We thank the four reviewers for their insightful comments and suggestions. Below, we have addressed most of the items given the time- and space-bounded aspects of the rebuttal, hoping we clarified the main questions of the reviewers.

Reviewer 1 (R1):

• "…I looked into the paper in ref[12] …": In [12], the greedy algorithm is generic, with no assumptions about models it forms an ensemble from. In particular, the models are not forced to start from the same initialization, which we will clarify in the paper. For hyper ensemble, we are further interested in using a fixed initialization to isolate the effect of just varying the hyperparameters (while deep ensembles vary only the initialization, with fixed hyperparameters).

• "…why o(mk) became o(k^2) …": Random search leads to a set of m models. If we were to stratify all of them, we would need k seeds for each of those m models, hence a total of O(mk) models to train. However, if we first apply the greedy procedure to extract k models out of the m available ones, then the stratification needs k seeds for each of those k models, thus O(k^2) models to train (as a reminder, the greedy procedure does not imply any training).

• "…hyper-ens, str hyper ens and deep ens are quite close to each other …": Recent work like [16] show that improvements on WRN-cifar10/100 benchmarks are typically in small ranges (with larger room for improvements on cifar100). For Tab. 1, we ran the Wilcoxon signed-rank test (paired along settings, datasets and model types) and observe statistically significant improvements (except for ece, known to be noisier Nixon et al. (2020)). Similar results were obtained with a paired t-test. For Tab. 2 (with more costly experiments), we do not have enough runs to apply such tests. We nonetheless report the standard errors in the paper, which seem to indicate significant improvements.

	ens size	p-value (nll)	p-value (acc)	p-value (ece)	ens size	p-value (nll)	p-value (acc)	p-value (ece)
deep ens ↔ str hyper ens	3	1.1×10^{-5}	2.1×10^{-5}	0.25	5	9.1×10^{-6}	1.9×10^{-5}	0.33
hyper ens ↔ str hyper ens	3	0.0725	0.0017	0.43	5	0.0088	0.0018	0.44

• "…numbers in brackets …": Those numbers indicate the size of the ensemble; we will clarify this point.

• "…reporting results on other deep models …": We thank R1 for the idea and ran our entire benchmark for ResNet-20:

ResNet-20 / cifar100	nll (↓)	acc (↑)	ece (↓)	*ResNet-20 / cifar100*	nll (↓)	acc (↑)	ece (↓)
single (1)	1.245	0.679	0.105	batch ens (4)	1.235	0.697	0.119
deep ens (4)	0.905	0.749	0.043	batch hyper ens (4)	1.141	0.702	0.059
str hyper ens (4)	0.905	0.751	0.048				

Reviewer 2 (R2):

• "…hierarchical Bayesian modeling of neural networks …": Hyper ensembles can indeed be viewed as a mixture variational posterior and the entropy penalty is the ELBO's KL divergence toward a uniform prior. There are many related works from Bayes, e.g., Kemp & Tenenbaum (2008), Adams et al. (2009), Grosse et al. (2012), Lake et al. (2015). They typically use Bayes nonparametric priors/posteriors and MCMC; we use mixtures and SGD. We will add more detailed discussion to the paper.

• "…with replacement …": When used *with replacement*, the greedy algorithm from Caruana et al. [12, Sec. 2.1] makes it possible to find a *weighted* combination of models (e.g., $\frac{1}{4}$ (2 model$_a$ + model$_b$ +model$_c$) would correspond to the situation where model$_a$ has been selected twice). To avoid the pitfall rightly mentioned by R2, Algorithm 1 and Algorithm 2 (in appendix) make use of ".unique()" to correctly count the number of members.

• "…skew …": Skew intensity refers to the severity of the distortion applied to the corrupted dataset; see [28, 55].

• "…BNN baselines …": We use the same data/training/evaluation pipeline as that used in the `baselines` of the `edward2` repository. We can thus directly compare with the reported metrics for BNN VI and MC dropout on cifar10/100. E.g., on cifar100: nll/acc/ece=0.944/0.778/0.097 and 0.830/0796/0.050, which we will add in the paper.

• "…OOD experiment …": We thank R2 for this suggestion. Along the line of Tab. 1 in Hein et al. (CVPR 2019), we computed the table below for our WRN experiments (MMC/AUROC/FPR@95 are defined in Hein et al. (2019))

	(trained on cifar100, MMC (↓)/ AUROC (↑) / FPR@95 (↓)) cifar10	SVHN	(trained on cifar10, MMC (↓)/ AUROC (↑) / FPR@95 (↓)) cifar100	SVHN
deep ens	0.502 / 0.818 / 0.758	0.495 / 0.826 / 0.756	0.737 / 0.914 / 0.477	0.644 / 0.964 / 0.265
str hyper ens	0.525 / 0.823 / 0.744	0.561 / 0.802 / 0.764	0.727 / 0.917 / 0.455	0.572 / 0.973 / 0.172
batch ens	0.626 / 0.810 / 0.784	0.621 / 0.825 / 0.796	0.806 / 0.907 / 0.504	0.681 / 0.968 / 0.211
batch hyper ens	0.583 / 0.811 / 0.748	0.574 / 0.823 / 0.736	0.714 / 0.911 / 0.507	0.634 / 0.956 / 0.329

• "…interpretation of the parameter ξ_t …": In our setting, the parameter ξ_t contains the lower and upper bounds of the log-uniform distribution at the step t. Given ξ_t, $p(\lambda|\xi_t)$ is a standard log-uniform distribution.

• "…How many samples do you use for computing the objectives in Eq. 8 and 9? …": We use one sample for each data point in the batch. Sec. 5.1 (MLP and LeNet) uses 256. Sec. 5.2 (WRN) uses 512—64 for each of 8 workers.

Reviewer 3 (R3):

• "…third source of diversity …": The distribution p_t is log-uniform. Its variance is implicitly controlled by the entropy regularization (see Eq. 9) since both the variance and entropy depend on the width of the support (see lines 232 to 234). At prediction time, the variance does not play an explicit role since we use the mean of p_t, like in [45] (see lines 230-231). However the variance has a direct impact during the optimization when the λ_k's are sampled.

• "…qualitative explanation …": While the K distributions $p_t(\lambda_k)$'s are independent, we stress that their parameters $\{\xi_{k,t}\}_{k=1}^K$ are *jointly* learned in the tuning phase (see Eq. 9). Indeed, the *ensemble cross-entropy loss* ties together the K members (and hence $\{p(\lambda_k|\xi_{k,t})\}_{k=1}^K$). E.g., we see the complementarity of the members by comparing the ensemble metrics (nll/acc=0.718/0.821; see Tab. 2 in the paper) with the *average ensemble-member* metrics (nll/acc=0.851/0.804).

Reviewer 4 (R4): We thank R4 for the comments and feedback.