TOWARD PRIVILEGED FOUNDATION MODELS: LUPI FOR ACCELERATED AND IMPROVED LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Training foundation models is computationally intensive and often slow to converge. We introduce PIQL[1], Privileged Information for Quick and Quality Learning, the first framework to systematically integrate privileged information (PI) to simultaneously accelerate learning and improve generalization in tabular foundation models (TFMs). We construct two complementary forms of PI: (i) aggregate dataset-level statistics that reduce the burden on in-context learning, and (ii) encodings of the underlying data-generating program, providing knowledge beyond observable data. We further design an architecture that effectively transfers the train-time-only PI by learning to reconstruct it from observed context at inference. We provide a theoretical analysis characterizing conditions under which PI reduces the population-level approximation gap and accelerates convergence in finite-data regimes. Empirical evidence shows that PIQL enables TFMs to achieve faster convergence, lower final loss, and better generalization, in effect, reducing data and compute requirements. Our work establishes PI-guided pretraining as a principled and practical paradigm for improving the efficiency and performance of foundation models.

1 INTRODUCTION

Training foundation models (FMs) poses a formidable learning problem. The combination of massive, heterogeneous datasets and high-dimensional parameter spaces substantially slow down convergence, often requiring weeks of intensive computation to achieve robust generalization.

Motivated by the insight that effective guidance can dramatically improve learning efficiency, as captured by "*Better than a thousand days of self-instruction is one day with a great teacher*" (Vapnik & Izmailov, 2015), we explore a new way to train foundation models (FMs): using a "teacher" to provide useful auxiliary information during training. We focus on tabular foundation models (TFMs), although the same idea can extend to other modalities.

Our goal for teacher-guided TFMs is twofold: **(1)** to accelerate convergence (i.e., learning rate), thereby reducing pretraining cost; and **(2)** to improve asymptotic generalization, which is central to downstream TFM performance on new tasks. To this end, we explore a **new paradigm for learning tabular foundation models** in which additional information is provided to the learner during training.

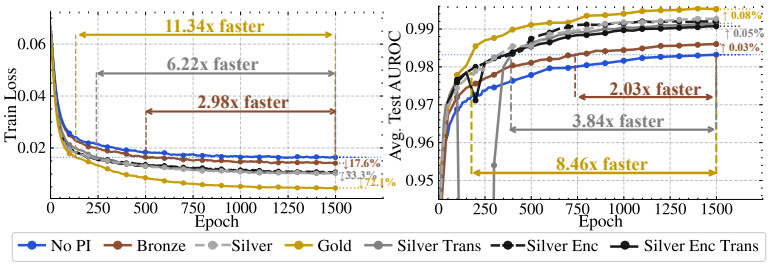
This raises a key question: *Does there exist auxiliary information that can systematically improve both learning speed and generalization?* We posit two complementary mechanisms for constructing such information: **(i)** targeting the in-context learning (ICL) capacity of the backbone Transformer, and **(ii)** leveraging prior knowledge of the data-generating process.

First, TFMs perform ICL by using training examples as context, with each layer contributing a computational step toward prediction Garg et al. (2022); Von Oswald et al. (2023). We provide *aggregate dataset-level statistics* as auxiliary inputs, giving direct access to global properties and freeing ICL capacity to increase effective depth. Second, TFMs are pretrained on synthetic datasets from known priors (Hollmann et al., 2023; 2025; Shen et al., 2025; Zhang et al., 2025b; Ding et al., 2026b), so the true generating process is available during pretraining. Instead of discarding it after sampling, we provide this *generative information* as auxiliary input, supplying knowledge beyond what can be inferred from observed data alone.

Fig. 1 presents striking results from a proof-of-concept study (Sec. 3), demonstrating the effectiveness of this new paradigm: progressively richer forms of **teacher-guided foundation models achieve**

[1]All source code and checkpoints are open at `https://anonymous.4open.science/r/PIQL`

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



	Prior Type		GMM #Clus.	
L	No-PI	PIQL	No-PI	PIQL
9	0.789	0.996	0.609	0.655
8	0.788	0.985	0.606	0.625
7	0.768	0.981	0.547	0.556
6	0.750	0.976	0.502	0.536
5	0.748	0.975	0.472	0.537
4	0.739	0.976	0.465	0.569
3	0.734	0.981	0.450	0.542
2	0.727	0.977	0.418	0.564
1	0.713	0.973	0.398	0.554
0	0.687	0.740	0.363	0.457

Figure 1: Privileged TFMs pretrained on a GMM-only prior and given bronze- (empirical dataset-level statistics), silver- (theoretical model-wise means and variances), or gold-grade (theoretical point-wise means and variances) PI outperform the no-PI baseline. **(left)** PI enables *faster learning and lower asymptotic loss*, with gains increasing from bronze to gold. **(middle)** PI also improves sample and compute efficiency, reaching strong in-distribution performance in fewer epochs. **(right)** PI increases effective depth by *freeing ICL capacity*: in a 10-layer TFM trained on mixed priors, PIQL yields linearly probe-able representations of prior type and GMM cluster count in earlier layers than the no-PIQL baseline. (best in color)

faster learning and lower asymptotic loss, reducing both data and computational requirements. Further, they produce representations that capture data distributional properties in earlier layers.

Our approach is grounded in the Learning Using Privileged Information (LUPI) framework (Vapnik & Vashist, 2009), which formalizes learning with auxiliary information available only during training (only). While prior theory characterizes when the so-called privileged information (PI) can accelerate learning Pechyony & Vapnik (2010), it does not provide constructive design principles. As a result, the design of effective PI remains a largely empirical quest[2] We address this gap by proposing concrete, novel forms of PI tailored to TFMs. The following summarizes our main contributions.

- **Novel Training Paradigm for TFMs:** We introduce PIQL, leveraging Privileged Information for Quick and Quality Learning, the first PI-guided framework for foundation model training toward both accelerated and improved learning.
- **Constructing Privileged Information (PI) for TFMs:** Constructing effective PI remains an empirical challenge. We design two novel forms of PI for TFMs: **(i)** aggregate dataset-level statistics that alleviate the demand on the model's ICL capacity (available during both training and inference), and **(ii)** encoding of the underlying data-generating program that provides guidance beyond what can be inferred from in-context observations alone (train-time-only).
- **Designing Privileged TFM Architecture:** We design a new architecture that transfers the train-time-only PI (program encoding) by learning to reconstruct it from observable input during inference, trained with annealed teacher forcing, gradually replacing the original with estimated PI.
- **Theoretical Analysis:** We establish the theoretical potential of PIQL for TFMs by characterizing the conditions under which PI reduces the population-level approximation gap between standard and PI-augmented hypothesis classes. We further analyze finite-data regimes, quantifying the rate at which models realize PI-induced gains in practice.
- **Empirical Effectiveness:** We support our theoretical analysis under idealized assumptions with empirical evidence, showing that our proposed PI mechanisms and architecture for TFMs yield faster convergence, lower final loss, and improved generalization.

2 Preliminaries

2.1 Learning Using Privileged Information (LUPI)

The LUPI paradigm (Vapnik & Vashist, 2009) introduces a "teacher" that provides **additional training-only information**. The learner receives triplets $(\mathbf{x}_i, \mathbf{x}_i^*, y_i)_{i=1}^n$, where $\mathbf{x}$ is the primary input **given by nature** and $\mathbf{x}^*$ is privileged information (PI) **constructed by the teacher**. PI acts as correcting information, helping the learner model sample difficulty and errors to learn better decision boundaries and generalize more efficiently. The learner chooses the hypothesis spaces $\mathcal{H}$ and Φ for the decision and correcting functions, respectively. While classical supervised learning has been theoretically shown to often converge at a rate of $O(n^{-1/2})$, a teacher that explains structure in the

[2]Empirical work demonstrated compelling examples Vapnik & Vashist (2009); where pairing handwritten digit images with poetic language descriptions as PI for digit prediction, and supplementing protein sequences with their 3D structure information as PI for protein-family classification improved learning rates.

data via PI can accelerate learning to a faster rate of $O(n^{-1})$ under certain conditions (Pechyony & Vapnik, 2010); effectively reducing the complexity of the learning problem.

The utility of LUPI has been shown in several settings where PI provides high-level structural context (Vapnik & Vashist, 2009; Vapnik & Izmailov, 2017). In protein function prediction, **3D molecular coordinates** complement amino acid sequences and improve sequence-only models. In MNIST, **human-written poetic descriptions** of digit style, such as stroke dynamics and visual rhythm, help identify structural invariants. In medical diagnosis, **expert descriptions and biopsy interpretations** complement biometric measurements and improve diagnostic performance.

2.2 Tabular Foundation Models (TFMs)

Tabular Foundation Models (TFMs). TFMs extend the foundation-model paradigm to tabular data by pretraining on many *synthetic* tasks sampled from data-generating priors, such as Structural Causal Models (SCMs) (Hollmann et al., 2023; 2025), Gaussian Mixture Models (GMMs), and Copulas (Shen et al., 2025; Ding et al., 2026b). Through pretraining, TFMs learn to approximate Bayesian inference across tasks. At inference, they take a labeled set as context and predict labels for new inputs in a *single forward pass*, without dataset-specific training (Xie et al., 2021; Falck et al., 2024).

Posterior Predictive Distribution (PPD). Let $\psi \in \Psi$ denote a data-generating hypothesis drawn from prior $p(\psi)$, and let $\mathcal{D}_{\text{train}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ be the observed training set. Bayesian prediction for a new input $\mathbf{x}_{\text{test}}$ marginalizes over the posterior distribution of hypotheses:

$$p(y_{\text{test}} \mid \mathbf{x}_{\text{test}}, \mathcal{D}_{\text{train}}) = \int_\Psi p(y_{\text{test}} \mid \mathbf{x}_{\text{test}}, \psi) p(\psi \mid \mathcal{D}_{\text{train}})\, d\psi, \tag{1}$$

where $p(\psi \mid \mathcal{D}_{\text{train}}) \propto p(\mathcal{D}_{\text{train}} \mid \psi) p(\psi)$.

PFNs for PPD approximation. Because the PPD is generally intractable, Prior-data Fitted Networks (PFNs) approximate it by training on synthetic datasets sampled from the prior (Müller et al., 2022).

During **pretraining**, a hypothesis $\psi \sim p(\psi)$ is sampled, followed by a dataset $\mathcal{D} \sim p(\mathcal{D} \mid \psi)$. The dataset is split into a context set $\mathcal{D}_{\text{train}}$ and a query set $\mathcal{D}_{\text{test}}$. A PFN $h_{\boldsymbol{\theta}}$ learns to predict query labels from the context by minimizing expected negative log-likelihood:

$$\mathcal{L} = \mathbb{E}_{(\{(\mathbf{x}_{\text{test}}, y_{\text{test}})\} \cup \mathcal{D}_{\text{train}}) \sim p(\mathcal{D})} \big[-\log h_{\boldsymbol{\theta}}(y_{\text{test}} \mid \mathbf{x}_{\text{test}}, \mathcal{D}_{\text{train}}) \big]. \tag{2}$$

During **inference**, given a new context set $\mathcal{D}_{\text{train}}$, the TFM outputs $h_{\boldsymbol{\theta}}(\cdot \mid \mathbf{x}_{\text{test}}, \mathcal{D}_{\text{train}})$ for each query in a single forward pass, relying entirely on in-context learning.

2.3 Notation and Problem Statement

The theory of LUPI does not offer constructive principles for designing privileged information (PI), therefore the quest for PI remains largely empirical. As such, the first problem we address is how to construct useful PI, denoted $\mathbf{P}^*$, for training TFMs.

Problem 1 (PI Construction). *Given datasets $\mathcal{D}_X = (\mathbf{X}, \mathbf{y})$ drawn from data priors; Construct privileged tokens $\mathbf{P}^*$ to prepend to the context window, where $\mathbf{P}^* = (\mathbf{P}_a^* \| \mathbf{P}_u^*)$ is composed of $\mathbf{P}_a^*$ available during both pretraining and inference, and $\mathbf{P}_u^*$ that is only available for training.*

The second problem is how to effectively train with the PI, and in particular, how to leverage $\mathbf{P}_u^*$ that is only available during training.

Problem 2 (PI Transfer). *Given the pretraining corpus $\{\big(\mathbf{P}^{*(i)}, \mathbf{X}_{\text{c}}^{(i)}, \mathbf{y}_{\text{c}}^{(i)} \mid \mathbf{X}_{\text{q}}^{(i)}\big)\}_{i=1}^N$; during training: 1. Estimate teacher $\phi \in \Phi : \big(\mathbf{P}_u^*, \mathbf{P}_a^*, \mathbf{X}_{\text{c}}, \mathbf{y}_{\text{c}} \mid \mathbf{X}_{\text{q}}\big) \mapsto \mathbf{y}_{\text{q}}$ that ingests privileged context-query pairs and maps to query labels, and 2. Estimate transfer model $s : \big(\mathbf{P}_a^*, \mathbf{X}_{\text{c}}, \mathbf{y}_{\text{c}}\big) \mapsto \mathbf{Z}_{\text{u}}^*$ that ingests inference-available context-query pairs and maps to intermediate teacher representations $\mathbf{Z}_{\text{u}}^*$ of $\mathbf{P}_u^*$, i.e. inference-unavailable privileged context.*

Note that we jointly optimize the teacher ϕ and the transfer model s end-to-end during pretraining, gradually transitioning ϕ to h that replaces the true $\mathbf{P}_u^*$ with the student-*predicted* $\widehat{\mathbf{P}}_u^*$. At inference, the learner $(h, s) \in \mathcal{H}$ is operates on new train-test pairs using available PI only by employing (1) s on $\big(\mathbf{P}_a^*, \mathbf{X}_{\text{tr}}, \mathbf{y}_{\text{tr}}\big)$ to obtain $\widehat{\mathbf{Z}}_{\text{u}}^*$ and then (2) h on $\big(\widehat{\mathbf{Z}}_{\text{u}}^*, \mathbf{P}_a^*, \mathbf{X}_{\text{tr}}, \mathbf{y}_{\text{tr}} \mid \mathbf{X}_{\text{test}}\big)$ to predict $\widehat{\mathbf{y}}_{\text{test}}$.

2.4 Related Work Overview

Our work extends tabular foundation models (TFMs) with privileged information (PI) by prepending PI tokens to the context. This may resemble **prefix tuning** (Li & Liang, 2021), **prompt optimization**

(Lester et al., 2021; Chen et al., 2025), **instruction tuning** (Wei et al., 2022; Mishra et al., 2022), **personalization** (Adomavicius & Tuzhilin, 2005; Ricci et al., 2010), **conditional generation** (Mirza & Osindero, 2014; Keskar et al., 2019; Ramesh et al., 2021; Rombach et al., 2022), and **auxiliary-data prediction** (Lim et al., 2021; Zhang et al., 2025a; Radford et al., 2021; Kim et al., 2021; Li et al., 2023); see Appdx. A and Table 7 for a comparison to our work.

In a nutshell, these paradigms all use information beyond the raw input, but differ in its nature, availability, and role. In particular, **none is designed for TFMs or assumes a *train-time-only* information regime**. In contrast, our framework follows LUPI, where part of the PI is available only during training. Rather than serving as persistent conditioning or task-defining input, our PI acts as a *transient* learning signal that improves TFM learning efficiency and generalization. Specifically, our PI consists of dataset meta-features and generator program specifications that encode latent structure in the data distribution.

3 A Proof-of-Concept for Privileged TFMs

In this section, we demonstrate the premise of leveraging privileged information (PI) in training TFMs using a simple scenario. We consider a TFM for outlier detection, namely FoMo-0D (Shen et al., 2025), which is trained on datasets synthesized solely from a data prior based on Gaussian Mixture Models (GMMs). We construct three different PI of varying strength, so-called **bronze**-, **silver**- and **gold**-grade PI, and show their impact on training loss as well as generalization performance in-distribution, i.e. on hold-out GMM datasets from the data-generating prior.

For each pretraining dataset, inliers are drawn from a GMM with up to $M{=}5$ components in up to $D{=}100$ dimensions, with centroids $\boldsymbol{\mu}_j^{(k)} \in [-5, 5]$ and diagonal covariance $\boldsymbol{\Sigma}_{jj}^{(k)} \in (0, 5]$ for $k \in [M]$ and $j \in [D]$. Outliers are drawn from the "inflated" GMM: each Gaussian's variance is scaled by s' within a random subspace $\mathcal{S}^{(k)}$ as $s' \times \Sigma_{jj}^{(k)}$ for $j \in \mathcal{S}^{(k)}$ while keeping the centroids as before.

- **Bronze**-grade PI consists of **(empirical) aggregate statistics at the *dataset-level***. Specifically, we compute first- to fourth-order moment statistics (mean, variance, skewness and kurtosis) of each feature across all points in $\mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{test}}$ (without access to any labels). The four resulting tokens in $\mathbb{R}^D$ are prepended to the context along with the points in $\mathcal{D}_{\text{train}}$.
- **Silver**-grade PI directly provides the **(theoretical) mean and variance for *each Gaussian*** that is used to generate inliers and outliers. Let $(\boldsymbol{\mu}, \boldsymbol{\sigma}^2_{\text{in}})$ denote the parameters of a Gaussian that generates inliers, and $(\boldsymbol{\mu}, \boldsymbol{\sigma}^2_{\text{out}})$ denote those of its inflated version that generates outliers. Then, we prepend the context with two tokens in $\mathbb{R}^D$ per Gaussian of the form $(\boldsymbol{\mu}, 1/2\boldsymbol{\sigma}^2_{\text{in}} + 1/2\boldsymbol{\sigma}^2_{\text{out}})$.
- **Gold**-grade PI provides the **(theoretical) mean and variance vector for *each individual point*** from which they were drawn. Specifically, we concatenate each inlier point $\mathbf{x}_i \sim \mathcal{N}(\boldsymbol{\mu}^{(k)}, \boldsymbol{\sigma}^{2,(k)}_{\text{in}})$ as $(\mathbf{x}_i \| \boldsymbol{\mu}^{(k)} \| \boldsymbol{\sigma}^{2,(k)}_{\text{in}}) \in \mathbb{R}^{3D}$. Outliers are also concatenated with the *original* parameters of their respective Gaussian (and *not* the inflated variances).

Remarks: PI strength increases with grade. **Bronze** PI provides empirical dataset-level feature moments, which can also be computed at inference. **Silver** PI provides the true GMM parameters, available only for synthetic pretraining data and difficult to recover exactly at inference; standard estimation via EM is iterative and not guaranteed to recover the ground-truth parameters (Dempster et al., 1977). GMMs contain both global mixture parameters and point-level component assignments: Silver provides the former, while **Gold** additionally provides the latter, including each point's component membership and corresponding Gaussian parameters.

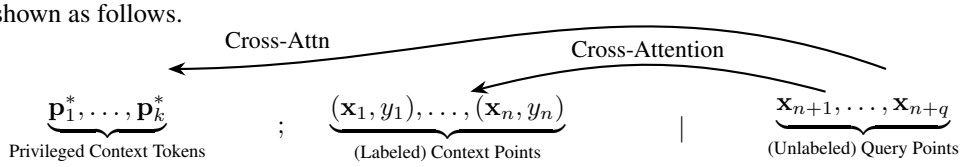
Transferring PI: Because silver- and gold-grade PI contain theoretical GMM latents, they are available only during pretraining and are difficult to recover at inference from limited data and depth. We therefore design a novel transformer architecture that reconstructs this PI from inference-available representations. We defer the architecture and training details to Sec. 4 and first show compelling results in the simplified GMM-only setting.

Results: Fig. 1 shows (left) training loss and (right) average in-distribution AUROC over epochs for the no-PI baseline (blue) and models with bronze- (brown), silver- (dashed gray), and gold-grade (yellow) PI. For the train-time-only silver and gold PI, we provide PI at inference as a proof of concept. We also include our *transferred* silver model (gray), which reconstructs PI from inference-available inputs and achieves comparable performance, demonstrating effective PI transfer during pretraining. Also shown are models using generator program encodings as PI (silver_enc; dashed black) and their transferred counterpart (black), both yielding gains similar to silver-grade PI. Program-encoding

details are deferred to Sec. 4. Strikingly, **privileged training accelerates loss reduction and lowers asymptotic loss**, with gains increasing from bronze to gold. Privileged models also reach strong in-distribution generalization earlier, requiring fewer epochs, and thus less data and compute. These results highlight the promise of privileged foundation models for improving training efficiency, convergence, and generalization.

4 PIQL: Privileged Information for Quick and Quality Learning for TFMs

We consider learning under privileged information (PI) for tabular foundation models (TFMs), where the labeled training points in the context window are prepended with additional tokens capturing the PI, shown as follows.



We introduce two key innovations: (1) construction of two novel PIs for TFMs (Sec. 4.1), and (2) a new privileged TFM architecture and pretraining algorithm to effectively accommodate train-time-only PI (Sec. 4.2). This section presents how to *operationalize* our innovations, followed by a theoretical analysis in the next section.

4.1 Constructing Privileged Information for TFMs

The LUPI framework provides little guidance for designing PI. However, our preliminary results show that PI quality matters, with clear performance differences across grades.

Toward PI-equipped TFMs, we introduce two complementary forms of information: **meta-features** and **generator programs**, differing in availability and role. **Meta-features**, available at both training and inference, provide high-level dataset statistics directly in context, reducing the burden on ICL to infer them. **Generator programs**, available only during training, exploit the synthetic-data setting where the true data-generating prior and parameters are known. As strict LUPI, they expose the underlying generative process, guiding representation learning and generalization beyond what can be inferred from finite in-context samples alone.

Aggregate data statistics (**meta-features**) can be directly added as privileged prefix tokens. **Generator programs** are more challenging because different data priors have different structures: Sec. 3 used theoretical means and variances for GMMs, while SCMs and Copulas require different representations. To unify them, we introduce a **program encoder** that maps each generator's textual–numeric specification into learned prefix representations that can be prepended to the context.

4.1.1 Meta-Features (weak PI, available at test time)

Meta-PI consists of dataset-level summary statistics computed directly from the input data and available at both training and inference. We prepend these statistics as prefix tokens, giving the TFM explicit access to global distributional information instead of requiring it to reconstruct such properties through in-context learning (ICL). By offloading this computation, meta-PI can free representational and computational capacity, effectively increasing the model's usable depth and improving downstream performance.

We use simple moment- and quantile-based summaries of the feature distributions: the first four moments (mean, variance, skewness, and kurtosis) and five quantiles (5, 25, 50, 75, and 90). Computed across samples for each feature, these form nine prefix tokens $\mathbf{p}_1^*, \ldots, \mathbf{p}_9^* \in \mathbb{R}^D$.

These statistics compactly characterize key aspects of a distribution, including its center, spread, asymmetry, tail behavior, and mass across the range (van der Vaart, 1998). Such information can be particularly useful for outlier detection, where mean, variance, quantiles, and tail statistics provide a reference for identifying atypical observations. Our goal is not to exhaustively search over possible meta-features, but to show that even simple statistical summaries can provide meaningful gains. We leave broader meta-feature design to future work.

Table 1: Meta-feature vs. Generator program based privileged information (PI) for TFMs.

Type	Availability	Architecture	Role
Meta-feature (weak PI)	Train & Test	Prefix tokens	Relieve ICL capacity
Generator program (strong PI)	Train only	Embd. transfer	Expose true generative model

4.1.2 GENERATOR PROGRAM (STRONG PI, TRAIN-TIME ONLY)

Unlike large language models pretrained on massive *real-world* corpora, TFMs occupy a unique setting in which training data are generated *synthetically*.This provides privileged access to the true generative process behind each sampled dataset. However, existing TFMs largely discard this information once the training data are generated. **Our work is the first to recognize synthetic data priors as a rich source of information and leverage them within the formal LUPI framework**, where the generator program serves as PI available during training but not at inference, as TFMs are deployed on real-world tasks downstream.

Earlier TFMs typically used a single prior family, such as SCMs (Hollmann et al., 2023) or GMMs (Shen et al., 2025), while recent TFMs train on mixtures of heterogeneous priors (Zhang et al., 2025b; Ding et al., 2026b), including SCMs, GMMs, Copulas, and tree-based generators. These priors have very different specifications; for example, means and covariances for GMMs, causal graphs and structural equations for SCMs, and marginals and dependencies for Copulas. The **central challenge** is therefore to map these heterogeneous generator specifications into a shared token space.

To this end, we introduce a **program encoder** encompassing two key innovations: First, we propose a serialization format and tokenization that presents a generative specification—the program, its structure, and its parameter values—as a *structured sequence*, analogous to a domain-specific language (DSL), while *respecting the numeric values* of the parameters. Second, we encode this sequence using a Transformer, prepending k learnable `[PREFIX]` tokens to the input. Through self-attention, these tokens accumulate information from the entire generator specification; e.g.:

$$\underbrace{[\mathbf{p}_1]\ [\mathbf{p}_2]\ \cdots\ [\mathbf{p}_k]}_{\text{learnable prefix tokens}}\ \underbrace{[\texttt{FAMILY:GMM}]\ [\texttt{DIM:5}]\ [\texttt{N_COMP:3}]\ [\texttt{COMP}]\ [\texttt{W:0.4}]\ \cdots}_{\text{serialized generator specification}}$$

The program encoder is trained on numerous programs from the heterogeneous data priors with varying configurations of the parameters using contrastive loss. Given a generator configuration $\mathcal{G}$, we construct a positive pair by slightly perturbing its parameters as $\mathcal{G}' = \mathrm{perturb}(\mathcal{G}, \epsilon)$, and encode both as $\mathbf{z} = \mathrm{pool}\big(\mathrm{Enc}(\mathrm{serialize}(\mathcal{G}))\big)$, and $\mathbf{z}' = \mathrm{pool}\big(\mathrm{Enc}(\mathrm{serialize}(\mathcal{G}'))\big)$. We perform hard negative mining and employ a curriculum to improve contrastive learning. Appdx. B provides additional details on serialization, tokenization, and encoder training, while Table 8 lists the configurations.

Program encoder training precedes the TFM pretraining. Upon its training, each pretraining generator program $\mathcal{G}$ is serialized and passed through the program encoder, which provides to the TFM the first k output tokens as the **Generator-PI** prefix tokens; capturing a compact, fixed-size representation of the generator, regardless of the underlying family or dimensionality. That is,

$$\{\mathbf{p}_1^*, \ldots, \mathbf{p}_k^*\} = \mathrm{Enc}\big(\,\mathrm{serialize}(\mathcal{G})\big)\ .$$

Table 1 summarizes the two PI types for TFMs. Since the generator program is train-time-only PI, we propose a transfer-based TFM architecture trained with annealed teacher forcing, where *estimated* Generator-PI gradually replaces the original, as described next.

4.2 DESIGNING AND TRAINING PRIVILEGED TFM ARCHITECTURES

A Teacher–Student TFM: Let a TFM be $h : (\mathcal{D}_{\text{context}}, \mathbf{x}_{\text{query}}) \mapsto y_{\text{query}}$. For a privileged TFM, the pretraining context is $\mathcal{D}_{\text{context}} = (\mathbf{P}_u^*, \mathbf{P}_a^*, \mathcal{D}_{\text{train}})$, where $\mathbf{P}_u^*$ are train-only generator-program tokens, $\mathbf{P}_a^*$ are inference-available meta-feature tokens, and $\mathcal{D}_{\text{train}}$ contains the training points. We augment h with a student

$$st : (\mathbf{P}_a^*, \mathcal{D}_{\text{train}}) \mapsto \widehat{\mathbf{P}}_u^*,$$

which learns to reconstruct the unavailable PI from inference-available context. At inference, (h, st) first predicts $\widehat{\mathbf{P}}_u^*$ with st, then h predicts using $(\widehat{\mathbf{P}}_u^*, \mathbf{P}_a^*, \mathcal{D}_{\text{train}})$.

Training to Transfer PI: Our goal is for (h, st), using only inference-available inputs, to approach the teacher

$$\phi : (\mathbf{P}_u^*, \mathbf{P}_a^*, \mathbf{X}_{\mathrm{c}}, \mathbf{y}_{\mathrm{c}} \mid \mathbf{X}_{\mathrm{q}}) \mapsto \mathbf{y}_{\mathrm{q}},$$

which has access to all PI. However, replacing true $\mathbf{P}_u^*$ with estimated $\widehat{\mathbf{P}}_u^*$ introduces a train–inference mismatch, or *exposure bias*. To mitigate this, we use scheduled sampling (Bengio et al., 2015): during pretraining, h conditions on either the true $\mathbf{P}_u^*$ or student prediction $\widehat{\mathbf{P}}_u^*$, with probability α_t annealed over training. Thus, true PI is progressively replaced by student-estimated PI, preparing the model for inference. Architecture and training details are provided in Appdx. C, with an overview in Fig. 4.

5 Theoretical Analysis of PI

Following the LUPI perspective, our analysis identifies usefulness, non-redundancy, and transferability as conditions for PI to help. **I: Representation improvement via meta-features.** Meta-PI consists of deterministic dataset-level statistics $\mathcal{P}_a^* = \mathbf{s}(\mathcal{D})$. It adds no information beyond the observed context, but can provide a computational shortcut for a fixed-depth Transformer. Explicitly supplying useful summaries may reduce the burden of reconstructing them through in-context computation. The conditional analysis associated with Theorem D.1 examines this architectural bottleneck. The meta-PI experiments and layer-wise probes provide empirical evidence consistent with this mechanism. **II: Information improvement via generator-PI.** Generator-PI exposes latent properties of the data-generating process that may be difficult to identify from finite observations. Such information can guide pretraining when it is relevant to prediction; uncertainty about the generator alone does not imply a predictive benefit. The oracle setting is considered in Theorem D.2. Its relevance to deployment additionally depends on how much of the oracle advantage survives when true PI is replaced by student-reconstructed PI.

These conditions motivate our controls: parameter-/token-matched and random-PI comparisons assess the contribution of PI content, while oracle-versus-student comparisons, reconstruction error, and annealing ablations assess transfer. The finite-sample analysis in Appdx. D.2 provides qualitative guidance rather than a prediction of measured training speedup.

6 Experiments

TFM Backbone, Data, Metrics: Our proof-of-concept study (Sec. 3) considered GMM-only data prior and a 2-layer Transformer backbone. Having developed a unified program-as-PI representation across heterogeneous priors, we pretrain 10-layer outlier detection TFMs on a mixture of five (5) different priors (Ding et al., 2026b). We also scale PIQL to a 14-layer, approximately 150M-parameter TFM. Test set includes 250×5 datasets sampled across these priors, as well as (**57**) datasets from the real-world benchmark ADBench (Han et al., 2022) and (**690**) datasets from OddBench (Ding et al., 2026a). We track and report training (cross-entropy) loss, as well as AUROC and AUPRC performances, over epochs. We perform paired permutation tests to statistically compare two methods and report the p-value. See Appdx. E.1 for details.

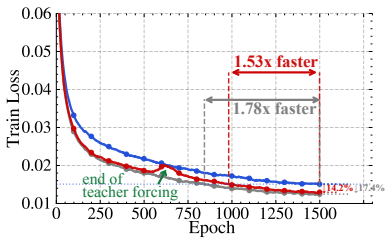
6.1 Main Results

Fig. 2 shows the training loss (left) and average test AUROC (right) across epochs for the PIQL-enabled TFM and the no-PI baseline. Consistent with the proof-of-concept results in Fig. 1, privileged TFMs learn more rapidly, reaching lower training loss and higher AUROC in earlier epochs, and achieve improved asymptotic performance. Besides the PIQL'ed model that transfers PI (unified program encoding) during training (red), Fig. 2 shows the variant trained without transfer (gray). While the training dynamics differ (e.g., loss bumps slightly when teacher forcing is annealed), PIQL reaches competitive loss and performance at convergence. Appdx. E.2 presents the paired tests.

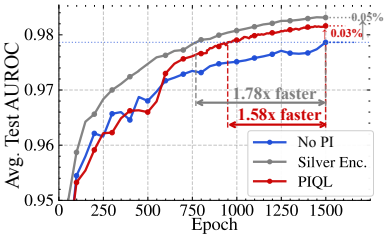
We report **real- world dataset** results on ADBench w.r.t. both AUROC and AUPRC in Appdx. E.3. In addition, we compare PIQL and no-PIQL on OddBench, a substantially larger real-world benchmark containing approximately 690 datasets. As shown in Table 2, paired tests across checkpoints indicate that the PIQL-enabled TFM matches

Table 2: Permutation-test p-values comparing PIQL and no-PI checkpoints on ADBench and OddBench. Values $p > 0.05$ indicate no statistically significant difference. Across both benchmarks, PIQL at 650–700 epochs matches the performance of no-PI models trained for 920–1,050 epochs, demonstrating that PIQL's training acceleration transfers to unseen real-world datasets.

	ADBench		**OddBench**	
No-PI	PIQL @650	PIQL @700	PIQL @650	PIQL @700
@920	0.299	0.084	0.301	0.293
@950	0.497	0.204	0.382	0.243
@1000	0.833	0.390	0.519	0.378
@1050	0.438	0.119	0.420	0.245



(a) Training loss



(b) Test AUROC

No-PI epoch	Target loss	PIQL epoch	Epoch speedup
500	0.0216	286	**1.75**×
800	0.0171	702	**1.14**×
1,000	0.0150	793	**1.26**×
1,300	0.0149	842	**1.54**×

(c) Epoch speedup for 14-layer TFMs at matched training loss.

Figure 2: **Convergence and scaling of** PIQL**. (a–b)** Training loss and average test AUROC for the 10-layer PIQL model (red), no-PI baseline (blue), and no-transfer variant (gray). PIQL reaches lower loss and higher AUROC in fewer epochs, approaching the no-transfer variant at convergence despite fluctuations during teacher-forcing annealing. **(c)** For the 14-layer, approximately 150M-parameter model, PIQL matches each no-PI reference loss in fewer epochs.

the no-PI baseline's performance at earlier epochs, with no statistically significant differences (e.g. on ADBench, p=0.299 for PIQL @650 vs. base @920 epochs, and p=0.119 for PIQL @700 vs. base @1050 epochs).

We also report the **total wall-clock time** saved by training with PIQL. Although PIQL introduces a separate program encoder and student model, its practical overhead can be kept small through several implementation choices. First, the program encoder does not need to run during PIQL training. Each generator program is a lightweight description of a data-generating process, so its embedding can be computed once offline and cached as a single vector. Training the program encoder itself is also a small, one-time cost of approximately 1–1.5 hours. As shown in Table 3, for the 10-layer model, PIQL reduces training from 1,200 to 780 epochs and saves approximately 804 minutes (13.4 hours) of wall-clock time on 4 A100 GPUs. Overall, PIQL achieves a ∼1.25× end-to-end speedup, even after accounting for the student module's online computation.

Table 3: **PIQL efficiency in real-time.** PIQL reduces end-to-end training time despite higher per-epoch cost and retains its acceleration.

A. Time for 10L on 4×A100 GPUs				
No-PI epoch	**PIQL epoch**	**Epoch reduc.**	**Total min.**	**Time Speedup**
1,000	715	28.5%	3,400 / 3,003	1.13×
1,200	780	35.0%	4,080 / 3,276	1.25×
B. Time for 14L (∼150M) on 4×A100 GPUs				
No-PI epoch	**PIQL epoch**	**Epoch reduc.**	**Total min.**	**Time Speedup**
500	286	42.8%	2,750 / 1,658	1.65×
1,000	793	20.7%	5,480 / 4,080	1.34×
1,300	842	35.2%	7,150 / 4,797	1.49×

6.2 Ablation Analyses

To isolate effective privileged information from other factors, we conduct additional ablations on matching model parameters, random tokens/PIs, annealing components (Appdx. E.4) and etc.

Controlling Model Capacity: PIQL has 2.67M parameters, including its lightweight student layer. We therefore construct two No-PI baselines that match or exceed its parameter count: ***Deeper no-PI***, we increase the transformer to 3 layers, ending-up with 3.17M parameters (more parameter count). ***Wider no-PI***, attention width increased from 256 to 288 and FFN width increased from 1,024 to 1,152, with 2.67M parameters (matching parameter count). Table 4 reports the earliest PIQL epoch that matches these models ($1.96-6.52\times$ earlier in training loss and $1.11-2.00\times$ earlier in AUROC). Thus, its acceleration cannot be explained by additional parameters alone.

Matching tokens: To control for prefix length, we give the No-PI, Bronze, and Silver models the same number of prepended tokens, replacing with randomly initialized tokens where necessary. In Table 5, matching the token count does not change the performance ordering: Silver consistently outperforms Bronze, which outperforms No-PI. Therefore, the gains come from the information encoded in PI, not simply from adding more prefix tokens that function like thinking tokens.

Learned PI versus random PI: To isolate PI's learned semantic content, we vary only program-encoder training. Pretraining is lightweight, taking about one hour for 256K synthetic programs. As Table 6 (full result with AUROC and p-values in Table 11) shows, random PI yields only a small gain over No PI (0.74 vs. 0.77 loss), while learned PI reduces loss to 0.53 with 64K programs and 0.49 with 256K, improving with PI quality.

Table 4: PIQL is compared with a larger deeper No-PI model and a parameter-matched wider No-PI model. We report the earliest PIQL epoch matching each No-PI target; parentheses denote speedup.

	Epoch	Loss		AUROC	
		Target	PIQL (×)	Target	PIQL (×)
Deeper	500	.062	254 (**1.96**)	.985	450 (**1.11**)
	1,000	.058	365 (**2.73**)	.989	750 (**1.33**)
	1,500	.057	380 (**3.94**)	.991	1,150 (**1.31**)
Wider	500	.070	131 (**3.81**)	.984	350 (**1.43**)
	1,000	.067	209 (**4.78**)	.987	550 (**1.82**)
	1,500	.065	230 (**6.52**)	.989	750 (**2.00**)

Table 5: No-PI, Bronze, and Silver use the same number of prepended tokens, with random tokens added where needed. We report the earliest Silver epoch matching each control target; parentheses denote speedup.

	Epoch	Loss		AUROC	
		Target	Silver (×)	Target	Silver (×)
No-PI	600	.071	211 (**2.84**)	.979	210 (**2.86**)
	900	.067	233 (**3.86**)	.980	250 (**3.60**)
	1,300	.063	364 (**3.57**)	.982	300 (**4.33**)
Bronze	600	.065	303 (**1.98**)	.978	210 (**2.86**)
	900	.058	451 (**2.00**)	.982	300 (**3.00**)
	1,300	.048	832 (**1.56**)	.984	390 (**3.33**)

Table 6: Learned vs. random PI Architecture and token count are fixed while only program-encoder training varies. We report training loss at TFM epoch 600.

PI	Loss @ 600
No PI	0.77
Random	0.74
Learned (64K)	0.53
Learned (256K)	**0.49**

LLM-as-Program-Encoder: We evaluate the utility of our unified program encoder by replacing it with embeddings from off-the-shelf language models. We consider general-purpose open-source model, `Qwen3-Embed-4B` Zhang et al. (2025c), and code-specialized, identifier-aware model, `codet5p-110m-embedding` Wang et al. (2021). Fig. 3 (left) shows that off-the-shelf embeddings do not provide PI benefits, highlighting the importance of our program-specialized encoder.

Generator-PI, Meta-PI and Meta-PI++: Fig. 1 studied meta-PI (bronze) and generator-PI (silver) in isolation, while the proposed PIQL-TFM can ingest both. Fig. 3 (middle) shows that augmenting generator-PI with meta-PI yields marginal gains, suggesting that train-time generator-PI provides more effective training signal. Our initial list contained only 9 meta-features (Sec. 4.1.1). To strengthen the meta-PI signal, we expand to 100 meta-features via an LLM (see Appdx. E.7). Fig. 3 (right) shows that meta-PI++ using these 100 LLM-generated features alone improves performance, reaching silver-grade levels. These results motivate further research into novel PI designs for FMs.

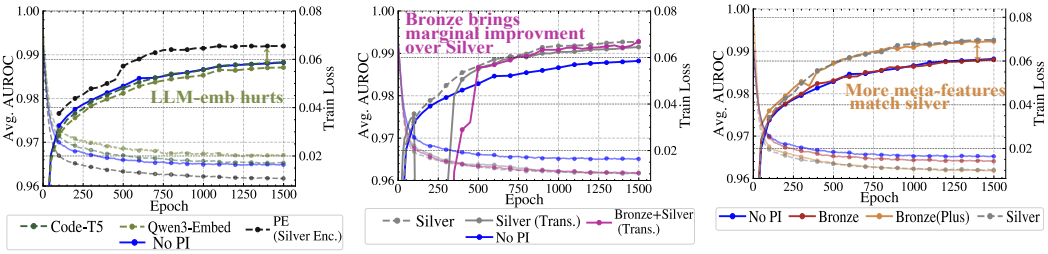


Figure 3: **(left)** LLM-as-Program Encoder; **(middle)** GeneratorPI + MetaPI; **(right)** MetaPI++

6.3 Layer-wise Representation Probing

Finally, we use linear probes (Conneau et al., 2018) (diagnostic classifiers) to analyze representations from each layer (0–9) of pretrained privileged vs. standard TFMs. Specifically, we predict two targets: (i) type of generator (5-way, balanced) and (ii) number of components (M=5-way balanced, for GMM-generated data only). Note that these latent variables are **not** available directly from the observed context samples at inference. As shown in Fig. 1 (right), PIQL'ed TFM representations reflect these latent information *more effectively* (higher classification accuracy) and at *earlier layers* (matching results highlighted). These results provide empirical support for two mechanisms by which PI enhances TFMs: by freeing up ICL capacity and by capturing latent distributional information.

7 Conclusion

We introduced PIQL, Privileged Information for Quick and Quality Learning, a new paradigm for tabular foundation models (TFMs) in which auxiliary information is provided during training. We constructed two forms of PI based on meta-features and the data-generating program, supporting two mechanisms for improving TFMs: freeing up ICL capacity and incorporating latent generative information. To operationalize PIQL, we developed a program encoder that unifies diverse generators and designed an architecture that transfers train-time-only generator-PI via teacher-annealed training. Experiments showed that PIQL enables TFMs to reach lower loss faster, in effect, reducing data and compute requirements. We expect our work to motivate future research on PI-guided pretraining as a principled and practical approach toward more efficient and effective foundation models.

AI USE STATEMENT

In this work, we used generative AI tools to interpret results, implement and debug program encoder code, and edit our research paper to improve readability. We have also asked for AI assitance for generate Meta++ privileged information for an ablation of PIQL. The remaining tasks requiring disclosure do not apply to our paper. We have reviewed all AI-assisted work: LLM-generated code was manually verified and tested for correctness. We have made sure that every AI interpretation is accurate and that readability changes do not alter the meaning of our text. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To support reproducibility, we publish both training and inference code, all used datasets, and the trained PIQL checkpoint at `https://anonymous.4open.science/r/PIQL`. We further provide detailed results for ADBench, time-complexity analysis and additional ablation studies in the Appendix sections.

REFERENCES

Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

Tarek Ait Baha, Mohamed El Hajji, Youssef Es-Saady, and Hammou Fadili. The power of personalization: A systematic review of personality-adaptive chatbots. *SN Computer Science*, 4(5):661, 2023.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

Banghao Chen, Zhaofeng Zhang, Nicolas Langrené, and Shengxin Zhu. Unleashing the potential of prompt engineering for large language models. *Patterns*, 6(6), 2025.

Eric Chu, Prashanth Vijayaraghavan, and Deb Roy. Learning personas from dialogue with attentive memory networks. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2638–2646, 2018.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. In *ACL*, 2018.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B*, 39(1):1–22, 1977.

Xueying Ding, Simon Klüttermann, Haomin Wen, Yilong Chen, and Leman Akoglu. Macrodata: New benchmarks of thousands of datasets for tabular outlier detection. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '26, pp. 8777–8788, New York, NY, USA, 2026a. Association for Computing Machinery. ISBN 9798400722592. doi: 10.1145/3770855.3817520. URL `https://doi.org/10.1145/3770855.3817520`.

Xueying Ding, Haomin Wen, Simon Klütterman, and Leman Akoglu. From zero to hero: Advancing zero-shot foundation models for tabular outlier detection. *arXiv preprint arXiv:2602.03018*, 2026b.

Fabian Falck, Ziyu Wang, and Christopher C. Holmes. Is in-context learning in large language models bayesian? a martingale perspective. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 12784–12805. PMLR, 2024.

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

Siavash Golkar, Mariel Pettee, Michael Eickenberg, Alberto Bietti, Miles Cranmer, Geraud Krawezik, Francois Lanusse, Michael McCabe, Ruben Ohana, Liam Parker, Bruno Régaldo-Saint Blancard, Tiberiu Tesileanu, Kyunghyun Cho, and Shirley Ho. xVal: A continuous number encoding for large language models. In *NeurIPS 2023 AI for Science Workshop*, 2023.

Songqiao Han, Xiyang Hu, Hailiang Huang, Minqi Jiang, and Yue Zhao. Adbench: Anomaly detection benchmark. *Advances in Neural Information Processing Systems*, 35, 2022.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

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

Bowen Jiang, Yuan Yuan, Maohao Shen, Zhuoqun Hao, Zhangchen Xu, Zichen Chen, Ziyi Liu, Anvesh Rao Vijjini, Jiashu He, Hanchao Yu, et al. Personamem-v2: Towards personalized intelligence via learning implicit user personas and agentic memory. *arXiv preprint arXiv:2512.06688*, 2025.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. CTRL: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*, 2019.

Wonjae Kim, Bokyung Son, and Ildoo Kim. Vilt: Vision-and-language transformer without convolution or region supervision. In *International conference on machine learning*, pp. 5583–5594. PMLR, 2021.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 3045–3059, 2021.

Guanrong Li, Xinyu Liu, Zhen Wu, and Xinyu Dai. Persona-aware alignment framework for personalized dialogue generation. *Transactions of the Association for Computational Linguistics*, 13:1722–1742, 2025.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pp. 19730–19742. PMLR, 2023.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, 2021.

Bryan Lim, Sercan Ö Arık, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International journal of forecasting*, 37(4): 1748–1764, 2021.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3470–3487, 2022.

Samuel Müller, Noah Hollmann, Sebastian Pineda-Arango, Josif Grabocka, and Frank Hutter. Transformers can do bayesian inference. In *International Conference on Learning Representations (ICLR)*, 2022.

Dmitry Pechyony and Vladimir Vapnik. On the theory of learning with privileged information. In *Advances in Neural Information Processing Systems (NIPS)*, volume 23, pp. 1894–1902, 2010.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International conference on machine learning*, pp. 8821–8831. Pmlr, 2021.

Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender Systems Handbook*, pp. 1–35. Springer, 2010.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Yuchen Shen, Haomin Wen, and Leman Akoglu. A foundation model for zero-shot tabular outlier detection. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856.

Aad W. van der Vaart. *Asymptotic Statistics*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, Cambridge, 1998.

Vladimir Vapnik and Rauf Izmailov. Learning using privileged information: Similarity control and knowledge transfer. *Journal of Machine Learning Research*, 16(61):2023–2049, 2015. URL `http://jmlr.org/papers/v16/vapnik15b.html`.

Vladimir Vapnik and Rauf Izmailov. Knowledge transfer in SVM and neural networks. *Annals of Mathematics and Artificial Intelligence*, 81(1):3–19, 2017.

Vladimir Vapnik and Akshay Vashist. A new learning paradigm: Learning using privileged information. *Neural Networks*, 22(5-6):544–557, 2009.

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pp. 35151–35174. PMLR, 2023.

Yue Wang, Weishi Wang, Shafiq Joty, and Steven CH Hoi. CodeT5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 8696–8708, 2021.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=gEZrGCozdqR`.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*, 2021.

Xiyuan Zhang, Boran Han, Haoyang Fang, Abdul Fatir Ansari, Shuai Zhang, Danielle C Maddix, Cuixiong Hu, Andrew Gordon Wilson, Michael W Mahoney, Hao Wang, et al. When does multimodality lead to better time series forecasting? *arXiv preprint arXiv:2506.21611*, 2025a.

Xiyuan Zhang, Danielle C. Maddix, Junming Yin, Nick Erickson, Abdul Fatir Ansari, Boran Han, Shuai Zhang, Leman Akoglu, Christos Faloutsos, Michael W. Mahoney, Cuixiong Hu, Huzefa Rangwala, George Karypis, and Bernie Wang. Mitra: Mixed synthetic priors for enhancing tabular foundation models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2025b.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025c.

APPENDIX

A EXTENDED RELATED WORK

Table 7: Comparison of learning paradigms by input source, task structure, and task–input assignment. Our setting uniquely differs from related lines of work in that (1) the additional input, called privileged information (PI) in our case, is not sourced externally but represents **internal** distribution characteristics derived empirically (meta-features) or theoretically (generator program); (2) the latter, generator-PI is **train-time-only** while other settings use external input at both training and inference; (3) PI represents **given**, and not learned, prefix tokens; (4) PI is employed at instance level **per dataset**, rather than being task-specific; (5) PI enriches the training data in the context by representing **distinct yet latent** distributions as opposed to different but known tasks.

/ Setting Problem	**Extra Input**	**Train-only?**	**Given vs. Learned**	**Instance- vs. Task-specific**	**Task–Input Assignment**
Prefix-tuning	N/A	No	Learned	Task	Known
Soft-prompt opt.	N/A	No	Learned	Task	Known
Instruction tuning	External	No	Given	Task	Known
Personalization	External	No	Given	Instance	N/A
Conditional generation	External	No	Given	Instance	N/A
Prediction with aux. data	External	No	Given	Instance	N/A
[Privileged TFMs]	Internal	Yes	Given	Instance	Latent

Our work extends tabular foundation models (TFMs) with privileged information (PI). Specifically, we design and prepend the training points in the context window with PI tokens. As stated, it appears similar to other learning settings including prefix tuning, instruction tuning, personalization, predictions with auxiliary data, and conditional generation. Table 7 highlights the key differences between our problem and these existing lines of work, on which we elaborate in this section.

Prefix-tuning (Li & Liang, 2021) aims to learn different soft-prompts per task while keeping the parameters of a pretrained model intact. As opposed to maintaining multiple model parameters, one per task, prefix tokens require a much smaller number of parameters.

Soft-prompt optimization (Lester et al., 2021; Chen et al., 2025) is created concurrently and shares the same objective. The difference is that prefix-tuning prepends a sequence of learned prefixes at every transformer layer while prompt tuning is simpler and limits prefixes to the input layer. In both settings, *the prepended tokens are learned* to optimize a specific task's performance, and unlike our setting, there exist no external auxiliary input.

Instruction tuning (Wei et al., 2022; Mishra et al., 2022), in contrast, fine-tunes a pretrained model's parameters on a collection of tasks expressed via natural-language instructions. Each training example includes an instruction that specifies the task, in addition to the input/output pairs, and the model is trained typically with a unified language modeling objective to follow these instructions and generalize to new ones. Unlike our train-time-only PI, instruction tuning relies on *explicit instructions given at both training and inference time*.

In essence, these approaches target strong performance across a collection of distinct tasks. Prefix tuning and soft-prompt optimization use *learned*, task-specific tokens while keeping the base model fixed, whereas instruction tuning relies on *given*, task-specific natural-language instructions and updates the model parameters accordingly. Crucially, in all cases the prepended tokens are *task-specific* and explicitly signal the *known* task to the model; in contrast, in our setting the PI tokens implicitly characterize the *latent* underlying data distribution rather than specifying a task.

Personalization has been a central research topic for decades, particularly in recommender systems, well predating large language models (Adomavicius & Tuzhilin, 2005; Ricci et al., 2010). The primary input space typically consists of user–item interactions, such as explicit ratings or implicit feedback (e.g., clicks, watch time). Personalization is often improved by augmenting these signals with auxiliary information about users (e.g., demographics, preferences) and items (e.g., attributes such as genre or director). In most cases, these auxiliary user/item attributes are obtained from

external sources, although in some cases they may also be inferred (e.g. buying `diapers` regularly implies `is-father`, watching `coming-of-age movies` implies `is-teenager`, etc.).

The modern version of this setting appears in personalizing large language model (LLM) based dialogue generation to individual users (Ait Baha et al., 2023). In such systems, user characteristics and preferences are recorded or inferred from historical interactions and retrieved, often from a memory module, to augment the model's context at generation time (Chu et al., 2018; Li et al., 2025; Jiang et al., 2025).

Conditional generation can be viewed as an instance of personalization for generative models, where LLMs and image generators produce outputs conditioned on auxiliary information such as user preferences, instruction, or retrieved context (Mirza & Osindero, 2014; Keskar et al., 2019; Ramesh et al., 2021; Rombach et al., 2022).

Similar to instruction tuning and personalization, the conditioning input is external and given at both training and inference time. Conditional generation focuses on instance-level conditioning, akin to personalization, rather than distinct tasks as in instruction tuning.

Learning with auxiliary data is one of the most widely used strategies for improving model performance. At its core, this paradigm seeks to incorporate information beyond the original input space by leveraging additional data sources. A prominent recent trend is multi-modal fusion, where heterogeneous modalities are combined to enhance performance. Representative examples include time series forecasting augmented with textual annotations (Lim et al., 2021; Zhang et al., 2025a), joint text–image representation learning (Radford et al., 2021), and vision–language pretraining frameworks (Kim et al., 2021; Li et al., 2023).

Across these paradigms, a unifying theme is the use of additional information beyond the raw input to improve learning. However, they differ fundamentally in the nature, availability, and role of this auxiliary information. In particular, none of the aforementioned settings is designed for tabular foundation models, nor do they explicitly operate under a *train-time-only* information regime. In contrast, our framework is grounded in the LUPI paradigm, where privileged information is available exclusively during training and is not required at inference time. This distinction is critical: rather than serving as persistent conditioning input (as in personalization, conditional generation, or auxiliary-data learning), or as task-defining signals shared across train and test (as in instruction tuning and prefix-based methods), our PI acts as a transient learning signal that accelerates learning and improves generalization of TFMs. Our auxiliary information, namely the designed privileged information comprising dataset meta-features and generator program specifications, distinctly encodes latent structure in the data distribution, enabling more efficient learning of tabular predictors under a foundation model paradigm.

B Unifying Generator Programs in a Shared Token Space

B.1 Goal and Challenges

We aim to **encode the generator specification**, consisting of the program, its structure, and its parameters, rather than the data it emits. Concretely, a generator configuration is derived from the following synthetic data priors:

- A Gaussian Mixture Model (GMM) with specific parameters (number of components, means, covariances, mixture weights),
- A Structural Causal Model (SCM) with a specific DAG and structural equations, and
- A Copula with specific feature marginals and dependence structure.

Our **goal** is to represent any such generator configuration as a small set of vector tokens. That is, we seek to encode the *generative program itself* into a compact token representation.

This poses a nontrivial challenge, as the three generator families are fundamentally different objects:

- **GMM:** defined by M Gaussian clusters, each with a mean vector $\boldsymbol{\mu}_k \in \mathbb{R}^d$, a covariance matrix $\boldsymbol{\Sigma}_k \in \mathbb{R}^{d\times d}$, and a mixture weight π_k;

- **SCM:** defined by a DAG adjacency matrix $\mathbf{A} \in \{0,1\}^{d\times d}$, structural equations with specified functional forms, and per-variable noise distributions;
- **Copula:** defined by a copula family with associated parameters, together with d marginal distributions, each specified by its own family and parameters.

In addition, each generator family contains distinctive outlier types. Overall, our mixture consists of five priors, which we briefly describe in the following. We refer to Ding et al. (2026b) for detailed descriptions.

(i) For GMM, outliers are generated by inflating the covariance matrices of subspaces and sampling from the resulting inflated Gaussian distributions.

For SCM, there are two types of outlier archetypes: (ii) measurement and (iii) structural outliers.

(ii) The former picks one node from the DAG and samples with inflated exogenous noise;
(iii) The latter breaks the causal dependencies within the DAG by removing or reversing edges.

Copulas also exhibit two types of outlier archetypes: (iv) dependence and (v) probabilistic outliers.

(iv) The former samples outliers by overwriting the copula vector and inverting a subset of dimensions;
(v) The latter samples outliers by selecting a subset of features and perturbing the copula coordinates by pushing towards the boundaries.

The central **challenge** is to map these five heterogeneous specifications into a common token space, i.e., unify the distinct prior programs.

B.2 Program Serialization and Tokenization

Serialization: We treat each generator specification as a **structured sequence**, analogous to a domain-specific language (DSL) or program. Below we illustrate the structured format for each generator family.

Example of GMM Prior Description.

```
[FAMILY:GMM] [DIM:5] [N_COMP:3]
[ENTITY] [TYPE:COMPONENT] [ID:0] [MEAN:-1.9,0.0,1.6,...] [COV_DIAG:0.8,0.5,...]
[ENTITY] [TYPE:COMPONENT] [ID:1] [MEAN:0.0,-0.2,-0.6,...] [COV_DIAG:0.9,0.4,...]
[ENTITY] [TYPE:COMPONENT] [ID:2] [MEAN:0.1,-0.9,0.0,...] [COV_DIAG:0.1,0.7,...]
[OUTLIER] [TYPE:inflated_cov] [SUB_DIMS:1.0,2.0...]
[ENTITY] [TYPE:OUTLIER_COMPONENT] [ID:0] [INF_COV:0.8,2.5,...]
[ENTITY] [TYPE:OUTLIER_COMPONENT] [ID:1] [INF_COV:0.9,2.3,...]
[ENTITY] [TYPE:OUTLIER_COMPONENT] [ID:2] [INF_COV:0.1,3.5,...]
```

The program above encodes a 5-dimensional GMM prior with three inlier components, where each component is described by its mean vector and diagonal covariance. The outlier mechanism is specified as inflated covariance on selected sub-dimensions, meaning outliers are generated by increasing variance along certain feature dimensions for each corresponding mixture component.

Example of SCM-Measurement Prior Description.

```
[FAMILY:SCM] [DIM:4]
[ENTITY] [TYPE:VAR][ID:0] [LAYER:0] [EQ:tanh] [PARENTS:none] [COEFFS:none]
[ENTITY] [TYPE:VAR][ID:1] [LAYER:0] [EQ:tanh] [PARENTS:none] [COEFFS:none]
[ENTITY] [TYPE:VAR][ID:2] [LAYER:1] [EQ:leaky_relu] [PARENTS:1.0] [COEFFS:-0.7]
[ENTITY] [TYPE:VAR][ID:3] [LAYER:2] [EQ:tanh] [PARENTS:0.0,2.0] [COEFFS:1.4,-0.1]
[OUTLIER] [TYPE:noise_scale] [HIGH_NOISE:5.0] [HIGH_NOISE_PROB:0.2][N_PROTO:2]
[ENTITY] [TYPE:OUTLIER_PROTOTYPE] [ID:0] [HIGH_NOISE_NODES:1.0,2.0]
[ENTITY] [TYPE:OUTLIER_PROTOTYPE] [ID:1] [HIGH_NOISE_NODES:2.000]
```

The program above encodes a 4-dimensional SCM, where variables are generated according to a layered causal graph: variables 0 and 1 are root nodes, variable 2 depends on variable 1 through a

leaky-ReLU equation, and variable 3 depends on variables 0 and 2 through a tanh equation. The outlier mechanism is noise-scale perturbation, where selected variables receive high exogenous noise with probability 0.2, producing two outlier prototypes: one affecting nodes 1 and 2, and another affecting node 2 only.

Example of SCM-Structural Prior Description.

```
[FAMILY:SCM] [DIM:4]
[ENTITY] [TYPE:VAR][ID:0] [LAYER:0] [EQ:tanh] [PARENTS:none] [COEFFS:none]
[ENTITY] [TYPE:VAR][ID:1] [LAYER:0] [EQ:tanh] [PARENTS:none] [COEFFS:none]
[ENTITY] [TYPE:VAR][ID:2] [LAYER:1] [EQ:leaky_relu] [PARENTS:1.0] [COEFFS:-0.7]
[ENTITY] [TYPE:VAR][ID:3] [LAYER:2] [EQ:tanh] [PARENTS:0.0,2.0] [COEFFS:1.4,-0.1]
[OUTLIER] [TYPE:weight_mask] [PERTURB_PROB:0.2]  [N_PROTO:2]
[ENTITY] [TYPE:OUTLIER_PROTOTYPE] [ID:0] [FLIPPED_NODES:0.0] [DROPPED_NODES:2.0]
[ENTITY] [TYPE:OUTLIER_PROTOTYPE] [ID:1] [FLIPPED_NODES:1.0] [DROPPED_NODES:]
```

The above encodes a similar 4-dimensional SCM: variables 0 and 1 are root variables, variable 2 depends on variable 1, and variable 3 depends on variables 0 and 2. The outlier mechanism is a weight-mask perturbation, where causal relationships are altered with probability 0.2 by flipping or dropping selected nodes/edges. Specifically, prototype 0 flips node 0 and drops node 2, while prototype 1 flips node 1 without dropping any node, creating structural outliers that violate the normal causal generation process.

Example of Copula-Probabilistic Prior Description.

```
[FAMILY:COPULA] [DIM:4]
[ENTITY] [TYPE:COPULA_BASE] [ID:0] [COPULA_PARAM:random] [CHOL:1.000,0.852,...]
[ENTITY] [TYPE:MARGINAL] [ID:0] [DIST:studentt] [PARAMS:8,0,1]
[ENTITY] [TYPE:MARGINAL] [ID:1] [DIST:beta] [PARAMS:1.2,3.0]
[ENTITY] [TYPE:MARGINAL] [ID:2] [DIST:normal] [PARAMS:0.7,1.6]
[ENTITY] [TYPE:MARGINAL] [ID:3] [DIST:exponential] [PARAMS:1.8]
[OUTLIER] [TYPE:PROBABILISTIC] [N_PROTO:2] [STRENGTH_RANGE:0.2~0.4]
[ENTITY] [TYPE:OUTLIER_PROTOTYPE] [ID:0] [PERTURBED_DIMS:0.0,1.0,2.0]
[ENTITY] [TYPE:OUTLIER_PROTOTYPE] [ID:1] [PERTURBED_DIMS:1.0]
```

The above encodes a 4-dimensional copula prior, where the dependence structure is specified by a copula base through a Cholesky factor, while each dimension has its own distinct marginal distribution: Student-t, Beta, Normal, and Exponential. The outlier mechanism is probabilistic perturbation, where selected dimensions are perturbed with a strength sampled from 0.2–0.4. Specifically, prototype 0 perturbs dimensions 0, 1, and 2, while prototype 1 perturbs only dimension 1, creating outliers that deviate from the normal marginal/dependence pattern.

Example of Copula-Dependence Prior Description.

```
[FAMILY:COPULA] [DIM:3]
[ENTITY] [TYPE:COPULA_BASE] [ID:0] [COPULA_PARAM:random] [CHOL:0.83,0.25,0.216]
[ENTITY] [TYPE:MARGINAL] [ID:0] [DIST:normal] [PARAMS:-0.88,1.76]
[ENTITY] [TYPE:MARGINAL] [ID:1] [DIST:interp] [PARAMS:lo=-5.3,hi=6.41,u_grid=2000]
[ENTITY] [TYPE:MARGINAL] [ID:2] [DIST:exponential] [PARAMS:1.4]
[OUTLIER] [TYPE:DEPENDENCE] [N_PROTO:1] [STRENGTH_RANGE:0.97~0.99]
[ENTITY] [TYPE:OUTLIER_PROTOTYPE] [ID:0] [DISTURBED_DIMS:0.0,2.0]
```

The description above encodes a similar 3-dimensional copula prior, where the copula base defines the dependence structure through a Cholesky factor. Each dimension has its own marginal distribution: Normal, Interpolated/Custom, and Exponential. The outlier mechanism is a dependence perturbation, meaning that outliers are generated by strongly disrupting the dependency pattern among selected dimensions. In this case, the single outlier prototype disturbs dimensions 0 and 2 with high strength, creating samples that may look marginally plausible but violate the normal cross-feature dependence.

Tokenization: With this **structured/serialized format**, we next describe how to **tokenize** the program.

Taking the first GMM as an example, the GMM prior is first serialized as a bracketed program string, where each block, such as `[FAMILY:GMM]`, `[ENTITY]`, `[MEAN:...]`, explicitly marks the semantic role of the information. The tokenizer then parses each bracket into typed tokens: categorical items like `FAMILY`, `GMM`, `ENTITY`, and `COMPONENT` become symbol tokens; scalar fields like `DIM`, `N_COMP`,and `ID` become scalar tokens; vector fields like `MEAN` and `COV_DIAG` become one vector entry token per dimension; and `SUB DIMS` becomes index entry tokens indicating which feature dimensions are affected. The box below shows an example tokenization of the GMM example above.

Example of Tokenized GMM Prior Description.


```
[
  {"type": "symbol", "name": "FAMILY"},
  {"type": "symbol", "name": "GMM"},
  {"type": "symbol", "name": "DIM"},
  { "type": "scalar","field": "DIM", "value": 5.0, "family": "GMM",
  "block": null
  },
  {"type": "symbol", "name": "N_COMP"},
  {"type": "scalar","field": "N_COMP", "value": 3.0, "family": "GMM",
  "block": null
  },
  {"type": "symbol", "name": "ENTITY"},
  {"type": "symbol", "name": "TYPE"},
  {"type": "symbol", "name": "COMPONENT"},
  {"type": "symbol", "name": "ID"},
  {"type": "scalar","field": "ID", "value": 0.0,
  "family": "GMM", "block": "ENTITY", "entity_type": "COMPONENT","entity_id": 0
  },
  {"type": "symbol", "name": "MEAN"},
  {"type": "vector_entry","field": "MEAN","dim": 0,"value": -1.9, "family": "GMM",
  "block": "ENTITY", "entity_type": "COMPONENT","entity_id": 0
  },
  "... remaining COMPONENT and OUTLIER_COMPONENT tokens omitted ..."
]
```


Given the **tokenized** prior description, each token is converted into a **dense embedding** by summing learnable embeddings for its semantic attributes. For example, consider the token:


```
{
  "type": "vector_entry",
  "field": "MEAN",
  "dim": 0,
  "value": -1.9,
  "family": "GMM",
  "block": "ENTITY",
  "entity_type": "COMPONENT",
  "entity_id": 0
}
```


This token represents "dimension 0 of the mean vector for component 0 in a GMM prior". Its embedding is obtained by summing several pieces: (i) field embedding for `MEAN`, (ii) family embedding for `GMM`, (iii) entity-type embedding for `COMPONENT`, (iv) entity-ID embedding for component `0`, (v) dimension embedding for dim `0`, and (vi) value embedding for `-1.9`. Importantly, the numeric value `-1.9` is first normalized and then passed through a small MLP to produce its embedding value. Therefore, the final token embedding does not only encode the number itself; it also encodes what that number means within the prior structure.

To build a meaningful embedding table, we also utilize a bootstrap step in the beginning to build vocabulary before training, by sampling many synthetic prior descriptions across GMM, SCM, and Copula families, tokenizing them, and collecting all possible symbols, fields, family names, and entity types needed to initialize the embedding tables. Without this step, some tokens appearing later could

be missing from the vocabulary and would collapse to the default unknown/PAD index, losing their semantic meaning. The bootstrap is saved and then utilized at TFM pre-training/ inference phase.

We remark that this special serialization and tokenization of programs produces more meaningful signals than simply using off-the-shelf text embeddings from language models. (See our LLM-as-Program-Encoder ablations in Sec. 6.2.) It also embeds more specific program information than existing scientific tokenization methods such as xVal (Golkar et al., 2023).

Encoding: Finally, we encode the serialized and tokenized sequence using a Transformer, prepending k learnable `[PREFIX]` tokens (with different embedding initialization) to the input. Through self-attention, these summary tokens accumulate information from the entire generator specification:

$$\underbrace{[\mathbf{p}_1]\ [\mathbf{p}_2]\ \cdots\ [\mathbf{p}_k]}_{\text{learnable prefix tokens}}\ \underbrace{[\texttt{FAMILY:GMM}]\ [\texttt{DIM:5}]\ [\texttt{N_COMP:3}]\ [\texttt{COMP}]\ [\texttt{W:0.4}]\ \cdots}_{\text{serialized generator specification}}$$

A full generator sequence $\mathcal{G}$ is passed through the Transformer encoder, and only the first k output positions are retained:

$$\{\mathbf{p}_1^*, \ldots, \mathbf{p}_k^*\} = \mathrm{Enc}\big(\,\mathrm{serialize}(\mathcal{G})\big)\ \in\ \mathbb{R}^{d_{\text{model}}}$$

These k vectors constitute the *generator tokens*; a compact, fixed-size representation of the generator, regardless of the underlying family or dimensionality.

B.3 Program Encoder Training

Program Contrastive Loss. We train the program encoder using a margin-based contrastive objective over serialized prior programs. For an anchor generator $\mathcal{G}$, we construct a positive $\mathcal{G}^+$ by weakly perturbing its parameters while preserving the same prior family, outlier mechanism, and dimensionality. Negatives are either easy negatives sampled from different prior families, or hard negatives generated by applying a stronger perturbation to the same anchor configuration. After serialization, tokenization, and encoding, we obtain embeddings $\mathbf{z}$, $\mathbf{z}^+$, and $\{\mathbf{z}_i^-\}_{i=1}^M$.

We optimize a weighted triplet-style objective:

$$\mathcal{L}_{\text{triplet}} = \frac{1}{M}\sum_{i=1}^{M} w_i\left[\|\mathbf{z}-\mathbf{z}^+\|_2 - \|\mathbf{z}-\mathbf{z}_i^-\|_2 + m_i\right]_+,$$

where hard negatives use a separate margin m_{hard} and weight w_{hard}, while easy negatives use margin m_{easy} and unit weight. We further add a hardest-negative mining term

$$\mathcal{L}_{\text{mine}} = \left[\|\mathbf{z}-\mathbf{z}^+\|_2 - \min_i \|\mathbf{z}-\mathbf{z}_i^-\|_2 + m_{\text{hard}}\right]_+.$$

Finally, to preserve high-level prior-family structure, we introduce learnable family proxies and apply a cross-entropy loss over normalized anchor-proxy similarities. The final objective is

$$\mathcal{L} = \mathcal{L}_{\text{triplet}} + \lambda_{\text{mine}}\mathcal{L}_{\text{mine}} + \lambda_{\text{proxy}}\mathcal{L}_{\text{proxy}}.$$

Program Encoder Architecture and Training. We train two versions of the program encoder: the first version is for GMM prior only and the second version is for Mix-priors. The GMM-only program encoder contains one `PREFIX` token while the latter contains $k = 10$ learnable prefix tokens to capture more information. We list the detailed hyperparameter configurations of the Transformer based encoders in Table 8.

Both program encoders are trained for 20 epochs with 80 steps per epoch, batch size 16, AdamW optimizer with learning rate 10^{-4}, warmup ratio 0.1, and minimum learning-rate ratio 5×10^{-6}. The contrastive training setup uses 8 negatives per anchor. The prior-generation range uses dimensionality $d \in [2, 100]$, GMM components in $[1, 5]$, maximum mean magnitude 6, maximum variance 6, covariance inflation scale 5.0, and 16 outlier prototypes. The training takes places over one single L40 GPU for approximately 24 hours.

Table 8: Hyperparameters of the two program encoder (PE) architecture variants.

Hyperparameter	GMM-only PE	Mixed-prior PE
Number of PREFIX tokens	1	10
Maximum sequence length	2048	2048
Hidden dimension d_{model}	256	256
Number of attention heads	8	8
Number of Transformer layers	6	4
Feedforward dimension	256	256
Dropout rate	0.1	0.1

Curriculum Hyperparameters. The GMM-only program encoder is not trained with easy/hard negatives due to all negatives being within the same prior family. However, we utilize a curriculum for the Mixed-prior program encoder. For curriculum-based contrastive training, we use a warm-up period of 3 epochs before gradually increasing the hard-negative difficulty. The curriculum is controlled by the hard-negative sampling ratio ρ_{hard}, which is annealed from $\rho_{\text{start}} = 0.1$ to $\rho_{\text{end}} = 0.5$, and the hard-negative perturbation strength ϵ_{hard}, which is annealed from an easier initial value to the target value $\epsilon_{\text{hard}} = 0.25$. We set easy margin $m_{\text{easy}} = 0.8$, hard margin $m_{\text{hard}} = 0.4$, hard-negative weight $w_{\text{hard}} = 1.8$, hard-mining weight $\lambda_{\text{mine}} = 0.5$, proxy loss weight $\lambda_{\text{proxy}} = 0.5$. This schedule exposes the encoder to easier negatives early in training and progressively introduces harder, more semantically similar negatives, encouraging stable coarse-to-fine representation learning.

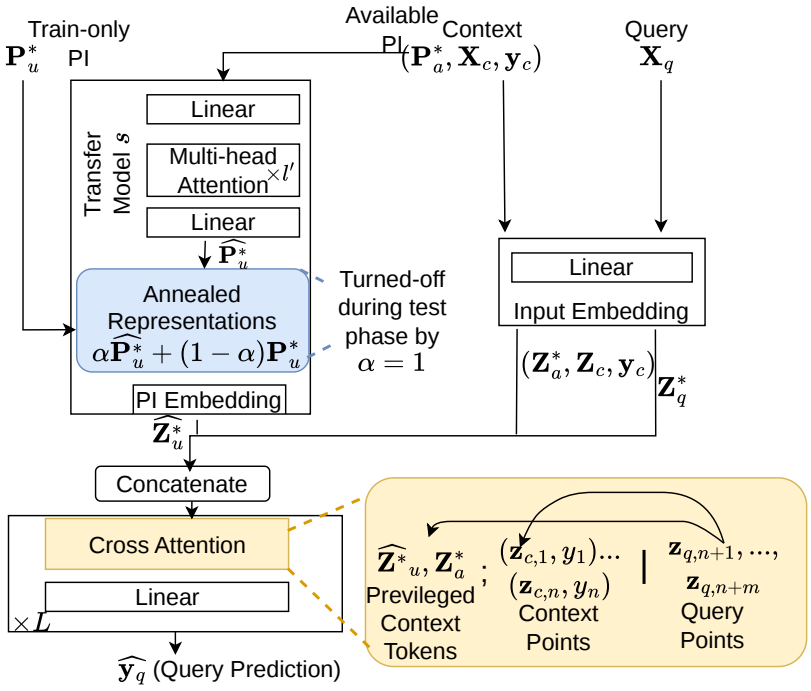


Figure 4: Illustration of our PIQL diagram: the teacher-student TFM and training to transfer PI.

C Privileged Transformer Architecture and Training with PI Transfer

Inspired by the knowledge transfer framework Vapnik & Izmailov (2017), we propose PIQL to transfer generative privileged information into the foundation model. The model consists of two coupled components: **(1)** a transferring student model s that learns the embedding of train-time-only

generative privileged information based on $(\mathbf{P}_a^*, \mathbf{X}_c, \mathbf{y}_c)$, and **(2)** a Transformer-based foundation model (FM) h that performs context-to-query prediction over an assembled token sequence.

The transfer model s takes the train-only privileged signal $\mathbf{P}_u^*$ and $(\mathbf{P}_a^*, \mathbf{X}_c, \mathbf{y}_c)$ to map it to a set of hidden PI embeddings $\widehat{\mathbf{Z}}_u^*$, which are utilized as prepended context tokens into the transformer model. Since $\mathbf{P}_u^*$ is un-available during inference time, we aim to reduce the model's dependence on this privileged information when learning the embedding $\widehat{\mathbf{Z}}_u^*$. To that end, we introduce a representation annealing mechanism. Specifically, the privileged representation at training step t is formed as

$$\widehat{\mathbf{P}}_u^{(t)} = (1 - \alpha_t)\mathbf{P}_u^* + \alpha_t \widehat{\mathbf{P}}_u^*, \tag{3}$$

where $\widehat{\mathbf{P}}_u^*$ is the model-predicted approximation of the privileged representation and α_t is increased during training. As such, early training relies primarily on the true privileged signal $\mathbf{P}_u^*$, while later training increasingly relies on the learned approximation $\widehat{\mathbf{P}}_u^*$.

The mixed representation $\widehat{\mathbf{P}}_u^{(t)}$ in Eq. equation 3 is embedded into privileged tokens $\widehat{\mathbf{Z}}_u^*$. These tokens are concatenated with available privileged/context tokens $\mathbf{Z}_a^*$, labeled context tokens $\{(\mathbf{z}_{c,i}, y_i)\}_{i=1}^n$, and query tokens $\{\mathbf{z}_{q,j}\}_{j=1}^m$:

$$\mathbf{S} = \mathrm{Concat}\left(\widehat{\mathbf{Z}}_u^*,\ \mathbf{Z}_a^*,\ \{(\mathbf{z}_{c,i}, y_i)\}_{i=1}^n,\ \{\mathbf{z}_{q,j}\}_{j=1}^m\right). \tag{4}$$

The assembled sequence is then processed by the Transformer backbone h(similar to the main backbone in Ding et al. (2026b); Shen et al. (2025)). The context tokens provide labeled in-context examples, while the privileged tokens provide additional information about the data-generating process during training. The decoder is applied only to the query-token outputs to obtain the final query predictions:

$$\widehat{\mathbf{y}}_q = h_\theta(\mathbf{S})_{\text{query}}\ . \tag{5}$$

Hyperparameters. We use a lightweight transfer model s with $l' = 1$ to limit the number of additional parameters. It projects the 100-dimensional input features into a 256-dimensional space, where learned query tokens perform attention-based pooling over the input sequence. The pooled representation is then projected to a compact 256-dimensional output. The other detailed hyperparameter configurations can be found in the additional experiment details in Appendix E.

D Theoretical Analysis

Let the prior define a space of hypotheses Ψ on the relationship of a set of features to the output label. Each hypothesis $\psi \in \Psi$ can be seen as a mechanism that generates a data distribution from which we can draw samples forming a dataset $\mathcal{D}$. Let $\psi \sim P_\Psi$ denote the parameters of a hypothesis sampled from the data prior (generator). Then, synthetic datasets are sampled as

$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n \sim P(\cdot \mid \psi)\ .$$

A tabular foundation model, denoted h_θ with parameters θ, is a meta-learner that is trained on numerous datasets to learn the mapping

$$h_\theta : (\mathcal{D}_{\text{context}}, \mathbf{x}_{\text{query}}) \mapsto y_{\text{query}}\ .$$

Let $\mathcal{H}$ denote the hypothesis class induced by a fixed transformer architecture (fixed depth, width, attention heads, and context length):

$$\mathcal{H} = \{h_\theta(\mathcal{D}, \mathbf{x}) : \theta \in \Theta\}\ .$$

The expected meta-risk is written as

$$R(h) = \mathbb{E}_{\psi \sim P_\Psi} \mathbb{E}_{\mathcal{D} \sim P(\cdot|\psi)} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}} \left[\ell(h(\mathcal{D}, \mathbf{x}), y)\right]\ .$$

Let h^* be the best possible decision function (in terms of generalization error) in $\mathcal{H}$, and let

$$R^* = \inf_{h \in \mathcal{H}} R(h) = R(h^*)\ .$$

Let $\mathcal{P}^*$ denote the privileged information (PI) associated with a data set constructed by the teacher. Define the augmented hypothesis class

$$\mathcal{H}_{\text{PI}} = \{h_\theta(\mathcal{P}^*, \mathcal{D}, \mathbf{x}) : \theta \in \Theta_{\text{PI}}\} ,$$

and let

$$R^*_{\text{PI}} = \inf_{h \in \mathcal{H}_{\text{PI}}} R(h) .$$

D.1 PI-Driven Representational Advantage

In this section, we establish the theoretical potential of our approach, by characterizing the conditions under which privileged information (PI) narrows the population-level approximation gap between a standard learner and a PI-augmented hypothesis class.

D.1.1 Meta-features based Privileged Information

Let $\mathcal{P}^* = \mathbf{s}(\mathcal{D})$ denote a set of meta-features comprising summary statistics computed from the context dataset $\mathcal{D}$ (e.g., empirical mean, higher moments, etc.).

Although $\mathbf{s}(\mathcal{D})$ is a deterministic function of $\mathcal{D}$, the class $\mathcal{H}$ may not contain functions that reliably compute or represent $\mathbf{s}(\mathcal{D})$ internally under fixed architectural constraints (e.g., bounded depth, width, and attention heads).

In such cases, $\mathcal{H} \subsetneq \mathcal{H}_{\text{PI}}$, which implies $R^*_{\text{PI}} \leq R^*$.

Thus, summary statistics can reduce achievable asymptotic meta-risk relative to the fixed architecture, even though they do not introduce new information in an information-theoretic sense.

Theorem D.1 (Model-Relative Risk Reduction via Meta-PI under Architectural Constraints)**.** *Let us define the set of useful teachers as*

$$\Phi_u = \{\phi \in \Phi \mid \ell_{(\mathcal{P}^*,\mathcal{D})}(\phi(\mathcal{P}^*, \mathcal{D}, \mathbf{x})) \approx \ell_{\mathcal{D}}(h^*(\mathcal{D}, \mathbf{x})) \quad \textit{for any } (\mathcal{P}^*, \mathcal{D}, \mathbf{x}) \textit{ with non-zero probability.}\} \tag{1}$$

In words, useful teacher (correcting) functions can, using the additional privileged information, emulate the oracle corrections of the best-in-class learner for typical inputs, and therefore have an approximation error smaller than that of the learner.

Assume:

(A1) $\Phi_u \neq \emptyset$*, i.e., there exists a useful teacher(s).*

(A2) For each useful teacher $\phi \in \Phi_u$*, there exists a decision function* $h \in \mathcal{H}$ *such that, for any* $(\mathcal{P}^*, \mathcal{D}, \mathbf{x})$ *with non-zero probability,*

$$\ell_{(\mathcal{P}^*,\mathcal{D})}(\phi(\mathcal{P}^*, \mathcal{D}, \mathbf{x})) \geq \ell_{\mathcal{D}}(h(\mathcal{D}, \mathbf{x})) \ .$$

That is, the learner's hypothesis class can implement the teacher space, where every useful teacher corresponds to a realizable learner that matches the teacher's predictions on the non-privileged inputs. Informally, useful teachers are not "too good" for the learner class.

(A3) There exists $\epsilon > 0$ *such that for all* $h \in \mathcal{H}$*,*

$$\mathbb{E}_{\mathcal{D} \sim P(\cdot|\psi)} \|\mathbf{z}_h(\mathcal{D}) - \mathbf{s}(\mathcal{D})\|^2 \geq \epsilon,$$

where $\mathbf{z}_h$ *denotes the implicit dataset-level representation computed by* h*. That is, the learner class cannot approximate the meta-features arbitrarily well under fixed architecture constraints.*

Then, under assumptions (A1)–(A3),

$$R^*_{PI} \leq R^* - c\epsilon$$

for some constant $c > 0$ *that depends on the sensitivity of the loss to the meta-features.*

Proof. We show that the PI-augmented class achieves strictly lower population risk by separating (i) realizability of useful PI-dependent predictors and (ii) irreducible representation error in $\mathcal{H}$.

Step 1: Existence of a useful PI-dependent predictor.

By (A1), there exists a useful teacher $\phi \in \Phi_u$ such that its risk is comparable to the best-in-class non-PI predictor:

$$R(\phi) \approx R(h^*).$$

Let ϕ act as a reference PI-dependent predictor.

Step 2: Alignment with the learner class.

By (A2), for this $\phi \in \Phi_u$, there exists $h_\phi \in \mathcal{H}$ such that for all $(\mathcal{P}^*, \mathcal{D}, \mathbf{x})$,

$$\ell_{(\mathcal{P}^*,\mathcal{D})}\big(\phi(\mathcal{P}^*, \mathcal{D}, \mathbf{x})\big) \;\geq\; \ell_{\mathcal{D}}\big(h_\phi(\mathcal{D}, \mathbf{x})\big).$$

Taking expectations over the data distribution yields

$$R(\phi) \geq R(h_\phi).$$

Hence, every useful PI-dependent predictor has a counterpart in $\mathcal{H}$ that is not worse when restricted to non-privileged inputs.

This step ensures that any advantage of ϕ is not due to a strictly richer hypothesis class, but must arise from how PI modifies the representation available during learning.

Step 3: Representation bottleneck in $\mathcal{H}$.

Each $h \in \mathcal{H}$ induces a dataset representation $\mathbf{z}_h(\mathcal{D})$. By (A3),

$$\mathbb{E}_{\mathcal{D}} \|\mathbf{z}_h(\mathcal{D}) - \mathbf{s}(\mathcal{D})\|^2 \geq \epsilon.$$

That is, no $h \in \mathcal{H}$ can recover the meta-features $\mathbf{s}(\mathcal{D})$ arbitrarily well under the fixed architecture. Thus, predictors in $\mathcal{H}$ must act on a distorted version of the input-relevant meta-features.

Step 4: Risk gap induced by representation distortion.

Because useful predictors in Φ_u depend non-trivially on $\mathbf{s}(\mathcal{D})$, there exists $c > 0$ such that for any $h \in \mathcal{H}$,

$$R(h) \geq R(\phi) + c \cdot \mathbb{E}_{\mathcal{D}} \|\mathbf{z}_h(\mathcal{D}) - \mathbf{s}(\mathcal{D})\|^2.$$

Using (A3), this implies

$$R(h) \geq R(\phi) + c\epsilon, \quad \forall h \in \mathcal{H}.$$

Taking the infimum over $\mathcal{H}$ yields

$$R^* \geq R(\phi) + c\epsilon. \tag{1}$$

Step 5: PI alleviates the representation bottleneck.

In $\mathcal{H}_{\text{PI}}$, the predictor has direct access to $\mathcal{P}^* = \mathbf{s}(\mathcal{D})$, eliminating the representation gap. Hence there exists $h_{\text{PI}} \in \mathcal{H}_{\text{PI}}$ such that

$$R(h_{\text{PI}}) \approx R(\phi), \quad \Rightarrow \quad R^*_{\text{PI}} \leq R(\phi). \tag{2}$$

Step 6: Combining the bounds.

Combining inequalities (1) and (2), we get

$$R^*_{\text{PI}} \leq R(\phi) \leq R^* - c\epsilon,$$

which yields

$$\boxed{R^*_{\text{PI}} \leq R^* - c\epsilon}\,.$$

□

In short, Theorem D.1 shows that the augmented hypothesis class can achieve reduced expected (meta-)risk, that is $R^*_{\text{PI}} < R^*$, assuming **(A1)** Useful teachers exist ($\Phi_u \neq \emptyset$), **(A2)** Learner class can mimic any useful teacher ($\mathcal{H}$ can realize Φ_u), and and **(A3)** Learner class cannot recover meta-features exactly ($\mathbf{z}_h(\mathcal{D})$ differs from $\mathbf{s}(\mathcal{D})$ by at least ϵ in expectation).

D.1.2 GENERATOR BASED PRIVILEGED INFORMATION

Next we move to the true LUPI case: generator program available during pretraining (i.e. meta-training) but not at inference time. This approach extends the utility of the meta-features as PI, as it enables information-theoretic gains in the asymptotic limit, beyond architecture-relative improvement.

Theorem D.2 (Information-Theoretic Risk Reduction via Generator based PI)**.** *Assume: (A1) and (A2) in Theorem D.1, as well as*

(A3) The conditional entropy $H(\psi \mid \mathcal{D})$, which measures the remaining uncertainty about the generator (specification and parameter values) after observing the dataset, is positive[3]*, that is,*

$$H(\psi \mid \mathcal{D}) = \mathbb{E}_{\mathcal{D}}\left[-\int p(\psi \mid \mathcal{D}) \log p(\psi \mid \mathcal{D})\, d\psi\right] > 0\ .$$

Then, under assumptions (A1)–(A3),

$$R^*_{PI} < R^*.$$

Proof. We show that access to generator $\mathcal{P}^*$ during training induces a hypothesis that encodes a more accurate inference of the latent generator ψ, leading to strictly lower population risk at test time, even though $\mathcal{P}^*$ is not available during inference.

Step 1: Bayes-optimal prediction without PI.

Given a dataset $\mathcal{D}$, any predictor without PI must act under the posterior $p(\psi \mid \mathcal{D})$. Define the Bayes-optimal decision rule

$$h^{\text{Bayes}}(\mathcal{D}, \mathbf{x}) := \arg\min_{\widehat{y}} \mathbb{E}_{\psi \sim p(\psi|\mathcal{D})}\ \mathbb{E}_{y \sim p(\cdot|\mathbf{x},\psi)}\big[\ell(\widehat{y}, y)\big].$$

By (A3), $H(\psi \mid \mathcal{D}) > 0$, so the posterior is non-degenerate with positive probability. Hence h^{Bayes} must average over multiple plausible generators, and therefore incurs strictly higher risk than the oracle predictor that knows ψ. Then,

$$R(h^{\psi}) < R(h^{\text{Bayes}}), \quad \text{and} \quad R^* \geq R(h^{\text{Bayes}})\ , \tag{1}$$

where

$$h^{\psi}(\mathcal{D}, \mathbf{x}) := \arg\min_{\widehat{y}} \mathbb{E}_{y \sim p(\cdot|\mathbf{x},\psi)}\big[\ell(\widehat{y}, y)\big].$$

Step 2: PI enables learning of generator-dependent predictors.

Privileged information $\mathcal{P}^*$ provides additional information about ψ at training time. By (A1), there exists a useful teacher $\phi \in \Phi_u$ that exploits $\mathcal{P}^*$ to approximate oracle corrections associated with ψ.

Training with $\mathcal{P}^*$ enables the learner to construct a predictor that approximates the mapping $\mathcal{D} \mapsto \psi$ and thereby approximates h^{ψ}. This induces a parameterization within $\mathcal{H}_{\text{PI}}$ that approximates h^{ψ} more closely than any predictor trained without PI:

$$R(h_{\text{PI}}) \approx R(h^{\psi}), \quad \Rightarrow \quad R^*_{\text{PI}} \leq R(h^{\psi}). \tag{2}$$

Step 3: Elimination of hypothesis-class confounding.

By (A2), for the useful teacher $\phi \in \Phi_u$, there exists $h_\phi \in \mathcal{H}$ such that its behavior over $(\mathcal{D}, \mathbf{x})$ is compatible with the base hypothesis class.

Thus, any improvement induced by ϕ cannot be attributed to a strict increase in the expressivity of $\mathcal{H}$, but must instead arise from improved inference of the latent generator ψ.

Step 4: Combining the bounds.

[3]For continuous ψ, this condition corresponds to the assumption $\text{Var}(\psi \mid \mathcal{D}) > 0$ with positive probability.

Combining inequalities in (1) and (2) above, we get

$$R^*_{\text{PI}} \leq R(h^{\psi}) < R(h^{\text{Bayes}}) \leq R^*,$$

which implies

$$\boxed{R^*_{\text{PI}} < R^*}\,.$$

□

In words, Theorem D.2 states that the generator based PI can reduce asymptotic Bayes risk assuming **(A1)** the teacher space encodes the oracle loss surface, **(A2)** the teacher space is within the representational limits of the learner class, and **(A3)** the generator prior and parameters are not identifiable from the observed dataset (i.e. multiple generators remain plausible given the dataset).

Remarks: In summary, (A1) and (A2) capture the usefulness and realizability of the privileged information: (A1) asserts that the teacher equipped with PI can emulate oracle corrections, while (A2) ensures that the learner can exploit the guidance during training. For example, summary statistics of a dataset, such as mean or variance, can aid with outlier detection. Similarly, knowing the task type (e.g., outlier detection under a GMM prior) via the generator parameters can allow the model to adapt its internal representations or weight updates according to the prior-specified distribution.

In both Meta-PI and Generator-PI cases, a key assumption is (A3), that **privileged information cannot be recovered from the context alone**, which is what allows it to strictly reduce risk. For **meta-features** as PI, (A3) reflects a **model-limited** scenario: even though the information is in principle computable from the dataset, the model cannot extract it efficiently due to architectural or in-context capacity constraints. For **generator program** as PI, (A3) reflects a **data-limited** scenario: the privileged parameters cannot be inferred from a limited dataset context, so they provide information that is fundamentally inaccessible without teacher guidance.

Theorems 3.1 and 3.2 characterize a *population-level* approximation gap between the learner and the PI-augmented hypothesis class, outlining conditions under which the teacher provides a **strictly lower achievable risk** due to enhanced representational capacity.

In the following, we consider a *finite-sample* convergence analysis, where we study learning rates and estimation effects, identifying conditions under which the teacher enables **faster convergence** to this improved risk.

D.2 Finite-Sample Learning and Convergence

Next, we analyze the empirical convergence properties of our approach in finite-data regimes, examining the rate at which the model realizes the benefits of PI in practice.

First, assume that the learner $h \in \mathcal{H}$ may learn the true function t at a slow rate $\alpha_h = 1/2$,

$$R(h) - R(t) \leq \mathcal{O}\left(\frac{|\mathcal{H}|_{\text{VC}}}{\sqrt{n}}\right) + \epsilon_h\,, \tag{6}$$

where $\mathcal{O}(\cdot)$ term is the estimation error and ϵ_h is the approximation error of the learner class $\mathcal{H}$ with respect to the target function t.

Second, assume that a teacher $\phi \in \Phi$ can learn at a faster rate $\beta > \alpha_t$ than the learner;

$$R(\phi) - R(t) \leq \mathcal{O}\left(\frac{|\Phi|_{\text{VC}}}{n^{\beta}}\right) + \epsilon_{\phi}\,, \tag{7}$$

where ϵ_{ϕ} is the approximation error of the teacher function class Φ with respect to the target t.

Finally, assume that the learner can match the teacher (correcting) function at a rate α_{ϕ}

$$R(h) - R(\phi) \leq \mathcal{O}\left(\frac{|\mathcal{H}|_{\text{VC}}}{n^{\alpha_{\phi}}}\right) + \epsilon_s\,, \tag{8}$$

where $\frac{1}{2} \leq \alpha_{\phi} \leq 1$ and ϵ_s ('s' for student; learner is called as student when mimicking the teacher) denotes the approximation error of the learner class $\mathcal{H}$ with respect to the teacher function $\phi \in \Phi_u$.

Summing the inequalities in equation 7 and equation 8 provides an alternative upper bound to the LHS in equation 6:

$$R(h) - R(t) \leq \mathcal{O}\left(\frac{|\mathcal{H}|_{\text{VC}} + |\Phi|_{\text{VC}}}{n^{\min\{\beta,\alpha_\phi\}}}\right) + \epsilon_\phi + \epsilon_s \ . \tag{9}$$

Benefits of learning with a teacher arise, i.e., RHS of equation 9 becomes lower than the RHS of equation 6, when:

(1) the approximation error of the teacher is smaller than that of the learner ($\epsilon_\phi \leq \epsilon_h$), hence the teacher is useful (A1);

(2) the learner can match the teacher ($\epsilon_s \approx 0$) with rate $\alpha_\phi > 1/2$ such that $\min\{\beta, \alpha_\phi\} > 1/2$ (A2); and

(3) the capacity of the teacher class ($|\Phi|_{\text{VC}}$) is small.

E Additional Experiments and Details

E.1 Experiment Setup

Dataset Description: To evaluate different models on in-distribution datasets, we generate data from five distinct priors. For each prior, we sample 250 datasets by uniformly drawing the dimensionality from $[2, 100]$ and randomly sampling other prior-specific variables, such as the number of GMM components, SCM DAG widths and depths, and the dimensions perturbed to create outliers. Since both the 2-layer and 10-layer models are trained with a maximum context length of 5,000, we restrict all evaluation datasets to have at most 5,000 context points. Outliers constitute 5%–20% of the test set. For the 2-layer models and their variants, we use the GMM subset for in-distribution evaluation. For the 10-layer models, we evaluate on the full mixed-prior test set.

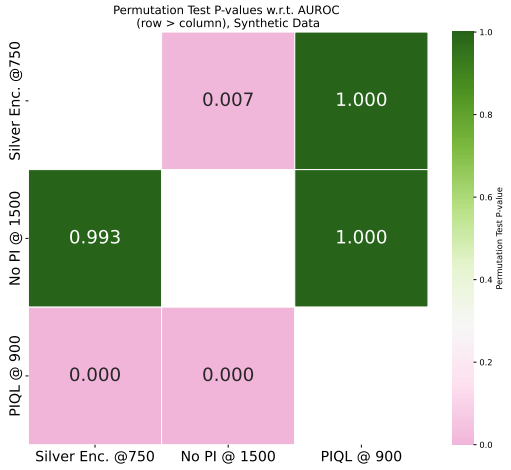
For out-of-distribution evaluation, we evaluate the 10-layer models on ADBench Han et al. (2022), a widely used benchmark suite for tabular outlier detection that aggregates diverse real-world datasets across multiple domains and anomaly types.

Training Configurations: We train the 2-layer model for 1,500 epochs on sequences of length 5,000, using 1,000 steps per epoch, batch size 2, and 4 v100-32G GPUs. The model uses a 256-dimensional embedding, 2 attention heads, hidden dimension 512, and 500 learned routing tokens, with the last layer also using routing attention. Optimization is performed with a peak learning rate of 10^{-4} with AdamW optimizer and cosine learning rate decay. For the enlarged model, the embedding dimension is increased from 256 to 512, the number of attention heads from 2 to 8, the hidden dimension from 512 to 1024, and the depth from 2 layers to 10 layers. This enlarged configuration substantially increases model capacity, enabling stronger sequence/context modeling while keeping the overall training protocol unchanged (we train on 4 L40 GPUs).

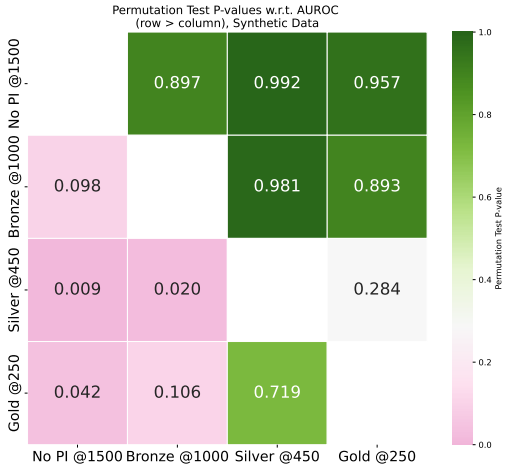
E.2 Comparing PI vs. No-PI TFMs across Training Epochs

We compare PIQL, Silver Enc., and No-PI when they all reach the same loss as shown in Figure 2. Specifically, PIQL (transferred) and Silver Enc. (no transfer) reach No-PI's loss at convergence @1500 epochs respectively @900 and @750 epochs. Figure 5a presents the corresponding permutation test results, showing that Silver Enc@750 outperforms No-PI@1500, and that PIQL@900 significantly outperforms both.

Similarly, we compare Gold@250, Silver@450, Bronze@1000, and No-PI@1500 epochs for 2-layer architecture when they all reach the same loss as shown in Figure 1. Figure 5b presents the corresponding permutation test results, showing that all privileged models significantly outperform the standard base model No-PI@1500 even at earlier epochs, with earlier gains observed as PI quality increases from bronze to gold. Moreover, generator-PI based Silver and Gold models are comparable at @450 and @250 epochs, respectively, both of which outperform Bronze @1000 epochs.



(a) Paired permutation test of PIQL @900, Silver Enc. @750, and No-PI @1500 epochs for the 10-layer architecture (as shown in Fig. 2), taken at varying epochs as they reach the same loss. PIQL@900 significantly outperforms both Silver Enc@750 and No-PI@1500.



(b) Paired permutation test of Gold @250, Silver @450, Bronze @1000, and No-PI @1500 epochs for the 2-layer architecture (as shown in Fig. 1), taken at varying epochs as they reach the same loss. All privileged models significantly outperform No-PI@1500 at much earlier epochs, improving from bronze to gold.

Figure 5: Pairwise permutation test results for the 10-layer and 2-layer models.

Table 9: **Training-loss comparison of fixed versus annealed teacher forcing.** For each annealed checkpoint, we report the earliest epoch at which the fixed model reaches the same training loss.

Annealed reference	Reference loss	Fixed matching epoch
Annealed @ 500	0.058	218 (**2.29× earlier**)
Annealed @ 750	0.046	355 (**2.11× earlier**)
Annealed @ 1,000	0.043	422 (**2.37× earlier**)

E.3 Individual Dataset Results in ADBench

Table 14 and Table 15 show individual AUROC and AUPRC results on ADBench. Table **??** shows the permutation test results between PIQL and the base model at varying epochs. PIQL shows statistically similar results to the standard base model at earlier epochs, exhibiting faster learning rate.

E.4 Ablation on Teacher Annealing vs. Fixed Teacher Forcing

To test whether annealing is necessary, we compare our default annealing schedule against fixed teacher forcing. Let α denote the weight on true PI.

- **Fixed:** $\alpha = 0.5$ throughout training, so the model always receives equal contributions from true PI and student-predicted PI.
- **Annealed:** α decreases from 1.0 to 0.0, gradually replacing true PI with student-predicted PI and matching student-only inference. This is the default setting used in the paper.

The results on training loss is shown in Table 9. Fixed teacher forcing achieves lower training loss earlier because it retains access to true PI, which acts as a shortcut unavailable at inference.

The AUROC result is showin in Table 10. Despite its faster loss reduction, fixed teacher forcing generalizes worse. The fixed model's Avg. AUROC decreases from 0.978 to 0.973 even as its training loss improves, revealing a train–inference gap: it learns to rely on the true-PI component that is unavailable at deployment. Annealing removes this shortcut by gradually transitioning to fully student-predicted PI, yielding consistently and significantly better AUROC. This ablation confirms that annealing is necessary for effective PI transfer rather than merely an auxiliary training choice.

Table 10: **Generalization under fixed versus annealed teacher forcing.** Annealing yields consistently higher Avg. AUROC on GMM in-distribution data.

Epoch	Fixed ($\alpha = 0.5$)	Annealed	p-value
500	0.978	**0.985**	**0.01**
750	0.976	**0.988**	< 0.001
1,000	0.973	**0.990**	< 0.001

E.5 Compute Overhead and Wall-Clock Efficiency

We provide a detailed analysis of the FLOP between no-PIQL and PIQL models. First, the program encoder does not run during PIQL training. Each generator program is a lightweight description of a data-generating process, so its embedding is computed once offline and cached as a single vector. The remaining online overhead comes primarily from the lightweight student module and PI input processing. For the 10-layer model, with $N = 5000$ context points, $R = 500$ router tokens, hidden dimension $d = 512$, and FFN dimension $f = 1024$, the transformer backbone requires approximately

$$10\left(8Nd^2 + 4N^2d + 4Ndf + 4(N+R)d^2 + 8NRd\right) \approx 887.6 \text{ B FLOPs.}$$

The PI input encoders require approximately 5.8 B FLOPs, while the PIQL-specific student requires approximately 30.4 B FLOPs. Thus, the total PIQL cost is approximately

$$887.6 + 5.8 + 30.4 = 923.8 \text{ B FLOPs,}$$

corresponding to an incremental arithmetic overhead of

$$\frac{5.8 + 30.4}{887.6} \approx 0.041,$$

or approximately 4.1%.

This theoretical FLOP overhead is smaller than the measured per-epoch runtime overhead. In our implementation, training time increases from 3.4 to 4.2 minutes per epoch, corresponding to approximately 24% additional wall-clock cost. The discrepancy arises because FLOP counts capture arithmetic operations but not several systems-level costs, including cached program-embedding retrieval and CPU-to-GPU transfer, additional activation and memory traffic, synchronization, and unfused student operations. Thus, the current implementation is not yet fully compute-efficient, particularly in the PI-loading and student pipelines.

Importantly, the reduction in required training epochs outweighs this per-epoch overhead. As shown in Table 3, PIQL reaches the matched training-loss target in 780 rather than 1,200 epochs, reducing total training time from 4,080 to 3,276 minutes, or a $1.25\times$ end-to-end wall-clock speedup. Caching PI embeddings directly on device, batching retrieval, and fusing student operations could further narrow the gap between theoretical FLOPs and measured runtime.

E.6 PI Quality and Student Approximation

Learned versus random PI. To isolate the contribution of PI's learned semantic content, we vary only the amount of program-encoder training while keeping the downstream architecture and PI token count fixed. Program-encoder pretraining is lightweight, taking approximately one hour for 256K synthetic programs. Table 11 shows that random, untrained PI provides little benefit over No PI, whereas learned PI substantially improves both training loss and AUROC. Increasing program-encoder training from 64K to 256K programs further improves the mean AUROC from 0.987 to 0.989, although this difference is not statistically significant ($p = 0.29$). In contrast, the fully learned 256K PI model significantly outperforms both No PI and random PI ($p < 0.001$).

Student approximation error. We additionally quantify how accurately the student reconstructs the privileged representation using its ℓ_2 reconstruction error. As shown in Table 12, the error decreases rapidly from 5.31 at epoch 50 to 2.62 at epoch 500, demonstrating that the student successfully learns to approximate the teacher representation. After the true-PI teacher signal is removed, the

Table 11: **Effect of PI quality.** AUROC is reported as mean ± standard deviation. The p-value compares each representation against learned PI trained on 256K programs.

PI representation	Loss @ 600	AUROC	p-value
No PI	0.77	0.979 ± 0.052	< 0.001
Random, untrained PI	0.74	0.978 ± 0.071	< 0.001
Learned PI (64K programs)	0.53	0.987 ± 0.052	0.29
Learned PI (256K programs)	**0.49**	$\mathbf{0.989 \pm 0.048}$	–

Table 12: **Student approximation error during PIQL training.** The student rapidly learns to approximate the privileged representation; after teacher removal, reconstruction error increases slightly even as downstream performance continues to improve.

Training epoch	ℓ_2 reconstruction error
50	5.31
300	2.86
500	**2.62**
700	2.92
1,000	3.14
1,300	3.02

reconstruction error increases modestly, reaching 3.14 at epoch 1,000, while downstream performance continues to improve. This suggests that exact reconstruction of the teacher embedding is not itself the optimization objective: once the student has acquired a useful approximation, its representation can continue adapting toward the downstream task after teacher removal.

E.7 LLM-GENERATED META-FEATURES FOR META-PI++

For Meta-PI++, we prompt a LLM (ChatGPT) to describe 91 additional summary statistics for any dataset besides the 9 existing ones: mean, variance, skewness, kurtosis, q10, q25, q50, q75, and q90. LLM provides the additional statistics as grouped into seven categories as shown in Table 13: 39 fine-grained quantile features describing the marginal distribution, 15 robust spread measures capturing variability and interval widths, 8 extreme-value statistics summarizing minima, maxima, ranges, and absolute magnitudes, 8 higher-moment statistics characterizing distributional shape beyond skewness and kurtosis, 12 tail/outlier proportion features measuring mass in extreme regions, 6 sign/zero/finite-value indicators capturing numerical and sparsity patterns, and 3 asymmetry measures quantifying skewness and mean–median imbalance. We also work with the LLM to generate the code that produces the described meta-features.

Table 13 presents the list and the detailed description of the additional dataset-level statistics we use as meta-features in Meta-PI++.

Table 13: List of the additional LLM-generated meta-features for Meta-PI++.

Category	Count	Additional features
Extra quantiles	39	$q_{0.001}$, $q_{0.01}$, $q_{0.025}$, $q_{0.05}$, $q_{0.075}$, $q_{0.125}$, $q_{0.15}$, $q_{0.175}$, $q_{0.20}$, $q_{0.225}$, $q_{0.275}$, $q_{0.30}$, $q_{0.325}$, $q_{0.35}$, $q_{0.375}$, $q_{0.40}$, $q_{0.425}$, $q_{0.45}$, $q_{0.475}$, $q_{0.525}$, $q_{0.55}$, $q_{0.575}$, $q_{0.60}$, $q_{0.625}$, $q_{0.65}$, $q_{0.675}$, $q_{0.70}$, $q_{0.725}$, $q_{0.775}$, $q_{0.80}$, $q_{0.825}$, $q_{0.85}$, $q_{0.875}$, $q_{0.925}$, $q_{0.95}$, $q_{0.975}$, $q_{0.99}$, $q_{0.995}$, $q_{0.999}$
Robust spread / interval widths	15	IQR, 90% range, 80% range, 50% range, lower-tail width $(q_{0.25} - q_{0.10})$, upper-tail width $(q_{0.90} - q_{0.75})$, lower extreme-tail width $(q_{0.10} - q_{0.01})$, upper extreme-tail width $(q_{0.99} - q_{0.90})$, median absolute deviation, mean absolute deviation, std-to-mean ratio, IQR-to-std ratio, range-to-std ratio, $q_{0.90}/\lvert q_{0.10}\rvert$, $q_{0.75}/\lvert q_{0.25}\rvert$
Min/max/extreme statistics	8	Minimum, maximum, range, absolute minimum, absolute maximum, mean absolute value, median absolute value, RMS
Higher moments	8	5th standardized moment, 6th standardized moment, 2nd raw moment, 3rd raw moment, 4th raw moment, 2nd central moment, 3rd central moment, 4th central moment
Tail / outlier proportions	12	Fraction below $q_{0.01}$, below $q_{0.05}$, below $q_{0.10}$, above $q_{0.90}$, above $q_{0.95}$, above $q_{0.99}$, fraction with $\lvert x - \mu\rvert > \sigma$, $\lvert x - \mu\rvert > 2\sigma$, $\lvert x - \mu\rvert > 3\sigma$, fraction below lower IQR bound, above upper IQR bound, total IQR outlier fraction
Sign / zero / finite behavior	6	Fraction positive, fraction negative, fraction zero, fraction NaN, fraction Inf, fraction finite
Shape asymmetry statistics	3	Bowley skewness, tail asymmetry, normalized mean–median gap
Total	**91**	Additional per-dimension statistics

Table 14: Individual dataset performances on ADBench w.r.t. **AUROC** (± standard dev. over five seeds). **Best** and Second-best results are highlighted. PIQL reaches statistically similar performance to base model (no-PI) at earlier epochs (see Table **??**)

Dataset	PIQL@650	PIQL@700	No-PI@920	No-PI@950	No-PI@1000	No-PI@1050
agnews	60.26$_{\pm 1.46}$	62.85$_{\pm 0.82}$	63.50$_{\pm 2.72}$	64.58$_{\pm 2.42}$	65.74$_{\pm 2.02}$	**65.82**$_{\pm 2.37}$
20news	57.12$_{\pm 1.25}$	58.91$_{\pm 2.34}$	59.50$_{\pm 1.96}$	60.98$_{\pm 1.91}$	60.69$_{\pm 1.57}$	**61.08**$_{\pm 1.74}$
SVHN	58.57$_{\pm 2.39}$	58.57$_{\pm 3.41}$	59.76$_{\pm 1.58}$	59.73$_{\pm 1.46}$	60.24$_{\pm 1.39}$	**61.19**$_{\pm 1.04}$
FashionMNIST	81.87$_{\pm 4.72}$	80.93$_{\pm 7.20}$	84.55$_{\pm 3.41}$	84.69$_{\pm 3.15}$	85.61$_{\pm 2.28}$	**86.72**$_{\pm 0.92}$
MNIST-C	76.43$_{\pm 4.71}$	77.15$_{\pm 4.07}$	78.33$_{\pm 1.26}$	78.35$_{\pm 2.00}$	79.05$_{\pm 2.38}$	**80.43**$_{\pm 1.97}$
CIFAR10	64.42$_{\pm 1.59}$	64.26$_{\pm 2.05}$	64.78$_{\pm 1.78}$	65.09$_{\pm 1.72}$	65.52$_{\pm 1.33}$	**65.96**$_{\pm 1.02}$
MVTec-AD	82.48$_{\pm 0.55}$	82.59$_{\pm 0.58}$	84.09$_{\pm 1.06}$	84.65$_{\pm 1.14}$	**84.91**$_{\pm 0.80}$	84.89$_{\pm 1.02}$
wilt	91.79$_{\pm 0.00}$	93.83$_{\pm 0.00}$	90.21$_{\pm 0.00}$	93.70$_{\pm 0.00}$	**94.18**$_{\pm 0.00}$	91.96$_{\pm 0.00}$
mnist	67.89$_{\pm 0.00}$	81.80$_{\pm 0.00}$	84.57$_{\pm 0.00}$	82.76$_{\pm 0.00}$	**84.67**$_{\pm 0.00}$	80.01$_{\pm 0.00}$
hepatitis	99.89$_{\pm 0.22}$	99.89$_{\pm 0.22}$	99.92$_{\pm 0.17}$	99.90$_{\pm 0.19}$	99.39$_{\pm 0.84}$	**99.92**$_{\pm 0.17}$
yelp	58.59$_{\pm 8.74}$	58.27$_{\pm 6.25}$	60.72$_{\pm 6.06}$	61.34$_{\pm 5.76}$	62.21$_{\pm 3.79}$	**62.81**$_{\pm 3.68}$
spambase	80.26$_{\pm 0.00}$	**83.44**$_{\pm 0.00}$	83.05$_{\pm 0.00}$	83.18$_{\pm 0.00}$	82.98$_{\pm 0.00}$	81.74$_{\pm 0.00}$
smtp	91.50$_{\pm 2.22}$	94.79$_{\pm 1.36}$	91.84$_{\pm 2.32}$	91.07$_{\pm 3.04}$	**95.31**$_{\pm 0.63}$	87.39$_{\pm 4.10}$
speech	40.30$_{\pm 3.18}$	**40.79**$_{\pm 2.61}$	40.02$_{\pm 2.99}$	40.11$_{\pm 1.90}$	40.16$_{\pm 2.01}$	39.13$_{\pm 1.81}$
pima	80.17$_{\pm 1.33}$	80.29$_{\pm 1.12}$	79.39$_{\pm 1.44}$	79.73$_{\pm 1.57}$	80.78$_{\pm 1.69}$	**81.18**$_{\pm 1.08}$
internetads	50.92$_{\pm 9.90}$	54.62$_{\pm 4.28}$	54.95$_{\pm 5.28}$	54.01$_{\pm 6.64}$	55.37$_{\pm 4.73}$	**55.84**$_{\pm 3.30}$
stamps	**98.75**$_{\pm 0.55}$	98.17$_{\pm 0.49}$	97.81$_{\pm 0.77}$	98.33$_{\pm 0.52}$	98.57$_{\pm 0.62}$	97.56$_{\pm 0.95}$
cardiotocography	**62.79**$_{\pm 0.00}$	60.18$_{\pm 0.00}$	59.82$_{\pm 0.00}$	62.62$_{\pm 0.00}$	59.58$_{\pm 0.00}$	61.46$_{\pm 0.00}$
imdb	**49.85**$_{\pm 3.42}$	48.06$_{\pm 3.26}$	48.75$_{\pm 3.80}$	48.33$_{\pm 3.61}$	47.94$_{\pm 4.01}$	49.36$_{\pm 4.66}$
aloi	51.97$_{\pm 0.46}$	51.22$_{\pm 0.69}$	52.01$_{\pm 0.70}$	51.92$_{\pm 0.38}$	**53.49**$_{\pm 1.16}$	52.03$_{\pm 0.24}$
wpbc	91.24$_{\pm 1.94}$	89.82$_{\pm 1.38}$	92.35$_{\pm 0.82}$	**93.03**$_{\pm 1.18}$	90.08$_{\pm 0.76}$	90.71$_{\pm 1.98}$
yeast	47.56$_{\pm 0.00}$	**51.58**$_{\pm 0.00}$	48.47$_{\pm 0.00}$	49.46$_{\pm 0.00}$	49.48$_{\pm 0.00}$	49.02$_{\pm 0.00}$
fraud	95.23$_{\pm 0.75}$	95.33$_{\pm 0.84}$	95.73$_{\pm 0.56}$	**96.11**$_{\pm 0.76}$	95.90$_{\pm 0.88}$	95.67$_{\pm 1.12}$
musk	99.99$_{\pm 0.01}$	**99.99**$_{\pm 0.01}$	99.96$_{\pm 0.07}$	99.98$_{\pm 0.02}$	99.96$_{\pm 0.02}$	99.97$_{\pm 0.04}$
fault	54.31$_{\pm 0.00}$	**57.44**$_{\pm 0.00}$	55.35$_{\pm 0.00}$	56.90$_{\pm 0.00}$	55.50$_{\pm 0.00}$	55.21$_{\pm 0.00}$
amazon	52.69$_{\pm 5.17}$	52.75$_{\pm 4.77}$	55.18$_{\pm 4.08}$	55.17$_{\pm 4.02}$	55.90$_{\pm 2.97}$	**56.70**$_{\pm 1.57}$
campaign	63.31$_{\pm 1.39}$	72.99$_{\pm 0.88}$	71.51$_{\pm 1.26}$	66.73$_{\pm 1.33}$	72.31$_{\pm 1.40}$	**73.29**$_{\pm 1.11}$
breastw	**98.81**$_{\pm 0.27}$	98.69$_{\pm 0.32}$	98.42$_{\pm 0.58}$	97.47$_{\pm 1.04}$	96.82$_{\pm 0.99}$	97.23$_{\pm 0.97}$
glass	98.15$_{\pm 0.91}$	98.70$_{\pm 0.96}$	97.80$_{\pm 0.95}$	95.29$_{\pm 3.86}$	98.15$_{\pm 1.30}$	**98.72**$_{\pm 1.12}$
thyroid	**98.40**$_{\pm 0.00}$	98.23$_{\pm 0.00}$	96.64$_{\pm 0.00}$	97.40$_{\pm 0.00}$	95.56$_{\pm 0.00}$	96.50$_{\pm 0.00}$
http	99.97$_{\pm 0.02}$	**99.99**$_{\pm 0.01}$	99.92$_{\pm 0.13}$	99.98$_{\pm 0.02}$	99.96$_{\pm 0.03}$	99.98$_{\pm 0.02}$
optdigits	82.01$_{\pm 0.00}$	75.13$_{\pm 0.00}$	82.78$_{\pm 0.00}$	72.82$_{\pm 0.00}$	82.54$_{\pm 0.00}$	**85.83**$_{\pm 0.00}$
skin	82.43$_{\pm 2.82}$	**86.35**$_{\pm 1.92}$	52.70$_{\pm 3.25}$	62.11$_{\pm 6.90}$	86.20$_{\pm 2.63}$	72.12$_{\pm 4.32}$
backdoor	**61.67**$_{\pm 16.45}$	60.50$_{\pm 9.22}$	54.49$_{\pm 19.27}$	55.44$_{\pm 19.54}$	57.21$_{\pm 21.43}$	46.53$_{\pm 9.69}$
letter	39.22$_{\pm 0.00}$	40.28$_{\pm 0.00}$	**40.61**$_{\pm 0.00}$	39.46$_{\pm 0.00}$	39.00$_{\pm 0.00}$	39.15$_{\pm 0.00}$
vertebral	79.61$_{\pm 2.98}$	81.20$_{\pm 1.64}$	**82.07**$_{\pm 2.69}$	81.74$_{\pm 2.86}$	77.37$_{\pm 5.43}$	70.07$_{\pm 4.12}$
satimage-2	**99.62**$_{\pm 0.00}$	99.24$_{\pm 0.00}$	99.14$_{\pm 0.00}$	99.08$_{\pm 0.00}$	98.84$_{\pm 0.00}$	99.24$_{\pm 0.00}$
wdbc	99.17$_{\pm 0.26}$	98.85$_{\pm 0.47}$	99.11$_{\pm 0.26}$	**99.18**$_{\pm 0.20}$	98.82$_{\pm 0.38}$	98.91$_{\pm 0.45}$
celeba	69.40$_{\pm 0.77}$	**72.92**$_{\pm 0.63}$	63.31$_{\pm 1.18}$	66.84$_{\pm 0.99}$	69.32$_{\pm 1.02}$	63.96$_{\pm 1.32}$
landsat	55.54$_{\pm 0.00}$	**59.61**$_{\pm 0.00}$	54.23$_{\pm 0.00}$	49.45$_{\pm 0.00}$	50.77$_{\pm 0.00}$	58.14$_{\pm 0.00}$
vowels	86.59$_{\pm 0.00}$	85.31$_{\pm 0.00}$	82.62$_{\pm 0.00}$	86.45$_{\pm 0.00}$	**86.73**$_{\pm 0.00}$	86.61$_{\pm 0.00}$
shuttle	99.38$_{\pm 0.30}$	99.29$_{\pm 0.40}$	99.39$_{\pm 0.22}$	**99.60**$_{\pm 0.15}$	99.54$_{\pm 0.22}$	99.45$_{\pm 0.42}$
wine	99.91$_{\pm 0.17}$	99.93$_{\pm 0.14}$	99.92$_{\pm 0.17}$	99.94$_{\pm 0.12}$	99.89$_{\pm 0.21}$	**99.94**$_{\pm 0.12}$
annthyroid	77.41$_{\pm 0.00}$	81.51$_{\pm 0.00}$	85.14$_{\pm 0.00}$	85.66$_{\pm 0.00}$	**86.95**$_{\pm 0.00}$	81.47$_{\pm 0.00}$
census	60.92$_{\pm 3.88}$	58.97$_{\pm 4.10}$	61.43$_{\pm 5.70}$	**62.53**$_{\pm 3.41}$	62.47$_{\pm 4.34}$	61.16$_{\pm 6.26}$
waveform	70.61$_{\pm 0.00}$	71.63$_{\pm 0.00}$	72.35$_{\pm 0.00}$	**72.88**$_{\pm 0.00}$	71.40$_{\pm 0.00}$	70.99$_{\pm 0.00}$
cardio	**92.31**$_{\pm 0.00}$	91.29$_{\pm 0.00}$	90.97$_{\pm 0.00}$	91.88$_{\pm 0.00}$	90.58$_{\pm 0.00}$	90.89$_{\pm 0.00}$
wbc	99.44$_{\pm 0.74}$	**99.54**$_{\pm 0.61}$	99.25$_{\pm 0.83}$	99.07$_{\pm 0.99}$	98.97$_{\pm 1.13}$	99.23$_{\pm 0.91}$
ionosphere	98.05$_{\pm 0.81}$	98.27$_{\pm 0.82}$	98.34$_{\pm 0.58}$	**98.69**$_{\pm 0.45}$	98.38$_{\pm 0.60}$	98.54$_{\pm 0.53}$
cover	99.60$_{\pm 0.09}$	99.54$_{\pm 0.11}$	99.51$_{\pm 0.13}$	99.60$_{\pm 0.10}$	99.66$_{\pm 0.10}$	**99.66**$_{\pm 0.10}$
pageblocks	88.29$_{\pm 0.00}$	87.85$_{\pm 0.00}$	86.82$_{\pm 0.00}$	88.23$_{\pm 0.00}$	87.50$_{\pm 0.00}$	**88.73**$_{\pm 0.00}$
pendigits	98.83$_{\pm 0.00}$	99.07$_{\pm 0.00}$	**99.52**$_{\pm 0.00}$	99.40$_{\pm 0.00}$	99.41$_{\pm 0.00}$	99.51$_{\pm 0.00}$
satellite	74.96$_{\pm 0.00}$	75.59$_{\pm 0.00}$	73.58$_{\pm 0.00}$	73.31$_{\pm 0.00}$	73.66$_{\pm 0.00}$	**76.05**$_{\pm 0.00}$
magic.gamma	85.43$_{\pm 0.14}$	86.70$_{\pm 0.12}$	86.96$_{\pm 0.21}$	**88.99**$_{\pm 0.19}$	84.12$_{\pm 0.22}$	86.64$_{\pm 0.10}$
donors	**99.72**$_{\pm 0.06}$	99.70$_{\pm 0.03}$	99.31$_{\pm 0.50}$	99.61$_{\pm 0.04}$	98.95$_{\pm 0.93}$	99.29$_{\pm 0.28}$
lymphography	99.98$_{\pm 0.04}$	99.98$_{\pm 0.04}$	**99.98**$_{\pm 0.04}$	99.93$_{\pm 0.09}$	99.93$_{\pm 0.09}$	99.96$_{\pm 0.05}$
mammography	83.22$_{\pm 4.02}$	86.31$_{\pm 1.70}$	83.69$_{\pm 1.14}$	**88.41**$_{\pm 0.59}$	65.52$_{\pm 1.13}$	77.80$_{\pm 1.78}$

Table 15: Individual dataset performances on ADBench w.r.t. **AUPRC** ($\pm$ standard dev. over five seeds). **Best** and Second-best results are highlighted. PIQL reaches statistically similar performance to base model (no-PI) at earlier epochs (see Table **??**).

Dataset	PIQL@650	PIQL@700	No-PI@920	No-PI@950	No-PI@1000	No-PI@1050
agnews	$14.96_{\pm 0.83}$	$16.41_{\pm 0.51}$	$16.86_{\pm 1.90}$	$17.20_{\pm 1.41}$	**$17.64_{\pm 1.38}$**	$17.63_{\pm 1.56}$
20news	$13.87_{\pm 0.53}$	$14.55_{\pm 1.27}$	$14.75_{\pm 0.57}$	**$16.08_{\pm 0.89}$**	$15.33_{\pm 0.53}$	$15.13_{\pm 0.60}$
SVHN	$13.57_{\pm 1.14}$	$13.68_{\pm 1.61}$	$14.33_{\pm 0.84}$	$14.24_{\pm 0.79}$	$14.44_{\pm 0.84}$	**$14.85_{\pm 0.63}$**
FashionMNIST	$44.32_{\pm 7.98}$	$44.71_{\pm 11.42}$	$49.68_{\pm 5.01}$	**$50.42_{\pm 5.66}$**	$48.92_{\pm 7.28}$	$50.38_{\pm 4.22}$
MNIST-C	$38.40_{\pm 7.07}$	$38.86_{\pm 6.82}$	$41.87_{\pm 2.17}$	$41.64_{\pm 3.09}$	$41.61_{\pm 4.97}$	**$44.58_{\pm 1.99}$**
CIFAR10	$17.69_{\pm 1.03}$	$17.61_{\pm 1.62}$	$18.12_{\pm 1.21}$	$18.36_{\pm 1.23}$	$18.53_{\pm 1.29}$	**$18.84_{\pm 0.89}$**
MVTec-AD	$79.09_{\pm 1.02}$	$78.98_{\pm 1.13}$	$80.01_{\pm 1.21}$	$80.81_{\pm 1.24}$	$81.00_{\pm 0.95}$	**$81.33_{\pm 1.01}$**
wilt	$46.61_{\pm 0.00}$	$55.84_{\pm 0.00}$	$42.02_{\pm 0.00}$	$59.61_{\pm 0.00}$	**$59.82_{\pm 0.00}$**	$50.02_{\pm 0.00}$
mnist	$44.25_{\pm 0.00}$	$51.35_{\pm 0.00}$	$58.19_{\pm 0.00}$	$55.11_{\pm 0.00}$	**$58.69_{\pm 0.00}$**	$56.72_{\pm 0.00}$
hepatitis	$99.57_{\pm 0.86}$	$99.57_{\pm 0.86}$	$99.71_{\pm 0.57}$	$99.65_{\pm 0.71}$	$98.82_{\pm 1.39}$	**$99.72_{\pm 0.57}$**
yelp	$13.61_{\pm 3.77}$	$12.97_{\pm 2.17}$	**$14.22_{\pm 2.26}$**	$13.72_{\pm 2.03}$	$14.01_{\pm 1.57}$	$14.11_{\pm 1.26}$
spambase	$82.05_{\pm 0.00}$	$82.15_{\pm 0.00}$	**$82.46_{\pm 0.00}$**	$82.08_{\pm 0.00}$	$81.69_{\pm 0.00}$	$81.31_{\pm 0.00}$
smtp	**$47.97_{\pm 14.45}$**	$43.34_{\pm 13.02}$	$24.10_{\pm 19.03}$	$37.56_{\pm 9.13}$	$42.13_{\pm 11.71}$	$28.82_{\pm 8.15}$
speech	$2.73_{\pm 0.30}$	$2.73_{\pm 0.21}$	$2.74_{\pm 0.29}$	$2.78_{\pm 0.30}$	**$2.82_{\pm 0.34}$**	$2.69_{\pm 0.24}$
pima	$79.29_{\pm 1.94}$	$79.71_{\pm 1.85}$	$78.59_{\pm 2.23}$	$78.98_{\pm 1.93}$	$79.62_{\pm 2.21}$	**$80.01_{\pm 1.94}$**
internetads	$38.04_{\pm 8.26}$	$40.88_{\pm 7.68}$	$40.24_{\pm 7.24}$	$39.38_{\pm 6.67}$	$40.53_{\pm 6.37}$	**$41.23_{\pm 5.88}$**
stamps	**$90.56_{\pm 5.29}$**	$87.77_{\pm 4.82}$	$86.90_{\pm 4.72}$	$89.04_{\pm 4.64}$	$89.79_{\pm 5.28}$	$88.80_{\pm 4.41}$
cardiotocography	**$57.56_{\pm 0.00}$**	$55.09_{\pm 0.00}$	$55.03_{\pm 0.00}$	$56.98_{\pm 0.00}$	$54.42_{\pm 0.00}$	$56.37_{\pm 0.00}$
imdb	**$9.46_{\pm 1.12}$**	$8.94_{\pm 0.76}$	$9.16_{\pm 1.06}$	$9.07_{\pm 0.97}$	$8.97_{\pm 0.95}$	$9.30_{\pm 1.29}$
aloi	**$6.68_{\pm 0.24}$**	$6.34_{\pm 0.02}$	$6.45_{\pm 0.07}$	$6.36_{\pm 0.06}$	$6.57_{\pm 0.13}$	$6.33_{\pm 0.03}$
wpbc	$80.84_{\pm 4.45}$	$79.70_{\pm 3.98}$	$83.11_{\pm 3.74}$	**$83.37_{\pm 4.06}$**	$81.31_{\pm 3.55}$	$81.22_{\pm 4.34}$
yeast	$50.52_{\pm 0.00}$	**$52.69_{\pm 0.00}$**	$51.56_{\pm 0.00}$	$52.24_{\pm 0.00}$	$52.13_{\pm 0.00}$	$51.68_{\pm 0.00}$
fraud	$42.27_{\pm 11.92}$	$37.40_{\pm 10.47}$	$45.84_{\pm 12.15}$	$41.85_{\pm 13.52}$	$42.54_{\pm 10.95}$	**$48.73_{\pm 14.70}$**
musk	**$99.88_{\pm 0.18}$**	$99.77_{\pm 0.24}$	$99.45_{\pm 0.89}$	$99.65_{\pm 0.34}$	$99.50_{\pm 0.30}$	$99.62_{\pm 0.59}$
fault	$61.22_{\pm 0.00}$	$62.34_{\pm 0.00}$	$61.30_{\pm 0.00}$	**$62.53_{\pm 0.00}$**	$61.04_{\pm 0.00}$	$61.64_{\pm 0.00}$
amazon	$10.38_{\pm 1.46}$	$10.32_{\pm 1.23}$	$10.97_{\pm 1.10}$	$10.87_{\pm 1.06}$	$11.00_{\pm 0.77}$	**$11.31_{\pm 0.40}$**
campaign	$38.62_{\pm 1.02}$	$43.33_{\pm 0.87}$	$43.46_{\pm 1.03}$	$40.21_{\pm 0.94}$	$43.52_{\pm 1.51}$	**$44.73_{\pm 0.90}$**
breastw	**$98.45_{\pm 0.30}$**	$98.09_{\pm 0.49}$	$97.60_{\pm 0.93}$	$96.02_{\pm 1.27}$	$94.94_{\pm 1.39}$	$95.27_{\pm 0.96}$
glass	$77.35_{\pm 10.46}$	$81.92_{\pm 9.60}$	$73.11_{\pm 12.12}$	$71.94_{\pm 13.48}$	$76.07_{\pm 13.64}$	**$82.38_{\pm 11.50}$**
thyroid	**$72.20_{\pm 0.00}$**	$69.61_{\pm 0.00}$	$57.88_{\pm 0.00}$	$66.19_{\pm 0.00}$	$52.60_{\pm 0.00}$	$64.09_{\pm 0.00}$
http	$97.63_{\pm 2.74}$	**$99.16_{\pm 1.21}$**	$97.76_{\pm 2.35}$	$97.47_{\pm 2.85}$	$94.19_{\pm 4.19}$	$97.33_{\pm 2.80}$
optdigits	$15.23_{\pm 0.00}$	$11.88_{\pm 0.00}$	$16.20_{\pm 0.00}$	$11.78_{\pm 0.00}$	$16.25_{\pm 0.00}$	**$19.30_{\pm 0.00}$**
skin	$64.06_{\pm 4.28}$	$76.97_{\pm 2.24}$	$43.78_{\pm 2.67}$	$46.52_{\pm 8.07}$	**$77.78_{\pm 2.91}$**	$59.51_{\pm 3.77}$
backdoor	$9.84_{\pm 3.74}$	$9.55_{\pm 1.40}$	$8.10_{\pm 4.26}$	$8.77_{\pm 3.01}$	**$10.45_{\pm 5.42}$**	$7.07_{\pm 2.17}$
letter	$9.06_{\pm 0.00}$	$9.19_{\pm 0.00}$	**$9.25_{\pm 0.00}$**	$9.10_{\pm 0.00}$	$9.04_{\pm 0.00}$	$9.08_{\pm 0.00}$
vertebral	$51.54_{\pm 6.33}$	$52.63_{\pm 4.30}$	$56.97_{\pm 6.83}$	**$58.91_{\pm 5.66}$**	$51.50_{\pm 8.45}$	$47.63_{\pm 5.89}$
satimage-2	$88.35_{\pm 0.00}$	**$90.05_{\pm 0.00}$**	$89.58_{\pm 0.00}$	$87.33_{\pm 0.00}$	$82.36_{\pm 0.00}$	$86.39_{\pm 0.00}$
wdbc	**$82.54_{\pm 4.97}$**	$74.76_{\pm 9.40}$	$77.10_{\pm 5.60}$	$78.85_{\pm 5.57}$	$70.67_{\pm 8.13}$	$75.09_{\pm 6.50}$
celeba	$8.05_{\pm 0.55}$	**$8.85_{\pm 0.64}$**	$6.66_{\pm 0.57}$	$7.61_{\pm 0.59}$	$7.98_{\pm 0.73}$	$6.42_{\pm 0.51}$
landsat	$38.72_{\pm 0.00}$	**$43.29_{\pm 0.00}$**	$37.26_{\pm 0.00}$	$33.92_{\pm 0.00}$	$35.48_{\pm 0.00}$	$41.83_{\pm 0.00}$
vowels	$42.19_{\pm 0.00}$	$42.20_{\pm 0.00}$	$36.20_{\pm 0.00}$	$41.82_{\pm 0.00}$	$43.53_{\pm 0.00}$	**$46.26_{\pm 0.00}$**
shuttle	$98.60_{\pm 0.46}$	$98.53_{\pm 0.83}$	$98.23_{\pm 0.59}$	**$99.11_{\pm 0.53}$**	$98.92_{\pm 0.55}$	$98.34_{\pm 1.24}$
wine	$99.38_{\pm 1.24}$	$99.61_{\pm 0.79}$	$99.51_{\pm 0.97}$	$99.62_{\pm 0.77}$	$99.40_{\pm 1.20}$	**$99.62_{\pm 0.77}$**
annthyroid	$49.51_{\pm 0.00}$	$50.76_{\pm 0.00}$	$47.56_{\pm 0.00}$	**$51.84_{\pm 0.00}$**	$47.08_{\pm 0.00}$	$47.77_{\pm 0.00}$
census	$15.63_{\pm 1.38}$	$14.96_{\pm 1.04}$	$15.78_{\pm 2.57}$	$16.01_{\pm 1.21}$	**$16.05_{\pm 1.61}$**	$15.61_{\pm 2.01}$
waveform	$11.92_{\pm 0.00}$	$12.19_{\pm 0.00}$	$12.87_{\pm 0.00}$	**$13.29_{\pm 0.00}$**	$12.10_{\pm 0.00}$	$11.56_{\pm 0.00}$
cardio	**$74.78_{\pm 0.00}$**	$71.32_{\pm 0.00}$	$67.28_{\pm 0.00}$	$71.60_{\pm 0.00}$	$66.73_{\pm 0.00}$	$72.01_{\pm 0.00}$
wbc	$93.52_{\pm 9.07}$	**$95.80_{\pm 5.20}$**	$91.12_{\pm 9.26}$	$87.00_{\pm 11.82}$	$86.24_{\pm 12.59}$	$90.95_{\pm 9.97}$
ionosphere	$98.60_{\pm 0.48}$	$98.79_{\pm 0.42}$	$98.71_{\pm 0.37}$	**$98.90_{\pm 0.33}$**	$98.76_{\pm 0.33}$	$98.84_{\pm 0.32}$
cover	$84.09_{\pm 2.78}$	$81.47_{\pm 4.19}$	$83.37_{\pm 2.73}$	**$86.87_{\pm 2.47}$**	$85.65_{\pm 2.43}$	$86.56_{\pm 2.91}$
pageblocks	**$61.90_{\pm 0.00}$**	$58.90_{\pm 0.00}$	$53.05_{\pm 0.00}$	$59.84_{\pm 0.00}$	$48.22_{\pm 0.00}$	$56.96_{\pm 0.00}$
pendigits	$76.32_{\pm 0.00}$	$81.49_{\pm 0.00}$	**$89.65_{\pm 0.00}$**	$87.81_{\pm 0.00}$	$87.87_{\pm 0.00}$	$89.16_{\pm 0.00}$
satellite	$80.27_{\pm 0.00}$	**$80.94_{\pm 0.00}$**	$79.53_{\pm 0.00}$	$79.09_{\pm 0.00}$	$78.91_{\pm 0.00}$	$80.91_{\pm 0.00}$
magic.gamma	$88.32_{\pm 0.15}$	$89.09_{\pm 0.10}$	$89.15_{\pm 0.15}$	**$90.54_{\pm 0.12}$**	$87.50_{\pm 0.20}$	$89.21_{\pm 0.09}$
donors	**$94.22_{\pm 1.05}$**	$93.84_{\pm 0.85}$	$90.01_{\pm 2.09}$	$92.41_{\pm 1.20}$	$88.46_{\pm 3.31}$	$88.46_{\pm 1.80}$
lymphography	$99.76_{\pm 0.49}$	$99.80_{\pm 0.41}$	**$99.80_{\pm 0.41}$**	$99.28_{\pm 0.96}$	$99.28_{\pm 0.96}$	$99.55_{\pm 0.55}$
mammography	$37.81_{\pm 1.36}$	$45.75_{\pm 1.58}$	$31.85_{\pm 1.20}$	**$49.29_{\pm 0.87}$**	$34.93_{\pm 1.41}$	$39.61_{\pm 0.84}$