# Mantis: Towards a Powerful Foundation Model for Time Series Classification

**Vasilii Feofanov**

Huawei Paris Noah's Ark Lab

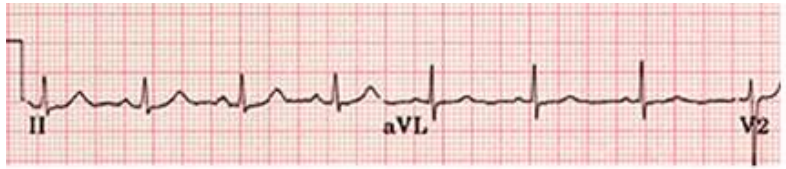
In collaboration with Songkang Wen, Marius Alonso, Romain Ilbert, Hongbo Guo, Malik Tiomoko, Lujia Pan, Jianfeng Zhang, and Ievgen Redko

GitHub

Paper

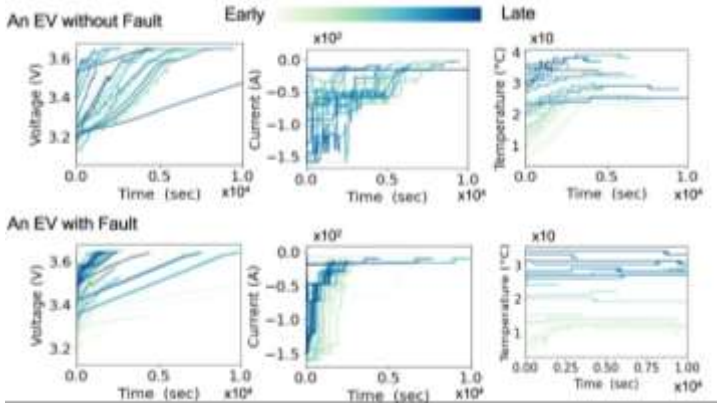
# Time Series Classification

Normal ECG

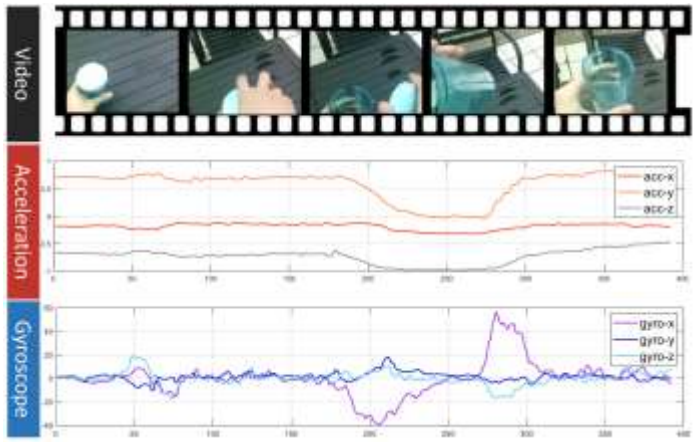



Abnormal ECG

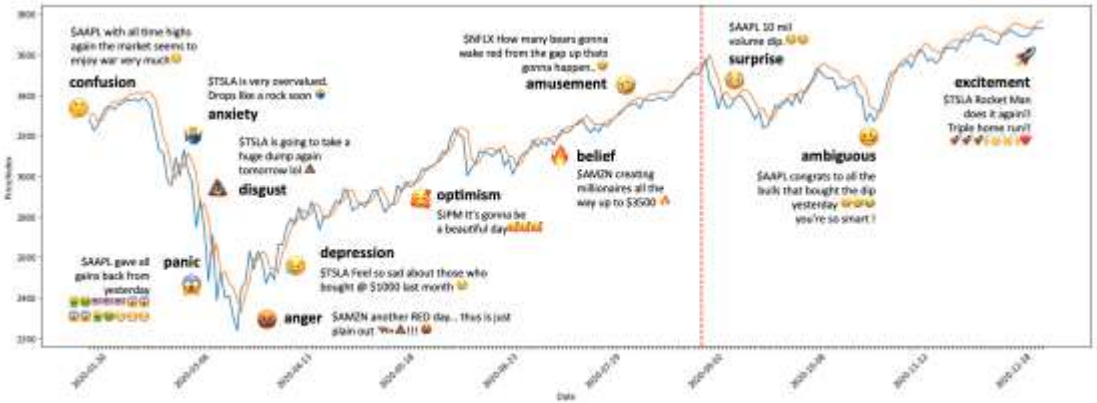
1. Cardiovascular Disease Diagnostic




2. Fault Detection of Electric Vehicle's Battery

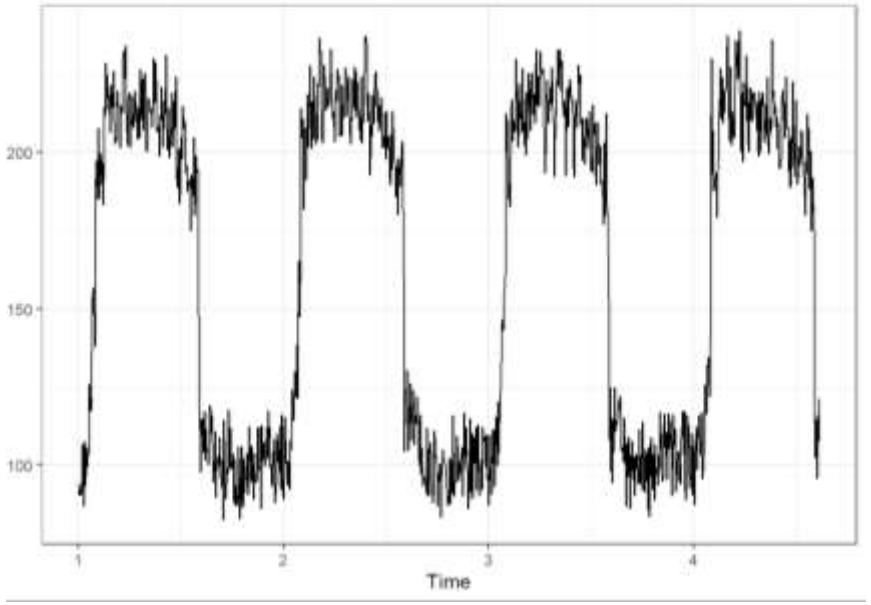



3. Human Activity Recognition




4. Financial Sentiment Analysis

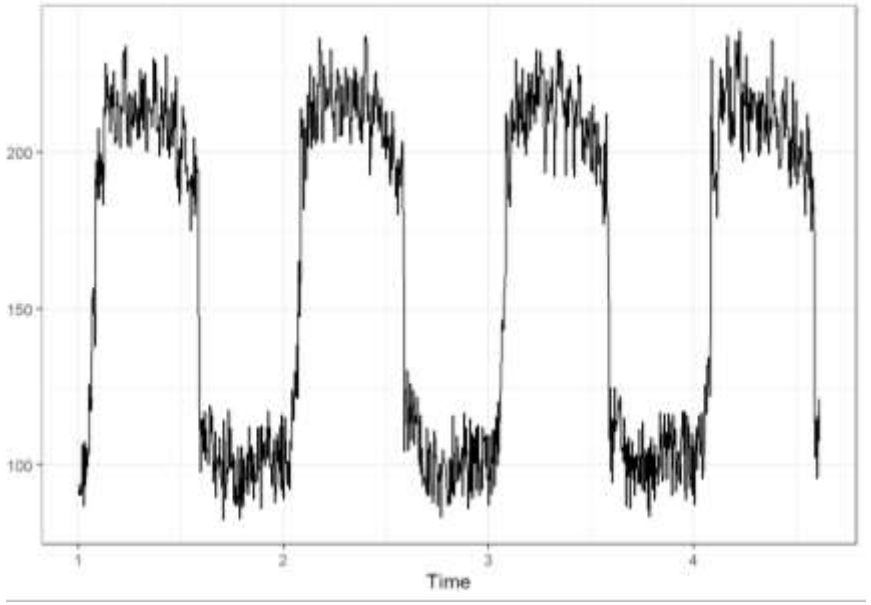
# Forecasting vs Classification

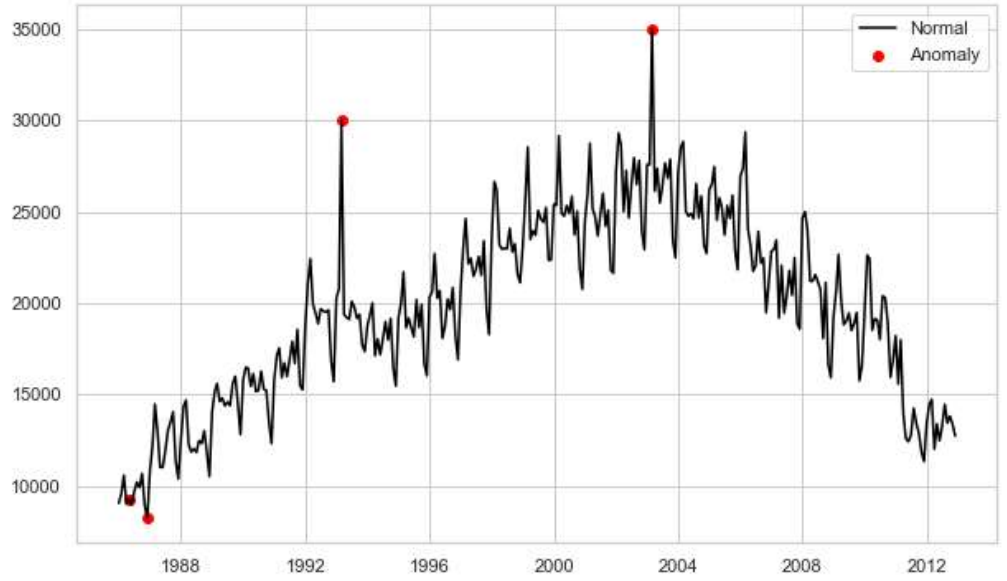



*Forecasting:* predict future patterns based on previously seen ones.

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

# Forecasting vs Classification




*Forecasting:* predict future patterns based on previously seen ones.

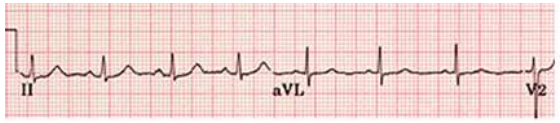
Jittering is often somewhat arbitrary => smoothing may be a good thing.

*Classification:* discriminate, detect anomalies.

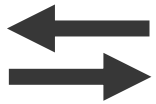
Some spikes are a very important source of information!

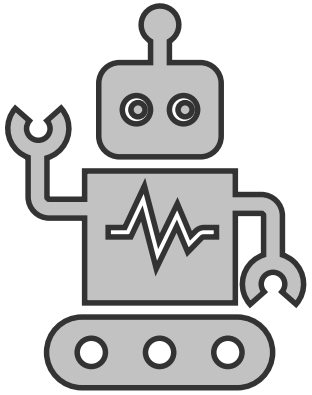
# Traditional Machine Learning

Normal ECG




Abnormal ECG



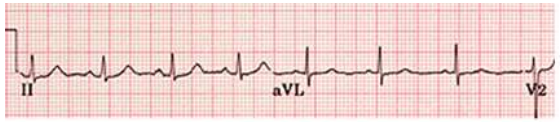


Traditional machine learning pipeline:

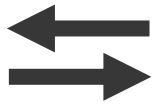
1. Collect training data.
2. Train a new model.
3. Deploy it.

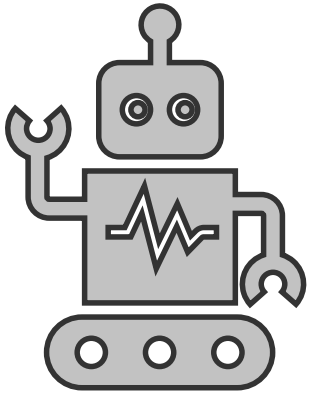
# Traditional Machine Learning

Normal ECG




Abnormal ECG



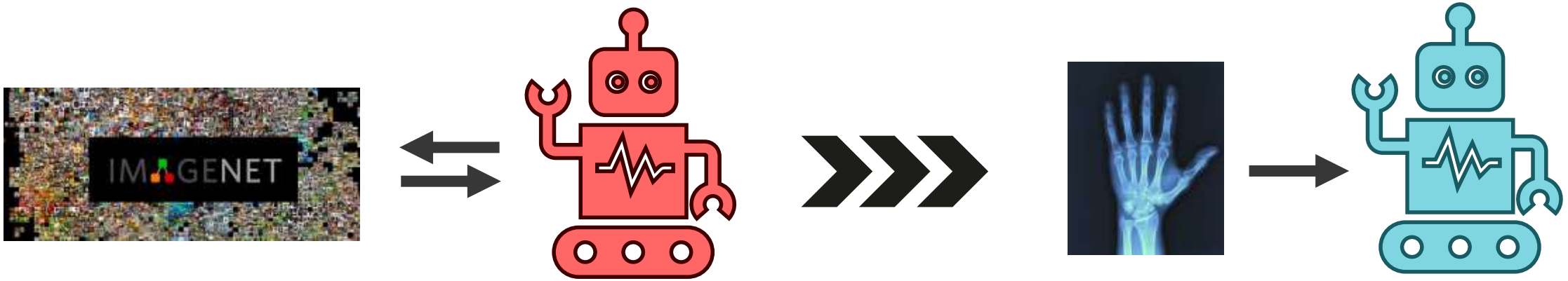


Traditional machine learning pipeline:

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.

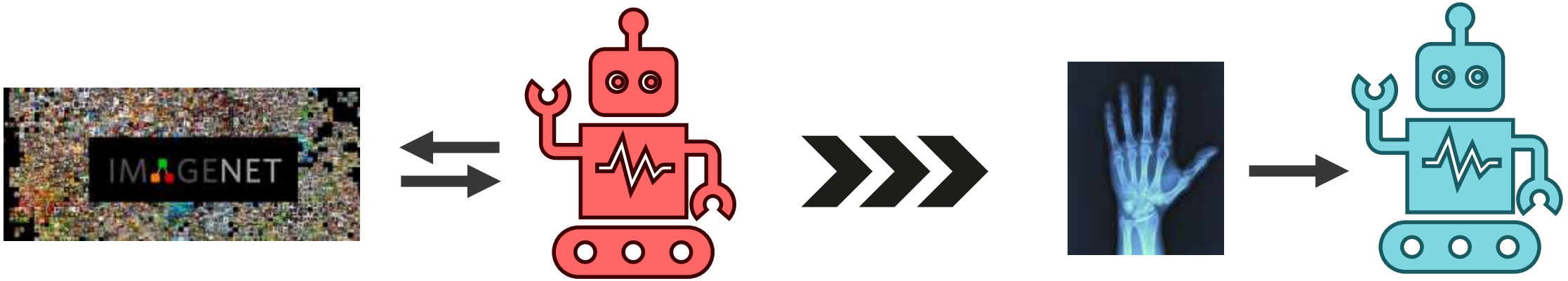
# Success of LLM and Pre-training

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

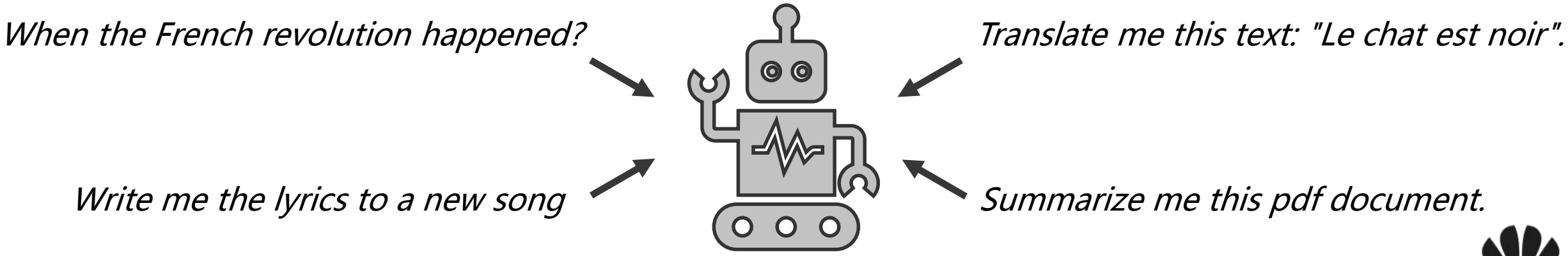


# Success of LLM and Pre-training

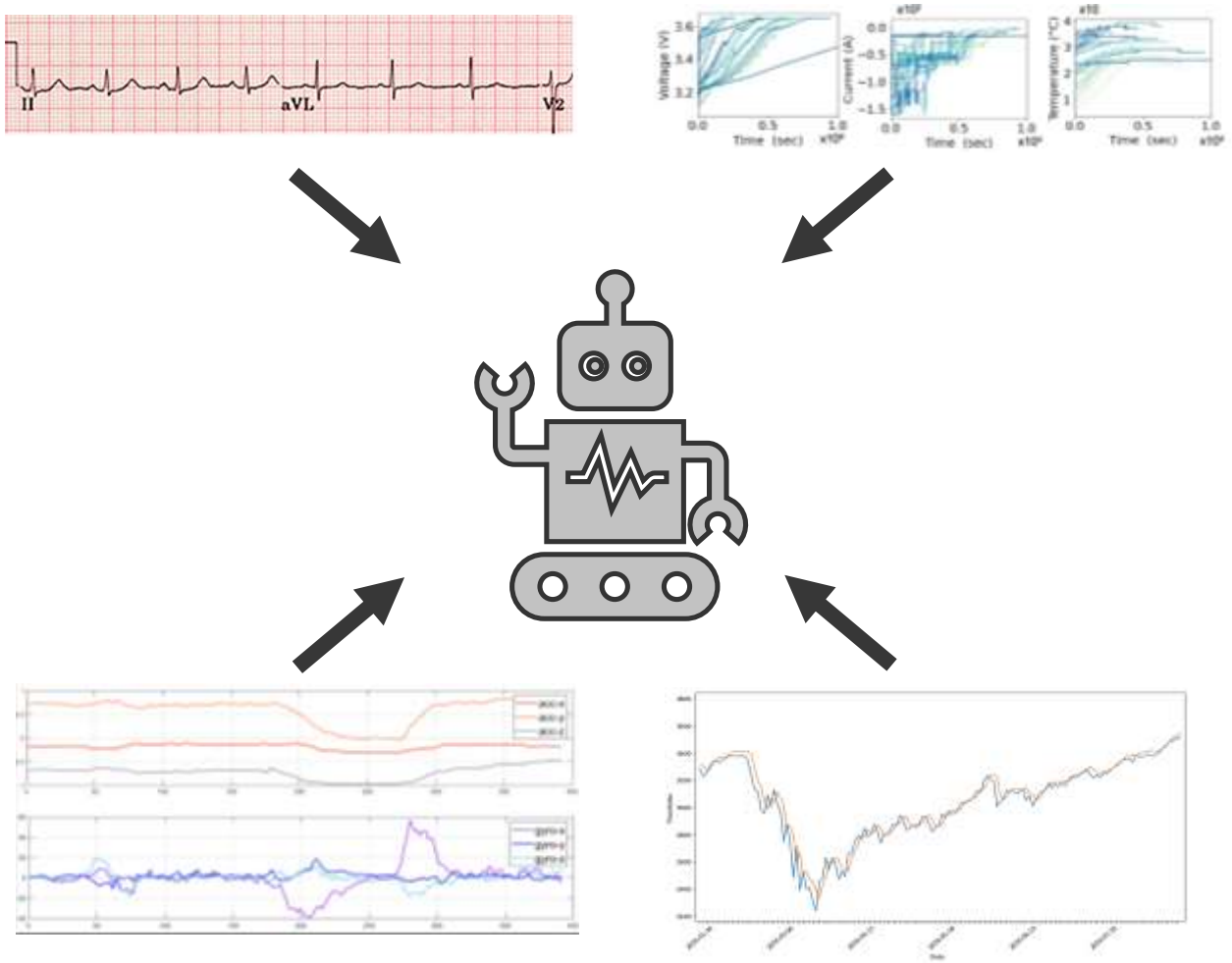
1. In Computer Vision, models are pre-trained on ImageNet and applied for new problems.




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



# Time Series Foundation Model (TSFM)



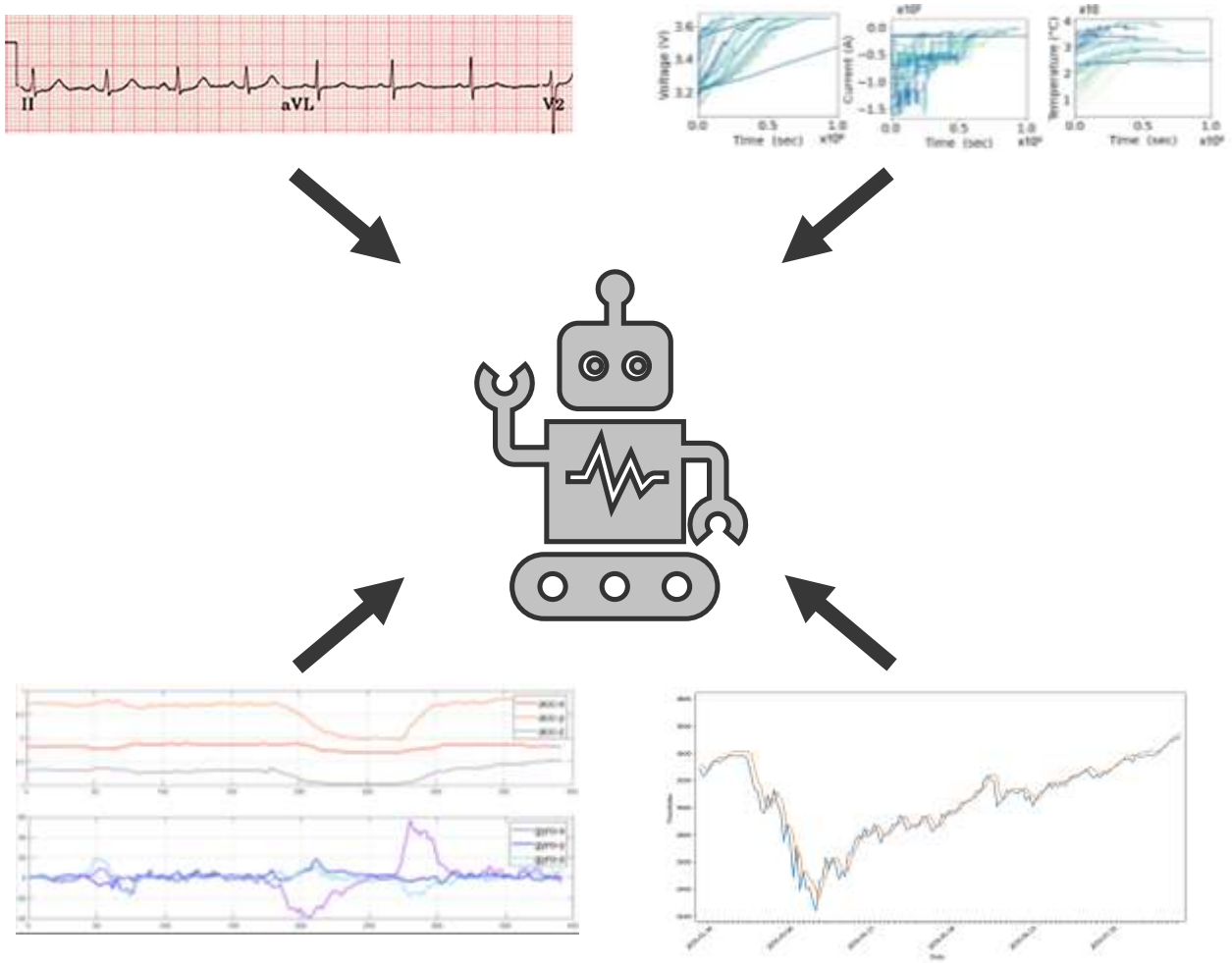

The goal of a TSFM is to learn a projector using a large corpus of various datasets.

Advantages:

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

# Mantis: Framework

# Time Series Foundation Model (TSFM)



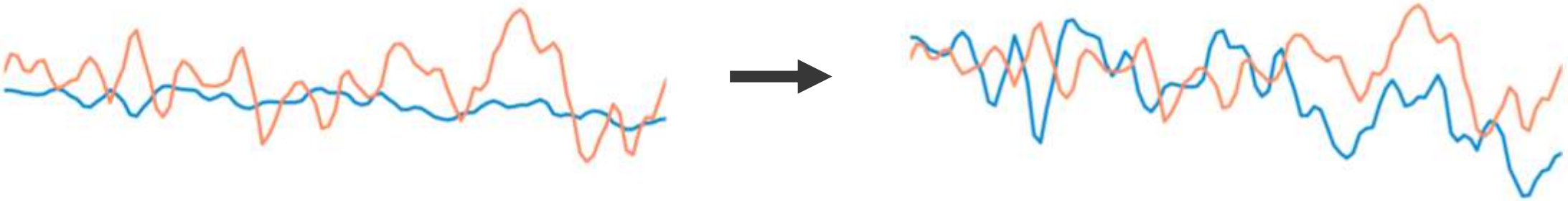

The goal of a TSFM is to learn a projector using a large corpus of various datasets.

**Step 0: Data Preparation**

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

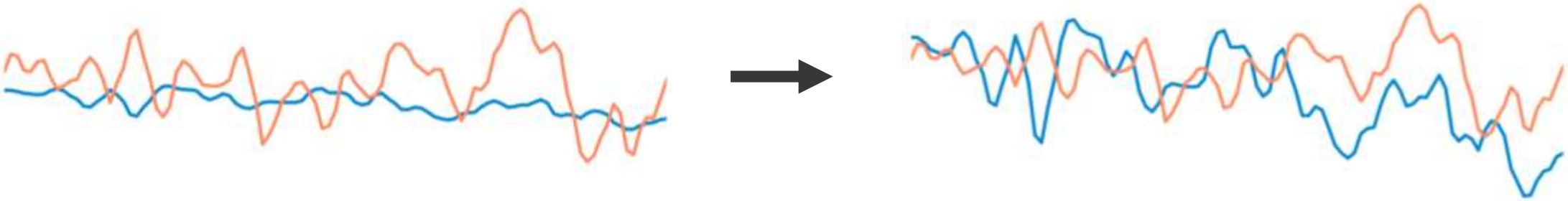
# Pre-training Dataset Preparation

1. Instance-Wise Normalization

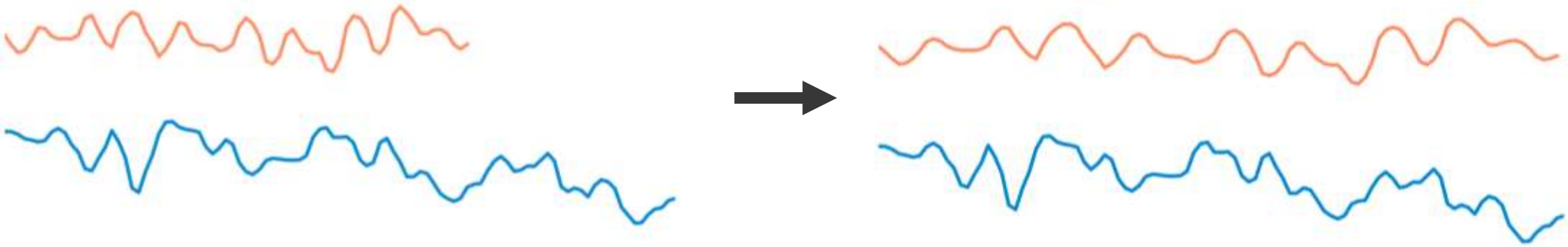


# Pre-training Dataset Preparation

1. Instance-Wise Normalization

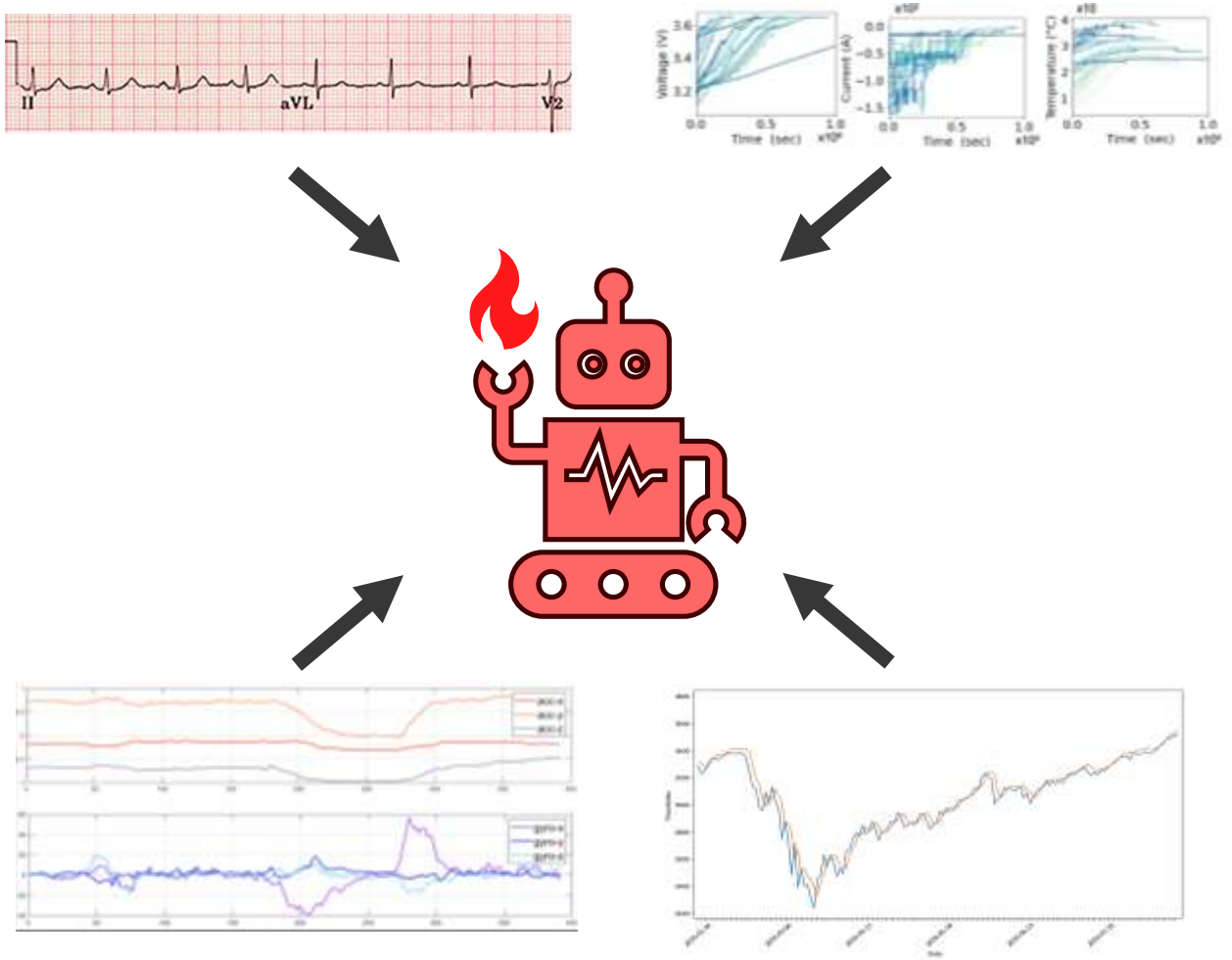


2. Resize by Interpolation



# Time Series Foundation Model (TSFM)

**Step 1: Pre-training**

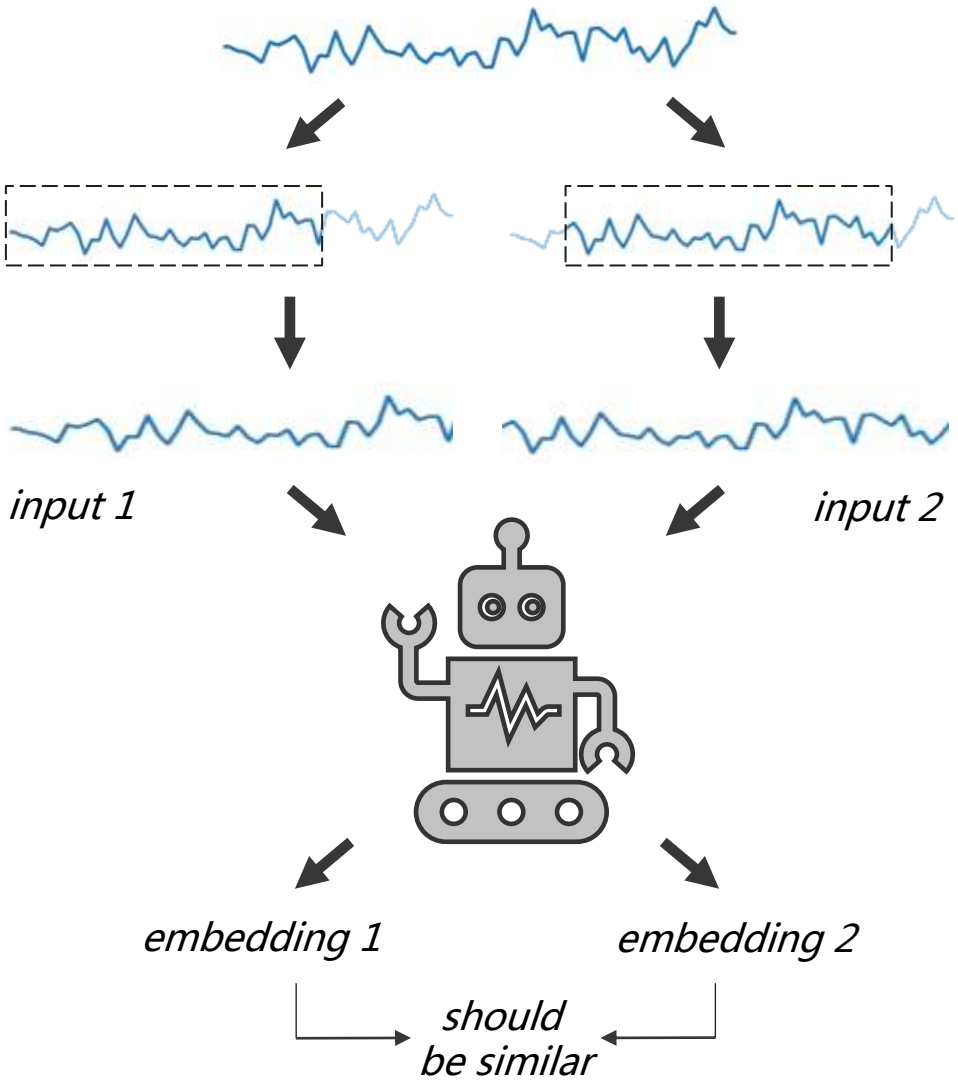



Pre-train a projector using one of the two options:

1. Unsupervised (self-supervised):
   a. Contrastive learning.
   b. Masked reconstruction.

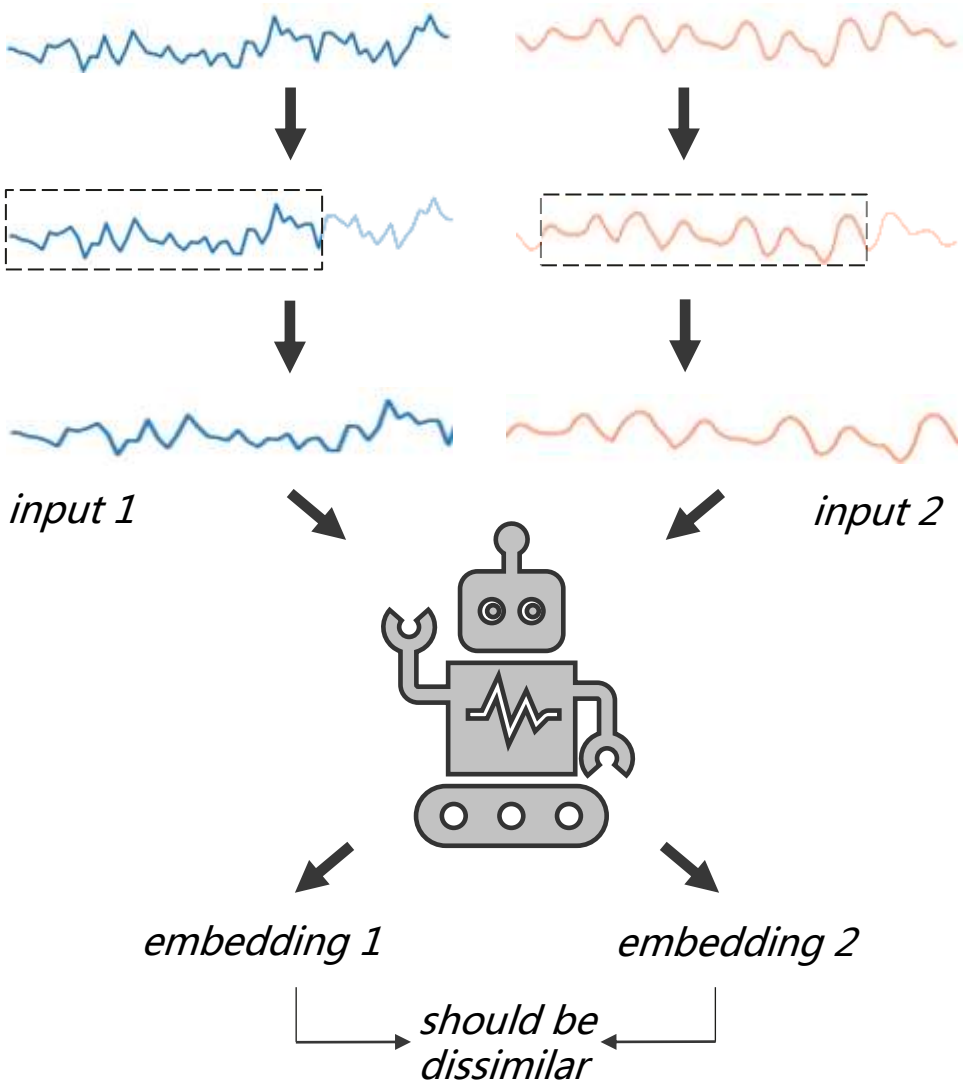
2. Supervised (multi-task).

# 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.

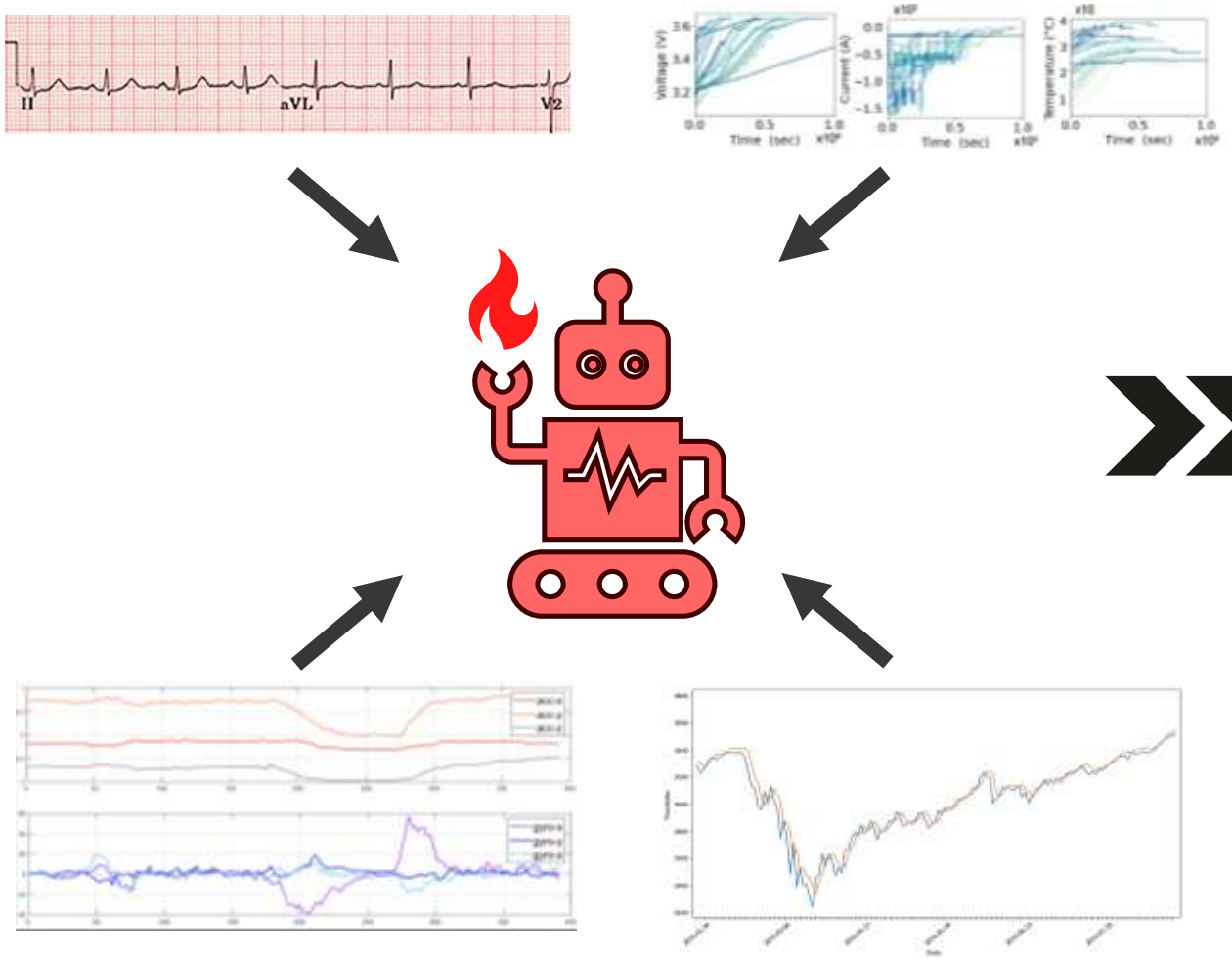
# Pre-training by Contrastive Learning




- 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

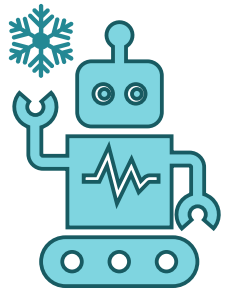
# Time Series Foundation Model (TSFM)

**Step 1: Pre-training**




**Step 2: Fitting to a New Task**

Option 1:

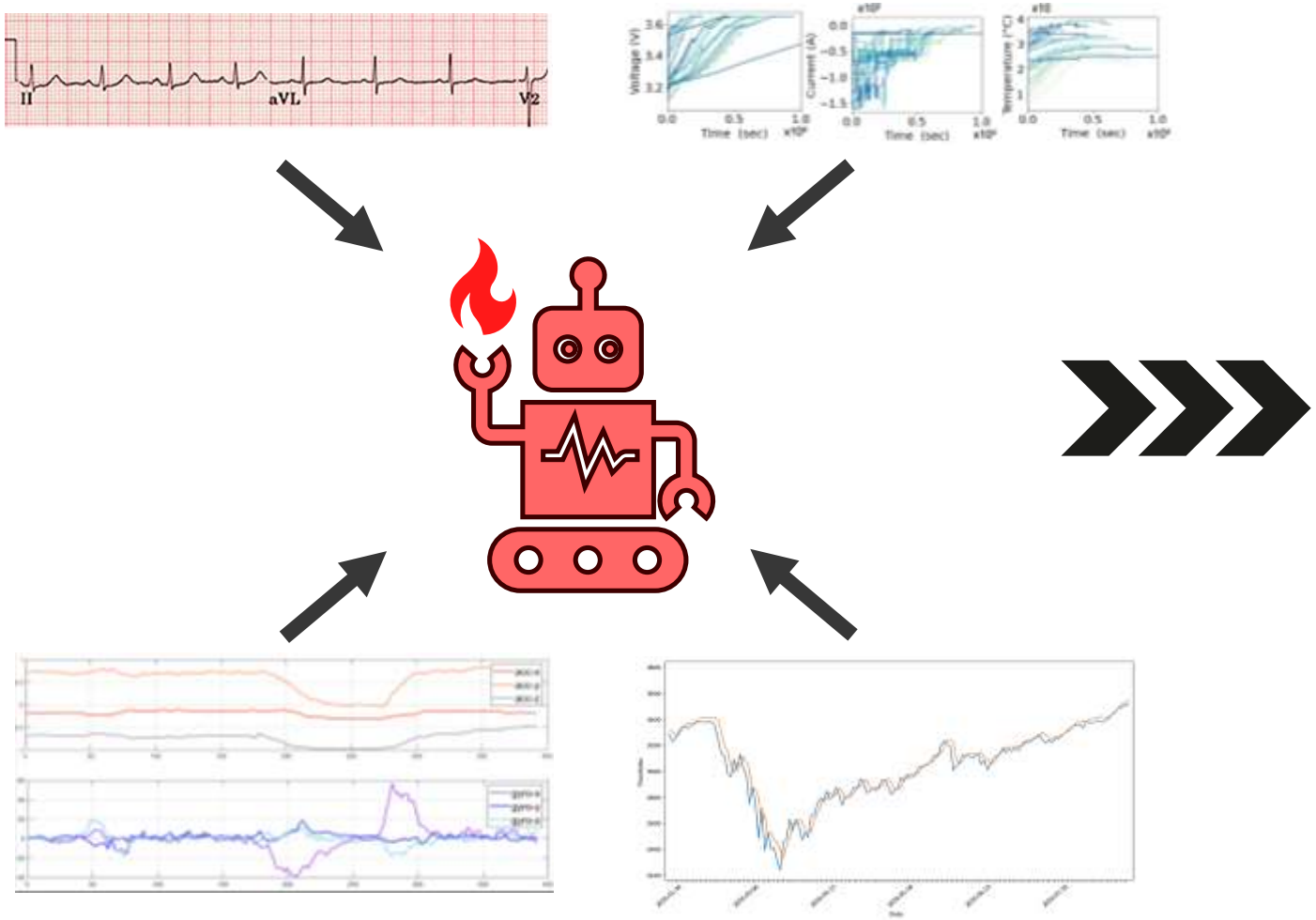



Random Forest

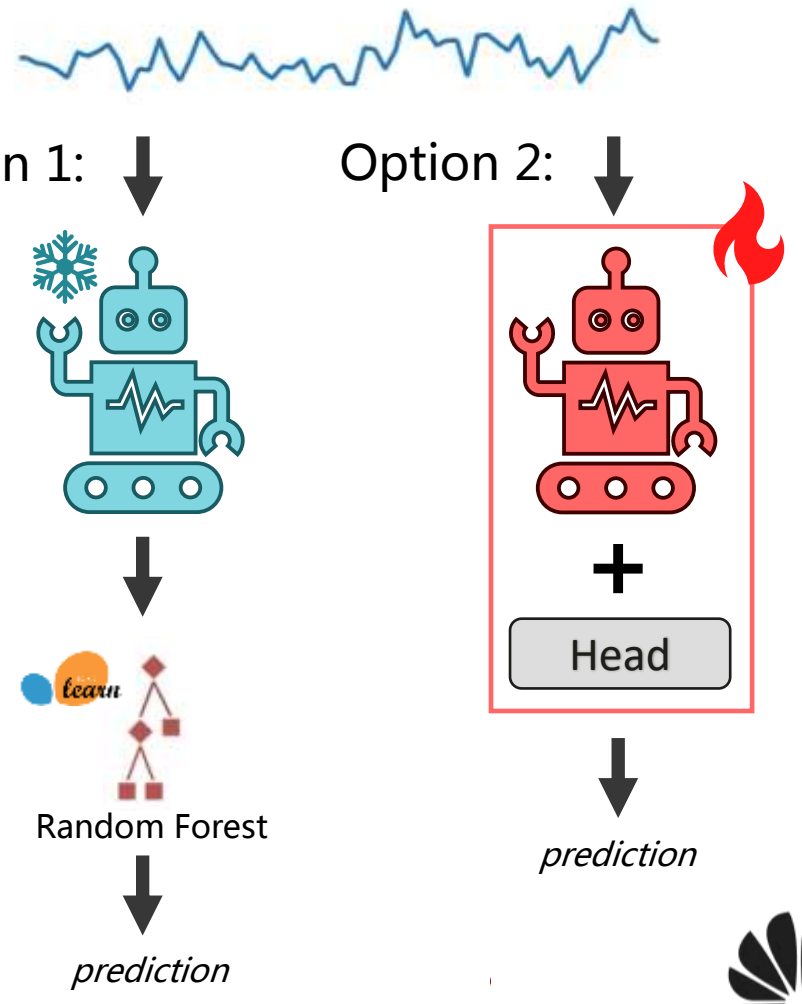
*prediction*

# Time Series Foundation Model (TSFM)

**Step 1: Pre-training**



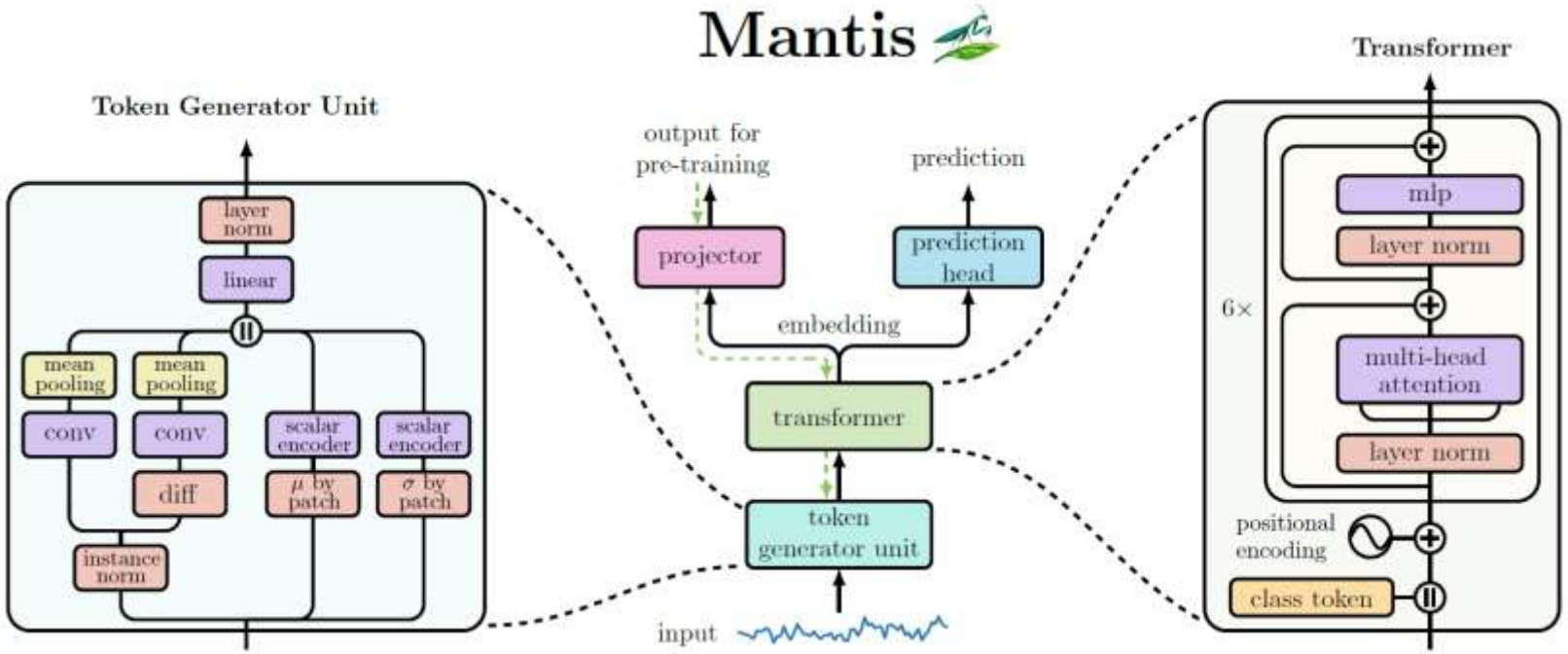

**Step 2: Fine-tuning to New Task**



# Mantis: Architecture



# Architecture

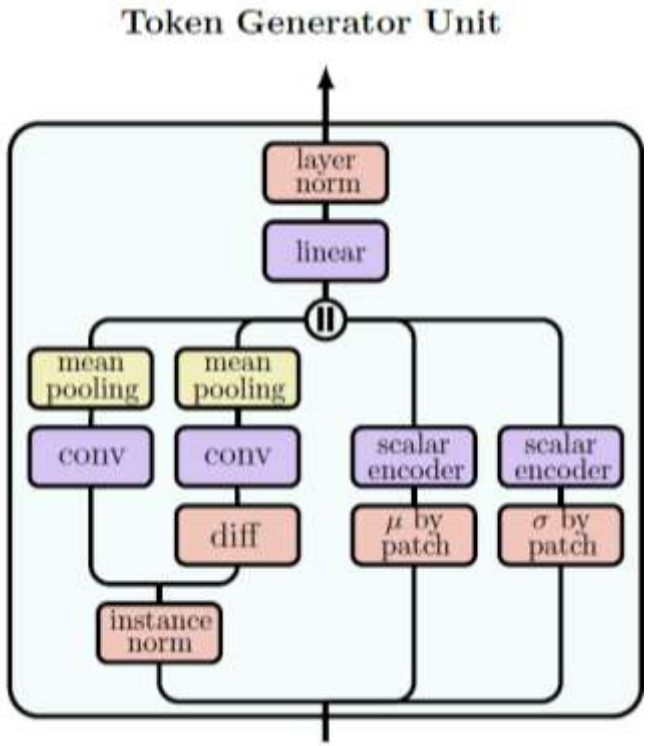



*Tokenization is key to unlock the power of the transformer.*

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

# Token Generator Unit

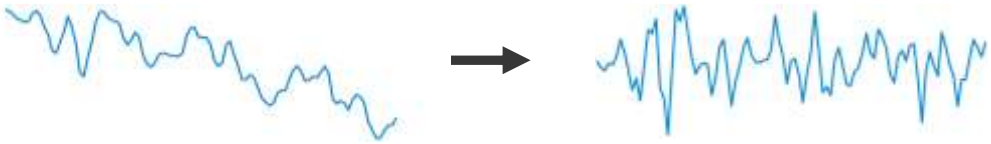
1. 32 x-patches:
   - norm x → conv → mean pooling.

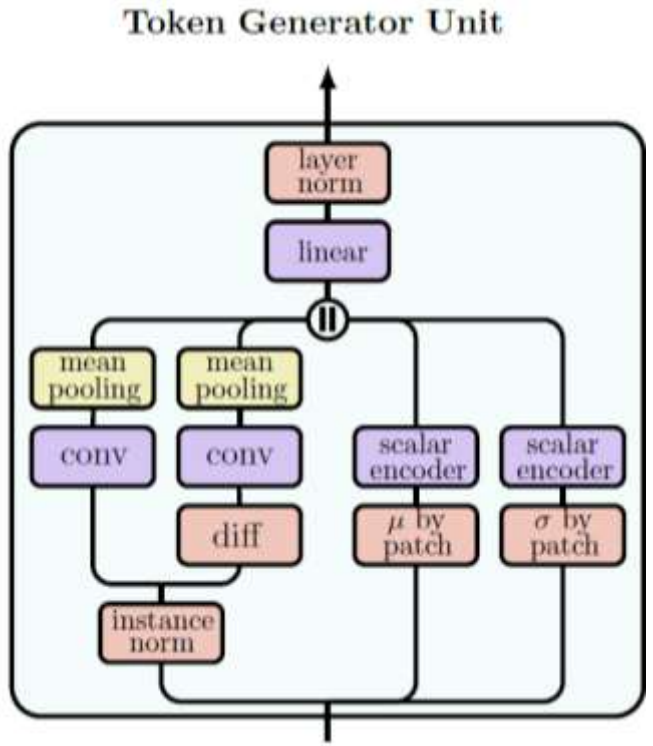


# Token Generator Unit

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.

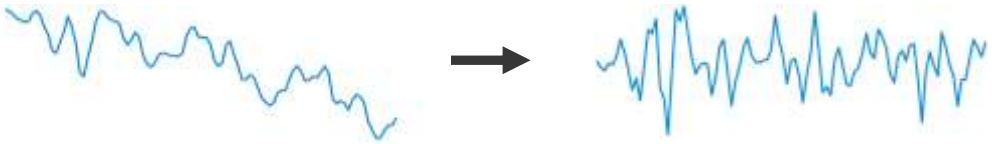




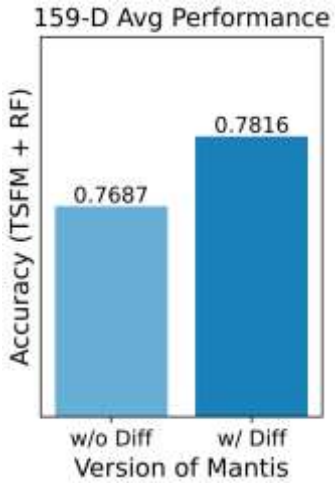
# Token Generator Unit

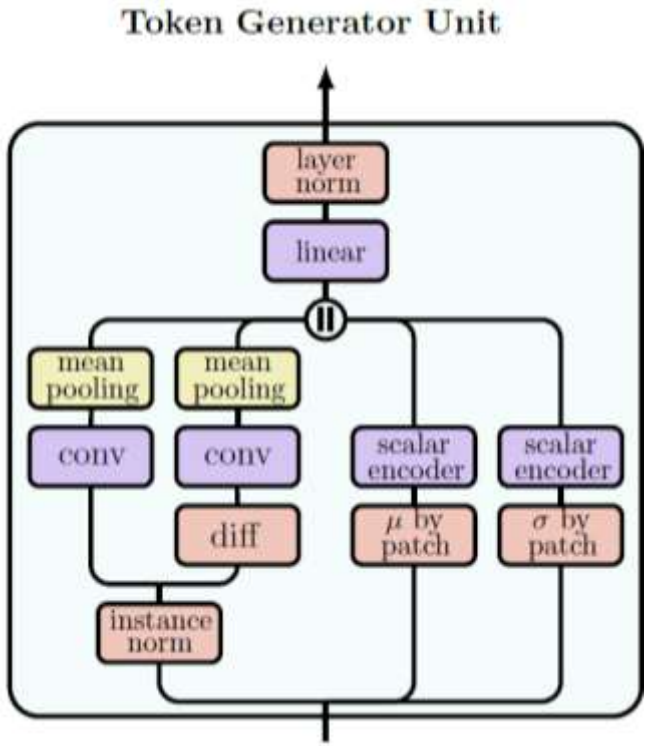
1. 32 x-patches:
   - norm x → conv → mean pooling.

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.

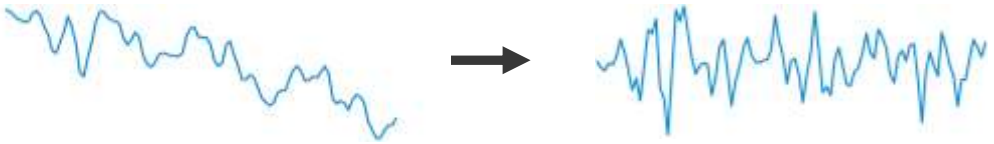


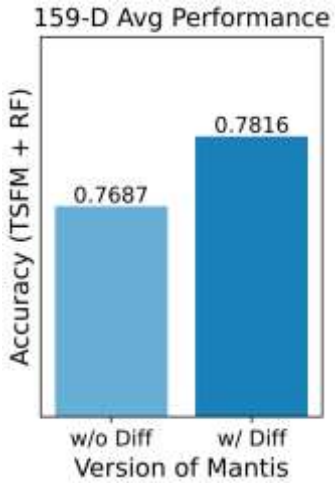
# Token Generator Unit

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

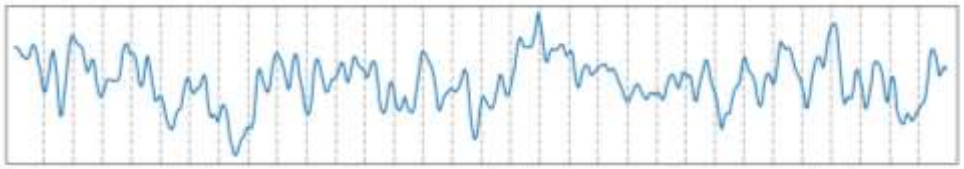
2. 32 diff-x-patches:
   - norm x → diff → conv → mean pooling.
   - diff: x[t]-x[t-1], makes time series stationary.

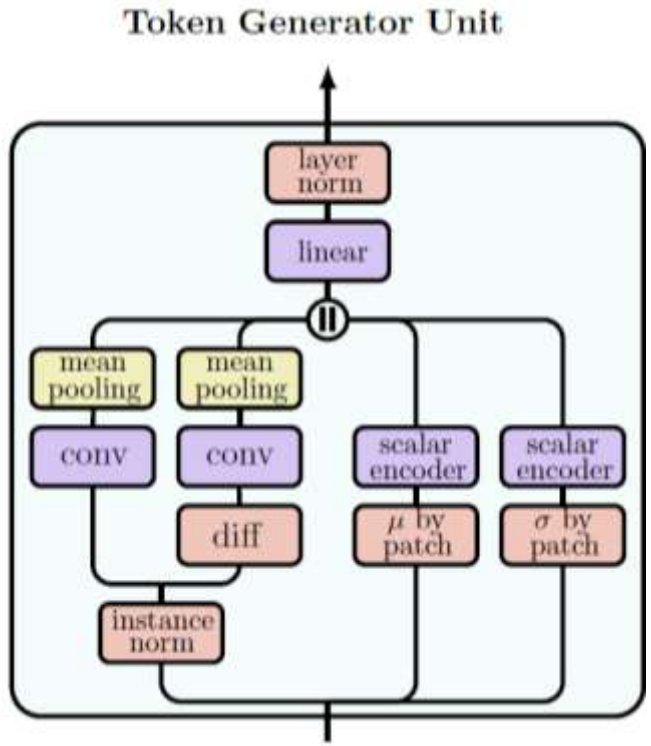


   - ablation study: diff improves performance.




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

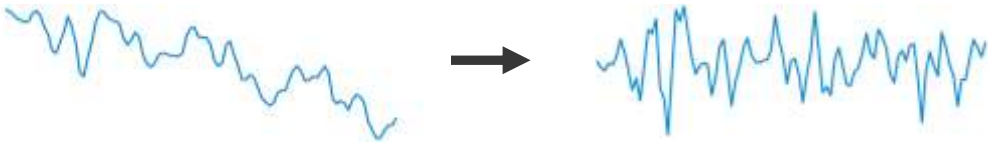




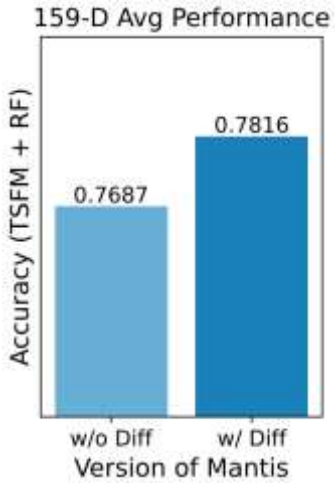
# Token Generator Unit

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

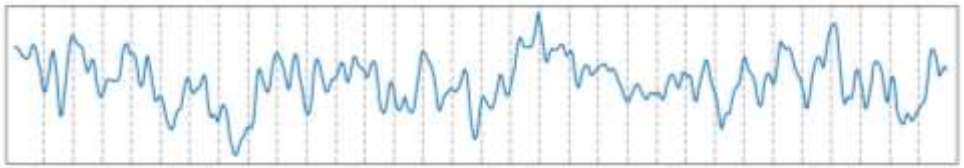
2. 32 diff-x-patches:
   - norm x → diff → conv → mean pooling.
   - diff: x[t]-x[t-1], makes time series stationary.

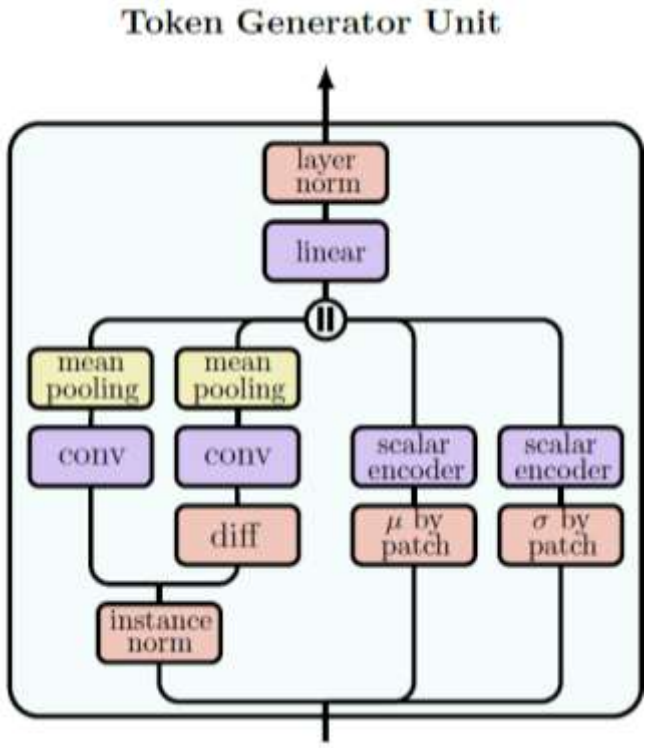


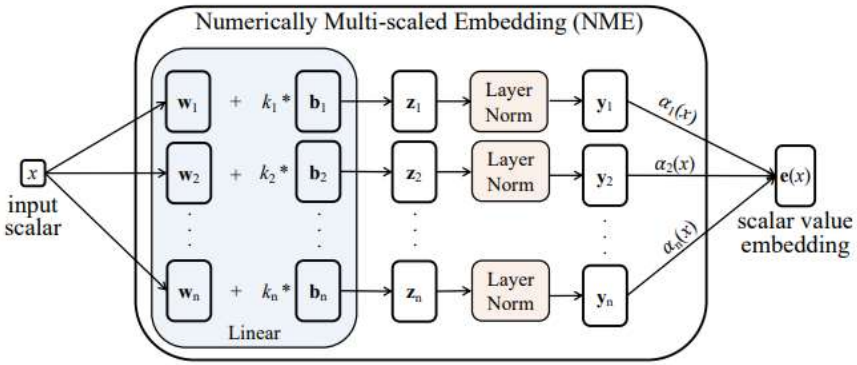
   - ablation study: diff improves performance.




3. Scalar encoder (Lin et al., 2024):
   - split time series into 32 non-overlapping patches.
   - compute $\mu$ and $\sigma$ for each patch.
   - encode each scalar by a high-dimensional vector.





# Mantis: Experimental Results

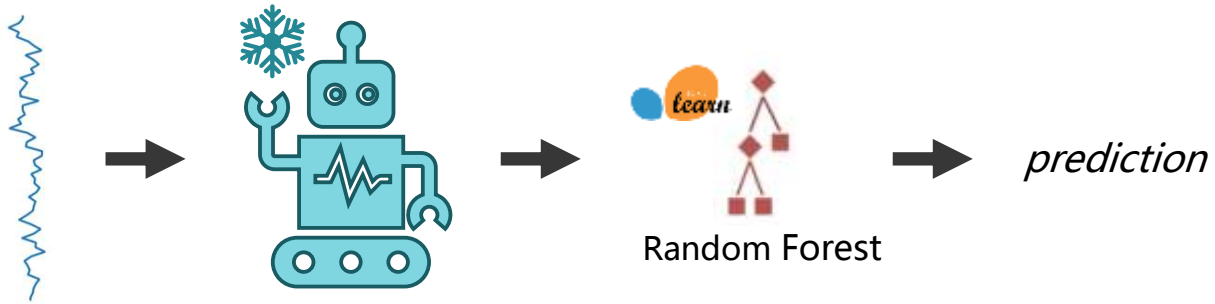
# Comparison with Other Methods

| Model name | Multivar Support | Zero-shot | HuggingFace Support | Model Size |
|---|---|---|---|---|
| Mantis | ✓ | ✓ | ✓ | ● ○ ○ |
| UniTS | ✓ | × | × | ● ○ ○ |
| NuTime | ✓ | ✓ | × | ● ○ ○ |
| GPT4TS | ✓ | × | × | ● ● ○ |
| MOMENT | × | ✓ | ✓ | ● ● ● |

- **Mantis: 8M parameters pre-trained on 1.8M samples.**
- UniTS: multi-task model with 1M params pre-trained in a supervised way on 0.3M samples.
- NuTime: classification TSFM with 2M params pre-trained on 1.8M samples.
- GPT4TS: 6 pre-trained layers from GPT2 + some layers to fine-tune, 80M params.
- MOMENT: based on T5 architecture, 385M params, 1.13B samples.

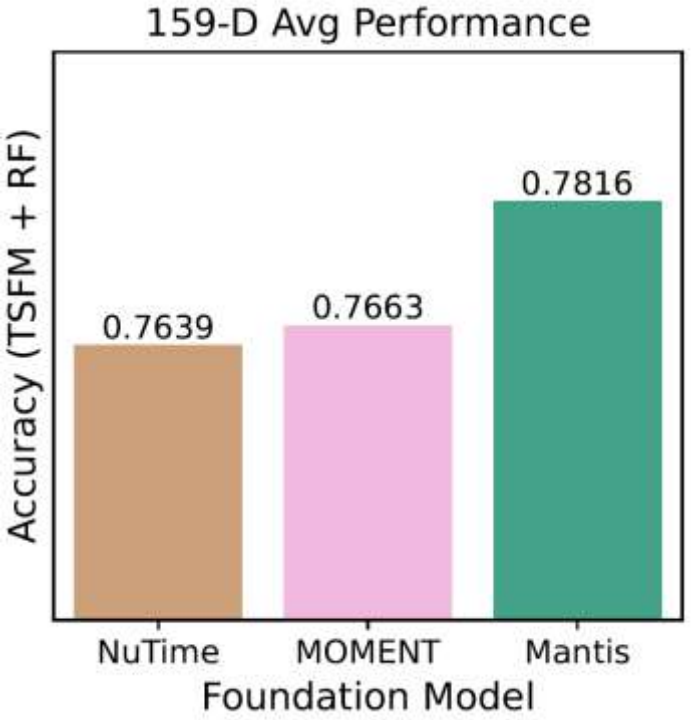


# Zero-shot Feature Extraction Results




For each dataset:

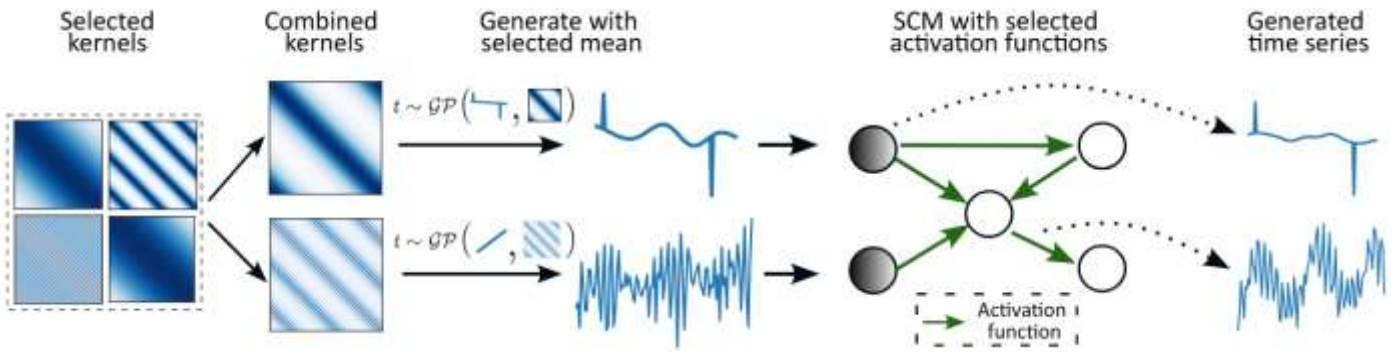
- Extract features for train and test sets
- Learn a Random Forest on train set
- Evaluate accuracy on test set

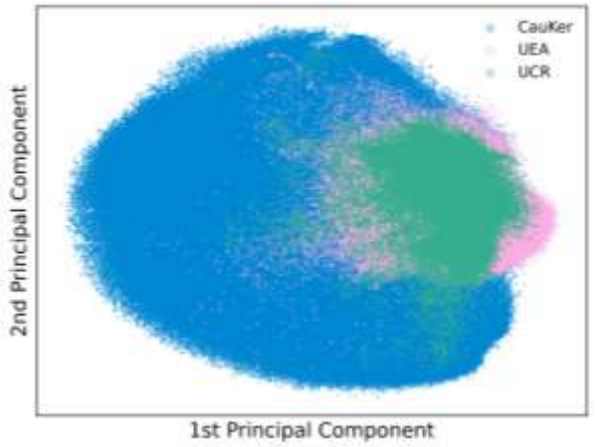


# Zero-shot with Synthetic Pre-training Data

CauKer: synthetic data generation algorithm that yields

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



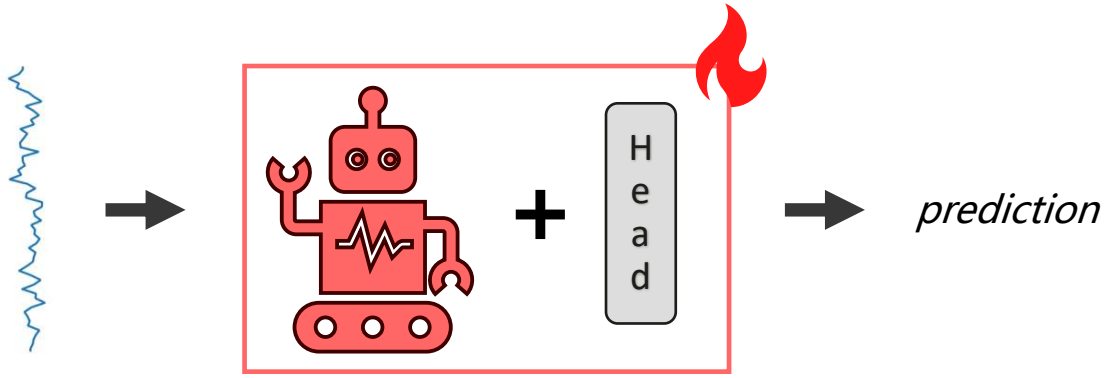




| Pre-train Data | Size | UCR Included? | UCR acc. (%) |
|---|---|---|---|
| Real | 100K | No | 78.29 |
| CAUKER | 100K | No | 78.81 |
| CAUKER | 1M | No | 78.98 |
| Real | 1.89M | Yes | 79.21 |



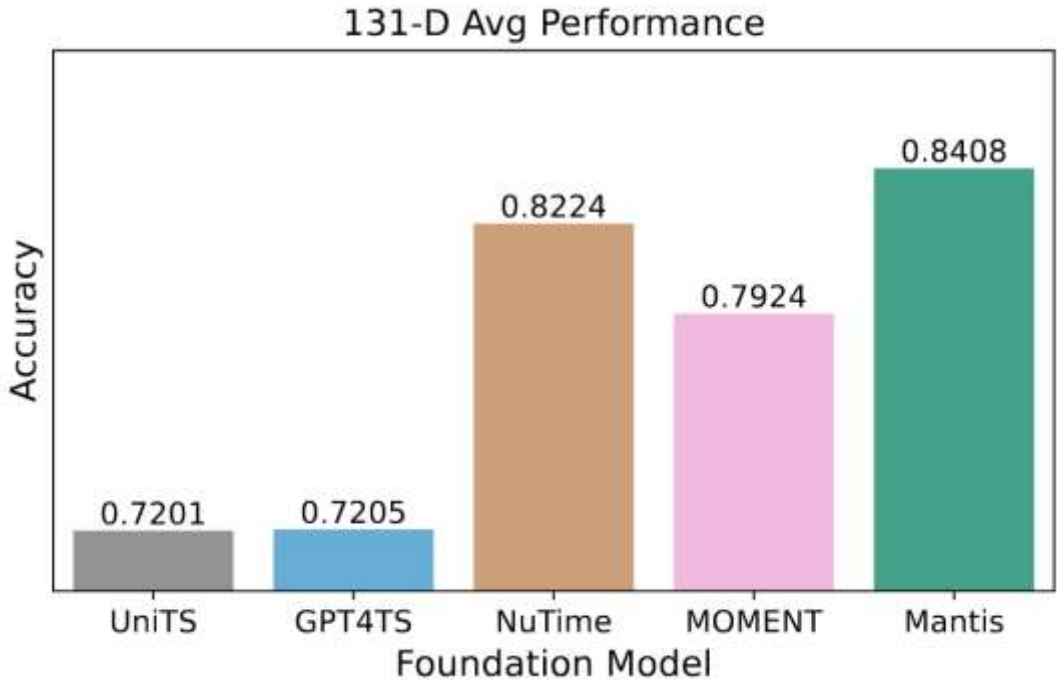
S. Xie et al. (2025). Cauker: classification time series foundation models can be pretrained on synthetic data only. *arXiv:2508.02879*.

# Fine-tuning Results



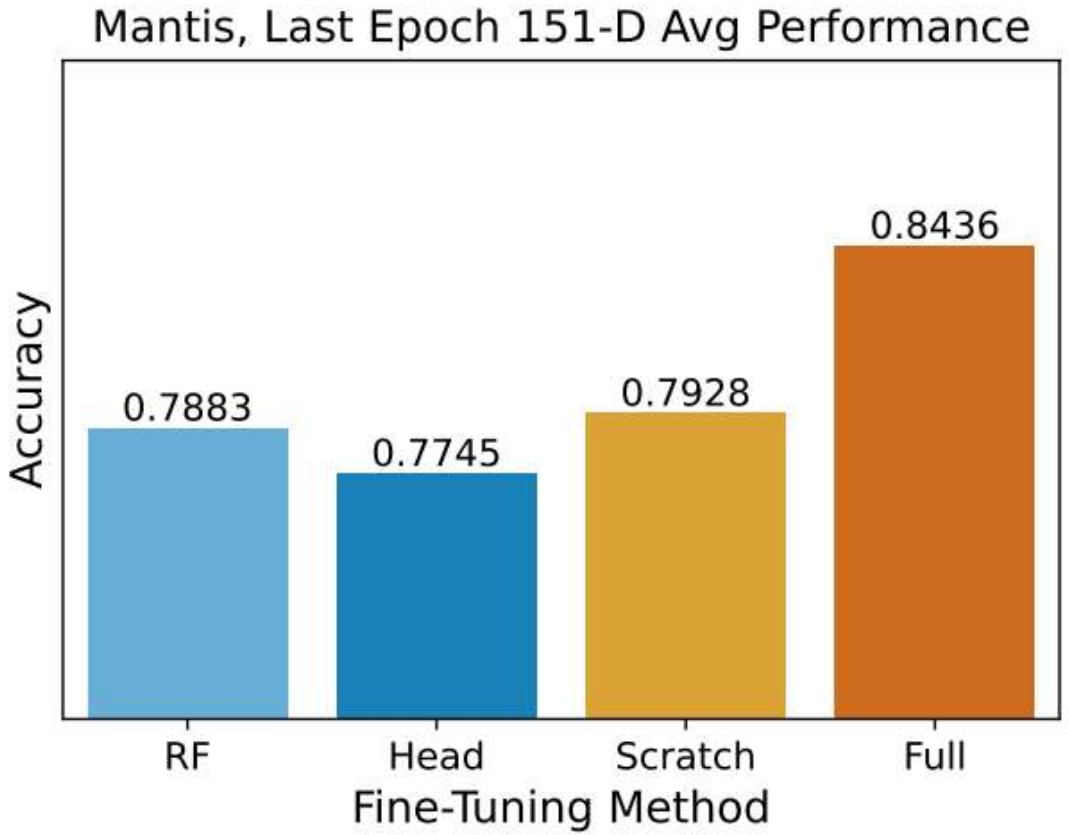

For each dataset:

- Start from the pre-trained model
- Fine-tune it on train set
- Choose best learning rate on validation
- Evaluate accuracy on test set



# Influence of Pre-training and Fine-tuning

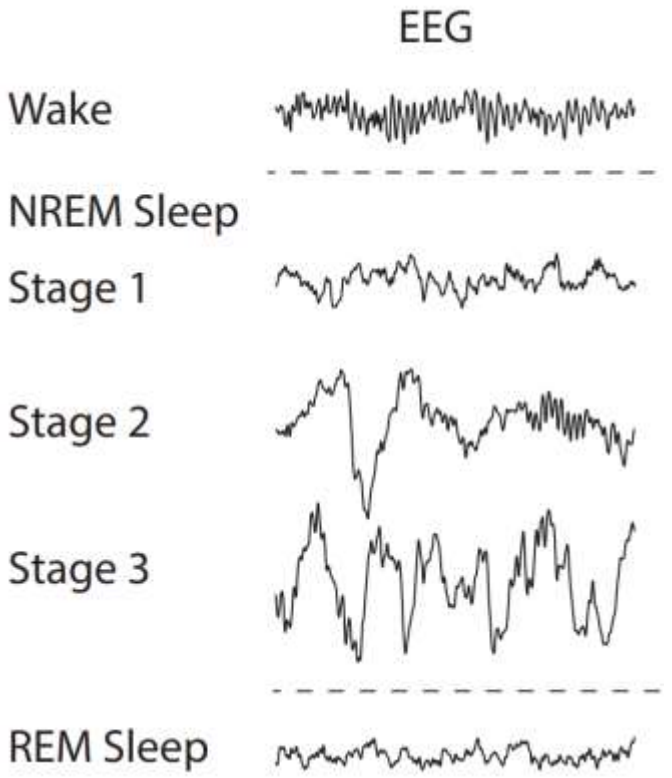
- RF: pre-trained frozen encoder + random forest for classification.
- Head: pre-trained frozen encoder + linear probing.
- Scratch: randomly initialized encoder, full fine-tuning.
- Full: pre-trained encoder, full fine-tuning.




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

# Application to EEG Sleep Staging

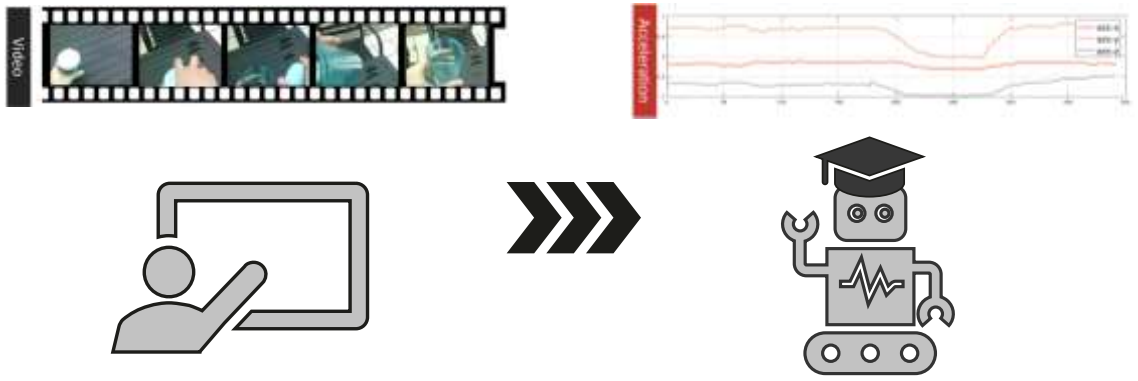
| Dataset | EEGNet | CBraMod | | Mantis | | |
|---|---|---|---|---|---|---|
| | | Random | EEG Pretrain | Random | Real Pretrain | Synth Pretrain |
| ABC | $67.94_{\pm 6.52}$ | $70.61_{\pm 3.29}$ | $74.90_{\pm 4.89}$ | $72.82_{\pm 3.89}$ | $75.50_{\pm 5.62}$ | $\mathbf{75.74_{\pm 4.32}}$ |
| CCSHS | $83.13_{\pm 0.10}$ | $87.01_{\pm 0.27}$ | $88.04_{\pm 0.59}$ | $88.55_{\pm 0.39}$ | $\mathbf{88.85_{\pm 0.48}}$ | $88.80_{\pm 0.30}$ |
| 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!

# Application to HAR

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




Video Teacher

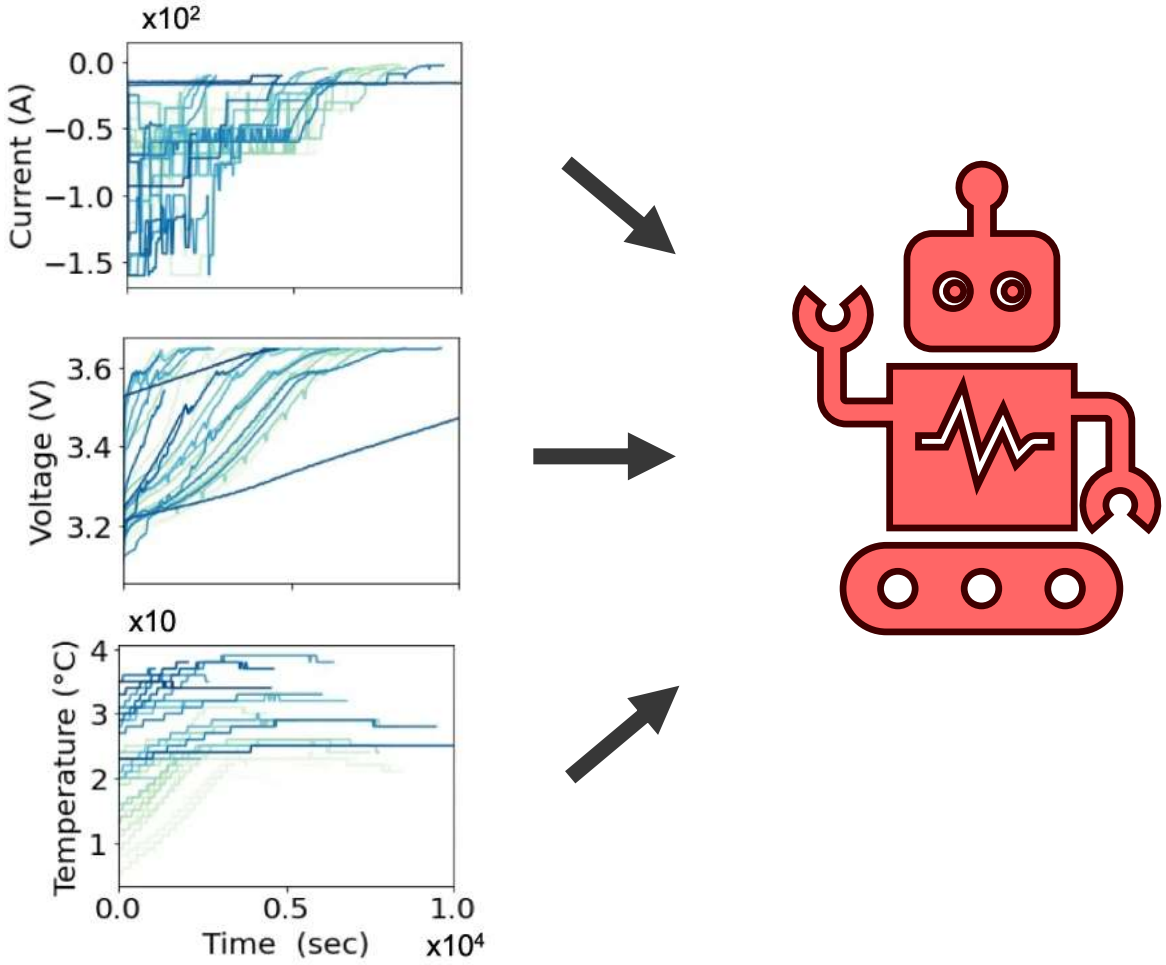
Time Series Student

| | 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 |



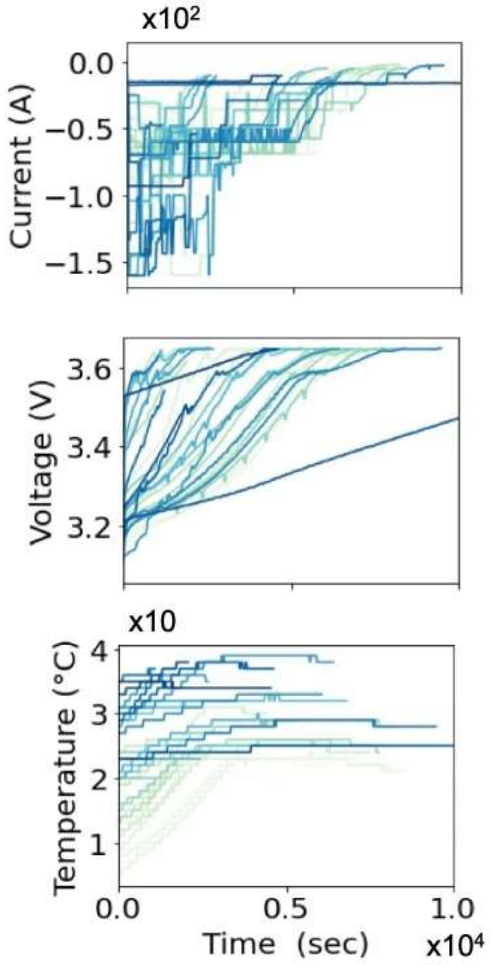
B. Chen et al. (2025). Comodo: Cross-modal video-to-imu distillation for efficient egocentric human activity recognition. *arXiv:2503.07259*.

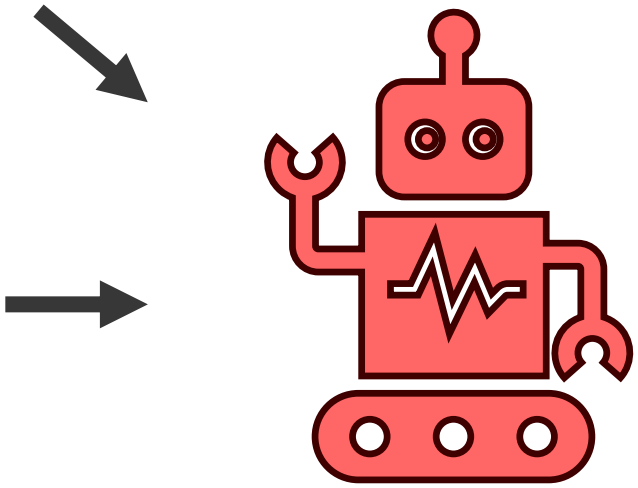
# Challenge 1: Multivariate Time Series




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

# Challenge 1: Multivariate Time Series



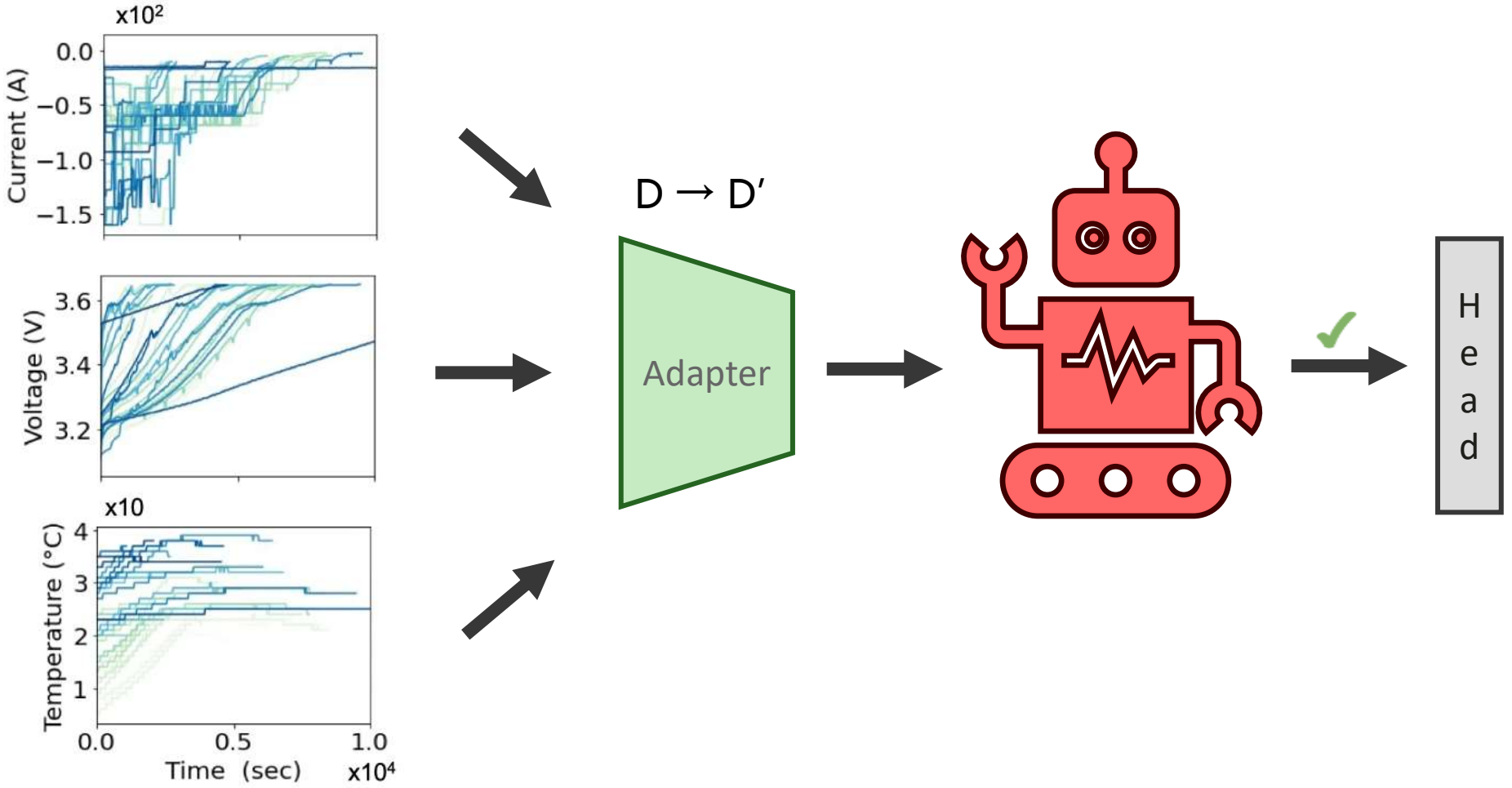



Mantis and other TSFMs are pre-trained on univariate time series data => treat channels independently.

*!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

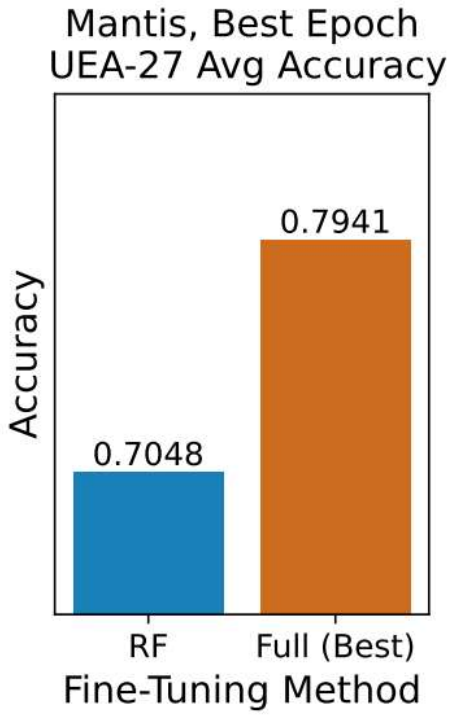
- Number of channels ≥28, full fine-tuning with batch size 256 is not possible on a single V100 without an adapter.

| | 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**$_{\pm 0.0108}$ |



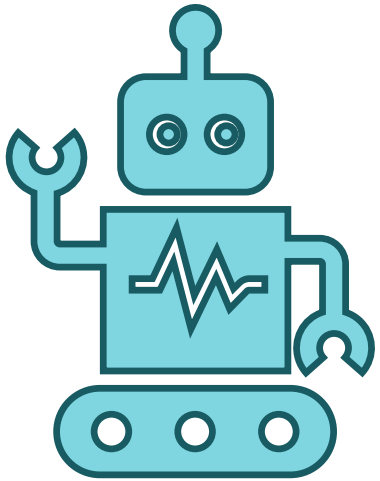
# Adapters for Multivariate Classification

- 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**$_{\pm 0.0108}$ |

# Challenge 2: Uncertainty Quantification



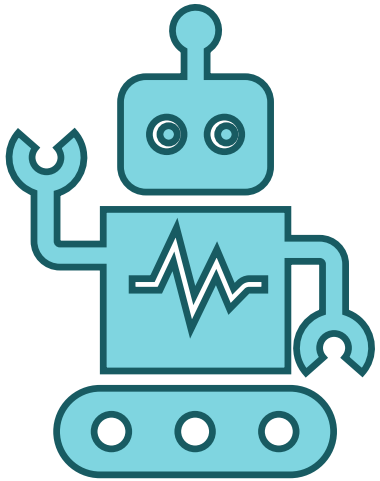
*output probs*

Test data

Are they calibrated?



# Challenge 2: Uncertainty Quantification



*output probs*

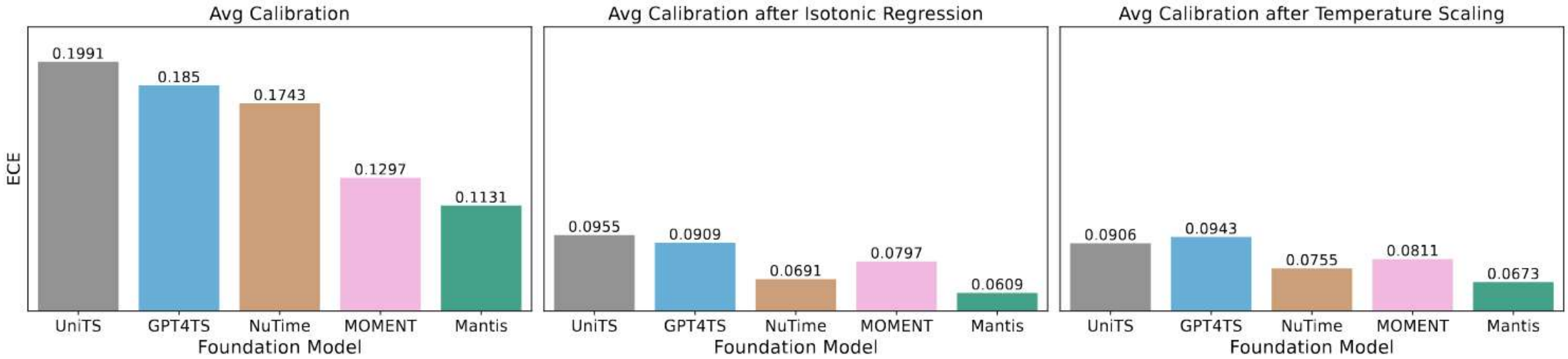
Test data

Are they calibrated?

$$\mathbb{P}(Y = \hat{y} \,|\, \mathrm{conf}(\mathbf{X}) = \alpha) = \alpha, \quad \forall \alpha \in [0, 1]$$



# Calibration Experimental Results




**Mantis is the most calibrated time series classification foundation model**
(Before and after post-hoc calibration).

# 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
```

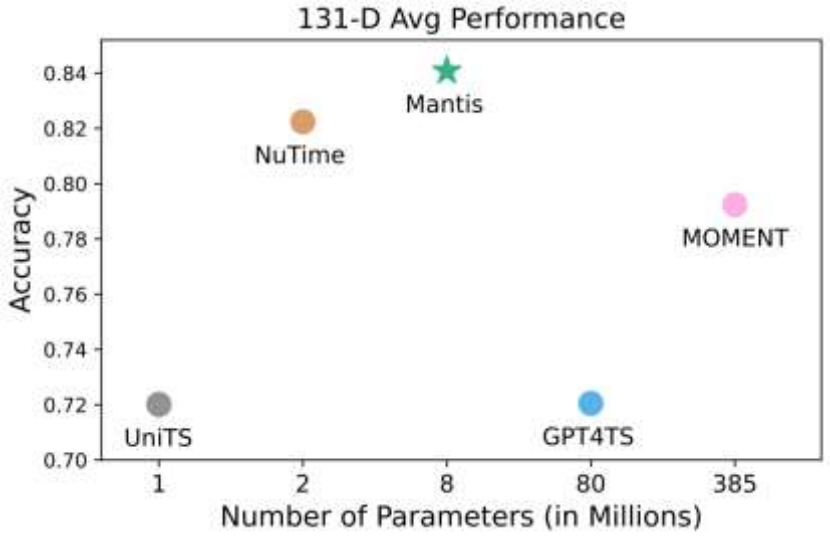
## Or directly fine-tune the model

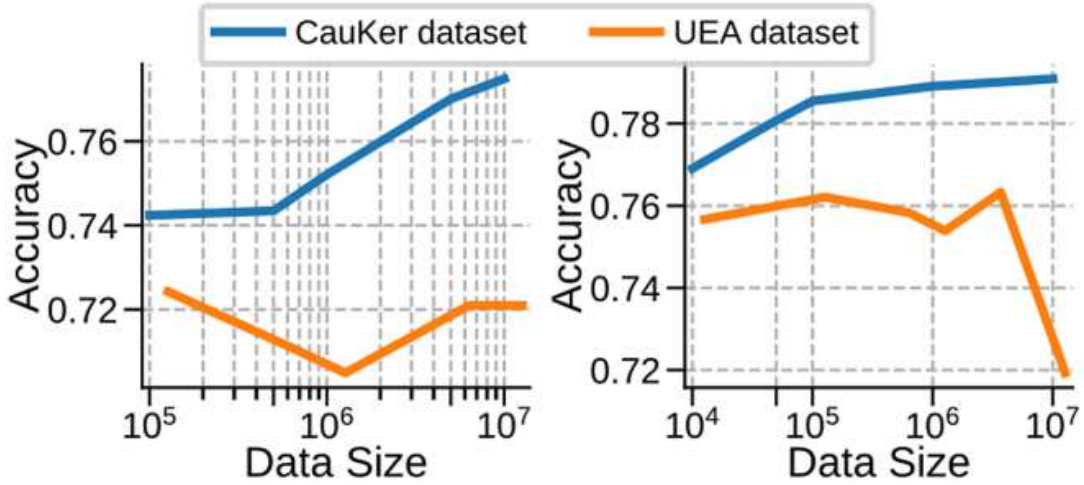
```
from mantis.trainer import MantisTrainer

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)
```



# Future Work

1. Ensure scaling law.



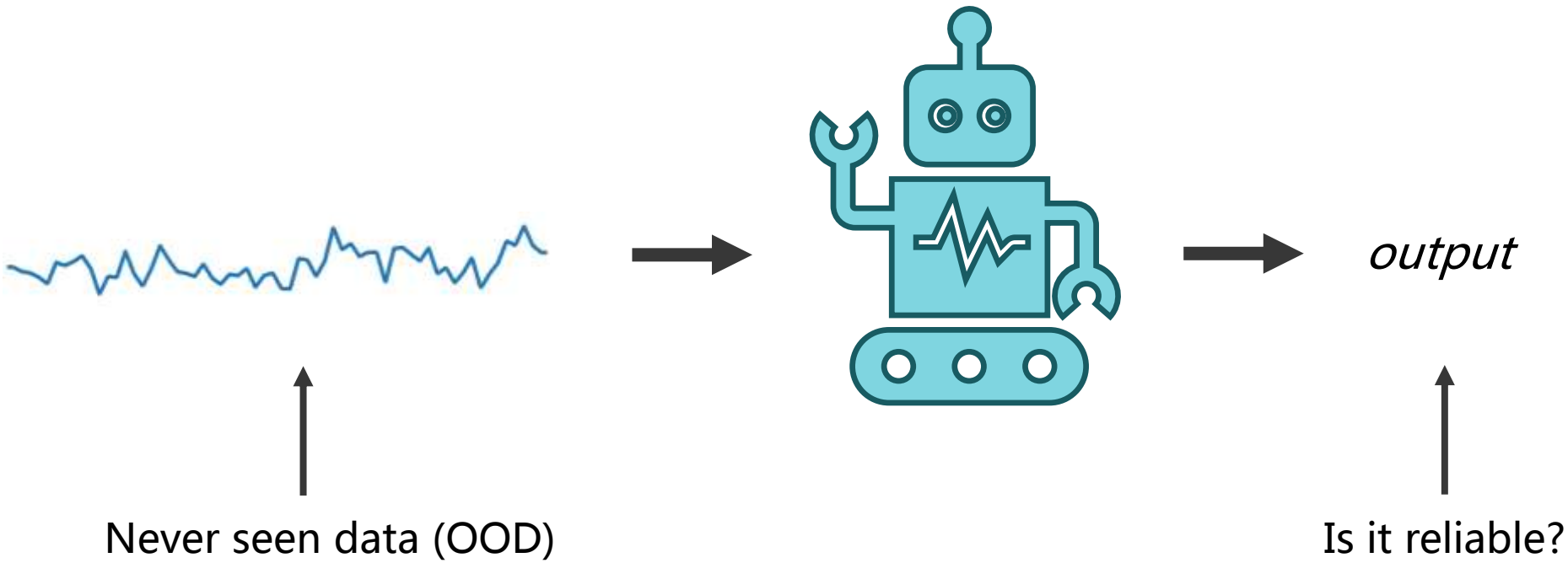




Moment Mantis

**No clear relationship between the model /data size and overall performance!**

# Future Work

2. Performance prediction.



# BERT2S NeurIPS Workshop

## Have we reached the BERT Moment?

**Location**: San Diego Convention Center, Upper Level Room 24ABC
**Date**: Sunday 7th December 2025


## Second Call for Papers

- Not peer-reviewed.
- Just for a poster presentation.
- Submission deadline: Oct 19, 2025 (11:59 pm AoE)

# Thank you for your attention!

Mantis

My LinkedIn