We thank the reviewers for the valuable feedback and address specific comments below.

Motivation [R3]: We emphasize that the major advantage of our method is fast identification of reasonable kernel hyperparameters since MLL-opt is costly and has no global optima guarantees. A side benefit is that our approach empirically does not result in catastrophically bad hyperparameters because collective training across different tasks helps avoid bad local minima: averaging the gradient over tasks seems to help the training escape from pathological solutions. Also, overparameterization in deep neural nets is known to help SGD find good optima both in terms of optimization and generalization, and sometimes with theoretical support [Allen-Zhu et al., 2019, Du et al., 2019].

Hierarchical attention architecture [R1 & R3]: We would like to first clarify that our architecture consists of three parts: *local transformer*, *aggregation function* and *global transformer*. The *local transformer* sees every dimension independently and an aggregated representation is calculated for each dimension. The **interactions between dimensions** are later modeled by the *global transformer*. The architecture is carefully designed for the GP problem with desired properties such as versatility and permutation equivariance. We are not aware of similar existing architecture for learning on datasets. Kim et al. [2019] uses attention to aggregate information of the context data and relates target data to context data for prediction. The major difference is that our method uses hierarchical attention and it is able to take in datasets with **different dimensionality** with **one model**. We will expand this explanation in the paper. We also expect the proposed architecture to be useful for other tasks that involve learning on sets (e.g., the neural statistician).

Training setup [R1, R3 & R4]: With regard to R1's comments on generating large synthetic datasets, we would like to mention that our model does not necessarily need to be trained on large synthetic datasets for it to generalize to large-scale problems. In our experiments, we train a single neural model using synthetic datasets generated with stationary kernels of dimensionality $2\sim15$ and 30 data points on average. (Full details are available in Appendix C.) This **single trained model** is applied to **all the real-world tasks** with varying dimensionality and cardinality, yet it is able to perform fairly well on those **unseen tasks**. This demonstrates the effectiveness of our proposed architecture: although it is trained on small synthetic datasets, it learns to extract the underlying structure for large unseen datasets. Our model is expected to work for a relatively low-dimensional GP regression task that follows the stationarity assumption.

Scalability [R1 & R3]: Full GP MLL-opt scales as $\mathcal{O}(N^3)$ using Cholesky decomposition (GPy) and $\mathcal{O}(TN^2)$ using the conjugate gradient method (GPyTorch). It can be reduced to $\mathcal{O}(NM^2)$ with a sparse GP. All the methods require iterative optimization, meaning the computation cost incurred in practice is the complexity multiplied by the number of opt iterations. In comparison, our method scales as $\mathcal{O}(N^2 + D^2)$ and only requires one forward pass of the trained neural net. In terms of actual running time, our method compares much more favorably with all the baselines above. The $2,000$ datapoints used in our experiments are already considered large-scale for GP. To give a larger-scale example, for *power plant* with $4,000$ data points, our method is 226 times faster than GPyTorch on GPU while 332 times faster than GPy on CPU. We will include these larger-scale experiments in the revised version. It is possible to go even larger for our model, but full GP MLL-opt methods either take too long on CPU or report out of memory issues on GPU. Note that our model's complexity can be further reduced to $\mathcal{O}(MN + MD)$ if M-sparse attention is introduced. With regards to R3's comments that BO is usually small-scale, it might be observed that 1) BO is small-scale in part because of the cost of hyperparameter inference, and 2) AHGP is often empirically more robust in the small-data regime.

Predictive variance [R1]: It was presented in Appendix C. We will improve the presentation to make it more accessible.

MLL when AHGP performs better than MLL-opt [R2]: Upon checking, we observed that AHGP found an MLL that is slightly smaller than that of MLL-opt. This illustrates that MLL-opt can potentially overfit in the low-data regime, while AHGP adds extra regularization through learning to optimize over many different GP tasks.

Noise variance [R1]: In all experiments, we normalize the data input and output. The noise variance is fixed at 0.01. It is possible for our method to learn the noise variance. In practice, we find the predictive performance is not sensitive to the noise variance and setting it to 0.01 gives competitive performance for all baselines. Also, spectral mixture kernel is flexible enough to model the noise as well (with $\mu = 0$, small σ^2). We will add the clarification to make it more clear.

Neural processes (NPs) [R1]: In terms of the problem formulation, our work targets efficient estimation of GP hyperparameters as part of a full GP inference procedure, while NP serves as an end-to-end approximation to the GP inference itself. We also hope to emphasize that our method tackles a richer distribution of tasks (i.e., regression tasks generated from stationary GP, with **different dimensionality** and cardinality) with **a single trained model**, while NP aims at solving one particular task or similar tasks (**same dimensionality**) with one neural model.

BO hyperparameter transfer learning [R3]: Thanks for the pointer. We will include them in related work. This type of approach focuses on better warm-starting BO through leveraging data of previous related BO runs, such that a good surrogate model can be built with very few evaluations of the target function. In comparison, our method focuses on fast inference of kernel hyperparameters, and our single trained model is applicable to a wider range of GP use cases.

Inducing point methods [R2]: Thanks for the pointer. We will include the reference on efficient inducing point method [Burt et al., 2019] and add experiments with an increasing number of inducing points for sparse GP.