We thank the reviewers for their time and their feedback. The reviewers highlight the interest of our method and find both the model and the experiments interesting and different from the literature. Criticism is focused on one key question: how is this model different from a GP, in particular in the case of a local model. In the following we address this question, clarify the structure of our model further, and answer individual points the reviewers raised.

(**R1, R2, R3, R4 - Representation & Advantages to GPs**) We propose a model which follows the flow: latent variables per unit->GP->weights and a local version of this which also includes a context variable as input to the GP. In the appendix, we explain that this model can be run per layer in two variants: either as a GP generating all weights in the network; per layer, with a per-layer kernel with individual kernel parameters and inducing points; or in a stacked fashion, where each layer's weights depend on the outputs of the layer below. In our global model, the parametrization of k_w uses "inducing weights", the local model adds "inducing inputs" and makes the model distance-dependent. Depending on whether the model is used per layer, and whether the context variable is the input x or the previous layer-activations, the model will have quite different applications and (dis)-advantages. When comparing to a GP, the closest variant may be when utilizing the local model conditioned on input x at all layers. Consider this thought-example to prove they differ even in that case: our model can -for an identical kernel choice as a GP- learn dramatically different functions than a GP since it can simply 'switch off' k_{aux} (i.e. with a large lengthscale which sets all entries in k_{aux} to 1) and revert to the global model k_w which only utilizes the inducing weights but ignores the influence of the input. A GP can only express functions induced by the input-kernel, whereas our model can revert to a parametrization that ignores the influence of the input on the weights. This becomes more evident when considering per-layer or stacked parametrizations that blend with the k_w kernel. We will add an example of this to the appendix. As such, the influence of k_{aux} is an additional modeling tool to help induce biases into the prior and allow inheritance of beneficial properties of GPs when useful.

(**R1, R2, R3, R4 - Experiments**) We note that Deep Kernel Learning is a GP with a learned kernel that we compare to favorably. As per suggestions, we are adding comparisons to sparse GPs with RBF kernels to the supplement and will update Fig.16 (including notMNIST and fashionMNIST) to show the benefits of our model.

(**R1, R2, R3 - Scalability, Practicality**) When used per layer, the complexity is $O(|W| * M^2 + M^3)$ where $|W|$ = number of weights in a layer, M = number of inducing weights so it will be $O(|W|M^2)$ in practice. Empirically we found that we only require a small M, especially in a per-layer parametrization. The model thus induces an overhead compared to a regular mean field BNN, but maintains scalability in terms of size of the model.

(**R1, R3, R4 - Global Model & Modeling Benefits**) Another point of interest is what structure the latent-node parametrization of the global model induces and why this parametrization makes sense. Consider a per-layer kernel model, given samples z, the model is equivalent to a Matrix-Variate Normal weight model with components for weight columns and rows, parametrized by inducing weights. This has been shown to be useful in BNNs. Our global model adds two variations: (1) marginally over z, we have a mixture of matrix-variate Normal weight models as each 'unit' can change and (2) we share components between adjacent layers by sharing the unit-samples z. This can be seen as tying together the matrix variate distributions at each layer. Our construction can be used on any network structure, utilize convolution or sparse graph NNs with no notion of layers, we leave elaborate such constructions to future work.

(**R2**) Thanks for the helpful suggestions and comments. We will compare to ensembles in the next version.

(**R3**) **weight correlations:** conditionally on latent samples z, weights are independent when using a diagonal approximation. However, when z are marginalized out, correlations between all weights connected to a unit are still retained. We demonstrated this in Fig 11, App. G. **block diagonal:** Thanks for the suggestion. This is similar to the Partially Independent Training Conditional method in the GP literature. Our per-layer parametrizations are also moving in that direction and we intend to explore this suggestion further. **Budget vs model size:** our treatment of NNs is not focused so much on computational budget, as it is on capturing inductive biases and uncertainty which is about sample efficiency and robustness. **Fig.15** DKL is actually not the best method, Fig 15 shows our model to perform best.

(**R4**) **GP-ness:** We note that the GP predictions are different to that of the proposed method, e.g. for the dataset in Fig 4, the high-confidence regions for GPs are smaller. We also do not believe the GP-ness would hurt transfer learning in our formulation, but this goes beyond the scope and space we have in this paper. **Evaluation:** We provided some regression results on UCI in the appendix I.3, and we will add additional experiments on UCI and MNIST for some variants of our model and CIFAR-10 (last layer). We focus on the OOD setting since this is an advantage of our model and we compare against strong GP-baselines such as DKL. We'd love to see the community utilize some of the ideas regarding distance dependent weights demonstrated here in large scale applications down the line. **Minor:** f is an entire function. We take the function space view, which is now widely adopted in the sparse GP literature, $w = f(z) + \epsilon$. **Line 110:** we used this kernel. For tasks with high dimensional inputs, an input warping can be used. **Periodic Kernel:** Thanks! It demonstrates that we can impose a controllable and non-trivial prior on weights, and periodicities in the past have been difficult to "cook into" NNs, see "Expressive priors in Bayesian neural networks" by Pearce et al. **CKL:** CKL is an exciting avenue to pursue in combination with our model to facilitate utilizing such rich priors for weight-based models.