APPROXIMATING FULL CONFORMAL PREDICTION FOR NEURAL NETWORK REGRESSION WITH GAUSS-NEWTON INFLUENCE

Dharmesh Tailor[1]*, **Alvaro H.C. Correia**[2], **Eric Nalisnick**[3], **Christos Louizos**[2]
[1]University of Amsterdam [2]Qualcomm AI Research† [3]Johns Hopkins University

ABSTRACT

Uncertainty quantification is an important prerequisite for the deployment of deep learning models in safety-critical areas. Yet, this hinges on the uncertainty estimates being *useful* to the extent the prediction intervals are well-calibrated and sharp. In the absence of inherent uncertainty estimates (*e.g.*, pretrained models predicting only point estimates), popular approaches that operate post-hoc include Laplace's method and split conformal prediction (split-CP). However, Laplace's method can be miscalibrated when the model is misspecified and split-CP requires sample splitting, and thus comes at the expense of statistical efficiency. In this work, we construct prediction intervals for neural network regressors post-hoc without held-out data. This is achieved by approximating the *full* conformal prediction method (full-CP). Whilst full-CP nominally requires retraining the model for every test point and candidate label, we propose to train just once and locally perturb model parameters using Gauss-Newton influence to approximate the effect of retraining. Coupled with linearization of the network, we express the absolute residual nonconformity score as a piecewise linear function of the candidate label allowing for an efficient procedure that avoids the exhaustive search over the output space. On standard regression benchmarks and bounding box localization, we show the resulting prediction intervals are locally-adaptive and often tighter than those of split-CP.

1 INTRODUCTION

Despite impressive advancements in machine learning, most models, particularly neural networks, are still designed and trained to provide only point estimates, lacking the ability to rigorously quantify uncertainty in their predictions. This poses a significant challenge, as reliable decision-making depends on trustworthy uncertainty representation. This need has drawn increased attention to uncertainty quantification in machine learning research, especially as the use of these models becomes more widespread and permeate safety-sensitive fields such as healthcare and autonomous driving.

Conformal Prediction (CP) (Vovk et al., 2005) is a class of uncertainty quantification methods that represent uncertainty via prediction intervals. These intervals intuitively convey the degree of uncertainty—the larger the interval, the greater the uncertainty. What sets CP apart is a rigorous, *distribution-free coverage* guarantee: conformal prediction intervals contain the true label with a probability of at least $1 - \alpha$, where α is a user-defined miscoverage rate. In recent years, a variant known as split (or inductive) CP (Papadopoulos et al., 2002) has gained traction in the machine learning community due to its ease of use and low computational cost. Split-CP, as the name suggests, divides the available data into training and calibration sets, using the former to fit a model and the latter to construct prediction intervals. Yet, split-CP is statistically inefficient as it cannot leverage all available data for model fitting, which hinders overall model performance. In contrast, full (or transductive) CP uses all data for both training and calibration but incurs high computational costs. It requires retraining the model multiple times—once for each test point and possible label—which is prohibitively expensive for deep learning applications.

*Work done while at Qualcomm AI Research. †Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc. and/or its subsidiaries.

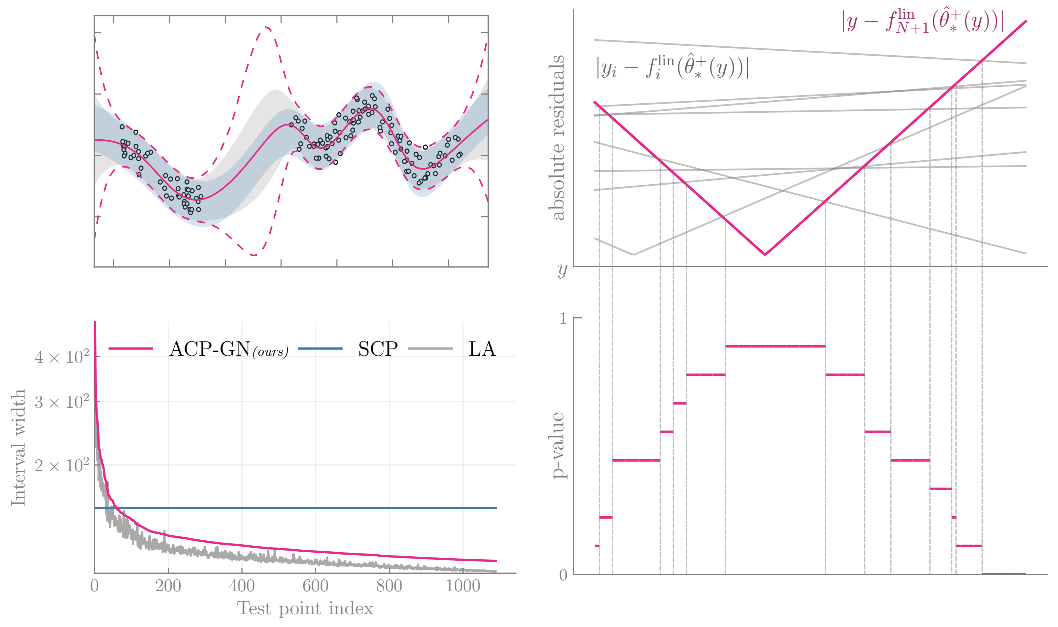

Figure 1: Our approx. full-CP via Gauss-Newton influence (ACP-GN) produces *adaptive* intervals (bottom left figure)—similar to Bayes via Laplace approximation (LA)—while satisfying coverage as seen in the high-overlap with split-CP (SCP) close to the data (white dots with black edge in top left figure). In the top right, we show the absolute residuals for the training points (in gray) and test point (in pink) as function of the postulated label for the test point $\mathbf{x}_{N+1}$. The conformal p-values, in the bottom right, only change when the pink line crosses a gray line, the so-called "changepoints", reducing the search space over candidate labels to incorporate in the prediction set.

In this work, we scale full-CP to neural network regression by (i) eliminating the need for retraining through a Gauss-Newton influence approximation, and (ii) avoiding an exhaustive search over the label space via network linearization. This gives a new, scalable full-CP method for neural network regression we dub *approximate full-CP via Gauss-Newton influence* (ACP-GN). As a second contribution, we show the same tools can be applied to enhance split-CP via normalization, yielding a new adaptable split-CP method we call SCP-GN. We validate ACP-GN and SCP-GN in several regression and bounding box prediction tasks, showing they satisfy coverage in most cases and produce tight, adaptable prediction intervals (see Fig. 1 for results with ACP-GN).

2 Background on Conformal Prediction

Neural network regression follows the canonical empirical risk minimization framework. Given a dataset $\mathcal{D}_N := \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ consisting of N examples with inputs $\mathbf{x}_i \in \mathbb{R}^I$ and targets $y_i \in \mathbb{R}$, we minimize the (regularized) empirical risk:

$$\boldsymbol{\theta}_* = \arg\min_{\boldsymbol{\theta}} \left(\sum_{i=1}^{N} \ell(y_i, f_i(\boldsymbol{\theta})) + \tfrac{1}{2}\delta \left\|\boldsymbol{\theta}\right\|^2 \right) \tag{1}$$

where $f_i(\boldsymbol{\theta})$ is shorthand for $f(\mathbf{x}_i; \boldsymbol{\theta})$, the output of a deep neural network (DNN) at $\mathbf{x}_i$ with parameters $\boldsymbol{\theta} \in \mathbb{R}^D$, $\ell(y, f)$ is a loss function, and δ is an L_2 regularizer (*i.e.* weight decay). In this work, we restrict our attention to the squared-error loss: $\ell(y, f) = \frac{1}{2}(y - f)^2$. As is standard practice, the problem in Eq. (1) is approached using stochastic-gradient methods. For an unseen input $\mathbf{x}_{N+1}$, this gives us a point prediction $f_{N+1}(\boldsymbol{\theta}_*)$. However, in an ideal scenario, we would like a prediction *interval* whose width reflects the uncertainty associated with that input. This is where conformal prediction (Vovk et al., 2005) comes into play. It provides a framework for constructing prediction intervals while satisfying the following frequentist coverage guarantee known as *marginal coverage*

$$\mathcal{P}\left(y_{N+1} \in C_\alpha(\mathbf{x}_{N+1})\right) \geq 1 - \alpha, \tag{2}$$

where y_{N+1} is the unseen target, $C_\alpha(\mathbf{x}_{N+1})$ is the prediction interval with desired miscoverage rate $\alpha \in (0, 1)$, and the probability is over all samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N+1}$, hence the name marginal coverage.

The prediction interval $C_\alpha(\mathbf{x}_{N+1})$ is constructed using *nonconformity* scores, which quantify how unusual a sample $(\mathbf{x}_i, y_i)$ is compared to other samples in a set. In the context of regression, which is the focus of this paper, the absolute residual $R_i = R(\mathbf{x}_i, y_i) = |y_i - f_i(\boldsymbol{\theta})|$ is the most common score (Kato et al., 2023). Given the data $\mathcal{D}_N$ and a score function, $C_\alpha(\mathbf{x}_{N+1})$ is defined as

$$C_\alpha(\mathbf{x}_{N+1}) = \{y \in \mathbb{R} : \pi(y) \leq \lceil (1-\alpha)(N+1) \rceil\}, \tag{3}$$

where $\pi(y) = \sum_{i=1}^{N+1} \mathbb{1}\{R_i \leq R(\mathbf{x}_{N+1}, y)\}$ is the rank of $R(\mathbf{x}_{N+1}, y)$ among the other N residuals. Remarkably, the only requirement for prediction intervals constructed as in Eq. (3) to satisfy marginal coverage is that the set of scores $\{R(\mathbf{x}_i, y_i)\}_{i=1}^{N+1}$ be exchangeable. This is equivalent to assuming the data is exchangeable and the score function (and consequently the regressor) preserves this exchangeability by treating data points symmetrically. Thus, Eq. (2) is a distribution-free guarantee that remains valid even if the model is misspecified. This contrasts with Bayesian methods, the dominant approach for uncertainty quantification in deep learning, which are often poorly calibrated under model misspecification (Dawid, 1982; Fraser, 2011; Grünwald & van Ommen, 2017).

2.1 Split Conformal Prediction

As stated in the introduction, CP methods can generally be categorized into split-CP and full-CP variants. The primary distinction between these two families of methods lies in how they ensure the exchangeability of the set of scores $\{R(\mathbf{x}_i, y_i)\}_{i=1}^{N} \cup \{R(\mathbf{x}_{N+1}, y)\}$. Essentially, this means that all samples, including the test sample $(\mathbf{x}_{N+1}, y_{N+1})$, must exert the same influence over the prediction model $f(\cdot; \boldsymbol{\theta})$. Split-CP offers the simplest and most computationally efficient solution. By keeping the model fixed when computing the scores of each of the $N + 1$ samples, it ensures the scores are exchangeable, as the score function is applied elementwise. However, this approach is data inefficient because it prevents using the first N samples to fit the model, requiring a separate training dataset. This statistical inefficiency can negatively impact the quality of the final prediction intervals in two key ways. Since any conformal prediction method is designed to achieve coverage as described in Eq. (2), different approaches are compared in terms of their *efficiency*—how small the prediction intervals are—and *adaptability*—how much the size of the prediction intervals varies across samples. To be informative about both the true label and predictive uncertainty, prediction intervals must be both efficient and adaptable. However, in split-CP, efficiency is compromised because not all data is used for calibration, while fixing the model parameters reduces adaptability.

2.2 Full Conformal Prediction

Full-CP uses all available data for both training the model and computing prediction intervals. The main challenge in full-CP is that exchangeability of scores requires treating all data points symmetrically, meaning the model should be fit on $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ as well as $(\mathbf{x}_{N+1}, y_{N+1})$. Since y_{N+1} is unknown a priori, this necessitates retraining the model for all possible values y_{N+1} can take. To be precise, full-CP requires the following modification to the optimization problem in Eq. (1)

$$\boldsymbol{\theta}_*^+(y) = \arg\min_{\boldsymbol{\theta}} \left(\sum_{i=1}^{N} \ell(y_i, f_i(\boldsymbol{\theta})) + \ell(y, f_{N+1}(\boldsymbol{\theta})) + \tfrac{1}{2}\delta \|\boldsymbol{\theta}\|^2 \right), \tag{4}$$

where $\boldsymbol{\theta}_*^+(y)$ is the optimal model parameters for the augmented training set $\mathcal{D}_{N+1}(y) := \mathcal{D}_N \cup \{(\mathbf{x}_{N+1}, y)\}$ that includes the test point $\mathbf{x}_{N+1}$ plus a candidate label y for y_{N+1}. With a slight abuse of notation, we use $R_i(y)$ to denote the residual with model parameters $\boldsymbol{\theta}_*^+(y)$, that is,

$$R_i(y) = |y_i - f_i(\boldsymbol{\theta}_*^+(y))| \;\; \forall i = 1, \ldots, N \;\text{ and }\; R_{N+1}(y) = |y - f_{N+1}(\boldsymbol{\theta}_*^+(y))|. \tag{5}$$

From here we can construct prediction intervals as in Eq. (3), but now the residuals vary for each test point $\mathbf{x}_{N+1}$ and candidate label $y \in \mathbb{R}$. As mentioned before, this has two major limitations. Firstly it requires retraining the model for every candidate label y which is infeasible for DNNs. Secondly in theory the method asks to consider an uncountable set (*i.e.* all real numbers). Therefore, in practice a finite grid of possible labels for y is used, typically delimited by the training targets. Evidently, the grid imposes computation-precision trade-off and has implications for the coverage if a valid candidate happens to lie between two grid points (Chen et al., 2018). In a few cases, the prediction set can be computed efficiently and exactly without the need to try candidate labels of y. In addition, this procedure only depends on a single fit on the original (unaugmented) dataset which can be efficiently updated not only for variations in y but also for different inputs $\mathbf{x}_{N+1}$. These include the Lasso (Lei, 2019), k-Nearest Neighbours Regression (Papadopoulos et al., 2011), and ridge regression (Nouretdinov et al., 2001; Burnaev & Vovk, 2014). There are also approaches based on homotopy continuation methods (Ndiaye & Takeuchi, 2019) and algorithmic stability (Ndiaye, 2022) that hold in more general settings. In the following section, we review conformalized ridge regression which is the basis of our approximate full-CP procedure.

2.3 Conformalized Ridge Regression (CRR)

Ridge regression is a special case of Eq. (1) where we have a linear model $f_i(\boldsymbol{\theta}) := \mathbf{x}_i^\top \boldsymbol{\theta}$. In this case, the absolute residual can be written as a piecewise linear function of the candidate y with $R_i(y) = |a_i + b_i y|$, where a_i and b_i are coefficients capturing information from the training data and test point, resp. It is then convenient to express the rank as $\pi(y) = \sum_{i=1}^{N+1} \mathbb{1}\{y \in S_i\}$, with

$$S_i = \{y : R_i(y) \leq R_{N+1}(y)\} = \{y : |a_i + b_i y| \leq |a_{N+1} + b_{N+1} y|\}. \tag{6}$$

Each set S_i can be an interval, a point, a ray, a union of two rays, the real line, or the empty set. Eq. (6) suggests that the rank for a given y can only change at points where $R_i(y) = R_{N+1}(y)$ to which we refer as "changepoints". For the absolute residual, these changepoints fall into one of the following two cases, where we assume $b_i \geq 0$ (if needed, multiplying a_i and b_i by -1):

- If $b_i \neq b_{N+1}$, then S_i is an interval or a union of two rays, and we have two changepoints, $-(a_i - a_{N+1})/(b_i - b_{N+1})$ and $-(a_i + a_{N+1})/(b_i + b_{N+1})$.
- If $b_i = b_{N+1} \neq 0$, then S_i is a ray, unless $a_i = a_{N+1}$, in which case $S_i = \mathbb{R}$. Here we have single changepoint $-(a_i + a_{N+1})/2b_i$.

This leads to an exact form of the prediction set by taking the union of finitely many intervals and rays whose endpoints are given by the changepoints sorted in increasing order. Given the changepoints, different implementations have been proposed with varying time complexity. For the smaller datasets, we use the ridge regression confidence machine algorithm (Nouretdinov et al., 2001; Vovk et al., 2005), which uses the absolute residual and changepoints described above.

There is also a simpler, asymmetric version of CRR that uses the signed residuals (Burnaev & Vovk, 2014). It allows us to compute the lower and upper bounds of $C_\alpha(\mathbf{x}_{N+1})$ separately, using residuals $f_i(\boldsymbol{\theta}) - y_i$ for the lower bound, and $y_i - f_i(\boldsymbol{\theta})$ for the upper bound. This is the version we use for the larger datasets and that appears in the depiction of our method in Alg. 2, where we use l_i and u_i to denote the changepoints for the lower and upper bounds, resp. When using signed residuals, S_i is either a ray with changepoint $l_i = u_i = (a_i - a_{N+1})/(b_{N+1} - b_i)$ if $b_{N+1} - b_i > 0$ or otherwise, $S_i = \mathbb{R}$ with $l_i = -\infty$ and $u_i = \infty$.

Finally, we need to write down the expression for coefficients a_i and b_i. Using the Sherman-Morrison formula, a widely-used tool of the regressions diagnostics literature (Cook, 1977), Burnaev & Vovk (2014) showed that the required coefficients $a_1, \ldots, a_{N+1}$ and $b_1, \ldots, b_{N+1}$ for the CRR procedure can be efficiently computed for different $\mathbf{x}_{N+1}$ by a simple rank-1 update or *perturbation* to the ridge solution on $\mathcal{D}_N$ (see App. A for derivation):

$$y_i - \mathbf{x}_i^T \boldsymbol{\theta}_*^+(y) = \underbrace{y_i - \mathbf{x}_i^T \boldsymbol{\theta}_* + \frac{h_{i,N+1}}{1 + h_{N+1}} \mathbf{x}_{N+1}^\top \boldsymbol{\theta}_*}_{a_i} \underbrace{- \frac{h_{i,N+1}}{1 + h_{N+1}}}_{b_i} y \tag{7}$$

$$y - \mathbf{x}_{N+1}^T \boldsymbol{\theta}_*^+(y) = \underbrace{-\frac{1}{1 + h_{N+1}} \mathbf{x}_{N+1}^\top \boldsymbol{\theta}_*}_{a_{N+1}} + \underbrace{\frac{1}{1 + h_{N+1}}}_{b_{N+1}} y \tag{8}$$

where $h_{N+1} = \mathbf{x}_{N+1}^\top \mathbf{H}_*^{-1} \mathbf{x}_{N+1}$ with Hessian matrix $\mathbf{H}_* = \sum_{i=1}^N \mathbf{x}_i \mathbf{x}_i^\top + \delta \mathbf{I}$.

2.4 Normalized Nonconformity Scores

In the previous section, we derived CRR using the nonconformity score given in Eq. (5), which is often referred to as add-one-in (AOI). In this section, we also consider the leave-one-out (LOO) and studentized scores, which lead to the variants CRR-deleted and CRR-studentized and their corresponding extensions to neural network regression given by our approximate full-CP method. All those variants are valid choices (Vovk et al., 2005; Shafer & Vovk, 2008), and the literature is not conclusive regarding which one is to be preferred (Fong & Holmes, 2021; Fontana et al., 2023). Yet, in our neural network regression experiments, the studentized variant outperformed the other two.

In the leave-one-out (LOO) variety (also known as jackknife), scores R_i^{LOO} are computed by excluding the i^{th} data point from the augmented data $\mathcal{D}_{N+1}(y)$ before retraining the model. The only exception being R_{N+1}^{LOO}, which requires no retraining, as $\boldsymbol{\theta}_*$ already ignores the $(N+1)^{\text{th}}$ data point. In the case of ridge regression with absolute residuals, we can derive the jackknife score from the standard one as (Vovk et al., 2005)

$$R_i^{\text{LOO}} = {}^{R_i}\!/\!_{(1-\bar{h}_i)}, \tag{9}$$

where we have included the leverage score with respect to the augmented problem $\bar{h}_i = \mathbf{x}_i^\top \bar{\mathbf{H}}_*^{-1} \mathbf{x}_i$, with $\bar{\mathbf{H}}_* = \sum_{i=1}^{N+1} \mathbf{x}_i \mathbf{x}_i^\top + \delta \mathbf{I}$. The leverage score (Chatterjee & Hadi, 1986) can be viewed as a diagnostics measure for measuring the sensitivity of the prediction to changes in the target. The formula is a consequence of the exact rank-1 updates available in ridge regression. This leads to the deleted-CRR method which proceeds in the same way as standard CRR except it uses the normalized coefficients: $a_i \leftarrow a_i/(1-\bar{h}_i)$ and $b_i \leftarrow b_i/(1-\bar{h}_i)$ for $i = 1, \dots, N+1$. Such relations can also be used to recover the standard score starting from the jackknife one:

$$R_i = {}^{R_i^{\text{LOO}}}\!/\!_{\left(1+\bar{h}_i^{\setminus i}\right)}, \tag{10}$$

where $\bar{h}_i^{\setminus i} = \mathbf{x}_i^\top \bar{\mathbf{H}}_*^{\setminus i} \mathbf{x}_i$ with $\bar{\mathbf{H}}_*^{\setminus i} = \sum_{j=1, j\neq i}^{N+1} \mathbf{x}_j \mathbf{x}_j^\top + \delta \mathbf{I}$. Papadopoulos (2024) gave an interpretation of this relation as locally-weighted conformal prediction (Papadopoulos et al., 2008) where the expression in the denominator can be seen as the leave-one-out predictive variance from a Bayesian perspective. This can be seen as a measure of the difficulty of the i^{th} example. Similar normalizations can also be applied to nonconformity scores in a split-CP framework (Vovk et al., 2005). Finally, the studentized-CRR can be interpreted as a compromise between standard and deleted-CRR. It uses a similar normalization and defines nonconformity scores as

$$R_i^{\text{student}} = {}^{R_i}\!/\!_{\sqrt{1-\bar{h}_i}}. \tag{11}$$

This transformation is applied analogously to the CRR coefficients with a_i and b_i.

3 Approximate Full-CP for Neural Network Regression

In previous work, (Martinez et al., 2023) used influence function (Jaeckel, 1972; Cook & Weisberg, 1980) to approximate the retraining step in full-CP by locally perturbing the solution on the unaugmented dataset. This ensures that a neural network is trained only once on the original data. However, this was restricted to classification problems and so their approach requires discretizing the (continuous) target space. This introduces an additional computational burden and must be done in a careful way to avoid further increasing the coverage gap (Chen et al., 2018). We propose to use a local perturbation closely related to influence function alongside network linearization in order to express the residual nonconformity score as a linear function of the candidate label that recovers Eqs. (7) and (8) as a special case. Then by leveraging the conformalized ridge regression framework (Nouretdinov et al., 2001), we can eliminate the need to specify a grid of candidate labels.

We propose *Gauss-Newton influence* to approximate the solution to the augmented problem $\hat{\boldsymbol{\theta}}_*^+(y) \approx \boldsymbol{\theta}_*^+(y)$,

$$\hat{\boldsymbol{\theta}}_*^+(y) = \boldsymbol{\theta}_* + \frac{\hat{e}_{N+1}(y)}{1 + \hat{h}_{N+1}} \mathbf{H}_{\text{GN}}^{-1} \boldsymbol{\phi}_{N+1}, \tag{12}$$

where $\boldsymbol{\phi}_i := \nabla_\theta f_i(\boldsymbol{\theta}_*)^\top$ is the Jacobian of the neural network at $\boldsymbol{\theta}_*$, $\mathbf{H}_{\text{GN}} = \sum_{i=1}^N \boldsymbol{\phi}_i \boldsymbol{\phi}_i^\top + \delta \mathbf{I}$ is the Gauss-Newton approximation to the Hessian, $\hat{e}_{N+1}(y) = y - f_{N+1}(\boldsymbol{\theta}_*)$ is the residual for the $(N+1)^{\text{th}}$ example and $\hat{h}_{N+1} = \boldsymbol{\phi}_{N+1}^\top \mathbf{H}_{\text{GN}}^{-1} \boldsymbol{\phi}_{N+1}$ can be interpreted as a kind of *generalized* leverage score (Wei et al., 1998). This is a specialization of Newton-step (NS) influence (Pregibon, 1981; Beirami et al., 2017), a technique closely related to influence function (Jaeckel, 1972; Koh & Liang, 2017), by approximating the Hessian by the Gauss-Newton matrix (Martens, 2010) followed by a rank-one update (see App. B for the derivation). Whilst such influence measures are commonly used for estimating the effect of *removing* a single example (or group of examples) on the model, we extend this for *adding* an example, that is add-one-in (AOI) estimation. Previous work has shown that NS influence more accurately estimates the solution to the modified problem than influence function (Pregibon, 1981; Rad & Maleki, 2020). However, at present influence function is the more common choice since in its standard form NS influence requires recomputing and inverting the Hessian matrix whenever a different target effect (*e.g.* change in removed example(s)) is desired, thus incurring a higher computational cost. Our use of the Gauss-Newton approximation leads to a form amenable to a rank-one update, similar to generalized linear models (Pregibon, 1981), bringing down the complexity to that of influence function.

Next we show that using Eq. (12) along with linearization of the neural network, we can approximate the residual as a linear function of the postulated label y recovering an identical form to that of conformalized ridge regression in Eqs. (7) and (8),

$$y_i - f_i(\boldsymbol{\theta}_*^+(y)) \approx \underbrace{y_i - f_i(\boldsymbol{\theta}_*) + \frac{\hat{h}_{i,N+1}}{1+\hat{h}_{N+1}} f_{N+1}(\boldsymbol{\theta}_*)}_{a_i} \underbrace{- \frac{\hat{h}_{i,N+1}}{1+\hat{h}_{N+1}}}_{b_i} y \tag{13}$$

$$y - f_{N+1}(\boldsymbol{\theta}_*^+(y)) \approx \underbrace{-\frac{1}{1+\hat{h}_{N+1}} f_{N+1}(\boldsymbol{\theta}_*)}_{a_{N+1}} + \underbrace{\frac{1}{1+\hat{h}_{N+1}}}_{b_{N+1}} y \tag{14}$$

where $\hat{h}_{i,N+1} = \boldsymbol{\phi}_i^\top \mathbf{H}_{\text{GN}}^{-1} \boldsymbol{\phi}_{N+1}$ (see App. C for the derivation and App. D for the extension to the multi-output setting). This is not surprising as it is known that NS influence recovers the exact leave-one-out diagnostics in linear regression (Allen, 1974; Cook, 1977), which is not the case for influence function. The coefficients can be readily adapted for normalized nonconformity scores as outlined in Sec. 2.4. Our use of network linearization is akin to the “direct” approach used in Martinez et al. (2023) to approximate the nonconformity scores on the augmented solution directly.

In Alg. 2, we outline a simplified but complete algorithmic depiction of our method, which we dub *approximate full-CP via Gauss-Newton influence* (ACP-GN). We contrast it with standard full-CP in Alg. 1, highlighting the costly grid search and retraining steps (see App. G for a time complexity analysis). Note that in ACP-GN there is no grid search and the model is fit only once on all available data $\mathcal{D}_N$, as indicated in blue. At test time, instead of retraining the model for each $\mathbf{x}_{N+1}$, ACP-GN only computes the CRR coefficients $\{(a_i, b_i)\}_{i=1}^{N+1}$ using Gauss-Newton influence as in Eq. (13) and Eq. (14). These are then used to derive the prediction set in closed form either using the method of (Nouretdinov et al., 2001) or the asymmetric one of (Burnaev & Vovk, 2014) that we show in Alg. 2.

Algorithm 1: Standard Full-CP.

```
for each test point $\mathbf{x}_{N+1}$ do
    for each $y$ in a given grid do
        optimize $\boldsymbol{\theta}_*^+(y)$ as in Eq. (4)
        $R_{N+1}(y) = |y - f_{N+1}(\boldsymbol{\theta}_*^+(y))|$
        for $i \in \{1, \dots N\}$ do
            $R_i(y) = |y_i - f_i(\boldsymbol{\theta}_*^+(y))|$
        $\pi(y) = \sum_{i=1}^{N+1} \mathbb{1}\{R_i(y) \le R_{N+1}(y)\}$
        if $\pi(y) \le \lceil (1-\alpha)(N+1) \rceil$ then
            include $y$ in $C_\alpha(\mathbf{x}_{N+1})$
```

Algorithm 2: ACP-GN (ours).

```
optimize $\boldsymbol{\theta}_*$ as in Eq. (1)
for each test point $\mathbf{x}_{N+1}$ do
    compute $a_{N+1}, b_{N+1}$ as in Eq. (14)
    for $i \in \{1, \dots N\}$ do
        compute $a_i, b_i$ as in Eq. (13)
        if $b_{N+1} - b_i > 0$ then
            $l_i = u_i = (a_i - a_{N+1})/(b_{N+1} - b_i)$
        else
            $l_i = -\infty$ and $u_i = \infty$
    sort $\{l_i\}_{i=1}^N$ and $\{u_i\}_{i=1}^N$ in ascending order
    $C_\alpha(\mathbf{x}_{N+1}) =$
      $[l_{(\lfloor (N+1)(\alpha/2) \rfloor)}, u_{(\lceil (N+1)(1-\alpha/2) \rceil)}]$
```

Conformalizing Linearized Laplace We can show that our ACP-GN method can be interpreted as conformalizing Linearized Laplace (MacKay, 1992; Khan et al., 2019; Immer et al., 2021b) for regression. This relates to the result of Burnaev & Vovk (2014) who showed the CRR procedure can be viewed as conformalizing or “de-Bayesing” Bayesian Linear Regression (with Gaussian assumptions) (Burnaev & Vovk, 2014). They provide asymptotic results indicating that in well-specified settings, the conformal prediction intervals and Bayesian credible intervals closely align. This places our method within the wider context of *Conformal Bayes* (Melluish et al., 2001; Wasserman, 2011) for recalibrating Bayesian intervals in case of model misspecification.

The Laplace approximation constructs a Gaussian posterior approximation centered around the point estimate $\boldsymbol{\theta}_*$ and covariance given by the inverse of the Hessian (local curvature) of the empirical risk evaluated at $\boldsymbol{\theta}_*$. When the Hessian is approximated by the generalized Gauss-Newton matrix, we refer to the resulting Laplace approximation as the Laplace-GGN posterior, $q_*(\boldsymbol{\theta}) = N(\boldsymbol{\theta}|\boldsymbol{\theta}_*, \boldsymbol{\Sigma}_*)$ where $\boldsymbol{\Sigma}_* = \mathbf{H}_{\text{GN}}^{-1}$. This is often accompanied by linearizing the output of the neural network about $\boldsymbol{\theta}_*$ (Foong et al., 2019; Immer et al., 2021b):

$$f_i(\boldsymbol{\theta}) \approx f_i^{\text{lin}}(\boldsymbol{\theta}) = f_i(\boldsymbol{\theta}_*) + \nabla_\theta f_i(\boldsymbol{\theta}_*)^\top (\boldsymbol{\theta} - \boldsymbol{\theta}_*) \,. \tag{15}$$

The overall method is referred to as Linearized Laplace and retains the original NN point prediction as the mean of the posterior predictive. Analogous to the add-one-in estimate in Eq. (12), we can derive an approximation to the add-one-in posterior $\hat{q}_*^+(\boldsymbol{\theta})$ by perturbing the Laplace-GGN posterior (see App. E for derivation) whose mean is equal to $\hat{\boldsymbol{\theta}}_*^+(y)$:

$$\hat{q}_*^+(\boldsymbol{\theta}) = \mathbb{N}(\boldsymbol{\theta}|\hat{\boldsymbol{\theta}}_*^+(y), \hat{\boldsymbol{\Sigma}}_*^+) \quad \text{where } \hat{\boldsymbol{\Sigma}}_*^+ = \left(\mathbf{H}_{\text{GN}} + \boldsymbol{\phi}_{N+1}\boldsymbol{\phi}_{N+1}^\top\right)^{-1} \tag{16}$$

This is a simple extension of the leave-one-out results in (Nickl et al., 2023) to the add-one-in case. Using this perturbed posterior in combination with the linearized predictor in Eq. (15), we recover Eqs. (13) and (14). From the perspective of Linearized Laplace, the Gauss-Newton influence gives the exact AOI solution with respect to the linearized network and in particular its feature expansion given the Jacobian of the network. However, it is often the case in practice that $\boldsymbol{\theta}_*$ is not a minimum of the empirical risk (*i.e.* neural network not trained to convergence). Thus, $\boldsymbol{\theta}_*$ is not a minima of the linearized network's objective. One can correct this by solving for the following objective,

$$\tilde{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \left(\textstyle\sum_{i=1}^N \frac{1}{2}(\tilde{y}_i - \boldsymbol{\phi}_i^\top \boldsymbol{\theta})^2 + \frac{1}{2}\delta \left\|\boldsymbol{\theta}\right\|^2 \right) \tag{17}$$

where $\tilde{y}_i := \boldsymbol{\phi}_i^\top \boldsymbol{\theta}_* + e_i$ with residual $e_i = y_i - f_i(\boldsymbol{\theta}_*)$. This is a linear-Gaussian system and hence can be solved for in a single step. This process is often referred to as "refinement" in the Linearized Laplace literature and has been shown to improve predictions (Immer et al., 2021b). The exact AOI solution with respect to Eq. (17) is then given by,

$$\tilde{\boldsymbol{\theta}}_*^+(y) = \tilde{\boldsymbol{\theta}} + \frac{y - f_{N+1}^{\text{lin}}(\tilde{\boldsymbol{\theta}})}{1 + \hat{h}_{N+1}} \mathbf{H}_{\text{GN}}^{-1} \boldsymbol{\phi}_{N+1} \tag{18}$$

which reduces to Eq. (12) when $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta}_*$. We can derive analogous expressions to Eqs. (13) and (14) that differ only in the use of linearized versions of the neural network prediction and hold exactly.

Normalized Split-CP with Gauss-Newton Influence It turns out the tools we used to derive our approximate full-CP method are also effective to improve the adaptability and efficiency of split-CP. In vanilla split-CP, all prediction intervals have the same width because the model remains fixed. One way to alleviate this issue is to *normalize* the scores as $R_i = R_i/\sigma_i$, where σ_i estimates the difficulty in predicting the i^{th} data point correctly (Papadopoulos et al., 2008). As observed in (Papadopoulos, 2024), the normalized scores described in Sec. 2.4 are scaled by the predictive variance, which is closely related to how difficult y_i is to predict. This motivates our Gauss-Newton split-CP variant, with scores,

$$R_i = {|y_i - f_i(\boldsymbol{\theta}_*)|}/{\sqrt{1+\hat{h}_i}}, \tag{19}$$

where $\hat{h}_i = \boldsymbol{\phi}_i^\top \mathbf{H}_{\text{GN}}^{-1} \boldsymbol{\phi}_i$ is the marginal variance given by Linearized Laplace. This is similar to Eq. 4.10 in (Vovk et al., 2005) and is analogous to the studentized scores in the full-CP case.

Validity of ACP-GN Similar to Martinez et al. (2023), we cannot assure that our ACP-GN satisfies the coverage guarantee of full-CP. This is simply because the retraining step is locally approximated. However, bounds on the approximation error have previously been derived for Newton-step influence under fairly standard assumptions (Beirami et al., 2017; Koh et al., 2019). Whilst not shown here, we anticipate that such bounds can be extended to Gauss-Newton influence, as it is just a specialization of Newton-step influence for a certain choice of Hessian approximation. That being said in a majority of settings (datasets and target coverage levels) we empirically observe that the validity of ACP-GN does hold in practice. To address concerns about validity, in Sec. 4 we propose a variant "ACP-GN (split+refine)" that is guaranteed to provide correct coverage. This uses a train-calibration split like in split-CP along with the refinement procedure from Linearized Laplace ensuring that the residual nonconformity score expressions are exact with respect to the linearized neural network.

4 Experiments & Results

We compare our method, approximate full-CP via Gauss-Newton influence (ACP-GN), against Linearized Laplace (LA) (MacKay, 1992; Immer et al., 2021b), a recently popular Bayesian method for post-hoc uncertainty quantification, split conformal prediction (SCP) (Papadopoulos et al., 2002), conformalized residual fitting (CRF) (Papadopoulos et al., 2002) and conformalized quantile regression (CQR) (Romano et al., 2019). We also evaluate two further proposals, "ACP-GN (split + refine)" and "SCP-GN" which we proceed to describe along with the aforementioned methods:

- **LA**: The Laplace approximation with the Hessian approximated by the generalized Gauss-Newton matrix. The linearized predictive in Eq. (15) is used for inference. We use the implementation provided in the `Laplace` PyTorch library (Daxberger et al., 2021a) as well as to implement our ACP-GN method.
- **SCP**: Uses absolute residual nonconformity score in the procedure outlined in Sec. 2.1.
- **CRF**: Trains an additional network to predict the absolute residuals of the original network which is then used to normalize the absolute residual nonconformity score.
- **CQR**: Trains a quantile regression network using the pinball loss. The predicted lower and upper quantile functions are then used in the split conformal quantile regression algorithm.
- **ACP-GN**: Uses the *studentized* nonconformity score of Eq. (11) as we found this score performed the best.
- **SCP-GN**: Normalizes the absolute residual nonconformity score by the posterior predictive standard deviation of the LA method (trained only on the same split as SCP) as in Eq. (19).
- **ACP-GN** (split + refine): Uses a train-calibration split like in SCP, where we pretrain the model on the training set before running ACP-GN on the calibration set. More precisely, it solves the linearized network's objective in Eq. (17) but defined on the calibration set. It then uses the linearized network prediction in Eq. (15) in lieu of the original network to evaluate the CRR coefficients, again on the calibration set.

4.1 UCI Regression

We conduct experiments on popular benchmark datasets for regression taken from the UCI Machine Learning repository (Nottingham et al., 2024). These vary in size and we adapt the experimental setup accordingly, placing the datasets into 3 groups for easy referencing: *small* (boston, concrete, energy, wine, yacht); *medium* (kin8nm, power); *large* (bike, community, protein, facebook_1, facebook_2). A subset of these are shown in Table 1 and the remainder are discussed in App. I.1. For the small datasets, the reported metrics result from 10 repeats of a 10-fold cross-validation process. For the other datasets, we perform 20 different train-test splits. In all cases, 90% of the examples are used for training and 10% for testing. When a calibration set is needed, the training set is divided into two chunks of equal size. We show results for a desired miscoverage rate of $\alpha \in \{0.1, 0.05, 0.01\}$.

All methods use neural networks trained to convergence with the Adam optimizer (Kingma & Ba, 2015). Throughout, we use fully-connected layers with 50 units and GeLU activations. The small and medium datasets have a single hidden layer whereas the large datasets use 3 hidden layers. For these architectures, it is feasible to evaluate the Gauss-Newton matrix without any approximations. However, we expect this to be prohibitive for larger architectures—inversion of the Gauss-Newton matrix scales cubically in the number of parameters. For this reason in App. J.1, we repeat the experiments using two scalable approximations: Kronecker-factored approximate curvature (KFAC) (Martens & Grosse, 2015) and last-layer approximation (Daxberger et al., 2021b) (*i.e.* neural linear model approach (Ober & Rasmussen, 2019)). As described in Sec. 2.3, to construct the predictive intervals from the coefficients in ACP-GN we use the ridge regression confidence machine algorithm (Nouretdinov et al., 2001) on the small datasets, and the asymmetric version (Burnaev & Vovk, 2014) for the medium and large datasets. See App. I.1 for further details on the experimental setup.

To assess the efficiency (tightness) and well-calibratedness of our proposed methods for obtaining prediction intervals, we report their average prediction interval width and coverage against the baselines in Table 1. A method is reported as satisfying validity if its empirical coverage lies within the 1% and 99% quantiles of the exact marginal coverage distribution as given by the train/calibration

Table 1: Our approximate full-CP via Gauss-Newton influence (ACP-GN) almost always gives the tightest intervals in limited-data regimes whilst satisfying the target coverage (cf. `yacht`, `boston`, `energy`). On larger datasets, ACP-GN remains competitive on efficiency compared with other conformal methods but can sometimes miscover. As a remedy, we propose two variants inspired by ACP-GN that generally fix the miscoverage issue. Average prediction interval width and coverage for our proposed approaches (shaded gray) against baselines (non-shaded) for three different settings of the confidence level. The best average widths over well-calibrated approaches (indicated by ✓/✗) appear in bold. Reported metrics are accompanied by standard error from repeated runs.

		Avg. Width			Avg. Coverage		
		90%	95%	99%	90%	95%	99%
	LA	1.690±0.017	2.014±0.020	2.647±0.027	88.73±0.61 (✓)	90.78±0.59 (✗)	93.89±0.60 (✗)
	SCP	2.553±0.093	4.001±0.115	10.018±0.361	89.56±0.66 (✓)	94.07±0.39 (✓)	99.32±0.08 (✓)
`yacht`	**CRF**	2.526±0.092	3.947±0.115	9.674±0.294	89.53±0.64 (✓)	94.10±0.38 (✓)	99.29±0.10 (✓)
N=308	**CQR**	4.090±0.105	5.845±0.187	18.650±0.484	89.94±0.42 (✓)	94.42±0.32 (✓)	99.02±0.17 (✓)
I=6	**ACP-GN**	**1.594**±0.016	**2.385**±0.029	**6.915**±0.067	87.36±0.58 (✓)	92.56±0.68 (✓)	99.03±0.11 (✓)
	SCP-GN	2.270±0.086	3.349±0.098	**7.216**±0.254	89.85±0.51 (✓)	94.91±0.32 (✓)	99.19±0.15 (✓)
	ACP-GN (split + refine)	1.993±0.020	2.954±0.037	**7.307**±0.178	89.35±0.62 (✓)	94.90±0.51 (✓)	99.45±0.08 (✓)
	LA	9.398±0.046	**11.199**±0.055	14.718±0.072	91.24±0.31 (✓)	94.34±0.22 (✓)	97.53±0.11 (✗)
	SCP	10.635±0.123	14.509±0.171	36.272±1.847	89.56±0.42 (✓)	94.64±0.32 (✓)	99.11±0.13 (✓)
`boston`	**CRF**	11.932±0.605	16.073±0.862	40.690±3.333	90.01±0.33 (✓)	94.77±0.22 (✓)	99.30±0.08 (✓)
N=506	**CQR**	11.692±0.129	15.115±0.213	31.628±1.822	90.10±0.33 (✓)	95.12±0.24 (✓)	99.07±0.14 (✓)
I=13	**ACP-GN**	**9.182**±0.046	12.111±0.038	**20.512**±0.057	90.64±0.26 (✓)	95.49±0.16 (✓)	99.11±0.08 (✓)
	SCP-GN	10.301±0.089	13.418±0.151	24.714±0.865	89.52±0.50 (✓)	94.82±0.32 (✓)	99.05±0.12 (✓)
	ACP-GN (split + refine)	13.103±0.072	16.729±0.134	27.561±0.445	90.12±0.26 (✓)	95.41±0.20 (✓)	99.27±0.10 (✓)
	LA	1.502±0.006	1.790±0.007	2.353±0.009	88.96±0.35 (✓)	92.92±0.33 (✗)	96.95±0.23 (✗)
	SCP	1.942±0.032	2.486±0.046	3.772±0.093	89.44±0.28 (✓)	94.80±0.20 (✓)	99.18±0.08 (✓)
`energy`	**CRF**	1.923±0.031	2.454±0.046	3.728±0.092	89.39±0.28 (✓)	94.78±0.22 (✓)	99.14±0.08 (✓)
N=768	**CQR**	4.670±0.030	5.139±0.029	6.438±0.120	90.08±0.26 (✓)	95.24±0.21 (✓)	98.96±0.09 (✓)
I=8	**ACP-GN**	**1.462**±0.006	**1.884**±0.008	**3.076**±0.015	88.28±0.33 (✓)	93.69±0.33 (✓)	98.88±0.11 (✓)
	SCP-GN	1.911±0.029	2.449±0.044	3.609±0.071	89.69±0.29 (✓)	94.79±0.18 (✓)	99.21±0.09 (✓)
	ACP-GN (split + refine)	1.745±0.016	2.174±0.021	3.300±0.045	90.54±0.25 (✓)	94.96±0.22 (✓)	99.18±0.10 (✓)
	LA	100.451±2.394	119.694±2.853	157.305±3.749	89.82±0.39 (✓)	93.29±0.33 (✗)	96.83±0.16 (✗)
	SCP	131.138±0.812	180.477±1.244	324.756±4.635	90.33±0.21 (✓)	95.17±0.15 (✓)	99.00±0.07 (✓)
`bike`	**CRF**	127.836±0.894	174.362±1.376	311.580±5.077	90.39±0.19 (✓)	95.26±0.14 (✓)	99.01±0.07 (✓)
N=10,886	**CQR**	141.329±5.943	**167.682**±5.835	**244.863**±4.952	89.83±0.23 (✓)	94.80±0.14 (✓)	98.89±0.07 (✓)
I=18	**ACP-GN**	**98.813**±2.485	130.893±3.231	213.131±5.630	89.36±0.43 (✓)	94.41±0.27 (✗)	98.67±0.09 (✗)
	SCP-GN	122.245±1.073	**160.505**±1.761	254.409±3.767	90.34±0.24 (✓)	95.26±0.15 (✓)	99.02±0.08 (✓)
	ACP-GN (split + refine)	128.336±4.336	**170.782**±5.859	281.632±10.176	89.98±0.22 (✓)	94.94±0.16 (✓)	99.01±0.06 (✓)
	LA	9.385±0.022	11.183±0.027	14.697±0.035	85.43±0.18 (✗)	89.69±0.15 (✗)	94.81±0.10 (✗)
	SCP	13.041±0.088	17.161±0.098	26.181±0.119	89.78±0.08 (✓)	94.83±0.06 (✓)	98.94±0.04 (✓)
`protein`	**CRF**	12.645±0.127	16.931±0.146	26.973±0.202	89.86±0.09 (✓)	94.84±0.07 (✓)	98.93±0.04 (✓)
N=45,730	**CQR**	13.541±0.144	**14.798**±0.129	**18.239**±0.041	90.07±0.10 (✓)	95.07±0.09 (✓)	98.96±0.04 (✓)
I=9	**ACP-GN**	10.243±0.019	13.294±0.027	20.101±0.053	87.54±0.15 (✗)	93.04±0.11 (✗)	98.24±0.05 (✗)
	SCP-GN	**12.426**±0.085	16.102±0.096	24.032±0.138	89.78±0.10 (✓)	94.86±0.08 (✓)	98.94±0.03 (✓)
	ACP-GN (split + refine)	12.660±0.028	16.073±0.031	23.445±0.057	89.83±0.09 (✓)	94.90±0.09 (✓)	98.97±0.05 (✓)
	LA	66.088±2.760	78.749±3.289	103.493±4.322	97.47±0.12 (✗)	98.01±0.09 (✗)	98.65±0.06 (✗)
	SCP	16.387±0.208	35.387±0.462	152.706±1.591	89.97±0.07 (✓)	95.00±0.06 (✓)	99.06±0.03 (✓)
`facebook_2`	**CRF**	**15.088**±0.188	29.679±0.396	102.326±2.552	89.94±0.07 (✓)	94.98±0.06 (✓)	99.03±0.02 (✓)
N=81,311	**CQR**	17.605±0.645	**21.571**±0.960	**30.852**±1.303	90.16±0.28 (✓)	95.13±0.12 (✓)	99.01±0.03 (✓)
I=53	**ACP-GN**	18.396±0.546	40.088±1.091	166.792±5.632	90.47±0.11 (✗)	95.45±0.08 (✗)	99.35±0.04 (✗)
	SCP-GN	16.287±0.202	33.655±0.563	118.489±4.305	89.99±0.07 (✓)	94.98±0.06 (✓)	99.00±0.03 (✓)
	ACP-GN (split + refine)	21.469±0.906	42.184±0.788	152.460±2.499	90.14±0.08 (✓)	95.10±0.06 (✓)	99.13±0.02 (✗)

set size (depending on the method) (Angelopoulos & Bates, 2021; Vovk, 2012). In the case of limited data regimes (yacht, boston, energy), ACP-GN gives the tighest intervals, with the exception of boston at 95% target coverage where LA is the most efficient (being one of the few cases when LA does not miscover). On the larger datasets, ACP-GN remains competitive on efficiency with the exception of facebook_2 at the higher values of target coverage, but we find it miscovers. Our proposed variant, "ACP-GN (split+refine)", generally gives the correct coverage albeit incurring a trade-off in efficiency due to sample splitting. We also observe that our novel normalization strategy inspired by ACP-GN, SCP-GN, improves over CRF for most datasets and settings of the target coverage.

4.2 Bounding Box Localization

We consider single-object localization and in particular adapt the task from Phan et al. (2018), which predicts bounding boxes localizing the face of different breeds of cats and dogs in varying conditions. Conformal methods have recently been adapted for this task (De Grancey et al., 2022). We construct two-sided intervals similar to Timans et al. (2024) but without considering uncertainty in the classifier, following De Grancey et al. (2022). All images with ground-truth bounding box annotations in the Oxford-IIIT Pet dataset (Parkhi et al., 2012) are extracted resulting in 3 686 images overall. The experiment is repeated with 20 different train-test splits where 20% of the data is used for testing. When needed, 25% of the training set is reserved as calibration data. The VGG-19 ar-

Table 2: On a bounding box regression task using a deep convolutional neural network, our split and refine variant of ACP-GN results in the most efficient confidence regions whilst achieving comparable coverage to the split-CP baselines. We target coverage rates of $\{85\%, 90\%, 95\%\}$ and reported metrics are accompanied by standard error from repeated runs.

	Avg. Volume ($\times 10^{-2}$)			Avg. Coverage		
	85%	90%	95%	85%	90%	95%
LA	$0.710_{\pm 0.009}$	$0.945_{\pm 0.013}$	$1.405_{\pm 0.019}$	$92.11_{\pm 0.18}$ (✗)	$94.49_{\pm 0.18}$ (✗)	$96.65_{\pm 0.16}$ (✗)
SCP	$0.809_{\pm 0.025}$	$1.166_{\pm 0.035}$	$2.261_{\pm 0.106}$	$87.80_{\pm 0.42}$ (✓)	$91.82_{\pm 0.41}$ (✓)	$95.85_{\pm 0.32}$ (✓)
CRF	$0.713_{\pm 0.021}$	$1.098_{\pm 0.044}$	$2.130_{\pm 0.087}$	$87.47_{\pm 0.44}$ (✓)	$91.61_{\pm 0.34}$ (✓)	$95.85_{\pm 0.23}$ (✓)
CQR	$0.947_{\pm 0.028}$	$1.451_{\pm 0.046}$	$3.680_{\pm 0.113}$	$87.45_{\pm 0.51}$ (✓)	$91.29_{\pm 0.38}$ (✓)	$96.06_{\pm 0.26}$ (✓)
ACP-GN	$0.384_{\pm 0.004}$	$0.605_{\pm 0.007}$	$1.225_{\pm 0.017}$	$90.97_{\pm 0.20}$ (✗)	$94.09_{\pm 0.20}$ (✗)	$97.41_{\pm 0.16}$ (✗)
SCP-GN	$0.768_{\pm 0.020}$	$1.071_{\pm 0.030}$	$1.854_{\pm 0.066}$	$87.17_{\pm 0.45}$ (✓)	$91.43_{\pm 0.40}$ (✓)	$95.58_{\pm 0.28}$ (✓)
ACP-GN (split + refine)	$\mathbf{0.311}_{\pm 0.006}$	$\mathbf{0.472}_{\pm 0.014}$	$\mathbf{1.031}_{\pm 0.046}$	$87.34_{\pm 0.41}$ (✓)	$91.31_{\pm 0.35}$ (✓)	$96.12_{\pm 0.27}$ (✓)

chitecture is used as the object detection backbone but with the original output layer removed. The network is trained jointly with two heads, a regression head predicting 2D bounding box coordinates (4 outputs) and a binary classification head. We only consider the regression head for constructing predictive intervals. Without a calibration split, the model gets 99.6% classification accuracy and 20.1% localization error with 0.5 IoU (Intersection over Union) threshold. With a calibration split, the model achieves 99.5% classification accuracy and 27.5% localization error.

We use the last-layer approximation to the Gauss-Newton matrix throughout for computational reasons. The asymmetric implementation of CRR is used due to its efficiency and we target miscoverage rates $\alpha \in \{0.15, 0.1, 0.05\}$. See App. I.2 for further experimental details. Since bounding box localization is a type of multi-output regression, we obtain confidence *regions* given by hyperrectangles except for LA that results in a hyperellipsoid. All conformal prediction methods are run for each output dimension independently and we evaluate the confidence region volumes by taking the product over interval widths per output dimension. We apply a multiple testing correction, the Bonferroni correction, to mitigate the miscoverage that arises when conformalizing the outputs separately. However, we find that all methods still consistently overcover suggesting further improvements are possible.

In Table 2, we observe that ACP-GN gives the tightest intervals as compared with the baseline conformal methods despite having a greater empirical coverage than those methods. Surprisingly our split and refine variant of ACP-GN gives even tighter intervals whilst matching the coverage of the conformal baselines. We observed a similar effect in our ablation of the previous UCI experiments with the last-layer approximation in App. J.1.

5 Conclusion

In this work, we show how to efficiently construct prediction intervals with full conformal prediction (CP) for neural networks in regression tasks. While full-CP requires retraining the model from scratch on all of the training data and for each postulated label for the test point, we show one can efficiently approximate the full-CP predictive intervals with a specialization of Newton-step influence (Pregibon, 1981; Beirami et al., 2017) and the ridge regression confidence machine of Nouretdinov et al. (2001) without retraining the model. In doing so, we also avoid relying on a grid over all possible labels for the test point, which incurs an undesirable accuracy/precision trade-off due to the uncountable set of real numbers. We demonstrate how this approach corresponds to exact full-CP on a linearized version of the neural network and further show how it corresponds to "conformalizing" the Linearized Laplace method (Khan et al., 2019; Immer et al., 2021b), a popular Bayesian approach for post-hoc uncertainty estimation in deep learning. Finally, through the lens of normalized nonconformity scores, we recover the leave-one-out variety of full-CP and easily extend to studentized scores that we find performs the best empirically and was left as future work in Martinez et al. (2023). This also leads us to propose a novel adaptive split-CP method similar to conformalized residual fitting (Papadopoulos et al., 2002) but without the need to train an additional network. Empirically, we see that our approximate full-CP methods typically provide tighter prediction intervals in limited data regimes across the well-calibrated approaches. For future work, we would like to extend our method to other real-world tasks such as pose estimation and tracking, and develop further the theoretical analysis.

REFERENCES

Ahmed Alaa and Mihaela Van Der Schaar. Discriminative Jackknife: Quantifying Uncertainty in Deep Learning via Higher-Order Influence Functions. In *International Conference on Machine Learning*, 2020.

David M Allen. The Relationship between Variable Selection and Data Agumentation and a Method for Prediction. *Technometrics*, 1974.

Anastasios N Angelopoulos and Stephen Bates. A Gentle Introduction to Conformal Prediction and Distribution-Free Uncertainty Quantification. *ArXiv e-prints*, 2021.

Javier Antorán, David Janz, James Urquhart Allingham, Erik Daxberger, Riccardo Barbano, Eric Nalisnick, and José Miguel Hernández-Lobato. Adapting the Linearised Laplace Model Evidence for Modern Deep Learning. In *International Conference on Machine Learning*, 2022.

Juhan Bae, Nathan Ng, Alston Lo, Marzyeh Ghassemi, and Roger B Grosse. If Influence Functions are the Answer, Then What is the Question? In *Advances in Neural Information Processing Systems*, 2022.

Rina Foygel Barber, Emmanuel J Candès, Aaditya Ramdas, and Ryan J Tibshirani. Predictive inference with the jackknife+. *The Annals of Statistics*, 2021.

Samyadeep Basu, Xuchen You, and Soheil Feizi. On Second-Order Group Influence Functions for Black-Box Predictions. In *International Conference on Machine Learning*, 2020.

Ahmad Beirami, Meisam Razaviyayn, Shahin Shahrampour, and Vahid Tarokh. On Optimal Generalizability in Parametric Learning. In *Advances in Neural Information Processing Systems*, 2017.

Evgeny Burnaev and Vladimir Vovk. Efficiency of conformalized ridge regression. In *Conference on Learning Theory*, 2014.

Samprit Chatterjee and Ali S Hadi. Influential Observations, High Leverage Points, and Outliers in Linear Regression. *Statistical Science*, 1986.

Wenyu Chen, Kelli-Jean Chun, and Rina Foygel Barber. Discretized conformal prediction for efficient distribution-free inference. *Stat*, 2018.

R Dennis Cook. Detection of Influential Observation in Linear Regression. *Technometrics*, 1977.

R Dennis Cook and Sanford Weisberg. Characterizations of an Empirical Influence Function for Detecting Influential Cases in Regression. *Technometrics*, 1980.

A Philip Dawid. The Well-Calibrated Bayesian. *Journal of the American Statistical Association*, 1982.

Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace Redux-Effortless Bayesian Deep Learning. In *Advances in Neural Information Processing Systems*, 2021a.

Erik Daxberger, Eric Nalisnick, James U Allingham, Javier Antorán, and José Miguel Hernández-Lobato. Bayesian Deep Learning via Subnetwork Inference. In *International Conference on Machine Learning*, 2021b.

Florence De Grancey, Jean-Luc Adam, Lucian Alecu, Sébastien Gerchinovitz, Franck Mamalet, and David Vigouroux. Object Detection with Probabilistic Guarantees: A Conformal Prediction Approach. In *International Conference on Computer Safety, Reliability, and Security*, 2022.

Bradley Efron and Robert Tibshirani. Bootstrap Methods for Standard Errors, Confidence Intervals, and Other Measures of Statistical Accuracy. *Statistical Science*, 1986.

Edwin Fong and Chris C Holmes. Conformal Bayesian Computation. *Advances in Neural Information Processing Systems*, 2021.

Matteo Fontana, Gianluca Zeni, and Simone Vantini. Conformal Prediction: a Unified Review of Theory and New Challenges. *Bernoulli*, 2023.

Andrew YK Foong, Yingzhen Li, José Miguel Hernández-Lobato, and Richard E Turner. 'In-Between' Uncertainty in Bayesian Neural Networks. In *ICML Workshop on Uncertainty and Robustness in Deep Learning*, 2019.

D A S Fraser. Is Bayes Posterior just Quick and Dirty Confidence? *Statistical Science*, 2011.

Ryan Giordano, Michael I Jordan, and Tamara Broderick. A Higher-Order Swiss Army Infinitesimal Jackknife. *ArXiv e-Prints*, 2019a.

Ryan Giordano, William Stephenson, Runjing Liu, Michael I Jordan, and Tamara Broderick. A Swiss Army Infinitesimal Jackknife. In *International Conference on Artificial Intelligence and Statistics*, 2019b.

R Girshick. Fast R-CNN. *ArXiv e-Prints*, 2015.

Peter Grünwald and Thijs van Ommen. Inconsistency of Bayesian Inference for Misspecified Linear Models, and a Proposal for Repairing It. *Bayesian Analysis*, 2017.

Etash Kumar Guha, Shlok Natarajan, Thomas Möllenhoff, Mohammad Emtiyaz Khan, and Eugene Ndiaye. Conformal Prediction via Regression-as-Classification. In *International Conference on Learning Representations*, 2024.

Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified Data Removal from Machine Learning Models. In *International Conference on Machine Learning*, 2020.

Frank R Hampel. The Influence Curve and its Role in Robust Estimation. *Journal of the American Statistical Association*, 1974.

Lars Kai Hansen and Jan Larsen. Linear unlearning for cross-validation. *Advances in Computational Mathematics*, 1996.

Alexander Immer, Matthias Bauer, Vincent Fortuin, Gunnar Rätsch, and Mohammad Emtiyaz Khan. Scalable Marginal Likelihood Estimation for Model Selection in Deep Learning. In *International Conference on Machine Learning*, 2021a.

Alexander Immer, Maciej Korzepa, and Matthias Bauer. Improving Predictions of Bayesian Neural Nets via Local Linearization. In *International Conference on Artificial Intelligence and Statistics*, 2021b.

Tommi Jaakkola and David Haussler. Exploiting Generative Models in Discriminative Classifiers. In *Advances in Neural Information Processing Systems*, 1998.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural Tangent Kernel: Convergence and Generalization in Neural Networks. In *Advances in Neural Information Processing Systems*, 2018.

Louis A Jaeckel. The Infinitesimal Jackknife. Technical report, Bell Lab., 1972.

Yuko Kato, David MJ Tax, and Marco Loog. A Review of Nonconformity Measures for Conformal Prediction in Regression. *Conformal and Probabilistic Prediction with Applications*, 2023.

Mohammad Emtiyaz Khan and Håvard Rue. The Bayesian Learning Rule. *Journal of Machine Learning Research*, 2023.

Mohammad Emtiyaz Khan, Alexander Immer, Ehsan Abedi, and Maciej Korzepa. Approximate Inference Turns Deep Networks into Gaussian Processes. In *Advances in Neural Information Processing Systems*, 2019.

Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2015.

Pang Wei Koh and Percy Liang. Understanding Black-Box Predictions via Influence Functions. In *International Conference on Machine Learning*, 2017.

Pang Wei Koh, Kai-Siang Ang, Hubert Teo, and Percy S Liang. On the Accuracy of Influence Functions for Measuring Group Effects. In *Advances in Neural Information Processing Systems*, 2019.

Roy T St Laurent and R Dennis Cook. Leverage and Superleverage in Nonlinear Regression. *Journal of the American Statistical Association*, 1992.

S le Cessie and J C van Houwelingen. Ridge Estimators in Logistic Regression. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 1992.

Jing Lei. Fast Exact Conformalization of Lasso using Piecewise Linear Homotopy. *Biometrika*, 2019.

Jing Lei, Max G'Sell, Alessandro Rinaldo, Ryan J Tibshirani, and Larry Wasserman. Distribution-Free Predictive Inference for Regression. *Journal of the American Statistical Association*, 2018.

David J C MacKay. A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation*, 1992.

Wesley Maddox, Shuai Tang, Pablo Moreno, Andrew Gordon Wilson, and Andreas Damianou. Fast Adaptation with Linearized Neural Networks. In *International Conference on Artificial Intelligence and Statistics*, 2021.

James Martens. Deep learning via Hessian-free optimization. In *International Conference on Machine Learning*, 2010.

James Martens and Roger Grosse. Optimizing Neural Networks with Kronecker-factored Approximate Curvature. In *International Conference on Machine Learning*, 2015.

Javier Abad Martinez, Umang Bhatt, Adrian Weller, and Giovanni Cherubin. Approximating Full Conformal Prediction at Scale via Influence Functions. In *AAAI Conference on Artificial Intelligence*, 2023.

Thomas Melluish, Craig Saunders, Ilia Nouretdinov, and Volodya Vovk. Comparing the Bayes and Typicalness Frameworks. In *European Conference on Machine Learning*, 2001.

Gaétan Monari and Gérard Dreyfus. Withdrawing an example from the training set: An analytic estimation of its effect on a non-linear parameterised model. *Neurocomputing*, 2000.

Fangzhou Mu, Yingyu Liang, and Yin Li. Gradients as Features for Deep Representation Learning. In *International Conference on Learning Representations*, 2020.

Eugene Ndiaye. Stable Conformal Prediction Sets. In *International Conference on Machine Learning*, 2022.

Eugene Ndiaye and Ichiro Takeuchi. Computing Full Conformal Prediction Set with Approximate Homotopy. In *Advances in Neural Information Processing Systems*, 2019.

Peter Nickl, Lu Xu, Dharmesh Tailor, Thomas Möllenhoff, and Mohammad Emtiyaz Khan. The Memory-Perturbation Equation: Understanding Model's Sensitivity to Data. In *Advances in Neural Information Processing Systems*, 2023.

Kolby Nottingham, Rachel Longjohn, and Markelle Kelly. UCI Machine Learning Repository, 2024. URL `https://archive.ics.uci.edu/datasets`. Accessed: September, 2024.

Ilia Nouretdinov, Thomas Melluish, and Volodya Vovk. Ridge Regression Confidence Machine. In *International Conference on Machine Learning*, 2001.

Roman Novak, Jascha Sohl-Dickstein, and Samuel S Schoenholz. Fast Finite Width Neural Tangent Kernel. In *International Conference on Machine Learning*, 2022.

Sebastian W Ober and Carl E Rasmussen. Benchmarking the Neural Linear Model for Regression. In *Second Symposium on Advances in Approximate Bayesian Inference*, 2019.

Harris Papadopoulos. Guaranteed Coverage Prediction Intervals with Gaussian Process Regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Harris Papadopoulos and Haris Haralambous. Reliable Prediction Intervals with Regression Neural Networks. *Neural Networks*, 2011.

Harris Papadopoulos, Kostas Proedrou, Volodya Vovk, and Alex Gammerman. Inductive Confidence Machines for Regression. In *European Conference on Machine Learning*, 2002.

Harris Papadopoulos, Alex Gammerman, and Volodya Vovk. Normalized nonconformity measures for regression conformal prediction. In *International Conference on Artificial Intelligence and Applications*, 2008.

Harris Papadopoulos, Vladimir Vovk, and Alex Gammerman. Regression Conformal Prediction with Nearest Neighbours. *Journal of Artificial Intelligence Research*, 2011.

Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and Dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

Buu Phan, Rick Salay, Krzysztof Czarnecki, Vahdat Abdelzad, Taylor Denouden, and Sachin Vernekar. Calibrating Uncertainties in Object Localization Task. *ArXiv e-Prints*, 2018.

Daryl Pregibon. Logistic Regression Diagnostics. *The Annals of Statistics*, 1981.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating Training Data Influence by Tracing Gradient Descent. In *Advances in Neural Information Processing Systems*, 2020.

Kamiar Rahnama Rad and Arian Maleki. A scalable estimate of the out-of-sample prediction error via approximate leave-one-out cross-validation. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 2020.

Hippolyt Ritter, Aleksandar Botev, and David Barber. A Scalable Laplace Approximation for Neural Networks. In *International Conference on Learning Representations*, 2018.

Yaniv Romano, Evan Patterson, and Emmanuel Candes. Conformalized Quantile Regression. In *Advances in Neural Information Processing Systems*, 2019.

Peter Schulam and Suchi Saria. Can You Trust This Prediction? Auditing Pointwise Reliability After Learning. In *International Conference on Artificial Intelligence and Statistics*, 2019.

Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. Remember What You Want to Forget: Algorithms for Machine Unlearning. In *Advances in Neural Information Processing Systems*, 2021.

Glenn Shafer and Vladimir Vovk. A Tutorial on Conformal Prediction. *Journal of Machine Learning Research*, 2008.

Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*, 2015.

Stefano Teso, Andrea Bontempelli, Fausto Giunchiglia, and Andrea Passerini. Interactive Label Cleaning with Example-based Explanations. In *Advances in Neural Information Processing Systems*, 2021.

Alexander Timans, Christoph-Nikolas Straehle, Kaspar Sakmann, and Eric Nalisnick. Adaptive Bounding Box Uncertainties via Two-Step Conformal Prediction. In *European Conference on Computer Vision*, 2024.

Vladimir Vovk. Conditional Validity of Inductive Conformal Predictors. In *Asian Conference on Machine Learning*, 2012.

Vladimir Vovk. Cross-conformal predictors. *Annals of Mathematics and Artificial Intelligence*, 2015.

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer, 2005.

Larry Wasserman. Frasian Inference. *Statistical Science*, 2011.

Bo-Cheng Wei, Yue-Qing Hu, and Wing-Kam Fung. Generalized Leverage and Its Applications. *Scandinavian Journal of Statistics*, 1998.

A Derivation of CRR coefficients

The ridge solution on $\mathcal{D}_N$ is given by $\boldsymbol{\theta}_* = \mathbf{H}_*^{-1}\mathbf{X}^\top\mathbf{y}$ where $\mathbf{X}$ is the $(N \times I)$ feature matrix with $\mathbf{x}_i^\top$ as rows and $\mathbf{y}$ is the N-dim vector of targets. We can express the ridge solution on $\mathcal{D}_{N+1}(y)$, referred to as the add-one-in (AOI) solution, as a deviation from $\boldsymbol{\theta}_*$:

$$\boldsymbol{\theta}_*^+(y) = (\mathbf{H}_* + \mathbf{x}_{N+1}\mathbf{x}_{N+1}^\top)^{-1}\left(\mathbf{X}^\top\mathbf{y} + \mathbf{x}_{N+1}y\right) \tag{20}$$

▷ use Sherman-Morrison formula

$$= \left(\mathbf{H}_*^{-1} - \frac{\mathbf{H}_*^{-1}\mathbf{x}_{N+1}\mathbf{x}_{N+1}^\top\mathbf{H}_*^{-1}}{1+\mathbf{x}_{N+1}^\top\mathbf{H}_*^{-1}\mathbf{x}_{N+1}}\right)\left(\mathbf{X}^\top\mathbf{y} + \mathbf{x}_{N+1}y\right) \tag{21}$$

$$= \mathbf{H}_*^{-1}\mathbf{X}^\top\mathbf{y} + \mathbf{H}_*^{-1}\mathbf{x}_{N+1}\left(y - \frac{\mathbf{x}_{N+1}^\top\mathbf{H}_*^{-1}\mathbf{X}^\top\mathbf{y}}{1+\mathbf{x}_{N+1}^\top\mathbf{H}_*^{-1}\mathbf{x}_{N+1}} - \frac{y\mathbf{x}_{N+1}^\top\mathbf{H}_*^{-1}\mathbf{x}_{N+1}}{1+\mathbf{x}_{N+1}^\top\mathbf{H}_*^{-1}\mathbf{x}_{N+1}}\right) \tag{22}$$

▷ substitute $\boldsymbol{\theta}_* = \mathbf{H}_*^{-1}\mathbf{X}^\top\mathbf{y}$ and $h_{N+1} = \mathbf{x}_{N+1}^\top\mathbf{H}_*^{-1}\mathbf{x}_{N+1}$

$$= \boldsymbol{\theta}_* + \mathbf{H}_*^{-1}\mathbf{x}_{N+1}\left(y - \frac{\mathbf{x}_{N+1}^\top\boldsymbol{\theta}_*}{1+h_{N+1}} - \frac{yh_{N+1}}{1+h_{N+1}}\right) \tag{23}$$

$$= \boldsymbol{\theta}_* + \mathbf{H}_*^{-1}\mathbf{x}_{N+1}\left(\frac{y - \mathbf{x}_{N+1}^\top\boldsymbol{\theta}_*}{1+h_{N+1}}\right) \tag{24}$$

Using this, it is easy to show the residuals can be expressed in terms of the postulated label y:

$$y_i - \mathbf{x}_i^\top\boldsymbol{\theta}_*^+(y) = y_i - \mathbf{x}_i^\top\boldsymbol{\theta}_* - \mathbf{x}_i^\top\mathbf{H}_*^{-1}\mathbf{x}_{N+1}\left(\frac{y - \mathbf{x}_{N+1}^\top\boldsymbol{\theta}_*}{1+h_{N+1}}\right) \tag{25}$$

▷ substitute $h_{i,N+1} = \mathbf{x}_i^\top\mathbf{H}_*^{-1}\mathbf{x}_{N+1}$

$$= y_i - \mathbf{x}_i^T\boldsymbol{\theta}_* + \frac{h_{i,N+1}}{1+h_{N+1}}\mathbf{x}_{N+1}^\top\boldsymbol{\theta}_* - \frac{h_{i,N+1}}{1+h_{N+1}}y \tag{26}$$

$$y - \mathbf{x}_{N+1}^\top\boldsymbol{\theta}_*^+(y) = y - \mathbf{x}_{N+1}^\top\boldsymbol{\theta}_* - \mathbf{x}_{N+1}^\top\mathbf{H}_*^{-1}\mathbf{x}_{N+1}\left(\frac{y - \mathbf{x}_{N+1}^\top\boldsymbol{\theta}_*}{1+h_{N+1}}\right) \tag{27}$$

▷ substitute $h_{N+1} = \mathbf{x}_{N+1}^\top\mathbf{H}_*^{-1}\mathbf{x}_{N+1}$

$$= -\frac{1}{1+h_{N+1}}\mathbf{x}_{N+1}^\top\boldsymbol{\theta}_* + \frac{1}{1+h_{N+1}}y \tag{28}$$

B Derivation of Gauss-Newton influence

The first two steps are similar to the usual derivation of Newton-step influence (Beirami et al., 2017) except that a single Newton step is performed on the *augmented* problem defined in Eq. (4) starting from $\boldsymbol{\theta}_*$. For brevity, we define $L(\boldsymbol{\theta}) = \sum_{i=1}^N \ell(y_i, f_i(\boldsymbol{\theta})) + \frac{1}{2}\delta\left\|\boldsymbol{\theta}\right\|^2$ as the empirical risk on $\mathcal{D}_N$.

$$\boldsymbol{\theta}_*^+(y) \approx \boldsymbol{\theta}_* - \left(\nabla_\theta^2 L(\boldsymbol{\theta}_*) + \nabla_\theta^2\ell(y, f_{N+1}(\boldsymbol{\theta}_*))\right)^{-1}\left(\nabla_\theta L(\boldsymbol{\theta}_*) + \nabla_{\boldsymbol{\theta}}\ell(y, f_{N+1}(\boldsymbol{\theta}_*))\right) \tag{29}$$

▷ substitute first-order stationarity condition $\nabla_\theta L(\boldsymbol{\theta}_*) = 0$

$$= \boldsymbol{\theta}_* - \left(\nabla_\theta^2 L(\boldsymbol{\theta}_*) + \nabla_\theta^2\ell(y, f_{N+1}(\boldsymbol{\theta}_*))\right)^{-1}\nabla_{\boldsymbol{\theta}}\ell(y, f_{N+1}(\boldsymbol{\theta}_*)) \tag{30}$$

▷ use Gauss-Newton approximation to Hessian

$$\approx \boldsymbol{\theta}_* - \left(\mathbf{H}_{\text{GN}} + \boldsymbol{\phi}_{N+1}\boldsymbol{\phi}_{N+1}^\top\right)^{-1}\nabla_{\boldsymbol{\theta}}\ell(y, f_{N+1}(\boldsymbol{\theta}_*)) \tag{31}$$

▷ expand the loss gradient on the $(N+1)^{\text{th}}$ example by chain rule

$$= \boldsymbol{\theta}_* + \left(\mathbf{H}_{\text{GN}} + \boldsymbol{\phi}_{N+1}\boldsymbol{\phi}_{N+1}^\top\right)^{-1}\boldsymbol{\phi}_{N+1}\hat{e}_{N+1}(y) \tag{32}$$

▷ use Sherman-Morrison formula

$$= \boldsymbol{\theta}_* + \frac{\hat{e}_{N+1}(y)}{1+\hat{h}_{N+1}}\mathbf{H}_{\text{GN}}^{-1}\boldsymbol{\phi}_{N+1} \tag{33}$$

C Derivation of ACP-GN coefficients

We present the derivation for expressing the residual in the neural network regression setting as a linear function of the postulated label y,

$$y_i - f_i(\boldsymbol{\theta}_*^+(y)) \tag{34}$$

$\triangleright$ linearize neural network about $\boldsymbol{\theta}_*$

$$\approx y_i - \left[f_i(\boldsymbol{\theta}_*) + \nabla_\theta f_i(\boldsymbol{\theta}_*)\left(\boldsymbol{\theta}_*^+(y) - \boldsymbol{\theta}_*\right)\right] \tag{35}$$

$\triangleright$ approximate $\boldsymbol{\theta}_*^+(y) \approx \hat{\boldsymbol{\theta}}_*^+(y)$ and substitute Gauss-Newton influence

$$= y_i - f_i(\boldsymbol{\theta}_*) - \boldsymbol{\phi}_i^\top \left(\frac{\hat{e}_{N+1}(y)}{1+\hat{h}_{N+1}}\mathbf{H}_{\text{GN}}^{-1}\boldsymbol{\phi}_{N+1}\right) \tag{36}$$

$\triangleright$ definition of residual $\hat{e}_{N+1}(y)$ to reveal postulated label y and substitute $\hat{h}_{i,N+1} = \boldsymbol{\phi}_i^\top \mathbf{H}_{\text{GN}}^{-1}\boldsymbol{\phi}_{N+1}$

$$= \underbrace{y_i - f_i(\boldsymbol{\theta}_*) + \frac{\hat{h}_{i,N+1}}{1+\hat{h}_{N+1}} f_{N+1}(\boldsymbol{\theta}_*)}_{a_i} \underbrace{- \frac{\hat{h}_{i,N+1}}{1+\hat{h}_{N+1}}}_{b_i} y \tag{37}$$

The residual of the postulated point proceds in an almost identical manner.

D Extension of ACP-GN for multi-output regression

In the case of multi-output regression with vector-valued targets $\mathbf{y}_i \in \mathbb{R}^O$ and outputs of a DNN $\mathbf{f}_i(\boldsymbol{\theta}) \in \mathbb{R}^O$, the coefficients needed in the CRR procedure $\{\mathbf{a}_i, \mathbf{B}_i\}$ now correspond to a O-dim vector and a $O \times O$ matrix respectively. Since CRR was only proposed for single-output regression, we also need to adapt the procedure. Considering the asymmetric version of CRR (Burnaev & Vovk, 2014), we simply need to solve for the O-dim changepoints which are given by $(\mathbf{B}_{N+1} - \mathbf{B}_i)^{-1}(\mathbf{a}_i - \mathbf{a}_{N+1})$ as long as $\mathbf{B}_{N+1} - \mathbf{B}_i$ is positive-definite (generalizing the positivity constraint in the single-output case). Then we propose to sort the set of changepoints followed by taking the quantile component-wise analogous to the existing algorithm. Now we proceed to derive $\{\mathbf{a}_i, \mathbf{B}_i\}$. Firstly the multi-output analogue of the AOI estimator in Eq. (12) is given by,

$$\hat{\boldsymbol{\theta}}_*^+(\mathbf{y}) = \boldsymbol{\theta}_* + \mathbf{H}_{\text{GN}}^{-1}\boldsymbol{\Phi}_{N+1}^\top\left(\mathbf{I} + \hat{\mathbf{V}}_{N+1}\right)^{-1}\hat{\mathbf{e}}_{N+1}(\mathbf{y}) \tag{38}$$

where $\boldsymbol{\Phi}_i := \nabla_{\boldsymbol{\theta}}\mathbf{f}_i(\boldsymbol{\theta}_*) \in \mathbb{R}^{O\times D}$, $\mathbf{H}_{\text{GN}} = \sum_{i=1}^N \boldsymbol{\Phi}_i^\top\boldsymbol{\Phi}_i + \delta\mathbf{I}$, $\hat{\mathbf{e}}_{N+1}(\mathbf{y}) = \mathbf{y} - \mathbf{f}_{N+1}(\boldsymbol{\theta}_*)$ and $\hat{\mathbf{V}}_{N+1} = \boldsymbol{\Phi}_{N+1}\mathbf{H}_{\text{GN}}^{-1}\boldsymbol{\Phi}_{N+1}^\top$. There is a change in notation in the last expression of the multi-output leverage score to avoid confusion with the Hessian. Using this we can derive analogous expressions to Eqs. (13) and (14):

$$\mathbf{y}_i - \mathbf{f}_i(\boldsymbol{\theta}_*^+(\mathbf{y})) \approx \mathbf{y}_i - \left[\mathbf{f}_i(\boldsymbol{\theta}_*) + \nabla_{\boldsymbol{\theta}}\mathbf{f}_i(\boldsymbol{\theta}_*)\left(\hat{\boldsymbol{\theta}}_*^+(\mathbf{y}) - \boldsymbol{\theta}_*\right)\right] \tag{39}$$

$$= \mathbf{y}_i - \mathbf{f}_i(\boldsymbol{\theta}_*) - \boldsymbol{\Phi}_i\mathbf{H}_{\text{GN}}^{-1}\boldsymbol{\Phi}_{N+1}^\top\left(\mathbf{I} + \hat{\mathbf{V}}_{N+1}\right)^{-1}\hat{\mathbf{e}}_{N+1}(\mathbf{y}) \tag{40}$$

$$= \underbrace{\mathbf{y}_i - \mathbf{f}_i(\boldsymbol{\theta}_*) + \hat{\mathbf{V}}_{i,N+1}\left(\mathbf{I} + \hat{\mathbf{V}}_{N+1}\right)^{-1}\mathbf{f}_{N+1}(\boldsymbol{\theta}_*)}_{\mathbf{a}_i} \tag{41}$$

$$\underbrace{-\hat{\mathbf{V}}_{i,N+1}\left(\mathbf{I} + \hat{\mathbf{V}}_{N+1}\right)^{-1}}_{\mathbf{B}_i}\mathbf{y}$$

and

$$\mathbf{y} - \mathbf{f}_{N+1}(\boldsymbol{\theta}_*^+(\mathbf{y})) \approx \underbrace{-\left(\mathbf{I} + \hat{\mathbf{V}}_{N+1}\right)^{-1}\mathbf{f}_{N+1}(\boldsymbol{\theta}_*)}_{\mathbf{a}_{N+1}} + \underbrace{\left(\mathbf{I} + \hat{\mathbf{V}}_{N+1}\right)^{-1}}_{\mathbf{B}_{N+1}}\mathbf{y} \tag{42}$$

where $\hat{\mathbf{V}}_{i,N+1} = \boldsymbol{\Phi}_i\mathbf{H}_{\text{GN}}^{-1}\boldsymbol{\Phi}_{N+1}^\top$. The normalized nonconformity scores can also be extended to the multi-output setting. In the case of deleted-CRR, we have:

$$\mathbf{a}_i \leftarrow \left(\mathbf{I} - \bar{\mathbf{V}}_i\right)^{-1}\mathbf{a}_i, \quad \mathbf{B}_i \leftarrow \left(\mathbf{I} - \bar{\mathbf{V}}_i\right)^{-1}\mathbf{B}_i \qquad \forall i = 1, \ldots, N+1 \tag{43}$$

where,

$$\bar{\mathbf{V}}_i = \hat{\mathbf{V}}_i - \hat{\mathbf{V}}_{i,N+1}\big(\mathbf{I} + \hat{\mathbf{V}}_{N+1}\big)^{-1}\hat{\mathbf{V}}_{i,N+1}^\top \quad \forall i = 1, \dots, N \tag{44}$$

$$\bar{\mathbf{V}}_{N+1} = \hat{\mathbf{V}}_{N+1}\big(\mathbf{I} + \hat{\mathbf{V}}_{N+1}\big)^{-1} \tag{45}$$

and we introduced $\hat{\mathbf{V}}_i = \boldsymbol{\Phi}_i \mathbf{H}_{\text{GN}}^{-1}\boldsymbol{\Phi}_i^\top$.

E Derivation of approximate add-one-in posterior with Laplace-GGN

Khan et al. (2019) show that the Laplace-GGN posterior can be equivalently stated as exact inference in the following linear regression model (see their Theorem 1):

$$q_*(\boldsymbol{\theta}) \propto \prod_{i=1}^{N} e^{-\frac{1}{2}\left(\tilde{y}_i - \boldsymbol{\phi}_i^\top \boldsymbol{\theta}\right)^2} p(\boldsymbol{\theta}) \tag{46}$$

where $p(\boldsymbol{\theta}) \propto \exp(\frac{1}{2}\delta \|\boldsymbol{\theta}\|^2)$, $\boldsymbol{\phi}_i := \nabla_\theta f_i(\boldsymbol{\theta}_*)^\top$ and $\tilde{y}_i := \boldsymbol{\phi}_i^\top \boldsymbol{\theta}_* + e_i$ with $e_i = y_i - f_i(\boldsymbol{\theta}_*)$. This can be viewed as approximating the original non-conjugate terms by conjugate factors (see Sec. 5.4 in (Khan & Rue, 2023)): $e^{-\ell(y_i, f_i(\boldsymbol{\theta}))} \approx e^{-\frac{1}{2}\left(\tilde{y}_i - \boldsymbol{\phi}_i^\top \boldsymbol{\theta}\right)^2}$, which take a similar interpretation to site functions in expectation propagation. This is used along with the standard formula for online Bayesian updating to derive the approximate AOI posterior $\hat{q}_*^+(\boldsymbol{\theta}) \approx p(\boldsymbol{\theta}|\mathcal{D}_{N+1})$,

$$p(\boldsymbol{\theta}|\mathcal{D}_{N+1}) \propto \prod_{i=1}^{N+1} e^{-\ell(y_i, f_i(\boldsymbol{\theta}))} p(\boldsymbol{\theta}) \tag{47}$$

▷ split off the $(N+1)^{\text{th}}$ likelihood from the others

$$= e^{-\ell(y, f_{N+1}(\boldsymbol{\theta}))} \prod_{i=1}^{N} e^{-\ell(y_i, f_i(\boldsymbol{\theta}))} p(\boldsymbol{\theta}) \tag{48}$$

▷ apply the Laplace-GGN posterior approximation

$$\approx e^{-\ell(y, f_{N+1}(\boldsymbol{\theta}))} q_*(\boldsymbol{\theta}) \tag{49}$$

▷ approximate the $(N+1)^{\text{th}}$ likelihood by its site function

$$\approx e^{-\frac{1}{2}\left(\tilde{y} - \boldsymbol{\phi}_{N+1}^\top \boldsymbol{\theta}\right)^2} q_*(\boldsymbol{\theta}) \tag{50}$$

▷ this corresponds to an unnormalized Gaussian distribution

$$\propto \mathbb{N}(\boldsymbol{\theta}|\hat{\boldsymbol{\theta}}_*^+(y), \hat{\boldsymbol{\Sigma}}_*^+) \tag{51}$$

F Multi-output regression prediction intervals

For a Bayesian posterior predictive distribution (under Gaussian assumptions) in the multi-output case, $p(\mathbf{y}_*|\mathbf{x}_*, \mathcal{D}) = \mathbb{N}(\mathbf{y}_*|\hat{\mathbf{y}}_*, \boldsymbol{\Sigma}_{y_*})$, the confidence *region* is given by an ellipsoid,

$$C_\alpha(\mathbf{x}_*) = \{\mathbf{y} \in \mathbb{R}^O : (\mathbf{y} - \hat{\mathbf{y}}_*)^\top \boldsymbol{\Sigma}_{y_*}^{-1}(\mathbf{y} - \hat{\mathbf{y}}_*) \leq \chi^2_{O,\alpha}\} \tag{52}$$

where $\chi^2_{O,\alpha}$ is the quantile function for the chi-squared distribution with O degrees of freedom. Using this definition, it is straightforward to evaluate empirical coverage. Efficiency is then given by the volume of the corresponding ellipsoid,

$$\text{Vol}\big[C_\alpha(\mathbf{x}_*)\big] = (\chi^2_{O,\alpha})^{\frac{O}{2}} \det(\boldsymbol{\Sigma}_{y_*})^{\frac{1}{2}} \text{Vol}[B_O] \tag{53}$$

where B_O is the unit ball with O dimensions.

Table 3: We have number of test points (M), number of grid points (K), number of train points (N), parameter count (D), total epochs (E), and cost of forward pass/gradient computation/Jacobian computation ($[FP]$). We highlight the additional complexity of FCP over ACP-GN (purple), ACP over ACP-GN (green), ACP-GN over ACP (blue) and ACP/ACP-GN over FCP (red).

	Train	**Predict**
FCP	—	$MKN^2E[FP]$
ACP	$NE[FP] + ND^2 + D^3$	$MK([FP] + ND^2)$
ACP-GN	$NE[FP] + ND^2 + D^3$	$M([FP] + ND^2 + N \log N)$

G Time complexity: FCP vs. ACP-GN

We write the time complexity for our ACP-GN and compare it against full conformal prediction (FCP) and approximate full conformal prediction (ACP) (Martinez et al., 2023) in Table 3. This is shown for the deleted nonconformity score similar to Sec. A.2 in Martinez et al. (2023) and for scalar targets only. For the standard score, ACP and ACP-GN are unchanged but the factor of N is dropped in FCP. "Train" refers to the upfront time complexity that can be re-used when constructing prediction intervals ("predict") for new batches of test points.

We state the best time complexity of the ridge regression confidence machine routine given the coefficients $\{(a_i, b_i)\}_{i=1}^{N+1}$ as $\mathcal{O}(N \log N)$ (see Sec. 2.3 in (Vovk et al., 2005)). This is the same as the asymmetric implementation of Burnaev & Vovk (2014) shown in Alg. 2 (sorting the N change-points). The cost of network prediction (single forward pass) is architecture-dependent but since this is constant across the methods we do not expand on this here. We also regard the cost of gradient and Jacobian computation as comparable to a forward pass (see the discussion in App. D in Novak et al. (2022)).

FCP has a complexity multiplicative in the number of test points, grid points, train points (twice) and epochs. Whereas for ACP-GN, it is just multiplicative in the number of test points, given an upfront cost which is cubic in the number of parameters. We can further reduce the time complexity for ACP-GN when scalable approximations to the Gauss-Newton Hessian are used (as investigated in App. J.1). These extensions could also be adapted for ACP which was in fact left as future work in Martinez et al. (2023). This is shown in Table 4 for the cases of Kronecker-factored approximate curvature (KFAC) and last-layer approximation (LL). Crucially, both these approximations relax the cubic dependence on the parameter count P which often exceeds millions of parameters for modern architectures. Instead, the cubic dependence shifts to the input and output dimensionality of the network layers in the case of KFAC or just the last layer, which are typically far smaller than P.

Table 4: We have number of network layers (L), layer input/output dimensionality ($I_{l,\text{in}}/I_{l,\text{out}}$), cost of Gauss-Newton Hessian evaluation/inversion (H_N) along with its KFAC (H_N^{kfac}) and last-layer (H_N^{LL}) approximation, and cost of operations related to Gauss-Newton influence in the KFAC case $[INF]^{kfac}$. Other terms are defined in Table 3.

	Train	**Predict**
ACP-GN	$NE[FP] + H_N$[a]	$M([FP] + ND^2 + N \log N)$
ACP-GN*(kfac)*	$NE[FP] + H_N^{kfac}$[b]	$M([FP] + [INF]^{kfac}$[d] $+ N \log N)$
ACP-GN*(LL)*	$NE[FP] + H_N^{LL}$[c]	$M([FP] + NI_{L,\text{in}}^2 + N \log N)$

[a] $H_N = N[FP] + ND^2 + D^3$
[b] $H_N^{kfac} = N[FP] + N\sum_{l=1}^{L}(I_{l,\text{in}}^2 + I_{l,\text{out}}^2) + \sum_{l=1}^{L}(I_{l,\text{in}}^3 + I_{l,\text{out}}^3)$
[c] $H_N^{LL} = N[FP] + NI_{L,\text{in}}^2 + I_{L,\text{in}}^3$
[d] $[INF]^{kfac} = N\left(D + \sum_{l=1}^{L}(I_{l,\text{out}}I_{l,\text{in}}^2 + I_{l,\text{in}}I_{l,\text{out}}^2)\right)$

H Related Work

Computationally efficient versions of conformal prediction There are approaches based on data splitting such as split-CP (Papadopoulos et al., 2002; Lei et al., 2018) and cross-conformal prediction (Vovk, 2015; Barber et al., 2021). Such approaches greatly reduce the number of models that need to be trained (in the case of split-CP just one), but this comes at the expense of statistical efficiency. There are also normalized variants of split-CP (Papadopoulos et al., 2008) and other extensions in the regression setting such as conformalized quantile regression (Romano et al., 2019) and regression-to-classification CP (Guha et al., 2024). For certain model classes, computationally efficient implementations of full-CP exist such as ridge regression (Nouretdinov et al., 2001; Burnaev & Vovk, 2014), Lasso regression (Lei, 2019) and k-Nearest Neighbours Regression (Papadopoulos et al., 2011). For more general settings, there are approaches based on homotopy continuation method (Ndiaye & Takeuchi, 2019) and algorithmic stability Ndiaye (2022). The method of Martinez et al. (2023) is closely related to ours where they use influence function (Jaeckel, 1972; Cook & Weisberg, 1980) to approximate the retraining step in the full-CP algorithm. However, they only consider classification problems and their solution still requires an exhaustive search over possible labels, which is unfeasible for the regression setting we consider in this paper. A related approach is Alaa & Van Der Schaar (2020) that used higher-order influence functions to approximate leave-one-out retraining in jackknife+ (Barber et al., 2021) and Schulam & Saria (2019) that approximates the bootstrap (Efron & Tibshirani, 1986) using similar sensitivity-based techniques.

Influence functions These are a family of techniques that estimate the effect on the model of deleting a single example (or group of examples) from the training set, without the computationally prohibitive cost of retraining the model. There are two main approaches, the first is the infinitesimal jackknife (IJ) or simply *influence function* that the wider family of techniques takes its namesake. This was first proposed in robust statistics (Jaeckel, 1972; Hampel, 1974) and then was introduced as an influence measure for regression diagnostics and outlier detection in Cook & Weisberg (1980). It was popularized in deep learning by Koh & Liang (2017) focusing on diagnosing model errors and data attribution. The other approach, that we use in this work, is the Newton-step (NS) influence which was first proposed in Pregibon (1981) for logistic regression and generalized to more general losses and regularizers in Beirami et al. (2017). Closely related to our use case, influence functions have been used to approximate cross-validation: IJ in Giordano et al. (2019b) and NS influence in le Cessie & van Houwelingen (1992); Beirami et al. (2017); Rad & Maleki (2020). They have also been proposed for machine unlearning: IJ in Guo et al. (2020) and NS influence in Sekhari et al. (2021). Influence functions derived using higher-order Taylor approximations have been proposed (Giordano et al., 2019a; Basu et al., 2020). Many recent works have focused on scaling IJ such as Fisher information matrix (Teso et al., 2021) or generalized Gauss-Newton (Bae et al., 2022) approximations to the Hessian, or measures defined using only gradients (Pruthi et al., 2020). Similar techniques to our Gauss-Newton influence (in the leave-one-out setting) have appeared in past works (Laurent & Cook, 1992; Hansen & Larsen, 1996; Monari & Dreyfus, 2000).

Network linearization The use of gradients or Jacobians as features first appeared in Jaakkola & Haussler (1998) which they called "Fisher vectors". Such features have been demonstrated for fast adaptation (Maddox et al., 2021), transfer learning (Mu et al., 2020) and improving predictions in Bayesian neural networks (Immer et al., 2021b). This local linearization has received much attention in recent years due to the neural tangent kernel (Jacot et al., 2018) of which its finite-width variety can be viewed as the kernel analogue of Fisher-vectors.

I Experimental details

I.1 UCI Regression

For every dataset, both the input features and targets are standardized to have zero mean and unit variance. We use a batch size of 256 in SGD training with an initial learning rate of 10^{-2} that is decayed to 10^{-5} using a cosine schedule. All methods require tuning the L2 regularizer/prior precision. Additionally, for Linearized Laplace we have the observation noise. For the small and medium datasets, these are tuned using online marginal likelihood optimization (Immer et al., 2021a) that alternates between standard neural network training and gradient-based updates to the hyperparam-

Table 5: We repeat the bounding box experiment in Table 2 but with 50% calibration split.

	Avg. Volume ($\times 10^{-2}$)			Avg. Coverage		
	85%	90%	95%	85%	90%	95%
LA	$0.710_{\pm 0.009}$	$0.945_{\pm 0.013}$	$1.405_{\pm 0.019}$	$92.11_{\pm 0.18}$ (✗)	$94.49_{\pm 0.18}$ (✗)	$96.65_{\pm 0.16}$ (✗)
SCP	$1.177_{\pm 0.023}$	$1.723_{\pm 0.037}$	$3.137_{\pm 0.081}$	$87.36_{\pm 0.27}$ (✗)	$91.87_{\pm 0.27}$ (✗)	$95.91_{\pm 0.19}$ (✓)
CRF	$1.248_{\pm 0.030}$	$1.900_{\pm 0.044}$	$3.781_{\pm 0.092}$	$87.48_{\pm 0.34}$ (✗)	$91.96_{\pm 0.25}$ (✗)	$96.02_{\pm 0.20}$ (✓)
CQR	$1.627_{\pm 0.027}$	$2.739_{\pm 0.052}$	$7.971_{\pm 0.159}$	$87.46_{\pm 0.31}$ (✗)	$91.42_{\pm 0.28}$ (✓)	$95.97_{\pm 0.14}$ (✓)
ACP-GN	$0.384_{\pm 0.004}$	$0.605_{\pm 0.007}$	$1.225_{\pm 0.017}$	$90.97_{\pm 0.20}$ (✗)	$94.09_{\pm 0.20}$ (✗)	$97.41_{\pm 0.16}$ (✗)
SCP-GN	$\mathbf{1.149}_{\pm 0.025}$	$1.615_{\pm 0.034}$	$2.643_{\pm 0.054}$	$86.84_{\pm 0.29}$ (✓)	$91.35_{\pm 0.31}$ (✓)	$95.56_{\pm 0.21}$ (✓)
ACP-GN (split + refine)	$0.365_{\pm 0.006}$	$\mathbf{0.556}_{\pm 0.010}$	$\mathbf{1.104}_{\pm 0.028}$	$87.78_{\pm 0.37}$ (✗)	$91.75_{\pm 0.31}$ (✓)	$95.62_{\pm 0.17}$ (✓)

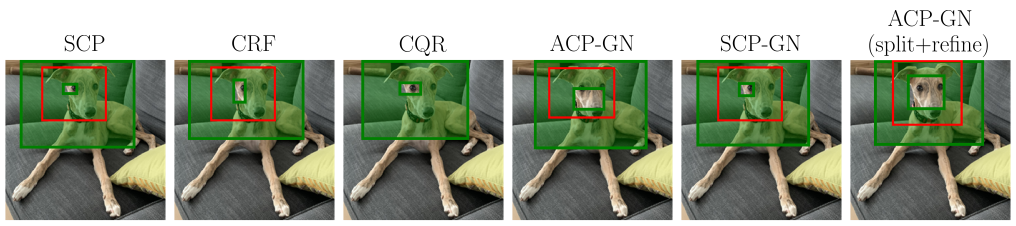

Figure 2: Demonstration of our method (starting 3rd from right) for object localization alongside common conformal prediction baselines. The two-sided prediction regions are shown in green and neural network prediction is shown in red.

eters using the differentiable marginal likelihood estimate. We use a layerwise structure in the prior precision. The marginal likelihood estimate is also used for early stopping. The prior precision and observation noise are initialized to 1. Overall, using Adam optimizer we run for 5000 epochs with a hyperparameter learning rate of 10^{-2} decayed to 10^{-3} using a cosine schedule, 100 burn-in epochs, and take 50 hyperparameter steps on single marginal likelihood evaluation every 50 epochs.

For the large datasets, the (scalar) prior precision is tuned via grid-search for each order of magnitude from 10^{-2} to 10^{4}. 10% of the training set is used for validation and once the best prior precision is found, the network is retrained on full training set. The observation noise is fit to the training data via maximum likelihood after training.

We follow the above procedure for all methods except "ACP-GN (split + refine)" where we first train with L2 regularizer fixed to $\delta/N = 10^{-4}$. For the small datasets, the hyperparameters are tuned via post-hoc marginal likelihood training with 5000 steps and the same learning rate schedule described earlier. In Fig. 3 we demonstrate the trade-off between coverage and efficiency in the split variant of ACP-GN. We set the sensitivity hyperparameter (β) of CRF (see Eq. 16 in (Papadopoulos & Haralambous, 2011)) to 1 as used in Romano et al. (2019). The additional network is identical to the original one and re-uses the same hyperparameters and training configuration. It is worth mentioning that the original proposal of CRF in the context of neural networks (Papadopoulos et al., 2002) used a ridge regression model to predict residuals.

The quantile regression network in CQR uses the same architecture as that used in the other methods except there are two output units. As done in CRF, hyperparameters from SCP are re-used along with the training configuration. Rather than retraining for each desired significance level, the output layer is appended with additional units and trained jointly. We do not perform tuning of the quantiles so it is expected the intervals can be made more efficient (Romano et al., 2019).

Results on additional UCI datasets are reported in Table 8. Furthermore, in Table 9 and 10 we repeat all experiments with 25% calibration split.

I.2 Bounding Box Localization

The training setup is adapted from Girshick (2015) using SGD optimizer and simple data augmentation involving random horizontal flips of probability 0.5. The robust L1 loss is used for the bounding box regression head and logistic loss for the classification head. With L1 loss, Eqs. (12) to (14) no longer hold but we demonstrate the efficacy of our procedure when the objectives for training and predictive interval construction differ.

Images are resized to 224×224 followed by scaling the pixel values to $[0, 1]$ and then normalizing to statistics computed from the ImageNet dataset as required by the pretrained VGG-19 backbone (Simonyan & Zisserman, 2015). Bounding box coordinates denote the top-left and bottom-right corners and are scaled to $[0, 1]$. The reported volumes use the standardized targets. A demonstration is shown in Fig. 2.

For SGD training we use a batch size of 128 for 200 epochs and an initial learning rate of 10^{-2}. A learning rate schedule is used that takes incremental steps towards the base learning rate for the first 5 epochs (warm-up) and then decays to 0 using a cosine schedule. The SGD optimizer uses nesterov momentum 0.9 and weight decay 5×10^{-4}. At inference with squared-error loss, we perform *post-hoc* finetuning of hyperparameters (regularization coefficient, observation noise) using the marginal likelihood. This is optimized using Adam for 250 epochs with a learning rate of 10^{-1}. For LA, we use the expressions outlined in App. F to evaluate the volume and coverage.

Due to the last-layer approximation, the off-diagonal entries of the $O \times O$ multi-output leverage scores are zero. We can understand this by realizing the Jacobian of an output with respect to a parameter tied to a different output is zero. Due to this inherent output independence, we can simply use the expressions given in the single-output case in parallel over the output dimensions rather than use the expressions derived for the multi-output setting in App. D.

When training the additional network in CRF and the quantile regression network in CQR, the backbone parameters are initialized to those of the original network and not updated. They are trained in a similar fashion except for 100 epochs, without warm-up in the learning rate schedule and without data augmentation. The sensitivity hyperparameter (β) of CRF is set to 0.01 after trying a few different values on a single seed.

In Table 5, we repeat the experiment but with 50% calibration split. In this case the model achieves 99.4% classification accuracy and 37.0% localization error.

J Additional experiments

J.1 Ablation with scalable approximations to the Gauss-Newton matrix

For most deep architectures, storing and inverting the full Gauss-Newton matrix is infeasible. We investigate two choices for scalable approximations to the Gauss-Newton matrix in the context of our experiments involving the large UCI datasets. These are inspired by two popular choices for scalable Laplace approximations.

The first is the use of Kronecker-factored approximate curvature (KFAC) (Martens & Grosse, 2015) as an approximation to the GN. By approximating each layer's Gauss Newton independently as a Kronecker product, this leads to a block-diagonal factorization for the overall GN matrix enabling efficient storage and computation of inverses. We use the specific form proposed in Immer et al. (2021b) that performs an eigendecomposition of the Kronecker factors and avoids the "dampening" approximation of Ritter et al. (2018). With regards to refinement, we observed that the one-step solve with KFAC approximation performs much worse than without refinement. Hence, we instead take a gradient-based approach as outlined in Antorán et al. (2022) optimizing the linearized network's objective using a similar configuration to the original NN training. The linearized network's loss gradients are evaluated without explicitly instantiating Jacobians via vector-Jacobian and Jacobian-vector products.

The results are shown in Table 11. For LA, we observe that KFAC leads to slightly tighter intervals but the miscoverage increases. In the case of SCP-GN, we find KFAC to be very competitive to the full Gauss-Newton with little change to the interval width or coverage. For ACP-GN we show results for the standard (*i.e.* AOI) nonconformity score as opposed to the studentized nonconformity score reported in Table 1 and 8. This is because we found that the deleted and studentized nonconformity scores combined with the KFAC approximation performed poorly. However for the reported standard nonconformity score, we observe KFAC leads to much tighter intervals with just a slight increase in miscoverage. For the "ACP-GN (split + refine)", on all datasets except bike, we observe tighter intervals with KFAC however the miscoverage is often slightly greater.

Table 6: We compare against Approximate full Conformal Prediction (ACP) that uses a target discretization strategy.

		Avg. Width			Avg. Coverage		
		90%	95%	99%	90%	95%	99%
`energy`	**ACP**	1.706±0.045	2.219±0.055	3.694±0.070	87.41±0.28 (✓)	93.44±0.31 (✓)	98.58±0.16 (✓)
N=768	**ACP-GN**	**1.467**±0.010	**1.882**±0.013	**3.097**±0.015	88.32±0.40 (✓)	93.87±0.29 (✓)	99.01±0.09 (✓)
I=8	**ACP-GN** (split + refine)	1.745±0.016	2.174±0.021	3.300±0.045	90.54±0.25 (✓)	94.96±0.22 (✓)	99.18±0.10 (✓)
`concrete`	**ACP**	17.854±0.104	21.966±0.119	31.068±0.139	90.36±0.15 (✓)	94.97±0.13 (✓)	98.43±0.07 (✓)
N=1,030	**ACP-GN**	**15.690**±0.112	**19.856**±0.141	**29.179**±0.190	89.95±0.18 (✓)	94.96±0.13 (✓)	98.91±0.06 (✓)
I=8	**ACP-GN** (split + refine)	18.907±0.103	24.012±0.129	35.949±0.387	90.26±0.20 (✓)	95.23±0.27 (✓)	99.17±0.06 (✓)
`wine`	**ACP**	2.158±0.003	2.676±0.005	**3.630**±0.008	90.46±0.13 (✓)	94.88±0.08 (✓)	98.22±0.07 (✓)
N=1,599	**ACP-GN**	**2.091**±0.005	**2.651**±0.006	3.797±0.013	90.91±0.11 (✓)	95.50±0.10 (✓)	99.14±0.05 (✓)
I=11	**ACP-GN** (split + refine)	2.474±0.007	3.054±0.008	4.324±0.020	89.56±0.27 (✓)	94.81±0.13 (✓)	98.99±0.09 (✓)
`kin8nm`	**ACP**	0.207±0.001	**0.255**±0.001	**0.355**±0.001	88.83±0.28 (✗)	94.16±0.19 (✓)	98.77±0.10 (✓)
N=8,192	**ACP-GN**	**0.213**±0.001	0.262±0.001	0.365±0.002	89.68±0.28 (✓)	94.58±0.20 (✓)	98.85±0.08 (✓)
I=8	**ACP-GN** (split + refine)	0.232±0.001	0.284±0.001	0.406±0.003	90.45±0.17 (✓)	95.02±0.22 (✓)	99.10±0.07 (✓)
`power`	**ACP**	**12.131**±0.022	**14.840**±0.017	**20.804**±0.050	89.28±0.25 (✓)	94.68±0.18 (✓)	98.82±0.10 (✓)
N=9,568	**ACP-GN**	12.526±0.024	15.248±0.020	21.592±0.062	90.16±0.26 (✓)	95.13±0.19 (✓)	98.89±0.08 (✓)
I=4	**ACP-GN** (split + refine)	12.744±0.045	15.442±0.039	22.043±0.151	90.17±0.29 (✓)	95.26±0.17 (✓)	98.98±0.09 (✓)
`community`	**ACP**	**0.401**±0.001	**0.502**±0.003	0.700±0.005	89.71±0.48 (✓)	94.00±0.36 (✓)	97.71±0.30 (✗)
N=1,994	**ACP-GN**	0.459±0.003	0.594±0.004	**0.936**±0.007	90.68±0.55 (✓)	95.21±0.38 (✓)	99.18±0.14 (✓)
I=100	**ACP-GN** (split + refine)	0.521±0.006	0.654±0.008	1.011±0.017	90.95±0.64 (✓)	95.26±0.43 (✓)	99.16±0.17 (✓)
`bike`	**ACP**	**93.647**±1.300	**121.850**±1.603	186.090±2.393	89.03±0.31 (✓)	94.36±0.22 (✓)	98.43±0.14 (✗)
N=10,886	**ACP-GN**	97.966±1.193	130.732±1.565	216.677±2.917	89.12±0.21 (✗)	94.32±0.18 (✗)	98.72±0.09 (✗)
I=18	**ACP-GN** (split + refine)	130.933±5.221	174.190±7.065	**288.603**±12.066	90.09±0.25 (✓)	94.93±0.19 (✓)	99.06±0.07 (✓)

The second choice is a last-layer approximation or neural linear model approach (Ober & Rasmussen, 2019) that can be considered a special case of subnetwork inference for Linearized Laplace (Daxberger et al., 2021b). This makes the AOI estimation exact with respect to a linear model whose basis features are given by the activations of the penultimate layer. This can be recovered as a special case of Gauss-Newton influence. This leads to storing and inverting a much smaller matrix whose size corresponds to the number of parameters in the final layer. In the context of Linearized Laplace, the last-layer approximation can be combined with KFAC for increased scalability but we do not investigate this configuration here.

The results are shown in Table 12. The last-layer approximation has the effect of making the existing undercoverage in LA much worse. In the case of SCP-GN, the approximation leads to the intervals being no more efficient than SCP, that is without normalization, whilst maintaining correct coverage. For ACP-GN, the intervals undercover by a large margin on the bike, community and protein datasets. The split and refine variant is still able to successfully correct this attaining the desired coverage whilst being competitive to the full Gauss-Newton on tightness. We also observed the different varieties of nonconformity score all performed similarly – results reported are those of the studentized variety as in Table 1 and 8.

J.2 Comparison against Approximate full Conformal Prediction

We compare ACP-GN and ACP-GN (split+refine) to Approximate full Conformal Prediction (ACP) (Martinez et al., 2023) on several UCI datasets taken from Table 1 and 8. ACP uses the AOI nonconformity score (referred to as the ordinary scheme in (Martinez et al., 2023)). Whilst the exact Hessian is computed in (Martinez et al., 2023) with a damping term to ensure positive eigenvalues, we approximate the Hessian by a (damped) Gauss-Newton matrix in order to keep approximations consistent in the comparison. The damping term is tuned in the same way as ACP-GN, as described in App. I.1. The Gauss-Newton matrix has been used in previous works when evaluating influence function (Bae et al., 2022; Nickl et al., 2023). Furthermore, we use "direct approach" (see Eq. 4 in (Martinez et al., 2023)). Martinez et al. (2023) only considered classification tasks so ACP needs to be adapted for regression. We define a grid of candidate targets using a simple discretization strategy that constructs a fine, uniform grid of 200 points delimited by the training targets.

Table 6 shows that in some datasets and target coverage levels, ACP-GN is more efficient but in others ACP is more efficient. However, we used 200 grid points which exceeds the upper end of what was investigated in prior work (Chen et al., 2018). We also emphasise that ACP-GN has lower space complexity since it only computes changepoints for each combination of test and train point, whereas ACP computes residuals for each combination of test point, train point and candidate target value. For the datasets considered we did not observe a considerable difference in the running time between the two methods but we do expect for larger datasets ACP to be slower due to the need for batched computation.

J.3 Comparison against Full Conformal Prediction

We compare full conformal prediction (FCP) against ACP-GN and ACP (Martinez et al., 2023) as well as LA and SCP. This is evaluated on a synthetic dataset with outliers (500 train points and 100 test points) taken from Papadopoulos (2024) (see Sec. 5.2). This generates data from a Gaussian Process prior with RBF kernel. There is a coin flip of probability 0.1 for which the observation noise standard deviation is increased by a factor of 10. The experiment is repeated 20 times with different seeds also controlling the data generation. A MLP with a single hidden layer, 100 units and Tanh activation function is used throughout. This is trained using Adam for 500 epochs (full-batch training) with an initial learning rate of 10^{-2} that decays following a cosine schedule to 10^{-5}. We use a uniform grid of 50 points for the target discretization in FCP which is on the lower end as suggested by previous works (Chen et al., 2018) and enough to give valid coverage. ACP follows the configuration outlined in App. J.2 with 200 grid points. The AOI nonconformity score is used throughout for FCP, ACP and ACP-GN. SCP uses a 50% calibration split. Hyperparameters are tuned using the online marginal likelihood procedure as described in App. I.1 which we exclude from the running time.

As Table 7 shows, all conformal methods give the correct coverage but FCP is indeed the most efficient. However, the running time is a factor of 10^4 slower than all other methods including ACP and ACP-GN. This experiment was run on a NVIDIA A100 GPU.

Table 7: We compare against Full Conformal Prediction (FCP) on a synthetically-generated dataset.

	Avg. Width			Avg. Coverage			Time $(\times 10^2)$
	90%	95%	99%	90%	95%	99%	
LA	1.061±0.025	**1.264**±0.030	1.661±0.039	94.45±0.56 (✗)	95.15±0.52 (✓)	96.60±0.45 (✗)	0.034±0.001
SCP	0.623±0.024	1.466±0.090	3.632±0.131	91.10±0.82 (✓)	96.10±0.45 (✓)	99.50±0.15 (✓)	0.023±0.001
FCP	**0.451**±0.011	**1.287**±0.053	**3.090**±0.092	87.00±1.06 (✓)	95.25±0.52 (✓)	98.90±0.32 (✓)	169.527±3.560
ACP	0.532±0.013	1.370±0.054	3.177±0.094	90.35±0.79 (✓)	95.50±0.49 (✓)	99.00±0.32 (✓)	0.035±0.001
ACP-GN	0.581±0.021	1.450±0.053	3.529±0.105	91.00±0.81 (✓)	95.60±0.48 (✓)	99.25±0.27 (✓)	0.034±0.000

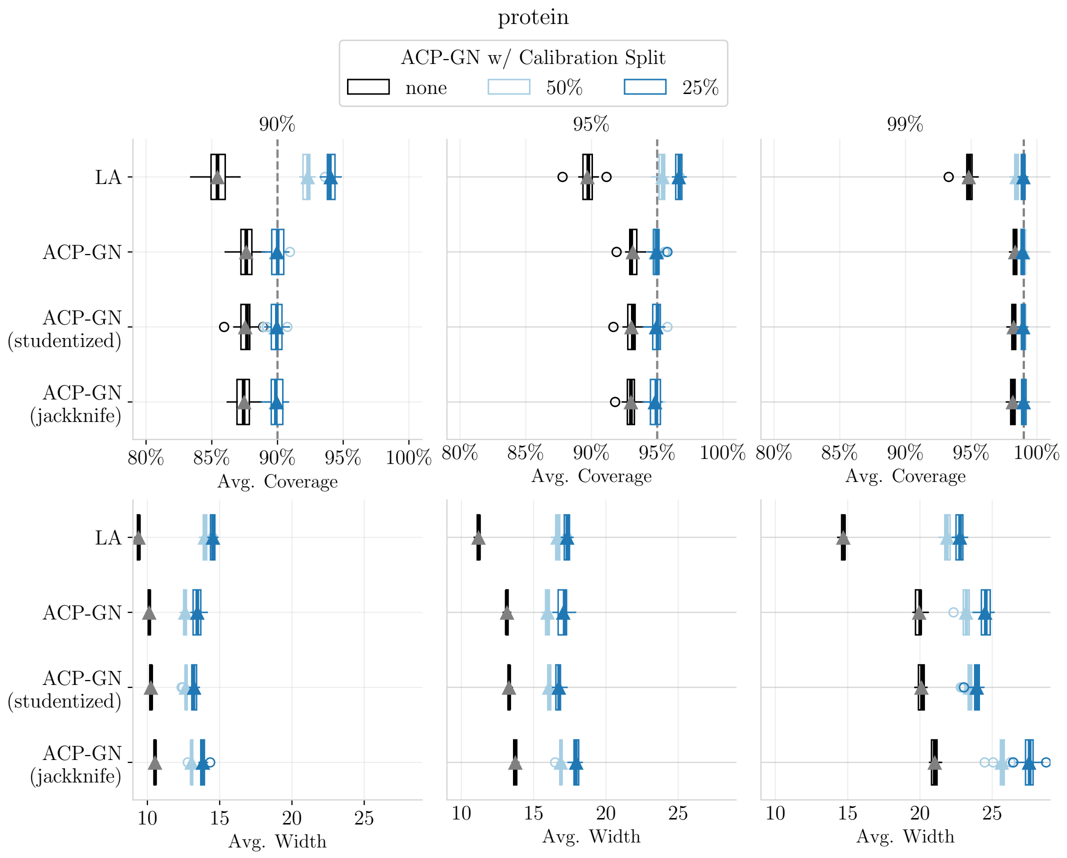

Figure 3: Our extension to ACP-GN that employs a separate calibration set to evaluate nonconformity scores combined with *refinement* of the linearized model shows considerable improvement to the coverage. Unsurprisingly it leads to larger average interval widths due to a smaller training set. This is shown for `protein` dataset at $\{90\%, 95\%, 99\%\}$ confidence levels.

Table 8: Additional results on UCI regression datasets.

			Avg. Width			Avg. Coverage	
		90%	95%	99%	90%	95%	99%
	LA	**15.523**±0.087	**18.497**±0.103	24.310±0.136	89.30±0.17 (✓)	93.53±0.10 (✓)	97.37±0.11 (✗)
	SCP	19.216±0.124	24.526±0.194	42.110±0.802	89.92±0.29 (✓)	94.97±0.14 (✓)	99.11±0.05 (✓)
`concrete`	**CRF**	18.623±0.117	23.737±0.167	40.390±0.506	89.75±0.36 (✓)	94.97±0.13 (✓)	99.05±0.06 (✓)
N=1,030	**CQR**	22.486±0.081	27.145±0.091	39.827±0.212	90.51±0.27 (✓)	95.24±0.16 (✓)	99.01±0.08 (✓)
I=8	**ACP-GN**	15.727±0.114	19.810±0.147	**29.192**±0.179	89.60±0.20 (✓)	94.77±0.14 (✓)	98.85±0.08 (✓)
	SCP-GN	18.851±0.118	23.780±0.121	36.686±0.313	89.68±0.28 (✓)	94.87±0.14 (✓)	99.15±0.07 (✓)
	ACP-GN (split + refine)	18.907±0.103	24.012±0.129	35.949±0.387	90.26±0.20 (✓)	95.23±0.27 (✓)	99.17±0.06 (✓)
	LA	2.099±0.001	**2.501**±0.002	**3.287**±0.002	91.12±0.13 (✓)	94.58±0.10 (✓)	98.34±0.05 (✓)
	SCP	2.183±0.009	2.768±0.012	3.941±0.020	90.41±0.19 (✓)	95.07±0.12 (✓)	99.12±0.04 (✓)
`wine`	**CRF**	2.304±0.009	2.921±0.015	4.381±0.031	90.14±0.18 (✓)	95.16±0.15 (✓)	99.02±0.05 (✓)
N=1,599	**CQR**	**1.977**±0.005	**2.490**±0.016	3.867±0.027	90.24±0.12 (✓)	95.05±0.06 (✓)	99.07±0.06 (✓)
I=11	**ACP-GN**	2.103±0.003	2.665±0.006	3.827±0.007	91.16±0.12 (✓)	95.80±0.05 (✓)	99.17±0.04 (✓)
	SCP-GN	2.134±0.007	2.676±0.015	3.753±0.026	90.06±0.24 (✓)	95.09±0.16 (✓)	99.01±0.05 (✓)
	ACP-GN (split + refine)	2.474±0.007	3.054±0.008	4.324±0.020	89.56±0.27 (✓)	94.81±0.13 (✓)	98.99±0.09 (✓)
	LA	**0.213**±0.001	0.254±0.001	0.334±0.001	89.66±0.29 (✓)	93.88±0.23 (✗)	98.09±0.10 (✗)
	SCP	0.231±0.001	0.285±0.001	0.409±0.003	90.54±0.29 (✓)	95.31±0.20 (✓)	99.18±0.08 (✓)
`kin8nm`	**CRF**	0.229±0.001	0.282±0.001	0.401±0.002	90.59±0.31 (✓)	95.41±0.23 (✓)	99.18±0.06 (✓)
N=8,192	**CQR**	0.254±0.001	0.304±0.001	0.426±0.003	90.24±0.27 (✓)	95.21±0.25 (✓)	99.02±0.09 (✓)
I=8	**ACP-GN**	**0.213**±0.001	**0.262**±0.001	**0.365**±0.002	89.68±0.28 (✓)	94.58±0.20 (✓)	98.85±0.08 (✓)
	SCP-GN	0.230±0.001	0.282±0.001	0.400±0.002	90.57±0.29 (✓)	95.20±0.22 (✓)	99.18±0.07 (✓)
	ACP-GN (split + refine)	0.232±0.001	0.284±0.001	0.406±0.003	90.45±0.17 (✓)	95.02±0.22 (✓)	99.10±0.07 (✓)
	LA	13.345±0.028	15.901±0.034	**20.898**±0.045	92.08±0.23 (✗)	95.86±0.17 (✗)	98.86±0.09 (✓)
	SCP	12.732±0.040	15.359±0.041	21.688±0.143	90.27±0.24 (✓)	94.94±0.17 (✓)	98.98±0.10 (✓)
`power`	**CRF**	**12.573**±0.039	**15.130**±0.039	21.548±0.145	90.26±0.27 (✓)	94.93±0.18 (✓)	98.98±0.09 (✓)
N=9,568	**CQR**	12.860±0.026	**15.180**±0.042	21.946±0.154	90.37±0.24 (✓)	94.93±0.14 (✓)	98.89±0.06 (✓)
I=4	**ACP-GN**	**12.526**±0.024	15.248±0.020	21.592±0.062	90.16±0.26 (✓)	95.13±0.19 (✓)	98.89±0.08 (✓)
	SCP-GN	12.736±0.041	15.362±0.044	21.680±0.146	90.25±0.25 (✓)	94.95±0.18 (✓)	98.98±0.10 (✓)
	ACP-GN (split + refine)	12.744±0.045	15.442±0.039	22.043±0.151	90.17±0.29 (✓)	95.26±0.17 (✓)	98.98±0.09 (✓)
	LA	0.548±0.074	0.653±0.088	0.858±0.116	90.90±0.59 (✓)	93.83±0.50 (✓)	97.05±0.33 (✗)
	SCP	0.534±0.010	0.731±0.020	1.159±0.028	90.30±0.42 (✓)	95.53±0.24 (✓)	99.12±0.16 (✓)
`community`	**CRF**	0.526±0.010	0.721±0.020	1.154±0.029	90.20±0.43 (✓)	95.33±0.25 (✓)	99.12±0.13 (✓)
N=1,994	**CQR**	0.554±0.020	**0.701**±0.034	**1.077**±0.038	90.90±0.57 (✓)	95.70±0.30 (✓)	99.35±0.15 (✓)
I=100	**ACP-GN**	**0.570**±0.108	**0.755**±0.158	**1.224**±0.285	90.90±0.62 (✓)	95.30±0.47 (✓)	99.25±0.14 (✓)
	SCP-GN	**0.474**±0.013	**0.656**±0.024	1.104±0.034	90.58±0.37 (✓)	94.95±0.27 (✓)	99.00±0.15 (✓)
	ACP-GN (split + refine)	0.519±0.006	**0.652**±0.008	**1.010**±0.016	90.97±0.61 (✓)	95.28±0.41 (✓)	99.12±0.16 (✓)
	LA	67.580±2.637	80.527±3.142	105.831±4.130	97.63±0.09 (✗)	98.15±0.08 (✗)	98.76±0.06 (✗)
	SCP	**20.771**±2.452	44.192±2.799	178.890±4.596	90.00±0.10 (✓)	95.03±0.08 (✓)	99.05±0.04 (✓)
`facebook_1`	**CRF**	**18.689**±1.944	35.765±1.870	121.181±4.148	89.98±0.12 (✓)	95.10±0.09 (✓)	99.08±0.04 (✓)
N=40,948	**CQR**	**19.625**±0.878	**25.970**±1.322	**59.922**±12.774	90.16±0.24 (✓)	94.91±0.11 (✓)	98.97±0.04 (✓)
I=53	**ACP-GN**	**17.986**±0.480	41.063±0.770	199.331±10.821	90.36±0.16 (✓)	95.56±0.17 (✗)	99.37±0.08 (✗)
	SCP-GN	**20.460**±2.463	39.712±3.148	125.078±8.075	90.00±0.10 (✓)	94.94±0.09 (✓)	99.04±0.03 (✓)
	ACP-GN (split + refine)	29.006±4.472	54.099±4.576	172.252±6.758	90.06±0.10 (✓)	95.20±0.08 (✓)	99.14±0.04 (✓)

Table 9: We repeat the UCI regression experiment in Table 1 but with 25% calibration split. In the case of yacht, the desired coverage of 99% was too small a significance level to accurately evaluate the quantile of the calibration scores.

		Avg. Width			Avg. Coverage		
		90%	95%	99%	90%	95%	99%
	LA	1.690±0.017	2.014±0.020	2.647±0.027	88.73±0.61 (✓)	90.78±0.59 (✗)	93.89±0.60 (✗)
`yacht`	**SCP**	2.306±0.117	3.866±0.133	—	90.11±0.70 (✓)	95.88±0.43 (✓)	—
N=308	**CRF**	2.281±0.112	3.818±0.130	—	90.11±0.67 (✓)	95.82±0.46 (✓)	—
I=6	**CQR**	3.236±0.151	4.895±0.211	—	90.33±0.65 (✓)	95.85±0.39 (✓)	—
	ACP-GN	**1.594**±0.016	**2.385**±0.029	**6.915**±0.067	87.36±0.58 (✓)	92.56±0.68 (✓)	99.03±0.11 (✓)
	SCP-GN	2.070±0.084	3.111±0.110	—	90.53±0.54 (✓)	95.95±0.51 (✓)	—
	ACP-GN (split + refine)	3.931±0.083	5.926±0.144	—	91.66±0.46 (✓)	96.66±0.41 (✓)	—
	LA	9.398±0.046	**11.199**±0.055	14.718±0.072	91.24±0.31 (✓)	94.34±0.22 (✓)	97.53±0.11 (✗)
	SCP	10.086±0.184	14.746±0.345	38.574±1.914	90.12±0.51 (✓)	96.03±0.34 (✓)	99.19±0.12 (✓)
`boston`	**CRF**	9.874±0.194	14.204±0.270	37.421±2.132	90.03±0.45 (✓)	95.52±0.24 (✓)	99.28±0.11 (✓)
N=506	**CQR**	10.918±0.121	14.618±0.208	31.130±0.823	90.09±0.31 (✓)	95.65±0.21 (✓)	99.17±0.10 (✓)
I=13	**ACP-GN**	**9.182**±0.046	12.111±0.038	**20.512**±0.057	90.64±0.26 (✓)	95.49±0.16 (✓)	99.11±0.08 (✓)
	SCP-GN	9.787±0.152	13.372±0.247	27.274±1.221	90.21±0.44 (✓)	95.49±0.29 (✓)	99.13±0.14 (✓)
	ACP-GN (split + refine)	17.372±0.252	22.852±0.371	42.081±1.170	90.68±0.44 (✓)	95.79±0.28 (✓)	98.95±0.15 (✓)
	LA	1.502±0.006	1.790±0.007	2.353±0.009	88.96±0.35 (✓)	92.92±0.33 (✗)	96.95±0.23 (✗)
	SCP	1.824±0.037	2.365±0.046	4.313±0.129	90.05±0.45 (✓)	95.04±0.25 (✓)	99.39±0.09 (✓)
`energy`	**CRF**	1.807±0.037	2.341±0.046	4.316±0.133	89.97±0.47 (✓)	95.02±0.23 (✓)	99.36±0.10 (✓)
N=768	**CQR**	4.779±0.032	5.181±0.035	7.692±0.171	90.55±0.26 (✓)	95.51±0.22 (✓)	99.35±0.06 (✓)
I=8	**ACP-GN**	**1.462**±0.006	**1.884**±0.008	**3.076**±0.015	88.28±0.33 (✓)	93.69±0.33 (✓)	98.88±0.11 (✓)
	SCP-GN	1.812±0.034	2.348±0.047	4.226±0.120	90.16±0.43 (✓)	95.19±0.24 (✓)	99.31±0.11 (✓)
	ACP-GN (split + refine)	2.498±0.056	3.251±0.072	5.601±0.168	90.19±0.53 (✓)	95.28±0.37 (✓)	99.43±0.07 (✓)
	LA	100.451±2.394	119.694±2.853	157.305±3.749	89.82±0.39 (✓)	93.29±0.33 (✗)	96.83±0.16 (✗)
	SCP	113.594±1.370	159.114±2.297	287.486±5.425	90.48±0.22 (✓)	95.11±0.20 (✓)	98.87±0.08 (✓)
`bike`	**CRF**	112.104±1.653	156.161±2.486	283.911±6.247	90.38±0.22 (✓)	95.17±0.19 (✓)	98.93±0.09 (✓)
N=10,886	**CQR**	**107.141**±3.598	**130.756**±3.259	**212.513**±5.131	90.34±0.26 (✓)	95.18±0.18 (✓)	98.97±0.09 (✓)
I=18	**ACP-GN**	**98.813**±2.485	130.893±3.231	213.131±5.630	89.36±0.43 (✓)	94.41±0.27 (✗)	98.67±0.09 (✗)
	SCP-GN	105.020±1.209	139.391±1.711	228.656±4.493	90.17±0.21 (✓)	95.10±0.16 (✓)	98.99±0.07 (✓)
	ACP-GN (split + refine)	137.114±0.759	178.639±1.307	285.997±3.834	90.20±0.18 (✓)	95.05±0.16 (✓)	99.16±0.10 (✓)
	LA	9.385±0.022	11.183±0.027	14.697±0.035	85.43±0.18 (✗)	89.69±0.15 (✗)	94.81±0.10 (✗)
	SCP	12.188±0.036	16.069±0.050	24.828±0.126	89.85±0.11 (✓)	94.91±0.07 (✓)	98.92±0.05 (✓)
`protein`	**CRF**	**11.532**±0.047	15.398±0.071	25.006±0.147	89.89±0.09 (✓)	94.93±0.08 (✓)	98.92±0.03 (✓)
N=45,730	**CQR**	13.174±0.162	**14.422**±0.148	**17.924**±0.078	90.10±0.09 (✓)	95.04±0.05 (✓)	98.92±0.04 (✓)
I=9	**ACP-GN**	10.243±0.019	13.294±0.027	20.101±0.053	87.54±0.15 (✗)	93.04±0.11 (✗)	98.24±0.05 (✗)
	SCP-GN	11.743±0.035	15.312±0.035	23.243±0.097	89.87±0.09 (✓)	94.91±0.07 (✓)	98.95±0.04 (✓)
	ACP-GN (split + refine)	13.229±0.042	16.741±0.055	23.927±0.068	89.96±0.11 (✓)	94.93±0.10 (✓)	98.94±0.05 (✓)
	LA	66.088±2.760	78.749±3.289	103.493±4.322	97.47±0.12 (✗)	98.01±0.09 (✗)	98.65±0.06 (✗)
	SCP	**15.739**±0.240	34.114±0.646	145.543±1.431	89.85±0.09 (✓)	94.96±0.06 (✓)	99.01±0.03 (✓)
`facebook_2`	**CRF**	**15.257**±0.663	29.889±1.398	100.709±6.136	89.82±0.08 (✓)	94.97±0.05 (✓)	99.01±0.03 (✓)
N=81,311	**CQR**	**17.929**±0.703	**22.314**±1.070	**32.257**±1.604	89.85±0.06 (✓)	95.00±0.04 (✓)	99.02±0.03 (✓)
I=53	**ACP-GN**	18.396±0.546	40.088±1.091	166.792±5.632	90.47±0.11 (✗)	95.45±0.08 (✗)	99.35±0.04 (✗)
	SCP-GN	**15.633**±0.220	33.192±0.708	122.079±4.142	89.81±0.07 (✓)	94.96±0.06 (✓)	99.00±0.02 (✓)
	ACP-GN (split + refine)	20.487±0.502	41.051±0.844	152.090±6.966	90.18±0.08 (✓)	95.08±0.07 (✓)	99.11±0.03 (✓)

Table 10: We repeat the UCI regression experiment in Table 8 but with 25% calibration split.

		Avg. Width			Avg. Coverage		
		90%	95%	99%	90%	95%	99%
	LA	**15.523**±0.087	**18.497**±0.103	24.310±0.136	89.30±0.17 (✓)	93.53±0.10 (✓)	97.37±0.11 (✗)
	SCP	17.563±0.209	22.787±0.277	39.972±1.300	89.35±0.43 (✓)	94.93±0.27 (✓)	99.02±0.12 (✓)
`concrete`	**CRF**	16.946±0.176	21.898±0.271	38.480±1.287	89.44±0.33 (✓)	94.88±0.30 (✓)	99.03±0.12 (✓)
N=1,030	**CQR**	20.063±0.092	24.620±0.144	37.035±0.329	89.88±0.27 (✓)	94.90±0.19 (✓)	99.11±0.07 (✓)
I=8	**ACP-GN**	15.727±0.114	19.810±0.147	**29.192**±0.179	89.60±0.20 (✓)	94.77±0.14 (✓)	98.85±0.08 (✓)
	SCP-GN	17.373±0.177	22.170±0.233	34.104±0.566	89.47±0.45 (✓)	94.94±0.25 (✓)	99.11±0.12 (✓)
	ACP-GN (split + refine)	25.331±0.182	32.279±0.244	47.494±0.795	90.30±0.40 (✓)	95.45±0.21 (✓)	99.13±0.11 (✓)
	LA	2.099±0.001	2.501±0.002	**3.287**±0.002	91.12±0.13 (✓)	94.58±0.10 (✓)	98.34±0.05 (✓)
	SCP	2.112±0.010	2.686±0.018	4.032±0.028	90.11±0.15 (✓)	95.00±0.11 (✓)	99.27±0.05 (✓)
`wine`	**CRF**	2.186±0.013	2.753±0.013	4.397±0.075	89.92±0.15 (✓)	94.92±0.13 (✓)	99.20±0.06 (✓)
N=1,599	**CQR**	**1.966**±0.004	**2.421**±0.016	3.920±0.031	89.97±0.22 (✓)	94.95±0.09 (✓)	99.12±0.08 (✓)
I=11	**ACP-GN**	2.103±0.003	2.665±0.006	3.827±0.007	91.16±0.12 (✓)	95.80±0.05 (✓)	99.17±0.04 (✓)
	SCP-GN	2.068±0.009	2.621±0.018	3.760±0.035	90.07±0.18 (✓)	95.00±0.11 (✓)	99.09±0.06 (✓)
	ACP-GN (split + refine)	2.908±0.016	3.527±0.023	5.089±0.037	90.14±0.20 (✓)	94.93±0.21 (✓)	99.09±0.09 (✓)
	LA	**0.213**±0.001	0.254±0.001	0.334±0.001	89.66±0.29 (✓)	93.88±0.23 (✗)	98.09±0.10 (✗)
	SCP	0.223±0.001	0.275±0.001	0.394±0.003	90.10±0.27 (✓)	95.35±0.20 (✓)	99.15±0.09 (✓)
`kin8nm`	**CRF**	0.221±0.001	0.271±0.001	0.381±0.003	90.15±0.27 (✓)	95.36±0.19 (✓)	99.12±0.09 (✓)
N=8,192	**CQR**	0.241±0.001	0.286±0.001	0.392±0.003	90.41±0.24 (✓)	95.18±0.20 (✓)	98.98±0.09 (✓)
I=8	**ACP-GN**	**0.213**±0.001	**0.262**±0.001	**0.365**±0.002	89.68±0.28 (✓)	94.58±0.20 (✓)	98.85±0.08 (✓)
	SCP-GN	0.222±0.001	0.273±0.001	0.386±0.003	90.05±0.27 (✓)	95.36±0.20 (✓)	99.12±0.08 (✓)
	ACP-GN (split + refine)	0.243±0.001	0.298±0.001	0.423±0.004	90.60±0.27 (✓)	95.27±0.16 (✓)	99.17±0.09 (✓)
	LA	13.345±0.028	15.901±0.034	**20.898**±0.045	92.08±0.23 (✗)	95.86±0.17 (✗)	98.86±0.09 (✓)
	SCP	12.620±0.051	15.212±0.069	21.720±0.245	90.29±0.22 (✓)	95.01±0.19 (✓)	98.99±0.10 (✓)
`power`	**CRF**	**12.457**±0.049	**15.019**±0.069	21.553±0.189	90.32±0.24 (✓)	95.06±0.17 (✓)	98.96±0.10 (✓)
N=9,568	**CQR**	12.818±0.041	**15.081**±0.058	21.903±0.170	90.52±0.22 (✓)	94.90±0.18 (✓)	99.02±0.08 (✓)
I=4	**ACP-GN**	**12.526**±0.024	15.248±0.020	21.592±0.062	90.16±0.26 (✓)	95.13±0.19 (✓)	98.89±0.08 (✓)
	SCP-GN	12.627±0.050	15.215±0.070	21.724±0.242	90.32±0.22 (✓)	95.01±0.18 (✓)	99.00±0.09 (✓)
	ACP-GN (split + refine)	12.834±0.054	15.561±0.067	22.548±0.255	90.18±0.26 (✓)	95.18±0.19 (✓)	99.05±0.10 (✓)
	LA	0.548±0.074	0.653±0.088	0.858±0.116	90.90±0.59 (✓)	93.83±0.50 (✓)	97.05±0.33 (✗)
	SCP	0.471±0.009	0.642±0.013	1.033±0.027	90.12±0.55 (✓)	95.20±0.46 (✓)	98.97±0.20 (✓)
`community`	**CRF**	**0.464**±0.010	**0.617**±0.015	0.992±0.029	90.35±0.53 (✓)	95.03±0.45 (✓)	98.90±0.18 (✓)
N=1,994	**CQR**	0.659±0.034	0.926±0.058	1.349±0.067	91.28±0.49 (✓)	96.00±0.37 (✓)	99.65±0.18 (✓)
I=100	**ACP-GN**	**0.460**±0.002	**0.593**±0.005	**0.932**±0.006	90.45±0.46 (✓)	95.05±0.42 (✓)	99.21±0.14 (✓)
	SCP-GN	**0.448**±0.008	**0.621**±0.013	1.069±0.048	90.10±0.57 (✓)	95.35±0.42 (✓)	99.03±0.19 (✓)
	ACP-GN (split + refine)	0.561±0.009	0.700±0.010	1.092±0.029	89.97±0.69 (✓)	94.92±0.45 (✓)	98.85±0.20 (✓)
	LA	67.580±2.637	80.527±3.142	105.831±4.130	97.63±0.09 (✗)	98.15±0.08 (✗)	98.76±0.06 (✗)
	SCP	18.756±0.625	42.110±1.381	176.332±5.531	89.98±0.12 (✓)	95.08±0.09 (✓)	99.03±0.04 (✓)
`facebook_1`	**CRF**	**16.845**±0.513	34.078±0.936	114.769±4.111	90.04±0.13 (✓)	95.12±0.09 (✓)	98.98±0.05 (✓)
N=40,948	**CQR**	21.120±1.049	**27.024**±1.543	**43.095**±2.232	90.05±0.14 (✓)	95.04±0.08 (✓)	98.95±0.03 (✓)
I=53	**ACP-GN**	17.986±0.480	41.063±0.770	199.331±10.821	90.36±0.16 (✓)	95.56±0.17 (✗)	99.37±0.08 (✗)
	SCP-GN	18.696±0.612	39.460±1.552	130.967±6.779	90.04±0.13 (✓)	95.04±0.10 (✓)	99.03±0.04 (✓)
	ACP-GN (split + refine)	24.639±1.195	51.864±2.687	181.585±11.155	90.27±0.13 (✓)	95.29±0.09 (✓)	99.14±0.04 (✓)

Table 11: KFAC approximation to the Gauss-Newton evaluated on large UCI datasets

		Avg. Width			Avg. Coverage		
		90%	95%	99%	90%	95%	99%
`bike` N=10,886 I=18	**LA**	100.451±2.394	119.694±2.853	157.305±3.749	89.82±0.39 (✓)	93.29±0.33 (✗)	96.83±0.16 (✗)
	LA*(kfac)*	91.501±3.058	109.030±3.644	143.290±4.789	87.24±1.01 (✗)	91.07±0.84 (✗)	95.20±0.55 (✗)
	SCP	131.138±0.812	180.477±1.244	324.756±4.635	90.33±0.21 (✓)	95.17±0.15 (✓)	99.00±0.07 (✓)
	SCP-GN	122.245±1.073	160.505±1.761	254.409±3.767	90.34±0.24 (✓)	95.26±0.15 (✓)	99.02±0.08 (✓)
	SCP-GN*(kfac)*	121.401±0.708	160.214±1.328	259.909±3.247	90.46±0.21 (✓)	95.22±0.18 (✓)	98.99±0.07 (✓)
	ACP-GN	102.401±6.037	133.781±7.394	212.144±11.677	89.27±0.35 (✗)	94.45±0.24 (✗)	98.69±0.10 (✗)
	ACP-GN*(kfac)*	94.964±2.095	124.501±3.011	198.018±5.012	88.67±0.68 (✗)	94.06±0.52 (✗)	98.49±0.18 (✗)
	ACP-GN (split + refine)	125.090±4.180	162.697±5.654	254.893±9.363	90.20±0.22 (✓)	94.95±0.13 (✓)	99.01±0.08 (✓)
	ACP-GN*(kfac)* (split + refine)	162.693±2.312	211.953±2.906	334.975±5.205	91.31±0.19 (✗)	95.81±0.11 (✗)	99.24±0.05 (✓)
`community` N=1,994 I=100	**LA**	0.548±0.074	0.653±0.088	0.858±0.116	90.90±0.59 (✓)	93.83±0.50 (✓)	97.05±0.33 (✗)
	LA*(kfac)*	0.472±0.003	0.562±0.004	0.739±0.005	90.45±0.42 (✓)	93.55±0.41 (✗)	96.95±0.30 (✗)
	SCP	0.534±0.010	0.735±0.020	1.164±0.029	90.17±0.41 (✓)	95.33±0.22 (✓)	99.17±0.17 (✓)
	SCP-GN	0.473±0.013	0.660±0.024	1.116±0.034	90.55±0.39 (✓)	95.10±0.25 (✓)	99.12±0.14 (✓)
	SCP-GN*(kfac)*	0.476±0.012	0.657±0.023	1.095±0.030	90.62±0.41 (✓)	95.30±0.23 (✓)	98.90±0.18 (✓)
	ACP-GN	0.611±0.151	0.784±0.191	1.222±0.289	90.92±0.62 (✓)	95.28±0.45 (✓)	99.25±0.13 (✓)
	ACP-GN*(kfac)*	0.476±0.018	0.612±0.023	0.964±0.035	90.78±0.48 (✓)	95.20±0.37 (✓)	99.25±0.12 (✓)
	ACP-GN (split + refine)	0.523±0.005	0.661±0.007	1.040±0.016	91.38±0.55 (✓)	95.45±0.44 (✓)	99.20±0.14 (✓)
	ACP-GN*(kfac)* (split + refine)	0.530±0.005	0.674±0.008	1.067±0.019	91.20±0.64 (✓)	95.62±0.44 (✓)	99.35±0.12 (✓)
`protein` N=45,730 I=9	**LA**	9.385±0.022	11.183±0.027	14.697±0.035	85.43±0.18 (✗)	89.69±0.15 (✗)	94.81±0.10 (✗)
	LA*(kfac)*	9.132±0.027	10.881±0.032	14.301±0.042	84.33±0.18 (✗)	88.63±0.16 (✗)	94.04±0.12 (✗)
	SCP	13.041±0.088	17.161±0.098	26.181±0.119	89.78±0.08 (✓)	94.83±0.06 (✓)	98.94±0.04 (✓)
	SCP-GN	12.426±0.085	16.102±0.096	24.032±0.138	89.78±0.10 (✓)	94.86±0.08 (✓)	98.94±0.03 (✓)
	SCP-GN*(kfac)*	12.679±0.084	16.570±0.096	24.981±0.133	89.76±0.10 (✓)	94.85±0.08 (✓)	98.94±0.03 (✓)
	ACP-GN	10.127±0.020	13.156±0.027	19.943±0.061	87.62±0.15 (✗)	93.14±0.10 (✗)	98.35±0.04 (✗)
	ACP-GN*(kfac)*	9.695±0.034	12.595±0.041	19.093±0.064	86.02±0.18 (✗)	91.92±0.15 (✗)	97.77±0.06 (✗)
	ACP-GN (split + refine)	12.614±0.034	15.961±0.044	23.188±0.069	89.88±0.10 (✓)	94.90±0.09 (✓)	98.97±0.05 (✓)
	ACP-GN*(kfac)* (split + refine)	12.129±0.105	15.363±0.128	22.260±0.152	88.39±0.14 (✗)	93.82±0.13 (✗)	98.49±0.07 (✗)
`facebook_1` N=40,948 I=53	**LA**	67.580±2.637	80.527±3.142	105.831±4.130	97.63±0.09 (✗)	98.15±0.08 (✗)	98.76±0.06 (✗)
	LA*(kfac)*	67.534±3.806	80.471±4.535	105.757±5.960	97.39±0.15 (✗)	97.88±0.12 (✗)	98.57±0.09 (✗)
	SCP	20.771±2.452	44.192±2.799	178.890±4.596	90.00±0.10 (✓)	95.03±0.08 (✓)	99.05±0.04 (✓)
	SCP-GN	20.460±2.463	39.712±3.148	125.078±8.075	90.01±0.10 (✓)	94.94±0.09 (✓)	99.04±0.03 (✓)
	SCP-GN*(kfac)*	20.303±2.476	40.531±2.973	128.583±5.746	90.01±0.11 (✓)	95.04±0.09 (✓)	99.03±0.03 (✓)
	ACP-GN	30.466±7.858	62.895±13.942	223.609±45.677	90.56±0.19 (✗)	95.61±0.16 (✗)	99.26±0.05 (✗)
	ACP-GN*(kfac)*	19.275±0.542	40.991±0.964	149.319±5.539	90.24±0.12 (✓)	95.38±0.09 (✗)	99.24±0.06 (✗)
	ACP-GN (split + refine)	33.780±5.321	57.292±5.730	141.074±6.581	90.13±0.11 (✓)	95.23±0.06 (✓)	99.08±0.04 (✓)
	ACP-GN*(kfac)* (split + refine)	23.732±1.180	45.239±2.220	121.631±3.699	89.81±0.10 (✓)	94.86±0.08 (✓)	98.90±0.04 (✓)
`facebook_2` N=81,311 I=53	**LA**	66.088±2.760	78.749±3.289	103.493±4.322	97.47±0.12 (✗)	98.01±0.09 (✗)	98.65±0.06 (✗)
	LA*(kfac)*	63.591±3.085	75.774±3.675	99.584±4.830	97.26±0.15 (✗)	97.84±0.12 (✗)	98.54±0.08 (✗)
	SCP	16.387±0.208	35.387±0.462	152.706±1.591	89.97±0.07 (✓)	95.00±0.06 (✓)	99.06±0.03 (✓)
	SCP-GN	16.287±0.202	33.655±0.563	118.489±4.305	89.99±0.07 (✓)	94.98±0.06 (✓)	99.00±0.03 (✓)
	SCP-GN*(kfac)*	16.213±0.195	33.686±0.530	116.672±2.618	89.96±0.08 (✓)	94.98±0.06 (✓)	99.02±0.02 (✓)
	ACP-GN	26.536±3.513	56.470±7.754	197.251±27.935	90.54±0.11 (✗)	95.46±0.08 (✗)	99.23±0.04 (✗)
	ACP-GN*(kfac)*	20.497±1.859	43.246±3.955	147.832±13.223	90.34±0.08 (✗)	95.22±0.05 (✗)	99.18±0.03 (✗)
	ACP-GN (split + refine)	23.095±1.120	42.653±1.161	123.337±2.954	90.17±0.08 (✓)	95.15±0.05 (✓)	99.08±0.03 (✓)
	ACP-GN*(kfac)* (split + refine)	20.487±0.257	39.636±0.582	117.994±2.407	90.11±0.09 (✓)	95.15±0.06 (✓)	99.11±0.03 (✓)

Table 12: Last-layer (LL) approximation to the Gauss-Newton evaluated on large UCI datasets

		Avg. Width			Avg. Coverage		
		90%	95%	99%	90%	95%	99%
`bike` N=10,886 I=18	**LA**	100.451±2.394	119.694±2.853	157.305±3.749	89.82±0.39 (✓)	93.29±0.33 (✗)	96.83±0.16 (✗)
	LA*(LL)*	79.923±3.087	95.234±3.679	125.159±4.835	82.80±1.32 (✗)	86.98±1.21 (✗)	92.19±0.95 (✗)
	SCP	131.138±0.812	180.477±1.244	324.756±4.635	90.33±0.21 (✓)	95.17±0.15 (✓)	99.00±0.07 (✓)
	SCP-GN	122.245±1.073	160.505±1.761	254.409±3.767	90.34±0.24 (✓)	95.26±0.15 (✓)	99.02±0.08 (✓)
	SCP-GN*(LL)*	130.965±0.796	180.227±1.213	324.427±4.404	90.26±0.22 (✓)	95.17±0.15 (✓)	99.00±0.07 (✓)
	ACP-GN	98.813±2.485	130.893±3.231	213.131±5.630	89.36±0.43 (✓)	94.41±0.27 (✗)	98.67±0.09 (✗)
	ACP-GN*(LL)*	76.808±2.933	100.920±4.078	160.448±6.648	81.69±1.31 (✗)	88.20±1.23 (✗)	95.17±0.78 (✗)
	ACP-GN (split + refine)	128.336±4.336	170.782±5.859	281.632±10.176	89.98±0.22 (✓)	94.94±0.16 (✓)	99.01±0.06 (✓)
	ACP-GN*(LL)* (split + refine)	128.853±1.101	178.079±1.464	325.171±3.916	90.12±0.23 (✓)	95.12±0.20 (✓)	99.02±0.07 (✓)
`community` N=1,994 I=100	**LA**	0.548±0.074	0.653±0.088	0.858±0.116	90.90±0.59 (✓)	93.83±0.50 (✓)	97.05±0.33 (✗)
	LA*(LL)*	0.455±0.013	0.542±0.015	0.712±0.020	89.28±1.00 (✓)	92.33±0.94 (✗)	96.05±0.79 (✗)
	SCP	0.534±0.010	0.735±0.020	1.164±0.029	90.17±0.41 (✓)	95.33±0.22 (✓)	99.17±0.17 (✓)
	SCP-GN	0.473±0.013	0.660±0.024	1.116±0.034	90.55±0.39 (✓)	95.10±0.25 (✓)	99.12±0.14 (✓)
	SCP-GN*(LL)*	0.533±0.010	0.730±0.020	1.157±0.028	90.33±0.42 (✓)	95.50±0.24 (✓)	99.15±0.16 (✓)
	ACP-GN	0.570±0.108	0.755±0.158	1.224±0.285	90.90±0.62 (✓)	95.30±0.47 (✓)	99.25±0.14 (✓)
	ACP-GN*(LL)*	0.438±0.012	0.564±0.016	0.887±0.026	88.83±1.06 (✓)	93.78±0.99 (✗)	98.42±0.65 (✗)
	ACP-GN (split + refine)	0.519±0.006	0.652±0.008	1.010±0.016	90.97±0.61 (✓)	95.28±0.41 (✓)	99.12±0.16 (✓)
	ACP-GN*(LL)* (split + refine)	0.495±0.005	0.648±0.006	1.040±0.014	90.75±0.40 (✓)	95.70±0.29 (✓)	99.35±0.12 (✓)
`protein` N=45,730 I=9	**LA**	9.385±0.022	11.183±0.027	14.697±0.035	85.43±0.18 (✗)	89.69±0.15 (✗)	94.81±0.10 (✗)
	LA*(LL)*	8.667±0.026	10.328±0.031	13.573±0.041	82.51±0.17 (✗)	87.08±0.17 (✗)	92.81±0.12 (✗)
	SCP	13.041±0.088	17.161±0.098	26.181±0.119	89.78±0.08 (✓)	94.83±0.06 (✓)	98.94±0.04 (✓)
	SCP-GN	12.426±0.085	16.102±0.096	24.032±0.138	89.78±0.10 (✓)	94.86±0.08 (✓)	98.94±0.03 (✓)
	SCP-GN*(LL)*	13.035±0.088	17.150±0.098	26.167±0.119	89.77±0.08 (✓)	94.82±0.06 (✓)	98.94±0.04 (✓)
	ACP-GN	10.243±0.019	13.294±0.027	20.101±0.053	87.54±0.15 (✗)	93.04±0.11 (✗)	98.24±0.05 (✗)
	ACP-GN*(LL)*	8.731±0.032	11.341±0.038	17.192±0.060	82.62±0.18 (✗)	89.16±0.15 (✗)	96.29±0.08 (✗)
	ACP-GN (split + refine)	12.660±0.028	16.073±0.031	23.445±0.057	89.83±0.09 (✓)	94.90±0.09 (✓)	98.97±0.05 (✓)
	ACP-GN*(LL)* (split + refine)	12.616±0.029	16.078±0.036	23.672±0.075	89.81±0.11 (✓)	94.77±0.09 (✓)	98.98±0.03 (✓)
`facebook_1` N=40,948 I=53	**LA**	67.580±2.637	80.527±3.142	105.831±4.130	97.63±0.09 (✗)	98.15±0.08 (✗)	98.76±0.06 (✗)
	LA*(LL)*	66.025±4.620	78.673±5.506	103.394±7.235	96.90±0.19 (✗)	97.45±0.17 (✗)	98.10±0.14 (✗)
	SCP	20.771±2.452	44.192±2.799	178.890±4.596	90.00±0.10 (✓)	95.03±0.08 (✓)	99.05±0.04 (✓)
	SCP-GN	20.460±2.463	39.712±3.148	125.078±8.075	90.01±0.10 (✓)	94.94±0.09 (✓)	99.04±0.03 (✓)
	SCP-GN*(LL)*	20.735±2.454	44.237±2.780	177.595±4.404	90.02±0.11 (✓)	95.03±0.08 (✓)	99.04±0.03 (✓)
	ACP-GN	17.986±0.480	41.063±0.770	199.331±10.821	90.36±0.16 (✓)	95.56±0.17 (✗)	99.37±0.08 (✗)
	ACP-GN*(LL)*	17.305±0.514	36.951±1.220	134.876±6.983	89.54±0.12 (✗)	94.53±0.09 (✗)	98.63±0.09 (✗)
	ACP-GN (split + refine)	29.006±4.472	54.099±4.576	172.252±6.758	90.06±0.10 (✓)	95.20±0.08 (✓)	99.14±0.04 (✓)
	ACP-GN*(LL)* (split + refine)	19.198±0.585	42.501±0.841	175.527±2.964	90.02±0.11 (✓)	95.12±0.08 (✓)	99.06±0.03 (✓)
`facebook_2` N=81,311 I=53	**LA**	66.088±2.760	78.749±3.289	103.493±4.322	97.47±0.12 (✗)	98.01±0.09 (✗)	98.65±0.06 (✗)
	LA*(LL)*	64.783±4.014	77.194±4.783	101.450±6.286	97.02±0.15 (✗)	97.56±0.13 (✗)	98.23±0.09 (✗)
	SCP	16.387±0.208	35.387±0.462	152.706±1.591	89.97±0.07 (✓)	95.00±0.06 (✓)	99.06±0.03 (✓)
	SCP-GN	16.287±0.202	33.655±0.563	118.489±4.305	89.99±0.07 (✓)	94.98±0.06 (✓)	99.00±0.03 (✓)
	SCP-GN*(LL)*	16.323±0.199	35.192±0.446	151.199±1.862	90.00±0.07 (✓)	94.99±0.06 (✓)	99.05±0.03 (✓)
	ACP-GN	18.396±0.546	40.088±1.091	166.792±5.632	90.47±0.11 (✗)	95.45±0.08 (✗)	99.35±0.04 (✗)
	ACP-GN*(LL)*	17.671±0.815	37.040±1.617	127.615±5.783	89.94±0.09 (✓)	94.74±0.06 (✗)	98.73±0.05 (✗)
	ACP-GN (split + refine)	21.469±0.906	42.184±0.788	152.460±2.499	90.14±0.08 (✓)	95.10±0.06 (✓)	99.13±0.02 (✗)
	ACP-GN*(LL)* (split + refine)	16.895±0.238	36.727±0.477	149.723±2.197	89.98±0.07 (✓)	95.03±0.06 (✓)	99.03±0.02 (✓)