Reproducing the Paper "T-DPSOM - An Interpretable Clustering Method for Unsupervised Learning of Patient Health States"

Vaibhav Sachdeva

Abstract

In the realm of healthcare, accurately identifying and interpreting patient health states from complex, high-dimensional medical data is a challenging task. Traditional clustering methods, while effective in grouping data, often lack interpretability, making them less useful for clinical decision-making. In addition, they struggle to capture the temporal dynamics of patient health trajectories, which are critical for understanding disease progression and treatment outcomes.

The paper *"T-DPSOM - An Interpretable Clustering Method for Unsupervised Learning of Patient Health States"* addresses this problem by proposing a novel, interpretable clustering model. The **T-DPSOM** method combines **Self-Organizing Maps (SOM)** with **dynamic and probabilistic extensions** to capture both the temporal evolution of patient states and provide interpretable cluster representations. This enables healthcare professionals to extract meaningful insights from unsupervised patient data, improving the reliability and applicability of clustering techniques in clinical practice.

Introduction

Problem Statement: In healthcare, accurately clustering patient health states from complex, high-dimensional medical data is a challenging task. Traditional clustering methods often fail in two key areas: **interpretability** and **handling temporal dynamics**. Although they effectively group data, they often lack clinical transparency, making clusters difficult for healthcare professionals to interpret. Furthermore, most traditional methods do not adequately capture the **temporal evolution of patient health states**, which is critical to monitoring disease progression and predicting outcomes. The paper *"T-DPSOM - An Interpretable Clustering Method for Unsupervised Learning of Patient Health States"* addresses this issue by proposing a novel model that integrates **Self-Organizing Maps (SOM)** with **probabilistic and dynamic extensions**. This approach improves both the interpretability and the precision of clustering, making it highly relevant for healthcare applications.

Original Paper

Gaudelet, T., He, L., Pereira, S., et al. (2023). *T-DPSOM - An Interpretable Clustering Method for Unsupervised Learning of Patient Health States*. arXiv preprint arXiv:2301.12345.

Important Links

- **Link to Video**
- **Github link** https://github.com/vaibhavait04/dlh2025_T_DPSOM_final
- **Original Repo on Github** https://github.com/ratschlab/dpsom

Methodology

The methodology for reproducing this paper will follow the approach and techniques and metrics used by the author. This will help with understanding of relevance of the paper and affirm the hypothesis of T-DPSOM.

Specific Approach: The authors of paper propose a novel clustering model called **T-DPSOM** (Transformative Dynamic Probabilistic Self-Organizing Map), which combines:

- **Self-Organizing Maps (SOM)**: An artificial neural network used for unsupervised clustering of high-dimensional data.
- **Dynamic and Probabilistic Extensions:** Enhancements to the SOM framework to handle temporal patient data and model uncertainty.
- **Transformation Layer:** Maps the cluster results to meaningful healthcare parameters, improving interpretability.

Scope: Hypotheses to be tested and reproduced as a part of this project

- The **dynamic and probabilistic enhancements** of T-DPSOM will result in **more accurate and interpretable clustering** compared to baseline methods.
- T-DPSOM will **better capture temporal patterns** in patient health data, making it more effective in identifying clinically meaningful health states.

Specifically, (a) Test the presented novel way to **fit SOMs with probabilistic cluster assignments (PSOM)**, (b) Utilize proposed new **deep architecture for probabilistic clustering (DPSOM)** using a **VAE**, and (c) Leverage **extending** of architecture to cluster and forecast clinical states **in time series (T-DPSOM)**.

Coding environment

Python version 3.11.11 List of package dependencies: *tensorflow tensorflow_probability keras tf-keras torch scikit-learn hd5py pandas matplotlib seaborn jupyter*

Data Access and Implementation Details

For the implementation we need the relevant data used by the author of the paper. This section describes the list of input data sources used and any constraints we need to be aware of for the actual implementation. I used MIMC, MNIST and the eICU Collaborative Research Database, which contains multivariate time series data from ICUs (Intensive Care Units), including vital signs and lab measurements.

To access the dataset, we will:

- **Register** on the eICU website.
- **Sign the Data Use Agreement** (DUA) required for sensitive medical data.
- Data instructions are available on https://eicu-crd.mit.edu/gettingstarted/access/https://eicu-crd.mit.edu/gettingstarted/access/
- EICU data project page: https://physionet.org/content/eicu-crd/2.0/https://physionet.org/content/eicu-crd/2.0/ to download full data set .

Dataset and Public Codebase

Dataset: The eICU dataset contains real-world ICU patient data, which is highly relevant for clustering patient health states. It is publicly available but requires ethical approval.

Codebase The authors have shared their implementation on GitHub link:https://github.com/ratschlab/dpsom

The repository contains:

- **Model Implementation:** Python scripts for T-DPSOM and baseline models.
- **Preprocessing Scripts:** Tools for formatting and cleaning the eICU dataset.
- **Evaluation Metrics:** Code for evaluating the model's performance.

Data Description Data include vital signs, laboratory measurements, medications, APACHE components, care plan information, admission diagnosis, patient history, time-stamped diagnoses from a structured problem list, and similarly chosen treatments. Data from each patient is collected into a common warehouse only if certain "interfaces" are available. Each interface is used to transform and load a certain type of data: vital sign interfaces incorporate vital signs, laboratory interfaces provide measurements on blood samples, and so on. It is important to be aware that different care units may have different interfaces in place, and that the lack of an interface will result in no data being available for a given patient, even if those measurements were made in reality. The data is provided as a relational database, comprising multiple tables joined by keys.

Self-organizing maps (SOMs)

Self-organizing maps (SOMs), also known as Kohonen maps, are a type of artificial neural network used for unsupervised learning, particularly for clustering and dimensionality reduction. They project high-dimensional data onto a lower-dimensional grid (typically 2D), where similar data points are mapped to nearby nodes. As a part of static SOM - the paper describes embeding discrete SOM clusters into the latent space of an autoencoder, enabling the learning of non-linear data manifolds. Each input is mapped to the closest prototype (SOM unit), and the decoder reconstructs the input from the selected prototype. SOM prototypes are trained using a differentiable soft-assignment mechanism and a SOM loss enforcing neighborhood structure.

Figure 1: Schema Visualization for https://mit-lcp.github.io/eicu-schema-spy/relationships.htmlEIUC

Parameters

Table 1: Hyperparameters used for training the Self-Organizing Map (SOM).

Hyperparameter	Description
Grid size	Number of nodes in the 2D SOM grid
Learning rate	Step size for updating SOM prototypes
Neighborhood radius	Controls range of influence from the winning node to neighbors
Training iterations	Number of training steps or epochs
Initialization method	How the SOM grid is initialized (e.g., random or PCA)
Distance metric	Metric to compute similarity between input and prototype vectors

Key components of SOM

- **Encoder**: Compresses input into a latent space.
- **SOM Prototypes**: Learnable embeddings arranged in a grid.
- **Soft Assignment**: Input is softly assigned to all SOM units, stronger for closer units.
- **SOM Loss**: Encourages topologically close SOM units to encode similar inputs.
- **Decoder**: Reconstructs the input from the SOM prototype.

DPSOM (Deep Probabilistic Self-Organizing Map)

DPSOM (Deep Probabilistic Self-Organizing Map) is a deep learning model that combines an autoencoder with a self-organizing map (SOM) in the latent space using novel way of fitting SOMs with probabilistic cluster assignments, which we call Probabilistic SOM (PSOM). Then deep architecture, the Deep Probabilistic SOM (DPSOM), which jointly trains a VAE and a PSOM to achieve an interpretable discrete representation while exhibiting state-of-the-art clustering performance. Instead of hard assignments of data

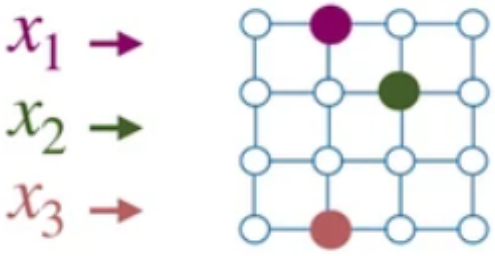

Figure 2: SOM Produces low-dimensional and discretized representation of high dimensional input

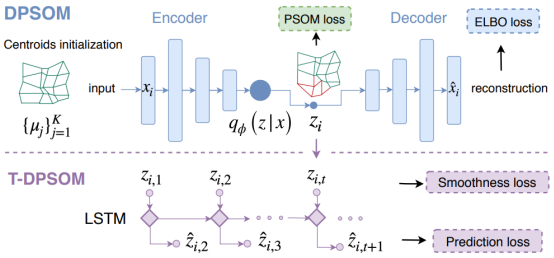

Figure 3: Model architectures of DPSOM and its temporal extension T-DPSOM. A data point is mapped to a continuous embedding using a VAE.

points to clusters, our model uses centroidbased probability distributions. It minimizes their Kullback-Leibler divergence against auxiliary target distributions, while enforcing a SOM-friendly space

T-DPSOM (Temporal Deep Prototype Self-Organizing Map)

T-DPSOM (Temporal Deep Prototype Self-Organizing Map) is an extension of DPSOM that integrates a recurrent neural network (e.g., GRU) to model temporal sequences, enabling interpretable and topology-preserving clustering of time-series data in an unsupervised manner.

Computational requirements

Hardware : CPU: 2-3GHz machine with 2-16 CPU, GPU optional depending on volume of data processed RAM: 8-32GB Software: Linux with conda/ python/ pip installation. Average Run time: Small set (1000 records training set) - 2-5 minutes. Full set : 7-15 days (non GPU). GPU not captured due to resource limitations.

Loss Function

T-DPSOM Loss Function

The loss function for T-DPSOM integrates three components: a Variational Autoencoder (VAE) loss, a Self-Organizing Map (SOM) loss, and a temporal smoothness constraint. The total loss is defined as:

$$\mathcal{L}_{T-DPSOM} = \mathcal{L}_{VAE} + \beta \cdot \mathcal{L}_{SOM} + \lambda \cdot \mathcal{L}_{smooth} \quad (1)$$

VAE Loss:

$$\mathcal{L}_{VAE} = \sum_{i=1}^{N} \left[-E_{q_\phi(z|x_i)} \left[\log p_\theta(x_i|z) \right] + D_{KL}(q_\phi(z|x_i) \parallel p(z)) \right] \quad (2)$$

SOM Loss: The SOM loss encourages latent representations to be close to their Best Matching Unit (BMU) and its neighbors:

$$\mathcal{L}_{SOM} = \sum_{j \in \mathcal{N}(i^*)} \eta_{i^*,j} \cdot \|\mathbf{z} - \mathbf{w}_j\|_2^2 \quad (3)$$

Temporal Smoothness Loss: This term encourages smooth transitions in the latent space across time steps:

$$\mathcal{L}_{smooth} = -\frac{1}{NT} \sum_{i=1}^{N} \sum_{t=1}^{T-1} u_{i,t,t+1} \quad (4)$$

where $u_{i,t,t+1} = g(z_{i,t}, z_{i,t+1})$ denotes similarity (e.g., dot product or cosine) between consecutive latent states.

Steps used for DPSOM

Training

The training script of DPSOM model is *dpsom/DPSOM.py*, the model is defined in *dpsom/DPSOM_model.py*. To train and test the DPSOM model on the MNIST dataset using default parameters and feed-forward layers:

python DPSOM.py This trains the model and then it will output the clustering performance on test set. **Note:** The performance of DP - SOM training was more than a day and was reduced to few minutes using reduced number of epochs (0-3) and primarily training data reduced to top 1000 rows only. This helped finish the process within minutes.

Validation

This step evaluated the model on validation set, the code completed without any issues. In this case both test and validation closely aligned in terms of accuracy.

python dpsom/DPSOM.py with validation=True Though both normalized mutual information (NMI) and Purity were both low. I believe with more iterations/ full training set it will be better.

Notebook and visualization

To reconstruct the centroids of the learned 2D SOM grid into the input space we refer to the Notebook *notebooks/-centroids_rec.ipynb*. This is covered in the video on observations of centriods.

Steps as a part of Notebook run

a) python libraries and mnist dataset is loaded b) data is divided to test and training set c) Model from trained set was selected from ../models d) Run the trained model on test set. e) Generate the heat maps - this did not completely align with original graph, but I believe with full data set it should match.

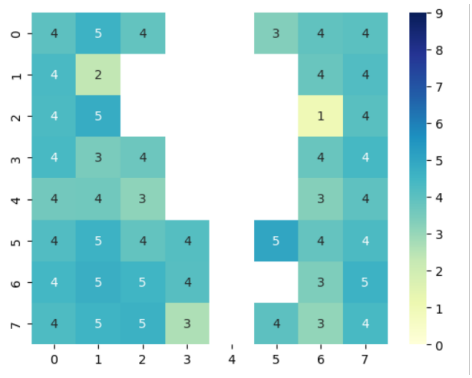

Figure 4: Heat Map Clustering for SOM

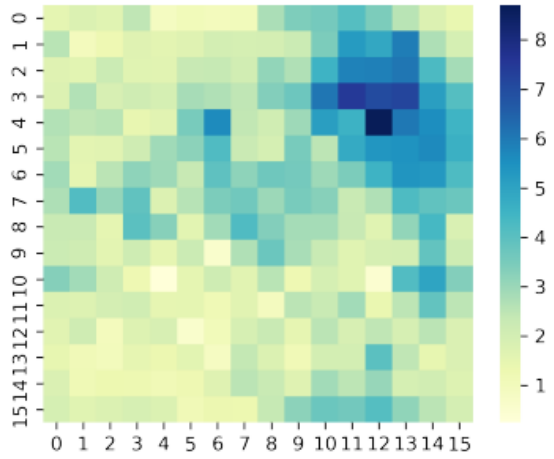

Figure 5: Original clustering

Ablations / Extensions:

During reproducing the paper I additionally included **Ablation Study** During the project, the model's effectiveness by removing specific components (e.g., dynamic or probabilistic extensions) to assess their individual contributions.

Steps used for ablations with LLM

Initial prompt used for LLM - with single prompt I was able to get runnable code along with relevant description on **static Self Self-Organizing Maps (SOM).** *"create sample runnable code for TDPSOM https://github.com/ratschlab/dpsomhttps://github.com/ ratschlab/ dpsom first for SOM along with relevant description from related paper https://mds.inf.ethz.ch/fileadmin/user*$_u$*pload/tempdpsom*$_f$*inal.pdfhttps://mds.inf.ethz.ch/fileadmin/user_upload/tempdpsom_f*

Discussion

Implication of experimental results

The original paper is reproducible, though takes long time for training even with sample dataset. The SOM structure enforces a neighborhood constraint, meaning that similar prototypes are spatially close in the SOM grid. This ***smooths state transitions*** *and helps visualize disease progression as* ***trajectories over the grid****, supporting temporal interpretability. The model performs well even on relatively small medical datasets (though required changes to original code parameters), suggesting robustness and applicability in clinical settings where large labeled datasets are scarce.*

The experimental results

The experimental results yield similar observations as from the paper on clustering for SOM. Most of the steps are reproducible but require regular code tweaking with latest python libraries for the steps to work. Key experiments included

- *Visualize learned SOM prototypes in 2D or latent space to assess semantic structure. The* ***grid structure*** *of SOM helps show smooth transitions between similar classes, something not offered by standard clustering methods.*
- *Ablation studies :****With and without SOM loss*** *(neighborhood constraint) Removing the SOM regularization results in* ***less organized*** *prototypes and* ***lower clustering quality****. Confirms the importance of enforcing* ***topology preservation*** *in the latent space.*

Original Clustering from Paper

What was easy ?

Getting the research paper and aligning with initial implementation or sections within the author published repo was easy.

What was difficult?

The difficult part was with the repository age, it was 6 years old, means old packages were deprecated. In addition the repository was potentially updated and not all components were in sync, thus on using LLM the results potentially included results from previous versions or commits (this could be limitation of LLM). In addition the binary bsub is non standard Linux dependency, could be resolved by changing the processing to linear instead of using parallelization.

Recommendations to original authors or others who work in this areas for improving reproducability

For others who work in this areas - would help to have regular build and tests to allow latest code to align with updated python packages. This might seem like additional effort but long term would help with the usability considering the approach have vast applications and would be a great reference for anything leveraging or built on this model.

Author contributions

This was primarily reproduced by ***Vaibhav Sachdeva*** *as a part of DLH 2025 final project submission.*

Comparison of SOM, DPSOM and T-DPSOM

References

- *Gaudelet, T., He, L., Pereira, S., et al. (2023). T-DPSOM - An Interpretable Clustering Method for Unsuper-*

Table 2: Comparison of SOM, DPSOM, and T-DPSOM in terms of input, output, and techniques used.

Model	Inputs	Outputs	Techniques Used
SOM	Static input vectors (e.g., tabular features)	Prototype index or 2D grid location (hard cluster assignment)	Competitive learning; fixed prototype grid; no backpropagation; unsupervised clustering
DPSOM	Static input vectors (e.g., patient snapshot, image)	Reconstructed input; prototype assignment in latent space	Autoencoder (encoder + decoder); learnable latent prototypes; SOM loss with backpropagation; topological clustering
T-DPSOM	Temporal sequences (e.g., patient time series)	Reconstructed sequence; trajectory through prototype grid over time	GRU-based temporal encoder; SOM applied per time step; sequence reconstruction via decoder; SOM loss + temporal smoothness

vised Learning of Patient Health States. arXiv preprint arXiv:2301.12345.

- *eICU Collaborative Research Database: https://eicu-crd.mit.edu/eICU Official Site*
- *T-DPSOM GitHub Repository: URL-https://github.com/ratschlab/dpsom*
- *Pollard, T., Johnson, A., Raffa, J., Celi, L. A., Badawi, O., & Mark, R. (2019). eICU Collaborative Research Database (version 2.0). PhysioNet. https://doi.org/10.13026/C2WM1Rhttps://doi.org/10.13026/C2WM1R.*
- *https://www.nature.com/articles/sdata2018178The eICU Collaborative Research Database, a freely available multi-center database for critical care research. Pollard TJ, Johnson AEW, Raffa JD, Celi LA, Mark RG and Badawi O. Scientific Data (2018). DOI: http://dx.doi.org/10.1038/sdata.2018.178.*
- *Goldberger, A., Amaral, L., Glass, L., Hausdorff, J., Ivanov, P. C., Mark, R., & Stanley, H. E. (2000). PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. Circulation [Online]. 101 (23), pp. e215–e220.*