Multivariate Time Series Clustering based on Transformer Model

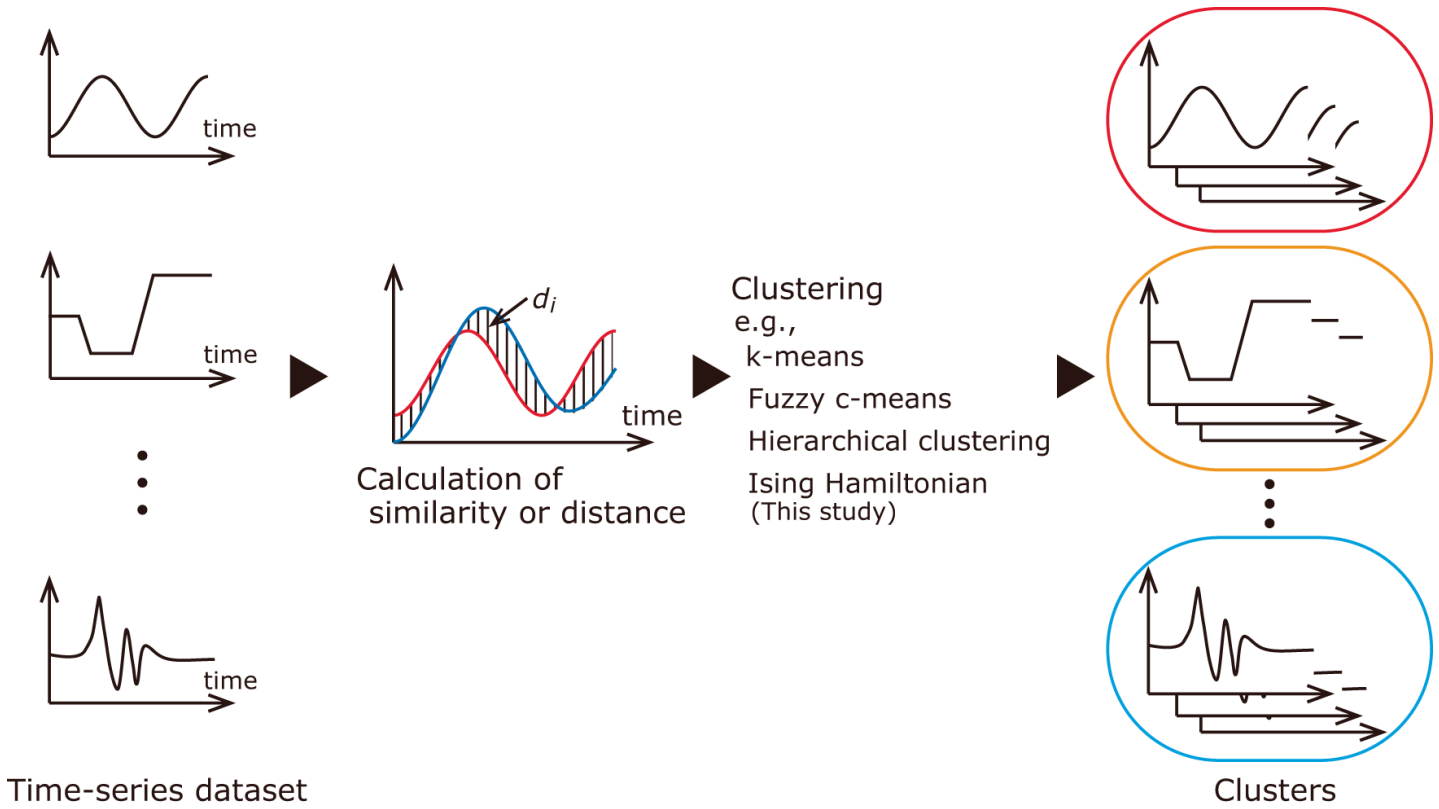
Jiazhou (Joe) Liang

UNIVERSITY OF TORONTO

DEFY GRAVITY

Multivariate Time Series Clustering

- **Time series:** a sequential data on the range of times
- **Multivariate:** when more one variables available for each samples
- **Clustering:** unsupervised learning, clustering a given time-series dataset without labels into groups



UNIVERSITY OF TORONTO

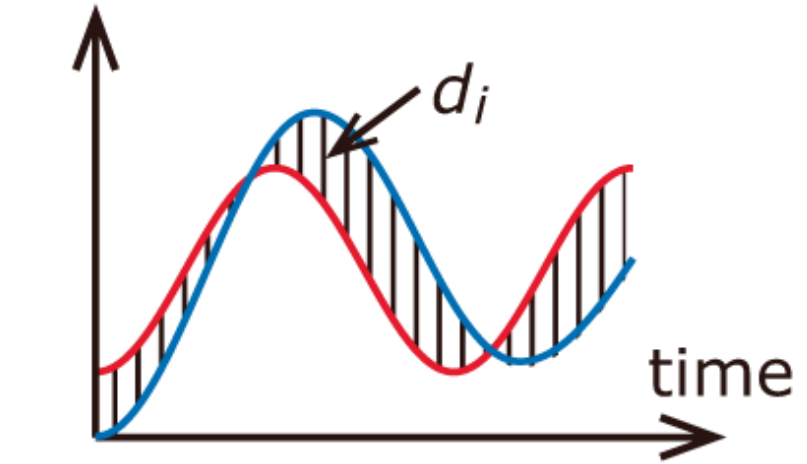
Two Approaches

Temporal-Proximity Based Clustering

- Comparing the raw time series

Problem:

- Noise in time series may reduce the performance
- Comparing different lengths is a problem
- Suffer in capturing multivariate relationship



Calculation of similarity or distance

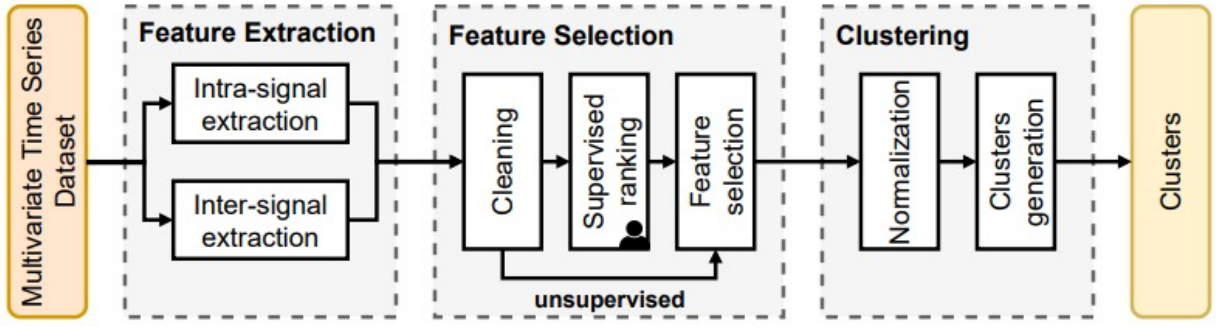
Representation-based Clustering

Extracting features as representation: seasonality pattern, Fourier coefficient, statistic, dimension reduction

Problem:

Required large amount of preprocessing work by human

Some methods might not suitable for multivariate data



Time2Feat

(Bonifati, 2022)

And more recently Deep-learning Based:
Autoencoder, seq2seq model

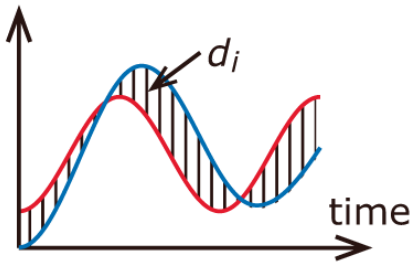
Research Question: Transformer-Based Clustering

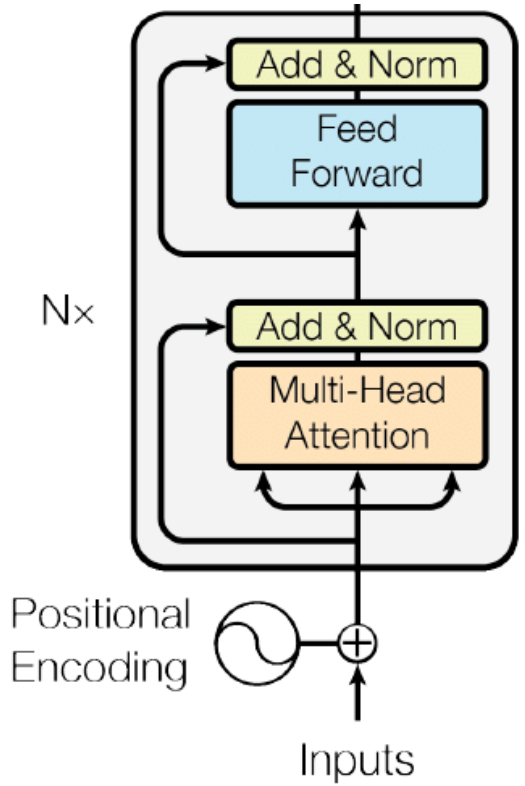
- Multi-head attention mechanism has potential to learn the relationship between each timestamps and variables
- More efficient compared to seq2seq model

Novel clustering framework using a transformer encoder

Deriving representation of samples using it latent space

Applied k-mean clustering on the representation of sample





UNIVERSITY OF TORONTO

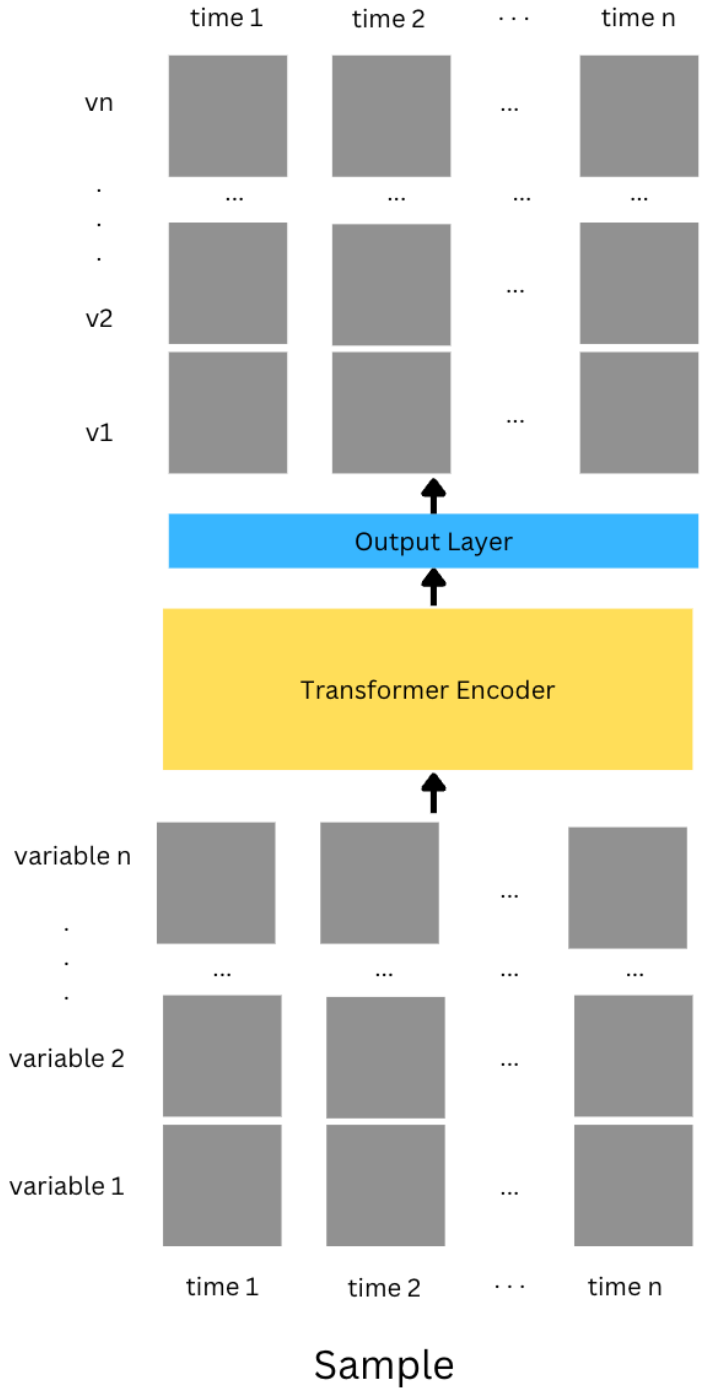
Self-supervised Learning

There is no true label for clustering dataset, how can we be training the transformer?

Self-supervised approach

- Using the raw multivariate time series samples as input
- Training the transform encoder to reconstruct the input time series
- OR: Masking the sequences (Zerveas, 2020)

The goal is let the encoder learn the representation



Auto-Encoder

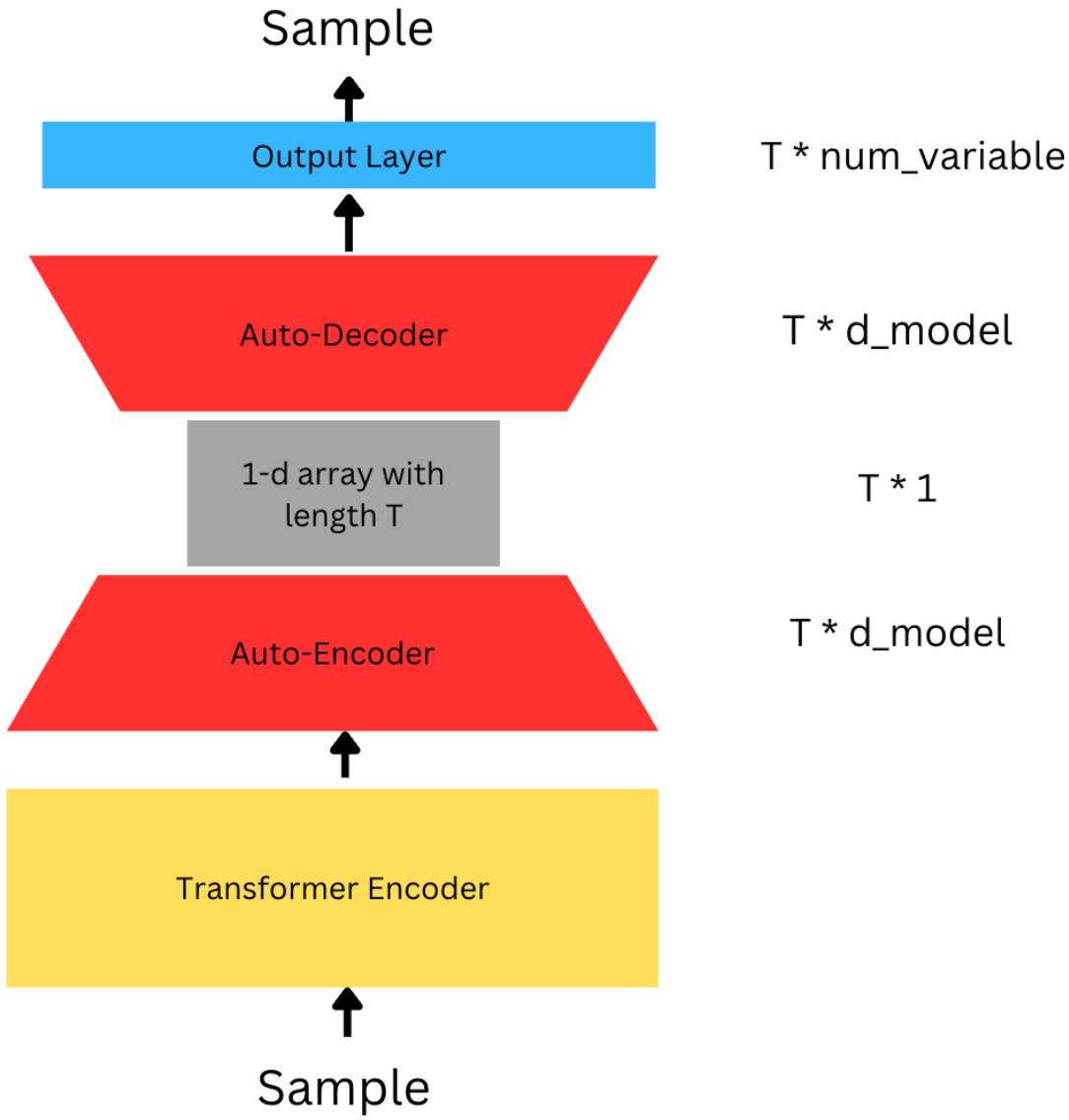
The latent space of transformer encoder usually is a high dimension array

- Defeat our purpose of not clustering a raw multivariate time series

Possible solution:

- Aggregation by mean
- Dimension Reduction

Using a small auto-encoder network to compress latent space into a single dimension array



Learning Representation for Clustering

Does not show improvement in original experiments

Possible problem:

- The representation learned from the reconstruction series might not be suitable for clustering purposes.
- We need to find a way to guide the model to learn a representation that can be beneficial for clustering.
- Original K-Means algorithm is not differentiable

Spectral Relaxation for K-means Clustering

Zha et al. purposed a reformulation of K-mean problem as a trace maximization (Zha, 2001)

Given a data matrix **H** that contains **N** samples

- Sum of Squares (Euclidean distance) between samples and its cluster's centroid:

$$Tr(\boldsymbol{H}^{\boldsymbol{T}}\boldsymbol{H}) - Tr(\boldsymbol{F}^{\boldsymbol{T}}\boldsymbol{H}^{\boldsymbol{T}}\boldsymbol{H}\boldsymbol{F})$$

where **F** *is an orthonormal matrix with dimension* **N* k** *as the indicating which clusters of each samples belongs to*

Minimized the Sum of Square Cost

$$\max_{\boldsymbol{F}} Tr(\boldsymbol{F}^{\boldsymbol{T}}\boldsymbol{H}^{\boldsymbol{T}}\boldsymbol{H}\boldsymbol{F}),\ s.t.\ \boldsymbol{F}^{\boldsymbol{T}}\boldsymbol{F} = \boldsymbol{I}$$

- Same as the maximization problem of the Trace of Matrix $\mathbf{F^T H^T HF}$,
  - where F indicate which cluster each sample belongs to

- Can be solved in **Closed form solution**
  - as a SVD problem by Ky Fan Theorem
  - Top k eigenvalues of the **H** as F, clusters each samples belong to

In The Model

$$\mathcal{L}_{K-means} = Tr(\boldsymbol{H}^{\boldsymbol{T}}\boldsymbol{H}) - Tr(\boldsymbol{F}^{\boldsymbol{T}}\boldsymbol{H}^{\boldsymbol{T}}\boldsymbol{H}\boldsymbol{F})$$

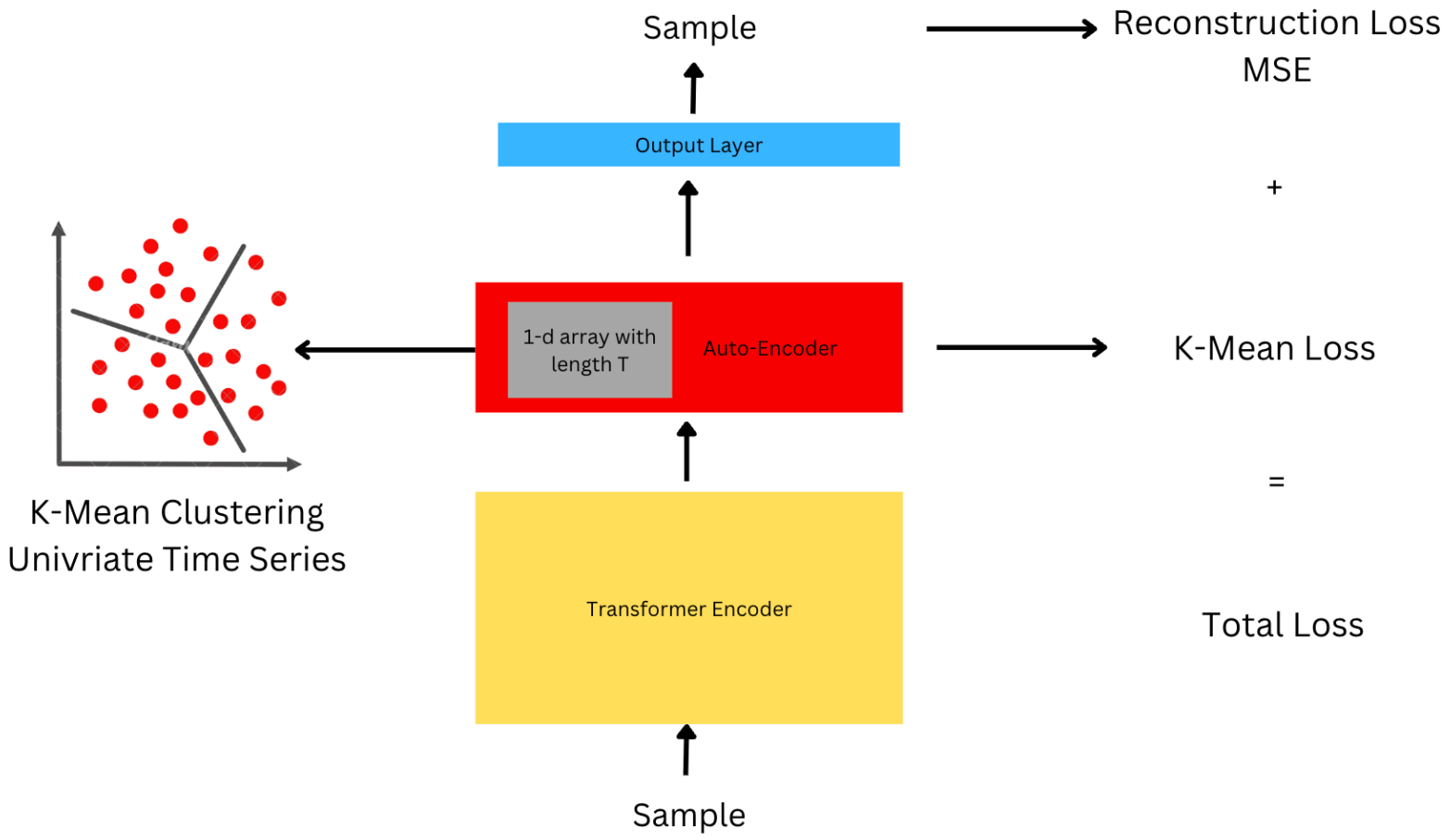
H = The compressed latent space after Auto-Encoder

- Length of time series **T** * number of samples in current batch **N_batch**
- Updated through Stochastic Gradient Descent

F = Random Initialization

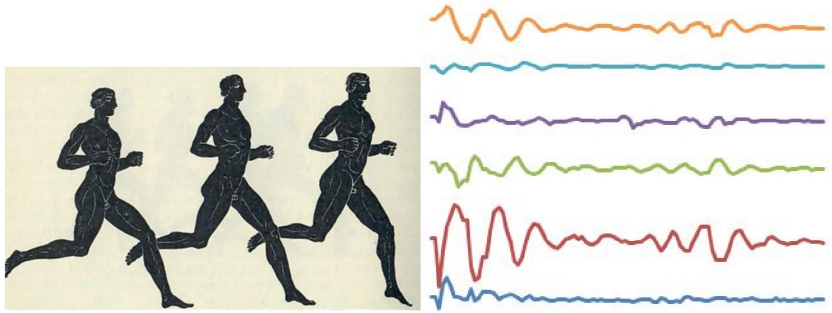
- **N_batch** * number of clusters **k**
- Updated periodically using SVD as mentioned before

By Combining Two Loss



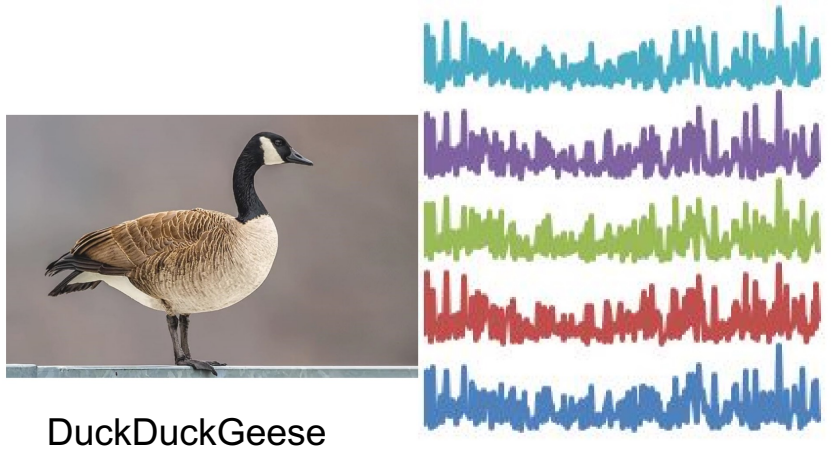
Experiment

- To test the performance of this novel framework on different types of Multivariate Time Series Data



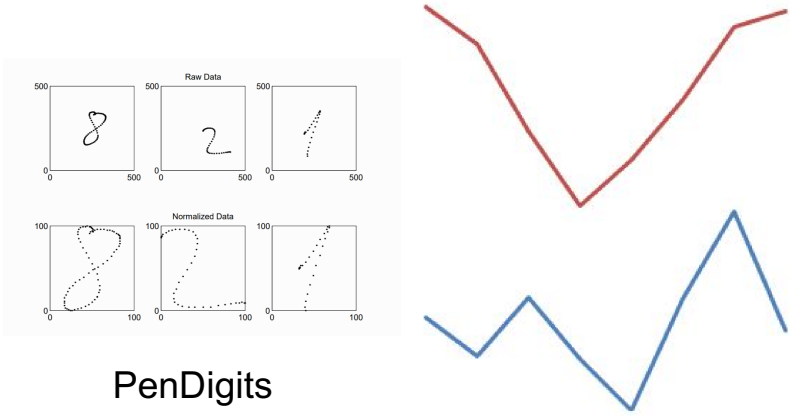
BasicMotion

- From UEA Multivariate Time Series Classification Archive (Bagnall, 2018)



DuckDuckGeese

- Why using classification?
  - true label and number of clusters available
  - helps in evaluate the performance



PenDigits

UNIVERSITY OF TORONTO

Datasets

General:

- BasicMotions: 60 samples, 6 dimensions, 100 timestamps, 4 classes
- Epilepsy: 137 samples, 3 dimension, 206 timestamps, 4 classes

Short Sequences, Lower dimension, Large Sample size (Easier case)

- PenDigits: 7494 samples, 2 dimension, 8 timestamps, 10 classes

Long Sequences

- EigenWorms: 128 samples, 6 dimension, 17984 timestamps, 5 classes

High Dimension

- NATOPS: 180 samples, 24 dimension, 52 timestamps, 6 classes
- DuckDuckGeese: 60 samples, 1345 dimension, 270 timestamps, 5 classes

Evaluation

- Self-supervised training the model using the train set
- Using the trained model to obtain representation of the test set

Rand Index: measure of the similarity between two data clustering

- True label vs Cluster Label
- (number of agreeing pairs) / (number of pairs)

$$RI = \frac{TP + TN}{TP + FP + FN + TN}$$

Clustering Time: only for reference

Rand Index

Baseline: K-mean Clustering

- using raw time series
- Perform well in the easier case
- While having a bad performance in higher dimensional cases

Transformer-Based:

- shows an improvement in many dataset
- significant in the extremely high dimension dataset

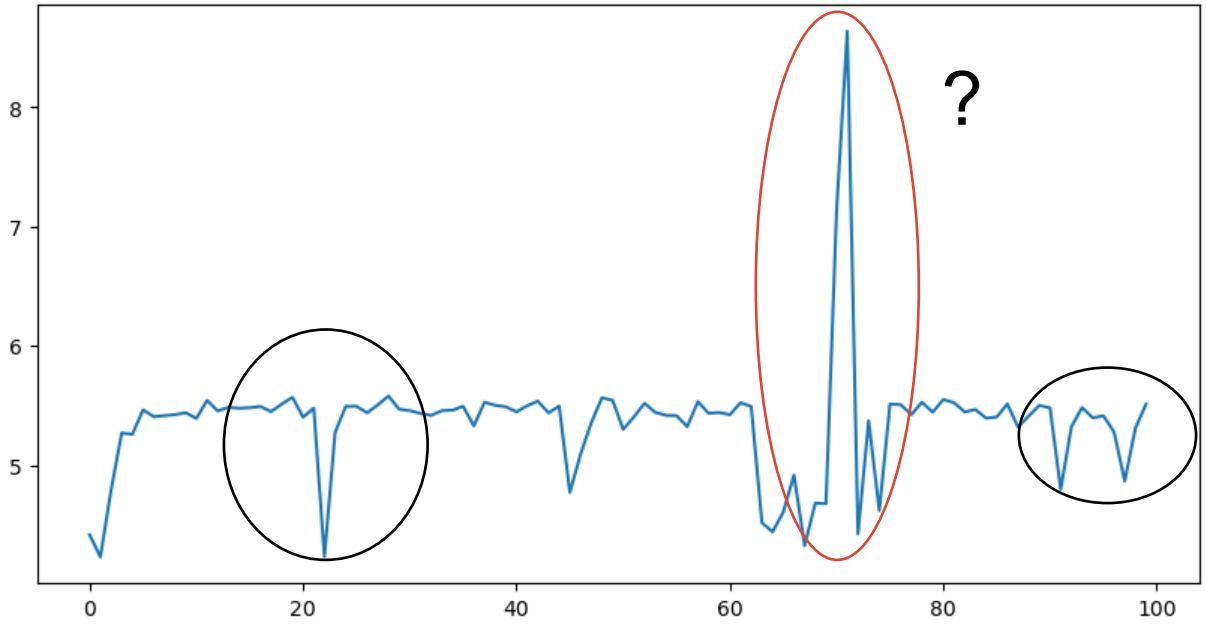
Dataset	Baseline	Transformer	Notes
BasicMotions	0.4192	**0.5270**	General
Epilepsy	0.6123	**0.6351**	General
PenDigits	**0.9140**	0.8184	Easy
EigenWorms	0.4912	**0.6860**	Long Sequence
NATOPS	**0.8002**	0.7577	High Dimension
DuckDuckGeese	0.2832	**0.5494**	Higher Dimension

Time (In Seconds)

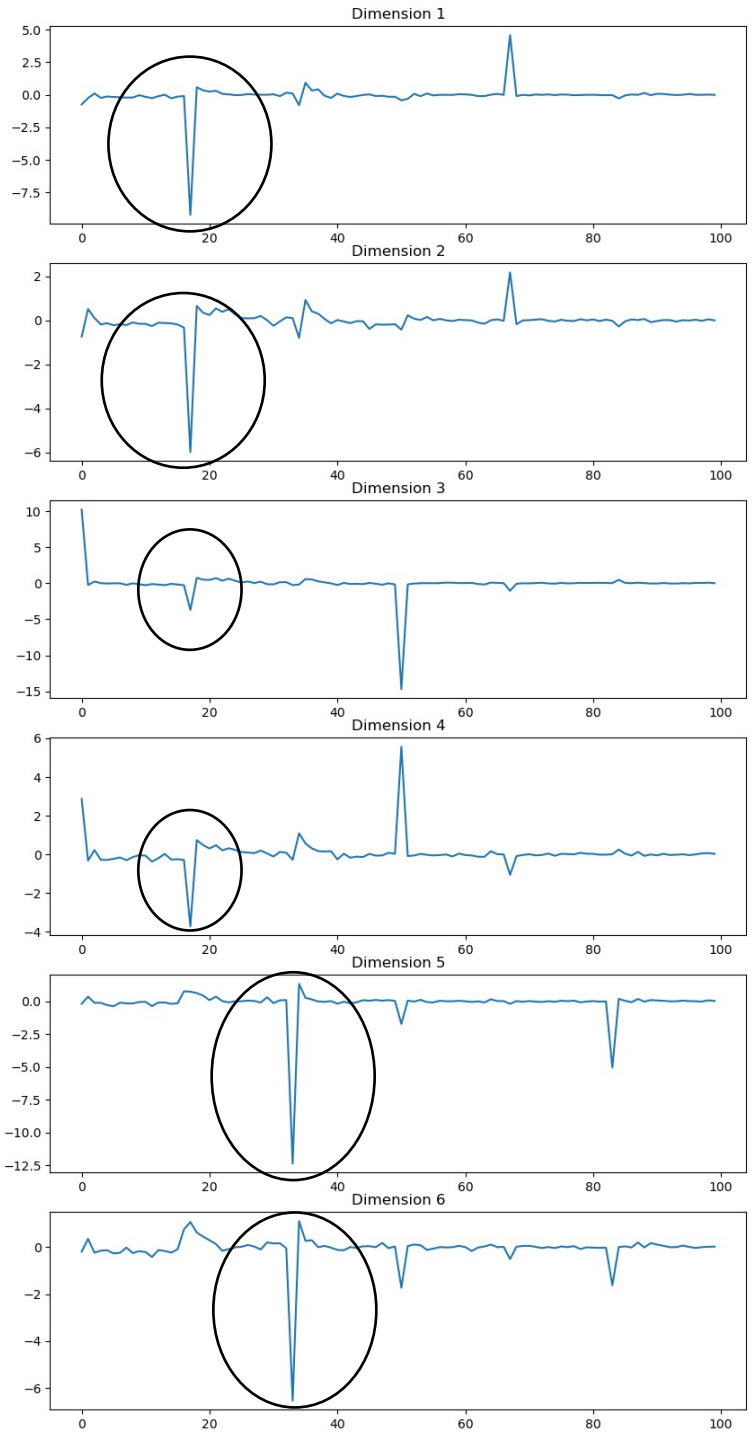
- Not considering the model training time
- Better performances in high dimension or large samples sizes
- since we convert it into univariate time series data

Dataset	Baseline	Transformer	Notes
BasicMotions	0.0152	0.0129	General
Epilepsy	0.0806	0.0536	General
PenDigits	3.7740	**0.7338**	Easy
EigenWorms	0.9896	0.0483	Long Sequence
NATOPS	0.0660	0.0313	High Dimension
DuckDuckGeese	1.3116	**0.0161**	Ex High Dimension

Can We Understand the Latent Space?



A sample's latent space in Basic Motion dataset



Original Space

UNIVERSITY OF TORONTO

Discussion

- We can see some improvement using Transformer Based Clustering, still need comparison between other representation-based clustering
- Sacrificing time (we need to train the model) compared to K-Mean Clustering
- Fine-Tuning the hyperparameter might improve the performance
- Does the latent space interpretable?

Next Steps:

- Try Other techniques: Encoder Classification (Ma, 2019), only the important part of the time series
- Large Pre-training models
- Cutting time series into different pieces, compared by a range instead of single points