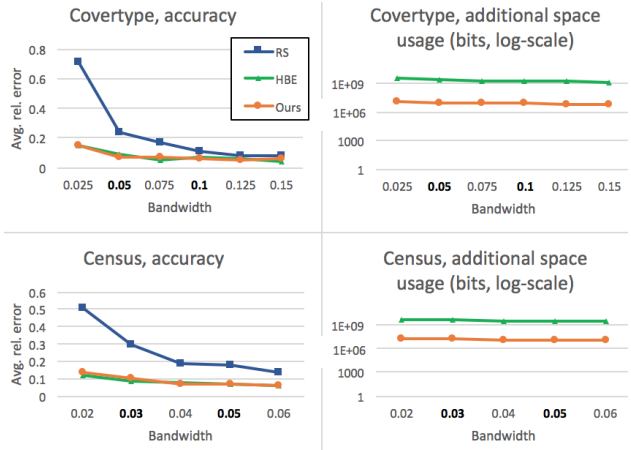
We thank the reviewers for their useful feedback and suggestions.

Significance of space improvement: The need to reduce the space requirements and the preprocessing time of the [CS17] algorithm was pointed out previously in the literature. For example [SRB+] states that *"...straightforward implementations of the proposed approach require a large amount [...] of preprocessing time and memory to create the requisite number of hash tables"* (Limitations section, pg. 2). This motivated [SRB+] to develop core-set-like methods for reducing the data size (cf. lines 93–99 in our submission).

Regarding theoretical analysis, we emphasize that our result is a direct theoretical improvement to the framework of [CS17]. While the algorithmic modification is simple (which has the advantage of being practical), the analysis is rather subtle (cf. Theorem 7). Moreover, it is nearly optimal: As mentioned on page 3, any algorithm must use at least $N = \Omega(1/(\tau \cdot \epsilon^2))$ data samples to guarantee $(1+\epsilon)$-approximation (see e.g., [PT18]). Thus, the $1/\sqrt{\tau}$ improvement that we provide is essentially the best possible, as our storage is proportional to the size N of the sample data set. We remark, however, that further improvements could be still possible by storing $O(N)$ bits instead of memory words.

Regarding the quantitative improvement, we note that the lower bound τ can be very close to zero in practice, which makes the $1/\sqrt{\tau}$ improvement very substantial. Concretely, if in Algorithm 1 we set the budget of kernel computations per KDE query to L (recall that analytically $L \sim 1/\sqrt{\tau}$), then our method stores L hashes per point (in expectation) while HBE stores L^2. Since L is typically of the order of at least dozens or hundreds, the gap in performance can be large. For example, the plots attached to this file are for $L = 250$, resulting in space reduction by a factor of 250.

Additional experiments: We thank you for the useful suggestion to study the performance as the bandwidth is varied. We have run those experiments and representative results are included here (as space permits). We will include the full results in the revised version. The plots shown here are for 250 kernel evaluations per query with varying bandwidths. The x-axis values in boldface are the bandwidth values used in our submission. As predicted, (a) HBE and our method achieve similar error, (b) our space usage is significantly better (by a factor of 250, as explained above), (c) as the bandwidth grows and the KDE values become bounded away from 0 the estimation task becomes easier, and in particular naïve random sampling improves and converges to the same error as HBE. We will also include results for other LSHable kernels.



Units of space: The plots measure space in single bits (however, the labels on the x-axis are accidentally scaled by 100; we have fixed this). We accept the reviewer's remark that there are various efficient ways to store each hash, so we redid the experiments by measuring the *number* of stored hashed points. The resulting plots are essentially scaled versions of the plots in the paper. In particular, the relative improvement (by a factor of L) over HBE is the same. For the sake of compatibility with our original submission, the plots in this rebuttal still measure space in bits.

Clarifications: All of the following points raised by the reviewers will be revised or clarified in the final version. Apologies for any confusion. **R1:** "scheme on page 5" Correct. More accurately for any L, the expected number of non-empty bins $b_j(y)$ is δL, and the estimate in line 152 has expectation τ, without having to know it. **R1:** "loss in variance" The variance affects the error of the estimate, so its impact is indirectly reported on Figs 1–8. We believe the *relative* error reported on those figures is more natural in our context than the additive squared error (i.e., the empirical variance), because it directly corresponds to the parameter ϵ. **R2:** "stated bound seem to assume $d = O(1)$" Indeed, the introduction alternates between the number of kernel evaluations and the running time, not always consistently. Generally, the running time (either of the preprocessing or the kernel evaluation) should have a factor of d in front of it. **R2:** "The new bound is only better when $d < \tau^{1/2}$" Indeed, the second column of Table 1 accurately describes the *preprocessing time* of both algorithms, as opposed to their extra space usage (which can be reduced for both algorithms by storing pointers not points). Theorem 1 summarizes the bounds correctly, parameterized by the relevant quantities. **R2:** "additive $\epsilon\tau$ error guarantee?" This is a subtle point. Our understanding is that without additional assumptions, such additive error guarantee must require $1/(\epsilon\tau)^2$ samples by anti-concentration. Whether it can be achieved under weaker assumptions needs further exploration. **R2:** "strictly better than quadratic?" Correct. **R2:** "query points from the data set?" Yes. **R3:** "why is the sampling probability chosen as $\delta = 1/(n\sqrt{\tau})$?" Intuitively, so each element appears in expectation in $O(1)$ hash tables (cf. lines 71–75). Technically, this equates the variance term of our modification to that of the original HBE algorithm, ensuring that our variance is at most twice that of HBE (cf. last two lines in proof of Theorem 7, with $\beta = 0.5$). **R3:** "refer to the proposed one, right?" Correct. **R4:** "what is meant by "typical kernel value"" Median KDE (which for all the considered settings is within a factor of 2 from the average KDE).