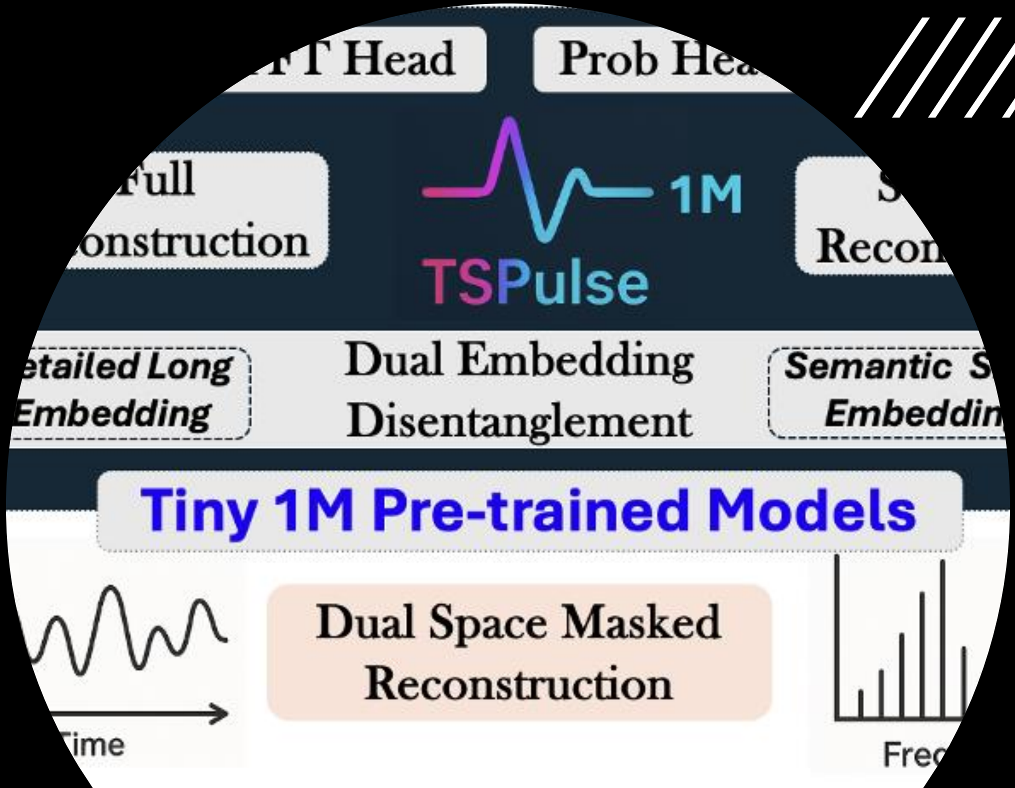
Granite Time Series Foundation Models

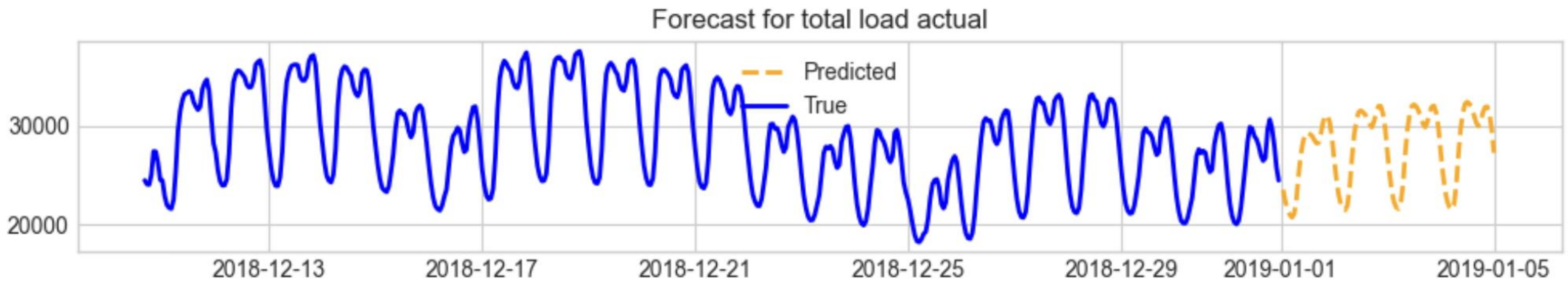


Presented @ AI Alliance Dev Meet. June 24, 2025
@ IBM IRL

Time-Series Analysis

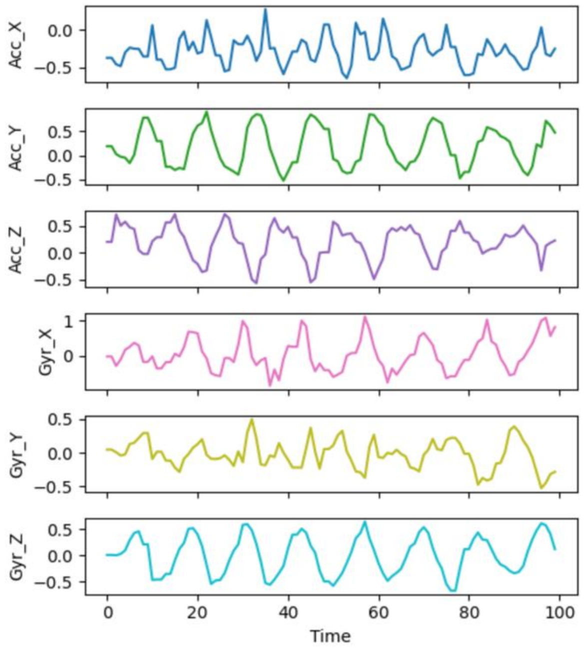
- Time-Series Analysis isn't all about Forecasting.
- It includes a wide-range of tasks and analysis that can be done on a time-series data.

1. Time –Series Forecasting

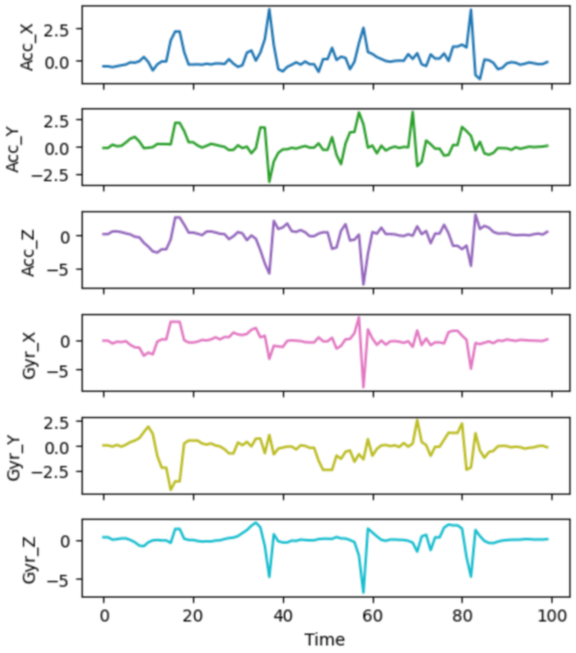


https://github.com/ibm-granite/granite-tsfm/blob/main/notebooks/recipes/energy_demand_forecasting/demand_forecast_zeroshot_recipe_minimal.ipynb

2. Time-Series Classification

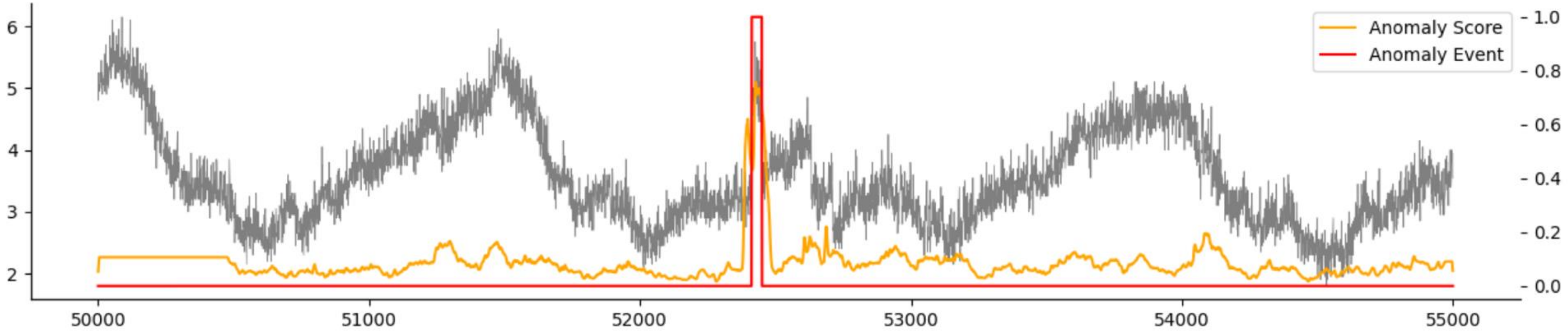


→ Walking



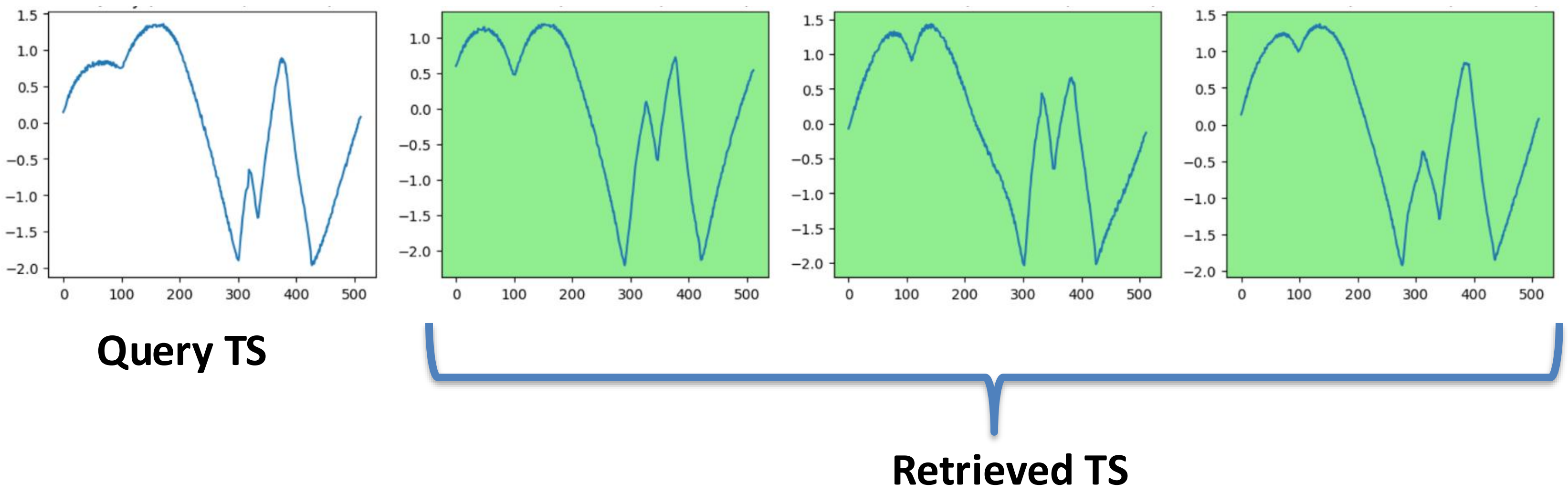
→ Playing Badminton

3. Time-Series Anomaly Detection

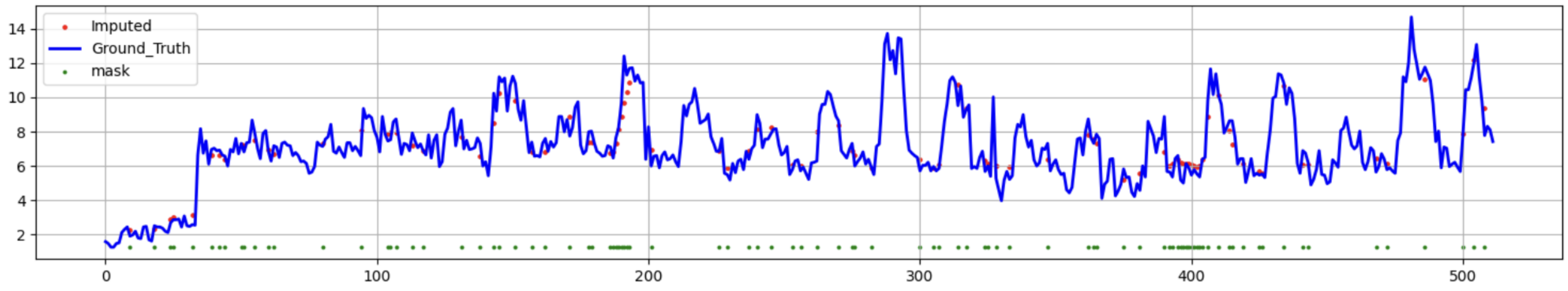


https://www.timeseriesclassification.com/description.php?Dataset=BasicMotions
https://github.com/ibm-granite/granite-tsfm/blob/main/notebooks/hfdemo/tspulse_anomaly_detection.ipynb

4. Time-Series Search



5. Time-Series Imputation



https://github.com/ibm-granite/granite-tsfm/blob/main/notebooks/hfdemo/tspulse_search_simple_example.ipynb
https://github.com/ibm-granite/granite-tsfm/blob/main/notebooks/hfdemo/tspulse_imputation.ipynb

IBM Granite Time Series Models

TSPULSE

First Light-weight FMs for
Anomaly Detection, Classification,
Imputation, Search

1M
TSPulse

TTM (NeurIPS 2024)

First Light-weight FMs for Time
Series Forecasting

PatchTSMixer (KDD '23)

Introducing Light-weight FM backbones

PatchTST (ICLR '23)

Unveiling Patch TS Transformers

2700+ citations

Open-Source Releases
Apache 2.0 License

Pretrained models on
Hugging Face
15M+ downloads

Model architecture and
supporting code available
on Granite-TSFM

500+	200+
Likes/stars	Forks

Ekambaram, V., Jati, A., Dayama, P., Mukherjee, S., Nguyen, N. H., Gifford, W. M., Reddy, C., & Kalagnanam, J. (2024). Tiny Time Mixers (TTMs): Fast Pre-trained Models for Enhanced Zero/Few-Shot Forecasting of Multivariate Time Series. Retrieved from [https://arxiv.org/abs/2401.0395, Accepted at NeurIPS 2024.

Ekambaram, V., Kumar, S., Jati, A., Mukherjee, S., Sakai, T., Dayama, P., Gifford, W.M., Kalagnanam, J. (2025). TSPulse: Dual Space Tiny Pre-Trained Models for Rapid Time-Series Analysis. arXiv preprint arXiv:2505.13033.

TTM Paper https://arxiv.org/pdf/2401.03955 (NeurIPS 24)

Tiny Time Mixers (TTMs): Fast Pre-trained Models for Enhanced Zero/Few-Shot Forecasting of Multivariate Time Series

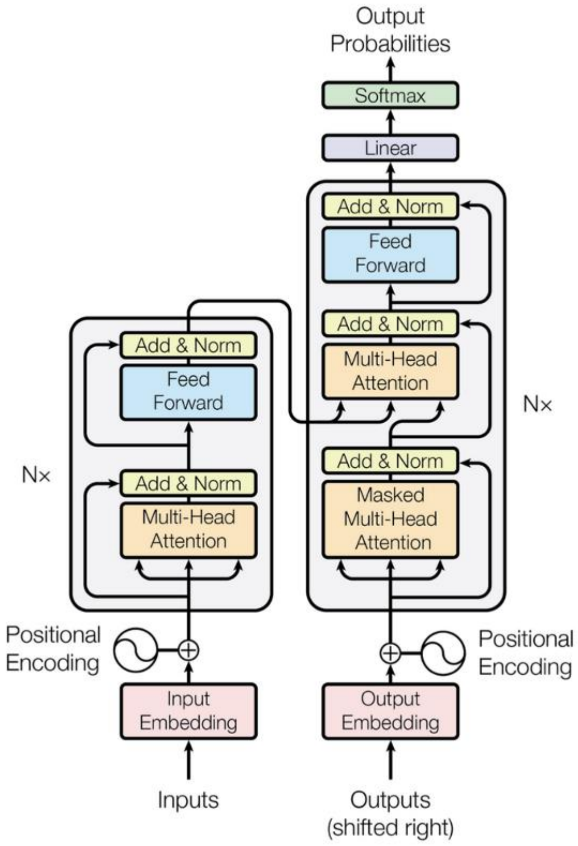
Vijay Ekambaram Arindam Jati Pankaj Dayama Sumanta Mukherjee
Nam H. Nguyen Wesley M. Gifford Chandra Reddy Jayant Kalagnanam

IBM Research*

Abstract

Large pre-trained models excel in zero/few-shot learning for language and vision tasks but face challenges in multivariate time series (TS) forecasting due to diverse data characteristics. Consequently, recent research efforts have focused on developing pre-trained TS forecasting models. These models, whether built from scratch or adapted from large language models (LLMs), excel in zero/few-shot forecasting tasks. However, they are limited by slow performance, high computational demands, and neglect of cross-channel and exogenous correlations. To address this, we introduce Tiny Time Mixers (TTM), a compact model (starting from 1M parameters) with effective transfer learning capabilities, trained exclusively on public TS datasets. TTM, based on the light-weight TSMixer architecture, incorporates innovations like adaptive patching, diverse resolution sampling, and resolution prefix tuning to handle pre-training on varied dataset resolutions with minimal model capacity. Additionally, it employs multi-level modeling to capture channel correlations and infuse exogenous signals during fine-tuning. TTM outperforms existing popular benchmarks in zero/few-shot forecasting by (4-40%), while reducing computational requirements significantly. Moreover, TTMs are lightweight and can be executed even on CPU-only machines, enhancing usability and fostering wider adoption in resource-constrained environments. Model weights for our initial variant TTM_Q are available here. Model weights for more sophisticated variants (TTM_B, TTM_E, and TTM_A) will be shared soon. The source code for TTM can be accessed here.

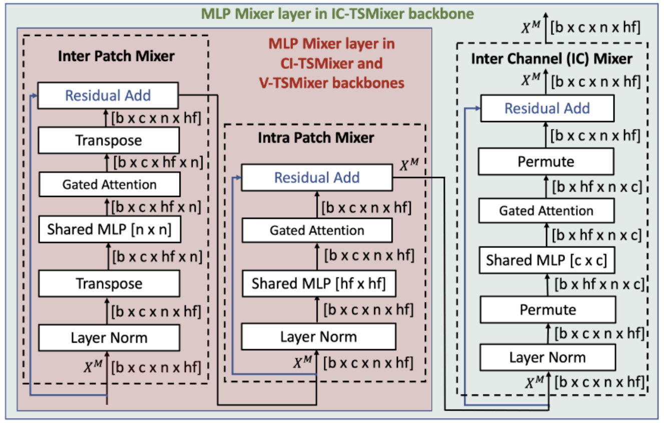
Transformer Backbone



Lighting Fast Pretraining and Finetuning with no compromise in accuracy

Model	GPU TIME (ms)	Params (M)	MEM (GB)	CPU TIME (s)
$\mathbf{TTM}_B$	**4.7**	**0.8**	**0.06**	**0.01**
Chronos_B (2024)	1395 (298X)	201 (251X)	16 (267X)	2340 (239KX)
Chronos_L (2024)	1393 (298X)	709 (886X)	41 (683X)	2352 (240KX)
Chronos_S (2024)	1386 (296X)	46 (58X)	6 (100X)	2349 (240KX)
Chronos_T (2024)	1389 (297X)	8 (10X)	2 (33X)	2504 (256KX)
GPT4TS (NeurIPS '23)	13.9 (3X)	87 (109X)	1.34 (36X)	0.3 (26X)
Lag-Llama (2024)	1619 (346X)	2.4 (3X)	0.2 (3X)	37.5 (3830X)
Moirai_S (ICML '24)	205 (44X)	14 (18X)	0.1 (2X)	1.4 (141X)
Moirai_L (ICML '24)	693 (148X)	311 (389X)	2 (33X)	10.5 (1070X)
Moirai_B (ICML '24)	335 (72X)	91 (114X)	1 (17X)	4.1 (421X)
Moment-L (ICML '24)	88 (19X)	348 (435X)	8 (133X)	1.4 (144X)
TimesFM (ICML '24)	24 (5X)	200 (250X)	2 (33X)	0.4 (46X)

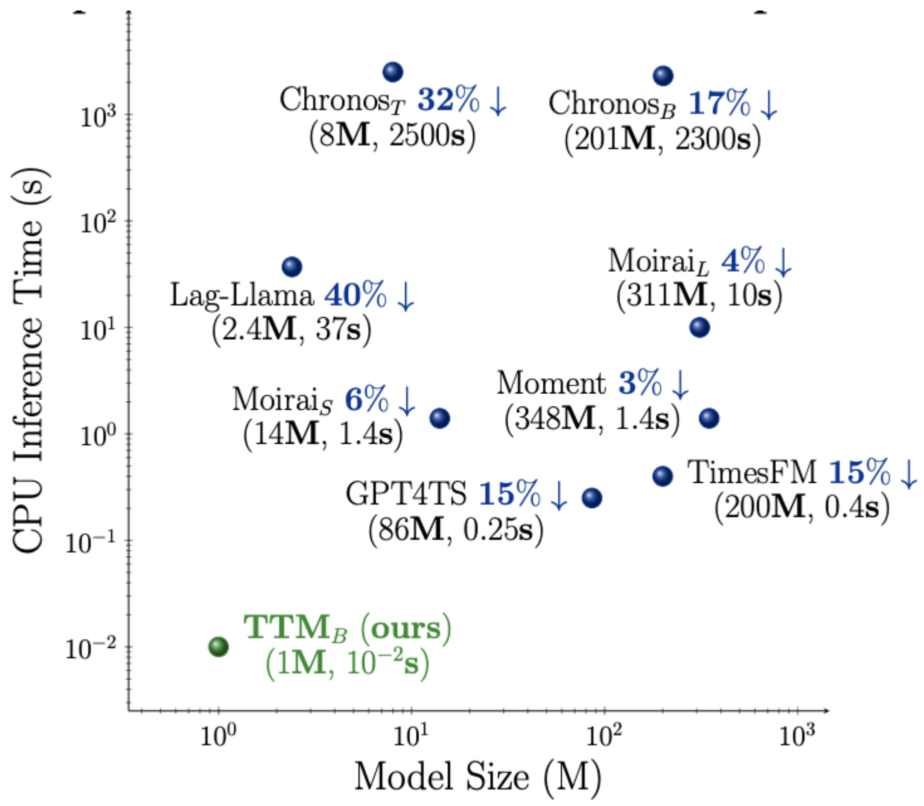
TTM is based on Light-Weight TSMixer Backbone



TTM provides state-of-the-art zero-shot forecasts and can easily be fine-tuned for multi-variate forecasts with just 5% of the training data to be competitive.

Note that zeroshot, fine-tuning and inference tasks using TTM can easily be executed on 1 GPU or on laptops.

Granite Time-Series TTM - The Power of Extremely Small Foundation Models



- TTM outperforms state of the art benchmarks.
- TTM is significantly smaller in size than most of the Transformer-based models (from 1M parameters)
- Quick Finetuning and Inference support (supports CPU finetuning too).
- TTM has significantly faster runtime and less memory footprint (no OOMs!!)

Getting Started with TTMs

- All the different sets of Pre-trained TTMs models are available on .
- TTMs support wide range of settings of context lengths from **52-1536** and forecasting lengths **16-720**.
- Automatic 'get_model()' takes care of selecting the best model for any arbitrary context and forecasting length given.
- All the public code to get started with using pre-trained TTM models for any forecasting task is available at the **granite_tsfm** repo.

DEMO : Getting Started with TTM

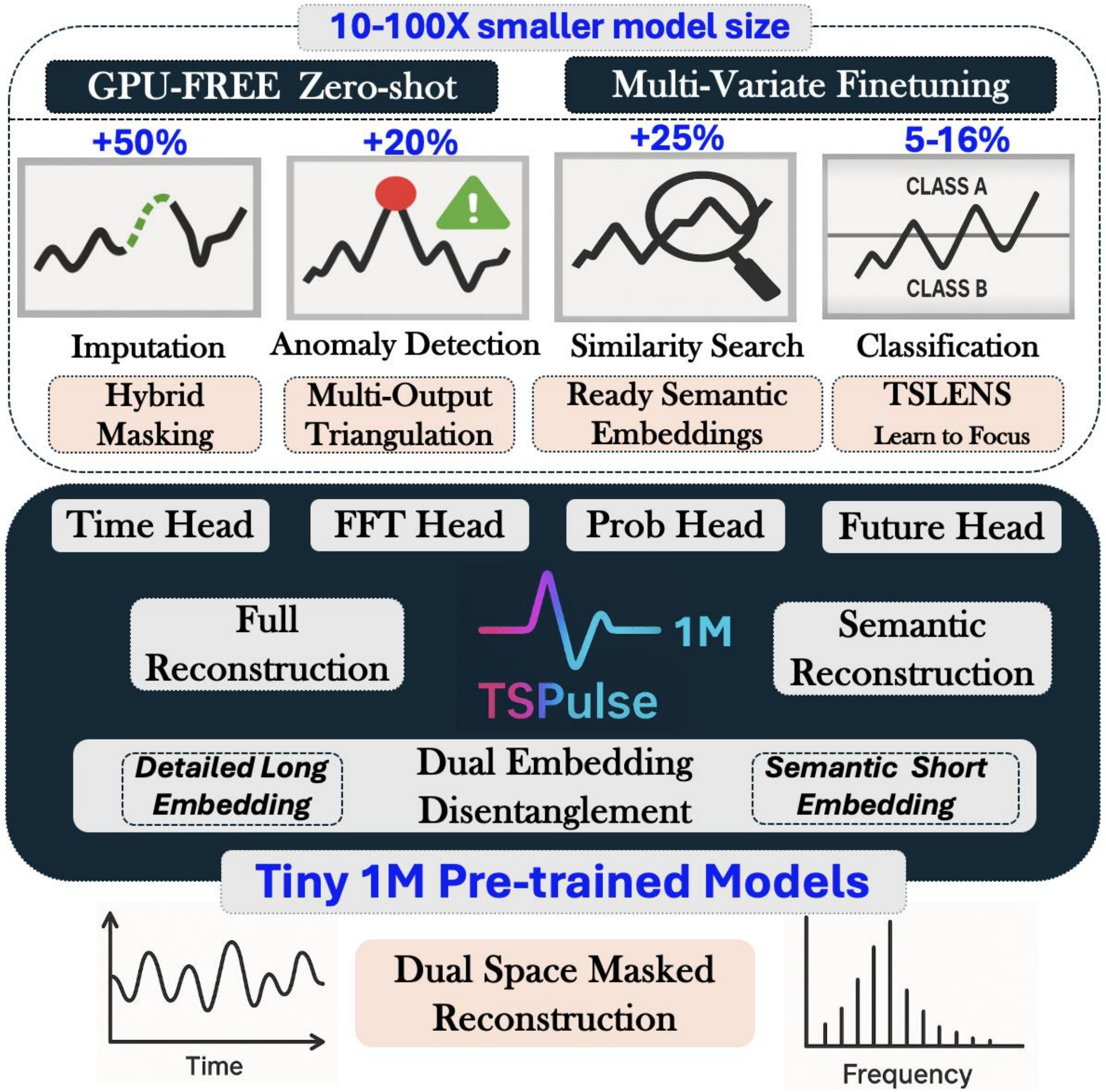
TSPulse Paper https://arxiv.org/pdf/2505.13033

TSPulse: Dual Space Tiny Pre-Trained Models for Rapid Time-Series Analysis

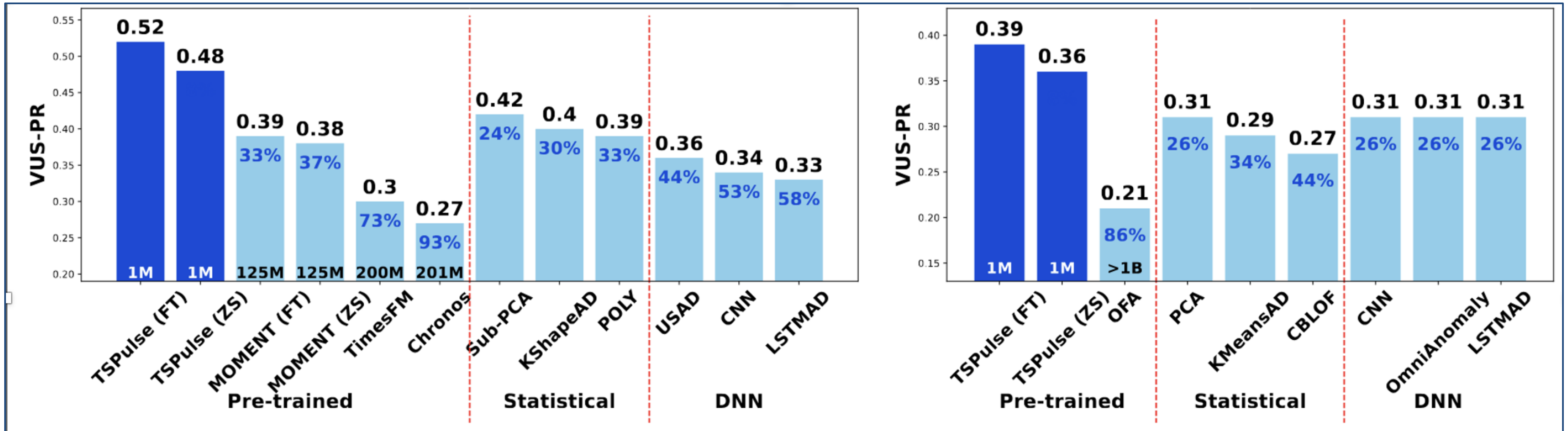
Vijay Ekambaram, Subodh Kumar*, Arindam Jati*, Sumanta Mukherjee†, Tomoya Sakai†, Pankaj Dayama, Wesley M. Gifford, Jayant Kalagnanam
IBM Research
vijaye12@in.ibm.com

Abstract

The rise of time-series pre-trained models has significantly advanced temporal representation learning, but current state-of-the-art models are often large-scale, requiring substantial compute. In this work, we introduce TSPulse, ultra-compact time-series pre-trained models with only 1M parameters, specialized to achieve strong performance across classification, anomaly detection, imputation, and retrieval tasks. TSPulse introduces innovations at both the architecture and task levels. At the architecture level, it employs a dual-space masked reconstruction strategy, learning from both time and frequency domains to capture complementary signal structures in a unified embedding space. This is further enhanced by a dual-embedding disentanglement approach, generating both detailed embeddings for fine-grained analysis and high-level semantic embeddings for broader task understanding. Notably, TSPulse's semantic embeddings are designed to be robust to shifts in time, magnitude, and noise, which is fundamental for time-series semantic search. At the task level, TSPulse incorporates TSLens, a fine-tuning component that enables task-specific feature extraction for classification. It also introduces a multi-head triangulation mechanism, leveraging reconstruction deviations from multiple prediction heads, enhancing anomaly detection by fusing complementary model outputs. Additionally, a hybrid masking strategy improves zero-shot imputation by reducing pre-training mask bias. These architecture and task innovations collectively contribute to TSPulse's significant performance improvements: 5-16% on the UEA classification benchmarks, +20% on the TSB-AD anomaly detection leaderboard, +50% in zero-shot imputation, and +25% in time-series retrieval. Remarkably, these results are achieved with just 1M parameters, making TSPulse 10-100x smaller than existing pre-trained models. Its efficiency enables GPU-free inference, rapid pre-training, and seamless task specialization, setting a new standard for efficient time-series pre-trained models. Models and source code will be open-sourced soon.



- TSPulse is also based on Light-Weight MLP mixer architecture.
- Tiny pre-trained models (~1M parameters) capable of multiple Time-Series Tasks : **Classification, Anomaly Detection, Imputation and Similarity Search.**
- **Out-performing** other **massive** SOTA pre-trained models/ supervised models on all the different tasks.



(a) **TSB-AD-U**: Univariate AD Performance (b) **TSB-AD-M**: Multivariate AD Performance

Figure 4: VUS-PR (official metric, higher is better) scores of different AD methods on the TSB-AD leaderboard; IMP(%)—the percentage improvement of TSPulse over baselines; .

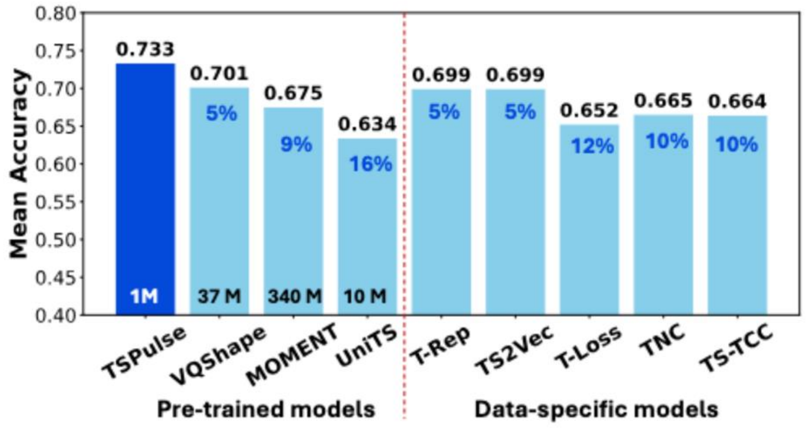


Figure 5: Classification Mean Accuracy results (higher is better); IMP(%)—the percentage improvement of TSPulse over baselines.

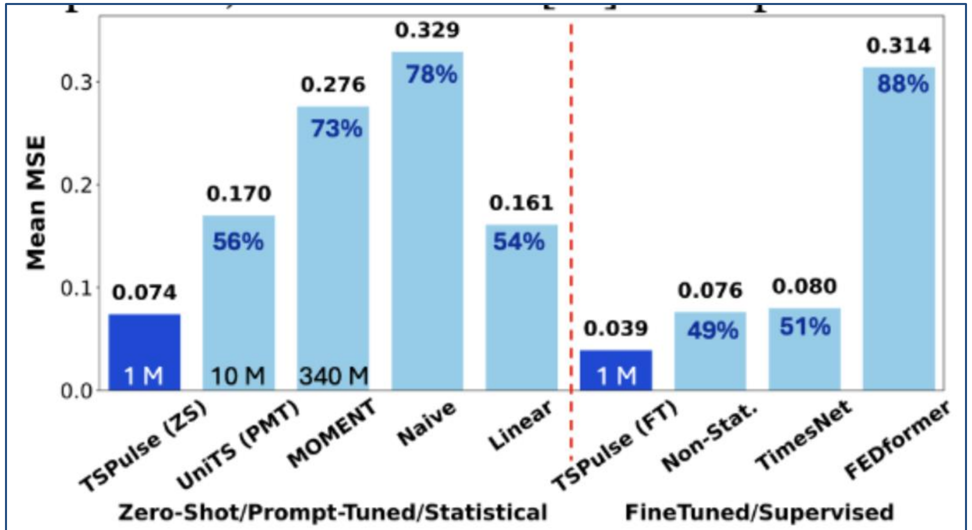
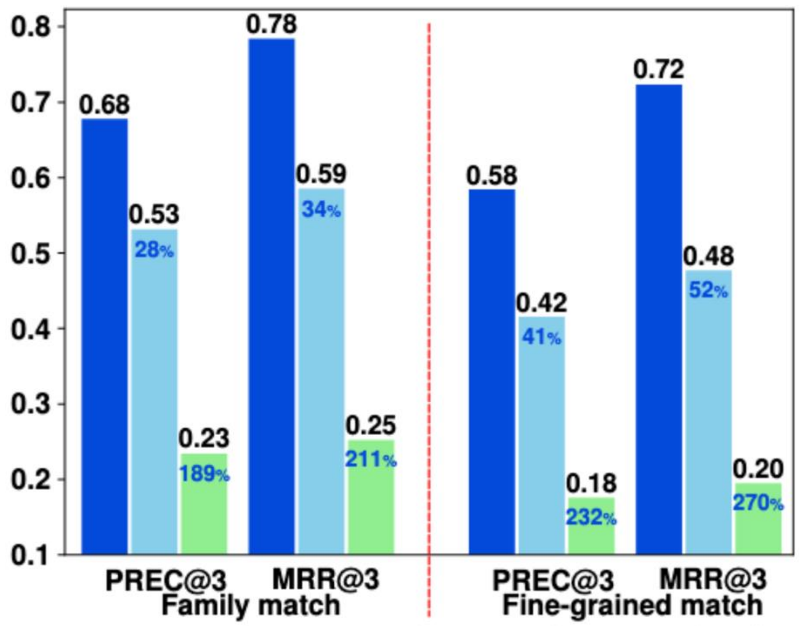


Figure 6: Hybrid Masking imputation MSE results (Lower is better). IMP(%)—the percentage improvement of TSPulse over baselines.



	CPU [ms]	GPU [ms]	Emb. dim.	Size
TSPulse	0.387	0.050	240	1M
MOMENT	5.51 (14X)	0.46 (9X)	512	40M
Chronos	40.71 (105X)	0.77 (15X)	512	46M

Figure 7: Similarity Search results using zero-shot embeddings (Higher is better). IMP(%)—the percentage improvement of TSPulse over baselines.

Getting Started with TSPulse

- Pre-trained TSPulse models are released and available on 🤗 .
- Specialized Pre-trained models can be accessed from 🤗 for different time-series analysis tasks.

```python
from tsfm_public.models.tspulse import TSPulseForReconstruction

# ✅ For Imputation and Semantic Search
model = TSPulseForReconstruction.from_pretrained(
    "ibm-granite/granite-timeseries-tspulse-r1",
    revision="tspulse-hybrid-dualhead-512-p8-r1"
)

# ✅ For Anomaly Detection
model = TSPulseForReconstruction.from_pretrained(
    "ibm-granite/granite-timeseries-tspulse-r1",
    revision="main"
)

# ✅ For Classification
from tsfm_public.models.tspulse import TSPulseForClassification

model = TSPulseForClassification.from_pretrained(
    "ibm-granite/granite-timeseries-tspulse-r1",
    revision="tspulse-block-dualhead-512-p16-r1"
)
```

- All the public code to get started with using pre-trained TSPulse models for all the different tasks is available at the granite_tsfm repo.

DEMO : Classification with TSPulse

DEMO : Anomaly-Detection with TSPulse

DEMO : Imputation with TSPulse

DEMO : Similarity Search with TSPulse

TTM : 🤗 Model Card

Granite_TSFM public repository

TSPulse : 🤗 Model Card