An Empirical Study of Neural Kernel Bandits

Michal Lisicki  
University of Guelph  
Vector Institute for AI  
mlisicki@uoguelph.ca  

Arash Afkanpour  
Google  
arashaf@google.com  

Graham W. Taylor  
University of Guelph  
Vector Institute for AI  
gwtaylor@uoguelph.ca  

Abstract

Neural bandits have enabled practitioners to operate efficiently on problems with non-linear reward functions. While in general contextual bandits commonly utilize Gaussian process (GP) predictive distributions for decision making, the most successful neural variants use only the last layer parameters in the derivation. Research on neural kernels (NK) has recently established a correspondence between deep networks and GPs that takes into account all the parameters of a NN and can be trained more efficiently than most Bayesian NNs. We propose to directly apply NK-induced distributions to guide an upper confidence bound or Thompson sampling-based policy. We show that NK bandits achieve state-of-the-art performance on highly non-linear structured data. Furthermore, we analyze practical considerations such as training frequency and model partitioning. We believe our work will help better understand the impact of utilizing NKs in applied settings.

1 Introduction

Contextual bandit algorithms, like upper confidence bound (UCB) [5] or Thompson sampling (TS) [25], balance exploration and exploitation with two terms: (1) a frequentist reward estimate and (2) an uncertainty term [15]. From a Bayesian perspective those terms represent the first two moments of a predictive posterior distribution. If little prior knowledge is available about the problem, this distribution can be chosen to be a Gaussian process (GP). GP-UCB [24] and GP-TS [7] are special cases of kernel bandits [26], in which the ridge regularizer is set to Gaussian noise to form a GP. In practical applications bandits are traditionally combined with well-known kernels, like linear, RBF or Matérn [22]. More recently a new family of neural kernels (NK) has emerged, which directly approximates the behavior of deep neural networks (NN) [6, 10]. The neural network Gaussian process (NNGP) kernel is derived from a forward pass of a NN, and corresponds to a random NN at initialization when combined with a GP [16]. The finding of a direct GP correspondence was later followed by establishing a similar relation for random linearized NNs and the neural tangent kernel (NTK), where NTK represents a network’s dynamics during gradient descent training [3, 13, 14, 17]. A review of all recent NK-GP models, with connections to other existing uncertainty models, like deep ensembles and randomized priors [21], can be found in [13]. NeuralUCB [29] and NeuralTS [28] are two related neural bandit policies that can be considered parameter space variants of NK bandits with a NTK-GP posterior [13], whose moments are defined by a standard NN estimate and the neural tangent features (NTF) covariance (Sec. B.5). These algorithms were shown to achieve a regret bound comparable to kernel bandits with the effective dimension induced by a corresponding NTK [29, 28]. The results presented in [29, 28], and later in [27, 19], however, show the algorithm to underperform in practical applications, which was linked to overparametrization and poor covariance approximation [27].

In this work we focus on the conditions in which NK bandits operating in the function space provide a competitive advantage over other neural bandit approaches. Even though NKs have been shown to lack the full representational power of the corresponding NNs, they outperform finite fully-connected
Algorithm 1

Require: Number of arms $k$, number of rounds $T$, kernel regularization parameter $\gamma$, exploration parameter $\eta$, NNGP kernel function $K(\cdot, \cdot)$, neural tangent kernel function $\Theta(\cdot, \cdot)$, initial number of steps $\iota$, policy $\pi$

Play each arm sequentially for $\iota$ steps to accumulate data $X_a \in \mathbb{R}^{\iota \times d}, y_a \in \mathbb{R}^\iota \forall a$

for round $t = 1, 2, \ldots, T$ do

Observe context $x_t$

for arm $a = 1, 2, \ldots, k$ do

$K_{a,t} \leftarrow K(X_a, X_a)$

$\Theta_{a,t} \leftarrow \Theta(X_a, X_a) + \gamma I$

// Calculate predictive distribution moments for all arms

$\mu_{a,t} \leftarrow \Theta(x_t, X_a)\Theta_{a,t}^{-1}y_a$

$\sigma_{a,t} \leftarrow \sqrt{K(x_t, x_t) + \Theta(x_t, X_a)\Theta_{a,t}^{-1}K_{a,t}\Theta_{a,t}^{-1}\Theta(x_t, x_t) - 2\Theta(x_t, X_a)\Theta_{a,t}^{-1}K_{a,t}K(x_a, x_t)}$

if $\pi$ is UCB then

$p_{a,t} \leftarrow \mu_{a,t} + \frac{\eta}{\gamma \iota^2} \sigma_{a,t}$

else if $\pi$ is TS then

$p_{a,t} \sim \mathcal{N}(\mu_{a,t}, \frac{\eta}{\gamma \iota^2} \sigma_{a,t})$

end if

end if

Choose $a_t \leftarrow \arg\max_a p_{a,t}$ and obtain reward $y_t$

Update $(X_{a_t}, y_{a_t})$ with $(x_t, y_t)$

end for
3 Empirical Results and Discussion

While NK bandits were conceptually proposed in the past [29, 28], they are not yet considered a reliable method for sequential decision making. We conduct experiments to show that NKs provide a competitive approach in bandit settings. We begin by evaluating our method within a Bayesian bandit framework [23] and present our main result w.r.t. performance of related approaches. We commit the subsequent subsections to measure the implications of practical implementation considerations.

3.1 NK bandits outperform neural-linear and NTF bandits on complex datasets

We tested the performance of NK bandits on a set of UCI datasets [11] commonly used in the bandits literature [23]. The details of our experimental setup can be found in Sec. A.2. NK bandits tend to achieve higher performance for higher-dimensional (over 50 features) non-linear datasets that correspond to classification tasks (Fig. 1). We attribute the increase in performance to the high accuracy of exploitation, achieved through the application of NTK to non-linear structured data problems [4], and exploration, achieved through the utilization of GP predictive distribution linked directly to the behavior of a deep neural network [17].

3.2 Training frequency

Frequent updates of complex models in bandit settings put high demand on compute resources. We measure the sensitivity of our method to the number of actions taken between subsequent model updates. We ran our NK bandit for numbers of actions specified in the recent literature [19, 23]: {1, 5, 20, 100, 400}, and report their respective cumulative rewards on all the datasets after 5000 steps. Fig. 2 shows a smooth degradation in performance as the number of actions increases, which is expected. We note, however, that the degradation is relatively small and preserves competitive performance on all tested datasets. As large numbers of actions help to save significant amounts of computational resources, we believe that the sustained performance provides a strong argument for applicability to real world scenarios. It also positions our method as a good candidate for a limited memory approach, which we leave for future work. For some datasets, we observe a significant spike in performance at 5 actions before retraining. The result suggests an additional role the training frequency plays in the exploration-exploitation trade-off. In Fig. 2 we also show the results of the best performing models, other than our method, as reported in Fig. 1. We observe that for some datasets of higher complexity (e.g. covertype), our method’s rank w.r.t. other algorithms is not affected, even with large numbers of actions before retraining, which additionally confirms its robustness.
3.3 Joint model

Figure 3: Cumulative rewards w.r.t. running times of a joint and a disjoint variant of our method.

Following the zero-padding approach (Sec. 2), we compared the performance of joint and disjoint models and noticed a considerable improvement in the last epoch performance for linear and stochastic datasets (financial and mushroom), slight improvement for datasets of medium complexity, and a large deterioration for some of the most difficult datasets — covertype and jester. When examined from a resource usage perspective, however, the joint model underperforms considerably. This result is significant, as most neural bandits use a joint (NeuralUCB, NeuralTS) or partially joint (neural-linear) model. It is important to note that in functional space a disjoint model reduces the overall number of kernel entries, while in the case of parameter space models we observe the opposite effect. We can only create a disjoint model at the expense of introducing additional \((k-1)|\theta|\) number of parameters, which puts a burden on resources. Our result highlights this difference, and can help practitioners to assess the impact of choosing joint or disjoint models when using NK bandits.

Acknowledgments and Disclosure of Funding

The authors would like to thank Carlos Riquelme, Blair Bilodeau, Angus Galloway, and Magdalena Sobol for discussions and support in reviewing this manuscript. The authors thank the Canada Foundation for Innovation and Compute Canada for computing resources.

References

[1] Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. In International Conference on Machine Learning, pages 127–135. PMLR, 2013.

[2] Robin Allesiardo, Raphael Feraud, and Djallel Bouneffouf. A Neural Networks Committee for the Contextual Bandit Problem. arXiv:1409.8191 [cs], September 2014. URL http://arxiv.org/abs/1409.8191. arXiv: 1409.8191.

[3] Sanjeev Arora, Simon S. Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On Exact Computation with an Infinitely Wide Neural Net. arXiv:1904.11955 [cs, stat], November 2019. URL http://arxiv.org/abs/1904.11955. arXiv: 1904.11955.

[4] Sanjeev Arora, Simon S. Du, Zhiyuan Li, Ruslan Salakhutdinov, Ruosong Wang, and Dingli Yu. Harnessing the Power of Infinitely Wide Deep Nets on Small-data Tasks. arXiv:1910.01663 [cs, stat], October 2019. URL http://arxiv.org/abs/1910.01663. arXiv:1910.01663.

[5] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time Analysis of the Multiarmed Bandit Problem. Machine Learning, 47(2):235–256, May 2002. ISSN 1573-0565. doi: 10.1023/A:1013689704352. URL https://doi.org/10.1023/A:1013689704352.

[6] Youngmin Cho and Lawrence Saul. Kernel Methods for Deep Learning. In Advances in Neural Information Processing Systems, volume 22. Curran Associates, Inc., 2009. URL https://papers.nips.cc/paper/2009/hash/5751e3e9a4feab575962e78e006250d-Abstract.html.

[7] Sayak Ray Chowdhury and Aditya Gopalan. On Kernelized Multi-armed Bandits. arXiv:1704.00445 [cs], May 2017. URL http://arxiv.org/abs/1704.00445. arXiv: 1704.00445.

[8] Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual bandits with linear payoff functions. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, pages 208–214. JMLR Workshop and Conference Proceedings, 2011.
[27] Pan Xu, Zheng Wen, Handong Zhao, and Quanquan Gu. Neural Contextual Bandits with Deep Representation and Shallow Exploration. arXiv:2012.01780 [cs, stat], December 2020. URL http://arxiv.org/abs/2012.01780 arXiv: 2012.01780.

[28] Weitong Zhang, Dongruo Zhou, Lihong Li, and Quanquan Gu. Neural Thompson Sampling. arXiv:2010.00827 [cs, stat], October 2020. URL http://arxiv.org/abs/2010.00827 arXiv: 2010.00827.

[29] Dongruo Zhou, Lihong Li, and Quanquan Gu. Neural Contextual Bandits with UCB-based Exploration. arXiv:1911.04462 [cs, stat], July 2020. URL http://arxiv.org/abs/1911.04462 arXiv: 1911.04462.

A Experimental setup

A.1 Datasets

Table 1 provides an summary of the key properties associated with each of the UCI datasets we considered. While [23] and [19] used the same datasets, there are some subtle differences in the way that they were implemented. More details are in Sec. A.2.

| Dataset | Description | Distinguishing factor(s) | \( d \) | \( k \) | \( n \) | Reward | Context |
|---------|-------------|--------------------------|---|---|---|---|---|
| Adult   | predict income based on personal information | binary classification | 13 | 2 | 50k | binary/categorical |
| Census  | predict occupation based on personal information | multivariate classification; large number of features | 67 | 9 | 250k | categorical/integer |
| Covertype| type of forest coverage in various areas | multivariate classification; requires non-linear models | 54 | 7 | 150k | categorical/integer |
| Financial | stock prices from NYSE and NASDAQ | linear problem; easy to over-explore | 21 | 8 | 4k | continuous |
| Jester  | joke recommender system | inputs and outputs in the same domain | 32 | 8 | 20k | continuous |
| Mushroom| poisonous vs. safe mushrooms | imbalanced stochastic rewards; captures aspects of classification and regression | 117 | 2 | 10k | continuous/categorical |
| Statlog | space shuttle flight indicators | one arm optimal 80% of the time; requires non-linear models | 9 | 7 | 45k | binary/integer |

A.2 Experimentation details

We primarily compare our method against the state-of-the-art neural-linear [23] and neural-linear LiM2 [19]. We followed the experimental setup of [23] and [19] and report the average and standard deviation of the cumulative reward over 10 runs. We also include important baselines reported in [23], linear TS and multitask GP, due to their significant performance on simpler datasets. Whenever applicable, we used the hyperparameters reported in [23]. Due to differences in number of layers, and units per layer, between [23] and [19], in Tab. 2 we report the results for both settings. We used the neural-tangents library [20] to obtain the NTK, NNGP kernel, and the mean and covariance of the predictive posterior. We computed the NKS based on a two-layer fully-connected architecture with ReLU activations and regularizer \( \gamma = 0.2 \). NKS were then used to compute a GP, as presented in [17], later referred to in [13] as the randomized prior approach [21]. We tested both UCB and TS policies with the exploration parameter \( \eta \) set to 0.1. Even though we compare primarily against TS approaches, we add UCB for reference. We note that removing UCB does not change the overall rank of NK approaches with respect to other methods in places where NK achieved the highest performance. As established in [19], normalized cumulative rewards are computed w.r.t. baseline uniform sampling and the best algorithm for each dataset: 

\[
\text{norm\_cum\_rew}_{\text{alg}} = \frac{\text{cum\_rew}_{\text{alg}} - \text{cum\_rew}_{\text{uniform}}}{\text{cum\_rew}_{\text{best}} - \text{cum\_rew}_{\text{uniform}}}.
\]

We complement our results by reporting the average (over 10 runs) times per round that our algorithms took w.r.t. the neural-linear approach (Tab. 3). As the times grow with increasing number of collected data points, we report the shortest, median, and the longest average round. Both TS and UCB variants of our method tend to be about 5 times slower than the LiM2 approach, taking around 2 seconds for

\[\text{For adult and census datasets the categorical features are subsequently encoded with one hot vectors, resulting in } \sim 80 \text{ features for adult and } \sim 370 \text{ features for census. The numbers vary due to subsampling.}\]
We denote the training data by $\{(x_t, y_t)\}$. Alternatively, the data points can be indexed by time $\forall k \in \{1, \ldots, n\}$.

The input data is composed of $\{x_t\}$, where $t \in [T]$ when we talk about a predetermined set. Alternatively, the data points can be indexed by time $t \in [T]$ in the context of bandits, where the data is collected sequentially. Parameter vectors are denoted by $\theta$ and the reward estimation models by $f: \mathcal{X} \to \mathcal{Y}$. The dimensionality of both the feature vector and the parameter vector is denoted by $d$. Depending on the parameters and the nature of the data, $\mathcal{X}$ and $\mathcal{Y}$ can be difficult to compute. Therefore in practice we often resort to sampling (hence “Thompson sampling”).

At each step the TS policy calculates the posterior over the parameters $\phi(\mathbf{X}, \mathbf{Y})$. The role of a policy $\pi$ is then to select an action $a \in [k]$ and observe the associated reward $y_t,a$. We name the triplet $(x_t,a) \sim \pi(a|x_t)$ an observation at time $t$. The theoretical objective is to minimize (instantaneous) regret $R_T = \mathbb{E}\left[\sum_{t=1}^T \max_a y_{t,a} - \sum_{t=1}^T y_{t,a}\right]$. In practical scenarios, however, where no information is available about the optimal reward, we often resort to reporting just the cumulative reward $\sum_{t=1}^T y_{t,a}$. Thompson sampling (TS) is a policy that operates on the Bayesian principle. The reward estimates for each arm are computed in terms of a predictive distribution $p(y_t|a,D_a,\theta) = \int p(y_t|\theta)p(\theta|D_a)d\theta$. Depending on the parameters and the nature of the data, this integral can be difficult to compute. Therefore in practice we often resort to sampling (hence “Thompson sampling”).

At each step the TS policy calculates the posterior over the parameters $p(\theta|D_a)$ and then collects a single parameter sample $\theta_a \sim p(\theta|D_a)$ per action. Then, instead of computing a full predictive posterior, the algorithm chooses actions according to the expected value $\mathbb{E}[y_t|\theta_a]$. In practice $p(\theta|D_a)$ can be difficult to derive analytically. For this reason, the prior and the likelihood are usually chosen to be Gaussian, which results in a closed form solution for $p(\theta|D_a)$ that also becomes Gaussian. Linear TS \textbf{[2]} (Alg \textbf{[2]}) is a contextual variant of TS, in which $\mathbb{E}[y_t|\theta_a] = x_t^T \theta_a$. A Gaussian formulation allows for straightforward computation of its parameters, covariance $\Sigma = \sum_t x_t x_t^T$ and the response vector $b = \sum_t x_t y_t$, by aggregating over the past epochs.

Instead of sampling parameters, TS can also be formulated such that rewards are sampled directly from the predictive distribution $\mathcal{N}(\mu, \sigma^2)$, which can be defined through Bayesian inference, or formulated

| Table 2: Neural-linear methods compared over two significant hyperparameter settings. |
|-----------------------------------------------|----------------|-----------|---------|---------|---------|
| adult | census | covertype | financial | jester | mushroom | statlog |
| NTK-TS $L = 1$, $\gamma = 0.2$ | 4119 ± 16 | 3152 ± 41 | 3402 ± 41 | 4407 ± 44 | 16779 ± 1234 | 4052 ± 4974 | 4720 ± 132 |
| NTK-TS $L = 2$, $\gamma = 0.2$ | 4113 ± 16 | 3186 ± 29 | 3441 ± 36 | 4162 ± 276 | 10856 ± 1212 | 2016 ± 3677 | 4820 ± 32 |
| NeuralLinearTS $L = 1$, $\eta_1 = 50$ | 3970 ± 36 | 2557 ± 54 | 2698 ± 76 | 4358 ± 363 | 12552 ± 1875 | 11111 ± 336 | 4771 ± 13 |
| NeuralLinearTS $L = 2$, $\eta_1 = 100$ | 3927 ± 26 | 2223 ± 181 | 2588 ± 74 | 4196 ± 350 | 11542 ± 2365 | 10516 ± 345 | 4801 ± 7 |
| NeuralLinearTS-LiM2 $L = 1$, $n_1 = 50$ | 4044 ± 17 | 2776 ± 36 | 2745 ± 141 | 4485 ± 360 | 14333 ± 1339 | 10962 ± 1070 | 4816 ± 66 |
| NeuralLinearTS-LiM2 $L = 2$, $n_1 = 100$ | 3993 ± 39 | 2304 ± 217 | 2748 ± 85 | 4357 ± 366 | 13235 ± 1005 | 10915 ± 631 | 4872 ± 14 |

Table 3: Times per epoch [sec] (min / median / max) rounded to 2 significant figures.

| Table 3: Times per epoch [sec] (min / median / max) rounded to 2 significant figures. |
|-----------------------------------------------|----------------|-----------|---------|---------|---------|
| adult | census | covertype | financial | jester | mushroom | statlog |
| NTK-TS | 0.0 / 0.8 / 2.25 | 0.00 / 0.07 / 1.93 | 0.00 / 0.04 / 2.47 | 0.00 / 0.12 / 1.79 | 0.01 / 0.14 / 2.66 | 0.08 / 0.82 / 2.66 |
| NTK-UCB | 0.0 / 0.8 / 1.85 | 0.00 / 0.07 / 1.93 | 0.00 / 0.04 / 2.47 | 0.00 / 0.12 / 1.79 | 0.01 / 0.14 / 2.66 | 0.08 / 0.82 / 2.66 |
| JointNTK-TS | 0.07 / 0.91 / 5.68 | 0.00 / 0.07 / 1.93 | 0.00 / 0.04 / 2.47 | 0.00 / 0.12 / 1.79 | 0.01 / 0.14 / 2.66 | 0.08 / 0.82 / 2.66 |
| JointNTK-UCB | 0.07 / 0.91 / 5.68 | 0.00 / 0.07 / 1.93 | 0.00 / 0.04 / 2.47 | 0.00 / 0.12 / 1.79 | 0.01 / 0.14 / 2.66 | 0.08 / 0.82 / 2.66 |
| NeuralLinearTS | 0.00 / 0.01 / 1.2 | 0.00 / 0.01 / 1.2 | 0.00 / 0.01 / 1.2 | 0.00 / 0.01 / 1.2 | 0.00 / 0.01 / 1.2 | 0.00 / 0.01 / 1.2 |
| NeuralLinearTS-LiM2 | 0.02 / 0.03 / 0.33 | 0.04 / 0.1 / 0.36 | 0.03 / 0.06 / 0.36 | 0.04 / 0.08 / 0.38 | 0.03 / 0.09 / 0.37 | 0.02 / 0.03 / 0.32 | 0.03 / 0.08 / 0.36 |

B Theoretical background

B.1 Mathematical notation

We denote the training data by $(X, y) \in (\mathcal{X}, \mathcal{Y})$, and the test inputs with predictions by $(x_s, y_s)$. The input data is composed of $(x_t, y_t)$, where $t \in [n]$ when we talk about a predetermined set. Alternatively, the data points can be indexed by time $t \in [T]$ in the context of bandits, where the data is collected sequentially. Parameter vectors are denoted by $\theta$ and the reward estimation models by $f: \mathcal{X} \to \mathcal{Y}$. The dimensionality of both the feature vector and the parameter vector is denoted by $d$.

The role of a policy $\pi$ is then to select an action $a \in [k]$ and observe the associated reward $y_t,a$. We name the triplet $(x_t,a) \sim \pi(a|x_t)$ an observation at time $t$. The theoretical objective is to minimize (instantaneous) regret $R_T = \mathbb{E}\left[\sum_{t=1}^T \max_a y_{t,a} - \sum_{t=1}^T y_{t,a}\right]$. In practical scenarios, however, where no information is available about the optimal reward, we often resort to reporting just the cumulative reward $\sum_{t=1}^T y_{t,a}$. Thompson sampling (TS) is a policy that operates on the Bayesian principle. The reward estimates for each arm are computed in terms of a predictive distribution $p(y_t|a,D_a,\theta) = \int p(y_t|\theta)p(\theta|D_a)d\theta$. Depending on the parameters and the nature of the data, this integral can be difficult to compute. Therefore in practice we often resort to sampling (hence “Thompson sampling”). At each step the TS policy calculates the posterior over the parameters $p(\theta|D_a)$ and then collects a single parameter sample $\theta_a \sim p(\theta|D_a)$ per action. Then, instead of computing a full predictive posterior, the algorithm chooses actions according to the expected value $\mathbb{E}[y_t|\theta_a]$. In practice $p(\theta|D_a)$ can be difficult to derive analytically. For this reason, the prior and the likelihood are usually chosen to be Gaussian, which results in a closed form solution for $p(\theta|D_a)$ that also becomes Gaussian. Linear TS \textbf{[1]} (Alg \textbf{[1]}) is a contextual variant of TS, in which $\mathbb{E}[y_t|\theta_a] = x_t^T \theta_a$. A Gaussian formulation allows for straightforward computation of its parameters, covariance $\Sigma = \sum_t x_t x_t^T$ and the response vector $b = \sum_t x_t y_t$, by aggregating over the past epochs.

Instead of sampling parameters, TS can also be formulated such that rewards are sampled directly from the predictive distribution $\mathcal{N}(\mu, \sigma^2)$, which can be defined through Bayesian inference, or formulated...
Algorithm 2 Linear Thompson sampling

Set $\Sigma_a = I_d, b_a = 0_d \quad \forall a$

for round $t = 1, 2, \ldots, T$ do
  Observe context $x_t$
  for arm $a = 1, 2, \ldots, k$ do
    $\hat{\mu}_a \leftarrow \Sigma_a^{-1} b_a$
    Sample parameter $\hat{\theta}_a \sim N(\mu_a, \sigma^2 \Sigma_a^{-1})$
  end for
  Choose $a_t \leftarrow \text{arg max}_a x_t^T \hat{\theta}_a$ and get reward $y_t$
  Update posterior:
  $\Sigma_a \leftarrow \Sigma_a + x_t x_t^T$
  $b_a \leftarrow b_a + x_t y_t$
end for

Algorithm 3 Linear UCB

Set $\Sigma_a = I_d, b_a = 0_d \quad \forall a$

for round $t = 1, 2, \ldots, T$ do
  Observe context $x_t$
  for arm $a = 1, 2, \ldots, k$ do
    $\hat{\theta}_a \leftarrow \Sigma_a^{-1} b_a$
    $\text{UCB}_{t,a} \leftarrow x_t^T \hat{\theta}_a + \eta \sqrt{x_t^T \Sigma_a^{-1} x_t}$
  end for
  Choose $a_t \leftarrow \text{arg max}_a \text{UCB}_{t,a}$ and get reward $y_t$
  Update posterior:
  $\Sigma_a \leftarrow \Sigma_a + x_t x_t^T$
  $b_a \leftarrow b_a + x_t y_t$
end for

independently, as in [17]. In our experiments we use the former, two-stage sampling process, for weight-space view algorithms, like neural-linear or NeuralUCB/TS, and the latter for function space (kernelized) algorithms, like multitask GPs or our method. While TS offers an explicit Bayesian interpretation, other stochastic policies can take advantage of the inference process in a similar way to model the reward distributions of each arm. The upper confidence bounds (UCB) algorithm [5] becomes highly related to TS when examined from Bayesian perspective. In each step of the UCB policy, the actions are chosen according to the highest reward estimate $a_t \leftarrow \text{arg max}_a \text{UCB}_{t,a}$. The reward estimates of each arm are updated at each round as $\text{UCB}_{t,a} = \mu_a + \sigma_a$, resulting in an exploration-exploitation trade-off between picking arms that give the best results based on the history of collected rewards (large $\mu_a$; e.g. $\mu_a = \bar{y}_a$) and the ones we are most uncertain about (large $\sigma_a$). From a Bayesian perspective the parameters $\mu_a$ and $\sigma_a$ correspond to moments of the predictive posterior $p(y_a | a, D, \theta_a)$. In this case it is advantageous to derive the predictive density in its full form. In fact, we can see in the linear UCB (aka LinUCB [8] [18]) algorithm (Alg. 3) below that UCBS for each action are calculated using exactly the mean $\mu_a = x_t^T \Sigma_a^{-1} X^T y = x_t^T \hat{\theta}$ and the standard deviation $\sigma_a = \sqrt{x_t^T \Sigma_a^{-1} x_t}$ of a normal distribution. It also includes the exploration parameter, denoted $\eta$, which can be optimized analytically w.r.t. a particular regret bound, or set by the user as a hyperparameter. In this work we distinguish between the exploration parameter $\eta$ and a regularizer that captures the randomness coming from the data and the parameters $\gamma = \frac{\sigma_y^2}{\sigma^2}$. In practical applications, however, this is of little importance, as $\sigma_y$ is often set to 1, in which case $\gamma$ can be combined with $\eta$ to form a single hyperparameter.

A contextual bandit can be either joint or disjoint. A joint model (e.g. [8]) uses a single model that makes predictions based on action-specific contexts, while a disjoint model (Alg. 2 and Alg. 3) uses a separate model per action, which is more suited for a global context scenario.
We mainly compare our model to two baselines — neural-linear, given by [23], and its limited
memory version by [19]. Neural-linear can be thought of as a mid-point between frequentist
and Bayesian methods. The model operates on the basis of Bayesian feature regression, with features
\phi(x) provided by the penultimate layer of the neural network. As in linear Thompson sampling the
parameters are sampled according to a posterior \( p(\theta | \Phi, y) \). Instead of setting the prior on likelihood
parameters manually, those models use another classic approach to Thompson sampling and apply a
conjugate prior for the parameters of the likelihood.

\[
p(\theta, \sigma^2_y) = p(\theta | \sigma^2_y)p(\sigma^2_y) = \mathcal{N}(\mu, \sigma^2_0\Sigma^{-1})IG(a, b)
\]

The neural-linear approach with conjugate priors was shown to obtain good empirical performance
[23], most likely due to more aggressive initial exploration and higher final accuracy of the model.
The limited memory version by [19] uses likelihood matching to preserve that performance, while
greatly reducing resource usage.

As any contextual bandit, neural bandits can be disjoint or joint. A disjoint model uses a separate NN
per action. A joint model is instead conditioned on an action, which can be achieved, for example, by
means of modulation, input concatenation or input padding.

### B.3 Non-linear bandits

KernelUCB [26] is a "kernelized" version of LinUCB [8], designed to perform efficiently in a \( p >> n \)
situation, where \( n \) is the number of collected data samples, and \( p \) is the dimensionality of data features.
The derivation is based on kernel feature regression: \( y = \phi(x)^T \theta \), with \( \phi(x) \) providing a mapping
to high-dimensional (possibly infinite-dimensional) space, also known as the reproducing kernel
Hilbert space (RKHS). Given the data \( (X, y) \) and a regularization parameter \( \gamma \), the analytical solution
to the feature regression model is given by:

\[
\hat{\theta} = (\phi(X)^T\phi(X) + \gamma I)^{-1}\phi(X)^Ty.
\]

If \( p \) is large, so is \( \phi(X)^T\phi(X) \), and computing its inverse is computationally heavy or impossible (e.g. when
\( p = \infty \)). The kernel trick is to use the identity \( (\Phi^T\Phi + \gamma I)^{-1}\Phi^Ty = \Phi^T(\Phi\Phi^T + \gamma I)^{-1} \)
to convert the \( p \times p \) covariance matrix \( \Sigma = \Sigma(X, X) = \phi(X)^T\phi(X) \) into a smaller \( n \times n \) kernel matrix
\( K = K(X, X) = \phi(X)\phi(X)^T \). In the case where the regularizer represents the model's noise,
\( \gamma = \sigma^2_n \), the resulting feature regression becomes a Gaussian process regression [26] [24], which has
a predictive posterior given by:

\[
p(y_*|x_*, D) = \mathcal{N}(\hat{K}(x_*, X)(\hat{K}(X, X) + \gamma I)^{-1}y, \sigma^2_n(\hat{K}(x_*, Xs) - \hat{K}(x_*, X)(\hat{K}(X, X) + \gamma I)^{-1}\hat{K}(X, Xs))).
\]

UCB algorithms commonly use the predictive posterior parameters to derive a tight upper bound on
the regret. We can, for example, specify UCB in terms of the mean and the standard deviation as follows:

\[
\mu \gets \hat{K}(x_*, X)(\hat{K}(X, X) + \gamma I)^{-1}y
\]

\[
\sigma \gets \eta\sigma_n\sqrt{\hat{K}(x_*, Xs) - \hat{K}(x_*, X)(\hat{K}(X, X) + \gamma I)^{-1}\hat{K}(X, Xs)}.
\]

### B.4 Neural bandits

Neural bandits operate on the premise of feature regression. Even though any type of network is
permissible, the prevalent architectures in the literature include: neural greedy, neural-linear, ensemble
models [2], Bayesian neural networks, and their approximations [23][9]. In all those approaches the
input \( x \) is first propagated through a neural network \( f(x) \) to yield learned features. The neural-greedy
(e.g. the base model in [2]) approach uses predictions of a NN directly to choose the next action. It
is considered a naive approach, tantamount to exploitation in feature space. An alternative is a full
Bayesian neural network. A comparison of Bayesian NNs that can be used to model a predictive
posterior in the Thompson sampling setup can be found in [23]. One of the key findings of [23] is
that current Bayesian NNs, while providing better estimates than regular networks, are often too slow
to operate efficiently as a subroutine for contextual bandits, while Bayesian approximation methods,
like dropout [9], underperform in practical scenarios.

We mainly compare our model to two baselines — neural-linear, given by [23], and its limited
memory version, given by [19]. Neural-linear can be thought of as a mid-point between frequentist
and Bayesian methods. The model operates on the basis of Bayesian feature regression, with features
\( \phi(x) \) provided by the penultimate layer of the neural network. As in linear Thompson sampling the
parameters are sampled according to a posterior \( p(\theta | \Phi, y) \). Instead of setting the prior on likelihood
parameters manually, those models use another classic approach to Thompson sampling and apply a
conjugate prior for the parameters of the likelihood.

The neural-linear approach with conjugate priors was shown to obtain good empirical performance
[23], most likely due to more aggressive initial exploration and higher final accuracy of the model.
The limited memory version by [19] uses likelihood matching to preserve that performance, while
greatly reducing resource usage.

As any contextual bandit, neural bandits can be disjoint or joint. A disjoint model uses a separate NN
per action. A joint model is instead conditioned on an action, which can be achieved, for example, by
means of modulation, input concatenation or input padding.

### B.5 Neural kernels

In this work we define a neural kernel to be any compositional kernel [6][10] that aims to mimic
a state or behaviour of a neural network. A form of the compositional kernel was analytically
where $K$

As we use only ReLU in our work, Eq. 5 applies directly in kernel derivations for our experiments.

The general form of NTK is given by [14, 3]:

\[ \text{RKHS corresponding to a network's gradients, further referred to as the neural tangent kernel (NTK).} \]

optimization process, when examined from functional perspective, can be captured by a kernel with dynamics of the network during GD training. Jacot et al. [14] showed that the gradient descent (GD) and allows for training by means of Bayesian inference. A related approach is to model the full

Regularized, the resulting formulas apply for composing the NNGP (aka conjugate) kernel [16, 3]:

\[ \text{following}\ [16], \text{but uses}\ [6]'s\ analytical\ derivation\ for\ ReLU\ activation.\ \text{The\ following\ general} \]

\[ \text{is\ a\ function\ specifying\ the\ angular\ dependence,}\ \text{and}\ \alpha\ \text{is\ the\ angle\ between\ inputs\ in\ RKHS.} \]

The line of work on NKs was recently extended [16] by (1) placing it in a GP framework, (2) adding the parameters controlling the variances of weights and biases of the neural network, and (3) deriving a numerical method for approximating the neural kernel to any well-behaved activation function. In our experiments we used the neural-tangents library [20], which allows for full GP inference, following [16], but uses [6]'s analytical derivation for ReLU activation. The following general formulas apply for composing the NNGP (aka conjugate) kernel [16, 3]:

\[ \Sigma^{(l)}(x, x') = x^T x' \]

\[ \Lambda^{(l)}(x, x') = \begin{pmatrix} \Sigma^{(l-1)}(x, x') & \Sigma^{(l-1)}(x, x') \\ \Sigma^{(l-1)}(x', x) & \Sigma^{(l-1)}(x', x') \end{pmatrix} \]

\[ \Sigma^{(l)}(x, x') = \sigma^2 E_{(u,v) \sim GP(0,\Lambda^{(l)})} [\phi(u)^T \phi(v)]. \]

As we use only ReLU in our work, Eq. 5 applies directly in kernel derivations for our experiments. Regularized, the resulting NNGP has the following moments:

\[ \mathbb{E}[y_*] = \mathcal{K}(x_*, X)(\mathcal{K}(X, X) + \gamma I)^{-1} y \]

\[ \mathbb{Var}[y_*] = \sigma^2 (\mathcal{K}(x_*, x_*) - \mathcal{K}(X, X)(\mathcal{K}(X, X) + \gamma I)^{-1} \mathcal{K}(X, x_*)), \]

where $\mathcal{K} = \Sigma^{(l)}$. The NNGP kernel models the network’s covariance at random initialization, and allows for training by means of Bayesian inference. A related approach is to model the full dynamics of the network during GD training. Jacot et al. [14] showed that the gradient descent (GD) optimization process, when examined from functional perspective, can be captured by a kernel with RKHS corresponding to a network’s gradients, further referred to as the neural tangent kernel (NTK). The general form of NTK is given by [14, 3]:

\[ \Sigma(x, x') = \sigma^2 \mathbb{E}_{(u,v) \sim GP(0,\Lambda^{(l)})} [\phi(u)^T \phi(v)] \]

\[ \Theta(x, x') = \sum_{l=1}^{L+1} \left( \Sigma^{(l-1)}(x, x') \cdot \prod_{l'=l}^{l} \hat{\Sigma}^{(l')}(x, x') \right), \]
where dot denotes gradients. NTK represents the tangent kernel of the form \( \nabla_\theta f(X)\nabla_\theta f(X)^T \) in the infinite width-limit. In this work we refer to its corresponding parameter space covariance \( Z = \nabla_\theta f(X)^T \nabla_\theta f(X) \) as neural tangent feature matrix (NTF). In the infinite width limit the parameters change so little throughout the training \( (O(1/\sqrt{p}) \), where \( p \) is the number of parameters) that the network can be approximated with a first order Taylor expansion around the initial parameters \( \theta_0 \) [17]:

\[
f_{\theta_t}(x^*) \approx f_{\theta_0}^{lin}(x^*) = f_{\theta_0}(x^*) + \nabla_\theta f_{\theta_0}(x^*)^T \omega_t,
\]

where \( \omega_t = \theta_t - \theta_0 \). The approximation \( f_{\theta_0}^{lin}(x^*) \) is referred to as the "linearized" network.

Because \( \Theta \) stays constant throughout training, the solution to the ODE above has a closed form and is straightforward to compute [17]. It yields the following model after training (when \( t \to \infty \)):

\[
f_{\theta_\infty}^{lin}(x^*) = f_{\theta_0}(x^*) - \Theta(x^*, X)\Theta(X, X)^{-1}(f_{\theta_0}(X) - y).
\]

By putting a Gaussian prior of \( \mathcal{N}(0; \mathcal{K}) \) on the functional form of the neural network, the linearized model results in a GP predictive posterior with the following moments:

\[
\mathbb{E}[y_*] = \Theta(x^*, X)\Theta(X, X)^{-1}y
\]

\[
\text{Var}[y_*] = \mathcal{K}(x_*, x_*) + \Theta(x_*, X)\Theta(X, X)^{-1}\mathcal{K}(X, X)\Theta(X, X)^{-1}\Theta(X, x_*) - 2\Theta(x_*, X)\Theta(X, X)^{-1}\mathcal{K}(X, x_*),
\]

This formulation of predictive distribution \( p(y_*|x_*, D) \) is given in [17] (eq 16). Lee et al. [17] points out, however, that their formulation does not admit the interpretation of the resulting GP as a full predictive posterior. Yet it can still be used as a predictive distribution. He et al. [13] attempted to remedy this by adding a missing variance (defined by a function \( \delta(\cdot) \)), of the parameters of all layers except the last one, to the linearized network formulation:

\[
f_{\theta_\infty}^{lin}(x^*) = f_{\theta_0}(x^*) + \nabla_{\theta_{L+1}} f_{\theta_0}(x^*)\omega_{L+1} + \delta(x^*) + \nabla_{\theta_{L}} f_{\theta_0}(x^*)\omega_{L}.
\]

This results in the following predictive distribution, denoted NTKGP, that can be interpreted as a predictive posterior:

\[
\mathbb{E}[y_*] = \Theta(x_*, X)(\Theta(X, X) + \gamma I)^{-1}y
\]

\[
\text{Var}[y_*] = \sigma^2_{\theta}(\Theta(x_*, x_*) - \Theta(x_*, X)(\Theta(X, X) + \gamma I)^{-1}\Theta(X, x_*)).
\]

All mentioned predictive distributions were categorized in [13]. Notably linearized network GPs (Eq. 6) without regularization were related to deep ensembles, and with regularization to randomized priors [21]. Unless otherwise stated we use the randomized prior approach in all our experiments.