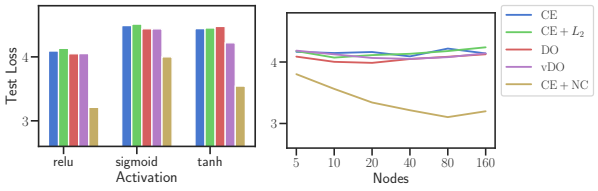
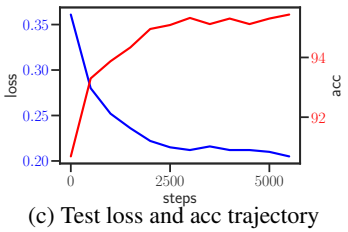
We thank all reviewers for their constructive and thorough comments. We will incorporate your many detailed comments to clean up our presentation in the next version. Most reviewers were happy with the significance and novelty of our method, while some concerns about experimental details were raised. We address your main concerns below.



(a) Stronger baselines for OOD experiment

	SVHN	CIFAR
Baseline	93.23	79.76
size ×2	93.59	79.64
NC (ours)	**93.83**	**81.15**
size ×4	93.88	80.47

(b) vs larger nets



(c) Test loss and acc trajectory

[R1] "confusing how the true loss is obtained" We estimate true loss $\mathcal{L}_T$ by computing the loss of a held-out set (X_{te}, Y_{te}), which is an unbiased estimator. We will update Alg 1 and Section 3.2 to be more clear about this point.

[R1,3] Is NC a 'complexity measure'? Our naming is similar to that of the Neural Process (Garnelo et al., 2018), which similarly is not a stochastic process in the strict sense. NC aims to bound the generalization gap just like traditional complexity measures such as VC-dim or Rademacher complexity. While NC does not come with theoretical guarantees, its data-driven prediction is much tighter, enabling the benefits we showed in our experiments.

[R2,5] More details of experiments Thank you for the suggestion. We will add more details about the task setup and training scheme to the supplementary material, and also publicly release the code used in our experiments.

[R2,3,4] NC takes whole dataset: expensive NC does not take entire training and test datasets as input. Instead, it takes subsets (i.e., minibatches) of the dataset at each gradient step (eq 4), just like standard SGD training. Note that the generalization gap calculated from random batches is an unbiased estimator of the true generalization gap. In the next version, we will make this critical point more apparent in Algorithm 1, eq 5, and Figure 2. We will also introduce a notation to distinguish between the entire dataset and a minibatch of it (e.g., $X^B_{tr} \subset X_{tr}, X^B_{te} \subset X_{te}, \ldots$).

[R3] "in a single task setting, not sure how much gain NC has over cross-validation" NC has much more expressive power compared to cross-validation as it actively alters the learning trajectory. Please note that we are already using cross-validation to select the hyperparameters for the baselines in Table 1.

[R3] "how this technique can be combined with data augmentation" We can use NC together with any sort of data augmentation, as long as we use the same augmentation scheme on all tasks.

[R3] "interpolation regime... how does NC distinguish between models that are trained to the same near perfect loss but exhibit different generalization behavior." We believe this is an important question. Our intuition is that NC compares the predictions for train and test data to see whether h is a smooth and consistent function over the entire data distribution, or if it just sharply predicts the train data. The Transformer learns how to do this comparison.

[R3] "PGDL competition is a good fit." We agree, and we plan to enter PGDL with an improved version of NC.

[R3,5] specific form of bilinear layer The form in eq (9) was shaped through experiments. Note that our ablation study (figure 6) shows that the bias ("ones"), loss ($\mathcal{L}$), and the bilinear layer itself each positively contribute to performance. The motivation for concatenating ones was to let the data embedding e_{tr} freely flow into the next layer.

[R4] "Proposition-1 only presents a one-sided bound...stronger claim?" In our proof in Appendix A, we actually proved the stronger two-sided version. We will edit the main text to show the two-sided bound.

[R4] "Weak experimental comparisons... compare w/ MetaReg on MNIST and CIFAR-10" Firstly, please note that we do compare against strong regularizers (label smoothing and mixup) on the datasets you mentioned in Table 1. MetaReg is not suitable as a baseline in our classification task because its task learners all share a "feature network", whereas we compare among methods that regularize networks trained from scratch.

[R5] "strong baselines... e.g., MAML. Figure 9 uses L2...not fair." First note that we showed NC's contribution is orthogonal to that of MAML in standard few-shot settings (fig 5). Fig 9 measures performance on *out-of-distribution task learners*: MAML can't be used here because initial parameters cannot be transferred to different architectures. Our additional experiments (1a) compare against stronger baselines: regular and variational Dropout (DO, vDO).

[R5] "simple learning with a larger model... capacity similar to base model + NC model" In table 1b, we evaluated nets with more capacity by jointly training n nets with a linear layer on top. The computation requirements for NC is on the scale of $n = 2$ since most of the compute is used in the data encoder. We also evaluated $n = 4$ for reference. NC's regularization is much more effective compared to simply using a larger model.

[R5] "concerned that NC might require a large labeled validation set in order to exhibit good performance" This is not a practical issue for NC, because the validation set size is simply a hyperparameter that we use to split the "true training set". Note that after training the NC model, we use unlabeled X_{te} to regularize task learners (Alg 1).

[R5] "comparison showing that NC is more stable across long learning trajectories" In 1c, we show the learning trajectory of an NC-regularized model on the KMNIST dataset. The model keeps improving for thousands of gradient steps, in contrast to other meta-learning methods like MAML which only work for a few gradient steps. The fact that NC is able to regularize learning in large single tasks (section 5.4) also demonstrates NC's stability in long trajectories.