BENCHMARK DATASETS FOR LEAD-LAG FORECASTING ON SOCIAL PLATFORMS

Kimia Kazemian[*1], **Zhenzhen Liu**[*1], **Yangfanyu Yang**[2], **Katie Z Luo**[1], **Shuhan Gu**[1], **Audrey Du**[1], **Xinyu Yang**[2], **Jack Jansons**[1], **Kilian Q Weinberger**[1], **John Thickstun**[1], **Yian Yin**[2], **Sarah Dean**[1]
[1] Department of Computer Science and [2] Department of Information Science at Cornell University

ABSTRACT

Social and collaborative platforms emit multivariate time-series traces in which early interactions—such as views, likes, or downloads—are followed, sometimes months or years later, by higher impact like citations, sales, or reviews. We formalize this setting as **Lead-Lag Forecasting (LLF): given an early usage channel (the lead), predict a correlated but temporally shifted outcome channel (the lag).** Despite the ubiquity of such patterns, LLF has not been treated as a unified forecasting problem within the time-series community, largely due to the absence of standardized datasets. To anchor research in LLF, here we present two high-volume benchmark datasets—arXiv (accesses $\rightarrow$ citations of 2.3M papers) and GitHub (pushes/stars $\rightarrow$ forks of 3M repositories)—and outline additional domains with analogous lead–lag dynamics, including Wikipedia (page-views $\rightarrow$ edits), Spotify (streams $\rightarrow$ concert attendance), e-commerce (click-throughs $\rightarrow$ purchases), and LinkedIn profile (views $\rightarrow$ messages). Our datasets provide ideal testbeds for lead–lag forecasting, by capturing long-horizon dynamics across years, spanning the full spectrum of outcomes, and avoiding survivorship bias in sampling. We documented all technical details of data curation and cleaning, verified the presence of lead-lag dynamics through statistical and classification tests, and benchmarked parametric and non-parametric baselines for regression. Our study establishes LLF as a novel forecasting paradigm and lays an empirical foundation for its systematic exploration in social and usage data. Our data portal with downloads and documentation is available at `https://lead-lag-forecasting.github.io/`.

1 INTRODUCTION

The success of human activities is often measured by their collective impact, ranging from music streams and movie box office revenues to product sales and social media popularity. These impact metrics typically follow heavy-tailed distributions (Clauset et al., 2009) and slow decay patterns across timescales (Candia et al., 2019), making early identification of future hits fundamentally challenging (Cheng et al., 2014; Martin et al., 2016). At the same time, digital platforms increasingly log online user interactions—searches, views, downloads, likes, and shares—that often precede these long-term dynamics. These temporal lead-lag dynamics are remarkably ubiquitous, spanning domains as diverse as science (Haque & Ginsparg, 2009), economics (Wu & Brynjolfsson, 2015), arts (Goel et al., 2010), culture (Gruhl et al., 2005), and social movements (Johnson et al., 2016). A systematic understanding of such lead-lag dynamics is not only crucial for anticipating and optimizing impact in digital ecosystems, but also essential for designing effective strategies that identify and promote emerging innovations and products.

In this paper, we introduce **Lead-Lag Forecasting (LLF)** defined as follows: given a "lead" usage channel, predict a correlated but temporally shifted "lag" outcome channel. LLF aims to build upon the framework of early-signal prediction and further clarify evaluation goals that differ from generic long-horizon forecasting—it addresses a complementary regime to traditional forecasting where the predictor must transfer information across channels (e.g., access $\rightarrow$ citations) while generalizing to new series.

[*] Equal contribution

LLF problems exhibit three key "dual process" properties: (i) *observable early-phase interactions*—e.g., views, likes, github repository pushes, (ii) *meaningful outcome signal*—e.g., paper citations, purchases, github repository forks, and (iii) *predictive dependency*—the early dynamics often **cause or forecast** later outcomes through complex, non-deterministic mechanisms. Despite the ubiquity of such dual process dynamics across digital ecosystems, our quantitative understanding of their predictability remains very limited, largely due to the lack of standardized datasets for systematic LLF research. Though frequently recorded in internal web logs, early "lead" signals are not often made publicly available. Popular forecasting benchmarks—traffic (California Department of Transportation, 2025), electricity load (Trindade, 2015), taxi demand (New York City Taxi and Limousine Commission, 2025), ETTh (Zhou et al., 2021), M4 (International Institute of Forecasters, 2025), and related corpora—have spurred substantial progress on short-horizon and seasonal forecasting within a single measurement channel. While these benchmarks serve their intended applications well, they focus primarily on same-channel prediction rather than cross-channel dynamics with series generalization.

To address this gap, here we build and release [1] two high-volume datasets in science (accesses and citations of about 2.3M **arXiv** papers) and technology (pushes, stars, and forks of about 3 million **GitHub** repositories). These two examples represent ideal testbeds for LLF research: (1) The broad impact of papers and software often extends beyond their original communities, making them significant, real-world settings rather than artificial test cases (Yin et al., 2022). (2) Both systems have observable effects that unfold over years, presenting complex real-world challenges that traditional methods struggle to model (Ke et al., 2015; Jiang et al., 2021). (3) As our records include both the high volume of low-impact items and the small number of high-impact breakthroughs, the datasets provide unbiased coverage across the entire impact spectrum without survivorship bias.

To summarize, our contributions are the following: (1) we provide two datasets, one of which has never before been publicly available, and document technical details of data curation and cleaning; (2) we analyze the lead-lag dynamics in both datasets and demonstrate predictive signals in the paired channels; (3) we benchmark deep learning, parametric, and non-parametric prediction baselines. Together, this study establishes LLF as a novel forecasting paradigm and lays an empirical foundation for its systematic exploration in social data.

2 Related Work

Time Series Prediction Tasks. Time series prediction is a longstanding and active research area in machine learning. Two prominent settings for this modality include **rolling forecasting** (Zhou et al., 2021; Wu et al., 2021; Zhang & Yan, 2023) and **temporal point process estimation** (Mei & Eisner, 2017; Shchur et al., 2019; Zuo et al., 2020; Xue et al., 2023). *Rolling forecasting* is commonly used for applications like weather forecasting or load monitoring. In this setting, observations are recorded at regular time intervals. At each time step t, one uses a lookback window of the past L observations $\{\mathbf{x}_{t-L}, \cdots \mathbf{x}_{t-1}\}$ to predict the next H future steps $\{\mathbf{x}_t, \cdots \mathbf{x}_{t+H-1}\}$ (Zhou et al., 2021). *Temporal point processes* are often used for tasks like behavior modeling and critical event forecasting. In this setting, one is interested in modeling discrete events that occur at potentially irregular intervals in continuous time. Given past events $\{(t_1, c_1), \cdots (t_{n-1}, c_{n-1})\}$, where t_i denotes the time and c_i the type of event i, the goal is to predict the arrival time and type of the next event (Xue et al., 2023). Our new **arXiv** and **GitHub** datasets are relevant for such settings: the dense lead channels can be used for rolling forecasting, while the sparse lag channels can be reframed as temporal point processes. Beyond this contribution to the broader time-series community, though, our proposed lead-lag forecasting task differs from both settings. Unlike rolling forecasting, which often predicts near-future outcomes immediately following the context window, we introduce a gap between the lead window and the prediction horizon. Unlike temporal point processes, the lead channels in lead-lag prediction can consist of dense, regularly sampled measurements available at each time step. Our datasets introduce exciting challenges for time series prediction: models must generalize to unseen series, transfer information across channels, and handle substantially longer prediction ranges than existing benchmarks. These open new opportunities for developing scalable, flexible forecasting architectures. A detailed discussion of these challenges appears in the Appendix.

[1]Dataset portal: `https://lead-lag-forecasting.github.io/` For the responsible usage terms rationale, see Appendix A.

Table 1: **Comparison of time series datasets.** See the Appendix for additional details and citations.

Name	**Domain**	**Labeled Lag Event?**	**Multi. Variate?**	**# Series**	**Time Series Task**		**Sugg. Pred. Horizon**
					Forecast	**TTP**	
Electricity	Energy	✗	✗	370	✓		1 day
Solar	Energy	✗	✗	137	✓		4 hrs
ETT	Energy	✗	✓	69	✓		30 days
Weather	Weather	✗	✓	20	✓		30 day
NYC Taxi	Transport.	✗	✓	12M	✓	✓	*N/A*
UberTLC	Transport.	✗	✓	200K	✓	✓	*N/A*
Traffic	Transport.	✗	✗	862		✓	1 day
KDD Cup	Security	✗	✓	5M	✓	✓	*N/A*
Exchange	Economics	✗	✗	8	✓		24 days
Wiki Traffic	Network	✗	✗	145K	✓	✓	*N/A*
Retweet	Social	✗	✓	166K	✓	✓	6 hr
M1–5	M Competitions	✗	✗	1–100K	✓		$<$ 1 yr
Ours (arXiv)	Science	✓	✓	2.3M	✓	✓	5 yrs
Ours (Github)	Technology	✓	✓	3M	✓	✓	5 yrs

Time Series Datasets. We provide a summary and comparison of existing time series datasets in Table 1, with additional details and citations in the Appendix. These datasets are mostly designed for the rolling forecasting and temporal point process tasks discussed above. Popular datasets for rolling forecasting cover domains like energy and weather (Trindade, 2015; National Renewable Energy Laboratory, 2025; Max-Planck-Institut für Biogeochemie, 2025; Lai et al., 2018; International Institute of Forecasters, 2025). Common datasets for temporal point processes encompass domains like transportation and online activity (New York City Taxi and Limousine Commission, 2025; Mo, Shuheng, 2023; California Department of Transportation, 2025; Stolfo et al., 1999; muonneutrino, 2017; Zhao et al., 2015). While these datasets have driven advancement in their respective areas, they do not capture clear lead-lag relationships central to our task—they omit two axes that are *central* to LLF: (i) *Cross-channel prediction:* models must map an early "lead" channel to a delayed "lag" channel related to the same entity. (ii) *Cross-series generalization:* models are trained on many entities and evaluated on *new* entities that appear only at test time.

Although there are lines of work developing cross-series/variate capability with these datasets (Zhou et al., 2021), primary benchmark tasks still focus on same-variate prediction. Furthermore, the methods are not geared toward learning from thousands—or even millions—of series in order to forecast the future for hundreds of unseen ones. For example, traffic datasets treat each sensor (e.g., a camera on a highway) as a variable and extrapolate its own future load; they rarely ask the model to forecast demand at a brand-new sensor from patterns observed at other sensors.

In contrast, **arXiv** provides training data on 1.46M papers' access and citation trajectories and predicts citations for newly submitted papers given only an early access window, while **GitHub** provides from pushes & stars of 938K existing repositories to forecast forks of repositories that are unseen during training. Both datasets are an order of magnitude larger than popular rolling-forecasting corpora and introduce research domains—scholarly communication and open-source software—that are absent from prior work. They therefore constitute the first large-scale benchmarks that *jointly* test cross-channel and cross-series generalization, filling a critical gap in the time-series literature.

3 Problem Setting

Consider an entity, e.g., an academic paper or a code repository, $i \in \{1, \ldots, N\}$ observed on a platform that logs data containing *lead* and *lag* channels. The lead and lag sequences are discrete-time signals sampled at a predefined interval (e.g., daily or weekly) starting when the entity initializes (e.g., a preprint is posted). Early in the timeline, the lag channel may be sparse or entirely zero, as measurable impact often takes time to manifest. In particular, we define **lead channel** as a multivariate sequence $\mathbf{x}^{(i)}_{1:T_i} = (x^{(i)}_1, \ldots, x^{(i)}_{T_i}) \in \mathbb{R}^{T_i \times d_x}$ which captures early-phase interactions (e.g., downloads, pushes, views). The **lag channel** is multivariate sequence $\mathbf{y}^{(i)}_{1:T_i} = (y^{(i)}_1, \ldots, y^{(i)}_{T_i}) \in \mathbb{R}^{T_i \times d_y}$ recording delayed signals of impact (e.g., citations, sales), aligned to the same absolute timeline.

We fix a prediction horizon $H \in \mathbb{Z}_{>0}$, and define a *lookback window* $\tau < H$. We are interested in making a prediction based only on early observation up to cutoff τ about some behavior of $\mathbf{y}^{(i)}$

at a much later timepoint H. In full generality, the learner may observe both early lead and lag sequences $\mathbf{x}^{(i)}_{1:\tau}, \mathbf{y}^{(i)}_{1:\tau}$, though the early lag sequence may be sparse or uninformative. At prediction time, there is no access to the future lead values $\mathbf{x}^{(i)}_{\tau+1:H}$ nor lag values $\mathbf{y}^{(i)}_{\tau+1:H}$. The gap between τ and H may be significant, and this sets the LLF setting apart from much related work on timeseries forecasting. The goal is to predict a downstream outcome derived from the lag channel at H:

$$\hat{z}^{(i)} = f_\theta\big(\mathbf{x}^{(i)}_{1:\tau}, \mathbf{y}^{(i)}_{1:\tau}\big) \quad \text{with target} \quad z^{(i)} = g\big(\mathbf{y}^{(i)}_{1:H}\big),$$

where $g(\cdot)$ extracts a quantity of interest, like the value at a fixed future timepoint (e.g., $y^{(i)}_H$), a cumulative sum, or a binary label (e.g., top-10% impact). For example, we may use a sequence of preprint accesses over the first $\tau = 30$ days to predict whether the paper will receive at least 50 citations in the next $H = 5$ years. Unlike rolling forecasting (Zhou et al., 2021), LLF (i) ignores the intermediate values between τ and H; (ii) leverages cross-channel dependence between $\mathbf{x}$ and $\mathbf{y}$; and (iii) focuses on horizons where H is large relative to typical seasonality patterns, making naïve extrapolation ineffective.

We provide and analyze two novel benchmark datasets for investigating LLF:

1. **arXiv** : accesses, look-back $\mathcal{L} = \{30, 100, 365\} \to$ citations, horizons $\mathcal{H} = \{1825\}$ days;
2. **GitHub** : pushes, stars $\to$ forks, same $\mathcal{L}$ and $\mathcal{H}$.

We fix a long-term forecast horizon $H = 1825$ days (5 years) across both datasets. Rather than varying the prediction target, we vary the *lookback window*: the length τ of the available early lead signal. In Section 4 and 5 we investigate whether early-phase signals $\mathbf{x}^{(i)}_{1:\tau}$ and $\mathbf{y}^{(i)}_{1:\tau}$, are predictive of a lag outcome $z^{(i)} = g(\mathbf{y}^{(i)}_{\tau+1825})$. This setup enables us to study how much predictive power accumulates as more early interaction data becomes available.

We additionally outline a standard methodology for evaluating predictions, and present accuracy results for several baseline supervised learning models in Section 6 across $\tau \in \mathcal{T} = \{30, 100, 365\}$ days. Many lead-lag signals of interest follow a heavy tailed, power law distribution, and therefore it's important to use appropriate metrics to measure performance. Given a labelled dataset of N_{test} examples for evaluating performance, we consider metrics for two types of tasks: **(1) Classification**, which aims to identify high-impact entities defined by a threshold binarization of the lag outcome. Here, the metrics are AUROC and F1 score: the AUROC captures the model's ability to rank positive examples higher than negative ones across all thresholds, while the F1 score summarizes the precision and recall at a fixed decision threshold (e.g., predicting whether a paper will receive more than 50 citations). **(2) Regression**, which aims to predict the numerical values of the lag outcomes. The three evaluation metrics are mean absolute error in the linear (**MAE**) and log (**MAE-log**) space (computed with $z \leftarrow \log(1+z)$), and mean absolute percentage error (**MAPE**) in the linear space for high-impact entities ($z \geq k$).

$$\text{MAE} = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \big|\hat{z}^{(i)} - z^{(i)}\big|, \quad \text{MAPE} = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \left| \frac{\hat{z}^{(i)} - z^{(i)}}{z^{(i)}} \right| \text{ for } \mathcal{I} = \Big\{ i \mid z^{(i)} \geq k \Big\} \tag{1}$$

MAE is a direct measure of the average prediction from the ground truth outcomes, while MAE-log accounts for the long-tailed nature. MAPE quantifies the relative errors between the predicted and ground truth outcomes. We restrict this metric to high impact entities with $z \geq k$, to avoid instability from low-impact instances, e.g. $\hat{z} = 2$ when $z = 1$ yields 100% error but is inconsequential.

4 ARXIV DATASET

4.1 PROCESSING, STATISTICS, AND DISTRIBUTION

Data Sources. The dataset is built by an *left join* between **(i) the arXiv access logs** and **(ii) the citation graph in the Semantic Scholar** JSON dump extracted via the Datasets API [2].

In *arXiv Access Logs*, the raw file records 2.3M arXiv preprint IDs together with their submission dates and hashed IDs of users who accessed the preprint during the time period. Coverage is weekly

[2] Semantic Scholar Datasets API: `https://api.semanticscholar.org/api-docs/datasets`.

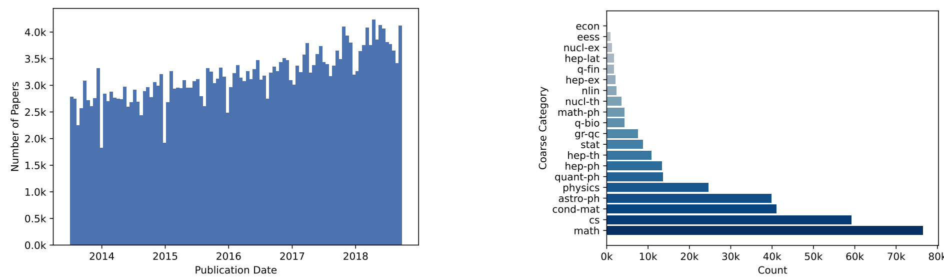

Figure 1: **Left:** Histogram of paper publication time in the `train` split. **Right:** Distribution across coarse categories in the `train` split.

from `2006-07-03` to `2013-06-30`, and daily from `2013-07-01` onward. Across the 4.87B total downloads, 44% originate from users averaging > 50 daily accesses or > 5000 total accesses; we regard these as bots and *remove* them. We then aggregate the access times over users into a (potentially sparse) access time series for each paper. In *Semantic Scholar Citations*, the dump contains 220M paper metadata records, from which we extracted 2.66B directed citation pairs $(\textit{citing}, \textit{cited})$. For any pair in which the *cited* paper has an arXiv ID, we approximate the citation time by the publication date of the *citing* paper (true citation dates are unavailable) and append it to that paper's citation sequence. This yields citation time series for 2M arXiv papers.

Merging and Anonymization. Performing an left join between access and citation data produces 2.3M unique papers with recorded arXiv submission, of which 2M have *both* citation and access records. For anonymity we drop paper IDs and submission dates in the released files. Feature descriptions appear in the Appendix.

Data Splits. Since arXiv switched from weekly to daily access logs on `2013-07-01`, and a small fraction (1.88%) of papers have other anomalies (see Appendix), we set aside these cases in a `train_extra` split. We then sample random `train` / `val` / `labeled_test` splits from papers published between `2013-07-01` and `2018-09-25`. The distribution of paper publication dates is illustrated in Figure 1 (Left). Papers submitted after `2018-09-25` form an `unlabeled_test` set. the final split sizes are 318K, 111K, 79K, 681K, and 1.14M for `train`, `val`, `labeled_test`, `unlabeled_test`, and `train_extra` respectively.

Subject-area Distribution. There are 152 fine-grained and 20 coarse first-choice categories. Figure 1 (Right) plots the coarse distribution for the train set. The four largest areas are `cs`, `math`, `cond-mat`, and `astro-ph`. Randomised splitting preserves this mix, whereas the `unlabeled test` split—being more recent—contains a noticeably higher share of `cs` papers.

4.2 Lead and Lag Analysis

We investigate the lead and lag dynamics in the arXiv dataset to assess whether early access metrics provide reliable indicators of future scholarly recognition. Our analysis focuses on cumulative accesses and citations at 30 days, 100 days, and 5 years.

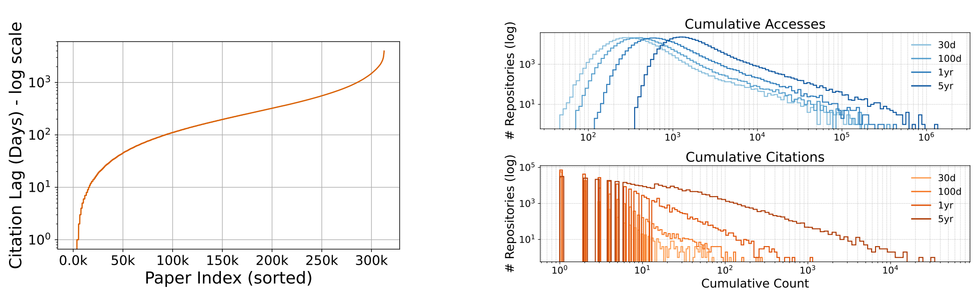

Figure 2: **Left:** Citation delay in days after first access. **Right:** Distribution of accesses and citations over time. Log-scaled histograms show cumulative accesses (top) and citations (bottom) at 30 days, 100 days, 1 year, and 5 years after publication. Colors progress from light to dark over time.

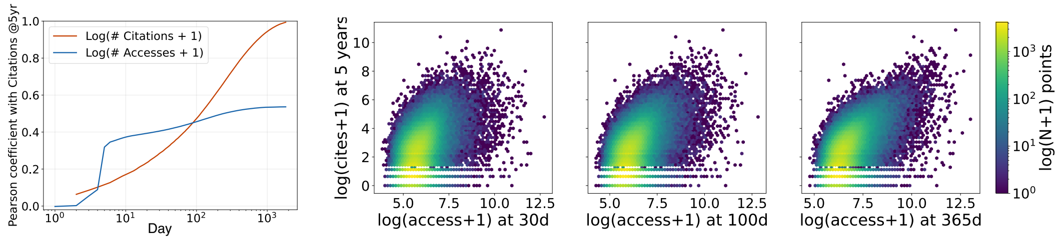

Figure 3: **Left:** Pearson correlation of early accesses (blue) and citations (orange) with 5-year citation count. Access data shows strong early correlation. After 3 months (90 days), citations gradually become more predictive. **Right:** Hexbin plots of log-transformed early accesses vs. five-year citations. Each subplot corresponds to the 30-, 100-, and 365-day access horizon. Color indicates the log-density of papers. These plots illustrate a clear positive association across all horizons, with the signal becoming sharper and more linear at longer horizons.

Channel Dynamics. Figure 2 illustrates the temporal dynamics of accesses and citations, showing that accesses clearly *lead* citations, which *lag* in numbers. Across the corpus, the first access consistently occurs before the first citation. At later times (darker shades), both distributions shift rightward and citations become increasingly dispersed and heavy tailed, reflecting variation in long-term impact. Notably, accesses distributions are bell-shaped even on shorter timescales, suggesting more predictable access patterns. These observations support early accesses as potential predictors of long-term scholarly recognition.

Predictive Power. Figure 3 (left) shows the Pearson correlation between early citations/accesses (cumulative counts) with 5-year citations. Early accesses exhibit stronger correlation with long-term impact than citations, which only surpass accesses around day 50. This highlights citation's delayed utility. Figure 3 (right) presents log-log hexbin plots of early accesses vs. long-term citations, revealing a clear, increasingly linear association over time. Density patterns also reveals a floor effect, where some papers receive early attention but few citations. These findings support early usage metrics as effective early indicators of scientific impact.

5 GITHUB DATASET

5.1 PROCESSING, STATISTICS, AND DISTRIBUTION

Data Sources. The GitHub event data are from GH Archive hosted on Google BigQuery (Grigorik, 2025), for which the coverage starts from `2011-02-12`. Events associated with 424M repositories are recorded between `2011-02-12` and `2024-12-31`. We select pushes, creates (creating a repository, branch, or Git tag), stars, and forks as our primary focus.

Due to the sparse nature of GitHub events, we focus on those associated with packages that have been released on some manager platform (e.g. PyPI, Conda, *etc.*). Specifically, we extract repository metadata associated to packages on 32 platforms recorded on Ecosyste.ms. We used an open-access data dump from Ecosyste.ms that covers 3M package repositories created between `2007-10-29` and `2024-06-04` [3]. We restrict to the package repositories created within the coverage of GH Archive (`2011-02-12` to `2024-12-31`). The left joining of the Ecosyste.ms metadata with the GH Archive event data results in 3M repositories in total, 2.9M of which have associated events recorded in GH Archive. The features of the resulting final dataset are detailed in the Appendix.

Data Splits. A random split gives a training data of size 938K, testing data of size 235K, and validation data of size 235K. The repositories with histories shorter than 1825 days (5 years) go into the unlabelled test set of size 1.57M, and the examples with problematic first events are saved to an extra training set of size 58K. Figure 4 shows the creation timeline of Github repositories in the dataset and the distribution of top 15 packages. Additional dataset details are included in the Appendix.

[3] Ecosyste.ms open data releases: `https://packages.ecosyste.ms/open-data`.

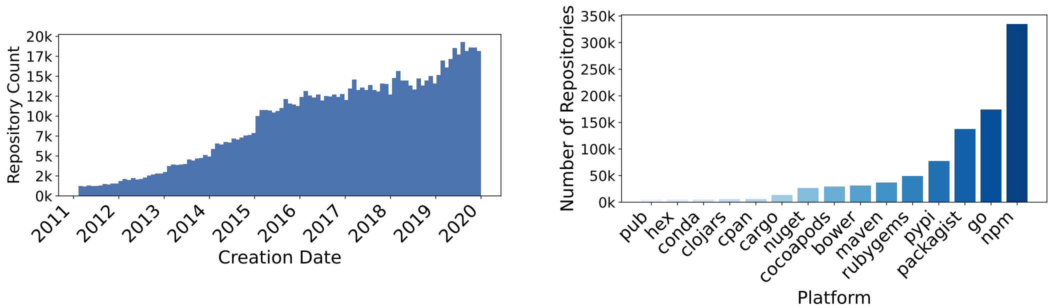

Figure 4: **Left:** Github Repository creation timeline. **Right:** Distribution across coarse packages.

5.2 Lead and Lag Analysis

We examine how different types of engagement signals on GitHub repositories evolve over time to determine whether early engagement patterns can predict long-term repository popularity and activity. Our analysis tracks cumulative counts of pushes, stars, and forks at days 30, 100, 365, and 5 years.

Channel Dynamics. Figure 5 (left) shows the first arrival time of events after the first push across repositories. The median repository receives its first star within approximately 10–30 days, whereas the first fork typically occurs after 100–200 days. By the 90th percentile, stars appear within a few hundred days, while forks may take several years. This temporal gap indicates that community engagement generally begins with a star and only later escalates to a fork. Figure 5 (right) shows engagement signal distributions across channels and time horizons. As time progresses from 30 days to 5 years (lighter to darker lines), all distributions shift right and widen, reflecting increasing heterogeneity in repository popularity and activity. Forks display the most rightward spread among the three, highlighting its role as a signal of sustained engagement akin to citations. Therefore, our benchmarks use forks as the target, and pushes and stars as early indicators. We leave the exploration of alternative choices of lead-lag pairs as future work.

Preditive Power. Figure 6 (left) shows their correlations with 5-year forks. Notably, early stars exhibit stronger correlation with long-term forks than early forks. Figure 6 (right) visualizes the relationship between early activity and long-term forks via log-log hexbin plots, revealing a consistent positive association that sharpens over time. Density patterns also show floor effects at low activity levels. These observations support early pushes and stars as practical early indicators of forks.

6 Baseline Experiments

We now present results for lead-lag prediction using baseline supervised learning methods. Our goal is twofold: to empirically validate the presence of predictive signal in our datasets, and to establish baseline performance benchmarks and evaluation methodology to ground future research.

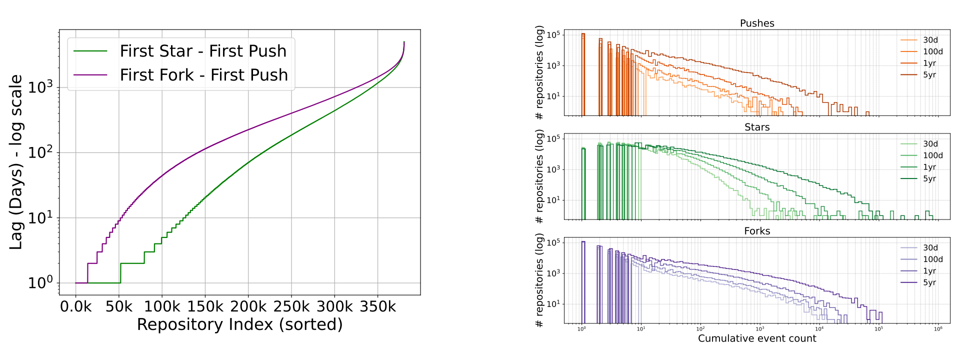

Figure 5: **Left:** Lag between first activity and first star and first fork event. **Right:** Distributions of GitHub engagement signals across time horizons. Log-scaled histograms showing cumulative counts of pushes (top), stars (middle), and forks (bottom) measured at 30 days, 100 days, 1 year, and 5 years after repository creation. Colors progress from lighter to darker as time advances.

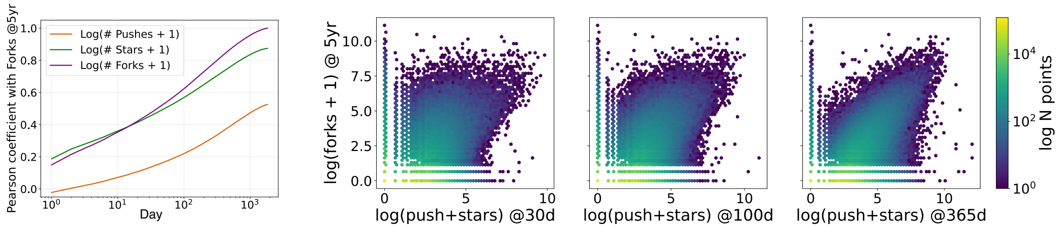

Figure 6: **Left:** Pearson correlation of early pushes (orange), stars (green) and forks (blue) with 5-year fork count. Stars have the strongest correlation in the first 2 weeks. Afterwards, forks gradually become more predictive. **Right:** Hexbin plots of log-transformed early push+star events vs. five-year forks. Each subplot corresponds to the 30-, 100-, and 365-day horizons. Color indicates the log-density of papers. These plots illustrate a positive association across all horizons, with the signal becoming sharper and more linear at longer horizons.

High-Impact Lag Classification. We first explore whether early activity alone can predict which items will land in the top 8.1% of arXiv papers (≥ 50 citations) or top 11.3% of GitHub repos (≥ 10 forks) after five years. We perform classification with logistic regrssion in three settings:

1. using one feature—the cumulative count of the lead signal at time τ;
2. using the raw features up to time τ;
3. using a time-series foundation model—Time-Moe (large) (Shi et al., 2024) - embeddings of raw features with a logistic regression head. We perform ablations across all channels, and pairs of channels, for both dataset and all model.

Details are presented in the appendix D.

The results in Table 2 and Table 3, illustrated in Figure 7, demonstrate predictive power even with one feature: using the single cumulative feature, the AUC climbs from 0.80 (day 30) to 0.86 (day 365) while F1 rises from 0.31 to 0.39 for arXiv from cross-channel prediction, and outperform same-channel predictions up to day 100, consistent with our correlation results in Figure 3. Github exhibits a similar pattern consistent with Figure 6. Additionally, we observe that using more channels (pushes + stars) performs better than single channel prediction across all models. Using all features generally outperforms using a single feature, and Time-MOE embeddings generally lead to enhanced results in most cases, indicating that more expressive features are potentially helpful. In all cases, predictive performance exceeds that of random baselines, providing clear evidence that early usage patterns contain meaningful signal about long-term impact.

Table 2: **Classification results for arXiv citation prediction across different horizons.**

Horizon	**Method**	**Citations from Cit.**		**Citations from Acc.**		**Citations from (Cit.+Acc.)**	
		F1	AUC	F1	AUC	F1	AUC
	Random	0.082	0.50	0.082	0.50	0.082	0.50
	All-positive	0.151	0.50	0.151	0.50	0.151	0.50
30 Days	One-feature LR	0.278	**0.623**	0.314	0.803	0.338	0.820
	Vanilla LR	**0.281**	0.622	**0.352**	0.824	0.369	0.842
	Time-MoE LR	**0.281**	0.622	0.349	**0.836**	**0.377**	**0.846**
100 Days	One-feature LR	0.300	**0.755**	0.347	0.833	0.395	0.870
	Vanilla LR	0.304	**0.755**	0.368	0.851	**0.434**	0.889
	Time-MoE LR	**0.334**	0.753	**0.425**	**0.866**	0.425	**0.895**
365 Days	One-feature LR	0.463	0.928	0.389	0.864	0.533	0.947
	Vanilla LR	0.485	**0.932**	**0.397**	0.870	0.562	0.954
	Time-MoE LR	**0.508**	0.929	0.388	**0.882**	**0.569**	**0.956**

Lag Outcome Regression. We next provide baseline benchmarks in the regression setting. We evaluate models using metrics defined in section 3. For MAPE, we use threshold $k = 50$ for arXiv and $k = 10$ for GitHub to designate high-impact items. We consider the standard application of five methods: MEY (Abrishami & Aliakbary, 2019), Linear Regression, KNN, MLP, Transformer (Vaswani et al., 2017) and KNN with embeddings from Time-MoE (large) (Shi et al., 2024). Description and implementation details are in the Appendix. We train on the training split of each dataset, and evaluate on the test split. We use all channels as inputs for all baselines except MEY, and use only the lag channel for MEY, since MEY does not support cross-channel prediction.

Table 3: **Classification results for GitHub fork prediction across different horizons.**

Horizon	Method	Forks from Pushes		Forks from Stars		Forks from Forks		Forks from (P+S)		Forks from (P+S+F)	
		F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC
	Random	0.113	0.50	0.113	0.50	0.113	0.50	0.113	0.50	0.113	0.50
	All-positive	0.204	0.50	0.204	0.50	0.204	0.50	0.204	0.50	0.204	0.50
10 Days	One-feature LR	0.189	0.534	0.296	0.639	0.281	**0.592**	0.294	0.679	0.308	0.695
	Vanilla LR	0.214	0.596	0.306	0.640	**0.283**	0.591	**0.308**	0.698	**0.327**	0.713
	Time-MoE LR	**0.245**	**0.623**	**0.325**	**0.641**	**0.283**	0.591	0.294	**0.725**	0.304	**0.722**
30 Days	One-feature LR	0.210	0.578	0.328	0.694	0.369	0.651	0.335	0.727	0.369	0.749
	Vanilla LR	0.249	0.631	0.369	**0.696**	**0.376**	0.650	**0.373**	0.751	**0.397**	0.769
	Time-MoE LR	**0.261**	**0.668**	**0.378**	0.689	**0.376**	**0.661**	0.314	**0.768**	0.339	**0.777**
100 Days	One-feature LR	0.248	0.640	0.435	0.773	0.456	**0.756**	0.426	0.800	0.466	0.831
	Vanilla LR	**0.294**	0.651	0.432	0.778	0.469	0.755	0.449	0.825	0.488	0.849
	Time-MoE LR	0.282	**0.717**	**0.468**	**0.777**	**0.482**	0.746	**0.487**	**0.844**	**0.505**	**0.864**

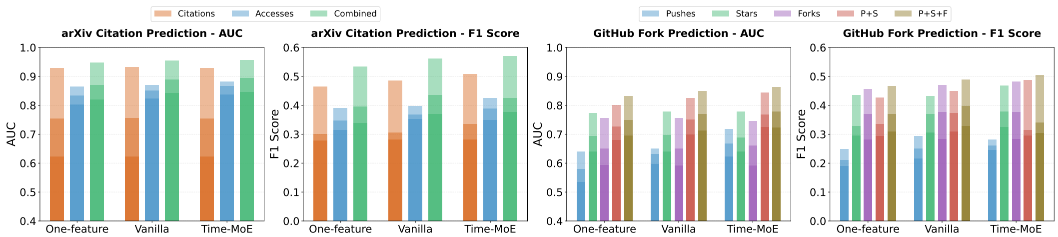

Figure 7: Classification results on arXiv (left) and GitHub (right) across different channels, models and input-horizon. Cross-channel prediction outperforms same-channel prediction for early input-horizon, consistent with correlations plots in Figure 6 and Figure 3. Deep features generally outperform non-transformed features, while even classification with a single feature outperforms random (F1 0.08 for arXiv and F1 0.11 for github). As we increase the input horizon models perform better.

We report the results on arXiv in Table 4 and results on GitHub in Table 5. In both cases, all methods generally achieve better performance as the input horizon increases from 30 days to 1 year, validating that more information facilitates prediction. Straightforward applications of deep learning methods yield performance gains, highlighting the potential for future research on more advanced deep learning approaches for this task. Notably, KNN attains strong performance, particularly on GitHub, for MAE metrics, but underperforms the deep learning methods on MAPE. We hypothesize that this effect arises due to the skewed data distribution: on GitHub, many repos have zero forks in 5 years. Deep learning methods show a clear advantage in predicting the high-impact instances. KNN using Time-MoE embeddings achieves comparable results to standard KNN on arXiv and GitHub. Thus, while we see that the extracted features contain some signal, we do not find compelling evidence that the time-series foundation model adds substantial value for this task.

Table 4: **Benchmarking results on the test split of the Arxiv dataset.**

Method	30 Days			100 Days			365 Days		
	MAE-log	MAE	MAPE	MAE-log	MAE	MAPE	MAE-log	MAE	MAPE
MEY	2.057	24.902	0.934	1.631	19.770	0.758	0.835	13.534	0.516
Linear Regression	0.841	22.895	1.393	0.767	16.725	0.756	0.574	11.247	0.463
KNN	0.932	19.121	**0.737**	0.856	17.613	0.690	0.656	12.967	0.512
MLP	0.842	17.651	0.763	0.771	16.193	0.678	0.576	11.656	0.459
Transformer	**0.832**	**17.422**	0.747	**0.759**	**15.669**	**0.668**	**0.568**	**10.987**	**0.443**
Time-MoE KNN	0.931	18.954	0.732	0.857	17.613	0.686	0.657	13.232	0.521

7 Discussion and Conclusion

We establish Lead-Lag Forecasting (LLF) as a formal prediction problem, motivated by the gap between observed *lead-lag dynamics* in important domains—including scientific and technological impact—and popular timeseries forecasting benchmarks. We catalyze research on LLF by curating and releasing two novel datasets: arXiv papers and GitHub repositories. We establish lead-lag relationships in streams of activity data and provide baseline numbers for several standard supervised machine learning methods on the task of predicting a 5 year outcome from as little as one month of observation. While our results demonstrate the existence of predictive signal, we speculate that there are opportunities for innovation to improve predictions.

Table 5: **Benchmarking results on the test split of the GitHub dataset.**

Method	30 Days			100 Days			365 Days		
	MAE-log	MAE	MAPE	MAE-log	MAE	MAPE	MAE-log	MAE	MAPE
MEY	0.775	25.467	2.526	0.649	15.721	1.464	0.423	9.438	0.834
Linear Regression	0.757	24.068	1.076	0.647	34.474	1.001	0.425	13.385	0.660
KNN	**0.675**	11.882	0.923	**0.556**	10.934	0.875	0.360	8.689	0.672
MLP	0.739	11.834	0.868	0.622	10.814	0.800	0.408	8.664	0.601
Transformer	0.732	**11.456**	**0.856**	0.612	**10.085**	**0.767**	**0.398**	**7.530**	**0.543**
Time-MoE KNN	0.716	12.05	0.925	0.578	11.11	0.872	0.355	8.92	0.686

Limitations. Our datasets contain only aggregate signals resulting from user activities, such as accesses, stars, and citations. They do not contain behavior traces at the individual level (i.e. *who* downloaded which paper), nor do they contain additional information about entities, such as author lists or content. It is likely that such information could provide complementary predictive power. Furthermore, the datasets that we curate reflect the social dynamics of particular platforms at particular times, and it is plausible that such dynamics could change or be altered by interventions based on these early signals. There are also limitations due to lag signals (citations, forks) serving only as approximations of quality, success, and impact. Nevertheless, we hope these datasets will enable better understanding of these promises and pitfalls of predicting long term outcomes.

8 ACKNOWLEDGMENTS

This work was funded by NSF CCF 2312774, NSF OAC-2311521, NSF IIS-2442137, NSF 2404035, NASA 80NSSC25M7039, a PCCW Affinito-Stewart Award, a gift to the LinkedIn-Cornell Bowers CIS Strategic Partnership, an AI2050 Early Career Fellowship program at Schmidt Sciences, and NewYork-Presbyterian for the NYP-Cornell Cardiovascular AI Collaboration.

REFERENCES

Ali Abrishami and Sadegh Aliakbary. Predicting citation counts based on deep neural network learning techniques. *Journal of Informetrics*, 13(2):485–499, 2019.

California Department of Transportation. Performance measurement system (pems). `https://pems.dot.ca.gov/`, 2025.

Cristian Candia, C Jara-Figueroa, Carlos Rodriguez-Sickert, Albert-László Barabási, and César A Hidalgo. The universal decay of collective memory and attention. *Nature human behaviour*, 3 (1):82–91, 2019.

Justin Cheng, Lada Adamic, P Alex Dow, Jon Michael Kleinberg, and Jure Leskovec. Can cascades be predicted? In *Proceedings of the 23rd international conference on World wide web*, pp. 925–936, 2014.

Aaron Clauset, Cosma Rohilla Shalizi, and Mark EJ Newman. Power-law distributions in empirical data. *SIAM review*, 51(4):661–703, 2009.

Sharad Goel, Jake M Hofman, Sébastien Lahaie, David M Pennock, and Duncan J Watts. Predicting consumer behavior with web search. *Proceedings of the National academy of sciences*, 107(41): 17486–17490, 2010.

Ilya Grigorik. Gh archive: Github public timeline archive. `https://www.gharchive.org/`, 2025. URL `https://www.gharchive.org/`. Project recording GitHub event data since 2011 with BigQuery integration.

Daniel Gruhl, Ramanathan Guha, Ravi Kumar, Jasmine Novak, and Andrew Tomkins. The predictive power of online chatter. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pp. 78–87, 2005.

Asif-ul Haque and Paul Ginsparg. Positional effects on citation and readership in arxiv. *Journal of the American Society for Information Science and Technology*, 60(11):2203–2218, 2009.

International Institute of Forecasters. Time series data. `https://forecasters.org/resources/time-series-data/`, 2025. Repository of time series datasets from the M-Competitions and other sources.

Song Jiang, Bernard Koch, and Yizhou Sun. Hints: Citation time series prediction for new publications via dynamic heterogeneous information network embedding. In *Proceedings of the web conference 2021*, pp. 3158–3167, 2021.

Neil F Johnson, Minzhang Zheng, Yulia Vorobyeva, Andrew Gabriel, Hong Qi, Nicolás Velásquez, Pedro Manrique, Daniela Johnson, Eduardo Restrepo, Chaoming Song, et al. New online ecology of adversarial aggregates: Isis and beyond. *Science*, 352(6292):1459–1463, 2016.

Qing Ke, Emilio Ferrara, Filippo Radicchi, and Alessandro Flammini. Defining and identifying sleeping beauties in science. *Proceedings of the National Academy of Sciences*, 112(24):7426–7431, 2015.

Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long- and short-term temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '18. ACM, June 2018. doi: 10.1145/3209978.3210006. URL `http://dx.doi.org/10.1145/3209978.3210006`.

Travis Martin, Jake M Hofman, Amit Sharma, Ashton Anderson, and Duncan J Watts. Exploring limits to prediction in complex social systems. In *Proceedings of the 25th international conference on world wide web*, pp. 683–694, 2016.

Max-Planck-Institut für Biogeochemie. Weather station data archive. `https://www.bgc-jena.mpg.de/wetter/`, 2025.

Hongyuan Mei and Jason M Eisner. The neural hawkes process: A neurally self-modulating multivariate point process. *Advances in neural information processing systems*, 30, 2017.

Mo, Shuheng. Uber nyc tlc data — exploratory data analysis. Kaggle, 2023. URL `https://www.kaggle.com/code/shuhengmo/uber-nyc-tlc-data-exploratory-data-analysis`. Using data from Uber NYC for-hire vehicles trip data (2021).

muonneutrino. Wikipedia traffic data exploration. Kaggle Notebook, 2017. URL `https://www.kaggle.com/code/muonneutrino/wikipedia-traffic-data-exploration`. Based on data from the Web Traffic Time Series Forecasting competition.

National Renewable Energy Laboratory. Solar power data for integration studies. `https://www.nrel.gov/grid/solar-power-data`, 2025. Last updated March 13, 2025.

New York City Taxi and Limousine Commission. Tlc trip record data. `https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page`, 2025.

Oleksandr Shchur, Marin Biloš, and Stephan Günnemann. Intensity-free learning of temporal point processes. *arXiv preprint arXiv:1909.12127*, 2019.

Xiaoming Shi, Shiyu Wang, Yuqi Nie, Dianqi Li, Zhou Ye, Qingsong Wen, and Ming Jin. Time-moe: Billion-scale time series foundation models with mixture of experts. *arXiv preprint arXiv:2409.16040*, 2024.

S. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. Chan. Kdd cup 1999 data, 1999. URL `https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`.

Artur Trindade. ElectricityLoadDiagrams20112014. UCI Machine Learning Repository, 2015. DOI: https://doi.org/10.24432/C58C86.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430, 2021.

Lynn Wu and Erik Brynjolfsson. The future of prediction: How google searches foreshadow housing prices and sales. In *Economic analysis of the digital economy*, pp. 89–118. University of Chicago Press, 2015.

Siqiao Xue, Xiaoming Shi, Zhixuan Chu, Yan Wang, Hongyan Hao, Fan Zhou, Caigao Jiang, Chen Pan, James Y Zhang, Qingsong Wen, et al. Easytpp: Towards open benchmarking temporal point processes. *arXiv preprint arXiv:2307.08097*, 2023.

Yian Yin, Yuxiao Dong, Kuansan Wang, Dashun Wang, and Benjamin F Jones. Public use and public funding of science. *Nature human behaviour*, 6(10):1344–1350, 2022.

Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The eleventh international conference on learning representations*, 2023.

Qingyuan Zhao, Murat A Erdogdu, Hera Y He, Anand Rajaraman, and Jure Leskovec. Seismic: A self-exciting point process model for predicting tweet popularity. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1513–1522, 2015.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11106–11115, 2021.

Simiao Zuo, Haoming Jiang, Zichong Li, Tuo Zhao, and Hongyuan Zha. Transformer hawkes process. In *International conference on machine learning*, pp. 11692–11702. PMLR, 2020.

Supplementary Material: Lead-Lag Forecasting on Social Platforms

A Data Release

The datasets can be downloaded at `https://lead-lag-forecasting.github.io/`.

arXiv Data. The original web logs contained fine grained access information by individual users; our timeseries data aggregates these patterns to avoid privacy concerns. In addition, there are two related concerns:

1. **Ethics:** the arXiv organization has not historically made information about paper views/downloads public. There is a "neutrality policy" against appearing to promote specific preprints on the site. By removing preprint identifiers from the timeseries, we address this concern.
2. **Re-identification:** The access timeseries are linked with citations, which are publicly available. In principle, this enables (partial) re-identification of the preprints. Agreement to responsible usage will reduces this risk while still allowing large-scale scholarly analysis.

GitHub Data. We also release the GitHub activity logs data used in our study. These files are publicly readable without credentialization.

B Additional Details about the arXiv Dataset

Dataset Features. Below we provide a description of each key present in the arXiv dataset:

- `first_publication`: difference in days between the publication date of the first version of the paper and arXiv submitted date;
- `first_access`: difference in days between the date of the first access and arXiv submitted date;
- `first_citation`: difference in days between the date of the first citation and arXiv submitted date;
- `cumulative_accesses`: cumulative number of accesses from the arXiv submitted date, e.g., `[1, 1, 2, ..., 3, 3]`;
- `cumulative_citations_offset`: cumulative number of accesses from the arXiv submitted date, subtracting the initial citations on Day 0;
- `initial_citations`: total number of citations before and on the arXiv submitted date.

`initial_citations` can be nonzero due to a mismatch between the actual citation date and our inferred date based on the first publication of the citing papers: most of the recorded citations before the submitted date actually come from later versions of the citing papers. In order to avoid exposing too much later signal, we subtracted the initial citations from cumulative citation counts to form `cumulative_citations_offset`. `cumulative_citations_offset` can be empty for a paper due to missing citation data. Other anomalies include late first access (after day 4 [4]) and publication before submitted to arXiv (S1).

Additional Details on Dataset Splits. The analysis and benchmark model training are performed over the `train` split of the arXiv dataset. The benchmark evaluations are conducted on the `test` split. We use the `cumulative_accesses` key for the accesses sequences, and `cumulative_citations_offset` for the citations sequences.

The `train_extra` split contains papers that are filtered out because of the following anomalies:

[4] Most papers gain immediate access after being published on arXiv. However, some are not immediately released after submission. Therefore, we used a threshold of 4 to quantify late first access.

Table S1: **Percentages of papers with data anomalies in the arXiv dataset.**

Anomaly	Submitted before daily precision < 2013-07-01 (37%)	Daily precision labeled 2013-07-01 - 2018-09-24 (25%)	Daily precision unlabeled > 2018-09-24 (38%)	Entire dataset
Missing cumulative_citations_offset data	13.73%	11.15%	20.10%	15.51%
first_access > 4	79.01%	2.97%	4.43%	31.45%
first_citation < 0	16.89%	20.17%	17.52%	17.96%
first_publication < 0	7.02%	7.87%	7.06%	7.25%

- Papers that are not in the citation data obtained from the Semantic Scholar Datasets API (16% filtered out).
- Papers that are published before or on 2013-06-30, as daily access records are not available before this date (37% from the previous step filtered out). Another 36% of papers after the previous filtering have histories shorter than 5 years and are put into the unlabelled test split.
- Papers with first access later than day 4 since the date of submission to arXiv, as this suggests that the preprint has abnormal access sequence with little early signal (2% from the previous step filtered out).

C ADDITIONAL DETAILS ABOUT THE GITHUB DATASET

Dataset Features. Below we provide a description of each key present in the GitHub dataset:

- id: unique identifier of the GitHub repository, as a string of {owner_name}/{repo_name};
- platform: list of name strings of platforms on which the repository is hosted, e.g., ['pypi', 'npm'];
- created_date: date of the repository creation recorded by Ecosyte.ms;
- first_create: difference in days between the date of the first CreateEvent (i.e., repository, branch, or tag creation) and created_date.
- first_push: difference in days between the date of the first PushEvent and created_date;
- first_star: difference in days between the date of the first WatchEvent (star) and created_date;
- first_fork: difference in days between the date of the first ForkEvent and created_date;
- cumulative_pushes_w_creates: cumulative number of PushEvents (pushes) plus CreateEvents indexed by day from created_date, e.g., [1, 1, 2, ..., 3, 3];
- cumulative_pushes: cumulative number of PushEvents (pushes) indexed by day from created_date;
- cumulative_stars: cumulative number of WatchEvents (stars) indexed by day from created_date;
- cumulative_forks: cumulative number of ForkEvents (forks) indexed by day from created_date.

In principle, first_create should be 0 as the first CreateEvent coincides with repository creation. However, it is nonzero for 35% of data, due to the repository's first CreateEvent missing in GH Archive or not matching the repository creation recorded by Ecosyte.ms. There can also be other data anomalies (S2), such as events before created_date due to repository renaming.

Additional Details on Dataset Splits. The analysis and benchmark model training are performed over the train split of the GitHub dataset. The benchmark evaluations are conducted on the test split. While the cumulative count features are taken as prediction inputs and targets, the first event features can be used for filtering. We moved the examples (2%) with negative first_push, first_star, or first_fork to train_extra.

Table S2: **Percentages of package repositories with data anomalies in the GitHub dataset.**

Anomaly	**Labeled** ¡ `2020-01-01` (48%)	**Unlabeled** $\geq$ `2020-01-01` (52%)	**Entire dataset**
`first_create != 0`	36.69%	34.21%	35.41%
`first_push < 0`	2.05%	2.30%	2.18%
`first_star < 0`	0.56%	0.53%	0.54%
`first_fork < 0`	0.36%	0.29%	0.32%

D Additional Experimental Details

D.1 Classification

In the following, for each horizon $t \in \{30, 100, 365\}$ for arXiv and $t \in \{10, 30, 100\}$ for GitHub, and each threshold of 50 for arXiv and 10 for GitHub we perform the following:

One Feature Logistic Regressoin: For each horizon t we fit a logistic regressor on log-transformed cumulative counts *at* time t:

$$\widehat{LR}_t^{(i)} = \sigma\big(w^\top x_t^{(i)} + b\big), \quad x_t^{(i)} = \begin{cases} \big[\text{log(cumulative accesses)}_t^{(i)}\big] & (arXiv) \\ \big[\text{log(cumulative pushes)}_t^{(i)},\ \text{log(cumulative stars)}_t^{(i)}\big] & (GitHub) \end{cases}$$

The logistic regression from Scikit-Learn library, with balanced class weights and random seed 42 is used to fit the model.

Vanilla Logistic Regression: For each horizon, we fit a logistic regressor on log-transformed cumulative counts *up to* that horizon.

The regressor is implemented in Pytorch, using the `BCEWithLogitsLoss`, with the positive weight set accordingly for each dataset (11.24 for arXiv and 7.81 for GitHub). Models are trained for 35 epochs with a learning rate of 0.01 and AdamW optimizer.

Time-MoE logistic regression: For each entity, for a fixed input horizon t we construct a feature vector by concatenating the daily cumulative counts in log space across the input horizon. We normalize the input with the global mean and standard deviation computed across all entities and time steps in the training set, separately for each channel. We inference Time-MoE (large) (Shi et al., 2024), a mixture of exports foundation model with 200M active parameters to embed each input sequence. The regressor is implemented in Pytorch, using `BCEWithLogitsLoss`, with the positive weight set accordingly for each dataset (11.24 for arXiv and 7.81 for GitHub). Models are trained for 35 epochs with a learning rate of 0.01 and AdamW optimizer. For the results shown in the main body, we used the final hidden layer of the model along with mean pooling to extract the embeddings.

D.2 Regression

We provide a description of each baseline in the lag outcome regression task benchmarking below. Each model was trained on a single GPU. While our experiments were conducted using NVIDIA Titan RTX and NVIDIA GeForce RTX 3090 GPUs, the models are also compatible with GPUs that have smaller memory capacity.

Mean of Early Years (MEY): MEY is a simple but strong baseline for citation prediction. For each paper, it calculates the average number of new citations observed at each time step within the input horizon, and uses this as the predicted number of new citations for each future time step. While some previous works (Abrishami & Aliakbary, 2019) use yearly resolution, our implementation uses daily resolution as our longest input horizon is one year. MEY is a univarate method, so we use only the lag channel for prediction, excluding the lead channels. And as MEY only leverages the each entity's own input horizon, predictions are directly made over the test set.

K-Nearest Neighbors (KNN): For each entity, we construct a feature vector by concatenating the daily cumulative counts in log space across the input horizon for all lead and lag channels, i.e. accesses and citations for arXiv, and pushes, stars and forks for GitHub. To make a prediction for a test entity, we identify its five nearest neighbors in the training set based on the L2 distance of the feature vectors, and use the median of the ground truth lag outcomes from these neighbors as the final prediction.

Linear Regression: Similar to KNN, the input variables consist of daily cumulative counts in log space across the input horizon for all lead and lag channels. The output variable is the lag outcome, i.e. cumulative count at year 5, in the log space. A linear regression model is fit on the training set, and then applied for prediction on the test set.

Multi-Layer Perceptron (MLP): The input and output variables are similar to those of linear regression. Additionally, we normalize the input with the global mean and standard deviation computed across all entities and time steps in the training set, separately for each channel. We use a 3-layer MLP with hidden size 1024 and `tanh` activations in-between. The model is trained on the training set for 5 epochs, and then applied for prediction on the test set.

Transformer: Each entity is represented as a sequence over the input horizon, where each time step is a vector containing the log-transformed cumulative counts for all channels. We apply the same input normalization as used for MLP. The architecture consists of learned positional encoding, followed by 3 attention blocks with bidirectional multi-head attention and pre-Layer Normalization. We use 4 attention heads, hidden size 256 and feed forward size 1024. To predict the lag outcome in the log space, we apply adaptive pooling across the sequence outputs, followed by a linear layer. The model is trained on the training set for 5 epochs, and then applied for prediction on the test set.

Time-MoE K-Nearest Neighbors (Time-MoE KNN): Each entity is same as transformer with the same normalizations. The entities are fed to the Time-MoE (Shi et al., 2024) and embeddings of sequences up to each input horizon are used as features for the KNN. We show ablations for embeddings from the middle decoder layers and last hidden layer. We also show ablations of taking the last token, and average and max pooling along the token axis. To make a prediction for a test entity, we identify its five nearest neighbors in the training set based on the L2 distance of the feature vectors, and use the median of the ground truth lag outcomes from these neighbors as the final prediction. The results using mean pooling + final layer are shown in the body of the paper for both arXiv and GitHub.

Table S3: **Ablation study results for Time-MoE KNN on the arXiv test split.**

Pooling Method	Layer	30 Days			100 Days			365 Days		
		MAE-log	MAE	MAPE	MAE-log	MAE	MAPE	MAE-log	MAE	MAPE
Max Pooling	Middle Layer	0.933	19.005	0.742	0.861	17.686	0.686	0.649	12.822	0.504
	Last Layer	0.930	19.018	**0.731**	0.867	17.982	0.693	0.660	13.307	0.533
Mean Pooling	Middle Layer	0.928	19.011	0.734	0.852	17.577	0.682	0.649	12.977	0.506
	Last Layer	0.931	18.954	0.732	0.857	17.613	0.686	0.657	13.232	0.521
Last Token	Middle Layer	**0.926**	18.877	0.738	**0.851**	17.349	0.677	**0.647**	**12.758**	**0.498**
	Last Layer	0.927	**18.856**	0.740	0.847	**17.158**	**0.670**	0.657	13.195	0.508

Table S4: **Ablation study results for Time-MoE KNN on the GitHub test split.**

Pooling Method	Layer	30 Days			100 Days			365 Days		
		MAE-log	MAE	MAPE	MAE-log	MAE	MAPE	MAE-log	MAE	MAPE
Max Pooling	Middle Layer	**0.666**	**11.88**	0.926	**0.552**	**10.90**	0.877	**0.347**	**8.38**	0.658
	Last Layer	0.675	12.00	0.929	0.558	11.08	0.886	0.356	8.75	**0.656**
Mean Pooling	Middle Layer	0.696	11.99	0.930	0.633	11.11	**0.855**	0.351	8.75	0.678
	Last Layer	0.716	12.05	0.925	0.578	11.11	0.872	0.355	8.92	0.686
Last Token	Middle Layer	**0.666**	11.93	**0.918**	0.561	11.13	0.867	0.404	9.36	0.707
	Last Layer	0.712	12.10	**0.918**	0.565	11.25	0.877	0.378	9.58	0.703

E DISCUSSION OF FORECASTING LIMITATIONS

Our datasets introduce exciting challenges beyond standard forecasting in several aspects:

Cross-series Generalization: With 2.4M papers and 3M repos, models need to be able to predict for unseen entities, requiring broad pattern recognition beyond entity-specific trends.

Cross-channel Prediction: Models must map early signals (accesses or stars/pushes) to different outcome types (citations or forks), not just extrapolate the same variable.

Temporal Gap and Sparsity: A long gap (30–365 day input $\rightarrow$ 5-year target) means the lag signal is often sparse or zero early on, forcing models to find predictive signals without short-term cues.

Long-tailed distributions: In our datasets, citations and forks follow power-law distributions. Only a small percentage achieve high impact, so models must learn from rare events without overfitting to common low-impact cases.

Current forecasting methods face two primary limitations when applied to lead-lag forecasting (LLF):

Cross-Series Generalization Constraints. Prevailing models are typically designed for forecasting future values within the same set of series, while LLF requires prediction for entirely unseen series.

- **Training/Testing Paradigm Mismatch:** Existing approaches assume chronological splits across all series (e.g., training on months 1–10, testing on months 11–12 for every energy station in the ETT dataset (Zhou et al., 2021)). In LLF, training and testing instead involve disjoint series (e.g., training on papers published up to 5 years ago, testing on entirely new papers), which current architectures are not designed to handle.
- **Scalability Constraints:** Most of the sophisticated models such as the Informer (Zhou et al., 2021) treat all variates within a time window as a single sample. With 1M+ variates in our datasets, this approach becomes computationally infeasible and architecturally rigid, since it prevents accommodating new variates at inference.

Temporal Prediction Scope Mismatches.

- **Long-Horizon Accumulation Error:** Conventional point-process models such as the Transformer Hawkes (Zuo et al., 2020) forecast only short horizons (1–96 steps ahead). Extending to 5-year predictions ($\approx$ 1825 steps) requires iterative forecasting, where compounding errors undermine reliability.
- **Periodicity Assumptions:** Architectures such as Autoformer (Wu et al., 2021) rely on capturing regular seasonal patterns. However, long-term dynamics in LLF do not exhibit such periodicity, rendering these assumptions ineffective.

F RELATED DATASETS

Electricity (Trindade, 2015): customers' electricity consumption data; *Solar* (National Renewable Energy Laboratory, 2025): solar power and day-ahead forecasts for simulated PV plants; *ETT* (Zhou et al., 2021): load, oil temperature, location, climate and demand for Electricity Transformer stations; *Weather* (Max-Planck-Institut für Biogeochemie, 2025): load, weather, renewable, and synchrophasor measurements for climate modeling; *NYC Taxi* (New York City Taxi and Limousine Commission, 2025): trip records for taxis in NYC; *UberTLC* (Mo, Shuheng, 2023): trip records for for-hire vehicles in NYC under the TLC regulation; *Traffic* (California Department of Transportation, 2025): sensor-based traffic measurements across California highways; *KDD Cup* (Stolfo et al., 1999): normal and attack traffic patterns for intrusion detection; *Exchange* (Lai et al., 2018): foreign exchange rates relative to USD; *Wiki Traffic* (muonneutrino, 2017): page views for Wikipedia articles; *Retweet* (Zhao et al., 2015): retweets of tweets on Twitter; *M Competitions* (International Institute of Forecasters, 2025): real-world time series used in forecasting competitions spanning many domains and frequencies.