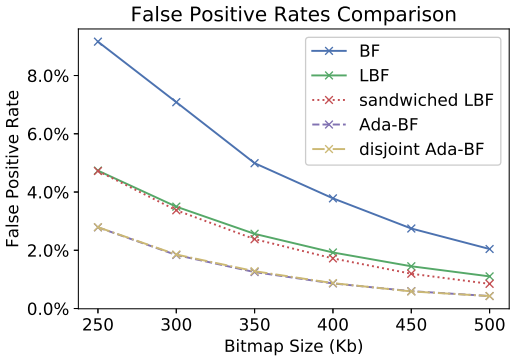
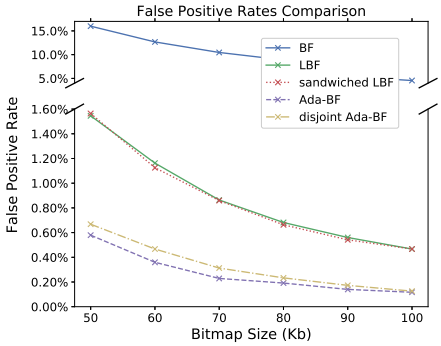
We thank all reviewers for helpful comments. We are excited to know that everyone feels that this work is worth a publication at NeurIPS. We are encouraged they found our idea is novel (R3,4), our insight is clear and valuable (R1), our improvements are notable (R1) and our applications are interesting (R1,3).

To R1: Thanks for your thoughtful comments and support for the paper! Regarding the process to set the number of hash tables and tune the boundary of each partitioned region, we proposed a simple hyper-parameter tuning strategy which only includes two knobs. Although the boundaries of the partitioned regions are not optimal, our Figure 5 in the appendix also suggests that the performance of Ada-BF and disjoint Ada-BF is not sensitive to the choice of hyper-parameters. Thanks for the suggestion, we will put a detailed discussion in the appendix on the effect of those knobs.

To R3: Thanks for your thoughtful comments and support for the paper! We like the idea to add a thorough discussion contrasting all the three variants of bloom filters with details. Regarding the comparison of model assumptions between Ada-BF and LBF (learned Bloom filter) or NBF (Neural Bloom filter). Our work is built on the assumption similar to LBF that the score distribution is static. While NBF assumes the meta-learning model can generalize to different classification tasks instead of a specific query distribution. So, if the meta-learning model can generalize well, NBF is a good idea. However, our methods and LBF may enjoy the following advantages:
1) Less Requirement of Model Accuracy: Compared to NBF, Ada-BF does not require a machine learning model to be very accurate. For example, we use a weaker learner, whose accuracy ≈ 0.90, for our malicious URL experiment. This classifier is not strong because a degenerate classifier which just labels everything as the most frequent class (just classify as benign URL as malicious is rare) gives you accuracy of 0.82. **Figure 1a** suggests Ada-BF still reduces the memory cost by 1/3 compared to LBF, and reduced over 45% memory usage compared to vanilla BF.
2) Smaller Model Size: If the model size is too large, all the learned Bloom filters are worse than vanilla (non-learned) Bloom filter. To achieve good generalization performance across different tasks, NBF requires a meta-learning model which is usually large in size. While LBF and Ada-BF can use a smaller model to adapt to a specific task. In our experiment, we use simple random forest models and even after taking the model size into account, our experiment results (see paper Figure 4a) suggest Ada-BF and LBF reduce the memory footprint by **50%** compared to the LBF and SLBF.
R3's idea is an excellent future direction. Our proposed Ada-BF can be used over any partitioning of the data. Partitioning of higher dimensional space using VAE instead of a one-dimensional classifier score can lead to novel variants. The nice thing about Ada-BF framework is that it only needs to get a (crude) estimate of the ratio of the density of keys over the density of non-keys ($\frac{f_{keys}(x)}{f_{non-keys}(x)}$) across different partitioned regions. With this information, Ada-BF will achieve better FPR compared to LBF!



(a) FPR of learned Bloom filters using weaker learner



(b) Fake news experiment

To R4: Thanks for your thoughtful comments and support for the paper! Here we address some concerns about our methods:
1) Why not just Ada-BF instead of disjoint Ada-BF: although disjoint Ada-BF is a bit worse than Ada-BF in terms of lower FPR. However, disjoint Ada-BF enjoys some benefits during deployment. Disjoint Ada-BF has several separate Bloom filters that can be deployed on distributed servers. To query a key, we only need a central server to infer the score then send the key to the corresponding Bloom filter to decide the membership.
2) When our algorithms perform better than LBF: Ada-BF and disjoint Ada-BF generalize the learned Bloom filter (LBF). If we only partition the score into two regions, Ada-BF and disjoint Ada-BF are reduced to a standard LBF. Hence, the performance of our algorithms are **at least as good as** LBF for any dataset.
3) Computation overhead: Our algorithms did not add a lot of extra computation overhead during the query process. Using the malicious URL experiment as an example, when sketch size = 400 Kb, vanilla BF, LBF, Ada-BF, disjoint Ada-BF use 3, 6.98, 9.40 and 12.07 hashing operations for each query. The slight extra hashing cost is negligible as it takes 10 nano seconds per hash evaluations Without parallelism. Thus Ada-BF adds like 25 nano seconds overhead. The inference is also fast, it only takes 1s to process half million URLs (2 nano seconds per url) using the random forest model. We will add a discussion on these overheads.
4) Reproducibility: the code and datasets used in the experiments are included in the supplementary materials for review. We also have uploaded the code to GitHub and provided detailed instructions to run the experiments.
5) Performance in other datasets: Given the suggestion, we experimented on a new task with dataset having different characteristics. The dataset consists of tweets flagged as fake. The task is to use bloom filter to quickly check (in very small memory) whether a tweet is known fake or not. **Figure 1b** suggests the Ada-BF and disjoint BF reduce the memory cost by **45% to 50%** compared to LBF and SLBF. We will add details to the paper. Clearly, the results make our paper stronger.