_3 \nabla_{\boldsymbol{\theta}^{3,(i-1)}} \mathcal{L}_3$
    end for
    Return $\boldsymbol{\theta}^{1,(K_1)}, \boldsymbol{\theta}^{2,(K_2)}, \boldsymbol{\theta}^{3,(K_3)}$
  end if
  Return $\boldsymbol{\theta}^{1,(K_1)}, \boldsymbol{\theta}^{2,(K_2)}, \boldsymbol{\theta}_0^3$
end procedure

Given an arm $\mathbf{x}_{t,i}$, we define $p_{t,i}$ as the probability of being the optimal arm for $\mathbf{x}_{t,i}$ in round t. For brevity, we denote by p_t the probability of being the optimal arm for the selected arm $\mathbf{x}_t$ in round t. According to different reward distributions, we have different approaches to determine p_t.

1. *Binary reward*. $\forall t \in [T]$, suppose r_t is a binary variable over $a, b (a < b)$, it is straightforward to set: $p_t = 1.0$ if $r_t = b$; $p_t = 0.0$, otherwise.
2. *Continuous reward*. $\forall t \in [T]$, suppose r_t is a continuous variable over the range $[a, b]$, we provide two ways to determine p_t. (1) p_t can be directly set as $\frac{r_t - a}{b - a}$. (2) The learner can set a threshold $\theta, (a < \theta < b)$. Then $p_t = 1.0$ if $r_t > \theta$; $p_t = 0.0$, otherwise.

Therefore, with the collected training samples $\{\left(f_1(\mathbf{x}_i; \boldsymbol{\theta}_i^1), f_2(\nabla_{\boldsymbol{\theta}^1} f_1; \boldsymbol{\theta}_i^2)\right), p_i\}_{i=1}^t$ in round t, we can conduct gradient descent to update parameters of $f_3(f_1, f_2; \boldsymbol{\theta}_t^3)$. Table 2 details the working structure of EE-Net and Algorithm 1 depicts the workflow of EE-Net.

Remark 4.1. The networks f_1, f_2, f_3 can be different structures according to different applications. For example, in the vision tasks, f_1 can be set up as convolutional layers (LeCun et al., 1995). For the exploration network f_2, the input $\nabla_{\boldsymbol{\theta}^1} f_1$ may have exploding dimensions when the exploitation

network f_1 becomes wide and deep, which may cause huge computation cost for f_2. To address this challenge, we can apply dimensionality reduction techniques (Roweis and Saul, 2000; Van Der Maaten et al., 2009) to obtain low-dimensional vectors of $\triangledown_{\boldsymbol{\theta}^1} f_1$. In the experiments, we use Roweis and Saul (2000) to acquire a 10-dimensional vector for $\triangledown_{\boldsymbol{\theta}^1} f_1$ and achieve the best performance among all baselines. □

Remark 4.2. EE-Net has the ability to determine exploration direction. Given an arm $\mathbf{x}_{t,i}$, when the estimation $f_1(\mathbf{x}_{t,i})$ is *lower* than the expected reward $h(\mathbf{x}_{t,i})$, the learner should make the "upward" exploration, i.e., increase the chance of $\mathbf{x}_{t,i}$ being explored; When $f_1(\mathbf{x}_{t,i})$ is *higher* than $h(\mathbf{x}_{t,i})$, the learner should do the "downward" exploration, i.e., decrease the chance of $\mathbf{x}_{t,i}$ being explored. EE-Net uses the neural network f_2 to learn $h(\mathbf{x}_{t,i}) - f_1(\mathbf{x}_{t,i})$ (which has positive and negative scores) and has the ability to determine the exploration direction. In contrast, NeuralUCB will always make "upward" exploration and NeuralTS will randomly choose between "upward" exploration and "downward" exploration (see selection criteria in Table 1). □

Remark 4.3. EE-Net reduces the space complexity from $\mathcal{O}(p^2)$ to $\mathcal{O}(p)$ (let $\boldsymbol{\theta}^1 \in \mathbb{R}^p$). NeuralUCB and NeuralTS have to maintain the gradient outer product matrix $\mathbf{A}_t = \mathbf{I} + \sum_{i=1}^{t} \triangledown_{\boldsymbol{\theta}^1} f_1(\mathbf{x}_i; \boldsymbol{\theta}_i^1) \triangledown_{\boldsymbol{\theta}^1} f_1(\mathbf{x}_i; \boldsymbol{\theta}_i^1)^\top \in \mathbb{R}^{p \times p}$ to store the historical gradients. On the contrary, EE-Net does not have this matrix and only regards $\triangledown_{\boldsymbol{\theta}^1} f_1$ as the input of f_2. □

5 Regret Analysis

In this section, we provide the regret analysis of EE-Net when f_3 is set as the linear function $f_3 = f_1 + f_2$, which can be thought of as the UCB-type trade-off between exploitation and exploration. For the sake of simplicity, we conduct the regret analysis on some unknown but fixed data distribution $\mathcal{D}$. In each round t, n samples $\{(\mathbf{x}_{t,1}, r_{t,1}), (\mathbf{x}_{t,2}, r_{t,2}), \ldots, (\mathbf{x}_{t,n}, r_{t,n})\}$ are drawn from $\mathcal{D}$, where $i \in [n]$, $\mathbf{x}_{t,i}$ is the representation of arm satisfying $\|\mathbf{x}_{t,i}\|_2 = 1$ and $r_{t,i}$ is the corresponding reward satisfying $r_{t,i} \in [0, 1]$, which are standard assumptions in neural bandits (Zhou et al., 2020; Zhang et al., 2020).

The analysis will focus on over-parameterized neural networks (Jacot et al., 2018; Du et al., 2019; Allen-Zhu et al., 2019). Given an input $\mathbf{x} \in \mathbb{R}^d$, without loss of generality, we define the fully-connected network f with depth $L \geq 2$ and width m:

$$f(\mathbf{x}; \boldsymbol{\theta}) = \mathbf{W}_L \sigma(\mathbf{W}_{L-1} \sigma(\mathbf{W}_{L-2} \ldots \sigma(\mathbf{W}_1 \mathbf{x}))) \tag{5.1}$$

where σ is the ReLU activation function, $\mathbf{W}_1 \in \mathbb{R}^{m \times d}$, $\mathbf{W}_l \in \mathbb{R}^{m \times m}$, for $2 \leq l \leq L-1$, $\mathbf{W}^L \in \mathbb{R}^{1 \times m}$, and $\boldsymbol{\theta} = [\text{vec}(\mathbf{W}_1)^\intercal, \text{vec}(\mathbf{W}_2)^\intercal, \ldots, \text{vec}(\mathbf{W}_L)^\intercal]^\intercal$. In round t, given the collected data $\{\mathbf{x}_i, r_i\}_{i=1}^t$, the loss function is defined as:

$$\mathcal{L} = \frac{1}{2} \sum_{i=1}^{t} \left(f(\mathbf{x}_i; \boldsymbol{\theta}) - r_i \right)^2 . \tag{5.2}$$

Initialization. For any $l \in [L]$, each entry of $\mathbf{W}_l$ is drawn from the normal distribution $\mathcal{N}(0, \frac{2}{m})$. Note that EE-Net at most has three networks f_1, f_2, f_3. We define them following the definition of f for brevity, although they may have different depth or width. Then, we have the following theorem for EE-Net. Recall that η_1, η_2 are the learning rates for f_1, f_2; K_1 is the number of iterations of gradient descent for f_1 in each round; and K_2 is the number of iterations for f_2.

Theorem 1. *Let f_1, f_2 follow the setting of f (Eq. (5.1)) with width m, m' respectively and same depth L. Let $\mathcal{L}_1, \mathcal{L}_2$ be loss function defined in Algorithm 1. Set f_3 as $f_3 = f_1 + f_2$. Given two constants ϵ_1, ϵ_2, $0 < \epsilon_1, \epsilon_2 < 1$, assume*

$$\begin{aligned} & m \geq \textit{poly}(T, n, L, \log(1/\delta) \cdot d \cdot e^{\sqrt{\log 1/\delta}}), \ m' \geq \Omega(m^2 L) \\ & \eta_1 = \Theta\left(\frac{d\delta}{\textit{poly}(T, n, L) \cdot m} \right), \ \ \eta_2 = \Theta\left(\frac{\mathcal{O}(m^2 L)\delta}{\textit{poly}(T, n, L) \cdot m'} \right) \\ & K_1 = \Theta\left(\frac{\textit{poly}(T, n, L)}{\delta^2} \cdot \log\left((\epsilon_1/2)^{-1} \right) \right), \ \ K_2 = \Theta\left(\frac{\textit{poly}(T, n, L)}{\delta^2} \cdot \log\left(\epsilon_2^{-1} \right) \right), \end{aligned} \tag{5.3}$$

then with probability at least $1 - \delta$, the expected cumulative regret of EE-Net in T rounds satisfies

$$\mathbf{R}_T \leq \mathcal{O}\left((2\sqrt{T} - 1)\sqrt{2\epsilon_2} \right) + \mathcal{O}\left((\xi_2 + \epsilon_1)(2\sqrt{T} - 1)\sqrt{2\log(\mathcal{O}(Tn)/\delta)} \right). \tag{5.4}$$

where

$$\xi_2 = \mathcal{O}\left(\frac{T^4 nL\sqrt{\mathcal{O}(m^2L)}\log m'}{\delta\sqrt{m'}}\right) + \mathcal{O}\left(\frac{T^5 nL^2\sqrt{\mathcal{O}(m^2L)}\log^{11/6} m'}{\delta m'^{1/6}}\right) < 1. \tag{5.5}$$

When $\epsilon_2 \leq 1/T$*, we have* $\mathbf{R}_T \leq \mathcal{O}(1) + \mathcal{O}\left((\xi_2+\epsilon_1)(2\sqrt{T}-1)\sqrt{2\log(\mathcal{O}(Tn)/\delta)}\right)$.

Comparison with NeuralUCB/TS. Under the same assumptions in over-parameterized neural networks, the regret bounds complexity of NeuralUCB (Zhou et al., 2020) and NeuralTS (Zhang et al., 2020) both are

$$\mathbf{R}_T \leq \mathcal{O}\left(\sqrt{\tilde{d}T\log T}\right)\cdot\mathcal{O}\left(\sqrt{\tilde{d}\log T}\right), \text{ and } \tilde{d} = \frac{\log\det(\mathbf{I}+\mathbf{H}/\lambda)}{\log(1+Tn/\lambda)}$$

where $\mathbf{H}$ is the neural tangent kernel matrix (NTK) (Jacot et al., 2018; Arora et al., 2019) and λ is a regularization parameter.

Remark 5.1. It is easy to observe that the regret bound of EE-Net is tighter than NeuralUCB/TS, which roughly improves by a multiplicative factor of $\sqrt{\log T}$, because our proof of EE-Net is directly built on recent advances in convergence theory (Allen-Zhu et al., 2019) and generalization bound (Cao and Gu, 2019) of over-parameterized neural networks. Instead, the analysis for NeuralUCB/TS contains three parts of approximation error by calculating the distances between the expected reward and ridge regression, ridge regression and NTK, and NTK and network function. □

Remark 5.2. The regret bound of EE-Net does not have the effective dimension $\tilde{d}$ which is a considerable multiplicative factor when the input dimension d is extremely large. The effective dimension is first introduced by Valko et al. (2013) to measure the underlying dimensions of observed context. Although $\tilde{d}$ can be upper bounded to some dimensional subspace of reproducing kernel hilbert space (RKHS) by NTK (Zhang et al., 2020), their regret bound still has the multiplicative factor $\tilde{d}$, but EE-Net does not have this factor. □

The proof of Theorem 1 is in **Appendix B**. Moreover, we provide the regret analysis of greedy approach with the only exploitation network f_1, i.e. $f_3 = f_1$, showing that EE-Net theoretically outperforms the greedy approach (see details in **Appendix C**).

6 Experiments

In this section, we evaluate EE-Net on four real-world datasets comparing with strong state-of-the-art baselines. We first present the setup of experiments, then show regret comparison and report ablation study. For the reproducibility, **all the code has been released anonymously**[1].

MNIST dataset. MNIST is a well-known image dataset (LeCun et al., 1998) for the 10-class classification problem. Following the evaluation setting of existing works (Valko et al., 2013; Zhou et al., 2020; Zhang et al., 2020), we transform this classification problem into bandit problem. Consider an image $\mathbf{x} \in \mathbb{R}^d$, we aim to classify it from 10 classes. First, in each round, the image $\mathbf{x}$ is transformed into 10 arms and presented to the learner, represented by 10 vectors in sequence $\mathbf{x}_1 = (\mathbf{x}, \mathbf{0}, \ldots, \mathbf{0}), \mathbf{x}_2 = (\mathbf{0}, \mathbf{x}, \ldots, \mathbf{0}), \ldots, \mathbf{x}_{10} = (\mathbf{0}, \mathbf{0}, \ldots, \mathbf{x}) \in \mathbb{R}^{10d}$. The reward is defined as 1 if the index of selected arm matches the index of $\mathbf{x}$'s ground-truth class; Otherwise, the reward is 0.

Yelp[2] **and Movielens (Harper and Konstan, 2015) datasets**. Yelp is a dataset released in the Yelp dataset challenge, which consists of 4.7 million rating entries for 1.57×10^5 restaurants by 1.18 million users. MovieLens is a dataset consisting of 25 million ratings between 1.6×10^5 users and 6×10^4 movies. We build the rating matrix by choosing the top 2000 users and top 10000 restaurants(movies) and use singular-value decomposition (SVD) to extract a 10-dimension feature vector for each user and restaurant(movie). In these two datasets, the bandit algorithm is to choose the restaurants(movies) with bad ratings. We generate the reward by using the restaurant(movie)'s gained stars scored by the users. In each rating record, if the user scores a restaurant(movie) less than 2 stars (5 stars totally), its reward is 1; Otherwise, its reward is 0. In each round, we set 10 arms as follows:

[1] https://anonymous.4open.science/r/iclr_2022-B085/README.md
[2] https://www.yelp.com/dataset

we randomly choose one with reward 1 and randomly pick the other 9 restaurants(movies) with 0 rewards; then, the representation of each arm is the concatenation of corresponding user feature vector and restaurant(movie) feature vector.

Disin (Ahmed et al., 2018) dataset. Disin is a fake news dataset on kaggle[3] including 12600 fake news articles and 12600 truthful news articles, where each article is represented by the text. To transform the text into vectors, we use the approach (Fu and He, 2021) to represent each article by a 300-dimension vector. Similarly, we form a 10-arm pool in each round, where 9 real news and 1 fake news are randomly selected. If the fake news is selected, the reward is 1; Otherwise, the reward is 0.

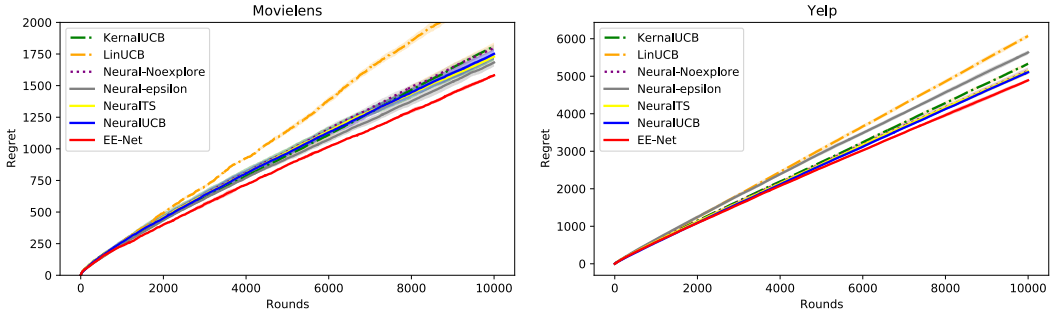

Figure 1: Regret comparison on Movielens and Yelp (mean of 10 runs with standard deviation (shadow)). With the same exploitation network f_1, EE-Net outperforms all baselines.

Baselines. To comprehensively evaluate EE-Net, we choose 4 neural-based bandit algorithms, one linear and one kernelized bandit algorithms.

1. LinUCB (Li et al., 2010) explicitly assumes the reward is a linear function of arm vector and unknown user parameter and then applies the ridge regression and un upper confidence bound to determine selected arm.
2. KernelUCB (Valko et al., 2013) adopts a predefined kernel matrix on the reward space combined with a UCB-based exploration strategy.
3. Neural-NoExplore only uses the exploitation network f_1 and selects an arm by the greedy strategy $\mathbf{x}_t = \arg\max_{i \in [n]} f_1(\mathbf{x}_{t,i}; \boldsymbol{\theta}^1)$.
4. Neural-Epsilon adapts the epsilon-greedy exploration strategy on exploitation network f_1. I.e., with probability $1 - \epsilon$, the arm is selected by $\mathbf{x}_t = \arg\max_{i \in [n]} f_1(\mathbf{x}_{t,i}; \boldsymbol{\theta}^1)$ and with probability ϵ, the arm is chosen randomly.
5. NeuralUCB (Zhou et al., 2020) uses the exploitation network f_1 to learn the reward function coming with an UCB-based exploration strategy.
6. NeuralTS (Zhang et al., 2020) adopts the exploitation network f_1 to learn the reward function coming with an Thompson Sampling exploration strategy.

Note that we do not report results of LinTS and KernelTS in experiments, because of the limited space in figures, but LinTS and KernelTS have been significantly outperformed by NeuralTS (Zhang et al., 2020).

Setup for EE-Net. To compare fairly, for all the neural-based methods including EE-Net, the exploitation network f_1 is built by a 2-layer fully-connected network with 100 width. For the exploration network f_2, we use a 2-layer fully-connected network with 100 width as well. For the decision maker f_3, by comprehensively evaluate both linear and nonlinear functions, we found that the most effective approach is combining them together, which we call " *hybrid decision maker*". In detail, for rounds $t \leq 500$, f_3 is set as $f_3 = f_2 + f_1$, and for $t > 500$, f_3 is set as a neural network with two 20-width fully-connected layers. Setting f_3 in this way is because the linear decision maker can maintain stable performance in each running (robustness) and the non-linear decision maker lacks the stability while can further improve the performance (see details in **Appendix A**). The hybrid decision maker can combine these two advantages together. For all the neural networks, we use the Adam optimizer (Kingma and Ba, 2014) and learning rate is set as 0.0001.

[3]https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset

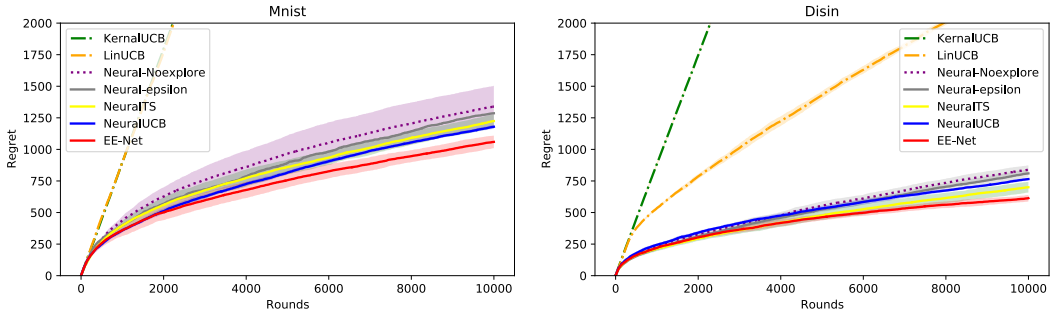

Figure 2: Regret comparison on Mnist and Disin (mean of 10 runs with standard deviation (shadow)). With the same exploitation network f_1, EE-Net outperforms all baselines.

6.1 Regret Comparison

Configurations. For LinUCB, following (Li et al., 2010), we do a grid search for the exploration constant α over $(\mathbf{0.01}, 0.1, 1)$ which is to tune the scale of UCB. For KernelUCB (Valko et al., 2013), we use the radial basis function kernel and stop adding contexts after 1000 rounds, following (Valko et al., 2013; Zhou et al., 2020). For the regularization parameter λ and exploration parameter ν in KernelUCB, we do the grid search for λ over $(\mathbf{0.1}, 1, 10)$ and for ν over $(0.01, \mathbf{0.1}, 1)$. For NeuralUCB and NeuralTS, following setting of (Zhou et al., 2020; Zhang et al., 2020), we use the exploiation network f_1 and conduct the grid search for the exploration parameter ν over $(\mathbf{0.001}, 0.01, 0.1, 1)$ and for the regularization parameter λ over $(0.01, \mathbf{0.1}, 1)$. For NeuralEpsilon, we use the same neural network f_1 and do the grid search for the exploration probability ϵ over $(\mathbf{0.01}, 0.1, 0.2)$. Neural-Noexplore uses the same neural network f_1 as well. For the neural bandits NeuralUCB/TS, following their setting, as they have expensive computation cost to store and compute the whole gradient matrix, we use a diagonal matrix to make approximation. For all grid-searched parameters, we choose the best of them for the comparison and report the averaged results of 10 runs for all methods.

Results. Figure 1 and Figure 2 show the regret comparison on these four datasets. EE-Net consistently outperforms all baselines across all datasets. For LinUCB and KernelUCN, the simple linear reward function or predefined kernel cannot properly formulate ground-truth reward function existed in real-world datasets. In particular, on Mnist and Disin datasets, the correlations between rewards and arm feature vectors are not linear or some simple mappings. Thus, LinUCB and KernelUCB barely exploit the past collected data samples and fail to select correct arms. For neural-based bandit algorithms, as Neural-Noexplore does not have exploration portion, its rates of collecting new samples and learning new knowledge are unstable and usually delayed. Therefore, Neural-Noexplore usually is inferior to the methods with exploration. The exploration probability of Neural-Epsilon is fixed and difficult to be adjustable. Thus it is usually hard to make effective exploration. To make exploration, NeuralUCB statistically calculates a gradient-based upper confidence bound and NeuralTS draws each arm's predicted reward from a normal distribution where the standard deviation is computed by gradient. However, the confidence bound or standard deviation they calculated only consider the worst cases and thus may not be able represent the actual potential of each arm. Instead, EE-Net uses a neural network f_2 to learn each arm's potential by neural network's powerful representation ability. Therefore, EE-Net can outperform these two state-of-the-art bandit algorithms. Note that NeuralUCB/TS does need two parameters to tune UCB/TS according to different scenarios while EE-Net only needs to set up a neural network and automatically learns it.

Ablation Study. In **Appendix A**, we conduct ablation study regarding the label function y of f_2 and the different setting of f_3.

7 Conclusion

In this paper, we propose a novel exploration strategy, EE-Net. In addition to a neural network that exploits collected data in past rounds , EE-Net has another neural network to learn the potential gain compared to current estimation for exploration. Then, a decision maker is built to make selections to further trade off between exploitation and exploration. We demonstrate that EE-Net outperforms NeuralUCB and NeuralTS both theoretically and empirically, becoming the new state-of-the-art exploration policy.

REFERENCES

Y. Abbasi-Yadkori, D. Pál, and C. Szepesvári. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*, pages 2312–2320, 2011.

M. Abeille and A. Lazaric. Linear thompson sampling revisited. In *Artificial Intelligence and Statistics*, pages 176–184. PMLR, 2017.

S. Agrawal and N. Goyal. Thompson sampling for contextual bandits with linear payoffs. In *International Conference on Machine Learning*, pages 127–135. PMLR, 2013.

H. Ahmed, I. Traore, and S. Saad. Detecting opinion spams and fake news using text classification. *Security and Privacy*, 1(1):e9, 2018.

Z. Allen-Zhu, Y. Li, and Z. Song. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, pages 242–252. PMLR, 2019.

S. Arora, S. S. Du, W. Hu, Z. Li, R. R. Salakhutdinov, and R. Wang. On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems*, pages 8141–8150, 2019.

P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

Y. Ban and J. He. Convolutional neural bandit: Provable algorithm for visual-aware advertising. *arXiv preprint arXiv:2107.07438*, 2021a.

Y. Ban and J. He. Local clustering in contextual multi-armed bandits. In *Proceedings of the Web Conference 2021*, pages 2335–2346, 2021b.

Y. Ban, J. He, and C. B. Cook. Multi-facet contextual bandits: A neural network perspective. In *The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, pages 35–45, 2021.

S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvári. X-armed bandits. *Journal of Machine Learning Research*, 12(5), 2011.

Y. Cao and Q. Gu. Generalization bounds of stochastic gradient descent for wide and deep neural networks. *Advances in Neural Information Processing Systems*, 32:10836–10846, 2019.

N. Cesa-Bianchi, A. Conconi, and C. Gentile. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057, 2004.

E. Chlebus. An approximate formula for a partial sum of the divergent p-series. *Applied Mathematics Letters*, 22(5):732–737, 2009.

W. Chu, L. Li, L. Reyzin, and R. Schapire. Contextual bandits with linear payoff functions. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 208–214, 2011.

V. Dani, T. P. Hayes, and S. M. Kakade. Stochastic linear optimization under bandit feedback. 2008.

S. Du, J. Lee, H. Li, L. Wang, and X. Zhai. Gradient descent finds global minima of deep neural networks. In *International Conference on Machine Learning*, pages 1675–1685. PMLR, 2019.

S. Filippi, O. Cappe, A. Garivier, and C. Szepesvári. Parametric bandits: The generalized linear case. In *Advances in Neural Information Processing Systems*, pages 586–594, 2010.

D. Fu and J. He. SDG: A simplified and dynamic graph neural network. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, pages 2273–2277. ACM, 2021. doi: 10.1145/3404835.3463059. URL https://doi.org/10.1145/3404835.3463059.

F. M. Harper and J. A. Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

J. Langford and T. Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in neural information processing systems*, pages 817–824, 2008.

T. Lattimore and C. Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

Y. LeCun, Y. Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670, 2010.

S. Li, A. Karatzoglou, and C. Gentile. Collaborative filtering bandits. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 539–548, 2016.

X. Lu and B. Van Roy. Ensemble sampling. *arXiv preprint arXiv:1705.07347*, 2017.

C. Riquelme, G. Tucker, and J. Snoek. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. *arXiv preprint arXiv:1802.09127*, 2018.

S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

M. Valko, N. Korda, R. Munos, I. Flaounas, and N. Cristianini. Finite-time analysis of kernelised contextual bandits. *arXiv preprint arXiv:1309.6869*, 2013.

L. Van Der Maaten, E. Postma, J. Van den Herik, et al. Dimensionality reduction: a comparative. *J Mach Learn Res*, 10(66-71):13, 2009.

Q. Wu, H. Wang, Q. Gu, and H. Wang. Contextual bandits in a collaborative environment. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 529–538, 2016.

P. Xu, Z. Wen, H. Zhao, and Q. Gu. Neural contextual bandits with deep representation and shallow exploration. *arXiv preprint arXiv:2012.01780*, 2020.

W. Zhang, D. Zhou, L. Li, and Q. Gu. Neural thompson sampling. *arXiv preprint arXiv:2010.00827*, 2020.

D. Zhou, L. Li, and Q. Gu. Neural contextual bandits with ucb-based exploration. In *International Conference on Machine Learning*, pages 11492–11502. PMLR, 2020.

A ABLATION STUDY

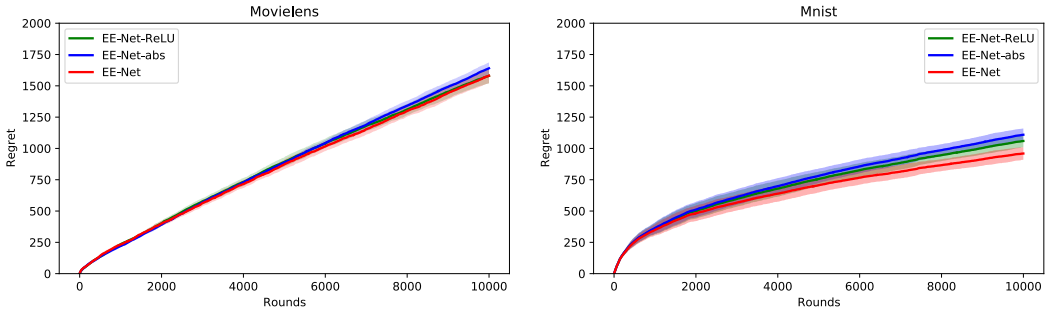

Figure 3: Ablation study on label function y for f_2. EE-Net denotes $y_1 = r - f_1$, EE-Net-abs denotes $y_2 = |r - f_1|$, and EE-Net-ReLU denotes $y_3 = \text{ReLU}(r - f_1)$. EE-Net shows the best performance on these two datasets.

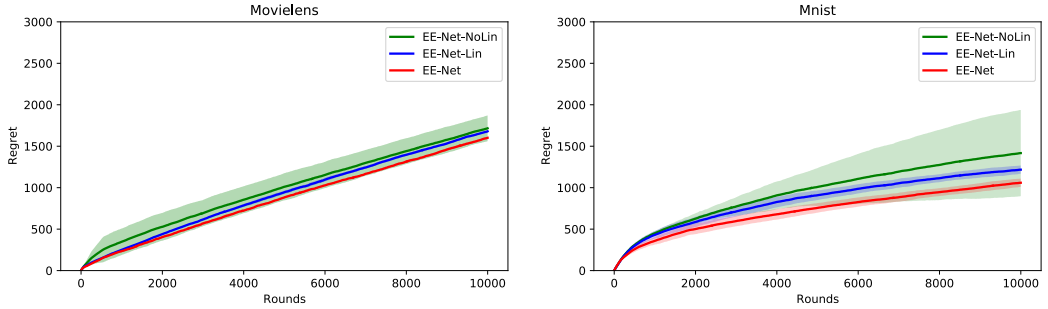

Figure 4: Ablation study on decision maker f_3. EE-Net-Lin denotes $f_3 = f_1 + f_2$, EE-Net-NoLin denote the nonlinear one where f_3 is a neural network (2 layer, width 20), EE-Net denotes the hybrid one where $f_3 = f_1 + f_2$ if $t \leq 500$ and f_3 is the neural network if $t > 500$. EE-Net has the most stable and best performance.

In this section, we conduct ablation study regarding the label function y for exploration network f_2 and seting of decision maker f_3 on two representative datasets Movielens and Mnist.

Label function y. In this paper, we use $y_1 = r - f_1$ to measure the potential gain of an arm, as the label of f_2. Moreover, we provide other two intuitive form $y_2 = |r - f_1|$ and $y_3 = ReLU(r - f_1)$. Figure 3 shows the regret with different y, where "EE-Net" denotes our method with default y_1, "EE-Net-abs" represents the one with y_2 and "EE-Net-ReLU" is with y_3. On Movielens and Mnist datasets, EE-Net slightly outperforms EE-Net-abs and EE-Net-ReLU. In fact, y_1 can effectively represent the positive potential gain and negative potential gain, such that f_2 intends to score the arm with positive gain higher and score the arm with negative gain lower. However, y_2 treats the positive/negative potential gain evenly, weakening the discriminative ability. y_3 can recognize the positive gain while neglecting the difference of negative gain. Therefore, y_1 usually is the most effective one for empirical performance.

Setting of f_3. f_3 can be set as linear function or non-linear function. In the experiment, we test the simple linear function $f_3 = f_1 + f_2$, denoted by "EE-Net-Lin", and a non-linear function represented by a 2-layer 20-width fully-connected neural network, denoted by "EE-Net-NoLin". For the default hybrid setting, denoted by "EE-Net", when rounds $t \leq 500$, $f_3 = f_1 + f_2$; Otherwise, f_3 is the neural network. Figure 4 reports the regret with these three different modes. EE-Net achieves the best performance with small standard deviation. In contrast, EE-Net-NoLin obtains the worst performance and largest standard deviation. However, notice that EE-Net-NoLin can achieve the best performance in certain running (the green shallow) but it is erratic. Because in the begin phase, without enough training samples, EE-Net-NoLin strongly relies on the quality of collected samples. With appropriate training samples, gradient descent can lead f_3 to global optimum. On the other hand, with misleading

training samples, gradient descent can deviate f_3 from global optimum. Therefore, EE-Net-NoLin shows very unstable performance. In contrast, EE-Net-Lin is inspired by the UCB strategy, i.e., the exploitation plus the exploration, exhibiting stable performance. To combine their advantages together, we propose the hybrid approach, EE-Net, achieving the best performance with strong stability.

B Proof of Theorem 1

In this section, we provide the proof of Theorem 1 and related lemmas.

Theorem 1. Let f_1, f_2 follow the setting of f (Eq. (5.1)) with width m, m' respectively and same depth L. Let $\mathcal{L}_1, \mathcal{L}_2$ be loss functions defined in Algorithm 1. Set f_3 as $f_3 = f_1 + f_2$. Given two constants ϵ_1, ϵ_2, $0 < \epsilon_1, \epsilon_2 < 1$, suppose

$$\begin{aligned}
&m \geq \text{poly}(T, n, L, \log(1/\delta) \cdot d \cdot e^{\sqrt{\log 1/\delta}}),\ m' \geq \Omega(m^2 L) \\
&\eta_1 = \Theta\left(\frac{d\delta}{\text{poly}(T,n,L)\cdot m}\right),\ \ \eta_2 = \Theta\left(\frac{\mathcal{O}(m^2L)\delta}{\text{poly}(T,n,L)\cdot m'}\right) \\
&K_1 = \Theta\left(\frac{\text{poly}(T,n,L)}{\delta^2}\cdot\log\left((\epsilon_1/2)^{-1}\right)\right),\ \ K_2 = \Theta\left(\frac{\text{poly}(T,n,L)}{\delta^2}\cdot\log\left(\epsilon_2^{-1}\right)\right),
\end{aligned} \tag{B.1}$$

then with probability at least $1-\delta$, the expected cumulative regret of EE-Net in T rounds satisfies

$$\mathbf{R}_T \leq \mathcal{O}\left((2\sqrt{T}-1)\sqrt{2\epsilon_2}\right) + \mathcal{O}\left((\xi_2+\epsilon_1)(2\sqrt{T}-1)\sqrt{2\log(\mathcal{O}(Tn)/\delta)}\right).$$

where

$$\xi_2 = \mathcal{O}\left(\frac{T^4 nL\sqrt{\mathcal{O}(m^2L)}\log m'}{\delta\sqrt{m'}}\right) + \mathcal{O}\left(\frac{T^5 nL^2\sqrt{\mathcal{O}(m^2L)}\log^{11/6} m'}{\delta m'^{1/6}}\right) < 1.$$

When $\epsilon_2 \leq 1/T$, we have

$$\mathbf{R}_T \leq \mathcal{O}(1) + \mathcal{O}\left((\xi_2+\epsilon_1)(2\sqrt{T}-1)\sqrt{2\log(\mathcal{O}(Tn)/\delta)}\right).$$

Proof. In round t, given the n arms $(\mathbf{x}_{t,1}, \dots \mathbf{x}_{t,n})$, let $H_t = (h(\mathbf{x}_{t,i}), \dots, h(\mathbf{x}_{t,n}))$ denote their expected rewards. For brevity, for the selected arm $\mathbf{x}_t$ in round t, let $h(\mathbf{x}_t)$ be its expected reward and $\mathbf{x}_t^*$ be the optimal arm in round t.

Then, the expected regret of round t is computed by

$$\begin{aligned}
\mathbf{R}_t &= h(\mathbf{x}_t^*) - h(\mathbf{x}_t) \\
&= h(\mathbf{x}_t^*) - f_3(\mathbf{x}_t) + f_3(\mathbf{x}_t) - h(\mathbf{x}_t) \\
&\leq \underbrace{h(\mathbf{x}_t^*) - f_3(\mathbf{x}_t^*) + f_3(\mathbf{x}_t) - f_3(\mathbf{x}_t)}_{I_1} + f_3(\mathbf{x}_t) - h(\mathbf{x}_t) \\
&= h(\mathbf{x}_t^*) - f_3(\mathbf{x}_t^*) + f_3(\mathbf{x}_t) - h(\mathbf{x}_t) \\
&\leq \max_{i\in[n]} \left(\mathbb{E}[(r_{t,i} - f_3(\mathbf{x}_{t,i}))\,|\mathbf{x}_{t,i}]\right) + \mathbb{E}[(f_3(\mathbf{x}_t) - r_t)\,|\mathbf{x}_t] \\
&= \max_{i\in[n]} \left(\mathbb{E}\left[\left(r_{t,i} - \left(f_1(\mathbf{x}_{t,i};\boldsymbol{\theta}_{t-1}^1) + f_2(\nabla_{\boldsymbol{\theta}_1} f_1/c\sqrt{mL};\boldsymbol{\theta}_{t-1}^2)\right)\right)|\mathbf{x}_{t,i}\right]\right) \\
&\quad + \mathbb{E}\left[\left(f_1(\mathbf{x}_i;\boldsymbol{\theta}_{t-1}^1) + f_2(\nabla_{\boldsymbol{\theta}_1} f_1/c\sqrt{mL};\boldsymbol{\theta}_{t-1}^2) - r_t\right)|\mathbf{x}_t\right] \\
&\leq \max_{i\in[n]} \left(\mathbb{E}[|f_1(\mathbf{x}_{t,i};\boldsymbol{\theta}_{t-1}^1) + f_2(\nabla_{\boldsymbol{\theta}_1} f_1/c\sqrt{mL};\boldsymbol{\theta}_{t-1}^2) - r_{t,i}||\mathbf{x}_{t,i}]\right) \\
&\quad + \mathbb{E}[|f_1(\mathbf{x}_i;\boldsymbol{\theta}_{t-1}^1) + f_2(\nabla_{\boldsymbol{\theta}_1} f_1/c\sqrt{mL};\boldsymbol{\theta}_{t-1}^2) - r_t||\mathbf{x}_t]
\end{aligned} \tag{B.2}$$

where I_1 is because $f_3(\mathbf{x}_t) = \max_{i\in[n]} f_3(\mathbf{x}_{t,i})$ and $f_3(\mathbf{x}_t) - f_3(\mathbf{x}_t^*) \geq 0$.

Then, based on Lemma B.1, with probability at least $1-\delta$, we have

$$\mathbf{R}_t \leq 2\left(\sqrt{\frac{2\epsilon_2}{t}} + (\xi_2+\epsilon_1)\sqrt{\frac{2\log \mathcal{O}(Tn)/\delta)}{t}}\right) \tag{B.3}$$

Then expected regret of T rounds is computed by

$$\begin{aligned}\mathbf{R}_T &= \sum_{t=1}^{T}\mathbf{R}_t \\ &\leq 2\sum_{t=1}^{T}\left(\sqrt{\frac{2\epsilon_2}{t}} + (\xi_2+\epsilon_1)\sqrt{\frac{2\log(\mathcal{O}(Tn)/\delta)}{t}}\right) \\ &\leq \underbrace{2(2\sqrt{T}-1)\sqrt{2\epsilon_2} + 2(\xi_2+\epsilon_1)(2\sqrt{T}-1)\sqrt{2\log(\mathcal{O}(Tn)/\delta)}}_{I_2}\end{aligned} \tag{B.4}$$

where I_2 is because $\sum_{t=1}^{T}\frac{1}{\sqrt{t}} \leq \int_1^T \frac{1}{\sqrt{t}}\,dx + 1 = 2\sqrt{T}-1$ (Chlebus, 2009).

When $K_2 = \Theta(\frac{\text{poly}(T,n,L)}{\delta^2}\cdot\log T)$, according to Lemma B.3 (1), with probability at least $1-\delta$, we have $\epsilon_2 = 1/T$. Therefore, we have

$$\mathbf{R}_T \leq \mathcal{O}(1) + 2(\xi_1+\epsilon_1)(2\sqrt{T}-1)\sqrt{2\log(\mathcal{O}(Tn)/\delta)}, \tag{B.5}$$

As the choice of m, we have $\xi_1 \leq 1$. Therefore, we have

$$\mathbf{R}_T \leq \mathcal{O}(1) + \mathcal{O}\left((1+\epsilon_1)(2\sqrt{T}-1)\sqrt{2\log(\mathcal{O}(Tn)/\delta)}\right). \tag{B.6}$$

The proof is completed. □

Lemma B.1. *Given $0<\epsilon_1,\epsilon_2<1$, suppose m,η,K_1,K_2 satisfy the conditions in Eq. (C.1). Then, with probability at least $1-\delta$, for any $t\in[T], i\in[n]$, it holds uniformly that*

$$\mathbb{E}_{(\mathbf{x}_{t,i},r_{t,i})\sim\mathcal{D}}[|f_2(\nabla_{\boldsymbol{\theta}_1}f_1/c_1\sqrt{mL};\boldsymbol{\theta}_t^2) - \big(r_{t,i} - f_1(\mathbf{x}_{t,i};\boldsymbol{\theta}_t^1)\big)\,|] \leq \sqrt{\frac{2\epsilon_2}{t}} + (\xi_2+\epsilon_1)\sqrt{\frac{2\log(\mathcal{O}(Tn)/\delta)}{t}}.$$

Proof. Using the Lemma B.4 (2) and Lemma B.3 (2), for $t\in[T]$, with probability at least $1-\delta$, we have $\|\nabla_{\boldsymbol{\theta}_t^1}f(\mathbf{x};\boldsymbol{\theta}_t^1)\|_2 \leq \mathcal{O}(\sqrt{mL})$ for any $\|\mathbf{x}\|\leq 1$. Now, consider the exploration network f_2. There must exist a constant c_1 such that

$$\left\|\frac{\nabla_{\boldsymbol{\theta}^1}f_1(\mathbf{x}_i;\boldsymbol{\theta}_t^1)}{c_1\sqrt{mL}}\right\|_2 \leq 1. \tag{B.7}$$

Consider $\nabla_{\boldsymbol{\theta}_t^1}f(\mathbf{x};\boldsymbol{\theta}_t^1)\in\mathbb{R}^p$ where $p = m + md + m^2(L-1)$. Then applying Lemma B.2 and union bound on f_2, replacing d by p, with probability at least $1-2\delta$, for any $t\in[T]$, we have

$$\left|f_2\left(\frac{\nabla_{\boldsymbol{\theta}^2}f_1}{c_1\sqrt{mL}};\boldsymbol{\theta}_t^2\right)\right| \leq \xi_2. \tag{B.8}$$

Then, applying the Lemma B.3 (1) on f_1, with probability at least $1-\delta$, for any $i\in[t]$, we have

$$|r_i - f_1(\mathbf{x}_i;\boldsymbol{\theta}_t^1))| \leq 2\cdot\frac{1}{2}\sum_{i=1}^{t}(r_i - f_1(\mathbf{x}_i;\boldsymbol{\theta}_t^1)^2 \leq \epsilon_1. \tag{B.9}$$

Applying union bound, with probability at least $1-3\delta$, we have

$$\left|f_2\left(\frac{\nabla_{\boldsymbol{\theta}^1}f_1}{c_1\sqrt{mL}};\boldsymbol{\theta}_t^2\right) - (r - f_1(\mathbf{x};\boldsymbol{\theta}_t^1))\right| \leq \xi_2+\epsilon_1. \tag{B.10}$$

Then, using the Lemma B.3 (1) and applying union bound, with probability at least $1-4\delta$, we have

$$\begin{aligned}
&\frac{1}{t}\sum_{i=1}^{t}\left|f_2\left(\frac{\nabla_{\boldsymbol{\theta}^1} f_1}{c_1\sqrt{mL}};\boldsymbol{\theta}_t^2\right)-\big(r_i-f_1(\mathbf{x}_i;\boldsymbol{\theta}_t^1)\big)\right| \\
\leq&\frac{1}{t}\sqrt{t}\sqrt{\sum_{i=1}^{t}\left(f_2\left(\frac{\nabla_{\boldsymbol{\theta}^1} f_1}{c_1\sqrt{mL}};\boldsymbol{\theta}_t^2\right)-\big(r_i-f_1(\mathbf{x}_i;\boldsymbol{\theta}_t^1)\big)\right)^2} \\
=&\sqrt{\frac{2\mathcal{L}_2}{t}}\leq\sqrt{\frac{2\epsilon_2}{t}}.
\end{aligned} \tag{B.11}$$

Set $V_i=\mathbb{E}[|f_2(\frac{\nabla_{\boldsymbol{\theta}^1} f_1}{c_1\sqrt{mL}};\boldsymbol{\theta}_t^2)-\big(r_i-f_1(\mathbf{x}_i;\boldsymbol{\theta}_t^1)\big)\,|]-|f_2(\frac{\nabla_{\boldsymbol{\theta}^1} f_1}{c_1\sqrt{mL}};\boldsymbol{\theta}_t^2)-\big(r_i-f_1(\mathbf{x}_i;\boldsymbol{\theta}_t^1)\big)\,|$. We have

$$\begin{aligned}
\frac{1}{t}\sum_{i=1}^{t}V_i=&\frac{1}{t}\sum_{i=1}^{t}\mathbb{E}\left[\left|f_2\left(\frac{\nabla_{\boldsymbol{\theta}^1} f_1}{c_1\sqrt{mL}};\boldsymbol{\theta}_t^2\right)-\big(r_i-f_1(\mathbf{x}_i;\boldsymbol{\theta}_t^1)\big)\right|\right] \\
&-\frac{1}{t}\sum_{i=1}^{t}\left|f_2\left(\frac{\nabla_{\boldsymbol{\theta}^1} f_1}{c_1\sqrt{mL}};\boldsymbol{\theta}_t^2\right)-\big(r_i-f_1(\mathbf{x}_i;\boldsymbol{\theta}_t^1)\big)\right|.
\end{aligned}$$

Then, we have

$$\begin{aligned}
\mathbb{E}[V_i|\mathbf{F}_i]=&\mathbb{E}\left[\left|f_2\left(\frac{\nabla_{\boldsymbol{\theta}^1} f_1}{c_1\sqrt{mL}};\boldsymbol{\theta}_t^2\right)-\big(r_i-f_1(\mathbf{x}_i;\boldsymbol{\theta}_t^1)\big)\right|\right] \\
&-\mathbb{E}\left[\left|f_2\left(\frac{\nabla_{\boldsymbol{\theta}^1} f_1}{c_1\sqrt{mL}};\boldsymbol{\theta}_t^2\right)-\big(r_i-f_1(\mathbf{x}_i;\boldsymbol{\theta}_t^1)\big)\right|\right]=0
\end{aligned}$$

where $\mathbf{F}_i$ denotes the σ-algebra generated by $(f_1(\boldsymbol{\theta}_1^1),f_2(\boldsymbol{\theta}_1^2)),\ldots,(f_1(\boldsymbol{\theta}_i^1),f_2(\boldsymbol{\theta}_i^2))$ and $f_1(\boldsymbol{\theta}_i^1),f_2(\boldsymbol{\theta}_i^2))$ are the hypothesis of neural networks f_1,f_2.

Then applying the Hoeffding-Azuma inequality to $V_1,\ldots,V_t$, with probability at least $1-5\delta$, we have

$$\begin{aligned}
&\mathbb{E}_{(\mathbf{x},r)\sim\mathcal{D}}\left[\left|f_2\left(\frac{\nabla_{\boldsymbol{\theta}^1} f_1}{c_1\sqrt{mL}};\boldsymbol{\theta}_t^2\right)-\big(r_i-f_1(\mathbf{x}_i;\boldsymbol{\theta}_t^1)\big)\right|\right] \\
=&\frac{1}{t}\sum_{i=1}^{t}\mathbb{E}_{(\mathbf{x}_i,r_i)\sim\mathcal{D}}\left[\left|f_2\left(\frac{\nabla_{\boldsymbol{\theta}^1} f_1}{c_1\sqrt{mL}};\boldsymbol{\theta}_t^2\right)-\big(r_i-f_1(\mathbf{x}_i;\boldsymbol{\theta}_t^1)\big)\right|\right] \\
\leq&\frac{1}{t}\sum_{i=1}^{t}\left|f_2\left(\frac{\nabla_{\boldsymbol{\theta}^1} f_1}{c_1\sqrt{mL}};\boldsymbol{\theta}_t^2\right)-\big(r_i-f_1(\mathbf{x}_i;\boldsymbol{\theta}_t^1)\big)\right|+(\xi_2+\epsilon_1)\sqrt{\frac{2\log(1/\delta)}{t}} \\
\leq&\sqrt{\frac{2\epsilon_2}{t}}+(\xi_2+\epsilon_1)\sqrt{\frac{2\log(1/\delta)}{t}}\,.
\end{aligned} \tag{B.12}$$

Then, applying union bound, with probability at least $1-\mathcal{O}(Tn)\delta$, Eq(B.12) always holds for any $t\in[T],i\in[n]$. Rescaling the δ completes the proof. □

Lemma B.2. *Let $f(\cdot;\boldsymbol{\theta}_t)$ follow the gradient descent of f_1 in Algorithm 1. Suppose m,η,K satisfy the conditions in Eq. (C.1). With probability at least $1-\delta$, for any $\mathbf{x}$ with $\|\mathbf{x}\|_2=1$ and $t\in[T]$, it holds that*

$$|f(\mathbf{x};\boldsymbol{\theta}_t)|\leq\mathcal{O}\left(\frac{t^4nL\sqrt{d}\log m}{\delta\sqrt{m}}\right)+\mathcal{O}\left(\frac{t^5nL^2\sqrt{d}\log^{11/6}m}{\delta m^{1/6}}\right).$$

Proof. Considering an inequality $|a-b| \leq c$, we have $|a| \leq |b| + c$. Let $\boldsymbol{\theta}_0$ be randomly initialized. Then applying Lemma B.4 (1), for any $\mathbf{x} \sim \mathcal{D}$, $\|\mathbf{x}\|_2 = 1$ and $\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_0\| \leq w$, we have

$$
\begin{aligned}
|f(\mathbf{x};\boldsymbol{\theta}_t)| &\leq |f(\mathbf{x};\boldsymbol{\theta}_0)| + |\langle \nabla_{\boldsymbol{\theta}_0} f(\mathbf{x}_i;\boldsymbol{\theta}'), \boldsymbol{\theta}_t - \boldsymbol{\theta}_0\rangle| + \mathcal{O}(L^2\sqrt{m\log(m)})\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_0\|_2 w^{1/3} \\
&\leq f(\mathbf{x};\boldsymbol{\theta}_0) + \underbrace{\|\nabla_{\boldsymbol{\theta}_0} f(\mathbf{x}_i;\boldsymbol{\theta}')\|_2 \|\boldsymbol{\theta}_t - \boldsymbol{\theta}_0\|_2}_{I_1} + \mathcal{O}(L^2\sqrt{m\log(m)})\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_0\|_2 w^{1/3} \\
&\leq \underbrace{\mathcal{O}(L)\cdot\mathcal{O}\left(\frac{t^3\sqrt{d}}{\delta\sqrt{m}}\log m\right)}_{I_2} + \underbrace{\mathcal{O}\left(L^2\sqrt{m\log(m)}\right)\cdot\mathcal{O}\left(\frac{t^3\sqrt{d}}{\delta\sqrt{m}}\log m\right)^{4/3}}_{I_3} \\
&= \mathcal{O}\left(\frac{t^3 L\sqrt{d}\log m}{\delta\sqrt{m}}\right) + \mathcal{O}\left(\frac{t^4 L^2\sqrt{d}\log^{11/6} m}{\delta m^{1/6}}\right)
\end{aligned}
\tag{B.13}
$$

where I_1 is an application of Cauchy–Schwarz inequality, I_2 is by initializing $f(\mathbf{x};\boldsymbol{\theta}_0) = 0$ and Lemma B.3 (3), and I_3 is due to Lemma B.3 (2). Then, apply the union bound and rescale δ. The proof is completed. □

Lemma B.3. *Suppose $m \geq \max\left(poly(n, L, \delta^{-1}\cdot d), \Omega(e^{\sqrt{\log 1/\delta}})\right)$, the learning rate $\eta = \Omega(\frac{\delta d}{poly(T,n,L)m})$, the number of iterations K satisfies the conditions in Eq. (C.1) , then with probability at least $1-\delta$, given a constant $0 < \epsilon < 1$, starting from random initialization,*

(1) (Theorem 1 in (Allen-Zhu et al., 2019)) The loss satisfies $\mathcal{L} \leq \epsilon$ (Eq. (5.2)) in $K = \Omega(\frac{poly(T,n,L)}{\delta^2}\cdot\log\epsilon^{-1})$ iterations,

(2) (Theorem 1 in (Allen-Zhu et al., 2019)) For any $k \in [K]$, it holds uniformly that $\|\boldsymbol{\theta}_t^{(k)} - \boldsymbol{\theta}_t^{(0)}\|_2 \leq \mathcal{O}\left(\frac{t^3\sqrt{d}}{\delta\sqrt{m}}\log m\right)$,

(3) Following the initialization, it holds that

$$\|\nabla_{\boldsymbol{\theta}_0} f(\mathbf{x}_i;\boldsymbol{\theta}_0)\|_2 \leq \mathcal{O}(L), \quad |f(\mathbf{x}_i;\boldsymbol{\theta}_0)| \leq \mathcal{O}(1)$$

where $\boldsymbol{\theta}_t^{(k)}$ represents the parameters of f after $k \in [K]$ iterations if gradient descent in round t.

Proof. For (1) and (2), the only different setting from (Allen-Zhu et al., 2019) is that the initialization of last layer $\mathbf{W}_L \sim \mathcal{N}(0, \frac{2}{m})$ in this paper while $\mathbf{W}_L \sim \mathcal{N}(0, \frac{1}{d})$ in (Allen-Zhu et al., 2019). Because $m > d$, the upper bound in (Allen-Zhu et al., 2019) still holds for $\mathbf{W}_L$: with probability at least $1 - \exp(-\Omega(m/L))$, $\|\mathbf{W}_L\| \leq \sqrt{m/d}$. Therefore, it is easy to verify that (1) and (2) still hold for the initialization of this paper.

For (3), based on Lemma 7.1 in Allen-Zhu et al. (2019), we have $|f(\mathbf{x}_i;\boldsymbol{\theta}_0)| \leq \mathcal{O}(1)$. Denote by D the ReLU function. For any $l \in [L]$,

$$\|\nabla_{\boldsymbol{\theta}_0} f(\mathbf{x}_i;\boldsymbol{\theta}_0)\| \leq \|\mathbf{W}_L D\mathbf{W}_{L-1}\cdots D\mathbf{W}_{l+1}\|\cdot\|D\mathbf{W}_{l+1}\cdots\mathbf{x}\| \leq \mathcal{O}(\sqrt{L})$$

where the inequality is according to Lemma 7.2 in Allen-Zhu et al. (2019). Therefore, we have $\|\nabla_{\boldsymbol{\theta}_0} f(\mathbf{x}_i;\boldsymbol{\theta}_0)\| \leq \mathcal{O}(L)$. □

Lemma B.4 (Lemma 4.1, (Cao and Gu, 2019))**.** *Suppose $\mathcal{O}(m^{-3/2}L^{-3/2}[\log(TnL^2/\delta)]^{3/2}) \leq w \leq \mathcal{O}(L^{-6}[\log m]^{-3/2})$. Then, with probability at least $1-\delta$ over randomness of $\boldsymbol{\theta}_0$, for any $t \in [T]$, $\|\mathbf{x}\|_2 = 1$, and $\boldsymbol{\theta}, \boldsymbol{\theta}'$ satisfying $\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\| \leq w$ and $\|\boldsymbol{\theta}' - \boldsymbol{\theta}_0\| \leq w$, it holds uniformly that*

(1) $|f(\mathbf{x}_i;\boldsymbol{\theta}) - f(\mathbf{x}_i;\boldsymbol{\theta}') - \langle\nabla_{\boldsymbol{\theta}'} f(\mathbf{x}_i;\boldsymbol{\theta}'), \boldsymbol{\theta} - \boldsymbol{\theta}'\rangle| \leq \mathcal{O}(w^{1/3}L^2\sqrt{m\log(m)})\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2$

(2) $\|\nabla_{\boldsymbol{\theta}} f(\mathbf{x};\boldsymbol{\theta})\|_F \leq \mathcal{O}(\sqrt{mL})$.

C COMPARISON WITH SINGLE REWARD FUNCTION

In this section, we provide the regret analysis of pure-exploitation approach, denoted by EE-Net', i.e., the EE-Net with $f_3 = f_1$.

Theorem 2. *Let f_1 follow the setting of f (Eq. (5.1)) and $f_3 = f_1$. Given one constant ϵ_1, $0 < \epsilon_1 < 1$, suppose*

$$\begin{aligned} &m \geq \Omega(poly(T, n, L, \log(1/\delta) \cdot d \cdot e^{\sqrt{\log 1/\delta}}), \quad \eta_1 = \Theta(\frac{d\delta}{poly(T, n, L) \cdot m}); \\ &K_1 = \Theta(\frac{poly(T, n, L)}{\delta^2} \cdot \log(\epsilon_1)^{-1}), \end{aligned} \tag{C.1}$$

then with probability at least $1 - \delta$, the expected regret of EE-Net' in T rounds satisfies

$$\mathbf{R}'_T \leq \mathcal{O}\left((2\sqrt{T} - 1)\sqrt{2\epsilon_1}\right) + \mathcal{O}\left((\xi_1 + 1)(2\sqrt{T} - 1)\sqrt{2\log(\mathcal{O}(Tn)/\delta)}\right),$$

where

$$\xi_1 = \mathcal{O}\left(\frac{T^4 nL\sqrt{d}\log m}{\delta\sqrt{m}}\right) + \mathcal{O}\left(\frac{T^5 nL^2\sqrt{d}\log^{11/6} m}{\delta m^{1/6}}\right) < 1.$$

When $\epsilon_1 \leq 1/T$, we have

$$\mathbf{R}'_T \leq \mathcal{O}(1) + \mathcal{O}\left((\xi_1 + 1)(2\sqrt{T} - 1)\sqrt{2\log(\mathcal{O}(Tn)/\delta)}\right).$$

Comparison with Theorem 1. With the regret upper bounds of EE-Net and EE-Net', we have

$$\begin{aligned} &\text{(EE-Net)} \ \ \mathbf{R}_T \leq \mathcal{O}(1) + \mathcal{O}\left((\xi_2 + \epsilon_1)(2\sqrt{T} - 1)\sqrt{2\log(\mathcal{O}(Tn)/\delta)}\right) \\ &\text{(EE-Net')} \ \ \mathbf{R}'_T \leq \mathcal{O}(1) + \mathcal{O}\left((\xi_1 + 1)(2\sqrt{T} - 1)\sqrt{2\log(\mathcal{O}(Tn)/\delta)}\right). \end{aligned} \tag{C.2}$$

Note that $\xi_1, \xi_2 < 1$ by setting of m, m'. It is easy to observe that regret bound of EE-Net is tighter than EE-Net', because ϵ_1 is a given number and can be $\epsilon_1 \ll 1$. Therefore, by adding a exploration network f_2, EE-Net can achieve a tighter bound than pure-exploitation approach.

Proof of Theorem 2. In round t, given the n arms $(\mathbf{x}_{t,1}, \dots \mathbf{x}_{t,n})$, let $H_t = (h(\mathbf{x}_{t,i}), \dots, h(\mathbf{x}_{t,n}))$ denote their expected rewards. For brevity, for the selected arm $\mathbf{x}_t$ in round t, let $h(\mathbf{x}_t)$ be its expected reward and $\mathbf{x}_t^*$ be the optimal arm in round t.

Then, based on the proof of Theorem 1, the expected regret of round t is computed by

$$\begin{aligned} \mathbf{R}_t &= h(\mathbf{x}_t^*) - h(\mathbf{x}_t) \\ &\leq h(\mathbf{x}_t^*) - f_3(\mathbf{x}_t^*) + f_3(\mathbf{x}_t) - h(\mathbf{x}_t) \\ &\leq \max_{i \in [n]} \left(\mathbb{E}[(f_3(r_{t,i} - \mathbf{x}_{t,i}))\,|\mathbf{x}_{t,i}]\right) + \mathbb{E}[(f_3(\mathbf{x}_t) - r_t)\,|\mathbf{x}_t] \\ &\leq \max_{i \in [n]} \mathbb{E}[|f_1(\mathbf{x}_{t,i}; \boldsymbol{\theta}^1_{t-1}) - r_{t,i}||\mathbf{x}_{t,i}] + \mathbb{E}[|f_1(\mathbf{x}_i; \boldsymbol{\theta}^1_{t-1}) - r_t||\mathbf{x}_t] \end{aligned}$$

Then, based on Lemma C.1, with probability at least $1 - \delta$, we have

$$\mathbf{R}_t \leq 2\left(\sqrt{\frac{2\epsilon_1}{t}} + (\xi_1 + 1)\sqrt{\frac{2\log(\mathcal{O}(Tn)/\delta)}{t}}\right)$$

Then expected regret of T rounds is computed by

$$\begin{aligned} \mathbf{R}_T &= \sum_{t=1}^{T} \mathbf{R}_t \\ &\leq 2\sum_{t=1}^{T}\left(\sqrt{\frac{2\epsilon_1}{t}} + (\xi_1 + 1)\sqrt{\frac{2\log(\mathcal{O}(Tn)/\delta)}{t}}\right) \\ &< \underbrace{2(2\sqrt{T} - 1)\sqrt{2\epsilon_1} + 2(\xi_1 + 1)(2\sqrt{T} - 1)\sqrt{2\log(\mathcal{O}(Tn)/\delta)}}_{I_2} \end{aligned} \tag{C.3}$$

where I_2 is because $\sum_{t=1}^{T} \frac{1}{\sqrt{t}} \leq \int_1^T \frac{1}{\sqrt{t}}\, dx + 1 = 2\sqrt{T} - 1$ (Chlebus, 2009).

When $K = \Omega(\frac{\text{poly}(T,n,L)}{\delta^2} \cdot \log T)$, according to Lemma 3.4, with probability at least $1-\delta$, we have $\epsilon_1 = 1/T$. Therefore, we have

$$\mathbf{R}_T \leq \mathcal{O}(1) + 2(\xi_1 + 1)(2\sqrt{T} - 1)\sqrt{2\log(\mathcal{O}(Tn)/\delta)}, \tag{C.4}$$

The proof is completed. □

Lemma C.1. *Given $0 < \epsilon_1 < 1$, suppose m, η satisfy the conditions in Eq. (C.1) and $K_1 = \Omega(\frac{poly(T,n,L)}{\delta^2} \cdot \log \epsilon_1^{-1})$. Then, with probability at least $1-\delta$, for any $t \in [T], i \in [n]$, it holds uniformly that*

$$\mathbb{E}_{(\mathbf{x}_t, r_t)\sim\mathcal{D}}[|r_t - f_1(\mathbf{x}_t; \boldsymbol{\theta}_t)|] \leq \sqrt{\frac{2\epsilon_1}{t}} + (\xi_1 + 1)\sqrt{\frac{2\log(\mathcal{O}(Tn)/\delta)}{t}}$$

Proof. Applying Lemma B.2 on f_1, with probability at least $1-\delta$, for any $t \in [T]$, it holds that $|f_1(\mathbf{x}; \theta_t)| \leq \xi_1$. And due to $|r| \leq 1$ for any $(\mathbf{x}, r) \sim \mathcal{D}$, we have

$$|r - f_1(\mathbf{x}; \theta_t)| \leq \xi_1 + 1.$$

Then, we have

$$\frac{1}{t}\sum_{i=1}^{t} |r_i - f_1(\mathbf{x}_i; \theta_t)| \leq \frac{1}{t}\sqrt{t}\sqrt{\sum_{i=1}^{t}(r_i - f_1(\mathbf{x}_i; \theta_t))^2} = \underbrace{\sqrt{\frac{2\mathcal{L}}{t}} \leq \sqrt{\frac{2\epsilon_1}{t}}}_{I_1}. \tag{C.5}$$

where I_1 is an application of Lemma B.3 (1).

Then, based on Lemma 1 in Cesa-Bianchi et al. (2004), applying the union bound, with probability at least $1-2\delta$, we have

$$\begin{aligned}
\mathbb{E}_{(\mathbf{x},r)\sim\mathcal{D}}[|r - f_1(\mathbf{x}; \boldsymbol{\theta}_t)|] &= \frac{1}{t}\sum_{i=1}^{t} \mathbb{E}_{(\mathbf{x}_i, r_i)\sim\mathcal{D}}[|r_i - f_1(\mathbf{x}_i; \boldsymbol{\theta}_t)|] \\
&\leq \frac{1}{t}\sum_{i=1}^{t} |r_i - f_1(\mathbf{x}_i; \theta_t)| + (\xi_1 + 1)\sqrt{\frac{2\log(1/\delta)}{t}} \\
&\leq \sqrt{\frac{2\epsilon_1}{t}} + (\xi_1 + 1)\sqrt{\frac{2\log(1/\delta)}{t}}
\end{aligned} \tag{C.6}$$

Finally, rescaling δ completes the proof. □

D Gradient-based Upper Confidence Bound

In this section, we list one gradient-based UCB from existing works (Ban et al., 2021; Zhou et al., 2020), which motivates our design of exploration network f_2. Let $g(\mathbf{x}_t; \boldsymbol{\theta}_t) = \nabla_{\boldsymbol{\theta}_t} f(\mathbf{x}_t; \boldsymbol{\theta}_t)$.

Lemma D.1. *(Lemma 5.2 in (Ban et al., 2021)). Given a set of context vectors $\{\mathbf{x}_t\}_{t=1}^T$ and the corresponding rewards $\{r_t\}_{t=1}^T$, $\mathbb{E}(r_t) = h(\mathbf{x}_t)$ for any $\mathbf{x}_t \in \{\mathbf{x}_t\}_{t=1}^T$. Let $f(\mathbf{x}_t; \boldsymbol{\theta})$ be the L-layers fully-connected neural network where the width is m, the learning rate is η, the number of iterations of gradient descent is K. Then, there exist positive constants C_1, C_2, S, such that if*

$$m \geq poly(T, n, L, \log(1/\delta) \cdot d \cdot e^{\sqrt{\log 1/\delta}}), \;\; \eta = \mathcal{O}(TmL + m\lambda)^{-1}, \;\; K \geq \widetilde{\mathcal{O}}(TL/\lambda),$$

then, with probability at least $1-\delta$, for any $\mathbf{x}_t \in \{\mathbf{x}_t\}_{t=1}^T$, we have the following upper confidence bound:

$$|h(\mathbf{x}_t) - f(\mathbf{x}_t; \boldsymbol{\theta}_t)| \leq \gamma_1 \|g(\mathbf{x}_t; \boldsymbol{\theta}_t)/\sqrt{m}\|_{\mathbf{A}_t^{-1}} + \gamma_2 + \gamma_1\gamma_3 + \gamma_4, \tag{D.1}$$

where

$$\gamma_1(m, L) = (\lambda + t\mathcal{O}(L)) \cdot ((1 - \eta m\lambda)^{J/2}\sqrt{t/\lambda}) + 1$$

$$\gamma_2(m, L, \delta) = \|g(\mathbf{x}_t; \boldsymbol{\theta}_0)/\sqrt{m}\|_{\mathbf{A}_t'^{-1}} \cdot \left(\sqrt{\log\left(\frac{\det(\mathbf{A}_t')}{\det(\lambda\mathbf{I})}\right) - 2\log\delta} + \lambda^{1/2}S\right)$$

$$\gamma_3(m, L) = C_2 m^{-1/6}\sqrt{\log m}t^{1/6}\lambda^{-7/6}L^{7/2}, \;\; \gamma_4(m, L) = C_1 m^{-1/6}\sqrt{\log m}t^{2/3}\lambda^{-2/3}L^3$$

$$\mathbf{A}_t = \lambda\mathbf{I} + \sum_{i=1}^{t} g(\mathbf{x}_t; \boldsymbol{\theta}_t)g(\mathbf{x}_t; \boldsymbol{\theta}_t)^\intercal/m, \;\; \mathbf{A}_t' = \lambda\mathbf{I} + \sum_{i=1}^{t} g(\mathbf{x}_t; \boldsymbol{\theta}_0)g(\mathbf{x}_t; \boldsymbol{\theta}_0)^\intercal/m.$$

Note that $g(\mathbf{x}_t; \boldsymbol{\theta}_0)$ is the gradient at initialization, which can be initialized as constants. Therefore, the above UCB can be represented as the following form for exploitation network f_1: $|h(\mathbf{x}_{t,i}) - f_1(\mathbf{x}_{t,i}; \boldsymbol{\theta}_t^1)| \leq \Psi(g(\mathbf{x}_t; \boldsymbol{\theta}_t))$.

Disadvantages of UCB. (1) The above UCB tries to measure the potential gain for $\mathbf{x}_t$ compared to current estimation $f_1(\mathbf{x}_t)$, i.e., $y = h(\mathbf{x}_t) - f(\mathbf{x}_t; \boldsymbol{\theta}_t)$. However, this UCB only considers the worst case for y and this UCB usually is a very large number. This is why they need to add turning parameters in their implementation (Ban et al., 2021; Zhou et al., 2020). (2) For these arms with $h(\mathbf{x}_t) < f(\mathbf{x}_t; \boldsymbol{\theta}_t)$, their exploration score should be lower than the arms with $h(\mathbf{x}_t) > f(\mathbf{x}_t; \boldsymbol{\theta}_t)$, while their added points are same in UCB strategy.

In contrast, EE-Net has a neural network f_2 to learn y and intends to predict y accurately. Moreover, f_2 can discriminate the arms with $h(\mathbf{x}_t) > f(\mathbf{x}_t; \boldsymbol{\theta}_t)$ from the arms with $h(\mathbf{x}_t) < f(\mathbf{x}_t; \boldsymbol{\theta}_t)$.

E Experiments (Added)

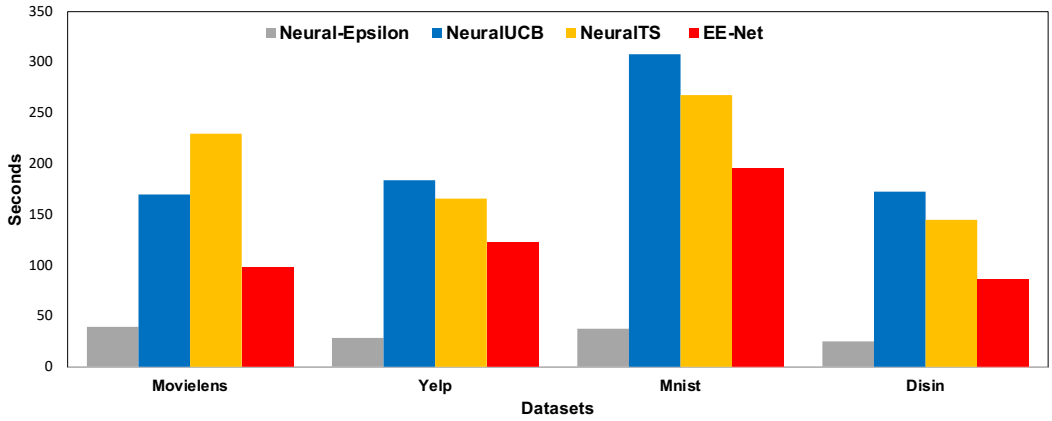

Figure 5: Decision-making time cost. The time cost of making the selection for each method is crucial to the performance in real applications.

Figure 5 reports the average decision-making time cost of 5 runs for neural-based methods. NeuralUCB and NeuralTS are about 1.5 to 2 times slower than EE-Net, because NeuralUCB and NeuralTS need additional computational cost for the gradient kernel matrix. Neural-Epsilon is the fastest approach because it only needs to compute one network output when making selections, while EE-Net needs to compute two network outputs.

Figure 6 reports the average overall time cost of 5 runs for neural-based methods. EE-Net is about 1.2-1.6 times slower than NeuralUCB and NeuralTS, because EE-Net needs to train two neural networks in each round while NeuralUCB and NeuralTS need to train one. Neural-Epsilon is still the fastest method because it only needs to train one neural network and it does not have the extra calculation of gradient.

Figure 7 reports the regret comparison on Movielens and Yelp data sets, where each method was run 30 times. These results are very similar to the results in Figure 1.

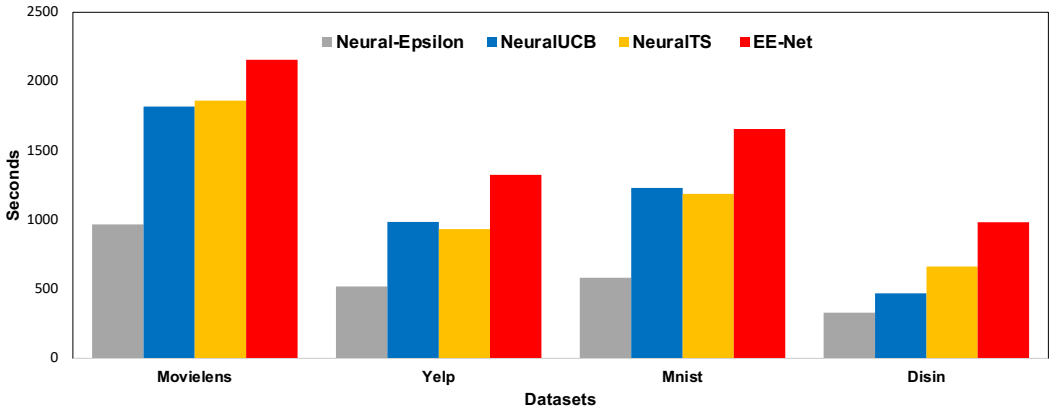

Figure 6: Overall time cost. This time cost includes the decision-making time cost and training time cost.

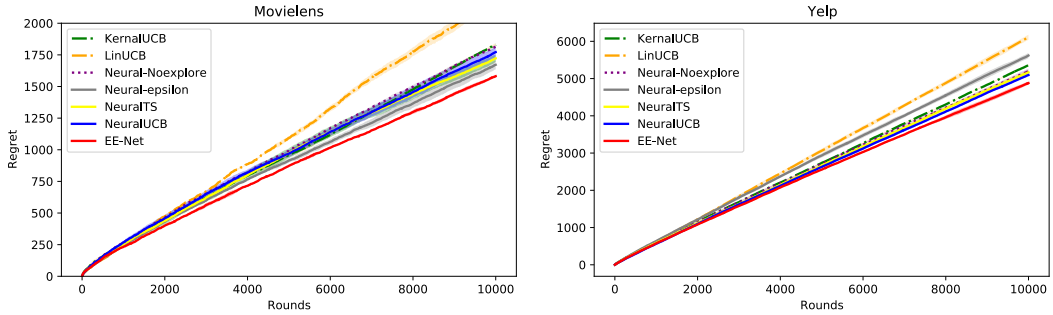

Figure 7: Regret report on Movielens and Yelp. Mean of 30 runs with standard deviation (shadow).