MoTime: A Dataset Suite for Multimodal Time Series Forecasting

Seminar Presentation

Xin Zhou, Weiqing Wang, Francisco J. Baldán, Wray Buntine, Christoph Bergmeir

16.12.2025

Presenter: Xinyi Cai
Heidelberg University

Content

Key Message

MoTime focuses on building a systematic benchmark for multimodal time series forecasting, rather than proposing a brand-new model.

Motivation

Motivation:

- **Time series forecasting underpins many real applications**
  - Finance, energy, retail, web/media, public health, etc.

Motivation:

- **Time series forecasting underpins many real applications**
  - Finance, energy, retail, web/media, public health, etc.
- **But real-world series are entity-centric and come with rich static context**
  - Products: titles, categories, descriptions, images
  - Movies & media items: plots, genres, tags, cast
  - News & pages: headlines, topics, summaries, sentiment, metadata

Motivation:

- **Time series forecasting underpins many real applications**
  - Finance, energy, retail, web/media, public health, etc.
- **But real-world series are entity-centric and come with rich static context**
  - Products: titles, categories, descriptions, images
  - Movies & media items: plots, genres, tags, cast
  - News & pages: headlines, topics, summaries, sentiment, metadata
- **Unimodal forecasting struggles with "what is this series?"**
  - Different items (portable fan vs blind box) can have almost identical demand curves for a long time
  - Their future diverges due to product attributes or lifecycle, which is invisible in the numeric series alone

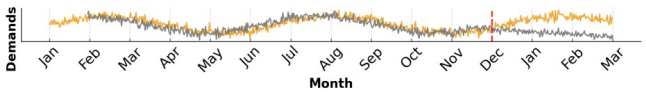

Figure 1: Monthly demand of a seasonal product, portable fan (orange curve), and a trend-sensitive product, blind box (blue curve), from Jan 2023 to Mar 2024. While both series follow similar patterns in 2023, they diverge in December (red dashed line) when the blind box demand drops, highlighting the limitations of unimodal forecasting without contextual information.

Motivation:

- **Time series forecasting underpins many real applications**
  - Finance, energy, retail, web/media, public health, etc.
- **But real-world series are entity-centric and come with rich static context**
  - Products: titles, categories, descriptions, images
  - Movies & media items: plots, genres, tags, cast
  - News & pages: headlines, topics, summaries, sentiment, metadata
- **Unimodal forecasting struggles with "what is this series?"**
  - Different items (portable fan vs blind box) can have almost identical demand curves for a long time
  - Their future diverges due to product attributes or lifecycle, which is invisible in the numeric series alone
- **Existing benchmarks are not enough**
  - Large suites & foundation models mostly use *unimodal* time series only
  - Multimodal datasets are small, domain-specific, or focus on dynamic event text and non-forecasting tasks

MoTime's goal

- Provide a large, entity-level multimodal dataset suite
- Pair time series with static text / images / metadata
- Enable systematic evaluation of modality utility under *varying history* and *cold-start* forecasting

Research Questions

- **RQ1. Uni vs. Multi**
  On different datasets, *how much* can external modalities (text / image / metadata) actually improve forecasting accuracy?
- **RQ2. Short vs. Long History**
  When history is very short, are multimodal signals more helpful than for long histories?
  - *Hypothesis:* with short history, the time-series signal is weak, so external context becomes more crucial.
- **RQ3. Cold-start**
  In a *pure* cold-start setting with **no historical series**, can we do meaningful forecasting using only a textual description plus an LLM?

Research Questions

- **RQ1. Uni vs. Multi**
  On different datasets, *how much* can external modalities (text / image / metadata) actually improve forecasting accuracy?
- **RQ2. Short vs. Long History**
  When history is very short, are multimodal signals more helpful than for long histories?
  - *Hypothesis:* with short history, the time-series signal is weak, so external context becomes more crucial.
- **RQ3. Cold-start**
  In a *pure* cold-start setting with **no historical series**, can we do meaningful forecasting using only a textual description plus an LLM?
- **Motivation for MoTime:**
  Build a **systematic benchmark** that allows us to study these research questions across datasets and scenarios.

MoTime: Datasets and Forecasting Tasks

MoTime: Datasets and Forecasting Tasks – Overview

Dataset Suite

- **8 multimodal time series datasets**, repurposed from large-scale real platforms
  - Domains: *e-commerce, media, web traffic, user behavior*

Entity-centric multimodal data

- Each *entity* (product, movie, news article, wiki page, etc.) includes:
  - **A univariate numeric time series**
    - e.g., daily demand, views, ratings, interactions
  - **Static external modalities**
    - **Text** (titles, descriptions, summaries)
    - **Images** (product or poster images)
    - **Structured metadata** (categories, attributes, tags)

MoTime: Datasets and Forecasting Tasks – Overview

Dataset Suite

- **8 multimodal time series datasets**, repurposed from large-scale real platforms
  - Domains: *e-commerce, media, web traffic, user behavior*

Entity-centric multimodal data

- Each *entity* (product, movie, news article, wiki page, etc.) includes:
  - **A univariate numeric time series**
    - e.g., daily demand, views, ratings, interactions
  - **Static external modalities**
    - **Text** (titles, descriptions, summaries)
    - **Images** (product or poster images)
    - **Structured metadata** (categories, attributes, tags)

Time series characteristics

- Wide variety of sequence lengths (short $\rightarrow$ long)
- Different sparsity patterns (dense vs intermittent)
- Multiple temporal resolutions (minutes, hours, days)

MoTime: Datasets and Forecasting Tasks – Overview

Two key forecasting scenarios

1. Varying-history forecasting

- Same forecasting task, but each entity has **different available history lengths**
- Designed to study *when external modalities compensate for limited history*

MoTime: Datasets and Forecasting Tasks – Overview

Two key forecasting scenarios

1. Varying-history forecasting

- Same forecasting task, but each entity has **different available history lengths**
- Designed to study *when external modalities compensate for limited history*

2. Cold-start forecasting

- The target entity has **no history at all**
- Forecast generated from:
  - **Entity description + retrieved similar entities + LLM reasoning**
- Models must rely on **semantics**, not past observations

MoTime: Datasets and Forecasting Tasks – Overview

Purpose of MoTime

- Provide the **first systematic benchmark** for multimodal TS forecasting
- Enable evaluation of **modality usefulness** across:
  - different **datasets**,
  - different **history lengths**, and
  - **cold-start** conditions

MoTime Datasets: Scale and Modalities

Table 2: Statistics of the eight multimodal time series datasets in MoTime.

Dataset	TS Shape	Density(%)	Text	Image	Metadata	Notes
PixelRec	4,865 × 43,082	4.41	✓	✓	✓	Long sparse multivariate TS
TaobaoFashion	365 × 890	68.01	✓	✓	–	One image per item
MovieLens	10,505 × 84,518	1.66	✓	–	✓	Text scraped externally
AmazonReview	3,934 × 668,756	6.18	✓	–	✓	29 categories, sparse TS
Tianchi	365 × 36,397	53.15	–	–	–	E-commerce purchase logs
News	144 × 26,612	17.61	✓	–	✓	20-min interval resolution
WikiPeople	550 × 3,856	99.96	–	–	✓	Multichannel access modes
VISUELLE	11 × 5,355	62.48	✓	✓	✓	Irregular time series

- The suite covers **long vs short** series, **sparse vs dense** series, and **single- vs multi-channel** series.
- Modality combinations are diverse: *text + image*, *text + meta*, *meta only*, etc.
- This diversity is crucial for evaluating *when* each modality is helpful.

MoTime Task Setting 1: Varying-History Forecasting

Content:

- For each item i, we have a time series $\{\mathbf{x}_{i,t}\}_{t=1}^{T_i}$ (possibly multi-channel).
- Unified forecasting setup:
  given a past window of length L, $\mathbf{x}_{i,t-L+1:t}$,
  predict the next H steps $\mathbf{y}_{i,t+1:t+H}$.
- Each original series is split into two subsets:
  - **Long history**: retaining the full historical windows
  - **Short history**: using only a few of each series.

MoTime Task Setting 1: Varying-History Forecasting

Content:

- For each item i, we have a time series $\{\mathbf{x}_{i,t}\}_{t=1}^{T_i}$ (possibly multi-channel).
- Unified forecasting setup:
  given a past window of length L, $\mathbf{x}_{i,t-L+1:t}$,
  predict the next H steps $\mathbf{y}_{i,t+1:t+H}$.
- Each original series is split into two subsets:
  - **Long history**: retaining the full historical windows
  - **Short history**: using only a few of each series.
- **Training:**
  - Train **jointly** on both short-history and long-history samples in the training set.
- **Evaluation:**
  - Report performance **separately** on
    - *short* subset
    - *long* subset
    - Combined mixture

MoTime Task Setting 2: Cold-start Forecasting

- **What is the retrieval base?**
  - A **collection of all non–cold-start entities** in MoTime
  - Each entity contributes:
    - its historical **time series**
    - its **text** description (titles, summaries, etc.)
    - **image captions** (from a VLM) and **metadata** when available
- **Why do we need it?**
  - Acts as a **semantic memory** of previously observed behaviors
  - Cold-start entities can find **similar neighbors** here
- **How is it built?**
  1. Use a **frozen LLM / VLM encoder** (GPT-4o-mini in the paper) to embed each entity's text / image into a vector $\mathbf{e}_i$
  2. Store $\mathbf{e}_i$ together with the entity ID and its time series
  3. Build a **semantic index** over $\{\mathbf{e}_i\}$ (cosine similarity search / k-NN)

MoTime Task Setting 2: Cold-start Forecasting

- **Goal**
  - For a **new entity with no historical time series**, forecast the next H steps.
- **Inputs**
  - Textual description of the **target entity**
    - e.g., product title/description, news headline
  - The **retrieval base** (from previous slide):
    - embeddings + time series of all other entities in the dataset

MoTime Task Setting 2: Cold-start Forecasting

- **Pipeline**
  1. **Encode the target**
     - Use the same frozen LLM / VLM encoder to get target embedding $\mathbf{e}_{\text{target}}$.
  2. **Retrieve neighbors**
     - Search the retrieval base with $\mathbf{e}_{\text{target}}$.
     - Select top-k most similar entities by cosine similarity.
  3. **Construct a structured prompt**
     - Target entity description
     - Text descriptions of the k retrieved entities
     - Their historical time series up to the prediction start time
     - IDs and timestamps
  4. **LLM-based generation**
     - Feed the structured prompt into an LLM.
     - LLM outputs a forecast vector $[v_1, \ldots, v_H]$ (plus a short reasoning explanation).
- **Baselines**
  - Compare against a simple **average-of-neighbors** baseline:
    - element-wise average of the retrieved entities' future values.
    - Same retrieval (LLM embeddings), but no LLM generation.

Models & Multimodal Extensions

Models & Multimodal Extensions

Unimodal baselines: representative time-series backbones

Model	Type	Keywords	Role in this paper
DLinear	Linear decomposition	trend / seasonal split, linear head	Strong but simple **linear** baseline
PatchTST	Transformer for TS patches	patching + TS encoder	Representative **Transformer-based** TS encoder
WPMixer	Wavelet + MLP-Mixer	multi-scale, local components	SOTA **wavelet + MLP-Mixer** model, captures local structure

- These are **existing models** chosen as **representative backbones**. The main contribution of *MoTime* is the **dataset & multimodal evaluation**, not proposing yet another forecasting architecture.

Models : a linear decomposition baseline (DLinear)

Models : a linear decomposition baseline (DLinear)

Main idea: Decompose each input series into trend and seasonal parts and forecast each part with a separate linear layer.

- **Model form**
  - Input window: $\mathbf{X} \in \mathbb{R}^{L \times C}$ (length L, C channels)
  - Trend: $T =$ moving-average smoothing of $\mathbf{X}$
  - Seasonal: $S = \mathbf{X} - T$
  - Forecast horizon H:
    $$\hat{\mathbf{Y}} = W_{\text{trend}}\, T \;+\; W_{\text{seasonal}}\, S$$
    where $W_{\text{trend}}, W_{\text{seasonal}} \in \mathbb{R}^{H \times L}$ are **linear projections**.
- **Characteristics**
  - Entirely **linear** operations, **no latent embedding**, **no attention**, **no MLP**
  - Very few parameters $\Rightarrow$ fast, stable, strong on many TS benchmarks
  - Not easy to extend to multimodal inputs (no natural "representation vector" to concatenate with text / image)

Models: patch-based Transformer for time series(PatchTST)

Models: patch-based Transformer for time series(PatchTST)

- **Input**
  - Multivariate time series $\mathbf{X} \in \mathbb{R}^{L \times C}$ (length L, C channels)

- **Processing pipeline**
  1. **Patching along time**
     - Split the time axis into non-overlapping patches of length $P \Rightarrow N_p = \lfloor L/P \rfloor$ patches.
  2. **Patch embedding**
     - Each patch has shape $P \times C$; flatten to a vector and project:

$$\mathbf{e}_p = W_{\text{emb}} \, \text{vec}(\text{patch}_p) + \mathbf{b} \in \mathbb{R}^d$$

  3. **Add position & form sequence**
     - Stack embeddings into $\mathbf{E} \in \mathbb{R}^{N_p \times d}$ and add positional encodings.
  4. **Transformer Encoder**
     - Pass $\mathbf{E}$ through several Transformer encoder layers (self-attention mixes information **across patches**).
  5. **Prediction head**
     - Take the final patch representations (e.g., all patches or pooled) and apply an MLP head to predict the next H steps $\hat{\mathbf{Y}}$.

Models: WPMixer(Wavelet + MLP-Mixer for multi-scale time series)

WPMixer: Wavelet + MLP-Mixer for multi-scale time series

- **High-level idea**
  - Combine **multi-scale wavelet decomposition** with a **patch + MLP-Mixer** backbone.
  - Capture both long-term trend and local spikes, **without attention**.

- **Step 1 – Multi-scale wavelet decomposition**
  - For each channel, apply a multi-level wavelet transform independently.
  - Obtain M sub-series at different scales:

$$\{S^{(m)}\}_{m=1}^{M}$$

  - Each scale has a shorter length $L_m < L$ due to down-sampling.

WPMixer: Wavelet + MLP-Mixer for multi-scale time series

- **Step 1 – Multi-scale wavelet decomposition**

- **Step 2 – Patch + linear embedding for each scale**
  - For a given scale $S^{(m)}$:
    - Cut the temporal axis into patches of length P, as in PatchTST $\Rightarrow N_p$ patches.
    - Flatten each patch and project to a d-dimensional embedding:

$$e_p^{(m)} = W_{\text{emb}}\,\text{vec}(\text{patch}_p^{(m)}) + b$$

    - Stack them into an embedding tensor

$$E^{(m)} \in \mathbb{R}^{C \times N_p \times d}$$

WPMixer: Wavelet + MLP-Mixer for multi-scale time series

- **Step 1 – Multi-scale wavelet decomposition**

- **Step 2 – Patch + linear embedding for each scale**

- **Step 3 – MLP-Mixer blocks**
  - **Patch-Mixer:**
    - View $E^{(m)}$ as (C, N_p, d).
    - For each channel, apply a 2-layer MLP **along the patch dimension** $N_p \Rightarrow$ mixes information across different time positions (patches).
  - **Embedding-Mixer:**
    - For each patch, apply another 2-layer MLP **along the feature dimension** $d \Rightarrow$ recombines embedding features within a patch.
  - Stack several Mixer blocks with residual connections $\Rightarrow$ final representation for this scale: $H^{(m)}$

WPMixer: Wavelet + MLP-Mixer for multi-scale time series

- **Step 1 – Multi-scale wavelet decomposition**

- **Step 2 – Patch + linear embedding for each scale**

- **Step 3 – MLP-Mixer blocks**

- **Step 4 – Multi-scale aggregation & prediction**
  - Aggregate $\{H^{(m)}\}_{m=1}^{M}$ from all scales (e.g., concatenate or sum across scales).
  - Feed the aggregated representation into a small MLP head $\Rightarrow$ predict the future H steps.

Models: MultiPatchTST(PatchTST + frozen context encoder)

MultiPatchTST: PatchTST + frozen context encoder

Idea:
Take a strong *time-series encoder* (PatchTST) and simply **concatenate** its representation with a frozen multimodal *context embedding*.

- **1. Time-series encoder (learned)**
  - Use the original PatchTST on the history window.
  - Get a time-series embedding

$$\mathbf{z}_{\text{ts}} \in \mathbb{R}^{d_{\text{ts}}}.$$

MultiPatchTST: PatchTST + frozen context encoder

Idea:Take a strong time-series encoder (PatchTST) and simply concatenate its representation with a frozen multimodal context embedding.

- **1. Time-series encoder (learned)**
- **2. Context encoder (frozen LLM / VLM)**
  - Text / tabular metadata $\rightarrow$ encode by a frozen LLM (e.g., GPT-4, BERT) $\rightarrow \mathbf{z}_{\text{text}}$.
  - Image (if any) $\rightarrow$ encode by a frozen VLM / CLIP $\rightarrow \mathbf{z}_{\text{img}}$.
  - Concatenate all available context to one vector

$$\mathbf{z}_{\text{ctx}} \in \mathbb{R}^{d_{\text{ctx}}}.$$

- **3. Fusion: concatenate**

$$\tilde{\mathbf{z}} = [\mathbf{z}_{\text{ts}}; \mathbf{z}_{\text{ctx}}] \in \mathbb{R}^{d_{\text{ts}}+d_{\text{ctx}}}.$$

- **4. Prediction head: small MLP**

$$\mathbf{h}_1 = \sigma(W_1\tilde{\mathbf{z}} + \mathbf{b}_1), \qquad \hat{\mathbf{Y}} = W_2\mathbf{h}_1 + \mathbf{b}_2.$$

Models: MultiWPMixer: (Wavelet + MLP-Mixer + Context)

MultiWPMixer: Wavelet + MLP-Mixer + Context

- **Recall WPMixer (unimodal)**
  - Wavelet DWT $\rightarrow$ multi-scale sub-series $S^{(m)}$
  - Each scale: patch + linear embedding $\rightarrow E^{(m)} \in \mathbb{R}^{C \times N_p \times d}$
  - Several Mixer blocks (Patch-Mixer + Embedding-Mixer) $\rightarrow$ scale representation $H^{(m)}$

- **MultiWPMixer: where does context enter?**
  1. **Context embedding (frozen, same as MultiPatchTST)**
     - Text / image / meta $\rightarrow$ LLM / VLM $\rightarrow$ global context vector

     $$\mathbf{z}_{\text{ctx}} \in \mathbb{R}^{d_{\text{ctx}}}$$

  2. **Scale-specific projection of context**
     - For each wavelet scale m, project context to the TS embedding dimension:

     $$\mathbf{b}_{\text{ctx}}^{(m)} = W^{(m)} \mathbf{z}_{\text{ctx}} \in \mathbb{R}^{d}$$

     - Broadcast $\mathbf{b}_{\text{ctx}}^{(m)}$ to all patches at scale m.

MultiWPMixer: Wavelet + MLP-Mixer + Context

- **MultiWPMixer: where does context enter?**
  1. **Context embedding (frozen, same as MultiPatchTST)**
  2. **Scale-specific projection of context**
  3. **Context-aware patch embeddings**
     - Augment each patch embedding at scale m:
       $$\tilde{E}^{(m)}_{c,p,:} = E^{(m)}_{c,p,:} + \mathbf{b}^{(m)}_{\text{ctx}}$$
       (equivalently: treat $\mathbf{b}^{(m)}_{\text{ctx}}$ as a "context bias" for that scale)
  4. **Context in Mixer blocks**
     - Patch-Mixer and Embedding-Mixer now operate on $\tilde{E}^{(m)}$.
     - Temporal mixing at every scale is conditioned on the global context.
  5. **Final prediction**
     - Aggregate all scale outputs $\{H^{(m)}\}$ (e.g., concat or sum).
     - Small MLP head $\rightarrow$ forecast horizon $\hat{\mathbf{Y}}$.

Experiment

Evaluation Setup: Baselines & Compared Models

- **Varying-history scenario**
  - **Unimodal baselines**
    - **DLinear** – linear decomposition model on raw time series only
    - **PatchTST** – Transformer encoder on patch embeddings of the time series
    - **WPMixer** – Wavelet + MLP-Mixer on the time series
  - **Multimodal variants (ours)**
    - **MultiPatchTST** – PatchTST + frozen context encoder + late fusion MLP head
    - **MultiWPMixer** – WPMixer + frozen context encoder + multi-scale fusion

Evaluation Setup: Baselines & Compared Models

- **Varying-history scenario**

- **Cold-start scenario**
  - **Baseline: average of retrieved neighbors**
    - Use same semantic retrieval as GPT method to find top-k similar entities
    - For each horizon h, take the *element-wise average* of their future time series as prediction
    - No LLM generation — purely "retrieval + averaging"
  - **GPT-based forecasting**
    - Retrieve the same top-k relevant entities
    - Build a structured prompt with their text + time series (Table 13 template)
    - Ask **GPT-4o** to output forecasted values (and reasoning text)

Evaluation Setup: Metrics

- **Root Mean Squared Error (RMSE)**
- **Formula:**

$$\text{RMSE} = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(y_t - \hat{y}_t)^2}$$

- **Properties**
  - Measures error **in the original scale** of y_t – units are the same as the target series.
  - **Very sensitive to large deviations** – the squaring step makes big errors dominate.
- **Role in this paper**
  - Good for judging **absolute error**, especially on datasets with **large magnitudes** (e.g., WikiPeople, Tianchi).
  - Keeps the **business scale** — easier for practitioners to interpret.

Evaluation Setup: Metrics

- **Weighted RMS Percent Error (WRMSPE)**

$$\text{WRMSPE} = \sqrt{\frac{\frac{1}{T}\sum_{t=1}^{T}(y_t - \hat{y}_t)^2}{\frac{1}{T}\sum_{t=1}^{T}|y_t|}}$$

- **Meaning**
  - Numerator: same mean squared error as RMSE.
  - Denominator: **mean absolute value** of the ground truth $\rightarrow$ acts as a **normalization factor**.
  - Result: units cancel out $\rightarrow$ a **dimensionless relative error**.
  - Useful when datasets / series have **very different scales**.

Evaluation Setup: Metrics

- **Weighted RMS Percent Error (WRMSPE)**

$$\text{WRMSPE} = \sqrt{\frac{\frac{1}{T}\sum_{t=1}^{T}(y_t - \hat{y}_t)^2}{\frac{1}{T}\sum_{t=1}^{T}|y_t|}}$$

- **Meaning**
- **Why WRMSPE instead of the popular RMSSE?**
  - RMSSE requires a **naive baseline** (e.g., seasonal naive).
  - In MoTime, datasets have **different forecast horizons** and setups → RMSSE becomes hard to compare.
  - The authors therefore choose:
    - **RMSE** → keeps original scale for absolute error.
    - **WRMSPE** → provides a **scale-invariant** relative error.

Evaluation Setup: Training Protocol

Varying-history forecasting

- **Data split**
  - Train / val / test = **7 : 1 : 2** on the *longest* version of each series.
- **Short vs. long history**
  - For each dataset, define a threshold for "short" based on typical length:
    - **PixelRec, AmazonReview, WikiPeople:** 100 steps
    - **MovieLens:** 50 steps
    - **TaobaoFashion, Tianchi:** 20 steps
    - **News:** 18 steps
  - Split series into **short** and **long** subsets at a **1 : 1 ratio**.
  - Train models on both groups together, then evaluate separately on:
    - short, long, and **mixture** subsets.
- **Input / output horizons**
  - Daily datasets: use **7-day input**, forecast **7–28 days** ahead.
  - News (high-frequency): use **6-step input**, forecast up to **12 steps** ahead.

Evaluation Setup: Training Protocol

Cold-start forecasting

- For each dataset, **randomly select 30 series** as cold-start targets.
- Start forecasting from the **first valid time step** (non-zero, non-placeholder value).
- Use only **7 previous daily steps** or **6 previous 20-minute steps** from a **relevant but non-target entity**, depending on dataset frequency.

Evaluation Setup: Implementation details

- **Reindexing** is applied so that each mini-batch mixes **short** and **long** series.
- **Sparse-series filtering:**
  - Remove PixelRec series with density < 0.4.
  - Randomly sample **1,000 series** from MovieLens to reduce computation.
- **Optimization**
  - Loss: **MSE**; Optimizer: **Adam**.
  - **Trainable parts**: time-series encoders (PatchTST / WPMixer) + multimodal fusion layers.
  - **Frozen**: all contextual encoders (LLM / VLM).
  - **Early stopping** based on validation loss.
- **Hardware**
  - All experiments run on a **single NVIDIA GPU** (A100 / A40 / RTX 3090), chosen depending on availability.

Results and Analysis: Long History Results (Varying-History)

Table 3: Evaluation on varying-training length forecasting (long history series). Due to space limitations, evaluation scores are rounded to three decimal places. In cases where multiple models appear to have identical scores under this rounding, we still mark the best and second-best results in bold and with an underline based on the full-precision metrics.

Models		DLinear		PatchTST		WPMixer		MultiPatchTST		MultiWPMixer	
Metric		rmse	wrmspe	rmse	wrmspe	rmse	wrmspe	rmse	wrmspe	rmse	wrmspe
PixelRec	1	1.379	1.606	1.322	1.541	1.356	1.580	1.328	1.548	**1.315**	**1.532**
	7	1.547	1.798	1.444	1.678	1.456	1.692	**1.427**	**1.658**	1.439	1.672
	14	1.577	1.824	1.501	1.736	1.509	1.746	**1.484**	**1.716**	1.496	1.730
	21	1.647	1.897	1.546	1.781	1.551	1.787	**1.529**	**1.761**	1.541	1.775
	28	1.648	1.891	1.584	1.817	1.593	1.828	**1.570**	**1.801**	1.577	1.809
Amazon	1	**0.373**	**3.604**	0.374	3.620	0.377	3.649	0.373	3.612	0.376	3.642
	7	0.394	3.797	0.390	3.763	0.390	3.765	**0.389**	**3.750**	0.390	3.763
	14	0.403	3.873	0.401	3.851	0.400	3.844	**0.400**	**3.838**	0.401	3.848
	21	0.411	3.937	0.409	3.909	0.408	3.906	**0.407**	**3.898**	0.409	3.911
	28	0.417	3.974	0.415	3.955	0.414	3.949	**0.414**	**3.945**	0.415	3.956
Taobao	1	6.203	1.808	5.864	1.709	6.414	1.869	6.177	1.800	**5.632**	**1.641**
	7	6.228	1.814	6.073	1.768	**6.066**	**1.767**	6.313	1.838	6.155	1.793
	14	7.066	2.047	**6.302**	**1.826**	6.496	1.882	6.405	1.856	6.375	1.847
	21	7.107	2.058	6.618	1.916	6.749	1.954	**6.613**	**1.915**	6.592	1.909
	28	7.289	2.113	6.879	1.994	6.974	2.021	**6.839**	**1.982**	6.853	1.986
Tianchi	1	186.20	11.42	184.10	11.29	185.50	11.37	180.50	11.07	**179.98**	**11.03**
	7	181.70	11.43	175.40	11.03	176.40	11.09	174.60	10.98	**174.28**	**10.96**
	14	189.00	12.17	**175.80**	**11.32**	176.00	11.33	175.90	11.33	176.02	11.34
Movielens	1	**0.260**	**6.030**	0.262	6.073	0.260	6.032	0.262	6.079	0.261	6.043
	7	**0.265**	**6.141**	0.268	6.213	0.267	6.172	0.274	6.338	0.266	6.164
	14	**0.267**	**6.189**	0.274	6.332	0.269	6.233	0.279	6.459	0.271	6.282
	21	**0.269**	**6.229**	0.279	6.466	0.272	6.303	0.286	6.624	0.275	6.360
	28	**0.271**	**6.274**	0.283	6.542	0.273	6.307	0.290	6.717	0.277	6.406
News	1	57.43	47.82	**61.33**	**51.07**	58.39	48.62	58.89	49.05	57.89	48.21
	3	60.02	47.42	60.30	47.63	**59.37**	**46.90**	59.87	47.29	59.48	46.99
	6	51.41	40.83	51.52	40.91	51.06	40.55	51.20	40.66	**51.06**	**40.55**
	9	46.56	36.41	46.54	36.39	46.18	36.10	46.42	36.29	**46.15**	**36.08**
	12	44.86	33.46	44.64	33.29	44.35	33.08	44.48	33.18	**44.31**	**33.05**
WikiPeople	1	21977	5.105	20767	4.824	20797	4.831	20844	4.841	**20651**	**4.796**
	7	22174	5.235	**21571**	**5.093**	21635	5.108	21583	5.096	21596	5.099
	14	19860	4.741	18527	4.423	18563	4.432	**18519**	**4.421**	18568	4.433
	21	18684	4.441	17666	4.199	17754	4.220	**17646**	**4.194**	17699	4.206
	28	19063	4.488	17493	4.119	17588	4.141	**17461**	**4.111**	17562	4.135

- **Multimodal models are usually strongest.**
  - On most datasets, at medium / long horizons, **MultiPatchTST** or **MultiWPMixer** achieves the best or second-best RMSE and WRMSPE.
  - Adding text / images / metadata *on top of* a strong TS encoder generally improves forecasting.
- **Unimodal baselines are still competitive.**
  - **DLinear** is a surprisingly strong unimodal baseline.
  - **PatchTST** and **WPMixer** also perform well, showing TS encoder choice matters.

Results and Analysis: Short History Results (Varying-History)

Models		DLinear		PatchTST		WPMixer		MultiPatchTST		MultiWPMixer	
Metric		rmse	wrmspe	rmse	wrmspe	rmse	wrmspe	rmse	wrmspe	rmse	wrmspe
PricERec	1	1.387	1.616	**1.319**	**1.536**	1.361	1.585	1.324	1.542	1.314	1.530
	7	1.585	1.844	1.480	1.721	1.495	1.739	**1.464**	**1.703**	1.474	1.715
	14	1.628	1.887	1.548	1.795	1.559	1.808	**1.532**	**1.777**	1.543	1.789
	21	1.700	1.963	1.594	1.840	1.602	1.850	**1.579**	**1.823**	1.589	1.835
	28	1.703	1.959	1.633	1.878	1.648	1.895	**1.619**	**1.862**	1.629	1.873
Amazon	1	**0.263**	**4.659**	0.263	4.672	0.264	4.687	0.263	4.667	0.264	4.682
	7	0.273	4.821	0.269	4.754	0.269	4.751	0.269	4.751	**0.269**	**4.750**
	14	0.275	4.840	0.273	4.804	**0.272**	**4.797**	0.272	4.801	0.272	4.800
	21	0.278	4.875	0.275	4.832	**0.275**	**4.828**	0.275	4.830	0.275	4.834
	28	0.279	4.883	0.277	4.852	**0.277**	**4.847**	0.277	4.852	0.277	4.855
Taobao	1	7.659	2.188	7.336	2.095	7.897	2.255	7.650	2.185	**7.012**	**2.003**
	7	7.729	2.203	7.549	2.152	**7.533**	**2.148**	7.834	2.233	7.639	2.178
	14	8.703	2.472	**7.750**	**2.201**	7.997	2.271	7.883	2.239	7.847	2.229
	21	8.476	2.413	7.969	2.269	8.078	2.300	7.943	2.261	**7.923**	**2.255**
	28	8.607	2.459	8.086	2.310	8.217	2.347	**8.060**	**2.303**	8.076	2.307
Tianchi	1	202.6	12.06	199.8	11.89	201.5	11.99	**196.8**	**11.72**	212.2	12.27
	7	202.4	12.31	194.7	11.84	195.9	11.92	**193.8**	**11.79**	211.5	12.46
	14	213.8	13.25	198.3	12.30	198.6	12.32	**198.1**	**12.28**	218.5	13.06
Movielens	1	**0.197**	**6.259**	0.198	6.295	0.197	6.265	0.197	6.273	0.197	6.274
	7	0.199	6.309	0.201	6.362	0.200	6.336	0.200	6.347	**0.198**	**6.278**
	14	0.200	6.321	0.203	6.415	0.201	6.348	0.203	6.397	**0.200**	**6.319**
	21	**0.201**	**6.332**	0.206	6.481	0.203	6.383	0.206	6.476	0.202	6.347
	28	**0.203**	**6.340**	0.208	6.507	0.203	6.366	0.208	6.501	0.204	6.373
News	1	22.98	37.22	21.30	34.50	23.84	38.60	**21.27**	**34.45**	23.12	37.44
	3	24.81	39.45	23.95	38.09	23.39	37.20	23.59	37.51	**23.26**	**36.99**
	6	22.12	35.32	22.09	35.28	21.21	33.87	21.46	34.26	**21.17**	**33.80**
	9	20.75	33.15	20.38	32.56	19.67	31.43	20.13	32.17	**19.56**	**31.25**
	12	20.78	32.86	20.02	31.65	19.29	30.51	19.64	31.06	**19.15**	**30.28**
WikiPeople	1	11173	3.371	**10431**	**3.146**	10444	3.151	10479	3.161	10416	3.142
	7	11872	3.620	11132	3.394	11210	3.418	**11127**	**3.393**	11130	3.394
	14	12039	3.677	11037	3.371	11056	3.377	**11021**	**3.366**	11055	3.377
	21	11424	3.489	10750	3.284	10792	3.297	**10745**	**3.282**	10778	3.292
	28	11809	3.592	10717	3.260	10762	3.274	**10701**	**3.255**	10761	3.273

- **Reminder of the hypothesis**
  - Authors originally expected **external modalities to help more** when the **history is short**, because the pure time-series signal is weak.
- **What Table 15 actually shows**
  - Across most datasets, the **ranking of models looks very similar** to the long-history case.
  - Multimodal models (**MultiPatchTST** / **MultiWPMixer**) **still usually outperform** unimodal models,
    but the improvement is **not uniformly bigger** than in long history.
  - Some datasets show clear multimodal gains; for others, gains are **small or almost flat**.

Results and Analysis: Varying-history Forecasting

Table 3: Evaluation on varying-training length forecasting (long history series). Due to space limitations, evaluation scores are rounded to three decimal places. In cases where multiple models appear to have identical scores under this rounding, we still mark the best and second-best results in bold and with an underline based on the full-precision metrics.

Models		DLinear		PatchTST		WPMixer		MultiPatchTST		MultiWPMixer	
Metric		rmse	wrmspe	rmse	wrmspe	rmse	wrmspe	rmse	wrmspe	rmse	wrmspe
PixelRec	1	1.379	1.606	1.322	1.541	1.356	1.580	1.328	1.548	**1.315**	**1.532**
	7	1.547	1.798	1.444	1.678	1.456	1.692	**1.427**	**1.658**	1.439	1.672
	14	1.577	1.824	1.501	1.736	1.509	1.746	**1.484**	**1.716**	1.496	1.730
	21	1.647	1.897	1.546	1.781	1.551	1.787	**1.529**	**1.761**	1.541	1.775
	28	1.648	1.891	1.584	1.817	1.593	1.828	**1.570**	**1.801**	1.577	1.809
Amazon	1	**0.373**	**3.604**	0.374	3.620	0.377	3.649	0.373	3.612	0.376	3.642
	7	0.394	3.797	0.390	3.763	0.390	3.765	**0.389**	**3.750**	0.390	3.763
	14	0.403	3.873	0.401	3.851	0.400	3.844	**0.400**	**3.838**	0.401	3.848
	21	0.411	3.937	0.409	3.909	0.408	3.906	**0.407**	**3.898**	0.409	3.911
	28	0.417	3.974	0.415	3.955	0.414	3.949	**0.414**	**3.945**	0.415	3.956
Taobao	1	6.203	1.808	5.864	1.709	6.414	1.869	6.177	1.800	**5.632**	**1.641**
	7	6.228	1.814	6.073	1.768	**6.066**	**1.767**	6.313	1.838	6.155	1.793
	14	7.066	2.047	**6.302**	**1.826**	6.496	1.882	6.405	1.856	6.375	1.847
	21	7.107	2.058	6.618	1.916	6.749	1.954	**6.613**	**1.915**	6.592	1.909
	28	7.289	2.113	6.879	1.994	6.974	2.021	**6.839**	**1.982**	6.853	1.986
Tianchi	1	186.20	11.42	184.10	11.29	185.50	11.37	180.50	11.07	**179.98**	**11.03**
	7	181.70	11.43	175.40	11.03	176.40	11.09	174.60	10.98	**174.28**	**10.96**
	14	189.00	12.17	**175.80**	**11.32**	176.00	11.33	175.90	11.33	176.02	11.34
Movielens	1	**0.260**	**6.030**	0.262	6.073	0.260	6.032	0.262	6.079	0.261	6.043
	7	**0.265**	**6.141**	0.268	6.213	0.267	6.172	0.274	6.338	0.266	6.164
	14	**0.267**	**6.189**	0.274	6.332	0.269	6.233	0.279	6.459	0.271	6.282
	21	**0.269**	**6.229**	0.279	6.466	0.272	6.303	0.286	6.624	0.275	6.360
	28	**0.271**	**6.274**	0.283	6.542	0.273	6.307	0.290	6.717	0.277	6.406
News	1	57.43	47.82	**61.33**	**51.07**	58.39	48.62	58.89	49.05	57.89	48.21
	3	60.02	47.42	60.30	47.63	**59.37**	**46.90**	59.87	47.29	59.48	46.99
	6	51.41	40.83	51.52	40.91	51.06	40.55	51.20	40.66	**51.06**	**40.55**
	9	46.56	36.41	46.54	36.39	46.18	36.10	46.42	36.29	**46.15**	**36.08**
	12	44.86	33.46	44.64	33.29	44.35	33.08	44.48	33.18	**44.31**	**33.05**
WikiPeople	1	21977	5.105	20767	4.824	20797	4.831	20844	4.841	**20651**	**4.796**
	7	22174	5.235	**21571**	**5.093**	21635	5.108	21583	5.096	21596	5.099
	14	19860	4.741	18527	4.423	18563	4.432	**18519**	**4.421**	18568	4.433
	21	18684	4.441	17666	4.199	17754	4.220	**17646**	**4.194**	17699	4.206
	28	19063	4.488	17493	4.119	17588	4.141	**17461**	**4.111**	17562	4.135

Table 3: Long-history results

Models		DLinear		PatchTST		WPMixer		MultiPatchTST		MultiWPMixer	
Metric		rmse	wrmspe	rmse	wrmspe	rmse	wrmspe	rmse	wrmspe	rmse	wrmspe
PixelRec	1	1.387	1.616	**1.319**	**1.536**	1.361	1.585	1.324	1.542	1.314	1.530
	7	1.585	1.844	1.480	1.721	1.495	1.739	**1.464**	**1.703**	1.474	1.715
	14	1.628	1.887	1.548	1.795	1.559	1.808	**1.532**	**1.777**	1.543	1.789
	21	1.700	1.963	1.594	1.840	1.602	1.850	**1.579**	**1.823**	1.589	1.835
	28	1.703	1.959	1.633	1.878	1.648	1.895	**1.619**	**1.862**	1.629	1.873
Amazon	1	**0.263**	**4.659**	0.263	4.672	0.264	4.687	0.263	4.667	0.264	4.682
	7	0.273	4.821	0.269	4.754	0.269	4.751	0.269	4.751	**0.269**	**4.750**
	14	0.275	4.840	0.273	4.804	**0.272**	**4.797**	0.272	4.801	0.272	4.800
	21	0.278	4.875	0.275	4.832	**0.275**	**4.828**	0.275	4.830	0.275	4.834
	28	0.279	4.883	0.277	4.852	**0.277**	**4.847**	0.277	4.852	0.277	4.855
Taobao	1	7.659	2.188	7.336	2.095	7.897	2.255	7.650	2.185	**7.012**	**2.003**
	7	7.729	2.203	7.549	2.152	**7.533**	**2.148**	7.834	2.233	7.639	2.178
	14	8.703	2.472	**7.750**	**2.201**	7.997	2.271	7.883	2.239	7.847	2.229
	21	8.476	2.413	7.969	2.269	8.078	2.300	7.943	2.261	**7.923**	**2.255**
	28	8.607	2.459	8.086	2.310	8.217	2.347	**8.060**	**2.303**	8.076	2.307
Tianchi	1	202.6	12.06	199.8	11.89	201.5	11.99	**196.8**	**11.72**	212.2	12.27
	7	202.4	12.31	194.7	11.84	195.9	11.92	**193.8**	**11.79**	211.5	12.46
	14	213.8	13.25	198.3	12.30	198.6	12.32	**198.1**	**12.28**	218.5	13.06
Movielens	1	**0.197**	**6.259**	0.198	6.295	0.197	6.265	0.197	6.273	0.197	6.274
	7	0.199	6.309	0.201	6.362	0.200	6.336	0.200	6.347	**0.198**	**6.278**
	14	0.200	6.321	0.203	6.415	0.201	6.348	0.203	6.397	**0.200**	**6.319**
	21	**0.201**	**6.332**	0.206	6.481	0.203	6.383	0.206	6.476	0.202	6.347
	28	**0.203**	**6.340**	0.208	6.507	0.203	6.366	0.208	6.501	0.204	6.373
News	1	22.98	37.22	21.30	34.50	23.84	38.60	**21.27**	**34.45**	23.12	37.44
	3	24.81	39.45	23.95	38.09	23.39	37.20	23.59	37.51	**23.26**	**36.99**
	6	22.12	35.32	22.09	35.28	21.21	33.87	21.46	34.26	**21.17**	**33.80**
	9	20.75	33.15	20.38	32.56	19.67	31.43	20.13	32.17	**19.56**	**31.25**
	12	20.78	32.86	20.02	31.65	19.29	30.51	19.64	31.06	**19.15**	**30.28**
WikiPeople	1	11173	3.371	**10431**	**3.146**	10444	3.151	10479	3.161	10416	3.142
	7	11872	3.620	11132	3.394	11210	3.418	**11127**	**3.393**	11130	3.394
	14	12039	3.677	11037	3.371	11056	3.377	**11021**	**3.366**	11055	3.377
	21	11424	3.489	10750	3.284	10792	3.297	**10745**	**3.282**	10778	3.292
	28	11809	3.592	10717	3.260	10762	3.274	**10701**	**3.255**	10761	3.273

Table 15: Short-history results

Special Case: MovieLens

- **Why might MovieLens-short benefit more from multimodality?**
  - MovieLens user–item interactions are **very sparse and spiky**.
    - In a short history window, you mostly see **zero ratings + a few isolated spikes** $\Rightarrow$ the time series alone carries weak signal.
  - Each movie, however, has **rich static metadata**: title, tags, synopsis, etc.
    - These text features describe *what kind of movie it is*, which is **highly predictive of future interest**.
  - **WPMixer** is already good at handling **local spikes**;
  - **MultiWPMixer = WPMixer + text embedding**, so it can:
    - use movie text to guess which items are likely to receive spikes,
    - compensate for the lack of temporal history in short windows.
- **Key message**
  - On MovieLens, **short history + sparse & spiky interactions + rich text metadata**

    $\Rightarrow$ multimodal (MultiWPMixer) helps more on the short subset than on the long subset.
  - This illustrates that **which history length benefits more from multimodality is dataset-dependent**, not a universal rule.

Cold-start Results

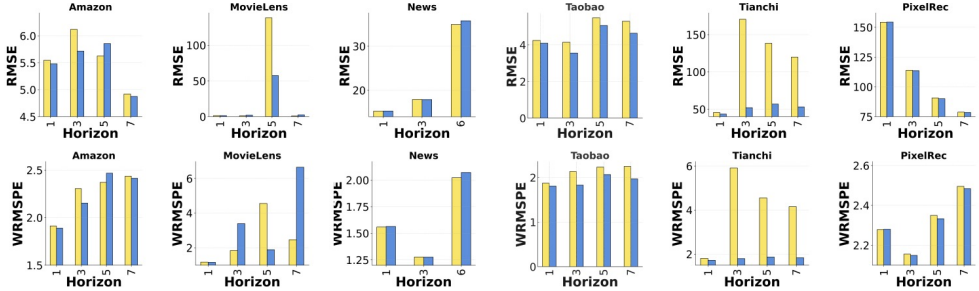

Figure 2: Cold-start forecasting results. Top: RMSE; Bottom: WRMSPE. The yellow bar is the baseline `average`, and the blue bar is `GPT` forecasting.

Cold-start Results

Cold-start Forecasting

- Across most datasets and horizons, **blue** $\leq$ **yellow** $\rightarrow$ GPT forecasting is **better or at least comparable** to the averaging baseline.
- Gains are most visible on **sparse datasets**:
  - **PixelRec, Tianchi, TaobaoFashion**: GPT clearly reduces both **RMSE and WRMSPE**, showing that **text / metadata can substitute for missing numeric history**.
- Some datasets show **limited or mixed benefit**:
  - **AmazonReview**: GPT and baseline are almost identical across horizons $\rightarrow$ text-based retrieval adds little, possibly due to **generic / noisy product text** or low similarity between items.
  - **MovieLens**: GPT brings **large improvements in at least one metric** (especially at short horizons) on this **spiky rating data**.

Cold-start Results

- **AmazonReview**
  - GPT $\approx$ average baseline
  - Likely due to **generic or noisy product descriptions**
- **MovieLens**
  - GPT significantly better, especially at **short horizons**
  - Sparse, **spike-like interaction patterns**
  - Retrieval context helps more than naive averaging
- **TaobaoFashion**
  - GPT improves **WRMSPE** more than RMSE
  - Reduces large relative errors, but volatility remains
- **Tianchi**
  - Clear and consistent **GPT advantage**
  - Large, sparse datasets with **bursty dynamics**
  - External modalities effectively substitute missing history

---

Overall, Cold-start performance depends on sparsity, text quality, and inter-entity similarity

Discussion & Limitations

Limitations:

- **Training imbalance**
  - Short-history series contribute fewer training windows
  - No upsampling was applied $\rightarrow$ may *underestimate* multimodal benefits
- **Sparsity not systematically studied**
  - Long but highly sparse series may carry less signal than short dense ones
  - Sparsity effects are acknowledged but not isolated or quantified

Conclusion

Conclusion:

What MoTime Provides

- A large-scale **multimodal time series dataset suite**
- Structured evaluation under:
  - **Varying-history lengths**
  - **Cold-start forecasting** conditions

Key Contributions

- Enables systematic analysis of **when external modalities help forecasting**
- Demonstrates that multimodality is especially valuable when:
  - historical data is limited
  - time-series signals are weak or sparse

Thank you for your attending.