We agree with all three reviewers that evaluating the predictive variances is important. We computed the test negative log likelihoods (NLL) as R2 suggests which reflect the variances' quality and present the results below. We will include the NLLs in the final version of the paper in addition to reporting averages and standard deviations in all of our other tables by running more trials.

Test NLL	PolTele ($n=9,600$)	Elevators ($n=10,623$)	Bike ($n=11,122$)	Kin40K ($n=25,600$)	Protein ($n=29,267$)	KeggD ($n=31,248$)	CTslice ($n=34,240$)	KeggU ($n=40,708$)	3DRoad ($n=278,319$)	Song ($n=329,820$)	Buzz ($n=373,280$)
Exact GP	$\mathbf{-0.217}$	$\mathbf{0.440}$	$\mathbf{0.189}$	$\mathbf{-0.456}$	$\mathbf{0.897}$	-0.895	$\mathbf{-1.044}$	-0.563	0.908	$\mathbf{1.151}$	$\mathbf{0.083}$
SGPR	0.051	0.659	0.449	0.475	0.989	$\mathbf{-0.986}$	1.419	$\mathbf{-0.728}$	0.944	1.359	0.777
SVGP	-0.011	0.504	0.308	0.249	1.027	-0.935	1.428	-0.661	$\mathbf{0.696}$	1.417	0.205

@R1 Regarding fit for NeurIPS: Thank you for your comments and suggestions. As R2 pointed out, there is a growing interest in using GPs for large scale problems. As a concrete example, the Bayesian optimization community within NeurIPS is turning toward large-scale optimization which will greatly benefit from the uncertainty calibration and flexibility of exact GPs while maintaining prediction times below 1s. GPs have long been a part of NeurIPS and conventional wisdom has been to compromise performance for scalability by using approximate GPs when $n \gg 1000$. Our paper shows that we can actually have both when $n < 10^6$ using modern hardware. In our ablations, we also show the gains from staying non-parametric and keeping inference exact, suggesting future research directions that scale up exact GP inference. Finally, we will clarify that SGPR is by (Titsias, 2009) and SVGP is by (Hensman et al., 2013).

@R2 Regarding the implications of this paper: We agree that we should clarify why we believe these approaches don't replace sparse GPs. Sparse GPs are still virtually mandatory when (1) $n \gg 10^6$, and (2) for models that require approximate inference like deep GPs regression or GPs classification. With that said, we agree that this paper raises an interesting question of when sparse GPs are useful when $n < 10^6$. Realistically, given the enormous hardware demands of running with $n \sim 10^6$, we expect sparse GPs may still be preferable in that regime. We will move sections of the paper to the appendix to add this discussion.

Still, we believe that sparse, non-deep GPs could potentially be all but obsolete in the $n \sim 10^4 - 10^5$ regime for regression, where in fact exact GPs are faster and more accurate than sparse GPs. This has important ramifications, e.g., in the large scale Bayesian optimization setting (see @R1). While we agree that many of these improvements are due to engineering effort, it is hard to overstate the value that engineering effort can provide to the GP community at this point in time. If this paper demonstrates anything, it is that many of the problems the scalable GP community face may be solved by sheer engineering effort. For example, we have recently been able to further decrease the running time on a dataset like 3dRoad ($n \sim 10^5$) from 7 hours to 1 hour through more efficient GPU routines. We believe that this work will enable researchers and practitioners to use GPs on problems for which it previously would have been intractable.

@R2 Regarding experimental setup and distributed Cholesky: Thank you for the suggestions on experimental setup and the reference on (Nguyen et al., 2019). We are happy to include results with ARD in the supplementary materials, as well as results on variational versus exact multitask GPs, deep kernel GPs, and other models in the appendix. Regarding distributed Cholesky, its main drawback are (1) the need to retain a $O(n^2)$ matrix in memory which may not be feasible and (2) the communication cost. Assuming we actually have $O(n^2)$ memory and have w workers, Algorithm 2 and 5 of Nguyen et al. require two broadcasts of a $b \times b$ matrix block among the workers every iteration, or $2w \cdot n/b$ exchanges. In contrast, using CG requires exactly $2w$ exchanges to do a linear solve. Furthermore, a matrix-vector multiplication can be cast as a map-reduce which is simple to implement and embarrassingly parallel. We were unaware of Nguyen's paper at submission and we will add this discussion to the paper.

@R3 Regarding predictive variances We agree that evaluating the variances is important and have done so (see above table). We note that the precomputation, like CG, can be run to a specified desired tolerance. While in (Pleiss et al., 2018) the authors were concerned with speed and would only run up to 100 iterations of Lanczos, running hundreds or even thousands of iterations still affords sufficiently fast and accurate variance computations, making it easy to simply run Lanczos to within tolerance. We will clarify this.

@R3 Regarding the number of inducing points: In (Titsias, 2009) the maximum number of inducing points considered is 500. Hensman et al. (2013) used 1000 inducing points on the massive Airline dataset. We note that even on small datasets like kin40k, running with even 4000 inducing points for SVGP would take multiple hours due to the cubic cost (even with CG) and quadratic number of parameters to fit.

We agree with you that we should improve our explanation for why using larger m is challenging. One reason is that SVGP requires up to 100x as many linear system solves as exact GPs. For example, training on a dataset with 500k datapoints and a minibatch size of 1024 results in 48828 solves for 100 epochs. There are other challenges that both SGPR and SVGP face. In particular, performing solves with the $m \times m$ inducing point kernel matrix K_{UU} requires either: (1) Cholesky decomposing K_{UU}, (2) precomputing $K_{UU}^{-1}K_{UX}$, or (3) solving with K_{UU} in an nested inner CG loop while solving with $K_{XU}K_{UU}^{-1}K_{UX}$, none of which are easily feasible with our approach due to the associated space and time complexities when m grows large.