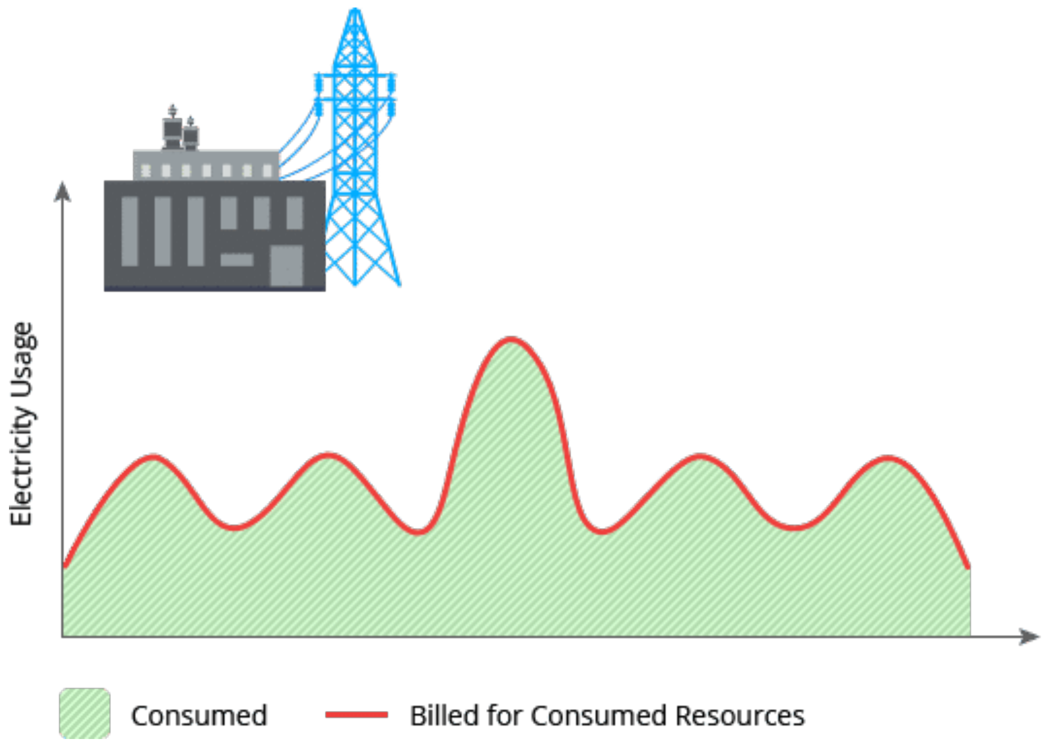
# Toward Using *Representation Learning* for Cloud Resource Usage Forecasting

**Razine Moundir Ghorab**, Thaleia Dimitra Doudali

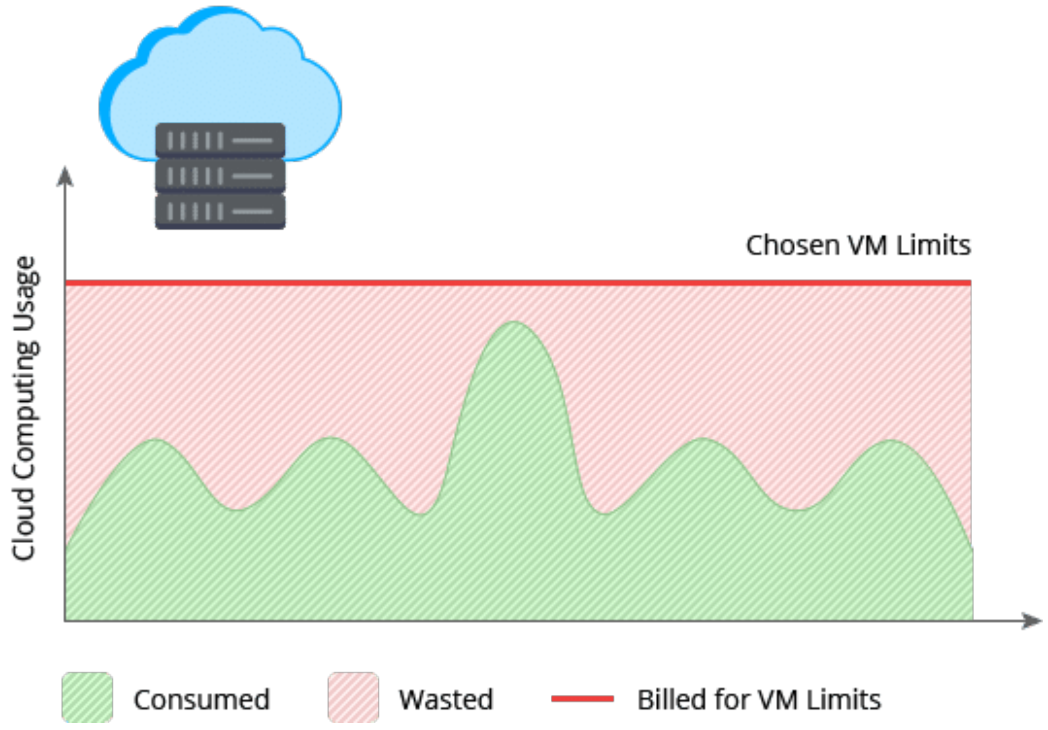
IMDEA Software Institute, Madrid, Spain

June 3rd 2024 @ AI4Sys workshop

# Problem: Low Resource Efficiency in the Cloud




VS




Electricity: we pay for what we use.

Hardware Resources: we pay for **more** than we use.

[Source] https://www.docktera.com/blog/deceptive-cloud-efficiency-do-you-really-pay-as-you-use/

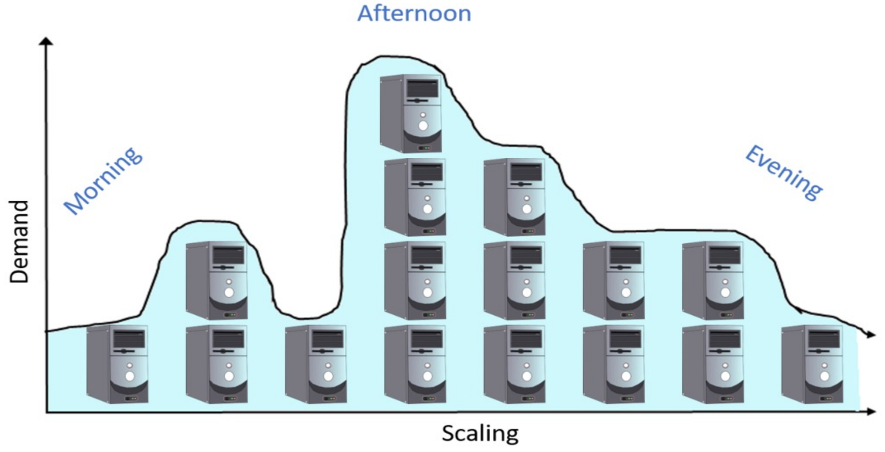
What causes this significant gap between consumed and paid resources?

- **Users**: over-estimate resource needs, ask for more than they end up using.
- **Cloud Providers:** allocate more resources to satisfy peak load and guarantee Service-Level-Agreements (SLAs).
- **Cloud Management System:** suboptimal resource management decisions.

# Solution: Resource Management Methods

## 1. Resource Autoscaling

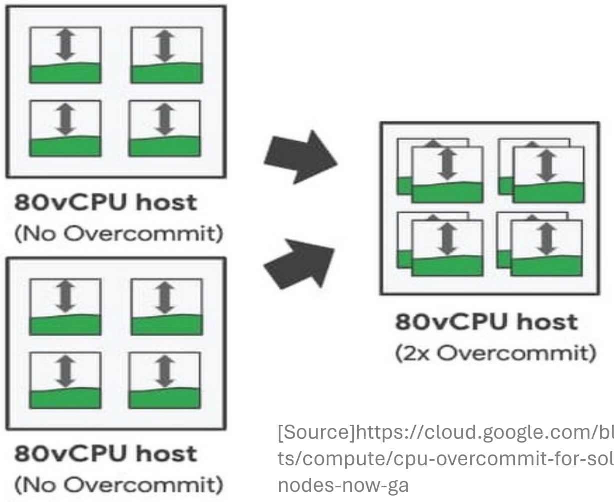
- Automatically adjust the amount of allocated resources based on demand.




[Source] https://www.linkedin.com/pulse/autoscaling-cloud-part-1-anshul-jindal

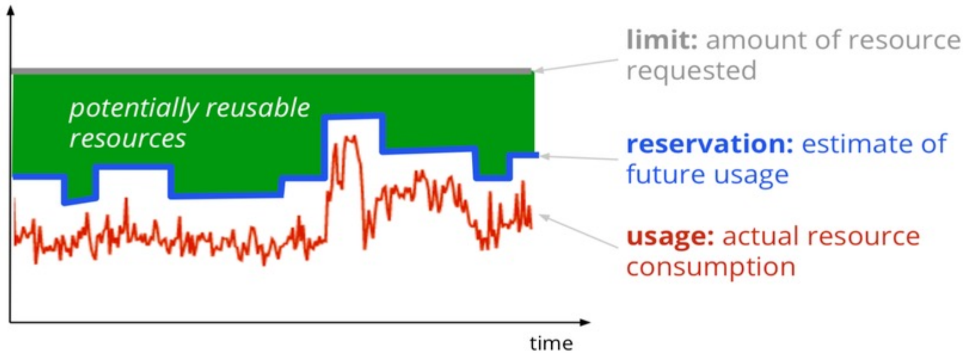
## 2. Resource Overcommitment

- Allocate more virtualized resources than physically available.




[Source]https://cloud.google.com/blog/products/compute/cpu-overcommit-for-sole-tenant-nodes-now-ga

What is the **key** for these methods to work?

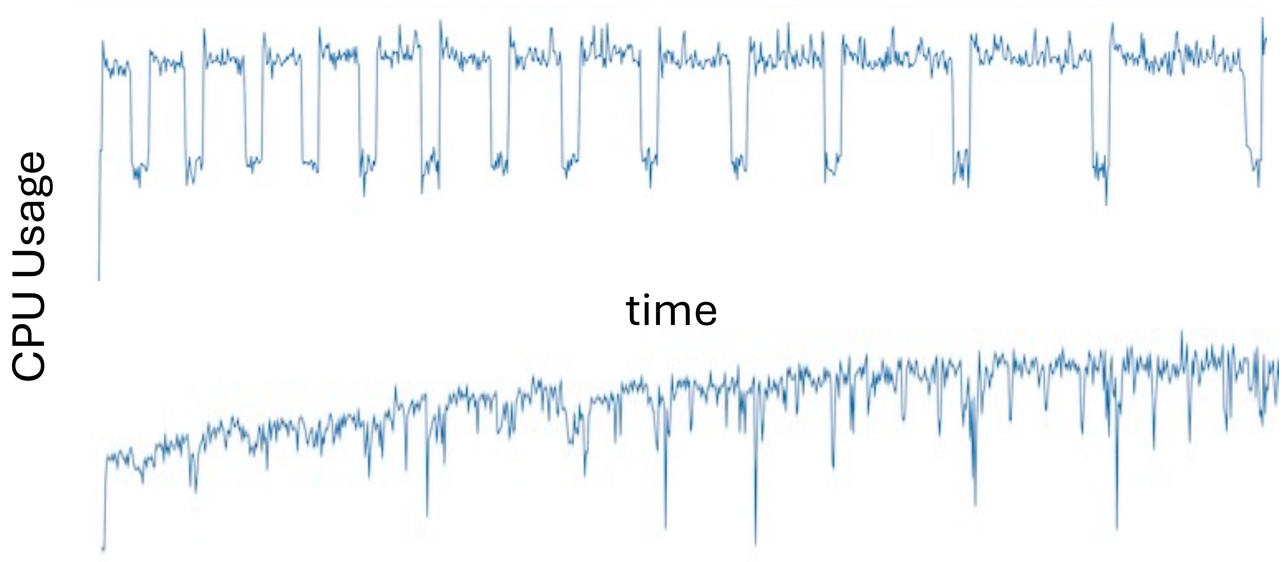


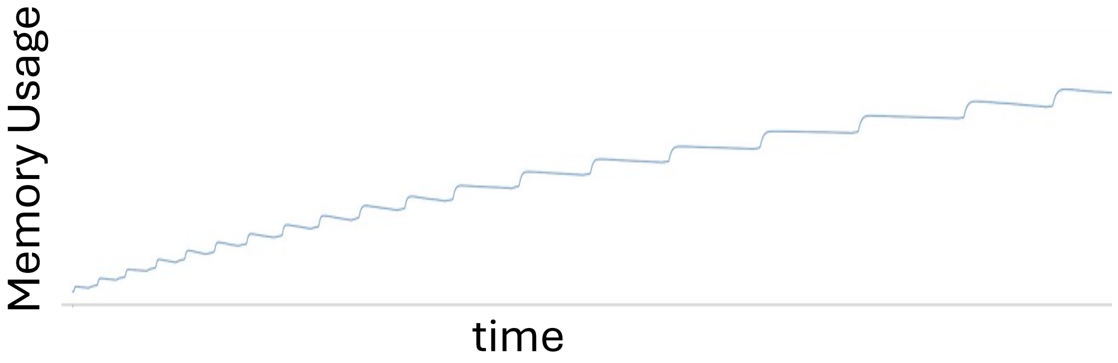

[Source] https://www.laitimes.com/en/article/30aoz_3h08q.html

The Key for Effective Resource Management is Accurate Forecasting of Resource Usage.

# Challenge: Accurate Forecasting is Hard to Achieve

**Reason 1:** Wide variety of observed patterns.

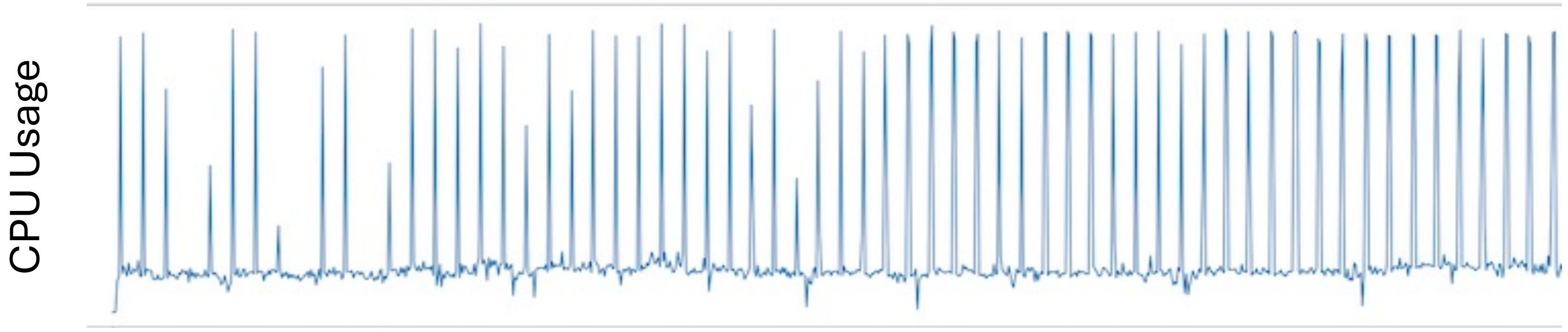



Resource usage patterns:

- application-level: wide variety.
- CPU: periodic cycles, memory: stable.

**Reason 2:** Unseen and unpredictable patterns.

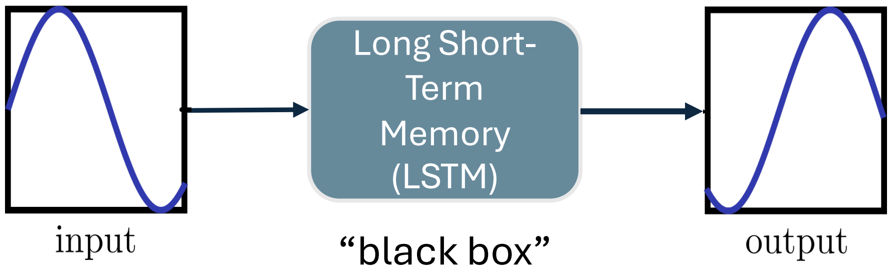



Emergence of new patterns:

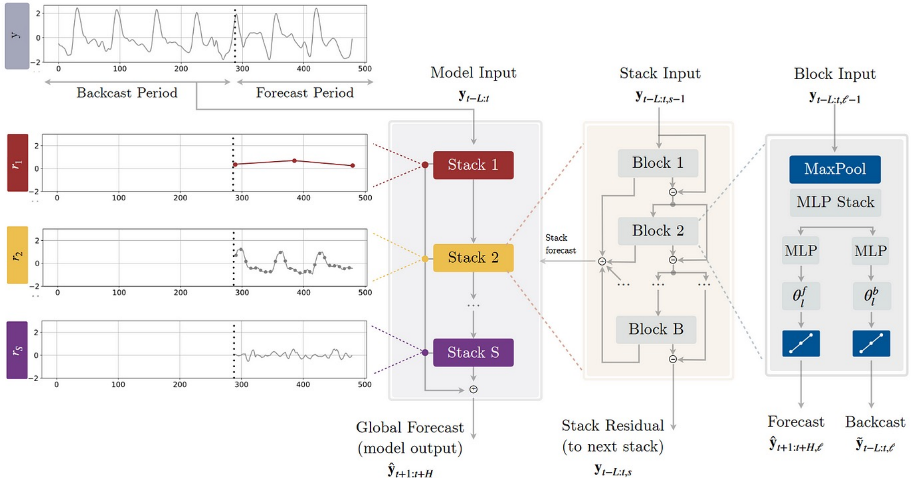
- New users / workloads.
- Same users with new behaviors.

# Current Machine Learning Methods for Time Series Forecasting

## LSTMs (Long Short-Term Memory Neural Networks)




## N-Hits (Neural Hierarchical Interpolation)




[Source] https://towardsdatascience.com/xai-for-forecasting-basis-expansion-17a16655b6e4

### Limitations

- High runtime overheads.
- High complexity.
- Hard to integrate in production.
- Low explainability.
- Limited predictive capabilities.

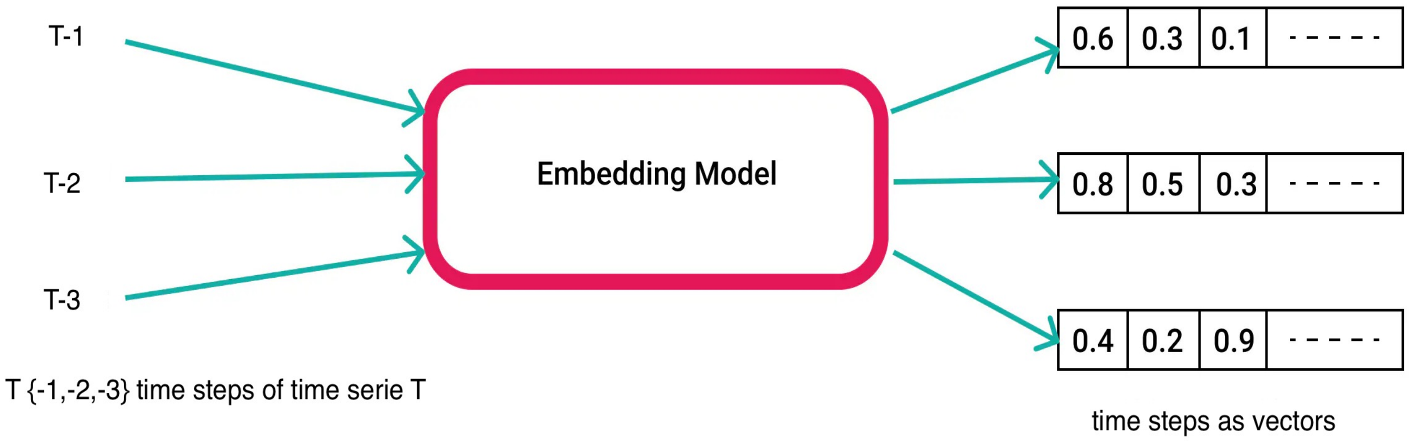
### Our Goal

Develop a predictor that is:

- Highly accurate.
- Fast, has low overheads.
- Practical to use.
- Explainable.

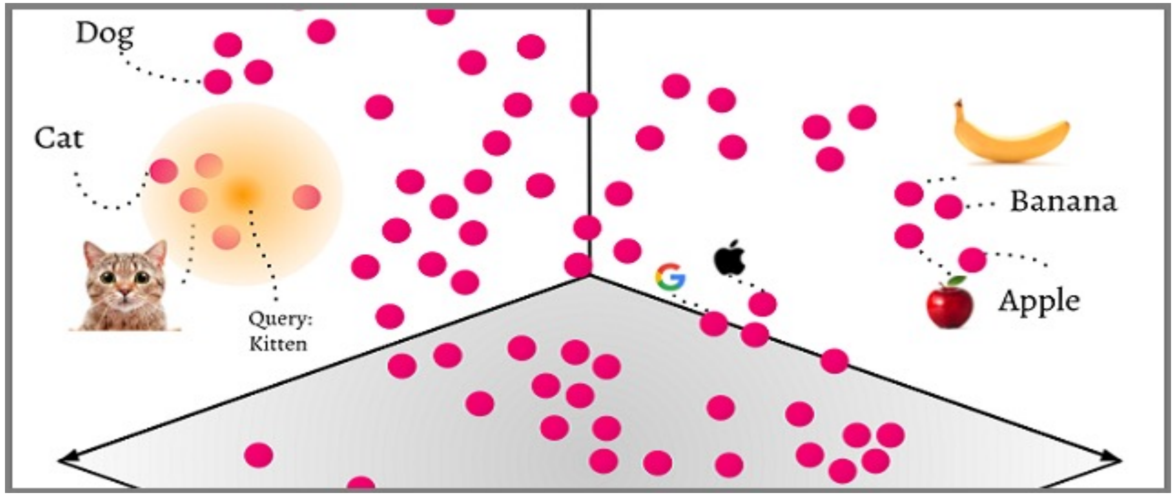
# Inspired by Representation Learning

Represent time series as embeddings in the vector space.




Can effectively capture context, patterns and the ordering of the time steps.

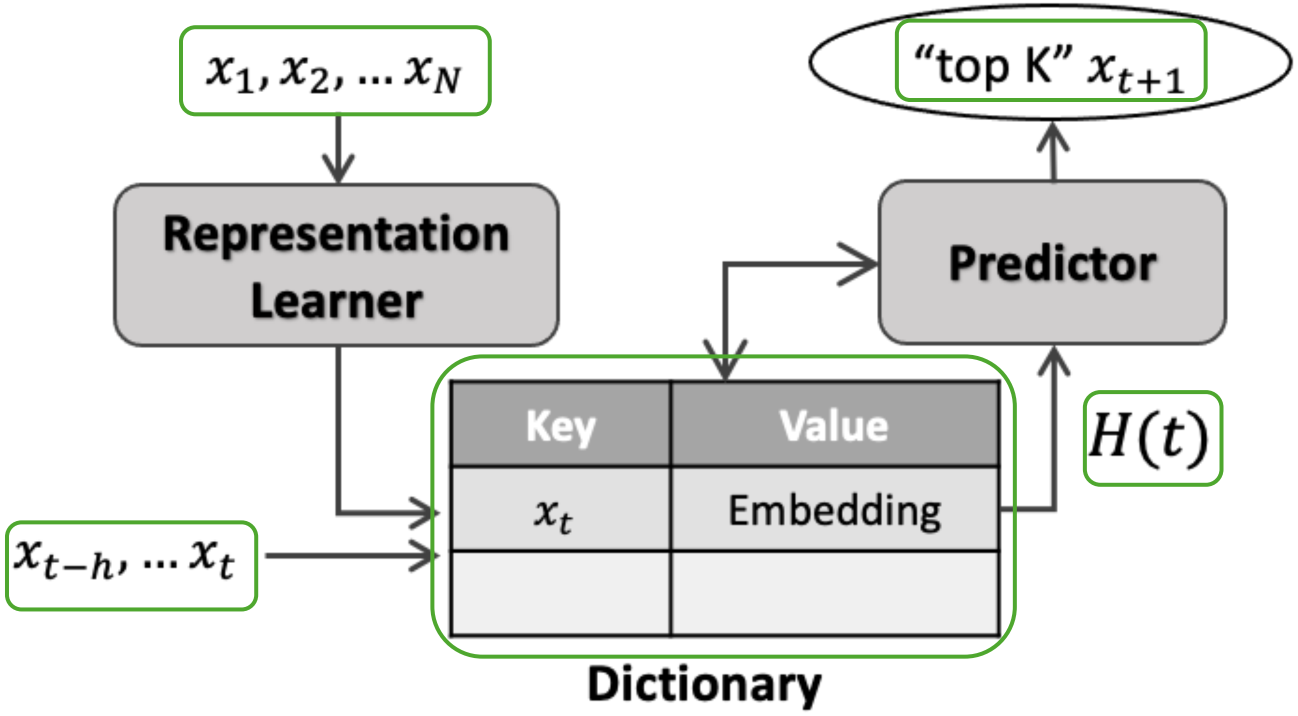
Explore spatial proximity to find similarities and make predictions.




[Source] https://www.mongodb.com/fr-fr/resources/basics/vector-databases

To find similarities in the vector space, we can use the K Nearest Neighbors (KNN) algorithm.

# Proposed System Overview (MTEBWY = May The Embedding Be With You)




*MTEBWY System Prototype*

## Representation Learner

- Creates a dictionary of embeddings using word2vec "**embedding model**" for each value $x_t$ of the timeseries $x_{1,} x_{2,} .. x_N$
- The **dictionary** size is the number of unique integer values in each time serie.

## Predictor

1. Takes as input a window of history $x_{t-h,} .. x_t$
2. Maps the $x_t$ values to their embeddings in the dictionary.
3. Calculates the weighted average $H(t)$ of these embeddings.
4. Finds the K nearest neighbors (KNN) of $H(t)$ using cosine similarity.
5. Uses the reverse mapping in the dictionary to return a set **of K predictions** for time $t + 1$.

# Experimental Setup

## Evaluation Baselines

- **Persistent Forecast:** Predicts next value as current value $y(t+1) = y(t)$.
- **LSTM** (Long Short-Term Memory network).
- **N-HiTS** (Neural Hierarchical Interpolation)

*Automatic hyperparameter fine-tuning via AutoLSTM and AutoNHITS from the framework NeuralForecast (Nixtla).*

## Dataset

Multiple time series of CPU usage from the DeathStarBench benchmark suite.

## Evaluation Metrics

**Is the ground truth among the set of K predictions (KNN) at each time step?**

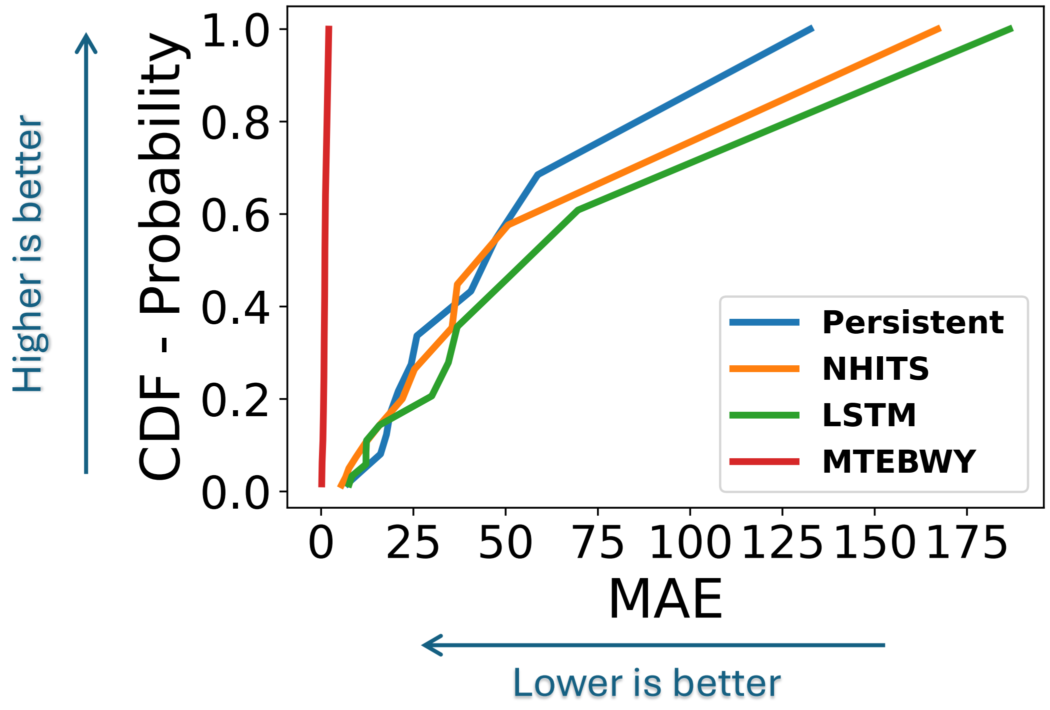
- **Yes:** top-K accuracy.

  Percentage of times when the ground truth is one of the K nearest neighbors.
- **No:** Mean Absolute Error (MAE).

  Average difference between the nearest neighbor and the ground truth.

# Key Results- Comparison with Baselines

## 1. Prediction Accuracy.




1- MTEBWY achieves extremely low MAE **close to zero** across all time series (vertical line).

**Why?** Because the **set of K predictions** either contains the ground truth or very similar values.

**This is the power of embeddings!**

2- All other methods perform significantly worse.

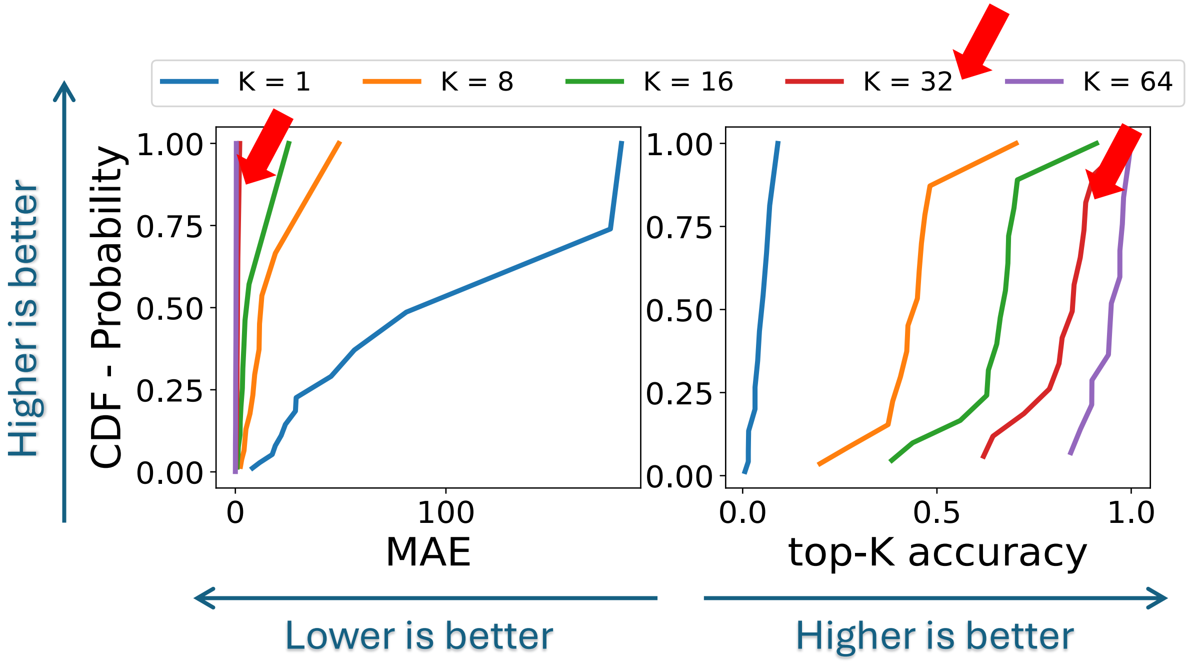
## 2. Training Overheads.

- MTEBWY: training in **1 second**.
- AutoLSTM and AutoN-HiTS: **6 hours.** (NVIDIA A100 GPU with 40 GB Memory)

**Takeaway:** MTEBWY it is **highly** accurate with **trivial** training overheads.

# Sensitivity Analysis of K

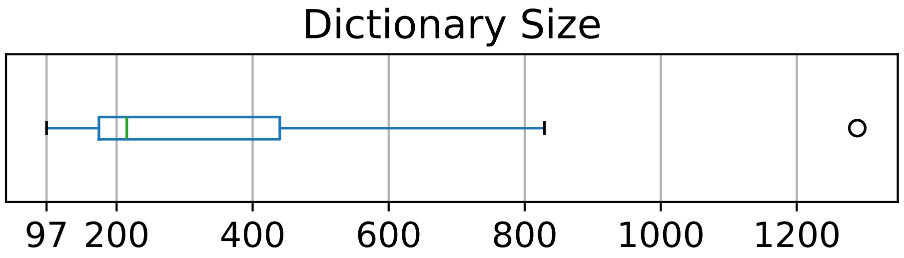
MTEBWY produces a set of K predictions at every time step.




**K = 32** provides **both** low MAE and high top-K accuracy

How does K compare to the dictionary size?

The dictionary size is the number of unique integer values in a time series. There is 1 dictionary per time series.




Although the dictionary size is large, 200 on average, **K=32** is robust across all time series.

# Summary and Conclusions

Github Repo

## ➢ Embedding is the Future !

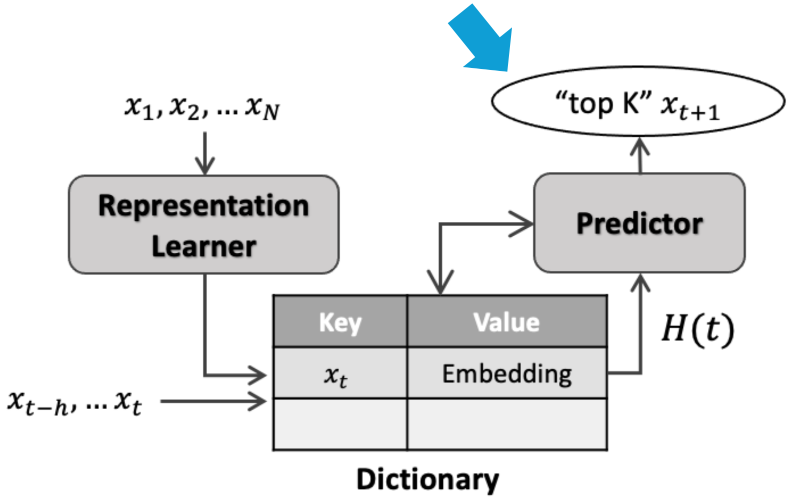
**Lessons learned:**

1. **Representation learning** unlocks very low error and negligible overheads.
2. The key to success is capturing the **spatial proximity** of the embeddings.

## Future Work:

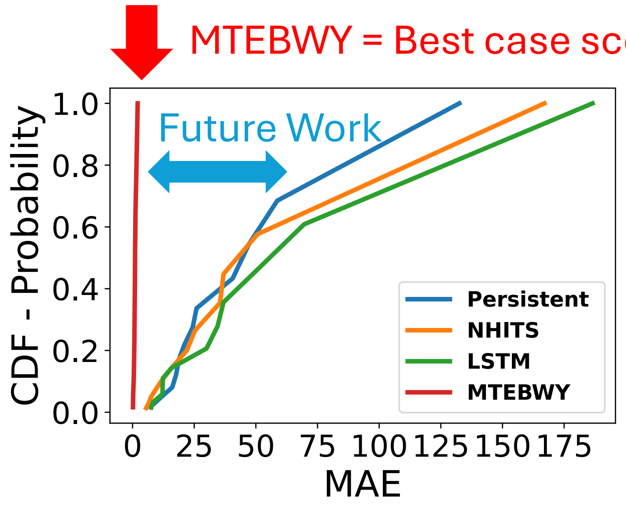
Complete the system prototype:

- ➢ Deliver 1 prediction, not K. (use classifier)
- ➢ How much will MAE increase?
- ➢ Evaluate against more datasets.
- ➢ Generalize to have 1 dictionary for more than 1 time series.



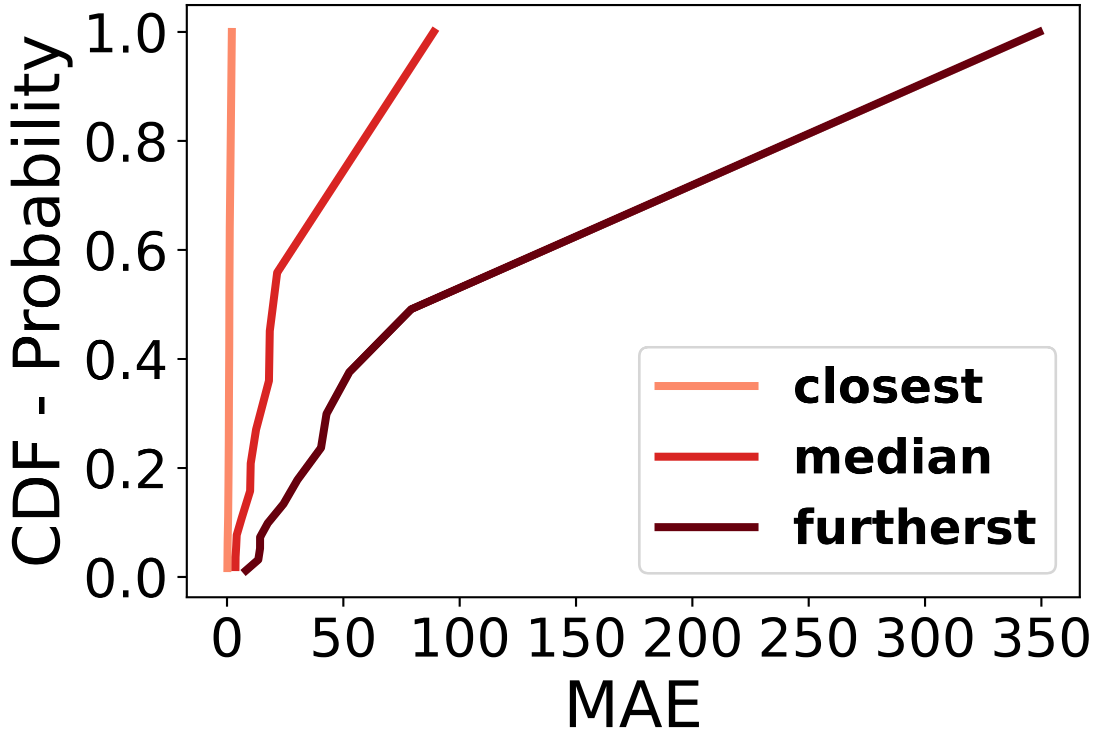

*MTEBWY System Prototype*

MTEBWY = Best case scenario



# Backup Slides

# Quality of top Ks



# Other Sensitivity Analysis