Towards Powerful Foundation Models for Time Series Classification

Vasilii Feofanov

Hi! PARIS Summer School, 2026

Hi! PARIS
HEC PARIS
INSTITUT POLYTECHNIQUE DE PARIS
PARIS ARTIFICIAL INTELLIGENCE FOR SOCIETY

Scope

1. **(Feofanov et al., 2026) Mantis: Lightweight foundation model for time series classification *(ICML'26)*.**
2. (Xie et al., 2026) CauKer: classification time series foundation models can be pretrained on synthetic data only. (*ICLR'26 oral)*.
3. (Moakher et al., 2026) UTICA: Multi-Objective Self-Distillation Foundation Model Pretraining for Time Series Classification (*ICLR'26 TSALM workshop*).
4. (Odonnat et al., 2026) Layer by layer, module by module: Choose both for optimal OOD probing of ViT (*ICLR'26 CAO workshop*).
5. (Roschmann et al., 2025) Time Series Representations for Classification Lie Hidden in Pretrained Vision Transformers (*NeurIPS'25 BERT2S workshop*)
6. (Gnassounou et al., 2025) Leveraging Generic Time Series Foundation Models for EEG Classification (*NeurIPS'25 BERT2S workshop*)
7. (Chen et al., 2025). Comodo: Cross-modal video-to-imu distillation for efficient egocentric human activity recognition.
8. (Ilbert et al., 2024) SAMFormer: Unlocking the potential of transformers in time series forecasting with sharpness-aware minimization and channel-wise attention (ICML'24).
9. (Lin et al., 2024) Nutime: Numerically multi-scaled embedding for large-scale time-series pretraining (TMLR).
10. (Ansari et al., 2024) Chronos: Learning the language of time series.
11. (Nie et al., 2023) A time series is worth 64 words: Long-term forecasting with transformers (ICLR'23).
12. (Zeng et al. 2022) Are transformers effective for time series forecasting? (AAAI'22).
13. (Kim et al., 2022) Reversible Instance Normalization for Accurate Time-Series Forecasting against Distribution Shift (ICLR'22)
14. (Tsai et al., 2019) Transformer Dissection: A Unified Understanding of Transformer's Attention via the Lens of Kernel.

Mantis: Available Models

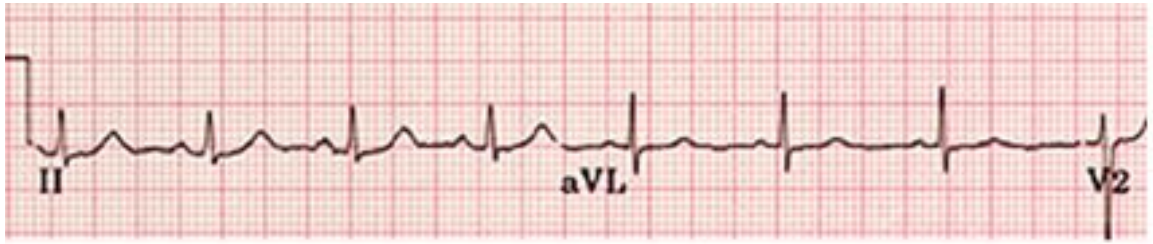
	Mantis	Mantis+	MantisV2
Module	`MantisV1`	`MantisV1`	`MantisV2`
Checkpoint	`paris-noah/Mantis-8M`	`paris-noah/MantisPlus`	`paris-noah/MantisV2`

- **Mantis**: original version (8M params) pre-trained on a collection of real datasets.
- **Mantis+**: new checkpoint when pre-training on synthetic data.
- **MantisV2**: architecture refinement (4M params) + synthetic pre-training.

GitHub

Time Series Classification: Examples

1. Cardiovascular disease diagnostic by electrocardiogram (ECG)

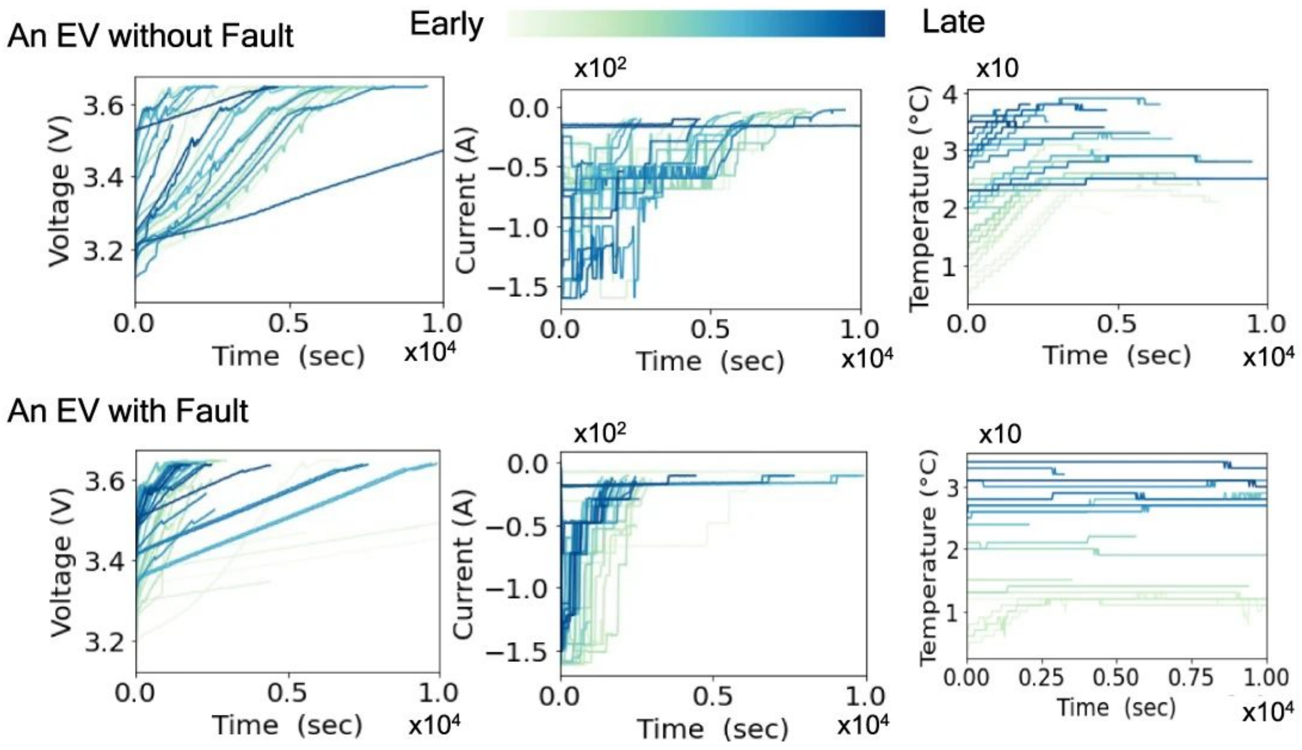


Class 0: Normal ECG

Class 1: Abnormal ECG

Time Series Classification: Examples

2. Fault detection of electric vehicle's battery.

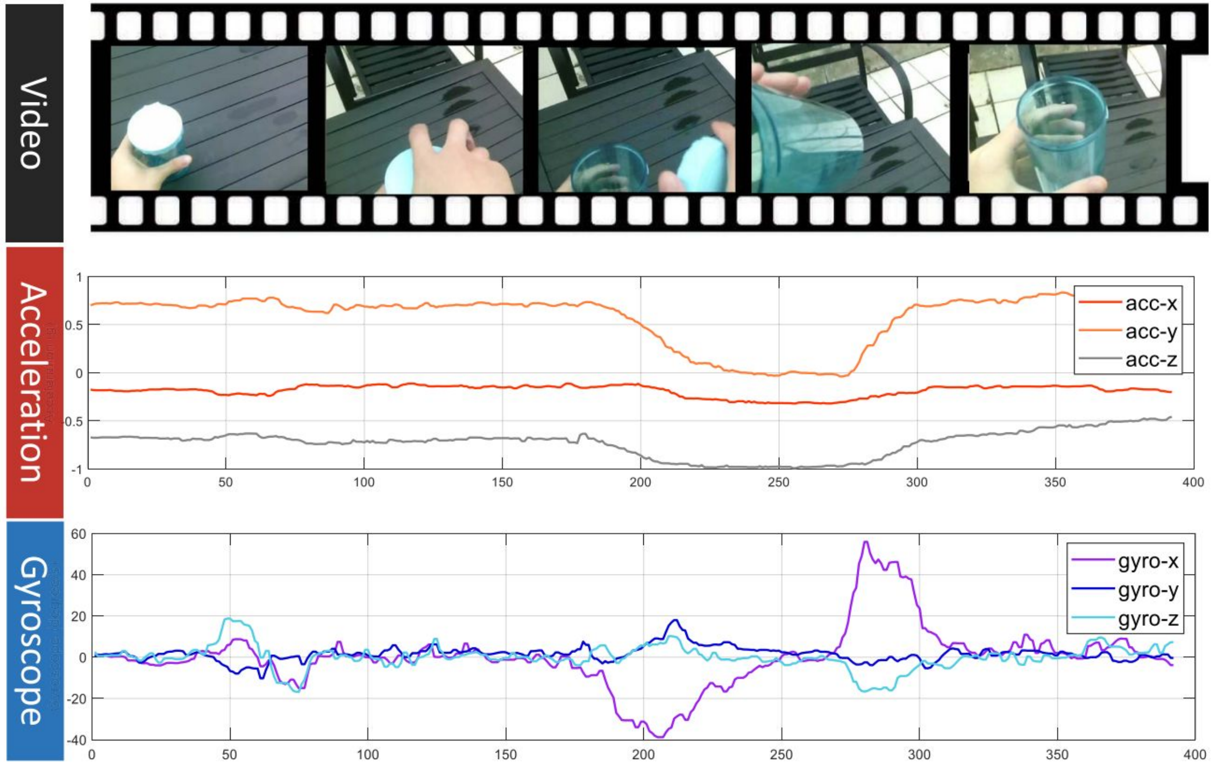


Class 0: No Fault

Class 1: Fault

Time Series Classification: Examples

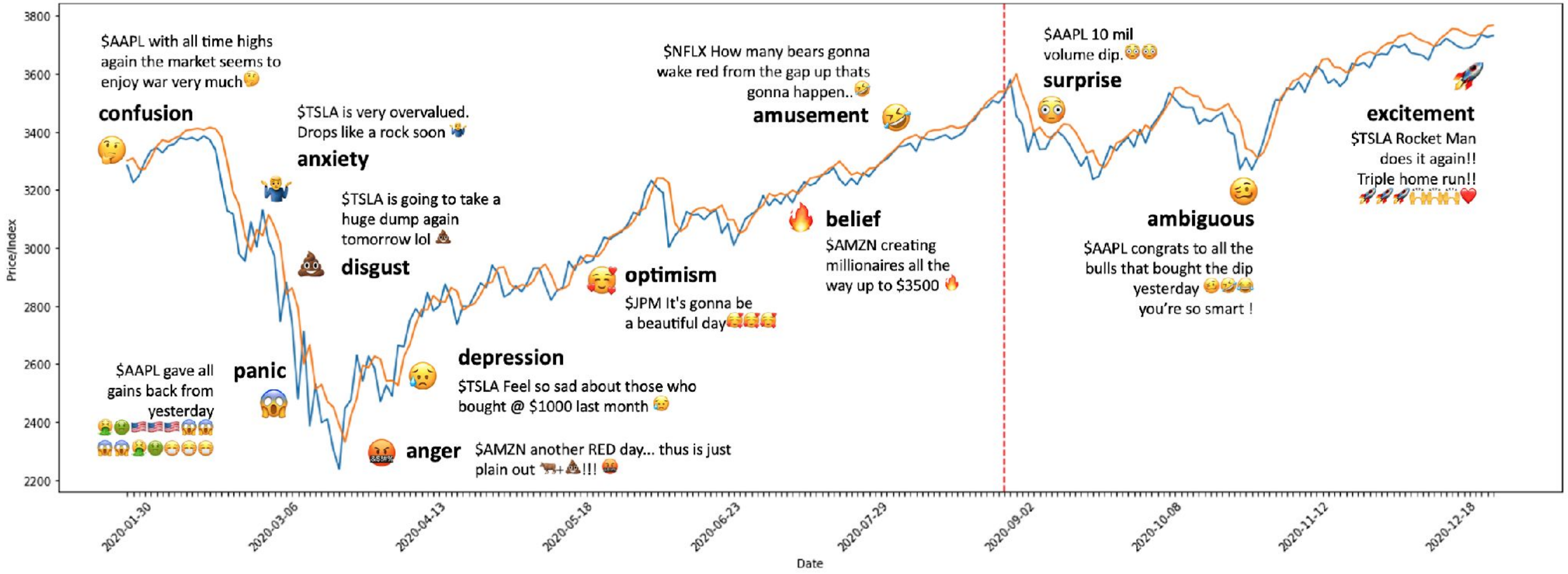
3. Human Activity Recognition



Class “Drinking”

Time Series Classification: Examples

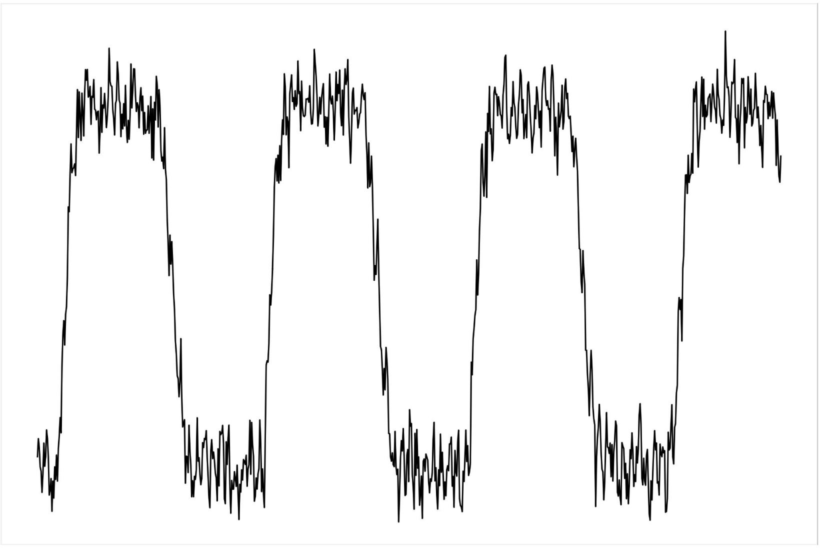
4. Finance: Prediction of Sentiment or Direction.

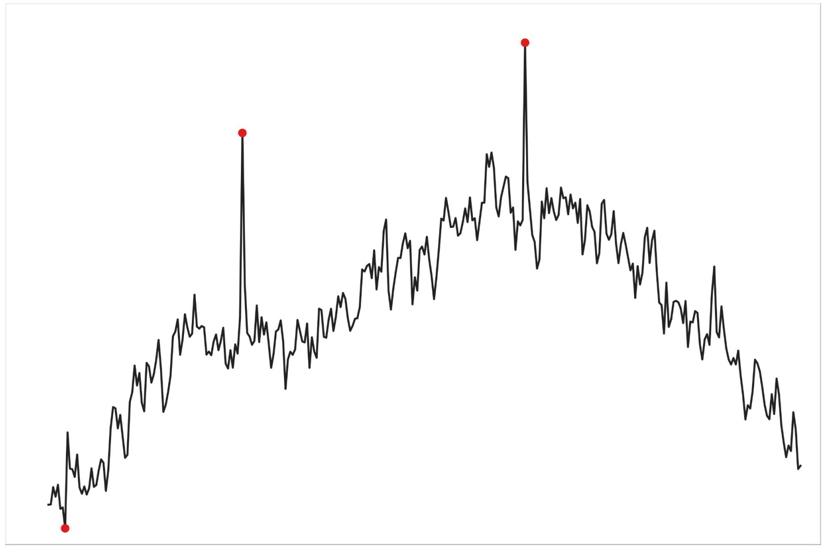


(a) Sentiment: mood of investors (confusion, anxiety, disgust, etc.)

(b) Direction: price goes up, down or stay neutral?

Forecasting vs Classification





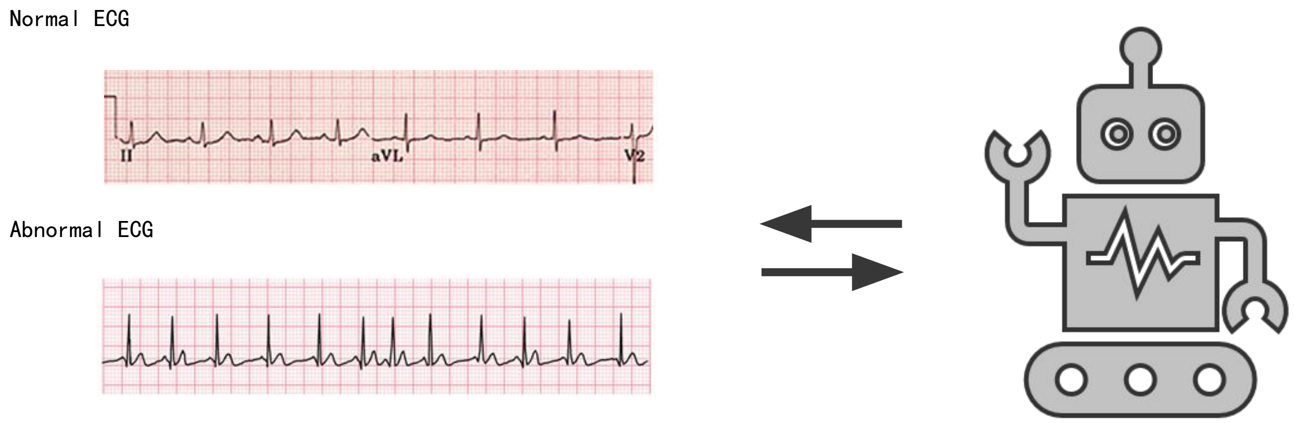
Forecasting: predict future patterns based on previously seen ones.

Jittering is often somewhat arbitrary => smoothing may be a good thing.

Classification: discriminate, detect anomalies.

Some spikes are a very important source of information!

Traditional Machine Learning

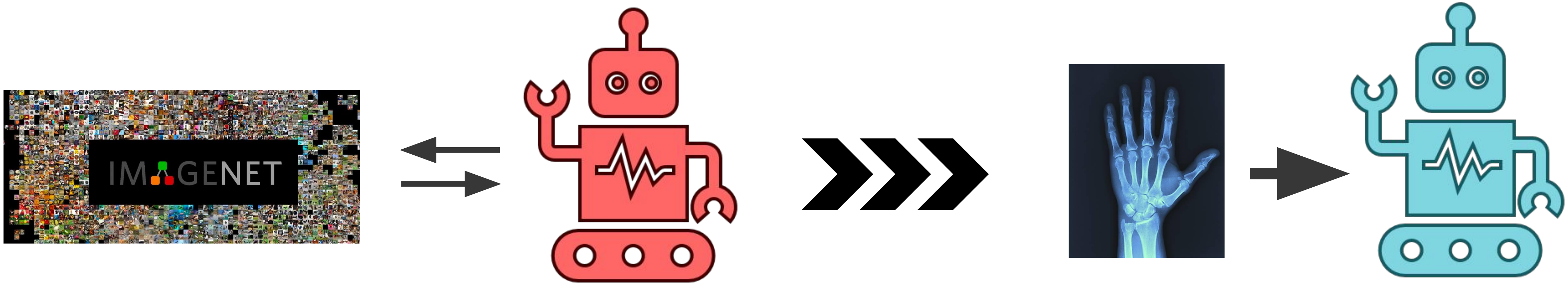


Traditional machine learning pipeline:

1. Collect training data. ← Issue 1: it requires a large training set.
2. Train a new model. ← Issue 2: for every new task a new model is needed.
3. Deploy it.

Success of LLM and Pre-training

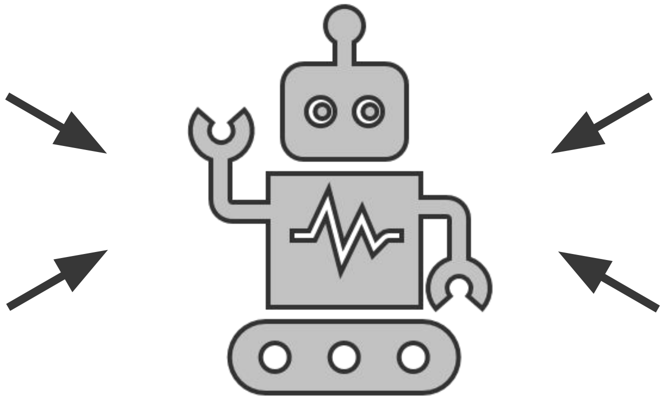
1. In Computer Vision, models are pre-trained on ImageNet and applied for new problems.



2. Large Language Models are versatile and able to solve different problems.

When the French revolution happened?

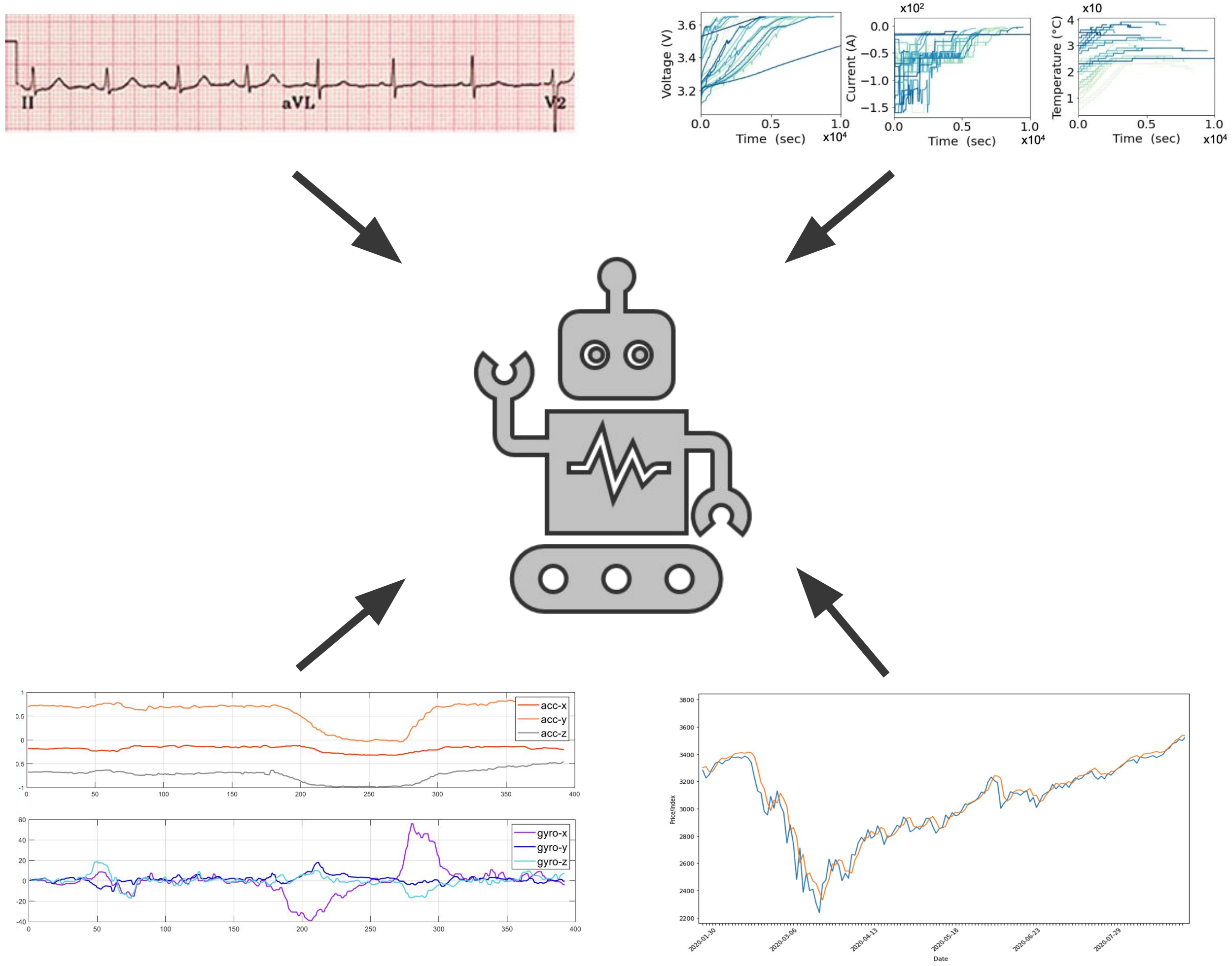
Translate me this text: "Le chat est noir".



Write me the lyrics to a new song

Summarize me this pdf document.

Time Series Foundation Model (TSFM)



The goal of a TSFM is to learn a projector using a large corpus of various datasets.

Advantages:

1. <u>Versatility</u>: one model for different problems.
2. <u>Knowledge alignment</u> with a new task.

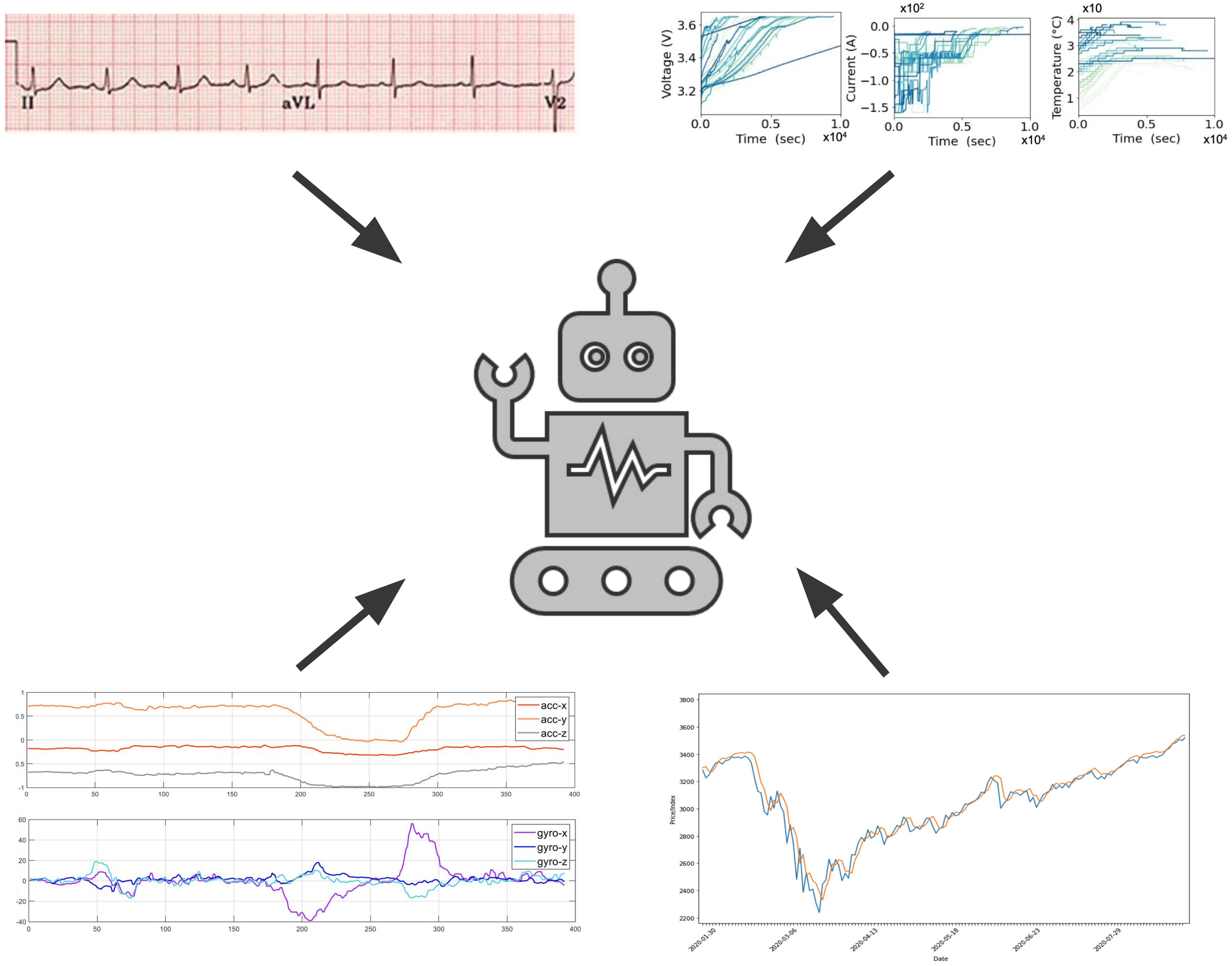
High-Level View on a TSFM

Notations

$$\mathbf{x} = [\mathbf{x}_1, \ldots, \mathbf{x}_d] \in \mathbb{R}^{d \times t}$$

- t timestamps (sequence length).
- d features (channels): for example, current, voltage, …
- d = 1 is called univariate.
- d > 1 is called multivariate.

Data Preparation



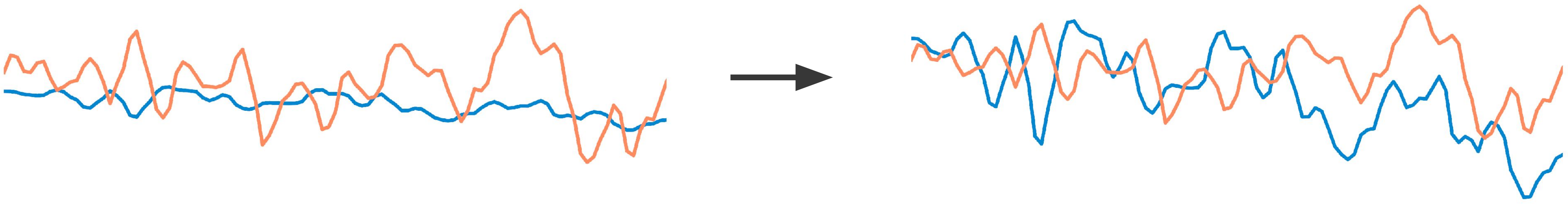
The goal of a TSFM is to learn a projector using a large corpus of various datasets.

Step 0: Data Preparation

a. Scale to same units.
b. Fix the context size.
c. Univariate dataset.

Pre-training Dataset Preparation

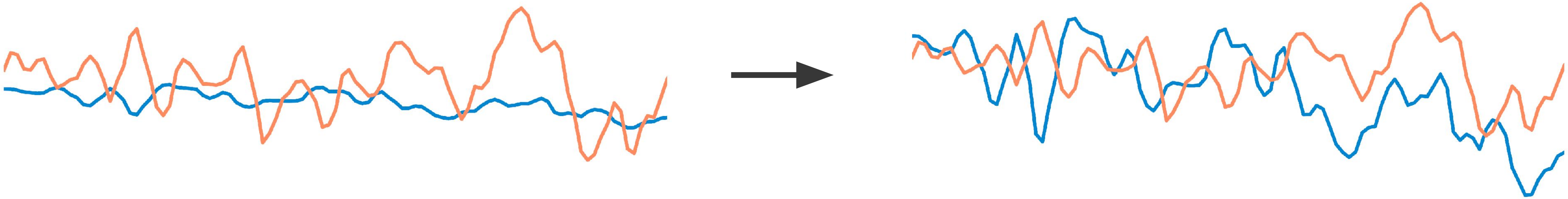
Instance-Wise Normalization



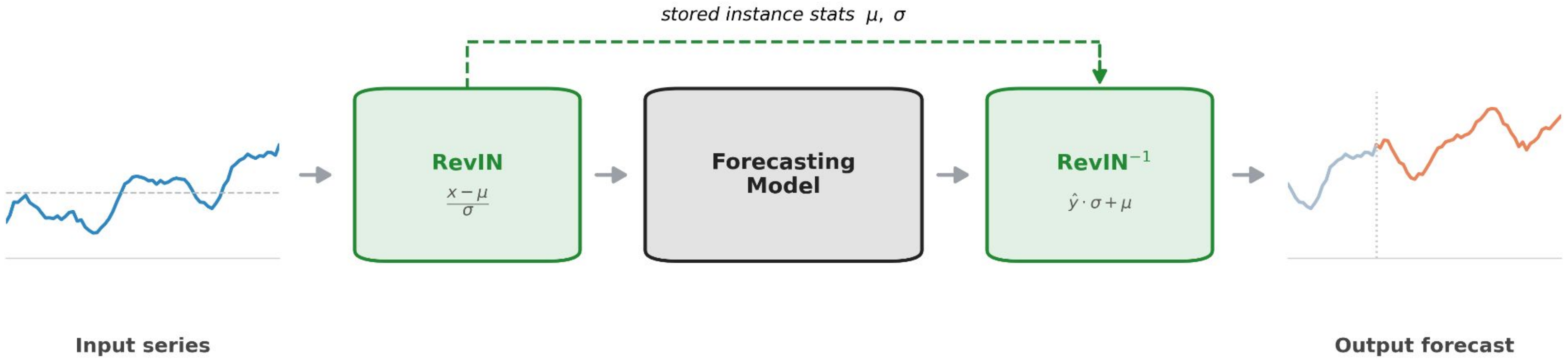
- ❑ For each time series, compute mean and std over the window.

Pre-training Dataset Preparation

Instance-Wise Normalization

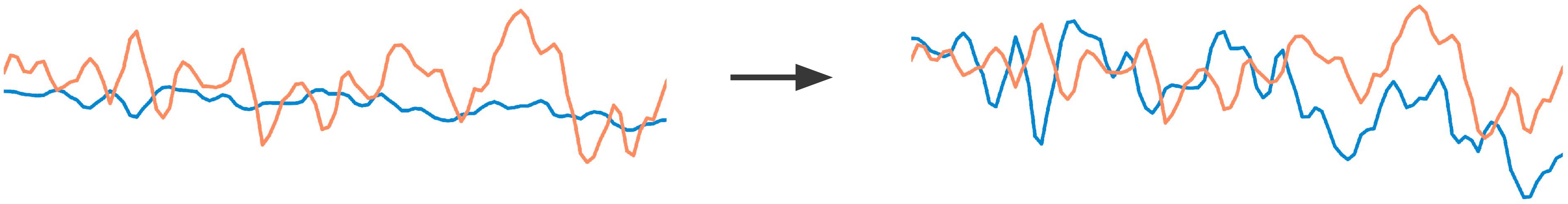


- ❑ For each time series, compute mean and std over the window.
- ❑ For time series forecasting, we also want to denormalize the prediction => RevIN.



Pre-training Dataset Preparation

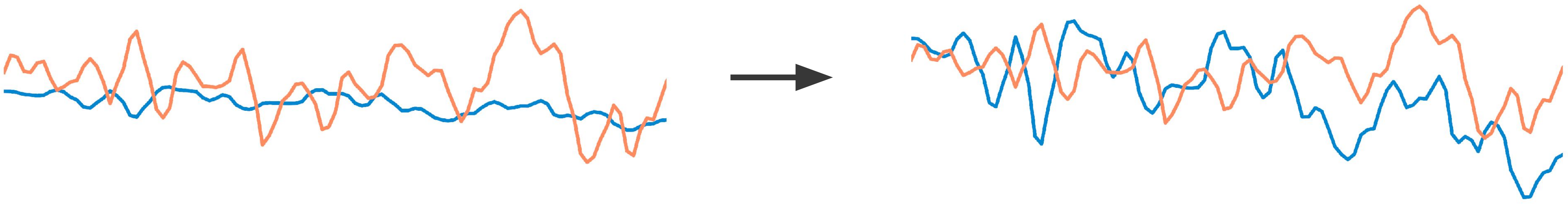
Instance-Wise Normalization



- ❑ For each time series, compute mean and std over the window.
- ❑ For time series forecasting, we also want to denormalize the prediction => RevIN.
- ❑ Cons:

Pre-training Dataset Preparation

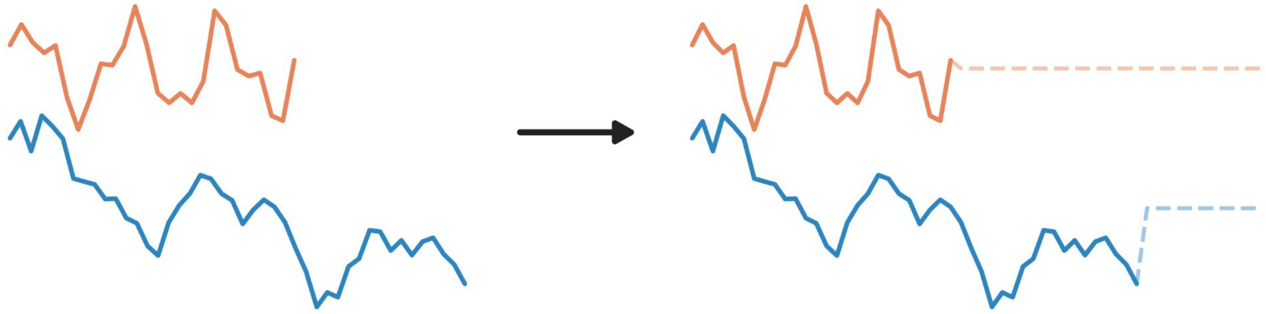
Instance-Wise Normalization



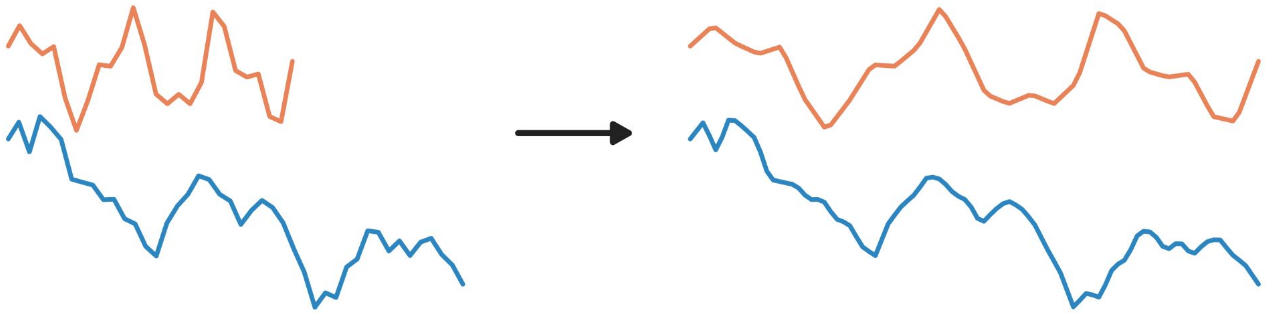
- ❑ For each time series, compute mean and std over the window.
- ❑ For time series forecasting, we also want to denormalize the prediction => RevIN.
- ❑ Cons:
  - ➔ Noise amplification, if a time series has no signal.
  - ➔ Bad for classification: absolute values might be important to distinguish two time series.

Fit the context length requirements

1. Padding

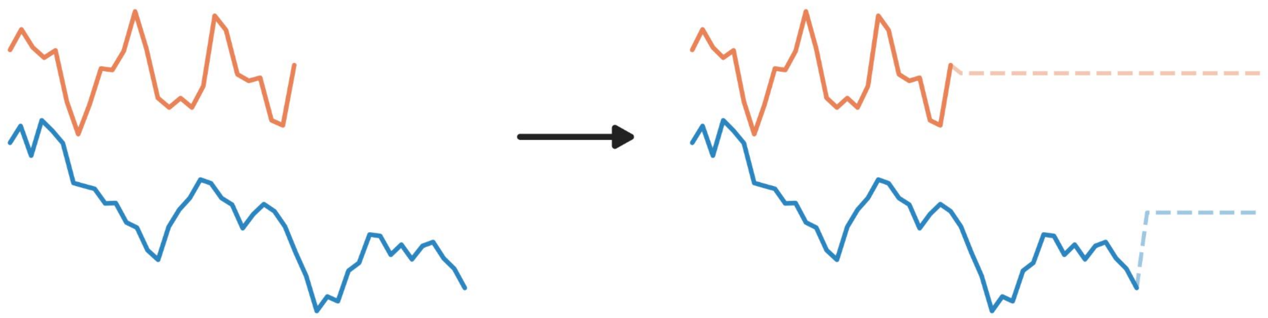


2. Resize by Interpolation

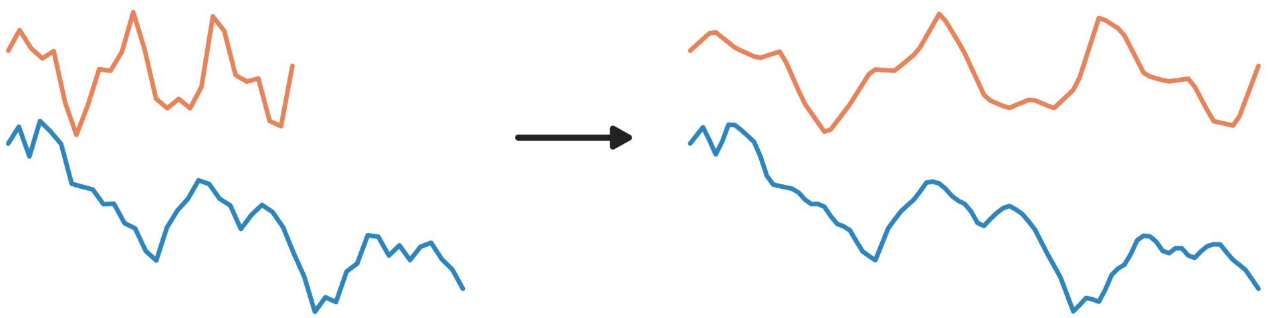


Fit the context length requirements

1. Padding



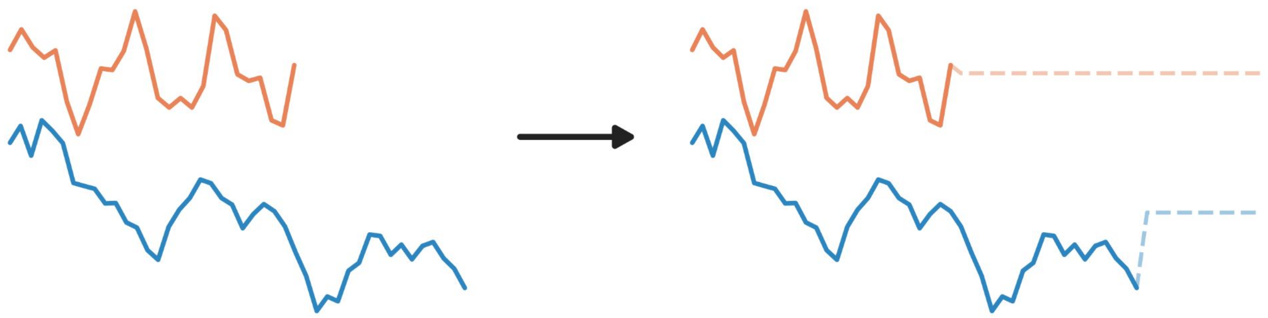
2. Resize by Interpolation



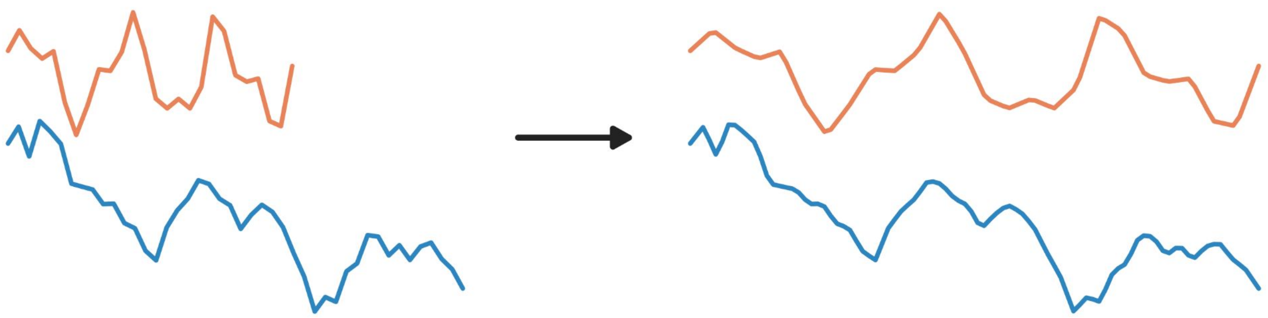
	Padding	Interpolation
Cons		
Forecasting or Classification?		

Fit the context length requirements

1. Padding



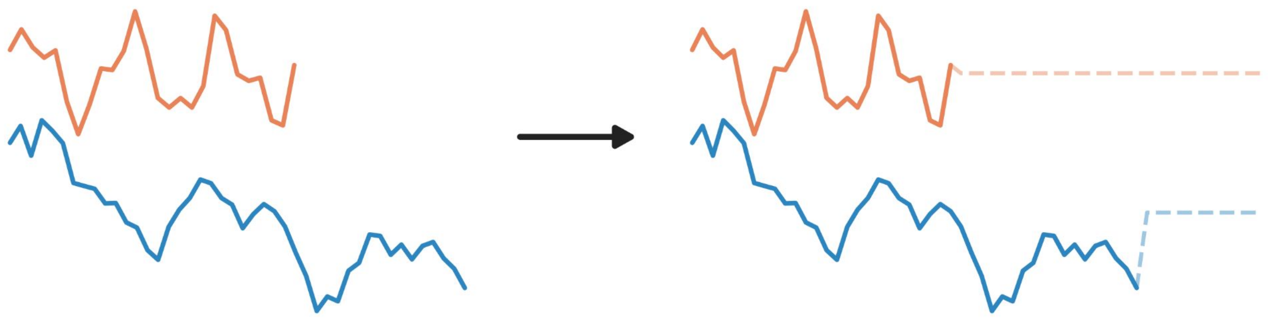
2. Resize by Interpolation



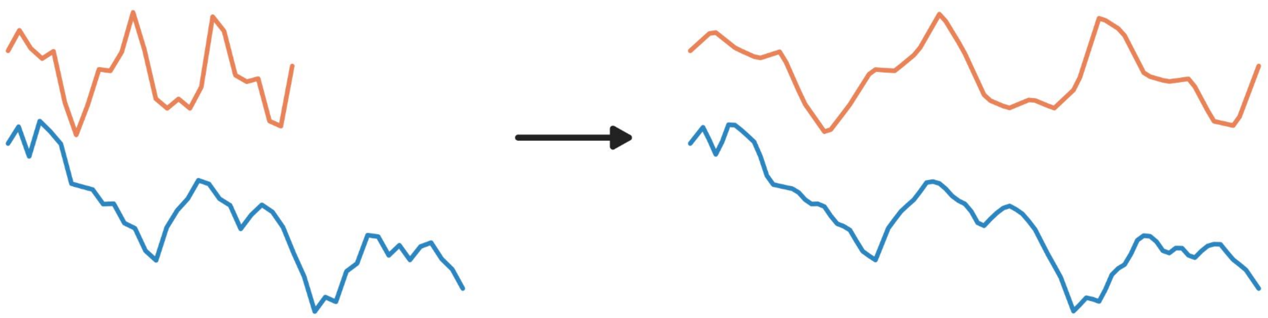
	Padding	Interpolation
Cons	Ambiguity: missing or signal?	Frequency spectrum shift
Forecasting or Classification?		

Fit the context length requirements

1. Padding



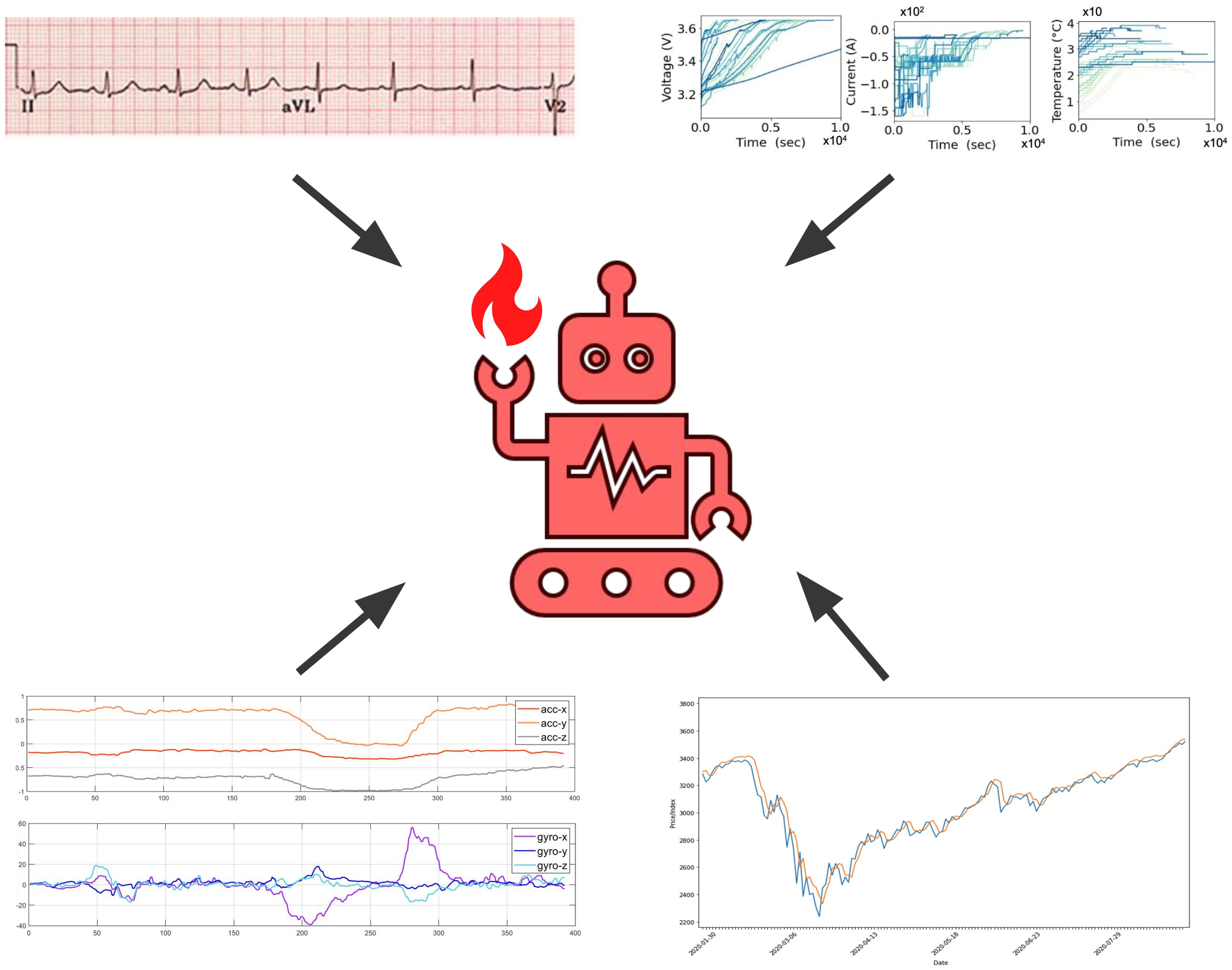
2. Resize by Interpolation



	Padding	Interpolation
Cons	Ambiguity: missing or signal?	Frequency spectrum shift
Forecasting or Classification?	Forecasting	Classification

Time Series Foundation Model (TSFM)

Step 1: Pre-training



Pre-train a projector using one of the two options:

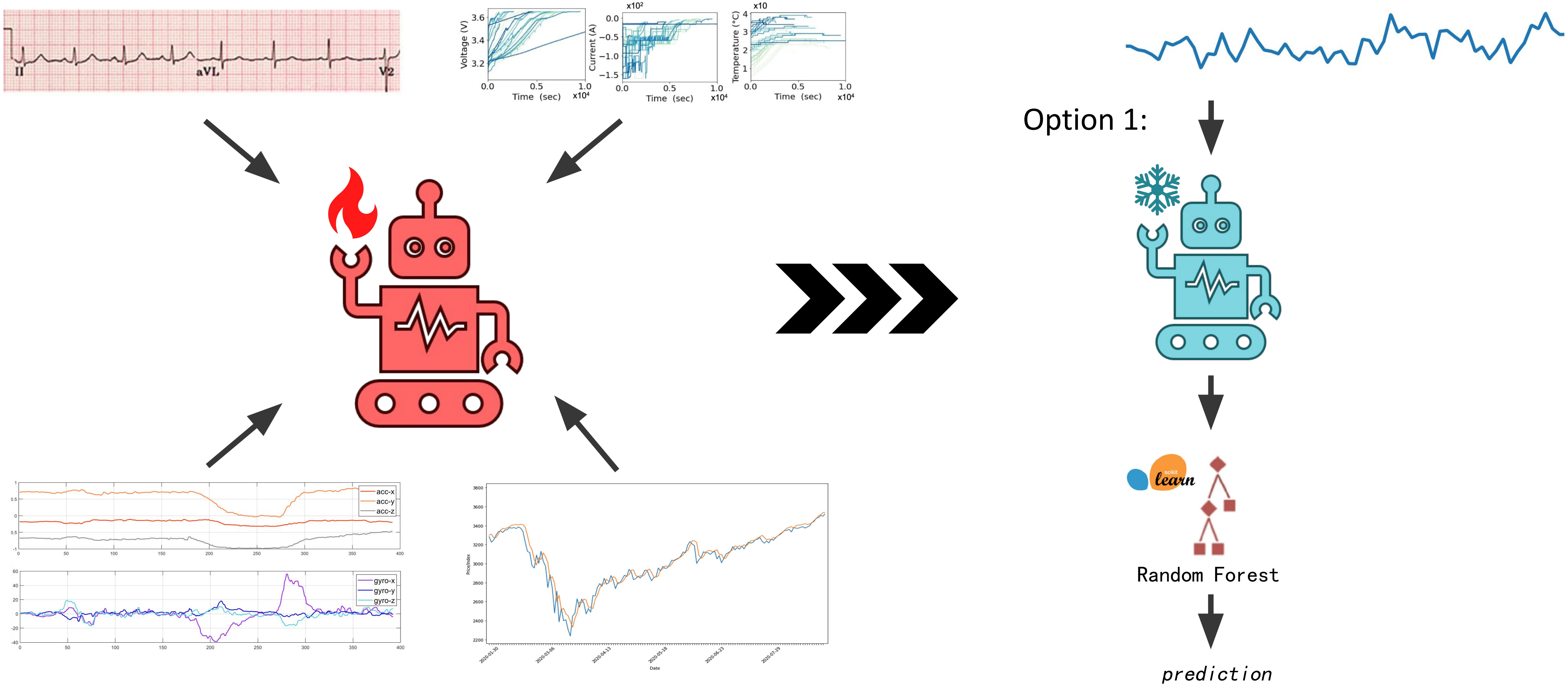
1. Unsupervised (self-supervised):
   a. <u>Contrastive learning</u>.
   b. Masked reconstruction.
   c. Self-distillation.

2. Supervised (multi-task).

Time Series Foundation Model (TSFM)

Step 1: Pre-training

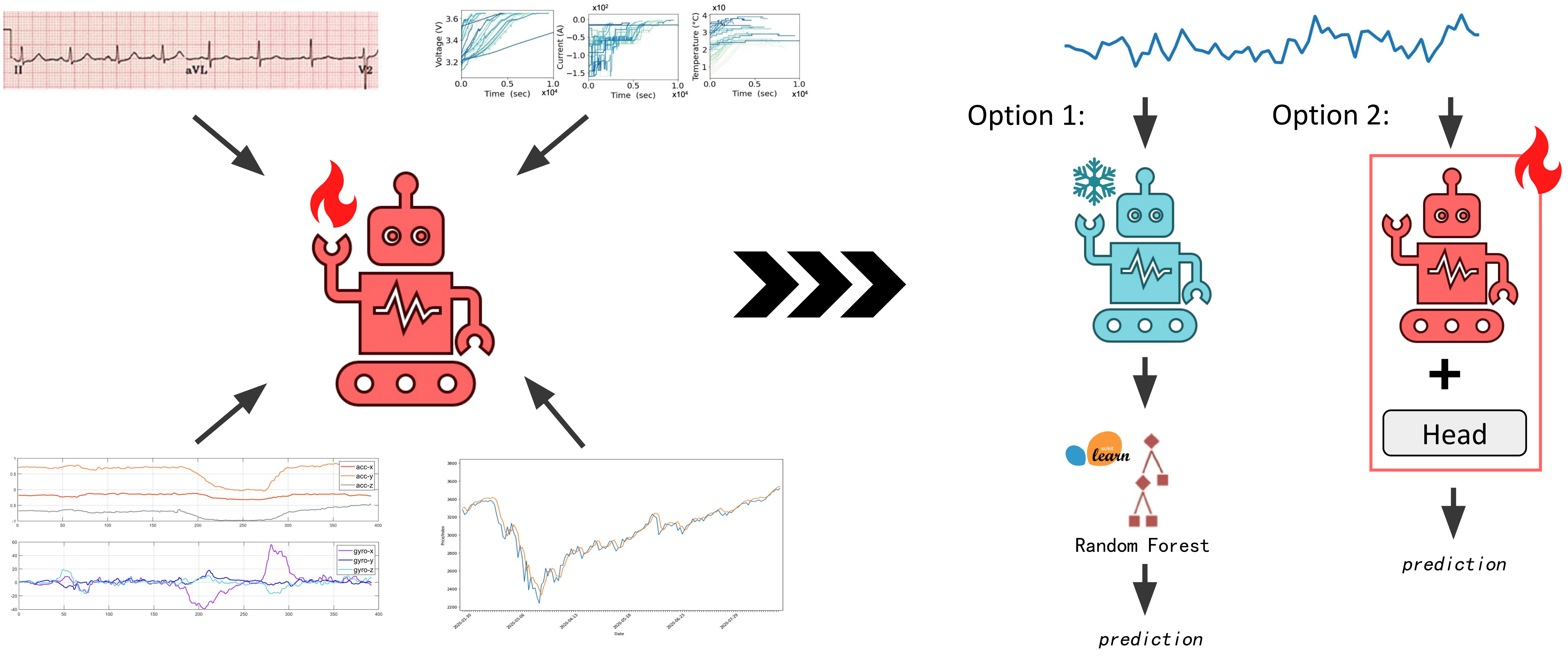
Step 2: Fitting to a New Task



Time Series Foundation Model (TSFM)

Step 1: Pre-training

Step 2: Fitting to a New Task



Transformers

Self-Attention is Kernel Smoothing

Scaled dot-product attention

$$\mathrm{Att}(\mathbf{X}) = \mathrm{softmax}\left[\frac{(\mathbf{X}\mathbf{W}_Q)(\mathbf{X}\mathbf{W}_K)^\top}{\sqrt{d_k}}\right]\mathbf{X}\mathbf{W}_V$$

$$\mathbf{X} \in \mathbb{R}^{t \times d}, \quad \mathbf{W}_Q, \mathbf{W}_K \in \mathbb{R}^{d \times d_k}$$

$$\mathbf{W}_V \in \mathbb{R}^{d \times d_v}$$

1. Queries, Keys, Values: linear transformations.
2. Quadratic expression inside of softmax.

Self-Attention is Kernel Smoothing

Scaled dot-product attention

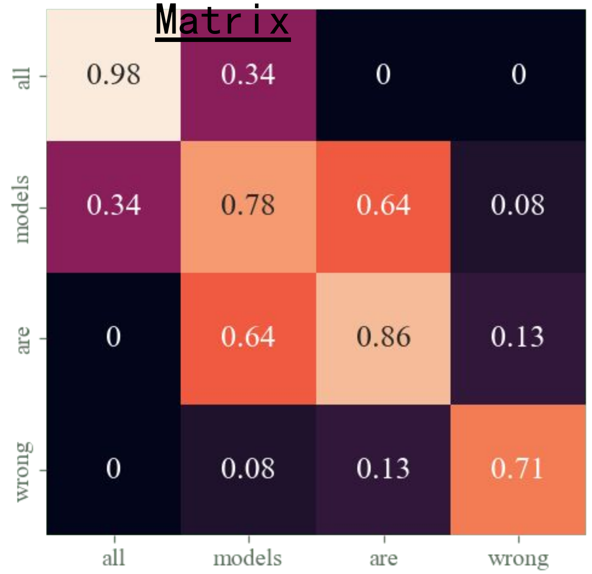
$$\mathrm{Att}(\mathbf{X}) = \mathrm{softmax}\left[\frac{(\mathbf{X}\mathbf{W}_Q)(\mathbf{X}\mathbf{W}_K)^\top}{\sqrt{d_k}}\right]\mathbf{X}\mathbf{W}_V$$

$$\mathbf{X} \in \mathbb{R}^{t\times d}, \quad \mathbf{W}_Q, \mathbf{W}_K \in \mathbb{R}^{d\times d_k}$$

$$\mathbf{W}_V \in \mathbb{R}^{d\times d_v}$$

1. Queries, Keys, Values: linear transformations.
2. Quadratic expression inside of softmax.
3. Softmax gives non-linearity + interpretability.
4. Attention matrix: relationship between tokens.

Attention Matrix



Self-Attention is Kernel Smoothing

Nadaraya–Watson estimator

$$\hat{y}(\mathbf{x}) = \sum_{i=1}^{t} \frac{k(\mathbf{x},\, \mathbf{x}_i)}{\sum_{j=1}^{t} k(\mathbf{x},\, \mathbf{x}_j)}\, y_i$$

with the (asymmetric) kernel

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\frac{(\mathbf{x}_i \mathbf{W}_Q)(\mathbf{x}_j \mathbf{W}_K)^\top}{\sqrt{d_k}}\right)$$

Scaled dot-product attention

$$\mathrm{Att}(\mathbf{X}) = \mathrm{softmax}\left[\frac{(\mathbf{X}\mathbf{W}_Q)(\mathbf{X}\mathbf{W}_K)^\top}{\sqrt{d_k}}\right]\mathbf{X}\mathbf{W}_V$$

$$\mathbf{X} \in \mathbb{R}^{t \times d}, \quad \mathbf{W}_Q, \mathbf{W}_K \in \mathbb{R}^{d \times d_k}$$

$$\mathbf{W}_V \in \mathbb{R}^{d \times d_v}$$

Each output row is a kernel-smoothed estimate

$$\mathrm{Att}(\mathbf{X})_i = \sum_j \frac{k(\mathbf{x}_i, \mathbf{x}_j)}{\sum_l k(\mathbf{x}_i, \mathbf{x}_l)}\, \mathbf{x}_j \mathbf{W}_V = \hat{y}(\mathbf{x}_i)$$

$\mathbf{x}$ *query* $\leftrightarrow$ $\mathbf{x}_i$ *token i*

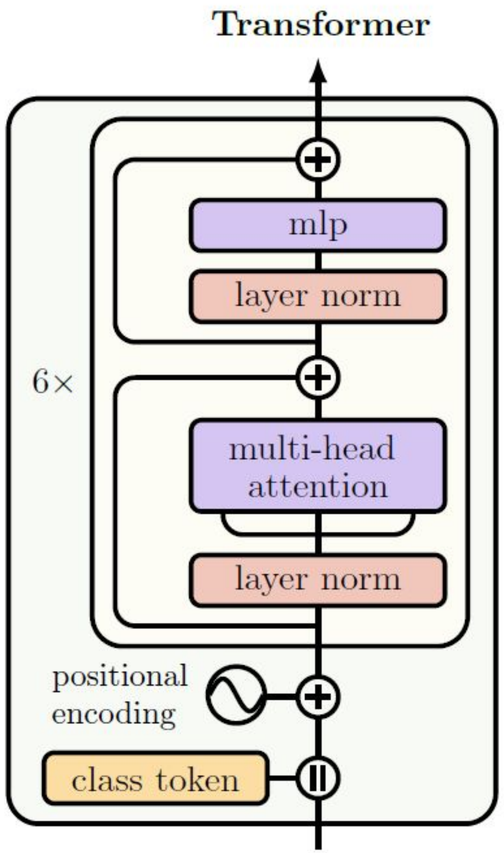
y_j *labels* $\leftrightarrow$ $\mathbf{x}_j \mathbf{W}_V$ *values*

Transformer

1. 1 transformer layer =
   Pre-norm + Att + Skip + Pre-norm + MLP + Skip.

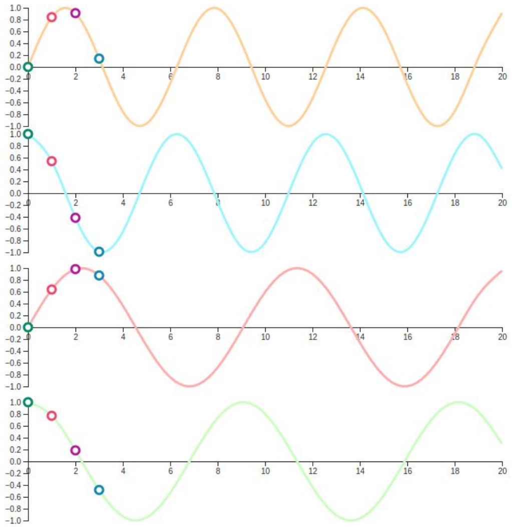
2. Class token is appended to the input tokens:
   - learnable vector,
   - aggregates information from the entire input,
   - its embedding is the final output.

3. Positional encoding:
   - incorporates the order of tokens,
   - encodes position into a high-dimensional vector.



Positional Encoding

Sinusoidal Positional Encoding



$d_{\text{model}} = 50$

p0	p1	p2	p3	
0.000	0.841	0.909	0.141	i=0
1.000	0.540	-0.416	-0.990	i=1
0.000	0.638	0.983	0.875	i=2
1.000	0.770	0.186	-0.484	i=3

Positional Encoding

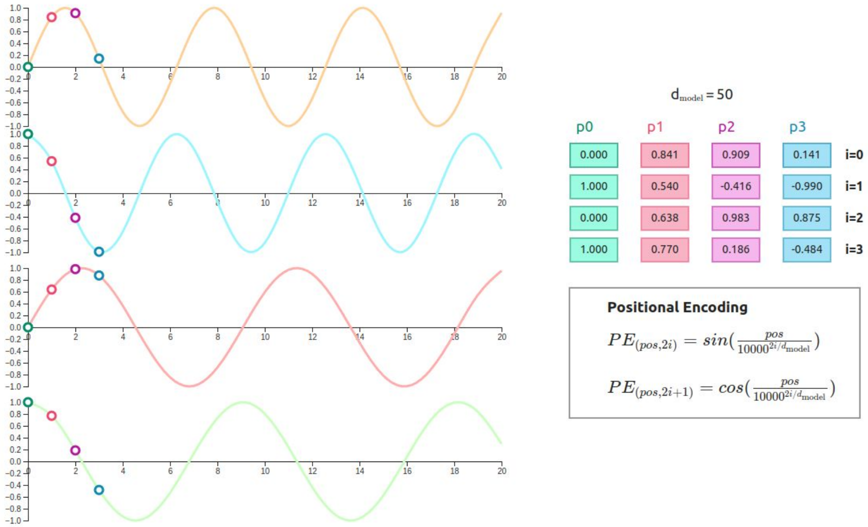
$$PE_{(pos,2i)} = sin(\frac{pos}{10000^{2i/d_{\text{model}}}})$$

$$PE_{(pos,2i+1)} = cos(\frac{pos}{10000^{2i/d_{\text{model}}}})$$

- Additive to the input.
- Absolute position.
- Simple, easy to try.

Positional Encoding

Sinusoidal Positional Encoding



- Additive to the input.
- Absolute position.
- Simple, easy to try.

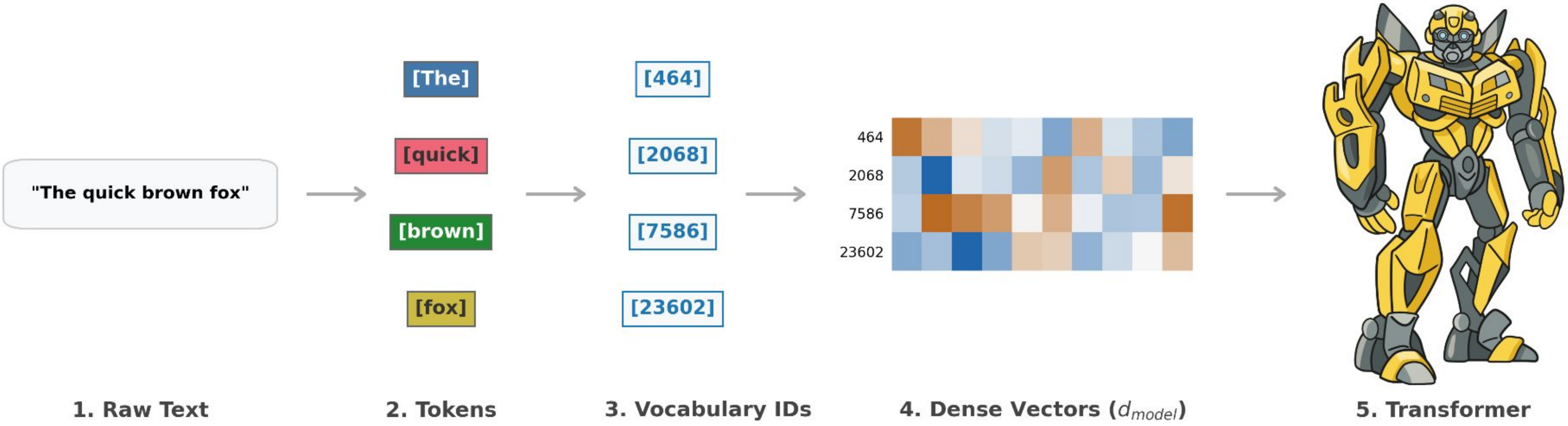
Rotary Positional Encoding

$$\boldsymbol{R}^d_{\Theta,m} = \begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos m\theta_{d/2} & -\sin m\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin m\theta_{d/2} & \cos m\theta_{d/2} \end{pmatrix}$$

$$\boldsymbol{q}_m^\intercal \boldsymbol{k}_n = (\boldsymbol{R}^d_{\Theta,m} \boldsymbol{W}_q \boldsymbol{x}_m)^\intercal (\boldsymbol{R}^d_{\Theta,n} \boldsymbol{W}_k \boldsymbol{x}_n) = \boldsymbol{x}^\intercal \boldsymbol{W}_q R^d_{\Theta,n-m} \boldsymbol{W}_k \boldsymbol{x}_n$$

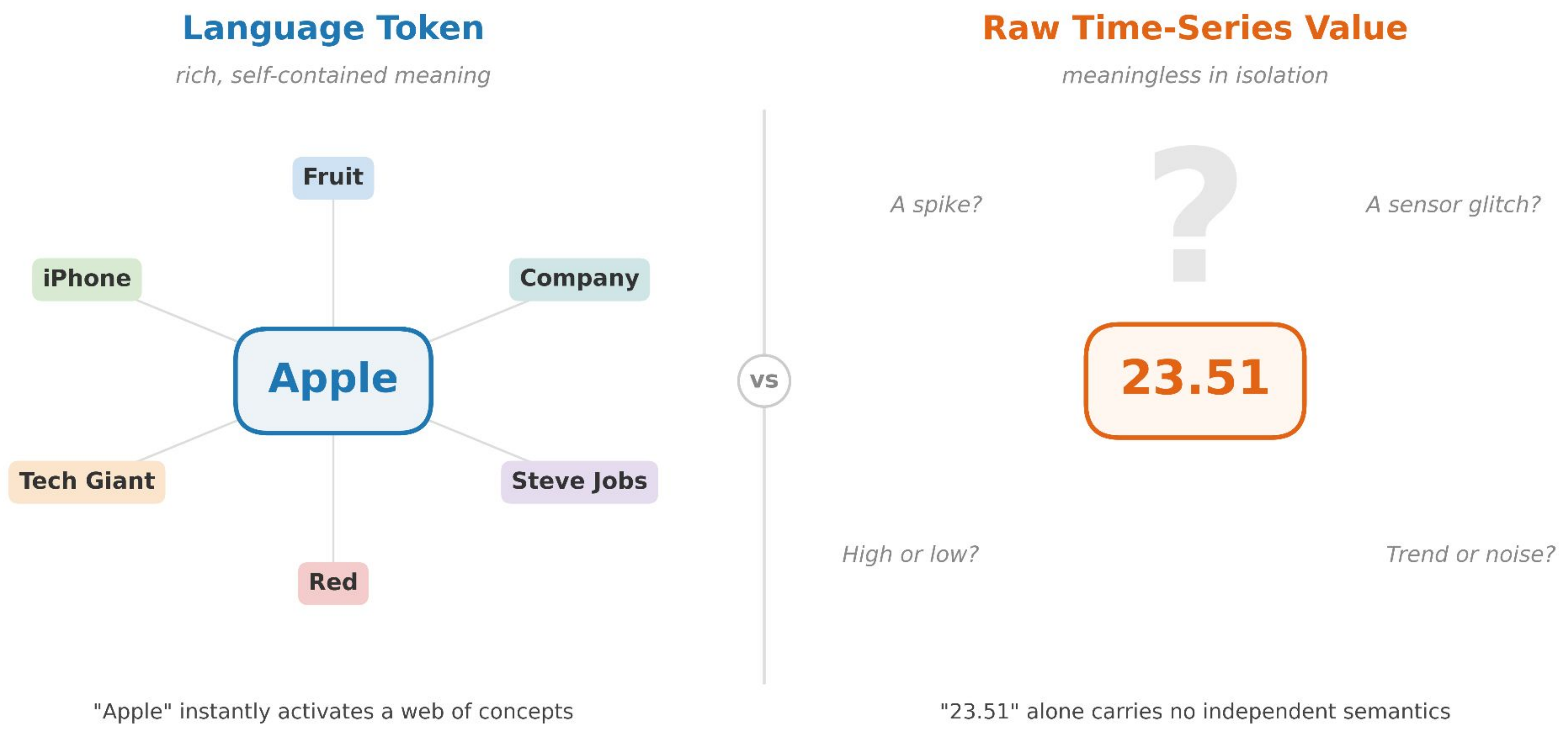
- Rotates keys and queries.
- Absolute rotation = relative distance.
- Standard for modern architectures.

Tokenization in LLMs



- 1 word is approx. 1-2 tokens.
- For each token ID, we learn a dense vector.
- These token embeddings are fed to Transformer.

What About Transformers for Time Series?



Time Series Transformers: Timeline

- Informer (AAAI 2021)
- Autoformer (NeuIPS 2021)
- Pyraformer (ICLR 2022)
- FEDformer (ICML 2022)

❑ Outperform a Vanilla Transformer,

❑ Roughly speaking, point-wise tokenization,

❑ Did not compare with baselines…

Time Series Transformers: Timeline

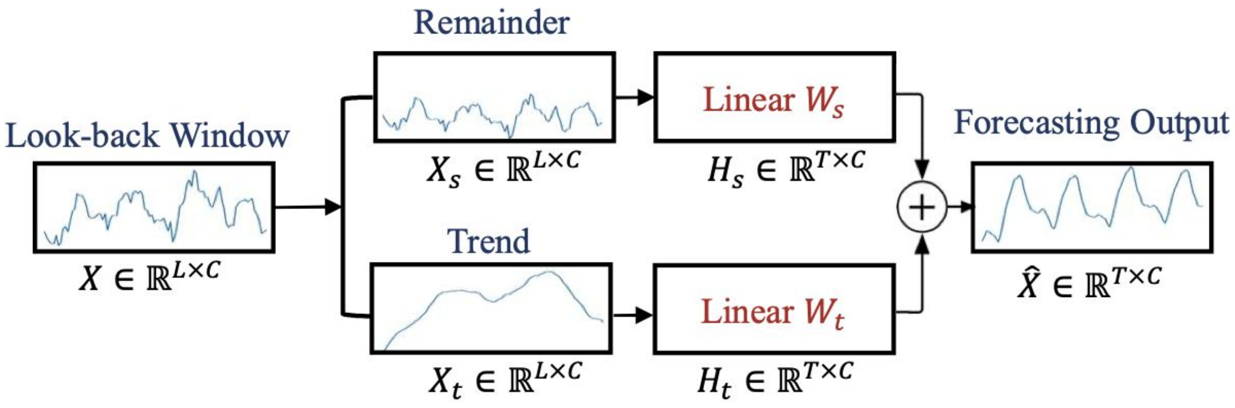
First TS Transformers | **What's Going On?** | **Foundation Models Era**

2017 — 2022 — 2024 — now

- Informer (AAAI 2021)
- Autoformer (NeuIPS 2021)
- Pyraformer (ICLR 2022)
- FEDformer (ICML 2022)

- DLinear (AAAI 2022)

❑ DLinear: linear forecasters easily beat all these transformers…



Time Series Transformers: Timeline

First TS Transformers | **What's Going On?** | **Foundation Models Era**

2017 — 2022 — 2024 — now

- Informer (AAAI 2021)
- Autoformer (NeuIPS 2021)
- Pyraformer (ICLR 2022)
- FEDformer (ICML 2022)

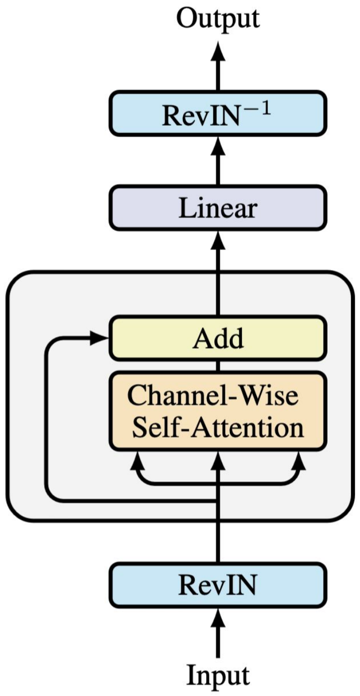
- DLinear (AAAI 2022)
- PatchTST (ICLR 2023)
- SAMFormer (ICML 2024)

❑ SAMFormer:

➔ Sharpness-aware minimization helps.

➔ The more shallow transformer, the better.

➔ Channel-wise attention is better.



Time Series Transformers: Timeline

First TS Transformers — 2017

- Informer (AAAI 2021)
- Autoformer (NeuIPS 2021)
- Pyraformer (ICLR 2022)
- FEDformer (ICML 2022)

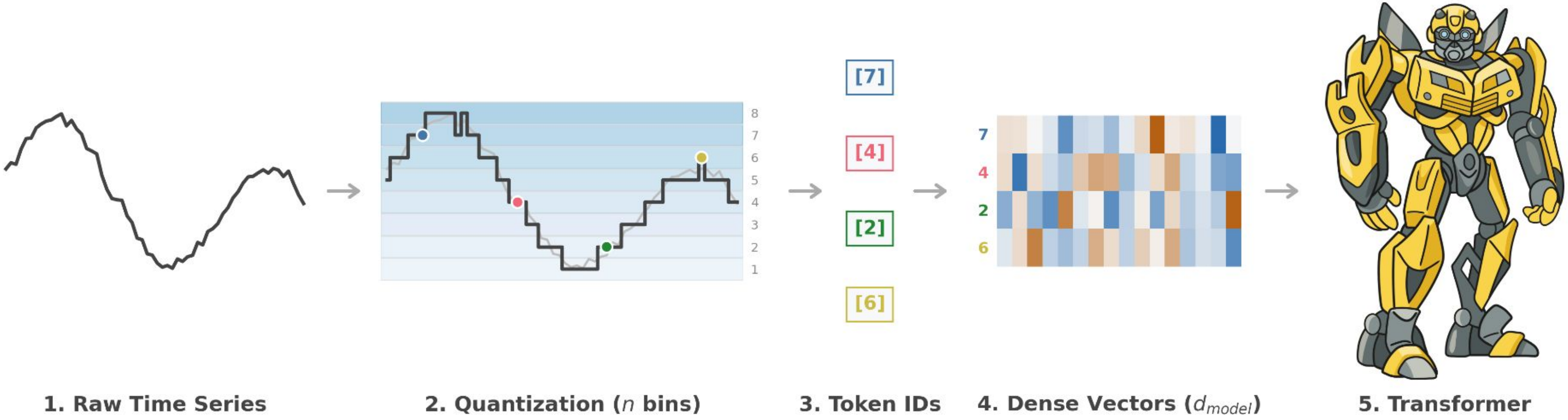
What's Going On? — 2022

- DLinear (AAAI 2022)
- PatchTST (ICLR 2023)
- SAMFormer (ICML 2024)

Foundation Models Era — 2024 – now

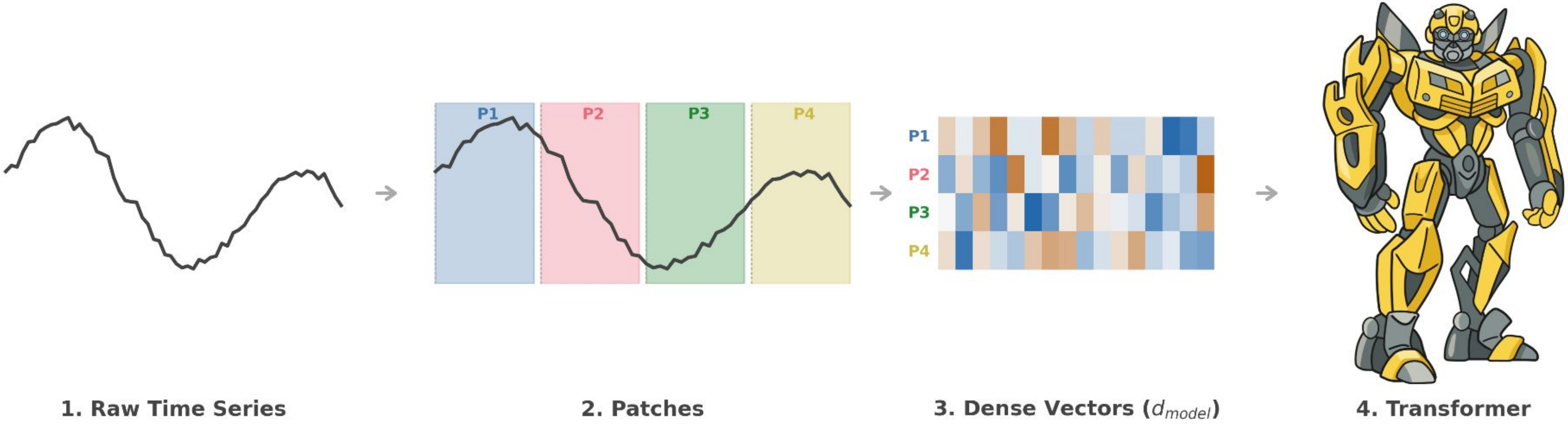
- Quantization (LLMTime, Chronos)
- Patching (Most of TSFMs)

Quantization



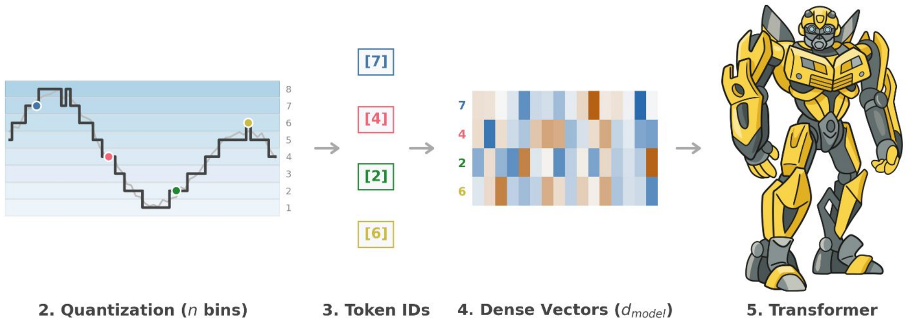
- ❑ Say all normalized time series live within [-5, 5].
- ❑ Vocabulary: bins that cover the range of values.
- ❑ A value fall into a bin *i* => token *i*.
- ❑ You can then use any LLM you want!

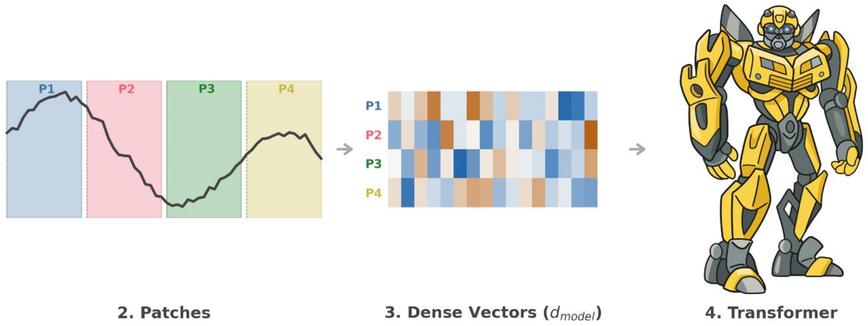
Patching



- Patch time series (non-overlapping or overlapping windows).
- 1 patch = 1 token.
- Each patch is projected to a dense vector (typically, by a linear layer).

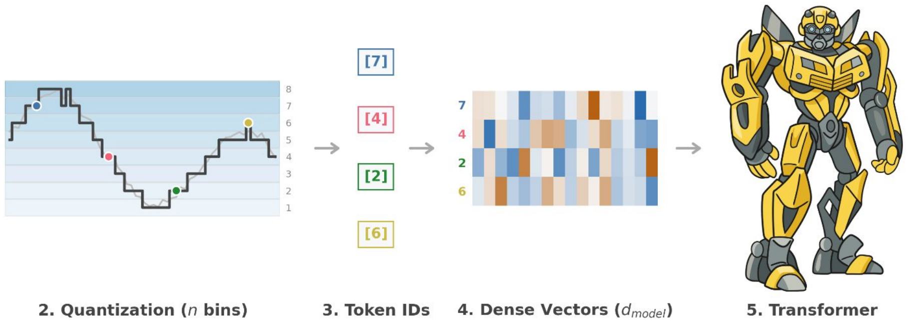
Quantization vs Patching

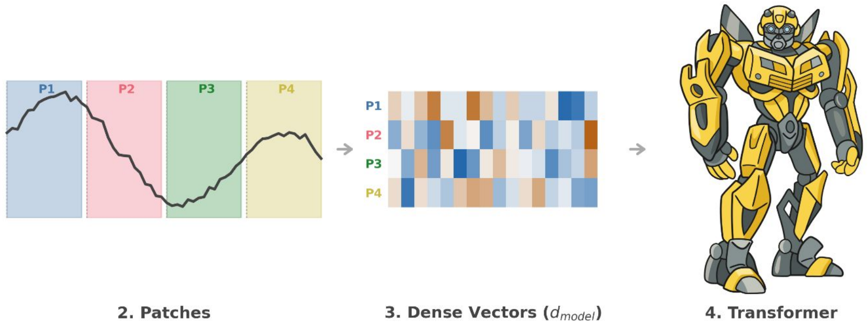




	Quantization	Patching
Space complexity		

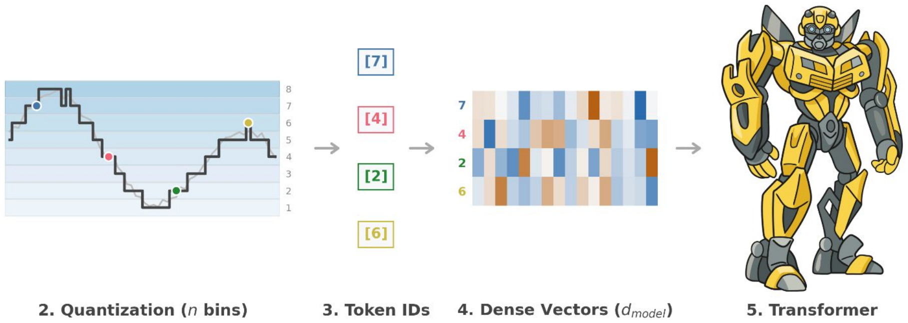
Quantization vs Patching

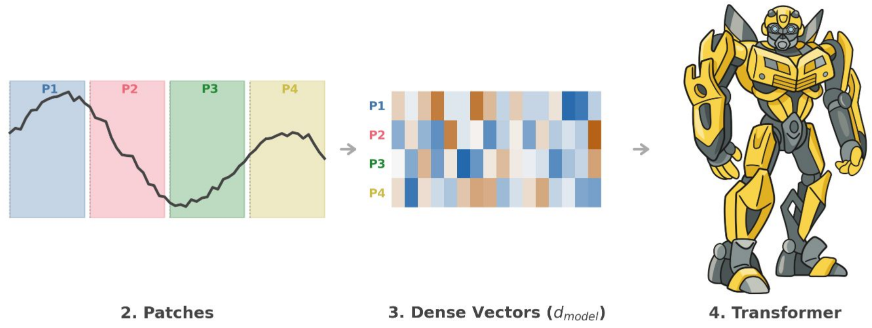




	Quantization	Patching
Space complexity	Higher	Lower
Robustness to noise		

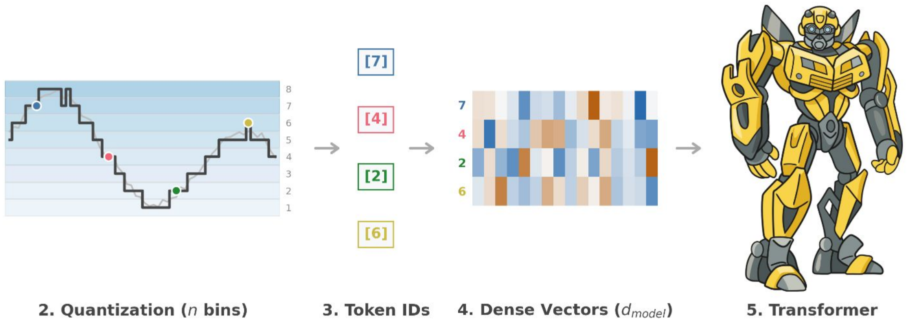
Quantization vs Patching

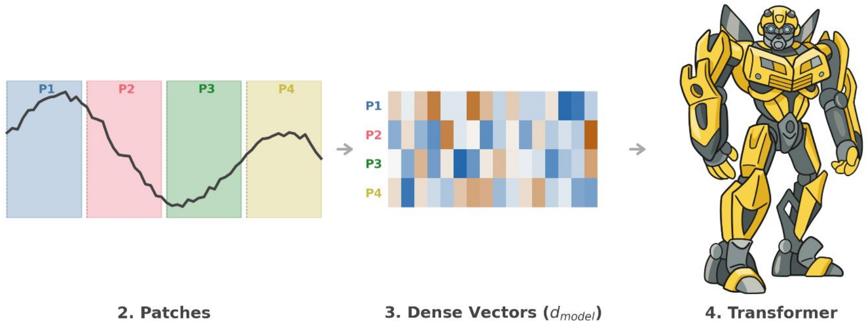




	Quantization	Patching
Space complexity	Higher	Lower
Robustness to noise	Lower	Higher
Input: Text + Time Series		

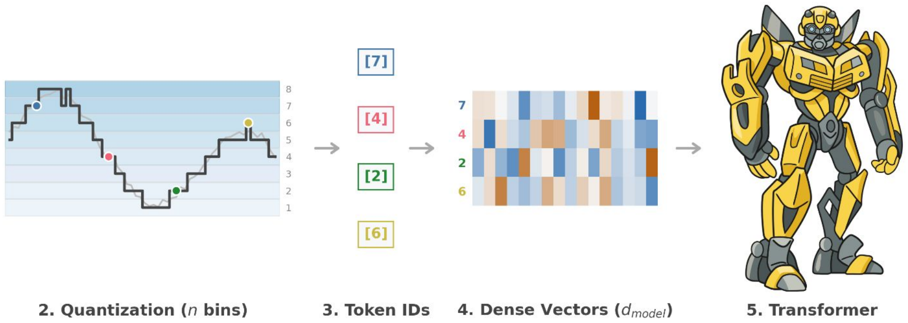
Quantization vs Patching

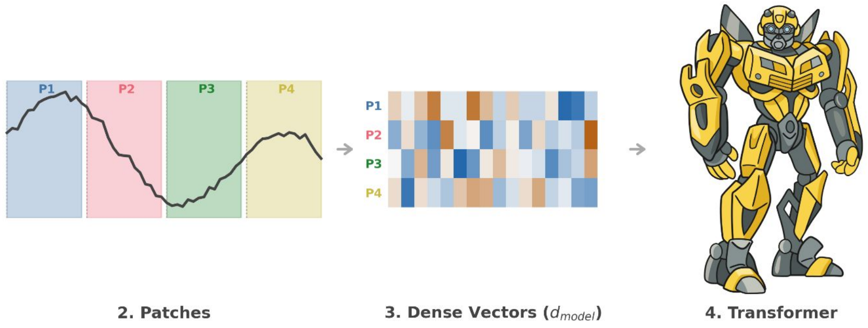




	Quantization	Patching
Space complexity	Higher	Lower
Robustness to noise	Lower	Higher
Input: Text + Time Series	Yes	No
Leverage a pre-trained FM from another modality	Yes: LLM	

Quantization vs Patching

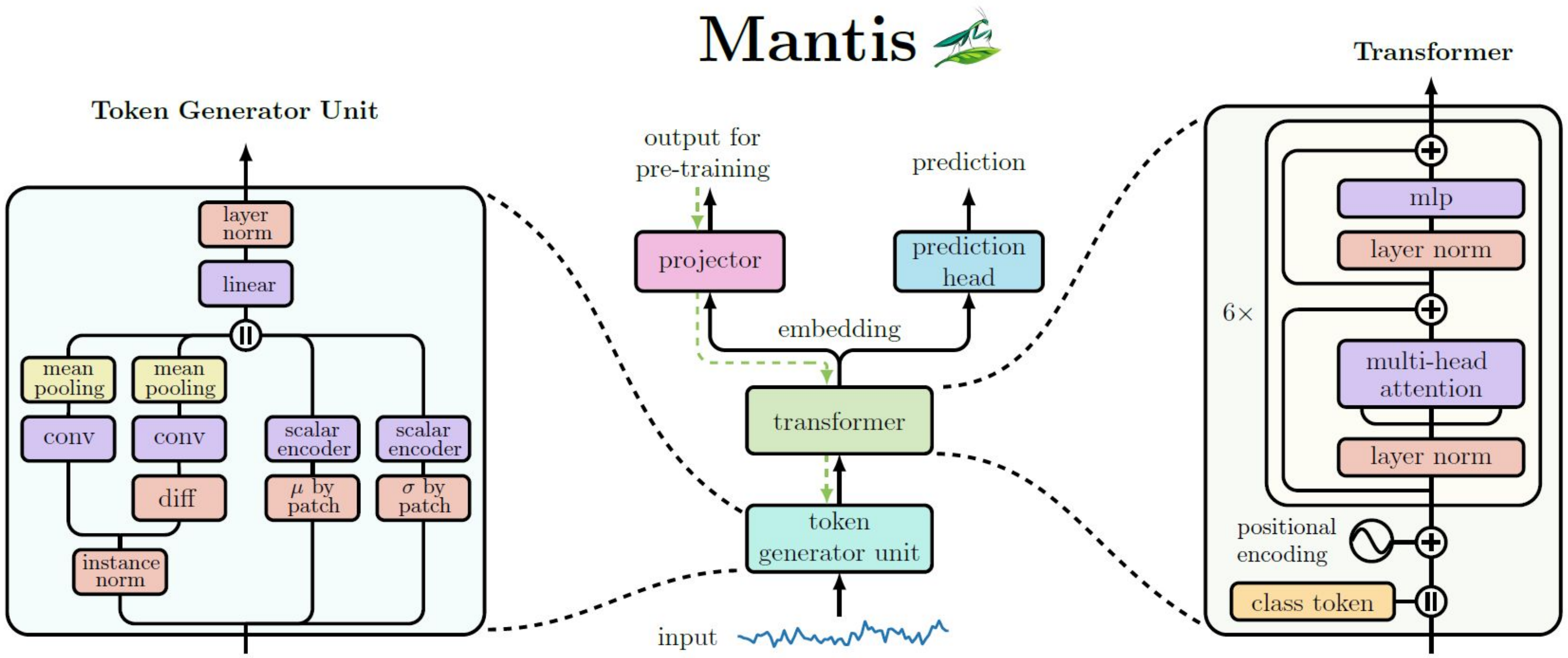




	Quantization	Patching
Space complexity	Higher	Lower
Robustness to noise	Lower	Higher
Input: Text + Time Series	Yes	No
Leverage a pre-trained FM from another modality	Yes: LLM	Yes: ViT

Mantis: Architecture

Architecture

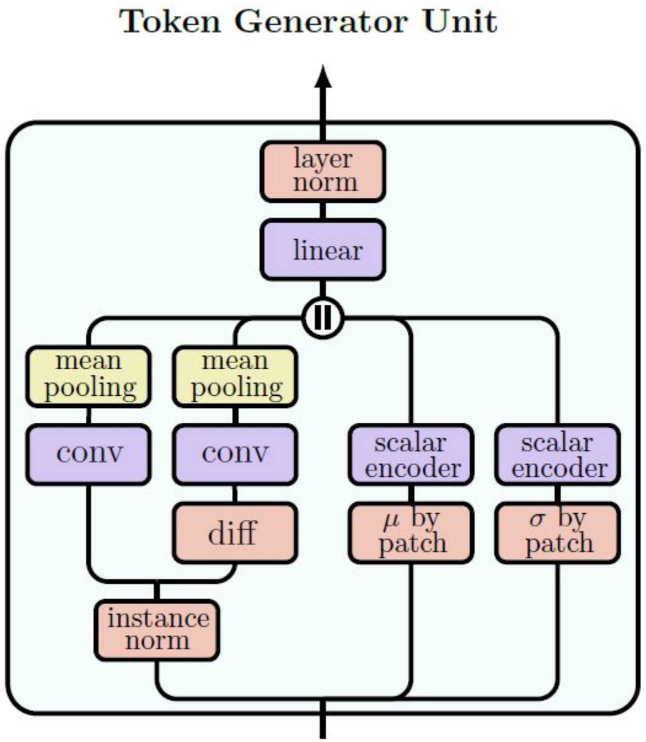


Tokenization is key to unlock the power of the transformer.

1. **Token Generator Unit:** converts time series into meaningful tokens.
2. **Transformer:** projects tokens to a new representation space.

Token Generator Unit

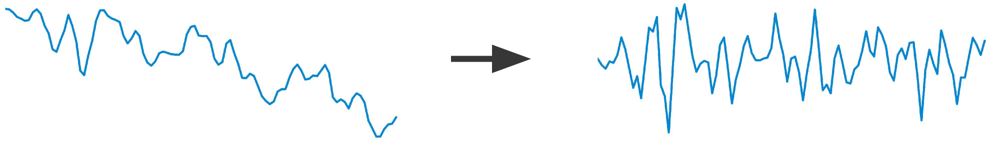
1. <u>32 x-patches</u>:
   - norm x → conv → mean pooling.

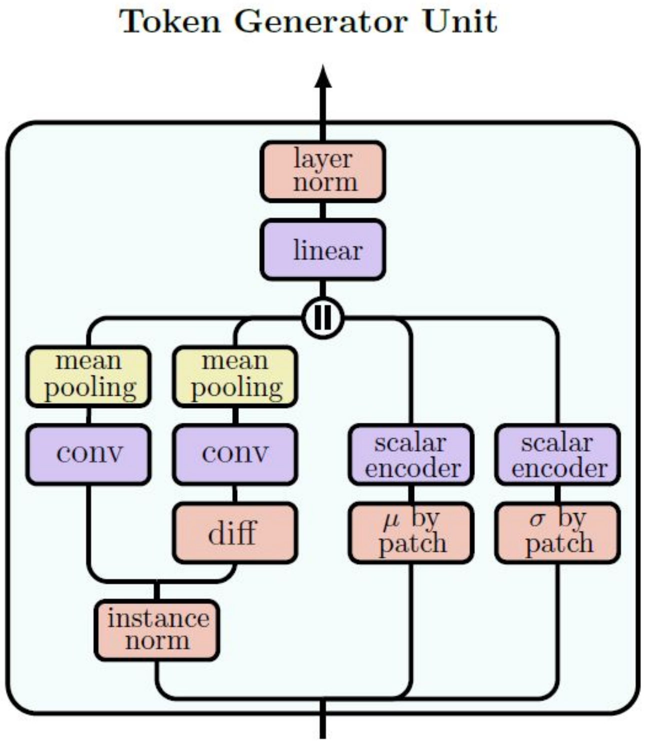


Token Generator Unit

1. 32 x-patches:
   - norm x ⟶ conv ⟶ mean pooling.

2. 32 diff-x-patches:
   - norm x ⟶ diff ⟶ conv ⟶ mean pooling.
   - diff: x[t]-x[t-1], makes time series stationary.

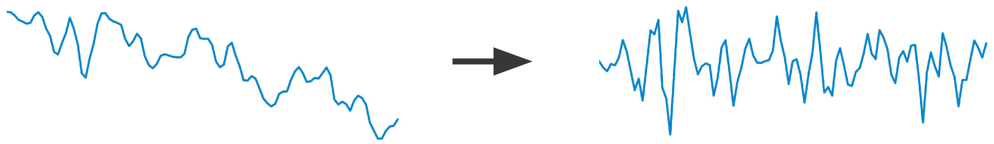




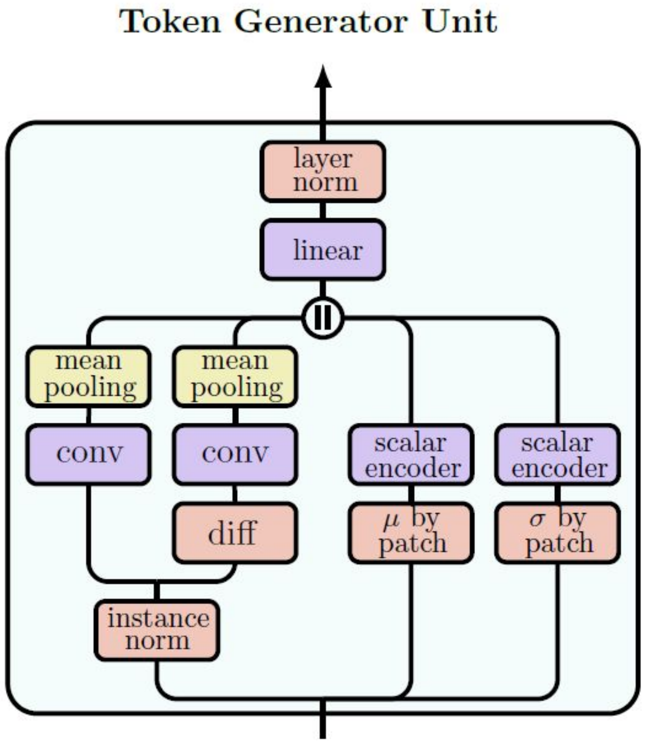
Token Generator Unit

1. <u>32 x-patches</u>:
   - norm x ⟶ conv ⟶ mean pooling.

2. <u>32 diff-x-patches</u>:
   - norm x ⟶ diff ⟶ conv ⟶ mean pooling.
   - diff: x[t]-x[t-1], makes time series stationary.



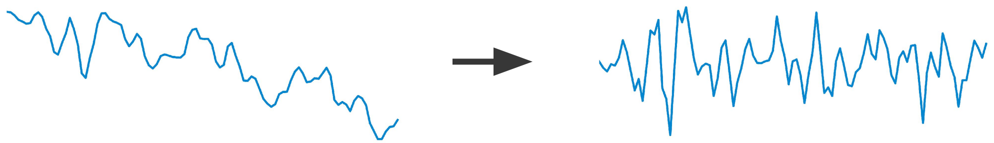
   - ablation study: diff improves performance.



Token Generator Unit

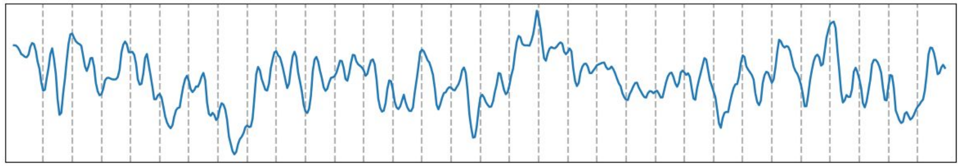
1. 32 x-patches:
   - norm x ⟶ conv ⟶ mean pooling.

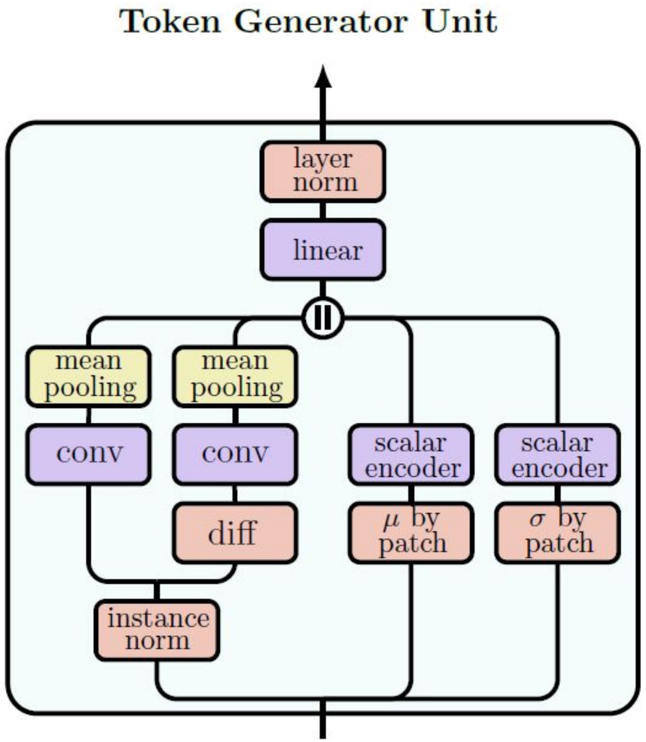
2. 32 diff-x-patches:
   - norm x ⟶ diff ⟶ conv ⟶ mean pooling.
   - diff: x[t]-x[t-1], makes time series stationary.



   - ablation study: diff improves performance.

3. Scalar encoder (Lin et al., 2024):
   - split time series into 32 non-overlapping patches.
   - compute mean and std for each patch.

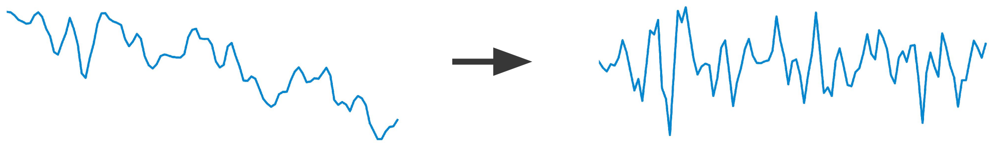




Token Generator Unit

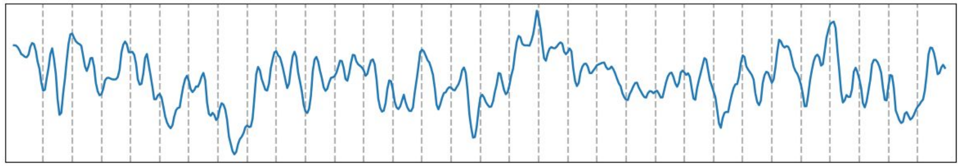
1. <u>32 x-patches</u>:
   - norm x ⟶ conv ⟶ mean pooling.

2. <u>32 diff-x-patches</u>:
   - norm x ⟶ diff ⟶ conv ⟶ mean pooling.
   - diff: x[t]-x[t-1], makes time series stationary.

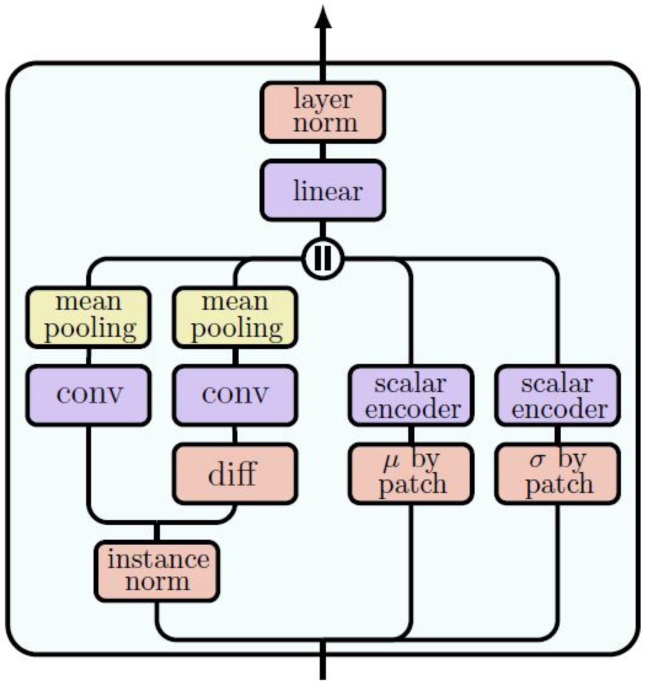


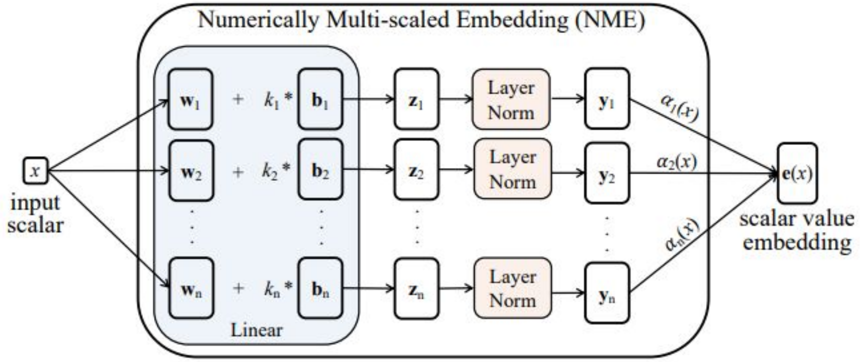
   - ablation study: diff improves performance.

3. <u>Scalar encoder</u> (Lin et al., 2024):
   - split time series into 32 non-overlapping patches.
   - compute mean and std for each patch.
   - encode each scalar by a high-dimensional vector.



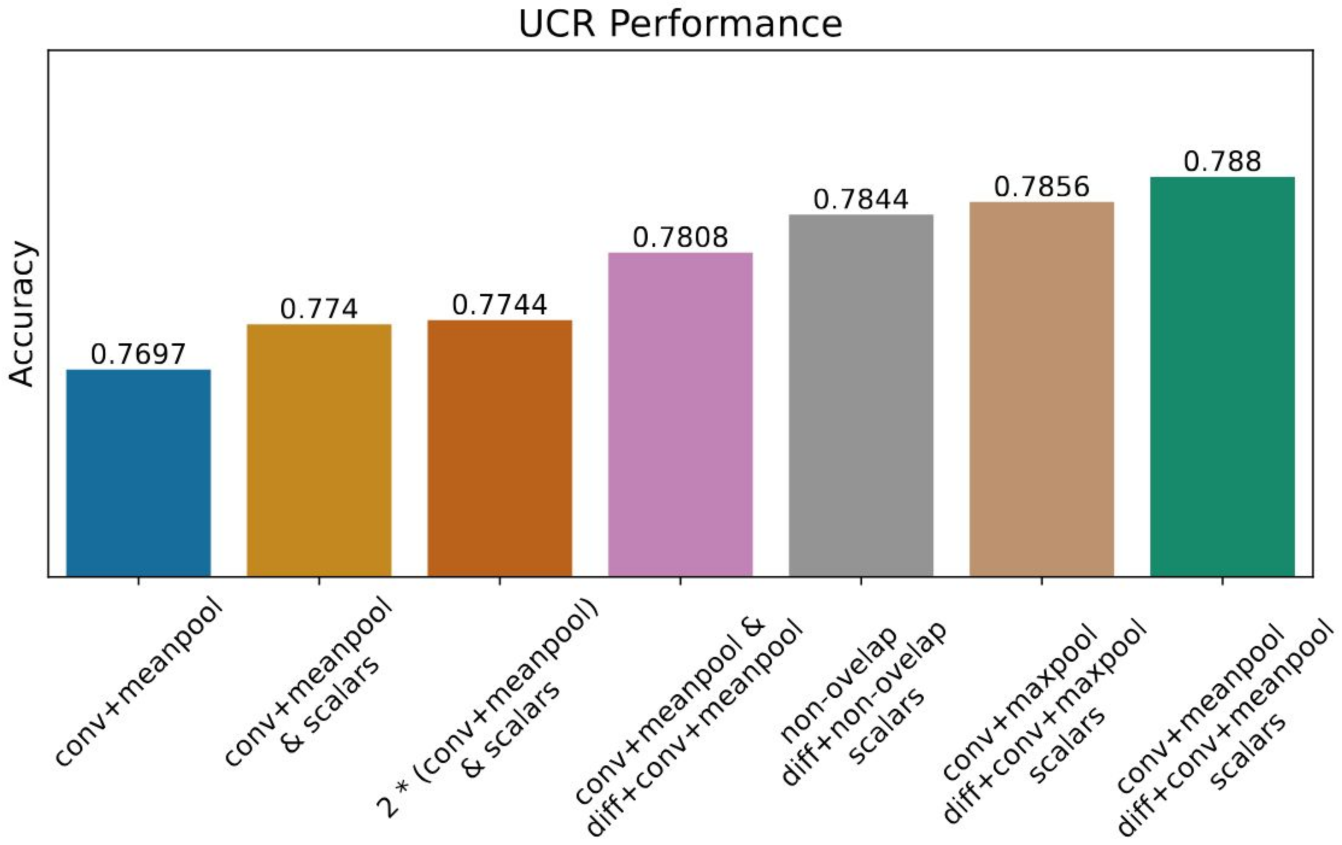
Token Generator Unit

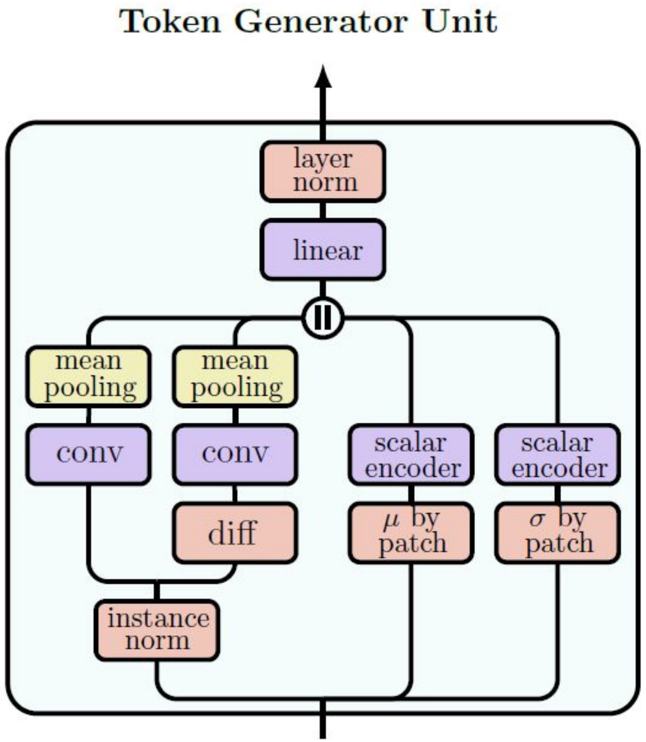




Token Generator Unit: Ablation Study

Ablation study validates the proposed scheme.

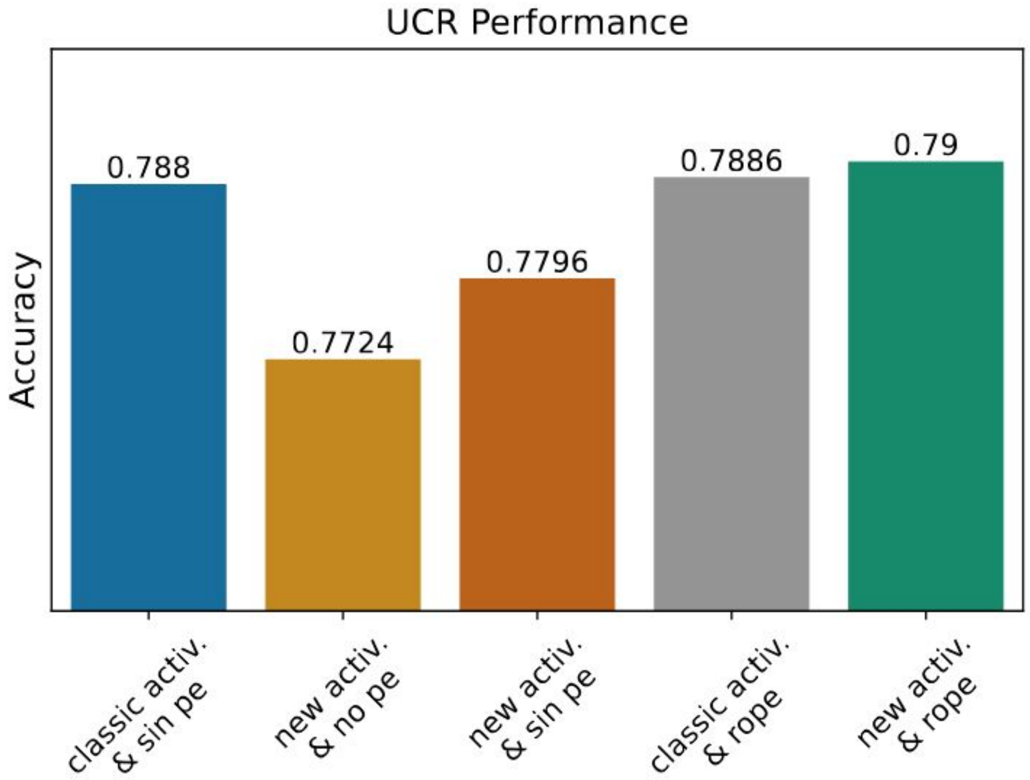




Transformer: Ablation Study

Some ablations:

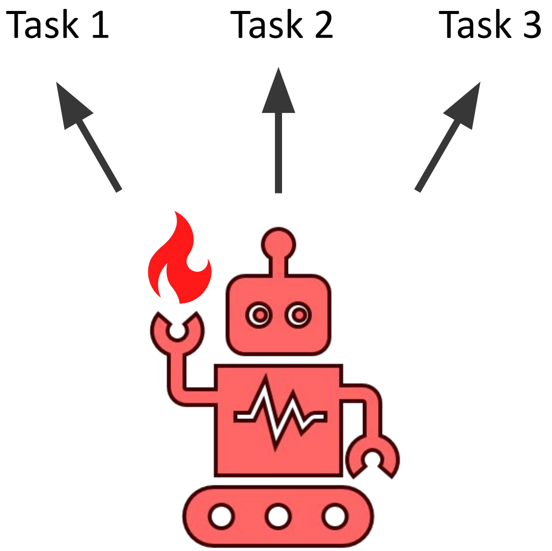
- Normalization: LayerNorm vs RMSNorm
- Activation: GELU vs SwiGLU
- Sin PE vs ROPE



Pre-training

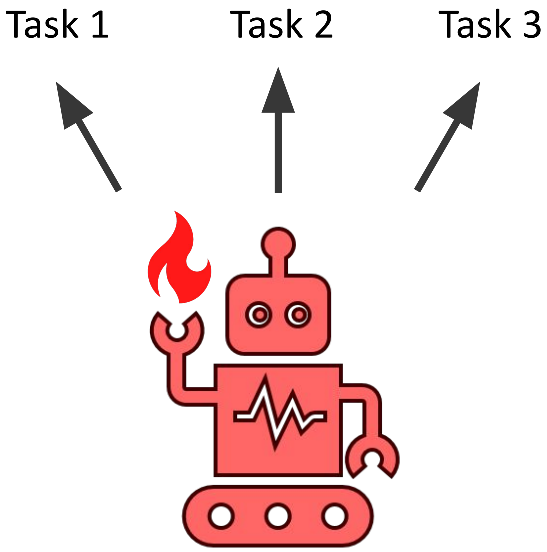
Types of Pre-training

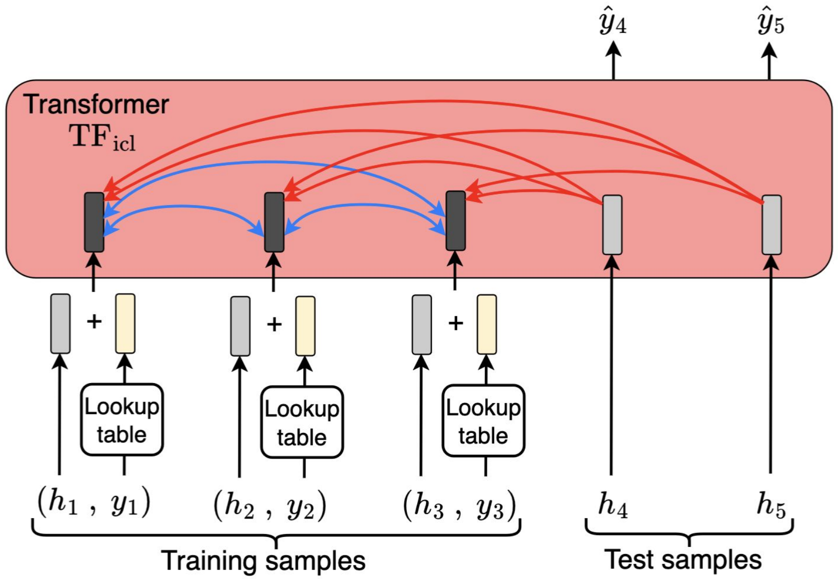
1. Supervised: use both features and targets for pre-training.
   a. Multi-task learning. ← hard to pre-train with many tasks.



Types of Pre-training

1. Supervised: use both features and targets for pre-training.
    a. Multi-task learning. ← hard to pre-train with many tasks.
    b. In-context learning. ← requires hard-fixing max. number of classes and features.

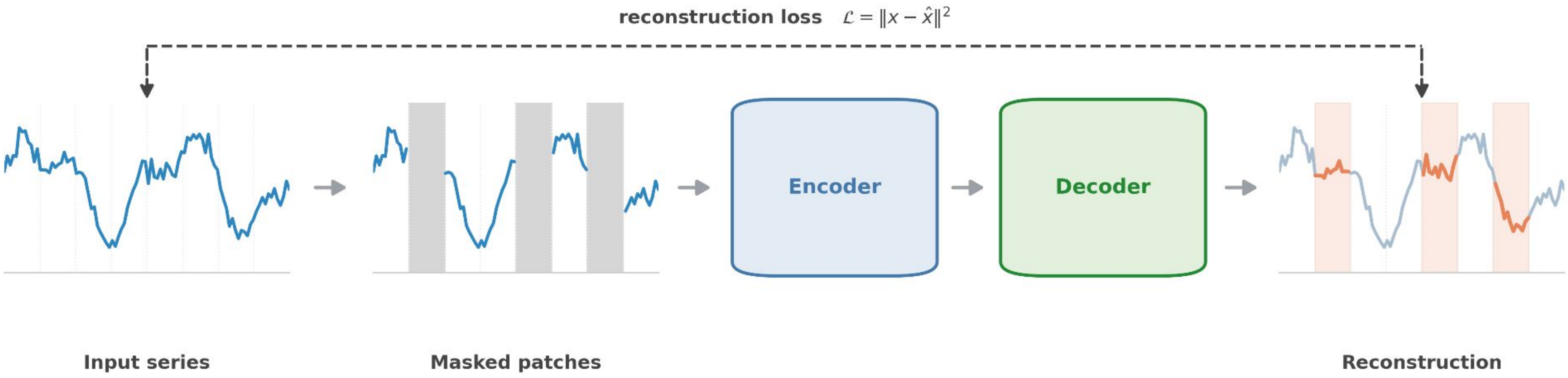




Types of Pre-training

1. Supervised: use both features and targets for pre-training.
   a. Multi-task learning.
   b. In-context learning.

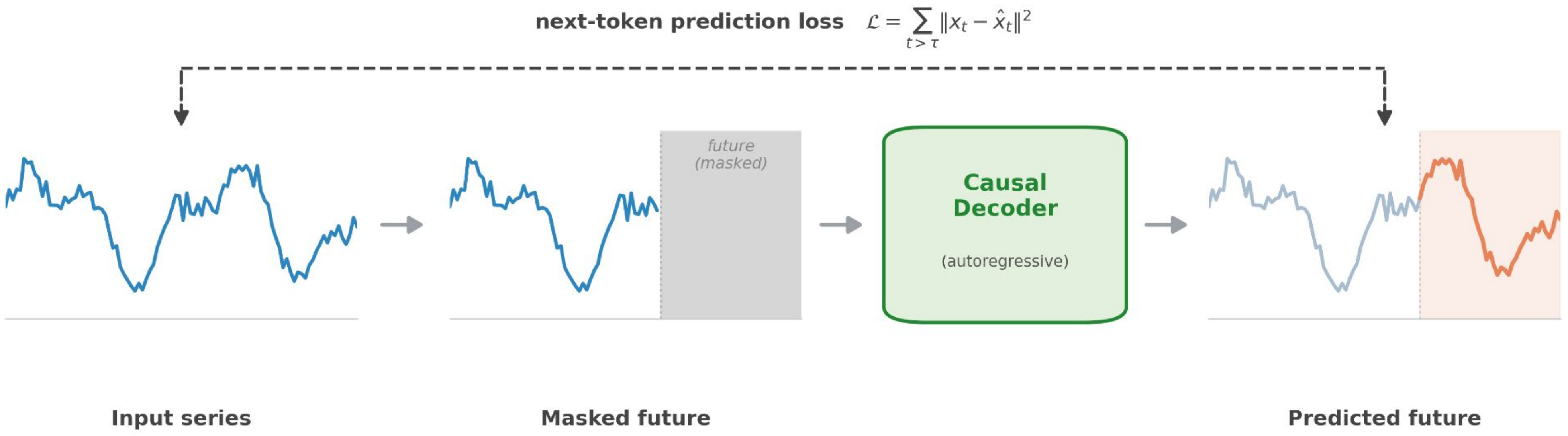
2. Unsupervised (self-supervised) pre-training: use only features
   a. Masked reconstruction (auto-encoder).



Types of Pre-training

1. Supervised: use both features and targets for pre-training.
   a. Multi-task learning.
   b. In-context learning.

2. Unsupervised (self-supervised) pre-training: use only features
   a. Masked reconstruction (auto-encoder).
   b. Next-toke prediction (decoder-only).

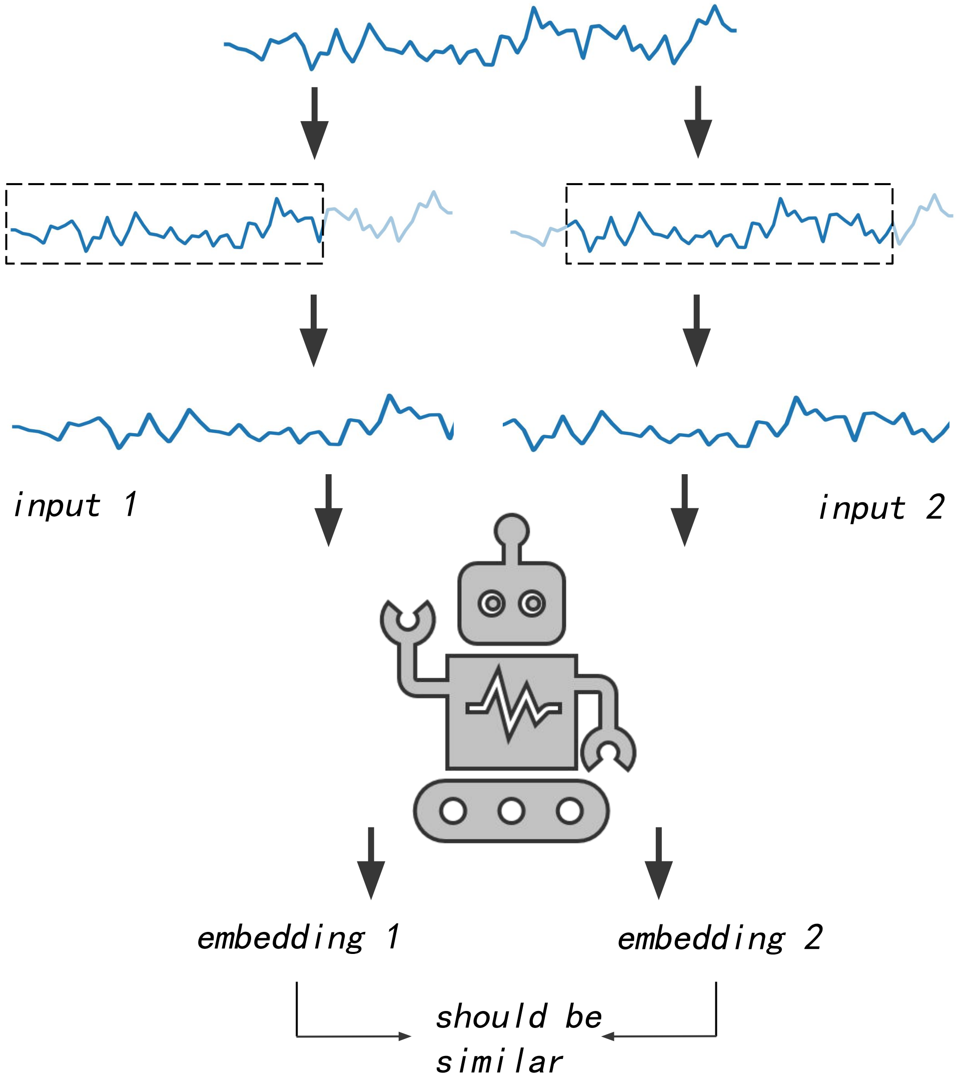


Types of Pre-training

1. Supervised: use both features and targets for pre-training.
   a. Multi-task learning.
   b. In-context learning.

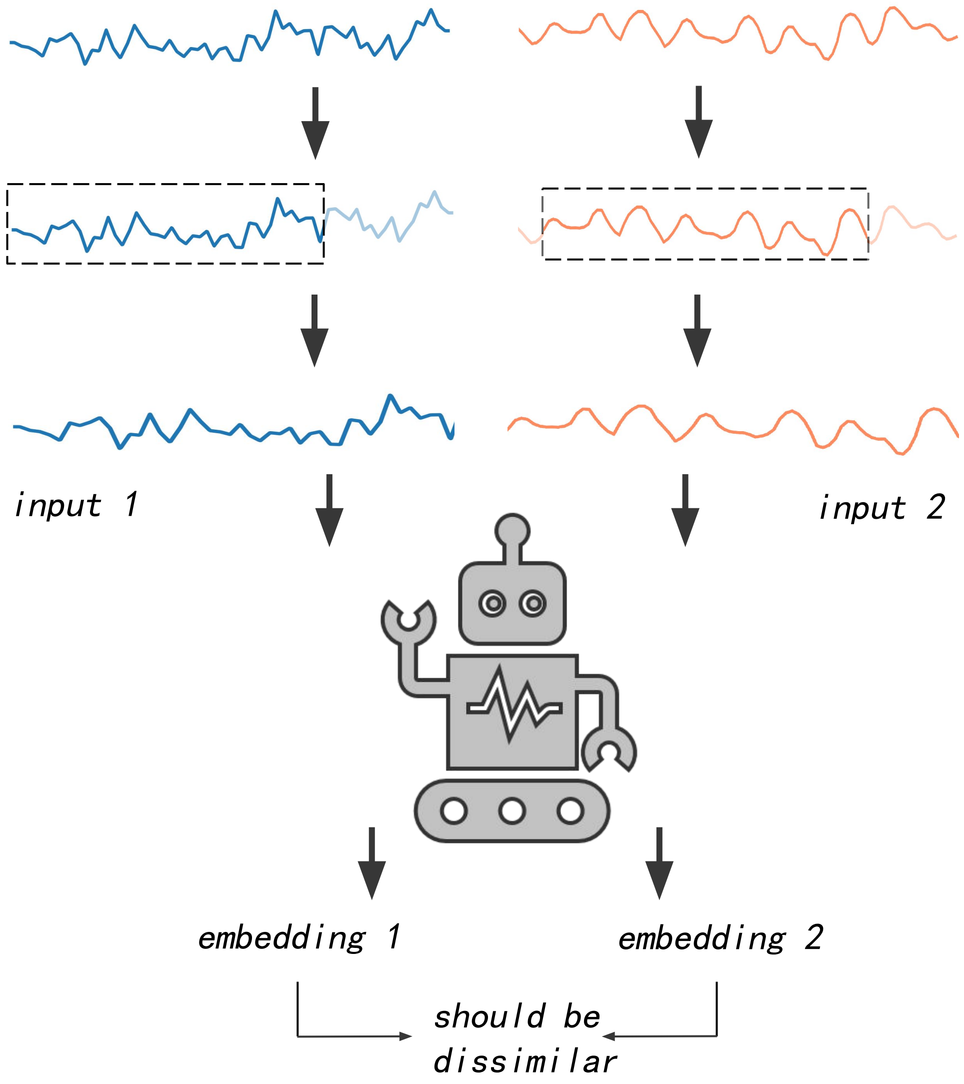
2. Unsupervised (self-supervised) pre-training: use only features
   a. Masked reconstruction (auto-encoder).
   b. Next-toke prediction (decoder-only).
   c. Contrastive learning and self-distillation (encoder-only).

Pre-training by Contrastive Learning



- Make positive pairs close to each other, negative pairs – far away from each other.
- Positive pair: 2 augmentations of the same time series.
- Augmentation: random-crop-resize.

Pre-training by Contrastive Learning



- Make positive pairs close to each other, negative pairs – far away from each other.
- Positive pair: 2 augmentations of the same time series.
- Augmentation: random-crop-resize.
- Negative pair: different examples

Pre-training by Contrastive Learning

- Notations:

Batch	$B = \{\mathbf{x}_i\}_{i=1}^{b}$
Augment. funcs	ϕ, ψ
Foundation model	$F : \mathbb{R}^t \to \mathbb{R}^q$
Projector	$g : \mathbb{R}^q \to \mathbb{R}^{q'}$

Pre-training by Contrastive Learning

- Notations:
- Cosine Distance:

$$s_{\cos}(\mathbf{a}, \mathbf{b}) := \frac{\mathbf{a}^\top \mathbf{b}}{\|\mathbf{a}\| \cdot \|\mathbf{b}\|}, \qquad \forall (\mathbf{a}, \mathbf{b}) \in \mathbb{R}^{2q'}$$

Batch	$B = \{\mathbf{x}_i\}_{i=1}^{b}$
Augment. funcs	ϕ, ψ
Foundation model	$F : \mathbb{R}^t \to \mathbb{R}^q$
Projector	$g : \mathbb{R}^q \to \mathbb{R}^{q'}$

Pre-training by Contrastive Learning

- Notations:

Batch	$B = \{\mathbf{x}_i\}_{i=1}^b$
Augment. funcs	ϕ, ψ
Foundation model	$F : \mathbb{R}^t \to \mathbb{R}^q$
Projector	$g : \mathbb{R}^q \to \mathbb{R}^{q'}$

- Cosine Distance:

$$s_{\cos}(\mathbf{a}, \mathbf{b}) := \frac{\mathbf{a}^\top \mathbf{b}}{\|\mathbf{a}\| \cdot \|\mathbf{b}\|}, \qquad \forall (\mathbf{a}, \mathbf{b}) \in \mathbb{R}^{2q'}$$

- Logits: cosine distances to all examples from the batch.

$$\mathbf{s}_i(\phi, \psi) = \left[s_{\cos}\left(g \circ F \circ \phi(\mathbf{x}_i),\ g \circ F \circ \psi(\mathbf{x}_j)\right)\right]_{j=1}^b \in \mathbb{R}^b$$

Pre-training by Contrastive Learning

Batch	$B = \{\mathbf{x}_i\}_{i=1}^{b}$
Augment. funcs	ϕ, ψ
Foundation model	$F : \mathbb{R}^t \to \mathbb{R}^q$
Projector	$g : \mathbb{R}^q \to \mathbb{R}^{q'}$

- Notations:
- Cosine Distance:

$$s_{\cos}(\mathbf{a}, \mathbf{b}) := \frac{\mathbf{a}^\top \mathbf{b}}{\|\mathbf{a}\| \cdot \|\mathbf{b}\|}, \qquad \forall (\mathbf{a}, \mathbf{b}) \in \mathbb{R}^{2q'}$$

- Logits: cosine distances to all examples from the batch.

$$\mathbf{s}_i(\phi, \psi) = \left[s_{\cos}\left(g \circ F \circ \phi(\mathbf{x}_i),\ g \circ F \circ \psi(\mathbf{x}_j)\right)\right]_{j=1}^{b} \in \mathbb{R}^b$$

- The true class for example $\mathbf{x}_i$ is *i* (i.e., the closest example to $\mathbf{x}_i$ is itself).
- Minimize cross-entropy loss:

$$\sum_{i=1}^{b} l_{\text{ce}}\left(\frac{\mathbf{s}_i(\phi, \psi)}{T},\ i\right)$$

Pre-training Data

Pre-training Data: Criteria

Good pre-training sata should be

1. Relevant, reflecting the downstream applications.
   - Just periodic data without spikes might be not enough for classification.
2. Diverse to be robust to out-of-distribution data.
   - Extracting many windows with stride=1 generate highly correlated data => useless.
3. Of reasonable quality.
   - If some time series have very low signal-to-noise ratios, they may disturb pre-training.

Mantis V1 Dataset

We concatenated 3 collections of publicly available datasets:

1. UCR – 128 univariate datasets,
2. UEA – 28/30 multivariate datasets,
3. + 8 datasets from (Zhang et al., 2022),

which results in **1.89 million** samples.

Mantis V1 Dataset

We concatenated 3 collections of publicly available datasets:

1. UCR – 128 univariate datasets,
2. UEA – 28/30 multivariate datasets,
3. + 8 datasets from (Zhang et al., 2022),

which results in **1.89 million** samples.

There were some issues:

- ❑ Overlap with test benchmarks.
- ❑ Imbalanced: some datasets dominate.

Mantis V1 Dataset

We concatenated 3 collections of publicly available datasets:

1. UCR – 128 univariate datasets,
2. UEA – 28/30 multivariate datasets,
3. + 8 datasets from (Zhang et al., 2022),

which results in **1.89 million** samples.

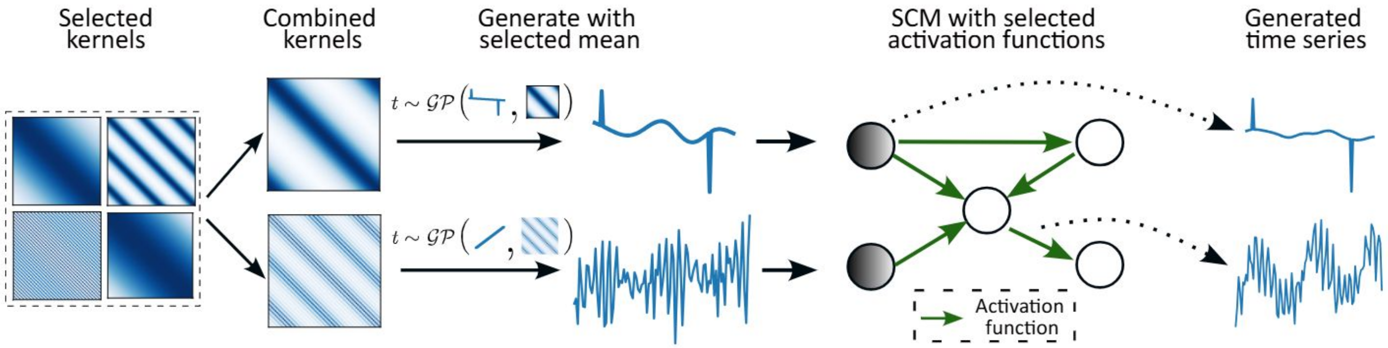
There were some issues:

- ❏ Overlap with test benchmarks.
- ❏ Imbalanced: some datasets dominate.

Idea: generate synthetic data!

Synthetic Pre-training Data

<u>CauKer:</u> synthetic data generation algorithm.



(1) *Priors*: multiple Gaussian processes with different mean functions and covariance kernels.

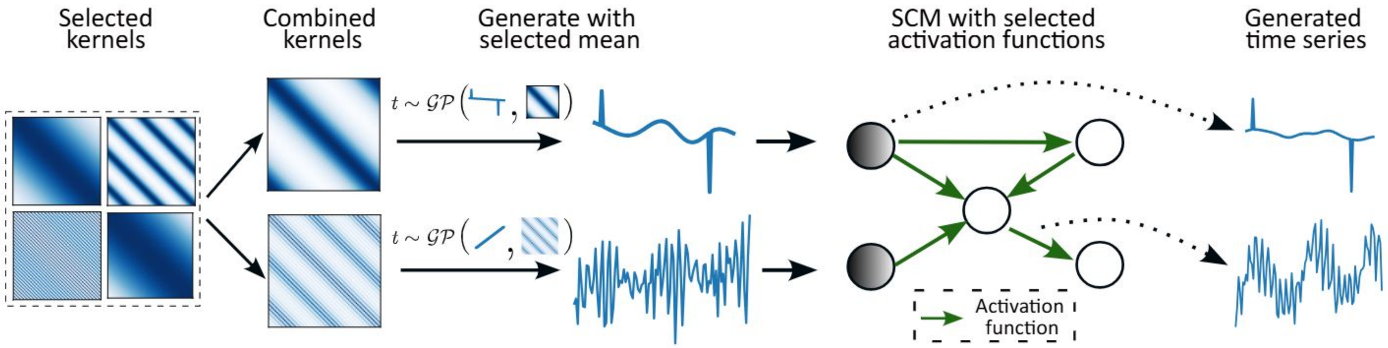
(2) *Causal graph*: priors are root nodes of a structural causal graph with a non-linear activation at each node.

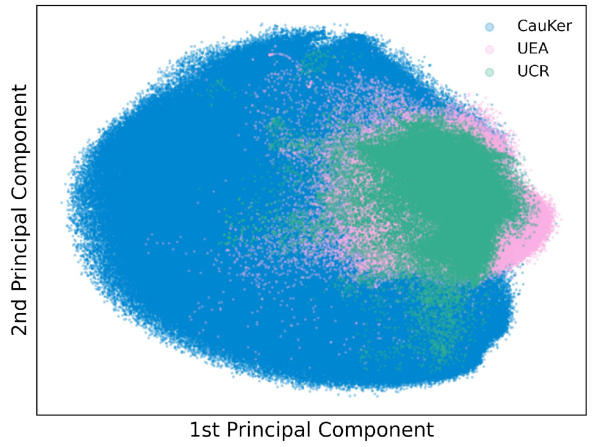
S. Xie et al. (2026). CauKer: classification time series foundation models can be pretrained on synthetic data only. (*ICLR'26 oral)*.

Synthetic Pre-training Data

<u>CauKer:</u> synthetic data generation algorithm that yields

- Large data diversity.
- Sample efficiency.
- Similar performance to real pre-training data.





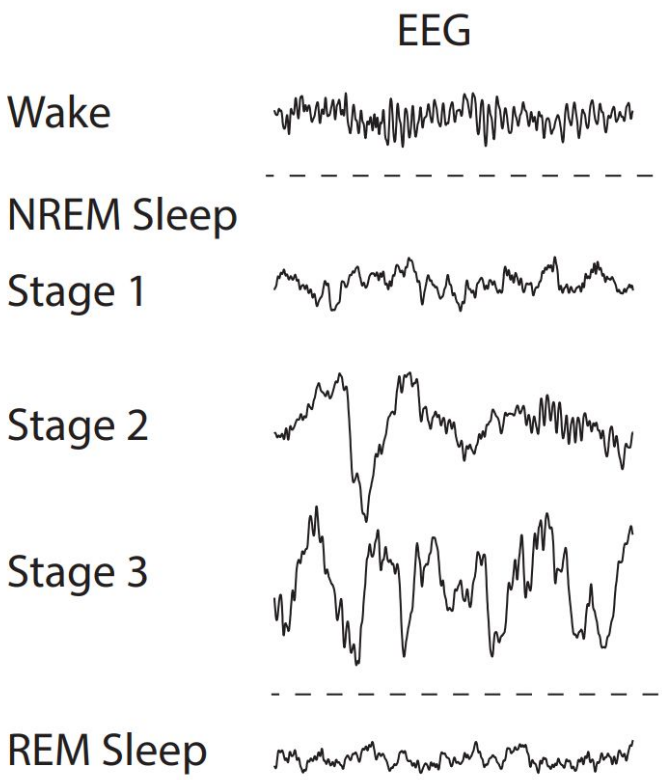
Pre-training Data	Nature	Size	UCR Included?	UCR acc. (%)
Anomaly	Real	38K	No	$0.7473_{\pm 0.0014}$
Subset of V1 Dataset	Real	100K	No	$0.7829_{\pm 0.0008}$
CauKer	Synth	100K	No	$78.81_{\pm 0.001}$
V1 Dataset	Real	1.89M	Yes	$79.21_{\pm 0.0012}$

S. Xie et al. (2026). CauKer: classification time series foundation models can be pretrained on synthetic data only. (*ICLR'26)*.

First Experimental Results

Application to EEG Sleep Staging

Dataset	EEGNet	CBraMod		Mantis		
		Random	EEG Pretrain	Random	Real Pretrain	Synth Pretrain
ABC	$67.94_{\pm 6.52}$	$70.61_{\pm 3.29}$	$74.90_{\pm 4.89}$	$72.82_{\pm 3.89}$	$75.50_{\pm 5.62}$	$\mathbf{75.74_{\pm 4.32}}$
CCSHS	$83.13_{\pm 0.10}$	$87.01_{\pm 0.27}$	$88.04_{\pm 0.59}$	$88.55_{\pm 0.39}$	$\mathbf{88.85_{\pm 0.48}}$	$88.80_{\pm 0.30}$
CFS	$78.60_{\pm 1.31}$	$83.48_{\pm 0.23}$	$84.30_{\pm 0.08}$	$84.96_{\pm 0.43}$	$\mathbf{85.35_{\pm 0.35}}$	$85.06_{\pm 0.75}$
CHAT	$78.91_{\pm 0.16}$	$84.11_{\pm 0.81}$	$85.01_{\pm 0.42}$	NaN	$\mathbf{85.94_{\pm 0.18}}$	$85.72_{\pm 0.29}$
HOMEPAP	$69.43_{\pm 0.08}$	$70.37_{\pm 1.90}$	$72.56_{\pm 2.35}$	$71.26_{\pm 1.93}$	$73.14_{\pm 2.09}$	$\mathbf{73.53_{\pm 2.00}}$
MASS	$79.85_{\pm 1.27}$	$77.40_{\pm 2.18}$	$81.12_{\pm 2.27}$	$79.06_{\pm 1.89}$	$\mathbf{84.09_{\pm 0.85}}$	$82.49_{\pm 1.22}$
PhysioNet	$75.73_{\pm 0.38}$	$77.19_{\pm 0.94}$	$78.97_{\pm 0.43}$	$77.98_{\pm 0.89}$	$\mathbf{79.82_{\pm 1.63}}$	$78.83_{\pm 1.60}$
SOF	$78.74_{\pm 1.81}$	$82.61_{\pm 0.35}$	$83.39_{\pm 0.67}$	$83.70_{\pm 1.01}$	$\mathbf{84.69_{\pm 0.73}}$	$84.31_{\pm 0.57}$



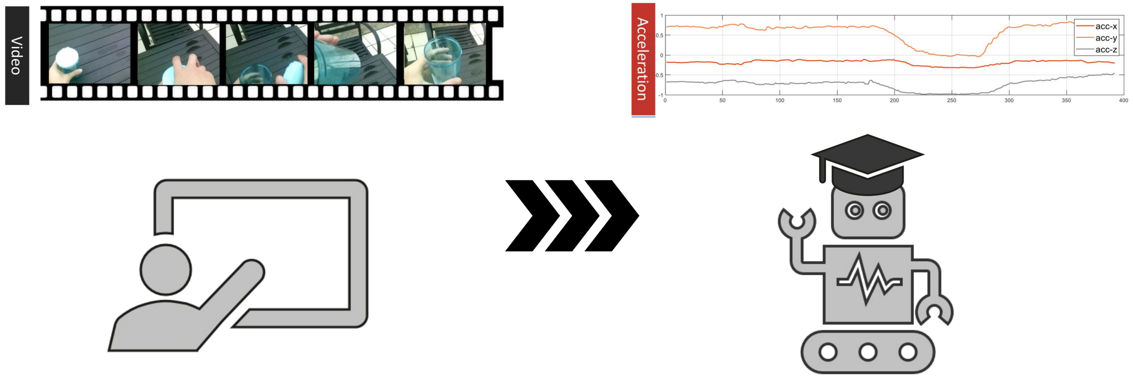
Mantis outperforms EEG foundation model

on sleep stage prediction task!

T. Gnassounou et al. (2025). Leveraging Generic Time Series Foundation Models for EEG Classification. *arXiv:2510.27522*.

Application to HAR

- Mantis is a good student from a video teacher.
- Zero-shot after distillation matches the fine-tuning performance.



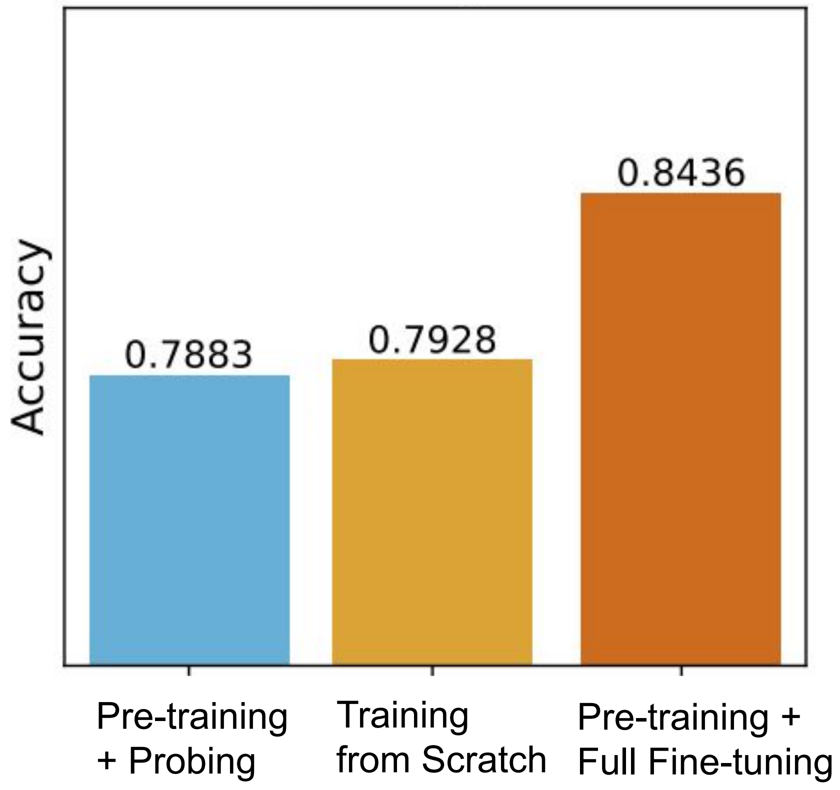
Video Teacher   Time Series Student

	Model	Ego4d			EgoExo4D			MMEA		
		acc@1	acc@3	acc@3	acc@1	acc@3	acc@3	acc@1	acc@3	acc@3
Zero-Shot	Moment-Small	39.70	64.47	75.32	68.14	93.55	98.43	83.66	95.52	97.42
	Mantis	47.49	71.63	81.24	76.47	96.98	99.21	90.96	98.56	99.39
	TSFormer→ Mantis	**59.13**	**78.79**	**85.65**	**84.92**	**98.28**	**99.59**	92.48	**99.01**	**99.77**
Fine-Tuned	Moment-Small	57.59	75.91	82.94	79.26	97.04	99.33	84.27	94.76	96.88
	Mantis	58.36	76.98	83.76	84.22	97.95	99.41	**93.01**	98.25	99.01

B. Chen et al. (2025). Comodo: Cross-modal video-to-imu distillation for efficient egocentric human activity recognition. *arXiv:2503.07259*.

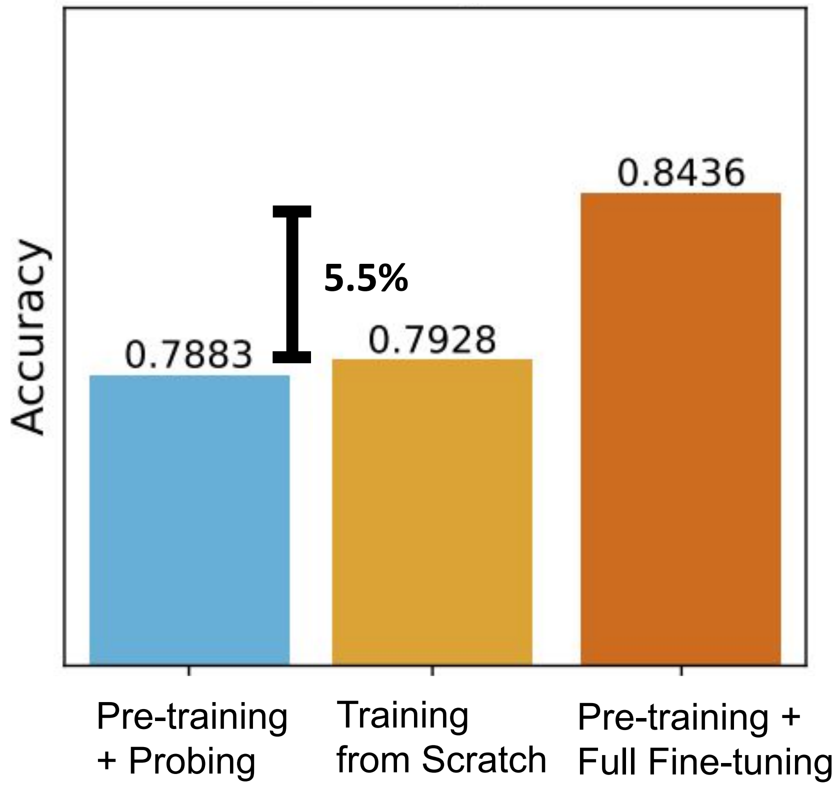
Influence of Pre-training and Fine-tuning

Best performance is achieved with pre-training + fine-tuning.



Influence of Pre-training and Fine-tuning

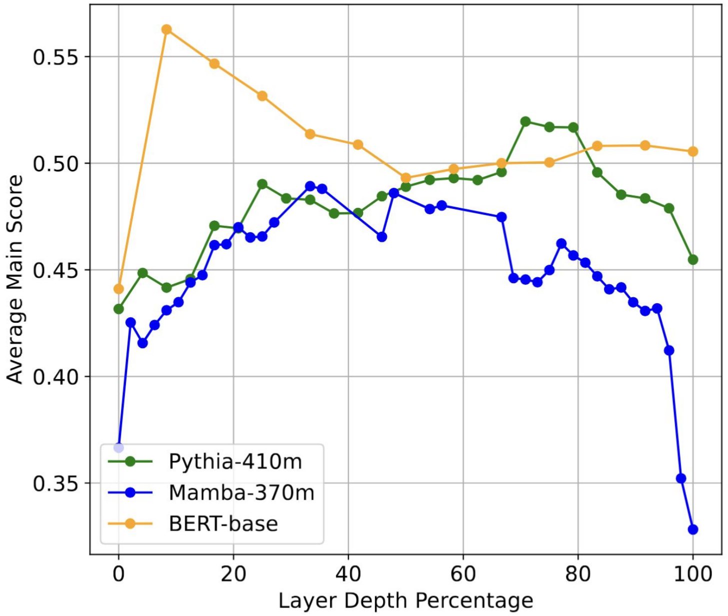
- **Issue:** A big gap in performance between frozen and fine-tuned Mantis.
- **Question:** Are we able to mitigate this gap?



Leveraging Intermediate Layers

Strong Intermediate Representations

- Skean et al. (2025) showed that intermediate representations of LLMs may outperform the final one.

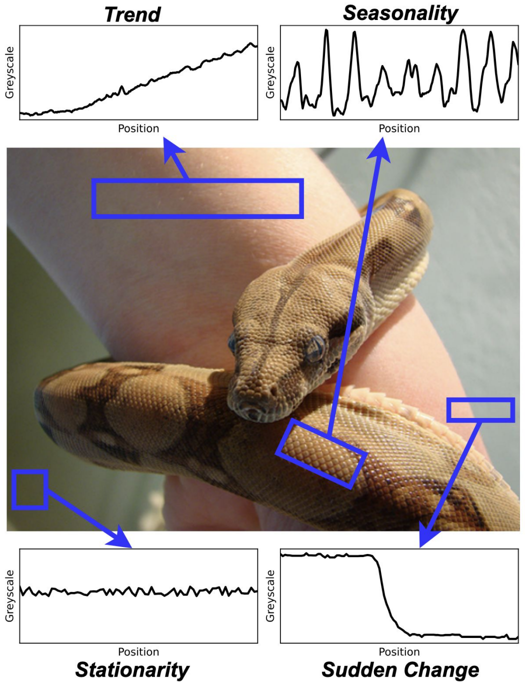


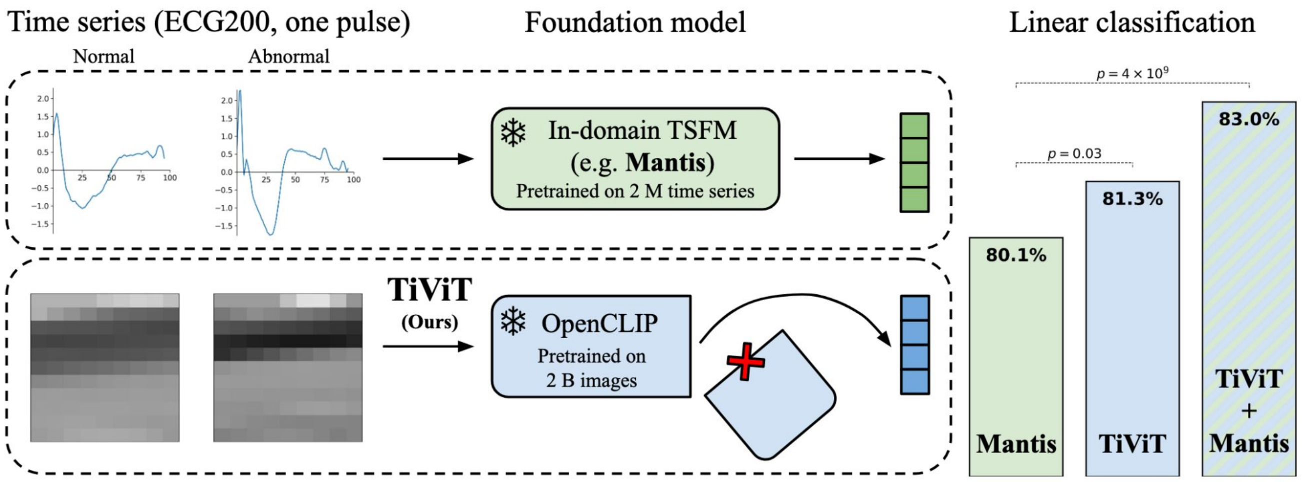
Performance of 32 embedding tasks from the Massive Text Embedding Benchmark (MTEB).

O. Skean et al. (2025). Layer by Layer: Uncovering Hidden Representations in Language Models.

Strong Intermediate Representations

- Skean et al. (2025) showed that intermediate representations of LLMs may outperform the final one.
- Roschmann et al. (2025) showed a cross-domain transfer: intermediate layers of pre-trained ViT are strong embeddings for time series classification!

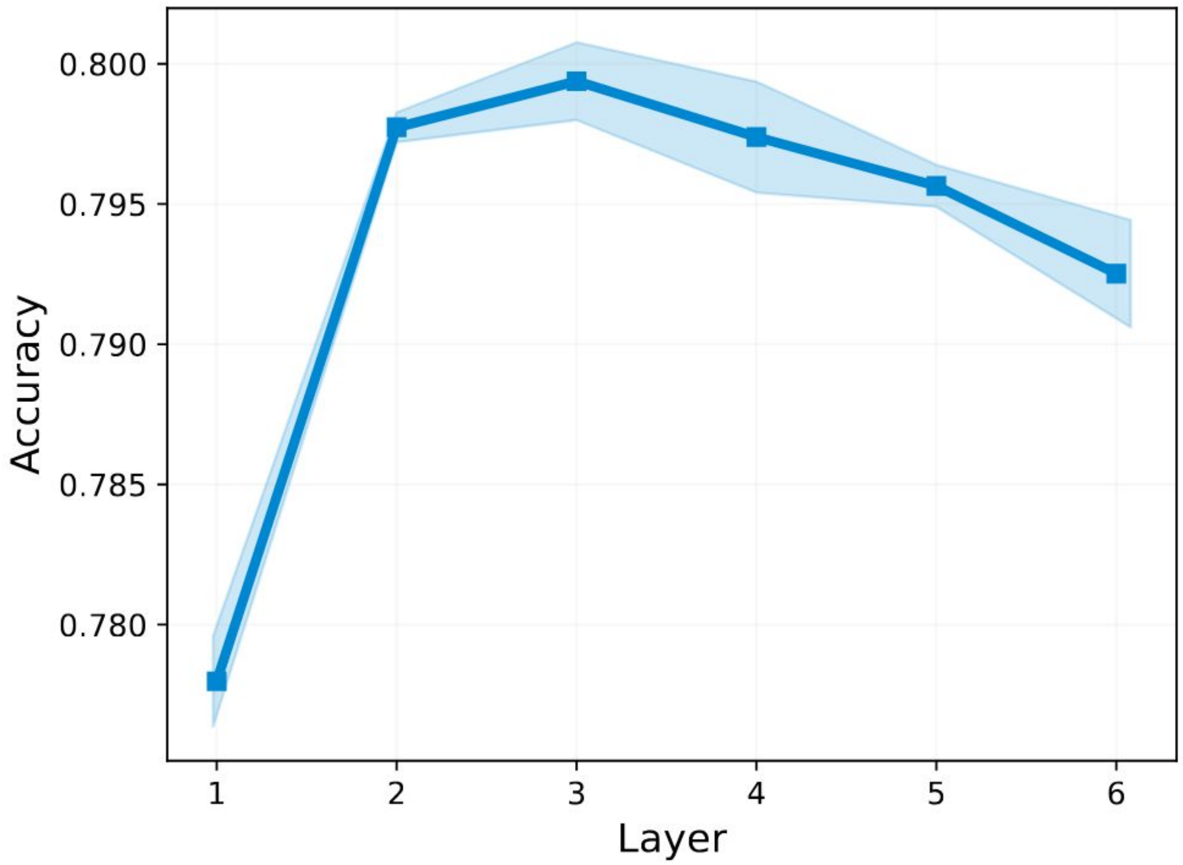




O. Skean et al. (2025). Layer by Layer: Uncovering Hidden Representations in Language Models.
M. Chen et al. (2025). VisionTS: Visual Masked Autoencoders Are Free-Lunch Zero-Shot Time Series Forecasters.
Roschmann et al. (2025). Time Series Representations for Classification Lie Hidden in Pretrained Vision Transformers.

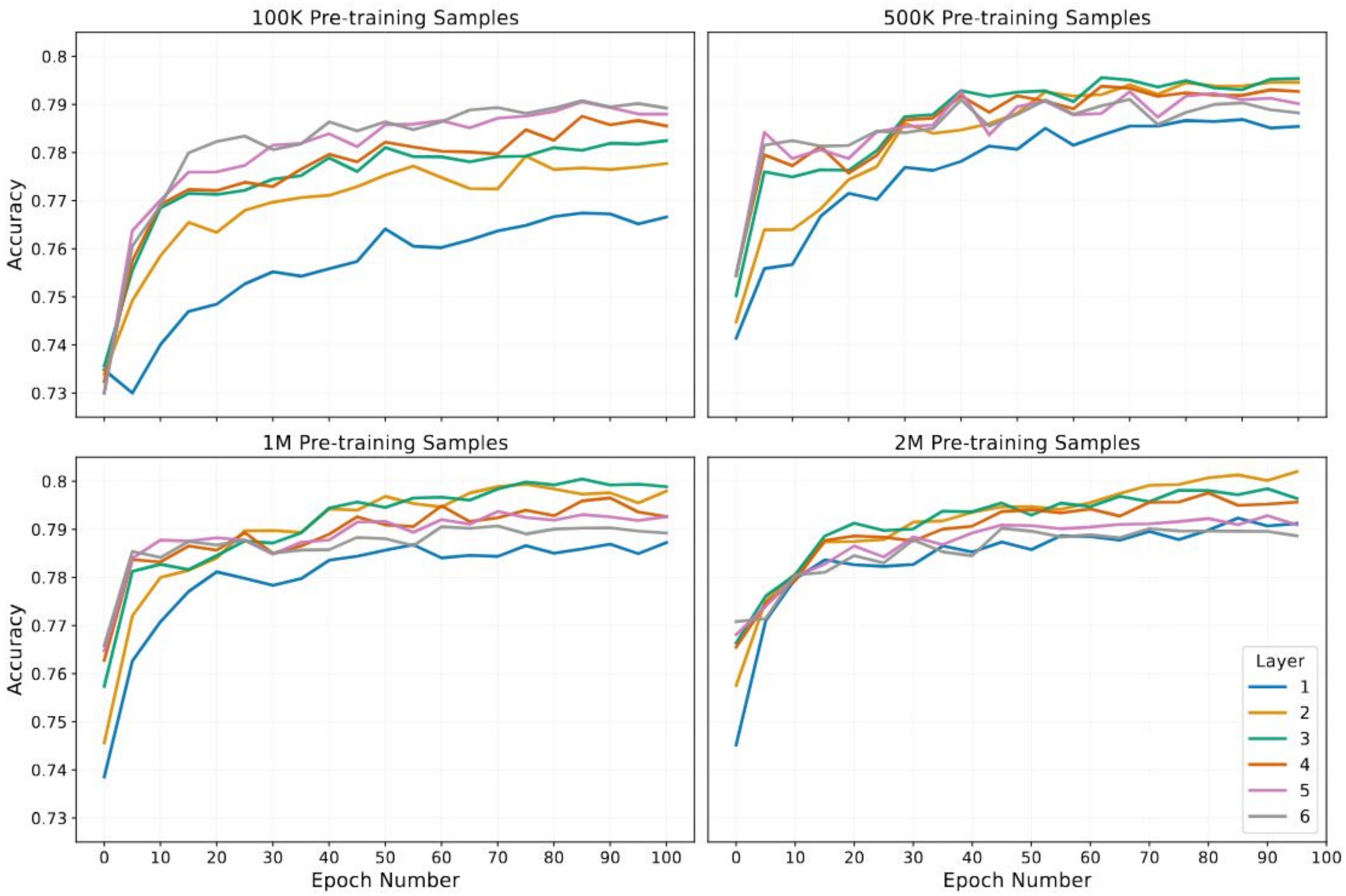
Layer by Layer for Mantis

<u>Observation 1:</u> Intermediate layers of transformers generalize better!



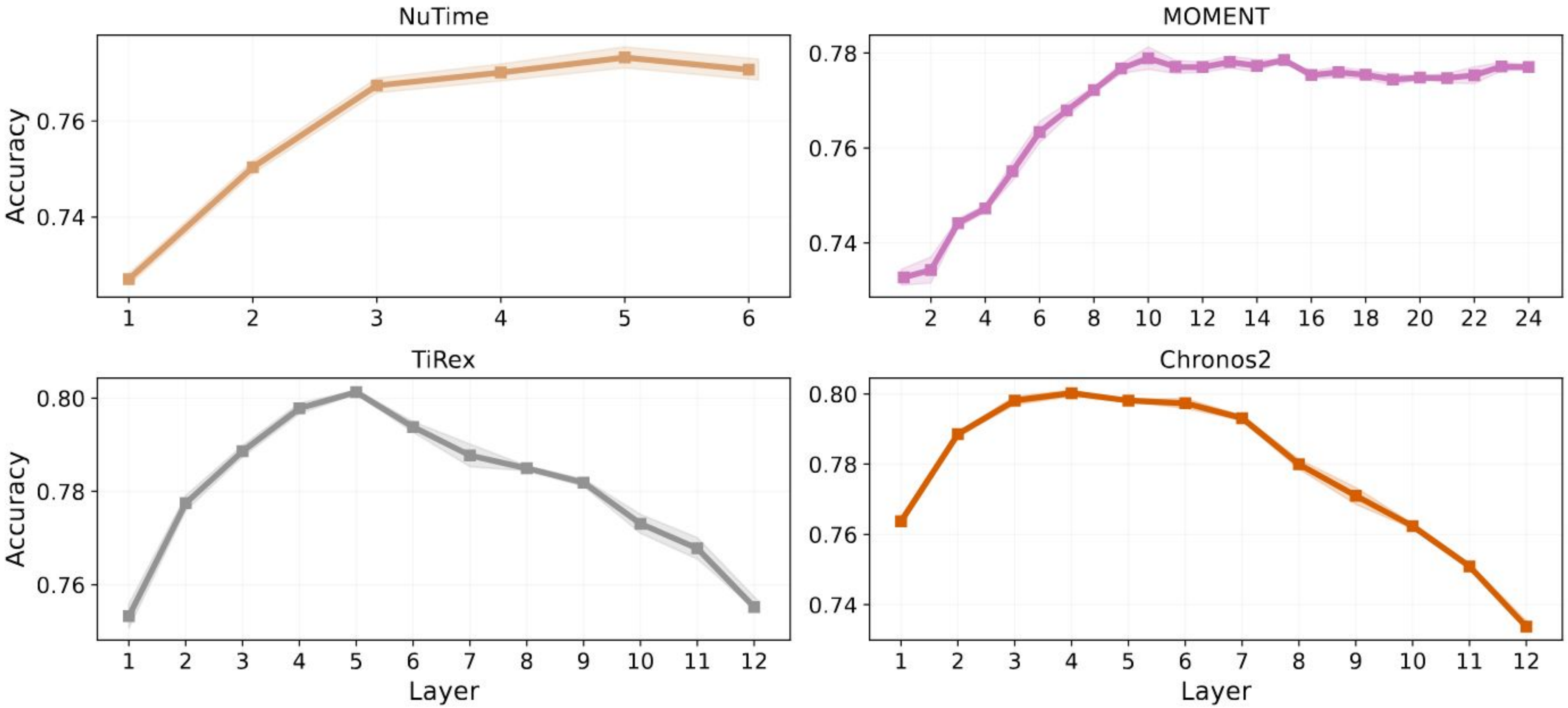
Layer by Layer, Epoch by Epoch

<u>Observation 2:</u> Intermediate layers unlock the data scaling law!



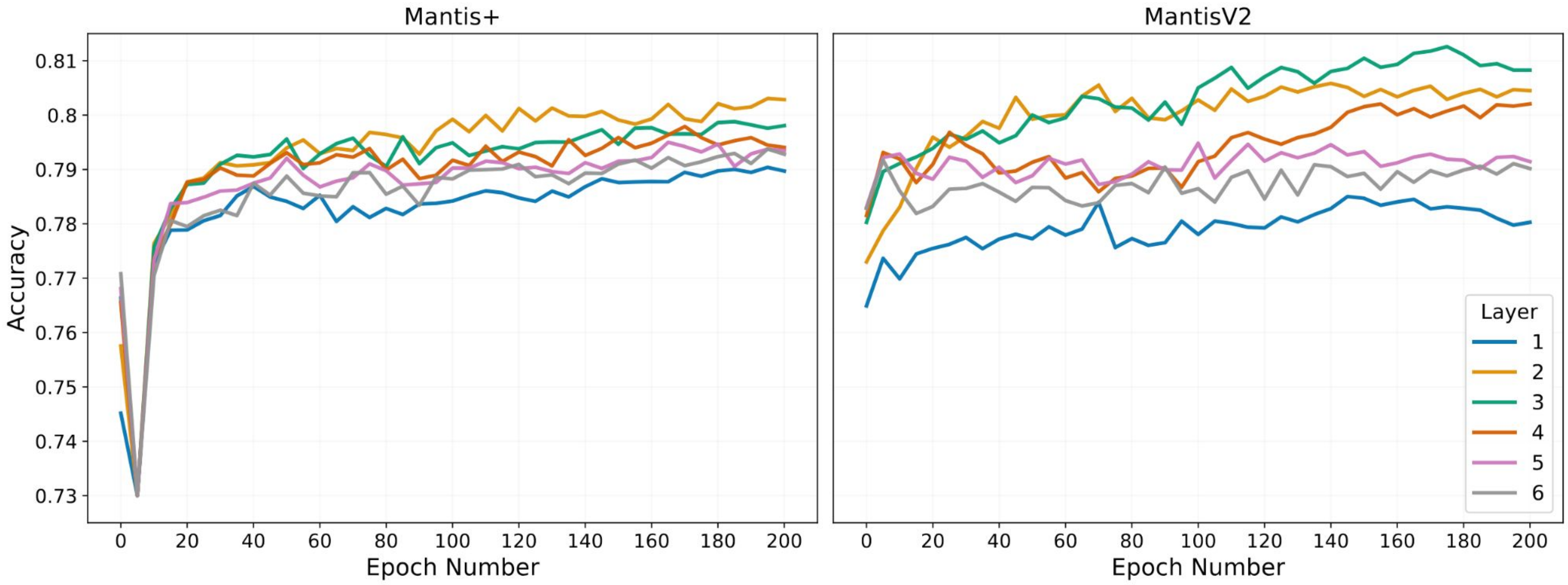
Layer by Layer for Other Methods

Observation 3: Intermediate layers make time series forecasting models relevant for time series classification.



Layer by Layer Epoch by Epoch

Final pre-training that gives Mantis+ and MantisV2.



Layer by Layer Module by Module

For image classification, Odonnat et al. (2026) showed that even better representations are found by looking after each module of a transformer layer!

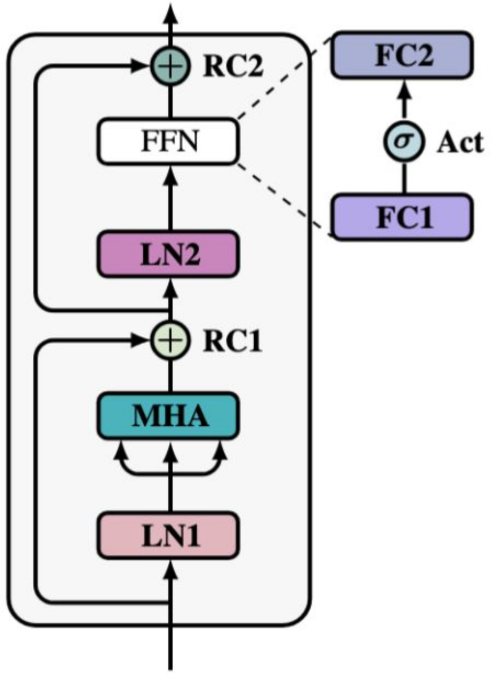


Figure 2: **Transformer block.**

Table 1: **Module by module.** For each module, we report the best linear probing accuracy over the layers. The best performance per dataset is in **bold** and the module with the highest win rate is in gray.

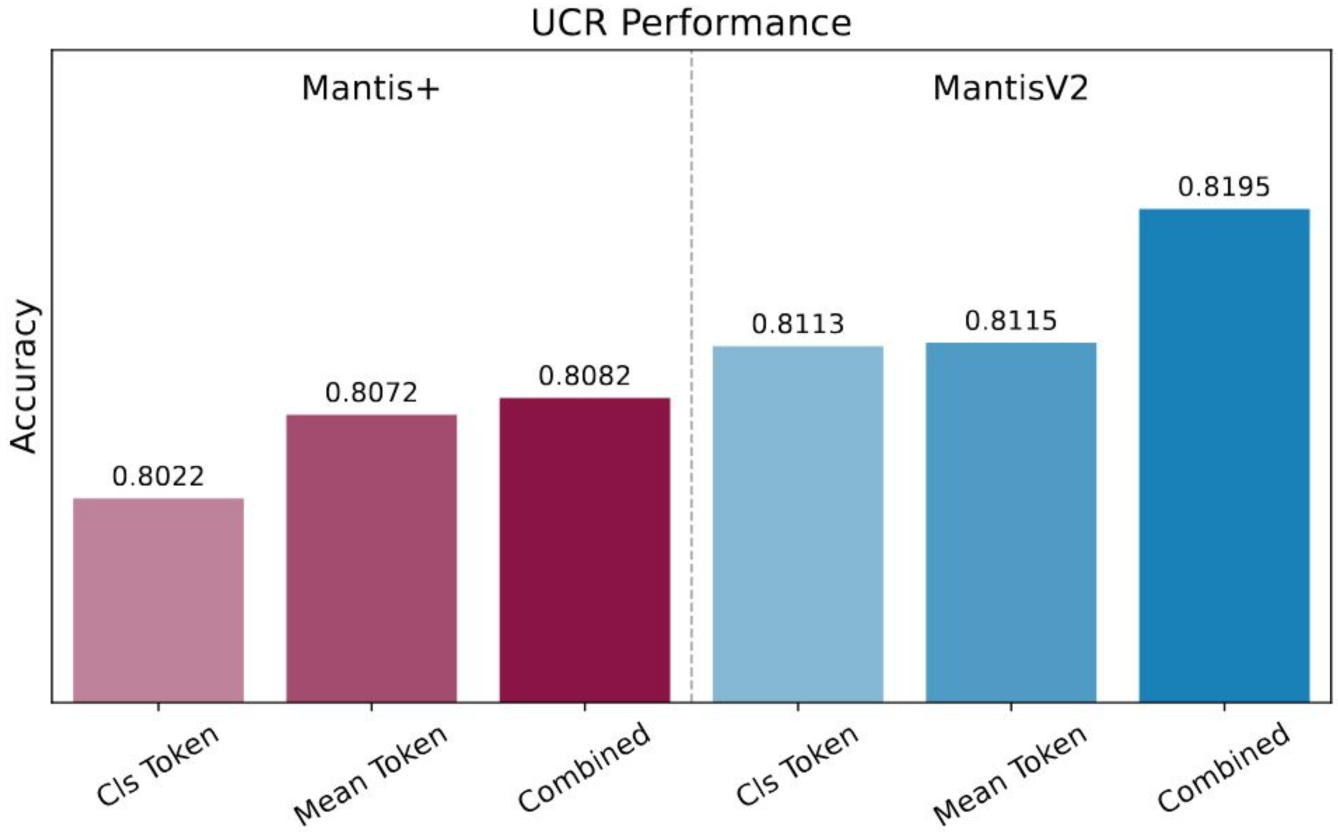
Dataset	LN1	MHA	RC1	LN2	FC1	Act	FC2	RC2
Cifar10	91.94	91.98	92.19	92.20	**92.28**	85.30	89.98	92.07
Cifar100	69.39	67.27	69.88	**69.97**	68.98	69.75	60.92	69.63
Contrast	78.05	74.55	78.30	79.05	78.15	**80.20**	70.85	77.95
Gaussian Noise	57.65	60.40	59.05	58.10	58.75	**61.85**	56.70	58.20
Motion Blur	66.85	64.30	68.50	67.75	66.80	**71.15**	60.90	67.75
Snow	67.30	66.15	68.10	67.30	67.35	**69.30**	61.50	67.70
Speckle Noise	59.75	61.85	60.25	59.95	59.90	**63.35**	58.00	59.80
Clipart	47.66	43.82	48.66	48.37	45.74	**49.34**	40.97	48.33
Sketch	32.36	30.95	33.32	33.07	31.11	**34.90**	28.45	32.99
Flowers102	96.58	96.34	96.58	**96.62**	96.44	91.64	95.23	**96.62**
Pet	88.36	88.33	89.48	**89.51**	88.47	83.46	85.80	89.18

A. Odonnat et al. (2026). Layer by layer, module by module: Choose both for optimal OOD probing of ViT (*ICLR'26 CAO workshop*)

Output-Token Aggregation

Usually, a transformer encoder outputs classification token's embedding as the final one.

- Not optimal for intermediate layers.
- Concatenation of cls token and the mean of all other tokens improves performance.



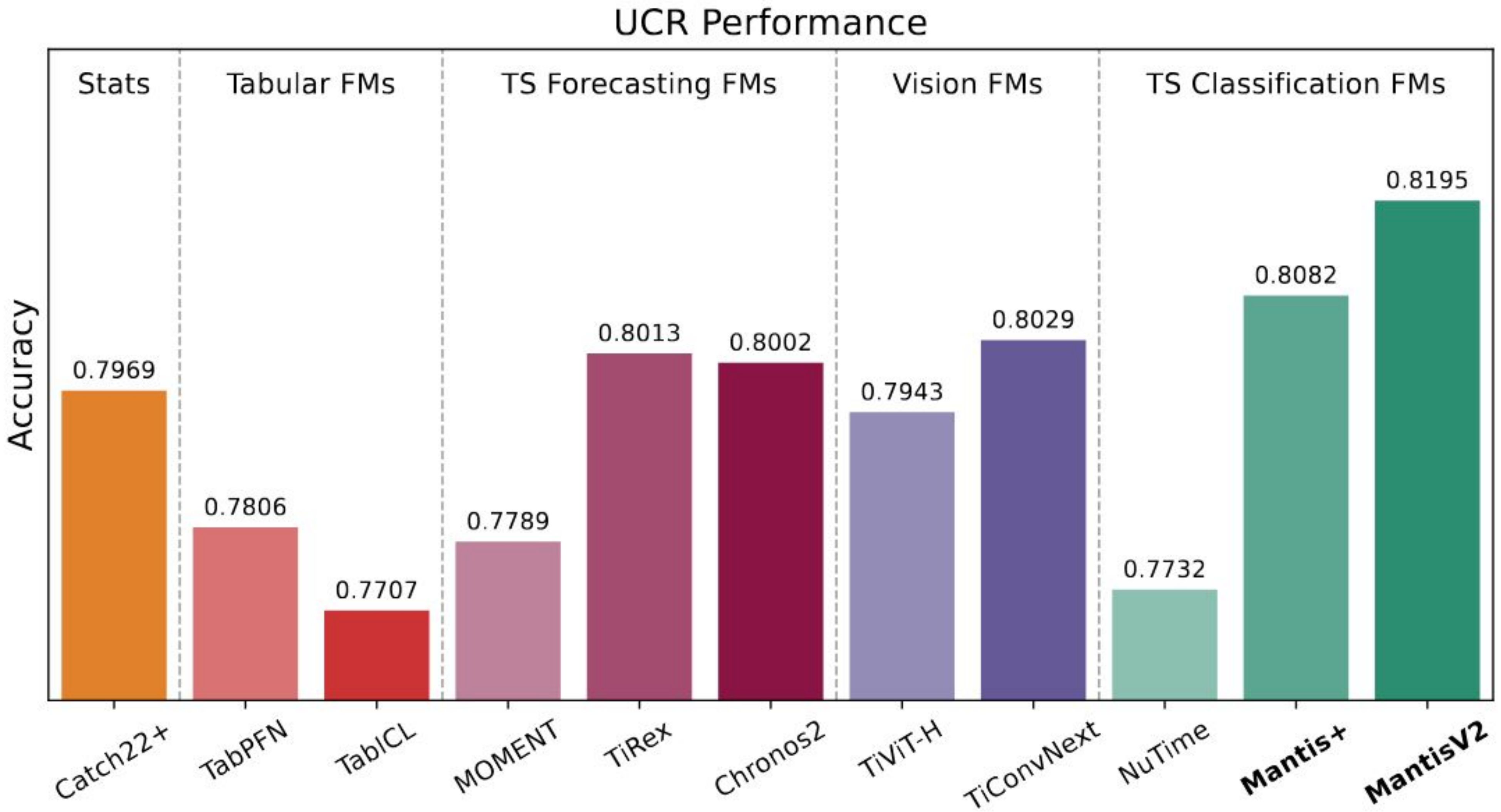
Main Experimental Results.

Baselines

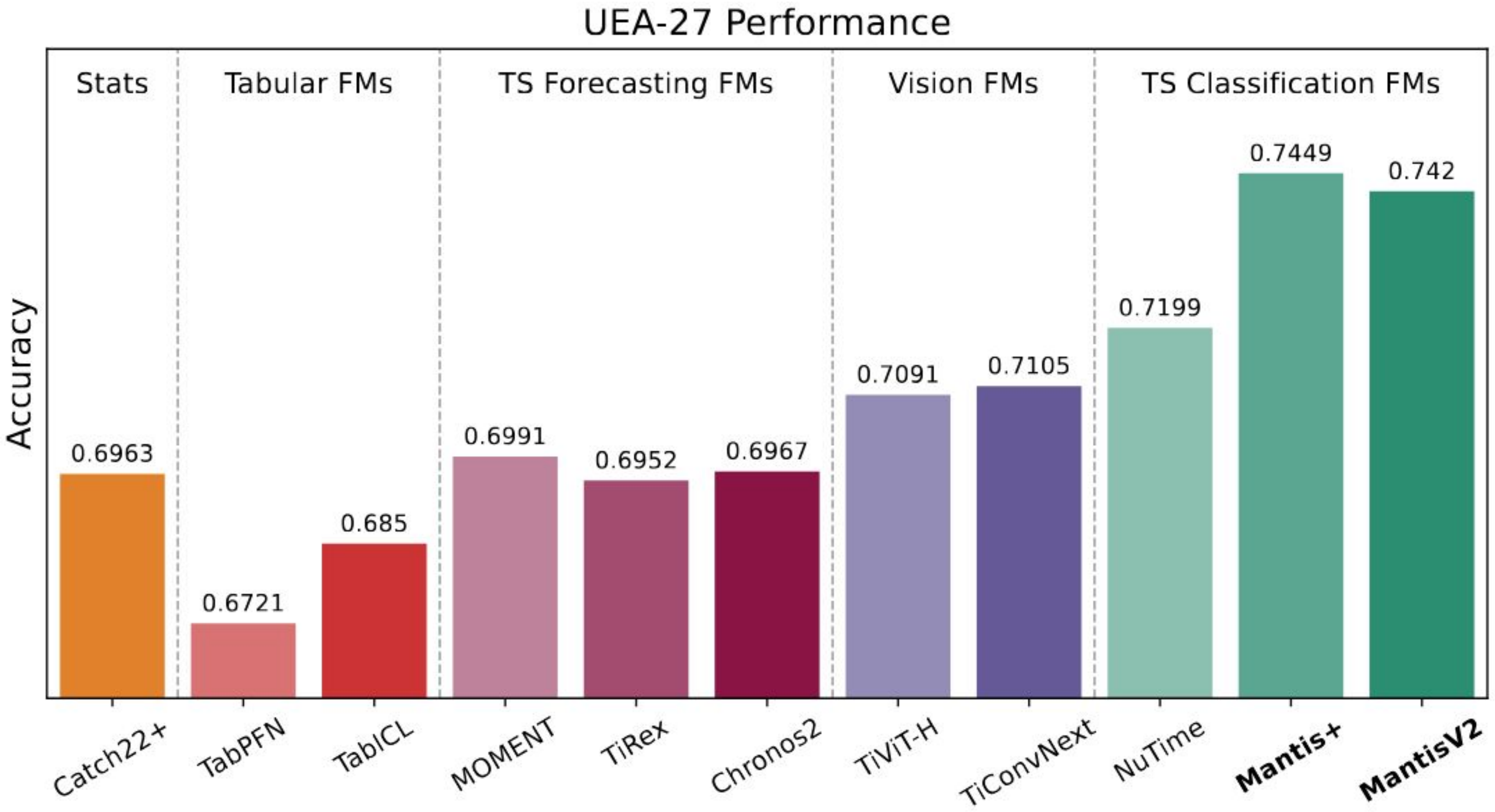
# of Params.	TabPFN	TabICL	MOMENT	TiRex	Chronos2	TiViT-H	TiConvNext	NuTime	Mantis+	MantisV2
Original	7.2M	27.1M	341.2M	35.3M	119.5M	630.8M	843.4M	2.4M	8.1M	4.2M
After Pruning	-	-	161.4M	16.5M	44M	276.6M	180.3M	2M	2.9M	2.2M

- **Catch22+**: Manually curated statistical features.
- 2 tabular FMs: **TabPFN** and **TabICL**.
- **MOMENT**: T5-based masked auto-encoder.
- 2 forecasting foundation models: **TiRex** and **Chronos2**.
- Adaptation of vision-based models to time series: **TiViT-H** and **TiConvNext**.
- **NuTime**: classification TSFM based on BYOL self-distillation.
- Our models: **Mantis+** and **MantisV2**.

UCR Performance (128 Univ. Datasets)



UEA Performance (27 Multivar. Datasets)



Performance on Human Activity Recognition Datasets

	Catch22+	TabPFN	TabICL	MOMENT	TiRex	Chronos2	TiViT-H	TiConvNext	NuTime	Mantis+	MantisV2
Ego4D	$0.4397_{\pm 0.002}$	NaN	NaN	$0.4068_{\pm 0.001}$	$0.5037_{\pm 0.001}$	$0.4742_{\pm 0.0}$	0.1912	0.1907	$0.5108_{\pm 0.0}$	$\mathbf{0.5273}_{\pm 0.0}$	$0.5258_{\pm 0.001}$
EMOPain	$0.8826_{\pm 0.006}$	0.7831	0.7831	$0.8469_{\pm 0.002}$	$0.7915_{\pm 0.003}$	$0.8075_{\pm 0.004}$	$0.83_{\pm 0.002}$	$0.8225_{\pm 0.003}$	$0.8901_{\pm 0.005}$	$\mathbf{0.8939}_{\pm 0.002}$	$0.8798_{\pm 0.009}$
HHAR-ID	$0.9738_{\pm 0.001}$	0.8938	0.9073	$0.9299_{\pm 0.0}$	$0.9481_{\pm 0.0}$	$0.9475_{\pm 0.002}$	$0.9338_{\pm 0.001}$	$0.9461_{\pm 0.001}$	$0.9808_{\pm 0.001}$	$0.9835_{\pm 0.001}$	$\mathbf{0.9845}_{\pm 0.001}$
HHAR-OOD	$0.4808_{\pm 0.008}$	0.5311	0.5441	$0.3081_{\pm 0.001}$	$0.5135_{\pm 0.014}$	$0.4174_{\pm 0.008}$	$0.3748_{\pm 0.007}$	$0.4031_{\pm 0.01}$	$0.56_{\pm 0.005}$	$0.5624_{\pm 0.016}$	$\mathbf{0.5822}_{\pm 0.005}$
MP8	$0.572_{\pm 0.008}$	0.6235	0.6185	$0.6392_{\pm 0.011}$	$0.5994_{\pm 0.016}$	$0.5966_{\pm 0.006}$	$0.6022_{\pm 0.009}$	$0.5966_{\pm 0.003}$	$0.6504_{\pm 0.008}$	$0.6403_{\pm 0.006}$	$\mathbf{0.6857}_{\pm 0.012}$
MP50	$0.3602_{\pm 0.016}$	0.5664	0.5042	$\mathbf{0.7434}_{\pm 0.017}$	$0.6644_{\pm 0.01}$	$0.6639_{\pm 0.007}$	$0.6908_{\pm 0.008}$	$0.6801_{\pm 0.008}$	$0.6913_{\pm 0.012}$	$0.7227_{\pm 0.008}$	$0.7345_{\pm 0.007}$
UCI-HAR	$0.8273_{\pm 0.003}$	0.809	0.8157	$0.8782_{\pm 0.001}$	$0.8744_{\pm 0.002}$	$0.8873_{\pm 0.002}$	$0.8936_{\pm 0.001}$	$0.8918_{\pm 0.001}$	$0.8842_{\pm 0.001}$	$0.8911_{\pm 0.0}$	$\mathbf{0.9013}_{\pm 0.002}$
Avg	0.6481	NaN	NaN	0.6789	0.6993	0.6849	0.6452	0.6473	0.7382	0.7459	**0.7562**
Avg UCR HAR	0.7564	0.7479	0.7285	0.7572	0.7687	0.7702	0.7726	0.772	0.756	0.7867	**0.8007**
Avg UEA HAR	0.8138	0.7591	0.764	0.8431	0.846	0.8492	0.8535	0.8488	0.8539	**0.8796**	0.8739

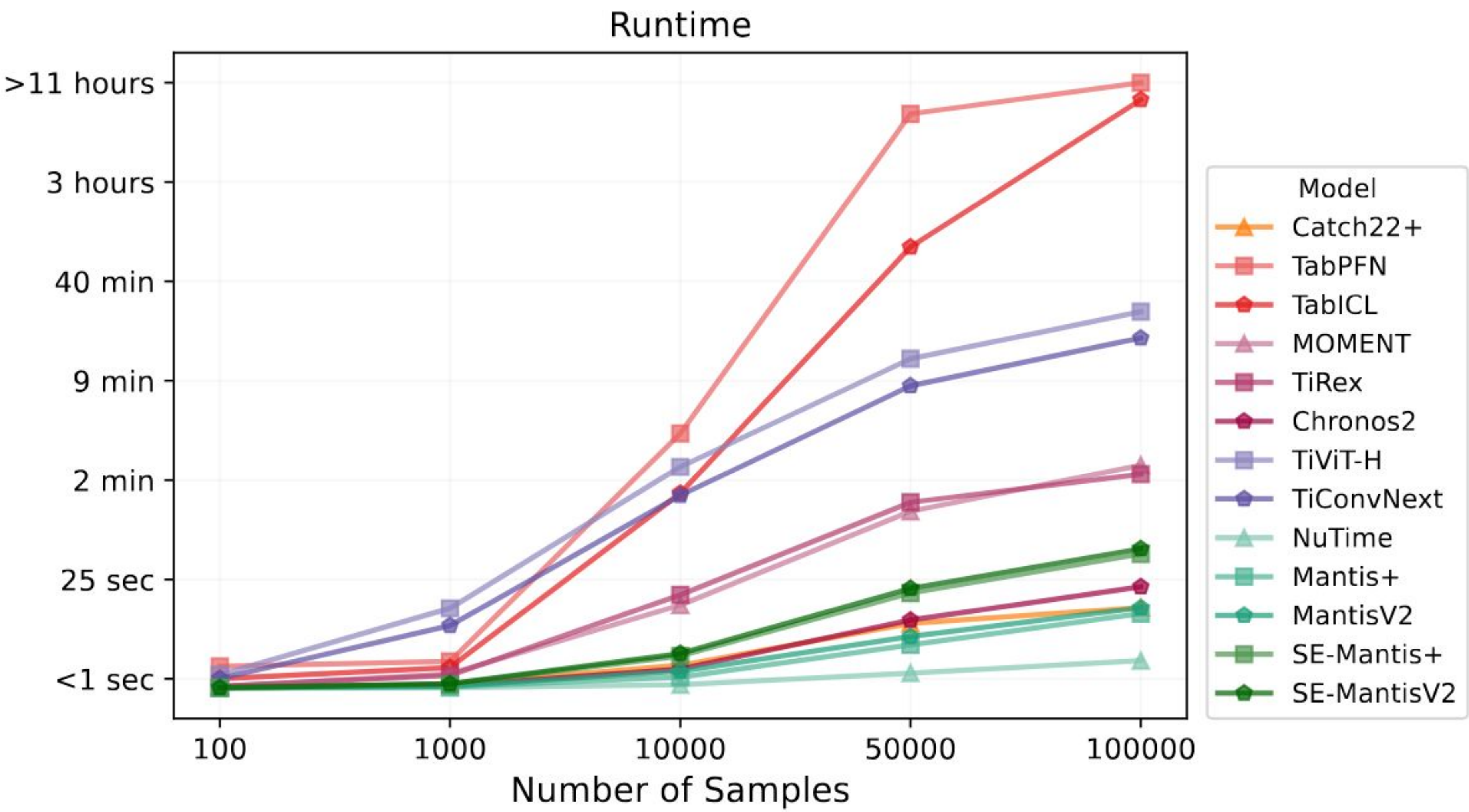
Lightweight models like MantisV2, Mantis+ and NuTime are the most appropriate ones for human activity recognition on device with highest performance and lowest complexity.

Performance on EEG Datasets

	Catch22+	TabPFN	TabICL	MOMENT	TiRex	Chronos2	TiViT-H	TiConvNext	NuTime	Mantis+	MantisV2
Blink	$0.9963_{\pm 0.001}$	0.9178	0.8978	$0.9674_{\pm 0.003}$	$\mathbf{0.997}_{\pm 0.001}$	$0.9733_{\pm 0.01}$	$0.9852_{\pm 0.006}$	$0.98_{\pm 0.002}$	$0.66_{\pm 0.006}$	$0.9778_{\pm 0.002}$	$0.9956_{\pm 0.002}$
CAP-ID	$0.751_{\pm 0.001}$	NaN	NaN	$0.7374_{\pm 0.002}$	$0.8104_{\pm 0.002}$	$0.8179_{\pm 0.001}$	$0.7983_{\pm 0.001}$	$0.8126_{\pm 0.001}$	$0.8044_{\pm 0.0}$	$\mathbf{0.8189}_{\pm 0.001}$	$0.8152_{\pm 0.001}$
CAP-OOD	$0.71_{\pm 0.003}$	NaN	NaN	$0.7408_{\pm 0.001}$	$0.7821_{\pm 0.001}$	$0.7857_{\pm 0.001}$	$0.7781_{\pm 0.001}$	$0.784_{\pm 0.001}$	$0.767_{\pm 0.002}$	$\mathbf{0.791}_{\pm 0.001}$	$0.7859_{\pm 0.0}$
Epilepsy-EEG	$0.9507_{\pm 0.002}$	0.9496	0.9447	$0.952_{\pm 0.001}$	$\mathbf{0.9614}_{\pm 0.001}$	$0.95_{\pm 0.001}$	$0.9548_{\pm 0.0}$	$0.9495_{\pm 0.001}$	$0.9234_{\pm 0.002}$	$0.9518_{\pm 0.001}$	$0.9558_{\pm 0.001}$
FingerMovements	$0.4967_{\pm 0.064}$	0.5	0.49	$0.53_{\pm 0.02}$	$0.52_{\pm 0.04}$	$0.5233_{\pm 0.021}$	$0.5367_{\pm 0.045}$	$0.5333_{\pm 0.059}$	$0.5267_{\pm 0.015}$	$0.51_{\pm 0.026}$	$\mathbf{0.55}_{\pm 0.01}$
PCL-ID	$0.5241_{\pm 0.003}$	NaN	NaN	$0.5431_{\pm 0.012}$	$0.5272_{\pm 0.011}$	$0.5308_{\pm 0.002}$	$0.5347_{\pm 0.007}$	$0.5385_{\pm 0.005}$	$0.5615_{\pm 0.003}$	$\mathbf{0.5764}_{\pm 0.006}$	$0.5716_{\pm 0.003}$
PCL-OOD	$0.5025_{\pm 0.004}$	NaN	NaN	$0.5129_{\pm 0.001}$	$0.4988_{\pm 0.001}$	$0.5072_{\pm 0.004}$	$0.5003_{\pm 0.005}$	$0.5132_{\pm 0.001}$	$0.5415_{\pm 0.004}$	$\mathbf{0.5427}_{\pm 0.002}$	$0.5311_{\pm 0.009}$
SEDFx-ID	$0.7574_{\pm 0.0}$	NaN	NaN	$0.7516_{\pm 0.001}$	$0.7904_{\pm 0.0}$	$0.8_{\pm 0.0}$	$0.7884_{\pm 0.001}$	$0.8008_{\pm 0.001}$	$0.7822_{\pm 0.001}$	$\mathbf{0.8066}_{\pm 0.0}$	$0.8_{\pm 0.0}$
SEDFx-OOD	$0.7142_{\pm 0.001}$	NaN	NaN	$0.7244_{\pm 0.001}$	$0.7714_{\pm 0.0}$	$\mathbf{0.7758}_{\pm 0.0}$	$0.7709_{\pm 0.0}$	$0.7755_{\pm 0.001}$	$0.741_{\pm 0.0}$	$0.7731_{\pm 0.0}$	$0.7636_{\pm 0.0}$
SelfRegulationSCP1	$0.7702_{\pm 0.007}$	**0.8942**	0.8874	$0.7747_{\pm 0.006}$	$0.7884_{\pm 0.012}$	$0.785_{\pm 0.003}$	$0.7986_{\pm 0.007}$	$0.7929_{\pm 0.01}$	$0.7952_{\pm 0.003}$	$0.7736_{\pm 0.005}$	$0.8134_{\pm 0.009}$
SelfRegulationSCP2	$0.4926_{\pm 0.031}$	0.4778	0.5056	$0.4907_{\pm 0.023}$	$0.4963_{\pm 0.049}$	$0.5167_{\pm 0.006}$	$0.4907_{\pm 0.033}$	$0.5056_{\pm 0.011}$	$0.5074_{\pm 0.018}$	$\mathbf{0.5611}_{\pm 0.02}$	$0.5167_{\pm 0.006}$
Avg	0.6969	NaN	NaN	0.7023	0.7221	0.7242	0.7215	0.726	0.6919	0.7348	**0.7363**

- Big datasets => Tabular FMs do not pass the scale.
- Complex data, small performance difference between models.

Running Time



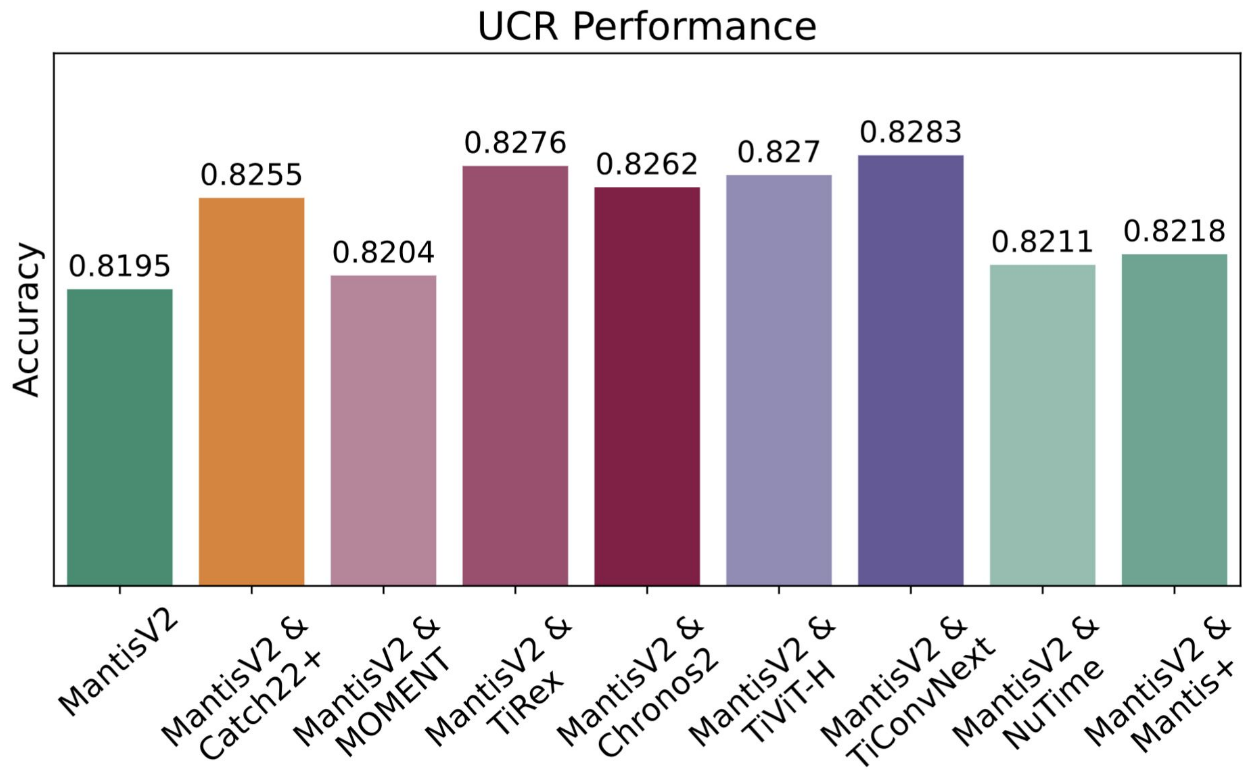
Choice of a Classifier for Probing

Log. Regression (L-BFGS-B)			PyTorch Linear (AdamW)			RF
W/o Norm	MinMax	Standard	W/o Norm	LayerNorm	BatchNorm	
0.763	0.829	**0.837**	0.669	0.71	0.827	0.82

- Self-supervised features are not uniformly scaled.
  - Contrastive loss does not impose identical variance for all dimensions.
  - Architecture contains layer norms, not batch norms.
  - ⇒ We should normalize for linear probing.

- Random forest: not the best, but faster than logistic regression!

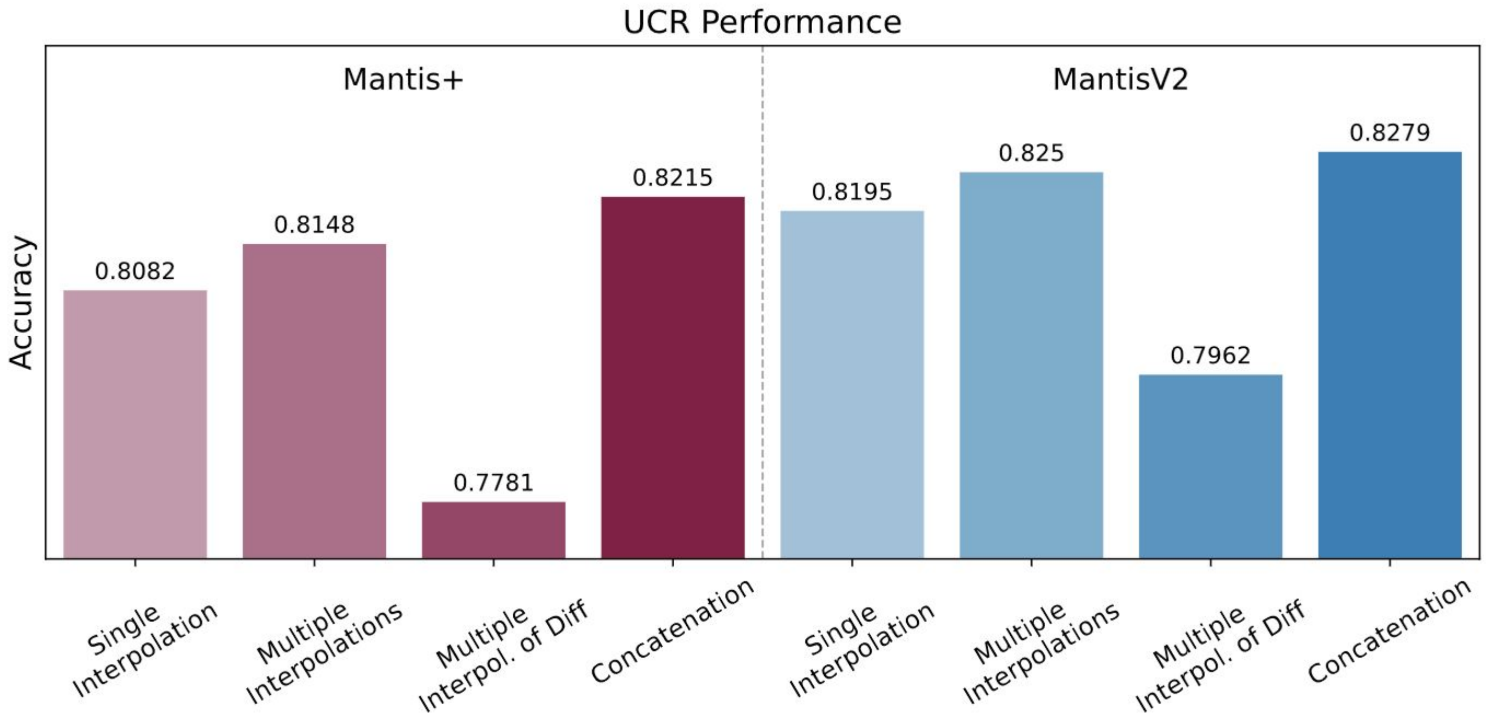
Cross-model Embedding Fusion



- Embeddings of different models are complementary to each other
  ⇒ we can combine them.

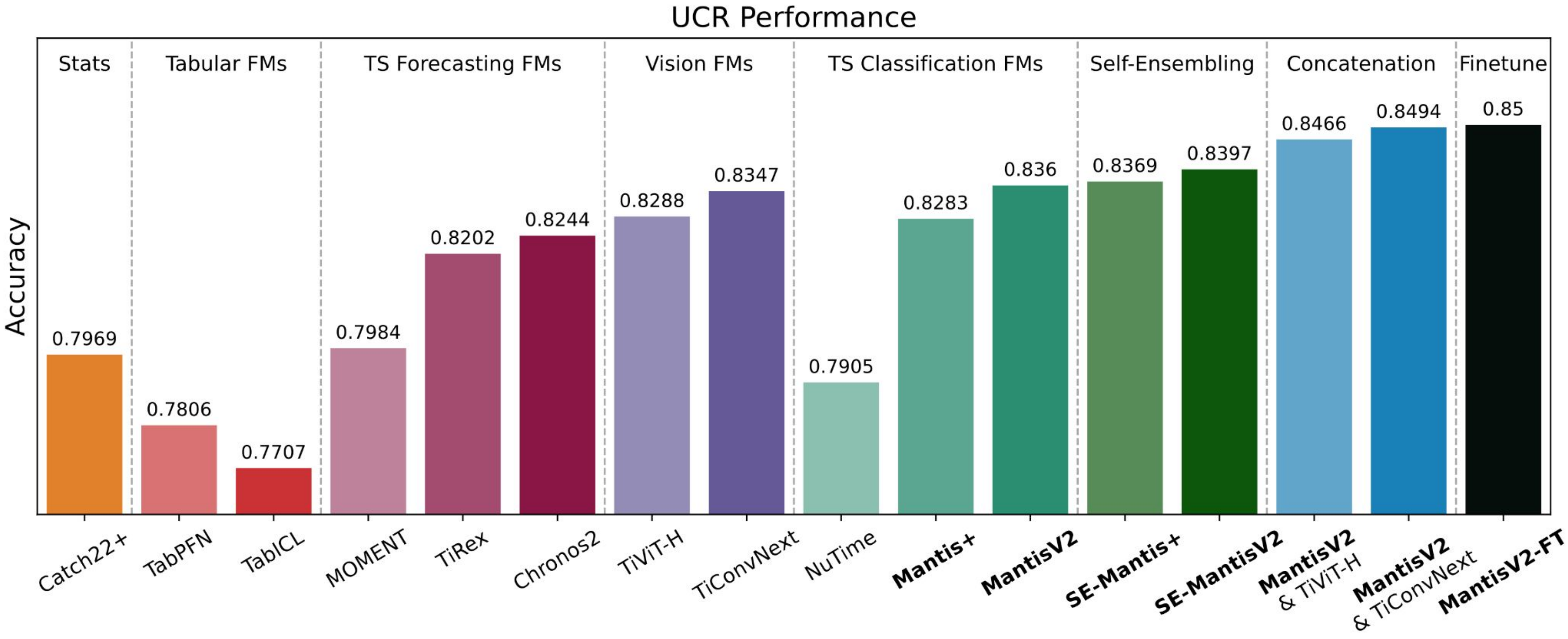
- More complimentary, bigger the boost.

Self-Ensembling (SE)



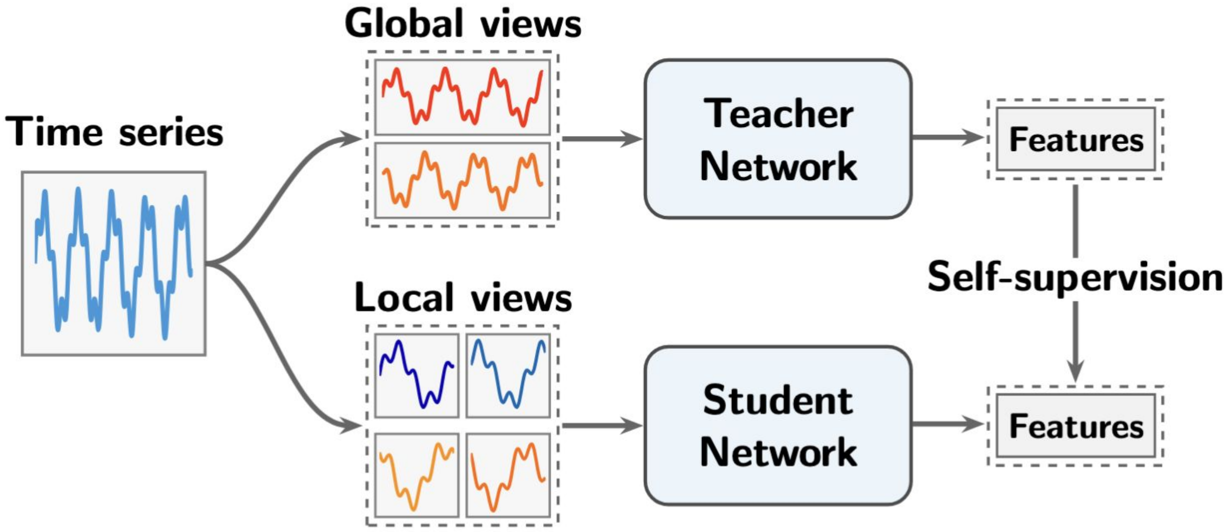
- By perturbing the input, we can generate different embeddings.
- Perturbation No. 1: interpolate a time series to different lengths and get embedding for each of them.
- Perturbation No. 2: do the same for the first-order difference of the input.
- Concatenation of all these embeddings boosts performance on some datasets.

Reaching the Fine-tuning Performance



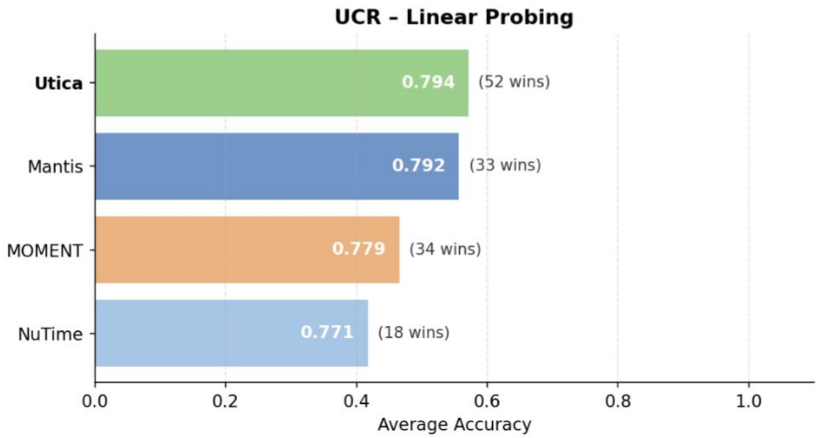
Proof-of-concept: frozen encoders can reach the fine-tuning performance!

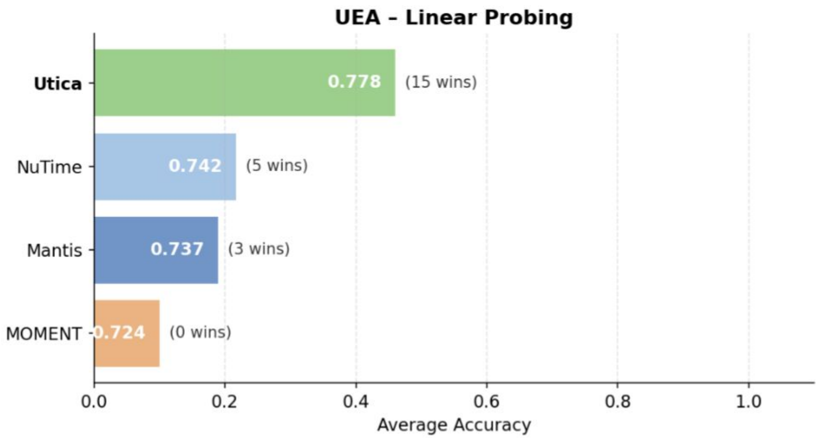
More Ideas: Self-distillation Loss



- Teacher and Student are identical encoders:

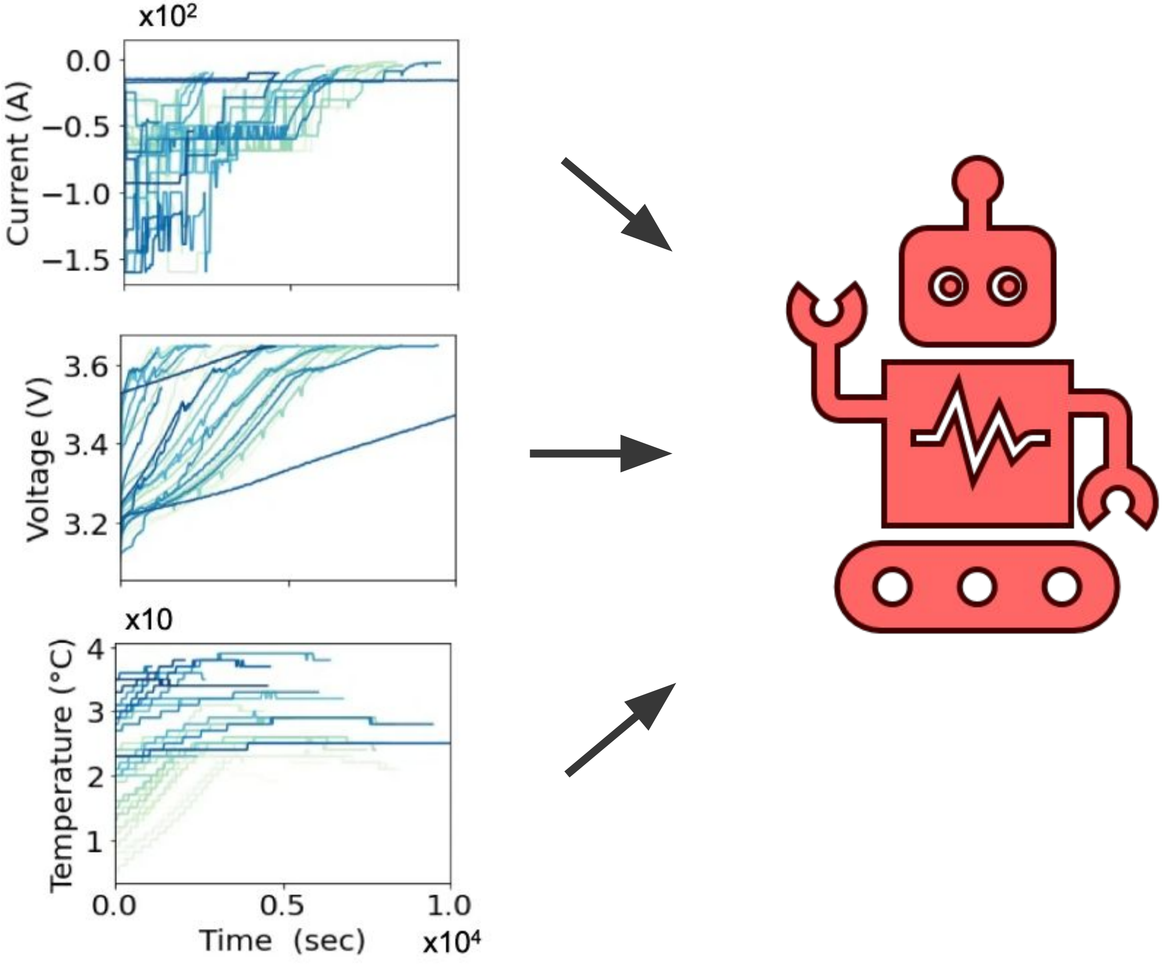
- DINO V2 Self-distillation Loss is a composition of:
  - DINO loss: local and global views give similar embeddings.
  - iBOT loss: masked and global views give similar embeddings.
  - KoLeo regularizer: prevent feature collapse.





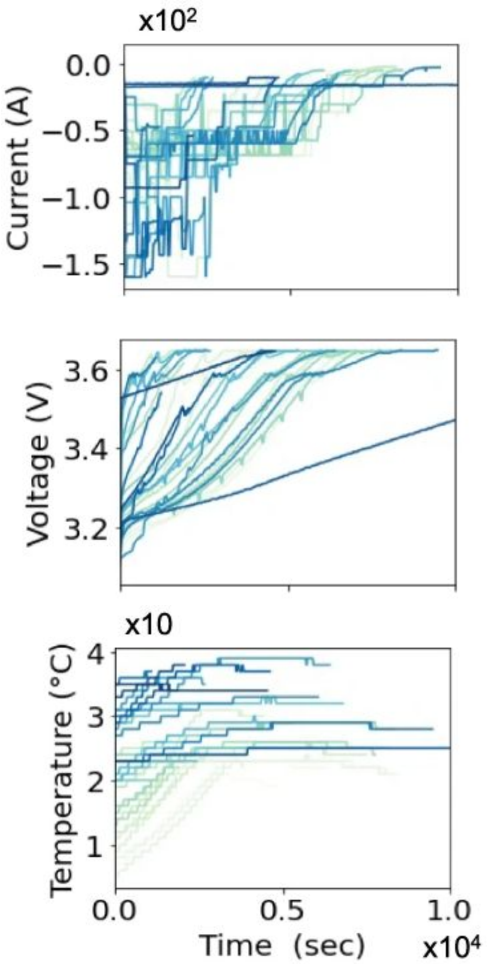
Moakher et al. (2026) UTICA: Multi-Objective Self-Distillation Foundation Model Pretraining for Time Series Classification (*ICLR'26 TSALM workshop*).

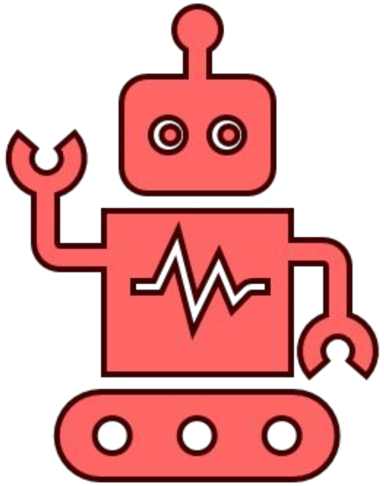
Challenge: Multivariate Time Series



Mantis and other TSFMs are pre-trained on univariate time series data => treat channels independently.

Challenge: Multivariate Time Series



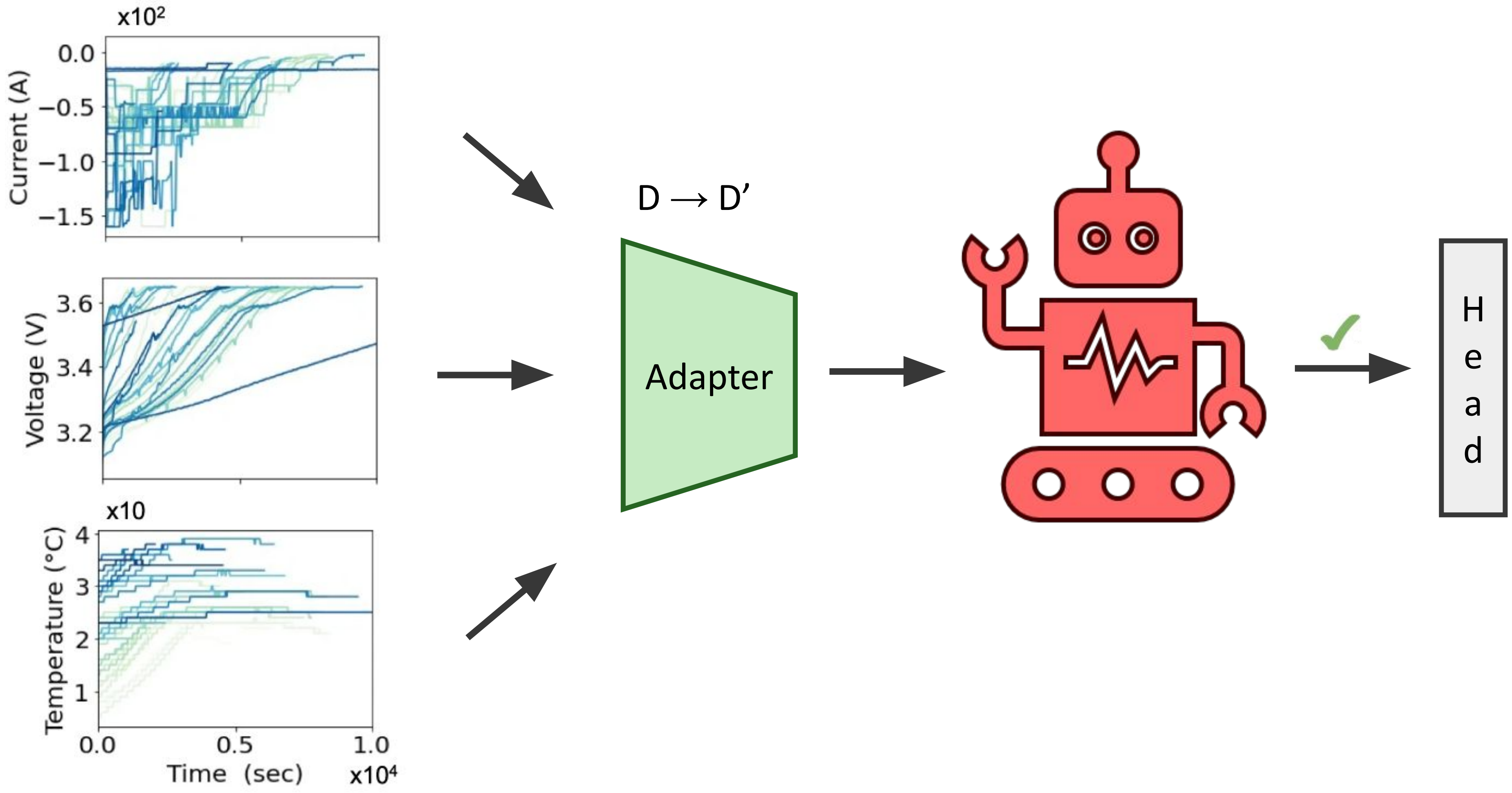


Mantis and other TSFMs are pre-trained on univariate time series data => treat channels independently.

!CUDA Out-of-Memory!

Problem: computationally costly, when number of channels is large when full fine-tuning is needed.

Adapters for Multivariate Classification



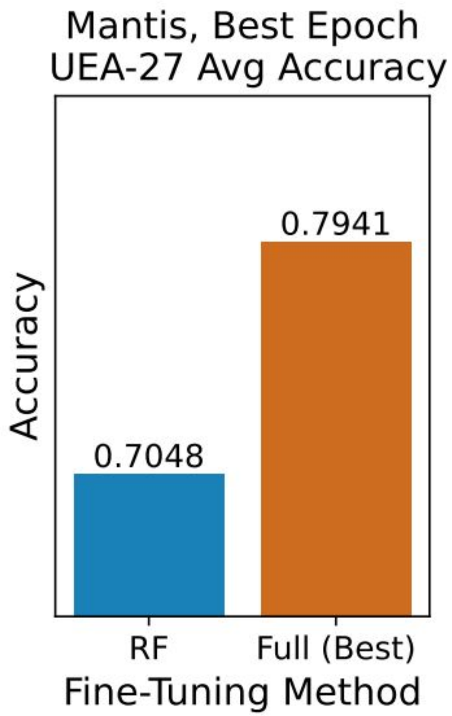
Adapters for Multivariate Classification

- Number of channels ≥28, full fine-tuning with batch size 256 is not possible on a single V100 without an adapter.

	d	No Adapter
ArticularyWordRecognition	9	**0.9933**$_{\pm 0.0}$
BasicMotions	6	**1.0**$_{\pm 0.0}$
CharacterTrajectories	3	0.9928$_{\pm 0.0004}$
Cricket	6	**1.0**$_{\pm 0.0}$
DuckDuckGeese	1345	NaN
ERing	4	**0.9926**$_{\pm 0.0074}$
EigenWorms	6	0.8372$_{\pm 0.0044}$
Epilepsy	3	**1.0**$_{\pm 0.0}$
EthanolConcentration	3	**0.4208**$_{\pm 0.0195}$
FaceDetection	144	NaN
FingerMovements	28	NaN
HandMovementDirection	10	0.4009$_{\pm 0.0206}$
Handwriting	3	0.482$_{\pm 0.0157}$
Heartbeat	61	NaN
InsectWingbeatSubset	200	NaN
JapaneseVowels	12	**0.9811**$_{\pm 0.0054}$
LSST	6	**0.7109**$_{\pm 0.0015}$
Libras	2	0.9389$_{\pm 0.0}$
MotorImagery	64	NaN
NATOPS	24	0.937$_{\pm 0.0116}$
PEMS-SF	963	NaN
PhonemeSpectra	11	0.3421$_{\pm 0.0023}$
RacketSports	6	**0.9408**$_{\pm 0.0}$
SelfRegulationSCP1	6	0.9135$_{\pm 0.0071}$
SelfRegulationSCP2	7	0.5389$_{\pm 0.0096}$
SpokenArabicDigits	13	0.987$_{\pm 0.0009}$
UWaveGestureLibrary	3	**0.9438**$_{\pm 0.0108}$

Adapters for Multivariate Classification

- Number of channels ≥28, full fine-tuning with batch size 256 is not possible on a single V100 without an adapter.
- With adapters, we can fit the computation budget,
- Making full fine-tuning possible on large-dimensional dataset.



	d	No Adapter
ArticularyWordRecognition	9	**0.9933**$_{\pm 0.0}$
BasicMotions	6	**1.0**$_{\pm 0.0}$
CharacterTrajectories	3	0.9928$_{\pm 0.0004}$
Cricket	6	**1.0**$_{\pm 0.0}$
DuckDuckGeese	1345	NaN
ERing	4	**0.9926**$_{\pm 0.0074}$
EigenWorms	6	0.8372$_{\pm 0.0044}$
Epilepsy	3	**1.0**$_{\pm 0.0}$
EthanolConcentration	3	**0.4208**$_{\pm 0.0195}$
FaceDetection	144	NaN
FingerMovements	28	NaN
HandMovementDirection	10	0.4009$_{\pm 0.0206}$
Handwriting	3	0.482$_{\pm 0.0157}$
Heartbeat	61	NaN
InsectWingbeatSubset	200	NaN
JapaneseVowels	12	**0.9811**$_{\pm 0.0054}$
LSST	6	**0.7109**$_{\pm 0.0015}$
Libras	2	0.9389$_{\pm 0.0}$
MotorImagery	64	NaN
NATOPS	24	0.937$_{\pm 0.0116}$
PEMS-SF	963	NaN
PhonemeSpectra	11	0.3421$_{\pm 0.0023}$
RacketSports	6	**0.9408**$_{\pm 0.0}$
SelfRegulationSCP1	6	0.9135$_{\pm 0.0071}$
SelfRegulationSCP2	7	0.5389$_{\pm 0.0096}$
SpokenArabicDigits	13	0.987$_{\pm 0.0009}$
UWaveGestureLibrary	3	**0.9438**$_{\pm 0.0108}$

Python Package

Installation

```
pip install mantis-tsfm
```

Init model / load pre-training weights

```
from mantis.architecture import Mantis8M

network = Mantis8M(device='cuda')
network = network.from_pretrained("paris-noah/Mantis-8M")
```

Extract features

```
from mantis.trainer import MantisTrainer

model = MantisTrainer(device='cuda', network=network)
Z = model.transform(X) # X is your time series dataset
```

Or directly fine-tune the model

```
from mantis.trainer import MantisTrainer

model = MantisTrainer(device='cuda', network=network)
model.fit(X, y) # y is a vector with class labels
probs = model.predict_proba(X)
y_pred = model.predict(X)
```

Thank you for your attention!

GitHub

Paper

My LinkedIn