===Reviewer 1===

The execution time for inference is not provided in the paper. It might be a problem in cases like web marketing, where the agents must make decisions in millisecond order. Median inference time to select an action is between 5 to 8 ms across all datasets in terms of the wall-clock time, and does not change as more examples are seen. This compares favourably versus other methods such as LINFULLPOST, whose action selection gets slower as more data is observed. We will state this in the next revision.

The advantage of the proposed algorithm is clear for the discrete tasks but not for continuous tasks. The published GLN formulation is specifically for Bernoulli modelling. We propose tree-based discretization as a method of adapting GLCB to continuous rewards. The results are competitive with SOTA so we have elected to include them for completeness.

I think the authors used seven datasets out of the eight datasets described in the paper [7]. The only one not used from the paper is "mushroom." Why is it excluded? The Mushroom dataset was the first dataset that we tried, and we obtain SOTA performance. We later dropped the results for clarity of exposition because the dataset fits neither a classification nor regression formulation out of the box, i.e. the correct decision must be based on expected utility computation.

[...] the synthesized dataset "wheel" seems to have a parameter δ. What is the value of δ? We used $\delta = 0.95$, which was the default at the time in the Deep Bayesian Bandits library. We now note this.

===Reviewer 2===

[. . .] in the appendix there is no explicit literature reference for the CTREE algorithm. CTREE is our own simple very method of tree-based discretization, and is defined in Algorithm 3. We will better explain this in the revision.

We will also expand our *Broader Impact* section as requested.

===Reviewer 3===

The rationale for why utilizing such a scheme for "Pseudocounts for GLNs" is not clearly explained. Pseudocounts have a strong track record for driving exploration in reinforcement learning (eg, [12]). Density estimation is typically utilized to compute pseudocounts, which is computationally expensive. By using the structural property of GLN gating we are able to approximate with essentially zero computational overhead. Moreover, our pseudocount proposal is closely related to “half-space depth” and “half-space mass”, which are statistical notions used within outlier detection to avoid density estimation. One can interpret our exploration bonus to be proportional to how much of an outlier a given context is using the existing GLN gating mechanisms.

The rationale for utilizing the proposed "Pseudocounts for GLNs". Why do you use such an aggregation scheme? Are there any other alternatives? We experimented with different aggregation schemes such as mean, median, and min. We discuss the “soft-min” aggregation in the paper because it has both strong empirical results and theoretical guarantees.

[...] the data size is very small as you only report experimental results till 5000. I wonder if the advantage of your proposed method can keep when the data size increases? We chose a sample size of 5k (with 500 seeds) to allow for fair evaluation across all baseline algorithms, some of which scale super-linearly in the data size (GLCB incurs constant cost). We did manage to run our experiments for 10-fold the number of steps (with fewer seeds), and found that our GLCB remains the best neural algorithm. We were unable to run full Bayesian Linear Regression (LINFULLPOST) within the rebuttal timeframe (it requires inverting matrices that grow with the amount of observed data), so it is possible that this might outperform GLCB despite being prohibitively difficult to scale.

===Reviewer 4===

We agree that our discussion of should be expanded and aim to do so using the additional page for accepted papers.

[I]t is not clear if the grid search over hyperparameters for GLCB is leaking information about the test set. Our experimental setup followed the guidelines set out in the cited Deep Bayesian Bandit Showdown paper, which is an established benchmark focusing on online performance given a single stream of data. For each competitor we tried both their previously published hyperparameters as well as performing our own sweeps to ensure the fairest comparison.

Furthermore, there is no idea of the sensitivity of GLCB to hyperparameter choices (i.e., what if tuning is not done?). Our model is robust to hyperparameter choices in our experience. This is supported by our experimental results, where we use a *shared* set of hyperparameters for all binary tasks and all continuous tasks, despite large differences in the shape and distribution of the datasets.

[D]oes GLCB do better on regression tasks if the (scaled and shifted) reward is randomly rounded to a Bernoulli to induce a classification problem? Thanks for the interesting suggestion. We ran this experiment by binarizing the “financial” dataset, where the best action has value 1 and the rest have 0, and GLCB does indeed perform better in this setting. We will follow up on this.