PREDICTIVE INFERENCE IS REALLY FREE WITH IN-CONTEXT LEARNING

Sohom Mukherjee*, **Ivane Antonov***, **Kai Günder***, **Magnus Josef Maichle**
Julius-Maximilians-Universität Würzburg
{sohom.mukherjee, ivane.antonov}@uni-wuerzburg.de
{kai.guender, magnus.maichle}@uni-wuerzburg.de

ABSTRACT

In this work, we consider the problem of constructing prediction intervals (PIs) for point predictions that are obtained using transformers. We propose a novel method for constructing PIs called in-context Jackknife+ (ICJ+), by using a meta-learned transformer trained via in-context learning (ICL) to perform training-free leave-one-out (LOO) predictions, i.e., by only prompting the transformer with LOO datasets and no retraining. We provide distribution-free coverage guarantees for our proposed ICJ+ algorithm under mild assumptions, by leveraging the stability of in-context trained transformers. We evaluate the coverage and width of the intervals obtained using ICJ+ on synthetic i.i.d. data for five classes of functions, and observe that their performance is comparable or superior to the benchmark J+ and true confidence intervals.

1 INTRODUCTION

Predicting a quantity of interest based on historical data of responses and covariates is necessary in numerous applications including the energy sector (Haben et al., 2023) and supply chain management (Syntetos et al., 2016). Traditionally, data-driven methods have addressed these problems using either point predictors or probabilistic predictors at their core. Recently, there has been growing interest in using modern transformer-based foundation models to meta-learn parametric and non-parametric estimators, by leveraging their in-context learning (ICL) (Hollmann et al., 2025; Maichle et al., 2024) capabilities. Garg et al. (2022) studies in-context learning in transformers for simple function classes such as linear regression and ReLU 2-layer neural networks, and Li et al. (2023) provides formal generalization bounds for the same setting using the stability of the algorithm meta-learned by the transformer. However, these methods produce only a point prediction with no uncertainty estimates.

This necessitates the construction of valid prediction intervals around these predictions. Such uncertainty representation becomes especially important for predictions (involved in high-stakes decisions) obtained using transformers that are known to hallucinate (Abbasi-Yadkori et al., 2024b). Very recently, there has been a growing interest in uncertainty quantification for transformers, but they mainly focus on text data with a finite vocabulary (Abbasi-Yadkori et al., 2024a). This motivates us to come up with new algorithms and theories for representing uncertainty in transformers that work with continuous numerical data. To do this, we leverage the key idea that a transformer trained in an in-context manner represents a meta-learned algorithm that can give different predictors based on different prompts. This enables us to obtain LOO predictors by only prompting a transformer using appropriate data without any retraining, and use these LOO predictions for constructing PIs using the J+ (Barber et al., 2021) method. The proposed in-context Jackknife+ (ICJ+) algorithm has the two-fold advantage of being faster (by avoiding retraining) as well as obtaining good quality PIs (due to enhanced accuracy of the underlying point predictor obtained using the ICL transformer).

Contributions. We make the following main contributions:

*Equal Contribution

1. We propose in-context Jackknife+ (ICJ+), a modified version of the Jacknife+ (J+) framework that leverages the meta-learning capability of in-context trained transformers for constructing prediction intervals around point predictions without any model retraining.

2. While the idea to prompt a transformer in a LOO manner is pretty simple and brings obvious benefits when used for predictive inference, the theoretical validity of such a method is not obvious. We build upon the idea of algorithmic stability—studied in both the predictive inference and transformer generalization literature—to come up with finite sample as well as asymptotic coverage guarantees for our proposed ICJ+ algorithm under mild assumptions.

3. We benchmark the performance of our proposed algorithm against the original J+ method with model retrainings as well as the true confidence intervals, based on synthetic i.i.d. data generated for various function classes. We observe that our method achieves superior coverage while maintaining reasonable interval width, and reducing the computational burden by many folds compared to J+.

2 Problem Setup

Suppose we are given training data $\mathcal{S}_n := \{(\boldsymbol{X}_i, Y_i)\}_{i=1}^n$, consisting of i.i.d. realizations of a random response $Y \in \mathcal{Y} \subseteq \mathbb{R}$ and associated contextual feature information $\boldsymbol{X} \in \mathcal{X} \subseteq \mathbb{R}^p$. Our goal is to predict the value of the response Y_{n+1} for a new data point given feature information $\boldsymbol{X}_{n+1}$, which is independently drawn from the same distribution as the training data. To achieve this, we aim to use a transformer model TF that takes as input a prompt consisting of the training data $\mathcal{S}_n$ and the test point $\boldsymbol{X}_{n+1}$. However, this raises the crucial question of how to quantify the uncertainty in our prediction. Specifically, how can we construct a data-dependent prediction interval $\hat{C}_\alpha(\boldsymbol{X}_{n+1}) \subset \mathcal{Y}$ around the transformer's point forecast, ensuring that it contains Y_{n+1} with a given probability of $1-\alpha$, without relying on assumptions about the underlying unknown distribution? To be specific, our goal is to construct $\hat{C}_\alpha(\boldsymbol{X}_{n+1})$ so that it satisfies either the marginal coverage guarantee

$$\mathbb{P}\left(Y_{n+1} \in \hat{C}_\alpha(\boldsymbol{X}_{n+1})\right) \geq 1-\alpha\,, \tag{1}$$

where the expectation is with respect to $\{(\boldsymbol{X}_i, Y_i)\}_{i=1}^{n+1}$, or the stronger conditional coverage guarantee

$$\mathbb{P}\left(Y_{n+1} \in \hat{C}_\alpha(\boldsymbol{X}_{n+1}) \,\middle|\, \mathcal{S}_n\right) \geq 1-\alpha \tag{2}$$

where the expectation is only taken with respect to $(\boldsymbol{X}_{n+1}, Y_{n+1})$ given $\mathcal{S}_n$. The conditional coverage $\mathbb{P}\left(Y_{n+1} \in \hat{C}_\alpha(\boldsymbol{X}_{n+1}) \,\middle|\, \mathcal{S}_n\right)$ is itself a random variable dependent on $\mathcal{S}_n$ and is our main object of interest in this paper. As we shall see in Section 3.2.1, obtaining (2) is usually only possible provided that the underlying algorithm is stable (i.e. is insensitive to small changes in the training data), which transformers in fact are.

In the field of distribution-free predictive inference, Conformalized Prediction (CP), refers to the construction of PIs using residuals of the form $R_i = |\hat{y}_i - y_i|$, where $\hat{y}_i$ is the point prediction of the conditional expectation of Y given $\boldsymbol{X} = \boldsymbol{x}_i$ obtained using some base algorithm. The class of CP methods that offer the best performance are called full conformal methods, in the sense that they do not need to split the data into an additional calibration set. One of the most popular full conformal methods is the Jackknife+ (J+), which uses leave-one-out (LOO) predictors for creating the residuals. However, the main limitation of this method is that it needs to be retrained in order of number of training examples. In this paper, we explore how PIs can be constructed if transformers are employed as the base algorithm that maps data to point predictors in the J+ method. We find that the ICL capability of modern transformers allows us to obtain LOO predictors and hence J+ PIs for free, i.e., without any retraining. In the following Section, we first introduce ICL for least squares regression and then describe how ICL can be leveraged for training-free predictive inference.

3 TRAINING-FREE PIS USING ICL

3.1 IN-CONTEXT LEARNING

ICL refers to the ability of a transformer TF, trained on *prompts* comprising input-output pairs for a given *task*, to generate an output for a new prompt corresponding to a different task, without requiring any fine-tuning (Garg et al., 2022). To illustrate this in our context, let $\mathcal{S}_m := \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$ represent a set of m feature-target pairs sampled from a common underlying data-generating process $(\boldsymbol{X}, Y)$. Using these example pairs, we define the prompt

$$\boldsymbol{x}_{\text{prompt}}^{(m+1)} := (\boldsymbol{x}_1, y_1, \ldots, \boldsymbol{x}_m, y_m, \boldsymbol{x}_{m+1}) \in \mathbb{R}^{(2m+1)\times d} \tag{3}$$

to make a prediction for the new input $\boldsymbol{x}_{m+1}$.[1] A transformer $\texttt{TF}(\cdot)$ is said to demonstrate ICL if it can generate a meaningful prediction $\texttt{TF}(\boldsymbol{x}_{\text{prompt}}^{(m+1)})$ for the response y_{m+1} given $\boldsymbol{x}_{m+1}$, solely based on the examples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$ provided in the prompt, without having encountered any examples from the same data-generating process $(\boldsymbol{X}, Y)$ during training. As such, TF conducts an implicit optimization on the sequence $\mathcal{S}_m$ to make a prediction for $\boldsymbol{x}_{m+1}$ (Li et al., 2023). A transformer TF can therefore be abstracted as a learning algorithm mapping a sequence $\mathcal{S}_m$ of arbitrary size m to a prediction function $f_{\mathcal{S}_m}^{\texttt{TF}}(\cdot) : \mathcal{X} \to \mathcal{Y}$ represented as:

$$\hat{f}_{\mathcal{S}_m}^{\texttt{TF}}(\boldsymbol{x}_{m+1}) := \texttt{TF}\left(\left(\mathcal{S}_m, \boldsymbol{x}_{m+1}\right)\right) := \texttt{TF}\left(\boldsymbol{x}_{\text{prompt}}^{m+1}\right), \tag{4}$$

where $\boldsymbol{x}_{\text{prompt}}^{m+1}$ is constructed as described earlier, using the m examples in $\mathcal{S}_m$ and $\boldsymbol{x}_{m+1}$. A learning algorithm $\texttt{Alg} \in \mathcal{A}$ denotes a function, that takes a training data set as input and returns a fitted regression function $\hat{f} : \mathcal{X} \to \mathcal{Y}$ (Liang & Barber, 2023). For a fixed architecture, each specific set of transformer weights defines a distinct learning algorithm. Thus, $\mathcal{A}$ represents the set of all possible transformer models within the given architecture.

To equip TF with the ICL ability, like Garg et al. (2022) or Li et al. (2023), we train the transformer TF using multi-task learning (MTL) over $t \in [T]$ tasks, each task characterized by a training sequence $\mathcal{S}_n^t := \{(\boldsymbol{x}_i^t, y_i^t)\}_{i=1}^n$ of n feature-target pairs sampled from a common data-generating process. The data-generating processes of all tasks are assumed to be mutually independent. Let $\mathcal{S}_m^t := \{(\boldsymbol{x}_i^t, y_i^t)\}_{i=1}^m$ denote a subsequence of $\mathcal{S}_m^t$ for $m \leq n$. Our ICL training objective is then given by:

$$\widehat{\texttt{TF}} \in \underset{\texttt{TF}\in\mathcal{A}}{\arg\min} \frac{1}{T}\sum_{t=1}^{T}\frac{1}{n}\sum_{i=0}^{n-1}\left(y_{i+1}^t - \hat{f}_{\mathcal{S}_i^t}^{\texttt{TF}}(\boldsymbol{x}_{i+1}^t)\right)^2$$

3.2 IN-CONTEXT JACKKNIFE+

We will now show how the meta-learning capability of in-context trained transformers can be leveraged to obtain PIs similar to (2) without any model retraining. One well-known distribution-free method to generate $\hat{C}_\alpha$ is the so-called Jackkinfe+ (J+) (Barber et al., 2021). Given a training dataset $\mathcal{S}_n = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, define the modified training dataset $S_n^{\setminus i} := ((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_{i-1}, y_{i-1}), (\boldsymbol{x}_{i+1}, y_{i+1}), \ldots, (\boldsymbol{x}_n, y_n))$ by removing point i. The J+ prediction interval (PI) for a test point $\boldsymbol{x}_{n+1} \in \mathcal{X}$ is then given by

$$\hat{C}_\alpha^{\texttt{J+}}(\boldsymbol{x}_{n+1}) = \left[Q_\alpha^-\left(\left\{\hat{f}_{\setminus i}(\boldsymbol{x}_{n+1}) - R_i\right\}_{i=1}^n\right), Q_\alpha^+\left(\left\{\hat{f}_{\setminus i}(\boldsymbol{x}_{n+1}) + R_i\right\}_{i=1}^n\right)\right], \tag{5}$$

where $\hat{f}_{\setminus i}$ is trained on $S_n^{\setminus i}$, $R_i := \left|y_i - \hat{f}_{\setminus i}(\boldsymbol{x}_i)\right|$ represents the $i-$th LOO residual, $Q_\alpha^+(\cdot)$ and $Q_\alpha^-(\cdot)$ represent the empirical upper and lower $\alpha-$quantiles[2].

While J+ provides robust and theoretically grounded prediction intervals, it is computationally expensive since the model must be retrained n times—once for each LOO subset. To address these

[1] Tokens are interpreted as row-vectors. TF outputs the last token as its prediction.

[2] For a set $\{v_i\}_{i=1}^n \subset \mathbb{R}$, we define $Q_\alpha^+(\{v_i\}_{i=1}^n)$ and $Q_\alpha^-(\{v_i\}_{i=1}^n)$ as the upper and lower-α quantiles, i.e., $Q_\alpha^+(\{v_i\}_{i=1}^n)$ is the $\lceil(1-\alpha)(n+1)\rceil-$th smallest value of $\{v_i\}_{i=1}^n$ and $Q_\alpha^-(\{v_i\}_{i=1}^n) = -Q_\alpha^+(\{v_i\}_{i=1}^n)$.

computational challenges, we propose the In-context Jackknife+ (ICJ+) approach that leverages the ICL capability of transformers to generate LOO predictors without requiring retraining. As outlined in (4), given a fixed set of feature-target pairs in $\mathcal{S}_n^{\setminus i}$ as context, a transformer implicitly defines a LOO predictor $\hat{f}^{\texttt{TF}}_{\mathcal{S}_n^{\setminus i}}(\cdot) : \mathcal{X} \to \mathcal{Y}$, $\hat{f}^{\texttt{TF}}_{\mathcal{S}_n^{\setminus i}}(\boldsymbol{x}) = \texttt{TF}(\boldsymbol{x}^{\setminus i}_{\text{prompt}})$ where

$$\boldsymbol{x}^{\setminus i}_{\text{prompt}} := (\boldsymbol{x}_1, y_1, \ldots, \boldsymbol{x}_{i-1}, y_{i-1}, \boldsymbol{x}_{i+1}, y_{i+1}, \ldots, \boldsymbol{x}_n, y_n, \boldsymbol{x}) \,.$$

Instead of retraining a model on each set $\mathcal{S}_n^{\setminus i}$, we simply prompt the transformer TF with the feature-target pairs in $\mathcal{S}_n^{\setminus i}$ and the point to predict. Further details can be found in Algorithm 1. The ICJ+ approach requires prompting the transformer $2n$ times: n times to compute $\{R_i\}_{i=1}^n$ and another n times to compute $\{\hat{f}^{\texttt{TF}}_{\mathcal{S}_n^{\setminus i}}(\boldsymbol{x}_{n+1})\}_{i=1}^n$. Table 1 compares the computational costs of J+ and ICJ+ in terms of the number of model retrainings.

Algorithm 1 `In-context Jackknife+ (ICJ+)`**.** Training-free J+ using an in-context trained transformer for constructing PIs around point estimates according to Equations (4) and (5).

Input: A trained transformer TF, Dataset $\mathcal{S}_n := \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, a test point $\boldsymbol{x}_{n+1} \in \mathcal{X}$

Output: Prediction interval $\hat{C}_\alpha^{\texttt{ICJ+}}(\boldsymbol{x}_{n+1})$

1: **for** $i = 1, \ldots, n$ **do**

2: $\quad \boldsymbol{x}^{\setminus i,(m)}_{\text{prompt}} = (\boldsymbol{x}_1, y_1, \ldots, \boldsymbol{x}_{i-1}, y_{i-1}, \boldsymbol{x}_{i+1}, y_{i+1}, \ldots, \boldsymbol{x}_n, y_n, \boldsymbol{x}_m)$ for $m \in \{i, n+1\}$

3: $\quad$ Compute $\hat{f}^{\texttt{TF}}_{\mathcal{S}_n^{\setminus i}} = \texttt{TF}\left(\boldsymbol{x}^{\setminus i,(m)}_{\text{prompt}}\right)$ for $m \in \{i, n+1\}$

4: $\quad$ Compute $R_i = \left|y_i - \hat{f}^{\texttt{TF}}_{\mathcal{S}_n^{\setminus i}}(\boldsymbol{x}_i)\right|$

5: **end for**

Compute the ICJ+ prediction interval at $\boldsymbol{x}_{n+1}$ as

$$\hat{C}_\alpha^{\texttt{ICJ+}}(\boldsymbol{x}_{n+1}) = \left[Q_\alpha^-\left(\{\hat{f}^{\texttt{TF}}_{\mathcal{S}_n^{\setminus i}}(\boldsymbol{x}_{n+1}) - R_i\}_{i=1}^n\right), Q_\alpha^+\left(\{\hat{f}^{\texttt{TF}}_{\mathcal{S}_n^{\setminus i}}(\boldsymbol{x}_{n+1}) + R_i\}_{i=1}^n\right)\right]$$

3.2.1 Conditional Coverage for ICJ+

There is an intrinsic connection between the theory of distribution-free prediction intervals and algorithmic stability. (Ndiaye, 2022) shows that if the base algorithm is stable, one can obtain conformal PIs without model retraining and without data splitting. Moreover, in recent years, several studies have established finite-sample guarantees for prediction intervals derived using the J+ algorithm, relying solely on the assumption of some form of algorithmic stability of the underlying point predictor (Barber et al., 2021; Amann et al., 2023). In this paper, we consider the replace-one stability that was introduced in (Shalev-Shwartz et al., 2010) and studied in the context of PIs in (Liang & Barber, 2023).

Definition 1 (Replace-one Algorithmic Stability)**.** *Let $\mathcal{S}_n = \{(\boldsymbol{X}_i, Y_i)\}_{i=1}^n$ be a set of n samples and denote by $\mathcal{S}_n^i$ the modification of $\mathcal{S}_n$ where the i-th sample is replaced by $(\boldsymbol{X}_i', Y_i')$. A learning algorithm `Alg` is β_n replace-one stable if*

$$\mathbb{E}\left(\left|\hat{f}_n(\boldsymbol{X}) - \hat{f}_n^i(\boldsymbol{X})\right|\right) \leq \beta_n \quad \textit{for all } i = 1, \ldots, n$$

where

$$\hat{f}_n := \texttt{Alg}(\mathcal{S}_n)\,, \hat{f}_n^i := \texttt{Alg}\left(\mathcal{S}_n^i\right)$$

and the expectation is taken with respect to all data points $((\boldsymbol{X}_i, Y_i))_{i=1}^n, (\boldsymbol{X}_i', Y_i'), \boldsymbol{X}$.

On the other hand, recent work on the generalization of transformers (Li et al., 2023) shows that the algorithm represented by an in-context trained transformer is replace-one stable according to Definition 1. We state this result in the following Proposition.

Proposition 1 (Replace-one Stability for Transformers)**.** *Under mild assumptions on the weight matrices as detailed in Theorem 4, a transformer `TF` with $R \in \mathbb{N}$ layers and an upper bound $\Gamma > 0$ on the operator norm of the combined key-query matrices across all self-attention heads, is replace-one stable according to Definition 2 with $\beta_n := \frac{2\left((1+\Gamma)e^\Gamma\right)^R}{2n-1}$, i.e.,*

$$\mathbb{E}\left(\left|\hat{f}^{TF}_{\mathcal{S}_n}(\boldsymbol{X}) - \hat{f}^{TF}_{\mathcal{S}_n^i}(\boldsymbol{X})\right|\right) \leq \frac{2\left((1+\Gamma)e^\Gamma\right)^R}{2n-1} \quad \textit{for all } i = 1, \ldots, n \tag{6}$$

Proposition 1 is an informal summary of the results presented in Theorem 4 in Appendix A.1.1. The result directly follows from recent work on transformer generalization theory (Li et al., 2023), and the key steps leading to (6) are outlined in A.1.1. The above result on the replace-one stability of transformers can be combined with the stability-based coverage guarantees provided in (Liang & Barber, 2023) to obtain the desired conditional coverage for our ICJ+ algorithm. To do that, we have to first define the symmetry of a learning algorithm, which is an assumption required for our finite sample coverage guarantee.

Definition 2 (Symmetry to Training Data). *For $n \in \mathbb{N}$, any set of training data $\mathcal{S}_n = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, and any permutation π on $\{1, \ldots, n\}$, a learning algorithm Alg is said to be symmetric if it holds that*

$$\texttt{Alg}((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)) = \texttt{Alg}((\boldsymbol{x}_{\pi(1)}, y_{\pi(1)}), \ldots, (\boldsymbol{x}_{\pi(n)}, y_{\pi(n)})),$$

i.e., Alg is invariant to the ordering of the input arguments.

In the context of transformers, symmetry implies that $\hat{f}^{\texttt{TF}}_{\mathcal{S}_n}(\boldsymbol{x}_{i+1}) = \hat{f}^{\texttt{TF}}_{\mathcal{S}_n^\pi}(\boldsymbol{x}_{i+1})$, where $\mathcal{S}_n^\pi$ is a permutation of the training data $\mathcal{S}_n$. While we cannot provide a formal proof of symmetry for our transformer TF, we do provide supporting empirical evidence in the Appendix A.2.5 (Figure 3)[3]. We are now ready to state our theoretical coverage guarantee for ICJ+, by assuming the symmetry of the transformer learning algorithm.

Theorem 1 (Training-conditional coverage for ICJ+). *Suppose $\mathcal{S}_n = \{(\boldsymbol{X}_i, Y_i)\}_{i=1}^n$ and $(\boldsymbol{X}_{n+1}, Y_{n+1})$ are i.i.d samples of $(\boldsymbol{X}, Y)$, and that the conditional distribution of $Y \mid \boldsymbol{X}$ has a density $h_{Y|\boldsymbol{X}}(y \mid \boldsymbol{x})$ with respect to the Lebesgue measure with*

$$B := \mathbb{E}\left[\sup_{y \in \mathbb{R}} h_{Y|\boldsymbol{X}}(y \mid \boldsymbol{X})\right] < \infty,$$

where the expected value is taken with respect to the marginal distribution of $\boldsymbol{X}$. Further assume that TF is symmetric as defined in Definition 2. Let us define the training-conditional miscoverage rate of the ICJ+ prediction interval as

$$\alpha^{ICJ+}(\mathcal{S}_n) = \mathbb{P}\left\{Y_{n+1} \notin \hat{C}_\alpha^{ICJ+}(\boldsymbol{X}_{n+1}) \mid \mathcal{S}_n\right\}.$$

Then for any $\delta \in (0, 1)$ it holds that

$$\mathbb{P}\left(\alpha^{ICJ+}(\mathcal{S}_n) < \alpha + 3\sqrt{\frac{\log(1/\delta)}{2}} + 4\sqrt[4]{2B\beta_n}\right) \geq 1 - 3\delta - \sqrt[3]{2B\beta_n}\,.$$

Proof. Once we have the stability result for transformers (Proposition 1), the statement of Theorem 1 follows from Theorem 4.4 of (Liang & Barber, 2023). The only difference is that their Theorem assumed replace-m stability, and we set $m = 1$ in their result for our case of replace-one stability. □

Note that to achieve our goal of ensuring $\alpha^{\texttt{ICJ+}}(\mathcal{S}_n)$ being close to α with high probability, β_n must be small. We observe that while β_n decreases as the number of in-context samples n grows, it increases exponentially with the number of transformer layers R. In general, the number of in-context samples has to be high to ensure training conditional coverage based on (1). Nevertheless, the stability of our transformer model ensures that training conditional coverage is at least given asymptotically:

Corollary 2 (Asymptotic results). *Assume the conditions of Theorem 1 hold. Then there exists sequences $(\varepsilon_n)_{n=1}^\infty, (\delta_n)_{n=1}^\infty \subset [0, \infty]$ with $\varepsilon_n, \delta_n \to 0$ such that*

$$\mathbb{P}\left(\alpha^{ICJ+}(\mathcal{S}_n) \leq \alpha + \varepsilon_n\right) \geq 1 - \delta_n$$

for all $n \geq 1$. As a direct consequence, we can additionally conclude $\alpha^{ICJ+}(\mathcal{S}_n) \xrightarrow{P} \alpha$ for $n \to \infty$.

[3]Moreover, it is reasonable in our case to assume symmetry to training data according to Definition 2 because the transformer model is solely trained on i.i.d. data, and not sequential data like in language models.

Table 1: Comparison of computational costs in terms of # of training runs.

PI Method	Base	ICL
J+	n (Barber et al., 2021)	1 (ICJ+, Alg. 1)
J+aB	B (Kim et al., 2020)	1 (ICJ+aB, Apx. A.3 Alg. 2)

A limitation of both Theorem 1 and Corollary 2 is the requirement that TF be symmetric with respect to the training data. However, by leveraging insights from Liang & Barber (2023) on the relationship between replace-stability and remove-stability, we can adopt the asymptotic coverage guarantees developed by Amann et al. (2023), which do not impose this symmetry requirement. Instead, we only need to assume that the prediction errors are bounded in probability—a mild condition when working with i.i.d. data.

Theorem 3 (Asymptotic conditional coverage for ICJ+ without symmetry)**.** *Suppose* $\mathcal{S}_n = \{(\boldsymbol{X}_i, Y_i)\}_{i=1}^n$ *and* $(\boldsymbol{X}_{n+1}, Y_{n+1})$ *are i.i.d samples of* $(\boldsymbol{X}, Y)$*, and assume that the conditional distribution of* $Y \mid \boldsymbol{X}$ *has a density* $h_{Y|\boldsymbol{X}}(y \mid \boldsymbol{x})$ *with respect to the Lebesgue measure with*

$$\mathbb{E}\left(\sup_{y\in\mathbb{R}} h_{Y|\boldsymbol{X}}(y \mid \boldsymbol{X})\right) < \infty,$$

where the expected value is taken with respect to the marginal distribution of $\boldsymbol{X}$*. Assume the prediction errors are bounded in probability, i.e., for every* $\varepsilon > 0$ *there exists an* $M > 0$ *such that*

$$\sup_{n\in\mathbb{N}} \mathbb{P}\left(\left|Y_{n+1} - \hat{f}^{TF}_{\mathcal{S}_n}(\boldsymbol{X})\right| \geq M\right) \leq \varepsilon,$$

where the expectation is taken with respect to $\{(\boldsymbol{X}_i, Y_i)\}_{i=1}^{n+1}$*. Then we have*

$$\lim_{n\to\infty} \mathbb{E}\left(\left|\alpha^{ICJ+}(\mathcal{S}_n) - \alpha\right|\right) = 0, \tag{7}$$

where the expectation is taken with respect to $\mathcal{S}_n$*. (7) directly implies* $\alpha^{ICJ+}(\mathcal{S}_n) \xrightarrow{P} \alpha$ *for* $n \to \infty$*.*

The proof of Theorem 3 can be found in Appendix A.1.2.

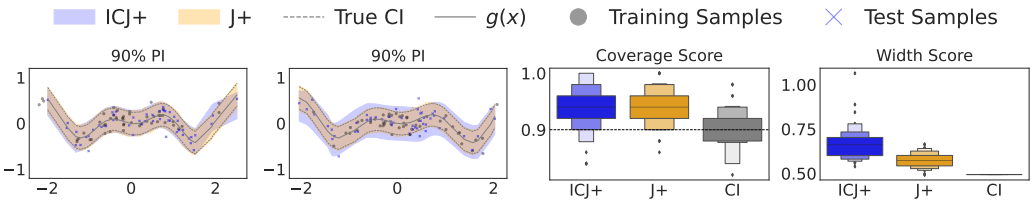
4 Experiments

In this Section, we demonstrate the predictive performance of our proposed ICJ+ algorithm and benchmark it against the J+ algorithm and the true uncertainty for synthetic i.i.d. data. We begin by explaining the data generation process and the transformer architecture along with its in-context training procedure. Following this, we explain the implementation of the predictive inference algorithms and compare their performances based on the metrics of coverage score and interval width score.

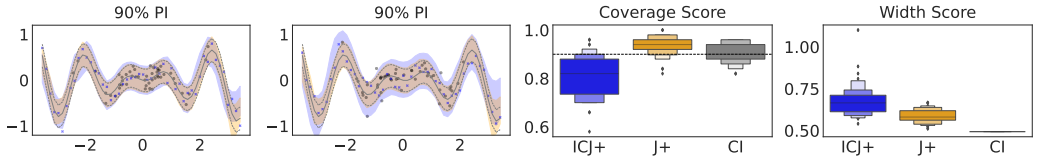
Dataset generation and function classes. We generate synthetic data for five function classes, where any function class, in general, is given by $\mathcal{H} = \{h \mid h(\boldsymbol{x}) = g(\boldsymbol{x}) + \xi;\ \xi \sim \mathcal{N}(0, \sigma^2)\}$, for features $\boldsymbol{x} \in \mathbb{R}^p$, response $y = h(\boldsymbol{x})$, and fixed noise $\sigma = 1.5$. We work with two feature dimensions $p \in \{1, 20\}$, and the features are sampled from the $\mathbb{R}^p$ dimensional standard normal distribution $\boldsymbol{x} \sim \mathcal{N}(0, \boldsymbol{I}_p)$. We study five function classes namely: linear (LR), quadratic (QR), cubic (CR), ReLU 2 layer neural network (NN), and a modified sinusoidal function (SR). For example, in LR, the parameterized noise-less function is defined according to $g(\boldsymbol{x}) = \boldsymbol{\beta}^\top \boldsymbol{x}$, the parameter being sampled from $\boldsymbol{\beta} \sim \mathcal{N}(0, \boldsymbol{I}_p)$. The definitions of the remaining function classes are relegated to Appendix A.2.1 for brevity.

Transformer architecture. Similar to Garg et al. (2022), we use transformer architectures from the GPT-2 family (Radford, 2018), in particular the small (for $p = 1$) and standard (for $p = 20$) architectures, which have been detailed in Appendix A.2.2, Table 4.

In-context transformer training. We train a transformer model per function class as a meta-learner, following the setting of Garg et al. (2022). For this, we first sample a random function h from the function class $\mathcal{H}$ we are training on. For example, for LR this would mean sampling $\boldsymbol{\beta} \sim \mathcal{N}(0, \boldsymbol{I}_p)$.



(a) Prediction intervals (PI) and metrics for two different feature samplings of 1D SR and in-sample evaluation.



(b) Prediction intervals (PI) and metrics for two different feature samplings of 1D SR and out-of-sample evaluation.

Figure 1: Comparison of PIs across different feature samplings of an instance of the one-dimensional SR function class ($\alpha = 0.1$). We observe that for in-sample evaluation ICJ+ reaches the target coverage, while for out-of-sample evaluation it performs slightly worse.

Then we sample inputs $\boldsymbol{x_i}$ from the multivariate standard normal distribution $\mathcal{N}(0, \boldsymbol{I}_p)$ and construct a training dataset as $\mathcal{S}_{m+1} = (\boldsymbol{x_1}, h(\boldsymbol{x_1}), \ldots, \boldsymbol{x}_{m+1}, h(\boldsymbol{x}_{m+1}))$. Given the training dataset $\mathcal{S}_{m+1}$, we obtain the transformer predictions $\left\{\hat{f}^{\mathrm{TF}}_{\mathcal{S}_i}(\boldsymbol{x}_{i+1})\right\}_{i=0}^{m}$ and compute the mean squared loss $\frac{1}{m+1}\sum_{i=0}^{m}\left(h(\boldsymbol{x}_{i+1}) - \hat{f}^{\mathrm{TF}}_{\mathcal{S}_i}(\boldsymbol{x}_{i+1})\right)^2$. At each training step, the loss is averaged over a batch of randomly generated prompts (with different functions from the chosen function class $h \in \mathcal{H}$), followed by an update step. A context length $m = 100$ is used across all function classes and models.

Construction and evaluation of PIs. While the base algorithm for ICJ+ is a meta-learned transformer that can produce predictors for any function belonging to the function class it is trained on, the base algorithm for the benchmark J+ is a standard machine learning algorithm such as MLP that is not capable of meta-learning across the function class. Therefore, for a fair comparison, we choose a particular instance of a function from a function class for comparison of PIs. For example, in LR this would mean choosing the function $g(\boldsymbol{x}) = \boldsymbol{\beta}_1^\top \boldsymbol{x}$, for some fixed $\boldsymbol{\beta}_1$.

We implement the J+ algorithm as described in (Barber et al., 2021) with a 2-layer neural network with ReLU activation (MLP) consisting of hidden dimension 512 as the base algorithm. Each LOO retraining is done for 1000 steps. The ICJ+ algorithm is implemented according to Algorithm 1. For any instance of a function class, we obtain 64 datasets, each with 150 points, where the different datasets represent different feature samplings of the same function instance to introduce statistical variability. Out of this, the first 100 points for each dataset are used for LOO retraining in J+ and equivalently LOO prompting in ICJ+, and the last 50 points are used for evaluation or testing. We also compare these two methods to the true $1-\alpha$ confidence interval (CI) according to the formula $CI_\alpha(\boldsymbol{x}_i) = g(\boldsymbol{x}_i) \pm Z_{\alpha/2} \times \sigma$, where $g(\boldsymbol{x}_i)$ is the response value without noise as defined above, $\sigma = 1.5$ is the standard deviation, and $Z_{\alpha/2}$ is the $Z-$score for the given confidence level. During the evaluation, we sample the features $\boldsymbol{x}$ for both the in-sample and out-of-sample settings as detailed below.

- *In-sample*. For in-sample evaluation, the features are sampled in the same way as training, that is, from the multivariate standard normal distribution, as we have described above $\boldsymbol{x} \sim \mathcal{N}(0, \boldsymbol{I}_p)$.
- *Out-of-sample.* For out-of-sample evaluation, the features are constructed on a grid spanning $[-3.5, 3.5]^p$ to simulate extrapolation. This grid is densely sampled near the origin (reflecting regions well-represented in training data) and increasingly sparse toward the extremes, deliberately exceeding the range where 99.9% of $\mathcal{N}(0, \boldsymbol{I}_p)$ samples lie: $\{x \in \mathbb{R}^p : \|x\|_2 \leq 3.29\}$.

Metrics and comparison. We use two metrics—regression coverage score and regression width score—to quantify the performance of our methods. Coverage Score (or empirical coverage proba-

Table 2: Average coverage (C) and interval width (W), with standard deviations, for J+, ICJ+, and the True 90% CI on 1D data, aggregated over 5 function samples in the same function class for both evaluation regimes.

Function	**Linear**		**Quadratic**		**Qubic**		**ReLu**		**Sinusoidal**	
Method	C	W	C	W	C	W	C	W	C	W
In Sample										
J+	0.94±0.04	0.57±0.05	0.94±0.04	0.57±0.05	0.94±0.04	0.57±0.05	0.94±0.04	0.58±0.05	0.94±0.04	0.58±0.05
ICJ+	0.94±0.04	0.57±0.05	0.94±0.04	0.57±0.05	0.94±0.04	0.59±0.05	0.94±0.04	0.59±0.06	0.94±0.04	0.73±0.11
True CI	0.90±0.04	0.49±0.00	0.90±0.04	0.49±0.00	0.90±0.04	0.49±0.00	0.90±0.04	0.49±0.00	0.90±0.04	0.49±0.00
Out Of Sample										
J+	0.92±0.04	0.58±0.05	0.93±0.04	0.57±0.05	0.93±0.04	0.59±0.05	0.94±0.04	0.58±0.05	0.90±0.04	0.75±0.05
ICJ+	0.93±0.05	0.57±0.05	0.91±0.06	0.57±0.05	0.88±0.08	0.59±0.05	0.94±0.04	0.59±0.05	0.80±0.07	0.77±0.11
True CI	0.90±0.04	0.49±0.00	0.90±0.04	0.49±0.00	0.90±0.04	0.49±0.00	0.90±0.04	0.49±0.00	0.90±0.04	0.49±0.00

bility) in the context of PIs for regression measures how well a model's predicted intervals capture the true values of the target variable. For a test set given by $\{(\boldsymbol{x}_i, y_i)\}_{i=n+1}^{n+N}$ and a predictive inference method that gives $[L_i, U_i]$ as the lower and upper bounds of the interval for test point $\boldsymbol{x}_i$, the coverage score is computed as $CP = \frac{1}{N}\mathbb{1}\{L_i \leq y_i \leq U_i\}$, where $\mathbb{1}\{\cdot\}$ is the indicator function. While we want to reach the desired coverage of $1-\alpha$, (i.e., $CP = 0.9$ for $\alpha = 0.1$ in our case) to ensure the validity of our PI, we also want to ensure that the intervals are not too wide. This is measured by the interval width score, which is computed as $IW = \frac{1}{N}\sum_{i=1}^{n}(U_i - L_i)$. The metrics are computed with $N = 100$ for our test set described above and across the 64 instances to obtain measures of statistical dispersion.

Results. In Figure 1 we use box plots to depict the coverage scores and the width scores across 64 different feature samplings of the same function instance from the one-dimensional modified sinusoidal (SR) class. For in-sample evaluation, we can observe that our proposed `ICJ+` as well as the benchmark `J+` always cover more points than the target on average. We also observe that this good coverage offered by `ICJ+` does not come at the cost of width, in fact, the width score of the `ICJ+` PI is at par with that of the benchmark. For out-of-sample evaluation, our `ICJ+` performs slightly worse that the benchmark on the SR function class. Table 2 and Figures 5, 6, 7, 8 (Appendix A.2.4) provide a more detailed comparison for all the function classes, where we observe similar trends except that `ICJ+` in fact performs at par with `J+` for most other function classes even out-of-sample. The results for 20-dimensional data are deferred to the Appendix A.2.3 in Figure 4 and Table 5. For higher dimensional data, we find that `ICJ+` outperforms `J+` by a large margin in terms of both coverage and width.

5 Conclusion

We introduced in-context Jackknife+ (`ICJ+`), a novel method for constructing prediction intervals (PIs) using transformers trained via in-context learning (ICL). By leveraging training-free leave-one-out (LOO) predictions, we are able to show that `J+` can be applied to transformer-based in-context learners. `ICJ+` eliminates the need for model retraining while maintaining distribution-free coverage guarantees under mild assumptions. Our theoretical analysis, grounded in algorithmic stability, establishes both finite-sample and asymptotic validity. Empirical evaluations on synthetic i.i.d. data across multiple function classes demonstrate that `ICJ+` achieves comparable coverage to Jackknife+ (`J+`) with retraining, while significantly reducing computational costs. Looking ahead, `ICJ+` could be extended to foundation models beyond transformers and adapted to permutation-invariant ICL methods for broader applicability. Addressing the exponential growth of the stability parameter β_n, remains a key challenge, where PAC-Bayes bounds and model compression techniques might possibly offer promising solutions (Lotfi et al., 2024). Our work paves the way for more efficient and theoretically grounded uncertainty quantification for transformer-based predictors. In Appendix A.3, we end with a discussion of how our proposed `ICJ+` can be extended to incorporate model (epistemic) uncertainty, i.e., the model's uncertainty in approximating the true conditional distribution, by using training-free bootstrapping (ref. `ICJ+aB` Algorithm 2). Future work would include benchmarking the computational efficiency of `ICJ+` using detailed timing comparisons, as well as experiments on real-world data possibly with temporal dependance.

ACKNOWLEDGMENTS

The authors would like to thank Richard Pibernik for extensive feedback on the manuscript at various stages of the work, as well as Nikolai Stein for help with experiment design and formatting. Sohom Mukherjee wishes to acknowledge the support of Michele Caprio, for discussions that helped shape the author's thoughts regarding uncertainty quantification in general, and epistemic uncertainty in particular.

REFERENCES

Yasin Abbasi-Yadkori, Ilja Kuzborskij, András György, and Csaba Szepesvari. To believe or not to believe your LLM: Iterativeprompting for estimating epistemic uncertainty. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a.

Yasin Abbasi-Yadkori, Ilja Kuzborskij, David Stutz, András György, Adam Fisch, Arnaud Doucet, Iuliya Beloshapka, Wei-Hung Weng, Yao-Yuan Yang, Csaba Szepesvári, et al. Mitigating llm hallucinations via conformal abstention. *arXiv preprint arXiv:2405.01563*, 2024b.

Nicolai Amann, Hannes Leeb, and Lukas Steinberger. Assumption-lean conditional predictive inference via the jackknife and the jackknife+, 2023. URL `https://arxiv.org/abs/2312.14596`.

Rina Barber, Emmanuel Candès, Aaditya Ramdas, and Ryan Tibshirani. Predictive inference with the jackknife+. *Annals of Statistics*, 49:486–507, 02 2021. doi: 10.1214/20-AOS1965.

Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems*, 35:30583–30598, 2022.

Stephen Haben, Marcus Voss, and William Holderbaum. *Core Concepts and Methods in Load Forecasting: With Applications in Distribution Networks*. Springer Nature, 2023.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.

Byol Kim, Chen Xu, and Rina Barber. Predictive inference is free with the jackknife+-after-bootstrap. *Advances in Neural Information Processing Systems*, 33:4138–4149, 2020.

Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Yingcong Li, Muhammed Emrullah Ildiz, Dimitris Papailiopoulos, and Samet Oymak. Transformers as algorithms: Generalization and stability in in-context learning. In *International Conference on Machine Learning*, pp. 19565–19594. PMLR, 2023.

Ruiting Liang and Rina Foygel Barber. Algorithmic stability implies training-conditional coverage for distribution-free prediction methods. *arXiv preprint arXiv:2311.04295*, 2023.

Sanae Lotfi, Yilun Kuang, Marc Anton Finzi, Brandon Amos, Micah Goldblum, and Andrew Gordon Wilson. Unlocking tokens as data points for generalization bounds on larger language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL `https://openreview.net/forum?id=5jRU8ufi8H`.

Magnus Josef Maichle, Sohom Mukherjee, Kai Günder, Ivane Antonov, Nikolai Stein, and Richard Pibernik. In-context quantile regression for multi-product inventory management using time-series transformers. In *NeurIPS Workshop on Time Series in the Age of Large Models*, 2024.

Eugene Ndiaye. Stable conformal prediction sets. In *International Conference on Machine Learning*, pp. 16462–16479. PMLR, 2022.

Alec Radford. Improving language understanding by generative pre-training. 2018.

Raphael Rossellini, Rina Foygel Barber, and Rebecca Willett. Integrating uncertainty awareness into conformalized quantile regression. In *International Conference on Artificial Intelligence and Statistics*, pp. 1540–1548. PMLR, 2024.

Yusuf Sale, Viktor Bengs, Michele Caprio, and Eyke Hüllermeier. Second-order uncertainty quantification: A distance-based approach. In *Forty-first International Conference on Machine Learning*, 2023.

Shai Shalev-Shwartz, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. Learnability, stability and uniform convergence. *The Journal of Machine Learning Research*, 11:2635–2670, 2010.

Aris A Syntetos, Zied Babai, John E Boylan, Stephan Kolassa, and Konstantinos Nikolopoulos. Supply chain forecasting: Theory, practice, their gap and the future. *European Journal of Operational Research*, 252(1):1–26, 2016.

A Appendix

A.1 Additional Theoretical Results

A.1.1 Transformer Stability

The replace-one stability of transformer models, as stated in Proposition 1, has been established in a uniform sense in Theorem 3.2 of Li et al. (2023). Theorem 4 presents a slightly modified version tailored to our requirements.

Before we can state the stability result for transformer models, we need to revisit the architecture of the transformer model `TF` under consideration. `TF` is an R-layer transformer model with an initial linear layer for embedding the input tokens into the embedding space $\mathbb{R}^{emb}$. The input tokens are then repeatedly transformed by a sequence of R subsequent self-attention and feed-forward sublayers with ReLU-activation. By $\boldsymbol{W}^{(r)}$ and $\boldsymbol{V}^{(r)}$ we denote the combined-key-query and value matrices of the r-th self-attention layer and by $\boldsymbol{M}_1^{(r)}, \boldsymbol{M}_2^{(r)}$ the weight matrices of the r-th feed-forward layer. `TF` outputs the last token of the final layer. Given a sequence of n feature-target samples $\mathcal{S}_n = \{(\boldsymbol{X}_i, Y_i)\}_{i=1}^n$, prompts are constructed as shown in (3).

Theorem 4 (Transformer Stability)**.** *Let `TF` be a R-layer transformer with the architecture specified above. Let $\mathcal{S}_n = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ be a sequence of n samples and denote by $\mathcal{S}_n^i$ the modification of $\mathcal{S}_n$ where the i-th sample is replaced by $(\boldsymbol{x}_i', y_i')$. Assume that $\|\boldsymbol{x}_i\|_2, \|y_i\|_2 \leq 1$ for all $i = 1, \ldots, n$. Further, assume that*

$$\begin{aligned} \|\boldsymbol{V}^{(r)}\| &\leq 1 \\ \|\boldsymbol{M}_1^{(r)}\|, \|\boldsymbol{M}_2^{(r)}\| &\leq 1 \\ \|\boldsymbol{W}^{(r)}\| &\leq \frac{\Gamma}{2} \end{aligned}$$

for all layers $r \in [R]$. Additionally, we require the matrix operator norm of the linear embedding layer to be smaller or equal to 1. Then we have

$$\left| \hat{f}_{\mathcal{S}_n}^{TF}(\boldsymbol{x}) - \hat{f}_{\mathcal{S}_n^i}^{TF}(\boldsymbol{x}) \right| \leq \frac{2\left((1+\Gamma)e^{\Gamma}\right)^R}{2n-1} \quad \forall x \in \mathcal{X} \tag{8}$$

and `TF` is replace-one-stable with $\beta_n := \frac{2\left((1+\Gamma)e^{\Gamma}\right)^R}{2n-1}$.

Proof. It is straightforward to verify that all the requirements of Theorem 3.2 in Li et al. (2023) are satisfied, which directly establishes (8). Taking the expectation of (8) with respect to $\{(\boldsymbol{X}_i, Y_i)\}_{i=1}^n$, $(\boldsymbol{X}_i', Y_i')$, and $\boldsymbol{X}$ immediately yields the claimed β_n replace-one stability. □

A.1.2 PROOFS OF ASYMPTOTIC COVERAGE IN THEOREM 3

To prove Theorem 3, we apply Corollary 5.6 of Amann et al. (2023). To adapt this result to our setting, we leverage insights of Liang & Barber (2023) on the relationship between replace-stability and remove-stability.

Proof of Theorem 3. As outlined above, our goal is to establish that the assumptions of Theorem 3 suffice to satisfy all conditions required for the application of Corollary 5.6 in Amann et al. (2023).

Continuous Case Assumption 1: It is easy to see, that the so-called Continuous Case Assumption 1 (**CC1** Assumption) as defined in (Amann et al., 2023) is satisfied. As the conditional distribution of $Y \mid \boldsymbol{X} = \boldsymbol{x}$ has a density for almost all $\boldsymbol{x} \in \mathcal{X}$, the absolute continuity of Y given $\boldsymbol{X} = \boldsymbol{x}$ immediately follows. Furthermore, as $\mathbb{E}\left(\sup_{y\in\mathbb{R}} h_{Y|\boldsymbol{X}}(y \mid \boldsymbol{X})\right) < \infty$, it follows that $\sup_{y\in\mathbb{R}} h_{Y|\boldsymbol{X}}(y \mid \boldsymbol{X})$ must be finite for almost all $\boldsymbol{x} \in \mathcal{X}$.

Boundedness in probability of density: Let $\varepsilon > 0$ and $M > 0$ be arbitrary. Set $B := \mathbb{E}\left(\sup_{y\in\mathbb{R}} h_{Y|\boldsymbol{X}}(y \mid \boldsymbol{X})\right)$. By applying Markov's inequality we get

$$\sup_{n\in\mathbb{N}} \mathbb{P}\left(\sup_{y\in\mathbb{R}} h_{Y_n|\boldsymbol{X}_n}(y \mid \boldsymbol{X}_n) \geq M\right) \leq \sup_{n\in\mathbb{N}} \frac{\mathbb{E}\left(\sup_{y\in\mathbb{R}} h_{Y_n|\boldsymbol{X}_n}(y \mid \boldsymbol{X}_n)\right)}{M} = \frac{B}{M}$$

Set $M_0 := \frac{B}{\varepsilon}$ to get

$$\sup_{n\in\mathbb{N}} \mathbb{P}\left(\sup_{y\in\mathbb{R}} h_{Y_n|\boldsymbol{X}_n}(y \mid \boldsymbol{X}_n) \geq M_0\right) \leq \frac{B}{M_0} = \varepsilon,$$

which shows that the sequence $\left(\sup_{y\in\mathbb{R}} h_{Y_n|\boldsymbol{X}_n}(y \mid \boldsymbol{X}_n)\right)_{n=1}^{\infty}$ is bounded in probability.

Asymptotic Stability: It remains to establish asymptotic stability as defined by Amann et al. (2023). To do that, we need to show for every $\varepsilon > 0$ that

$$\frac{1}{n}\sum_{i=1}^{n} \mathbb{P}\left(\left|\hat{f}^{\texttt{TF}}_{\mathcal{S}_n}(\boldsymbol{X}) - \hat{f}^{\texttt{TF}}_{\mathcal{S}_n^{\setminus i}}(\boldsymbol{X}) \geq \varepsilon\right|\right) \to 0 \tag{9}$$

for $n \to \infty$. From Proposition 1 we know that `TF` is β_n replace-one-stable with $\beta_n \to 0$ for $n \to \infty$. Proposition 4.6 of Liang & Barber (2023) shows that if `TF` is replace-one stable, then there exists an alternative learning algorithm `Alg` such that

1. $\hat{f}^{\texttt{Alg}}_{\mathcal{S}_m} = \hat{f}^{\texttt{TF}}_{\mathcal{S}_m}$ for any set of training data $\mathcal{S}_m$ with size $m = 1, \ldots, n-1$, i.e., `Alg` and `TF` are identical learning algorithms for training data sizes of $n-1$ and smaller.
2. `Alg` is β_n remove-one-stable, i.e.,

$$\sup_{i=1,\ldots,n} \mathbb{E}\left(\hat{f}^{\texttt{Alg}}_{\mathcal{S}_n}(\boldsymbol{X}) - \hat{f}^{\texttt{Alg}}_{\mathcal{S}_n^{\setminus i}}(\boldsymbol{X})\right) \leq \beta_n$$

For arbitrary $\varepsilon > 0$, applying Markov's inequality then yields

$$\frac{1}{n}\sum_{i=1}^{n} \mathbb{P}\left(\left|\hat{f}^{\texttt{Alg}}_{\mathcal{S}_n}(\boldsymbol{X}) - \hat{f}^{\texttt{Alg}}_{\mathcal{S}_n^{\setminus i}}(\boldsymbol{X}) \geq \varepsilon\right|\right) \leq \frac{1}{n}\sum_{i=1}^{n} \frac{1}{\varepsilon}\mathbb{E}\left(\hat{f}^{\texttt{Alg}}_{\mathcal{S}_n}(\boldsymbol{X}) - \hat{f}^{\texttt{Alg}}_{\mathcal{S}_n^{\setminus i}}(\boldsymbol{X})\right) \leq \frac{\beta_n}{\varepsilon} \to 0$$

for $n \to \infty$, which shows that `Alg` is asymptotically stable as defined in (9). Consequently, with `Alg` as the learning algorithm, all requirements of Corollary 4.6 of Amann et al. (2023) are fulfilled, so we can conclude

$$\mathbb{E}\left(\left|\mathbb{P}\left\{Y_{n+1} \notin \hat{C}_{\alpha}^{\texttt{J+(Alg)}}(\boldsymbol{X}_{n+1}) \;\middle|\; \mathcal{S}_n\right\} - \alpha\right|\right) \to 0 \tag{10}$$

for $n \to \infty$, where $C_{\alpha}^{\texttt{J+(Alg)}}$ is the `J+` prediction interval resulting from generating the leave-one-out predictors $\hat{f}_n^{\setminus i}$ with `Alg`. We now follow the reasoning in the proof of Theorem 4.4 in (Liang & Barber, 2023) to show that this also ensures the validity of (10) when computing $\hat{C}_{\alpha}$ using `ICJ+`

with `TF`, rather than applying `J+` with `Alg` as the learning algorithm. Observe, that when performing `ICJ+`, `TF` conducts in-context learning on samples of size $n-1$, where by definition, `TF` and `Alg` are identical. Consequently, the resulting prediction interval $C_\alpha^{\texttt{ICJ+}}$ from `TF` is exactly the same as $C_\alpha^{\texttt{J+(Alg)}}$, so the desired result immediately follows.

□

A.2 Additional Experiments

A.2.1 Function Classes for Dataset Generation

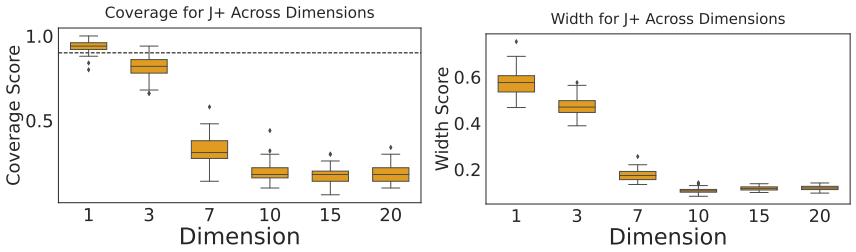
Now, we define each of the particular function classes. As described in the Section 4, any function class, in general, is given by $\mathcal{H} = \{h \mid h(\boldsymbol{x}) = g(\boldsymbol{x}) + \xi;\ \xi \sim \mathcal{N}(0, \sigma^2)\}$. According to the function class, $g(\boldsymbol{x})$ is defined in Table 3 below.

Table 3: Details regarding function classes for data generation. Symbols as defined in Section 4.

Function Class	Abbreviation	$g(\boldsymbol{x})$	Parameters	Remarks
Linear	LR	$\boldsymbol{\beta}^\top \boldsymbol{x}$	$\boldsymbol{\beta} \sim \mathcal{N}(0, \boldsymbol{I}_p)$	-
Quadratic	QR	$\boldsymbol{\beta}^\top \boldsymbol{x}^2$	$\boldsymbol{\beta} \sim \mathcal{N}(0, \boldsymbol{I}_p)$	Pointwise
Cubic	CR	$\boldsymbol{\beta}^\top \boldsymbol{x}^3$	$\boldsymbol{\beta} \sim \mathcal{N}(0, \boldsymbol{I}_p)$	Pointwise
ReLU 2-layer NN	NN	$\sum_{k=1}^r \alpha_k \sigma(\boldsymbol{\beta}_k^\top \boldsymbol{x})$	$\boldsymbol{\beta}_k \sim \mathcal{N}(0, \boldsymbol{I}_p)$, $\alpha_k \sim \mathcal{N}(0, 2/r)$	$r = 100$ is # hidden nodes $\sigma(\cdot) = \max(0, \cdot)$ is ReLU act. fn.
Modified Sinusoidal	SR	$\boldsymbol{\beta}^\top A \sin(f\boldsymbol{x} + \boldsymbol{\theta})$	$\boldsymbol{\beta} \sim \mathcal{N}(0, \boldsymbol{I}_p)$, $\boldsymbol{\theta} \sim \mathcal{N}(0, \boldsymbol{I}_p)$ $A \sim \mathcal{N}(0, \sigma^2)$, $f \sim \mathcal{N}(0, \sigma^2)$	Pointwise

A.2.2 Transformer Architecture and Training

Here we provide a detailed account of the transformer architectures used in our work. This has been detailed in Table 4. Training is conducted on a single NVIDIA GeForce RTX 4090 GPU with 24 GB of RAM, and runtime varying from 30 minutes to 11 hours depending on the model size. We use the Adam optimizer (Kingma, 2014) and train for a total of 500,000 steps with a batch size of 64. A fixed learning rate of 10^{-4} is used across all function classes and models.



(a) Coverage across dimensions

(b) Width across dimensions

Figure 2: Comparison of coverage and width across data dimensions using `J+` for a target coverage level of 90%. We observe that the coverage of `J+` has a roughly decreasing trend with an increase in the dimensions of the data. The poor performance of `J+` is however not due poor model fitting, we always obtain a MSE loss of 0.

A.2.3 PIs for 20 Dimensional Data

The comparison of coverage score and width score metrics for the 20-dimensional data is plotted in Figure 4, for all function classes. We observe that the performance of the proposed `ICJ+` is superior to that of the benchmark `J+` by a wide margin. To perform a sanity check for our `J+` results, we plot its PI metrics across dimensions (Figure 2). We observe that its performance does decrease with an increasing number of dimensions, and the degradation in performance is not due to poor model training.

Table 4: Details regarding variants of the GPT-2 transformer architecture.

Data dimension (d)	Model	Embedding size	#Layers	#Heads
1	Small	128	6	4
20	Standard	256	12	8

Table 5: Average coverage (C) and interval width (W), with standard deviations, for J+, ICJ+, and the True 90% CI on 20D data, aggregated over 5 function samples in the same function class for both evaluation regimes.

Function	**Linear**		**Quadratic**		**Qubic**		**ReLu**		**Sinusoidal**	
Method	C	W	C	W	C	W	C	W	C	W
					In Sample					
J+	0.30±0.01	0.15±0.01	0.30±0.04	0.33±0.04	0.24±0.02	2.03±0.13	0.18±0.03	0.09±0.01	0.13±0.02	0.19±0.10
ICJ+	0.94±0.00	0.72±0.01	0.94±0.00	0.79±0.02	0.94±0.01	1.42±0.09	0.94±0.00	1.57±0.64	0.94±0.00	0.61±0.03
True CI	0.90±0.00	0.49±0.00	0.90±0.00	0.49±0.00	0.90±0.00	0.49±0.00	0.90±0.00	0.49±0.00	0.90±0.00	0.49±0.00

A.2.4 PIs for 1 Dimensional Data

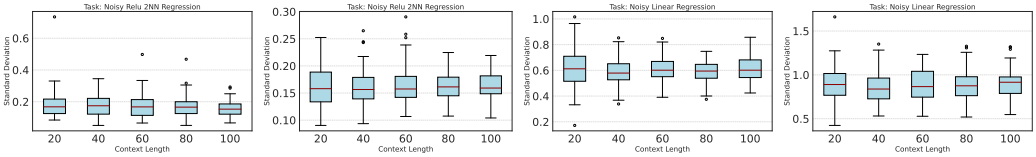
We provide the prediction intervals and metrics for the remaining four function classes in Figures 5, 6, 7, 8.

A.2.5 Symmetry to Training Data

We conduct an ablation study to check the symmetry to training data for transformers, which was an assumption for our theoretical result (ref. Definition 2). For our proposed method, symmetry to training data would translate to permutation invariance of the transformer prompt. For each function class (defined above), we create $N(\pi) = 100$ permutations of the input prompt and evaluate it on 64 test points. For each test point, we record the standard deviation for each of the 100 predictions. The same is repeated for each of the 64 test points and depicted via box plots in Figure 3. A lower standard deviation in predictions across the 100 permutations would point to the validity of the symmetry assumption. One can observe that the standard deviation values are quite low for the first row (ReLU 2-layer neural network) and a bit higher for the second row (linear regression). However, the comparison is also not completely fair since the standard deviation is not scale invariant, and data from each function class has a different scale. Therefore, we can conclude that transformers loosely obey the symmetry assumption, and that verifying this assumption can be challenging. Our future work would involve more exhaustive experiments to verify symmetry by taking into account scale invariant measures of variability, as well as exploring permutation invariant transformer architectures.

A.3 Epistemic Uncertainty-Aware Prediction Intervals

In the Conclusion, we highlighted the importance of incorporating epistemic uncertainty in PIs. Intuitively, estimating the epistemic or second-order uncertainty (Sale et al., 2023; Rossellini et al., 2024) would require access to multiple point estimates or multiple (conditional) probability distributions. One of the simplest ways to obtain this is by bootstrapping, i.e., creating models on resampled subsets. We can do this using ICL without retraining, and just by prompting as depicted in Algorithm 2. We propose a modified version of the Jackknife+-after-Bootstrap (Kim et al., 2020) called `ICJ+aB` (Algorithm 2) to account for epistemic uncertainty in its PIs. The main difference to J+aB is the use of in-context bootstrapping for creating the LOO point estimates. This results in epistemic uncertainty aware PIs without any additional training overhead that was incurred by the previously proposed J+aB. A comparison of the computational costs is depicted in Table 1.



(a) 1-D NN (b) 20-D NN (c) 1-D LR (d) 20-D LR

Figure 3: Transformer symmetry to input permutations. Each point in a box represents standard deviations of predictions across 100 permutations for a particular context length, and the box represents variability across 64 test points. The first row represents the function class ReLU 2-layer neural network, and the second row represents the function class linear regression.

Algorithm 2 `ICJ+aB`. In-context Jackknife+-after-Bootstrap for constructing epistemic uncertainty-aware PIs around point estimates.

Input: A trained transformer `TF`, Dataset $\mathcal{S}_n := \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, a test point $\boldsymbol{x}_{n+1} \in \mathcal{X}$

Output: Prediction interval $\hat{C}_\alpha^{\texttt{ICJ+aB}}(\boldsymbol{x}_{n+1})$

for $b = 1, \ldots, B$ **do**

  Sample index set $I_b = \left(i_1^b, \ldots, i_k^b\right)$ with replacement from set $\{1, \ldots, n\}$

end for

for $i = 1, \ldots, n$ **do**

  Define $J^{\setminus i} := \{b \in [B] \,:\, i \notin I_b\}$

  for $b \in J^{\setminus i}$ **do**

    Construct prompt $\boldsymbol{x}_{\text{prompt}}^{b,(m)} = \left(\boldsymbol{x}_{i_1^b}, y_{i_1^b}, \ldots, \boldsymbol{x}_{i_k^b}, y_{i_k^b}, \boldsymbol{x}_m\right)$ for $m \in \{i, n+1\}$

  end for

  Compute LOO predictions as

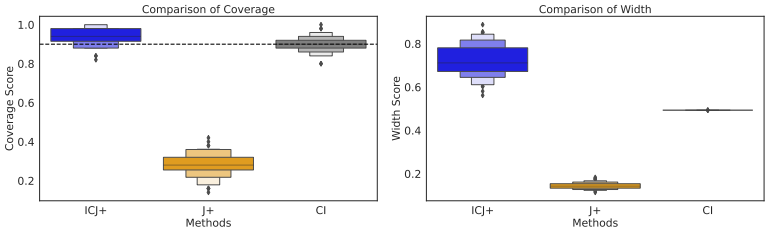
$$\hat{f}_{\setminus i}(\boldsymbol{x}_m) = \frac{1}{|J^{\setminus i}|} \sum_{b \in J^{\setminus i}} \texttt{TF}\left(\boldsymbol{x}_{\text{prompt}}^{b,(m)}\right) \text{ for } m \in \{i, n+1\}$$

  Compute $R_i = \left|y_i - \hat{f}_{\setminus i}(\boldsymbol{x}_i)\right|$

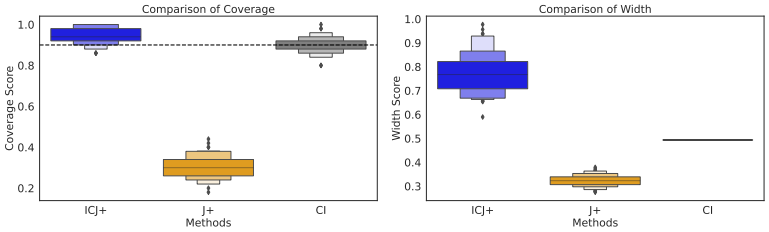
end for

Compute the `ICJ+aB` prediction interval at $\boldsymbol{x}_{n+1}$ as

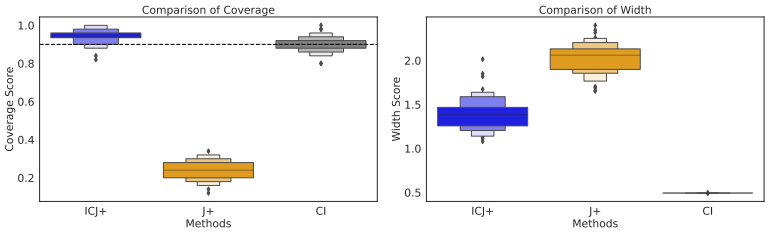
$$\hat{C}_\alpha^{\texttt{ICJ+aB}}(\boldsymbol{x}_{n+1}) = \left[Q_\alpha^-\left(\left\{\hat{f}_{\setminus i}(\boldsymbol{x}_{n+1}) - R_i\right\}_{i=1}^n\right), Q_\alpha^+\left(\left\{\hat{f}_{\setminus i}(\boldsymbol{x}_{n+1}) + R_i\right\}_{i=1}^n\right)\right]$$



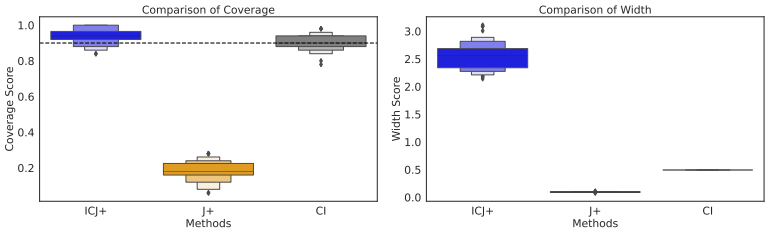
(a) Coverage and Width for LR.



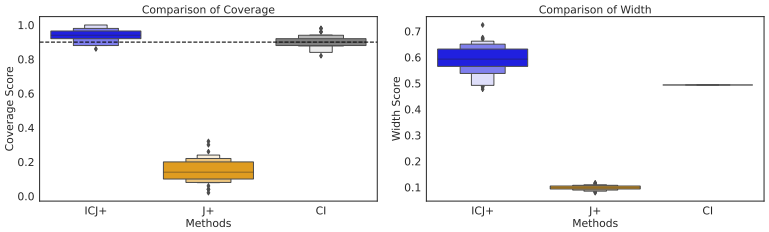
(b) Coverage and Width for QR.



(c) Coverage and Width for CR.

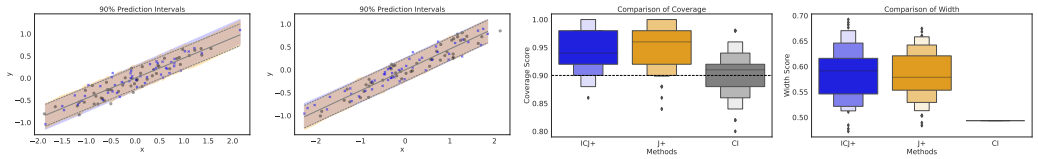


(d) Coverage and Width for NN.

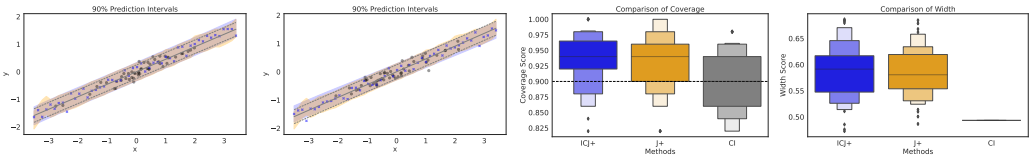


(e) Coverage and Width for SR.

Figure 4: Comparison of coverage and width scores for 20-dimensional data. We observe that our transformer-based `ICJ+` outperforms the benchmark `J+` consistently across all function classes.



(a) Prediction intervals and metrics for two different feature samplings of 1D LR and in-sample evaluation.



(b) Prediction intervals and metrics for two different feature samplings of 1D LR and out-of-sample evaluation.

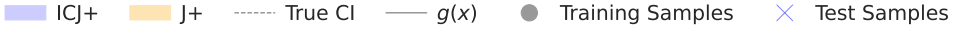
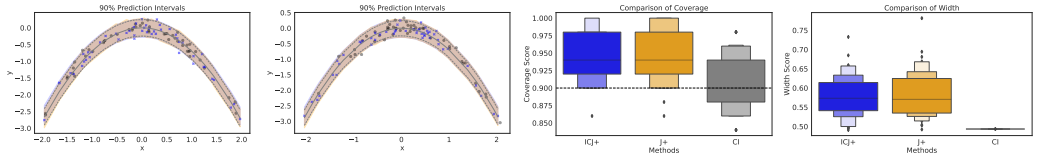
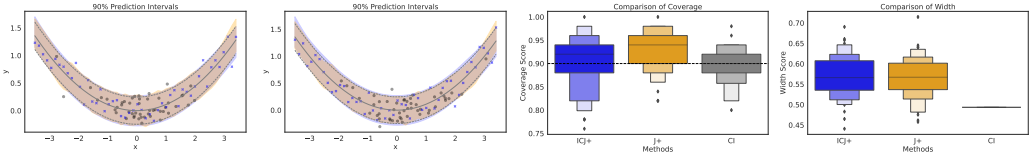


Figure 5: Comparison of PIs across different feature samplings of an instance of the one-dimensional LR function class ($\alpha = 0.1$).We observe that both methods reach the target coverage while maintaining comparable width, for in-sample out-of-sample evaluation.



(a) Prediction intervals and metrics for two different feature samplings of 1D QR and in-sample evaluation.



(b) Prediction intervals and metrics for two different feature samplings of 1D QR and out-of-sample evaluation.

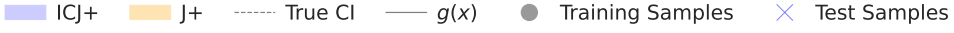
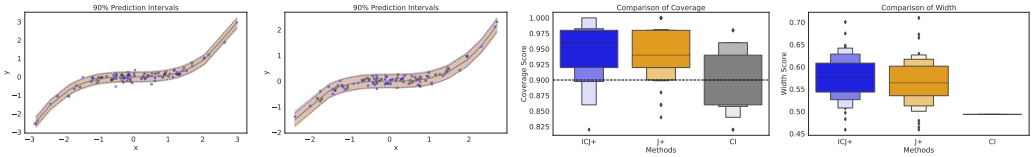
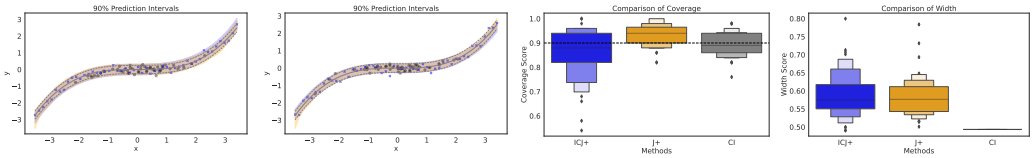


Figure 6: Comparison of PIs across different feature samplings of an instance of the one-dimensional QR function class ($\alpha = 0.1$). We observe that for in-sample evaluation `ICJ+` reaches the target coverage, while for out-of-sample evaluation it performs slightly worse.



(a) Prediction intervals and metrics for two different feature samplings of 1D CR and in-sample evaluation.



(b) Prediction intervals and metrics for two different feature samplings of 1D CR and out-of-sample evaluation.

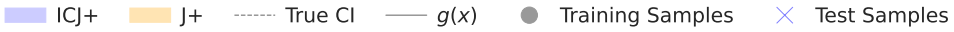
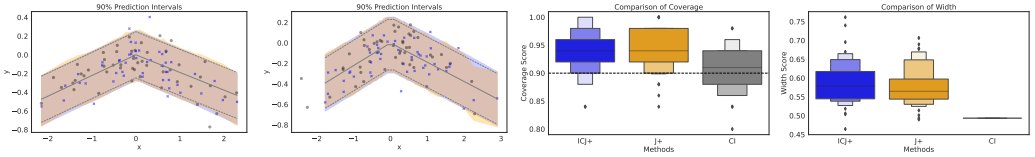
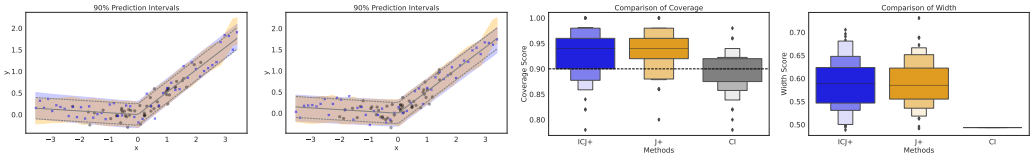


Figure 7: Comparison of PIs across different feature samplings of an instance of the one-dimensional CR function class ($\alpha = 0.1$). We observe that for in-sample evaluation `ICJ+` reaches the target coverage, while for out-of-sample evaluation it performs slightly worse.



(a) Prediction intervals and metrics for two different feature samplings of 1D ReLU and in-sample evaluation.



(b) Prediction intervals and metrics for two different feature samplings of 1D ReLU and out-of-sample evaluation.

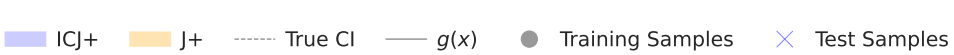


Figure 8: Comparison of PIs across different feature samplings of an instance of the one-dimensional NN function class ($\alpha = 0.1$). We observe that both methods reach the target coverage while maintaining comparable width, for in-sample out-of-sample evaluation.