Semi-Supervised Deep Clustering with Convolutional Autoencoders

Nicola Fiorentino
Department of Computer Science
University of Bari “Aldo Moro”
n.fiorentino3@studenti.uniba.it

Abstract

The problem of clustering data into groups falls within the scope of unsupervised learning. When dealing with high dimensional data, we can exploit neural networks to learn features that are suitable for clustering purposes. This approach is called Deep Clustering. In this work, we show how performance can be significantly improved by injecting a small amount of supervision. The experimental phase will be performed on the Fashion MNIST dataset.

1. Introduction

The goal of clustering is grouping a set of items so that those within a cluster are similar to each other and different from items in other clusters. Traditional clustering algorithms do not work well with high dimensional inputs, due to unreliable similarity metrics. This phenomenon is known as the curse of dimensionality.

A common way to mitigate the problem is transforming data into a lower dimensional space using feature selection or dimension reduction techniques. Nevertheless, extracting meaningful features can be difficult when dealing with raw pixel data. In addition, dimension reduction techniques like Principal Component Analysis ignore non-linear relationships between the original input and the latent space. To overcome these limits, researchers exploited deep neural networks to combine feature learning and clustering. This approach, known as Deep Clustering, provides a unified framework in which the task of mapping the input data to a lower dimensional space is jointly optimized with the task of finding a set of clusters in this embedded space.

Conventional clustering looks for similarities between unlabeled data. Unlike supervised approaches, deep clustering is not forced to learn useful features for a classification task. Nevertheless, in many situations some information regarding clusters is provided in advance [1]. For example, the cluster labels of a subset of instances are available or some items are known to belong to a single cluster. In this context, semi-supervised learning tries to exploit both labeled and unlabeled data. This approach has the advantage of not requiring large amounts of human annotations, while providing a form of background knowledge.

2. Related Work

In literature outstanding performances have been achieved with classification tasks. The drawback of classification algorithms is the need for large amounts of labeled data, which are often expensive to collect. This work attempts to leverage a small amount of supervision to improve the performance of a clustering framework. The model used as backbone is *Deep Convolutional Embedded Clustering* (DCEC) [2].

DCEC exploits Convolutional AutoEncoders (CAE) to learn features from unlabeled images. A traditional autoencoder consists of an encoder and a decoder and learns to reconstruct its input data. Compared to traditional autoencoders, CAE allow to incorporate the spatial relationships between pixels. Specifically, some convolutional layers are stacked on the input images to extract hierarchical features. The last convolutional layer is then flattened, followed by a fully connected layer with 10 units. The small size of this embedded layer forces the autoencoder to capture the most salient features in the data. After that, a fully connected layer and some convolutional transpose layers transform the embedded features back to the original image. The model is trained by minimizing the mean squared error between the input and the output.

An earlier attempt to extend Deep Clustering through supervision is provided in [5]. Since the mentioned approach does not use convolution, performances are poor when applied to complex image datasets. Moreover, supervision is limited to pairwise constraints, which simply specify if two given instances belong to the same cluster (must-link constraint) or to different clusters (cannot-link constraint). In this work, we consider a different kind of supervision. We assume that the labels of some instances are provided as ground truth.

3. Data

Original authors tested DCEC on MNIST and USPS datasets. These consist of simple handwritten digits and the framework is able to achieve high performance even without supervision. However, the situation changes if we consider more complex datasets. The proposed approach will be tested on the Fashion MNIST dataset, containing 70,000 images depicting Zalando's articles. Each example is a 28×28 grayscale image associated with one of 10 classes. Pixel values are normalized to range from 0 to 1.

4. Methods

4.1. Deep Convolutional Embedded Clustering

In the following we provide a brief description of Deep Convolutional Embedded Clustering. The framework consists of CAE with a clustering layer connected to the embedded layer. The clustering layer computes the probability of assigning the embedded point z_i of input image x_i to a given cluster. Specifically, cluster centers $\{\mu_j\}_1^K$ are treated as trainable weights and each embedded point z_i is mapped into soft label q_i by Student's t-distribution:

$$q_{ij} = \frac{\left(1 + \|z_i - \mu_j\|^2\right)^{-1}}{\sum_j \left(1 + \|z_i - \mu_j\|^2\right)^{-1}} \tag{1}$$

where q_{ij} is the jth entry of q_i and represents the probability of z_i belonging to cluster j. The probabilities are then used to calculate a target distribution:

$$p_{ij} = \frac{q_{ij}^2 / \sum_i q_{ij}}{\sum_j \left(q_{ij}^2 / \sum_i q_{ij}\right)} \tag{2}$$

which is designed to place more emphasis on data points assigned with greater confidence. The clustering loss L_c is defined as Kullback-Leibler divergence between the distribution of soft labels and the target distribution:

$$L_c = KL(P\|Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{3}$$

Since the embedded space could be distorted by only using the clustering loss, the reconstruction loss L_r of CAE is added to the objective and optimized along with clustering loss simultaneously. In this way, the structure of data is preserved, avoiding the corruption of feature space. Overall, the objective of DCEC is given by:

$$L = L_r + \lambda L_c \tag{4}$$

where λ is a coefficient that controls the degree of distorting embedded space. Intuitively, CAE is used to learn embedded features while the clustering loss guides the learned features to be prone to forming clusters.

The training phase consists of two steps. First, CAE is pretrained to learn the initial embedded features. Cluster centers are initialized by performing k-means on embedded feature space. Secondly, feature learning and clustering are jointly optimized using (4). CAE's weights and cluster centers are updated through backpropagation and mini-batch SGD. To avoid instability, the target distribution P is updated every T iterations.

The training process stops if the change of label assignments between two consecutive updates of the target distribution is less than a threshold δ or the maximum number of iterations is reached.

4.2. Supervision injection

In DCEC the target distribution P simulates a ground truth soft label. The proposed approach modifies the target distribution by injecting a small amount of real supervision. Let's assume the labels of a subset S of instances are known in advance. The improved target distribution is defined as follows:

$$p_{ij}^* = \begin{cases} p_{ij} & \text{if } x_i \notin S \\ \mathbb{1}_j(x_i) & \text{if } x_i \in S \end{cases} \tag{5}$$

where $\mathbb{1}_j(x_i)$ is an indicator function which assumes value 1 if x_i belongs to cluster j and 0 otherwise.

When $S = \emptyset$ we use k-means to initialize cluster centers. However, when $S \neq \emptyset$ we perform the initialization in the following way:

$$\mu_j = \frac{1}{|S_j|} \sum_{i: x_i \in S_j} z_i \tag{6}$$

where S_j is the set of instances we know belong to cluster j. Proceeding in this way, the framework guarantees the consistency between the learned features and the background knowledge.

4.3. Visualization

When the training stops, embedded features are projected in a two-dimensional space to allow for visualization. To this end, we employ a dimension reduction technique known as UMAP (Uniform Manifold Approximation and Projection) [3]. This method attempts to reduce dimensions in a way that preserves as much of the structure as possible and scales well with large datasets.

5. Experiment Setup

The structure of the framework reflects the one adopted in the original paper. In particular, the encoder network is $\text{conv}_{32}^5 \rightarrow \text{conv}_{64}^5 \rightarrow \text{conv}_{128}^3 \rightarrow \text{FC}_{10}$, where conv_n^k denotes a convolutional layer with n filters, kernel size of $k \times k$ and stride length 2 as default. The decoder mirrors the encoder.

CAE are pretrained for 200 epochs using Adam optimization. The size of the batch is 256. The number of clusters is set equal to the number of ground-truth categories. When no supervision is provided, cluster centers are initialized using k-means with 20 restarts. The update interval is $T = 140$ and the parameter $\lambda = 1$. The convergence threshold is set to $\delta = 0.1\%$, with a maximum number of iterations of $20,000$. The implementation is based on Keras.

The framework is evaluated when different amounts of supervision are considered. The supervision is controlled by a parameter denoting the percentage of instances whose cluster label is known. To select these instances we use random sampling without replacement.

The performance are evaluated by adopting two external metrics [4]. The first metric is the unsupervised clustering accuracy:

$$ACC = \max_{m} \frac{\sum_{i=1}^{N} \mathbb{1}\left\{y_i = m\left(c_i\right)\right\}}{N} \tag{7}$$

where y_i is the ground-truth label, c_i is the cluster assignment generated by the algorithm, and m is a mapping function which ranges over all possible one-to-one mappings between assignments and labels. Intuitively, this metric searches for the best matching between cluster assignments and ground truth. This is necessary because unsupervised algorithms can use a different label than the ground truth to represent a cluster. The second metric is the normalized mutual information:

$$NMI(Y, C) = \frac{I(Y, C)}{\frac{1}{2}[H(Y) + H(C)]} \tag{8}$$

where Y denotes the ground-truth labels, C denotes the clusters labels, I is the mutual information and H is the entropy. For both metrics, values range from 0 to 1.

6. Results

Table 1 and Figure 1 describe the performance on the Fashion MNIST dataset as the percentage of supervision varies. When no supervision is provided, the model achieves a clustering accuracy of 0.58. Assuming only 1% of labeled data, accuracy increases by 15%. This can be seen as a significant improvement. The obtained accuracy reaches 0.80 if we consider a supervision equal to 10% of the instances. By providing greater amounts of supervision, the accuracy grows at a lower rate. Since labeled data is expensive to acquire, this approach allows to reach a fair trade-off between costs and performance.

Figure 2 shows the two-dimensional projections of the features learned as the amount of supervision varies. Instances with different class labels are marked with different colors. By injecting increasing amounts of supervision, the purity of the clusters increases as evidenced by greater separation of colors.

Supervision	ACC	NMI
0%	0.5847	0.6294
1%	0.7371	0.6666
3%	0.7604	0.6872
5%	0.7722	0.6968
10%	0.8022	0.7254
20%	0.8653	0.7891
30%	0.9034	0.8331
50%	0.9391	0.8800

Table 1: Performance of the framework on Fashion MNIST as the amount of supervision varies.

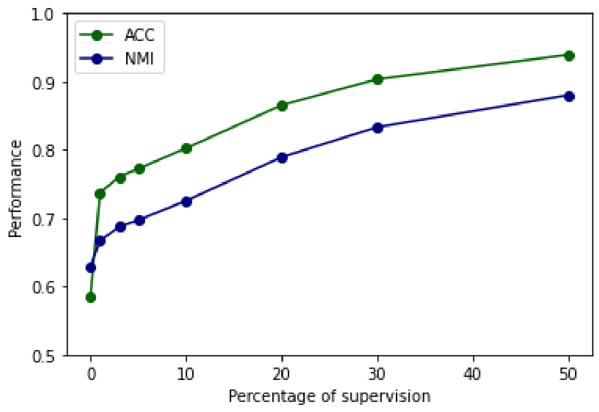


Figure 1: Performance chart on Fashion MNIST as the amount of supervision varies.

7. Conclusion

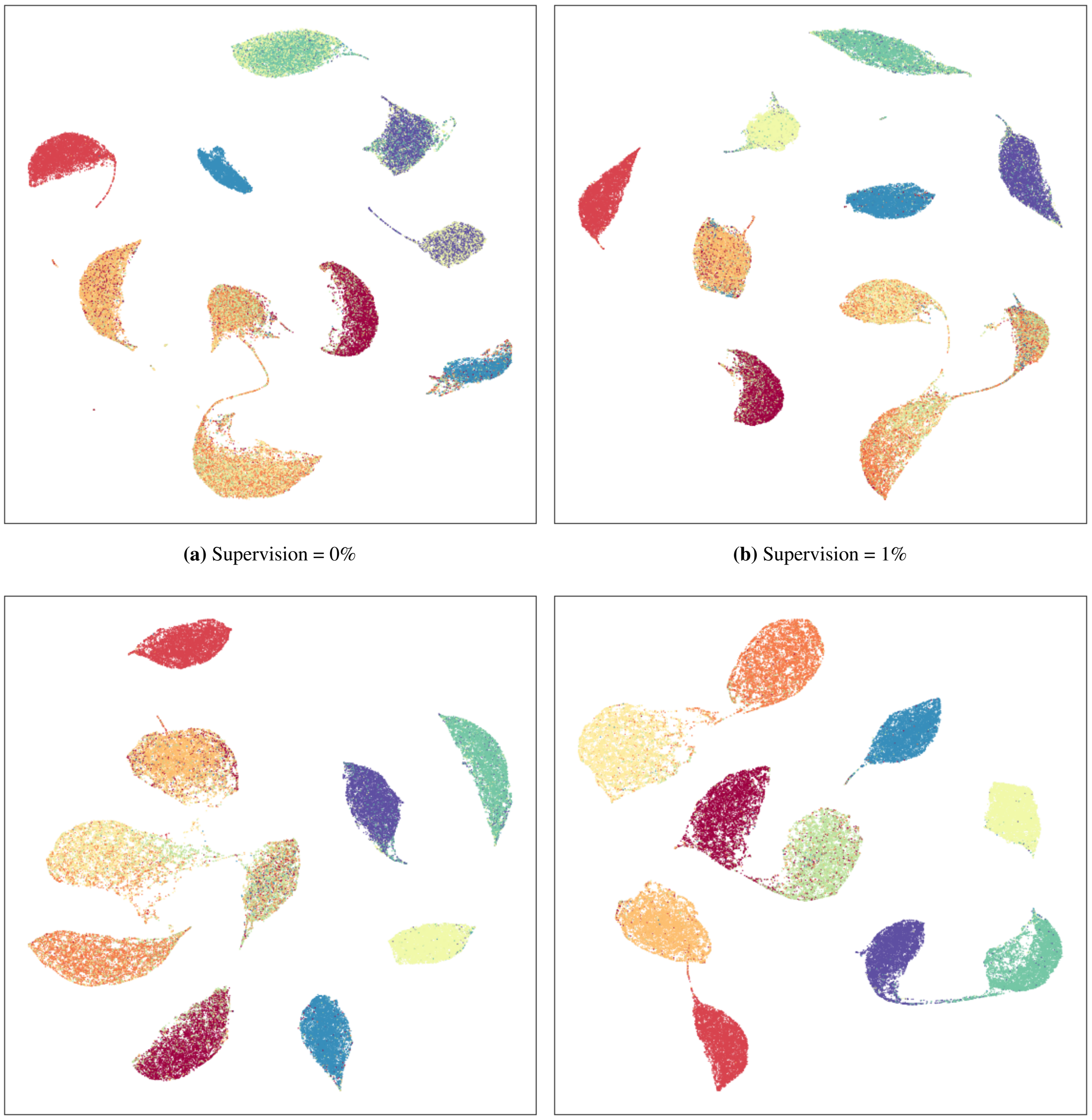
DCEC is able to exploit the features extracted with CAE to cluster images into groups. The obtained results show how the use of a small amount of supervision can significantly improve the performance of the clustering task. Since the percentage of supervision is set by the user, the achievable performance increases as the availability of labeled data increases. To extensively verify the contribution of supervision, the framework should be tested on more complex image datasets.

References

[1] Eric Bair. Semi-supervised clustering methods. *WIREs Computational Statistics*, 5(5):349–361, 2013.

[2] Xifeng Guo, Xinwang Liu, En Zhu, and Jianping Yin. Deep clustering with convolutional autoencoders. In *Neural Information Processing*, pages 373–382. Springer International Publishing, 2017.

[3] Leland McInnes, John Healy, and James Melville. UMAP: Uniform Manifold Approximation and Projection for Dimen-



(a) Supervision = 0%

(b) Supervision = 1%

(c) Supervision = 10%

(d) Supervision = 50%

Figure 2: Two-dimensional projections of the features learned as the amount of supervision varies. Images are generated with Uniform Manifold Approximation and Projection. Different colors mark different class labels.

sion Reduction. *arXiv e-prints*, page arXiv:1802.03426, Feb. 2018.

[4] Erxue Min, Xifeng Guo, Qiang Liu, Gen Zhang, Jianjing Cui, and Jun Long. A survey of clustering with deep learning: From the perspective of network architecture. *IEEE Access*, 6:39501–39514, 2018.

[5] Yazhou Ren, Kangrong Hu, Xinyi Dai, Lili Pan, Steven C.H. Hoi, and Zenglin Xu. Semi-supervised deep embedded clustering. *Neurocomputing*, 325:121–130, 2019.