A. Experimental Details

For obtaining approximate posteriors with SWAG and KFAC-Laplace, we follow the exact training procedures given in Maddox et al. (2019). We then implement ACNML on top of the diagonal SWAG posterior and the KFAC-Laplace posterior.

The variance of the SWAG posterior depends in a complex way on the learning rate and gradient covariances. To account for this, we introduce an additional temperature hyperparameter α and solve for the ACNML approximation using

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmax}} \log p_\theta(y_n|x_n) + \frac{1}{\alpha} \log q(\theta). \quad (16)$$

To calibrate α, we can calculate the CNML distribution using a validation set, by training on the entire training set and the validation point, and then selecting α such that our ACNML procedure produces similar likelihoods. We can also treat α as a tunable hyperparameter and select it using a validation set, similarly to how temperature scaling (Guo et al., 2017) is used to achieve better calibration for prediction, or how the relative weighting of priors and likelihoods are used in generalized Bayesian inference (Vovk, 1990) or safe Bayesian inference (Grünwald et al., 2017) as a way to deal with model misspecification. For our experiments using the SWAGD posterior, we heuristically tune α to be as large as possible without degrading the accuracy compared to the MAP solution. Note, however, that this procedure is specific to the particular way in which SWAG estimates the parameter distribution, and any posterior inference procedure that explicitly approximates the posterior likelihood (e.g., Blundell et al. (2015)) would not require this step. To select α for each model class, we swept over values $[0.25, 0.5, 1, 1.5, 2]$ and selected the highest value such that accuracy and NLL on the validation set did not degrade significantly compared to SWA. For VGG16, we use $\alpha = 0.5$ and for WideResNet28x10, we used $\alpha = 1.5$.

ACNML Optimization Details: With our approximate posterior $q(\theta)$ being a Gaussian with covariance Σ, we approximately compute the MAP solution for each label y as per Algorithm 1 by initializing θ_0 to be the posterior mean and iterating

$$\theta_{t+1} = \theta_t + \epsilon_t \Sigma(\alpha \nabla \log p_{\theta_t}(y|x_n) + \nabla \log q(\theta_t)), \quad (17)$$

using the covariance as a preconditioner. Similarly to the influence function calculation for the post update parameters discussed in section 3.2, this corresponds to taking approximate Newton steps at each iteration, using the Hessian approximation of the training set given by our approximate posterior. For our experiments, we used a constant step size $\epsilon = 0.5$ for the SWAG-D and BBP posteriors, and $\epsilon = 0.25$ with KFAC-Laplace. We empirically found that 5 steps was often enough to find an approximate stationary point with the SWAG-D posterior, and 10 steps for the KFAC-Laplace posterior.

For the reliability diagrams in Figure 5, we again follow the procedure used by Maddox et al. (2019). We first divide the points into twenty bins uniformly based on confidence (each bin has the same number of points), then plot the mean accuracy vs mean confidence within each bin. This differs from the reliability diagrams used by Guo et al. (2017), where they divide the range of confidence values into bins uniformly, resulting in unevenly filled bins.

For our expected calibration error (ECE) numbers, we use the same bins as computed for our reliability diagrams, and compute

$$ECE = \sum_{i=1}^{K} P(i) \cdot |o_i - e_i|, \quad (18)$$

where $P(i)$ is the empirical probability a randomly chosen point lies in bin i, o_i is the accuracy within bin i, and e_i is the average confidence in bin i.

We adapted the SWAG authors' implementation at https://github.com/wjmaddox/swa_gaussian to include the ACNML procedure for test time evaluation. Experiments were conducted using a mix of local GPU servers and Google Cloud Program compute resources.

MNIST Experimental Details: For the MNIST experiments, we used a feedforward network with 2 hidden layers of size 1200, with no data augmentation. The posterior is factored as independent Gaussians for each parameter, with the prior for each parameter being a zero-mean Gaussian with standard deviation 0.1.

We include an expanded results with additional metrics in Figure 13.

B. Further Experimental Results and Comparisons on CIFAR10

In addition to the comparisons in the main paper, we additionally compare to SWA-Gaussian (SWAG), which uses a more expressive posterior than SWAG-D, SWA with Monte Carlo Dropout (Gal and Ghahramani, 2015) (SWA-Drop), and SWA-Ensemble, which averages the predictions of independent runs of SWA as with regular deep ensembles (Lakshminarayanan et al., 2016). For reference, we show in-distribution performance of all methods in Table 2. Overall, performance differences between all methods are quite small, and ACNML's conservative predictions do not improve on NLL or ECE over some baselines on in-distribution performance, which is to be expected, since the main aim of our method is produce more calibrated predictions on **out-of-distribution** tasks.

For all Bayesian marginalization methods, we marginalize over 30 model samples, with the exception of SWA Ensembles, for which we average over 10 models, as each model sample requires training an model independently from scratch.

For completeness, we show expanded results on CIFAR10-Corrupted in Figures 7, 8, and 9, which include additional baselines and metrics. ACNML consistently achieves significantly better ECE than prior methods on the more severe corruptions, and generally comparable or slightly better NLL and Brier scores to the best performing baselines. With the same architecture, all methods generally have very similar accuracy, with the exception of SWA-Ensemble slightly outperforming better than other methods in accuracy.

While evaluating MC-Dropout, we found that adding dropout before each layer in VGG16 (labelled VGG16Drop in 8) significantly improved performance on CIFAR10-C. For fair comparisons, we reran all methods with the VGG16Drop architecture as well. We again find that ACNML performs the best in terms of calibration on the more severe corruptions.

C. Comparisons between ACNML and naive CNML on MNIST

In this section, we include expanded comparisons between ACNML and a naive implementation of CNML from Bibas et al. (2019) that computes the MLE/MAP $\hat{\theta}_y$ for each label by appending the query point and label to the dataset and finetuning for N epochs. Both ACNML and naive CNML are initialized from the same MAP solution, with ACNML taking 5 gradient steps on the query point and posterior and naive CNML finetuning with the query point and training set for 5 epochs. For the OOD dataset, instead of computing results for every level of corruption, we instead simply average out over all corruption levels by randomly rotating the test inputs.

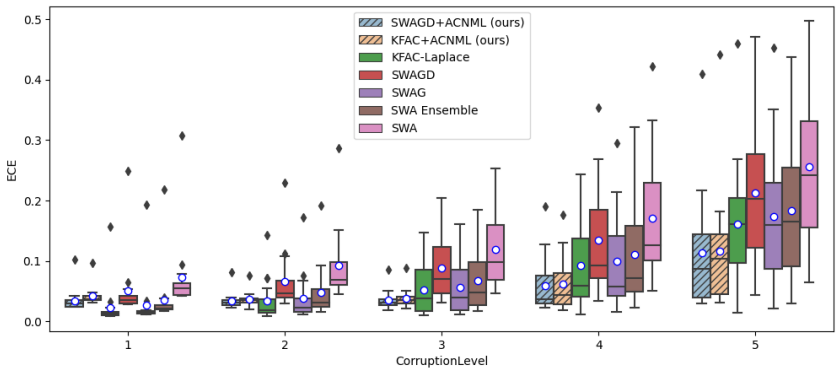
This naive implementation of CNML differs slightly from Bibas et al. (2019) in that we finetune the entire network, while Bibas et al. (2019) proposed only tuning the last few layers. During the finetuning, we also append the query point and label to every batch in optimization, and down-weighting that portion of the loss accordingly to get unbiased gradient estimates. We found this led to more efficient optimization than randomly sampling

We first examine how closely ACNML and naive CNML's predictions match on the same datapoint. To assess this, we compare the CNML normalization terms $\sum_y p_{\hat{\theta}_y}(y|x)$, NLLs, and the confidences of the two methods. The CNML normalization term captures how much each procedure was able to adapt to different labels for that input. A higher normalization term for an input means that we were flexible enough to fit multiple different labels well together with the training set (or approximate posterior in the case of ACNML), and typically means a less confident prediction on that input.

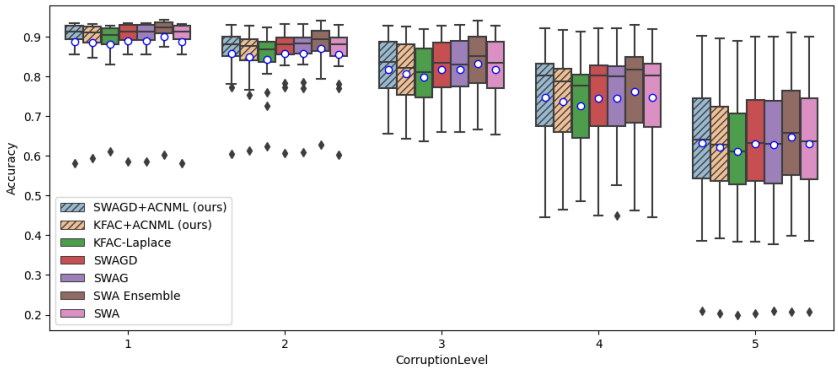
In Figures 10 and 11, We show scatter plots over 1000 randomly selected test points (from the in-distribution test set and the randomly rotated OOD images respectively) comparing the CNML normalizers, NLLs, and confidences of ACNML and naive CNML. In each scatter plot, we include a diagonal red line to illustrate where points would lie if predictions of ACNML and naive CNML matched exactly.

We additionally plot reliability diagrams for MNIST experiments in Figure 12, showing that ACNML provides very conservative predictions.

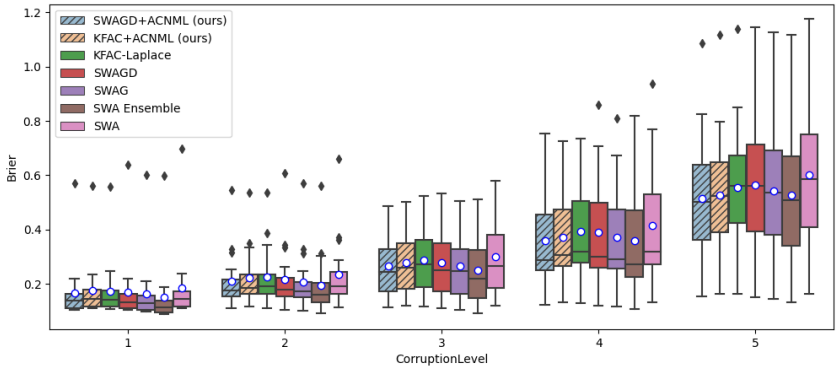
For the in-distribution test set, we see from the CNML normalizer plot that the ACNML adaptation procedure using the approximate posterior is much less constraining than using the training set, resulting in the normalizers being higher for ACNML than naive CNML for almost all inputs. This leads to excess conservatism, with ACNML almost always having lower confidence its predictions. As a result,



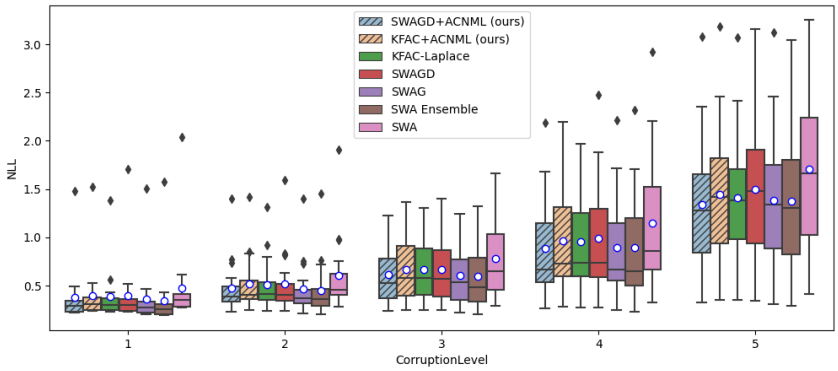
(a) CIFAR10C VGG16 ECEs (lower is better)



(b) CIFAR10C VGG16 Accuracies (higher is better)

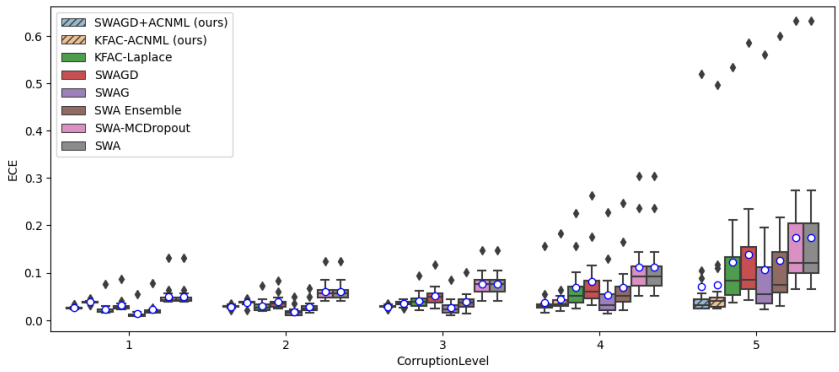


(c) CIFAR10C VGG16 Brier scores (lower is better)

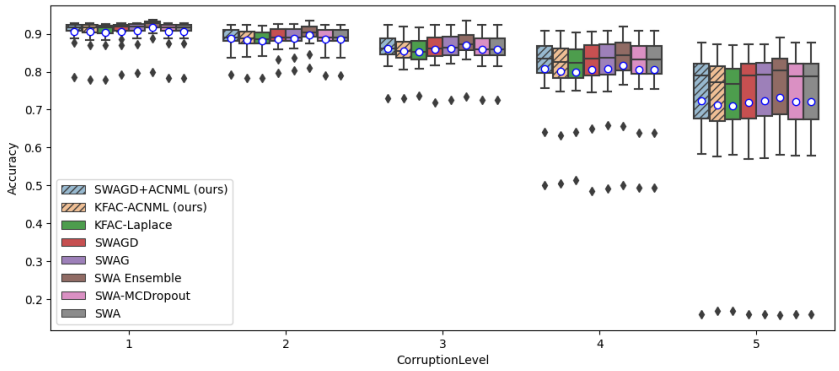


(d) CIFAR10C VGG16 NLLs (lower is better)

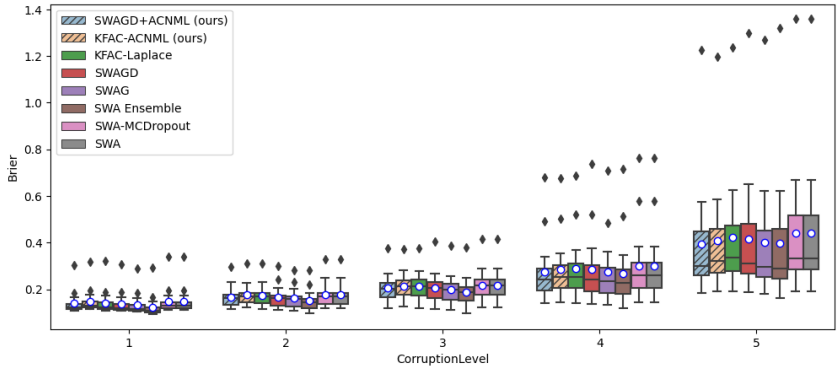
Figure 7. CIFAR10-C performance with the VGG16 architecture. Instantations of our methods are shown in stripes. Boxplots show quartiles of each statistic over all different corruption types of the given intensity, with the mean indicated by a circle. Both ACNML variants attain significantly better ECE (a) on the more severe corruptions, as the images move further out of distribution.



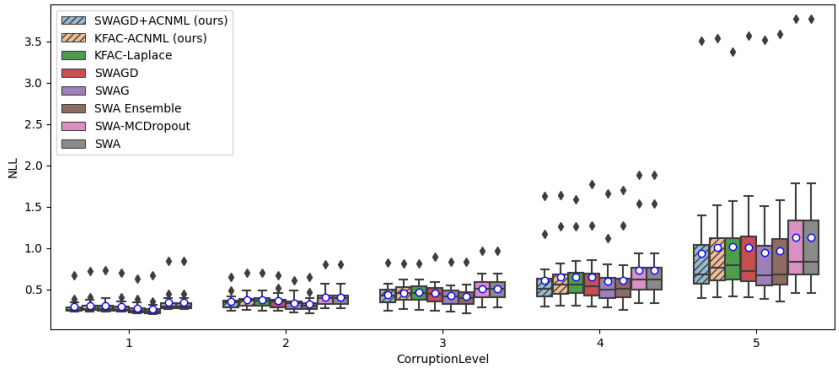
(a) CIFAR10C VGG16Drop ECEs (lower is better)



(b) CIFAR10C VGG16Drop Accuracies (higher is better)

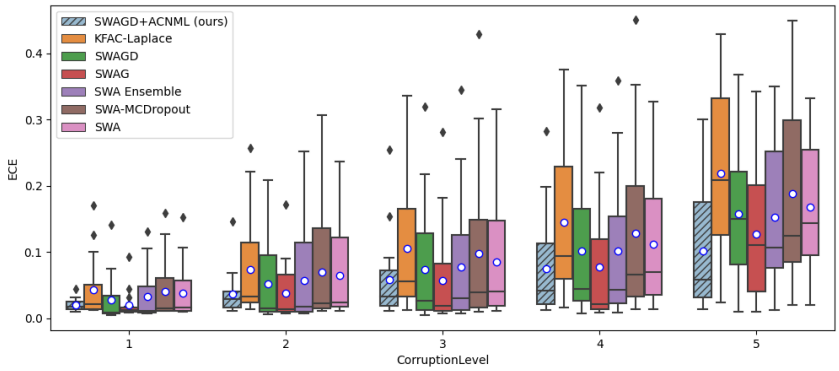


(c) CIFAR10C VGG16Drop Brier scores (lower is better)

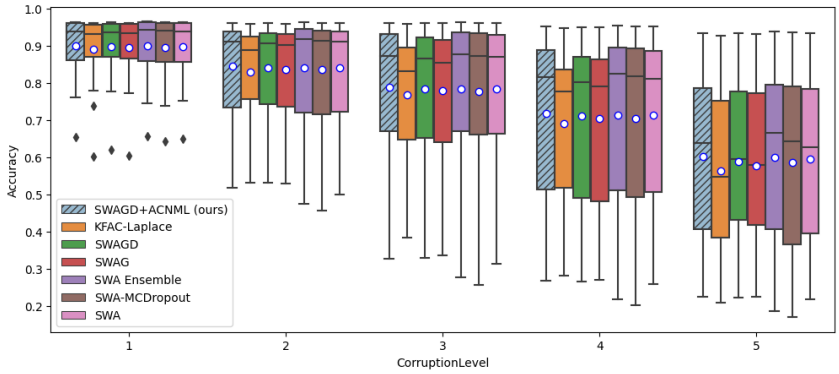


(d) CIFAR10C VGG16Drop NLLs (lower is better)

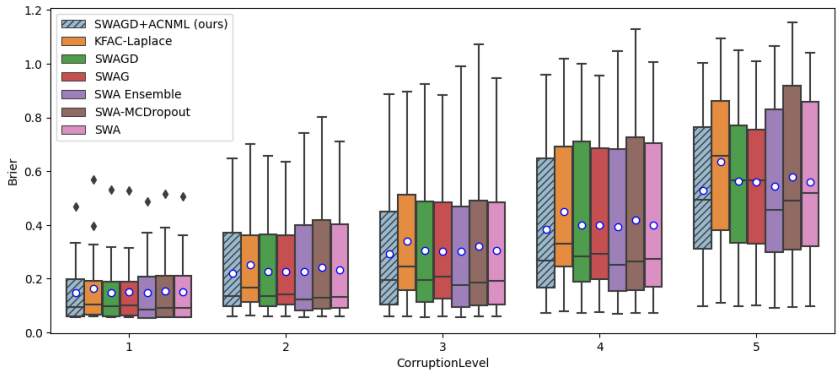
Figure 8. CIFAR10-C performance with the VGG16Drop architecture. Instantations of our methods are shown in stripes. Boxplots show quartiles of each statistic over all different corruption types of the given intensity, with the mean indicated by a circle. Again, both ACNML variants attain significantly better ECE (a) on the more severe corruptions, as the images move further out of distribution.



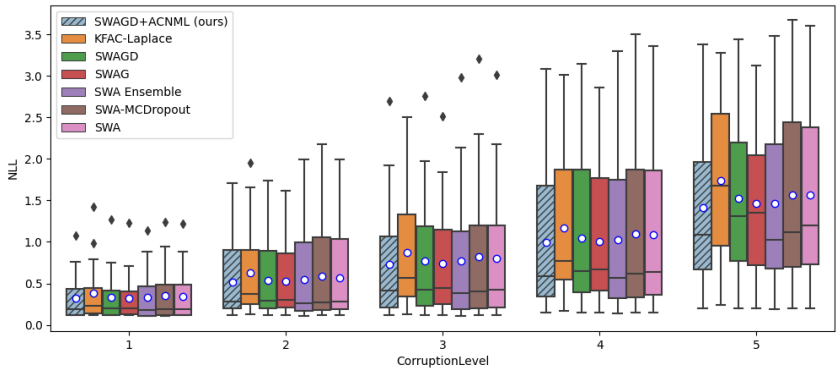
(a) CIFAR10C WRN28x10 ECEs (lower is better)



(b) CIFAR10C WRN28x10 Accuracies (higher is better)



(c) CIFAR10C WRN28x10 Brier scores (lower is better)

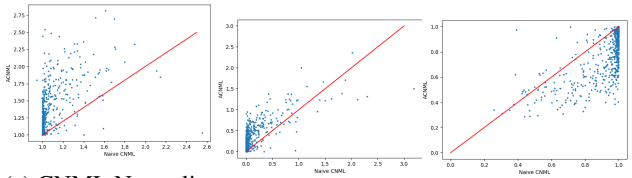


(d) CIFAR10C WRN28x10 NLLs (lower is better)

Figure 9. CIFAR10-C performance with the WideResNet28x10 architecture. Instantations of our methods are shown in stripes. Boxplots show quartiles of each statistic over all different corruption types of the given intensity, with the mean indicated by a circle. Again, we see that ACNML attains better ECE values than comparable methods on the heavier corruptions (b).

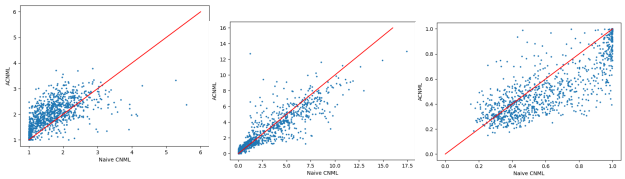
CIFAR10 Results	VGG16			WideResNet28x10		
	NLL	Accuracy	ECE	NLL	Accuracy	ECE
ACNML-SWAGD (ours)	0.2167 ± 0.0041	93.23 ± 0.09	0.0115 ± 0.0010	0.1130 ± 0.0012	96.38 ± 0.03	0.0122 ± 0.0006
ACNML-KFAC (ours)	0.2329 ± 0.0028	93.14 ± 0.08	0.0361 ± 0.0016	-	-	-
MAP (SWA)	0.2694 ± 0.0056	93.23 ± 0.13	0.0430 ± 0.0010	0.1128 ± 0.0014	96.41 ± 0.01	0.0099 ± 0.0004
SWAGD	0.2257 ± 0.0047	93.31 ± 0.04	0.0284 ± 0.0002	0.1125 ± 0.0012	96.28 ± 0.04	$\mathbf{0.0042} \pm 0.0003$
SWAG	0.2016 ± 0.0031	93.60 ± 0.10	0.0158 ± 0.0030	0.1122 ± 0.0009	96.32 ± 0.08	0.0088 ± 0.0006
KFAC-Laplace	0.2236 ± 0.0013	92.76 ± 0.11	$\mathbf{0.0097} \pm 0.0005$	0.1197 ± 0.0031	96.23 ± 0.02	0.0111 ± 0.0006
SWA-Dropout	0.2562 ± 0.0025	92.85 ± 0.14	0.0380 ± 0.0007	0.1111 ± 0.0024	96.36 ± 0.09	0.0107 ± 0.0008
SWA-Temp	0.2481 ± 0.0245	93.61 ± 0.11	0.0366 ± 0.0063	$\mathbf{0.1064} \pm 0.0004$	96.46 ± 0.04	0.0080 ± 0.0007
SGD	0.3285 ± 0.0139	93.17 ± 0.14	0.0483 ± 0.0022	0.1294 ± 0.0022	96.41 ± 0.10	0.0166 ± 0.0007
SWA-Ensemble	**0.17867**	**94.36**	0.0148	**0.1036**	**96.53**	0.0068

Table 2. **In-distribution comparative results** We see that for in-distribution performance, ACNML variants perform comparably to other methods, without large separations between most methods. Results for SWA-Temp and SGD are taken from Maddox et al. (2019).



(a) CNML Normalizers $\sum_y p_{\hat{\theta}_y}(y|x)$ (b) NLLs (c) Confidences

Figure 10. **In Distribution Comparisons between ACNML and naive CNML.** We plot scatter plots of the values of each statistic for naive CNML (x-axis) vs ACNML (y-axis), with the red line indicating Looking at the CNML normalizers, we see that the ACNML adaptation procedure using the approximate posterior is much less constraining than using the training set, resulting in the normalizers being higher for ACNML than naive CNML for almost all inputs. This leads to excess conservatism, with ACNML almost always having lower confidence its predictions, and many inputs with close to 0 NLL with naive CNML having higher NLL with ACNML.



(a) CNML Normalizers $\sum_y p_{\hat{\theta}_y}(y|x)$ (b) NLLs (c) Confidences

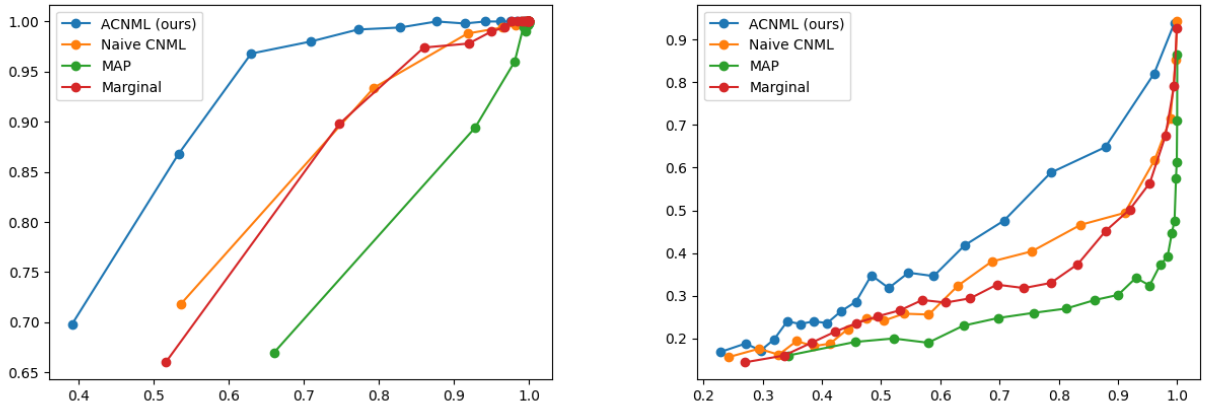
Figure 11. **OOD Comparisons between ACNML and naive CNML.** We plot scatter plots of the values of each statistic for naive CNML (x-axis) vs ACNML (y-axis). Looking at the CNML normalizers, we again see that the ACNML adaptation procedure using the approximate posterior is less constraining than using the training set, with the normalizers being higher for ACNML than naive CNML for most inputs (though to lesser extent than the in-distribution data). ACNML again outputs more conservative predictions with lower confidence on many inputs, which leads to better NLL and calibration on the OOD dataset, unlike with the in-distribution test set.

we see that on many points where naive CNML outputted confident correct answers and achieved close to 0 NLL loss, ACNML still incurs some higher losses due to its less confident predictions.

On the OOD rotated images, we again see that ACNML typically adapts more than CNML as measured by the CNML normalizers, though the difference is much less extreme compared to the in-distribution dataset. In the confidence scatter plot, we again see that ACNML tends to make lower confidence predictions than naive CNML (especially when naive CNML's predictions are confident), and as seen in Figure 13, result in ACNML having better Brier scores, NLL and calibration on the OOD inputs.

Handling multiple MLEs in CNML: Strictly speaking, the CNML distribution is not well defined when there exist multiple potential MLEs $\hat{\theta}_y$ that can output different predictions (prior references to CNML typically assume such MLEs are unique). However, the non-convexity of the objective for deep neural networks means multiple MLEs can exist, and to properly define CNML in this case, we would need to select a particular MLE to use when assigning probabilities in CNML. In line with the min-max formulation of CNML, we propose to select the MLE $\hat{\theta}_y$ that maximizes the likelihood $p_{\hat{\theta}_y}(y|x)$ of the query point and proposed label, as this is the choice that maximizes the regret for that particular label over all MLEs.

With our naive CNML instantiation, we observe that during the finetuning for each query point x and label y, the predicted probability of that label $p_\theta(y|x)$ does not monotonically increase over iterations as we might hope (since we initialize θ to be the MLE of the training set, then update it to maximize likelihood of the training set with the query point and label), but can potentially oscillate substantially throughout the finetuning process. We suspect this is due to the stochasticity in the optimization procedure from sampling minibatches of the training data, which causes the trajectory of parameters can potentially visit several different (approximate) local optima that output different predictions on the query point. While our instantiation of naive CNML simply used the parameter found at the end of 5 epochs, we additionally compare against a variant that explicitly tries to select the MLE that maximizes the likelihood



(a) MNIST Test Set

(b) Randomly Rotated MNIST (OOD data)

Figure 12. Reliability diagrams plotting confidence vs. accuracy for Bayes-by-Backprop experiments on the MNIST test set and a randomly rotated MNIST test set (OOD). ACNML's conservative predictions provided better calibrated predictions on the OOD test set.

of the proposed label. This variant heuristically uses the bset value of $p_\theta(y|x)$ over all θ encountered in the last epoch of finetuning. We see in Figure 13 that this variant, denoted naive CNML (max), gives more conservative predictions than naive CNML and improves in NLL and calibration on the more OOD rotated datasets. However, it is still not as conservative as ACNML using the Bayes-by-Backprop posterior, and so does not perform as well on the more severe rotations.

D. NMAP and ACNML

NML type methods can be extended with a prior-like regularization term on the selected parameter, resulting in Normalized Maximum a Posteriori (NMAP)(Kakade et al., 2006), also referred to as Luckiness NML (Grunwald, 2004). For a regularizer given by $\log p(\theta)$, NMAP assigns probabilities according to

$$p^{\text{NMAP}}(x^n) \propto p_{\hat{\theta}(x^n)}(x^n)$$
$$\hat{\theta}(x^n) = \operatorname*{argmax}_\theta \log p_\theta(x^n) + \log p(\theta).$$

Similarly to CNML, there are several variations on NMAP that predict slightly different distributions, but we adopt the one of the same form as our CNML. Similarly to how NML was extended to CNML, NMAP can be extended to a conditional version, again with the $\hat{\theta}$'s being chosen via MAP rather than MLE. As mentioned in Section 3.1, with a non-uniform prior, ACNML actually approximates a version of conditional NMAP, with the Bayesian prior term on the parameters corresponding to the additional regularizer.

We also note that with the calculations in section 3.1, CNML can be viewed as performing NMAP on a single new test point, with a regularizer corresponding to the posterior likelihood from the training set. In this perspective, ACNML approximates CNML by using an approximation to that training set regularizer.

E. Details of Analysis in Section 3.2

E.1. Bounding Error in Parameter Estimation

Here we state the primary theorem of Giordano et al. (2019) along with the necessary definitions and assumptions.

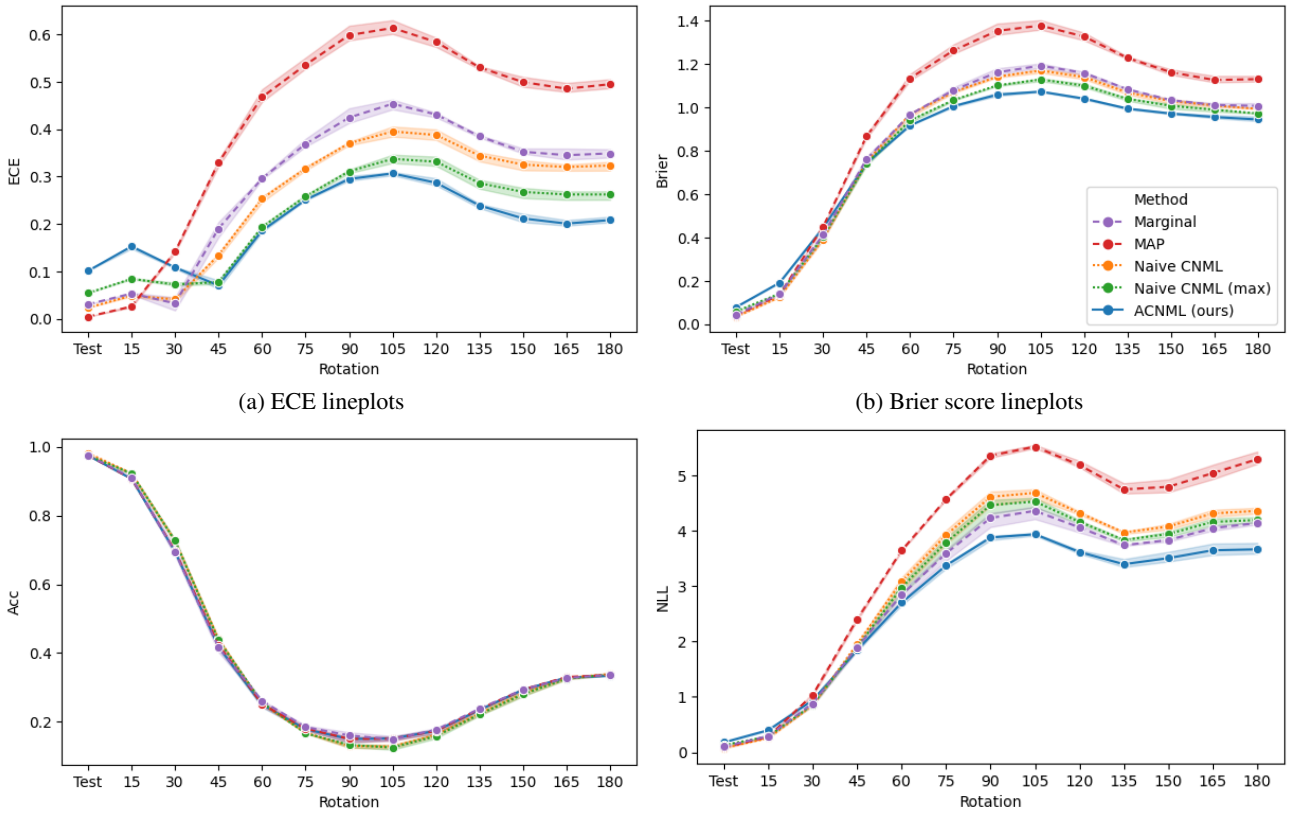
Here, we attempt to estimate an unknown parameter $\theta \in \Omega_\theta \subseteq \mathbb{R}^D$ where Ω_θ is compact. Suppose we have a dataset N datapoints and a weight vector $w_1, \ldots, w_N$. Let $g_i(\theta)$ denote the gradient of the loss at datapoint i evaluated at θ, and $h_i(\theta)$ the Hessian. We can then define

$$G(\theta, w) = \frac{1}{N}\sum_{i=1}^{N} w_i g_i(\theta) \tag{19}$$

$$H(\theta, w) = \frac{1}{N}\sum_{i=1}^{N} w_i h_i(\theta). \tag{20}$$

The MLE $\hat{\theta}(w)$ for the dataset weighted by w is given by solving for $G(\hat{\theta}(w), w) = 0$. Let 1_w denote the vector of weights consisting of all 1s. We define $\hat{\theta}_1$ to be the MLE for the whole unweighted dataset, which is equivalent to evaluating $\hat{\theta}(1_w)$ and also define the corresponding Hessian $H_1 = H(\hat{\theta}_1, 1_w)$. We now wish to estimate $\hat{\theta}(w)$ using a first order approximation around $\hat{\theta}_1$ given by

$$\hat{\theta}_{\text{IJ}}(w) = \hat{\theta}_1 - H_1^{-1} G(\hat{\theta}_1, \Delta w), \tag{21}$$



(a) ECE lineplots

(b) Brier score lineplots

(c) Accuracy lineplots

(d) NLL lineplots

Figure 13. **Expanded MNIST Results**: We include the accuracy and negative-log-likelihood metrics as well as ECE and Brier score. We see that all methods perform similarly in accuracy, and that, and ACNML also has better calibration (ECE), Brier scores, and NLLs on the more OOD datasets compared to other methods. We also additionally compare to the Naive CNML (max) method we designed to handle non-unique maximizers with naive CNML. We see that while the Naive CNML (max) variant outperforms Naive CNML on the more OOD datasets, ACNML is still more conservative, resulting in better calibrated estimates on the more severe rotations.

where we define $\Delta_w = w - 1_w$. The theorem will proceed to bound $\left\| \hat{\theta}(w) - \hat{\theta}_{\text{IJ}} \right\|_2$ for suitable weights w.

Now we further define $g(\theta) \in \mathbb{R}^{N \times D}$ to be the concatenation of all $g_i(\theta)$s and similarly for $h(\theta) \in \mathbb{R}^{N \times D \times D}$. We let $\|g(\theta)\|_p$ and $\|h(\theta)\|_p$ to refer to the p-norms when treating those as vector quantities.

Assumption 1 (Smoothness): For all $\theta \in \Omega_\theta$ each $g_n(\theta)$ is continuously differentiable.

Assumption 2 (Non-degeneracy): For all $\theta \in \Omega_\theta$, $H(\theta, 1_w)$ is nonsingular and

$$\sup_{\theta \in \Omega_\theta} \left\| H(\theta, 1_w)^{-1} \right\|_{op} \leq C_{op} \leq \infty. \tag{22}$$

Assumption 3 (Bounded averages): There exist finite constants C_g and C_h such that $\sup_{\theta \in \Omega_\theta} \frac{1}{\sqrt{N}} \|g(\theta)\|_2 \leq C_g$ and $\sup_{\theta \in \Omega_\theta} \frac{1}{\sqrt{N}} \|h(\theta)\|_2 \leq C_h$.

Assumption 4 (Local Smoothness): There exists a $\Delta_\theta > 0$ and a finite constant L_h such that $\left\| \theta - \hat{\theta}_1 \right\|_2 \leq \Delta_\theta$ implies $\frac{\left\|h(\theta) - h(\hat{\theta}_1)\right\|_2}{\sqrt{N}} \leq L_h \left\| \theta - \hat{\theta}_1 \right\|_2$.

Assumption 5 (Bounded weight averages). $\frac{1}{\sqrt{N}} \|w\|_2$ is uniformly bounded for all $w \in W$ by a finite constant C_w.

We note that assumption 2 is equivalent to H_1 being strongly positive definite. Assumption 5 is not relevant for our use cases, but is stated for completeness.

Condition 1 (Set Complexity): There exists a $\delta \geq 0$ and corresponding set $W_\delta \subseteq W$ such that

$$\max_{w \in W_\delta} \sup_{\theta \in \Omega_\theta} \left\| \frac{1}{N} \sum_{i=1}^{N} (w_i - 1) g_i(\theta) \right\|_1 \leq \delta. \tag{23}$$

$$\max_{w \in W_\delta} \sup_{\theta \in \Omega_\theta} \left\| \frac{1}{N} \sum_{i=1}^{N} (w_i - 1) h_i(\theta) \right\|_1 \leq \delta. \tag{24}$$

Condition 1 essentially describes the set of weight vectors for which $\hat{\theta}_{\text{IJ}}$ will be an accurate approximation within order δ.

Definition 1: Given assumptions 1-5, define

$$C_{\text{IJ}} = 1 + D C_w L_h C_{op} \tag{25}$$

$$\Delta_\delta = \min\{\Delta_\theta C_{op}^{-1}, \frac{1}{n} C_{\text{IJ}}^{-1} C_{op}^{-1}\}. \tag{26}$$

We now state the main theorem of Giordano et al. (2019).

Theorem (Error Bound for the approximation). Under assumptions 1-5 and condition 1,

$$\delta \leq \Delta_\delta \Rightarrow \max_{w \in W_\delta} \left\| \hat{\theta}_{\text{IJ}}(w) - \hat{\theta}(w) \right\|_2 \leq 2 C_{op}^2 C_{\text{IJ}} \delta^2. \tag{27}$$

We can now apply the above theorem to provide error bounds for a setting where we have a training set of n datapoints and wish to consider the MLE after adding a new datapoint z. The issue is that the theorem as stated bounds the error of the approximation when the approximation is centered around the uniform weighting over all the datapoints, which would be appropriate for considering the impact of *removing* datapoints from the dataset.

To apply the theorem to bound the effects of *adding* a datapoint, we have to do some slight manipulation. We apply the previous theorem with $N = n + 2$, where $g_i(\theta)$ correspond to the gradients of training data point i for i in $(1, \ldots, n)$, $g_{n+1} = -\nabla \log p_\theta(z)$, and $g_{n+2} = \nabla \log p_\theta(z)$, and similarly for the Hessians $h_i(\theta)$. We have thus added the query point to the dataset, as well as another fake point that serves to cancel out the contribution of the query point under a uniform weighting, so $G(\theta, 1_w)$ and $H(\theta, 1_w)$ are the mean gradients and Hessians for just the training set. Now supposing assumptions 1-5 are met for this problem, then we need to check condition 1 for the particular W_δ that contains the vector $\bar{w}$ of all 1s, except for a 2 in the last entry. We can then find the smallest δ that satisfies

$$\sup_{\theta \in \Omega_\theta} \left\| \frac{1}{N+2} g_{n+2}(\theta) \right\|_1 \leq \delta \tag{28}$$

$$\sup_{\theta \in \Omega_\theta} \left\| \frac{1}{N+2} h_{n+2}(\theta) \right\|_1 \leq \delta, \tag{29}$$

and so long as $\delta \leq \Delta_\delta$, applying the theorem bounds $\left\| \hat{\theta}_{\text{IJ}}(\bar{w}) - \hat{\theta}(\bar{w}) \right\|_2$.

Commentary: The above theorem gives explicit conditions for the accuracy of the approximation that we can verify for a particular training set and query point. Under assumptions that we have some limiting procedure for growing the training set such that the constants defined hold uniformly, we can extend this to an asymptotic statement to explicitly say that the approximation error decays as $O(n^{-2})$.

E.2. Bounding error in the resulting CNML distribution

We now provide the proof for Proposition 3.2, which we restate here. For notational simplicity, we ignore any dependence on the input x, which we consider fixed.

Proposition E.1 (3.2). *Suppose $z \in \mathcal{Z}$ with $|\mathcal{Z}| = k$ (for example classification with k classes). Let $\hat{\theta}_z$ be the exact MLE after appending z to the training set, and let $\tilde{\theta}_z$ be an approximate MLE with $\left\| \hat{\theta}_z - \tilde{\theta}_z \right\| \leq \delta$ for all z. Further suppose $\log p_\theta(z)$ is L-Lipschitz in θ.*

Denote the exact CNML distribution $p_{CNML}(z) \propto p_{\hat{\theta}_z}(z)$ and an approximate CNML distribution $p_{ACNML}(z) \propto$

$p_{\tilde{\theta}_z}(z)$. *Then, we have the bound*

$$\sup_z |\log p_{CNML}(z) - \log p_{ACNML}(z)| \leq 2L\delta. \tag{30}$$

Proof. The assumed bound $\left\|\hat{\theta}_z - \tilde{\theta}_z\right\|_2 \leq \delta$ combined with L-Lipschitzness implies a bound on differences of logits of each class

$$\left|\log p_{\hat{\theta}_z}(z) - \log p_{\tilde{\theta}_z}(z)\right| \leq L\delta. \tag{31}$$

We note that the log probabilities of the exact CNML distribution p_{CNML} (p_{ACNML} is given by a similar expression using $\tilde{\theta}_z$ instead of $\hat{\theta}_z$) is given by

$$\log p_{\text{CNML}}(z) = \log p_{\hat{\theta}_z}(z) - \log \sum_{z' \in \mathcal{Z}} p_{\hat{\theta}_{z'}}(z'). \tag{32}$$

For any $z \in \mathcal{Z}$, we can then expand, apply the triangle inequality and then Equation 31 to obtain

$$\begin{aligned}
&|\log p_{\text{CNML}}(z) - \log p_{\text{ACNML}}(z)| \\
&= |\log p_{\hat{\theta}_z}(z) - \log p_{\tilde{\theta}_z}(z) \\
&- \log \sum_{z' \in \mathcal{Z}} p_{\hat{\theta}_{z'}}(z') + \log \sum_{z' \in \mathcal{Z}} p_{\tilde{\theta}_{z'}}(z')| && (33) \\
&\leq \left|\log p_{\hat{\theta}_z}(z) - \log p_{\tilde{\theta}_z}(z)\right| \\
&+ \left|\log \sum_{z' \in \mathcal{Z}} p_{\hat{\theta}_{z'}}(z') - \log \sum_{z' \in \mathcal{Z}} p_{\tilde{\theta}_{z'}}(z')\right| && (34) \\
&\leq L\delta + \left|\log \sum_{z' \in \mathcal{Z}} p_{\hat{\theta}_{z'}}(z') - \log \sum_{z' \in \mathcal{Z}} p_{\tilde{\theta}_{z'}}(z')\right|. && (35)
\end{aligned}$$

We now bound the difference between the log-normalizers $\left|\log \sum_{z'} p_{\hat{\theta}_{z'}}(z') - \log \sum_{z'} p_{\tilde{\theta}_{z'}}(z')\right|$.

We first let $p_{\min}(z) = \min\{p_{\hat{\theta}_z}(z), p_{\tilde{\theta}_z}(z)\}$ and $p_{\max}(z) = \max\{p_{\hat{\theta}_z}(z), p_{\tilde{\theta}_z}(z)\}$, and note that Equation 31 implies $\log p_{\max}(z) \leq \log p_{\min}(z) + L\delta$ for all z. We then bound the difference in log-normalizers

$$\begin{aligned}
&\left|\log \sum_{z' \in \mathcal{Z}} p_{\hat{\theta}_{z'}}(z') - \log \sum_{z' \in \mathcal{Z}} p_{\tilde{\theta}_{z'}}(z')\right| \\
&\leq \log \sum_{z' \in \mathcal{Z}} p_{\max}(z') - \log \sum_{z' \in \mathcal{Z}} p_{\min}(z') && (36) \\
&= \log \frac{\sum_{z' \in \mathcal{Z}} p_{\max}(z')}{\sum_{z' \in \mathcal{Z}} p_{\min}(z')} && (37) \\
&= \log \frac{\sum_{z' \in \mathcal{Z}} \exp(\log p_{\max}(z'))}{\sum_{z' \in \mathcal{Z}} p_{\min}(z')} && (38) \\
&\leq \log \frac{\sum_{z' \in \mathcal{Z}} \exp(\log p_{\min}(z') + L\delta)}{\sum_{z' \in \mathcal{Z}} p_{\min}(z')} && (39) \\
&= \log \frac{\exp(L\delta) \sum_{z' \in \mathcal{Z}} p_{\min}(z')}{\sum_{z' \in \mathcal{Z}} p_{\min}(z')} && (40) \\
&= L\delta. && (41)
\end{aligned}$$

Plugging back into Equation 37, we have the following bound for all $z \in \mathcal{Z}$

$$|\log p_{\text{CNML}}(z) - \log p_{\text{ACNML}}(z)| \leq 2L\delta. \tag{42}$$

□