Transformers can reinforcement learn to approximate Gittins Index

Vladimir Petrov
Harvard College
vpetrov@college.harvard.edu

Nikhil Vyas
Harvard University
nikhil@g.harvard.edu

Lucas Janson
Harvard University
ljanson@fas.harvard.edu

Abstract

Transformers have demonstrated the ability to approximate *in-context* a rich class of functions in supervised learning and more recently in reinforcement learning (RL) settings. In this work, we investigate the transformer's ability to in-context learn the Gittins index, an online RL algorithm computed via dynamic programming (DP) and known to be optimal in Bayesian Bernoulli bandits. Our experiments show that the transformer can learn to approximate this strategy very well in a pure RL manner, without expert demonstrations, especially after we account for the problem's underlying symmetric properties. Our results, therefore, serve as empirical evidence that the class of RL algorithms transformers can learn in context extends to include certain DP-based algorithms.

1 Introduction

LLMs like GPT4 have shown the ability to generalize well across a wide variety of tasks through a phenomenon called in-context learning (ICL) [2]. When given a prompt of a few input-output examples, the model can generate relevant predictions for novel queries without parameter updates. The remarkable generalizability of ICL has motivated extensive research into which functions can be in-context approximated by a transformer, the core architecture of LLMs [15]. Significant progress has been made in supervised learning settings (e.g., [5, 16, 1]). Concurrently, transformers have garnered increasing attention in RL [14], where the sequential nature of the problem is well-aligned with transformers' demonstrated success on sequential tasks (e.g., [3, 7, 9, 10]).

Our paper provides empirical insights into in-context reinforcement learning (ICRL), where the goal is to train models to *act like* RL algorithms. The model should learn how to adapt the optimal strategy to the current environment based on the observed past interactions. This fast adaptation draws a connection to the field of *meta reinforcement learning*, where the goal is to learn algorithms that generalize well across the distribution of RL tasks rather than excel at a single RL task. We emphasize the *learning* aspect of these fields: rather than imitating the expert, the model should independently discover the optimum (via online RL), possibly outperforming existing SOTA methods. This motivates the major question for our paper:

Which RL algorithms can transformers efficiently meta-learn in-context?

Researchers have explored this question for other architectures (e.g., Wang et al. [17] trained RNNs and Duan et al. [4] trained LSTMs to solve different RL problems). Other works have studied the transformer's performance in the *supervised* pretraining setting, i.e. where the model aims to

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

approximate expert solutions through demonstrations [9, 10]. The closest to our work, Mishra et al. [11] proposes a *modified* transformer architecture called SNAIL which they show approximates Gittins index [6] well. SNAIL manages to achieve this optimal strategy but at the expense of only a finite context size. Namely, their model would have to grow as $O(\log(T))$ in depth as the context size T (horizon) increases. In general, SNAIL's architecture significantly differs from the *original* transformer, which Mishra et al. claim was incapable of solving the bandits in their experiments. In contrast, our work shows that the *original* transformer (and in fact, a transformer of modest size) can approximate this Gittins index strategy very well, especially after being encouraged to respect the problem's underlying symmetry. Our results, therefore, serve as empirical evidence that the class of RL algorithms transformers can learn in context extends to include certain DP-based algorithms.

2 Methodology

2.1 Bayesian Bernoulli Bandits and Gittins index

First, we remind the reader about the Bayesian Bernoulli bandit formulation. A *single* Bernoulli bandit M is defined by a vector $(\mu_1, \mu_2, ..., \mu_K) \in [0,1]^K$ of unobserved arm means, where at each step $t = 0, ..., T-1$ the agent chooses some action a_t and receives the reward $r_t \sim \mathrm{Bernoulli}(\mu_{a_t})$. With the transformer f_θ parametrized by θ (all weights and biases), at each step t the agent makes a decision based on the output $f_\theta(H_t) = f_\theta(a_0, r_0, ..., a_{t-1}, r_{t-1})$, a vector of probabilities for choosing different arms. At the end of the episode, the agent observes the final trajectory $\tau = (a_0, r_0, ..., a_{T-1}, r_{T-1})$ and receives a discounted reward $R(\tau) = \sum_{t=0}^{T-1} \gamma^t r_t$. Thus the distribution of a trajectory τ is defined by the policy and reward distributions, and so for a given bandit M and policy f_θ, we will denote τ's distribution by (M, f_θ). The *Bayesian* MAB assumes a prior distribution of single bandits $\langle \mu_1, \mu_2, ..., \mu_K \rangle \sim \mathbf{D}$ (e.g., $\mu_i \overset{\text{i.i.d.}}{\sim} \mathrm{Unif}(0,1)$), with the goal to maximize (or, equivalently, minimize the negative of) the expected reward across this *whole* distribution:

$$J(\theta) = (-1) \cdot \mathbb{E}_{M \sim \mathbf{D}}[R(M)] = -\,\mathbb{E}_{M \sim \mathbf{D}}\left[\mathbb{E}_{\tau \sim (M, f_\theta)}\, R(\tau)\right] \tag{1}$$

and we aim to find $\theta^* = \arg\min_\theta J(\theta)$. During evaluation, at each step t the model decides about optimal choices sequentially by analyzing the history of past interactions $H_t = (a_0, r_0, ..., a_{t-1}, r_{t-1})$ entirely in-context (parameters θ are frozen), positioning our problem as in-context RL.

For any prior distribution $\mathbf{D}$, the strategy minimizing (1) with the *infinite* horizon $T = \infty$ is known to be the Gittins index [6]. At each step, it updates a scalar value independently for each arm using *dynamic programming* and pulls the arm with the largest value. In reality, we can train the transformer only for finite horizon T, but in our settings of (γ, T) Gittins index becomes almost identical to the optimal solution for the finite horizon case.[1]

2.2 Gradient Estimate and Optimization

To minimize the objective (1), we use on-policy stochastic gradient descent. We follow the classic REINFORCE algorithm [18] to derive an unbiased gradient expression (which is still valid if we allow the policy to condition on the actions-rewards history $H_t = (a_0, r_0, ..., a_{t-1}, r_{t-1})$):

$$\nabla_\theta\, J(\theta) = -\,\mathbb{E}_{M \sim D}\left[\mathbb{E}_{\tau \sim (M, f_\theta)}\left[\sum_{t=1}^{T-1} \nabla_\theta \log \pi_\theta(a_t | H_t)\right] \cdot R(\tau)\right]. \tag{2}$$

This gradient is estimated using a batch of B trajectory rollouts τ_i via Monte Carlo: we first draw a bandit $M_i \sim \mathbf{D}$ and then interact with this bandit for T timesteps according to the current policy π_θ (this is repeated in parallel for $i = 1, ..., B$). To further reduce the variance of this estimate, we use baselines (also from [18]), which we estimate with the value network, generalized advantage estimates [13] (making the estimate slightly biased but with much lower variance), and entropy regularization to encourage more exploration. The final total loss is the weighted sum between these components mirroring the "A3C" approach ([12], further details can be found in the appendix A.1). The transformer dimensions we used for our major experiments: n_embd=96, n_layer=6, n_head=3, which results in approximately 0.5 million parameters. The code for the algorithm, along with all other components of the work, can be found on the anonymized GitHub link.

[1] This is because γ^T becomes negligible, e.g., with $\gamma = 0.9, T = 100, 0.9^{100} \approx 2e-5$.

2.3 Symmetry modifications

In our problem, the optimal solution intuitively should be symmetric: if we swap the evidence (past data rewards) between two arms, our beliefs about pulling those arms should also switch accordingly. Formally, f_θ is considered to be symmetric if and only if for any permutation of arm indices $\sigma : \{1, .., K\} \xrightarrow{\mathsf{B}} \{1, .., K\}$ (B stands for bijection) and history $H_t = (a_0, r_0, ..., a_{t-1}, r_{t-1})$, the model's forward pass operator and permutation σ on arm indexes are interchangeable:

$$\sigma(f_\theta(H_t)) = f_\theta(\sigma(H_t)). \tag{3}$$

In this equation, the left side applies permutation σ on the resulting vector $f_\theta(H_t)$ of K probabilities for choosing different arms, while the right side first applies σ on arm choices $\sigma(H_t) = (\sigma(a_0), r_0, ..., \sigma(a_{t-1}), r_{t-1})$ (rewards kept the same), and then applies the forward pass f_θ.

Looking ahead to the experiment results, we found that the transformer converges to asymmetric solutions. This led us to incorporate into our experiments techniques that could alleviate the asymmetry, which we briefly describe here:

(1) Symmetrization transformation f_θ^S on the original model f_θ modifies its forward pass by applying all possible permutations σ to the input and averaging out the results. This makes the model satisfy (3) and we call this modification a "symmetrized transformer".

(2) Symmetry regularization penalizes the original transformer model f_θ for asymmetric behavior, namely any deviations between $f_\theta(H_t)$ and $\sigma^{-1}(f_\theta(\sigma(H_t)))$ for histories H_t encountered during the training (note that for a symmetric model those vectors should be the same and hence the penalty is zero). As opposed to the symmetrization, the regularization does *not* change the original architecture (forward pass remains the same), and therefore, sheds light on whether the original transformer can represent the optimal solution without any modifications. Formal definitions of both methods can be found in appendix A.2.

3 Experiments and Results

All our experiments are conducted for Bayesian Bernoulli bandits with different settings (arms K, horizon T, prior distribution $\mathbf{D}$) and different transformer designs (with/without symmetry modifications). We evaluate the models based on the final expected reward per episode (scalar value) and training convergence dynamics. The major results for $T = 100$ and $K = 5, 10, 50$ are presented in Figure 1, and more extended analysis can be found in the Appendix A.3.1.

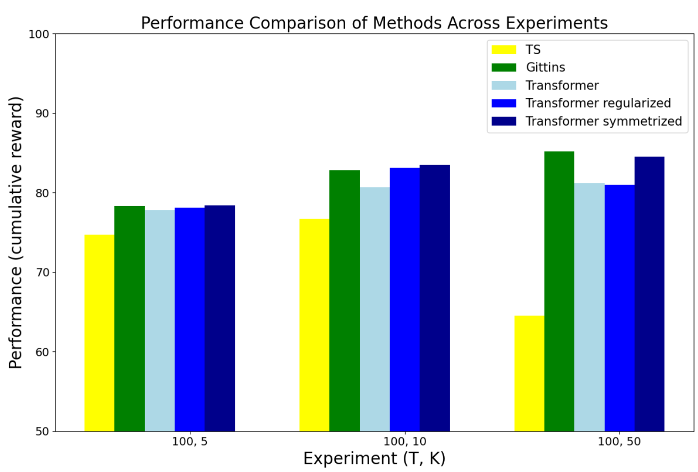

Figure 1: Total undiscounted reward on multi-arm bandits at the end of the episode: Thompson Sampling (TS; yellow), Gittins index (green, optimal as $T >> K$), and three transformer methods (blue colors).

There are two key takeaways from these barplots. **First**, even without symmetry adjustments, the original transformer (light blue) achieves expected reward close to the best possible, at times significantly outperforming Thompson Sampling. **Second**, regularizing toward and/or imposing symmetry helped, with the symmetrized transformer performing best across settings. This is supported by the convergence dynamics of all three methods, which we present for $T = 100, K = 10$ in Figure 2.

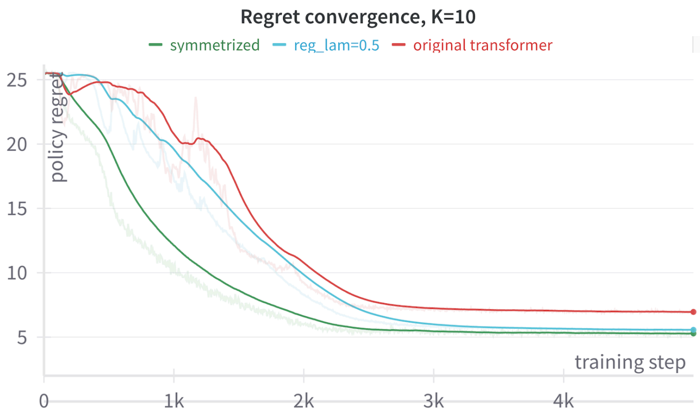

Figure 2: Training convergence for 10 arms for 3 models: original transformer (red), symmetrized (green), and regularized (blue).

The symmetrized transformer (green) has the fastest rate of convergence and the most stable training. In contrast, the original transformer (red) has sharp spikes where the empirical loss goes up significantly. We also find that these spikes usually correlate with asymmetric behavior: it goes up exactly when the model becomes biased towards some arms. Regularization stabilizes the training well (the blue plot smoothes out the red one), resulting in the optimal solution in the end.

Note that the green/blue lines essentially achieve the optimal loss of the Gittins index at the end of the training, while the red line plateaus at a higher level, suggesting that the original transformer gets stuck with an asymmetric solution.

Surprisingly, the symmetrized version has the fastest and most stable convergence even though we use K times fewer independent samples at each training step. We do this because for the symmetrized transformer we perform K cyclic permutations for every input trajectory, and hence we had to reduce our batch size K times in the Monte Carlo gradient estimates to fit into the training time/memory constraints (details can be found in Appendix A.2). Given this, a clear lead over the original transformer demonstrates how advantageous symmetrization is in our bandits problem.

While the regularization helped in smaller K settings ($K = 2, 5, 10$), we found it ineffective for $K = 50$, often resulting in unstable training and higher loss than the original transformer (appendix A.3.1). Further investigation is needed to robustify the regularization function and optimization process in this regime. In general, as K increases, the problem becomes harder since there is little time for exploration (e.g., for $K = 50$ and $T = 100$ there are ≤ 2 trials per arm on average, and the problem becomes very noisy). Larger K seems to lead to larger asymmetry gaps as showcased in our convergence plots for $K = 50$ (appendix A.3.1).

Furthermore, for $K = 2$, we compare decisions made by our model (arm a_t) and Gittins (arm g_t) when they observe the same history of past interactions H_t by calculating the 'difference scores' $\sum_{t=0}^{T-1} \gamma^t \cdot \mathbb{1}(a_t \neq g_t)$. This is done for every 'region' of size $\frac{1}{7} \times \frac{1}{7}$ where the true arm means μ_1, μ_2 can be situated within the unit square, which allows observing *directly* how our model deviates from Gittins depending on the underlying bandit. These scores are presented as a heatmap in Figure 3.

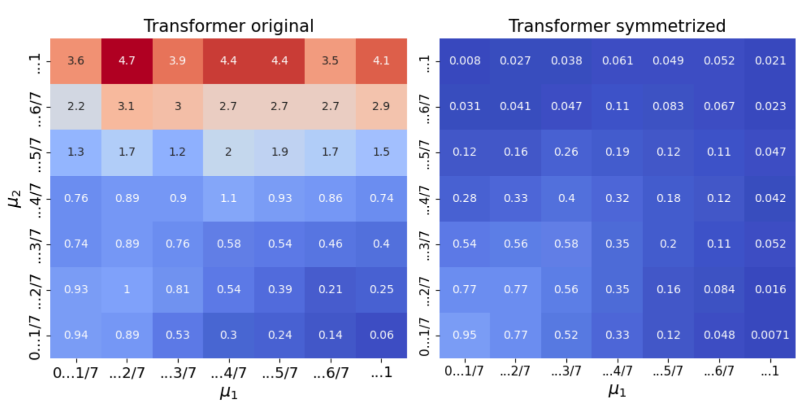

Figure 3: Difference scores with Gittins for the original (left) and symmetrized (right) transformers. Higher scores indicate more frequent deviations. For each region estimates are averaged across 250 repeated rollouts.

Now asymmetry is clear: on the left heatmap, entries above the diagonal $\mu_1 = \mu_2$ have higher difference scores than those below the diagonal, and hence the transformer underperforms. In contrast, the symmetrized transformer (on the right) produces a perfectly symmetric heatmap and much more closely matches Gittins index.

Another notable pattern is the decrease in difference scores from low to high μ's (bottom left to top right for the symmetrized transformer). We believe, to some extent, this is just the nature of the our problem and provide a detailed discussion in Appendix A.3.2.

We further test the effectiveness of the transformer by conducting an out-of-distribution (OOD) experiment. For $K = 2$ we change the prior of arm means from Uniform to $\mathrm{Beta}(10, 10)$, making the chances of μ_1, μ_2 falling close to 0 or 1 negligibly small. During test time we compare the performance on the "held-out" regions when those means fall on the boundary of a unit square. We find that the fitted transformer solution generalizes very well out-of-distribution, on par with the Gittins index. Details about this experiment can be found in the Appendix A.4.

To conclude, our work aims to deepen the understanding of which reinforcement learning algorithms transformers can learn in-context. In this paper specifically, we study the Gittins Index algorithm, which is typically computed via dynamic programming. Our scientific approach hypothesized that the transformer could approximate it well in context. Our empirical evidence, however, indicates that this is not entirely the case for the original transformer, as it demonstrated suboptimal performance across various settings (# of arms and horizon). This led us to formulate a new hypothesis: that the asymmetry was causing the problem, which was further validated via experiments with symmetrizing interventions. These interventions helped to stabilize the training and obtain the optimum in most regimes.

One of the interventions, regularization, does not affect the transformer's architecture but simply adjusts the optimization process via loss modifications. This result serves as empirical evidence that the *original* transformer can approximate the Gittins index, and, as a result, achieve a perfect exploration-exploitation balance. In contrast, recent works [8] have shown that LLMs like GPT4 fail to do even basic exploration in bandits, let alone approximate the optimum. We, therefore, hypothesize that this lack of exploration stems from the training procedure (next-token prediction) rather than inherent limitations in the transformer's architecture, and we hope that further research into online in-context RL at scale can open up novel powerful model capabilities.

References

[1] Yu Bai, Fan Chen, Huan Wang, Caiming Xiong, and Song Mei. Transformers as statisticians: Provable in-context learning with in-context algorithm selection. *Advances in neural information processing systems*, 36, 2024.

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[3] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[4] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl ˆ 2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

[5] Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems*, 35:30583–30598, 2022.

[6] John C Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 41(2):148–164, 1979.

[7] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

[8] Akshay Krishnamurthy, Keegan Harris, Dylan J Foster, Cyril Zhang, and Aleksandrs Slivkins. Can large language models explore in-context? *arXiv preprint arXiv:2403.15371*, 2024.

[9] Michael Laskin, Luyu Wang, Junhyuk Oh, Emilio Parisotto, Stephen Spencer, Richie Steigerwald, DJ Strouse, Steven Hansen, Angelos Filos, Ethan Brooks, et al. In-context reinforcement learning with algorithm distillation. *arXiv preprint arXiv:2210.14215*, 2022.

[10] Jonathan Lee, Annie Xie, Aldo Pacchiano, Yash Chandak, Chelsea Finn, Ofir Nachum, and Emma Brunskill. Supervised pretraining can learn in-context reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[11] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017.

[12] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[13] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[14] R. S. Sutton and A. G. Barto. Reinforcement learning: An introduction (2nd edition). *MIT press*, 2018.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. pages 6000–6010, 2017.

[16] Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pages 35151–35174. PMLR, 2023.

[17] Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

[18] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

A Appendix

A.1 Training details

The primary "policy" loss during our training can be defined as:

$$L_P(\theta) = -\,\mathbb{E}_M\,\mathbb{E}_{\tau\sim M, f_\theta}\left(\sum_{t=1}^{T}(\log\pi_\theta(a_t|H_t)\cdot(R_{[t:]}(\tau)-\gamma^t V_{\theta_V}(H_t))\right) \tag{4}$$

This closely follows the gradient expression from (2), except that we use $R_{[t:]}(\tau) = \gamma^0 r_t + \gamma^1 r_{t+1} + \ldots + \gamma^{T-t-1} r_{T-1}$ instead of $R(\tau)$ and estimate baseline with the value network $V_{\theta_V}(H_t)$, which mirrors the approach from REINFORCE [18] and A3C [12] papers. We further modify this difference with the generalized advantage estimate (GAE, [13]) $\widetilde{A}^{GAE(\gamma,\lambda)}(\tau)$, where λ is the tuned hyperparameter that balances the bias-variance tradeoff. Finally, we add the value $\widehat{L_V}(\theta)$ and entropy $\widehat{L}_{entropy}(\theta)$ losses, mirroring the approach from [12].

During training, we cannot compute *exactly* either of the three losses (that would require integrating over all MABs and trajectories τ), so they are replaced with the Monte-Carlo estimates. For instance, the estimate for the policy loss is:

$$\widehat{L_P}(\theta) = -\frac{1}{B}\sum_{\tau_i}\left((\sum_{t=1}^{T}\log\pi_\theta(a_t|H_t))\cdot(R_{[t:]}(\tau_i)-\gamma^t V_{\theta_V}(H_{i,t}))\right), \tag{5}$$

We use the largest batch size that fits into the memory: for $T = 10$, $B \approx 20000$ was enough, but for $T = 100$ we used $B = 5000$. Putting all parts together, the pseudocode is presented in the algorithm 1, and the code implementation, along with all other components of the work can be found on the anonymized GitHub link.

From the computing perspective, we used a single 80GB A-100 GPU for our major transformer model config (n_embd=96, n_layer=6, n_head=3). The number of training steps was usually in the range of 1K to 5K, which resulted in an average of 1.5-2 days of runtime.

Algorithm 1: Training procedure

```
Initialize θ_π and θ_V (all shared except the last layer)
for step in range(total steps) do
    Draw B bandits M_i from the distribution D. Generate actions a_0 by random (tensor of
      B × 1), get rewards r_0[i] by pulling a_0[i] in bandit M_i for each i.
    for t = 1...T-1 do
        Perform model's forward pass on H_t = (a_0, r_0, ..., a_{t-1}, r_{t-1}) // this input
          tensor has shape B × t × 2 (because we group each action-reward
          pair together)
        Use model's output logits to sample actions a_t // tensor B × 1
        Obtain rewards r_t by pulling arms a_t[i] for each bandit M_i.
    Using the last history H_T = (a_0, r_0, ..., a_T, r_T), calculate Monte Carlo estimates for the
      policy, values, and entropy losses and symmetry loss in (7) when using regularization.
    Perform backward pass on the combined sum
```

$$\widehat{L_P}(\theta, \lambda_{GAE}[t]) + c_V\cdot\widehat{L_V}(\theta) + c_{entropy}[\text{step}]\cdot\widehat{L}_{entropy}(\theta) + \lambda_{symmetry}\cdot\widehat{L}_{symmetry}(\theta).$$

```
    Update the parameters by performing one gradient step with learning rate = lr[step].
```

There are many hyperparameters in this algorithm: calibration of $c_V, \lambda_{GAE}, c_{entropy}, \lambda_{symmetry}$ coefficients in the loss calculation, learning rate scheduling, and the number of training steps:

– c_V is the coefficient of the value function and is typically set between 0 and 0.5; in our case, we chose $c_V = 0.1$, being constant throughout the training.

– $\lambda_{symmetry}$ controls the symmetry regularization strength; we chose the values empirically depending on the convergence dynamics.

– $c_{entropy}$ follows the dynamic annealing scheduling: it starts with 0.2, decreases to 0 as an arithmetic progression for the first half of the training, and stays at 0 for the remaining half. We remind the reader that higher $c_{entropy}$ penalizes the model for deterministic outputs $\pi_\theta(H_t)$, implying that the model is artificially encouraged for exploration.

– λ_{GAE} follows the dynamically rising scheduling: it starts with 0.3, increases to 1 as an arithmetic progression for the first half of training, and stays at 1 for the remaining half. We remind the reader that $\lambda_{GAE} = 1$ corresponds to the unbiased estimates for the gradient, and $\lambda_{GAE} < 1$ implies biased estimates with a lower variance, which decreases the influence of future rewards.

– The Learning rate in all experiments was tuned. It followed a warm-up and then a constant schedule with Adam Optimizer. In several experiments, we observed that the learning rate can affect the convergence outcome (i.e., converge / not converge to the optimum). Therefore, we did some basic learning rate exploration. We usually chose the lowest learning rate and the largest number of training steps when reporting the final findings, as those produced consistent results.

We experimented with different parameter dimensions (depth/width, number of attention heads) and were able to get to the optimum performance with a small-sized transformer (maximum dimensions sizes we tried were: N_embed = 256, N_layer = 12, n_head = 8.) One further direction for research is to investigate how the training convergence/asymmetry phenomenon (of the original transformer) develops with the model and data scale, which we did not have a chance to explore fully in our case. In particular, we would like to know how much deeper and wider the network should be to find the global optimum without any modifications. From the data perspective, increased batch size could also help the model to get out of asymmetric local minimum.

A.2 Symmetry modifications

A.2.1 Symmetrization

For a given model f_θ (e.g. transformer), *symmetrization* f_θ^S can be defined as:

$$f_\theta^S(H_t) \coloneqq \mathrm{Ave}_\sigma[\sigma^{-1}(f_\theta(\sigma(H_t)))|\forall\sigma : \{1,..,K\} \xrightarrow{\mathsf{B}} \{1,..,K\}], \tag{6}$$

i.e. we apply the $\sigma^{-1}(f_\theta(\sigma(\cdot)))$ operator on H_t for all possible permutations σ and average out results. It is easy to see that this definition makes the model f_θ^S symmetric: assume we have any permutation σ_0 and we want to check $f_\theta^S(H_t) \stackrel{?}{=} \sigma_0^{-1}(f_\theta^S(\sigma_0(H_t)))$, which is equivalent to the original symmetry condition (3). Substituting the definition of f_θ^S on the right side:

$$\sigma_0^{-1}(f_\theta^S(\sigma_0(H_t))) = \sigma_0^{-1}(\mathrm{Ave}_\sigma\left[\sigma^{-1}(f_\theta(\sigma(\sigma_0(H_t))))\right]) = \mathrm{Ave}_\sigma\left[\sigma_0^{-1}(\sigma^{-1}(f_\theta(\sigma(\sigma_0(H_t)))))\right]) =$$
$$\mathrm{Ave}_\sigma\left[(\sigma\circ\sigma_0)^{-1}(f_\theta((\sigma\circ\sigma_0)(H_t)))\right] = f_\theta^S(H_t),$$

where the second equation follows because Ave and σ_0^{-1} operators are interchangeable, the third equation simply combines σ and σ_0 into the composition of permutations, and last equation follows because $\sigma\circ\sigma_0$ still spans all permutations as σ did before.

There are $K!$ different permutations, so for large K this becomes computationally infeasible, and hence we restrict the permutations σ to some subgroup G_K. A natural choice of such G_K is the group of cyclic shifts, i.e. $\sigma_s(i) = i + s \mod K$ for $s = 0, ..., K-1$, and it serves our goal of eliminating bias towards any one of the arms: none of the arms will be dominate the others now. This is because if, say, original model f_θ favors arm 1, then large output probabilities for this arm 1 will be transferred to all other arms $i > 1$ after applying cyclical shift σ_{i-1} on the output f_θ^S.

A.2.2 Symmetry Regularization

With symmetrization, we "engineer" our prior knowledge about the optimal solution (namely that it should be symmetric) and change its architecture accordingly. Therefore, it remains unclear whether the original transformer is just incapable[2] of representing the optimal, symmetric solution, or it is capable of doing that, but it faces very non-convex optimization problem and gets stuck in a local minimum. Our second method addresses these concerns via **symmetry regularization**, which adds

[2]Remark: in theory, transformer model can approximate well any function if its architecture is big enough; here "incapable" refers to the abilities of a small-size transformer

the penalty component for asymmetric behaviors in the total combined loss, without any changes to the architecture / forward pass. An additional benefit of the regularization is the ability to control its strength (via the calibration of the penalizing λ constant) and its flexibility of what kind of beliefs can be reflected in our penalties (e.g., it is unclear how symmetrization can help when the optimal solution requires symmetry along only a subset of coordinates or even asymmetry). We provide a formal description in the Algorithm 2:

Algorithm 2: Symmetry Regularization

Perform the regular training procedure: roll out trajectories and calculate the policy, value, and entropy losses as previously. This step is unmodified and produces B final trajectories $\tau_i = a_0, r_0, a_1, r_1, ..., a_T, r_T$ (i = 1 ... B);

Choose a random subset τ_j of size [B / K] among τ_i ; `// K is the number of arms`

For each chosen τ_j, consider K augmented trajectories by applying all permutations from G_K:

$$\tau_{j,s} := \sigma_s(\tau_j) = ((a_0 + s) \mod K, r_0, (a_1 + s) \mod K, r_1, ..., (a_T + s) \mod K, r_T)$$

where $s = 0, ..., K-1$ ("s" for shift). There will be $[B/K] \cdot K \leq B$ trajectories overall ; `// all the actions are shifted by some number` s `modulo K and rewards are kept the same`

Stack these trajectories together in a single batch of size B. Perform **one forward pass** and obtain the policy output P: the tensor with the shape (B, T, K) ; `// the third dimension is` K `because it corresponds to` K `logits of the policy output`

For simplicity, we assume the first dimension can be indexed via pair (j, s) so that $P[(j,s),t] = f_\theta(\sigma_s(\tau_j))[t]$ – the result of the original model on permuted $\tau_{j,s} = \sigma_s(\tau_j)$ at time step t. Recall that each of $P[(j,s),t]$ is still a K-dimensional vector, not a scalar.

For each index j and time step t, calculate the symmetry loss $L_{j,t}$ as the variance between the *permuted back* versions of the outputs:

$$L_{j,t} = \text{Ave}\left[\text{Var}_{s=0...K}\{\sigma_s^{-1}(P[(j,s),t])\}\right] = \text{Ave}\left[\text{Var}_{s=0...K}\{\sigma_s^{-1}(f_\theta(\sigma_s(\tau_j))[t])\}\right],$$

where the variance inside is taken coordinate-wise and then Averaged out across those K coordinates to obtain the final scalar value.

Finally, This loss is summed up across time horizons and averaged over the batch, where discounting happens for the same reason as with value and entropy losses:

$$L_{symmetry} := \lambda_{symmetry} \cdot \frac{1}{B} \sum_{j=1}^{B} \sum_{t=1}^{T} \gamma^t L_{j,t}. \tag{7}$$

Return this loss and add it to the other components of the total loss before the gradient update step. $\lambda_{symmetry}$ is a hyperparameter controlling the strength of regularization.

One significant benefit of the regularization, as opposed to symmetrization, is computational efficiency: we perform **only one** additional forward pass, while the symmetrized transformer "needs" to perform K times more forward passes at every step t of the rollout trajectory (because we shift every input K times).

Due to these computational and memory constraints, we had to reduce the batch size of the symmetrized transformer from B to B/K, which effectively means that it learns from K times less of independent trajectories ("independent" means not the ones created by us via permutation). Surprisingly, the training dynamics, as presented in Figure 2 in the results section, shows that even then the symmetrized transformer has significantly more stable and faster convergence. This emphasizes that symmetry represents a very powerful inductive bias in our bandits problem with transformers.

A.3 Experiments

A.3.1 Comparison with other methods

In this section we focused on evaluating transformer's performance against other methods in the setting where all arm means followed Uniform prior, i.e., $\mu_i \overset{\text{i.i.d.}}{\sim} \text{Unif}(0,1)$. We take the results of [11] as a benchmark as their "SNAIL" meta-learner has achieved state-of-the-art performance in most of the settings for bandits. Following their experiments, we tested all combinations of $T = 10, 100$ and $K = 5, 10, 50$. Table 1 presents the results. We also present the convergence dynamics of all three models for horizon $T = 100$ (and all possible numbers of arms $K = 5, 10, 50$). We remind the reader that the empirical loss function in bandits is the discounted regret, $\sum_{t=0}^{T-1} \gamma^t(\mu_{t^*} - \mu_{a_t})$, where $\mu_{t^*} = \max_k\{\mu_k\}$ is the highest available of the arm means (note: the discounted regret also equals minus expected discounted reward up to an additive constant).

Methods						
Experiment (T, K)	Gittins (optimal with $T \to \infty$)	TS	SNAIL	Transformer	Transformer regularized	Transformer symmetrized
10, 5	6.6	5.7	**6.6 ± 0.1**	6.3 ± 0.1	**6.6 ± 0.1**	**6.6 ± 0.1**
10, 10	6.6	5.5	**6.7 ± 0.1**	6.4 ± 0.1	**6.7 ± 0.1**	**6.7 ± 0.0**
10, 50	6.5	5.2	**6.7 ± 0.1**	6.3 ± 0.0	6.4 ± 0.1	**6.7 ± 0.1**
100, 5	78.3	74.7	**79.1 ± 1.0**	77.8 ± 0.2	**78.1 ± 0.1**	**78.4 ± 0.3**
100, 10	82.8	76.7	**83.5 ± 0.8**	80.7 ± 0.1	**83.1 ± 0.1**	**83.5 ± 0.1**
100, 50	85.2	64.5	**85.1 ± 0.6**	81.2 ± 0.1	81.0 ± 0.1	**84.5 ± 0.5**

Table 1: Total undiscounted reward on multi-arm bandits. Estimates are given with 95% confidence intervals. For each experiment, we highlight all models whose performance is not statistically significantly different from the best model (based on a one-sided t-test with p = 0.05)

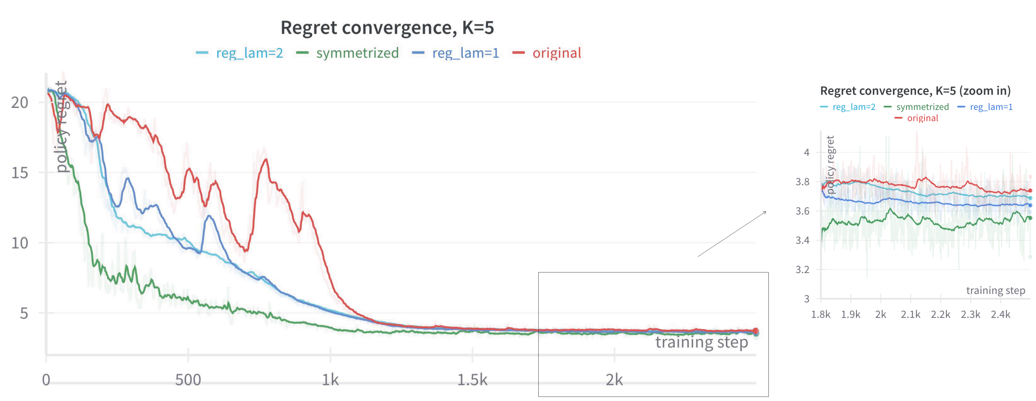

Figure 4: Training convergence for 5 arms, comparing 4 models: original transformer (red), symmetrized (green), and two regularized ones (blue) with different penalties. The right figure zooms in on the later stage of training (highlighted with a black rectangle) for closer comparison

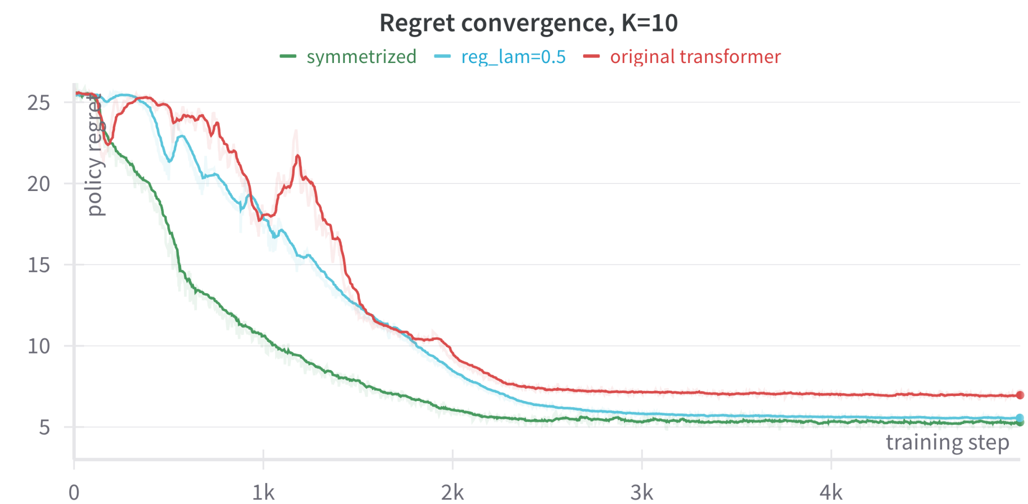

Figure 5: Training convergence for 10 arms

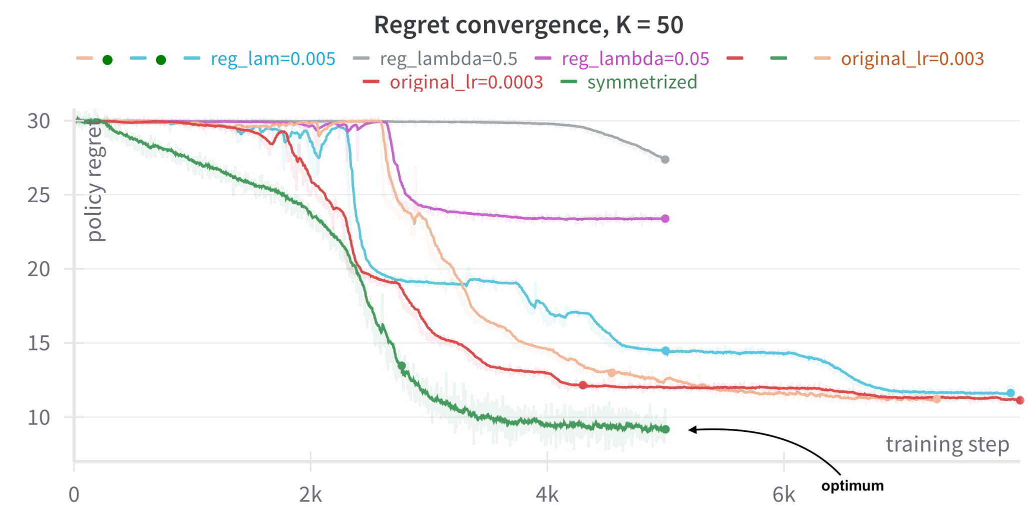

Figure 6: Training convergence for 50 arms. This problem was particularly hard for both the original and regularized transformers. Different curves show learning rate exploration and $\lambda_{symmetry}$ exploration. Symmetrized transformer still achieves the global optimum. (Note: the dots in the middle of some curves represent the runs that were interrupted and continued later)

Note that Regularization failed to work for $K = 50$. In general, this experiment turned out to be the hardest for both the original and regularized transformers, as both the table of rewards and the convergence plots indicate. Here, the regularized transformer converged very close to the original one, but the convergence was way longer and more unstable (light blue curve on the figure A.3.1). There are some interesting patterns that we do not understand about its convergence, for instance, a significant drop in regret next to the 2500th training step: observe that the light blue curve becomes close to red (original) and green (symmetrized). Further investigation is needed of what kind of learning happens in these moments.

The symmetrized transformer still achieved state-of-the-art performance (as we can judge from the cumulative rewards table), and had a stable convergence curve. Again, this is quite surprising because, in this experiment, it only learned from $\frac{B}{K} = \frac{5000}{50} = 100$ "independent" trajectories as opposed to 5000 that of the original transformer.

A.3.2 Direct comparison with Gittins

As mentioned in the experiments section, we noticed an interesting pattern on the diagonal of the heatmaps 3: the progression of difference scores from low μ-s to high μ-s. Seems like the model is almost identical to Gittins when arm means are high, but it struggles more with lower means. We believe that, to some extent, this is just the nature of the multi-armed bandit's problem: observing positive reward influences the model's actions "more deterministically" rather than observing negative reward. Namely, when both arm means are high (for instance, in the top right corner of the heatmap), Gittins pulls some arm, observes success, pulls it again, observes another success, etc. – continues pulling it until the end of the episode. It is reasonable to expect that the model will have the same behavior. However, the situation is different when both arm means are low (e.g., bottom left corner): in the beginning, we mostly see failures until the first couple of successes. One hypothesis could be that the model is not deterministic enough (recall, we do not force it to be so), i.e., maybe it has higher output chances for the correct arm, but it still gives some non-negligible chance for the incorrect one. When we change its decisions to be deterministic, the difference scores overall decrease, yet just slightly, and the diagonal pattern remains, so it does not explain the full picture.

A.4 Out-of-Distribution analysis (OOD)

We train the model on the bandits with arm means drawn from the Beta(10, 10) distribution. The density for this distribution is presented below (figure A.4). This is a symmetric distribution, which is very centered around the middle 0.5 point. In particular, the probability of being below $\frac{1}{7}$ or above $\frac{6}{7}$ is below 0.01%. Therefore, all regions on the boundary of the square in the previous heatmaps can be treated as OOD regions. The Gittins index, which changes its priors from Uniform to Beta, still remains optimal for the same goal of minimizing discounted regret when drawing bandits from this prior.

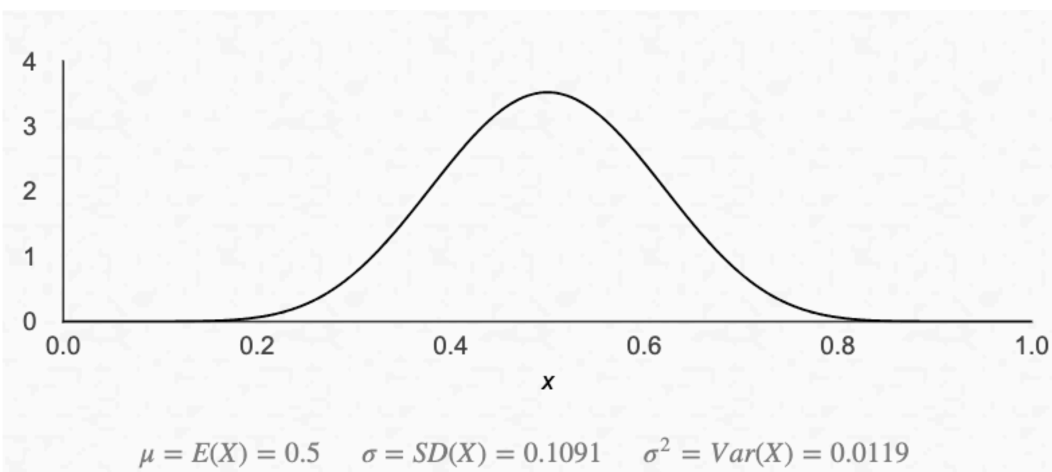

Figure 7: Beta(10, 10) density

Methods				
Gittins with Beta(10, 10) prior (optimal)	Thompson	Transformer	Transformer regularized	Transformer symmetrized
0.62 ± 0.10	0.84 ± 0.01	0.64 ± 0.01	0.638 ± 0.01	0.635 ± 0.01

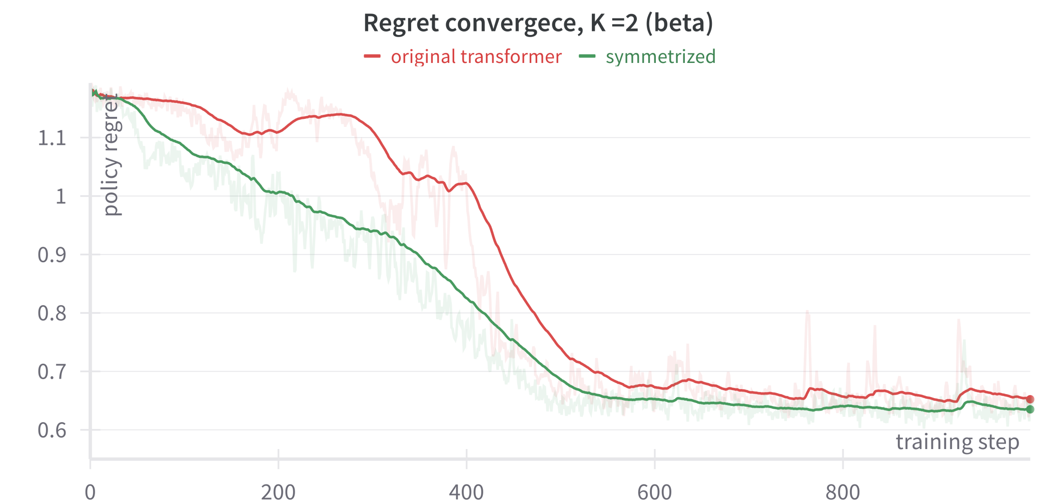

Figure 8: Training convergence with Beta(10, 10) priors: the original transformer (red) and symmetrized (green)

This time, we noticed that the gap between the original and symmetrized transformers was much smaller. This suggests that during regular training with Uniform priors, the original transformer is more prone to be greedy/biased towards one of the arms because of observing arms with high means. The lack of asymmetry is also demonstrated in the heat maps below (similar to the previous section). The asymmetry of the original transformer (on the left) is barely noticeable as the heatmap seems to be relatively symmetric with respect to the $\mu_1 = \mu_2$ diagonal.

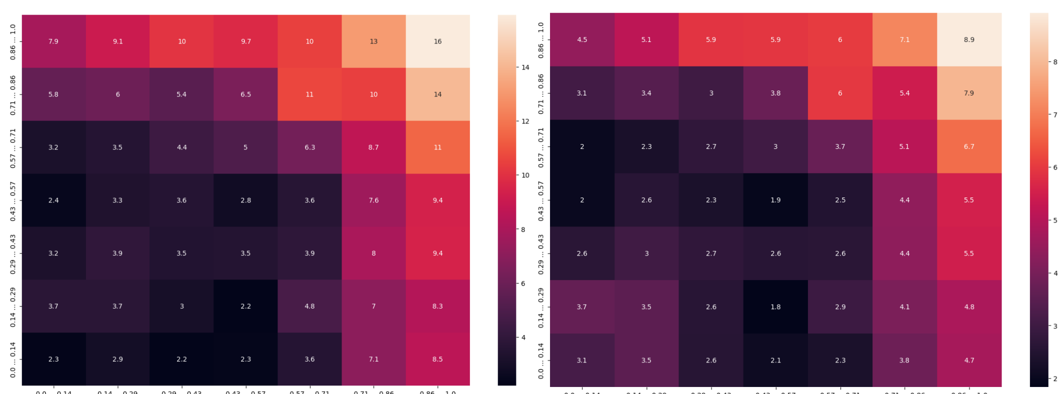

Figure 9: Heatmaps of difference scores with Gittins and Beta(10, 10) prior; the original (left) and symmetrized (right) transformers

Perhaps another surprising phenomenon is the pattern on the diagonal "flipped": the difference scores go up as the arm means become higher, and we also found that the model actually outperforms the Gittins index in that top right corner. One interpretation of this pattern is that the model became "less greedy" than the Gittins and explores more, which might pay off in the long run when dealing with arm means that are both high (close to 1).

In general, this experiment demonstrates that the fitted solution generalizes well out-of-distribution (since the model almost never observed any of the bandits in the regions of the heatmaps during the training). Hence, the transformer learned a quite effective in-context reinforcement learning strategy. In addition, we also see that the training data from the Uniform experiment seemingly pushes the transformer the transformer towards more greedy, asymmetric policies, whereas the more balanced Beta(10,10) training data encourages more exploratory solutions.

B Acknowledgements

LJ and NV were partially supported by NSF DMS-2134157.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We investigate the transformer's capabilities in Bayesian bandits, claim their optimal performance when trained in a pure RL manner (this is mentioned both in the abstract and introduction), and benchmark our results against the current state-of-the-art methods (mentioned in the introduction). We also investigate the asymmetry phenomenon (mentioned in the abstract).

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations of the training settings (model size/batch size limitations) are discussed in Appendix A.1. We also point out the limitations of the symmetrization method in its definition in appendix A.2 and symmetry regularization as empirically found for K = 50 in the appendix A.3.1. Major assumptions for our data (e.g., i.i.d. of bandits) are robust due to the synthetically generation procedure.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: The paper focuses mostly on the transformers' empirical performance, without theoretical analysis. The only theory includes the symmetrization/regularization justifications (in the appendix), which are supplementary and not central to the main contribution of the work.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All the major details are included in the Experiments section 3, and further discussion is in the appendix. The code has supportive explanations in the documentation of functions/training methods.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example

(a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide the anonymized GitHub link in the core part of the paper.The instructions for reproducing the training/results are included in the ReadMe file of that folder. Data was synthetically generated during the training.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so No is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: As the core part of the paper discusses, data was synthetically generated (we draw data from the desired distribution at every step of online RL training). All training details (hyperparameters, learning rate schedules, optimizer, etc.) are provided in the appendix A.1.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.

- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The major experiment comparing our models with other methods reports results with 95% confidence intervals as indicated in the table 1 of the Appendix A.3.1. Other quantitative results, such as difference scores comparing the transformer with Gittins in the heatmaps 3, were provided with Monte-Carlo estimates with statistical significance (though, the exact errors were omitted for better readability).

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Compute requirements were modest (just a single 80 GB GPU), which is discussed in the training details in the appendix A.1.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

Justification: We have reviewed the Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: The paper focuses on scientific findings with little immediate societal impact.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: we use synthetically generated data and our models are trained from scratch.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [NA]

    Justification: The only external source we use is the GPT-2 configuration from the Transformers library by Hugging Face, which is open-source software.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.