MOMENT: A FAMILY OF OPEN TIME-SERIES FOUNDATION MODELS

Mononito Goswami* CMU†
Konrad Szafer‡ CMU†
Arjun Choudhry‡ CMU†
Yifu Cai CMU†
Shuo Li UPenn§
Artur Dubrawski CMU†

ABSTRACT

We introduce MOMENT, a family of open-source foundation models for general-purpose time-series analysis. Pre-training large models on time-series data is challenging due to (1) the absence of a large and cohesive public time-series repository, and (2) diverse time-series characteristics which make multi-dataset training onerous. Additionally, (3) experimental benchmarks to evaluate these models, especially in scenarios with limited resources, time, and supervision, are still in their nascent stages. To address these challenges, we compile a large and diverse collection of public time-series, called the Time-series Pile, and systematically tackle time-series-specific challenges to unlock large-scale multi-dataset pre-training. Finally, we build on recent work to design a benchmark to evaluate time-series foundation models on diverse tasks and datasets in limited supervision settings. Experiments on this benchmark demonstrate the effectiveness of our pre-trained models with minimal data and task-specific fine-tuning. Finally, we present several interesting empirical observations about large pre-trained time-series models. Our code is available anonymously at `anonymous.4open.science/r/BETT-773F/`.

1 Introduction

Time-series analysis is an important field encompassing a wide range of applications ranging from forecasting weather patterns Schneider and Dickinson [1974] or detecting irregular heartbeats using Electrocardiograms Goswami et al. [2021], to identifying anomalous software deployments Xu et al. [2018]. Due to its significant practical value and the unique challenges that modeling time-series data poses, time-series analysis continues to receive substantial interest from academia and industry alike. However, modeling such data typically requires substantial domain expertise, time, and task-specific design.

Large pre-trained language Touvron et al. [2023], Devlin et al. [2019], Chung et al. [2022], vision Li et al. [2023a], and video Day et al. [2023] models, typically perform well on a variety of tasks on data from diverse domains, with little or no supervision, and they can be specialized to perform well on specific tasks. We unlock these key capabilities for time-series data and release the **first family of open-source large pre-trained time-series models**, which we call MOMENT. The models in this family (1) serve as a building block for diverse **time-series analysis tasks** (e.g., forecasting, classification, anomaly detection, and imputation, etc.), (2) are effective **out-of-the-box**, i.e., with no (or few) particular task-specific exemplars (enabling e.g., zero-shot forecasting, few-shot classification, etc.), and (3) are **tunable** using in-distribution and task-specific data to improve performance.

MOMENT is a family of high-capacity transformer models, pre-trained using a masked time-series prediction task on large amounts of time-series data drawn from diverse domains. Below we summarize our key contributions.

C1: Pre-training data. A key limiting factor for pre-training large time-series models from scratch was the lack of a large cohesive public time-series data repositories Zhou et al. [2023], Gruver et al. [2023], Jin et al. [2023], Ekambaram et al. [2024], Cao et al. [2023]. Therefore, we compiled **The Time-series Pile**, a large collection of publicly available

*Correspondence at `mgoswami@andrew.cmu.edu`.
†Auton Lab, Robotics Institute, Carnegie Mellon University
‡KS and AC contributed equally, order decided using random generator.
§Department of Computer and Information Science, University of Pennsylvania

data from diverse domains, ranging from healthcare to engineering to finance. The Time-series Pile comprises of over 5 public time-series databases, from several diverse domains for pre-training and evaluation (Tab. 9).

C2: Multi-dataset pre-training. Unlike text and images, which have largely consistent sampling rates and number of channels, time-series frequently vary in their temporal resolution, number of channels[5], lengths, and amplitudes, and sometimes have missing values. As a result, large-scale mixed dataset pre-training is largely unexplored. Instead, most methods are trained on a single dataset, and transferred across multiple datasets, but with modest success Wu et al. [2023], Oreshkin et al. [2021], Narwariya et al. [2020].

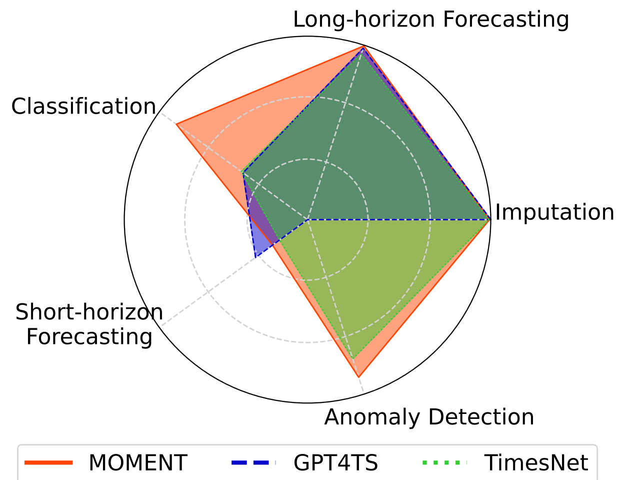


Figure 1: `MOMENT` can solve multiple time-series analysis tasks well (App. D).

C3: Evaluation. Holistic benchmarks to evaluate time-series foundation models on diverse datasets and tasks are in their nascent stages. To evaluate `MOMENT`, we build on the multi-task time-series modeling benchmark first proposed by Wu et al. [2023] along multiple dimensions. For each of the **5** time-series modeling tasks, namely, short- and long-horizon forecasting, classification, anomaly detection, and imputation we evaluate `MOMENT` against (1) both state-of-the-art deep learning as well as statistical baselines, on (2) more task-specific datasets, (3) using multiple evaluation metrics, (4) exclusively in limited supervision settings (e.g., zero-shot imputation, linear probing for forecasting, unsupervised representation learning for classification).

Finally, we explore various properties of these pre-trained time-series models. In particular, we study whether `MOMENT` is aware of intuitive time-series characteristics such as frequency and trend, and the impact of initialization, model size scaling, and cross-modal transfer.

2 Related Work

Transformers and patching for time-series modeling. There is a growing body of work utilizing transformers for various time-series analysis tasks Wen et al. [2023]. One issue with applying transformers to time-series data is the complexity of the self-attention mechanism, which grows quadratically with the size of input tokens (or length of time-series). Nie et al. [2023] demonstrated that treating time-series sub-sequences (or patches) as tokens instead of individual time points is a simple, efficient, and effective mechanism for learning useful representations for forecasting. Drawing inspiration from prior work, we build on top of the transformer architecture which takes disjoint time-series sub-sequences (or patches) as input.

Masked Representation Learning. Masked pre-training is a widely-used self-supervised learning task where a model learns to accurately reconstruct masked portions of its input. Masked language Devlin et al. [2019], Raffel et al. [2020] and image modeling Xie et al. [2022], Li et al. [2023b] have been successfully utilized to learn models from vast quantities of unlabeled data, which can generalize to a variety of downstream tasks.

For time-series data, prior work has primarily focused on contrastive representation learning Yue et al. [2022], Eldele et al. [2021], Franceschi et al. [2019]. However, contrastive learning relies on data augmentation, which is both subjective and data-dependent. In contrast, some studies mask portions of time-series using zeros and learn a model to reconstruct them Nie et al. [2023], Zerveas et al. [2021], Dong et al. [2023], Li et al. [2023c].

Representation learning via masking is well-suited to all the downstream tasks we care about, especially forecasting and imputation, as they are instances of the masked reconstruction problem. Owing to its simplicity and success in vision and language domains, we use the masked prediction task to pre-train our model, using a special embedding (see `[MASK]` in Fig. 3) to mask time-series patches instead of zeros.

Cross-modal transfer learning using language models. Lu et al. [2022] had first shown that transformers pre-trained on text data (LLMs) can effectively solve sequence modeling tasks in other modalities. Some recent studies have

[5] Temporal resolution reflects sampling frequency of time-series (e.g., hourly, daily); Channel is a single univariate time-series in multivariate data Ekambaram et al. [2024].

leveraged this inherent ability of language pre-trained transformers to “reprogram" LLMs for time-series analysis using parameter efficient fine-tuning and suitable tokenization strategies Zhou et al. [2023], Gruver et al. [2023], Jin et al. [2023], Cao et al. [2023], Ekambaram et al. [2024]. However, some of these models Jin et al. [2023], Gruver et al. [2023] with billions of parameters demand significant memory and computational resources to perform well. We complement this line of research with three empirical observations (Sec 4.3): we show that (1) transformers trained on time-series can also model sequences across modalities, (2) during pre-training, randomly initializing weights lead to lower pre-training loss, than initializing with language modeling weights, and (3) models pre-trained on time-series outperform LLM-based models such as Zhou et al. [2023], Jin et al. [2023] on many tasks and datasets.

Unanswered Questions. To the best of our knowledge, two questions remain largely unanswered in prior work on time-series modeling. First, all existing time-series models are (pre-)trained and fine-tuned on individual datasets Nie et al. [2023], Yue et al. [2022], Wu et al. [2023], Zhou et al. [2023], and the benefits (or drawbacks) of large-scale multi-dataset pre-training remains unexplored Wen et al. [2023]. Second, there is very limited work on time-series modeling in limited supervision settings, such as zero-shot forecasting Oreshkin et al. [2021], or few-shot classification Narwariya et al. [2020]. In our work, we consider both these questions and *show that pre-training a model of sufficient capacity on a large corpus of unlabeled time-series data can in fact enable it to provide reasonably accurate predictions in limited-supervision settings.*

3 Methodology

We first collect a large number of public time-series data into the **Time-series Pile** and then use it to pre-train a **transformer model** on the **masked time-series prediction task**. We discuss each of these steps in the following sections.

3.1 The Time-series Pile

Unlike natural language processing and computer vision, where large-scale datasets such as The Pile Gao et al. [2020], and ImageNet-1K Russakovsky et al. [2015] are easily available for pre-training, public time-series datasets are much smaller, scattered, and largely task-specific Ma et al. [2023], Zhou et al. [2023], Gruver et al. [2023]. To bridge this gap, we collate multiple time-series from 4 task-specific, widely-used **public** repositories resulting in a large number of time-series spanning diverse domains, and time-series characteristics such as lengths, amplitudes, and temporal resolutions. We call this collection the Time-series Pile.

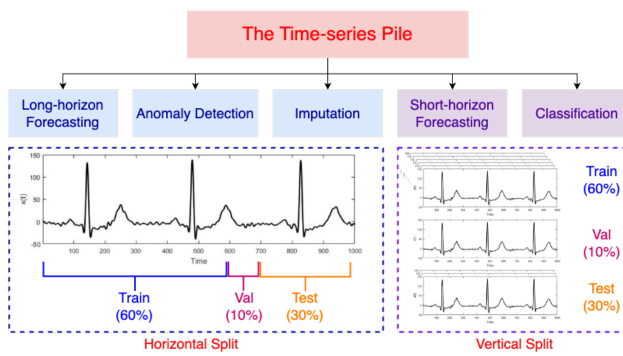


Figure 2: **Time-series Pile data splits**. To prevent data contamination, we carefully partition all datasets into disjoint train, validation, and test splits. We adhere to the predefined splits provided by the creators of each dataset. In cases where such splits are unavailable, we randomly sample 60% of the data for training, 10% for validation, and 30% for testing. We only use the training splits of all datasets for pre-training.

Informer long-horizon forecasting datasets Zhou et al. [2021] is a collection of 9 datasets that are widely used to evaluate long-horizon forecasting performance Wu et al. [2023], Nie et al. [2023], Challu et al. [2023]: 2 hourly and minutely subsets of the Electricity Transformer Temperature (ETT) Zhou et al. [2021], Electricity Trindade [2015], Traffic California Department of Transportation [2024], Weather Max Planck Institute for Biogeochemistry [2024], Influenza-like Illness (ILI) Centers for Disease Control and Prevention [2024], and Exchange-rate Lai et al. [2018].

Monash time-series forecasting archive Godahewa et al. [2021] is a collection of 58 publicly available short-horizon forecasting datasets with a total of over 100K time-series, spanning a variety of domains and temporal resolutions.

UCR/UEA classification archive Dau et al. [2018] comprises of 159 time-series datasets which are frequently used to benchmark classification algorithms Ismail Fawaz et al. [2019]. These datasets belonging to seven different categories (Image Outline, Sensor Readings, Motion Capture, Spectrographs, ECG, Electric Devices, and Simulated Data), vary substantially in terms of the number of classes and the size of the training set.

TSB-UAD anomaly benchmark Paparrizos et al. [2022a] is a recent collection of 1980 univariate time-series with labeled anomalies from 18 anomaly detection datasets proposed over the past decade. This collection includes both synthetic and real-world time-series originating from a wide range of sources such as the human body, spaceships, environment, and web serves.

Minimizing data contamination using careful train-test splitting. We carefully split each dataset into disjoint training, validation, and test splits, based on splits specified by data creators. When these splits are not available, we randomly sample 60% of the data for training, 10% for validation, and 30% for testing. Long-horizon forecasting and anomaly detection datasets are typically long time-series, which are split horizontally as shown in Fig. 2. Conversely, short-horizon forecasting and classification datasets often contain multiple short time-series. For these datasets, a complete time-series is either training, validation, or testing. We use the same random seed, set to 13, throughout our experiments, from pre-training to downstream evaluation, thus ensuring that the MOMENT only observes the training splits of datasets during pre-training.

3.2 Model Architecture

MOMENT receives a univariate time-series $\mathcal{T} \in \mathbb{R}^{1 \times T}$, and a mask $M = \{0, 1\}^{1 \times T}$ of length T. 0 and 1 denote unobserved and observed time-stamps, respectively. Reversible instance normalization Kim et al. [2022] is applied to the observed time-series before breaking it into N disjoint patches of length P. Each patch is then mapped to a D-dimensional embedding, using a trainable linear projection if all time steps are observed, and a designated learnable mask embedding [MASK] $\in \mathbb{R}^{1 \times D}$, otherwise. These N *patch embeddings* serve as input to the transformer model which retains their shape ($1 \times D$) throughout its operations. Each transformed patch embedding is then used to reconstruct both masked and unmasked time-series patches, using a *lightweight* prediction head. The goal of the prediction head is to map the transformed patch embeddings to the desired output dimensions. Since this particular prediction head enables time-series reconstruction, we call it the *reconstruction head*. Fig. 3 shows an overview of our model.

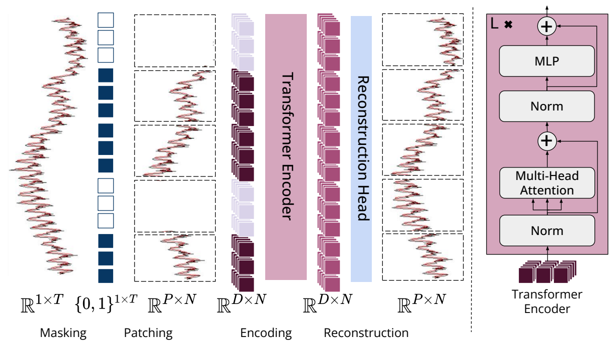


Figure 3: **Overview of** MOMENT. A time-series is broken into disjoint fixed-length sub-sequences called patches, and each patch is mapped into a D-dimensional patch embedding. During pre-training, we mask patches uniformly at random by replacing their patch embeddings using a special mask embedding [MASK]. The goal of pre-training is to learn patch embeddings which can be used to reconstruct the input time-series using a light-weight reconstruction head.

Our transformer encoder retains the modifications proposed by Raffel et al. [2020] to the original Transformer Vaswani et al. [2017]. Specifically, we remove the additive bias from the Layer Norm Ba et al. [2016], and place it before the residual skip connections He et al. [2016], and use the relation positional embedding scheme Shaw et al. [2018]. Below we summarize the intuition behind some of our key design decisions.

Handling varying time-series characteristics. Time-series vary in length, number of channels, amplitudes, and temporal resolutions. We address variable length by restricting MOMENT's input to a univariate time-series of a fixed length $T = 512$. As is common practice, we sub-sample longer time-series, and pad shorter ones with zeros on the left[6]. Moreover, segmenting time-series into patches quadratically reduces MOMENT's memory footprint and computational complexity, and linearly increases the length of time-series it can take as input. We handle multi-variate time-series by independently operating on each channel along the batch dimension. Like recent studies Zhou et al. [2023], Nie et al. [2023], we found that modeling each channel independently is an effective strategy for modeling multivariate time-series. Finally, re-scaling and centering time-series using reversible instance normalization enables MOMENT to model time-series with significantly different temporal distributions Kim et al. [2022]. We did not explicitly model the temporal resolution of time-series, since this information is often unavailable outside of time-series forecasting datasets.

Intentionally simple encoder. Closely following the design of transformers in the language domain allows us to leverage their scalable and efficient implementations (e.g., gradient checkpointing, mixed precision training).

Light-weight prediction head. We use a lightweight prediction head instead of a decoder of the same size as the encoder, to enable the necessary architectural modifications for task-specific fine-tuning of a limited number of trainable parameters while keeping the bulk of parameters and the high-level features learned by the encoder intact.

[6] We found a large majority of classification datasets to have time-series shorter than 512. Besides, a look-back window of length 512 was found to be sufficient for accurate long-horizon forecasting Nie et al. [2023].

Tasks	Supervision	Datasets	Metrics	Baselines	Experimental Setting
Long-horizon Forecasting	Linear Probing	ETT-h1/h2/m1/m2, Electricity, Traffic, Weather, Exchange, ILI	MSE, MAE	Time-LLM, GPT4TS, TimesNet, PatchTST, FEDFormer, DLinear, N-BEATS, Stationary, LightTS	Look-back window $L = 512$, Forecast horizon $H = \{24, 60\}$ (ILI), $\{96, 720\}$ (rest)
Short-horizon Forecasting	Zero-shot	M3 and M4 competition datasets (subset)	sMAPE[7].	GPT4TS, TimesNet, N-BEATS, AutoARIMA, AutoTheta, AutoETS, Seasonal Naive, Naive, Random Walk	Statistical methods fit on individual time-series. Deep learning methods are trained on a source dataset & evaluated on a target dataset of the same temporal resolution.
Classification	Unsupervised representation learning	UCR Classification Archive (subset)	Accuracy	GPT4TS, TimesNet, TS2Vec, T-Loss, TNC, TS-TCC, TST, CNN, Encoder, FCN, MCNN, MLP, ResNet, t-LeNet, TWIESN DTW	All models except `MOMENT` were trained on each individual dataset. Quality of unsupervised representations measured using the accuracy of a SVM trained on them.
Anomaly Detection	Linear probing, Zero-shot	UCR Anomaly Archive (subset)	Adjusted Best F1 VUS-ROC	GPT4TS, TimesNet, Anomaly Transformer, DGHL, Anomaly Nearest Neighbor	Reconstruction-based anomaly detection with window size $= 512$ MSE between observed and predicted time-series is used as the anomaly criterion
Imputation	Linear probing, Zero-shot	ETT-h1/h2/m1/m2, Electricity, Weather	MSE, MAE	GPT4TS, TimesNet, Linear, Naive, Cubic Spline, Nearest Neighbors	Randomly mask contiguous sub-sequences of length 8 Masking ratios: $\{12.5\%, 25\%, 37.5\%, 50\%\}$

Table 1: **Experimental benchmark.** We evaluate `MOMENT` on 5 time-series analysis tasks with an emphasis on limited memory, compute, and supervision settings.

3.3 Pre-training using Masked Time-series Modeling

We pre-train `MOMENT` using the masked time-series modeling task. Fig. 3 presents an overview of our pre-training procedure. During training, we first mask a small number of patches uniformly at random by replacing their patch embeddings with a learnable mask embedding `[MASK]`. The corrupted time-series patches are then fed into the transformer encoder to learn patch representations, which are used to reconstruct the original time-series using a lightweight reconstruction head. The pre-training objective is to minimize the *masked reconstruction error i.e.* the Mean Squared Error between the ground truth and the prediction, averaged over the masked patches.

Pre-training Setup. We pre-train three different sizes of `MOMENT`, roughly corresponding to the sizes of encoders in `T5-Small`, `Base`, and `Large`. Specifically, the `Base` (`Small`, `Large`) model uses a 12 (6, 24) layer Transform with hidden dimensions of size $D = 768$ (512, 1024), 12 (8, 16) attention heads, and feed-forward networks of size 3072 (2048, 4096), resulting in approximately 125 (40, 385) million parameters. All weights are randomly initialized before pre-training. All models take an input time-series of length $T = 512$, breaking it into $N = 64$ disjoint patches of length $P = 8$. We mask 30% of the patches uniformly at random during pre-training.

We use the Adam optimizer with weight decay Loshchilov and Hutter [2019] with $\lambda = 0.05$, $\beta_1 = 0.9$, $\beta_2 = 0.999$. We clip the gradient at 5.0, train models using a batch size of 2048, and use cosine learning rate schedule with initial and final learning rates of $1e^{-4}$ and $1e^{-5}$, respectively. We use gradient checkpointing Radford et al. [2021] to improve training throughput and save memory, and train all models in a mixed precision setting, using `float-32` for numerically unstable operations, e.g. layer normalization, and `bfloat-16`[8], otherwise. We train all models for 2 epochs.

3.4 Fine-tuning on Downstream Tasks

`MOMENT` can be seamlessly used for multiple time-series analysis tasks. In this work, we consider 5 practical time-series analysis tasks as examples, namely: long- and short-horizon forecasting, classification, anomaly detection, and imputation. For forecasting tasks with horizon H, we replace the reconstruction head with a forecasting head, which first flattens all the N D-dimensional patch embeddings into a $N \times D$ dimensional vector, and then projects it into a H-dimensional time-series via a linear projection layer. For all other tasks, we retain the reconstruction head. We provide detailed descriptions of each task and `MOMENT`'s configuration in App. D.

Fine-tuning settings. `MOMENT` can either be fine-tuned end-to-end, or linear probed ($\texttt{MOMENT}_{\texttt{LP}}$) by freezing all parameters except for those in the reconstruction or forecasting head. Additionally, for some tasks such as anomaly detection, unsupervised representation learning and imputation, `MOMENT` can also be used in a zero-shot ($\texttt{MOMENT}_{\texttt{0}}$) setting by retaining its reconstruction head.

4 Experimental Setup and Results

We extend the experimental benchmark introduced by Wu et al. [2023] across along various dimensions. Below, we outline the design choices of our benchmark and highlight its key distinctions from TimesNet[9].

[8]`https://cloud.google.com/tpu/docs/bfloat16`

[9]In this section, we use TimesNet to refer to the benchmark proposed by Wu et al. [2023] instead of their model.

Methods		MOMENT$_{\text{LP}}$		Time-LLM		GPT4TS		PatchTST		DLinear		TimesNet		FEDFormer		Stationary		LightTS		N-BEATS	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	0.154	0.209	-	-	0.162	0.212	0.149	0.198	0.176	0.237	0.172	0.220	0.217	0.296	0.173	0.223	0.182	0.242	0.152	0.210
	720	0.315	0.336	-	-	0.326	0.337	0.314	0.334	0.333	0.362	0.365	0.359	0.403	0.428	0.414	0.410	0.352	0.386	0.331	0.359
ETTh1	96	0.387	0.410	0.408	0.429	0.376	0.397	0.370	0.399	0.375	0.399	0.384	0.402	0.376	0.419	0.513	0.491	0.424	0.432	0.399	0.428
	720	0.454	0.472	0.523	0.514	0.477	0.456	0.447	0.466	0.472	0.490	0.521	0.500	0.506	0.507	0.643	0.616	0.547	0.533	0.608	0.573
ETTh2	96	0.288	0.345	0.285	0.348	0.285	0.342	0.274	0.336	0.289	0.353	0.340	0.374	0.358	0.397	0.476	0.458	0.397	0.437	0.327	0.387
	720	0.403	0.439	0.399	0.435	0.406	0.441	0.379	0.422	0.605	0.551	0.462	0.468	0.463	0.474	0.562	0.560	0.863	0.672	1.454	0.847
ETTm1	96	0.293	0.349	0.384	0.403	0.292	0.346	0.290	0.342	0.299	0.343	0.338	0.375	0.379	0.419	0.386	0.398	0.374	0.400	0.318	0.367
	720	0.405	0.416	0.437	0.429	0.417	0.421	0.416	0.420	0.425	0.421	0.478	0.450	0.543	0.490	0.585	0.516	0.527	0.502	0.448	0.448
ETTm2	96	0.170	0.260	0.181	0.269	0.173	0.262	0.165	0.255	0.167	0.269	0.187	0.267	0.203	0.287	0.192	0.274	0.209	0.308	0.197	0.271
	720	0.363	0.387	0.366	0.388	0.378	0.401	0.362	0.385	0.397	0.421	0.408	0.403	0.421	0.415	0.417	0.413	0.675	0.587	0.395	0.419
ILI	96	2.728	1.114	3.025	1.195	2.063	0.881	1.319	0.754	2.215	1.081	2.317	0.934	3.228	1.260	2.294	0.945	8.313	2.144	4.539	1.528
	720	2.893	1.132	3.245	1.221	1.979	0.957	1.470	0.788	2.368	1.096	2.027	0.928	2.857	1.157	2.178	0.963	7.283	1.985	5.429	1.661
ECL	96	0.138	0.242	-	-	0.139	0.238	0.129	0.222	0.140	0.237	0.168	0.272	0.193	0.308	0.169	0.273	0.207	0.307	0.131	0.228
	720	0.211	0.305	-	-	0.206	0.297	0.197	0.290	0.203	0.301	0.220	0.320	0.246	0.355	0.222	0.321	0.265	0.360	0.208	0.298
Traffic	96	0.391	0.282	-	-	0.388	0.282	0.360	0.249	0.410	0.282	0.593	0.321	0.587	0.366	0.612	0.338	0.615	0.391	0.375	0.259
	720	0.450	0.310	-	-	0.450	0.312	0.432	0.286	0.466	0.315	0.640	0.350	0.626	0.382	0.653	0.355	0.658	0.407	0.508	0.335

Table 2: Long-term forecasting performance measured using Mean Squared Error (MSE) and Mean Absolute Error (MAE). PatchTST performs the best across most settings, closely followed by MOMENT. We could not run Time-LLM on weather, electricity, and traffic datasets, due to time constraints, and since we could not fit them into a single GPU (see Tab. 21). Complete results in Tab. 11.

Datasets		MOMENT$_{\text{LP}}$ M4	MOMENT$_{\text{LP}}$ FR	GPT4TS M4	GPT4TS FR	TimesNet M4	TimesNet FR	N-BEATS M4	N-BEATS FR	ARIMA	Theta	ETS	Seasonal Naive	Naive	Random Walk
M3	Yearly	**16.74**	**16.97**	18.39	**17.40**	27.48	**16.21**	**16.82**	**15.92**	17.90	16.70	16.47	17.54	17.54	16.77
	Quarterly	**10.09**	10.62	**10.18**	10.29	14.41	12.68	11.26	11.30	10.18	9.19	8.99	11.02	11.45	11.72
	Monthly	16.04	16.90	**15.21**	16.37	**15.58**	16.23	**15.63**	16.37	15.95	14.96	14.41	17.74	18.53	19.19
M4	Yearly	-	**14.84**	-	**14.80**	-	**14.40**	-	14.18	16.19	14.04	14.06	16.33	16.33	14.22
	Quarterly	-	12.02	-	11.77	-	13.21	-	12.25	10.86	10.21	10.24	12.55	11.65	11.46
	Monthly	-	15.80	-	15.36	-	15.67	-	15.24	13.68	13.19	13.58	16.00	15.24	15.48

Table 3: Zero-shot short-horizon forecasting performance on a subset of the M3 and M4 datasets measured using sMAPE. Statistical methods outperformed their deeper counterparts. However, on some datasets (in **bold**), MOMENT, GPT4TS and N-BEATS achieved lower sMAPE than ARIMA.

Time-series modeling with limited supervision. Our benchmark comprises of 5 major time-series modeling tasks of significant practical value, namely long- and short-horizon forecasting, imputation, classification, and anomaly detection, as outlined in Tab. 1. In contrast to TimesNet, we exclusively consider scenarios characterized by limited compute and supervision resources. These scenarios mimic practical situations where training (or fine-tuning) a deep neural network is infeasible due to resource limitations or insufficiently characterized data. Accordingly, we assess MOMENT in zero-shot settings whenever feasible and through linear probing for a few epochs otherwise.

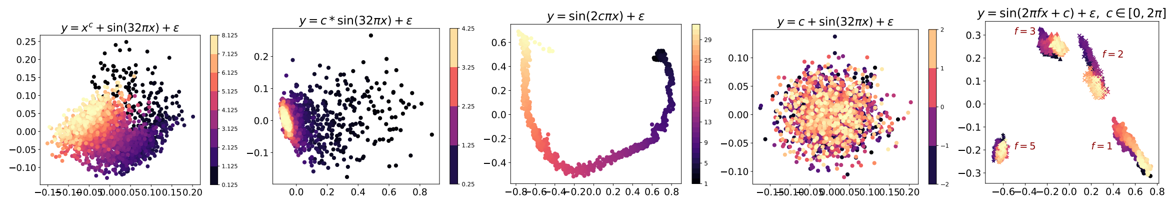
For classification, we consider the unsupervised representation learning problem, where the goal is to learn representations of time-series that are useful for downstream classification, without access to labeled data. As in common in prior work Yue et al. [2022], Franceschi et al. [2019], the quality of representations is measured using the accuracy of a Support Vector Machine trained on them (App. D.2). For short-horizon forecasting, we consider the zero-shot setting introduced by Oreshkin et al. [2021]. In particular, we fine-tune MOMENT on a source dataset using a forecasting head, and evaluate its performance on a target dataset without any fine-tuning (App D.1.2, Tab. 13).

Datasets. We use the same datasets as TimesNet for forecasting and imputation. However, for classification and anomaly detection, we conduct experiments on larger and systematically chosen subset of datasets from the UCR classification archive Dau et al. [2018] and UCR anomaly archive Wu and Keogh [2023]. Specifically, we run classification experiments on all 91 time-series datasets with each time-series shorter than 512 time steps (Tab.15). For anomaly detection, while choosing the subset of time-series, we prioritized coverage over different domains and data sources represented in the UCR anomaly archive (Tab. 14). We also note that the UCR anomaly archive was proposed as an improvement over pre-existing anomaly detection datasets such as the SMD Su et al. [2019], and SMAP Hundman et al. [2018], many of which are also used in TimesNet. Our proposed experimental setup is summarized in Tab. 1 and detailed in App. D.

Metrics. We evaluate each experiment using *multiple* metrics used in task-specific benchmarks, such as MSE and MAE for long-horizon forecasting, and sMAPE for short-horizon forecasting. We also note that TimesNet and GPT4TS Zhou et al. [2023] evaluate anomaly detection performance using vanilla F_1 score which ignores the sequential nature of

	$\texttt{MOMENT}_0$	**GPT4TS**	**TimesNet**	**TS2Vec**	**T-Loss**	**TNC**	**TS-TCC**	**TST**	**CNN**	**Encoder**	**FCN**	**MCNN**	**MLP**	**ResNet**	**t-LeNet**	**TWIESN**	**DTW**
Mean	0.794	0.567	0.573	**0.852**	**0.833**	0.793	0.793	0.659	0.752	0.743	**0.810**	0.702	0.750	**0.826**	0.348	0.727	0.764
Median	0.815	0.583	0.565	**0.871**	**0.849**	0.802	0.802	0.720	0.773	0.753	**0.837**	0.718	0.767	**0.853**	0.333	0.725	0.768
Std.	0.148	0.235	0.238	0.134	0.137	0.176	0.176	0.221	0.180	0.160	0.188	0.195	0.169	0.178	0.222	0.164	0.153

Table 4: **Classification accuracy** of methods across 91 UCR datasets. Methods with mean and median accuracy higher than MOMENT are in **bold**. MOMENT without fine-tuning on individual datasets demonstrates promising accuracy. Complete results in Tab. 15.



(i) Trend *(ii)* Amplitude *(iii)* Frequency *(iv)* Baseline Shift *(v)* Phase

Figure 4: What is MOMENT learning? Principal components of the embeddings of synthetically generated sinusoids suggest that MOMENT can capture subtle trend, scale, frequency, and phase information. In each experiment, c controls the factor of interest, for example the power of the trend polynomial $c \in (\frac{1}{8}, 8)$ Oreshkin et al. [2020] (Fig. 8), and frequency $c \in (1, 32)$ of the generated sine waves (Fig. 8). We generate multiple sine waves by varying c, derive their sequence-level representations using MOMENT, and visualize them in a 2- dimensional space using PCA and t-SNE van der Maaten [2014] in Fig. 4 and Fig. 6.

time-series. Instead, we measure anomaly detection performance with the widely used adjusted best F_1 score Goswami et al. [2023], Challu et al. [2022], and the recently proposed VUS-ROC Paparrizos et al. [2022b].

Baselines. We compare MOMENT with state-of-the-art deep learning and statistical machine learning models across tasks (Tab. 23). This is in contrast to TimesNet which primarily compared with transformer-based approaches. These comparisons are crucial for assessing the practical utility of the proposed methods. We found that statistical and non-transformer-based approaches like ARIMA for short-horizon forecasting, N-BEATS for long-horizon forecasting, and k-nearest neighbors for anomaly detection outperform many deep and transformer-based models.

Hyper-parameter tuning. We do not perform hyper-parameter tuning. In all experiments that follow, unless mentioned otherwise, we fine-tune MOMENT-Large with a batch size of 64, and one cycle learning rate schedule with a peak learning rate between $5e-5$ and $1e-3$ Smith and Topin [2019]. For baseline methods, we capture recommended settings from their papers and public repositories. We report all hyper-parameters settings for MOMENT and baselines in App. D.

Research questions. Through the following experiments we aim to answer 3 broad research questions.

RQ1: Effectiveness. Is MOMENT effective for multiple time-series analysis tasks in limited supervision settings?

RQ2: Interpretability. What is MOMENT learning? Does it capture intuitive time-series characteristics such as varying frequencies, trends, and amplitudes?

RQ3: Properties. What is the impact of the size of scaling model size? Can MOMENT, akin to LLMs, be used for cross-modal transfer learning?

4.1 MOMENT can solve multiple time-series modeling tasks in limited supervision settings

Long-horizon forecasting. Linearly probing MOMENT achieves near state-of-the-art performance on most datasets and horizons, and is only second to PatchTST which generally achieves the lowest MSE (Tab. 2). On many datasets and horizons, forecasting models based on LLMs– TimeLLM and GPT4TS perform worse than MOMENT. Notably, N-BEATS outperforms several recent methods, emphasizing the importance of comparing forecasting performance beyond transformer-based approaches.

Zero-shot short-horizon forecasting. Among all tasks, we found zero-shot short-horizon forecasting to have the largest scope for improvement (Tab. 3). Statistical methods such as Theta and ETS outperformed their deeper counterparts. However, on some datasets, MOMENT achieved lower sMAPE than ARIMA.

Dataset	$\texttt{MOMENT}_0$		$\texttt{MOMENT}_{\texttt{LP}}$		**GPT4TS**		**TimesNet**		**Naive**		**Linear**		**Nearest**		**Cubic**	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	0.082	0.130	0.035	0.075	**0.031**	**0.071**	0.036	0.098	0.119	0.108	0.065	0.067	0.083	0.078	0.601	0.153
ETTh1	0.402	0.403	**0.139**	**0.234**	0.227	0.254	0.175	0.264	1.185	0.658	0.775	0.534	0.900	0.579	2.178	0.916
ETTh2	0.125	0.238	**0.061**	**0.159**	0.109	0.213	0.170	0.286	0.225	0.304	0.135	0.234	0.166	0.252	1.920	0.641
ETTm1	0.202	0.288	**0.074**	**0.168**	0.076	0.146	0.087	0.198	0.455	0.365	0.165	0.229	0.230	0.260	0.858	0.494
ETTm2	0.078	0.184	**0.031**	**0.108**	0.052	0.133	0.112	0.220	0.113	0.191	0.062	0.138	0.079	0.152	0.534	0.356
Electricity	0.250	0.371	0.094	0.211	**0.072**	**0.183**	0.124	0.248	1.474	0.869	0.737	0.592	0.923	0.629	2.257	0.888

Table 6: **Imputation Results.** MOMENT with linear probing achieved the lowest reconstruction error on all ETT datasets. In the zero-shot setting, MOMENT consistently outperformed all statistical interpolation methods with the exception of linear interpolation. Complete results in Tab. 20.

Metric		$\texttt{MOMENT}_0$	$\texttt{MOMENT}_{\texttt{LP}}$	**GPT4TS**	**TimesNet**	**Anomaly Transformer**	**DGHL**	**k-NN**
Adj. F_1	Mean	0.636	**0.679**	0.444	0.562	0.463	0.412	0.580
	Median	0.704	**0.842**	0.314	0.529	0.400	0.340	0.670
	Std.	0.352	0.338	0.366	0.366	0.394	0.334	0.377
VUS ROC	Mean	0.683	0.715	0.612	0.703	0.664	0.678	**0.726**
	Median	0.701	0.724	0.604	0.710	0.681	0.690	**0.736**
	Std.	0.129	0.125	0.123	0.126	0.125	0.141	0.110

Table 7: Anomaly detection performance averaged over 44 time-series from the UCR Anomaly Archive. $\texttt{MOMENT}_{\texttt{LP}}$ achieves near state-of-the-art anomaly detection results. Complete results in Tab. 14.

Classification. Without any data-specific fine-tuning, MOMENT can learn distinct representations for different classes of data (Fig. 5a), and an SVM trained on its representations, performs better than all but 4 methods specifically built for time-series classification models and trained on each individual dataset. Recently proposed GPT4TS and TimesNet perform poorly despite being trained on each individual dataset with labels.

Model	Bit Memory	MNIST	CIFAR-10	IMDb
GPT-2	**1.000**	0.975	**0.711**	0.867
Flan-T5	**1.000**	**0.987**	0.672	0.861
MOMENT	**1.000**	0.982	0.620	**0.872**

Table 5: **Cross-modal transfer experiments.** Accuracy measured on the test set, from the checkpoint with the lowest train loss. Even with frozen self-attention and feed-forward layers, MOMENT is able to model cross-modal sequences on par with GPT-2 and Flan-T5 models of similar scale.

Anomaly detection. On 44 time-series from the UCR anomaly detection archive, MOMENT consistently outperformed both TimesNet and GPT4TS, as well as 2 state-of-the-art deep learning models tailored for anomaly detection, in both zero-shot and linear probing configurations. However, k-nearest neighbors performed marginally better in terms of VUS-ROC score, but had a lower adjusted best F_1 score

Imputation. Tab. 6 contains imputation performance of all models averaged over 4 different masking rates. MOMENT with linear probing achieved the lowest reconstruction error on all ETT datasets. In the zero-shot setting, MOMENT consistently outperformed all statistical interpolation methods with the exception of linear interpolation.

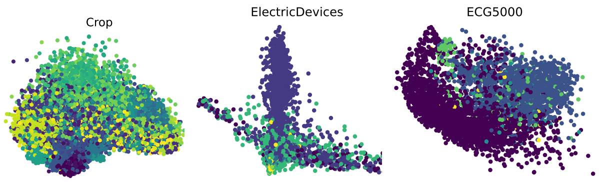
4.2 What is MOMENT Learning?

We found that MOMENT can capture changes in intuitive time-series characteristics such as trend, amplitude, frequencies, and phases of time-series. However, it cannot differentiate between vertically shifted time-series as it normalizes each signal prior to modeling (Fig. 4,6). Furthermore, on many classification datasets, MOMENT learns distinct representations of different classes, even in a zero-shot setting without access to labels (Fig. 5a, 7).

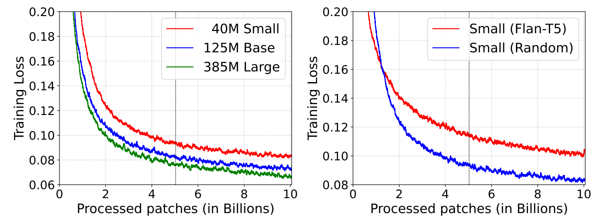
4.3 Properties of Large Time-series Models

Model scaling improves training loss. Like LLMs, we found that increasing the size of the model leads to lower training loss, even before the first epoch (Fig. 5b, left). An immediate next step is to assess how effectively this phenomenon extends to time-series modeling tasks under limited supervision.

MOMENT **can solve cross-modal sequence learning tasks.** Lu et al. [2022] first showed that large pre-trained language and vision transformers can solve general sequence learning tasks for modalities outside of text and images with minimal fine-tuning. Several recent studies have leveraged these properties to reprogram LLMs for time-series tasks.



(a) PCA and t-SNE visualizations of representations learned by `MOMENT` on the 3 largest UCR datasets. Different colors represent different classes. Even without dataset-specific fine-tuning, `MOMENT` learns distinct representations for different classes.



(b) **Training losses (MSE).** A dashed vertical line denotes the first epoch. All models were trained with a batch size of 131072 patches. (*left*) Larger models obtain lower training loss. *right* Eventually, randomly initialized `MOMENT-small` outperform the same model initialized with Flan-T5 weights.

We explore whether transformers pre-trained on time-series can also be used to solve sequence classification tasks on image, text, and binary data. Our results confirm that by freezing the self-attention and feed-forward layers, `MOMENT` can model sequences comparable to GPT-2 and Flan-T5 models of similar scale (Tab. 5).

`MOMENT` with randomly initialized weights converges to a lower training loss. Our observations suggest that with sufficient data, pre-training our model from scratch results in a lower training loss than continually pre-training a model of similar size initialized with language modeling weights (Fig. 5b, 11). This also underscores that there is sufficient publicly accessible pre-training data available in the Time-series Pile to facilitate pre-training time-series foundation models from scratch.

5 Conclusion and Future Work

We release the first open-source family of time-series foundation models and make contributions at all stages of the development and evaluation process. We first compile a large and diverse collection of public time-series, called the Time-series Pile, and demonstrate its efficacy by pre-training high-performing time-series foundation models from scratch. Then, we systematically address several time-series-specific challenges, which have hitherto hindered extensive exploration of large-scale multi-dataset pre-training. We use the Time-series Pile and these strategies to pre-train transformer models of three different sizes. Finally, we design an experimental benchmark to evaluate time-series foundation models on multiple practical time-series tasks, particularly focusing on scenarios with constrained compute and supervision, building on prior work by Wu et al. [2023]. Using this benchmark, we show that `MOMENT` is effective for the considered tasks with minimal fine-tuning. `MOMENT`'s superior performance, especially on anomaly detection and classification problems which typically have small datasets, can be attributed to pre-training. Moreover, we demonstrate that across many tasks, smaller statistical and shallower deep learning methods perform reasonably well. Lastly, we make several interesting empirical observations about time-series foundation models. Our overarching goal is to push the boundaries of open science by publicly releasing the Time-series Pile, along with code, model weights, and training logs.

We note several interesting directions of future work, including the application of `MOMENT` to real-world challenges, investigating multi-modal time-series and text foundation models Cai et al. [2023], and enhancing forecasting performance by pre-training `MOMENT` using causal attention and forecasting objectives.

Acknowledgments

Discussions. We would like to express our sincerest gratitude to Barış Kurt, Andrey Kan, Laurent Callot, Gauthier Guinet, Jingchao Ni, and Jonas M. Kübler for insightful discussions regarding the problem setting and experimental design. Their unwavering support was instrumental in the development of `MOMENT`. We are also thankful to Laurent, Barış, Jingchao and Andrey for their constructive feedback on the writing of this manuscript. Additionally, we acknowledge the insightful exchanges with Yuyang (Bernie) Wang, Abdul Fatir Ansari, Ingo Guering, Xiyuan Zhang, and Anoop Deoras. Special thanks to Cherie Ho for suggesting a creative and befitting name for our model.

Data. We extend our gratitude to the authors and data curators whose meticulous efforts were instrumental in curating the datasets utilized for both pre-training and evaluation purposes: UCR Time-series Classification Archive Dau et al. [2018], TSB-UAD Anomaly Benchmark Paparrizos et al. [2022a], Monash Forecasting Archive Godahewa et al. [2021], and the long-horizon forecasting datasets Zhou et al. [2021].

Software and Models. Our training and evaluation library was inspired from `Time-Series-Library`. We would also like to thank the authors of the following libraries for their implementations: `universal-computation`, `Anomaly-Transformer`, `VUS`, `tsad-model-selection`, `One-Fits-All` and `Statsforecast`.

Reproducibility statement

All models were trained and evaluated on a computing cluster consisting of 128 AMD EPYC 7502 CPUs, 503 GB of RAM, and 8 NVIDIA RTX A6000 GPUs each with 49 GiB RAM. All `MOMENT` variants were trained on a single A6000 GPU (with any data or model parallelism). We will release all our model artifacts (`MOMENT-small`, `MOMENT-base`, and `MOMENT-large`) upon acceptance. We have made the code to compile `TILE`, pre-train and fine-tune `MOMENT`, and reproduce our results anonymously available at `https://anonymous.4open.science/r/BETT-773F/README.md`. We enlist an exhaustive list of hyper-parameters in App. D to aid reproducibility. We would like to emphasize that all datasets used in this study are publicly available.

Impact statement

Transparency Index. Given the exponential rise in societal reliance on large foundation models, ensuring transparency in their training approach, architecture, and downstream application is crucial for public accountability, scientific advancement, and effective governance. o uphold this objective, we publicly release our training code base, data sources, and evaluation pipeline. We assess the transparency of *MOMENT* using the criteria outlined by Bommasani et al. [2023], focusing on upstream resources utilized during training and model description, encompassing 32 and 33 transparency indicators, respectively. We report expected upstream and model transparency scores for `MOMENT` in Tab. 22. Notably, `MOMENT` is *expected* to have one of the highest levels of upstream transparency. However, it's model transparency scores are lower, primarily due to comprehensive (external and third-party) harm and trustworthiness evaluations, which are not well understood in the context of time-series modeling.

Environmental Impact. We train multiple models over many days resulting in significant energy usage and a sizeable carbon footprint. However, we hope that releasing our models will ensure that future time-series modeling efforts are quicker and more efficient, resulting in lower carbon emissions.

We follow prior work Bender et al. [2021], Patterson et al. [2021], Touvron et al. [2023], Wu et al. [2022], Dodge et al. [2022] and estimate the carbon footprint of pre-training all variants of `MOMENT` based on the GPU device used and the carbon efficiency of the electricity grid. Our estimated $\texttt{CO}_2$ generation estimates are shown in Tab. 8.

Model Variant	# Parameters (M)	GPU Hours	Power Consumption (W)	Carbon Emission (tCO2eq)
Small	40	308.378	300	31.136
Base	125	308.306	300	31.129
Large	385	404.389	300	40.831
Upper Bound Total	-	1021.073	300	103.096
Actual Total	-	712.767	300	71.967

Table 8: Total carbon emission induced upon training the `MOMENT` family of models. `MOMENT-small` and `MOMENT-base` were trained simultaneously on a single GPU, thus the TGP required for each model would likely be much less than 300W, and the total time for both models combined is equal to the maximum of the time required for each model. Actual total power consumption and carbon emission values account for this.

We use the Total Graphics Power (TGP) to calculate the total power consumed for training `MOMENT` models, although the total power consumed by the GPU will likely vary a little based on the GPU utilization while training our model. Our calculations do not account for power demands from other sources of our compute. We use 336.566 Kg $C0_2$/MWH as the standard value of CO_2 emission per megawatt hour of energy consumed for Pittsburgh[10].

We share an upper limit of the individual CO_2 emission for each model, as well as a more realistic actual estimate for the carbon emissions from `MOMENT-small` and `MOMENT-base`, since they were trained simultaneously on a single Nvidia RTX A6000 GPU, and thus the power consumed by the GPU was shared for the training of both variants. `MOMENT-large` was trained independently on a single RTX A6000 GPU, and thus the carbon emissions for its pre-training are decidedly more realistic.

[10] `https://emissionsindex.org/`

Ethical considerations and potential misuse. Despite MOMENT's promising performance in limited-data settings, it is important to use its predictions with care, especially in high-stakes settings such as healthcare. Before MOMENT is used for high-stakes decision-making, we recommend fine-tuning and evaluating the model with task-specific in-domain data.

References

Stephen H Schneider and Robert E Dickinson. Climate modeling. *Reviews of Geophysics*, 12(3):447–493, 1974.

Mononito Goswami, Benedikt Boecking, and Artur Dubrawski. Weak supervision for affordable modeling of electrocardiogram data. In *AMIA Annual Symposium Proceedings*, volume 2021, page 536. American Medical Informatics Association, 2021.

Haowen Xu, Wenxiao Chen, Nengwen Zhao, Zeyan Li, Jiahao Bu, Zhihan Li, Ying Liu, Youjian Zhao, Dan Pei, Yang Feng, et al. Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications. In *Proceedings of the 2018 world wide web conference*, pages 187–196, 2018.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023a.

Kastan Day, Daniel Christl, Rohan Salvi, and Pranav Sriram. Video pre-trained transformer: A multimodal mixture of pre-trained experts, 2023.

Tian Zhou, Peisong Niu, Xue Wang, Liang Sun, and Rong Jin. One fits all: Power general time series analysis by pretrained LM. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=gMS6FVZvmF.

Nate Gruver, Marc Anton Finzi, Shikai Qiu, and Andrew Gordon Wilson. Large language models are zero-shot time series forecasters. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=md68e8iZK1.

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. Time-llm: Time series forecasting by reprogramming large language models, 2023.

Vijay Ekambaram, Arindam Jati, Nam H. Nguyen, Pankaj Dayama, Chandra Reddy, Wesley M. Gifford, and Jayant Kalagnanam. Tiny time mixers (ttms): Fast pre-trained models for enhanced zero/few-shot forecasting of multivariate time series, 2024.

Defu Cao, Furong Jia, Sercan O Arik, Tomas Pfister, Yixiang Zheng, Wen Ye, and Yan Liu. Tempo: Prompt-based generative pre-trained transformer for time series forecasting, 2023.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ju_Uqw384Oq.

Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. Meta-learning framework with applications to zero-shot time-series forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(10):9242–9250, May 2021. doi: 10.1609/aaai.v35i10.17115. URL https://ojs.aaai.org/index.php/AAAI/article/view/17115.

Jyoti Narwariya, Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and T. V. Vishnu. Meta-learning for few-shot time series classification. In *Proceedings of the 7th ACM IKDD CoDS and 25th COMAD*, CoDS COMAD 2020, page 28–36, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450377386. doi: 10.1145/3371158.3371162. URL https://doi.org/10.1145/3371158.3371162.

Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. Transformers in time series: A survey. In Edith Elkind, editor, *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pages 6778–6786. International Joint Conferences on Artificial Intelligence Organization, 8 2023. doi: 10.24963/ijcai.2023/759. URL https://doi.org/10.24963/ijcai.2023/759. Survey Track.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Jbdc0vTOcol.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. Simmim: a simple framework for masked image modeling. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9643–9653, 2022. doi: 10.1109/CVPR52688.2022.00943.

Y. Li, H. Fan, R. Hu, C. Feichtenhofer, and K. He. Scaling language-image pre-training via masking. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 23390–23400, Los Alamitos, CA, USA, jun 2023b. IEEE Computer Society. doi: 10.1109/CVPR52729.2023.02240. URL https://doi.ieeecomputersociety.org/10.1109/CVPR52729.2023.02240.

Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. Ts2vec: Towards universal representation of time series. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36 (8):8980–8987, Jun. 2022. doi: 10.1609/aaai.v36i8.20881. URL https://ojs.aaai.org/index.php/AAAI/article/view/20881.

Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee Keong Kwoh, Xiaoli Li, and Cuntai Guan. Time-series representation learning via temporal and contextual contrasting. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 2352–2359. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/324. URL https://doi.org/10.24963/ijcai.2021/324. Main Track.

Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. Unsupervised scalable representation learning for multivariate time series. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/53c6de78244e9f528eb3e1cda69699bb-Paper.pdf.

George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. A transformer-based framework for multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 2114–2124, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383325. doi: 10.1145/3447548.3467401. URL https://doi.org/10.1145/3447548.3467401.

Jiaxiang Dong, Haixu Wu, Haoran Zhang, Li Zhang, Jianmin Wang, and Mingsheng Long. Simmtm: A simple pre-training framework for masked time-series modeling. In *Advances in Neural Information Processing Systems*, 2023.

Zhe Li, Zhongwen Rao, Lujia Pan, Pengyun Wang, and Zenglin Xu. Ti-mae: Self-supervised masked time series autoencoders, 2023c.

Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. Frozen pretrained transformers as universal computation engines. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(7):7628–7636, Jun. 2022. doi: 10.1609/aaai.v36i7.20729. URL https://ojs.aaai.org/index.php/AAAI/article/view/20729.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The pile: An 800gb dataset of diverse text for language modeling, 2020.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Qianli Ma, Zhen Liu, Zhenjing Zheng, Ziyang Huang, Siying Zhu, Zhongzhong Yu, and James T. Kwok. A survey on time-series pre-trained models, 2023.

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. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12):11106–11115, May 2021. doi: 10.1609/aaai.v35i12.17325. URL https://ojs.aaai.org/index.php/AAAI/article/view/17325.

Cristian Challu, Kin G. Olivares, Boris N. Oreshkin, Federico Garza Ramirez, Max Mergenthaler Canseco, and Artur Dubrawski. NHITS: Neural hierarchical interpolation for time series forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(6):6989–6997, Jun. 2023. doi: 10.1609/aaai.v37i6.25854. URL https://ojs.aaai.org/index.php/AAAI/article/view/25854.

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

California Department of Transportation. Performance measurement system (pems), 2024. URL http://pems.dot.ca.gov/. Accessed: 2024-02-01.

Max Planck Institute for Biogeochemistry. Weather data, 2024. URL https://www.bgc-jena.mpg.de/wetter/. Accessed: 2024-02-01.

Centers for Disease Control and Prevention. Fluview: Flu activity & surveillance, 2024. URL https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html. Accessed: 2024-02-01.

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 & Development in Information Retrieval*, SIGIR '18, page 95–104, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356572. doi: 10.1145/3209978.3210006. URL https://doi.org/10.1145/3209978.3210006.

Rakshitha Wathsadini Godahewa, Christoph Bergmeir, Geoffrey I. Webb, Rob Hyndman, and Pablo Montero-Manso. Monash time series forecasting archive. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=wEc1mgAjU-.

Hoang Anh Dau, Eamonn Keogh, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, Yanping, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, Gustavo Batista, and Hexagon-ML. The ucr time series classification archive, October 2018. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.

Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963, 2019.

John Paparrizos, Yuhao Kang, Paul Boniol, Ruey S. Tsay, Themis Palpanas, and Michael J. Franklin. Tsb-uad: An end-to-end benchmark suite for univariate time-series anomaly detection. *Proc. VLDB Endow.*, 15(8):1697–1711, apr 2022a. ISSN 2150-8097. doi: 10.14778/3529337.3529354. URL https://doi.org/10.14778/3529337.3529354.

Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=cGDAkQo1C0p.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/radford21a.html`.

R. Wu and E. J. Keogh. Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress. *IEEE Transactions on Knowledge & Data Engineering*, 35(03):2421–2429, mar 2023. ISSN 1558-2191. doi: 10.1109/TKDE.2021.3112126.

Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2828–2837, 2019.

Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 387–395, 2018.

Mononito Goswami, Cristian Ignacio Challu, Laurent Callot, Lenon Minorics, and Andrey Kan. Unsupervised model selection for time series anomaly detection. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=gOZ_pKANaPW`.

Cristian I. Challu, Peihong Jiang, Ying Nian Wu, and Laurent Callot. Deep generative model with hierarchical latent factors for time series anomaly detection. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 1643–1654. PMLR, 28–30 Mar 2022. URL `https://proceedings.mlr.press/v151/challu22a.html`.

John Paparrizos, Paul Boniol, Themis Palpanas, Ruey S. Tsay, Aaron Elmore, and Michael J. Franklin. Volume under the surface: A new accuracy evaluation measure for time-series anomaly detection. *Proc. VLDB Endow.*, 15(11): 2774–2787, jul 2022b. ISSN 2150-8097. doi: 10.14778/3551793.3551830. URL `https://doi.org/10.14778/3551793.3551830`.

Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, volume 11006, pages 369–386. SPIE, 2019.

Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=r1ecqn4YwB`.

Laurens van der Maaten. Accelerating t-sne using tree-based algorithms. *Journal of Machine Learning Research*, 15 (93):3221–3245, 2014. URL `http://jmlr.org/papers/v15/vandermaaten14a.html`.

Yifu Cai, Mononito Goswami, Arjun Choudhry, Arvind Srinivasan, and Artur Dubrawski. Jolt: Jointly learned representations of language and time-series. In *Deep Generative Models for Health Workshop NeurIPS 2023*, 2023.

Rishi Bommasani, Kevin Klyman, Shayne Longpre, Sayash Kapoor, Nestor Maslej, Betty Xiong, Daniel Zhang, and Percy Liang. The foundation model transparency index, 2023.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383097. doi: 10.1145/3442188.3445922. URL `https://doi.org/10.1145/3442188.3445922`.

David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training, 2021.

Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga Behram, James Huang, Charles Bai, Michael Gschwind, Anurag Gupta, Myle Ott, Anastasia Melnikov, Salvatore Candido, David Brooks, Geeta Chauhan, Benjamin Lee, Hsien-Hsin S. Lee, Bugra Akyildiz, Maximilian Balandat, Joe Spisak, Ravi Jain, Mike Rabbat, and Kim Hazelwood. Sustainable ai: Environmental implications, challenges and opportunities, 2022.

Jesse Dodge, Taylor Prewitt, Remi Tachet des Combes, Erika Odmark, Roy Schwartz, Emma Strubell, Alexandra Sasha Luccioni, Noah A. Smith, Nicole DeCario, and Will Buchanan. Measuring the carbon intensity of ai in cloud instances. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22, page 1877–1894, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393522. doi: 10.1145/3531146.3533234. URL `https://doi.org/10.1145/3531146.3533234`.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=I55UqU-M11y.

Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *Proc. 39th International Conference on Machine Learning (ICML 2022)*, 2022.

Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X. Liu, and Schahram Dustdar. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=0EXmFzUn5I.

Jiehui Xu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Anomaly transformer: Time series anomaly detection with association discrepancy. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=LzQQ89U1qm_.

Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEit: BERT pre-training of image transformers. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=p-BhZSz59o4.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Sana Tonekaboni, Danny Eytan, and Anna Goldenberg. Unsupervised representation learning for time series with temporal neighborhood coding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=8qDwejCuCN.

Mariusz Zebik, Marcin Korytkowski, Rafal Angryk, and Rafał Scherer. *Convolutional Neural Networks for Time Series Classification*, pages 635–642. Springer International Publishing, Cham, 2017. ISBN 978-3-319-59060-8. doi: 10.1007/978-3-319-59060-8_57. URL https://doi.org/10.1007/978-3-319-59060-8_57.

Joan Serrà, Santiago Pascual, and Alexandros Karatzoglou. Towards a universal neural network encoder for time series. In *International Conference of the Catalan Association for Artificial Intelligence*, 2018. URL https://api.semanticscholar.org/CorpusID:13675490.

Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 1578–1585, 2017. doi: 10.1109/IJCNN.2017.7966039.

Zhicheng Cui, Wenlin Chen, and Yixin Chen. Multi-scale convolutional neural networks for time series classification, 2016.

Arthur Le Guennec, Simon Malinowski, and Romain Tavenard. Data Augmentation for Time Series Classification using Convolutional Neural Networks. In *ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data*, Riva Del Garda, Italy, September 2016. URL https://shs.hal.science/halshs-01357973.

Pattreeya Tanisaro and Gunther Heidemann. Time series classification using time warping invariant echo state networks. In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 831–836, 2016. doi: 10.1109/ICMLA.2016.0149.

Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. Anomaly detection in time series: A comprehensive evaluation. *Proc. VLDB Endow.*, 15(9):1779–1797, may 2022. ISSN 2150-8097. doi: 10.14778/3538598.3538602. URL https://doi.org/10.14778/3538598.3538602.

Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, page 427–438, New York, NY, USA, 2000. Association for Computing Machinery. ISBN 1581132174. doi: 10.1145/342009.335437. URL https://doi.org/10.1145/342009.335437.

A Related Work

Transformers and Patching for Time-series Modeling. There is a growing body of work utilizing transformers for various time-series analysis tasks, for example PatchTST Nie et al. [2023], Informer Zhou et al. [2021], Autoformer Wu et al. [2021], FEDformer Zhou et al. [2022], Pyraformer Liu et al. [2022] for forecasting; Anomaly Transformer Xu et al. [2022] for anomaly detection, and TST Zerveas et al. [2021], TS-TCC Eldele et al. [2021] for representation learning.

One issue with applying transformers to time-series data is the complexity of the self-attention mechanism, which grows quadratically with the size of input tokens (or length of time-series). Consequently, the primary focus of most initial applications of transformers to time-series, especially for forecasting where longer look-back windows typically improve performance, was to redesign the self-attention mechanism to reduce its complexity Zhou et al. [2021, 2022], Liu et al. [2022]. Nie et al. [2023] demonstrated that treating time-series sub-sequences (or patches) as tokens instead of individual time points is a simple, efficient yet effective mechanism for learning useful representations for forecasting. The authors drew inspiration from language and vision domains where sub-words (vs. characters) Devlin et al. [2019] and 2-D patches (vs. raw pixels) Bao et al. [2022], Dosovitskiy et al. [2021] are used as inputs to transformers. Drawing inspiration from prior work, we build on top of the transformer architecture which takes disjoint time-series sub-sequences (or patches) as input.

Masked Representation Learning. Masked pre-training is a widely-used self-supervised learning task where a model learns to accurately reconstruct masked portions of its input. Masked language Devlin et al. [2019], Raffel et al. [2020] and image modeling Xie et al. [2022], Li et al. [2023b] have been successfully utilized to learn models from vast quantities of unlabeled data, which can generalize to a variety of downstream tasks.

For time-series data, prior work has primarily focused on contrastive representation learning Yue et al. [2022], Eldele et al. [2021], Franceschi et al. [2019]. The goal of contrastive learning is to learn a representation space where "positive" pairs of time-series are close while "negative" pairs are far apart. However, the notion of positive and negative pairs is subjective and data-dependent, and popular transformations such as flipping and cropping invariance may not be appropriate for time-series data Yue et al. [2022]. In contrast, some studies mask portions of time-series using zeros and learn a model to reconstruct them Nie et al. [2023], Zerveas et al. [2021], Dong et al. [2023], Li et al. [2023c].

Representation learning via masking is well-suited to all the downstream tasks we care about, especially forecasting and imputation, as they are instances of the masked reconstruction problem. Owing to its simplicity and success in vision and language domains, we use the masked prediction task to pre-train our model, using a special embedding (see `[MASK]` in Fig. 3) to mask time-series patches instead of zeros.

Cross-modal transfer learning using language models. Lu et al. [2022] had first shown that transformers pre-trained on text data (LLMs) can effectively solve sequence modeling tasks in other modalities. Some recent studies have leveraged this inherent ability of language pre-trained transformers to "reprogram" LLMs for time-series analysis using parameter efficient fine-tuning and suitable tokenization strategies Zhou et al. [2023], Gruver et al. [2023], Jin et al. [2023], Cao et al. [2023], Ekambaram et al. [2024]. However, some of these models Jin et al. [2023], Gruver et al. [2023] with billions of parameters demand significant memory and computational resources to perform well. We complement this line of research with three empirical observations (Sec 4.3): we show that (1) transformers trained on time-series can also model sequences across modalities, (2) during pre-training, randomly initializing weights lead to lower pre-training loss, than initializing with language modeling weights, and (3) models pre-trained on time-series outperform LLM-based models such as Zhou et al. [2023], Jin et al. [2023] on many tasks and datasets.

Unanswered Questions. To the best of our knowledge, two questions remain largely unanswered in prior work on time-series modeling. First, all existing time-series models are (pre-)trained and fine-tuned on individual datasets Nie et al. [2023], Yue et al. [2022], Wu et al. [2023], Zhou et al. [2023], and the benefits (or drawbacks) of large-scale multi-dataset pre-training remains unexplored Wen et al. [2023]. Second, there is very limited work on time-series modeling in limited supervision settings, such as zero-shot forecasting Oreshkin et al. [2021], or few-shot classification Narwariya et al. [2020]. In our work, we consider both these questions and *show that pre-training a model of sufficient capacity on a large corpus of unlabeled time-series data can in fact enable it to provide reasonably accurate predictions in limited-supervision settings.*

B Interesting directions for future work

We note some interesting directions of future work:

- Study the impact of design choices such as the impact of the choice of the loss function (Huber, L_1, L_2), patch length $(4, 8)$, and masking percentage $(0.3, 0.6)$ on pre-training loss and time-series modeling performance.

Task	Dataset	Channels	Series Length	**Data Size** (Train, Val, Test)	**Information** (Frequency/Number of Classes)
Long horizon forecasting (Informer)	ETTm1, ETTm2	7	{96, 720}	(33953, 11425, 11425)	Electricity (15 mins)
	ETTh1, ETTh2	7		(8033, 2785, 2785)	Electricity (15 mins)
	Electricity	321		(17805, 2537, 5165)	Electricity (Hourly)
	Traffic	862		(11673, 1661, 3413)	Transportation (Hourly)
	Weather	21		(36280, 5175, 10444)	Weather (10 mins)
	Exchange	8		(4704, 665, 1422)	Exchange rate (Daily)
	ILI	7	{24, 60}	(69, 2, 98)	Illness (Weekly)
Short horizon forecasting (Monash)	M4-Yearly	1	6	(16099, 2301, 4600)	-
	M4-Quarterly		8	(16800, 2400, 4800)	-
	M4-Monthly		18	(33600, 4800, 9600)	-
	M3-Yearly		6	(451, 65, 129)	-
	M3-Quarterly		8	(529, 76, 151)	-
	M3-Monthly		18	(999, 144, 285)	-
Imputation (Informer)	ETTm1, ETTm2	7	512	(33953, 11425, 11425)	Electricity (15 mins)
	ETTh1, ETTh2	7		(8033, 2785, 2785)	Electricity (15 mins)
	Electricity	321		(17805, 2537, 5165)	Electricity (Hourly)
	Weather	21		(36280, 5175, 10444)	Weather (10 mins)
Classification (UCR)	UWaveGestureLibraryX	1	315	(640, 256, 3582)	Motion Gesture (8 classes)
	ECG5000		140	(357, 143, 4500)	ECG Record (5 classes)
	OSULeaf		427	(142, 58, 242)	Leaf Outlines (6 classes)
	MedicalImages		99	(272, 109, 760)	Pixel Intensity (10 classes)
	Ham		431	(77, 32, 105)	Food spectrographs (2 classes)
Anomaly detection (TSB-UAD)	1sddb40	1	-	(24489, 9489, 3969)	Beats
	BIDMC1		-	(1274, 204, 7988)	PVC
	CIMIS44AirTemperature3		-	(2346, 632, 3672)	Weather Data
	CIMIS44AirTemperature5		-	(2346, 632, 3672)	Weather Data
	ECG2		-	(10203, 3775, 14488)	ECG2 Lead

Table 9: **The Time-series Pile.** A brief description of datasets that collectively make the Time-series Pile. Due to space constraints, we only include metadata for the subsets of the M3 and M4 datasets in our experiments, as well as 5 classification and anomaly detection datasets. Characteristics of all short-horizon forecasting, classification and anomaly detection datasets in the Time-series Pile can be found in our official repository, and Monash archive, UCR/UEA classification archive, and TSB-UAD anomaly benchmark, respectively.

- Pre-training data. Two interesting directions include using augmentation and synthetic data to improve the quality of pre-training, and looking at tuning dataset mixtures in the Time-series Pile.

C The Time-series Pile

D Experimental Setup and Results

Through our experiments, our goal is to answer the following research questions.

Is `MOMENT` effective for multiple time-series analysis tasks in limited and rich supervision settings? We conduct large-scale experiments on widely used benchmarks to evaluate `MOMENT` on forecasting, classification, anomaly detection, and imputation as outlined in Table 8. The *limited supervision* setting mimics practical scenarios in which it is infeasible to train (or fine-tune) a deep neural network due to limited compute and, little or inadequately characterized data. In these settings, `MOMENT` provides predictions without any explicit (re)training on target data[11]. On the other hand, the rich supervision setting allows us to examine whether `MOMENT` can utilize task-specific data to improve its performance via end-to-end fine-tuning or linear probing.

What does `MOMENT` learn? We evaluated `MOMENT`'s ability to model time-series characteristics such as varying frequencies, trends, and scales. Structure in the PCA and t-SNE (Fig. 9) visualizations of the embeddings of synthetically generated sinusoids suggest that MOMENT can capture subtle trend, scale, frequency, and auto-correlation information. ϵ denotes gaussian noise with 0 mean and 0.1 standard deviation. c controls the factor of interest, i.e. the power of the trend polynomial, amplitude, and frequency of the sine waves in experiments (i), (ii) & (iii), respectively.

[11]For classification, the quality of `MOMENT`'s representations is measured using the accuracy of a Support Vector Machine trained on them, as is common in prior work on unsupervised representation learning Yue et al. [2022], Franceschi et al. [2019]. However, unlike prior work, `MOMENT` embeds time-series without any data-specific training.

Hyper-parameter Tuning. We do not perform extensive hyper-parameter tuning. In all experiments that follow, unless mentioned otherwise, we fine-tune `MOMENT-Base` with a batch size of 16, and cosine learning rate schedule with an initial learning rate of $1e^{-5}$. For baseline methods, we capture recommended settings from their respective papers and public repositories. We report all hyper-parameters settings for `MOMENT` and baselines in Appendix D.

D.1 Forecasting

Task description. Given a time-series $\mathcal{T} = [x_1, ..., x_L]$ where $x_i \in \mathbb{R}$, the univariate forecasting problem is to predict the next H time-steps $[x_{L+1}, ..., x_{L+H}]$. Depending on the length of the horizon, forecasting can be categorized as *short* or *long-horizon*[12]. We consider both tasks in our experiments. We propose two configurations of `MOMENT` for the forecasting problem: (1) we can produce short-horizon forecasts without any explicit training or fine-tuning, by appending masked patches and predicting them using the default reconstruction head (Fig. 4 (ii)); (2) alternatively, we can replace the reconstruction head to a forecasting head and then fine-tune it (Fig. 4 (i)).

D.1.1 Long-Horizon Forecasting

Datasets. We use all the long-horizon forecasting datasets (Sec 3.1). But to speed up our experiments, we drop all exogenous variables from multi-variate datasets and only consider the target time-series for forecasting.

Baselines. We compare our methods with various transformer-based and deep learning baselines. These models can be found in Table 11. For Time-LLM we could not run experiments on Weather, electricity, and traffic datasets, due to time constraints, and since we could not fit them into a single GPU.

Experimental Setting. We train all models with a look-back window of length $L = 512$ to forecast $T = 24$, 60 time-steps for the `ILI` dataset and $T = 96$, 720 for the rest. We evaluate the Mean Squared Error (MSE) and Mean Absolute Error (MAE) as metrics.

Hyperparameters. The hyperparameters used for training all models in our long-horizon forecasting experiments are shown in Table 10.

Model	Hyper-parameters	
`MOMENT`	`sequence length` :	512
	`patch length` :	8
	`patch stride length` :	8
	`initial learning rate` :	0.0001
	`forecast horizon` :	$\{96, 720\}$
`Time-LLM`	`sequence length` :	512
	`patch length` :	16
	`patch stride length` :	8
	`initial learning rate` :	0.001
	`dimension of feedforward layer` :	2048
	`llm layers` :	32
	`number of heads` :	8
`N-BEATS`	`sequence length` :	512
	`stack types` :	{trend, seasonality}
	`number of blocks per stack` :	3
	`thetas dimensions` :	$\{4, 8\}$
	`hidden layer units` :	256

Table 10: Hyper-parameter values for long-horizon forecasting models.

D.1.2 Zero-shot Short-Horizon Forecasting

Datasets. To evaluate zero-shot forecasting performance, we conduct experiments on the M3 and M4 datasets (Sec. 3.1).

[12]There distinction between long and short-horizon forecasting is rather arbitrary. For instance, most of the default forecasting horizons for the long-horizon forecasting benchmark Influenza-like Illness (24, 36, 48, 60) are shorter than the Hourly subset of the M4 dataset, a popular short-horizon forecasting benchmark.

Methods		MOMENT$_{LP}$		Time-LLM		GPT4TS		PatchTST		DLinear		TimesNet		FEDformer		Pyraformer		Autoformer		Stationary		ETSformer		LightTS		Informer		Reformer		LogTrans		N-BEATS	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	0.154	0.209	-	-	0.162	0.212	0.149	0.198	0.176	0.237	0.172	0.220	0.217	0.296	0.896	0.556	0.266	0.336	0.173	0.223	0.197	0.281	0.182	0.242	0.300	0.384	0.689	0.596	0.458	0.490	0.152	0.210
	720	0.315	0.336	-	-	0.326	0.337	0.314	0.334	0.333	0.362	0.365	0.359	0.403	0.428	1.004	0.934	0.419	0.428	0.414	0.410	0.352	0.288	0.352	0.386	1.059	0.741	1.130	0.792	0.869	0.675	0.331	0.359
ETTh1	96	0.387	0.410	0.408	0.429	0.376	0.397	0.370	0.399	0.375	0.399	0.384	0.402	0.376	0.419	0.664	0.612	0.449	0.459	0.513	0.491	0.494	0.479	0.424	0.432	0.865	0.713	0.837	0.728	0.878	0.740	0.399	0.428
	720	0.454	0.472	0.523	0.514	0.477	0.456	0.447	0.466	0.472	0.490	0.521	0.500	0.506	0.507	0.963	0.782	0.514	0.512	0.643	0.616	0.562	0.535	0.547	0.533	1.181	0.865	1.257	0.889	1.135	0.852	0.608	0.573
ETTh2	96	0.288	0.345	0.285	0.348	0.285	0.342	0.274	0.336	0.289	0.353	0.340	0.374	0.358	0.397	0.645	0.597	0.346	0.388	0.476	0.458	0.340	0.391	0.397	0.437	3.755	1.525	2.626	1.317	2.116	1.197	0.327	0.387
	720	0.403	0.439	0.399	0.435	0.406	0.441	0.379	0.422	0.605	0.551	0.462	0.468	0.463	0.474	0.963	0.783	0.515	0.511	0.562	0.560	0.500	0.497	0.863	0.672	3.647	1.625	3.874	1.697	3.188	1.540	1.454	0.847
ETTm1	96	0.293	0.349	0.384	0.403	0.292	0.346	0.290	0.342	0.299	0.343	0.338	0.375	0.379	0.419	0.543	0.510	0.505	0.475	0.386	0.398	0.375	0.398	0.374	0.400	0.672	0.571	0.538	0.528	0.600	0.546	0.318	0.367
	720	0.405	0.416	0.437	0.429	0.417	0.421	0.416	0.420	0.425	0.421	0.478	0.450	0.543	0.490	0.908	0.724	0.671	0.561	0.585	0.516	0.499	0.462	0.527	0.502	1.166	0.823	1.102	0.841	1.153	0.820	0.448	0.448
ETTm2	96	0.170	0.260	0.181	0.269	0.173	0.262	0.165	0.255	0.167	0.269	0.187	0.267	0.203	0.287	0.435	0.507	0.255	0.339	0.192	0.274	0.189	0.280	0.209	0.308	0.365	0.453	0.658	0.619	0.768	0.642	0.197	0.271
	720	0.363	0.387	0.366	0.388	0.378	0.401	0.362	0.385	0.397	0.421	0.408	0.403	0.421	0.415	3.625	1.451	0.433	0.432	0.417	0.413	0.414	0.413	0.675	0.587	3.379	1.338	2.631	1.242	3.048	1.328	0.395	0.419
ILI	24	2.728	1.114	3.025	1.195	2.063	0.881	1.319	0.754	2.215	1.081	2.317	0.934	3.228	1.260	1.420	2.012	3.483	1.287	2.294	0.945	2.527	1.020	8.313	2.144	5.764	1.677	4.400	1.382	4.480	1.444	4.539	1.528
	60	2.893	1.132	3.245	1.221	1.979	0.957	1.470	0.788	2.368	1.096	2.027	0.928	2.857	1.157	7.662	2.100	2.770	1.125	2.178	0.963	2.487	1.016	7.283	1.985	5.264	1.564	4.882	1.483	5.278	1.560	5.429	1.661
ECL	96	0.138	0.242	-	-	0.139	0.238	0.129	0.222	0.140	0.237	0.168	0.272	0.193	0.308	0.386	0.449	0.201	0.317	0.169	0.273	0.187	0.304	0.207	0.307	0.274	0.368	0.312	0.402	0.258	0.357	0.131	0.228
	720	0.211	0.305	-	-	0.206	0.297	0.197	0.290	0.203	0.301	0.220	0.320	0.246	0.355	0.376	0.445	0.254	0.361	0.222	0.321	0.233	0.345	0.265	0.360	0.373	0.439	0.340	0.420	0.283	0.376	0.208	0.298
Traffic	96	0.391	0.282	-	-	0.388	0.282	0.360	0.249	0.410	0.282	0.593	0.321	0.587	0.366	2.085	0.468	0.613	0.388	0.612	0.338	0.607	0.392	0.615	0.391	0.719	0.391	0.732	0.423	0.684	0.384	0.375	0.259
	720	0.450	0.310	-	-	0.450	0.312	0.432	0.286	0.466	0.315	0.640	0.350	0.626	0.382	0.881	0.473	0.660	0.408	0.653	0.355	0.632	0.396	0.658	0.407	0.864	0.472	0.755	0.423	0.717	0.396	0.508	0.335

Table 11: Long-term forecasting performance measured using Mean Squared Error (MSE) and Mean Absolute Error (MAE).

Baselines. We compare MOMENT with GPT4TS Zhou et al. [2023], TimesNet Wu et al. [2023], N-BEATS Oreshkin et al. [2020], 3 statistical and 3 benchmarking forecasting methods: AutoARIMA, AutoTheta, AutoETS, Naive, Seasonal Naive, and Random Walk (Makridakis et al., 2020).

Experimental Setting. Each statistical method is *fit* on individual time-series before producing a forecast. We follow the same train-test split and forecasting horizons from the M3 and M4 competitions, and report sMAPE as is common in prior work Oreshkin et al. [2020], Wu et al. [2023][13]. We follow the same experimental procedure as outlined in Oreshkin et al. [2021] with two exceptions: our results are reported only (1) on 40% of the M3 and M4 datasets that were unseen during pre-training, (2) a subset of frequencies with largest support in the datasets. Daily, hourly, and weekly frequencies had very little data and we could not get promising zero-shot performance for any of the deep learning models. Some ways that prior work Oreshkin et al. [2021] had overcome this issue was by leveraging data from frequencies with plenty of data. We also believe that ensembling played an important part in N-BEATS promising zero-shot performance.

Hyperparameters. The hyperparameters used for training all models in our short-horizon forecasting experiments are shown in Table 12.

Model	Hyper-parameters	
MOMENT$_{LP}$	sequence length :	512
	patch length :	8
	patch stride length :	8
	initial learning rate :	0.002
	max epochs :	$\{5, 10\}$
MOMENT$_0$	sequence length :	512
	patch length :	8
	patch stride length :	8
	initial learning rate :	0.001
N-BEATS	sequence length :	512
	stack types :	{'trend', 'seasonality'}
	number of blocks per stack :	3
	thetas dimensions :	$\{4, 8\}$
	hidden layer units :	256
GPT4TS	forecast horizon :	0
	gpt layers :	3
	patch length :	1
	patch stride length :	1
	sequence length :	512
TimesNet	sequence length :	512
	model dimension :	32
	dimension of feedforward layer :	32
	top-k :	5

Table 12: Hyper-parameter values for short-horizon forecasting models.

[13]The definitions of sMAPE were different in the M3 and M4 competitions. In our experiments, we used the same definition as the M4 competition.

Source Dataset → Target Dataset ↓	M4	Fred
M4		
Yearly	-	Yearly
Quarterly	-	Quarterly
Monthly	-	Monthly
M3		
Yearly	Yearly	Yearly
Quarterly	Quarterly	Quarterly
Monthly	Monthly	Monthly

Table 13: Experimental settings for short-horizon forecasting experiments for varying source and target datasets.

	Adjusted Best F_1						VUS-ROC					
Model name Dataset name	**Anomaly Transformer**	`MOMENT`$_0$	`MOMENT`$_{LP}$	**DGHL**	**GPT4TS**	**TimesNet**	AnomalyTransformer	`MOMENT`$_0$	`MOMENT`$_{LP}$	**DGHL**	**GPT4TS**	**TimesNet**
1sddb40	0.030	0.560	0.540	0.390	0.190	0.680	0.640	0.740	0.750	0.640	0.660	0.720
BIDMC1	0.990	1.000	1.000	1.000	1.000	1.000	0.690	0.560	0.650	0.720	0.630	0.740
CHARISfive	0.010	0.070	0.130	0.020	0.020	0.080	0.360	0.430	0.400	0.510	0.450	0.460
CHARISten	0.020	0.060	0.110	0.040	0.100	0.030	0.430	0.500	0.540	0.520	0.510	0.530
CIMIS44AirTemperature3	0.060	1.000	0.980	0.500	0.180	0.470	0.640	0.740	0.750	0.740	0.620	0.740
CIMIS44AirTemperature5	0.390	0.990	0.990	0.960	0.200	0.710	0.780	0.750	0.810	0.920	0.560	0.720
ECG2	1.000	1.000	1.000	0.620	0.900	1.000	0.830	0.740	0.840	0.630	0.780	0.600
ECG3	0.360	0.810	0.980	0.800	0.840	0.480	0.540	0.700	0.770	0.680	0.450	0.610
Fantasia	0.750	1.000	0.950	0.660	0.870	0.550	0.730	0.630	0.640	0.710	0.650	0.610
GP711MarkerLFM5z4	0.930	0.810	1.000	0.500	0.640	0.950	0.540	0.630	0.730	0.600	0.620	0.720
GP711MarkerLFM5z5	0.760	0.690	0.970	0.310	0.480	0.900	0.690	0.760	0.720	0.520	0.630	0.840
InternalBleeding4	NaN	1.000	NaN	NaN	NaN	NaN	NaN	0.650	NaN	NaN	NaN	NaN
InternalBleeding5	0.940	1.000	1.000	1.000	0.920	1.000	0.460	0.600	0.690	0.760	0.630	0.940
Italianpowerdemand	0.010	0.390	0.740	0.590	0.010	0.440	0.450	0.800	0.770	0.700	0.480	0.710
Lab2Cmac011215EPG5	0.990	0.970	0.980	0.340	0.600	0.990	0.770	0.620	0.630	0.710	0.640	0.610
Lab2Cmac011215EPG6	0.410	0.090	0.100	0.260	0.100	0.170	0.700	0.480	0.480	0.600	0.520	0.450
MesoplodonDensirostris	1.000	0.910	0.840	0.790	1.000	1.000	0.850	0.730	0.720	0.740	0.690	0.790
PowerDemand1	0.870	0.260	0.440	0.490	0.760	0.950	0.720	0.520	0.540	0.530	0.600	0.750
TkeepFirstMARS	0.010	0.080	0.150	0.020	0.020	0.230	0.520	0.570	0.760	0.460	0.500	0.790
TkeepSecondMARS	0.830	0.950	1.000	0.160	0.120	0.950	0.720	0.950	0.910	0.970	0.810	0.980
WalkingAceleration5	0.990	1.000	1.000	0.910	0.870	0.930	0.940	0.860	0.870	0.930	0.910	0.850
apneaecg	0.400	0.210	0.200	0.250	0.310	0.260	0.580	0.690	0.690	0.590	0.580	0.760
apneaecg2	0.650	0.940	1.000	1.000	1.000	0.650	0.790	0.750	0.740	0.730	0.650	0.610
gait1	0.180	0.710	0.360	0.070	0.410	0.520	0.630	0.650	0.570	0.600	0.580	0.600
gaitHunt1	0.080	0.500	0.430	0.020	0.100	0.300	0.810	0.640	0.680	0.570	0.710	0.840
insectEPG2	0.120	0.110	0.230	0.140	0.810	0.960	0.650	0.570	0.820	0.650	0.560	0.730
insectEPG4	0.980	1.000	1.000	0.460	0.210	0.850	0.690	0.700	0.720	0.730	0.490	0.650
ltstdbs30791AS	1.000	1.000	1.000	1.000	1.000	1.000	0.780	0.760	0.810	0.770	0.740	0.670
mit14046longtermecg	0.450	0.560	0.590	0.530	0.580	0.600	0.790	0.660	0.660	0.640	0.610	0.840
park3m	0.150	0.560	0.640	0.200	0.630	0.930	0.630	0.750	0.780	0.650	0.540	0.780
qtdbSel1005V	0.410	0.570	0.650	0.400	0.390	0.530	0.520	0.640	0.640	0.490	0.610	0.540
qtdbSel100MLII	0.420	0.780	0.840	0.410	0.600	0.870	0.620	0.580	0.620	0.590	0.580	0.650
resperation1	0.000	0.040	0.150	0.030	0.010	0.030	0.750	0.500	0.670	0.740	0.470	0.670
s20101mML2	0.690	0.650	0.710	0.150	0.050	0.080	0.640	0.760	0.720	0.690	0.640	0.690
sddb49	0.890	1.000	1.000	0.880	0.940	1.000	0.660	0.730	0.730	0.740	0.580	0.680
sel840mECG1	0.160	0.610	0.660	0.280	0.210	0.360	0.620	0.720	0.720	0.870	0.650	0.600
sel840mECG2	0.150	0.360	0.390	0.320	0.280	0.210	0.590	0.710	0.690	0.490	0.520	0.520
tilt12744mtable	0.070	0.110	0.240	0.100	0.000	0.030	0.480	0.670	0.740	0.660	0.510	0.640
tilt12754table	0.230	0.590	0.640	0.040	0.060	0.050	0.600	0.750	0.820	0.790	0.550	0.750
tiltAPB2	0.920	0.960	0.980	0.360	0.830	0.380	0.770	0.750	0.770	0.710	0.600	0.700
tiltAPB3	0.170	0.480	0.850	0.030	0.050	0.090	0.680	0.610	0.650	0.540	0.440	0.580
weallwalk	0.000	0.520	0.580	0.070	0.130	0.170	0.730	0.930	0.930	0.860	0.870	0.850

Table 14: Anomaly detection performance measured using adj. best F_1 and VUS-ROC for a subset of 45 datasets sampled from the UCR Anomaly archive.

D.2 Classification

Task Description. The classification problem comprises of learning a mapping $f : \mathcal{T} \rightarrow \{1, ..., C\}$ from a time-series to a finite set of classes, using a training dataset of the form $\{(\mathcal{T}_0, c_0), ..., (\mathcal{T}_n, c_n)\}$, $c_i \in \{1, ..., C\}$. One straightforward way to use `MOMENT` to learn f is to replace its reconstruction head with a linear head that maps patch representations to the C logits. Another way would be to learn f in two stages, as is common in prior work on unsupervised representation learning Yue et al. [2022], Franceschi et al. [2019]: in the first stage, we obtain sequence-level representations for each time-series without access to labels. The second stage involves learning any ML classifier (e.g., Support Vector Machine with RBF kernel) using these representations and labels.

Datasets. We conduct experiments on a subset of 95 datasets from the UCR Classification Archive Dau et al. [2018]. These datasets (listed in Table 10) comprise of equal-length univariate time-series shorter than 512 time steps.

Baselines. We compare `MOMENT` against 5 **unsupervised representation learning** methods (TS2Vec Yue et al. [2022], TST Zerveas et al. [2021], TS-TCC Eldele et al. [2021], TNC Tonekaboni et al. [2021], and T-Loss Franceschi et al. [2019]), 8 **supervised deep learning** (CNN Zebik et al. [2017], Encoder Serrà et al. [2018], FCN Wang et al.

Dataset	$\texttt{MOMENT}_0$	TimesNet	GPT4TS	TS2Vec	T-Loss	TNC	TS-TCC	TST	CNN	Encoder	FCN	MCDNN	MLP	ResNet	t-LeNet	TWIESN	DTW
GestureMidAirD2	0.608	0.131	0.200	0.469	0.546	0.254	0.254	0.138	0.518	0.480	0.631	0.500	0.545	0.668	0.038	0.575	0.608
UWaveGestureLibraryX	0.821	0.688	0.749	0.795	0.785	0.733	0.733	0.569	0.721	0.771	0.754	0.726	0.768	0.781	0.127	0.608	0.728
GesturePebbleZ2	0.816	0.310	0.285	0.873	0.899	0.430	0.430	0.380	0.778	0.796	0.781	0.720	0.701	0.777	0.184	0.843	0.671
ECG5000	0.942	0.584	0.584	0.935	0.933	0.941	0.941	0.928	0.928	0.941	0.940	0.933	0.930	0.935	0.584	0.922	0.924
OSULeaf	0.785	0.397	0.231	0.851	0.760	0.723	0.723	0.545	0.482	0.554	0.979	0.419	0.560	0.980	0.182	0.628	0.591
MedicalImages	0.762	0.571	0.496	0.789	0.750	0.747	0.747	0.632	0.671	0.664	0.778	0.627	0.719	0.770	0.514	0.649	0.737
Ham	0.581	0.686	0.781	0.714	0.724	0.743	0.743	0.524	0.720	0.682	0.707	0.718	0.699	0.758	0.514	0.768	0.467
DistalPhalanxTW	0.612	0.604	0.619	0.698	0.676	0.676	0.676	0.568	0.671	0.694	0.695	0.685	0.610	0.663	0.285	0.591	0.590
ProximalPhalanxOutlineCorrect	0.856	0.869	0.801	0.887	0.859	0.873	0.873	0.770	0.807	0.768	0.907	0.866	0.730	0.920	0.684	0.817	0.784
FreezerRegularTrain	0.982	0.926	0.829	0.986	0.956	0.989	0.989	0.922	0.987	0.760	0.997	0.973	0.906	0.998	0.500	0.946	0.899
TwoLeadECG	0.847	0.633	0.658	0.986	0.999	0.976	0.976	0.871	0.877	0.784	0.999	0.806	0.753	1.000	0.500	0.949	0.905
GunPointMaleVersusFemale	0.991	0.601	0.475	1.000	0.997	0.997	0.997	1.000	0.977	0.978	0.997	0.952	0.980	0.992	0.525	0.988	0.997
Trace	1.000	0.760	0.710	1.000	0.990	1.000	1.000	1.000	0.952	0.740	1.000	0.902	0.806	1.000	0.240	0.934	1.000
SmoothSubspace	0.820	0.440	0.453	0.980	0.960	0.953	0.953	0.827	0.976	0.964	0.975	0.963	0.980	0.980	0.333	0.849	0.827
MiddlePhalanxTW	0.532	0.506	0.571	0.584	0.591	0.610	0.610	0.506	0.551	0.597	0.501	0.562	0.536	0.495	0.286	0.569	0.506
SyntheticControl	0.990	0.467	0.437	0.997	0.987	0.990	0.990	0.490	0.987	0.973	0.989	0.953	0.973	0.997	0.167	0.879	0.993
ShapesAll	0.815	0.238	0.237	0.902	0.848	0.773	0.773	0.733	0.617	0.679	0.894	0.599	0.776	0.926	0.017	0.643	0.768
AllGestureWiimoteX	0.607	0.209	0.237	0.777	0.763	0.697	0.697	0.259	0.411	0.475	0.713	0.261	0.477	0.741	0.100	0.522	0.716
Wafer	0.997	0.989	0.994	0.998	0.992	0.994	0.994	0.991	0.961	0.998	0.997	0.992	0.996	0.998	0.892	0.916	0.980
FaceFour	0.852	0.830	0.659	0.932	0.920	0.773	0.773	0.511	0.905	0.852	0.930	0.711	0.836	0.955	0.295	0.857	0.830
CricketX	0.749	0.523	0.531	0.782	0.713	0.731	0.731	0.385	0.535	0.644	0.794	0.513	0.591	0.799	0.074	0.627	0.754
DistalPhalanxOutlineCorrect	0.717	0.786	0.659	0.761	0.775	0.754	0.754	0.728	0.772	0.724	0.760	0.759	0.727	0.770	0.583	0.711	0.717
ChlorineConcentration	0.765	0.618	0.565	0.832	0.749	0.753	0.753	0.562	0.608	0.583	0.817	0.662	0.800	0.853	0.533	0.554	0.648
Chinatown	0.965	0.274	0.857	0.965	0.951	0.983	0.983	0.936	0.977	0.966	0.980	0.945	0.872	0.978	0.726	0.825	0.957
GestureMidAirD1	0.646	0.285	0.292	0.608	0.608	0.369	0.369	0.208	0.534	0.528	0.695	0.518	0.575	0.698	0.038	0.549	0.569
MiddlePhalanxOutlineAgeGroup	0.461	0.344	0.526	0.636	0.656	0.630	0.630	0.617	0.534	0.577	0.535	0.558	0.522	0.545	0.571	0.578	0.500
UMD	0.993	0.681	0.368	1.000	0.993	0.986	0.986	0.910	0.960	0.771	0.988	0.842	0.949	0.990	0.333	0.835	0.993
Crop	0.734	0.388	0.341	0.756	0.722	0.742	0.742	0.710	0.670	0.760	0.738	0.687	0.618	0.743	0.042	0.489	0.665
GesturePebbleZ1	0.849	0.512	0.605	0.930	0.919	0.395	0.395	0.500	0.844	0.821	0.880	0.769	0.792	0.901	0.163	0.840	0.791
WordSynonyms	0.688	0.335	0.451	0.676	0.691	0.531	0.531	0.422	0.568	0.557	0.561	0.470	0.599	0.617	0.219	0.506	0.649
ArrowHead	0.743	0.360	0.429	0.857	0.766	0.737	0.737	0.771	0.717	0.630	0.843	0.678	0.784	0.838	0.303	0.689	0.703
Wine	0.537	0.519	0.611	0.870	0.815	0.778	0.778	0.500	0.519	0.556	0.611	0.500	0.541	0.722	0.500	0.744	0.574
Coffee	0.893	0.964	0.679	1.000	1.000	1.000	1.000	0.821	1.000	0.886	1.000	0.979	0.993	1.000	0.507	0.979	1.000
Earthquakes	0.748	0.741	0.748	0.748	0.748	0.748	0.748	0.748	0.709	0.740	0.725	0.748	0.727	0.712	0.748	0.748	0.719
Herring	0.594	0.531	0.578	0.641	0.594	0.594	0.594	0.594	0.531	0.512	0.644	0.572	0.491	0.600	0.594	0.625	0.531
Beef	0.833	0.400	0.167	0.767	0.667	0.600	0.600	0.500	0.767	0.707	0.680	0.507	0.713	0.753	0.200	0.527	0.633
MiddlePhalanxOutlineCorrect	0.467	0.512	0.519	0.838	0.825	0.818	0.818	0.753	0.744	0.752	0.795	0.796	0.755	0.826	0.570	0.743	0.698
ECGFiveDays	0.804	0.519	0.561	1.000	1.000	0.878	0.878	0.763	0.874	0.842	0.985	0.800	0.973	0.966	0.497	0.723	0.768
Yoga	0.834	0.672	0.691	0.887	0.837	0.791	0.791	0.830	0.786	0.753	0.837	0.741	0.856	0.867	0.536	0.626	0.837
Adiac	0.688	0.565	0.598	0.762	0.675	0.767	0.767	0.550	0.393	0.318	0.841	0.620	0.391	0.833	0.023	0.428	0.604
MoteStrain	0.774	0.700	0.681	0.861	0.851	0.843	0.843	0.768	0.885	0.872	0.936	0.691	0.855	0.924	0.539	0.809	0.835
Strawberry	0.951	0.946	0.935	0.962	0.954	0.965	0.965	0.916	0.952	0.959	0.975	0.958	0.959	0.980	0.643	0.911	0.941
InsectWingbeatSound	0.607	0.529	0.598	0.630	0.597	0.415	0.415	0.266	0.585	0.630	0.392	0.587	0.604	0.499	0.091	0.435	0.355
DodgerLoopWeekend	0.826	0.638	0.804	0.964	NaN	NaN	NaN	0.732	0.974	0.983	0.904	0.978	0.978	0.952	0.739	0.954	0.949
Meat	0.917	0.433	0.667	0.950	0.950	0.883	0.883	0.900	0.913	0.787	0.803	0.787	0.893	0.990	0.333	0.970	0.933
MelbournePedestrian	0.876	0.718	0.207	0.959	0.944	0.949	0.949	0.741	0.813	0.884	0.912	0.840	0.863	0.909	0.100	0.730	0.791
FaceAll	0.791	0.177	0.147	0.771	0.786	0.813	0.813	0.504	0.774	0.794	0.938	0.720	0.794	0.867	0.080	0.673	0.808
FacesUCR	0.811	0.679	0.462	0.924	0.884	0.863	0.863	0.543	0.873	0.867	0.943	0.775	0.831	0.954	0.143	0.641	0.905
AllGestureWiimoteY	0.666	0.223	0.160	0.793	0.726	0.741	0.741	0.423	0.479	0.509	0.784	0.420	0.571	0.794	0.100	0.600	0.729
ShakeGestureWiimoteZ	0.960	0.020	0.080	0.940	0.920	0.860	0.860	0.760	0.580	0.756	0.884	0.516	0.548	0.880	0.100	0.864	0.860
BME	0.960	0.467	0.367	0.993	0.993	0.933	0.933	0.760	0.947	0.827	0.836	0.896	0.905	0.999	0.333	0.819	0.900
FordB	0.798	0.754	0.677	0.794	0.793	0.815	0.815	0.507	0.749	0.777	0.772	0.698	0.707	0.813	0.503	0.512	0.620
Fish	0.800	0.726	0.731	0.926	0.891	0.817	0.817	0.720	0.855	0.734	0.961	0.720	0.848	0.981	0.126	0.878	0.823
SonyAIBORobotSurface2	0.829	0.646	0.650	0.871	0.889	0.907	0.907	0.745	0.831	0.844	0.980	0.804	0.831	0.975	0.617	0.635	0.831
FiftyWords	0.802	0.499	0.492	0.771	0.732	0.653	0.653	0.525	0.624	0.658	0.646	0.611	0.708	0.740	0.125	0.518	0.690
ToeSegmentation1	0.925	0.456	0.561	0.917	0.939	0.930	0.930	0.807	0.598	0.706	0.961	0.559	0.589	0.957	0.526	0.882	0.772
FreezerSmallTrain	0.902	0.704	0.500	0.870	0.933	0.979	0.979	0.920	0.739	0.676	0.683	0.688	0.686	0.832	0.500	0.917	0.753
TwoPatterns	0.994	0.989	0.923	1.000	0.999	0.999	0.999	0.466	0.991	1.000	0.870	0.976	0.948	1.000	0.259	0.875	1.000
ShapeletSim	0.961	0.500	0.489	1.000	0.672	0.683	0.683	0.489	0.497	0.510	0.706	0.498	0.513	0.782	0.500	0.546	0.650
Plane	0.990	0.981	0.924	1.000	0.990	1.000	1.000	0.933	0.962	0.964	1.000	0.952	0.977	1.000	0.143	1.000	1.000
GestureMidAirD3	0.369	0.085	0.162	0.292	0.285	0.177	0.177	0.154	0.317	0.368	0.326	0.278	0.382	0.340	0.038	0.275	0.323
DiatomSizeReduction	0.879	0.967	0.987	0.984	0.984	0.977	0.977	0.961	0.954	0.880	0.346	0.646	0.909	0.301	0.301	0.914	0.967
CricketZ	0.731	0.459	0.397	0.792	0.708	0.713	0.713	0.403	0.501	0.651	0.810	0.484	0.629	0.809	0.062	0.643	0.754
Lightning7	0.726	0.575	0.562	0.863	0.795	0.685	0.685	0.411	0.647	0.696	0.825	0.559	0.616	0.827	0.260	0.608	0.726
UWaveGestureLibraryY	0.738	0.547	0.648	0.719	0.710	0.641	0.641	0.348	0.626	0.676	0.642	0.639	0.699	0.666	0.121	0.497	0.634
GunPointAgeSpan	0.962	0.494	0.494	0.987	0.994	0.994	0.994	0.991	0.912	0.890	0.996	0.887	0.934	0.997	0.494	0.965	0.918
DistalPhalanxOutlineAgeGroup	0.669	0.597	0.489	0.727	0.727	0.755	0.755	0.741	0.758	0.761	0.718	0.729	0.647	0.718	0.433	0.705	0.770
SwedishLeaf	0.923	0.894	0.899	0.941	0.914	0.923	0.923	0.738	0.884	0.902	0.967	0.841	0.845	0.963	0.064	0.837	0.792
CBF	0.960	0.761	0.830	1.000	0.983	0.998	0.998	0.898	0.959	0.977	0.994	0.908	0.869	0.996	0.332	0.896	0.997
BeetleFly	0.900	0.400	0.700	0.900	0.800	0.800	0.800	1.000	0.900	0.620	0.910	0.630	0.880	0.850	0.500	0.790	0.700
AllGestureWiimoteZ	0.537	0.221	0.116	0.746	0.723	0.689	0.689	0.447	0.375	0.396	0.692	0.287	0.439	0.726	0.100	0.516	0.643
DodgerLoopDay	0.438	0.237	0.200	0.562	NaN	NaN	NaN	0.200	0.312	0.487	0.143	0.305	0.160	0.150	0.160	0.593	0.500
GunPointOldVersusYoung	0.981	0.508	0.524	1.000	1.000	1.000	1.000	1.000	0.922	0.923	0.989	0.926	0.941	0.989	0.524	0.975	0.838
FordA	0.936	0.913	0.914	0.936	0.928	0.930	0.930	0.568	0.896	0.928	0.914	0.863	0.816	0.937	0.510	0.555	0.555
ItalyPowerDemand	0.911	0.837	0.880	0.925	0.954	0.955	0.955	0.845	0.954	0.964	0.963	0.966	0.953	0.962	0.499	0.871	0.950
ProximalPhalanxOutlineAgeGroup	0.863	0.868	0.839	0.834	0.844	0.839	0.839	0.854	0.812	0.872	0.825	0.839	0.849	0.847	0.488	0.839	0.805
GunPoint	0.927	0.887	0.847	0.980	0.980	0.993	0.993	0.827	0.948	0.784	1.000	0.907	0.928	0.991	0.493	0.989	0.907
ProximalPhalanxTW	0.712	0.800	0.712	0.824	0.771	0.800	0.800	0.780	0.777	0.791	0.761	0.775	0.767	0.773	0.341	0.784	0.761
PickupGestureWiimoteZ	0.620	0.100	0.080	0.820	0.740	0.600	0.600	0.240	0.608	0.496	0.744	0.412	0.604	0.704	0.100	0.616	0.660
SonyAIBORobotSurface1	0.729	0.542	0.589	0.903	0.902	0.899	0.899	0.724	0.690	0.729	0.958	0.655	0.692	0.961	0.429	0.725	0.725
PowerCons	0.894	0.956	0.989	0.961	0.900	0.961	0.961	0.911	0.960	0.971	0.863	0.929	0.977	0.879	0.500	0.852	0.878
PhalangesOutlinesCorrect	0.652	0.614	0.663	0.809	0.784	0.804	0.804	0.773	0.799	0.745	0.818	0.795	0.756	0.845	0.613	0.656	0.728
BirdChicken	0.850	0.450	0.550	0.800	0.850	0.650	0.650	0.650	0.710	0.510	0.940	0.540	0.740	0.880	0.500	0.620	0.750
ToeSegmentation2	0.915	0.731	0.731	0.892	0.900	0.877	0.877	0.615	0.752	0.702	0.889	0.649	0.745	0.894	0.815	0.794	0.838
CricketY	0.746	0.531	0.521	0.749	0.728	0.718	0.718	0.467	0.582	0.639	0.793	0.521	0.598	0.810	0.085	0.652	0.744
ElectricDevices	0.646	0.552	0.506	0.721	0.707	0.686	0.686	0.676	0.686	0.702	0.706	0.653	0.593	0.728	0.242	0.605	0.602
DodgerLoopGame	0.623	0.471	0.717	0.841	NaN	NaN	NaN	0.696	0.816	0.810	0.768	0.877	0.865	0.710	0.478	0.716	0.877
Fungi	0.898	0.043	0.054	0.957	1.000	0.753	0.753	0.366	0.961	0.934	0.018	0.051	0.863	0.177	0.063	0.439	0.839
Symbols	0.936	0.864	0.694	0.976	0.963	0.916	0.916	0.786	0.808	0.754	0.955	0.644	0.836	0.893	0.174	0.798	0.950
UWaveGestureLibraryZ	0.765	0.632	0.643	0.770	0.757	0.690	0.690	0.655	0.630	0.684	0.727	0.645	0.697	0.749	0.121	0.573	0.658
ECG200	0.760	0.830	0.790	0.920	0.940	0.880	0.880	0.830	0.816	0.884	0.888	0.838	0.914	0.874	0.640	0.874	0.770

Table 15: Classification accuracy of methods across 91 UCR datasets. MOMENT without fine-tuning on individual datasets demonstrates promising accuracy.

[2017], MCNN Cui et al. [2016], MLP Wang et al. [2017], ResNet Wang et al. [2017], t-LeNet Le Guennec et al. [2016], TWIESN Tanisaro and Heidemann [2016]), 1 **supervised statistical learning** method DTW Dau et al. [2018]), TimesNet Wu et al. [2023] and GPT4TS Zhou et al. [2023].

Experimental Setting. All models except for MOMENT were trained on each dataset individually, either with labels for supervised deep and statistical learning methods), or without labels for representation learning methods. We collect baseline results for deep learning methods from Ismail Fawaz et al. [2019], representation learning methods from Yue et al. [2022], and DTW from Dau et al. [2018]. We report accuracy as the evaluation metric.

Hyperparameters. The hyperparameters used for evaluating classification experiments are shown in Table 16.

Model	Hyper-parameters	
$\texttt{MOMENT}_0$	sequence length :	512
	patch length :	8
	patch stride length :	8
SVM	C :	$\{0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000\}$
	kernel :	RBF
	degree :	3
	cache size :	200
	max iterations :	10000000
	decision function shape :	One versus rest

Table 16: Hyper-parameter values for classification.

D.3 Anomaly Detection

Task Description. Given a time-series $\mathcal{T}$, anomaly detection is a binary classification problem, where the goal is to detect whether a time step x_i is indicative of an anomaly or not. As shown in Fig. 4 (v), to detect anomalies in $\mathcal{T}$, we retain MOMENT's reconstruction head and use it to reconstruct the input time-series. Then, time steps where observations and predictions differ beyond a certain threshold are classified as anomalies[14].

Datasets. We conduct experiments on a subset of 46 univariate time-series from the UCR Anomaly Archive Wu and Keogh [2023], as enumerated in Table 11. When choosing the subset of time-series, we prioritized coverage over different domains and data sources represented in the archive.

Baselines. We compare MOMENT with 2 state-of-the-art anomaly detection methods DGHL Challu et al. [2022] and Anomaly Transformer Xu et al. [2022] along with TimesNet and GPT4TS. We also include k-Nearest Neighbors (with $k = 5$) Ramaswamy et al. [2000], a classical anomaly detection method in our experiments. In the zero-shot setting, we compare MOMENT to randomly initialized DGHL ($\texttt{DGHL}_0$)[15] and k-NN.

Experimental Setting. All algorithms use a fixed anomaly detection window size (= 512). Based on prior work Wu et al. [2023], Zhou et al. [2023], we use the mean squared error between predictions and observations as the anomaly criterion[16]. Following prior work Goswami et al. [2023], we downsample all time-series longer than 2560 timesteps by a factor of 10 to speed up the training and evaluation process.

We report two anomaly detection metrics: adjusted best F_1 which is frequently used in practice Goswami et al. [2023], Challu et al. [2022], and the recently proposed volume under ROC surface (VUS-ROC) metric Paparrizos et al. [2022b]. For both metrics, higher scores are better.

Hyperparameters. The hyperparameters used for training all models in our anomaly detection experiments are shown in Table 17.

D.4 Imputation

Task Description. Consider a time-series $\mathcal{T} = [x_1, ..., x_L]$ and an observation mask $\mathcal{M} = [m_1, ..., m_L]$, where $m_i = 0$ if x_i is missing and $m_i = 1$ if x_i is observed. Then imputation is the task of estimating the missing values $\mathcal{T}$ by

[14] Estimating good thresholds for anomaly detection is beyond the scope of this study and an active area of research Goswami et al. [2023], Schmidl et al. [2022].

[15] Randomly initialized DGHL is not a trivial zero-shot baseline, since it performs gradient descent to find the best latent z that minimizes reconstruction error during inference time Challu et al. [2022].

[16] To ensure a fair comparison, we do not use Anomaly Transformer's joint criterion as the anomaly score. We believe that this might put the Anomaly Transformer at some disadvantage in our experiments.

Model	Hyper-parameters	
MOMENT$_0$	sequence length :	512
	patch length :	8
	patch stride length :	8
MOMENT$_{LP}$	sequence length :	512
	patch length :	8
	patch stride length :	8
	initial lr :	$5e-5$
Anomaly Transformer	sequence length :	512
	number of channels :	1
	k :	3
	anomaly ratio :	4.00
	model dimensions :	512
	number of heads :	8
	embedding layers :	3
	dimension of feedforward layer :	512
DGHL	sequence length :	512
	number of channels :	1
	hidden multiplier :	32
	max filters :	256
	kernel multiplier :	1
	sub-windows :	4
	size of latent z vector :	50
	number of iteration in the Langevyn dynamics inference formula :	100
	z step size :	0.1
	noise std :	0.001
GPT4TS	sequence length :	512
	gpt layers :	3
	patch length :	1
	patch stride length :	1
	transformer backbone :	GPT-2
TimesNet	sequence length :	512
	dimension of model :	16
	dimension of feedforward layer :	16
	top k :	3
	number of kernels :	6
k-NN	k :	5

Table 17: Hyperparameter values for anomaly detection.

exploiting its observed values. We treat a patch as observed only if all its time steps are observed. For the remaining patches, we replace their patch embeddings with [MASK] and use MOMENT's default reconstruction head to impute its values (Fig. 4 (iv)).

Datasets. We evaluate imputation performance on 6 real-world datasets from domains where missing data is a common problem: 4 subsets of Electricity Transformer Temperature (ETT), Weather, and Electricity Wu et al. [2023], Zhou et al. [2023].

Baselines. We compare the two variants of MOMENT with 3 state-of-the-art deep learning methods, TimesNet, FPT, and DGHL; and 3 statistical interpolation methods, Cubic Spline, Linear, and 1-D Nearest Neighbor interpolation.

Experimental Setting. To evaluate the models' ability to interpolate missing values, we randomly mask contiguous sub-sequences of length 8. Instead of masking contiguous sub-sequences, previous studies Wu et al. [2023], Zhou et al. [2023] mask individual time points, making the imputation task much easier. The results from prior studies are shown in Table 19. We observe that the statistical methods perform similarly to transformer methods, owing to the ease of the task. For our experiments involving randomly masking patches of length 8, our results are shown in Table 20. We measure the imputation performance of models using mean squared error, over 4 different masking rates: 12.5%, 25%, 37.5%, and 50%. f

Hyperparameters. The hyperparameters used for training all models in our imputation experiments are shown in Table 18.

Model	Hyper-parameters	
$\texttt{MOMENT}_0$	`sequence length` :	512
	`patch length` :	8
	`patch stride length` :	8
$\texttt{MOMENT}_{\texttt{LP}}$	`sequence length` :	512
	`patch length` :	8
	`patch stride length` :	8
	`initial lr` :	0.0001
`GPT4TS`	`sequence length` :	512
	`gpt layers` :	3
	`patch length` :	1
	`patch stride length` :	1
	`transformer backbone` :	GPT-2
	`dimension of feedforward layer` :	16
`TimesNet`	`sequence length` :	512
	`dimension of model` :	64
	`dimension of feedforward layer` :	64
	`top k` :	3
	`number of kernels` :	6

Table 18: Hyperparameter values for imputation.

Dataset	Mask Ratio	GPT4TS MSE	GPT4TS MAE	TimesNet MSE	TimesNet MAE	PatchTST MSE	PatchTST MAE	ETSformer MSE	ETSformer MAE	LightTS MSE	LightTS MAE	DLinear MSE	DLinear MAE	FEDformer MSE	FEDformer MAE
ETTm1	12.5%	**0.017**	**0.085**	0.023	0.101	0.041	0.130	0.096	0.229	0.093	0.206	0.080	0.193	0.052	0.166
	25%	**0.022**	**0.096**	0.023	0.101	0.044	0.135	0.096	0.229	0.093	0.206	0.080	0.193	0.052	0.166
	37.5%	**0.029**	**0.111**	0.029	0.111	0.049	0.143	0.133	0.271	0.113	0.231	0.103	0.219	0.069	0.191
	50%	0.040	0.128	**0.036**	**0.124**	0.055	0.151	0.186	0.323	0.134	0.255	0.132	0.248	0.089	0.218
	Avg	0.028	**0.105**	**0.027**	0.107	0.047	0.140	0.120	0.253	0.104	0.218	0.093	0.206	0.062	0.177
ETTm2	12.5%	**0.017**	**0.076**	0.018	0.080	0.026	0.094	0.108	0.239	0.034	0.127	0.062	0.166	0.056	0.159
	25%	**0.020**	**0.080**	0.020	0.085	0.028	0.099	0.164	0.294	0.042	0.143	0.085	0.196	0.080	0.195
	37.5%	**0.022**	**0.087**	0.023	0.091	0.030	0.104	0.237	0.356	0.051	0.159	0.106	0.222	0.110	0.231
	50%	**0.025**	**0.095**	0.026	0.098	0.034	0.110	0.323	0.421	0.059	0.174	0.131	0.247	0.156	0.276
	Avg	**0.021**	**0.084**	0.022	0.088	0.029	0.102	0.208	0.327	0.046	0.151	0.096	0.208	0.101	0.215
ETTh1	12.5%	**0.043**	**0.140**	0.057	0.159	0.093	0.201	0.126	0.263	0.240	0.345	0.151	0.267	0.070	0.190
	25%	**0.054**	**0.156**	0.069	0.178	0.107	0.217	0.169	0.304	0.265	0.364	0.180	0.292	0.106	0.236
	37.5%	**0.072**	**0.180**	0.084	0.196	0.120	0.230	0.220	0.347	0.296	0.382	0.215	0.318	0.124	0.258
	50%	0.107	0.216	**0.102**	**0.215**	0.141	0.248	0.293	0.402	0.334	0.404	0.257	0.347	0.165	0.299
	Avg	**0.069**	**0.173**	0.078	0.187	0.115	0.224	0.202	0.329	0.284	0.373	0.201	0.306	0.117	0.246
ETTh2	12.5%	**0.039**	**0.125**	0.040	0.130	0.057	0.152	0.187	0.319	0.101	0.231	0.100	0.216	0.095	0.212
	25%	**0.044**	**0.135**	0.046	0.141	0.061	0.158	0.279	0.390	0.115	0.246	0.127	0.247	0.137	0.258
	37.5%	**0.051**	**0.147**	0.052	0.151	0.067	0.166	0.400	0.465	0.126	0.257	0.158	0.276	0.187	0.304
	50%	**0.059**	**0.158**	0.060	0.162	0.073	0.174	0.602	0.572	0.136	0.268	0.183	0.299	0.232	0.341
	Avg	**0.048**	**0.141**	0.049	0.146	0.065	0.163	0.367	0.436	0.119	0.250	0.142	0.259	0.163	0.279
ECL	12.5%	**0.080**	**0.194**	0.085	0.202	0.055	0.160	0.196	0.321	0.102	0.229	0.092	0.214	0.107	0.237
	25%	**0.087**	**0.203**	0.089	0.206	0.065	0.175	0.207	0.332	0.121	0.252	0.118	0.247	0.120	0.251
	37.5%	0.094	**0.211**	**0.094**	0.213	0.076	0.189	0.219	0.344	0.141	0.273	0.144	0.276	0.136	0.266
	50%	0.101	**0.220**	**0.100**	0.221	0.091	0.208	0.235	0.357	0.160	0.293	0.175	0.305	0.158	0.284
	Avg	**0.090**	**0.207**	0.092	0.210	0.072	0.183	0.214	0.339	0.131	0.262	0.132	0.260	0.130	0.259
Weather	12.5%	0.026	0.049	**0.025**	**0.045**	0.029	0.049	0.057	0.141	0.047	0.101	0.039	0.084	0.041	0.107
	25%	**0.028**	**0.052**	0.029	0.052	0.031	0.053	0.065	0.155	0.052	0.111	0.048	0.103	0.064	0.163
	37.5%	0.033	0.060	**0.031**	**0.057**	0.035	0.058	0.081	0.180	0.058	0.121	0.057	0.117	0.107	0.229
	50%	0.037	0.065	**0.034**	**0.062**	0.038	0.063	0.102	0.207	0.065	0.133	0.066	0.134	0.183	0.312
	Avg	0.031	0.056	**0.030**	**0.054**	0.060	0.144	0.076	0.171	0.055	0.117	0.052	0.110	0.099	0.203

Dataset	Mask Ratio	Stationary MSE	Stationary MAE	Autoformer MSE	Autoformer MAE	Informer MSE	Informer MAE	Reformer MSE	Reformer MAE	Naive MSE	Naive MAE	Linear MSE	Linear MAE	Nearest MSE	Nearest MAE	Cubic MSE	Cubic MAE
ETTm1	12.5%	0.032	0.119	0.046	0.144	0.063	0.180	0.042	0.146	0.059	0.145	0.034	0.109	0.055	0.140	0.052	0.135
	25%	0.032	0.119	0.046	0.144	0.063	0.180	0.042	0.146	0.066	0.153	0.036	0.112	0.056	0.142	0.060	0.142
	37.5%	0.039	0.131	0.057	0.161	0.079	0.200	0.063	0.182	0.077	0.164	0.038	0.117	0.060	0.146	0.071	0.151
	50%	0.047	0.145	0.067	0.174	0.093	0.218	0.082	0.208	0.094	0.178	0.042	0.123	0.066	0.153	0.100	0.164
	Avg	0.036	0.126	0.051	0.150	0.071	0.188	0.055	0.166	0.074	0.160	0.038	0.115	0.059	0.145	0.071	0.148
ETTm2	12.5%	0.021	0.088	0.023	0.092	0.133	0.270	0.108	0.228	0.038	0.095	0.023	0.077	0.035	0.091	0.033	0.097
	25%	0.024	0.096	0.026	0.101	0.135	0.272	0.136	0.262	0.041	0.100	0.025	0.081	0.036	0.093	0.039	0.103
	37.5%	0.027	0.103	0.030	0.108	0.155	0.293	0.175	0.300	0.046	0.106	0.027	0.085	0.038	0.095	0.047	0.111
	50%	0.030	0.108	0.035	0.119	0.200	0.333	0.211	0.329	0.051	0.115	0.030	0.090	0.041	0.100	0.062	0.122
	Avg	0.026	0.099	0.029	0.105	0.156	0.292	0.157	0.280	0.044	0.104	0.026	0.083	0.038	0.095	0.045	0.109
ETTh1	12.5%	0.060	0.165	0.074	0.182	0.114	0.234	0.074	0.194	0.211	0.275	0.083	0.183	0.181	0.260	0.107	0.207
	25%	0.080	0.189	0.090	0.203	0.140	0.262	0.102	0.227	0.259	0.298	0.098	0.197	0.192	0.266	0.127	0.224
	37.5%	0.102	0.212	0.109	0.222	0.174	0.293	0.135	0.261	0.323	0.326	0.119	0.215	0.215	0.277	0.160	0.245
	50%	0.133	0.240	0.137	0.248	0.215	0.325	0.179	0.298	0.423	0.366	0.158	0.242	0.257	0.297	0.235	0.279
	Avg	0.094	0.201	0.103	0.214	0.161	0.279	0.122	0.245	0.304	0.317	0.114	0.209	0.211	0.275	0.157	0.239
ETTh2	12.5%	0.042	0.133	0.044	0.138	0.305	0.431	0.163	0.289	0.090	0.167	0.058	0.134	0.085	0.162	0.091	0.172
	25%	0.049	0.147	0.050	0.149	0.322	0.444	0.206	0.331	0.097	0.176	0.060	0.138	0.088	0.165	0.101	0.179
	37.5%	0.056	0.158	0.060	0.163	0.353	0.462	0.252	0.370	0.105	0.185	0.064	0.144	0.091	0.169	0.118	0.190
	50%	0.065	0.170	0.068	0.173	0.369	0.472	0.316	0.419	0.118	0.199	0.071	0.153	0.097	0.176	0.150	0.205
	Avg	0.053	0.152	0.055	0.156	0.337	0.452	0.234	0.352	0.102	0.182	0.063	0.142	0.090	0.168	0.115	0.186
ECL	12.5%	0.093	0.210	0.089	0.210	0.218	0.326	0.190	0.308	0.214	0.293	0.079	0.182	0.181	0.271	0.091	0.196
	25%	0.097	0.214	0.096	0.220	0.219	0.326	0.197	0.312	0.266	0.324	0.094	0.199	0.194	0.280	0.115	0.217
	37.5%	0.102	0.220	0.104	0.229	0.222	0.328	0.203	0.315	0.339	0.366	0.117	0.223	0.220	0.296	0.152	0.245
	50%	0.108	0.228	0.113	0.239	0.228	0.331	0.210	0.319	0.447	0.424	0.156	0.259	0.264	0.324	0.222	0.286
	Avg	0.100	0.218	0.101	0.225	0.222	0.328	0.200	0.313	0.316	0.352	0.112	0.216	0.215	0.293	0.145	0.236
Weather	12.5%	0.027	0.051	0.026	0.047	0.037	0.093	0.031	0.076	0.041	0.042	0.026	0.031	0.038	0.041	0.038	0.036
	25%	0.029	0.056	0.030	0.054	0.042	0.100	0.035	0.082	0.045	0.045	0.027	0.032	0.040	0.041	0.043	0.039
	37.5%	0.033	0.062	0.032	0.060	0.049	0.111	0.040	0.091	0.048	0.049	0.030	0.034	0.042	0.043	0.055	0.043
	50%	0.037	0.068	0.037	0.067	0.053	0.114	0.046	0.099	0.054	0.054	0.033	0.037	0.044	0.045	0.069	0.048
	Avg	0.032	0.059	0.031	0.057	0.045	0.104	0.038	0.087	0.047	0.048	0.029	0.034	0.041	0.043	0.051	0.042

Table 19: Results for the imputation task in the time-steps missing at random setting. Results averaged across 4 different masking rates: $\{12.5\%, 25\%, 37.5\%, 50\%\}$. Statistical interpolation methods such as forward and backward fill (naive), linear, nearest, and cubic interpolation perform better than many transformer-based baselines. Therefore, we consider the much harder, patches missing at random setting in our experiments.

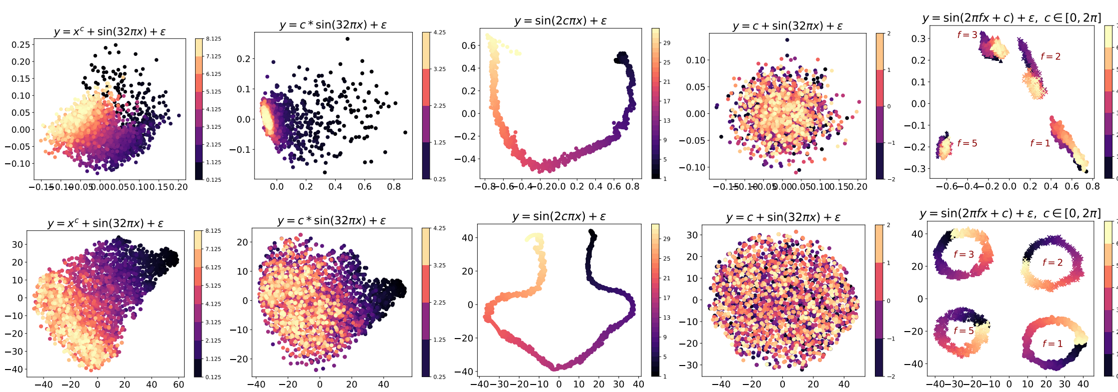
D.5 What is `MOMENT` Learning?

To investigate what `MOMENT` is learning, we conducted a series of experiments using synthetically generated sine waves to evaluate `MOMENT`'s ability to capture changes in trend, amplitude, frequencies, baselines, and phase of time-series. In each experiment, c controls the factor of interest, for example the power of the trend polynomial $c \in (\frac{1}{8}, 8)$ Oreshkin et al. [2020] (Fig. 8), and frequency $c \in (1, 32)$ of the generated sine waves (Fig. 8). We generate multiple sine waves by varying c, derive their sequence-level representations using `MOMENT` (Sec. 3.4), and visualize them in a 2- dimensional space using PCA and t-SNE van der Maaten [2014] in Fig. 4 and Fig. 6.

We also study the composition of the learnable mask embedding and the relationship between frequency and reconstruction error in a zero-shot setting. We find that the learned mask embedding is approximately composed of numbers drawn from the standard normal and that `MOMENT` can reconstruct lower frequency signals better. We observed a curious spike in reconstruction error around time-series of frequency $c = 64$. (Fig. 10)

Dataset	Mask Ratio	MOMENT$_0$		MOMENT$_{LP}$		GPT4TS		TimesNet		Naive		Linear		Nearest		Cubic	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	0.125	0.085	0.131	0.033	0.073	0.036	0.076	0.035	0.096	0.105	0.089	0.050	0.055	0.069	0.067	0.373	0.115
	0.250	0.079	0.130	0.036	0.078	0.030	0.071	0.037	0.100	0.127	0.104	0.075	0.065	0.094	0.076	0.297	0.125
	0.375	0.081	0.128	0.034	0.075	0.030	0.070	0.035	0.099	0.120	0.111	0.066	0.069	0.082	0.080	0.904	0.176
	0.500	0.081	0.129	0.035	0.075	0.026	0.069	0.035	0.098	0.124	0.127	0.066	0.078	0.086	0.090	0.831	0.197
	Mean	0.082	0.130	0.035	0.075	0.031	0.071	0.036	0.098	0.119	0.108	0.065	0.067	0.083	0.078	0.601	0.153
ETTh1	0.125	0.430	0.417	0.160	0.239	0.183	0.242	0.158	0.254	1.008	0.602	0.583	0.466	0.712	0.523	0.985	0.661
	0.250	0.373	0.392	0.142	0.238	0.278	0.267	0.154	0.261	1.311	0.686	0.833	0.540	0.954	0.581	1.433	0.772
	0.375	0.398	0.398	0.121	0.228	0.232	0.263	0.195	0.274	1.317	0.703	0.843	0.572	0.973	0.613	2.615	1.028
	0.500	0.408	0.403	0.132	0.231	0.213	0.243	0.192	0.267	1.103	0.643	0.840	0.559	0.963	0.601	3.681	1.204
	Mean	0.402	0.403	0.139	0.234	0.227	0.254	0.175	0.264	1.185	0.658	0.775	0.534	0.900	0.579	2.178	0.916
ETTh2	0.125	0.122	0.235	0.051	0.150	0.115	0.215	0.163	0.277	0.196	0.285	0.105	0.208	0.134	0.225	0.452	0.404
	0.250	0.127	0.237	0.079	0.177	0.114	0.216	0.161	0.280	0.210	0.291	0.120	0.220	0.154	0.240	0.831	0.524
	0.375	0.124	0.237	0.056	0.155	0.110	0.216	0.170	0.291	0.229	0.310	0.142	0.243	0.175	0.262	1.571	0.672
	0.500	0.127	0.242	0.056	0.154	0.098	0.207	0.186	0.295	0.265	0.329	0.171	0.264	0.199	0.279	4.823	0.966
	Mean	0.125	0.238	0.061	0.159	0.109	0.213	0.170	0.286	0.225	0.304	0.135	0.234	0.166	0.252	1.920	0.641
ETTm1	0.125	0.179	0.278	0.069	0.170	0.078	0.147	0.089	0.200	0.273	0.293	0.094	0.183	0.147	0.217	0.334	0.345
	0.250	0.206	0.290	0.071	0.169	0.071	0.144	0.080	0.194	0.395	0.341	0.114	0.202	0.171	0.234	0.539	0.424
	0.375	0.209	0.289	0.069	0.163	0.076	0.146	0.091	0.199	0.475	0.378	0.188	0.242	0.257	0.274	0.842	0.528
	0.500	0.215	0.294	0.086	0.169	0.081	0.149	0.088	0.197	0.679	0.448	0.265	0.291	0.346	0.316	1.715	0.680
	Mean	0.202	0.288	0.074	0.168	0.076	0.146	0.087	0.198	0.455	0.365	0.165	0.229	0.230	0.260	0.858	0.494
ETTm2	0.125	0.076	0.183	0.032	0.108	0.043	0.126	0.128	0.233	0.087	0.164	0.049	0.117	0.062	0.132	0.237	0.262
	0.250	0.084	0.187	0.029	0.105	0.046	0.129	0.101	0.207	0.104	0.182	0.057	0.132	0.073	0.146	0.373	0.309
	0.375	0.076	0.181	0.032	0.109	0.059	0.137	0.116	0.225	0.115	0.196	0.063	0.141	0.078	0.154	0.626	0.376
	0.500	0.077	0.183	0.031	0.110	0.059	0.140	0.103	0.212	0.144	0.222	0.080	0.162	0.102	0.177	0.899	0.477
	Mean	0.078	0.184	0.031	0.108	0.052	0.133	0.112	0.220	0.113	0.191	0.062	0.138	0.079	0.152	0.534	0.356
Electricity	0.125	0.251	0.370	0.095	0.211	0.069	0.180	0.126	0.248	1.350	0.818	0.458	0.466	0.608	0.492	0.924	0.610
	0.250	0.249	0.372	0.093	0.211	0.073	0.184	0.121	0.246	1.447	0.857	0.654	0.554	0.815	0.582	1.619	0.769
	0.375	0.250	0.371	0.094	0.211	0.071	0.181	0.125	0.248	1.518	0.888	0.836	0.637	1.031	0.675	2.507	0.959
	0.500	0.250	0.371	0.092	0.210	0.075	0.185	0.126	0.249	1.581	0.915	1.002	0.712	1.239	0.766	3.978	1.213
	Mean	0.250	0.371	0.094	0.211	0.072	0.183	0.124	0.248	1.474	0.869	0.737	0.592	0.923	0.629	2.257	0.888

Table 20: **Imputation Results.** MOMENT achieves state-of-the-art imputation results in both zero-shot and linear probe fine-tuning settings.



(i) Trend *(ii)* Amplitude *(iii)* Frequency *(iv)* Basline Shift *(v)* Auto-correlation

Figure 6: What is MOMENT learning? Structure in the PCA (top) and t-SNE (bottom) visualizations of the embeddings of synthetically generated sinusoids suggest that MOMENT can capture subtle trend, scale, frequency, and auto-correlation information. ϵ denotes gaussian noise with 0 mean and 0.1 standard deviation. c controls the factor of interest, i.e. the power of the trend polynomial, amplitude, and frequency of the sine waves in experiments (i), (ii) & (iii), respectively.

D.6 Impact of Model Size

We studied the impact of scaling the size of the model and training data on zero-shot forecasting, imputation, and anomaly detection performance. As shown in Fig. x, we found that increasing the size of the model generally improved zero-shot performance (lower MSE and sMAPE, higher VUS-ROC). Since varying the size of the pre-training dataset is expensive, we instead look at the zero-shot performance of model checkpoints before completing the first epoch. Our findings suggest that increasing the diversity in training data may also improve zero-shot performance.

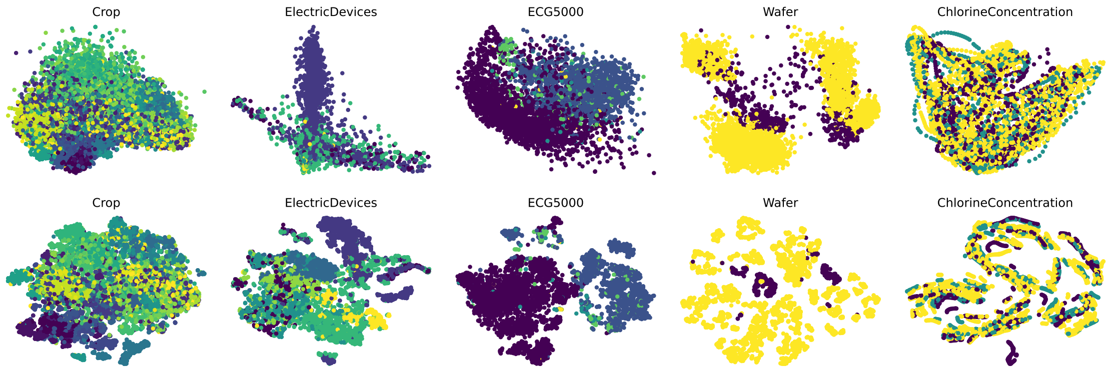


Figure 7: PCA (top) and t-SNE (bottom) visualizations of representations learned by MOMENT on the 5 largest UCR datasets. Different colors represent different classes. Even without dataset-specific fine-tuning, MOMENT learns distinct representations for different classes

D.7 Training losses

D.8 Efficiency Analysis

Model	**ETTh1-96** Total Param. (M)	Trainable Param. (M)	Mem. (MiB)
MOMENT	347.53	6.29	2079
GPT4TS	82.28	1.12	1031
TimesNet	0.89	0.89	683
Time-LLM	3623.71	254.37	4537

Table 21: Efficiency analysis of MOMENT against other forecasting models on the ETTh1 with prediction horizon set to 96. MOMENT outperforms all the listed models and has a fraction of parameters as the most recent LLM-based forecasting method.

E Transparency Index

F Results Sources

G Radar Plot

We generate a radar plot (Fig. 1) to visually compare MOMENT with GPT4TS and TimesNet. The values obtained by each method for a given task are min-max normalized with respect to the other methods for each of the 5 downstream tasks. For imputation, long- and short-horizon forecasting, we report $1-$ the normalized MSE or sMAPE for the methods on the weather and (subset of) M4 datasets, respectively. For classification and anomaly detection, we report the average accuracy and VUS-ROC of the methods across all the datasets.

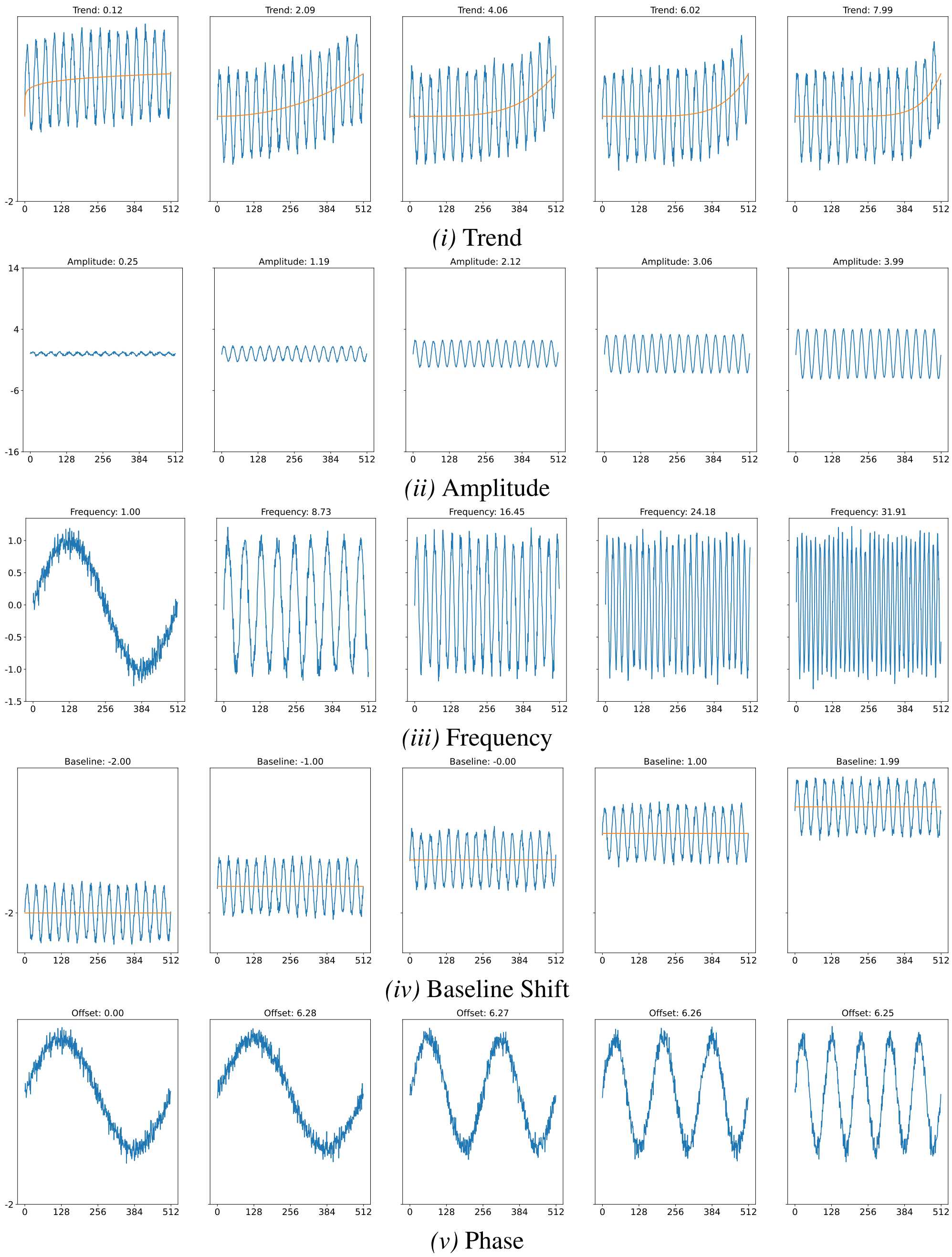


Figure 8: Examples of sinusoids used in the interpretability experiments.

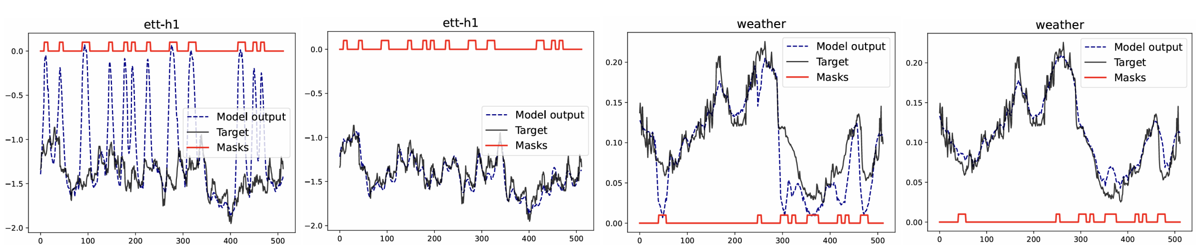


Figure 9: Masking using a `[MASK]` tokens allows `MOMENT` to reconstruct time-series in a zero-shot setting. Since zeros contain information, they bias model predictions. For two datasets ETTh1 and Weather, we mask the time-series with zeros on the left and special mask tokens on the right.

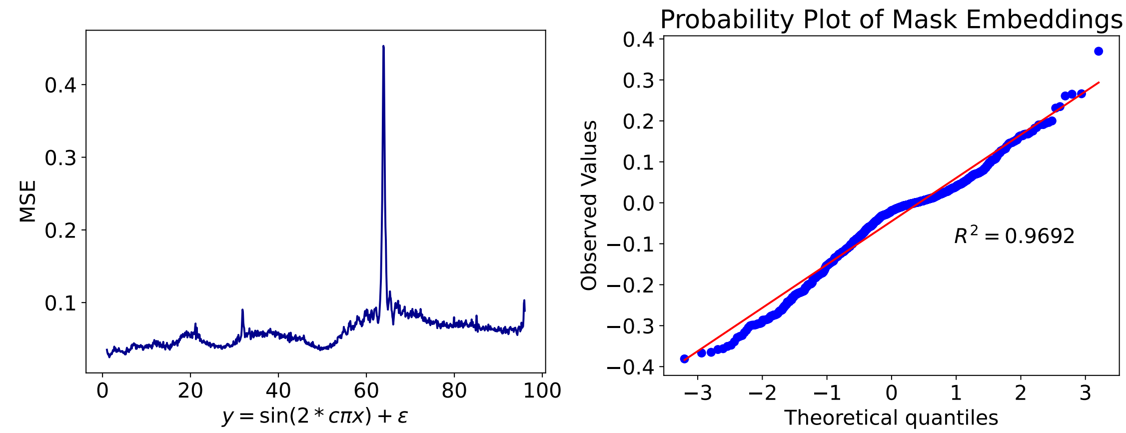


Figure 10: (*Left*) `MOMENT` can reconstruct lower frequency time-series better in a zero-shot setting. (*Right*) The learned mask token is approximately composed of numbers drawn from a standard normal.

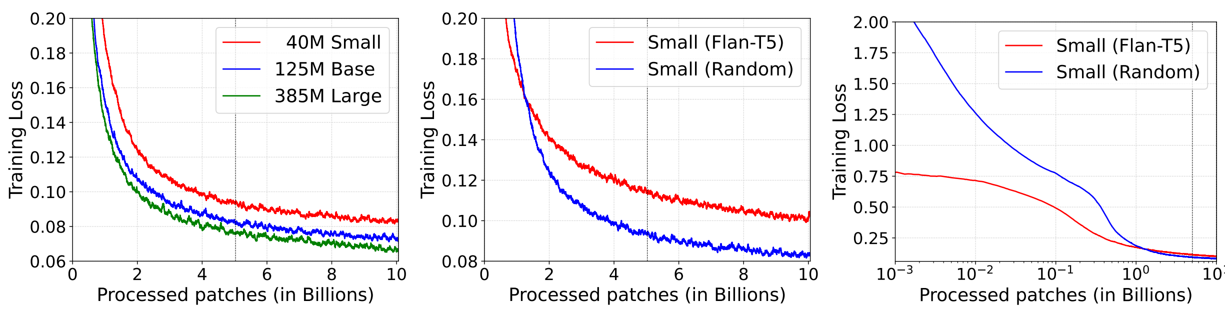


Figure 11: **Training losses (MSE).** A dashed vertical line denotes the first epoch. All models were trained with a batch size of 131072 patches. (*left*) Larger models obtain lower training loss. *right* Eventually, randomly initialized `MOMENT-small` outperform the same model initialized with Flan-T5 weights. The figure on the right is in log scale.

Sub-domain	Indicator	MOMENT
Data	Data size	1
	Data sources	1
	Data creators	0
	Data source selection	1
	Data curation	1
	Data augmentation	1
	Harmful data filtration	0
	Copyrighted data	1
	Data license	1
	Personal information in data	1
Data Labor	Use of human labor	1
	Employment of data laborers	1
	Geographic distribution of data laborers	1
	Wages	1
	Instructions for creating data	1
	Labor protections	1
	Third party partners	1
Data Access	Queryable external data access	1
	Direct external data access	1
Compute	Compute usage	1
	Development duration	1
	Compute hardware	1
	Hardware owner	1
	Energy usage	1
	Carbon emissions	1
	Broader environmental impact	0
Methods	Model stages	1
	Model objectives	1
	Core frameworks	1
	Additional dependencies	1
Data Mitigations	Mitigations for privacy	0
	Mitigations for copyright	0
	Upstream Subtotal	75%

Sub-domain	Indicator	MOMENT
Model Basics	Input modality	1
	Output modality	1
	Model components	1
	Model size	1
	Model architecture	1
	Centralized model documentation	1
Model Access	External model access protocol	1
	Blackbox external model access	1
	Full external model access	1
Capabilities	Capabilities description	1
	Capabilities demonstration	1
	Evaluation of capabilities	1
	External reproducibility of capabilities evaluation	0
	Third party capabilities evaluation	0
Limitations	Limitations description	1
	Limitations demonstration	1
	Third party evaluation of limitations	0
Risks	Risks description	1
	Risks demonstration	0
	Unintentional harm evaluation	0
	External reproducibility of unintentional harm evaluation	0
	Intentional harm evaluation	0
	External reproducibility of intentional harm evaluation	0
	Third party risks evaluation	0
Model Mitigations	Mitigations description	0
	Mitigations demonstration	0
Mitigations	Mitigations evaluation	0
	External reproducibility of mitigations evaluation	0
	Third party mitigations evaluation	0
Trustworthiness	Trustworthiness evaluation	0
	External reproducibility of trustworthiness evaluation	0
Inference	Inference duration evaluation	1
	Inference compute evaluation	1
	Model Subtotal	51.5%

Table 22: Expected (*left*) upstream and (*right*) model transparency scores. MOMENT has one of the highest upstream transparency. Our model transparency scores are lower due to (third-party) harm, mitigations, trustworthiness evaluation, which are not well understood for time-series modeling.

Task	Method	Type	Reimplementation/ Rerun	Source
Long-horizon Forecasting	Time-LLM	LLM-based	✓	Time-LLM
	GPT4TS	LLM-based	×	One Fits All
	PatchTST, Fedformer, Autoformer, Stationary, ETSformer, LightTS, Informer, Reformer	Transformer-based	×	One Fits All
	Pyraformer, LogTrans	Transformer-based	×	TimesNet
	TimesNet, DLinear	Deep learning	×	One Fits All
	N-BEATS	Deep learning	✓	N-BEATS
Short-horizon Forecasting	GPT4TS	LLM-based	✓	One Fits All
	TimesNet	Deep learning	✓	TimesNet
	N-BEATS	Deep learning	✓	N-BEATS
	AutoARIMA, AutoTheta, AutoETS, Seasonal Naive, Naive, Random Walk	Statistical learning	✓	Nixtla Statsforecast Repository
Classification	GPT4TS	LLM-based	✓	One Fits All
	TimesNet	Deep learning	✓	TimesNet
	TS2Vec, T-Loss, TNC, TS-TCC, TST	Unsupervised Representation learning	×	TS2Vec
	CNN, Encoder, FCN, MCNN, MLP, ResNet, t-LeNet, TWIESN	Deep learning	×	DL4TSC Repository
	DTW	Statistical learning	×	TS2Vec
Anomaly Detection	GPT4TS	LLM-based	✓	One Fits All
	TimesNet	Deep learning	✓	TimesNet
	Anomaly Transformer	Transformer-based	✓	Anomaly Transformer
	DGHL	Deep learning	✓	Time-Series Model Selection
	k-NN	Statistical learning	✓	Time-Series Model Selection
Imputation	GPT4TS	LLM-based	✓	One Fits All
	TimesNet	Deep learning	✓	TimesNet
	PatchTST, ETSformer, LightTS, Fedformer,Stationary, Autoformer, Informer, Reformer	Transformer-based	×	One Fits All
	DLinear	Deep learning	×	One Fits All
	Naive	Statistical learning	✓	Pandas FFill, Pandas BFill
	Linear, Nearest, Cubic	Statistical learning	✓	Scipy Interp1D

Table 23: Source for the results for each baseline for all downstream task.