Prompting Underestimates LLM Capability for Time Series Classification

Anonymous ACL submission

Abstract

Prompt-based prediction pipelines suggest that large language models (LLMs) perform poorly on time-series classification, raising doubts about whether they encode meaningful temporal structure. We show that this conclusion reflects limitations of prompt-based generation rather than the model's representational capacity by directly comparing prompt outputs with linear probes over the same internal representations while controlling for important confounding factors. While prompting performs near chance, linear probes improve average F1 from 0.15–0.26 to 0.61–0.67, often matching or exceeding specialized state-of-the-art (SoTA) time-series models. Layer-wise analyses further show that class-discriminative time-series information emerges in early transformer layers and is amplified with multimodal inputs and the presence of pretrained weights. Together, these results demonstrate a systematic mismatch between what LLMs internally represent and what prompt-based prediction pipelines reveals, leading current prompt-based prediction pipelines to underestimate the degree to which these models function as useful time-series feature extractors.[1]

1 Introduction

Time series analysis focuses on patterns in ordered sequences of events and is central to domains such as healthcare (Ayala Solares et al., 2020; Rajkomar et al., 2018; Choi et al., 2016), finance (Fama, 1970; Sirignano and Cont, 2019), and climate science (Reichstein et al., 2019). Time series classification plays a key role in these areas by assigning each sequence to a label based on its temporal structure and dynamics.

LLMs demonstrate broad generalization across natural language tasks which has motivated efforts to extend LLMs to non-linguistic data (Fons et al., 2024; Mai et al., 2024). However, there has been substantial disagreement about the usefulness of language models for time series tasks.

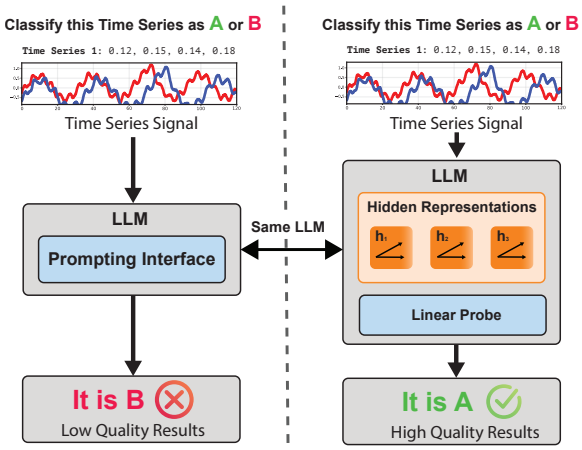

Figure 1: Main Idea.

Zeng et al. (2023) argue that transformers fail to adequately preserve the ordering information inherent in time series data. They show that many reported gains stem from the use of direct multi-step forecasting paradigms and support this claim using a simple one-layer linear model. Building on this critique, Tan et al. (2024) focus specifically on language models. They find that removing the LLM component or replacing it with a simpler alternative either does not affect performance or actually improves it. In parallel, Merrill et al. (2024) introduce a benchmark designed to evaluate time series reasoning across multiple subtasks. They find that language models perform nearly randomly. Despite these criticisms, these works converge on the view that further research is warranted: Zeng et al. (2023) encourage continued investigation in non-forecasting time series domains, Tan et al. (2024) argue that LLMs may be more appropriate for multimodal settings, and Merrill et al. (2024) emphasize the need to develop language models that are capable of robust temporal reasoning.

Other work highlights the potential of LLM-

[1] https://anonymous.4open.science/r/PromptingUnderestimates-02F1

based approaches for time series. Jin et al. (2024b) position the field as evolving from *task-specific* to *task-agnostic* to *domain-agnostic* approaches and argue that LLMs represent the next step toward *universal task solvers*. Furthermore, they take the position and that current limitations reflect the infancy of LLMs rather than a fundamental inability. Motivated by this view, a growing body of work has explored LLMs for time series forecasting, classification, and analysis (Jin et al., 2024a; Gruver et al., 2023; Morales-García et al., 2025; Zhou et al., 2023). However, these methods all rely on major confounding factors.

As the critics above emphasize, *strong end-to-end performance in these settings does not necessarily imply that LLMs encode useful temporal representations*. Prior work on LLMs for time-series analysis has not directly assessed their intrinsic representational capacity for classification. Instead, performance is often conflated with three different confounding factors (CF) that obscure the utility of the LLM. **CF1**: *results can depend heavily on prompt engineering choices*, making it difficult to distinguish genuine temporal understanding from prompt-specific behavior. **CF2**: *many approaches rely on task-specific fine-tuning*, which alters the original pretrained model and prevents evaluation of its native capabilities. **CF3**: *LLMs are often augmented with additional learned components*, making it difficult to attribute improvements to the model itself. Consequently, existing evaluations treat LLMs as components of end-to-end systems, offering limited insight into the usefulness of the model in isolation.

In this paper, we evaluate prompting and hidden representation probing side by side to assess an LLM's *observed* ability for time series classification versus its underlying *potential* (see Figure 1). To eliminate CF1, we use the same prompt for both generation and representation extraction. To eliminate CF2, we don't fine tune and operate exclusively on OOTB models. To eliminate CF3, we do not introduce auxiliary components such as specialized time-series embedders.

To summarize, this work makes four primary contributions. **(1)** To the best of our knowledge, we are the first to explicitly disentangle prompt-level performance from representational capacity in time series classification. **(2)** We show that simple classifiers over hidden representations consistently outperform prompt-based methods and even some SoTA pipelines. **(3)** We examine multiple input modalities to determine which are most effective. **(4)** Layer-wise analysis shows that temporal structure emerges early, while pretrained weights drive gains in deeper layers.

Taken together, these contributions suggest that LLMs already encode useful time-series structure, but prompt-based pipelines fail to access it.

2 Related Work

Probing and Hidden Representations in LLMs. Early work on Layer-wise Relevance Propagation introduced methods for attributing model predictions to intermediate representations (Bach et al., 2015) which later extended to more complex architectures (Binder et al., 2016; Arras et al., 2017). Building on this work, probing methods emerged to explicitly measure what information these representations encode by testing their linear separability with lightweight classifiers (Alain and Bengio, 2018). In recent years, probing methods have increasingly been used to extract features for a wide range of downstream applications. Zhu et al. (2025) and Wang et al. (2025) show that hidden states can be leveraged to estimate the difficulty an LLM is anticipated to have on a query prior to generation. More broadly, prior work has demonstrated that hidden representations encode rich and actionable signals that can be exploited across diverse settings, including alignment (Kong et al., 2024; Li et al., 2023; Zhang et al., 2025), safety (Wang et al., 2024; Lu et al., 2025), interpretability (Ghandeharioun et al., 2024; Jacobi and Niv, 2025), and robustness (Lad et al., 2024; Yan et al., 2024).

The most closely related works are Roschmann et al. (2025) and Auer et al. (2025), which both examine OOTB representations for time-series classification. Roschmann et al. (2025) show that pretrained vision transformers can produce representations competitive with time-series-specific models, while we extend this insight to text and multimodal LLMs. Auer et al. (2025) instead repurpose forecasting models for representation learning. However, *neither work directly compares probing-based classification with prompt-based generation. Nor do they focus on LLMs and vLLMs.*

Time-Series Research. Time series research spans a wide range of objectives, with forecasting and imputation forming the historical foundation of the field. Classical statistical methods, such as ARIMA and Kalman filters, model linear temporal dependencies (Box and Jenkins, 1990; Hamilton, 1994;

Hyndman and Athanasopoulos, 2018), while neural architectures, including LSTMs, convolutional networks, and transformers, enable richer nonlinear and long-range temporal modeling (Hochreiter and Schmidhuber, 1997; Bai et al., 2018; Zhou et al., 2021; Wu et al., 2021; Lim et al., 2021).

Building on this progression, recent work explores how large language models can be adapted for time series analysis. Existing approaches largely follow a small number of recurring patterns (Zhang et al., 2024): direct prompting that treats time series as text, often with modified numerical tokenization (Gruver et al., 2023; Xue and Salim, 2024; Liu et al., 2023; Xie et al., 2023); discretization or quantization schemes that map signals to token sequences (Chung et al., 2023; Duan et al., 2023; Yang et al., 2024; Rubenstein et al., 2023; Yu et al., 2023); and architectures that introduce dedicated time-series encoders aligned with language model representations (King et al., 2023; Sun et al., 2024; Zhou et al., 2026; Qiu et al., 2023; Zhou et al., 2023; Chang et al., 2025; Jin et al., 2024a). Related work also explores visual time-series representations processed by vision–language models or uses LLMs as controllers for external analysis tools (Girdhar et al., 2023; Su et al., 2023; Moon et al., 2023; Zhang et al., 2023).

In contrast, we do not propose a new end-to-end system or optimize for task performance. Instead, we use probing as a diagnostic tool to isolate the representational capacity of OOTB LLMs and vLLMs for time-series classification.

3 Preliminaries and Problem Setup

This section introduces the problem definition, the modalities used to encode time series data, and the prediction paradigms evaluated in our study.

3.1 Problem Definition

Each time-series dataset is represented as a tensor $X \in \mathbb{R}^{N \times V \times T}$, where N denotes the number of sequences, V the number of variables or channels, and T the temporal length of each sequence. Time series classification is defined as follows: given a dataset $\{(X_i, y_i)\}_{i=1}^{N}$, the objective is to predict class labels $y_j \in \{1, \ldots, C\}$ for each observation by identifying temporal patterns that distinguish between classes.

3.2 Time Series Modalities

We consider how LLMs and vLLMs perform with three different modalities of time series data. These modalities are used consistently across both prompting and probing methods and are described below:

Text Modality: Digit-Based Tokenization (d). Directly serializing numerical values can lead to unstable tokenization due to byte-pair encoding (BPE), which may fragment or merge digits unpredictably. To mitigate this issue, we adopt a digit-based preprocessing strategy from Gruver et al. (2023). Each numerical value is converted into a digit-space format with fixed precision, where commas separate time steps and decimal points are removed. For example, the time series `[1.0, 20, 0.33]` is transformed into `"1 0 0 , 2 0 0 0 , 0 3 3"`. This design choice ensures that any observed performance differences are not driven by tokenization artifacts, allowing a more faithful assessment of representational capacity[2]

Visual Modality (v). Each time series is also rendered as an image, using either line plots or spectrograms, depending on the dataset. Plots include appropriate titles, axis labels, and legends for multivariate data. These visualizations provide an alternative modality that leverages the strengths of vLLMs for pattern recognition. Additional details can be found in Appendix D.1.

Multimodal Modality (dv). Finally, we evaluate a multimodal representation that combines digit-based text images as a single multimodal input.

3.3 Prediction Paradigms

We study two complementary approaches to time series classification: direct prompt-based classification (PBC) and representation probing (RP). Figure 2 provides an overview of the representations and prediction pipelines evaluated in this work.

Prompt-Based Classification (PBC). For a given test time series X_j, we construct a prompt $p_j = \tau + q + m$, where τ denotes the task description, q is a multiple-choice question with answer options corresponding to all class labels $\{1, \ldots, C\}$, and $m \in \{d, v, dv\}$ specifies the input modality. A model f (either an LLM or vLLM) then produces a predicted label $y_j \in \{1, \ldots, C\}$ via text generation. Formally, $y_j = f(p_j), \quad \text{where } p_j = \phi(X_j, \tau, q, m)$.

As for our prompting strategy, we use zero shot

[2]We experimented with raw numeric serialization (e.g., "1.22, 4.01, ..."), but found that digit-based tokenization performed slightly better.

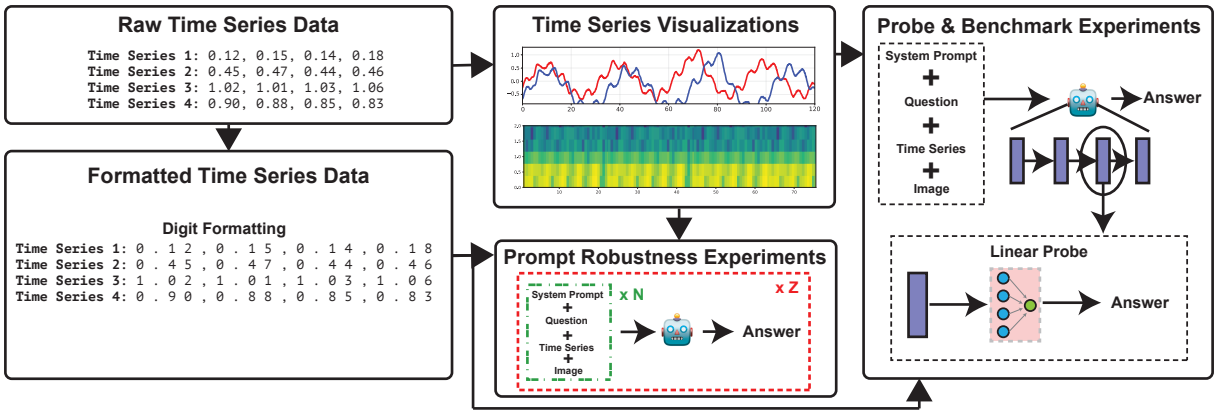

Figure 2: Overview of prompting vs. probing paradigms. Raw time series are transformed into digit-based text representations and visualizations which are incorporated into prompts for direct prediction or probing.

prompting for two reasons. First, *it is the best performing*. We test in context examples as well as chain-of-thought prompting and found that zero shot performed best (Please see Section 4: Ablations for discussion and Appendix A.2 for full result tables). Second, it allows us to keep the prompts as simple as possible which further alleviates CF1 (prompt engineering choices). Please see the Appendix D for a listing of all prompts.

Representation Probing (RP). Using the same prompt p_j, we evaluate whether class-relevant information is encoded in the OOTB model's internal representations by extracting hidden states rather than using generated outputs. For each layer $\ell \in \{1, \ldots, L\}$, we obtain a final-token embedding $h_j^{(\ell)} \in \mathbb{R}^D$. We train linear probes on these embeddings using up to 1,000 training samples, a maximum of 1,000 iterations, and 5-fold cross-validation to select the regularization parameter C. Formally, embeddings are given by $h_j^{(\ell)} = \psi_\ell(f, p_j)$, and predictions are computed as $\hat{y}_j^{(\ell)} = g_\ell(h_j^{(\ell)})$, where g_ℓ is trained on $\{(h_i^{(\ell)}, y_i)\}_{i=1}^{N_{\text{train}}}$ and evaluated on $\{h_j^{(\ell)}\}_{j=1}^{N_{\text{test}}}$.

To ensure a fair comparison with prompting, all reported classification results use only the final-layer representation $h_j^{(L)}$, which corresponds to the representation directly used during generation. This avoids aggregating across layers, which can introduce information unavailable at inference time (e.g., mean pooling Auer et al., 2025) and ensures a one-to-one correspondence between probing and prompting. Embeddings from all layers are retained only for diagnostic analysis (Section 4, Layer Analysis).

3.4 Accounting for Supervised Learning

Following Belinkov (2022), we treat probing as a diagnostic tool for assessing representational content rather than model behavior. A central concern is that probe performance may reflect the added supervision budget rather than information encoded in the underlying representations. To mitigate this, we adopt several standard controls (Belinkov, 2022; Conneau et al., 2018).

(1) We restrict all probes to simple linear classifiers trained on frozen representations. **(2)** We evaluate performance across layers and datasets, focusing on consistent trends rather than isolated improvements. **(3)** Except for layer-wise analysis, all reported results use final-layer representations, ensuring alignment with the representations used during prompting. **(4)** We contextualize probe performance using established baselines (Conneau et al., 2018), including: (i) heuristic baselines based on labels alone ; (ii) a digit-based classifier (DBC), which operates directly on raw numerical inputs to estimate signal prior to model encoding ; and (iii) probes on randomly initialized (RI) models to establish a lower bound on representation quality. As an upper-bound reference, we compare against (iv) SoTA time-series models . **(5)** We additionally include non-parametric classifiers, specifically a kNN with cosine similarity that operate on embeddings *without supervised training*. Appendix B.1 includes details of all of these baselines, SoTA methods, and the kNN model.

4 Experiments

We evaluate PBC and RP on three different vLLMs (specific models listed in Appendix B.2) six time-series classification datasets and compare them against a range of heuristic, random, and SoTA

Method	CTU	EMG	HAD	HAR	RWC	TEE	Avg
Attend	.848	.985	.732	.933	.726	.524	.791
TS2Vec	.631	.933	.667	.865	.672	.794	.761
Moment	.660	1.00	.671	.878	.778	.564	.758
OneFitsAll	.587	.296	.681	.885	.790	.475	.615
InstructTime	.627	.167	.278	.527	.434	.235	.378
R-Maj	.333	.167	.015	.048	.434	.036	.195
R-Prior	.439	.265	.088	.166	.504	.072	.282
R-Uniform	.442	.333	.084	.164	.463	.141	.296
DBC	.487	.463	.134	.488	.497	.399	.411
RI-Llama RP	.573	.933	.369	.592	.597	.436	.583
Llama RP	.640	1.00	.406	.723	.672	.223	**.611**
Llama PBC	.356	.175	.033	.111	.190	.036	.150
Mistral RP	.684	.933	.438	.730	.645	.462	**.649**
Mistral PBC	.548	.393	.021	.127	.190	.288	.261
Qwen RP	.676	1.00	.374	.737	.656	.592	**.672**
Qwen PBC	.324	.413	.016	.174	.222	.304	.242

Table 1: F1 scores across all datasets. Prompt (PBC) and Probe (RP) results pertain only to the dv modality

Method	EMG	HAD	HAR	RWC	TEE	Avg
InstructTime	.167	.278	.527	.434	.235	.328
Mistral kNN	.682	.326	.552	.558	.346	**.493**
Mistral PBC	.393	.035	.162	.383	.288	.252
Llama kNN	.776	.345	.561	.597	.295	**.515**
Llama PBC	.156	.032	.091	.161	.036	.095

Table 2: Unsupervised kNN compared to prompting approaches. Results are averaged across d, v, and dv modalities.

baselines. See Appendix B.3 for dataset descriptions and Table 14 for statistics of the datasets.

Main Findings. Table 1 summarizes performance across prompting and probing methods relative to heuristic and SoTA baselines. We highlight our main findings below.

Probing substantially outperforms prompting. Across all models and datasets, probing consistently yields significantly higher performance than direct prompting. For example, Llama improves from .150 with prompting to .611 with probing, Mistral from .261 to .649, and Qwen from .242 to .672. This gap persists across all datasets and is especially pronounced on EMG, where probing achieves near-perfect classification (up to 1.00 F1), while prompting remains near chance. These results indicate that LLMs and vLLMs encode strong time-series representations, but prompting fails to reliably access them.

Probing results are competitive with, and in some cases exceed, SoTA baselines. vLLM probing results are competitive with, and often exceed, established time-series baselines. Qwen (.672) and Mistral (.649) outperform OneFitsAll (.615), while Llama (.611) remains competitive. In contrast, even the strongest prompt-based method in our study, InstructTime (.378), performs worse than the simple Digit-Based Classifier (.411) and falls substantially behind all probe-based approaches. On individual datasets, probing achieves state-of-the-art performance in several cases. On EMG, Llama and Qwen both achieve perfect F1 scores of 1.00, exceeding Attend (.985) and TS2Vec (.933) while matching Moment. On CTU, Mistral (.684) and Qwen (.676) outperform all baselines except Attend. These results highlight the effectiveness of vLLMs as time-series feature extractors; for qualitative evidence Figure 6 in Appendix A.8 for a t-SNE visualizations. Finally, we further observe that these representations capture generalizable features beyond a single dataset; Appendix A.7 reports a cross-dataset transferability analysis.

The performance gap is not solely explained by supervision. We observe a clear hierarchy of methods: heuristic baselines (R-Maj, R-Prior, R-Uniform) perform worst, followed by the digit-based classifier (DBC), then probes on randomly initialized models, and finally probes on pretrained models. While random probes achieve strong performance (.583), pretrained probes consistently outperform them (Llama .611, Mistral .649, Qwen .672), indicating that pretrained weights contribute meaningful additional structure.

Table 3 shows that the performance gap between the random probes and pretrained probes is substantially larger when considering other modalities. As an extreme example, removing pretrained weights in the vision modality is especially damaging for LLaMA, with average F1 dropping from 0.678 to 0.172.[3]

To further isolate the role of supervision, we evaluate kNN classifiers, which do not involve any training. As shown in Table 2, kNN performance remains strong (Llama .515, Mistral .493), substantially exceeding prompting performance (.095 and .252, respectively), and even outperforming the prompt-based SoTA pipeline InstructTime (.328). Together, these results demonstrate that the observed gains are not solely due to supervised probes, but rather reflect the presence of rich, linearly separable features in the embeddings that simple prompting fails to access.

[3]Tables 1 & 3 only include Llama's Random initialization, however full apples-to-apples comparisons can be found in Table A.1 in Appendix 5.

Method	CTU	EMG	HAD	HAR	RWC	TEE	Avg
Lla RP d	.616	.866	.313	.621	.626	.335	.547
Lla RP v	.718	.933	.468	.799	.690	.512	**.678**
Lla RP dv	.654	1.00	.376	.708	.672	.223	.597
Lla PBC d	.333	.171	.032	.054	.104	.036	.132
Lla PBC v	.333	.159	.024	.077	.190	.036	.132
Lla PBC dv	.344	.171	.033	.111	.190	.036	.160
Mis RP d	.682	.712	.386	.735	.628	.265	.579
Mis RP v	.696	.796	.352	.684	.664	.439	.596
Mis RP dv	.698	.933	.393	.729	.645	.462	**.631**
Mis PBC d	.402	.280	.030	.153	.497	.260	.272
Mis PBC v	.445	.663	.055	.225	.460	.302	.354
Mis PBC dv	.511	.280	.021	.173	.190	.200	.254
Qwen RP d	.672	.933	.361	.764	.647	.419	.633
Qwen RP v	.688	.866	.304	.655	.640	.575	.621
Qwen RP dv	.676	1.00	.374	.737	.656	.592	**.672**
Qwen PBC d	.393	.372	.039	.146	.432	.159	.259
Qwen PBC v	.341	.618	.028	.220	.296	.246	.326
Qwen PBC dv	.324	.459	.016	.174	.222	.194	.255
RI-Lla RP d	.587	.861	.366	.583	.605	.433	.572
RI-Lla RP v	.333	.167	.015	.048	.434	.036	.172
RI-Lla RP dv	.573	.933	.369	.592	.597	.436	.583

Table 3: F1 for each dataset split by model, method, and modality. To save space, we have abbreviated Llama to 'Lla' and Mistral to 'Mis'; 'RI' stands for 'Randomly Initialized'

Probing and prompting benefit from visual information. As shown in Table 3, both probing and prompting improve with the inclusion of visual input, though probing remains substantially stronger. For probing, Llama performs best with v (avg .678), while Mistral and Qwen perform best with dv (.631 and .672, respectively). Prompting also benefits from visual input, particularly for Mistral and Qwen: Mistral improves from .272 (d) to .354 (v), and Qwen from .259 (d) to .326 (v).

The gains from v is most pronounced on EMG, where it substantially improves prompting performance. For example, Mistral achieves .663 with v compared to .280 with other modalities, and Qwen achieves .618 with v compared to .372 (d) and .459 (dv). In this setting, prompting comes closest to closing the gap with probing under the same modality (Mistral v: .354 vs .796, Qwen v: .326 vs .866), though a significant gap remains.

While v tends to enhance performance, *text-only input (d) is more robust than expected.* d performs surprisingly well across models and methods. For Qwen, probing with d (.633) is close to its best-performing modality (.672). For Mistral, probing with d (.579) remains competitive with v (.596). Even in prompting, d frequently outperforms the combined dv modality. This suggests that the textual representation captures meaningful time-series structure.

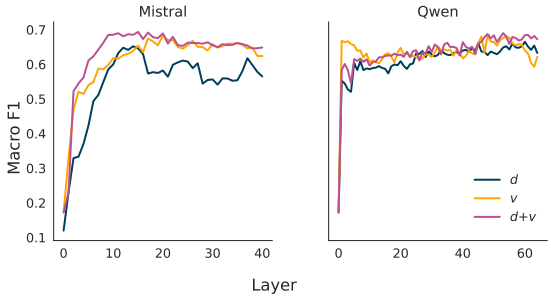

Figure 3: Layer-wise probe macro F1 for Mistral and Qwen across modalities.

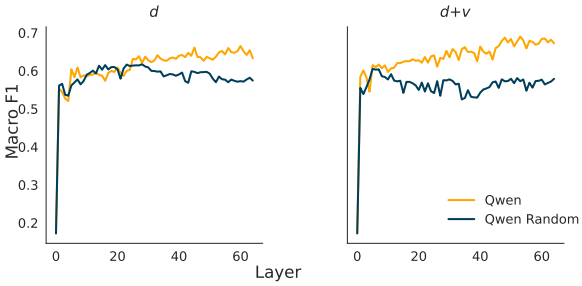

Figure 4: Layer-wise macro F1 of probes for Qwen versus the same model with randomly initialized weights.

Layer Analysis. In both plots of Figure 3, we see a quick improvement. Qwen encodes useful information in layers 1 and 2. Mistral is more gradual but encodes around layer 12 for d and dv or layer 20 for v. When comparing Qwen to its random initialization (Figure 4), both models exhibit a rapid early improvement in F1 score. However, while the random initialization stagnates at an F1 of approximately .6 across both modalities, the pretrained Qwen continues to gain performance. This suggests that while *temporal information emerges early as a structural property of the architecture, the pretrained weights are responsible for the sustained refinement and superior performance seen in higher layers.*

Ablations. Prompting is commonly augmented with in-context examples and chain-of-thought reasoning. We therefore include an ablation to assess the impact of these components. Table 7 in Appendix A.2 reports F1 scores averaged across modalities for different prompting strategies.

CoT does not have much effect on representation quality. For Llama and Qwen, average F1 decreases when using CoT, from .609 to .598 and from .667 to .654, respectively. Mistral shows a small improvement, increasing from .638 to .648. Overall, these results indicate that CoT prompting does not reliably enhance the quality of the extracted representations.

Similarly, CoT does not consistently improve

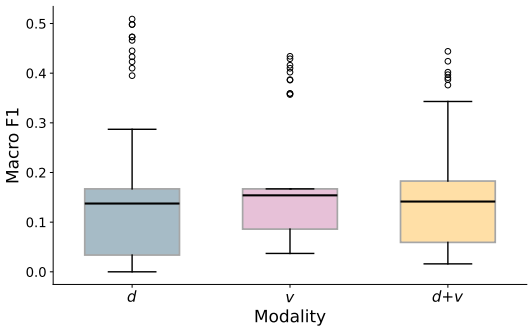

Figure 5: Prompt-based macro F1 distributions by input modalities , showing a bit more stable performance for visual-only prompting.

prompting. Llama benefits from CoT, improving from .174 with direct prompting to .220 with CoT. However, Mistral performs worse with CoT (.296) than with direct zero-shot prompting (.340). Qwen shows no meaningful difference, with CoT (.289) nearly matching direct prompting (.290). These results suggest that CoT benefits are model-dependent and not universal.

In-context examples do not improve prompting performance. Across models, in-context examples either degrade performance or provide negligible gains. For Llama, average F1 decreases from .174 in the zero-shot setting to .163 with two-shot prompting. For Qwen, performance drops from .290 to .280. Mistral shows a small decrease from .340 to .321.

Taken together, these results indicate that on our collection of datasets and models, *zero-shot direct prompting serves as the most robust and effective setting for PBC.*[4]

Results extend to text-only LLMs. To evaluate whether our findings depend on multimodal architectures, we repeat our experiments using text-only LLMs. Specifically, we evaluate Gemma and GPT-oss using the d modality. Table 9 in Appendix A.4 reports the corresponding probe-based and prompt-based results. In both LLMs, probing substantially outperforms prompting. Gemma achieves an average F1 of .596 with probing, compared to .206 with prompting. GPT-oss shows a similar pattern, achieving .466 with probing and .202 with prompting. This gap is consistent across all datasets. *These results indicate that the probing-prompting gap is not specific to visual inputs or vLLMs.*

Prompt Stability. We hypothesize that performance in LLM-based time-series classification is highly sensitive to prompt-engineering choices (CF1). We evaluate this hypothesis in Table 4, and Figure 5, which reports the results from two experiments.

[4]In Table 8 in Appendix A.3, we also evaluate how much self-consistency benefits prompting. Results are negligible.

Dataset	Modality	Mean F1	Δ F1	P@1	ΔP@K
CTU	d	.463	.099	.506	.098
	v	.394	.077	.457	.052
	dv	.380	.109	.471	.000
EMG	d	.202	.228	.343	.057
	v	.167	.000	.333	.000
	dv	.197	.236	.333	.000
HAD	d	.031	.035	.043	.062
	v	.049	.019	.163	.077
	dv	.020	.009	.059	.099
HAR	d	.138	.029	.175	.103
	v	.143	.077	.310	.040
	dv	.130	.057	.274	.034
TEE	d	.027	.081	.140	.057
	v	.110	.129	.357	.015
	dv	.102	.136	.400	.035
Avg	d	.172	**.094**	.242	**.075**
	v	.173	**.060**	.324	**.034**
	dv	.166	**.109**	.308	**.037**
Total Avg	–	.170	**.088**	.291	**.048**

Table 4: Prompt stability. Orange columns reflect sensitivity to prompt wording; blue columns reflect sampling variability. Larger Δ and ΔP@K indicate less stable prompting.

First, the sensitivity to prompt choice can be seen in Figure 5, which shows performance ranging from near 0 macro f1 to above .5. Next, (orange-shaded columns in Table 4), we assess *sensitivity to prompt wording* by using `gpt-4.1-mini` to generate ten meaning-preserving variations of the original prompt. A larger Δ (max−min macro F1) indicates more variability in performance across prompts, and thus higher sensitivity to phrasing. Across datasets and modalities, we observe an average Δ of .088 between the best- and worst-performing prompt variants, indicating *substantial sensitivity* even under minor prompt modifications. Examples of the prompt-variants can be found in Figures 14, 15, and 16 and the prompts used to generate them in Figures 12 and 13.

Second (blue-shaded columns), we evaluate *output consistency* by prompting the model twenty times with the same prompt. The resulting P@K scores (more information found in Appendix B.4) measure the probability that at least one of K sampled completions is correct. The corresponding ΔP@K (P@20−P@1) quantifies the benefit of repeated sampling. On average, allowing twenty attempts increases accuracy by .048.

Notably, *visual-only prompting (v) is the most stable modality under both criteria*. It exhibits the lowest average prompt-variant sensitivity ($\Delta = .060$) and the smallest sampling gain (ΔP@K $= .034$), suggesting more consistent behavior both across prompt wording and repeated generations.

Together, these results support our claim that prompt-based prediction pipelines in time series classification is highly sensitive to prompt-engineering choices (CF1), and that reported performance may reflect prompt design and sampling strategy as much as underlying model capability.

5 Implications

Our findings have several important implications for the evaluation and use of LLMs in time-series classification. Importantly, we do not claim that these models perform robust temporal reasoning; rather, our results show that their hidden representations encode linearly separable temporal features useful for classification.

Rethinking evaluation and prompting limitations. First, *PBC is not a reliable measure of LLM capability for time-series classification*. Across all datasets, prompt-based performance substantially underestimates the information encoded in model representations. Moreover, *prompting is unstable*: performance varies significantly with prompt wording and sampling, indicating sensitivity to factors unrelated to underlying model capability. Together, these findings suggest that prior work reporting poor LLM performance may reflect limitations of the PBC rather than the underlying model, and that prompt-based pipelines are inherently unreliable for evaluating model capability.

Pretrained weights enhance representations. Tan et al. (2024) report that re-initializing pretrained LLM weights does not significantly degrade performance. In contrast, *we find that removing pretrained weights consistently degrades performance* in the classification setting. This gap shows that, independent of their architectures, OOTB models encode useful time-series representations.

General-purpose models encode time-series structure across modalities. Prior work has shown that useful time-series representations can be obtained from models adapted to this domain, such as vision transformers with modality transformations (Roschmann et al., 2025) and pretrained forecasting models repurposed for classification (Auer et al., 2025). Our results extend this line of work by demonstrating that such representations are not limited to these settings, but also *emerge in general-purpose models*, including LLMs and vLLMs, without task-specific adaptation. While modality transformations and specialized architectures can improve performance, *they are not necessary* for models to encode discriminative temporal structure.

Representation formation and access. Our layer-wise analysis suggests that temporal structure emerges early as a property of the architecture, while pretrained weights refine and amplify this structure in deeper layers. Despite this, *final-layer representations already contain substantial class-relevant information*. Prior work improves performance through layer aggregation or direct selection (Auer et al., 2025; Roschmann et al., 2025), but our results show that these optimizations are not necessary to reveal useful structure. Crucially, they do not explain why prompt-based methods underperform, which instead reflects limitations in how these representations are accessed rather than how they are formed.

6 Conclusion

This research provides a rigorous reassessment of LLMs' ability to represent and classify functional time-series data. By contrasting direct prompting with linear probing across diverse architectures and modalities, we identify a persistent gap in representation performance. Our evidence suggests that, although current prompting interfaces fail to capture temporal dynamics, the internal hidden states of these models contain highly discriminative, class-relevant information. The results indicate that the perceived limitations of LLMs in time series tasks are primarily an artifact of the decoding interface rather than a fundamental deficiency in representational capacity. The early emergence of class discriminative signals, often within the first five transformer layers, suggests that these models function as effective general-purpose sequence encoders before higher-level reasoning stages. Furthermore, the competitive performance of simple linear probes against specialized, SoTA time series baselines underscores the quality of pretrained LLMs as robust feature extractors for non-linguistic modalities.

Limitations

This paper is primarily diagnostic. We focus on measuring and explaining the gap between prompt-level performance and representational capacity, rather than proposing a new prompting method to close this gap.

Next, while we address standard probing concerns (see Section 3.4), strong probe performance should not be interpreted as evidence of high-level temporal reasoning or semantic understanding. Rather, it is plausible that probes exploit low-level temporal cues such as amplitude ranges, monotonic trends, periodicity, or short-range autocorrelation. These cues are precisely the signals required for accurate classification in many standard time-series benchmarks.

Finally, our evaluation is necessarily scoped. We study six time-series datasets, five LLM/vLLMs, and a broad set of SoTA and heuristic baselines. The consistency of the observed trends across these diverse settings suggests that the underestimation of representational capacity by PBC is not driven by any single dataset or model choice.

Building on our diagnostic findings, developing more robust prompting strategies is an important direction for future work. We take initial steps toward addressing the prompt–probe gap in Appendix A.9, where we test whether fine-tuning reduces it. However, our results are not conclusive, and further work is needed to better understand and close this performance gap.

Further research should also explore LLMs for more holistic time-series reasoning. While our current focus is primarily on classification, generalizing these results to additional datasets, diverse time-series tasks, and new models remains a critical next step. To that end, we provide an initial forecasting analysis in Appendix A.5; however, these results are preliminary and limited in scope.

References

Alireza Abedin, Mahsa Ehsanpour, Qinfeng Shi, Hamid Rezatofighi, and Damith C. Ranasinghe. 2021. Attend and discriminate: Beyond the state-of-the-art for human activity recognition using wearable sensors. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 5(1).

Guillaume Alain and Yoshua Bengio. 2018. Understanding intermediate layers using linear classifier probes. *Preprint*, arXiv:1610.01644.

D. Anguita, Alessandro Ghio, L. Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. 2013. A public domain dataset for human activity recognition using smartphones. In *The European Symposium on Artificial Neural Networks*.

Leila Arras, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2017. Explaining recurrent neural network predictions in sentiment analysis. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 159–168, Copenhagen, Denmark. Association for Computational Linguistics.

Andreas Auer, Daniel Klotz, Sebastinan Böck, and Sepp Hochreiter. 2025. Pre-trained forecasting models: Strong zero-shot feature extractors for time series classification. *Preprint*, arXiv:2510.26777.

Jose Roberto Ayala Solares, Francesca Elisa Diletta Raimondi, Yajie Zhu, Fatemeh Rahimian, Dexter Canoy, Jenny Tran, Ana Catarina Pinho Gomes, Amir H. Payberah, Mariagrazia Zottoli, Milad Nazarzadeh, Nathalie Conrad, Kazem Rahimi, and Gholamreza Salimi-Khorshidi. 2020. Deep learning for electronic health records: A comparative review of multiple deep neural architectures. *Journal of Biomedical Informatics*, 101:103337.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE*, 10(7):1–46.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *Preprint*, arXiv:1803.01271.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, and 45 others. 2025. Qwen3-vl technical report. *Preprint*, arXiv:2511.21631.

Yonatan Belinkov. 2022. Probing Classifiers: Promises, Shortcomings, and Advances. *Computational Linguistics*, 48(1):207–219. _eprint: https://direct.mit.edu/coli/article-pdf/48/1/207/2006605/coli_a_00422.pdf.

Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, Klaus-Robert Müller, and Wojciech Samek. 2016. Layer-wise relevance propagation for neural networks with local renormalization layers. In *Artificial Neural Networks and Machine Learning – ICANN 2016*, pages 63–71, Cham. Springer International Publishing.

George Edward Pelham Box and Gwilym Jenkins. 1990. *Time Series Analysis, Forecasting and Control*. Holden-Day, Inc., USA.

Ching Chang, Wei-Yao Wang, Wen-Chih Peng, and Tien-Fu Chen. 2025. Llm4ts: Aligning pre-trained llms as data-efficient time-series forecasters. *ACM Trans. Intell. Syst. Technol.*, 16(3).

Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. 2015. The ucr time series classification archive. www.cs.ucr.edu/~eamonn/time_series_data/.

Mingyue Cheng, Yiheng Chen, Qi Liu, Zhiding Liu, Yucong Luo, and Enhong Chen. 2025. Instructime: Advancing time series classification with multimodal language modeling. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, WSDM '25, page 792–800, New York, NY, USA. Association for Computing Machinery.

Edward Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter F. Stewart, and Jimeng Sun. 2016. Doctor ai: Predicting clinical events via recurrent neural networks. In *Proceedings of the 1st Machine Learning for Healthcare Conference*, volume 56 of *Proceedings of Machine Learning Research*, pages 301–318, Northeastern University, Boston, MA, USA. PMLR.

Hyunseung Chung, Jiho Kim, Joon-Myoung Kwon, Ki-Hyun Jeon, Min Sung Lee, and Edward Choi. 2023. Text-to-ecg: 12-lead electrocardiogram synthesis conditioned on clinical text reports. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

TM Cox, TJ Ragen, AJ Read, E Vos, RW Baird, K Balcomb, J Barlow, J Caldwell, T Cranford, L Crum, and 1 others. 2006. Understanding the impacts of anthropogenic sound on beaked whales. *Journal of Cetacean Research and Management*, 7:177–187.

Yiqun Duan, Jinzhao Zhou, Zhen Wang, Yu-Kai Wang, and Chin-teng Lin. 2023. Dewave: Discrete encoding of eeg waves for eeg to text translation. In *Advances in Neural Information Processing Systems*, volume 36, pages 9907–9918. Curran Associates, Inc.

Damian R. Eads, Daniel Hill, Sean Davis, Simon J. Perkins, Junshui Ma, Reid B. Porter, and James P. Theiler. 2002. Genetic algorithms and support vector machines for time series classification. In *Applications and Science of Neural Networks, Fuzzy Systems, and Evolutionary Computation V*, volume 4787, pages 74 – 85. International Society for Optics and Photonics, SPIE.

Eugene F. Fama. 1970. Efficient capital markets: A review of theory and empirical work. *The Journal of Finance*, 25(2):383–417.

Elizabeth Fons, Rachneet Kaur, Soham Palande, Zhen Zeng, Tucker Balch, Manuela Veloso, and Svitlana Vyetrenko. 2024. Evaluating large language models on time series feature understanding: A comprehensive taxonomy and benchmark. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 21598–21634, Miami, Florida, USA. Association for Computational Linguistics.

Asma Ghandeharioun, Avi Caciularu, Adam Pearce, Lucas Dixon, and Mor Geva. 2024. Patchscopes: A unifying framework for inspecting hidden representations of language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 15466–15490. PMLR.

Rohit Girdhar, Alaaeldin El-Nouby, Zhuang Liu, Mannat Singh, Kalyan Vasudev Alwala, Armand Joulin, and Ishan Misra. 2023. Imagebind one embedding space to bind them all. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15180–15190.

Arye L. Goldberger, Luis A. N. Amaral, Leon Glass, Jeffrey M. Hausdorff, Plamen Ch. Ivanov, Roger G. Mark, Joseph E. Mietus, George B. Moody, Chung-Kang Peng, and H. Eugene Stanley. 2000. Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220. RRID:SCR_007345.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. 2024. Moment: a family of open time-series foundation models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. 2023. Large language models are zero-shot time series forecasters. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

James D. Hamilton. 1994. *Time Series Analysis*. Princeton University Press.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

R. J. Hyndman and G. Athanasopoulos. 2018. *Forecasting: Principles and Practice*, 2nd edition. OTexts, Melbourne, Australia. Accessed on March 11, 2026.

Jonathan Jacobi and Gal Niv. 2025. Superscopes: Amplifying internal feature representations for language model interpretation. *Preprint*, arXiv:2503.02078.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. 2024a. Time-LLM: Time series forecasting by reprogramming large language models. In *The Twelfth International Conference on Learning Representations*.

Ming Jin, Yifan Zhang, Wei Chen, Kexin Zhang, Yuxuan Liang, Bin Yang, Jindong Wang, Shirui Pan, and Qingsong Wen. 2024b. Position: What can large language models tell us about time series analysis. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 22260–22276. PMLR.

Ryan King, Tianbao Yang, and Bobak J. Mortazavi. 2023. Multimodal pretraining of medical time series and notes. In *Proceedings of the 3rd Machine Learning for Health Symposium*, volume 225 of *Proceedings of Machine Learning Research*, pages 244–255. PMLR.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization. *Preprint*, arXiv:1412.6980.

Lingkai Kong, Haorui Wang, Wenhao Mu, Yuanqi Du, Yuchen Zhuang, Yifei Zhou, Yue Song, Rongzhi Zhang, Kai Wang, and Chao Zhang. 2024. Aligning large language models with representation editing: A control perspective. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Vedang Lad, Wes Gurnee, and Max Tegmark. 2024. The remarkable robustness of LLMs: Stages of inference? In *ICML 2024 Workshop on Mechanistic Interpretability*.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2023. Inference-time intervention: Eliciting truthful answers from a language model. In *Advances in Neural Information Processing Systems*, volume 36, pages 41451–41530. Curran Associates, Inc.

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

Haoxin Liu, Chenghao Liu, and B. Aditya Prakash. 2025. A picture is worth a thousand numbers: Enabling LLMs reason about time series via visualization. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7486–7518, Albuquerque, New Mexico. Association for Computational Linguistics.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2020. On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations*.

Xin Liu, Daniel McDuff, Geza Kovacs, Isaac Galatzer-Levy, Jacob Sunshine, Jiening Zhan, Ming-Zher Poh, Shun Liao, Paolo Di Achille, and Shwetak Patel. 2023. Large language models are few-shot health learners. *Preprint*, arXiv:2305.15525.

Ilya Loshchilov and Frank Hutter. 2017. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. *Preprint*, arXiv:1711.05101.

Xiaoya Lu, Dongrui Liu, Yi Yu, Luxin Xu, and Jing Shao. 2025. X-boundary: Establishing exact safety boundary to shield LLMs from jailbreak attacks without compromising usability. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 5247–5272, Suzhou, China. Association for Computational Linguistics.

Gengchen Mai, Weiming Huang, Jin Sun, Suhang Song, Deepak Mishra, Ninghao Liu, Song Gao, Tianming Liu, Gao Cong, Yingjie Hu, Chris Cundy, Ziyuan Li, Rui Zhu, and Ni Lao. 2024. On the opportunities and challenges of foundation models for geoai (vision paper). *ACM Trans. Spatial Algorithms Syst.*, 10(2).

Mike A Merrill, Mingtian Tan, Vinayak Gupta, Thomas Hartvigsen, and Tim Althoff. 2024. Language models still struggle to zero-shot reason about time series. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3512–3533, Miami, Florida, USA. Association for Computational Linguistics.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2018. Mixed precision training. In *International Conference on Learning Representations*.

MistralAI. Model Card for Mistral-Small-3.1-24B-Instruct-2503. Accessed 12-01-2025.

Seungwhan Moon, Andrea Madotto, Zhaojiang Lin, Aparajita Saraf, Amy Bearman, and Babak Damavandi. 2023. IMU2CLIP: Language-grounded motion sensor translation with multimodal contrastive learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 13246–13253, Singapore. Association for Computational Linguistics.

Juan Morales-García, Antonio Llanes, Francisco Arcas-Túnez, and Fernando Terroso-Sáenz. 2025. Developing time series forecasting models with generative large language models. *ACM Trans. Intell. Syst. Technol.*, 16(6).

OpenAI, :, Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett, Eugene Brevdo, Greg Brockman, Sebastien Bubeck, and 108 others. 2025. gpt-oss-120b gpt-oss-20b model card. *Preprint*, arXiv:2508.10925.

Jielin Qiu, William Han, Jiacheng Zhu, Mengdi Xu, Michael Rosenberg, Emerson Liu, Douglas Weber, and Ding Zhao. 2023. Transfer knowledge from natural language to electrocardiography: Can we detect cardiovascular disease through language models? In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 442–453, Dubrovnik, Croatia. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, page 9.

Alvin Rajkomar, Eyal Oren, Kai Chen, Andrew M. Dai, Nissan Hajaj, Michaela Hardt, Peter J. Liu, Xiaobing Liu, Jake Marcus, Mimi Sun, Patrik Sundberg, Hector Yee, Kun Zhang, Yi Zhang, Gerardo Flores, Gavin E. Duggan, Jamie Irvine, Quoc Le, Kurt Litsch, and 16 others. 2018. Scalable and accurate deep learning with electronic health records. *npj Digital Medicine*, 1(1):18.

Markus Reichstein, Gustau Camps-Valls, Bjorn Stevens, Martin Jung, Joachim Denzler, Nuno Carvalhais, and Prabhat. 2019. Deep learning and process understanding for data-driven Earth system science. *Nature*, 566(7743):195–204.

Simon Roschmann, Quentin Bouniot, Vasilii Feofanov, Ievgen Redko, and Zeynep Akata. 2025. Time series representations for classification lie hidden in pretrained vision transformers. *Preprint*, arXiv:2506.08641.

Paul K. Rubenstein, Chulayuth Asawaroengchai, Duc Dung Nguyen, Ankur Bapna, Zalán Borsos, Félix de Chaumont Quitry, Peter Chen, Dalia El Badawy, Wei Han, Eugene Kharitonov, Hannah Muckenhirn, Dirk Padfield, James Qin, Danny Rozenberg, Tara Sainath, Johan Schalkwyk, Matt Sharifi, Michelle Tadmor Ramanovich, Marco Tagliasacchi, and 11 others. 2023. Audiopalm: A large language model that can speak and listen. *Preprint*, arXiv:2306.12925.

Justin Sirignano and Rama Cont. 2019. Universal features of price formation in financial markets: perspectives from deep learning. *Quantitative Finance*, 19(9):1449–1459.

Yixuan Su, Tian Lan, Huayang Li, Jialu Xu, Yan Wang, and Deng Cai. 2023. PandaGPT: One model to instruction-follow them all. In *Proceedings of the 1st Workshop on Taming Large Language Models: Controllability in the era of Interactive Assistants!*, pages 11–23, Prague, Czech Republic. Association for Computational Linguistics.

Chenxi Sun, Hongyan Li, Yaliang Li, and Shenda Hong. 2024. TEST: Text prototype aligned embedding to activate LLM's ability for time series. In *The Twelfth International Conference on Learning Representations*.

Mingtian Tan, Mike A Merrill, Vinayak Gupta, Tim Althoff, and Thomas Hartvigsen. 2024. Are language models actually useful for time series forecasting? In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, and 179 others. 2024. Gemma 2: Improving open language models at a practical size. *Preprint*, arXiv:2408.00118.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. 2017. Neural discrete representation learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6309–6318, Red Hook, NY, USA. Curran Associates Inc.

Haoyu Wang, Bingzhe Wu, Yatao Bian, Yongzhe Chang, Xueqian Wang, and Peilin Zhao. 2024. Probing the safety response boundary of large language models via unsafe decoding path generation. *Preprint*, arXiv:2408.10668.

Yanling Wang, Haoyang Li, Hao Zou, Jing Zhang, Xinlei He, Qi Li, and Ke Xu. 2025. FacLens: Transferable probe for foreseeing non-factuality in fact-seeking question answering of large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 18563–18582, Suzhou, China. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2021. Autoformer: decomposition transformers with auto-correlation for long-term series forecasting. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA. Curran Associates Inc.

Qianqian Xie, Weiguang Han, Yanzhao Lai, Min Peng, and Jimin Huang. 2023. The wall street neophyte: A zero-shot analysis of chatgpt over multimodal stock movement prediction challenges. *Preprint*, arXiv:2304.05351.

Hao Xue and Flora D. Salim. 2024. Promptcast: A new prompt-based learning paradigm for time series forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 36(11):6851–6864.

Tianyi Yan, Fei Wang, James Y. Huang, Wenxuan Zhou, Fan Yin, Aram Galstyan, Wenpeng Yin, and Muhao Chen. 2024. Contrastive instruction tuning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10288–10302, Bangkok, Thailand. Association for Computational Linguistics.

Dongchao Yang, Jinchuan Tian, Xu Tan, Rongjie Huang, Songxiang Liu, Haohan Guo, Xuankai Chang, Jiatong Shi, Sheng Zhao, Jiang Bian, Zhou Zhao, Xixin Wu, and Helen M. Meng. 2024. UniAudio: Towards universal audio generation with large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 56422–56447. PMLR.

Xinli Yu, Zheng Chen, and Yanbin Lu. 2023. Harnessing LLMs for temporal data - a study on explainable financial time series forecasting. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 739–753, Singapore. Association for Computational Linguistics.

Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. 2022. Ts2vec: Towards universal representation of time series. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(8):8980–8987.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. 2023. Are transformers effective for time series forecasting? *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(9):11121–11128.

Honggen Zhang, Xufeng Zhao, Igor Molybog, and June Zhang. 2025. Real: Response embedding-based alignment for llms. *Preprint*, arXiv:2409.17169.

Mi Zhang and Alexander A. Sawchuk. 2012. Usc-had: a daily activity dataset for ubiquitous activity recognition using wearable sensors. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, page 1036–1043, New York, NY, USA. Association for Computing Machinery.

Xiyuan Zhang, Ranak Roy Chowdhury, Rajesh K. Gupta, and Jingbo Shang. 2024. Large language models for time series: A survey. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 8335–8343. International Joint Conferences on Artificial Intelligence Organization. Survey Track.

Yunkai Zhang, Yawen Zhang, Ming Zheng, Kezhen Chen, Chongyang Gao, Ruian Ge, Siyuan Teng, Amine Jelloul, Jinmeng Rao, Xiaoyuan Guo, Chiang-Wei Fang, Zeyu Zheng, and Jie Yang. 2023. Insight miner: A large-scale multimodal model for insight mining from time series. In *NeurIPS 2023 AI for Science Workshop*.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12):11106–11115.

Tian Zhou, Peisong Niu, Xue Wang, Liang Sun, and Rong Jin. 2023. One fits all: power general time series analysis by pretrained lm. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Yunjiao Zhou, Jianfei Yang, Han Zou, and Lihua Xie. 2026. Tent: Connect language models with iot sensors for zero-shot activity recognition. *IEEE Transactions on Mobile Computing*, pages 1–12.

Yubo Zhu, Dongrui Liu, Zecheng Lin, Wei Tong, Sheng Zhong, and Jing Shao. 2025. The LLM already knows: Estimating LLM-perceived question difficulty via hidden representations. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 1160–1176, Suzhou, China. Association for Computational Linguistics.

A Additional Experiments and Results

This section provides supplementary experiments and detailed results to further contextualize our findings. Specifically, we present: **(1)** a one-to-one comparison of probing results between pretrained vLLMs and their randomly initialized counterparts; **(2)** an evaluation of zero-shot, Chain-of-Thought (CoT), and few-shot prompting/probing; **(3)** prompting performance using self consistency; **(4)** text-only LLM (gpt-oss & Gemma) results; **(5)** an extension of our framework to time-series forecasting; **(6)** a cost analysis **(7)** an in-domain vs out-of-domain generalizability experiment; **(8)** a visualization comparing feature separability; and **(9)** We explore the effect of fine-tuning on the probe-prompt gap.

A.1 One-To-One Random Baseline

In the main paper, we utilized a randomly initialized version of Llama as a baseline to control for supervised training effects. However, to enable a true apples-to-apples comparison, we expanded this evaluation across all model architectures. Table 5 details the performance of each pretrained model alongside its corresponding randomly initialized baseline.

Across all models, pretrained versions consistently outperform their randomly initialized counterparts. While Mistral exhibits only a marginal advantage over its random twin, primarily due to an atypical performance dip on the EMG dataset (.814 vs. .954), a more substantial gap is evident for Llama (0.443 to 0.607) and Qwen (0.592 to 0.642)

Model	CTU	EMG	HAD	HAR	RWC	TEE	Avg
Llama	.663	.933	.386	.709	.662	.357	.607
RI-Llama	.498	.653	.250	.408	.545	.302	.443
Mistral	.692	.814	.377	.716	.645	.389	.602
RI-Mistarl	.615	.954	.364	.647	.660	.323	.594
Qwen	.679	.933	.346	.719	.647	.529	.642
RI-Qwen	.645	.933	.355	.627	.649	.341	.592

Table 5: Comparison of probe F1 between all pretrained and randomly initialized models averaged over all modalities.

We further test if there is high variance depending on the random initialization. To do this we made three separate random initializations of each model. Table 6 reports the resulting mean, standard deviation, range, and 95% confidence intervals across these runs to establish a statistically robust performance floor. We observe that while the baselines are generally stable, performance can be highly sensitive to initialization on specific dataset-model pairings. For instance, Llama exhibits massive variance on the EMG dataset (StdDev: 0.312, Range: 0.667) as well as on our next smallest dataset, TEE (StdDev: 0.196, Range: 0.429). Conversely, as dataset size increases, baseline stability notably improves; both models demonstrate much tighter variance on the larger CTU dataset, with Llama and Qwen yielding standard deviations of just 0.057 (Range: 0.128) and 0.080 (Range: 0.176), respectively.

A.2 CoT and Few-Shot Results

Table 7 extends our comparison of probing and prompting by evaluating the impact of few-shot examples and Chain-of-Thought (CoT) prompting. Surprisingly, zero-shot Direct prompting outperforms both CoT and 2-shot approaches for Mistral (0.340 vs. 0.296 CoT and 0.321 2-shot) and Qwen (0.290 vs. 0.289 CoT and 0.280 2-shot). Furthermore, CoT does not consistently enhance probing representations. Although Mistral sees a marginal improvement with CoT probing (0.648 vs. 0.638 Direct), Llama and Qwen both achieve higher performance using the Direct setting. Given that the *zero-shot Direct method yields the best results in two out of three models for both prompting and probing, we adopt it as our primary prompting strategy*. Zero-shot is also the simplest, mitigating possible confounding factors like prompt sensitivity.

Dataset	Model	Mean	StdDev	Range	95%CI
CTU (n=250)	Llama	.575	.057	.128	.044
	Qwen	.651	.080	.176	.061
EMG (n=47)	Llama	.748	.312	.667	.240
	Qwen	.948	.029	.067	.023
TEE (n=73)	Llama	.399	.196	.429	.150
	Qwen	.323	.087	.238	.067

Table 6: Probing stability of random models. Mean, Standard Deviation, Range and 95% Confidence Intervals of three different initialized models.

A.3 Self-Consistency

In an additional effort to fairly represent prompting as a method, we evaluate self-consistency on three selected datasets. The results, presented in Table 8, compare zero-shot performance (ZS-F1) against self-consistency (SelfC-F1) using Llama. For each instance in the dataset, we generated seven independent responses and selected the final answer via majority voting. Across the evaluated datasets and modalities, we observe a modest overall improvement; self-consistency yields an average F1 score of 0.200, representing a 2.2 gain over the zero-shot average of 0.178. However, despite this slight enhancement, results are far lower than the best prompting method InstructTime. Furthermore, overall prompting performance under both settings remains substantially lower than the results achieved by our probing methods.

A.4 Text-Only Models

To ensure that our results generalize to text-only models (i.e., standard LLMs rather than vLLMs), we repeat the same experiments using `GPT-oss` and `gemma-2-9b-it`. Table 9 shows the results of

Method	Model	Shots	Style	CTU	EMG	HAR	RWC	TEE	Avg
Probe	Llama	0	CoT	.651	.789	.629	.566	.351	.598
		0	Direct	.650	.784	.645	.608	.355	.609
	Mistral	0	CoT	.688	.899	.653	.581	.418	.648
		0	Direct	.674	.871	.655	.585	.408	.638
	Qwen	0	CoT	.697	.920	.677	.595	.442	.654
		0	Direct	.702	.938	.704	.592	.473	.667
Prompt	Llama	0	CoT	.434	.233	.126	.231	.077	**.220**
		0	Direct	.333	.185	.083	.231	.036	.174
		2	Direct	.333	.173	.041	.231	.036	.163
	Mistral	0	CoT	.544	.324	.168	.339	.105	.296
		0	Direct	.420	.393	.199	.398	.288	**.340**
		2	Direct	.423	.445	.230	.320	.188	.321
	Qwen	0	CoT	.498	.127	.225	.401	.195	.289
		0	Direct	.359	.393	.127	.288	.283	**.290**
		2	Direct	.472	.360	.228	.234	.107	.280

Table 7: F1 results between prompting and probing. We explore whether Chain-of-Thought (CoT) prompting and few-shot examples improve prompting performance, as well as whether CoT improves the representations during probing. *Shots* represent the number of in-line examples *per class* in the dataset. For example *har* has 6 possible classes, there are 2 examples per class totaling to 12 examples overall. These results come from a smaller sample using 500 training rows and 100 test rows.

Dataset	Modality	ZS-F1	SelfC-F1
emg	d	.175	.167
	v	.118	.199
	dv	.175	.196
ctu	d	.333	.333
	v	.333	.333
	dv	.356	.355
tee	d	.036	.099
	v	.036	.036
	dv	.036	.086
	Avg:	.178	.200

Table 8: A comparison of zero-shot vs self-consistency F1 scores using Llama on a selection of the three smallest datasets.

these experiments. Please note that these evaluations only utilize the d domain, as these models are incapable of processing images. We observe that both models perform better with probing than they do with prompting. This finding, combined with our evaluation of vLLMs in the d-only domain, provides strong support for generalization.

A.5 What About Forecasting?

Our paper focuses on time-series classification as a proxy for temporal reasoning. However, time-series reasoning encompasses far more than classification; tasks such as forecasting, imputation, and anomaly detection are essential components of a holistic temporal understanding. While a comprehensive evaluation of each task is outside the current scope, we explore forecasting here to provide a foundation and motivation for future research.

With this purpose, use the ETT-small dataset. It comprises of two years of hourly electricity transformer data. To prepare the data, we set Oil Temperature (OT) as our target feature, and apply z-score normalization. In our experiments, We used a 96-hour look back window to predict the next 6 hours. Finally, our prompts (Which can be found in Appendix D.1 Figure 17) generate one value at a time, and do so in an auto-regressive fashion.

Results are shown in Table 10. They show inconsistency. The most notable finding is the success of prompting observed for Mistral (0.97) and Qwen (1.15) in the text-only (d) modality. These are the lowest MSE scores in the study, dramatically outperforming all other configurations. Conversely, other prompting results are substantially worse than probing; for instance, Llama's prompting performance in the d modality (2032.77) fails entirely compared to its probe (11.61).

Despite these variances, we observe that on average, every model's probe MSE is lower than its randomly initialized counterpart. While these results are preliminary, involving a single small-scale dataset and short prediction window, they suggest that internal representations *can* encode useful features for not just classification but also forecasting. However, the decoding interface of prompt-

Model	Method	CTU	EMG	HAD	HAR	RWC	TEE	Avg
Gemma	Probe	.680	1.00	.145	.739	.556	.458	.596
GPT-oss		.579	.792	.112	.498	.518	.295	.466
Gemma	Prompt	.286	.314	.023	.075	.346	.100	.206
GPT-oss		.282	.358	.031	.142	.272	.132	.202

Table 9: F1 scores for probe-based and prompt-based inference on text-only large language models.

Method	Model	d	dv	v	Avg
Probe	llama	11.61	9.21	9.82	10.22
	llama rand	11.10	10.23	10.77	10.70
	mistral	7.19	10.17	9.82	9.06
	mistral rand	11.74	12.56	7.66	10.66
	qwen	7.64	11.68	10.28	9.86
	qwen rand	11.79	10.41	9.34	10.51
Prompt	llama	2032.77	835.73	307.12	1058.54
	mistral	0.97	22.51	779.66	267.71
	qwen	1.15	1157.98	1114.13	757.76

Table 10: Comparison (MSE) of zero-shot prompting and linear probing for time-series forecasting.

Modality	Avg Tokens	Projected Cost
d	4,067.46	\$10.17
v	1,996.50	\$4.99
$d + v$	5,115.80	\$12.79

Table 11: Estimated prompting costs aggregated across all datasets. Avg Tokens shows the expected number of tokens per instance. Projected cost shows the predicted cost of 1,000 instances.

Modality	Prompt HAD	HAD→HAD	HAR→HAD
d	.032	.349	.389
v	.032	.561	.422
dv	.033	.406	.131

Table 12: Generalization check: F1 scores for in-distribution versus cross-dataset probing and zero-shot prompting

ing remains highly brittle and model/modality-dependent.

A.6 Cost

We evaluate the resource requirements and supervision budgets. Zero-shot prompting operates under the strictest constraints, with no access to training data. Few-shot prompting is provided with two examples per unique label. In contrast, our linear probes are trained on a subset of the training data, capped at a maximum of 1,000 rows. Table 11 details the financial cost of inference based on average token counts across modalities. Costs are projected using a standard rate of \$2.50 per million input tokens. For visual inputs, we account for GPT-4o's high-detail tiling (85 base tokens plus 170 tokens per 512×512 tile). Interestingly, because probing utilizes the same underlying hidden representations as the zero-shot interface without the need for autoregressive token generation, it is marginally cheaper in practice while providing significantly higher performance floor.

A.7 Transferability Check

To strengthen our claims regarding representation quality beyond simple in-distribution separability, we perform a cross-dataset generalization check. We focus on the HAD and HAR datasets, as both represent human activity recognition tasks with similar underlying temporal dynamics. Because HAD contains a larger label space, we first consolidated labels to create a one-to-one mapping (e.g., merging various directional walking labels into a single "walking" class) and removed activities unique to a single dataset (e.g., "elevator usage"). We then evaluated the transferability of Llama representations by training a linear probe on the source dataset (HAR) and evaluating its performance on the target (HAD). Table 12 compares these cross-dataset results against in-distribution probing (HAD → HAD) and zero-shot prompting. We observe that cross-dataset probing maintains a significant performance advantage over prompting, which is very low (≈ 0.03 F1) across all modalities. Notably, in the d modality, the HAR-trained probe achieved an F1 of 0.389, outperforming the in-distribution probe (0.349). These results overall suggest that the LLM's internal representations capture generalizable features rather than dataset specific ones.

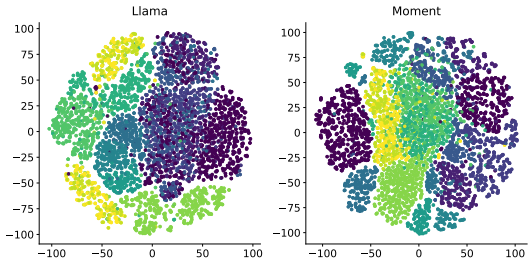

Figure 6: t-SNE of probe embeddings for Llama and Moment on the HAD dataset.

Method	Model	CTU	EMG	HAD	HAR	RWC	TEE	Avg
kNN	OOTB	.684	.603	.451	.713	.609	.350	.568
	FT	.688	1.00	.501	.747	.689	.390	.669
RP	OOTB	.704	.933	.561	.839	.690	.512	.706
	FT	.744	1.00	.565	.854	.746	.391	.716
PBC	OOTB	.333	.118	.032	.103	.190	.036	.135
	FT	.650	.222	.105	.256	.433	.162	.305

Table 13: Effect of LoRA fine-tuning on kNN, probing, and prompting performance (vision modality).

A.8 A Visualization of vLLM feature embeddings

Figure 6 presents t-SNE plots comparing Llama v embeddings with Moment embeddings on the HAD dataset. This visualization highlights differences in feature separability and illustrates the utility of Llama as a feature extractor relative to a specialized baseline.

A.9 The Effect of Fine-Tuning on the Probe–Prompt Gap

To assess whether the gap between probing and prompting can be reduced through additional fine-tuning, we conduct a controlled study using `LLaMA-3.2-11B-Vision-Instruct`. Specifically, we test whether fine-tuning enables information that is already accessible via linear probes to be more effectively utilized through prompting.

We fine-tune the model using LoRA (rank = 16) on each dataset's training split. After fine-tuning, we re-evaluate the adapted model using both PBC and RB, allowing us to directly compare performance before and after adaptation. This setup enables us to test whether fine-tuning closes the gap between probing and prompting, or instead improves both methods proportionally.

Results for the vision modality are summarized in Table 13. Fine-tuning (FT) improves overall representation separability across all evaluation methods. The kNN classifier improves from an average F1 of .568 to .669, indicating stronger clustering in the embedding space without additional supervision. PBC performance also improves substantially, increasing from .135 to .305. In contrast, probing shows only modest gains, improving from .706 to .716.

The relatively smaller gains in probing are partly driven by high variance on smaller datasets (e.g., TEE), where fine-tuning can degrade performance due to limited test samples. Despite these improvements, a substantial gap between probing and prompting remains. Notably, even under an asymmetric comparison—comparing pre-trained kNN (.568) to fine-tuned prompting (.305)—there remains a gap of .263 F1.

These results suggest that while fine-tuning improves overall separability of representations, it does not fundamentally resolve the gap between what is linearly accessible in model representations and what can be reliably extracted through prompting. In other words, fine-tuning enhances representation quality, but does not eliminate the limitations of prompt-based access.

B Baselines, Models, Datasets, & Metrics

In this section, we start by describing the baselines used as an upper-bound comparison. From there, we discuss the vLLMs and LLMs implemented in our experiments. Finally, we discuss the datasets in detail, followed by a comprehensive explanation of our evaluation metrics.

B.1 Baselines

We include random baselines, a digit-based classifier as a lower bounds, and severl SoTA baselines to act as upper bound.

B.1.1 Random Baselines

We include several label-only baselines to establish lower-bound performance and characterize dataset structure. The **uniform** baseline samples predictions uniformly at random from the set of class labels observed in the training data. The **prior** baseline samples predictions according to the empirical class distribution in the training set. The **majority** baseline assigns all instances to the most frequent class. These baselines do not use input features of any kind and serve to quantify chance-level performance and dataset imbalance effects.

B.1.2 Digit-Based Classifier

We additionally include a fully supervised **digit-based classifier** that operates directly on raw nu-

merical time-series values without any pretrained representations, prompting, or hidden states. Each time series is represented as a fixed-length vector $x = (x_1, x_2, \ldots, x_T)$, where each x_t is a scalar value at timestep t. These values are flattened and provided as input features to a standard logistic regression classifier.

The model learns a mapping $y = f(x_1, x_2, \ldots, x_T)$ using supervised training with class-balanced loss. Hyperparameters are selected via cross-validated grid search over the regularization parameter C. This baseline treats each timestep as an independent feature and does not explicitly model temporal dynamics beyond what can be captured linearly.

Its purpose is to quantify how much classification performance can be achieved using explicit numerical features alone, without relying on language models or learned representations. As such, it provides a non-LLM reference point for downstream performance.

B.1.3 Random Probe

To isolate the contribution of learned representations, we evaluate probing on randomly initialized models. Specifically, we use the same architecture as the pretrained models but with randomly initialized weights, and train linear classifiers on the resulting embeddings.

This baseline provides a practical lower bound on representation quality. Since no meaningful representations are learned, performance reflects the extent to which a classifier can extract signal from architectural structure alone. It therefore controls for the contribution of supervised learning applied after feature extraction and helps distinguish gains due to pretrained representations from those attributable to the probe.

B.1.4 kNN Classifier

We additionally include a non-parametric **k-nearest neighbors (kNN)** classifier using cosine similarity over embedding representations. Unlike linear probes, kNN does not involve parameter learning and operates directly on the geometry of the embedding space.

This baseline provides a complementary perspective on representation quality by evaluating whether class structure is preserved in the embedding space without supervised training. Strong kNN performance indicates that samples from the same class are naturally clustered, while weaker performance suggests that separability depends on learned classifiers.

B.1.5 SoTA Baselines

Attend follows the dedicated time-series encoder family. It uses self-attention to model cross-channel and temporal dependencies in multivariate sensor data and learns discriminative representations directly from raw time series, without adapting language-model token spaces (Abedin et al., 2021). **InstructTime** belongs to the time-series quantization and instruction-based prompting family. It discretizes time series into vector-quantized tokens, aligns them with domain-specific textual instructions, and uses generative instruction tuning to produce label text via next-token prediction (Cheng et al., 2025). **OneFitsAll** follows the encoder-alignment family. It reuses a frozen pretrained transformer from language or vision domains and learns lightweight input projections and normalization layers to map time-series patches into the model's embedding space, enabling a single backbone across tasks (Zhou et al., 2023). **TS2Vec** represents dedicated time-series representation learning. It learns multi-scale contextual embeddings through hierarchical contrastive learning over augmented temporal views, which are aggregated and used by simple downstream classifiers without language models (Yue et al., 2022). **Moment** belongs to the TS-Encoder family. It is a high-capacity transformer encoder pre-trained on large-scale, multi-domain time series using masked time-series modeling, and serves as a general-purpose representation learner that can be adapted to classification, forecasting, anomaly detection, and imputation via lightweight task-specific heads or linear probing (Goswami et al., 2024).

B.2 Models

To assess the generality of our findings, we evaluate a diverse set of pretrained models spanning different architectures and scales.[5] Motivated by prior work showing that visualizations provide a strong representation for time series classification (Liu et al., 2025), we primarily focus on vLLMs. Specifically, we evaluate `Llama-3.2-11B-Vision-Instruct` (Grattafiori et al., 2024), `Qwen/Qwen2.5-VL-32B-Instruct` (Bai et al., 2025), and

[5] All models are open source, as access to internal hidden states is required for probing.

`Mistral-Small-3.1-24B-Instruct-2503` (MistralAI). To verify that our conclusions extend beyond multimodal architectures, we also evaluate two text-only LLMs: `GPT-oss` (OpenAI et al., 2025) and `google/gemma-2-9b-it` (Team et al., 2024).

B.3 Datasets

Appendix C provides direct links to all source datasets used in our experiments, while Table 14 reports descriptive statistics. We selected datasets to span a wide range of temporal scales, sequence lengths, variable dimensionality, and label complexity. This diversity allows us to evaluate whether LLM-based prompting generalizes across fundamentally different forms of temporal structure. Short descriptions of each dataset can be found below.

CTU records 24-hour household electricity usage patterns and classifies computers as desktops or laptops, capturing differences in device power-consumption cycles (Chen et al., 2015). **EMG** consists of muscle response recordings under electrical stimulation, categorized into healthy, myopathy, and neuropathy classes (Goldberger et al., 2000). **HAR** contains multivariate accelerometer recordings of human activities such as walking, sitting, and stair climbing, and exhibits substantial inter-subject variability (Anguita et al., 2013). **HAD** similarly contains multivariate human activity data; we use both the tri-axis accelerometer data and tri-axis gyroscopic data (Zhang and Sawchuk, 2012). **RWC** comprises underwater acoustic recordings used to detect North Atlantic right whale calls in the presence of ocean noise (Cox et al., 2006). **TEE** originates from the FORTE satellite and classifies transient electromagnetic events (e.g., lightning strokes) based on waveform signatures (Eads et al., 2002).

B.4 Metrics

Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ denote a test set of N labeled time series, where $y_i \in \{1, \ldots, C\}$ and $\hat{y}_i$ is the predicted class. Let TP_c, FP_c, FN_c denote the counts of true positives, false positives, and false negatives for class c under the one-vs-rest convention.

Macro F1. For each class c, define precision and recall:

$$P_c = \frac{TP_c}{TP_c + FP_c}, \qquad R_c = \frac{TP_c}{TP_c + FN_c}.$$

Dataset	N-Channels	Series-Length	N-Classes	Train	Test
cpu	1	720	2	250	250
emg	1	1500	3	267	47
had	6	128	12	34779	7929
har	3	206	6	7,352	2,947
rwc	1	4000	2	25,499	4,501
tee	1	319	7	70	73

Table 14: Summary statistics for all time-series datasets used in our classification experiments.

The classwise F1 is

$$F1_c = \frac{2P_cR_c}{P_c + R_c}.$$

Macro F1 averages uniformly across classes:

$$F1_{\text{macro}} = \frac{1}{C}\sum_{c=1}^{C} F1_c.$$

Pass@K. For each test example i, we obtain n independent generations and mark each generation as correct/incorrect. Let c_i be the number of correct generations among the n samples for example i (in our experiments, $n = 20$). Following the standard unbiased estimator for pass@K, we define:

$$\widehat{\text{Pass@}K}_i = 1 - \frac{\binom{n-c_i}{K}}{\binom{n}{K}},$$

with the conventions $\widehat{\text{Pass@}K}_i = 0$ if $c_i = 0$, and $\widehat{\text{Pass@}K}_i = 1$ if $n - c_i < K$. We report the dataset-level score by averaging across examples:

$$\widehat{\text{Pass@}K} = \frac{1}{N}\sum_{i=1}^{N} \widehat{\text{Pass@}K}_i.$$

ΔPass@K. We quantify the benefit of repeated sampling by comparing multiple attempts against a single attempt:

$$\Delta\widehat{\text{Pass@}K} = \widehat{\text{Pass@}K} - \widehat{\text{Pass@}1}.$$

In particular, we report $\Delta\widehat{\text{Pass@}20} = \widehat{\text{Pass@}20} - \widehat{\text{Pass@}1}$.

C Dataset Statistics

In Table 14 we report the descriptive statistics of each of the datasets used in this study. Links for each dataset can be found below.

Dataset Links

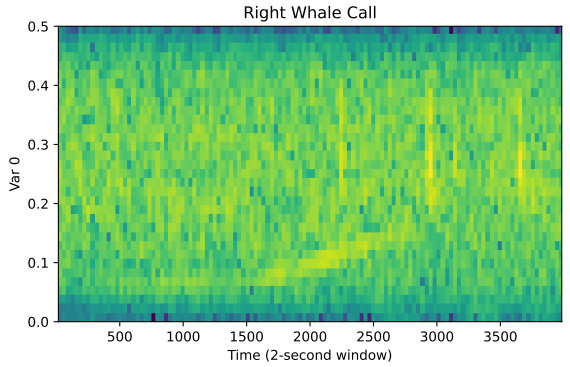

Figure 7: For the RWC dataset only, we represent the data as a spectrogram.

- **CTU**: https://www.timeseriesclassification.com/description.php?Dataset=Computers
- **HAD**: http://sipi.usc.edu/HAD/
- **EMG**: https://physionet.org/content/emgdb/1.0.0/
- **HAR**: https://archive.ics.uci.edu/dataset/240/human+activity+recognition+using+smartphones
- **RWC**: https://www.kaggle.com/competitions/whale-detection-challenge/data
- **TEE**: https://www.timeseriesclassification.com/description.php?Dataset=Lightning7

D Prompting Examples

This section illustrates how time series data are visualized in the visual domain and how these representations are incorporated into the prompting structures used for evaluation. We show both line-plot and spectrogram-based visualizations, text-only (d) prompts for inference, prompts used to generate prompt-variants, and examples of prompt-variants themselves.

D.1 Example Plots

Figures 7, 9, & 10 demonstrate how time series information we represent time series in the visual domain. For the RWC dataset (Figure 7), following the recommendations of Liu et al. (2025), the time series is rendered as a spectrogram. All other univariate datasets are visualized as line plots (see Figure 9 for an example). For multivariate datasets, each variable is plotted as a separate line with color-coded legends, expanding to multiple subplots when necessary (Figure10). Axis labels, titles, and legends are specified by the user prior to inference. Finally, in Appendix **??** we discuss a regression experiment serving to encourage additional research. Figure 8 shows an example of the type of plot used in these experiments.

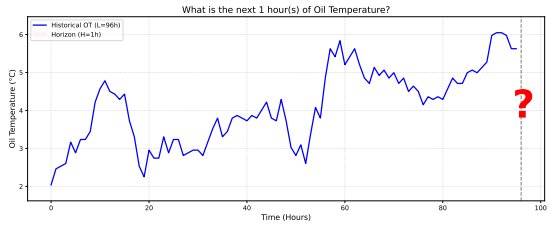

Figure 8: Example of a plot used for forecasting. This setup is auto regressive. This means that the look back window here is 96, but in the next prompt it will be 97 and include the previously predicted value. This repeats until a preset number of predictions are made (in our case 6).

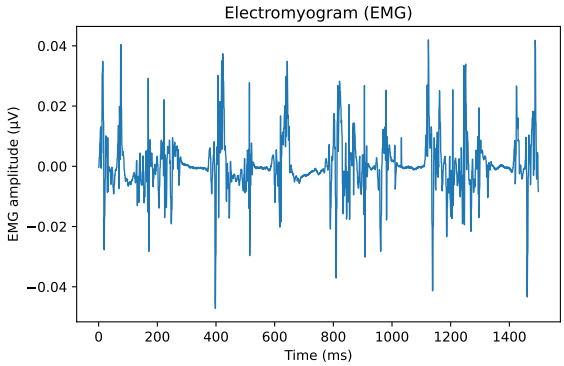

Figure 9: Example of a univariate time series line plot.

D.2 Prompting

Below we show the prompting structure for inference as well as the prompt used to make the variants.

Prompts for Inference. Figure 11 shows the instantiated prompt template used to classify the CTU dataset in the text-only (d) modality. In the visual (v) modality, the numerical time series is replaced with an appropriate visualization (see Section D.1). In the combined (d+v) modality, both the numerical sequence and the corresponding visualization are provided. The *Additional information that may help* is specified *a priori* and is adapted directly from Liu et al. (2025), who demonstrated the effectiveness of this prompting strategy.

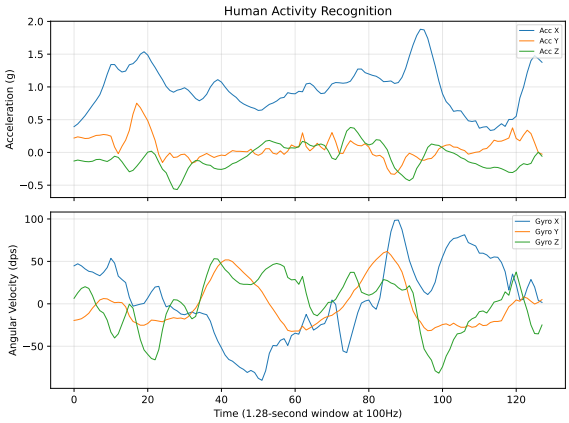

Figure 10: Example of a multivariate time series line plot.

Prompts for Prompt-Variant Generation. Figures 12 and 13 illustrate the prompts used to generate diverse, meaning-preserving variants. Figure 12 is used to generate variations of the system prompt, while Figure 13 is used to generate variations of the user message. For both system prompt and question generation, we issue two batch requests with batch size B=5, yielding a total of ten unique variants.

Examples of Prompt Variants. Figures 14, 15, & 16 present three prompt variants generated for the HAR dataset. Each variant exhibits distinct characteristics. For example, Variant 2 (Figure 15) is substantially more verbose than the others, whereas Variant 1 (Figure 14) is the most concise and omits expanded definitions for acronyms (e.g., using `RMS` without explicitly stating "Root Mean Square," as done in the other examples).

System Prompt

Play as a computer energy consumption analysis expert: determine whether this computer is a desktop or a laptop based on the 24-hour power consumption time series.

You will be given a multiple choice question and a time series. Your job is to use the time series to answer the question. **Do not explain your reasoning.** Use exactly this format: `The answer is [X] CLASS_NAME`. Example: `The answer is [D] CABBAGE`.

Additional information that may help:

- **Total daily consumption:** laptops typically consume less power than desktops.
- **Patterns over time:** laptops may show more uniform usage (sleep/hibernate); desktops may show sharper on/off contrasts.
- **Spikes:** desktops may exhibit higher spikes during active use.
- **Minimum/baseline:** laptops often have lower baselines; desktops may drop near zero if off or stay higher with peripherals.
- **Charging cycles:** periodic rises/falls consistent with battery charging suggest a laptop.
- **Variability:** laptops often have lower variance; desktops may fluctuate more across modes.
- **Day/night behavior:** strong overnight reductions may indicate sleep behavior typical of laptops.

User Prompt

Dataset: `CTU`

Question: determine whether this computer is a desktop or a laptop based on the 24-hour power consumption time series.

Options:

- `[A] Desktop`
- `[B] Laptop`

Time series:
`0 2 0 4, 0 3 9 5, 0 2 0 4, ... , 1 7 0 9, 1 3 2 6, 2 2 8 3`

Output format requirement: `The answer is [X] CLASS_NAME`

Figure 11: System and user prompts for the CTU dataset. The time series is abbreviated for readability. The *Additional information that may help*, *question*, and *options* change depending on the dataset.

Variation Generator: System Prompt Rewriter (System Prompt)

You are a prompt rewriter.
Rewrite the provided SYSTEM PROMPT into N distinct SYSTEM PROMPT variants that preserve meaning, constraints, and all factual task content, while changing phrasing, structure, and formatting.

Hard constraints (must obey):

- Do NOT add, remove, rename, or reorder any class names.
- Do NOT alter the semantic meaning of any class description.
- Preserve answer-choice letters (e.g., [A], [B]) exactly and in order.
- Preserve required output format strings exactly (punctuation/casing), e.g., `"The answer is [X] CLASS_NAME"`.
- Do NOT introduce new task instructions (e.g., reasoning, confidence).
- Maintain an academically appropriate tone.

Diversity requirements:

- Each variant must differ meaningfully in organization and wording.
- Use multiple presentation styles (headings, bullets, numbered steps, etc.).
- Avoid trivial paraphrases.

Output format (strict): return valid JSON only:

```
{
  "variants": [
    {"id": 1, "system_prompt": "..."},
    {"id": 2, "system_prompt": "..."}
  ]
}
```

No extra keys. No markdown. No commentary.

Variation Generator: System Prompt Rewriter (User Message Template)

Per-batch request (batch size = B):

```
Generate B rewritten variants of the following.

IMPORTANT:
- Do NOT change class names.
- Do NOT change answer letters or their order.
- Do NOT change the required answer format.
- Output must be valid JSON only.

<BASE_SYSTEM_PROMPT>
```

Batching: if total variants = N and batch size = B, the script makes N/B calls and then renumbers variant ids globally.

Figure 12: System and user prompts for system prompt variation generation. `<BASE_SYSTEM_PROMPT>` is replaced by the system prompt shown in Figure 11 or by the dataset-specific base system prompt used for evaluation. Two batch requests with batch size B=5 are used, producing ten total system prompt variants.

Variation Generator: General Question Rewriter (System Prompt)

You are a question rewriter.
Your task is to rewrite a GENERAL QUESTION into N distinct variants that preserve the task, label space, and decision criteria, while changing wording, structure, and phrasing.

Hard constraints (must obey):

- Do NOT change, rename, remove, or reorder answer choices.
- Preserve answer-choice letters exactly (e.g., [A], [B]) and their order.
- Do NOT introduce new labels, hints, or constraints.
- Do NOT add explanations, reasoning instructions, or output formatting rules.
- The rewritten question must ask the same classification decision.

Diversity requirements:

- Each variant must differ meaningfully in wording and structure.
- Vary tone (instructional vs. role-based), sentence structure, and framing.
- Avoid trivial paraphrases.

Output format (strict): return valid JSON only:

```
{
  "variants": [
    {"id": 1, "question": "..."},
    {"id": 2, "question": "..."}
  ]
}
```

No extra keys. No markdown. No commentary.

Variation Generator: General Question Rewriter (User Message Template)

Per-batch request (batch size = B):

```
Generate B rewritten variants of the following.

IMPORTANT:
- Do NOT change the answer choice letters, names, or their order.
- Do NOT add output formatting rules.
- Do NOT add reasoning/explanation instructions.
- Output must be valid JSON only.

<BASE_GENERAL_QUESTION>
```

Batching: if total variants = N and batch size = B, the script makes N/B calls and then renumbers variant ids globally.

Figure 13: System and user prompts for general question variation generation. `<BASE_GENERAL_QUESTION>` is replaced by the dataset-specific general question used for evaluation (e.g., *determine whether this computer* . . . from Figure 11). As in system prompt variation generation, we issue two batch requests with batch size B=5 to obtain ten total question variants.

Example Prompt Variations (Feature list with enumerated structure)

Variant 1.

```
You are tasked with human activity recognition based on tri-axial accelerometer
data (x, y, z) collected over time. Presented with a multiple-choice question
and corresponding time series data, your responsibility is to select the correct
activity. Provide your answer strictly in the following format:
"The answer is [X] CLASS_NAME" (e.g., "The answer is [D] CABBAGE"), without any
explanations.

Key features to consider for accurate classification include:
1. Statistical: mean, standard deviation, variance, median, range, IQR
2. Time-domain: RMS, zero crossing rate, signal magnitude area, time between peaks
3. Frequency-domain: FFT, PSD, spectral entropy, peak frequency
4. Correlation-based: correlations between axes
5. Geometrical: angles between axes, magnitude of acceleration vector
6. Entropy-based: signal entropy

Utilize these feature types to interpret the accelerometer signals and determine
the human activity. Analyze the sequence of tri-axial accelerometer measurements
(x, y, z) collected over time and classify the observed human activity into one
of these categories:
[A] WALKING [B] WALKING_UPSTAIRS [C] WALKING_DOWNSTAIRS [D] SITTING [E] STANDING [F] LAYING
```

Figure 14: Example one of three meaning-preserving prompt variations for human activity recognition.

Example Prompt Variations (Sectioned guidance with descriptive bullets)

Variant 2.

```
As an expert in human activity recognition, your task is to identify the activity
from the tri-axial accelerometer data (x, y, z) recorded over time. You will
receive a multiple-choice question along with the time series data. Use the time
series to select the correct answer without providing any explanation. Respond
strictly in the following format: "The answer is [X] CLASS_NAME". For example:
"The answer is [D] CABBAGE".

Additional guidance:
When analyzing accelerometer data for activity recognition, consider extracting
the following features:

1. Statistical Features:
- Mean: Average acceleration per axis. - Standard Deviation: Variation in acceleration values.
- Variance: Spread of data points around the mean. - Median: Middle value splitting the data.
- Range: Difference between max and min values.
- Interquartile Range (IQR): Spread between 25th and 75th percentiles.
2. Time-domain Features:
- Root Mean Square (RMS): Magnitude of acceleration. - Zero Crossing Rate (ZCR): Frequency of signal sign changes.
- Signal Magnitude Area (SMA): Energy expenditure over time. - Time between Peaks: Duration between local maxima.
3. Frequency-domain Features:
- Fast Fourier Transform (FFT): Frequency components analysis.
- Power Spectral Density (PSD): Power distribution across frequencies.
- Spectral Entropy: Complexity of frequency distribution.
- Peak Frequency: Dominant frequency component.
4. Correlation-based Features:
- Correlation between Axes: Coordination of movement.
5. Geometrical Features:
- Angles between Axes: Body orientation.
- Magnitude of Acceleration Vector: Overall acceleration magnitude.
6. Entropy-based Features:
- Signal Entropy: Complexity of the acceleration signal.

Using the tri-axial accelerometer data (x, y, z) recorded over time, identify
which human activity is being performed. Choose from the following options:
[A] WALKING [B] WALKING_UPSTAIRS [C] WALKING_DOWNSTAIRS [D] SITTING [E] STANDING [F] LAYING
```

Figure 15: Example two of three meaning-preserving prompt variations for human activity recognition.

Example Prompt Variations (Compact rubric-style presentation)

Variant 3.

```
As a professional in human activity recognition, your responsibility is to
identify the activity from tri-axial accelerometer data (x, y, z) recorded over
time. You will be provided with a multiple-choice question and the time series
data. Use the data to select the correct answer without any explanation,
adhering strictly to the format: "The answer is [X] CLASS_NAME". Example:
"The answer is [D] CABBAGE".

Consider these feature types when analyzing the accelerometer signals:
1) Statistical Features: mean, standard deviation, variance, median, range,
   interquartile range.
2) Time-domain Features: root mean square (RMS), zero crossing rate (ZCR),
   signal magnitude area (SMA), time between peaks.
3) Frequency-domain Features: fast Fourier transform (FFT), power spectral
   density (PSD), spectral entropy, peak frequency.
4) Correlation-based Features: correlation between axes.
5) Geometrical Features: angles between axes, magnitude of acceleration vector.
6) Entropy-based Features: signal entropy.

Acting as a specialist in human activity recognition, use the tri-axial
accelerometer readings (x, y, z) collected over time to classify the performed
activity. Your options are:
[A] WALKING [B] WALKING_UPSTAIRS [C] WALKING_DOWNSTAIRS [D] SITTING [E] STANDING [F] LAYING
```

Figure 16: Example three of three meaning-preserving prompt variations for human activity recognition.

System Prompt

You are an expert in energy grid analytics.

You are analyzing Oil Temperature (OT) from an electricity transformer.

User Prompt (Multimodal dv Example)

Attached is a plot showing the last 96 hours. The raw values for those hours are:

`12.55, 12.82, 13.10, ... , 14.22, 14.50`

[Plot is injected here (See Figure 8)]

TASK
Forecast the next hour of OT values (a single value).

CONSTRAINTS

- Your output should strictly follow the format: `'I predict the next hour will be <value>'`

OUTPUT Example
`I predict the next hour will be 37.6`

Figure 17: System and user prompts for the forecasting experiment on the ETT-small dataset. The example represents the multimodal (dv) setting.

Hyperparameter	Value
Model Architecture	GPT-2 (6 layers, pre-trained)
Model Dimension	768
Patch Size	16
Stride	16
Optimizer	RAdam
Learning Rate	0.001
Batch Size	64
Training Epochs	50
Dropout	0.1
Loss Function	Cross-Entropy

Table 15: Hyperparameters and training configuration for OneFitsAll model

Hyperparameter	Value
Epochs	300
Batch size	256
Learning rate	10^{-3}
LR decay factor	0.9
LR decay step	10 epochs
Sliding window (W)	24
Window overlap	50%
Mixup α	0.8
Center loss weight (β)	0.3
Dropout (features)	0.5
Dropout (RNN)	0.25
Dropout (classifier)	0.5
Hidden dimension	128
Filter size	5
Number of filters	64

Table 16: Training hyperparameters for Attend and Discriminate model

E Hyperparameters

wFor reproducibility, we provide the complete hyperparameter configurations for all baseline methods. These settings were either adopted from the original publications or carefully tuned to ensure fair comparison across all methods on our datasets.

E.1 OneFitsAll

All deep learning models are implemented in PyTorch using the Hugging Face Transformers library (Wolf et al., 2020). We employ a GPT-2-based architecture (Radford et al., 2019) where the pre-trained GPT-2 model serves as the backbone, with most parameters frozen, only the layer normalization and positional embedding layers remain trainable. The input time series are processed using a patching strategy, where patches are embedded into a high-dimensional space before being fed to the transformer layers. Training is performed using the RAdam optimizer (Liu et al., 2020) with cross-entropy loss, and model performance is evaluated using 5-fold stratified cross-validation on the training set. Standardization normalization is applied to input features, and the model is evaluated on a separate held-out test set using accuracy as the key metric. All hyperparameters are summarized in Table 15.

E.2 Attend

We implement our experiments using PyTorch. Our network is trained end-to-end using the Adam optimizer (Kingma and Ba, 2017) with a learning rate that decays periodically. The training data is partitioned into segments using a sliding window approach with 50% overlap between adjacent windows. We employ mixup augmentation as a regularization technique and incorporate center loss to encourage intra-class compactness in the learned feature space. Dropout is applied at multiple stages of the network, including the RNN layers and the classifier, to prevent overfitting. Model performance is evaluated using 5-fold cross-validation to ensure robust and reliable results. All hyperparameters are summarized in Table 16.

E.3 InstructTime

We implement InstructTime using Python 3.12.7 and PyTorch 2.7.1 on an Nvidia H200 GPU. The framework consists of two stages: (1) a TStokenizer (VQ-VAE) (van den Oord et al., 2017) for time series discretization, and (2) an InstructTime model built upon the pre-trained GPT-2 language model (Radford et al., 2019) for classification. For the TStokenizer, we employ standard VQ-VAE training settings with mean pooling. For the InstructTime model, we use AdamW optimizer (Loshchilov and Hutter, 2019) with weight decay and employ a cosine annealing learning rate scheduler with warmup (Loshchilov and Hutter, 2017). Following prior work on pre-trained language models, we use a smaller learning rate and batch size compared to training from scratch, as the PLM has been pre-trained with a large number of parameters. Mixed-precision training (Micikevicius et al., 2018) is enabled using PyTorch's automatic mixed precision (AMP) to accelerate computation. The detailed hyperparameter settings

Hyperparameter	Value
TStokenizer (VQ-VAE)	
Hidden dimension (d_{model})	64
Codebook size (n_{embed})	256
Wave length	24
Training epochs	60
Batch size (train/test)	32 / 64
Learning rate	5×10^{-4}
Learning rate decay	0.99
Weight decay	$1e^{-5}$
InstructTime	
Batch size	16
Learning rate	$1e^{-5}$
Weight decay	0.01
Warmup ratio	0.05
Training epochs	300
Max sequence length	230
Random seed	2024

Table 17: Hyperparameter settings for TStokenizer and InstructTime models

Hyperparameter	Value
Backbone model	`AutonLab/MOMENT-1-large`
Task name	classification
Series batch size	64
Window length (W)	512
Window stride	512
Patch length	8

Table 18: Hyperparameters for Moment baseline.

for both stages are summarized in Table 17.

E.4 Moment Embedding Extraction

We extract fixed-length embeddings using the pretrained `AutonLab/MOMENT-1-large` pipeline. Each time series is z-normalized using a per-channel `StandardScaler` fit on the training split and reused for the test split. To handle sequences longer than the model window, each series is segmented into non-overlapping windows of length W=512 with left-padding for the final partial window; an input mask marks padded positions. MOMENT outputs patch embeddings with patch length 8, which we average over the valid (non-padded) patches to obtain a window embedding. Window embeddings are then aggregated into a single series embedding using a length-weighted average across windows. All configuration values are summarized in Table 18.

Hyperparameter	Value
Representation Dimension	320
Hidden Channels	64
Residual Blocks	10
Kernel Size	3
Batch Size	8
Crop Length	3000
Normalization	Z-score (per channel)
Learning Rate	.0001
Epochs	10 or 30

Table 19: Hyperparameters and training configuration for TS2Vec.

E.5 TS2Vec

We implement TS2Vec using the authors' official PyTorch implementation, and, with the exception of epochs, used a fixed set of hyperparameters across all datasets. We used 10 training epochs for larger datasets (HAD, RWC, HAR) and 30 epochs for our smaller datasets (CTU, EMG, TEE). Input time series are standardized prior to training, and representations are extracted from the final encoder layer for downstream evaluation. All hyperparameters are summarized in Table 19.

F Use of AI Assistants

We used AI to help clean up and help with writing, but all thoughts and work are our own.