applied sciences

MDPI

Article

Soft Semi-Supervised Deep Learning-Based Clustering

Mona Suliman AlZuhair *, Mohamed Maher Ben Ismail and Ouiem Bchir

Department of Computer Science, College of Computer and Information Sciences, King Saud University, Riyadh 11362, Saudi Arabia; mbenismail@ksu.edu.sa (M.M.B.I.); obchir@ksu.edu.sa (O.B.)
* Correspondence: 439204075@student.ksu.edu.sa

Abstract: Semi-supervised clustering typically relies on both labeled and unlabeled data to guide the learning process towards the optimal data partition and to prevent falling into local minima. However, researchers' efforts made to improve existing semi-supervised clustering approaches are relatively scarce compared to the contributions made to enhance the state-of-the-art fully unsupervised clustering approaches. In this paper, we propose a novel semi-supervised deep clustering approach, named Soft Constrained Deep Clustering (SC-DEC), that aims to address the limitations exhibited by existing semi-supervised clustering approaches. Specifically, the proposed approach leverages a deep neural network architecture and generates fuzzy membership degrees that better reflect the true partition of the data. In particular, the proposed approach uses side-information and formulates it as a set of soft pairwise constraints to supervise the machine learning process. This supervision information is expressed using rather relaxed constraints named "should-link" constraints. Such constraints determine whether the pairs of data instances should be assigned to the same or different cluster(s). In fact, the clustering task was formulated as an optimization problem via the minimization of a novel objective function. Moreover, the proposed approach's performance was assessed via extensive experiments using benchmark datasets. Furthermore, the proposed approach was compared to relevant state-of-the-art clustering algorithms, and the obtained results demonstrate the impact of using minimal previous knowledge about the data in improving the overall clustering performance.

Keywords: deep clustering; semi-supervised clustering; soft constraints; fuzzy clustering

Citation: AlZuhair, M.S.; Ben Ismail, M.M.; Bchir, O. Soft Semi-Supervised Deep Learning-Based Clustering. *Appl. Sci.* **2023**, *13*, 9673. https://doi.org/10.3390/app13179673

Academic Editor: Wenjie Zhang

Received: 18 July 2023
Revised: 20 August 2023
Accepted: 23 August 2023
Published: 27 August 2023

1. Introduction

Supervised machine learning is an extensively studied and applied topic in the artificial intelligence field [1–3]. This learning paradigm is a powerful tool for data classification using machine language. However, supervised learning requires the availability of accurately labeled datasets to infer learning models [4]. Despite the learning capability of the state-of-the-art supervised machine learning algorithms, the continuously growing size of the datasets used to train the models has made the labeling an even more costly process in terms of time and effort. In contrast, when only a relatively small dataset is available, it typically becomes prone to overfitting the training set. This disadvantage of supervised learning has promoted research on unsupervised machine learning which does not require labeled data for the training stage. In fact, unsupervised machine learning can cope better with a huge amount of unlabeled data as it eliminates the requirement of carefully annotating the training sets. Clustering is an unsupervised learning technique that is meant to draw conclusions and discover the hidden patterns in unlabeled datasets.

Clustering has been adapted to address various problems in many areas, such as data mining, pattern recognition, and computer vision [5]. In many fields, there are obvious benefits to be acquired from grouping instances that share similar properties into different clusters in an unsupervised manner [6]. Clustering algorithms have been extensively studied, as a descriptive data mining tool, from various aspects including similarity measures, feature selection, and grouping methods [7]. Among the well-known

and widely used clustering algorithms, one can cite k-means [8], hierarchical clustering [9], and Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [10]. The main concept of DBSCAN is that for a data object to belong to a cluster, the density in a neighborhood for that data object should be high enough, satisfying a user-specified density threshold [11]. This threshold is represented by two hyper-parameters; the radius of the neighborhood (eps) and the minimum number of points required to from the neighborhood (MinPts). Despite DBSCAN algorithm's robustness to outliers and capability of handling irregularly shaped clusters, it deteriorates when dealing with high dimensional data and is sensitive to the input parameters [12].

The typical goal of clustering is to group sets of data objects together in a way that instances assigned to the same cluster are more similar to each other compared to those belonging to other clusters. In fact, grouping similar data entities together is intended to discover the data partition and provide insights on the patterns underlying the different categories. Commonly, for a clustering algorithm to achieve the data mining goal, it should be preceded by a feature extraction stage that is intended to encode and extract the data properties that can better discriminate between the hidden clusters. However, this feature extraction phase may yield a highly dimensional data representation and a curse of dimensionality problem [13].

To meet the curse of dimensionality challenges, several dimensionality reduction methods [14,15] have been introduced to transform the data from the original feature space into a new feature space with fewer dimensions. However, despite these contributions, the data that exhibit a highly complex latent structure remain challenging to cluster using the existing clustering methods [5].

The emergence and rapid development of deep neural networks (DNNs) has triggered revisiting the clustering-related research. The rationale was to exploit the deep neural networks' ability to automatically determine the most relevant features and relax the need for feature handcrafting and engineering [5,16]. Moreover, the researchers' interest in deep clustering paradigms was intended to address challenges such as the inability to handle datasets that lie on nonlinear manifolds, the curse of dimensionality, and the sensitivity to noise [17,18]. Basically, the earliest deep clustering works [19,20] focused on feature transformation and clustering as two independent processes. In other words, the data were first mapped into a new feature space and then fed into a clustering algorithm. Recently, deep clustering has been adapted to jointly perform feature learning, transformation, and the clustering of data that exhibit a highly complex latent structure [5]. The fundamental component of deep clustering approaches relies on deep neural network architectures such as autoencoders [21], the network loss, and the clustering loss optimization [7].

Deep clustering potentials were mainly explored in the context of entirely unsupervised learning. However, the datasets are naturally weak or poorly labeled in the real world [22]. This boosted the researchers' efforts to exploit this knowledge to inject some supervision to guide the clustering process, i.e., introduce the semi-supervised learning paradigm. Semi-supervised learning uses both labeled and unlabeled data to train a model. Moreover, semi-supervised learning exploits prior knowledge and formulates it as constraints to ease the learning process. Despite researchers' efforts, the state-of-the-art, semi-supervised, deep clustering approaches remain far below expectations compared to the fully unsupervised approaches. In particular, the use of fuzzy logic was not investigated to represent the data partition in the context of semi-supervised deep learning. Moreover, the existing semi-supervised clustering methods only proposed the integration of the supervision information as crisp pairwise constraints rather than soft ones.

In this research, we propose a novel semi-supervised deep clustering approach, named Soft Constrained Deep Clustering (SC-DEC), to overcome the limitations in current approaches. The proposed approach leverages a deep neural network architecture for feature learning and performs clustering with fuzzy membership degrees to better represent the true partition of the data. Specifically, we formulated the deep clustering problem as an optimization of a novel objective function. This function is designed to simultaneously

discover the hidden data clusters and optimize the deep neural network. Moreover, soft pairwise constraints were incorporated within the objective function to represent the available supervision knowledge. These constraints were formulated in a relaxing way in which the compliance to a constraint is not strictly obligated, which makes the proposed approach more suitable for real-world data clustering applications where the available side information is not mature enough to be strictly imposed. Additionally, the fuzzy-based representation of data partition was adopted in the proposed method to reflect the grouping of data in a more accurate way.

The main contributions of this research can be summarized as follows: (i) We formulated the proposed semi-supervised deep clustering approach as an optimization problem and designed the objective functions that carry the proposed tasks to simultaneously learn a discriminative embedded representation of the original data along with the optimal data clusters. (ii) We solved the formulated optimization problem and derived updated equations of the respective parameters. (iii) We designed and implemented a novel deep semi-supervised clustering algorithm using state-of-the-art technology, platforms, and tools. (iv) We evaluated the performance of the proposed approach using real datasets and standard performance measures in addition to an objective comparison with relevant state-of-the-art approaches.

The rest of this paper is organized as follows: Section 2 presents the related works while Section 3 introduces the proposed method. The experimental settings as well as the analysis and discussion of the results are presented in Section 4. Finally, the conclusion and future work are discussed in Section 5.

2. Related Works

Research work on deep clustering includes deep embedded clustering and semi-supervised deep embedded clustering (SS-DEC) approaches. For deep embedded clustering, Deep Neural Networks (DNNs) have been used for dimensionality reduction to learn a clustering-oriented representation that favors the clustering tasks. In fact, different deep clustering frameworks were created on top of various DNN architectures. Namely, autoencoders (AEs), Deep Belief Networks (DBNs) [23], Convolutional Neural Networks (CNNs) [24,25], and Generative Adversarial Networks (GANs) [26] have been introduced and used in various deep clustering applications. In particular, autoencoders have been widely adapted to address challenges relevant to the deep clustering architectures [21,27–29]. A recent survey [30] distinguishes deep clustering methods in terms of methodology, prior knowledge, and architecture. Specifically, the authors listed semi-supervised deep clustering as one of the main four categories of deep clustering methods.

For AE-based deep clustering approaches, the encoder layers of the trained autoencoder are the ones leveraged for feature transformation into a lower-dimensional space, which serves as the input for the clustering algorithm. Recently, an AE-based deep clustering architecture, deep embedded clustering (DEC) [21], was proposed, which was then followed by a number of variants [19,27–29,31–36] that used DEC as the basis for their framework. From then, deep clustering has become a growing research field, with DEC [21] being the experimental benchmark for many deep clustering approaches [29,34].

For DEC [21], the DNN architecture consists mainly of an autoencoder that is used to automatically learn the feature representations via nonlinear embedding into a lower-dimensional feature space. After the autoencoder is trained and its parameters are initialized, it is finetuned further to minimize the reconstruction loss. Only the encoder layers are kept and used as the initial mapping function between the data space and the embedded feature space. To initialize the cluster centers, the original data are passed through the initialized encoder to obtain the embedded data points and perform standard k-means clustering in the embedded feature space to obtain k initial centroids and memberships. After that, DEC updates its parameters by iterating between computing an auxiliary target distribution and minimizing the following objective function in form of the

Kullback–Leibler (KL) divergence from the computed soft assignments to the computed auxiliary target distribution:

$$L = KL(P||Q) = \sum_i \sum_j p_{ij} log \frac{P_{ij}}{q_{ij}} \quad (1)$$

where Q is the soft assignments of the data points and P is the computed auxiliary distribution.

In the above equation, q_{ij} is a centroid-based probability distribution (i.e., soft assignment) that represents the probability of assigning sample i to cluster j, and it is computed using the Student's t-distribution. Moreover, p_{ij} in Equation (1) is the auxiliary target distribution that q_{ij} is matched to by minimizing a KL divergence metric between the two distributions. Further elaboration on the computation of q_{ij} and p_{ij} can be found in Equations (3) and (4) below.

This objective function in Equation (1) is minimized using a Stochastic Gradient Descent (SGD) to learn cluster centers and DNN parameters from the embedded space Z. Lastly, a cluster assignment hardening loss is applied to obtain the soft assignment probabilities of the embedded points. DEC reported an 84.3% clustering accuracy on the MNIST images dataset and 75.63% on the REUTERS text dataset, outperforming several state-of-the-art spectral clustering-based algorithms.

Motivated by the need to address certain limitations, other deep clustering works were proposed to incorporate improvements to the DEC [21] framework. Specifically, the IDEC model [37] extended DEC by preserving the local structure of data in the feature space through keeping the decoder layers to avoid feature space distortion by the clustering loss. Another model introduced in [27] extended IDEC by adopting a convolutional autoencoder (CAE) as the deep network. Later, several deep clustering works based on (CAE) were suggested [32]. Data augmentation was also associated with DEC in [28] to improve the model's generalization by making the DNN learn more representative features. Moreover, a k-means based DEC framework was presented in [19]. In particular, a cost function that consists of dimensionality reduction, data reconstruction, and a k-means clustering structure promoting regularization loss terms was introduced. Another related DEC-based work [33] leverages symmetry-based distances within the DEC framework as a powerful tool to distinguish symmetric shapes.

The authors in [38] observed the fact that DEC [21] does not make use of prior knowledge to guide the learning process. They extended DEC [21] and proposed a new scheme of Semi-Supervised Deep Embedded Clustering (SDEC) to overcome this limitation. SDEC's best performance was reported as an 86.11% clustering accuracy and 82.89% NMI on the MNIST image dataset. Since then, Semi-Supervised Deep Embedded Clustering (SS-DEC) became a growing field. Actually, most of the SS-DEC literature addresses general clustering purposes [39–42]. On the other hand, some works targeted anomaly detection [43], software fault proneness classification (fault prediction) [44], image classification and segmentation [45], and deep generative purposes [46,47]. The supervision injected in SS-DEC models takes several forms. The researchers in [41] imposed standard must-link and cannot-link pairwise constraints to their model, while the authors in [31] extended the encoding of standard pairwise constraints to more complex constraints such as continuous values (triplet constraints), instance difficulty constraints, and cluster-level balancing constraints. Triplet constraints are useful in the cases where no strong pairwise guidance is available. As for the instance difficulty constraints, it allows the user to a priori specify which instances are easier to cluster (i.e., they belong strongly to only one cluster), whereas the cluster-level balancing constraints enable the experts to guide the clustering process via the prior cardinality information. Another approach, Ts2DEC, outlined in [48], used triplet constraints.

In [49], the pairwise constraints are self-generating from a mutual KNN graph which makes the clustering approach unsupervised. The method then extends it to semi-supervised clustering by including human intervention to finetune these self-generating constraints by analyzing the losses associated with the pairs to form a set of false positive candidates.

ClusterNet [50] also uses pairwise semantic constraints from very few labeled data samples (<5% of total data) and exploits the abundant unlabeled data to drive the clustering. The network is optimized by minimizing a combination of k-means-based clustering loss and pairwise KL-divergence loss where the two are regularized via an autoencoder's reconstruction loss and each are defined for both the labeled as well as unlabeled data. A different approach to include supervision information was outlined in [47] where a small, labeled dataset is used to assign classes to components of Gaussian mixtures. The resulting mixture describes the distribution of the whole data.

The authors in [39] extended deep embedded clustering approaches to Electronic Health Record (I) patient cohorts. Specifically, supervision was applied by modifying the latent representation according to known patient subgroups through applying transfer learning of the encoder and fine-tuning of layers.

One should note that most DEC-based works [38,39,45,48] use k-means clustering algorithms to initialize the cluster centers, followed by cluster assignment hardening in the form of clustering loss. However, the authors in [40] used a graph-based clustering method, while in [49], the mutual k nearest neighbor (MKNN) neighborhood method was employed to automatically extract appropriate pairs for clustering, and these pairs were used as must-link constraints. On the other hand, in [51], the researchers replaced the classical k-means clustering with a density-based clustering approach to cluster the learned low-dimensional embedded features. The concept of soft constraints was introduced in SS-DEC models by the authors in [42]. They extended a previous work [52] to neural networks with instance-level constraints. In [52], soft constraints were introduced by allowing the constraints to be violated with violation costs.

Generally, deep clustering approaches' potential was mainly explored in the context of entirely unsupervised learning. The resulting approaches are still prone to the local minima due to the NP-hardness of the clustering problem. On the other hand, most real-world datasets are naturally weakly labeled. Therefore, exploiting some prior knowledge as supervision information for the cluster analysis was later investigated to carry the clustering process away from the local minima. This conforms with the assumption that feeding the learning algorithm with some supervision in the form of a reward would improve the learning. A semi-supervised learning paradigm has produced a considerable impact on various machine learning-based applications [53]. Despite these achievements, the efforts made to extend and improve the existing semi-supervised deep clustering approaches remain far below expectations compared to the researchers' contributions to enhance fully unsupervised approaches. Several potentials for improvements can be investigated in this domain. Specifically, incorporating supervision information in a relaxed way was not investigated in previous works. In particular, such supervision can be softly expressed within the models' objective using "should-link" pairwise constraints to determine whether the pairs of data instances should be assigned to the same or different cluster(s). Moreover, the use of fuzzy logic [53] was not explored to represent the data partition in the context of the existing semi-supervised deep learning approaches. Thus, we introduce a novel, semi-supervised, deep clustering algorithm that incorporates the abovementioned aspects that we believe would fill several gaps and upscale the deep clustering performance.

3. Proposed Soft Constrained Deep Clustering (SC-DEC)

The proposed semi-supervised deep clustering approach, named Soft Constrained Deep Clustering (SC-DEC), leverages a deep neural network architecture for feature learning and performs clustering with fuzzy membership degrees. Specifically, the deep clustering problem is formulated as an optimization of a novel objective function that is designed to simultaneously discover the hidden data clusters and optimize the deep neural network. Furthermore, the available supervision knowledge is incorporated as soft pairwise constraints within the objective function in a soft way such that the compliance to a constraint is not strictly obligated. Moreover, data partition is represented in a fuzzy manner within the objective to accurately reflect the grouping of data.

This section introduces the design details of the proposed SC-DEC approach. The block diagram of SC-DEC is depicted in Figure 1. The proposed semi-supervised deep embedded clustering approach relies on two main components: (i) an objective function that carries out the clustering tasks as proposed, and (ii) an autoencoder AE network that learns the discriminative embedded representation of the original data.

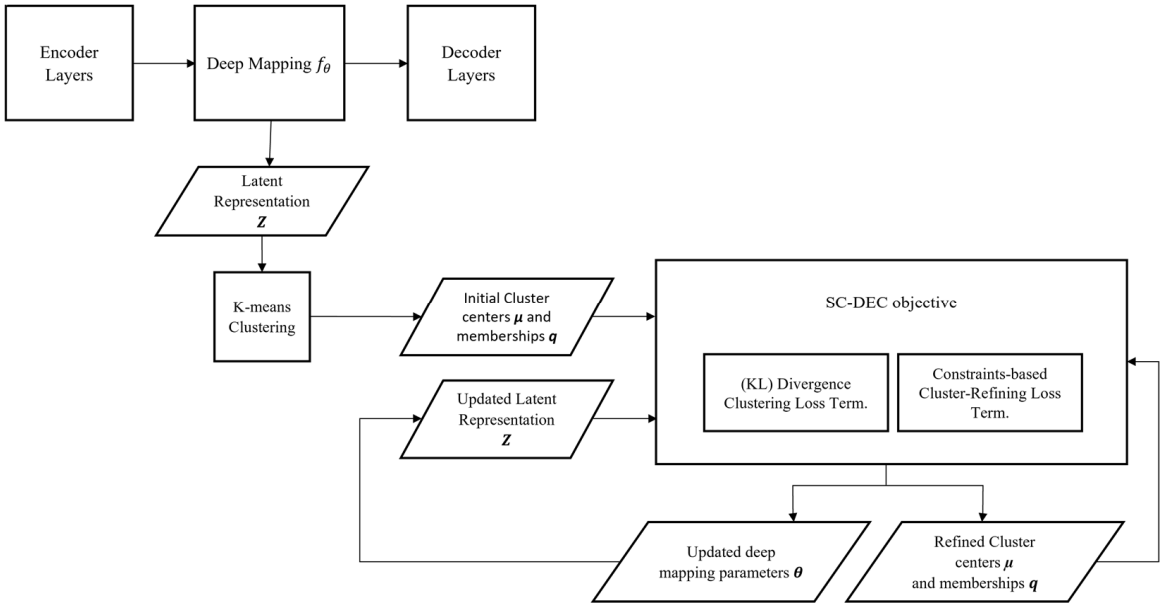


Figure 1. Overview of the proposed approach (SC-DEC).

Let $X = \{x_i\}_{i=1}^{n}$ be a set of unlabeled data that serves as the input to our proposed clustering method, where each sample $x_i \in \mathbb{R}_d$ and d is the dimensions of the input space. Instead of performing the clustering task directly in the original data space, we define a nonlinear mapping function f_θ to transform the original input data X into a latent feature space Z; $f_\theta : X \rightarrow Z$, where θ represents the learnable parameters. The dimensionality of the latent space Z is much lower than the original space dimensionality. The main goal of the proposed SC-DEC method is to output an appropriate clustering of data in the embedded feature space Z by utilizing the unlabeled data and the injected supervision. Specifically, it aims to partition the data input into k clusters, where each cluster is defined by a centroid $\mu_{j=1,\ldots,k}$ and $\mu_j \in \mathbb{R}_d$. Specifically, for k-means algorithms, the number of clusters k is based on the related literature and the considered benchmark datasets. Basically, we aim to find a cluster-friendly f_θ such that the learned parameters are biased towards the clustering task and the available knowledge.

To initialize the parameters of the f_θ, a deep autoencoder network [54] is built and trained in an unsupervised manner. Later, only the encoder layers are used within our model to receive the input data and transform it to the embedded space. The supervision knowledge are defined in the proposed approach as a set of should-link soft constraints and denoted by $\mathbb{C} = \{(x_i, x_k) : x_i \text{ and } x_k \text{ should be assigned to the same cluster, } 1 \leq i, j \leq N.\}$. These soft constraint sets are pre-defined and randomly generated from the dataset and generally, the supervision information should be available for a subset of the dataset only. In the Experiments section, we vary its ratio and investigate the performance.

We introduce our algorithm as an improvement to deep embedded clustering algorithms. Specifically, we propose to integrate should-link constraints into the clustering loss objective of DEC [21] to find clustering-friendly representations. Consequently, we tackle the proposed optimization problem by defining the following objective function

and obtaining the partial derivatives of the parameters to be updated via minimization, as shown below:

$$J = \sum_{i=1}^{n} \sum_{j=1}^{C} p_{ij} log \frac{p_{ij}}{q_{ij}} + \gamma \left[\sum_{x_i, x_k \in \mathbb{C}} \sum_{j=1}^{C} q_{x_i j}^{m} q_{x_k j}^{m} \right] \quad (2)$$

where n is the number of data points, C is the number of clusters, m is the fuzziness parameter that reflects the fuzzy-based representation of the data clusters, and $\mathbb{C}$ is the set of should-link constraints as shown above.

The first term is the deep embedded clustering term of DEC [21], while the second term is designed to learn the compact, fuzzy-based clusters given the supervision soft constraints. The second term basically rewards the model for correctly clustering, i.e., having a high membership value of a should-link pair of points to a certain cluster. In (2), q_{ij} and p_{ij} are computed as shown below [21]:

$$q_{ij} = \frac{(1 + \|z_i - \mu_j\|^2)^{-1}}{\sum_{j'} (1 + \|z_i - \mu_{j'}\|^2)^{-1}}, \quad (3)$$

$$p_{ij} = \frac{q_{ij}^2 / f_j}{\sum_{j'} q_{ij'}^2 / f_{j'}} \quad (4)$$

where $f_j = \sum_i q_{ij}$ are soft cluster frequencies; $z_i = f_\theta(x_i)$ is the embedded representation of x_i; μ_j is the j^{th} cluster centroid in the embedded space; and $\|.\|$ denotes the L2-norm.

Note that γ in the above equation is a predefined trade-off parameter that balances the influence of the supervision soft constraints. It balances the amount of penalty imposed on the data batch for misclustering.

Minimizing the above objective function is preceded by the training of an autoencoder AE for DNN parameter initialization. This AE training is performed by minimizing the following objective function:

$$J_{AE} = \sum_{i=1}^{n} \|x_i - z_i\|_2^2 + \gamma \left[\sum_{x_i, x_k \in \mathbb{C}} \sum_{j=1}^{C} q_{x_i j}^{m} q_{x_k j}^{m} \right] \quad (5)$$

where $\|\boldsymbol{x_i} - \boldsymbol{z_i}\|_2^2$ is the squared Euclidean distance between data point x_i and its embedding z_i.

The proposed algorithm would minimize the objective function in (2) iteratively and converge towards the clustering results under specific criteria (tolerance threshold %). Concretely, the clustering process carried out via our proposed approach follows DEC [21] in its two phases: 1. model parameter initialization. In this phase, the deep embedding parameters are initialized by training a deep autoencoder network. Then, the k-means clustering algorithm [55] is applied to the embedded space Z to initialize the k cluster center μ_j. 2. Model parameter optimization, i.e., updating the deep mapping f_θ and refining the cluster centers μ_j. In phase 2, the proposed objective loss is minimized to learn from the current high confidence predictions by iterating between computing an auxiliary target distribution and minimizing the Kullback–Leibler (KL) divergence to it.

To guide the learning process away from the local minima, some side information is adopted within our approach as a set of pairwise constraints. Basically, such information is incorporated to reward or penalize the traditional clustering objective as well as to adapt the distance measure to each cluster. Specifically, soft pairwise constraints are introduced and incorporated to represent the available prior knowledge where we formulate the second term of the model's objective according to these constraints. These pairwise constraints are randomly generated with a size equal to $\beta \times N$, where β is the number (ratio) of pairwise constraints and N is the dataset size. Specifically, for each randomly selected data point, we check the corresponding ground-truth labels of the whole dataset. If the ground-truth labels of the two points are similar, a "should-link" pairwise link is formed between these points and represented by 1. Otherwise, the pairwise link is represented by 0.

Unlike the constraints referred to as "must-link" and "cannot-link", the constraints in our formulation are soft and not obligated. They can be viewed as a reward for correctly clustering a point, which is suitable for the poorly labeled knowledge available in the real world. The soft constraint term of the objective function governs the consistency between the learned representation and clustering with the side information provided.

One should note that the Adam optimizer [56] was used to jointly optimize our objective in (2) with respect to its parameters, the cluster centers μ_j, and the DNN parameters. The following algorithm (Algorithm 1) depicts the steps designed and implemented to solve the clustering task of the proposed SC-DEC approach.

Algorithm 1 Soft Constrained Deep Clustering

INPUT: Dataset $\boldsymbol{X}$; number of clusters $\boldsymbol{K}$; update interval $\boldsymbol{T}$; stopping threshold $\boldsymbol{tol\%}$; maximum number of training iterations $\boldsymbol{maxIter}$; γ: the constraint term weight; fuzzifier $\boldsymbol{m}$; AE pretrain epochs; $\boldsymbol{\beta}$: number of pairwise constraints; data batch size.
OUTPUT: Cluster centers $\{\boldsymbol{\mu_i}\}_{i=1}^{k}$; deep mapping weights $\boldsymbol{\theta}$; cluster label assignments $\{\boldsymbol{y_i}\}_{i=1}^{n}$.

STEP1: Pretrain the deep network (AE) on input $\boldsymbol{X}$ and according to the input hyper-parameters to obtain initial deep mapping weights θ and the data in latent space $\boldsymbol{Z}$.
STEP2: Initialize the values of cluster centers $\{\boldsymbol{\mu_i}\}_{i=1}^{k}$ and cluster assignments $\{\boldsymbol{y_i}\}_{i=1}^{n}$ by running k-means in $\boldsymbol{Z}$ from step1.
STEP3: Begin model fitting:
→ Create pairwise constraints set $\mathbb{C}$ according to the input parameters **X, $\boldsymbol{\beta}$**.
→ **for** iter $\in \{0, 1, \ldots, maxIter\}$ **do**
    → **if** update_interval $\boldsymbol{T}$ is reached **then**
        → Compute embedded points on all dataset $\boldsymbol{X}$ using $f_{\boldsymbol{\theta}} : \boldsymbol{X} \rightarrow \boldsymbol{Z}$.
        → Compute soft assignments $\boldsymbol{q}$ and target distribution $\boldsymbol{p}$ using $\boldsymbol{z_i}$ and $\boldsymbol{q_{ij}}, \boldsymbol{p_{ij}}$ formulas in Equations (3) and (4).
        → Update label assignments $\{\boldsymbol{y_i}\}_{i=1}^{n}$.
        → Compute cluster performance metrics.
        → Save old label assignments.
        → Compute change in label assignment and stop training if it is < $\boldsymbol{tol\%}$.
    → **end if**
    → Begin custom training on sequential data batches $\boldsymbol{S} \subset \boldsymbol{X}$.
    → Compute constraint term per batch using Equation (2) and according to $\mathbb{C}$ and to the input hyper-parameters $\boldsymbol{m}$, γ.
    → Add the constraint term to the custom loss in Equation (2).
    → Train the model by minimizing Equation (2) using the selected optimizer.
    → Update cluster centers $\left\{\boldsymbol{\mu_j}\right\}_{j=1}^{K}$ and deep mapping weights θ on S via the respective update equations.
end for

4. Experiments

The performance of the proposed SC-DEC was validated through several experimental scenarios using benchmark datasets. Moreover, the obtained results were compared with those achieved via relevant state-of-the-art approaches. Below are the hypotheses tested via the experiments:

- Null hypothesis: introducing supervision to relevant unsupervised learning approaches, as described in Section 3, does not improve the clustering results.
- Alternate hypothesis: introducing supervision to relevant unsupervised learning approaches improves the clustering results. This means that the proposed SC-DEC yields higher accuracy than the relevant existing methods.

In the following subsections, we will outline the experiments including the datasets and performance measures, the implementation details, the experimental scenarios, parameter settings, and lastly, the results and discussion of the experiments.

4.1. Datasets and Performance Measures

The proposed approach was mainly assessed using multiple benchmarking datasets that are widely used by the deep clustering research community. The MNIST dataset [57], which consists of 70,000 images of handwritten digits, was used. One should note that the MNIST digits are size-normalized and centered in a fixed-size image. Each digit in MNIST is represented using a gray image with a size of 28 × 28 pixels. This results in a 784-dimensional vector for each image. In addition, the USPS dataset [58] that contains 9298 gray-scale, handwritten, 16 × 16 pixel digit images was also used. Moreover, the STL-10 dataset [59] that includes 13,000 color images, categorized into 10 classes, was considered for the experiments. Each STL-10 image is a 96 × 96 pixel size. Table 1 details these datasets and depicts the relevant ground-truth information for the clustering task.

Table 1. Details of the datasets used in this research.

Dataset Name and Refence	Number of Points	Number of Classes	Number of Dimensions
MNIST [57]	70,000	10	784
USPS [58]	9298	10	256
STL-10 [59]	13,000	10	1428

The performance of the proposed approach was evaluated using standard metrics that are widely used by the clustering research community: the classification accuracy (ACC), which indicates the percentage of the correctly clustered samples, and the Normalized Mutual Information (NMI) measure which represents the normalized similarity between the true and predicted labels for the data records. Both ACC and NMI values range between 0 and 1.

4.2. Implementation Details

Python language, along with the required libraries, were associated with high-performance resources to implement the intended experiments. For the non-linear transformation f_θ, we select a fully connected autoencoder deep network with d–500–500–2000–10 dimensions for all datasets, where d is the data-space dimension. All internal layers of the autoencoder are activated via a ReLU nonlinearity function [60] except for the input, output, and embedding layers.

The autoencoder weights were initialized using greedy layer-wise pretraining. The optimization method for the pretraining was Stochastic gradient descent (SGD) with a learning rate of 0.01 and a momentum of 0.9 across all datasets. To initialize the cluster centers μ_j, we ran k-means 20 times and selected the best solution. In the parameter optimization phase, we trained our (SC-DEC) model using the Adam optimizer with a default learning rate of 0.001. The stopping threshold was set to 0.001.

4.3. Experimental Scenarios

After conducting preliminary investigations and experiments using the proposed approach, it was observed that higher numbers of training epochs can result in a poor clustering performance. Moreover, the clustering results are sensitive to the initialization. Furthermore, the Adam optimizer outperformed the other optimizers, with a significant accuracy advantage. Accordingly, a lower range of training epochs was set for all datasets. Moreover, random seeds were fixed for all implementation libraries. Specifically, the seeds values were set based on the performances recorded in three trial runs.

During the preliminary experimentation on model optimizers, the hyper-parameters used for all datasets were set as follows: batch size = 256, γ and β = 1, and m = 1.2 (hyper-parameters are defined in Section 3). As for the number of pretraining epochs, 1000 was used for USPS, 2000 for MNIST, and 20 for STL-10. Table 2 shows the accuracy values resulting from using the different model optimizers and optimizers properties. It is

noticeable from Table 2 that the Adam optimizer outperformed the other optimizers for all datasets and with a large margin in the STL-10 dataset.

Table 2. Performance achieved via the proposed approach using different optimizer settings and datasets.

Optimizer Setting	Learning Rate	Momentum	MNIST	USPS	STL-10
SGD	0.01	0.9	85.94%	53.86%	25.49%
	0.001	0.9	79.42%	34.42%	22.97%
	0.1	0.9	86.55%	68.78%	28.84%
	0.01	0.9	79.19%	34.39%	26.36%
Adam	0.001	0	87.36%	74.12%	91.50%

After setting Adam as the model optimizer for our proposed approach, we performed other preliminary experiments to investigate the proper range for tuning the number of AE pretraining epochs. Table 3 shows the settings adopted to cluster the datasets using the proposed approach along with the Adam optimizer.

Table 3. Settings for the experiments investigating the number of AE pretraining epochs using the proposed approach along with Adam optimizer using the different datasets.

Hyper-Parameter	MNIST	USPS	STL-10
batch size	256	256	256
γ	1	10	1
β	1	1	1
m	1.2	1.2	2

Regarding the settings in Table 3, some of the experiments had specific settings. For USPS, in the 50-epoch case, $\gamma = -100$, $\beta = 0.5$, and $m = 1.5$. In the 100-epoch case, $\gamma = 1$ and $m = 2$. For the MNIST dataset, $m = 1.5$ in the 300-epoch case and $m = 2$ in the 400-epoch case. Table 4 shows the preliminary experiment results in terms of the accuracy values over the number of AE pretraining epochs. For the STL-10 dataset, a smaller number of epochs led to better clustering accuracy. Moreover, for all datasets, a higher number of pretraining epochs was not correlated with the increasing clustering accuracy. Based on that, we selected the range for tuning the number of AE pretraining epochs to be within smaller values that are relative to the size of the dataset.

It should also be mentioned that as the Adam optimizer was used, the tuning scenario dedicated for the setting of the (optimizer) hyper-parameter was excluded. For all experiments conducted using the different datasets, the relevant hyper-parameters were tuned for a better initialization and setting, as shown in Table 5.

In Table 5, since it is the first parameter to be tuned, the number of pretraining epochs does not require default value initialization. As for the batch size's default value, it was initialized according to the value that is widely used by the clustering research community. Regarding β and γ default values, they were initialized with neutral values. As for the initialization of the Fuzzifier m default value, the most frequently used and accepted value in various applications is $m = 2$ [61]. However, when datasets have significant uneven distributions in the cluster sizes, a smaller fuzzifier value has been suggested for the FCM-based clustering algorithms [62].

Table 4. Performance achieved via the proposed approach using different numbers of AE pretraining epochs on the different datasets. Bold formatting indicates the best obtained results.

MNIST		USPS		STL-10	
# of Pretraining epochs	**ACC**	**# of Pretraining Epochs**	**ACC**	**# Pretraining Epochs**	**ACC**
20	83.44%	20	74.45%	20	**91.02%**
30	83.51%	30	75.73%	30	78.67%
50	85.54%	50	**76.62%**	50	77.62%
100	85.45%	100	75.16%	100	62.25%
200	86.08%	200	75.15%	200	74.98%
300	**89.00%**	300	75.04%	300	66.87%
400	87.00%	500	75.78%	500	66.58%
1000	85.47%	1000	74.70%	1000	66.52%
2000	87.41%	2000	60.82%	2000	65.25%

Table 5. Hyper-parameters considered for hyper-parameter tuning along with the default values.

Hyper-Parameter	Default Value MNIST	Default Value USPS	Default Value STL-10
Number of AE pretraining epochs	-	-	-
Data batch size	256	256	256
Gamma γ	1	1	1
Beta β	1	1	1
Fuzzifier m	2	2	2

Next, a set of experiments for tuning the hyper-parameters were conducted to target the optimal solution in terms of clustering accuracy. Namely, we investigated the number of pretraining epochs, the amount of supervision (number of constraints β), the fuzziness parameter m, the trade-off parameter (i.e., the constraint term weight) γ, and the data batch size. The applicable range for these hyper-parameters is as follows:$\beta \in \{x \mid x > 0, x \in \mathbb{R}\}$, $\gamma \in \{x \mid x \in \mathbb{R}\}$, and the data batch size $\in \{x \mid x > 0, x \in \mathbb{Z}\}$. The latter values used to tune the data batch size are based on the related literature and the considered benchmark datasets. On the other hand, according to [63], the fuzzifier m is recommended to be within the interval [2, 3.5], while the authors in [61] suggested the interval [1.5, 2.5] for m. Based on that, we set the range of values {1.2, 1.5, 1.7, 2, 3} to tune the fuzzifier m for SC-DEC. Specifically, the tuning process was performed by dedicating and running one experiment for each hyper-parameter as reported in Algorithm 2. This sequential tuning strategy was intended to select the hyper-parameter value that yields the highest clustering accuracy using SC-DEC.

4.4. Results and Discussion

As outlined above, extensive experiments were conducted to assess the performance of the proposed approach, SC-DEC. The results were quantitatively analyzed, visualized, and compared to those achieved using relevant unsupervised and semi-supervised deep clustering algorithms. Namely, the proposed approach was compared with DEC [21], Semi-Supervised Deep Embedded Clustering (SDEC) [38], and Improved Deep Embedded Clustering with Local Structure Preservation (IDEC) [37]. Additionally, we added the results of applying simple k-means [8] for clustering the deep embeddings of the respective datasets (called DL + k-means).

Algorithm 2 Hyper-parameter tuning strategy

INPUT: Dataset $\boldsymbol{X}$; hyper-parameter set considered for tuning: {number of AE pretraining epochs; γ: the constraint term weight; $\boldsymbol{m}$: the fuzzifier; $\boldsymbol{\beta}$: number of pairwise constraints; data batch size}; default values of the hyper-parameters from Table 5; random sets of seed values.
OUTPUT: Hyper-parameters values that achieve the highest accuracy, per dataset.

STEP1: Tuning of the (number of pretraining epochs) hyper-parameter.
a. Other hyper-parameters are set to the default values.
b. Tuning values per dataset:
- USPS: {50, 100, 150, 200};
- MNIST: {100, 200, 300, 400, 500};
- STL-10: {20, 50, 70, 100, 150, 200},

c. Run the model for each value 3 times, with different seeds per run.
d. Save the pretraining epochs number and the seeds that resulted in the highest accuracy among runs.
e. Update the default value for the number of pretraining epochs according to (d).
f. Use the seeds in (d) for the rest of the experiments.

STEP2: Tuning of the (fuzzifier $\boldsymbol{m}$) hyper-parameter.
a. Other hyper-parameters are set to the default values.
b. Tuning values for all datasets: {1.2, 1.5, 1.7, 2, 3}.
c. Run the model for each value.
d. Save and use the $\boldsymbol{m}$ value that resulted in the highest accuracy for the rest of the experiments.
e. Update the default $\boldsymbol{m}$ value according to (d).

STEP3: Tuning of (γ) hyper-parameter.
a. Other hyper-parameters are set to the default values.
b. Tuning values for all datasets: {+1, +10, +100, +1000, −1, −10, −100, −1000}.
c. Run the model for each value.
d. Save and use the γ value that resulted in the highest accuracy for the rest of the experiments.
e. Update the default γ value according to (d).

STEP4: Tuning of the ($\boldsymbol{\beta}$) hyper-parameter.
a. Other hyper-parameters are set to the default values.
b. Tuning values for all datasets: {0.1, 0.3, 0.5, 0.7, 0.9, 1, 1.2, 1.5, 2}.
c. Run the model for each value.
d. Save and use the $\boldsymbol{\beta}$ value that resulted in the highest accuracy for the rest of the experiments.
e. Update the default $\boldsymbol{\beta}$ value according to (d).

STEP5: Tuning of the (data batch size) hyper-parameter.
a. Other hyper-parameters are set to the default values.
b. Tuning values for all datasets: {64, 128, 256, 512}.
c. Run the model for each value.
d. Save and use the batch size value that resulted in the highest accuracy for the rest of the experiments.
e. Update the default batch size value according to (d).

After hyper-parameter tuning, detailed in Algorithm 2, we ran 20 experiments using the hyper-parameter values that achieved the best accuracy. Table 6 reports the results achieved using the proposed approach along with the state-of-the-art methods on the MNIST, USPS, and STL-10 datasets. As can be seen in Table 6, the results mainly showed the positive impact of utilizing minimal prior knowledge about the data on the clustering performance. According to Table 6, the most notable improvement was obtained using the STL-10 dataset, with a clustering accuracy of 91.65%, which represents a drastic improvement compared to DEC [21] and SDEC [38].

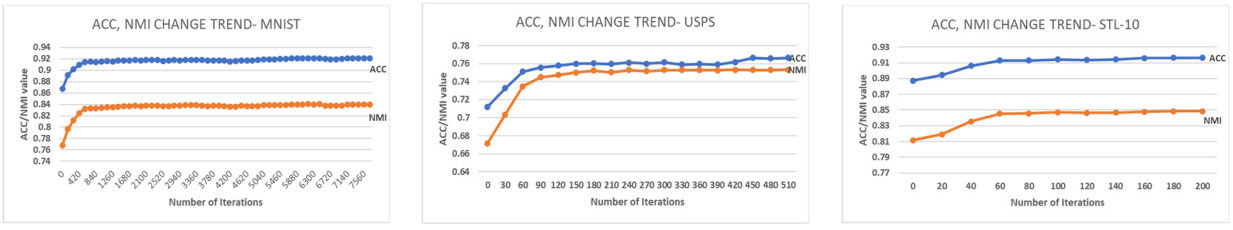
Table 6. Clustering results measured in ACC and NMI. Bold formatting indicates the best results.

Method	MNIST [57]		STL-10 [59]		USPS [58]	
Metric	**ACC**	**NMI**	**ACC**	**NMI**	**ACC**	**NMI**
DL + k-means	86.83%	76.89%	86.94%	78.84%	70.93%	68.44%
DEC [21]	84.30%	NA	35.90%	NA	NA	NA
SDEC [38]	86.11%	82.89%	38.86%	32.84%	76.39%	**77.68%**
IDEC [37]	83.841%	77.885%	NA	NA	72.693%	71.135%
SC-DEC	**92.11%**	**84.01%**	**91.65%**	**84.85%**	**76.62%**	75.31%

One can notice from Table 6 that the SC-DEC performance exceeded that of DL + k-means on all three datasets. Moreover, we can see that SC-DEC outperformed the non-constrained approach DEC [21] on the MNIST and STL-10 datasets. This proves the importance of the proposed semi-supervision information formulated as pairwise constraints in guiding the deep clustering process. Moreover, the results confirmed the importance of the fuzzy membership representations in improving the clustering partition. Furthermore, the proposed approach outperformed the non-fuzzy approach SDEC [38] in terms of clustering accuracy for all datasets.

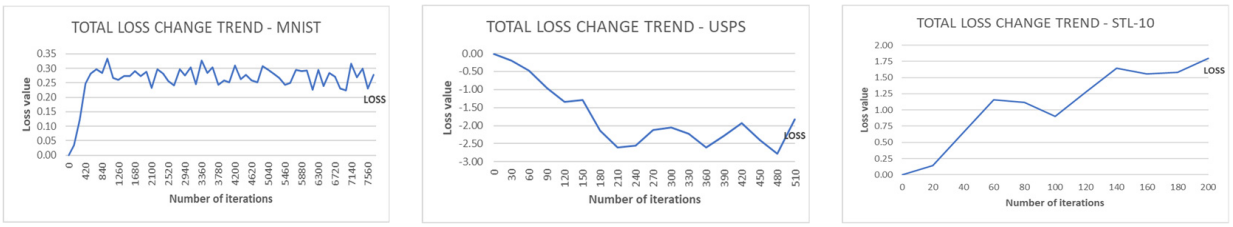
In addition, we retested the IDEC model in [37] using the settings reported in the paper: SGD optimizer with a 0.01 learning rate and 0.9 momentum, with the set of pretrained AE weights used for the implementation available online. The results in Table 6 show that the proposed SC-DEC outperformed the IDEC results in terms of clustering accuracy and NMI metrics.

Moreover, Figure 2 plots the change trends of the ACC and NMI metrics for the best run on the three datasets. For all datasets, one can observe that the improvement in both metrics stabilized after roughly 30% of the total training time. Then, a slight improvement occurred later that led the model into convergence. Additionally, Figure 3 shows the loss change trend (learning curve) of the run that resulted in the best accuracy for the three datasets.



(**a**) (**b**) (**c**)

Figure 2. Change trend recorded for the performance metrics obtained using SC-DEC on MNIST (**a**), USPS (**b**), and STL-10 (**c**) datasets.



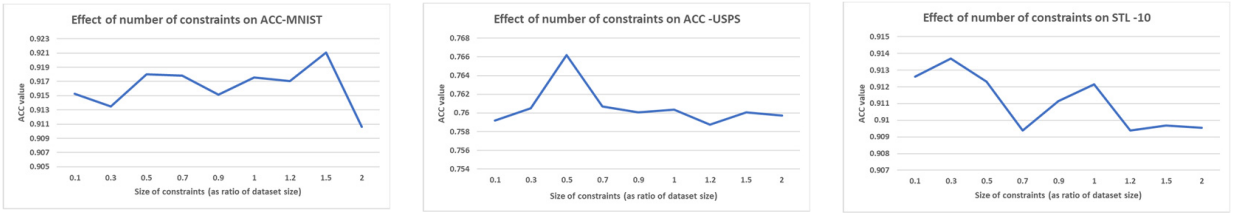
(**a**) (**b**) (**c**)

Figure 3. Loss change over training iterations on MNIST (**a**), USPS (**b**), and STL-10 (**c**) datasets.

The results of the hyper-parameter tuning presented earlier are shown in Table 7. Specifically, this table shows the hyper-parameter values that yielded the best accuracy for the proposed model when associated with each of the datasets. Moreover, Figures 4–7 plot the trends in accuracy changes achieved via SC-DEC according to the different hyper-parameter values on the three datasets.

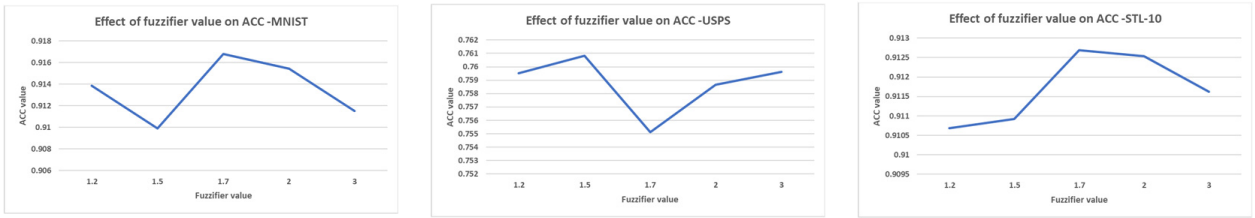
Table 7. Hyper-parameter values that yielded the highest accuracy for the different datasets.

Hyper-Parameter	Value for MNIST	Value for USPS	Value for STL-10
Number of AE pretraining epochs	300	50	20
Number of constraints β	1.5	0.5	0.3
Fuzzifier m	1.7	1.5	1.7
Constraint Term weight γ	1	−100	100
Data batch size	256	256	512



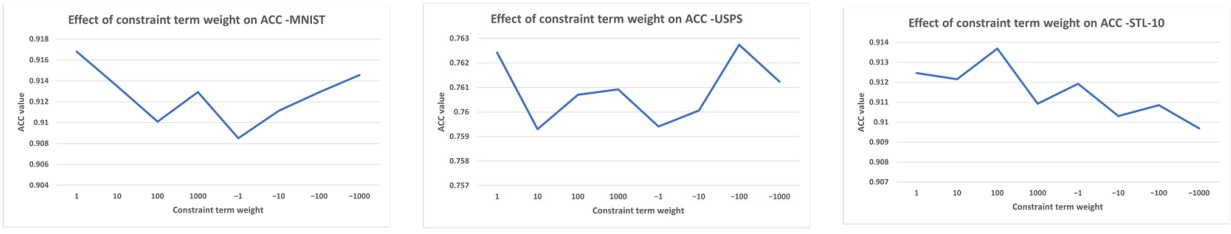
(**a**) (**b**) (**c**)

Figure 4. Effect of number of constraints β on SC-DEC clustering accuracy for MNIST (**a**), USPS (**b**), and STL-10 (**c**) datasets.



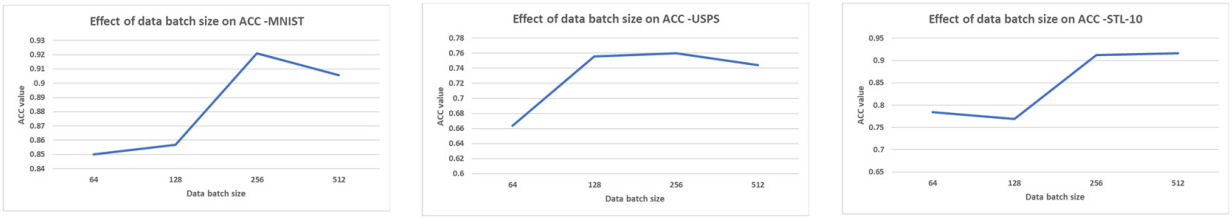
(**a**) (**b**) (**c**)

Figure 5. Effect of fuzzifier value m on SC-DEC clustering accuracy for MNIST (**a**), USPS (**b**), and STL-10 (**c**) datasets.



(**a**) (**b**) (**c**)

Figure 6. Effect of constraint term weight γ on SC-DEC clustering accuracy for MNIST (**a**), USPS (**b**), and STL-10 (**c**) datasets.



(**a**) (**b**) (**c**)

Figure 7. Effect of data batch size on SC-DEC clustering accuracy for MNIST (**a**), USPS (**b**), and STL-10 (**c**) datasets.

We can notice from Table 7 and Figure 4 that the number of pairwise constraints (β hyper-parameter) was correlated with the dataset size. In the largest dataset, MNIST, the best value was $\beta = 1.5$, compared to $\beta = 0.3$ and $\beta = 0.5$ for the STL-10 and USPS datasets, respectively. However, the change in accuracy in response to the increased β value was not substantial, as the difference between the highest and lowest accuracy equaled 1.05% (MNIST), 0.43% (STL-10), and 0.74% (USPS). This is consistent with the findings in the SDEC paper [38] that the initial introduction of pairwise constraints into deep embedded clustering will lead to a significant increase in performance, and then the performance becomes stable, indicating that enough prior information has been captured. According to SDEC, this observation is generally consistent with the semi-supervised learning literature.

Moreover, the effect of the number of constraints β on the performance of the proposed approach may be subject to the pairwise constraints generated, as the generation is random for each new value of the hyper-parameter β. This would explain the relative randomness that characterizes the results shown in Figure 4.

Furthermore, Table 7 and Figure 5 show that the highest accuracy values for the fuzzy parameter m among the three datasets are within the interval suggested by [61] as a heuristic for selecting the m value for FCM-based clustering. The calculation in [61] suggests that the best choice for m is probably in the interval of [1.5, 2.5]. Interestingly, training our model on the STL-10 dataset using a 512-batch size yielded better results than using the default 256-batch size.

5. Conclusions and Future Work

Most real-world datasets are weakly labeled by nature, which inspired the utilization of the available prior knowledge as supervision information by clustering algorithms to carry the clustering process away from the local minima. This led to the introduction of semi-supervised learning within the clustering paradigm. However, the state-of-the-art semi-supervised deep clustering approaches remain below expectations compared to the fully unsupervised approaches, with several potentials to be explored. In this paper, we propose a novel semi-supervised deep clustering approach (named Soft Constrained Deep Clustering, SC-DEC) to overcome the limitations in the existing semi-supervised clustering approaches. Specifically, the proposed approach leverages a deep neural network architecture and generates fuzzy membership degrees that better reflect the true partition of the data. Furthermore, the scarcely available prior knowledge was used as side information and formulated as a set of soft pairwise constraints to direct the machine learning process into clustering unlabeled data. This clustering task was formulated as an optimization problem where a novel objective function was minimized to simultaneously discover the hidden data clusters and optimize the deep neural network. The proposed approach was assessed using standard datasets and performance measures. The experiments proved that the proposed approach can automatically learn the partitioning of data. Moreover, various calibrations of the model's hyper-parameters were investigated during the experiments. The experimental results showed that the size of the pairwise constraints was positively correlated with the dataset size. However, after enough prior information has been captured via the initial insertion of the pairwise constraints into the proposed model,

the model performance became stable. Furthermore, when compared to the state-of-the-art approaches, SC-DEC produced competitive results on the STL-10 dataset and outperformed the other models on MNIST and USPS.

Some limitations of the proposed approach should be stated, which include 1. sensitivity issues toward the presetting of the number of clusters and 2. the trial-and-error approach in selecting some of the hyper-parameter values for tuning. However, to address these limitations as well as to investigate new potentials, some directions are suggested for future studies. Specifically, we suggest researching the following approaches: 1. developing an automatic determination of the number of clusters through designing an additional term to the proposed objective; 2. researching robust heuristics to guide the selection process for hyper-parameters β and γ; 3.dropping the noise points and considering only the points with membership values over a certain threshold for the constraint term formulation; 4. investigating the effect of "should-not-link" pairwise constraints; and 5. studying the proposed method using text datasets.

Author Contributions: Conceptualization, M.S.A., M.M.B.I. and O.B.; methodology, M.S.A., M.M.B.I. and O.B.; software, M.S.A.; validation, M.S.A. and M.M.B.I.; formal analysis, M.S.A. and M.M.B.I.; investigation, M.S.A. and M.M.B.I.; resources, M.S.A. and M.M.B.I.; data curation, M.S.A.; writing—original draft preparation, M.S.A.; writing—review and editing, M.S.A., M.M.B.I. and O.B.; visualization, M.S.A. and M.M.B.I.; supervision, O.B. and M.M.B.I. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. The MNIST data can be found at http://yann.lecun.com/exdb/mnist/ (accessed on 18 July 2023). The USPS data can be found at [https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html#usps] (accessed on 18 July 2023). The STL-10 data can be found at https://cs.stanford.edu/~acoates/stl10/ (accessed on 18 July 2023).

Acknowledgments: The authors would like to thank the Deanship of Scientific Research in King Saud University for supporting this research through the initiative of Graduate Students Research Support (GSR).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Jha, J.; Ragha, L. Intrusion detection system using support vector machine. *Int. J. Appl. Inf. Syst.* **2013**, *3*, 25–30.
2. Dhankhad, S.; Mohammed, E.; Far, B. Supervised Machine Learning Algorithms for Credit Card Fraudulent Transaction Detection: A Comparative Study. In Proceedings of the 2018 IEEE International Conference on Information Reuse and Integration (IRI), Salt Lake City, UT, USA, 6–9 July 2018; pp. 122–125. [CrossRef]
3. Jain, P.; Garibaldi, J.M.; Hirst, J.D. Supervised machine learning algorithms for protein structure classification. *Comput. Biol. Chem.* **2009**, *33*, 216–223. [CrossRef]
4. Talabis, M.; McPherson, R.; Miyamoto, I.; Martin, J. *Information Security Analytics: Finding Security Insights, Patterns, and Anomalies in Big Data*; Syngress: Waltham, MA, USA, 2014.
5. Min, E.; Guo, X.; Liu, Q.; Zhang, G.; Cui, J.; Long, J. A Survey of Clustering With Deep Learning: From the Perspective of Network Architecture. *IEEE Access* **2018**, *6*, 39501–39514. [CrossRef]
6. Bramer, M. *Principles of Data Mining*; Springer: Berlin/Heidelberg, Germany, 2007; Volume 180.
7. Dahal, P. Deep Clustering. DeepNotes. 24 April 2019. Available online: http://deepnotes.io/deep-clustering (accessed on 28 February 2021).
8. Kanungo, T.; Mount, D.M.; Netanyahu, N.S.; Piatko, C.D.; Silverman, R.; Wu, A.Y. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2002**, *24*, 881–892. [CrossRef]
9. Johnson, S.C. Hierarchical clustering schemes. *Psychometrika* **1967**, *32*, 241–254. [CrossRef] [PubMed]
10. Ester, M.; Kriegel, H.-P.; Sander, J.; Xu, X. Density-based spatial clustering of applications with noise. *Int. Conf. Knowl. Discov. Data Min.* **1996**, *6*.
11. Xu, R.; Wunsch, D. Survey of Clustering Algorithms. *IEEE Trans. Neural Netw.* **2005**, *16*, 645–678. [CrossRef]

12. Sisodia, D.; Singh, L.; Sisodia, S.; Saxena, K. Clustering techniques: A brief survey of different clustering algorithms. *Int. J. Latest Trends Eng. Technol.* **2012**, *1*, 82–87.
13. Steinbach, M.; Ertöz, L.; Kumar, V. The Challenges of Clustering High Dimensional Data. In *New Directions in Statistical Physics*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 273–309.
14. Saul, L.K.; Weinberger, K.Q.; Sha, F.; Ham, J.; Lee, D.D. Spectral Methods for Dimensionality Reduction. *Semi-Supervised Learn.* **2006**, *3*, 293–308. [CrossRef]
15. Vasan, K.K.; Surendiran, B. Dimensionality reduction using Principal Component Analysis for network intrusion detection. *Perspect. Sci.* **2016**, *8*, 510–512. [CrossRef]
16. Bengio, Y.; Courville, A.; Vincent, P. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 1798–1828. [CrossRef]
17. Rolnick, D.; Veit, A.; Belongie, S.; Shavit, N. Deep learning is robust to massive label noise. *arXiv* **2017**, arXiv:1705.10694.
18. Aljalbout, E.; Golkov, V.; Siddiqui, Y.; Strobel, M.; Cremers, D. Clustering with deep learning: Taxonomy and new methods. *arXiv* **2018**, arXiv:1801.07648.
19. Yang, B.; Fu, X.; Sidiropoulos, N.D.; Hong, M. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In Proceedings of the International Conference on Machine Learning, PMLR, Sydney, Australia, 6–11 August 2017; pp. 3861–3870.
20. Shaham, U.; Stanton, K.; Li, H.; Nadler, B.; Basri, R.; Kluger, Y. SpectralNet: Spectral Clustering using Deep Neural Networks. *arXiv* **2018**. [CrossRef]
21. Xie, J.; Girshick, R.; Farhadi, A. Unsupervised deep embedding for clustering analysis. In Proceedings of the International Conference on Machine Learning, PMLR, New York, NY, USA, 20–22 June 2016; pp. 478–487.
22. Parsazad, S.; Saboori, E.; Allahyar, A. Data selection for semi-supervised learning. *arXiv* **2012**, arXiv:1208.1315.
23. Chen, G. Deep learning with nonparametric clustering. *arXiv* **2015**, arXiv:1501.03084.
24. Kampffmeyer, M.; Løkse, S.; Bianchi, F.M.; Livi, L.; Salberg, A.-B.; Jenssen, R. Deep divergence-based approach to clustering. *Neural Netw.* **2019**, *113*, 91–101. [CrossRef]
25. Caron, M.; Bojanowski, P.; Joulin, A.; Douze, M. Deep clustering for unsupervised learning of visual features. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 132–149.
26. Premachandran, V.; Yuille, A.L. Unsupervised Learning Using Generative Adversarial Training and Clustering; 2016. Available online: https://openreview.net/forum?id=SJ8BZTjeg (accessed on 17 July 2023).
27. Guo, X.; Liu, X.; Zhu, E.; Yin, J. Deep clustering with convolutional autoencoders. In Proceedings of the International Conference on Neural Information Processing, ICONIP 2017, Guangzhou, China, 14–18 November 2017; pp. 373–382.
28. Guo, X.; Zhu, E.; Liu, X.; Yin, J. Deep embedded clustering with data augmentation. In Proceedings of the Asian Conference on Machine Learning, PMLR, Beijing, China, 14–16 November 2018; pp. 550–565.
29. Shah, S.A.; Koltun, V. Deep continuous clustering. *arXiv* **2018**, arXiv:1803.01449.
30. Ren, Y.; Pu, J.; Yang, Z.; Xu, J.; Li, G.; Pu, X.; Yu, P.S.; He, L. Deep Clustering: A Comprehensive Survey. *arXiv* **2022**. [CrossRef]
31. Zhang, H.; Basu, S.; Davidson, I. A framework for deep constrained clustering-algorithms and advances. In Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Würzburg, Germany, 16 September 2019; pp. 57–72.
32. Dizaji, K.G.; Herandi, A.; Deng, C.; Cai, W.; Huang, H. Deep Clustering via Joint Convolutional Autoencoder Embedding and Relative Entropy Minimization. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 5747–5756. [CrossRef]
33. Moreno, J.G. Point Symmetry-based deep clustering. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management, Torino, Italy, 22–26 October 2018; pp. 1747–1750.
34. Jiang, Z.; Zheng, Y.; Tan, H.; Tang, B.; Zhou, H. Variational Deep Embedding: An Unsupervised and Generative Approach to Clustering. *arXiv* **2016**, arXiv:1611.05148.
35. Wang, C.; Pan, S.; Hu, R.; Long, G.; Jiang, J.; Zhang, C. Attributed Graph Clustering: A Deep Attentional Embedding Approach. *arXiv* **2019**, arXiv:1906.06532.
36. Li, K.; Wang, F.; Yang, L.; Liu, R. Deep feature screening: Feature selection for ultra high-dimensional data via deep neural networks. *Neurocomputing* **2023**, *538*, 126186. [CrossRef]
37. Guo, X.; Gao, L.; Liu, X.; Yin, J. Improved Deep Embedded Clustering with Local Structure Preservation. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, Melbourne, Australia, 19–25 August 2017; pp. 1753–1759. [CrossRef]
38. Ren, Y.; Hu, K.; Dai, X.; Pan, L.; Hoi, S.C.; Xu, Z. Semi-supervised deep embedded clustering. *Neurocomputing* **2019**, *325*, 121–130. [CrossRef]
39. Carr, O.; Jovanovic, S.; Albergante, L.; Andreotti, F.; Dürichen, R.; Lipunova, N.; Baxter, J.; Khan, R.; Irving, B. Deep Semi-Supervised Embedded Clustering (DSEC) for Stratification of Heart Failure Patients. *arXiv* **2020**, arXiv:2012.13233.
40. Li, X.; Yin, H.; Zhou, K.; Zhou, X. Semi-supervised clustering with deep metric learning and graph embedding. *World Wide Web* **2020**, *23*, 781–798. [CrossRef]
41. Vilhagra, L.A.; Fernandes, E.R.; Nogueira, B.M. TextCSN: A semi-supervised approach for text clustering using pairwise constraints and convolutional siamese network. In Proceedings of the 35th Annual ACM Symposium on Applied Computing, Brno, Czech Republic, 30 March–3 April 2020; pp. 1135–1142.

42. Wang, Z.; Mi, H.; Ittycheriah, A. Semi-supervised Clustering for Short Text via Deep Representation Learning. *arXiv* **2016**, arXiv:1602.06797.
43. Ruff, L.; Vandermeulen, R.A.; Görnitz, N.; Binder, A.; Müller, E.; Müller, K.R.; Kloft, M. Deep semi-supervised anomaly detection. *arXiv* **2019**, arXiv:1906.02694.
44. Arshad, A.; Riaz, S.; Jiao, L.; Murthy, A. Semi-Supervised Deep Fuzzy C-Mean Clustering for Software Fault Prediction. *IEEE Access* **2018**, *6*, 25675–25685. [CrossRef]
45. Enguehard, J.; O'Halloran, P.; Gholipour, A. Semi-Supervised Learning With Deep Embedded Clustering for Image Classification and Segmentation. *IEEE Access* **2019**, *7*, 11093–11104. [CrossRef]
46. Kingma, D.P.; Rezende, D.J.; Mohamed, S.; Welling, M. Semi-supervised learning with deep generative models. *arXiv* **2014**, arXiv:1406.5298.
47. Śmieja, M.; Wołczyk, M.; Tabor, J.; Geiger, B.C. SeGMA: Semi-Supervised Gaussian Mixture Autoencoder. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 3930–3941. [CrossRef]
48. Ienco, D.; Pensa, R.G. Deep triplet-driven semi-supervised embedding clustering. In Proceedings of the International Conference on Discovery Science, Split, Croatia, 28–30 October 2019; pp. 220–234.
49. Fogel, S.; Averbuch-Elor, H.; Cohen-Or, D.; Goldberger, J. Clustering-Driven Deep Embedding With Pairwise Constraints. *IEEE Comput. Graph. Appl.* **2019**, *39*, 16–27. [CrossRef]
50. Shukla, A.; Cheema, G.S.; Anand, S. Semi-supervised clustering with neural networks. In Proceedings of the 2020 IEEE Sixth International Conference on Multimedia Big Data (BigMM), IEEE, New Delhi, India, 24–26 September 2020; pp. 152–161.
51. Ren, Y.; Wang, N.; Li, M.; Xu, Z. Deep density-based image clustering. *Knowl.-Based Syst.* **2020**, *197*, 105841. [CrossRef]
52. Basu, S.; Banerjee, A.; Mooney, R.J. Active semi-supervision for pairwise constrained clustering. In Proceedings of the 2004 SIAM International Conference on Data Mining, SIAM, Lake Buena Vista, FL, USA, 22–24 April 2004; pp. 333–344.
53. Zhu, X. *Semi-Supervised Learning Literature Survey (Technical Report)*; Computer Sciences; University of Wisconsin-Madison: Madison, WI, USA, 2005.
54. Vincent, P.; Larochelle, H.; Lajoie, I.; Bengio, Y.; Manzagol, P.-A. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *J. Mach. Learn. Res.* **2010**, *11*, 3371–3408.
55. Hartigan, J.A.; Wong, M.A. A k-means clustering algorithm. *Appl. Stat.* **1979**, *28*, 100–108. [CrossRef]
56. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2017**, arXiv:1412.6980.
57. LeCun, Y. The MNIST Database of Handwritten Digits. Available online: http://yann.lecun.com/exdb/mnist/ (accessed on 17 August 2023).
58. Hull, J.J. A database for handwritten text recognition research. *IEEE Trans. Pattern Anal. Mach. Intell.* **1994**, *16*, 550–554. [CrossRef]
59. Coates, A.; Ng, A.; Lee, H. An analysis of single-layer networks in unsupervised feature learning. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, Fort Lauderdale, FL, USA, 11–13 April 2011; pp. 215–223.
60. Glorot, X.; Bordes, A.; Bengio, Y. Deep sparse rectifier neural networks. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, Fort Lauderdale, FL, USA, 11–13 April 2011; pp. 315–323.
61. Pal, N.; Bezdek, J. On cluster validity for the fuzzy c-means model. *IEEE Trans. Fuzzy Syst.* **1995**, *3*, 370–379. [CrossRef]
62. Zhou, K.; Yang, S. Effect of cluster size distribution on clustering: A comparative study of k-means and fuzzy c-means clustering. *Pattern Anal. Appl.* **2020**, *23*, 455–466. [CrossRef]
63. Zhou, K.; Fu, C.; Yang, S. Fuzziness parameter selection in fuzzy c-means: The perspective of cluster validation. *Sci. China Inf. Sci.* **2014**, *57*, 1–8. [CrossRef]