# EE-Net: Exploitation-Exploration Neural Networks in Contextual Bandits

**Yikun Ban, Yuchen Yan, Arindam Banerjee, Jingrui He**

**University of Illinois at Urbana-Champaign**

April, 2022

# Roadmap

1. **Background**
2. Problem Definition and Related Work
3. Proposed Algorithm: EE-Net
4. Comparison with Existing Work

# Background

1. Sequential decision-making problem is everywhere
   - Personalized recommendation.
   - Online advertising
   - Clinical trials

2. Exploitation-exploration dilemma exists in decision making
   - Exploitation: Making greedy decisions by exploiting past knowledge
   - Exploration: Taking risks to explore new information

3. Powerful tool: Contextual multi-armed bandits
   - $\epsilon$-greedy
   - Upper Confidence Bound (UCB)
   - Thompson Sampling (TS)

# Background: Contextual Bandit

$n$-arm contextual bandit problem:

- Learner observes $n$ $d$-dimensional contextual vectors (arms) in round $t$

$$\mathbf{X}_t = \{\mathbf{x}_{t,i} \in \mathbb{R}^d | i \in [n]\} \tag{1}$$

- Learner selects an arm $\mathbf{x}_{t,i'}$ and receives a reward $r_{t,i'}$. For brevity, denote by $\mathbf{x}_t$ the selected arm in round $t$ and by $r_t$ its reward.
- The goal is to minimize the following pseudo regret:

$$R_T = \mathbb{E}\left[\sum_{t=1}^{T}(r_t^* - r_t)\right] \tag{2}$$

where $r_t^* = \max_{i\in[n]} \mathbb{E}[r_{t,i}]$ is the maximal expected reward.

# Background: Linear Contextual Bandit

- Given an arm $\mathbf{x}_{t,i}, i \in [n]$, its reward $r_{t,i}$ is assumed to be a linear function:

$$r_{t,i} = \boldsymbol{\theta}^\top \mathbf{x}_{t,i} + \eta_{t,i}, \quad \eta_{t,i} \sim \nu - \text{sub-Gaussian} \tag{3}$$

  where $\boldsymbol{\theta}$ is unknown.
- To approximate $\boldsymbol{\theta}$, in round $t$, based on the past data $\{\mathbf{x}_\tau, r_\tau\}_{\tau=1}^{t-1}$, Ridge regression is applied

$$\hat{\boldsymbol{\theta}}_t = \mathbf{A}_{i_t,t}{}^{-\mathbf{1}}\mathbf{b}_{i_t,t}, \quad \mathbf{A}_{i_t,t} = \mathbf{I} + \sum_{\tau=1}^{t-1} \mathbf{x}_\tau \mathbf{x}_\tau^\intercal, \quad \mathbf{b}_{i_t,t} = \sum_{\tau=1}^{t} \mathbf{x}_\tau r_\tau, \tag{4}$$

  where $\mathbf{I}$ is a $d \times d$ identity matrix.

# Roadmap

1. Background

2. **Problem Definition and Related Work**

3. Proposed Algorithm: EE-Net

4. Comparison with Existing Work

# Problem Definition: Neural Contextual Bandit

- Given an arm $\mathbf{x}_{t,i}, i \in [n]$, its reward $r_{t,i}$ is assumed to be a linear/non-linear function:

$$r_{t,i} = h(\mathbf{x}_{t,i}) + \eta_{t,i}, \quad \mathbb{E}[\eta_{t,i}] = 0.0$$

  where $h$ is unknown and $0 \leq r_{t,i} \leq 1$.

- The goal is to minimize the following pesudo regret:

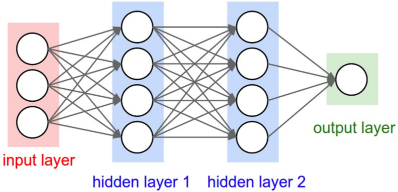
$$R_T = \mathbb{E}\left[\sum_{t=1}^{T}(r_t^* - r_t)\right]$$

  where $\mathbb{E}[r_{t,i} \mid \mathbf{x}_{t,i}] = h(\mathbf{x}_{t,i}), \forall i \in [n]$.

# Reward Estimation

- To learn some universal reward function $h$, use the universal function approximator, such as neural networks.
- Here, we use fully-connected neural network:

$$f(\mathbf{x}_{t,i}; \boldsymbol{\theta}) = \mathbf{W}_L \sigma(\mathbf{W}_{L-1} \sigma(\dots \sigma(\mathbf{W}_1 \mathbf{x}_{t,i}))).$$




where $\sigma$ is the ReLU activation function and $\boldsymbol{\theta} = \big(\text{vec}(\mathbf{W}_L)^\intercal, \dots,$ $\text{vec}(\mathbf{W}_1)^\intercal\big)^\intercal \in \mathbb{R}^p$.

# Related Work: NeuralUCB

NeuralUCB (Zhou et al., 2020):

- Let $g(\mathbf{x}_{t,i}; \boldsymbol{\theta})$ be the gradient $\nabla_{\boldsymbol{\theta}} f(\mathbf{x}_{t,i}; \boldsymbol{\theta})$.
- In round $t$, given $n$ arms $\{\mathbf{x}_{t,1}, \ldots, \mathbf{x}_{t,n}\}$, we select arm by

$$\mathbf{x}_t = \arg_{i \in [n]} \max \left( \underbrace{f(\mathbf{x}_{t,i}; \boldsymbol{\theta}_{t-1})}_{\text{Exploitation: Estimated reward}} + \underbrace{\gamma_{t-1} \sqrt{g(\mathbf{x}_{t,i}; \boldsymbol{\theta}_{t-1})^\top \mathbf{Z}_{t-1}^{-1} g(\mathbf{x}_{t,i}; \boldsymbol{\theta}_{t-1}) / m}}_{\text{Exploration: UCB}} \right) \tag{5}$$

  where $\gamma_{t-1}$ is a tuning parameter and $\mathbf{Z}_{t-1} = \mathbf{I} + \sum_{t'=1}^{t} g(\mathbf{x}_{t'}; \boldsymbol{\theta}) g(\mathbf{x}_{t'}; \boldsymbol{\theta})^\top$ is the gradient outer product matrix.
- $f$ is trained based on the historical data $\{\mathbf{x}_\tau, r_\tau\}_{\tau=1}^{t}$.

# Related Work: Neural Thompson Sampling (NeuralTS)

NeuralTS (Zhang et al., 2021):

- Given an arm $\mathbf{x}_{t,i}$, to learn the expected reward $h(\mathbf{x}_{t,i})$, use the neural network

$$f(\mathbf{x}_{t,i}; \boldsymbol{\theta}) = \mathbf{W}_L \sigma(\mathbf{W}_{L-1}\sigma(\dots \sigma(\mathbf{W}_1 \mathbf{x}_{t,i}))).$$

- In round $t$, given $n$ arms $\{\mathbf{x}_{t,1}, \dots, \mathbf{x}_{t,n}\}$, select an arm by

$$\forall i \in [n], \text{draw } \hat{r}_{t,i} \sim \mathcal{N}( \underbrace{f(\mathbf{x}_{t,i}; \boldsymbol{\theta})}_{\text{Mean: Exploitation}}, \underbrace{\sigma^2}_{\text{Variance: Exploration}} ) \tag{6}$$

$$\text{Select } \mathbf{x}_t = \arg_{i \in [n]} \max \hat{r}_{t,i}.$$

where $\sigma = \nu g(\mathbf{x}_{t,i}; \boldsymbol{\theta}_{t-1})^\top \mathbf{Z}_{t-1}^{-1} g(\mathbf{x}_{t,i}; \boldsymbol{\theta}_{t-1})$.

- Receive reward and update parameters.

# Roadmap

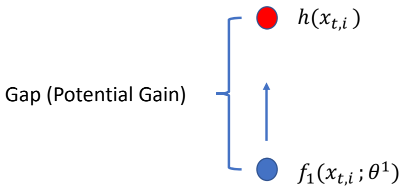
1. Background
2. Problem Definition and Related Work
3. **Proposed Algorithm: EE-Net**
4. Comparison with Existing Work

# EE-Net: Exploitation Neural Network

- To learn the expected reward function $h(\cdot)$, we use one neural network to learn it,

$$f_1(\mathbf{x}_{t,i}; \boldsymbol{\theta}^1) = \mathbf{W}_L \sigma(\mathbf{W}_{L-1} \sigma(\dots \sigma(\mathbf{W}_1 \mathbf{x}_{t,i}))),$$

  where $f_1$ is trained based on historical data $\{\mathbf{x}_\tau, r_\tau\}_{\tau=1}^{t}$ (**Exploitation Network**).
- Why explore? To fill the gap between expected reward and estimated reward.



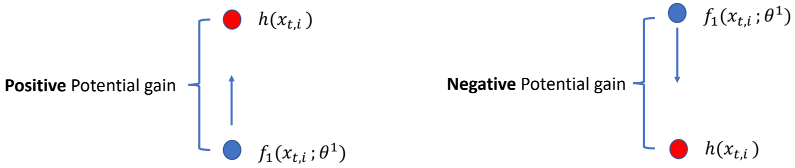

Case 1: When expected reward is larger than estimated reward.

# EE-Net: Exploration Neural Network

- Instead of calculating a statistic upper bound for $|h(\mathbf{x}_{t,i}) - f_1(\mathbf{x}_{t,i}; \boldsymbol{\theta}^2)|$, EE-Net uses another neural network $f_2$ to learn $h(\mathbf{x}_{t,i}) - f_1(\mathbf{x}_{t,i}; \boldsymbol{\theta}^2)$.

$$f_2(\mathbf{x}_{t,i}; \boldsymbol{\theta}^2) = \mathbf{W}_L \sigma(\mathbf{W}_{L-1} \sigma(\dots \sigma(\mathbf{W}_1 \mathbf{x}_{t,i}))).$$

- Ground truth: $h(\mathbf{x}_{t,i}) - f_1(\mathbf{x}_{t,i}; \boldsymbol{\theta}^1)$, i.e., $r_{t,i} - f_1(\mathbf{x}_{t,i}; \boldsymbol{\theta}^1)$.
- $h(\mathbf{x}_{t,i}) - f_1(\mathbf{x}_{t,i}; \boldsymbol{\theta}^1)$ indicates exploration direction: "Upward" or "Downward" exploration.




Case 1: **Upward Exploration**

Case 2: **Downward Exploration**

# EE-Net: Exploration Neural Network

- Input: Gradient $\triangledown_{\boldsymbol{\theta}_1} f_1(\mathbf{x}_{t,i}; \boldsymbol{\theta}^1)$. Why?
- $\triangledown_{\boldsymbol{\theta}_1} f_1(\mathbf{x}_{t,i}; \boldsymbol{\theta}^1)$ contains two sides of information.
  1. Arm feature $\mathbf{x}_{t,i}$.
  2. Discriminative ability of $f_1$(Exploration depending on the exploitation).
- Build loss function $\mathcal{L}_2$

$$\mathcal{L}_2 = \frac{1}{2} \sum_{i=1}^{t} \left( f_2 \left( \triangledown_{\boldsymbol{\theta}^1} f_1(\mathbf{x}_{t,i}; \boldsymbol{\theta}^1); \boldsymbol{\theta}^2 \right) - \underbrace{(r_i - f_1(\mathbf{x}_{t,i}; \boldsymbol{\theta}^1))}_{\text{Ground truth}} \right)^2$$

- After receiving $r_t$ in round $t$, based on $\left\{ \triangledown_{\boldsymbol{\theta}_1} f_1(\mathbf{x}_\tau; \boldsymbol{\theta}^1_{\tau-1}), r_\tau - f_1(\mathbf{x}_\tau; \boldsymbol{\theta}^1_{\tau-1}) \right\}_{\tau=1}^{t}$, use gradient descent to update $\boldsymbol{\theta}^2$.

# EE-Net: Selection Criterion 1

- In round $t$, given $n$ arms $\{\mathbf{x}_{t,1}, \ldots, \mathbf{x}_{t,n}\}$, we select arm by

$$\mathbf{x}_t = \arg \max_{\mathbf{x}_{t,i}, i \in [n]} \left( \underbrace{f_1(\mathbf{x}_{t,i}; \boldsymbol{\theta}^1_{t-1})}_{\text{Exploitation}} + \underbrace{f_2\left( \nabla_{\boldsymbol{\theta}^1_{t-1}} f_1(\mathbf{x}_{t,i}; \boldsymbol{\theta}^1_{t-1}); \boldsymbol{\theta}^2_{t-1} \right)}_{\text{Exploration}} \right) \quad (7)$$

- Receive reward $r_t$ and update $\boldsymbol{\theta}^1, \boldsymbol{\theta}^2$.

# EE-Net: Decision Maker

Build Decision Maker $f_3(\cdot;\boldsymbol{\theta}^3)$.

- In roung $t$, given an arm $\mathbf{x}_{t,i}$, calculate its $f_1, f_2$ scores.
- Build a neural network $f_3(\cdot;\boldsymbol{\theta}^3)$.
- Input: $f_1(\mathbf{x}_{t,i};\boldsymbol{\theta}^1_{t-1}), f_2(\nabla_{\boldsymbol{\theta}^1_{t-1}} f_1(\mathbf{x}_{t,i});\boldsymbol{\theta}^2_{t-1})$.
- Ground truth: $p_{t,i}$, i.e., the probability of $\mathbf{x}_{t,i}$ being the optimal arm in round $t$.
  1. Binary reward $(0,1)$: $p_{t,i} = 1.0$ if $r_{t,i} = 1$; Otherwise, $p_{t,i} = 0.0$ if $r_{t,i} = 0$.
  2. Continuous reward $[0,1]$: (1) $p_{t,i} = \frac{r_{t,i}-0}{1-0} = r_{t,i}$; (2) Set a threshold $\gamma$. $p_{t,i} = 1.0$ if $r_{t,i} > \gamma$; Otherwise $p_{t,i} = 0.0$.
- Build loss function:

$$\mathcal{L}_3 = -\frac{1}{t}\sum_{i=1}^{t}\left[p_t \log f_3((f_1, f_2);\boldsymbol{\theta}^3) + (1-p_t)\log(1 - f_3((f_1, f_2);\boldsymbol{\theta}^3))\right]. \quad (8)$$

- Update $\boldsymbol{\theta}^3$ in each round.

# EE-Net: Selection Criterion 2

- In round $t$, given $n$ arms $\{\mathbf{x}_{t,1}, \ldots, \mathbf{x}_{t,n}\}$, we select arm by

$$1.\ \text{Calculated } f_1(\mathbf{x}_{t,i}; \boldsymbol{\theta}^1_{t-1}),\ f_2(\nabla_{\boldsymbol{\theta}^1_{t-1}} f_1(\mathbf{x}_{t,i}); \boldsymbol{\theta}^2_{t-1}) \tag{9}$$

$$2.\ \mathbf{x}_t = \arg \max_{\mathbf{x}_{t,i}, i \in [n]} f_3\left( (f_1(\mathbf{x}_{t,i}; \boldsymbol{\theta}^1_{t-1}), f_2(\nabla_{\boldsymbol{\theta}^1_{t-1}} f_1; \boldsymbol{\theta}^2_{t-1})); \boldsymbol{\theta}^3_{t-1} \right) \tag{10}$$

- Receive reward $r_t$ and update $\boldsymbol{\theta}^1, \boldsymbol{\theta}^2, \boldsymbol{\theta}^3$.

# EE-Net: Regret Upper Bound

- Regret bound complexity:

$$R_T \leq \mathcal{O}(\sqrt{T \log T}),$$

tighter than existing works.

**Theorem 1.** *Let $f_1, f_2$ follow the setting of $f$ (Eq. (5.1) ) with the same width $m$ and 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 $\delta \in (0,1), \epsilon \in (0, \mathcal{O}(\frac{1}{T})], \rho \in (0, \mathcal{O}(\frac{1}{L})]$, suppose*

$$m \geq \widetilde{\Omega}\left( poly(T, n, L, \rho^{-1}) \cdot \log(1/\delta) \cdot e^{\sqrt{\log(Tn/\delta)}} \right),$$

$$\eta_1 = \eta_2 = \min\left( \Theta\left( \frac{T^5}{\sqrt{2}\delta^2 m} \right), \Theta\left( \frac{\rho}{poly(T,n,L) \cdot m} \right) \right), \tag{5.2}$$

$$K_1 = K_2 = \Theta\left( \frac{poly(T,n,L)}{\rho\delta^2} \cdot \log\left(\epsilon^{-1}\right) \right).$$

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

$$\mathbf{R}_T \leq (2\sqrt{T}-1)(2\sqrt{2\epsilon} + 3\sqrt{2}\mathcal{O}(L)) + 2(1+2\xi)(2\sqrt{T}-1)\sqrt{2\log\frac{\mathcal{O}(Tn)}{\delta}}, \tag{5.3}$$

# EE-Net: Generalization Bound

- Generalization bound of neural networks in bandit framework: decrease with a fixed $\tilde{\mathcal{O}}(\frac{1}{\sqrt{t}})$-rate.

**Lemma 5.1.** *Given $\delta, \epsilon \in (0,1), \rho \in (0, \mathcal{O}(\frac{1}{L}))$, suppose $m, \eta_1, \eta_2, K_1, K_2$ satisfy the conditions in Eq. (5.2) and $(\mathbf{x}_{\tau,i}, r_{\tau,i}) \sim \mathcal{D}, \forall \tau \in [t], i \in [n]$. Let*

$$\mathbf{x}_t = \arg \max_{\mathbf{x}_{t,i}, i \in [n]} \left[ f_2\left( \frac{\nabla_{\boldsymbol{\theta}^1_{t-1}} f_1(\mathbf{x}_{t,i}; \boldsymbol{\theta}^1_{t-1})}{c_1 \sqrt{mL}}; \boldsymbol{\theta}^2_{t-1} \right) + f_1(\mathbf{x}_{t,i}; \boldsymbol{\theta}^1_{t-1}) \right],$$

*and $r_t$ is the corresponding reward, given $(\mathbf{x}_{t,i}, r_{t,i}), i \in [n]$. Then, with probability at least $(1-\delta)$ over the random of the initialization, it holds that*

$$\begin{aligned} &\mathop{\mathbb{E}}_{(\mathbf{x}_{t,i}, r_{t,i}), i \in [n]} \left[ \left| f_2\left( \frac{\nabla_{\boldsymbol{\theta}^1_{t-1}} f_1(\mathbf{x}_t; \boldsymbol{\theta}^1_{t-1})}{c_1 \sqrt{mL}}; \boldsymbol{\theta}^2_{t-1} \right) - \left( r_t - f_1(\mathbf{x}_t; \boldsymbol{\theta}^1_{t-1}) \right) \right| \mid \{\mathbf{x}_\tau, r_\tau\}_{\tau=1}^{t-1} \right] \\ &\leq \sqrt{\frac{2\epsilon}{t}} + \mathcal{O}\left( \frac{3L}{\sqrt{2t}} \right) + (1+2\xi) \sqrt{\frac{2\log(\mathcal{O}(tn/\delta))}{t}}, \end{aligned} \tag{5.6}$$

*where the expectation is also taken over $(\boldsymbol{\theta}^1_{t-1}, \boldsymbol{\theta}^2_{t-1})$ that are uniformly drawn from $(\widehat{\boldsymbol{\theta}}^1_\tau, \widehat{\boldsymbol{\theta}}^2_\tau), \tau \in [t-1]$.*

# Roadmap

1. Background
2. Problem Definition and Related Work
3. Proposed Algorithm: EE-Net
4. **Comparison with Existing Work**

# Comparison 1: Selection Criterion

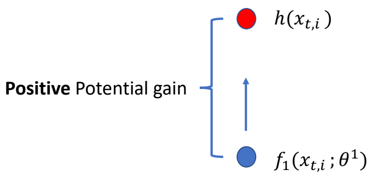
Table 1: Selection Criterion Comparison ($\mathbf{x}_t$: selected arm in round $t$).

| Methods | Selection Criterion |
|---|---|
| Neural Epsilon-greedy | With probability $1-\epsilon$, $\mathbf{x}_t = \arg\max_{\mathbf{x}_{t,i}, i\in[n]} f_1(\mathbf{x}_{t,i};\boldsymbol{\theta}^1)$; Otherwise, select $\mathbf{x}_t$ randomly. |
| NeuralTS (Zhang et al., 2021) | 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, select $\mathbf{x}_{t,\hat{i}}$, $\hat{i} = \arg\max_{i\in[n]} \hat{r}_{t,i}$. |
| NeuralUCB (Zhou et al., 2020) | $\mathbf{x}_t = \arg\max_{\mathbf{x}_{t,i}, i\in[n]} \left(f_1(\mathbf{x}_{t,i};\boldsymbol{\theta}^1) + \mathrm{UCB}_{t,i}\right)$. |
| 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 $\mathbf{x}_t = \arg\max_{\mathbf{x}_{t,i} i\in[n]} f_3(f_1, f_2;\boldsymbol{\theta}^3)$. |

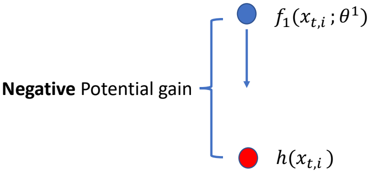
# Comparison 2: Exploration Direction

Table 3: Exploration Direction Comparison.

| Methods | "Upward" Exploration | "Downward" Exploration |
|---|---|---|
| NeuralUCB | $\surd$ | $\times$ |
| NeuralTS | Randomly | Randomly |
| EE-Net | $\surd$ | $\surd$ |




Case 1: **Upward Exploration**




Case 2: **Downward Exploration**

# Comparison 3: Complexity

Table 4: Regret Bound Comparison.

| Methods | Regret Upper Bound | Effective Dimension $\tilde{d}$ |
|---|---|---|
| NeuralUCB | $\mathcal{O}(\sqrt{\tilde{d}T}\log T)$ | Yes |
| NeuralTS | $\mathcal{O}(\sqrt{\tilde{d}T}\log T)$ | Yes |
| EE-Net | $\mathcal{O}(\sqrt{T}\sqrt{\log T})$ | No |

Table 5: Running Time/Space Complexity Comparison ($p$ is number of parameters of $f_1$).

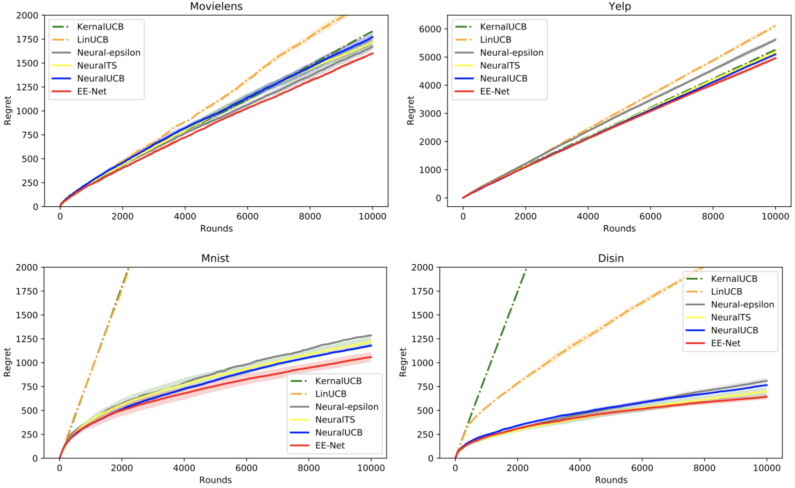
| Methods | Time | Space | Training Time (# Neural Networks) |
|---|---|---|---|
| NeuralUCB | $\mathcal{O}(p^2)$ | $\mathcal{O}(p^2)$ | 1 |
| NeuralTS | $\mathcal{O}(p^2)$ | $\mathcal{O}(p^2)$ | 1 |
| EE-Net | $\mathcal{O}(p)$ | $\mathcal{O}(p)$ | 2-3 |

# Comparison 4: Empirical Performance

- Public data sets: Mnist, Yelp, MovieLens, Disinformation
- Baselines:
  1. LinUCB (Li et al., 2010)
  2. KernelUCB (Valko et al., 2013)
  3. Neural-Epsilon
  4. NeuralUCB (Zhou et al., 2020)
  5. NeuralTS (Zhang et al., 2021)
- Report: Average regret of 10 runs with standard deviation (shadow)

# Comparison 4: Empirical Performance

- **EE-Net** outperforms all baselines cross all data sets.



# Summary

- **Main contributions**

1. We propose a novel neural exploration strategy, EE-Net, where another neural network is assigned to learn the potential gain compared to the current reward estimate.
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 contextual bandit algorithms and independent of the input dimension.
3. We conduct extensive experiments on four real-world data sets, showing that EE-Net outperforms baselines including linear and neural versions of $\epsilon$-greedy, TS, and UCB.

Thanks!