Universiteit Leiden
The Netherlands

Opleiding Informatica

A study of TabPFN as surrogate Model in Bayesian Optimization

Ocean Wang

Supervisors:
Elena Raponi & PhD. candidate Ivan Olarte-Rodriguez

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

01/06/2025

Abstract

Bayesian Optimization (BO) is the standard approach for optimizing expensive black-box functions. However, the traditional Gaussian Process (GP) surrogate has signi cant scaling limitations and poor performance in higher dimension problems beyond 20 variables. In this study, TabPFN (Tabular Prior-data Fitted Networks), a transformer-based model, is studied if it would perform decently as an alternative surrogate for BO. As the model uses in-context learning and was pre-trained on 130 million synthetic datasets, it has the potential to adapt to optimization landscapes without parameter updates. A framework is developed that extracts the values and made compatible, needed for the BoTorch optimization framework to work. The 2 algorithms are compared on the COCO platform's BBOB test suite, which contains 24 functions, divided into 5 problem categories. All of them are available in di erent dimensions ranging from 2D to 40D.

Contents

1 Introduction

Optimizing expensive black-box functions has always been a challenging problem in current computational science and engineering. Optimization scenarios could range from hyperparameter tuning in machine learning to materials discovery, where each evaluation would come at computational or financial cost. BO tries to solve these problems by constructing probabilistic models of the objective function based on limited observations. These models are then used with a configuration balancing exploration and exploitation to determine where to select the next evaluation point. This approach naturally would depend on the quality of the surrogate model, as it should provide accurate predictions and reliable uncertainty estimates for the exploration-exploitation trade-off. GPs have been the traditional standard surrogate model due to their uncertainty quantification [RW06]. However, this traditional method does have limitations that become problematic with increasing problem complexity and dimensionality. Computationally, GPs have a $O(n^3)$ scaling due to matrix inversion operations. The most important issue is that the performance of GPs deteriorates significantly with problems in higher dimensions. Ideally, GPs should be used to problems in dimensions of smaller than 20 [Fra18]. TabPFN (Tabular Prior-Data Fitted Networks) represents a breakthrough in tabular learning, and has the potential to overcome the limitations of GPs [HMP+25]. The transformer-based model uses in-context learning to achieve good performance on a variety of datasets without the need for task-specific training. Its architecture processes the datasets through a dual attention mechanism, giving it the ability to handle feature interactions while also providing uncertainty quantification through distributional predictions. This bachelor thesis, conducted at LIACS under Professor Elena Raponi's supervision and PhD. candidate Ivan Olarte-Rodriguez who is part of the supervision team, investigates TabPFN's potential as an alternative surrogate model for BO. Comprehensive benchmarking will be conducted on the COCO platform's BBOB test suite, comparing TabPFN's performance withn the performance of traditional GPs across 24 black-box functions spanning five different problem classes. The focus will be on optimization convergence rates, and scaling behavior with larger problem dimensions. Expected Improvement (EI) will be used as acquisition function as its well established across the optimization research field [RP21]. This study aims to study the performance of TabPFN as surrogate model In BO, hoping to lead to advancements in efficient BO methodologies. The code for experimenting and implementation can be found in the public repository **TabPFN_VanillaGP_Benchmark** in the appendix (A).

2 Background information

2.1 Bayesian Optimization Framework

Bayesian Optimization (BO) addresses a fundamental challenge in optimization: e ciently nding the global optimum of expensive, black-box functions where traditional optimization methods fail due to computational constraints and lack of structural information. Such problem can be de ned as:

Black-box Optimization Problem:

$$x^* = \arg\min_{x \in \mathcal{X}} f(x) \tag{1}$$

where:

- $f : \mathcal{X} \to \mathbb{R}$ is the unknown objective function, with $\mathcal{X} \subseteq \mathbb{R}^d$
- $\mathcal{X}$ represents the feasible search space (domain) in d-dimensional real space
- d is the dimensionality of the optimization problem
- f is expensive to evaluate, black-box (no derivatives), and lacks exploitable structure
- Evaluations are limited to a budget $B \geq 1$ (total number of allowed function evaluations, typically a few hundred)

The BO framework operates with the following key notation: observed data points $\mathcal{D}_t = \{(x_i, y_i)\}_{i=1}^t$ where $X_t = \{x_1, \ldots, x_t\}$ represents the input locations and $Y_t = \{y_1, \ldots, y_t\}$ represents the corresponding function values, with $y_i = f(x_i)$. The tted surrogate model at iteration t is denoted M_t, and m represents the initial sample size for the design of experiments (DoE).

The BO framework addresses this challenge through four components that work together to solve the problem. First, a surrogate model constructs a probabilistic approximation of the unknown function f using observed data $\mathcal{D}_t$. This surrogate, traditionally a GP, provides both predictive capabilities and uncertainty quanti cation across the search space $\mathcal{X}$. Second, an acquisition function $A(x \mid M_t)$ leverages the surrogate model M_t to determine the most promising point for the next evaluation by balancing the exploration-exploitation trade-o . Third, critical design choices including initial DoE and kernel selection fundamentally shape the surrogate's performance and optimization behavior. Finally, the optimization strategy iteratively re nes the surrogate model as new observations become available, creating a principled sequential decision-making process that converges toward the global optimum with minimal function evaluations.

The mathematical elegance of BO lies in its Bayesian treatment of uncertainty. Unlike deterministic optimization methods that require complete knowledge of the objective function, BO maintains a probabilistic belief about f that evolves as evidence accumulates. This probabilistic framework enables the method to make informed decisions about where to sample next, considering both the potential for improvement and the uncertainty in unexplored regions. All these components

together-surrogate modeling, acquisition optimization, design choices, and sequential updating-creates a powerful algorithm for problems where an evaluation would carry a large computational or nancial cost.

2.1.1 Problem Characteristics and Assumptions

BO is de ned as solving a class of problems with very speci c characteristics by nding the global optimum, where traditional optimization methods would fail. These functions are typically expensive to evaluate, so a speci c budget of evaluations is set, usually limited to a few hundred, as each evaluation has a real nancial and time-wise cost, unlike traditional optimization where evaluations could be run up to a million times. Usually these functions are of black-box nature, which can be thought of as having a machine that takes input and gives output, without any knowledge of how the machine works. There is no derivative information available and the function lacks properties like concavity or linearity that would allow exploitation.

A critical assumption for such problems is that f should be continuous, as it enables the core BO strategy to use nearby observations $f(x)$ to predict f's behaviour at unobserved points [Fra18]. The COCO (Comparing Continuous Optimizers) software includes the BBOB benchmarking test suite that contains 24 noiseless black-box functions with known behavior that adhere to the aforementioned characteristics [HAR+20]. Each function has multiple instances where they are translated and rotated, and they are classi ed into the following classes: separable, unimodal with moderate conditioning, unimodal with high conditioning, multi-modal with adequate global structure, multi-modal with weak global structure. The last 2 classes would contain the problems that are the closest to real-world problems.

Another assumption is that the statistical properties (smoothness, variance, correlation structure) of the function remain consistent across the entire search space [RW06]. Traditional BO relies on this, that the function can be captured by a single surrogate model (GPs) with a stationary kernel. This means that the correlation between any two points depends only on their distance. However, as dimensions increase, the data becomes more sparse, and all points become approximately equidistant from each other. Frazier's paper acknowledges this limitation and mentions that BO is best suited over domains with less than 20 dimensions [Fra18]. A study on extensively testing traditional BO on BBOB functions with di erent con gurations supports this with evidence that their performance is competitive in dimensions less or equal to 5 for certain function types [RP21]. The shown performance variation suggests that a single con guration does not capture di erent function types equally well. The benchmarking study by Bliek et al. speci cally addresses the gap between synthetic test functions and real-world expensive optimization problems, where the real-world functions would not have the same statistical properties across the search space [BOR+22].

2.1.2 Mathematical Framework and Surrogate Modeling

The mathematical heart of BO is a method for statistical inference, a surrogate model, traditionally the GP regression. It provides both predictive capabilities and uncertainty quanti cation critical to guiding the optimization process [RW06, Fra18]. A GP can be de ned as a way of de ning a

probability distribution over functions. Formalized:

GP Definition: For a function $f : \quad ! \quad R$ defined over a domain $\quad R^d$, a GP prior is specified by:

$$f(x) \quad GP(\ _0(x); k(x; x^0)) \tag{2}$$

where:

- $_0(x) : \quad ! \quad R$ is the mean function
- $k(x; x^0) : \quad ! \quad R$ is the covariance kernel
- This prior encodes beliefs about the function's behavior before observing data

The GP contains hyperparameters including kernel parameters (lengthscales, variance), noise levels, and mean function parameters that control the model's behavior and must be learned from observed data, typically via maximum likelihood estimation.

Given observations $D_t = f(x_i; y_i)g_{i=1}^t$, the GP provides a posterior distribution at any point x:

GP Posterior Distribution:

$$f(x) \mid D_t \quad GP(\ _t(x); \ _t^2(x)) \tag{3}$$

where:

- $_t(x)$ is the posterior mean (best estimate of $f(x)$)
- $_t^2(x)$ is the posterior variance (uncertainty quantification)

In practice, function observations may contain noise due to measurement errors or stochastic processes. The GP framework naturally accommodates this by modeling observations as $y_i = f(x_i) + \ _i$ where $_i \quad N(0; \ _n^2)$ represents independent Gaussian noise with variance $_n^2$. This noise parameter becomes part of the hyperparameters learned during model fitting.

The power of GPs lies in their ability to update beliefs about unknown functions as new data becomes available. The posterior gives us the posterior mean, which is the best guess of the function at an observation, and the posterior variance, which quantifies the prediction's uncertainty. This creates an intuitive behavior where near observed points, we have low uncertainty since we have directly observed the function at these locations, while far from observed points, we experience high uncertainty as we are extrapolating beyond our known samples.

(a) Low noise (= 0:001) (b) High noise (= 1)

Figure 1: Gaussian Process regression demonstrating uncertainty quanti cation. The GP posterior mean (orange line) provides the best estimate of the unknown true function (red dashed line) based on six observed training points (blue dots). The shaded purple region represents the 95% con dence interval (4), illustrating how uncertainty increases with distance from observed data points. Near training points, the con dence interval is narrow, indicating high certainty, while in regions far from observations, the uncertainty grows. This uncertainty quanti cation is important to BO's ability to balance exploration of uncertain regions with exploitation of promising areas through acquisition functions. Note how the added noise on for the plot on the left is almost non-existent, causing the prediction to be interpolating with the training points. The opposite can be observed for the prediction on the right.

The choice of kernel function determines the GP's prior knowledge about the function's smoothness and structure [RW06]. Common kernels include the Squared Exponential (RBF) kernel, which assumes in nitely di erentiable and very smooth functions, and the Matern kernel, which uses a smoothness parameter to control di erentiability. Speci cally, the Matern 5/2 variant is popular in BO for its exibility in capturing di erent function behaviors [RP21]. However, kernel choice signi cantly impacts performance depending on the problem characteristics. The study by Le Riche et al. demonstrated that while the Matern kernel outperforms the exponential kernel on majority of functions, the exponential kernel performed better on multimodal functions with adequate global structure, as it maintains higher uncertainty further from observed points, advantageous for exploration [RP21].

2.1.3 Design Choices and Implementation

Critical design decisions fundamentally shape BO performance and must be carefully considered based on problem characteristics. The selection of the initial DoE serves as the foundation for the surrogate model. The study by Le Riche et al. [RP21] on benchmarking BO on COCO uses maximin Latin Hypercube Sampling (LHS), which divides the parameter space into equally probable intervals, ensuring good representation across all dimensions.

Their research demonstrates that initial DoE size signi cantly impacts surrogate performance. Testing 24 functions from COCO's BBOB test suite with di erent con gurations|small (S) with $d + 4$ points, medium (M) with $7:5d$ points, and large (L) with $20d$ points, assuming a total budget

of 30d evaluations|revealed that starting with a small initial DoE leads to better performance for the majority of BBOB functions in the rst 30 d evaluations. This counterintuitive result challenges the common belief that larger initial DoE would lead to better performance. However, for highly multimodal functions, the medium initial DoE outperforms the small one in later evaluations, and speci c functions with funnels (functions 6 and 24) bene t from larger initial DoE sizes requiring su cient global exploration.

Beyond initial DoE, kernel selection represents another crucial design choice with performance implications dependent on problem characteristics. The key insight is that kernel choice should match the expected smoothness of the problem|the Matern kernel assumes smooth interpolations while the exponential kernel is more suitable for functions requiring continued exploration due to multimodal landscapes.

2.1.4 Acquisition Functions

The acquisition function determines where the next point will be sampled and is critical to BO performance. These functions quantify the utility of evaluating the objective function at any given point by balancing exploitation and exploration. Given current observations $\mathcal{D}_t = \{(x_i, y_i)\}_{i=1}^t$ and the surrogate model providing posterior distributions $f(x) \sim GP(\mu_t(x), \sigma_t^2(x))$, the goal is to find the next evaluation point $x_{t+1} = \arg\max_{x \in \mathcal{X}} A(x \mid M_t)$ where A balances exploration of uncertain regions against exploitation of promising areas.

EI is the most common acquisition function [JSW98, Fra18], measuring the expected amount by which a new evaluation will improve over the current best observation:

Expected Improvement Formulation:

$$EI_t(x) = E[(f_t - f(x))^+] \tag{4}$$

Closed-form solution:

$$EI_t(x) = (f_t - \mu_t(x))\,\Phi(Z) + \sigma_t(x)\,\phi(Z) \tag{5}$$

where:

- $Z = \frac{f_t - \mu_t(x)}{\sigma_t(x)}$ (standardized improvement)
- $\Phi(Z)$ is the standard normal cumulative distribution function evaluated at Z
- $\phi(Z)$ is the standard normal probability density function evaluated at Z
- $f_t = \min_{y \in Y_t} y$ is the current best observed function value

EI naturally balances exploration and exploitation without manual parameter tuning|values are high when either the predicted improvement is large (decided by the exploration term $(f_t - \mu_t(x))\,\Phi(Z)$) or posterior uncertainty is high (decided by the exploration term $\sigma_t(x)\,\phi(Z)$). However, EI su ers from numerical instability problems as dimensionality grows or observations

increase, leading to numerically zero values across large regions [AD25].

Alternative acquisition functions include Upper Con dence Bound (UCB), which combines the posterior mean with a scaled uncertainty term but requires manual parameter tuning [SKKS12]. Knowledge Gradient (KG) measures the expected improvement in the maximum of the posterior mean after observing a new point, making it e ective when the nal solution need not be previously evaluated [Fra18]. Entropy Search (ES) and Predictive Entropy Search (PES) focus on reducing uncertainty about the location of the global optimum by maximizing information gain about the optimizer's location [HLHG14].

2.1.5 Sequential Optimization Algorithm

The complete BO algorithm integrates all components into such sequential framework:

Algorithm 1: BO Framework Input: Black-box function $f : ! R$ where R^d, evaluation budget B, initial sample size m, acquisition strategy A

1. Initialization:
   (a) Generate initial design $X_0 = f x_1; x_2; : : : ; x_m g$ using Latin Hypercube Sampling
   (b) Evaluate objectives: $Y_0 = f f (x_1); f (x_2); : : : ; f (x_m) g$
   (c) Form dataset: $D_0 = f (x_i ; y_i) g_{i=1}^m$ where $y_i = f (x_i)$
   (d) Set iteration counter $t = m$
2. Surrogate Construction:
   (a) Fit probabilistic surrogate model M_t to observations D_t
   (b) Learn hyperparameters via maximum likelihood estimation
3. Optimization Loop:
   (a) repeat
   (b) Update surrogate posterior using all available data
   (c) Optimize acquisition function: $x_{t+1} = \arg\max_{x2} A (x j M_t)$
   (d) Query objective: $y_{t+1} = f (x_{t+1})$
   (e) Augment dataset: $D_{t+1} = D_t [f (x_{t+1} ; y_{t+1}) g$
   (f) Increment: $t = t + 1$
   (g) until $t B$
4. Output: Best found solution $x = \arg\min_{(x;y)2D_B} y$

Figure 2 contains a sausageplot that visualizes a few iterations from a BO loop on a 1D function. It shows the acquisition function being maximized to determine the next candidate point. We can

see that the con dence is the highest at the evaluated points, and the lowest at the middle between 2 points.

Figure 2: BO iterations 4 to 7, on a 1D function $x^2 \sin(x) + 5$, using the EI acquisition function. The orange line represents the mean, red dotted line the true function, the red cross as the current best point (minimum), and the green start representing the next to be evaluated point determined by maximizing the EI acquisition function. The lightest shade of purple represents the uncertainty band for 4 standard deviations away from the mean, and the darkest representing 1 standard deviation.

2.1.6 Applications and Limitations

BO nds extensive application in hyperparameter optimization for machine learning models, particularly deep neural networks containing hundreds of adjustable parameters [Sra18]. Traditional approaches like grid search become computationally intractable with increasing hyperparameters, while random search fails to exploit structure and dependencies between parameters. BO treats hyperparameter optimization as a black-box problem where each evaluation involves training and

validating a complete model.

However, BO using GPs faces signi cant limitations that motivate alternative surrogate approaches. Performance degrades with increasing dimensions, typically being applied to problems with less than 10 variables [BGK+23], as points become equidistant and local correlation structures become unreliable. Computational complexity increases with every function evaluation due to GP's reliance on computing and inverting covariance matrices. Additionally, GPs assume smooth and stationary functions that can be modeled through kernel similarity measures, while real-world problems contain plateaus and discontinuities that violate these assumptions.

These limitations establish requirements for e ective surrogate models: maintaining computational e ciency that scales well with both problem dimension and evaluation budget, successfully handling mixed-type variables without extensive preprocessing, and adapting assumptions to problem characteristics rather than relying on xed kernel assumptions.

2.2 Tabular Prior-Data Fitted Networks (TabPFN)

2.2.1 From Traditional Machine Learning to In-Context Learning

Traditional machine learning works by rst training the model on individual datasets, saving the parameters, and then applying the trained model on new test samples. This approach requires complete retraining whenever new data becomes available, making it computationally expensive for applications like BO where we acquire new observations continuously. Each time a new observation is evaluated, traditional GP-based BO must re t the whole surrogate model, involving matrix inversions and hyperparameter optimization that scales expensively with dataset size.

2.2.2 Theoretical Foundation: Bayesian Inference via In-Context Learning

TabPFN's theoretical foundation is built upon Prior-data Fitted Networks (PFNs), which demonstrate that transformer architectures can perform approximate Bayesian inference through in-context learning [MFHH23]. To understand this approach, we rst clarify what in-context learning means and how it di ers from traditional machine learning.

In-context learning refers to a model's ability to adapt to new tasks using only the information provided in its input context, without updating any model parameters. Unlike traditional supervised learning where models are trained on speci c datasets through parameter optimization, in-context learning models are pre-trained once and learn to recognize patterns and make predictions based on the examples provided in the current context, performing inference without explicit training.

TabPFN applies this in-context learning principle to tabular prediction tasks. The main di erence lies between traditional function learning and meta-function learning approaches. Standard supervised learning tries to learn a mapping $g : \to R$ that transforms individual inputs to outputs for a speci c dataset, where R^d is our domain. The parameters are optimized through iterative training on a xed dataset, and the resulting function is then applied to new test inputs. This

method requires retraining whenever new data becomes available or when applied to new datasets.

TabPFN instead learns a meta-function that operates on entire datasets rather than single data points. While standard functions learn mappings from single inputs to outputs, a meta-function learns $F : (D_t; x_{test}) \to P(R)$ that takes entire datasets $D_t = \{(x_i; y_i)\}_{i=1}^{t}$ plus test inputs $x_{test} \in$ and outputs predictive distributions over R. This meta-function encodes a general learning algorithm that can perform pattern recognition and uncertainty quantification for any tabular dataset in a single forward pass. The parameters are learned once during pre-training across millions of synthetic datasets and remain fixed during inference, enabling zero-shot adaptation, which is the ability to make predictions on completely new datasets without any additional training.

The theoretical basis for this approach lies in approximating the posterior predictive distribution from Bayesian inference. Given observed data $D_t = \{(x_i; y_i)\}_{i=1}^{t}$ where $x_i \in R^d$ and $y_i = f(x_i)$ for some unknown function $f : \to R$, and a test input $x_{test} \in$, the posterior predictive distribution is:

$$p(y_{test} | x_{test}; D_t) = \int p(y_{test} | x_{test}; f) p(f | D_t) df \tag{6}$$

where $p(f | D_t)$ represents the posterior distribution over functions given the observed data. This integral cannot be computed exactly for most real-world functions, so traditional methods must use approximations or make strong assumptions about the function's behavior.

TabPFN learns to approximate this posterior predictive distribution directly through its in-context learning mechanism:

$$p(y_{test} | x_{test}; D_t) \approx q_\theta(y_{test} | x_{test}; D_t) \tag{7}$$

where q_θ represents the transformer model's output distribution parameterized by θ. The model achieves this approximation by processing the training dataset D_t as contextual information that informs the prediction for test input x_{test}, without requiring any parameter updates or iterative optimization procedures.

TabPFN is trained to predict masked target values across millions of synthetic datasets. During training, each synthetic dataset has a subset of samples with their target values hidden, simulating supervised prediction problems. The model learns to predict these masked targets given the input features and the unmasked samples as context. This training process enables TabPFN to learn general inference patterns that transfer to unseen datasets through in-context adaptation rather than parameter updates.

The meta-learning framework provides TabPFN with several advantages. The model can make predictions on new datasets without parameter updates, removing the need for hyperparameter tuning that traditional machine learning methods require for each new dataset. TabPFN demonstrates strong performance out-of-the-box across multiple tabular datasets. It performs well even with missing values, outliers, and uninformative features. This robustness is because of the model's exposure to a large amount of synthetic datasets during training, allowing it to handle various data

characteristics without manual preprocessing or model adjustments. For sequential optimization applications like BO, these characteristics are potential advantages over traditional GP-based surrogates, which require re tting and hyperparameter optimization each time new function evaluations are acquired [HMP+ 25].

2.2.3 Architecture Design for Tabular Structure

Standard transformer architectures are designed for sequential data processing, but tabular data has a 2D structure where both rows (samples) and columns (features) have semantic meaning. TabPFN addresses this challenge through a specialized dual attention mechanism that utilizes the structure of tabular data while keeping the computational advantages of transformer architectures.

Figure 3: The TabPFN architecture with the input dataset, a simple visualization of its dual attention mechanism, and its unique output [HMP+ 25].

TabPFN treats each cell in the table as a separate token and applies attention in two directions. Feature attention works horizontally across columns, letting each cell look at other features in the same row to understand complex relationships between variables. Sample attention works vertically down columns, allowing each cell to examine the same feature across di erent rows to nd patterns in the data.

The paper reports that TabPFN's memory scales linearly with dataset size at roughly 1,000 bytes per cell, allowing processing of datasets up to 50 million cells on modern hardware. The architecture can cache training data representations and reuse them for multiple predictions, which could bene t BO applications where the same training data is used repeatedly with new candidate points.

2.2.4 Synthetic Data Generation via Structural Causal Models

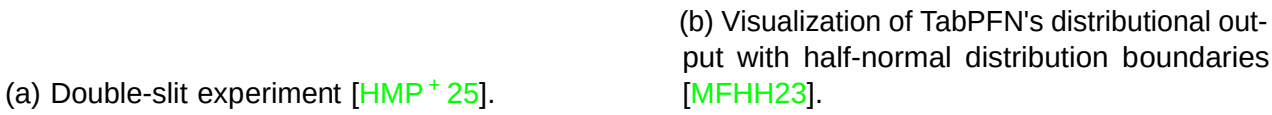
TabPFN takes a di erent approach to training compared to traditional machine learning models. Rather than collecting real-world datasets, TabPFN is trained on about 130 million synthetic datasets created using Structural Causal Models (SCMs). This covers a much wider range of data patterns than any collection of real datasets could provide. They represent causal relationships between variables through directed acyclic graphs where nodes represent variables and edges encode causal dependencies. The training process samples high-level hyperparameters like dataset size,

number of features, and di culty level to control each synthetic dataset's properties. Based on these hyperparameters, a computational graph is constructed that speci es the causal structure.

Figure 4: A: Sampling of high-level parameters. B: Build computational graph based on the sampled hyperparameters. C: Post-processing of the synthetic data [HMP25].

To generate each sample within a dataset, randomly generated noise is propagated through the root nodes of the causal graph. As this data moves through the graph edges, various computational mappings are applied: small neural networks with di erent activation functions, discretization mechanisms for categorical features, decision tree structures for rule-based dependencies, and Gaussian noise injection at each edge. After traversing the graph, intermediate representations are extracted at randomly sampled feature and target node positions to create the nal dataset. The training process includes common data problems on purpose. Missing values are randomly introduced in di erent patterns, outliers are added, and irrelevant features are included. Post-processing techniques like quantization and feature warping make the synthetic data more realistic and challenging. By training on millions of these di erent patterns, TabPFN learns general strategies that work across di erent types of tabular data. The model handles high percentages of missing values, many outliers, and lots of irrelevant features without special preprocessing. For BO applications, this robustness could be valuable since objective functions usually have complex, irregular behavior that violates the smooth assumptions made by traditional surrogate models. The main purpose of this is that by training on millions of datasets with di erent causal structures, feature types, and relationships, TabPFN learns to recognize and adapt to diverse patterns that occur in real tabular data, without being over tted to any speci c dataset characteristics [HMP+25].

2.2.5 Distributional Predictions and Uncertainty Quantification



(a) Double-slit experiment [HMP+25].

(b) Visualization of TabPFN's distributional output with half-normal distribution boundaries [MFHH23].

Figure 5: Combined figure showing both visualizations

TabPFN produces piece-wise probability distributions rather than single point predictions, which allows it to handle uncertainty in more flexible ways than traditional methods. The paper demonstrates this capability using a double-slit experiment example, where TabPFN predicts the multi-modal intensity patterns that emerge when photons pass through two slits. These patterns have multiple peaks and valleys that would be impossible to capture with simple Gaussian distributions. This piece-wise output allows for predicting probabilities to different regions rather than just a single prediction. These distributions can be asymmetric, meaning that the uncertainty could be unequal above and below the predicted value.

Bar distribution output implementation pseudocode

1. Range Discretization:
   (a) Divide prediction range into K discrete intervals (buckets)
   (b) Define bucket borders: $[b_i, b_{i+1}]$ for $i = 1, \ldots, K$
   (c) Compute bucket widths: $w_i = b_{i+1} - b_i$
2. Probability Computation:

(a) Model outputs logits $\ell_1, \ldots, \ell_K$ for each bucket

(b) Convert to probabilities: $p_i = \frac{\exp(\ell_i)}{\sum_j \exp(\ell_j)}$

(c) Scale by bucket width for density: $\phi_i = \frac{p_i}{w_i}$

3. Full Support Extension:

(a) Add half-normal distributions at boundaries

(b) Enable predictions beyond discrete bucket range

(c) Ensure valid probability distribution over entire real line

Unlike traditional regression methods that output point estimates, TabPFN's distributions provide information about the full range of possible outcomes. This is different from GP-based surrogate models, which provide posterior distributions under strict assumptions about function smoothness and stationarity. While GPs assume that functions can be modeled through kernel similarity measures and that statistical properties remain consistent across the search space, TabPFN's distributional approach can handle functions with plateaus, discontinuities, and sharp transitions. This would be especially useful in BO applications where the true functions might have complex behavior that violates GP assumptions.

Figure 6: TabPFN prediction on simple functions [HMP+25].

2.2.6 Potential for Surrogate Modeling in BO

TabPFN offers a fundamentally different approach to surrogate modeling in BO compared to traditional GPs. While GPs require matrix inversions that scale as $O(n^3)$ with dataset size [RW06], TabPFN's memory scales linearly through its transformer architecture [HMP+25]. However, this scaling advantage comes with trade-offs that affect practical BO applications. TabPFN requires processing entire datasets through transformer forward passes with multiple ensemble members at each BO iteration, which likely increases fitting time compared to GPs that only update matrix decompositions with new observations. The main question to ask is whether TabPFN's capabilities in handling non-stationary functions without kernel assumptions and its robustness to outliers and missing values can justify the increased computational cost per iteration. TabPFN's training on millions of synthetic datasets with diverse causal structures may enable it to capture complex function behaviors that violate GP assumptions about smoothness and stationarity. This could be especially valuable for higher-dimensional problems where GPs typically struggle [Fea18, BGK+23], though empirical validation is needed to determine if these theoretical advantages translate to

improved optimization performance within BO evaluation budgets.

While this research was conducted, Yu et al. (2025) independently developed GIT-BO, which uses TabPFN as a surrogate model for high-dimensional Bayesian optimization problems [YPA25]. Their approach extracts gradient information from TabPFN's forward pass to find low-dimensional subspaces for optimization in problems with more than 100 dimensions. Both studies examine TabPFN as a BO surrogate, but the methods are different. GIT-BO approaches high-dimensional optimization by reducing the search space using gradients, while this work tests TabPFN's performance on standard BBOB benchmarks in dimensions 2-40 without reducing dimensions. This shows that tabular foundation models are becoming recognized as serious alternatives to traditional GP methods.

Figure 7: Flowchart for the Bayesian Optimization loop, with the option of using GP regression or TabPFN regression to build the surrogate.

2.3 Benchmarking in BO

As mentioned earlier in the problem definitions, the COCO platform with its BBOB test suite provides the standard framework for benchmarking BO algorithms. It consists of 24 BBOB functions, categorized into 5 groups from separable to multimodal with weak global structure. Separable functions (f1-f5) can be decomposed into sums of univariate functions, allowing each dimension to be optimized independently without variable interactions. Functions with low/moderate conditioning (f6-f9) are unimodal with conditioning numbers between 10^1 and 10^3, creating

landscapes with directional bias but manageable numerical properties. Unimodal functions with high conditioning (f10-f14) has severe ill-conditioning around 10^6, where the most sensitive direction is approximately one million times more responsive than the least sensitive, creating narrow valleys. Multimodal functions with adequate global structure (f15-f19) combine multiple local optima with underlying learnable patterns such as periodic structures or predictable relationships between basins. Multimodal functions with weak global structure (f20-f24) are the most challenging landscapes with multiple optima but irregular or random characteristics that are difficult to exploit, making them difficult for surrogate-based methods that rely on learning global function behaviors. Each function generates multiple instances by having different optimums, rotations, and shifts of function values, preventing algorithm overfitting to specific function characteristics [HAR+20]. Key findings of Le Riche's study on benchmarking using COCO confirms that BO is sensitive to increasing dimension sizes, that there is no single BO configuration that dominated across all 24 BBOB functions, and that BO using GPs underperforms on the group of unimodal functions with high conditioning [RP21]. Another comparative study extends this benchmarking to high-dimensional scenarios. Santoni et al. conducted a comparison of high-dimensional optimization algorithms on the BBOB suite, evaluating algorithms on dimensions from 10 to 60 [SRLD24]. The study showed importance to choosing the appropiate algorithm depending on the specific problem characteristic, and further reinforces the importance of standardized benchmarking frameworks like COCO for fair algorithm comparison.

3 Research question

Transformer-based architectures have demonstrated amazing capabilities across different tasks. Recently, this has been extended to tabular data through models like TabPFN. The model utilizes in-context learning to achieve great performance on medium-sized datasets. As traditional GP-BO has limitations like scalability, this research investigates whether TabPFN can improve BO and address the challenges of traditional BO.

3.1 Primary research question

The primary research question guiding this investigation is: How effectively does TabPFN perform as a surrogate model for BO in regression tasks when evaluated on standardized benchmarks? To provide an answer to this question, multiple aspects are included in the research:

Performance Comparison: How does TabPFN compare to traditional GPs across the BBOB test suite in terms of optimization convergence rates?

Function Landscape Sensitivity: Does TabPFN exhibit differential performance across the five BBOB function groups, and when might it be preferred over traditional surrogates?

The experiments will be using the COCO environment's BBOB test suite, enabling standardized comparison with existing benchmark data in IOHprofiler.

3.2 Hypotheses

Several hypotheses guide this research based on TabPFN's demonstrated capabilities and GP limitations:

Better Performance on Complex Landscapes in higher dimensions: TabPFN is expected to outperform GPs on multimodal functions (BBOB groups 4-5) due to its dual-attention mechanism and ability to capture complex feature dependencies through its SCM-based prior.

Function-Specific Performance Patterns: Expected differential performance across BBOB groups - moderate improvement on separable functions, significant improvement on conditioned unimodal functions, and substantial gains on multimodal functions where TabPFN's non-stationary modeling capabilities should excel.

These hypotheses will be systematically tested through comprehensive benchmarking experiments on the BBOB test suite.

4 Methodology

4.1 Implementation framework

This section presents the implementation of TabPFN as a surrogate model within the BO framework, integrated with BoTorch for acquisition function optimization. The main challenge is extracting the mean and variance of TabPFN's native bar distribution output, presented earlier (section 2.2.5).

4.1.1 Architecture design

The integration requires a wrapper class that connects TabPFN's numpy-based transformer architecture with BoTorch's tensor operations. The TabPFNSurrogateModel class implements BoTorch's Model interface, so it preserves TabPFN's in-context learning capabilities, and allows the use of BoTorch's optimization pipeline.

Algorithm 1: TabPFN Surrogate Model Setup

Initialize TabPFNSurrogateModel:

1. Convert training data to appropriate tensor format
2. Con gure TabPFN with desired hyperparameters
3. Fit TabPFN on initial training data

Provide BoTorch Interface:

1. posterior() - Extract predictions with uncertainty
2. condition_on_observations() - Update model with new data

The wrapper first initializes the computational environment and model con guration. The default con guration from the developers of TabPFN is 8 ensemble estimators and 0.9 softmax temperature for regression. Each ensemble member applies di erent pre-processing or post-processing techniques to the same underlying model. Transformations include quantile normalization, outlier removal, SVD compression, or power transforms to create diversity. The softmax temperature calibration adjusts prediction con dence. To determine what hyperparameter con guration to use, we will conduct an initial exploratory analysis to investigate how di erent hyperparameters in uence the model's prediction.

BoTorch's acquisition functions expect Gaussian posteriors, while TabPFN naturally outputs the earlier described bar distribution. This means that we need a method to extract and convert this output so it can be passed onto the BoTorch pipeline.

Algorithm 2: Extract Predictions from TabPFN Function: Convert TabPFN output to BoTorch format

1. Prepare input points for TabPFN prediction

2. Get TabPFN's full probabilistic output (logits + distribution)
3. Extract mean and variance
4. Preserve gradient information for acquisition optimization
5. Return as Gaussian distribution for BoTorch compatibility

This process shows how TabPFN's distributional output is compressed into Gaussian parameters for BoTorch compatibility. It enables us to use the standard acquisition functions with the necessary uncertainty information for exploration-exploitation. TabPFN internally uses numpy computations, which would break the gradient ow for acquisition optimization. In order to maintain PyTorch's computational graph for gradient-based acquisition optimization within BoTorch, we attached a dummy tensor that has zero-impact to TabPFN's output.

After retrieving a new observation to evaluate from the acquisition function, it will be combined with the existing observations. TabPFN will be re- tted, and an updated surrogate model will be returned.

Algorithm 3: Update Model with New Data Function: Add new observations using in-context learning

1. Combine new observations with existing training data
2. Re- t TabPFN on expanded dataset
3. Update internal model state
4. Return updated surrogate model

TabPFN is able to update itself in-place due to it's ability to adapt to new data through its pre-trained transformer weights. Unlike GP methods, that would require the instantiation of a new model and re-solve matrix inversions with the new data.

4.1.2 Complete optimization loop

We will refer to the BO loop using the TabPFN regression model as surrogate as TabPFN-BO, and the one using traditional GP as GP-BO. The complete TabPFN-BO framework integrates all the presented components into a sequential algorithm shown in algorithm 4. It follows the detailed pseudocode of BO presented in the introduction section of BO (section 2.1.5).

Algorithm 4: Complete TabPFN-BO Process Input: Objective function, search bounds, evaluation budget

1. Initialization:
   (a) Generate initial design using Latin Hypercube Sampling

(b) Evaluate objective function at initial points

(c) Create TabPFN surrogate model

2. Optimization Loop:

(a) Create Expected Improvement acquisition function

(b) Optimize acquisition to nd next evaluation point

(c) Evaluate objective function at selected point

(d) Update TabPFN model with new observation

3. Output: Return best solution found

4.1.3 Alternative uncertainty quanti cation

Our current implementation uses statistical moments from TabPFN's output. However, an alternative approach we brie y experimented with is using quantiles to quantify uncertainty.

Algorithm 5: Quantile-Based Uncertainty Method Function: Extract uncertainty using quantiles

1. Access TabPFN's full distributional output
2. Calculate con dence intervals using inverse CDF
3. Use median as central prediction estimate
4. Convert quantile spread to variance estimate
5. Return compatible mean and variance for BoTorch, with median as the mean

This approach uses TabPFN's cumulative distribution function to compute con dence intervals directly from the learned distribution. Instead of using the mean, the median serves as the central tendency measure, which might be more robust to extreme values or a highly skewed/m

5 Exploratory analysis

In this section, we will brie y study how the approximation model looks for regression using GPs and TabPFN. For GPs, they will be tted on di erent con gurations of noise. TabPFN will be tted using di erent values for the number of ensemble estimators and the softmax temperature. Both regression models will be used in a 1D BO loop, and visualized with increasing observations alongside the corresponding acquisition curve, with TabPFN using a con guration determined from its individual behavior exploration. GP regression will be visualized under low noise ($\leq 0{:}001$) and high noise ($= 2{:}000$). The main goal of this section is to approximate what con guration TabPFN should be used for the full-scale experiments in a BO loop on the BBOB test suite.

5.1 Individual model behavior

5.1.1 Gaussian Process regression behavior

Figure 8: GP regression with 10 samples across increasing noise levels. Rows 1–3: $\sigma_n = 0.25$, 1.5; Bottom: $\sigma_n = 2.0$.

Figure 8 demonstrates how noise levels affect GP regression behavior on a 1D function with 10 training points. The GP mean prediction (orange line) remains consistent across all noise levels from $\sigma_n = 0.5$ to $\sigma_n = 2.0$, showing that the underlying function estimation is robust to noise parameter choices. The main difference we see across the plots is how confident or uncertain the model is at observations. Lower noise levels force the GP to interpolate more through observed points, while higher noise levels allows the model to deviate from training points. The confidence bands between training points remains the same across noise levels, but higher noise prevents overconfident predictions at observed locations. This would be beneficial for BO applications with unknown functions, so a moderate noise level between 1.0-1.25 might lead to better surrogate behavior. Decent uncertainty is maintained for exploration while also avoiding overfitting.

5.1.2 TabPFN regression behavior

This section explores TabPFN's regression behavior on basic functions from the original TabPFN paper to understand its suitability for Bayesian optimization [HMP+25]. We investigate how the number of training points affects prediction accuracy on symmetric functions like absolute value and sphere functions, where the location of critical points (minima, maxima) plays a key role in model performance. Additionally, we analyze two hyperparameters that influence TabPFN's uncertainty quantification: ensemble size and softmax temperature. These parameters directly impact the exploration-exploitation balance in BO, making their proper calibration crucial for optimization performance, especially in high-dimensional problems where traditional Gaussian processes struggle.

Figure 9: TabPFN regression on different functions: (top) abs and sin+x, (bottom) sphere and step functions

Figure 9 shows an attempt at recreating the TabPFN regression results on basic functions from figure 6. The exact code to recreate these plots could not be found, so the amount of data points to use has been counted manually. The first function what appears to be a scaled version of $\sin(x) + x$ uses 41 training points, the sphere function uses 11 training points, the absolute function 11 training points, and the step function seems to be using 11 training points for each plateau. We can see that the results do resemble the original results.

Figure 10: TabPFN regression on di erent functions with 1 less training point than original: (top) abs and sin+x, (bottom) sphere and step functions

Figure 10 shows the results on the same functions but with 1 less training point. It slightly a ected the prediction accuracy on the sphere function, but severely on the absolute value function, especially around the minimum of the function. We can hypothesize here that using an even amount of training points on symmetric functions with an optimum would degrade the regression performance heavily. This is because the extra point that would lead to an uneven amount of training points holds the information about the location where the 2 sides of the symmetric function connect, leading to a better approximation of the global landscape of the function. The following gures will use an uneven amount of training points again, 2 less than the original.

Figure 11: TabPFN regression on di erent functions with 2 less training points than original: (top) abs and sin+x, (bottom) sphere and step functions

From gure 11 we can observe that the connecting training point with an uneven amount does indeed lead to a better approximation of the global landscape of the function, although with less interpolation and precision. However, the prediction on $\sin(5x) + x$ seems to maintain its accuracy. The following gures will show the prediction on these functions again, but with another training point reduced, leading to an even amount of 8 for the absolute value, 38 for the $\sin(5x) + x$, 8 for the sphere function, and 38 for the step function.

Figure 12: TabPFN regression on di erent functions with 3 less training points than original: (top) abs and sin+x, (bottom) sphere and step functions

The results from gure 12 show a decrease in prediction accuracy in all functions besides the $\sin(x) + x$ function. The regression interpolates even less, and fails severely in capturing the main characteristic of the absolute value function at $x = 0{:}0$.

Figure 13: TabPFN regression on di erent functions with 4 less training points than original: (top) abs and sin+x, (bottom) sphere and step functions

In the nal plots of gure 13 we end up with an uneven amount again, with 4 less training points than original, and we can observe that the prediction got noisier and less accurate. However, the regression does capture the main characteristic of the absolute value function slightly.

With these results, we can infer that TabPFN would require more training points to get a decent prediction accuracy on the function, suggesting a larger intial DoE in BO applications compared to GPs, previously discussed in section 2.1.3. While a DoE of 4 might lead to a decent regression performance using GPs, TabPFN would fail to capture the global landscape, and have a quite noisy prediction. Our attempt at reproducing the plots from the original TabPFN article are all functions in 1D, meaning that we would start with a DoE of 4 in BO. From our previous plots, we can infer that this amount of training points would lead to a very inaccurate and noisy prediction, that would fail to capture the global trend of the function.

Figure 14: TabPFN regression with increasing ensemble estimator size, tted on 10 samples and a softmax temperature of 0.9. The lightest shade of purple represents the 90-10% quantile, while the darkest shade represents the 60-40% quantile.

Figure 14 shows how ensemble size a ects TabPFN's regression behavior on 10 training points with temperature 0.9. Increasing from 2 to 8 estimators generally smooths the mean and median predictions while making con dence bands more symmetric and smooth. However, 16 estimators produces more jagged predictions than 4 estimators, what suggests that more ensembles can lead to instability rather than improvement. Since each ensemble member requires separate preprocessing, larger ensemble sizes increase computational cost without guaranteed performance gains. The plateau in smoothness beyond 8 estimators, combined with diminishing returns and increased inference time, supports using 8 ensembles as the optimal con guration, which con rms the original articles choice of using 8 ensembles [HMP+25]. This choice balances prediction quality with computational e ciency. This is important for BO applications where the surrogate model is repeatedly updated at each iteration. The asymmetric con dence bands at lower ensemble sizes (2-4) also suggest insu cient uncertainty quanti cation, which could lead to bad exploration-exploitation

balance in BO.

Figure 15: TabPFN regression with increasing softmax temperature value, tted on 10 samples and 8 ensemble members. The lightest shade of purple represents the 90-10% quantile, while the darkest shade represents the 60-40% quantile.

In gure 15 we observe how softmax temperature a ects TabPFN's uncertainty calibration with 8 ensemble members on 10 training points. Lower temperatures (0.5-0.75) lead to overcon dent predictions where the model interpolates training points more with narrow con dence bands, similar to GP behavior with very low noise. As temperature increases to 1.0-1.25, the mean and median predictions diverge and con dence bands become wider. This suggests that the model becomes more uncertain about its prediction between the observations. At the higher temperatures (1.5-1.75), the model shows reduced tting to training data and worsens in approximating the function, but keeps high uncertainty throughout the domain. This temperature-uncertainty relationship would impact BO performance, as it directly a ects the acquisition function's exploration-exploitation balance. Higher temperatures would suggest more exploration, which would be important in high-dimensional problems where GP-based BO typically struggles due to sparse data coverage. A temperature

of 1.25 maintains reasonable predictive accuracy while also providing su cient uncertainty for exploration.

5.1.3 Visualization of TabPFN's output

Additionally, we visualize how a downsampled version of the TabPFN's bar distribution output looks like, paired with the regression model to see how they correspond to each other.

Figure 16: Visualizes TabPFN regression models mean and median prediction for 6 sample points with default parameters, along with con dence bands where the most outer one represents 90-10%, and the darkest 60-40%. The corresponding 'bucket' distribution for each training point is plotted, where the darkest shade represents the darkest con dence bound in the regression model.

In gure 16, we can observe that the distributions are highly multimodal and skewed, which causes the uncertainty bands to be assymetrical in the regression plot. There seems to be some kind of upper and lower limit across the sample points for the con dence bounds. This corresponds with the tallest peaks in the distribution. For sample point 5, the bar is the highest at the sample point and it matches with the what appears to be a lower bound on the regression plot, indicating that it's the most con dent about the value being at that location. This is not suprising as it is the initial training point. There is a second peak around 4-6 on the x-axis. This corresponds with the upperbound that we are seeing above the sample point. The same but opposite behavior can be observed at -5. We can infer from this that the model is con dent that the function will stay within these limits. While the tallest bar represents where TabPFN thinks the most likely value is, it still has the expectancy of the value being higher/lower on average, as the mean and median both

appear to be in the 60-40% range. It almost seems like there is a systematic bias.

Figure 17: Visualizes TabPFN regression models mean and median prediction for 7 sample points with default parameters, along with con dence bands where the most outer one represents 90-10%, and the darkest 60-40%. The corresponding 'bucket' distribution for each training point is plotted, where the darkest shade represents the darkest con dence bound in the regression model.

However, once we increase the initial points to 7 in gure 17, the prediction appears to turn linear. We can see that the con dence bounds become almost symmetrical, which is also to be observed in the histograms that have a light skew and are almost unimodal. It corresponds with how the median and mean have a slight divergence.

This mixed behavior would require a deeper investigation of TabPFN's internal workings, or on the data that was used for pretraining. It remains unclear why the model behaves this way. It appears to be interpolating for the known points, but for more points it assumes a linear trend and that any deviation would be noise rather than true function structure.

The regression turning linear paired with con dence regions of almost the same width across the

whole prediction poses a problem for BO applications as it is uninformative for the acquisition function. There would no accurate information about where to explore due to the constant variance, and the exploration term part of the expected improvement acquisition function$_t(x)$ (Z) would become primarily dependent on (Z), the standard normal probability density function evaluated at Z (section 2.1.4. (Z) peaks at $Z = 0$, so exploration only happens near points where the predicted mean (x) equals the current best f . There is no incentive to explore distant regions where the uncertainty should be the most. As result, expected improvement becomes dominated by the exploitation term $(f_t \quad _t(x))$ (Z), and the algorithm gets stuck exploring only around current best regions, leading to early convergence to local optima.

5.1.4 Side by side comparison on increasing samples

Figure 18: Comparison of both regression models with increasing sample points in steps of 3, TabPFN regressor tted with default parameters. The median curve is plotted as the con dence bands that represents quantiles are based around it.

The side-by-side comparison in gure 18 shows that GP regression does have a better approximation of the 1D function with lower amount of samples. We can also observe that TabPFN regression does not really interpolate through the points, but rather gets the general structure of the true function. With fewer samples at 5 points, GP has a reasonable approximation of the function's shape, while TabPFN shows instable behavior with high and constant uncertainty. As sample size increases

to 11-14 points, TabPFN's predictions stabilize and begin to approximate the true function more reliably, though still without exact interpolation. This behavior implies that TabPFN's pre-training biases it toward identifying global patterns rather than local tting, which could be useful in BO when function evaluations contain noise or when the goal is to avoid over tting to individual observations. However, the poor performance with small sample sizes shows that TabPFN may struggle in early BO iterations with limited data points, potentially requiring larger initial design sizes compared to GP-based approaches.

5.2 TabPFN-BO and GP-BO on a 1D function

Here we present the results from implementing GP regression and TabPFN regression in a basic BO loop using EI acquisition function, optimizing the same 1D function from before. Full iteration plots can be found in the appendix (appendix B). Figure 19, gure 20, and gure 21 shows the approximation model along the acquisition optimization plot. The rst thing we notice across all the plots is that EI has a much less informative, smooth acquisition curve with TabPFN-BO, and contains quite a lot of noise. This is not suprising as TabPFN outputs a complex probability distribution described earlier. TabPFN-BO does not explore as much as GP-BO due to its noisy acquisition function, it stays around the best current point, exploiting more. However, compared to TabPFN-BO, GP-BO has a lot of excessive uncertainty in unexplored regions with very wide con dence intervals. GP-BO does succeed in immediately nding the, what appears to be the minimum, at the rst iteration. We can also con rm our ndings in the previous section, that TabPFN struggles with capturing the global function landscape with a small amount of data points, and that the optimization behavior is dominated by exploitation around 1 region, and does not consider unexplored regions.

(a) GP iter 2

(b) TabPFN iter 2

(c) GP iter 4

(d) TabPFN iter 4

Figure 19: GP vs TabPFN comparison: Early iterations (2, 4)

(a) GP iter 6

(b) TabPFN iter 6

(c) GP iter 8

(d) TabPFN iter 8

Figure 20: GP vs TabPFN comparison: Mid iterations (6, 8)

(a) GP iter 10

(b) TabPFN iter 10

(c) GP iter 12

(d) TabPFN iter 12

Figure 21: GP vs TabPFN comparison: Later iterations (10, 12)

6 Experimental Design

6.1 Experimental setup

The experimental setup systematically compares TabPFN-BO against GP-BO using the COCO platform's BBOB test suite. The domain is set to $[-5, 5]^d$ by default. This test suite is accessed using the IOHpro ler library [DWY+18]. The complete suite of 24 functions cover 5 di erent function landscapes: separable functions (f1-f5), functions with low/moderate conditioning (f6-f9), unimodal functions with high conditioning (f10-f14), multimodal functions with adequate global structure (f15-f19), and multimodal functions with weak global structure (f20-f24). This categorization allows us to systematically analyze TabPFN-BO's performance across di erent problem types, addressing our hypothesis about di erent performance across landscapes.

Performance is benchmarked systematically across the dimensions 2D, 5D, 20D, and 40D. 20D and 40D is where GP-based methods start to degrade in performance signi cantly. TabPFN-BO might provide better performance here. Each algorithm-function-dimension combination undergoes evaluation across 3 instances with 5 repetitions each, yielding 15 runs per combination. The evaluation budget is calculated based on the dimension $budget = 10 \times dimension$. As presented earlier section 2.1.3 discussing design choices, smaller initial DoE sizes leads to better performance in the rst 30d evaluations. As we are using a small budget, we will calculate the DoE using the formula $DoE = 3 \times dimension$. It will be generated using maximin Latin Hypercube Sampling, which is the standard choice in BO. It maximizes the minimum distance between sample points for good space- lling design [FLS05].

6.2 Algorithm Con guration

Traditional GP is implemented using BoTorch's `SingleTaskGP` model, with the default Matern 5/2 kernel [S+20, RW06], maximum likelihood estimation for hyperparameter learning, and multi-start gradient-based L-BFGS-B for acquisition optimization, which is BoTorch's default optimizer. TabPFN-BO uses the con guration determined through the earlier exploratory analysis (section 5): 8 ensemble members and softmax temperature of 1.25. This con guration was selected based on a brief analysis showing that 8 estimators appears to provide optimal balance between prediction quality and computational e ciency, while temperature 1.25 would deliver appropiate uncertainty quanti cation for exploration in higher-dimensional problems. Both algorithms use the same gradient-based acquisition function optimization through BoTorch to ensure fair comparison.

6.3 Experimental Infrastructure

All experiments are executed on the computer equipped with an Intel i9-13900KF CPU, RTX 3080 FE GPU, and 32GB of available memory. GPU acceleration is enabled for both algorithms through CUDA for consistent computational conditions. The `BenchmarkManager` class coordinates the experimental work ow, managing result storage, runtime tracking, and convergence visualization.

For tracking, the IOHExperimenter [DWY+18] framework serves as the primary logger, tracking the optimization values through triggers that record function evaluations, current best values, and improvements. The data from the logs will later be used to generate regret-based convergence plots.

Finally, convergence plots are created using the tracking files generated by the logger. The plots contains 2 curves, aggregated over multiple runs and instances, representing the performance of each algorithm. This plot is created for each function-dimension combination.

6.4 Evaluation metrics

6.4.1 Optimization performance metrics

The primary performance metric keeps track of the logarithmic regret reduction, where regret is defined as $r(x) = f(x) - f^*$ with f^* representing the known global optimum. Convergence curves plot $\log_{10}(r(x_{best}))$ vs. function evaluations, allowing us to gain insight about the convergence behavior across different algorithms and problem dimensions.

These metrics should help us understand how quickly each algorithm approaches the global optimum, especially in higher dimensions where GP performance would degrade. It should also provide insight for TabPFN, if it has superior performance on less stationary function landscapes (BBOB groups 4-5) due to its in-context learning capabilities, and higher dimension problems. With this, we can infer when TabPFN-BO should be preferred over traditional GP-BO based on problem characteristics.

6.5 Statistical Analysis and Significance Testing

Significance Testing Implementation:
We employ the Wilcoxon signed-rank test for paired algorithm comparisons since optimization performance metrics typically exhibit non-normal distributions with potential outliers and skewness. For each function-dimension combination, we collect final regret values from 15 runs per algorithm, creating paired samples where each pair represents TabPFN-BO versus GP-BO performance on the same problem instance and random seed.

The test procedure computes differences between paired performances, ranks their absolute values, and assigns signs based on which algorithm performed better. Wilcoxons test with the two-sided alternative hypothesis will be used. With this we test whether the median difference between algorithm performances significantly differs from zero. This approach handles tied performance cases through midrank assignment and provides exact p-values for our sample size of 15 runs.

Pattern Detection Through Function Group Analysis:
Besides comparing individual functions, we aggregate results by BBOB function groups to identify algorithmic advantages. For each BBOB function category we calculate the proportion of functions where TabPFN-BO performs a lot better than GP-BO. We will use Fischer's exact test to decide whether TabPFN's rate of success for each function category would exceed chance performance

(50%). This will allow us to test our hypothesis of TabPFN performing better on multimodal (groups 4-5) functions compared to unimodal functions (groups 1-3).

6.6 Experimental Scope and Design Decisions

This study speci cally focuses on EI, the acknowledged standard of acquisition functions [ADE+25]. With this, we establish a baseline for TabPFN-BO performance without the need to consider factors for selecting alternative acquisition functions. We gain a clearer insight into its performance as a surrogate model, rather than di erent acquisition strategies.

Another limitation of our method is that we might be bottlenecking TabPFN's full potential. Our implementation extracts the mean and variance from TabPFN's output, compressing the complex and multimodal posterior information. The uncertainty based o this variance turns constant as observed in the earlier sections, which would lead to a BO algorithm that is exploitation dominant. Further elaboration and a potential solution will be discussed in the later sections. The priority lies more in making the model compatible with using BoTorch's acquisition pipeline. This provides a foundation for future work developing acquisition functions that makes full use of TabPFN's rich output.

The hyperparameter con guration we are using for TabPFN is determined from a brief exploratory analysis on a 1D function, rather than optimization across all BBOB function types. This approach may not represent optimal con gurations for maximum performance or speci c function landscapes.

7 Results

7.1 Benchmark performance

7.1.1 D2

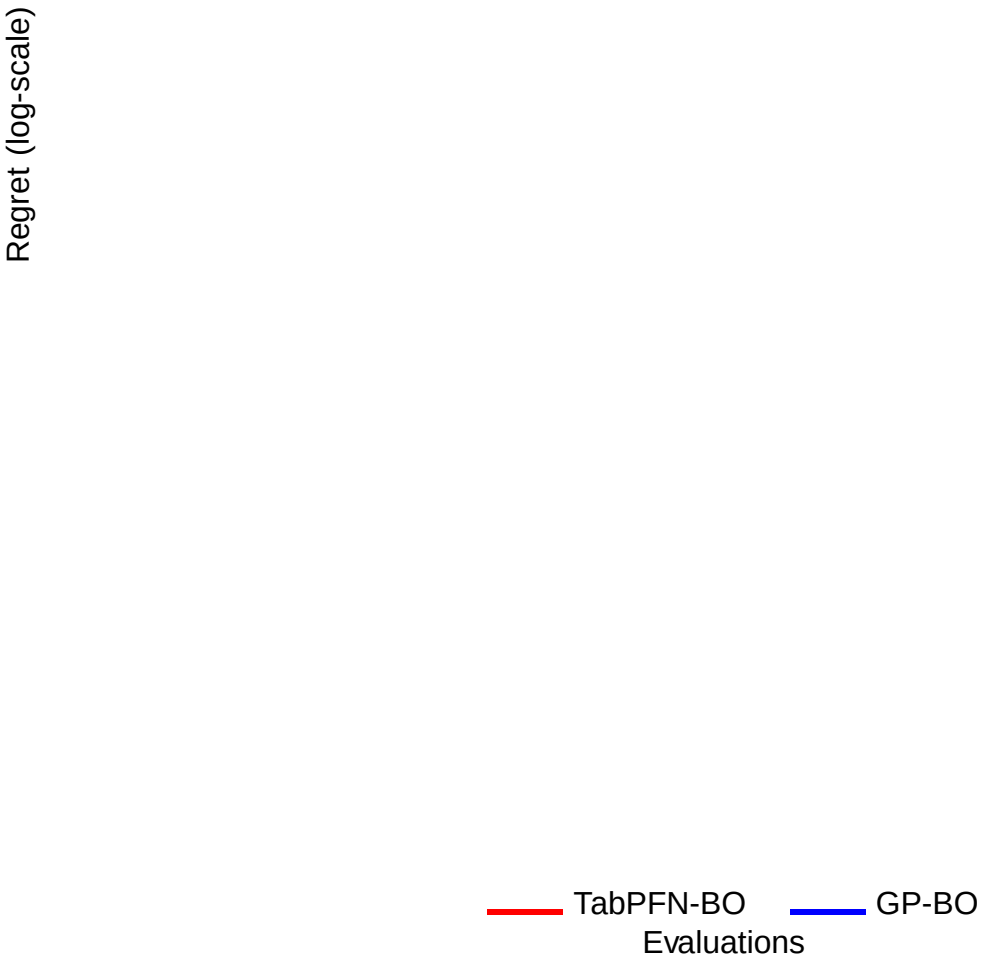


Figure 22: Convergence comparison between TabPFN-BO and GP-BO across all 24 BBOB functions in 2D.

In dimension 2, we can observe that GP-BO generally performs better than TabPFN-BO. This is especially noticeable in function 20, where it overtakes TabPFN-BO with a big gap, and the functions 14, 15, and 17 where it appears that TabPFN-BO stagnates for the evaluations starting from 5. the only function where TabPFN-BO outperforms GP-BO clearly over the whole budget is

function 5, although not with a huge gap. A convergence behavior that is noticeable across most of the plots, is that both algorithms perform similarly between budget 5-7, and then they start to diverge. In function 2, and function 24 this is visible too, but with the addition that the two curves cross each other at a later stage of the evaluations. The exceptions are function 11, 16, 18, and 23, where the performance starts to diverge around budget 10-12. We can see similar performance for both algorithms in functions 6, 12, 21, and 23.

As expected, GP-BO performs better than TabPFN due to the problems being low-dimensional. What is suprising, is TabPFN-BO performing slightly better than GP-BO on the linear slope function 5, while it is the most unimodal function with a monotonic landscape.

7.1.2 D5

In gure 23 we observe the similar behavior as in D2 where both algorithms perform similarly during the rst quarter of the evaluations. After this mark, they start diverging for majority of the functions, besides function 23, where they seem to diverge after the rst half of the evaluations. This divergence gap is especially large on function 5 and 6, both unimodal functions with a more 'monotonic' landscape. TabPFN-BO outperforms GP-BO on these 2 functions and function 1, 2, 3, and 23, although not by a huge gap. On the other majority of the functions, GP-BO seems to be performing better than TabPFN-BO. It is quite unexpected for GP-BO to be performing worse than TabPFN-BO on majority of the unimodal functions, as we are still in what is considered a dimension that should be favorable for GPs.

TabPFN-BO's better performance on the separable functions 1, 2, and 3 suggests that its training has developed strategies for functions that can be decomposed as sums of univariate functions, where the algorithm optimizes independently along each dimension without complex interactions. The advantage on the linear slope function 5 indicates TabPFN recognizes functions with consistent directional trends where movement in certain directions keeps improving the objective value. The performance on the attractive sector function 6, which has a highly asymmetric landscape, shows TabPFN may handle directional bias better than standard GP approaches. GP-BO's better performance on functions with high conditioning 10-14 and complex multimodal landscapes 15-22 shows the strength of its uncertainty quanti cation in navigating ill-conditioned and deceptive function surfaces. The mixed performance on the highly rugged function, Katsuura 23, demonstrates the challenge both methods face with irregular, non-smooth landscapes. It is quite unexpected for GP-BO to be performing worse than TabPFN-BO on majority of the unimodal functions, as we are still in what is considered a dimension that should be favorable for GPs.

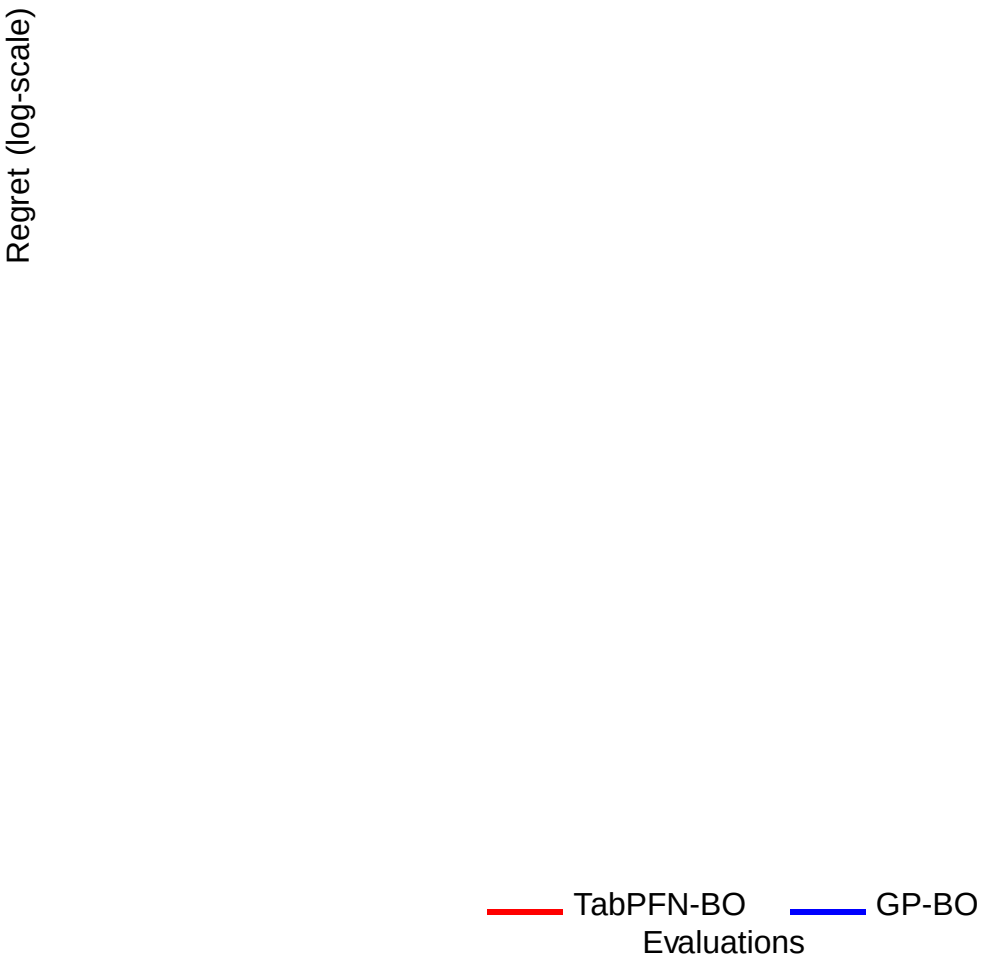


Figure 23: Convergence comparison between TabPFN-BO and GP-BO across all 24 BBOB functions in 5D.

7.1.3 D20

The same behavior for the rst early half of the evaluations can be observed in gure 24 as in the previous dimensions. After this point, the 2 convergence curves diverge. This is valid for majority of the functions, although functions 12, 14-17, 23, and 24 diverge minimally, or rst diverges and then converges back to each other. It is not expected to see TabPFN-BO perform similarly to GP-BO on majority of the functions within the multimodal group, as we D20 is where GP-BO should start to struggle. TabPFN-BO outperforms GP-BO on the functions that are seperable or have low/moderate conditioning, while GP-BO performs better on the unimodal functions with high-conditioning besides function 13, contradictory to the earlier mentioned ndings in the study

of Le Riche in section 2.3. Function 12 contains a narrow valley that must be followed to find the optimum. Function 13 is similar, but the valley has a v-shape making it non-differentiable. This suggests that TabPFN-BO performs better than GP-BO on functions with non-differentiable characteristics. The sudden drop around evaluation 60 in majority of the functions within the first 2 groups suggests that TabPFN-BO has reached an important amount of context information needed for its in-context learning mechanism to become more effective than the GP-based approach. GP-BO seems to be converging more stable without drastic changes in performance. This can be tied to GPs updating their uncertainty estimates incrementally with each new observation. The posterior variance decreases smoothly as more points are added, gradually turning more confident.

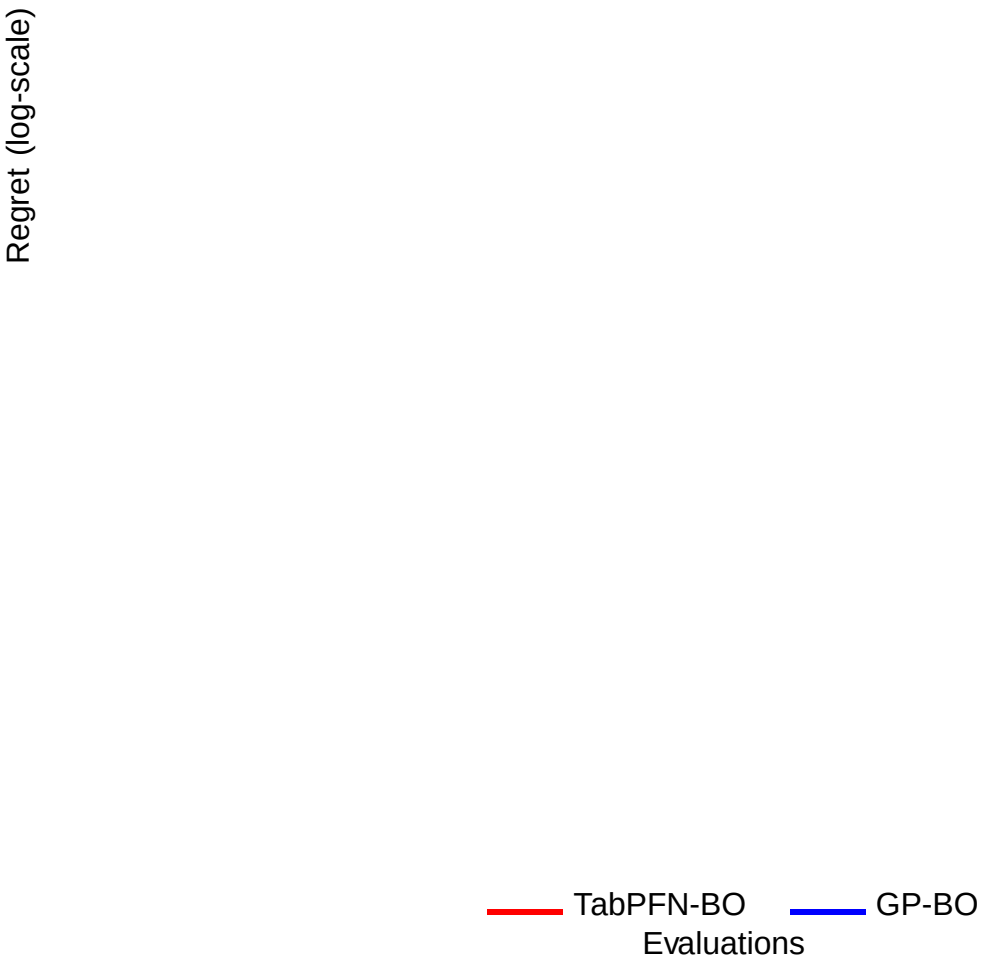


Figure 24: Convergence comparison between TabPFN-BO and GP-BO across all 24 BBOB functions in 20D.

7.1.4 D40

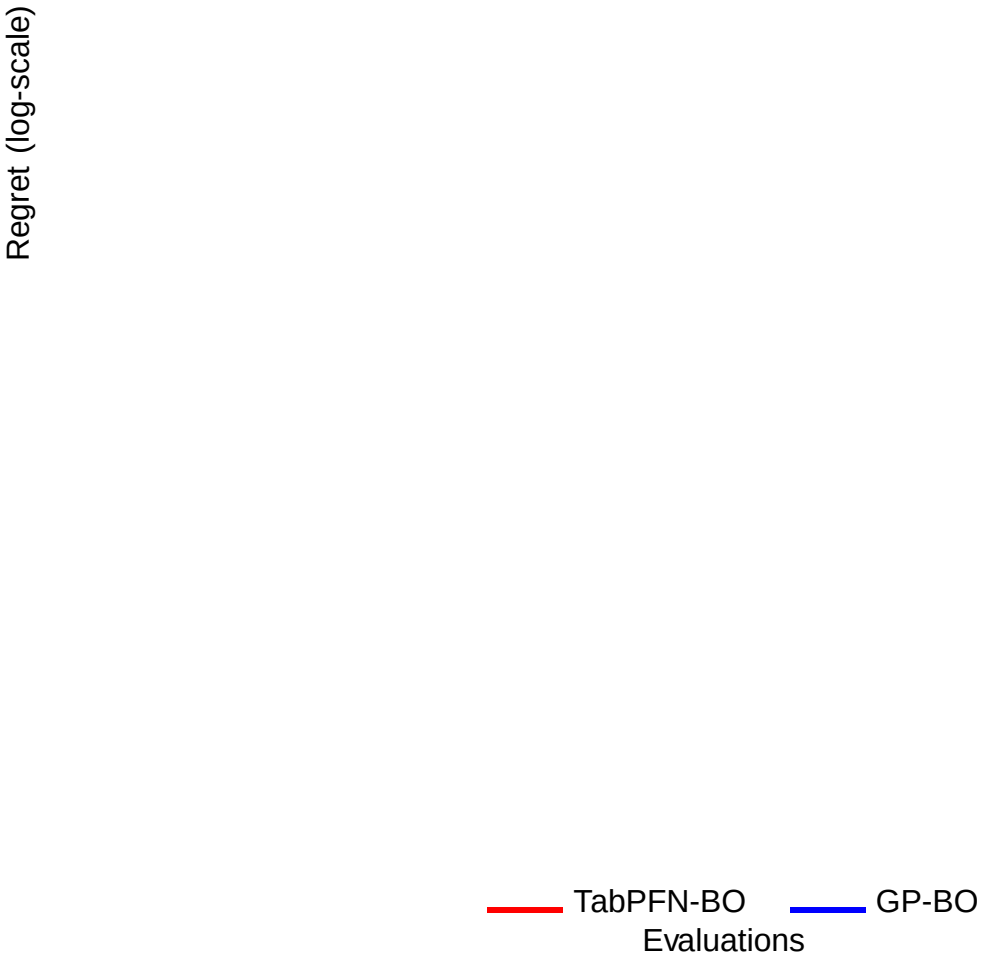


Figure 25: Convergence comparison between TabPFN-BO and GP-BO across all 24 BBOB functions in 40D.

Figure 25 shows how GP-BO lags behind in performance across majority of functions. Especially on functions 1-8, 13, 15, 20-22, and 24, the convergence gap between the two curves widens sooner or later during the evaluations. TabPFN-BO's superior performance on separable functions (f1 to f3) demonstrates that its learned representations can still exploit the independent optimization structure even at 40D, while GP-BO struggles with the curse of dimensionality in modeling high dimension separable landscapes. The advantage extends to functions with moderate conditioning (f6 to f8) and the linear slope function (f5), suggesting TabPFN maintains its ability to recognize

directional trends in high dimensions. The only function where GP-BO performs slightly better than TabPFN-BO is function 12 (Bent cigar), where the algorithm must navigate a narrow valley with extreme conditioning. Similarly to the performance in D20, TabPFN-BO outperforms GP-BO on function 13 (Sharp ridge), indicating that its learned patterns can handle the non-di erentiable characteristics better than GP's smoothness assumptions. On all functions we can see that GP-BO stagnates early and struggles to nd lower values at a slow pace, while TabPFN-BO shows drastic changes in converging behavior sooner or later on majority of functions. This behavior is particularly visible on the multimodal functions with weak global structure (f20 to f24), where TabPFN's sudden performance improvements suggest it recognizes exploitable patterns even in these irregular landscapes. Function 21 shows a sudden drop at the end, indicating TabPFN may have found useful structure among the 101 randomly positioned Gaussian peaks. However, both algorithms perform similarly on functions 11, 12, 14, 16, 17, and 23, suggesting these functions present challenges that neither approach can e ectively overcome at 40D. At the end stages of functions 18 and 19, TabPFN-BO appears to be widening the gap with GP-BO, suggesting that TabPFN-BO is discovering the underlying landscape structure and that additional evaluations would likely lead to a bigger convergence gap.

8 Discussion and analysis

8.1 Performance Patterns

The convergence analysis across dimensions shows how the two algorithms perform di erently as problem complexity increases. In 2D, GP-BO performs better on most functions, which matches expectations for low-dimensional problems. Both methods start with similar performance during the rst 5-7 evaluations, then their paths diverge. This pattern holds for most functions, though 11, 16, 18, and 23 don't diverge until around evaluations 10-12. TabPFN-BO does better on the linear slope function (f5), which is surprising since this monotonic function should favor GP methods in low dimensions.

Moving to 5D changes the performance balance. TabPFN-BO starts showing advantages on separable functions (f1 to f3) and functions with moderate conditioning (f5 to f6). This suggests TabPFN has learned to recognize separable structures where each dimension can be optimized independently. GP-BO still performs better on highly conditioned functions (f10 to f14) and complex multimodal problems (f15 to f22), where its uncertainty estimates help navigate di cult landscapes. Both methods struggle with Katsuura (f23), demonstrating the di culty of this function with an irregular and spiky landscape.

At 20D, the pattern of similar initial performance followed by divergence continues, but TabPFN-BO begins showing sudden performance jumps around evaluation 60 on separable and moderately conditioned functions. These abrupt changes indicate TabPFN needs enough context before its learning mechanism becomes e ective. GP-BO maintains steadier convergence because it updates uncertainty estimates gradually with each new observation. The di erence between functions 12 and 13 is telling: GP-BO handles the smooth but narrow Bent Cigar valley better, while TabPFN-BO outperforms on the Sharp Ridge with its non-di erentiable bottom.

The 40D results demonstrate that GP-BO struggles within high dimensions. TabPFN-BO outperforms GP-BO on most function types, including separable functions, moderate conditioning problems, and some multimodal landscapes (f20 to f24). GP-BO stagnates early and converges slowly, re ecting the di culty of modeling high-dimensional spaces with Gaussian processes. TabPFN-BO's sudden performance improvements suggest it nds useful patterns even in irregular high-dimensional problems. The functions where both methods show similar performance (11, 12, 14, 16, 17, 23) share characteristics that create fundamental optimization di culties: the Discus function (11) has extreme directional sensitivity, the Weierstrass function (16) has rugged surfaces with no clear structure, and Katsuura (23) has a spiky irregular landscape that provides no useful information for to exploit. Analysis of BBOB function groups shows TabPFN-BO performed better on unimodal functions than multimodal ones, achieving 53.6% success on groups 1 to 3 vs 22.5% on groups 4 and 5. Fisher's exact test found no signi cant di erence between these categories ($p = 1.000$), which contradicts the initial hypothesis that TabPFN-BO would perform good on multimodal problems. Again as discussed earlier, this can be explained by the uninformative variance that does not allow for a lot of exploration. Which leads to the algorithm being stuck at local optima on multimodal landscapes.

Figure 26: TabPFN success rates on unimodal vs multimodal BBOB functions. Despite expectations that TabPFN would perform better on multimodal problems due to its diverse training data, the results show higher success rates on unimodal functions (53.6%) compared to multimodal functions (22.5%). Fisher's exact test found no signi cant di erence between function categories ($p = 1.000$).

The benchmark evaluation across all the BBOB problem con gurations showed that TabPFN-BO achieved a win rate with slight advantage over GP-BO (51.4% vs 48.6%). However, the Wilcoxon signed-rank test revealed no statistically signi cant di erence between the algorithms ($W = 20230$, $p = 0.637$), indicating that this di erence likely represents random variation rather than actual better performance of an algorithm. The performance does vary across dimensions as expected. GP-BO in low-dimensional settings, winning 79.2% of 2-dimensional and 61.1% of 5-dimensional problems. This is reversed dimension 20 and 40, with TabPFN achieving 62.5% and 83,3% win rates for 20-dimensional and 40-dimensional problems. This suggests that TabPFN's

in-context learning mechanism better captures complex feature interactions in high-dimensional spaces, while GPs are better for low dimensions. This aligns with the claims that GPs in BO work well up until dimension 20.

Figure 27: Algorithm performance comparison across 288 BBOB benchmark problems. TabPFN achieved a marginal advantage (51.4% win rate), but the di erence was not statistically signi cant (Wilcoxon signed-rank test, $p = 0:637$).

Figure 28: Win rates by problem dimension showing dimensional crossover behavior. Vanilla BO dominates in low dimensions while TabPFN excels in high-dimensional problems.

8.2 Limitations and challenges

TabPFN-BO performs inconsistently in low dimensions while GP-BO performs well. The method needs a certain amount of context information before it becomes e ective, which creates problems during early optimization when quick progress matters most. This threshold e ect appears around evaluation 60 in higher dimensions, meaning TabPFN-BO may waste valuable function evaluations before it starts working properly. This corresponds with our earlier ndings during the exploratory analysis where we observed that TabPFN has di culties with capturing the function landscape with less data or context, combined with the models behavior to exploit around known regions instead of exploring.

The training data constrains TabPFN's performance in ways that aren't immediately obvious. Since TabPFN learns from synthetic datasets created during pretraining, it can only handle problems similar to what it saw during training. If a real optimization problem has characteristics missing from the training distribution, TabPFN will likely perform poorly. This explains why both methods struggle on functions like Katsuura (f23), but TabPFN can't even apply learned patterns because we can assume that the function is too di erent from its training examples.

GP-BO degrades in performance in high dimensions as expected. The method requires matrix operations that become expensive with more data points, and modeling high-dimensional surfaces accurately becomes nearly impossible. This leads to early stagnation where GP-BO stops making meaningful progress because its uncertainty estimates become unreliable as the data points become

sparse. The algorithm essentially gives up exploring because it has difficulty in distinguishing between promising and unpromising regions.

Certain function types cause problems for both approaches. Functions with extreme conditioning ratios around 10^6 create numerical challenges that both methods can't solve. Problems with random structures and irregular landscapes do not have a lot of useful information that either algorithm can exploit. GP-BO assumes functions are smooth, which fails on non-differentiable problems like Sharp ridge, while TabPFN-BO struggles with the precise navigation required for narrow valleys like Bent cigar where small errors have large consequences.

These limitations align with recent concurrent work by Yu et al. (2025), who demonstrated through ablation studies that direct application of TabPFN fails in high-dimensional problems [YPA25]. Their experiments showed vanilla TabPFN v2 performed 8.6 times worse than their gradient-informed approach, with drastic degradation in both convergence speed and final results. The authors concluded that TabPFN cannot capture crucial directions in high-dimensional spaces without assistance from subspace identification methods. Their success on problems up to 500 dimensions suggests TabPFN could work beyond the ranges tested here, but only when combined with proper dimensional reduction techniques like their gradient-informed subspace approach.

8.2.1 A potential solution: Spatial Density-Based Variance

Hereby an alternative solution to the method created by Yu et al. (2025) is presented. As BO relies on the exploration-exploitation balance of the acquisition function, a variance quantification is needed that is non-uniform across the search space. TabPFN produces such variance as shown earlier in this paper, and creates a major problem for the acquisition function, leading to no exploration and only exploitation around known regions. Which ultimately, leads to convergence at local optima. The Spatial Density-Based Variance (SDV) solution creates artificial uncertainty estimates based on the spatial distribution of observed data. The variance will be computed at any point x using 3 scalars: α (0.1-1.0) sets the minimum scaling factor near training data, β (1.0-5.0) determines the maximum additional scaling in unexplored regions, and σ (0.1-1.0) acts as a lengthscale controlling the rate of uncertainty increase with distance. Lower alpha values reduce uncertainty near training points more dramatically while higher values maintain baseline uncertainty everywhere. Lower beta increases uncertainty in distant regions modestly while higher beta creates dramatic uncertainty increases in unexplored areas. Smaller sigma produces sharp transitions between explored and unexplored regions while larger sigma creates smooth transitions. This gives us the following formula for computing the artificial variance:

$$\sigma^2_{\text{artificial}}(x) = \alpha + \beta\,(1 - \exp(-d_{\min}(x)/\sigma)) \tag{8}$$

where $d_{\min}(x)$ represents the distance to the nearest training point. This method would restore spatial awareness to TabPFN's predictions, enabling a proper balance between exploration and exploitation.

Figure 29 shows how the regression looks like before and after the scaling of variance. We can see that the uncertainty bands start to resemble the sausages of a sausageplot for GPs. The transitions between uncertainty regions is sharp as we have a low σ value, and the uncertainty at observations is dampened a lot because of the low α value.

Figure 29: Regression of TabPFN with and without arti cial variance. The right plot shows how the uncertainty is scaled.

The following gures shows the progress of optimization using TabPFN-BO with SDV (SDV-BO) on the same 1D function used in our exploratory analysis. The full iteration can be found in the appendix (C). We can observe that the exploration behavior is comparable now with GP-BO and what we would expect from EI's exploitation-exploitation trade-o . The acquisition function has smoothened as well compared to TabPFN-BO without SDV from section 5.2 due to the scaling of variance.

(a) Iteration 3

(b) Iteration 4

(c) Iteration 5

(d) Iteration 6

Figure 30: SVD-BO optimization iterations 3-6 showing the evolution of the optimization process, with the middle plot showing how much the uncertainty is scaled.

(a) Iteration 7

(b) Iteration 8

(c) Iteration 9

(d) Iteration 10

Figure 31: SVD-BO optimization iterations 7-10 showing the continued evolution of the optimization process

9 Contributions and Future Directions

9.1 Summary of Contributions

In this thesis, TabPFN is adapted as a surrogate model for BO instead of the traditional GP regressor surrogate. It is studied if TabPFN can address the scalability issues of GPs. We developed an integration framework that can extract values from TabPFN's distributional predictions for use within the BoTorch framework. With this, we can compare its performance against GP-based optimization. Using the benchmarking test suite BBOB, we can run both algorithms on black-box functions with di erent landscapes in multiple dimensions. Our current results show competitive performance in 2D, and superior convergence in 20D. This shows promising directions for overcoming the dimensional limitations for problems with fewer than 20 variables within BO. However, TabPFN-BO does not outperform GP-BO by a huge gap, and as con rmed by the study of Yu et al. (2025), TabPFN implemented within BO without any major adjustments does not perform well [YPA25]. This is the motivation for the proposed alternative solution of SDV-BO that scales the variance outputted by TabPFN arti cially. This research builds fundamental understanding of how TabPFN performs across di erent function types and dimensions, setting the foundation for the proposed SDV-BO approach that suggests further research directions.

9.2 Future Research Directions

Findings of this initial investigation of TabPFN as a surrogate model in BO open several promising avenues for future work:

Methodological improvements: This study focuses on EI as the acquisition function. However, it would be an important research direction to investigate TabPFN's performance with alternative acquisition functions, such as Upper Con dence Bound (UCB), or Probability of Improvement. Each of them has a di erent exploration-exploitation balance, and TabPFN's uncertainty quanti cation might interact di erently with each of them. Additionally, a potential limitation with our approach is that we are compressing TabPFN's multimodal posterior distributions into Gaussian assumptions for compatibility with BoTorch. Developing native acquisition functions that can fully exploit the multimodal distributions could provide better performance as surrogate model. A thorough investigation on computational resources and running times should be investigated as well to evaluate how the requirements scale for TabPFN against GPs.

Expanded benchmarking and evaluation: Benchmarking on di erent benchmarking frameworks would provide better validation of TabPFN's capabilities. EXPOBench is speci cally designed for expensive optimization scenarios that are more like real-world problems with mixed-variable types and constraints, and would provide insight on TabPFN's performance across those problems. Such studies should include alternative optimization algorithms to compare against, as presented in [BGK+23, SRLD24]. A more comprehensive HPO study investigating the tunable internal parameters of TabPFN would provide insight on how suitable TabPFN regression is as surrogate for BO.

High-dimensional optimization: Yu et al. (2025) showed that TabPFN can handle high-dimensional optimization when combined with gradient-informed subspace methods [YPA25]. Future work could combine their dimensional reduction approach with the function-speci c insights from this BBOB study. This might lead to TabPFN-based BO methods that work well in both low

and high dimensions.

References

[ADE+25] Sebastian Ament, Samuel Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. Unexpected improvements to expected improvement for bayesian optimization, 2025.

[BGK+23] Laurens Bliek, Arthur Guijt, Rickard Karlsson, Sicco Verwer, and Mathijs de Weerdt. Benchmarking surrogate-based optimisation algorithms on expensive black-box functions. Applied Soft Computing, 147:110744, 2023.

[DWY+18] Carola Doerr, Hao Wang, Furong Ye, Sander van Rijn, and Thomas Bäck. Iohprofiler: A benchmarking and profiling tool for iterative optimization heuristics. arXiv preprint arXiv:1810.05281, 2018.

[FLS05] Kai-Tai Fang, Runze Li, and Agus Sudjianto. Design and Modeling for Computer Experiments. Chapman and Hall/CRC, New York, 2005.

[Fra18] Peter I. Frazier. A tutorial on bayesian optimization. arXiv preprint arXiv:1807.02811, July 2018.

[HAR+20] Nikolaus Hansen, Anne Auger, Raymond Ros, Olaf Mersmann, Tea Tušar, and Dimo Brockhoff. Coco: a platform for comparing continuous optimizers in a black-box setting. Optimization Methods and Software, 36(1):114–144, 2020.

[HLHG14] José Miguel Hernández-Lobato, Matthew W. Hoffman, and Zoubin Ghahramani. Predictive entropy search for efficient global optimization of black-box functions, 2014.

[HMP+25] 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, 637:319–326, January 2025.

[JSW98] Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. Journal of Global Optimization, 13:455–492, 1998.

[MFHH23] Samuel Müller, Matthias Feurer, Noah Hollmann, and Frank Hutter. Pfns4bo: In-context learning for bayesian optimization, 2023.

[RP21] Rodolphe Le Riche and Victor Picheny. Revisiting bayesian optimization in the light of the coco benchmark. Structural and Multidisciplinary Optimization, 64:3063–3087, 2021.

[RW06] Carl Edward Rasmussen and Christopher K. I. Williams. Gaussian Processes for Machine Learning. The MIT Press, 2006.

[S+20] Agnieszka Slowik et al. Exploring bayesian optimization. Distill, 2020. States: 'We use a Matern 5/2 kernel due to its property of favoring doubly differentiable functions.'.

[SKKS12] Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias W. Seeger. Information-theoretic regret bounds for gaussian process optimization in the bandit setting. IEEE Transactions on Information Theory, 58(5):3250–3265, May 2012.

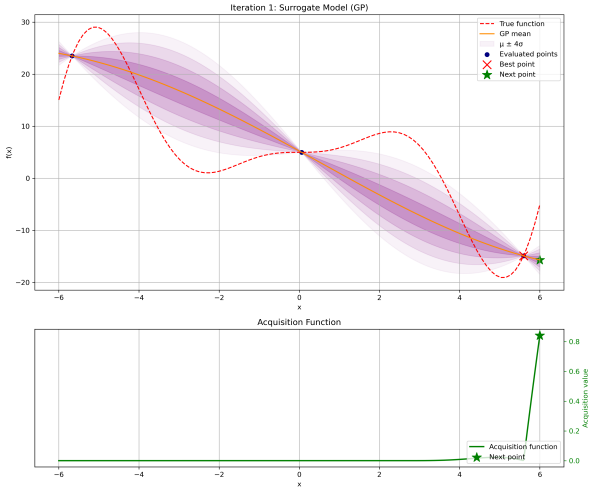
[SRLD24] Maria Laura Santoni, Elena Raponi, Renato De Leone, and Carola Doerr. Comparison of high-dimensional bayesian optimization algorithms on bbob, 2024.

[YPA25] Rosen Ting-Ying Yu, Cyril Picard, and Faez Ahmed. Git-bo: High-dimensional bayesian optimization with tabular foundation models, 2025.

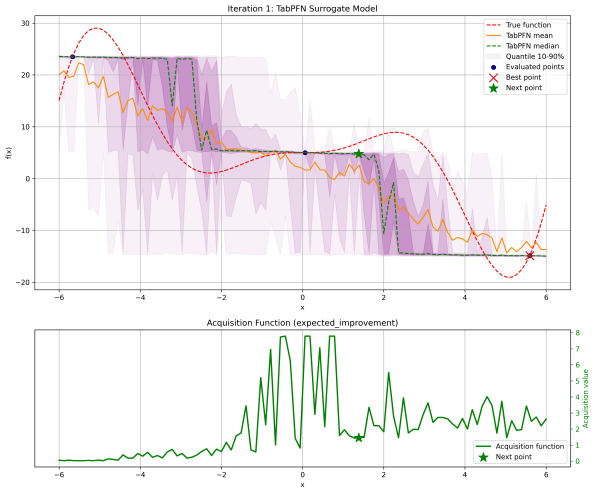
A Code for reproducibility

All the code can be found at https://github.com/Yuunjisan/TabPFN_VanillaGP_Benchmark.git.
The full TabPFN repository can be found at the public repository of PriorLabs https://github.com/PriorLabs/tabpfn.
The BBOB test suite can be found at https://coco-platform.org/testsuites/bbob/overview.html.

B Full iteration plots for GP-BO and TabPFN-BO on 1D problem



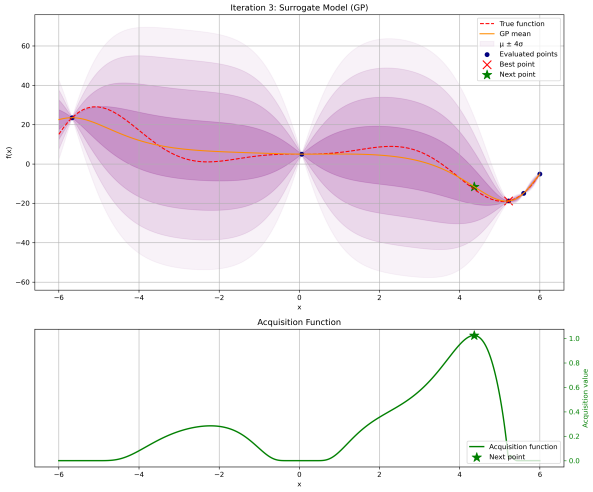
(a) GP iter 1



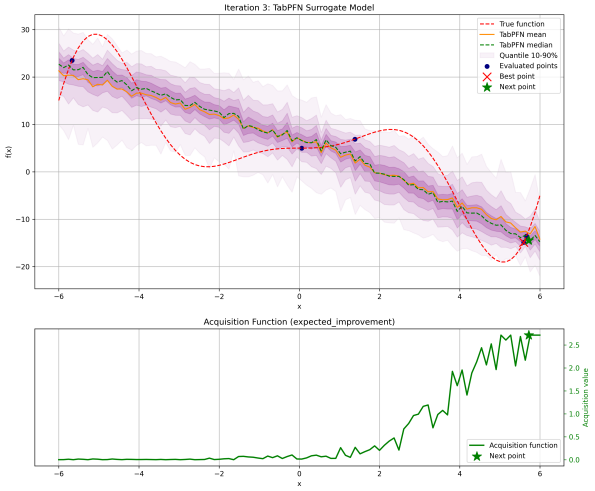
(b) TabPFN iter 1

(c) GP iter 2

(d) TabPFN iter 2



(e) GP iter 3

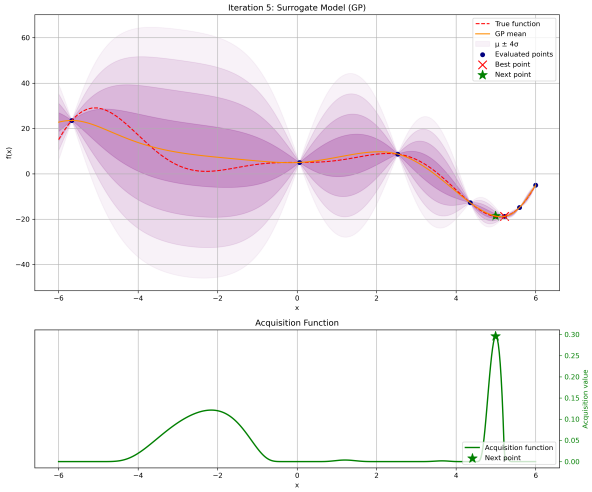


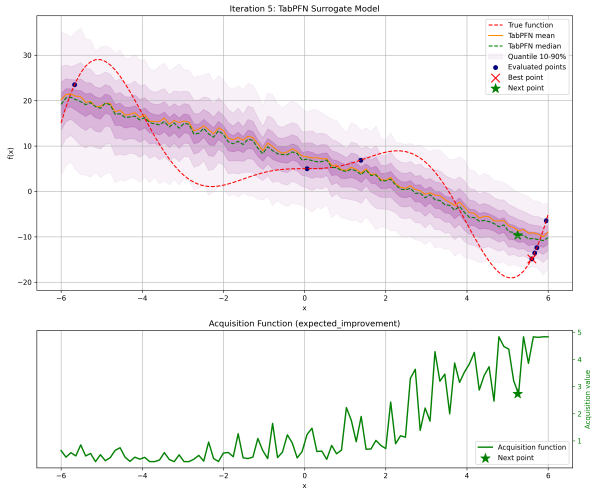
(f) TabPFN iter 3

Figure 32: GP vs TabPFN comparison: Iterations 1-3. Initial exploration and early model building phase.

(a) GP iter 4

(b) TabPFN iter 4





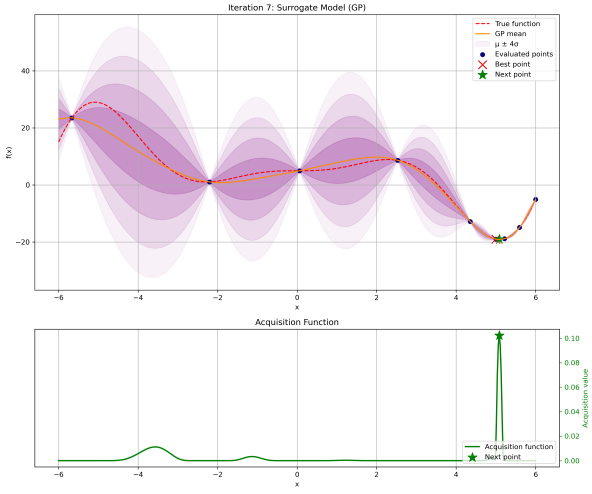
(c) GP iter 5

(d) TabPFN iter 5

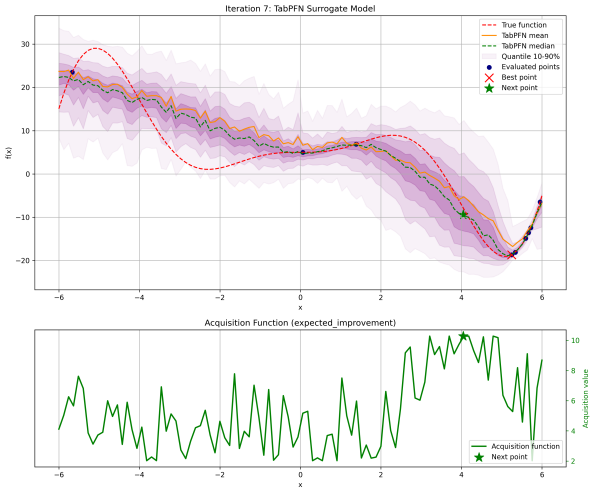
(e) GP iter 6

(f) TabPFN iter 6

Figure 33: GP vs TabPFN comparison: Iterations 4-6. Continued exploration and model refinement.



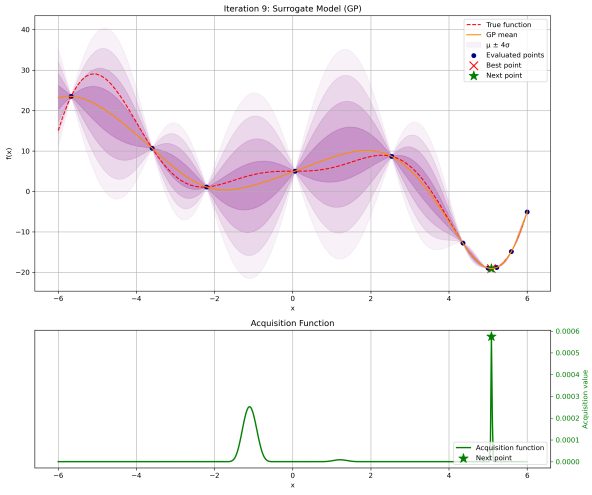
(a) GP iter 7



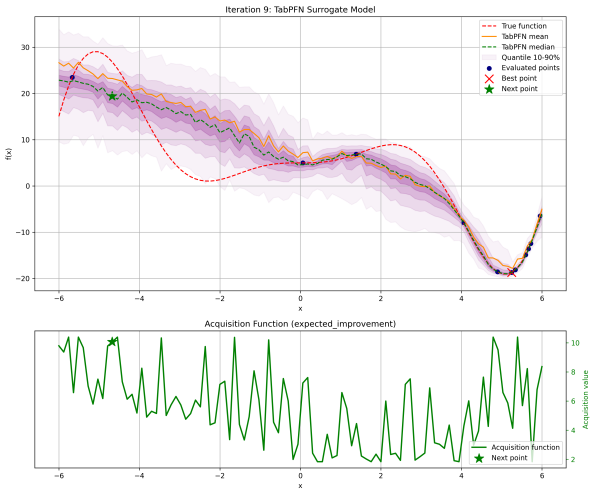
(b) TabPFN iter 7

(c) GP iter 8

(d) TabPFN iter 8



(e) GP iter 9

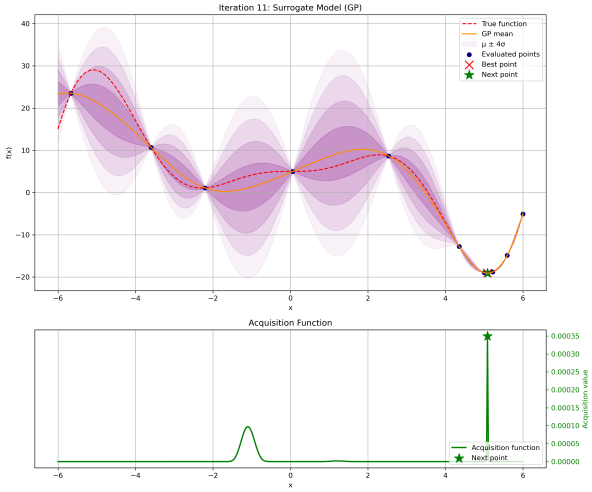


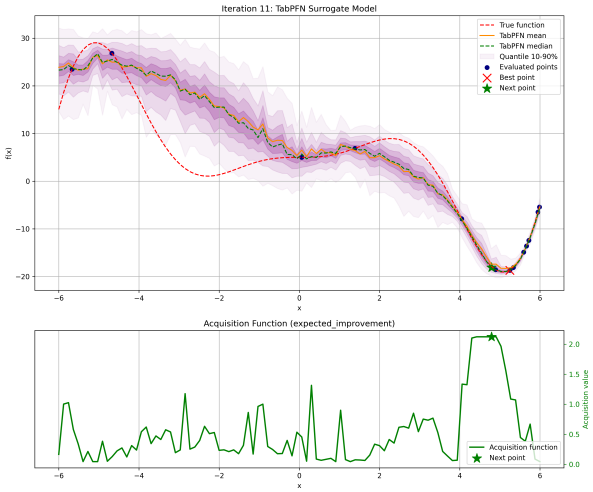
(f) TabPFN iter 9

Figure 34: GP vs TabPFN comparison: Iterations 7-9. Mid-phase optimization progression.

(a) GP iter 10

(b) TabPFN iter 10





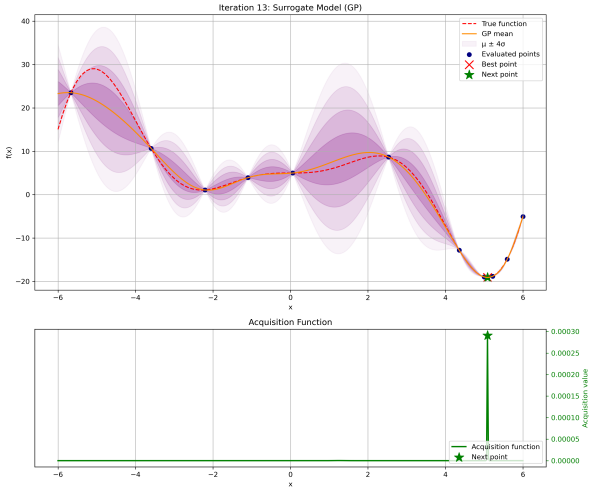
(c) GP iter 11

(d) TabPFN iter 11

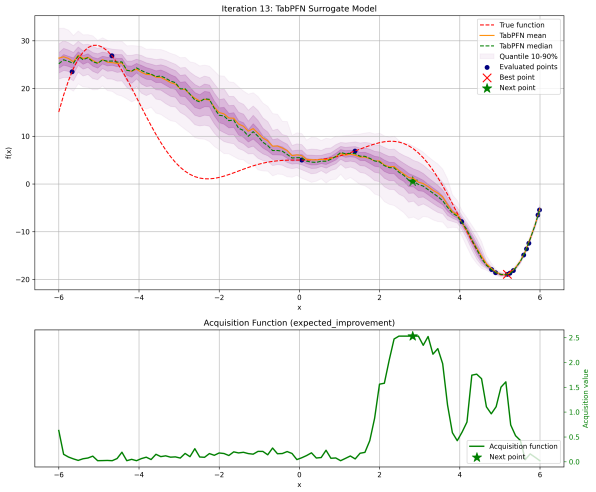
(e) GP iter 12

(f) TabPFN iter 12

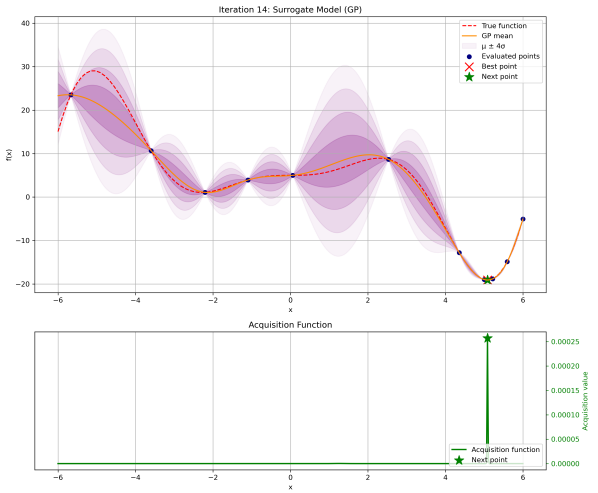
Figure 35: GP vs TabPFN comparison: Iterations 10-12. Later optimization phase with increased model confidence.



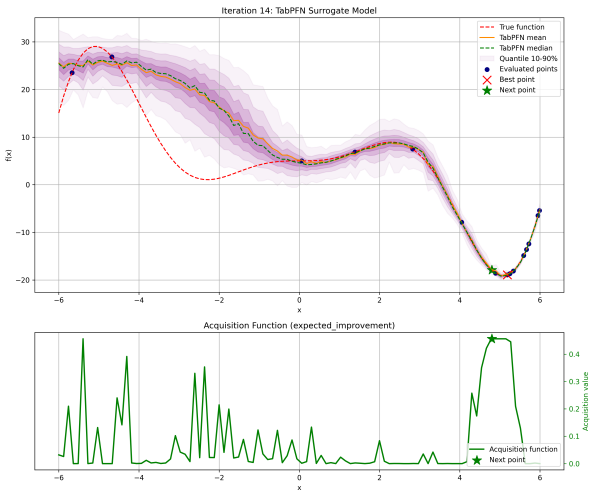
(a) GP iter 13



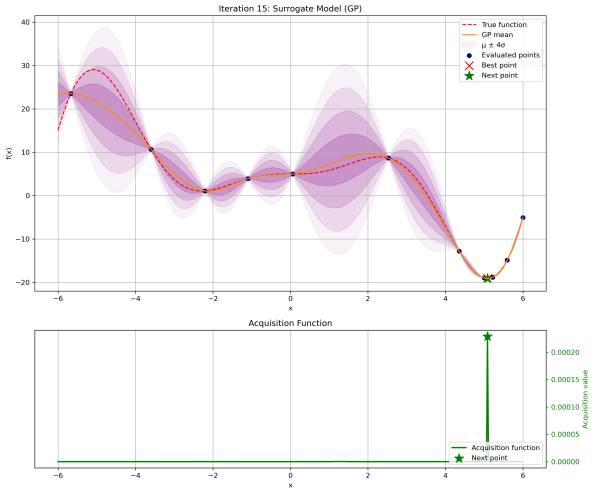
(b) TabPFN iter 13



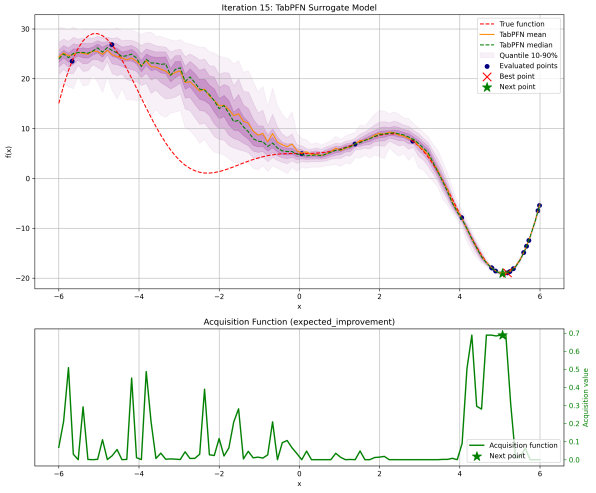
(c) GP iter 14



(d) TabPFN iter 14



(e) GP iter 15



(f) TabPFN iter 15

Figure 36: GP vs TabPFN comparison: Iterations 13-15. Advanced optimization with fine-tuning behavior.