

RESEARCH ARTICLE

TRUNC: A Transfer Learning Unsupervised Network for Data Clustering

RITA XAVIER¹, JOHN PELLER², AND LEANDRO NUNES DE CASTRO^{1,2}¹School of Technology, Universidade Estadual de Campinas, Limeira, São Paulo 13484-332, Brazil²Department of Computing and Software Engineering, Dendritic: A Human-Centered AI and Data Science Institute, Florida Gulf Coast University, Fort Myers, FL 10501, USA

Corresponding author: Rita Xavier (r223447@dac.unicamp.br)

This work was supported in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior–Brasil (CAPES) under Grant 001, and in part by São Paulo Research Foundation (FAPESP) under Grant #2021/11905-0.

ABSTRACT There is a demand for effective clustering methods, especially given the increasing complexity of data scenarios in modern applications. Motivated by the limitations of traditional clustering algorithms, particularly in handling noise, imbalanced data, and large-scale datasets, this work introduces TRUNC, a Transfer Learning Unsupervised Network for Data Clustering. TRUNC leverages a bio-inspired approach, utilizing a single-layer feed-forward winner-takes-all neural network enhanced with transfer learning to optimize clustering performance. The proposed algorithm is evaluated across a range of synthetic and real-world datasets, demonstrating its robustness and performance over conventional and state-of-the-art clustering methods. Key contributions include the integration of transfer learning into the clustering process, a detailed sensitivity analysis of the TL parameter, and extensive experimentation that confirms the efficiency and generalization capability of TRUNC. Results indicate that TRUNC improves clustering quality and convergence speed, offering a competitive and scalable solution for various clustering tasks. This proposal opens new avenues for the integration of TRUNC's principles with other bio-inspired algorithms and its application across diverse domains.

INDEX TERMS Unsupervised learning, self-organizing neural network, transfer learning, data clustering, bio-inspired computing.

I. INTRODUCTION


Unsupervised machine learning involves training a model on unlabeled data [1]. In this type of learning, the algorithm must discover patterns and structures in the data without using information about the group membership. This is usually used for tasks like clustering, dimensionality reduction, data summarization, data segmentation, and anomaly detection [2].

One of the approaches that can be used for unsupervised learning is transfer learning (TL), in which a pre-trained model is used to provide learning parameters for other models, thus being able to solve or optimize similar problems [3]. In this technique, knowledge from a pre-trained model (source domain), which obtained good results in solving previous problems (source tasks), is transferred to a new

model to be trained (target domain), with the objective of solving a new problem (target task), which may or not be similar to the previous one [2].

Clustering is a fundamental task in unsupervised machine learning and seeks to identify hidden structures or patterns in a dataset [4]. The objective is to organize or divide the data into groups, called clusters, so that objects within the same group are more similar to each other than to objects from other groups [5]. Due to the unsupervised nature of clustering, TL is expected to be beneficial for this type of task, as it can take advantage of good problem solutions and prior domains to improve the efficiency of algorithms intended to solve this type of task [5].

Although TL is widely used in Deep Learning (DL) [6], it is still at an early stage among bio-inspired algorithms, such as evolutionary algorithms, swarm intelligence, artificial neural networks and others, which compose a category of

The associate editor coordinating the review of this manuscript and approving it for publication was Tyson Brooks .

algorithms inspired by biological systems aiming to solve complex problems [7], [8].

The review carried out in [8] shows various studies about bio-inspired algorithms with TL for solving different problems. Despite that, the number of studies focusing on the clustering task is small. This gap in knowledge led to the intention of developing novel transfer-based bio-inspired algorithms that would provide good performance in solving clustering tasks.

The main goal of this study is to introduce a novel bioinspired algorithm named Transfer Learning-Based Unsupervised Network for Data Clustering (TRUNC), which consists of a single-layer feed-forward winner-takes-all neural network, with the implementation of TL. The objective is to solve clustering tasks with optimal quality and good execution time, thus becoming a competitive algorithm with good generalization capability.

The main contributions of this paper are as follows:

- A novel Transfer Learning-Based Unsupervised Network for Data Clustering (TRUNC) is proposed, which leverages transfer learning to enhance clustering performance.
- The implementation of a single-layer feed-forward winner-takes-all neural network, integrated with transfer learning, to optimize clustering tasks.
- A comprehensive evaluation of TRUNC using synthetic and real-world datasets, showing its robustness and performance compared to traditional unsupervised clustering methods and other transfer-based unsupervised algorithms.
- A sensitivity analysis of the transfer learning parameter λ , showing its impact on clustering quality and convergence speed.
- Statistical validation of TRUNC's performance using the Student's Wilcoxon Signed-Rank Test, confirming the reliability of the obtained results.
- Comparison of TRUNC with other classical algorithms, like K-Means and DBSCAN.
- Proposal of future research directions, including the integration of TRUNC with other bio-inspired algorithms and its application in different domains (like temporal datasets), and validation with applications in text analysis and image segmentation.

This paper is organized as follows. Section II presents the theoretical background for the paper, in Section III related works with the state-of-the-art algorithms using Prototype-Based TL (PBTL) are presented. Section IV presents the proposed algorithm TRUNC, and Section V presents the performance evaluation and validation of TRUNC. Section VI brings the conclusion and future trends.

II. THEORETICAL BACKGROUND

This section aims to review the concepts of TL and unsupervised neural networks so as to form the basis for the proposal of a transfer learning network for clustering.

A. THE UNSUPERVISED NEURAL NETWORK FOR DATA CLUSTERING

An Unsupervised Network for Clustering (UNC) is a type of network designed to perform unsupervised learning tasks [9]. For these networks, training is competitive, not requiring labeled data, as their neurons compete with each other to represent input patterns. The architecture used in this study is a single-layer feed-forward network, with postsynaptic neurons positioned at the nodes of a one-dimensional grid. A feed-forward neural network is the one in which data only flows in one direction: from input to output. This means that connections between neurons do not form cycles, and the network has no feedback loop [10], [11], [12].

A single-layer feed-forward network is illustrated in Figure 1. This network is trained using a winner-takes-all competitive learning scheme, where output neurons selectively adjust to different input patterns or groups of patterns through unsupervised learning. Only one neuron actively responds at a time for a given input pattern.

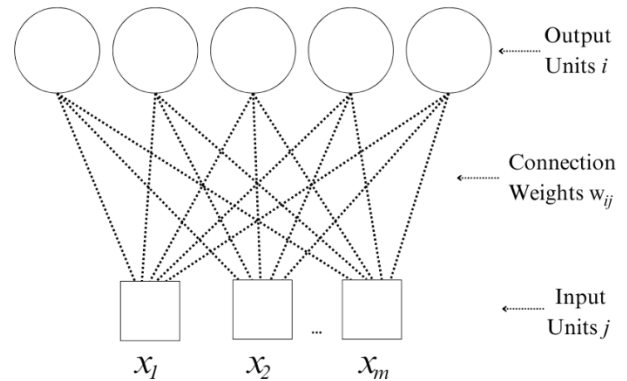


FIGURE 1. Representation of a single-layer feed-forward neural network.

An UNC begins by initializing the synaptic weights of the neurons, usually by assigning them small random values. This step is crucial so that no prior order is imposed on the network. After initialization, two essential processes occur in the formation of the self-organizing network: competition, where neurons compete to determine the winner for each input pattern; and adaptation, where the winning neuron has its associated weight vector updated.

The UNC can be formalized as follows. Let $\mathbf{x} = [x_1, x_2, \dots, x_m]^T \in \mathbb{R}^m$ be an input vector (training pattern) that is assumed to be presented in parallel to all neurons $i = 1, \dots, o$ throughout the network. The complete dataset \mathbf{X} consists of N patterns, $\mathbf{X} \in \mathbb{R}^{m \times N}$. The weight vector of unit i is $\mathbf{w}_i = [w_{i1}, w_{i2}, \dots, w_{im}] \in \mathbb{R}^{1 \times m}$, and the weight matrix of the network is $\mathbf{W} \in \mathbb{R}^{o \times m}$, where o is the number of output (post-synaptic) units of the network.

Regarding competition, for each input pattern, the o neurons in the network will compete to become active. That is, for a neuron i to be the winner, the distance between its corresponding weight vector \mathbf{w}_i and the input pattern \mathbf{x} must be the smallest one (among all the output units in the network),

given a certain distance measure $|\cdot|$, usually considered to be the Euclidean distance. Let $i(\mathbf{x})$ denote the winning neuron for the input pattern \mathbf{x} :

$$i(\mathbf{x}) = \arg \min_j |\mathbf{x} - \mathbf{w}_j|, \quad j = 1, \dots, o \quad (1)$$

As for adaptation, the weight updating rule is given by:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \alpha(t) [\mathbf{x}(t) - \mathbf{w}_i(t)] \quad (2)$$

where the learning rate $\alpha(t)$ decreases over time, $0 < \alpha(t) < 1$.

Sometimes, it is useful to keep the initial value of α fixed for several iterations, usually 1,000, before starting to reduce it. Following this approach may result in an initialization corresponding to a self-organization phase of the network, followed by a convergence phase where, as the values of α are systematically reduced, the network undergoes ever smaller updates. A simple way to reduce α is to adopt a geometric decay, i.e., multiplying α by a constant value less than one.

B. TRANSFER LEARNING FOR CLUSTERING

Clustering refers to the process of organizing or segmenting objects into groups of similar objects, called clusters. Clusters must satisfy two main principles [13], [14], [15], [16], [17]: each cluster must be homogeneous, meaning that objects within the same group are similar to each other; and each cluster must differ from the others, which means that objects within a cluster must be distinguished from objects in other clusters.

From a partitional data clustering perspective, the clustering task can be defined as obtaining k subsets (groups or categories) of a dataset \mathbf{X} with N objects $\{\mathbf{X} = \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ where each $\mathbf{x}_i \in \mathbb{R}^m$ is an m -dimensional vector. The objects $\mathbf{x}_i, i = \{1, 2, \dots, N\}$, must be grouped into k non-empty and disjoint groups $C = \{C_1, C_2, \dots, C_k\}$, such that: $C_1 \cup C_2 \cup \dots \cup C_k = \mathbf{X}$; $C_j \neq \emptyset$; $C_i \cap C_j = \emptyset, i \neq j$. Another important aspect of clustering is its performance evaluation, which can be measured *externally*, comparing the clusters found with a pre-defined structure, or *internally*, by evaluating the compactness and/or separation of the clusters [15], [17].

TL operates on the principle that knowledge acquired in one context can be transferred to another, improving the learning process. This approach reflects the idea that, just as humans build on existing knowledge to acquire new skills or concepts, learning models must have the ability to do the same. The fundamental idea is to efficiently exploit the acquired knowledge, allowing models to adapt and generalize to new tasks or domains [3]. This approach arose from the realization that training learning models, especially neural networks, can be computationally expensive. TL provides a strategic shortcut by allowing models to inherit knowledge from related domains or tasks, resulting in better performance, faster convergence, and greater ability to handle problems with limited data availability [2], [18].

TL is characterized by the adaptation of a model or system from a source domain, known as \mathcal{D}_S , with a corresponding domain task \mathcal{T}_S , to improve the performance of a target task \mathcal{T}_T , in a target domain \mathcal{D}_T , where $\mathcal{D}_S \neq \mathcal{D}_T$ and $\mathcal{T}_S \neq \mathcal{T}_T$:

$$\mathcal{D}_S, \mathcal{T}_S \rightarrow \mathcal{D}_T, \mathcal{T}_T$$

where \rightarrow corresponds to the process of transferring knowledge.

The primary objective is to facilitate the learning of \mathcal{T}_T by capitalizing on the insights and experience obtained from \mathcal{T}_S . Traditional TL can be divided into four transfer strategies [3]:

- **Instance-Transfer:** The objective is to reuse parts of instances from a source domain to improve clustering performance in a target domain. This is done through instance re-weighting, where the importance of each instance is adjusted based on its relevance to the target domain. This technique can improve clustering performance by allowing useful instances to be prioritized during the clustering process.
- **Feature-Representation-Transfer:** The focus is on learning a “good” feature representation for the target domain. This is achieved by encoding the knowledge used for cross-domain transfer into the learned feature representation. With a new feature representation, the clustering performance in the target domain can improve significantly as the learned features are better suited for the specific clustering task.
- **Parameter-Transfer:** In this strategy, it is assumed that the source domain and the target domain share some parameters or prior distributions of model hyperparameters. The transferred knowledge is then encoded in the shared parameters or distributions. This allows knowledge to be transferred between domains, improving cluster performance in the target domain by taking advantage of shared parameters or distributions.
- **Relational-Knowledge-Transfer:** Here the focus is on transferring knowledge about the relationships between instances in the source and target domains. The basic assumption is that the relationships between instances are similar across domains. This means that the knowledge to be transferred is the relationship between instances, which can improve clustering performance in the target domain as relationships between instances are better captured.

In TL, as shown in Figure 2, the data sampling and knowledge gained are typically from the source domain. Due to the abundance of data and limited (or absent) labeling, and different distribution, direct data clustering is not feasible [19].

Another issue that may arise is the inaccessibility of the source data, as often happens with real datasets that contain sensitive information. Thus, the collected data (knowledge) differs from the sampled data, with only the characteristics necessary for transfer being extracted, such as representations of characteristics, parameters and relationships [19].

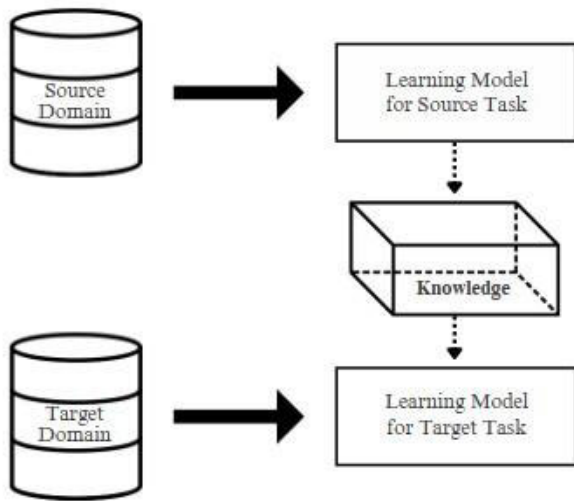


FIGURE 2. Transfer learning useful knowledge transfer framework.

III. RELATED WORKS

After carrying out a systematic review (under consideration and peer review for publication), the authors found several TL techniques, such as Manifold TL [20], Self-Adaptive TL [21], Two-Level TL [22] and prototype-based TL [19], [23], [24], [25], [42], among others.

By analyzing these techniques, it was observed that one of them stands out for its greater adaptability for use in clustering tasks: the prototype-based TL. In this technique, the prototypes found (in general, the centers of the groups) are transferred as useful knowledge to be used in the learning step of a clustering algorithm [19].

PBTL was observed in six algorithms found in the review made by the authors: the Transfer Fuzzy C-Means (TFCM) and Transfer Fuzzy Subspace Clustering (TFSC) [19], Transfer Learning-based Evidential C-Means Clustering (TECM) [23], ProtoTransfer [24], Differential Evolution-based Transfer Rough Clustering Algorithm (DE-TRC) [24], and Evidential Transfer Evidential C-Means (ETECM) [42].

TFCM and TFSC are algorithms that improve clustering when there is little data in the target domain. TFCM incorporates knowledge from the source domain by adapting cluster centers, whilst TFSC, which targets high-dimensional data, also uses feature weights to identify relevant attributes. Both methods balance the influence of data from the target domain and the source domain, iterating until convergence. These methods outperform their traditional counterparts, especially in scenarios with limited data in the target domain [19].

The TECM algorithm improves clustering in target domains by integrating knowledge from source domains through barycenter's and a custom objective function. It generates a credal partition, allowing objects to belong to multiple clusters, and is flexible to handle different numbers of clusters across domains. TECM improves performance in scenarios with limited or contaminated target data,

effectively identifies ambiguous objects, and achieves good results through iterative optimization [23].

ProtoTransfer is a few-shot classification algorithm composed of two stages: ProtoCLR, a self-supervised pre-training phase that learns an embedding metric via contrastive pooling; and ProtoTune, a fine-tuning phase that computes class prototypes from the support set samples and fine-tunes a linear layer. Its techniques, such as the use of larger batch sizes during pre-training and efficient fine-tuning, provide good performance, especially on cross-domain tasks [24].

The DE-TRC algorithm combines transfer learning and differential evolution to optimize the clustering process in challenging scenarios. It uses knowledge of the source domain, such as cluster centers, to assist in clustering limited or noisy data in the target domain, applying imprecise set theory to deal with uncertainty. Differential evolution optimizes the objective function, improving robustness and avoiding local optima. The process involves initializing a population of candidate solutions, evolving them via mutation, crossover, and selection, and evaluating them based on the compactness of the target domain data and their similarity to the source domain centers. The algorithm iterates until it meets a stopping criterion, and the best solution is used to determine the final cluster centers [25].

ETECM is an unsupervised transfer learning algorithm for clustering in scenarios where the source and target domains have different numbers of clusters. It updates three main components: the membership matrix, the prototypes, and the similarity matrix, using knowledge from the source domain and data from the target domain. The algorithm applies belief function theory to deal with uncertainty and demonstrated effectiveness in tasks with uncertain or insufficient target data [42].

The proposed algorithm TRUNC shares with these other TL algorithms the goal of improving clustering with limited or noisy data, but it differs by its simpler and more straightforward approach. While TFCM and TFSC [19], TECM [23], ETECM [42] and DE-TRC [25] use complex theories and represent uncertainties, TRUNC focuses on combining cluster compactness in the target domain and similarity between prototypes from both domains, without resorting to complex representations. Furthermore, unlike ProtoTransfer [24], which focuses on few-shot classification, TRUNC focuses on clustering and simplifying knowledge transfer. Thus, TRUNC is an efficient and less complex solution compared to other alternatives, bringing a more agile process with less computational cost.

IV. TRUNC: A TRANSFER LEARNING UNSUPERVISED NETWORK FOR DATA CLUSTERING

This section provides a comprehensive overview of the main concepts relating to the implementation of the proposal. The details of the algorithm will be investigated, highlighting its characteristics and functionalities.

A. PROTOTYPE-BASED TRANSFER PROCESS

Prototype-based TL is a broadly used technique, especially in clustering algorithms, that uses clusters identified in a source domain as representative models to aid in clustering data in a target domain. It is particularly beneficial when data in the target domain is sparse or of lower quality, as it allows for more effective data analysis across multiple application domains [25].

PBTL is a clustering process that typically involves minimizing an objective function J that considers a partition matrix U and a set of prototypes W [19]:

$$\min J = \min f(U, W) \quad (3)$$

$$\min_w J_{\text{UNC}} = \sum_i \sum_j |\mathbf{x}_j - \mathbf{w}_i|^2 \quad (4)$$

where J_{UNC} is the cost function to be optimized (minimized), \mathbf{x}_j is the j -th input pattern, \mathbf{w}_i is the i -th prototype (weight vector for neuron i), $i = 1, \dots, o$, and $|\cdot|$ is the distance function, usually the Euclidean distance, that quantifies the distance between the weight vector and the input pattern.

TL can be used in several ways to assist this learning process, but there is a problem in understanding what information is relevant and what knowledge should be transferred from one problem domain to another, or vice versa. In the PBTL technique, the most valuable information is the location of the prototypes (or centroids) of the group [19], [25]:

$$\Delta(\widetilde{W}_S, W_T) = \sum_{k=1}^o |\widetilde{\mathbf{w}}_k - \mathbf{w}_k|^2 \quad (5)$$

where \widetilde{W}_S is the set of prototypes for the source domain, W_T is the set of prototypes for the target domain, $\widetilde{\mathbf{w}}_k$ is the centroid of the k -th cluster (weight vector of the network) for the source domain, \mathbf{w}_k is the centroid of the k -th cluster for the target domain, and o is the number of clusters in the source and target domains. Note that in the case of the unsupervised network, each weight vector of each neuron represents the centroid of a cluster.

Equation (5) states that the PBTL process must consider the centroids of the source domain and move the centroids of the target domain in their direction as a way of transferring knowledge from one domain to another.

B. PROTOTYPE-BASED TRANSFER PROCESS

The objective of the learning process was presented in Equation (4) and corresponds to the process of finding the most compact clusters, that is, those whose distance between the input patterns and their corresponding centroids is minimum.

The objective function of the TRUNC algorithm uses the target domain data X_T and the cluster centers $\widetilde{W}_S = \widetilde{\mathbf{w}}_k$, $k = 1, \dots, o$, from the source domain as auxiliary knowledge. The cluster centers of the source domain are the weight vectors of the network trained on the source data and the new objective function becomes:

$$\min_w J_{\text{TRUNC}} = \sum_i \sum_j |\mathbf{x}_j - \mathbf{w}_i|^2 + \lambda \sum_k \sum_i |\mathbf{w}_k - \mathbf{w}_i|^2 \quad (6)$$

where $i = 1, \dots, c_T$ is the index of neurons (prototypes) in the target domain, $j = 1, \dots, N$ is the index of input patterns (objects), $k = 1, \dots, c_S$ is the index of neurons in the source domain, and λ is the transfer learning parameters, that is, the trade-off parameter responsible for weighting the influence of the source domain on the learning of the target domain. In this case, $c_S = c_T = o$, meaning both networks have the same number of output neurons.

Note that the first term of Equation (6) measures the compactness of the groups in the target domain, whilst the second term measures the similarity between the weight vectors (prototypes) in the source and target domains. This TL integration introduces new ways to improve clustering performance by leveraging prototype-based TL strategies. Based on this proposal; to adapt the learning process and incorporate TL into the self-organizing network, it is necessary to adjust the weight update presented in Equation (2) to accommodate the PBTL mechanism.

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \alpha(t) [\mathbf{x}(t) - \mathbf{w}_i(t)] + \lambda(t) [\widetilde{\mathbf{w}}_k(t) - \mathbf{w}_i(t)] \quad (7)$$

where $\lambda(t)$ is the TL learning parameter, and $\widetilde{\mathbf{w}}_k$ is the trained weight vector from the source domain that is closest to the winning neuron i . Hyperparameter $\lambda(t)$ is subject to a linear decrease over time, similar to the one of the network learning rate, $\alpha(t)$. The pseudocode for TRUNC is summarized in Algorithm 1.

Algorithm 1 TRUNC: Transfer Learning Unsupervised Network for Clustering

TRUNC pseudocode

1. **function** TRUNC (*data*, *n_protos*, *mode*, *source_weight*, *source_protos*, *max_iter*, *learning_rate*, *convergence_limit*)
 2. Initialize prototypes randomly
 3. **for** $i = 1$ to max_iter :
 4. **for** each point in data:
 5. Find closest prototype
 6. Compute movement direction
 7. **if** $\text{mode} == \text{target_tl}$ and source_protos :
 8. Add weighted movement from *source_protos*
 9. Update prototype position
 10. Check convergence (movements < limit)
 11. **if** converged: **break**
 12. **return** prototypes and movement log
 13. **end function**
-

V. PERFORMANCE EVALUATION

The assessment of TRUNC involves various different experiments and performance measures. Initially we are going to do a sensitivity analysis aimed at understanding the influence of the transfer learning parameter λ at different performance measures and at the convergence speed of the algorithm. Then, some experiments will be performed with eight synthetic datasets with different characteristics in the source and

target domains and finally experiments will be performed with datasets from the literature.

A. EXPERIMENTAL METHODOLOGY

This section describes the experiments to be performed, the datasets to be analyzed, and the performance measures to be used. The experimental results and corresponding analyses are presented in the following section.

1) EXPERIMENTS AND BENCHMARKS

To assess the performance of TRUNC, three sets of experiments were carried out:

- **Sensitivity Analysis to the TL Parameter:** The first set of experiments to be performed aim at investigating the sensitivity of TRUNC in relation to λ . The goal of the algorithm in terms of clustering quality and convergence speed for increasing values of λ .
- **Synthetic Data:** Four synthetic datasets created by [25], plus four synthetic datasets developed by us to explore specific properties of TRUNC.
- **Real-world Data:** Five real-world datasets available at the UCI and Kaggle repositories.

The experiments were inspired by the works of [19] and [25], which also inspired the development of TRUNC. In [19] the authors proposed two Prototype-Based Fuzzy Clustering (TPFC) algorithms for TL. The first, Transfer Fuzzy C-Mean (TFCM), is an adaptation of Fuzzy C-Means (FCM) that uses transfer learning to improve clustering by combining information from the source domain with data from the target domain. TFCM optimizes an objective function that considers both the data from the target domain and the cluster centers of the source domain, adjusting parameters to balance these influences [19].

The second algorithm, Transfer Fuzzy Subspace Clustering (TFSC), also presented in [19], is an adapted version of FSC for transfer learning, using fuzzy partitions and adjusting the partition matrix based on knowledge of the source domain. TFSC is also implemented through an iterative process, updated until convergence or the maximum number of iterations is reached. The results show that both TFCM and TFSC are effective compared with traditional methods and related algorithms such as multi-task clustering [19].

The DE-TRC algorithm, proposed by [25], uses a transfer learning mechanism to design an objective function for unsupervised rough clustering. It combines PBTL with rough clustering algorithms to improve the classification of sparse and impure data. In addition, a Differential Evolution (DE) algorithm is employed to optimize the objective function, increasing the robustness of the proposal. The introduction of the transfer learning mechanism helps DE-TRC deal with limited or noisy data, and differential evolution solves problems related to the sensitivity of initial cluster centers.

For the experiments and results to be presented here, we did our own implementation of DE-TRC using the pseudocodes and details provided by its developers in their published

works. This was done so that there could be a direct comparison among the competing algorithms, using the same parameters and information and, thus, providing a fair comparison.

In terms of development, the algorithms were implemented in Python, using the Numpy, Pandas, Matplotlib, Seaborn, Time, IterTools, JSon, Scikit-Learn and Scipy libraries. The experiments were carried out in the Google Collaboratory environment (real datasets) and Visual Studio Code (synthetic datasets).

2) DATASETS

For the synthetic data experiments, the datasets are two-dimensional and are designed to evaluate the effectiveness of TL for clustering in some scenarios. Four datasets were generated as presented in [25] and described in Table 1. These datasets were used to establish a benchmark for TRUNC. The visual distribution of the data can be seen in Figure 3, where the \mathcal{D}_S (source domain) objects of each dataset are presented, and in Figure 4, where the \mathcal{D}_T (target domain) objects of each dataset are illustrated.

Furthermore, there is a need to validate the robustness of the algorithm in various scenarios, in special with datasets with different number of dimensions. Thus, new synthetic datasets with different characteristics were created, including other ways of carrying out training (for example, training the algorithm with fewer objects in the source domain than those in the target domain), with a substantially larger number of objects than in the datasets provided in the previous studies by [25]. The visual distribution of these datasets can be seen in Figures 3 and 4.

TABLE 1. Main characteristics of the real-world datasets.

Dataset	Features' Types	Instances	N° of Features
Wine	Real, Integer	178	13
Iris	Real	150	4
Breast Cancer Wisc.	Integer	699	9
Haberman's Survival	Integer	306	3
Wir. Ind. Localization	Real	2000	7

In addition to the synthetic datasets, experiments were carried out using datasets from the literature [25] available at the UCI or Kaggle repositories: Wine [26], Iris [27], Breast Cancer Wisconsin [28], Haberman's Survival [29] and Wireless Indoor Localization [30]. The characteristics of all these datasets are summarized in Table 2.

3) PERFORMANCE MEASURES

The performance of TRUNC will be assessed using five external measures [31], [32], [33], [40], [41]: Accuracy (ACC), Normalized Mutual Information (NMI), Rand Index (RI), Adjusted Rand Index (ARI), and Silhouette Score. The number of epochs for convergence (EC), that is, the number of epochs until the algorithm does not show significant

TABLE 2. Synthetic datasets characteristics.

	Cluster	Source			Target		
		Mean	Covariance	Size	Mean	Covariance	Size
Dataset 1: S1/T1	C1	[- 5 5]	[5 0;0 5]	200	[- 5 5]	[8 0;0 8]	10
	C2	[0 15]	[5 0;0 5]	200	[0 15]	[8 0;0 8]	10
	C3	[5 5]	[5 0;0 5]	200	[5 5]	[8 0;0 8]	10
Dataset 2: S2/T2	C1	[2 4]	[10 0;0 10]	250	[3 4]	[10 0;0 11]	125
	C2	[9 15]	[25 0;0 7]	250	[10 12]	[25 0;0 7]	125
	C3	[8 30]	[30 0;0 20]	250	[9 29]	[30 0;0 20]	125
Dataset 3: S3/T3	C1	[2 4]	[10 0;0 10]	250	[3 4]	[10 0;0 11]	125
	C2	[9 15]	[25 0;0 7]	250	[10 12]	[25 0;0 7]	125
	C3	[8 30]	[30 0;0 20]	250	[9 29]	[30 0;0 20]	125
	C4	-	-	-	[20 20]	[4 0;0 4]	35
Dataset 4: S4/T4	C1	[10 5]	[5 0;0 5]	450	[10 5]	[5 0;0 5]	40
	C2	[10 15]	[5 0;0 5]	300	[10 15]	[5 0;0 5]	40
	C3	[15 -5]	[2 0;0 1]	150	[15 -5]	[2 0;0 1]	10
Dataset 5: S5/T5	C1	[2, 4]	[10 0; 0 10]	200	[3, 4]	[10 0; 0, 11]	1.000
	C2	[9, 15]	[25 0; 0 7]	200	[10, 12]	[25 0; 0 7]	1.000
	C3	[8, 30]	[30 0; 0 20]	200	[9, 29]	[30 0; 0 20]	1.000
Dataset 6: S6/T6	C1	[-5, 5]	[5 0; 0 5]	10.000	[-5, 5]	[8 0; 0 8]	2.000
	C2	[0, 15]	[5 0; 0 5]	10.000	[0, 15]	[8 0; 0 8]	2.000
	C3	[5, 5]	[5 0; 0 5]	10.000	[5, 5]	[8 0; 0 8]	2.000
Dataset 7: S7/T7	C1	[-20, 20]	[30 10; 10 30]	600	[-20, 20]	[30 10; 10 30]	300
	C2	[-30, 30]	[25 5; 5 25]	400	[60, -60]	[20 5; 5 20]	200
	C3	[-10, 10]	[40 15; 15 40]	500	[-70, 70]	[35 15; 15 35]	250
	C4	[-50, 50]	[50 10; 10 50]	700	-	-	-
	C5	[60, -60]	[20 5; 5 20]	350	-	-	-
	C6	[-70, 70]	[35 15; 15 35]	450	-	-	-
Dataset 8: S8/T8	C1	[-15, 15]	[20 5; 5 20]	1.000	[-14, 14]	[18 4; 4 18]	250
	C2	[0, 30]	[25 -10; -10 25]	1.000	[1, 28]	[22 -8; -8 28]	250
	C3	[20, 20]	[30 10; 10 30]	1.000	[18, 18]	[28 9; 9 28]	250
	C4	[35, -15]	[15 5; 5 15]	800	[33, -13]	[14 4; 4 14]	150
	C5	[50, 10]	[10 0; 0 10]	800	[48, 8]	[9 0; 0 9]	150
	C6	[60, -20]	[20 -5; -5 20]	800	[58, -18]	[18 -4; -4 18]	150

improvement from one epoch to another, is also going to be used as a performance measure.

A significance statistical test was also needed. So, the Wilcoxon Signed-Rank Test was performed with a significance threshold at 0.05 [35]. The test was developed as a non-parametric alternative to the Student's T-Test [34], is indicated for data that do not follow a normal distribution, using difference ranks instead of dispersion metrics, such as the standard deviation [36]. This method is particularly suitable for comparing algorithm performance in clustering tasks, as in the study in question.

B. SENSITIVITY ANALYSIS

Sensitivity analysis is a method to understand the behavior of an algorithm when some specific hyper-parameters vary, and to choose suitable ones to be used in the final applications. Thus, two types of sensitivity analysis will be carried out here: a convergence analysis, and a performance analysis based on the evaluation measures (ACC, NMI, RI, ARI and Silhouette Score).

In TRUNC, parameter λ is used to weight the influence of TL on the unsupervised network. To understand its influence, a sensitivity analysis was performed on the synthetic datasets

with the value of λ varying from 0 to 1 with a fluctuation of 0.05.

In Figure 5 the clustering performance can be observed for synthetic datasets. The line charts show the values of the performance measures on the y-axis, and the values of λ on the x-axis.

According to the sensitivity test, the best results are obtained with the parameter λ between 0.8 and 0.9, since between 0.0 and 0.79 the values are not significant, and between 0.91 and 1.00 they begin to decrease. Then, according to the performed tests, the parameters λ were used at 0.90, the update rate of the unsupervised network at 0.01. As a parameter to understand the ideal number of clusters, it was used the elbow method with WCSS [37].

C. PERFORMANCE FOR THE SYNTHETIC DATASETS

Each synthetic dataset aims at assessing the TL-based clustering algorithm under a specific context. The first dataset (S1/T1) presents a common situation where there is little data available for training. The second and third datasets (S2/T2 and S3/T3) simulate situations where noise appears during data capture, potentially reducing the performance of the clustering algorithm. In these datasets there is additive

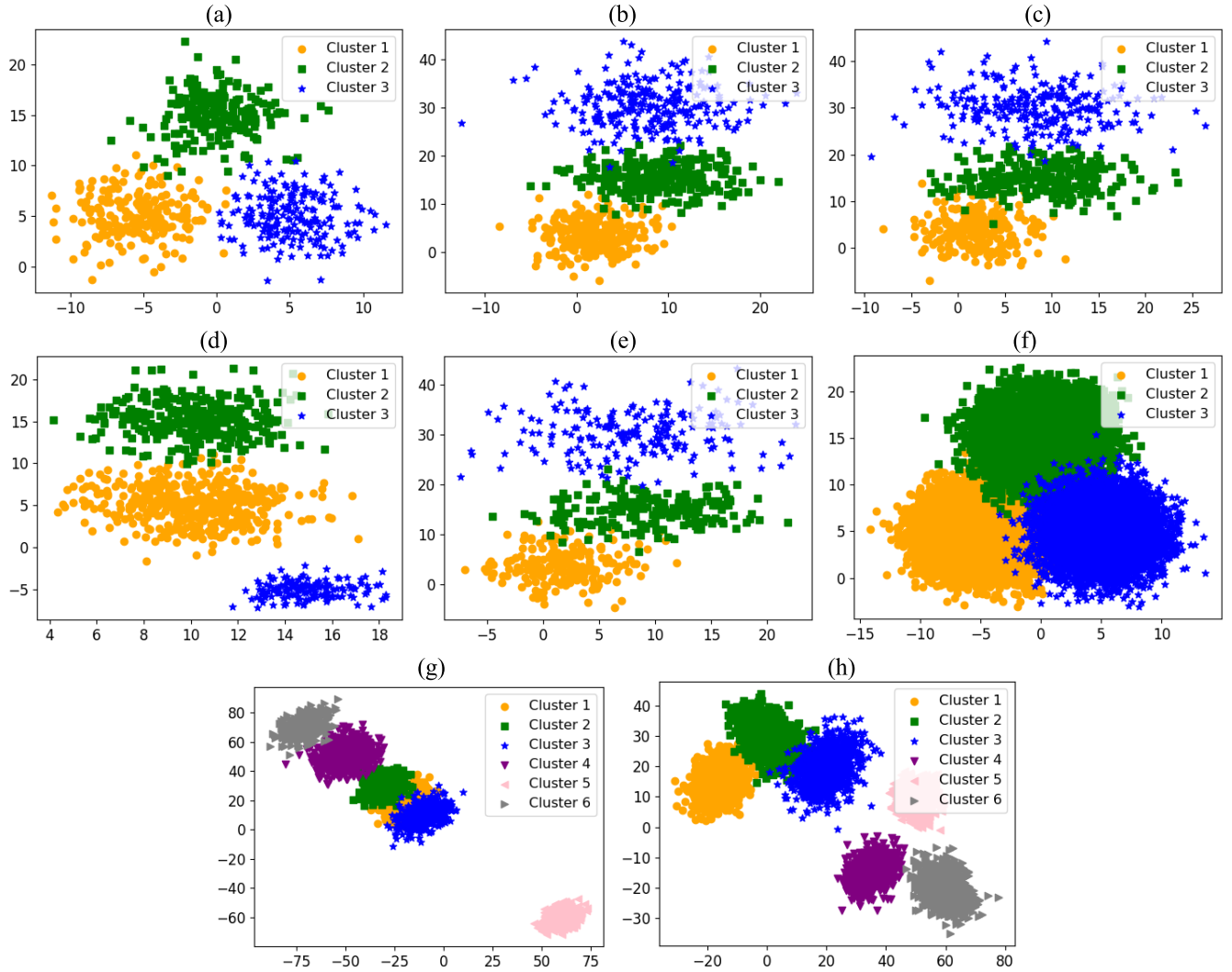


FIGURE 3. Synthetic datasets used for performance evaluation. \mathcal{D}_S data of synthetic datasets, (a) S1, (b) S2, (c) S3, (d) S4, (e) S5, (f) S6, (g) S7, (h) S8.

(S2/T2) and multiplicative (S3/T3) noise, respectively. The fourth dataset (S4/T4) simulates situations in which there is imbalance among the clusters, that is, the samples between the source and target domains are different.

In dataset 5 (S5/T5) the target domain has many more objects than the source domain. In dataset 6 (S6/T6) there are substantially more objects in the source domain than in the target domain, with the same distribution. In datasets 7 (S7/T7) and 8 (S8/T8) there are different distributions between the domains and slightly different numbers of objects and clusters.

Table 3 summarizes the results of the ACC, NMI, RI, ARI and the average silhouette score evaluation measures for the synthetic data. The average \pm standard deviation taken over 20 runs is shown for TRUNC and compared with the standard unsupervised network for clustering (UNC), that is, the network without TL, with K-Means [38] and DBSCAN [39], two traditional clustering algorithms, and also with the results

of the state-of-art algorithm DE-TRC [25]. The results show that the unsupervised network convergence is substantially improved by TL.

Regarding the clustering results, in Figure 6 it is possible to observe the original centroids (black hexagons) and those found by the UNC algorithm (red x), for the 8 synthetic datasets. In Figure 7 it is possible to observe the clustering results for synthetic datasets 1 to 8, observing the original centroids (black hexagons) and those found by the TRUNC algorithm (red x).

When looking at these results, it is possible to observe that there are centroids of two different colors: in black are the centroids for the source domain, and the grey centroids represent those found during the experiments with the UNC and the proposed TRUNC algorithm. It is possible to observe that in all representations of the grouping results with the application of TRUNC there is an excellent approximation between the \mathcal{D}_S prototypes and those found in \mathcal{D}_T . With the

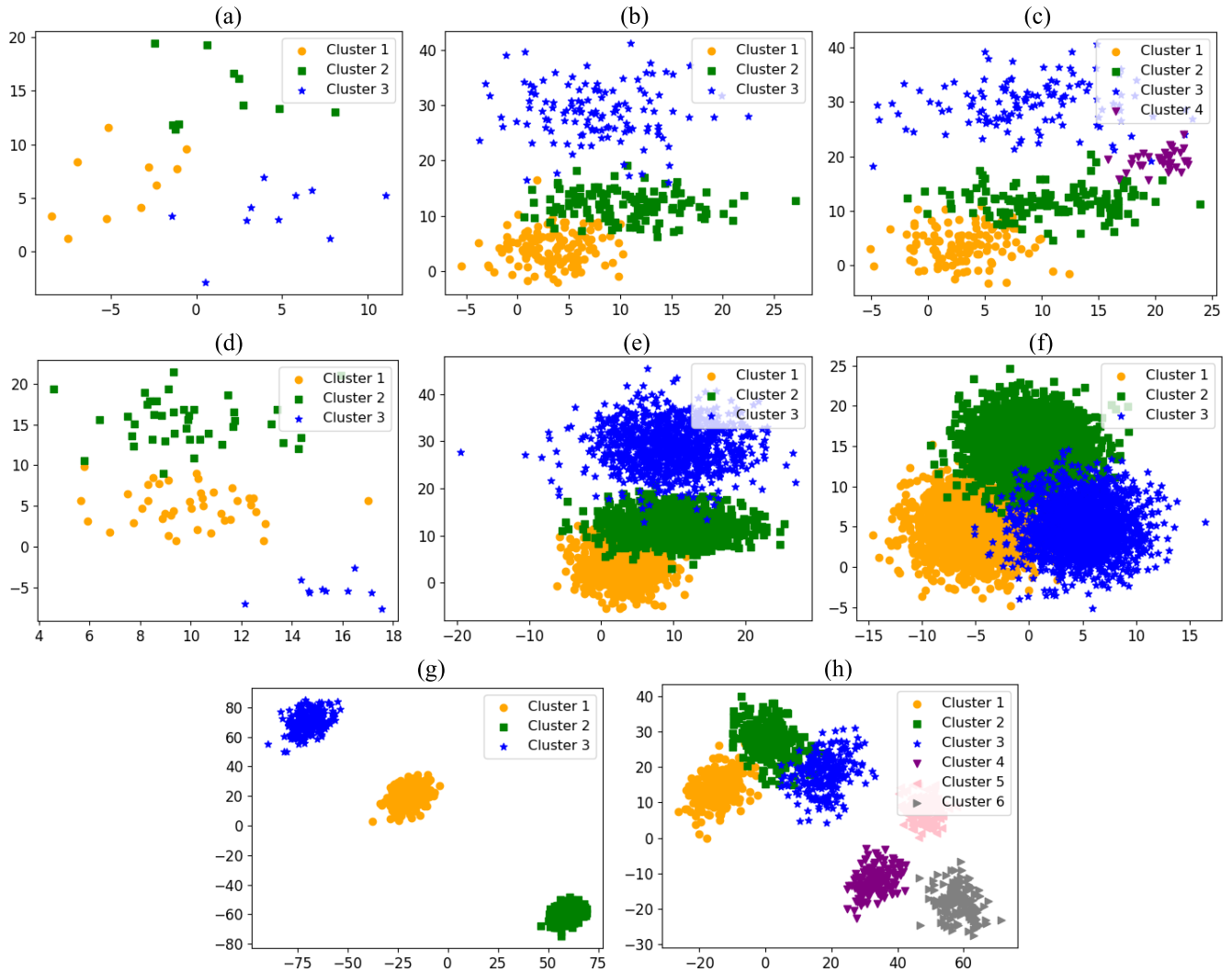


FIGURE 4. Synthetic datasets used for performance evaluation. $\mathcal{D}_{\mathcal{T}}$ data of synthetic datasets, (a) S1, (b) S2, (c) S3, (d) S4, (e) S5, (f) S6, (g) S7, (h) S8.

application of the UNC network, it is possible to see that there is a certain discrepancy between the real $\mathcal{D}_{\mathcal{S}}$ prototypes and the prototypes found by the standard UNC algorithm.

In relation to the performance measures, the UNC algorithm had a moderate performance for ACC, NMI, RI and Silhouette Score, but had a high computational cost measured by EC. K-Means had good performance for ACC, NMI, RI and ARI, but showed a decrease in clustering quality with its Silhouette Score results, presenting instability and variation in performance for complex data. DBSCAN presented uncompetitive results in terms of accuracy but could be useful for identifying dense clusters and noise without the need of multiple iterations. The DE-TRC algorithm presented reasonably accurate results, but had slightly greater limitations in simple scenarios, making it an algorithm with less variability and consistency compared with TRUNC, for example. TRUNC, on the other hand, presented the best performance of all, with consistent results for ACC, NMI, RI, ARI and

Silhouette Score, and had a low computational cost, with an average number of epochs for convergence between 1 and 4.

Figure 8 summarizes the results of all algorithms for all performance measures, showing the average values of the results grouped by evaluation measures. In this grouped bar chart, it can be seen that the algorithms are competing fiercely with each other, observing the bars that vary with each measurement result. However, it can also be seen that the TRUNC algorithm maintains the lead in most of the results collected.

In terms of iterations or epochs for convergence, the DBSCAN algorithm does not perform clustering with multiple traditional iterations, making it not directly comparable in terms of CE. UNC presented faster convergence in datasets such as 6 and 7 (10-11 epochs) and worse performance in datasets such as 8, reaching 55 epochs for the algorithm to converge. K-Means presented better average performance than UNC, but still with large variance, from 3 epochs (dataset 4) to 12 epochs (in dataset 3).

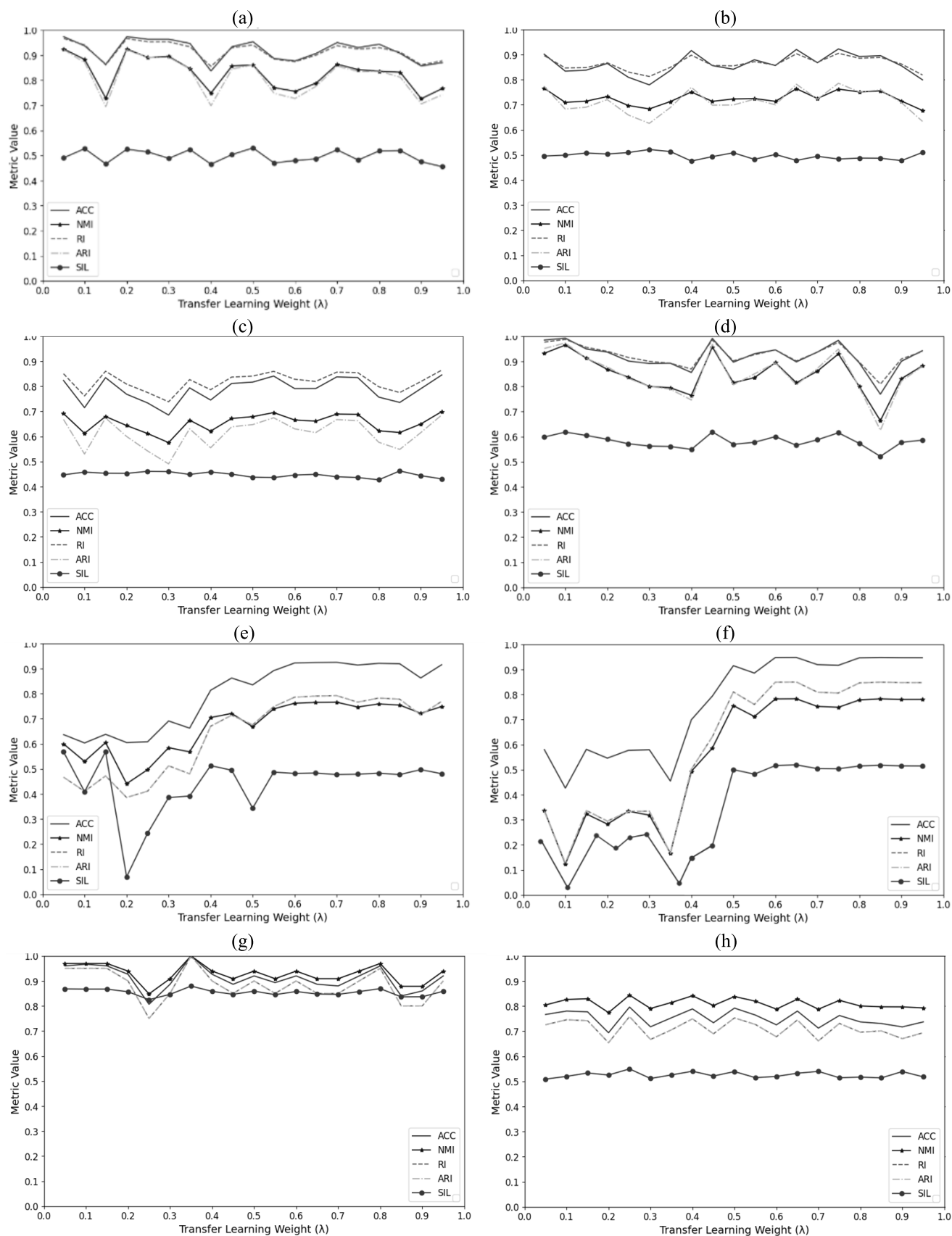


FIGURE 5. Sensitivity analysis for evaluation metrics for data of synthetic datasets, (a) D1, (b) D2, (c) D3, (d) D4, (e) D5, (f) D6, (g) D7, (h) D8.

TABLE 3. Evaluation results for synthetic datasets.

DATASETS	ACC (Av±Std.)				
	UNC	TRUNC	K-MEANS	DBSCAN	DE-TRC
1	0.83±0.00	1.00±0.00	0.86±0.11	0.64±0.01	0.80±0.00
2	0.66±0.00	0.96±0.00	0.89±0.00	0.47±0.00	0.85±0.00
3	0.68±0.00	0.85±0.00	0.86±0.00	0.32±0.00	0.78±0.00
4	0.67±0.00	1.00±0.00	0.95±0.00	0.56±0.00	0.96±0.00
5	0.65±0.00	0.93±0.00	0.92±0.00	0.33±0.00	0.87±0.00
6	0.84±0.00	0.94±0.00	0.93±0.00	0.34±0.00	0.94±0.00
7	0.94±0.00	0.98±0.00	0.70±0.00	0.86±0.00	0.33±0.00
8	0.54±0.00	0.99±0.00	0.98±0.00	0.53±0.00	0.97±0.00
DATASETS	NMI (Av±Std.)				
	UNC	TRUNC	K-MEANS	DBSCAN	DE-TRC
1	0.66±0.00	1.00±0.00	0.60±0.04	0.42±0.00	0.57±0.00
2	0.67±0.00	0.85±0.00	0.73±0.00	0.26±0.00	0.63±0.00
3	0.63±0.00	0.75±0.00	0.73±0.00	0.20±0.00	0.59±0.00
4	0.41±0.00	1.00±0.00	0.84±0.00	0.51±0.00	0.85±0.00
5	0.66±0.00	0.88±0.00	0.77±0.00	0.20±0.00	0.65±0.00
6	0.68±0.00	0.78±0.00	0.76±0.00	0.30±0.00	0.76±0.00
7	0.77±0.00	0.81±0.00	0.75±0.10	0.51±0.00	0.80±0.00
8	0.61±0.00	0.97±0.00	0.95±0.00	0.72±0.00	0.93±0.00
DATASETS	RI (Av±Std.)				
	UNC	TRUNC	K-MEANS	DBSCAN	DE-TRC
1	0.59±0.00	1.00±0.00	0.83±0.11	0.66±0.00	0.77±0.00
2	0.54±0.00	0.90±0.00	0.88±0.00	0.51±0.00	0.83±0.00
3	0.53±0.00	0.73±0.00	0.88±0.00	0.30±0.00	0.81±0.00
4	0.32±0.00	1.00±0.00	0.92±0.10	0.60±0.00	0.94±0.00
5	0.54±0.00	0.82±0.00	0.91±0.00	0.33±0.00	0.85±0.00
6	0.55±0.00	0.85±0.00	0.91±0.00	0.34±0.00	0.92±0.00
7	0.84±0.00	0.86±0.00	0.82±0.00	0.59±0.00	0.83±0.00
8	0.48±0.00	0.96±0.00	0.98±0.20	0.73±0.00	0.91±0.00
DATASETS	ARI (Av±Std.)				
	UNC	TRUNC	K-MEANS	DBSCAN	DE-TRC
1	0.85±0.00	0.91±0.00	0.62±0.00	0.29±0.00	0.49±0.00
2	0.86±0.00	0.89±0.00	0.73±1.00	0.20±0.00	0.63±0.00
3	0.89±0.00	0.91±0.00	0.71±0.10	0.31±0.00	0.56±0.00
4	0.92±0.00	0.97±0.00	0.84±0.00	0.28±0.00	0.87±0.00
5	0.89±0.00	0.90±0.00	0.80±0.00	0.34±0.00	0.66±0.00
6	0.85±0.00	0.95±0.00	0.81±0.00	0.44±0.00	0.83±0.00
7	0.94±0.00	0.96±0.00	0.56±0.00	0.49±0.00	0.85±0.00
8	0.75±0.00	0.97±0.00	0.94±0.00	0.41±0.00	0.93±0.00
DATASETS	Silhouette (Av±Std.)				
	UNC	TRUNC	K-MEANS	DBSCAN	DE-TRC
1	0.60±0.00	0.68±0.00	0.47±0.00	0.22±0.00	0.44±0.00
2	0.51±0.00	0.54±0.00	0.47±0.55	0.35±0.00	0.43±0.00
3	0.50±0.00	0.53±0.00	0.44±0.05	0.19±0.00	0.44±0.00
4	0.60±0.00	0.65±0.00	0.62±0.00	0.35±0.00	0.60±0.00
5	0.52±0.00	0.61±0.00	0.49±0.05	0.36±0.00	0.45±0.00
6	0.50±0.00	0.60±0.00	0.47±0.00	0.42±0.00	0.41±0.00
7	0.59±0.00	0.61±0.00	0.37±0.00	0.38±0.00	0.47±0.00
8	0.60±0.00	0.65±0.00	0.64±0.00	0.50±0.00	0.62±0.00
DATASETS	Epochs to Convergence (Av±Std.)				
	UNC	TRUNC	K-MEANS	DBSCAN	DE-TRC
1	51.00±0.00	3.00±0.00	4.00±0.90	-	2.00±0.00
2	49.00±0.00	2.00±0.00	6.00±0.35	-	4.00±0.00
3	46.00±0.00	1.00±0.00	12.00±0.00	-	2.00±0.00
4	30.00±0.00	2.00±0.00	3.00±0.07	-	3.00±0.00
5	41.00±0.00	3.00±0.00	7.00±0.00	-	4.00±0.00
6	10.00±0.00	2.00±0.00	5.00±0.00	-	3.00±0.00
7	11.00±0.00	1.00±0.00	6.00±0.00	-	2.00±0.00
8	55.00±0.00	4.00±0.00	4.00±0.20	-	5.00±0.00

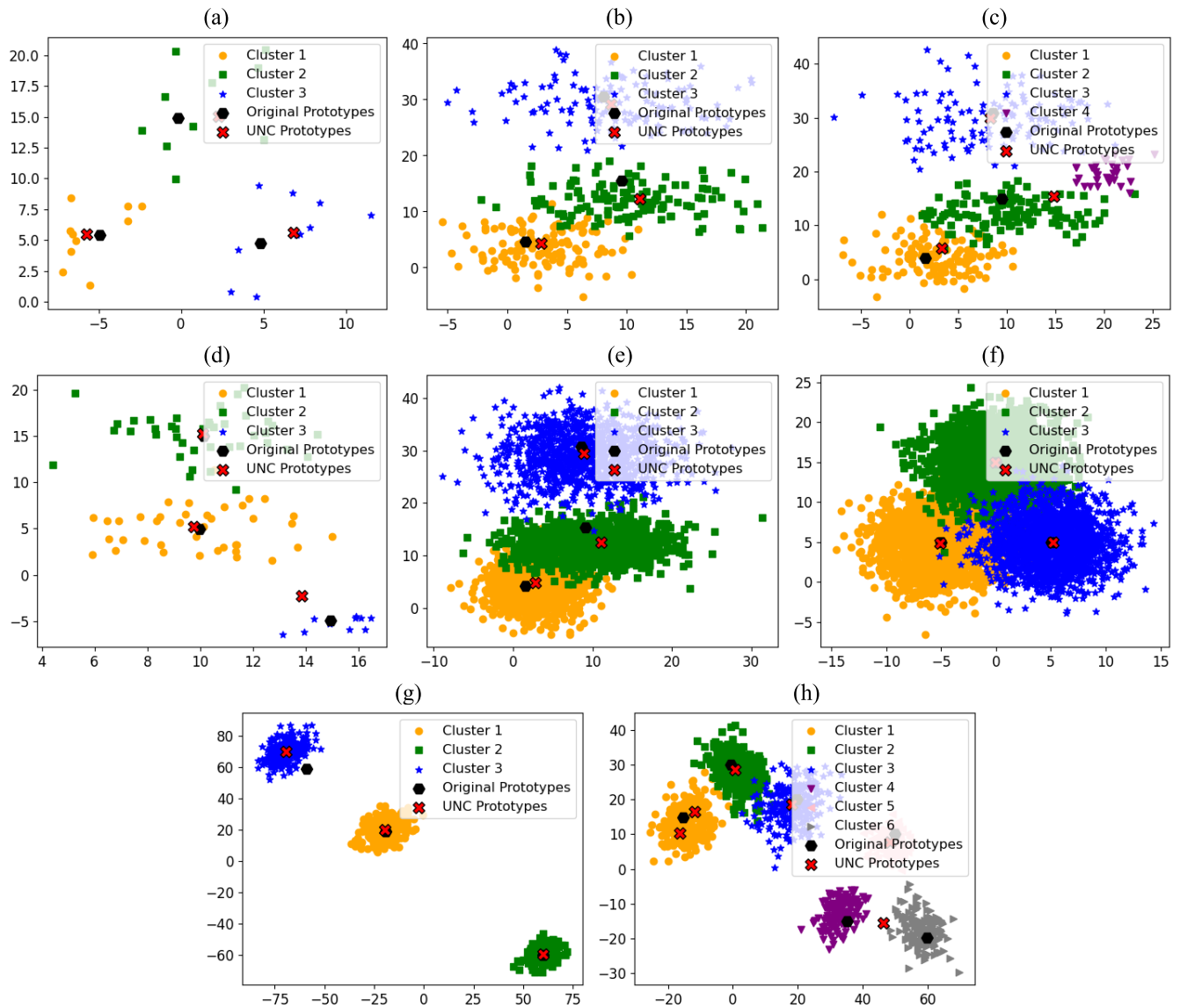


FIGURE 6. Clustering results of UNC implementation in synthetic datasets, (a) T1, (b) T2, (c) DT3, (d) T4, (e) T5, (f) T6, (g) T7, (h) T8.

The DE-TRC algorithm presented fast convergence in almost all scenarios, however, unlike the other metrics, in challenging scenarios, such as dataset 8, it obtained the worst performance. TRUNC had results that indicate faster convergence than its competitors, between 1 and 4, also presenting a small loss of performance in the scenario of dataset 8, with 4 epochs for the algorithm to converge. Despite this value, TRUNC is still the most computationally efficient among the algorithms tested.

It is also necessary to assess if there is a statistically significant difference among the results of Table 3, with the Wilcoxon signed-rank test. The results of the comparisons showed statistically significant differences between the algorithms, with p -values below 0.05.

In the comparison between TRUNC and UNC, TRUNC showed superior performance, with p -values. Compared to

K-Means, TRUNC was also superior, with p -values between 0.000 and 0.021, although some differences were moderate. Regarding DBSCAN, the results reinforce the advantage of TRUNC, with the lowest p -value in datasets 7 and significant values in datasets 2 and 4. Finally, in comparison with DE-TRC, TRUNC also demonstrated better performance, with more pronounced differences in datasets such as 4 and 7. In summary, the p -values found confirm that TRUNC presents statistically superior and consistent performance in relation to the other algorithms, evidencing its effectiveness and robustness for different dataset characteristics.

Overall, it is possible to observe that the proposed algorithm is more successful with datasets that present noise, such as datasets 2 and 3, which means that it shows quality in unbalanced environments, representing robustness in its

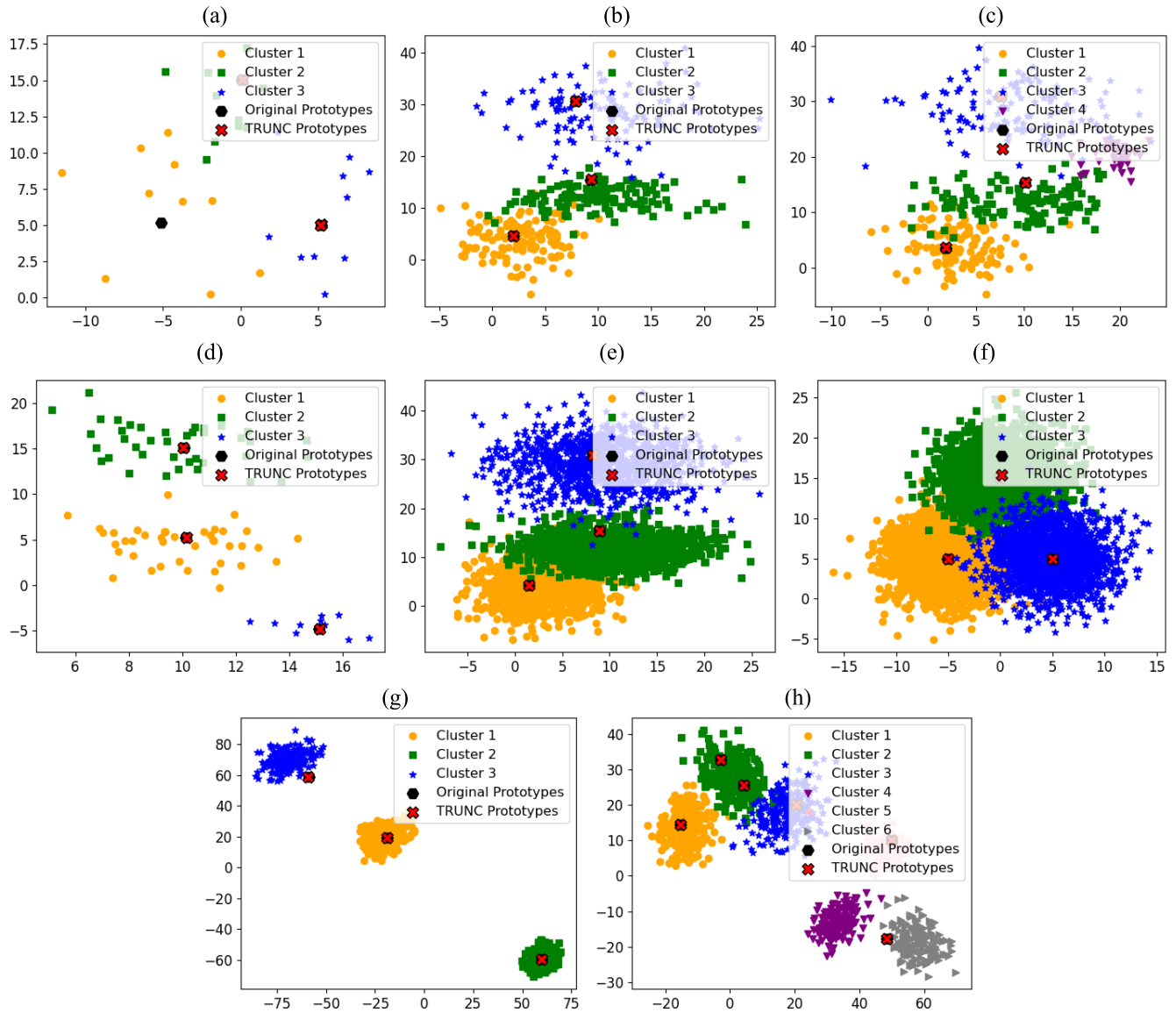


FIGURE 7. Clustering results of TRUNC implementation in synthetic datasets, (a) T1, (b) T2, (c) DT3, (d) T4, (e) T5, (f) T6, (g) T7, (h) T8.

performance. TRUNC also presented good clustering quality in datasets with fewer samples in D_S than in D_T , showing its effectiveness in transferring learning into this type of environment.

D. PERFORMANCE FOR THE REAL-WORLD DATASETS

For the experiments with real-world datasets, the ones used in [25] to assess their proposed algorithm (DE-TRC), which also implements PBTL with a bio-inspired approach (Differential Evolution), were chosen. This decision was made based on the similarity of the problem being solved by both algorithms, TRUNC and DE-TRC. The results of the clustering made by applying UNC and TRUNC to the aforementioned real-world datasets are presented in Table 4. Based on these

results, a grouped bar chart was created to show the efficiency of the algorithms (Figure 9).

In this chart, it is possible to observe, using the averages of the evaluation measures for the datasets, that TRUNC has the best performance, followed by DE-TRC with slightly lower performance. The UNC, K-Means and DBSCAN algorithms, respectively, have the worst performances for the real datasets.

About the epochs for convergence, for the Iris dataset, TRUNC and DE-TRC converged quickly, both in 2 epochs, while K-Means took 4 epochs, and UNC required 17 epochs. For the Breast Cancer Wisconsin, TRUNC was the fastest, converging in 1 epoch, followed by DE-TRC, K-Means, and UNC. For the Wine dataset, TRUNC was the most efficient, followed by DE-TRC, K-MEANS, and UNC. For

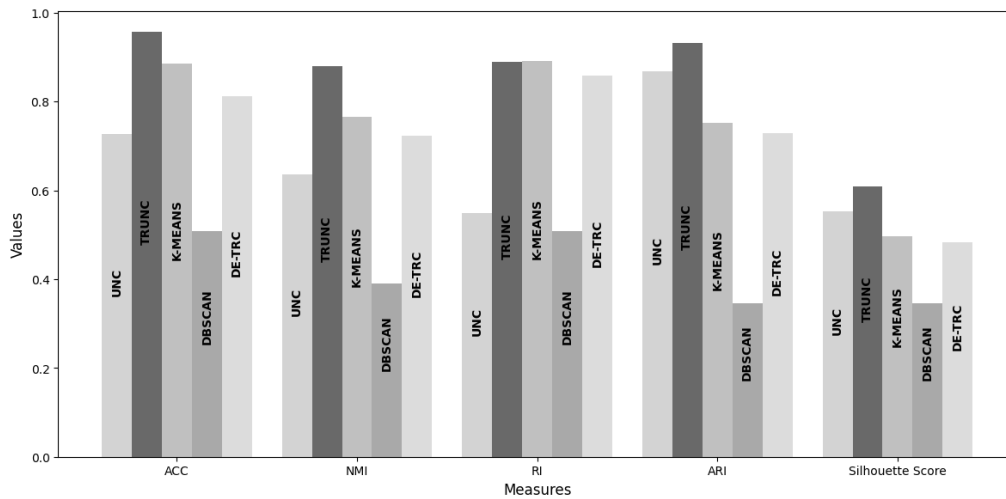


FIGURE 8. Average measures' values for the synthetic datasets.

Haberman's Survival, TRUNC and K-Means were the fastest, with DE-TRC being slightly slower and UNC taking the longest. Finally, for Wireless Indoor Localization, DE-TRC was the fastest, followed by K-Means, TRUNC, and UNC.

The general performance analysis of the TRUNC, DE-TRC, K-Means, UNC and DBSCAN algorithms for the different datasets shows that TRUNC was the most consistent algorithm, standing out in most cases, with good results in metrics such as RI, NMI, ACC and Silhouette Score, and proved to be efficient for various types of datasets. DE-TRC and K-Means performed well, especially on some datasets, but did not consistently outperform TRUNC. UNC, while performing reasonably well, lagged behind TRUNC and DE-TRC on most metrics. DBSCAN performed the worst in most cases, especially on more complex datasets, with the exception of Wireless Indoor Localization, where it performed reasonably well.

In terms of convergence, TRUNC was the fastest algorithm on several datasets, except for the Wireless Indoor Localization data, where DE-TRC was faster. UNC was the algorithm that took the longest number of epochs to converge, especially on datasets such as Breast Cancer Wisconsin and Wine.

As with the synthetic data, the Wilcoxon Signed-Rank Test was also applied with a significance threshold of 0.05. The results of the Wilcoxon Signed-Rank test suggested that, for most of the data sets and measures evaluated, there is no significant difference between the other algorithms and TRUNC. This is because the results for the p -values were all greater than 0.05, meaning that there is no statistically significant difference between the results.

VI. DISCUSSION AND FUTURE TRENDS

This work addressed the challenge of improving clustering performance in unsupervised learning tasks by leveraging

transfer learning. Clustering, a fundamental task in unsupervised machine learning, often suffers from issues such as noise, imbalanced data, and limited data availability. Traditional methods may not adequately handle these complexities, necessitating the development of more robust techniques. The motivation behind this study was to integrate transfer learning with bioinspired algorithms to enhance their effectiveness in clustering tasks, ultimately aiming to solve complex problems with better efficiency and accuracy, and smaller convergence time.

In this study, we proposed TRUNC (Transfer Learning-Based Unsupervised Network for Data Clustering), a novel bioinspired algorithm that incorporates transfer learning into a single-layer feed-forward winner-takes-all neural network. The key contributions of TRUNC include its ability to use knowledge from a source domain to improve clustering performance in a target domain, a detailed sensitivity analysis of the transfer learning parameter λ , and a comprehensive evaluation using both synthetic and real-world datasets.

The experimental results showed that TRUNC outperforms traditional unsupervised neural networks in various scenarios. Specifically, TRUNC showed significant improvements in Accuracy, Normalized Mutual Information, Rand Index, Adjusted Rand Index and Silhouette Score across synthetic datasets with different complexities. TRUNC also had better performance for most measures and datasets when compared to some traditional algorithms like K-Means and DBSCAN, specially with the last one. DE-TRC had a better performance in some datasets, but TRUNC outperformed it in EC, showing that TRUNC converged in less time than DE-TRC. For the real-world datasets, TRUNC achieved superior convergence times and maintained competitive clustering performance.

With the experiments performed and results collected, especially when we analyze the epochs for convergence

TABLE 4. Evaluation metrics results for real-world datasets.

DATASETS	ACC (Av±Std.)				
	UNC	TRUNC	K-MEANS	DBSCAN	DE-TRC
Iris	0.80±0.00	0.85±0.00	0.42±0.05	0.54±0.00	0.83±0.00
B. C. Wis.	0.85±0.00	0.94±0.00	0.42±0.00	0.51±0.00	0.91±0.00
Wine	0.70±0.00	0.92±0.00	0.35±0.02	0.56±0.00	0.81±0.00
Hab. Surv.	0.62±0.00	0.74±0.00	0.50±0.02	0.51±0.00	0.71±0.00
W. Ind. Loc.	0.94±0.00	0.95±0.00	0.45±0.00	0.69±0.00	0.82±0.00
DATASETS	NMI (Av±Std.)				
	UNC	TRUNC	K-MEANS	DBSCAN	DE-TRC
Iris	0.70±0.00	0.82±0.00	0.75±0.00	0.60±0.00	0.65±0.00
B. C. Wis.	0.46±0.00	0.59±0.00	0.46±0.00	0.20±0.00	0.55±0.00
Wine	0.40±0.00	0.91±0.00	0.42±0.00	0.41±0.00	0.87±0.00
Hab. Surv.	0.41±0.00	0.68±0.00	0.39±0.00	0.51±0.00	0.60±0.00
W. Ind. Loc.	0.88±0.00	0.92±0.00	0.88±0.00	0.42±0.00	0.80±0.00
DATASETS	RI (Av±Std.)				
	UNC	TRUNC	K-MEANS	DBSCAN	DE-TRC
Iris	0.64±0.00	0.87±0.00	0.87±0.00	0.73±0.00	0.83±0.00
B. C. Wis.	0.49±0.00	0.89±0.00	0.75±0.00	0.52±0.00	0.84±0.00
Wine	0.37±0.00	0.89±0.00	0.71±0.00	0.38±0.00	0.85±0.00
Hab. Surv.	0.63±0.00	0.72±0.00	0.49±0.00	0.61±0.00	0.49±0.00
W. Ind. Loc.	0.87±0.00	0.98±0.00	0.95±0.00	0.38±0.00	0.84±0.00
DATASETS	ARI (Av±Std.)				
	UNC	TRUNC	K-MEANS	DBSCAN	DE-TRC
Iris	0.66±0.00	0.77±0.00	0.73±0.00	0.47±0.00	0.62±0.00
B. C. Wis.	0.49±0.00	0.69±0.00	0.49±0.00	0.21±0.00	0.67±0.00
Wine	0.38±0.00	0.79±0.00	0.37±0.00	0.40±0.00	0.79±0.00
Hab. Surv.	0.58±0.00	0.63±0.00	0.50±0.00	0.43±0.00	0.40±0.00
W. Ind. Loc.	0.88±0.00	0.88±0.00	0.88±0.00	0.41±0.00	0.71±0.00
DATASETS	Average Silhouette Score (Av±Std.)				
	UNC	TRUNC	K-MEANS	DBSCAN	DE-TRC
Iris	0.60±0.00	0.68±0.00	0.55±0.00	0.40±0.00	0.45±0.00
B. C. Wis.	0.59±0.00	0.66±0.00	0.69±0.00	0.27±0.00	0.34±0.00
Wine	0.50±0.00	0.62±0.00	0.57±0.00	0.39±0.00	0.38±0.00
Hab. Surv.	0.58±0.00	0.61±0.00	0.49±0.10	0.58±0.00	0.27±0.00
W. Ind. Loc.	0.41±0.00	0.41±0.00	0.41±0.00	0.50±0.00	0.29±0.00
DATASETS	Epochs to Convergence (Av±Std.)				
	UNC	TRUNC	K-MEANS	DBSCAN	DE-TRC
Iris	17.00±0.00	2.00±0.00	4.00±0.00	-	2.00±0.00
B. C. Wis.	76.00±0.00	1.00±0.00	6.00±0.00	-	3.00±0.00
Wine	71.00±0.00	4.00±0.00	6.00±0.10	-	5.00±0.00
Hab. Surv.	60.00±0.00	6.00±0.00	6.00±0.00	-	8.00±0.00
W. Ind. Loc.	60.00±0.00	8.00±0.00	7.00±0.00	-	6.00±0.00

measure, it is possible to observe that TRUNC has a substantial improvement in computational efficiency compared to competing algorithms. Despite having tied results with the DE-TRC algorithm, there is still a faster and more consistent performance with the TRUNC algorithm, which represents less computational cost in its processing.

Although the results of the TRUNC algorithm are higher in absolute terms, there is still a need to conduct experiments with other datasets from the literature to understand which types of datasets would have a significant performance improvement. These findings confirm that the integration of transfer learning into the clustering process can lead to more effective and efficient solutions, validating the robustness and generalization capability of TRUNC. Some limitations of the TRUNC algorithm are the need for future

improvements to apply it in text mining, image segmentation and others.

Overall, TRUNC represents an advancement in the use of transfer learning for unsupervised clustering tasks, with potential for broad applicability in various fields. Future research directions include exploring the integration of concepts like prototype-based transfer with other bioinspired algorithms, such as particle swarm optimization and evolutionary algorithms, to further enhance its performance. Additionally, applying these concepts to other domains, such as image segmentation and text clustering, could provide valuable insights into its versatility. Further work could also focus on optimizing the transfer learning parameter λ and developing adaptive mechanisms to automatically tune this parameter based on dataset characteristics. The use of

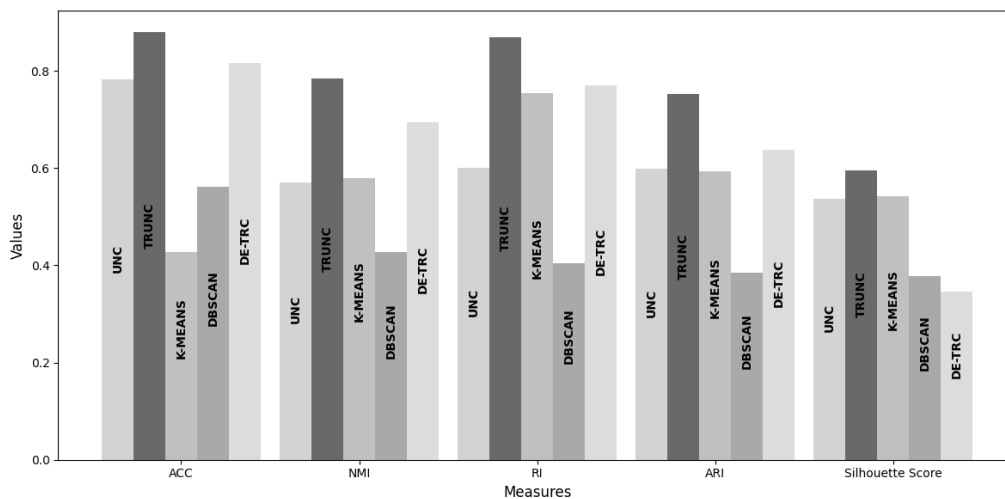


FIGURE 9. Average measures' values for real-world datasets.

Explainable AI in the context of PBTL also seems to be an interesting future path for research.

REFERENCES

- [1] C. M. Bishop and N. M. Nasrabadi, *Pattern Recognition and Machine Learning*, vol. 4, no. 4., New York, NY, USA: Springer, 2006.
- [2] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *J. Big Data*, vol. 3, pp. 1–40, Dec. 2016.
- [3] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Jan. 2009.
- [4] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, 1999.
- [5] D. Xu and Y. Tian, "A comprehensive survey of clustering algorithms," *Ann. Data Sci.*, vol. 2, no. 2, pp. 165–193, Jun. 2015.
- [6] R. Ribani and M. Marengoni, "A survey of transfer learning for convolutional neural networks," in *Proc. 32nd SIBGRAPI Conf. Graph., Patterns Images Tuts. (SIBGRAPI-T)*, Oct. 2019, pp. 47–57.
- [7] L. Decastro, "Fundamentals of natural computing: An overview," *Phys. Life Rev.*, vol. 4, no. 1, pp. 1–36, Mar. 2007.
- [8] R. Xavier and L. N. D. Castro, "On the use of evolutionary and swarm intelligence algorithms in transfer learning approaches: A review," *Int. J. Biosensors Bioelectronics*, vol. 8, no. 2, pp. 58–64, Dec. 2023.
- [9] O. G. Aliu, A. Imran, M. A. Imran, and B. Evans, "A survey of self organisation in future cellular networks," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 1, pp. 336–361, 1st Quart., 2013.
- [10] T. D. Sanger, "Optimal unsupervised learning in a single-layer linear feedforward neural network," *Neural Netw.*, vol. 2, no. 6, pp. 459–473, Jan. 1989.
- [11] M. I. V. Furtado, *Redes Neurais Artificiais: Uma Abordagem Para Sala De Aula*. Ponta Grossa, Brazil: Ponta Grossa, PR: Atena Editora, 2019, p. 19.
- [12] P. V. Klaine, M. A. Imran, O. Onireti, and R. D. Souza, "A survey of machine learning techniques applied to self-organizing cellular networks," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2392–2431, 4th Quart., 2017.
- [13] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Upper Saddle River, NJ, USA: Prentice-Hall, 1988.
- [14] R. Linden, "Técnicas de agrupamento," *Revista de Sistemas de Informação da FSMA*, vol. 4, no. 4, pp. 18–36, 2009.
- [15] Y. Liu, Z. Li, H. Xiong, X. Gao, and J. Wu, "Understanding of internal clustering validation measures," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2010, pp. 911–916.
- [16] M. C. Naldi, "Técnicas de combinação para agrupamento centralizado e distribuído de dados," Ph.D. dissertation, Universidade de São Paulo, 2011.
- [17] L. N. De Castro and D. G. Ferrari, *Introdução à Mineração de Dados*. Sao Paulo, Brazil: Saraiva Educação S.A., 2017.
- [18] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proc. IEEE*, vol. 109, no. 1, pp. 43–76, Jan. 2021.
- [19] Z. Deng, Y. Jiang, F.-L. Chung, H. Ishibuchi, K.-S. Choi, and S. Wang, "Transfer prototype-based fuzzy clustering," *IEEE Trans. Fuzzy Syst.*, vol. 24, no. 5, pp. 1210–1232, Oct. 2016.
- [20] M. Jiang, Z. Wang, L. Qiu, S. Guo, X. Gao, and K. C. Tan, "A fast dynamic evolutionary multiobjective algorithm via manifold transfer learning," *IEEE Trans. Cybern.*, vol. 51, no. 7, pp. 3417–3428, Jul. 2021.
- [21] X. Zhang, Y. Jin, and F. Qian, "A self-adaptive dynamic multi-objective optimization algorithm based on transfer learning and elitism-based mutation," *Neurocomputing*, vol. 559, Nov. 2023, Art. no. 126761.
- [22] X. Ma, Q. Chen, Y. Yu, Y. Sun, L. Ma, and Z. Zhu, "A two-level transfer learning algorithm for evolutionary multitasking," *Frontiers Neurosci.*, vol. 13, p. 1408, Jan. 2020.
- [23] L. Jiao, F. Wang, Z.-G. Liu, and Q. Pan, "TECM: Transfer learning-based evidential c-means clustering," *Knowl.-Based Syst.*, vol. 257, Dec. 2022, Art. no. 109937.
- [24] C. Medina, A. Devos, and M. Grossglauer, "Self-supervised prototypical transfer learning for few-shot classification," 2020, *arXiv:2006.11325*.
- [25] F. Zhao, C. Wang, and H. Liu, "Differential evolution-based transfer rough clustering algorithm," *Complex Intell. Syst.*, vol. 9, no. 5, pp. 5033–5047, Oct. 2023.
- [26] S. Aeberhard and M. Forina, *Wine*. Irvine, CA, USA: UCI Machine Learning Repository, 1991.
- [27] R. Fisher, *Iris*. Irvine, CA, USA: UCI Machine Learning Repository, 1988.
- [28] W. H. Wolberg, W. N. Street, and O. L. Mangasarian, *Breast Cancer Wisconsin (Diagnostic) Dataset*. Irvine, CA, USA: UCI Machine Learning Repository, 1992.
- [29] S. Haberman, *Haberman's Survival*. Irvine, CA, USA: UCI Machine Learning Repository, 1999, pp. 10–51.
- [30] R. Bhatt, *Wireless Indoor Localization*. Irvine, CA, USA: UCI Machine Learning Repository, 2017, pp. 20–55.
- [31] J. Ranstam, L. Ryd, and I. Önsten, "Accurate accuracy assessment: Review of basic principles," *Acta Orthopaedica Scandinavica*, vol. 70, no. 4, pp. 319–321, Jan. 1999.
- [32] T. Kålseth, "On normalized mutual information: Measure derivations and properties," *Entropy*, vol. 19, no. 11, p. 631, Nov. 2017.
- [33] E. Hullermeier, M. Rifqi, S. Henzgen, and R. Senge, "Comparing fuzzy partitions: A generalization of the Rand index and related measures," *IEEE Trans. Fuzzy Syst.*, vol. 20, no. 3, pp. 546–556, Jun. 2012.
- [34] S. Biometrika, "The probable error of a mean," *Biometrika*, vol. 6, no. 1, p. 1, Mar. 1908.
- [35] F. Wilcoxon, "Individual comparisons by ranking methods," in *Breakthroughs in Statistics: Methodology and Distribution*. New York, NY, USA: Springer, 1992, pp. 196–202.

- [36] B. Rosner, R. J. Glynn, and M. T. Lee, "The Wilcoxon signed rank test for paired comparisons of clustered data," *Biometrics*, vol. 62, no. 1, pp. 185–192, Mar. 2006.
- [37] M. Cui, "Introduction to the k-means clustering algorithm based on the elbow method," *Accounting, Auditing Finance*, vol. 1, no. 1, pp. 5–8, 2020.
- [38] J. A. Hartigan and M. A. Wong, "A K-means clustering algorithm," *Appl. Statist.*, vol. 28, no. 1, pp. 100–108, 1979.
- [39] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. KDD*, vol. 96, no. 34, Aug. 1996, pp. 226–231).
- [40] J. M. Santos and M. Embrechts, "On the use of the adjusted Rand index as a metric for evaluating supervised classification," in *Proc. Int. Conf. Artif. Neural Netw.* Berlin, Germany: Springer, Sep. 2009, pp. 175–184.
- [41] K. R. Shahapure and C. Nicholas, "Cluster quality analysis using silhouette score," in *Proc. IEEE 7th Int. Conf. Data Sci. Adv. Anal. (DSAA)*, Oct. 2020, pp. 747–748.
- [42] K. Zhou, M. Guo, and A. Martin, "Evidential prototype-based clustering based on transfer learning," *Int. J. Approx. Reasoning*, vol. 151, pp. 322–343, Dec. 2022.

RITA XAVIER received the B.A. degree in social communication and audiovisual from the Methodist University of Piracicaba (UNIMEP), São Paulo, Brazil, in 2013, and the B.Sc. degree in information security from the School of Technology of Americana (FATEC), São Paulo, in 2018. She is currently pursuing the master's degree. From 2015 to 2018, she was a Research Assistant in quantum computing and literacy research with the School of Technology of Americana. She is a Research Assistant in natural computation and new machine learning paradigms with the School of Technology, Universidade Estadual de Campinas (FT-UNICAMP).

JOHN PELLER received the B.Sc. degree in biology from the Davidson College, North Carolina. He is currently pursuing the degree in software engineering with Florida Gulf Coast University, Fort Myers, FL, USA, with a focus on computer science and data science. Since 2023, he has been with the University as a Student Researcher, exploring the application of data gaps analysis to water quality monitoring. He has been involved as a Contributor in research projects encompassing machine learning techniques, natural language processing, and geospatial analysis. His research interests include computational statistics, natural computing, machine learning, and data mining.

LEANDRO NUNES DE CASTRO received the B.Sc. degree in electrical engineering from the Federal University of Goiás, in 1996, the M.Sc. and Ph.D. degrees in computer engineering from UNICAMP, in 1998 and 2001, respectively, and the M.B.A. degree in strategic business management from the Catholic University of Santos, in 2008. He was a Research Associate with the Computing Laboratory, University of Kent, Canterbury, from 2001 to 2002, a Visiting Professor with Malaysian Technological University, in 2005, a Visiting Professor with UNICAMP, in 2012, and a Visiting Research Professor with the University of Salamanca, in 2014. He was an Assistant Professor with the Master's Program in Informatics, Unisantos, from 2003 to 2008, an Associate Professor with the Graduate Program in Electrical Engineering and Computing, Mackenzie Presbyterian University, from 2008 to 2022, and the Chief Executive Officer of the Center for Research, Development, and Innovation of AI Applied to Health (In.lab), from 2022 to 2023. He is currently a Visiting Associate Professor with the School of Technology, UNICAMP, a Principal Investigator with the Center of Science, Technology, and Development for Innovation in Medicine and Health, University of São Paulo, and a Full Professor with Florida Gulf Coast University. His research interests include artificial intelligence, natural computing, and machine learning, with applications in data science and optimization. He was the Founding Editor-in-Chief of *International Journal of Natural Computing Research*, from 2010 to 2015.

...