Stacked Filters Author's Feedback

Anonymous Author(s)

1 LEARNED FILTERS IN INTEGER EXPERIMENTS

To further complete our synthetic data experiments, we evaluated the Learned Filter on our synthetic dataset. The Learned Filter performs poorly in this instance because it is unable to learn a pattern in the randomly generated keys. This means that its performance becomes similar to a standard Bloom Filter. However, it is slightly larger because it stores a full ML model as well.

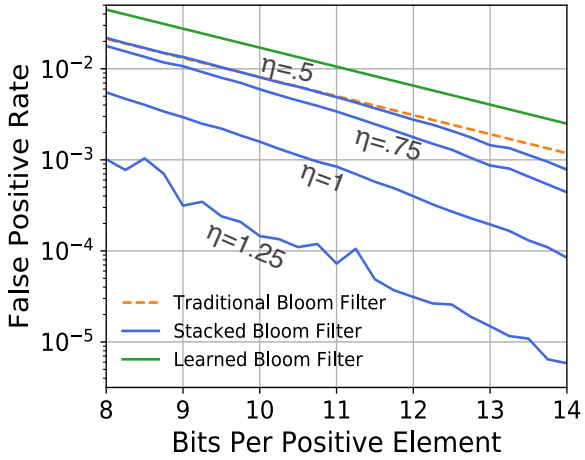

Figure 1: Stacked Bloom Filters signficantly outperform Learned Filters with synthetic data.

2 ADDITIONAL WORKLOADS: FIREHOSE DATASET

In order to demonstrate the variety of uses for Stacked Filters, we have run an additional set of experiments against a standard benchmark in packet filtering caled Firehose [1]. This benchmark simulates an environment where some subset of packets each keyed by a unique id is labeled suspicious and need to be filtered. 10% of the packets are marked suspicious as the positive set, which left the other 90% of packets as a negative set. We then used the generator provided by Firehose to produce packets which we filtered based on the the packet ID. We further restricted the known negative set of the Stacked Filter to 20% of the negative set in the same manner as we did for the URL blacklisting set. From this, we measured the latency and false positive rate of Stacked, Learned, and Traditional filters in the same manner as we did for URL blacklisting, results for which can be found in Figure 4 of the main paper.

As can be seen by figure 2, Stacked Filters perform very favorably in this context, providing multiple magnitudes lower FPRs than either traditional or Learned Filters. Learned Filters perform poorly in this use case because the Packet IDs carry no intrinsic meaning and therefore lack a pattern for the Learned Filters to classify based on. Further, these trends confirm those from the URL Blacklisting experiment in the main paper as Stacked Filters provide a lower false positive rate than traditional filters, multiple magnitudes lower computational cost than Learned Filters, and a significantly improved overall performance as compared with both.

Focusing on the overall performance as compared to URL blacklisting, we find that the Stacked Filters perform significantly better in this context as the Firehose benchmark is more skewed than the distribution we provided for URL queries. The Learned Filters perform significantly worse because of a high false positive rate resulting from a lack of learnability, a phenomenon which we analyzed in the final graph of Figure 6 in the main paper. Learned Filters' computational performance is the same because we once again substitute our own, slower, query latency numbers with the query latency provided by [2].

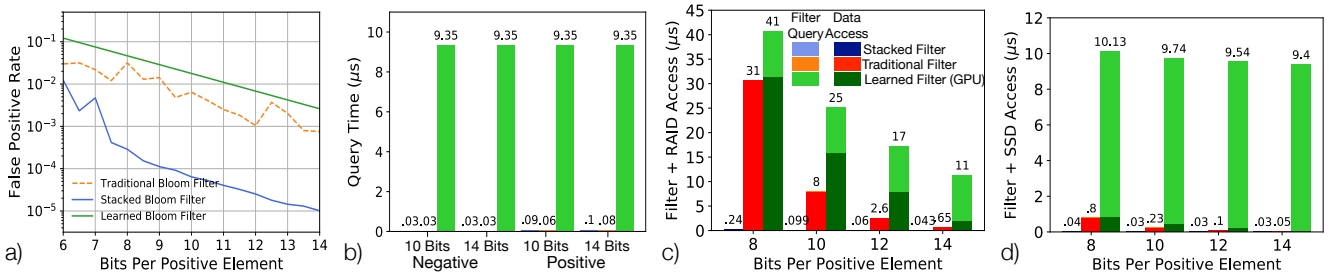

Figure 2: Stacked Filters maintain their strong properties of lowering the false positive rate for the same space and computational efficiency across new workloads (Firehose).

REFERENCES

[1] K. Anderson and S. Plimpton. Firehose streaming benchmarks. Technical report, Sandia National Laboratory, 2015.

[2] J. W. Rae, S. Bartunov, and T. P. Lillicrap. Meta-learning neural bloom filters. *arXiv preprint arXiv:1906.04304*, 2019.