Thanks to all reviewers for their careful reading and thoughtful comments.

Reviewer #1. The rationale in the choice of the comparisons for the experiments was to be as extensive as possible: covering the methods which solve Nyström KRR first, then related methods such as those which solve kernel regression with different smooth losses or other sketching techniques, and finally we wished to cover partially related kernel based methods i.e. approaches computing similar quantities as the predictive mean of variational GPs (VGPs). Indeed, we agree with Rev1 on the difference between Nyström KRR and VGPs, in terms of methods and objectives, as we already clarified in Section 4 (there we also pointed out the greater generality of SVGP). To be more explicit we will further clarify the scope of our experiments and the difference between Nyström KRR and VGPs in Section 4. From this viewpoint we agree that FITC / SGPR with KeOps would better fit the third category than SVGP and we will include the suggested methods in our experiments. Note that we had considered the option of comparing with FITC / SGPR while designing the experiments. However, in the end we opted for SVGP since it can be trained in minibatches, allowing a higher number of inducing points, contrary to the implementations of SGPR in GPyTorch and GPflow that require all data and gradients to be stored on the GPU, thus strongly limiting the number of inducing points. For example on our hardware, GPflow SGPR runs out of GPU memory with only 20 inducing points on the HIGGS dataset. This shows that current implementations of SGPR cannot scale on big dataset. However, for the sake of comparison, we decided to reduce the training set to $n = 300000$ points for each dataset. This allows SGPR to fit 1000 inducing points in memory. We then ran one experiment with fixed SGPR points and the same number (1000) of Falkon centers, and another where we trained the SGPR points, and ran Falkon by tuning the number of centers such that the computational time matched that of training SGPR to convergence. In this second case we can compare the accuracy of the two methods. In Table 1 we provide some preliminary results on a subset of our datasets. Notice that in the first experiment Falkon is much faster than SGPR, and the two methods have comparable accuracy; in the second experiment, the accuracy of Falkon is higher or equal to that of SGPR. The final version of the paper will compare the two methods on all considered datasets.

Reviewer #2. As suggested by the reviewer we will clarify in Table 2 and in Figure 1 that the results for GPyTorch and GPflow correspond to the SVGP algorithm. In particular, as suggested by Rev2 we will report the training times of SVGP clarifying how the procedure differs from the one of Falkon. Indeed by "time" we mean *total training time* since in general, this is the limiting factor. The crucial difference in the optimization of the two algorithms is due to the convexity of the FALKON objective which allows us to use the more efficient preconditioned conjugate gradients optimization, based on out-of-core matrix vector operations, instead of ADAM. We will recall this difference in the experimental section, before Table 2. As suggested by Rev2 we will include a table on how performance scales with more GPUs on the considered datasets. According with our current hardware availability, we will consider how the performance scales with 1 to 4 GPUs (see Fig. 1 for the TAXI dataset). To conclude, with "blocks" we mean that matrices are divided into blocks which fit in GPU memory. Each GPU will then be assigned to work on different subsets of blocks which it will process one at a time. In Figure 4 we neglected to mention that $TT^\top$ is a symmetric matrix, so we store only its lower triangular part.

Reviewer #3. We thank the reviewer for the thoughtful comments and interesting questions. We agree with Rev3, right now Nyström methods based on preconditioning can scale up to 10^6 centers since they are limited by the available memory. This can be addressed in different ways: 1) achieving the same accuracy with fewer centers is possible by picking better centers e.g. using leverage scores or DPP sampling instead of random sampling. 2) recomputing the preconditioner on the fly, in blocks, at each iteration; this would remove the memory bottleneck, but would make the algorithm $O(\log n)$ times slower 3) further sketching the preconditioner itself with randomized linear algebra. We will add a remark with these considerations at the end of Section 5. Reducing the precision is also an interesting direction to increase the number of Nyström points and to leverage the GPU and TPU architectures. The crucial aspect in this direction is to guarantee numerical stability of the computation of the kernel and of Cholesky decomposition. Interesting starting points in this direction are stochastic rounding techniques (see e.g. "Stochastic Rounding and its Probabilistic Backward Error Analysis" of Connolly et al. 2020). To conclude, CPU power is not crucial to run our implementation of FALKON since all the computations are done by GPU, while CPU essentially controls only the logic of the program.

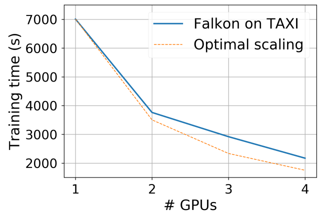

Figure 1: multi-GPU scaling of the TAXI dataset (experiment run on the workstation Nvidia DGX-Station).

	MSD-300k		SUSY-300k		HIGGS-300k	
	time	MSE	time	AUC	time	AUC
GPflow SGPR (fixed 1k points)	7s	83.06	8s	**87.43**	6s	74.48
Falkon (the same 1k points)	1.5s	**83.01**	1s	87.39	0.4s	**75.58**
GPflow SGPR (trained points)	40s	77.24	24s	87.49	11s	77.35
Falkon (match time)	42s	**76.15**	19s	**87.51**	9s	**77.66**

Table 1: Falkon is faster and more accurate than SGPR in the first and third setting; in the second, with very few inducing points, Falkon is much faster but has slightly worse accuracy.