

# Transferable Deep Metric Learning for Clustering

Simo Alami.C<sup>1,2</sup>, Rim Kaddah<sup>2</sup>, and Jesse Read<sup>1</sup>

<sup>1</sup> Department of Computer Science, Ecole Polytechnique, Palaiseau, France  
`{mohamed.alami-chehboune, jesse.read}@polytechnique.edu`

<sup>2</sup> IRT SystemX, Palaiseau, France  
`rim.kaddah@irt-systemx.fr`

**Abstract.** Clustering in high dimension spaces is a difficult task; the usual distance metrics may no longer be appropriate under the curse of dimensionality. Indeed, the choice of the metric is crucial, and it is highly dependent on the dataset characteristics. However a single metric could be used to correctly perform clustering on multiple datasets of different domains. We propose to do so, providing a framework for learning a transferable metric. We show that we can learn a metric on a labelled dataset, then apply it to cluster a different dataset, using an embedding space that characterises a desired clustering in the generic sense. We learn and test such metrics on several datasets of variable complexity (synthetic, MNIST, SVHN, omniglot) and achieve results competitive with the state-of-the-art while using only a small number of labelled training datasets and shallow networks.

**Keywords:** Clustering · Transfer Learning · Metric Learning.

## 1 Introduction

Clustering is the unsupervised task of assigning a categorical value  $y_i \in \{1, \dots, k\}$  to each data point  $x_i \in \mathbf{X}$ , where no such example categories are given in the training data; i.e., we should map  $\mathbf{X} = \{x_1, \dots, x_n\} \mapsto \mathbf{Y} = \{y_1, \dots, y_n\}$  with  $\mathbf{X}$  the input matrix of  $n$  data points, each of dimension  $d$ ; where  $y_i = \kappa$  implies that data point  $x_i$  is assigned to the  $\kappa$ -th cluster.

Clustering methods complete this task by measuring similarity (the distance) between training pairs, using a similarity function  $s(x_i, x_j) \in \mathbb{R}_+$ . This similarity function should typically reflect subjective criteria fixed by the user. Basically, this means that the user decides what makes a good clustering. As mentioned in [6], “since classes are a high-level abstraction, discovering them automatically is challenging, and perhaps impossible since there are many criteria that could be used to cluster data (e.g., we may equally well cluster objects by colour, size, or shape). Knowledge about some classes is not only a realistic assumption, but also indispensable to narrow down the meaning of clustering”. Taking the example of MNIST [11], one usually groups the same numbers together because these numbers share the highest amount of features (e.g., mutual

information based models do that). However one may want to group numbers given their roundness. In this case, we may obtain two clusters, namely straight shaped numbers (i.e., 1, 4,7) and round shaped numbers (i.e., all the others). Both clustering solutions are relevant, since each clustering addresses a different yet possible user subjective criteria (i.e., clustering semantics).

Finding an automated way to derive and incorporate user criteria in a clustering task based on intended semantics can be very hard. Nowadays, the wide availability of shared annotated datasets is a valuable asset and provides examples of possible user criteria. Hence, we argue that, given “similar” annotated data, classification logic can be used to derive a user criteria that one can apply to clustering similar non-annotated data. For example, we consider the situation where a human is placed in front of two datasets, each one consisting of letters of a certain alphabet she does not understand. The first dataset is annotated, grouping the same letters together. Only by seeing the first dataset, the person can understand the grouping logic used (grouping same geometrical shapes together) and replicate that logic to the second non annotated dataset and cluster correctly its letters.

In this paper, we are interested in tackling the problem of clustering data when the logic (i.e., user clustering criteria) is encoded into some available labelled datasets. This raises two main challenges, namely (1) find a solution that works well on the classification task but (2) ensure transferability in its decision mechanism so it is applicable to clustering data from a different domain.

We believe that addressing these challenges calls for the design of a scoring function that should be as general as possible to ensure transferability but is specific enough not to miss the user criteria. More specifically, the scoring function should be a comparing the logic used to produce a certain clustering to the one used to produce clusterings of the already seen training datasets. Using the concept of logic is useful as a logic is general enough to be used on any dataset and specific enough as it is the main common property shared by all training dataset. Our goal is then to find a suitable metric that retrieves and encapsulate the seen concept for scoring a clustering outcome.

Moreover, modern applications require solutions that are effective when data is of high dimension (i.e., large  $d$ ). While distance-based approaches are broadly used for clustering (e.g., Euclidean distance), we argue that they are not suitable for our problem since they would yield in data specific models in addition to their poor performance in high dimensional spaces due to the curse of dimensionality. To lower dimensionality, a solution is to perform instance-wise embeddings  $x_i \mapsto z_i$ , e.g., with an autoencoder. However this mechanism is still domain specific.

To achieve training on more general patterns, we think it is necessary to take the dataset in its entirety. Therefore, instead of learning a metric that compares pairs of data points in a dataset instance (like a similarity measure), a learned metric is applied to sets of data points so comparison is done between sets. The

metric can be intuitively understood as a distance between the logic underlying a given clustering and the general logic that was used to produce clusterings in training datasets.

For this, we propose a solution where we use a graph autoencoder [9] to embed a set of data points into a vector of chosen dimension. Then, we use the critic part of a Wasserstein GAN (WGAN) [1] to produce a continuous score of the embedded clustering outcome. This critic represents the metric we seek. Thus, our main contributions are:

- We provide a framework for joint metric learning and clustering tasks.
- We show that our proposed solution yields a learned metric that is transferable to datasets of different sizes and dimensions, and across different domains (either vision or tabular) and tasks.
- We obtain results competitive to the state-of-the-art with only a small number of training datasets, relatively simple networks, and no prior knowledge (only an upper bound of the cluster number that can be set to a high value).
- Our method is scalable to large datasets both in terms of number of points or dimensions (e.g the SVHN dataset used in section 4) as it does not have to compute pairwise distances and therefore does not heavily suffer when the number of points or dimensions increase.
- We test the metric on datasets of varying complexity and perform on par with the state-of-the-art while maintaining all the advantages cited above.

## 2 Related Work

Using auto-encoders before applying classic clustering algorithms resulted in a significant increase of clustering performance, while still being limited by these algorithms capacity. Deep Embedding Clustering (DEC) [19] gets rid of this limitation at the cost of more complex objective functions. It uses an auto-encoder along with a cluster assignment loss as a regularisation. The obtained clusters are refined by minimising the KL-divergence between the distribution of soft labels and an auxiliary target distribution. DEC became a baseline for deep clustering algorithms. Most deep clustering algorithms are based on classical center-based, divergence-based or hierarchical clustering formulations and hence bear limitations like the need for an *a priori* number of clusters.

MPCKMeans [2] is more related to metric learning as they use constraints for both metric learning and the clustering objective. However, their learned metrics remain dataset specific and are not transferable.

Constrained Clustering Network (CCN) [8], learns a metric that is transferable across domains and tasks. Categorical information is reduced to pairwise constraints using a similarity network. Along with the learned similarity function, the authors designed a loss function to regularise the clustering classification. But, using similarity networks only captures local properties instance-wise rather than global geometric properties of dataset clustering. Hence, the learned

metric remains non fully transferable, and requires to adapt the loss to the domain to which the metric is transferred to.

In Deep Transfer Clustering (DTC) [6] and Autonovel [7], the authors tackle the problem of discovering novel classes in an image collection given labelled examples of other classes. They extended DEC to a transfer learning setting while estimating the number of classes in the unlabelled data. Autonovel uses self-supervised learning to train the representation from scratch on the union of labelled and unlabelled datasets then trains the data representation by optimizing a joint objective function on the labelled and unlabelled subsets of data. We consider these two approaches as our state of the art baselines.

### 3 Our Framework

To restate our objective, we seek an evaluation metric

$$\begin{aligned} r : \mathbb{R}^{n \times d} \times \mathbb{N}^n &\rightarrow \mathbb{R} \\ (\mathbf{X}, \mathbf{y}) &\mapsto \mathbf{r}(\mathbf{X}, \mathbf{y}) \end{aligned} \quad (1)$$

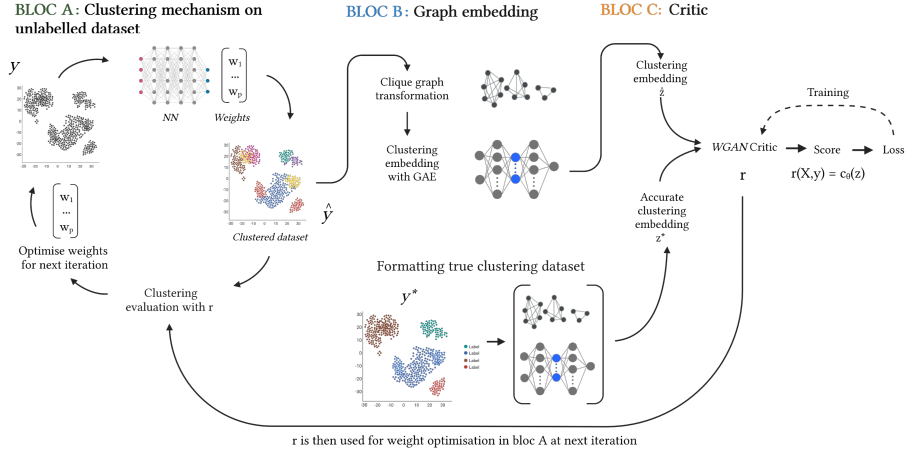
where  $\mathbf{X} \in \mathbb{R}^{n \times d}$  is a dataset of  $n$  points in  $d$  dimensions and  $\mathbf{y} \in \mathbb{N}^n$  a partition of  $\mathbf{X}$  (i.e. a clustering of  $\mathbf{X}$ ). Metric  $r$  should provide a score for *any* labelled dataset of any dimensionality; and in particular this score should be such that  $r(\mathbf{X}, \mathbf{y})$  is high when the hamming distance between the ground truth labels  $\mathbf{y}^*$  and  $\mathbf{y}$  is small (taking cluster label permutations into account). This would mean that we could perform clustering on any given dataset, simply by solving an optimisation problem even if such a dataset had not been seen before.

Formally stated, our goal is: (1) to produce a metric  $r$  that grades the quality of a clustering such that  $\mathbf{y}^* = \arg \max_{\mathbf{y}} \mathbf{r}(\mathbf{X}, \mathbf{y})$ ; (2) Implement an optimisation algorithm that finds  $\mathbf{y}^*$ ; (3) use (1) and (2) to perform a clustering on a new unrelated and unlabelled dataset. We use a collection  $\mathcal{D} = \{\mathbf{X}_l, \mathbf{y}_l^*\}_{l=1}^\ell$  of labelled datasets as examples of correctly ‘clustered’ datasets, and learn  $r$  such that  $\mathbb{E}[r(\mathbf{X}, \mathbf{y})]$  is high. In order to make  $r$  transferable between datasets, we embed each dataset with its corresponding clustering  $(\mathbf{X}_l, \mathbf{y}_l)$  into a vector  $\mathbf{z}_l \in \mathbb{R}^e$ . More formally, the embedding function is of the form:

$$\begin{aligned} g : \mathbb{R}^{n \times d} \times \mathbf{Y} &\rightarrow \mathbb{R}^e \\ (\mathbf{X}, \mathbf{y}) &\mapsto \mathbf{z} \end{aligned} \quad (2)$$

Therefore, the metric  $r$  is actually the composition of two functions  $g$  and  $c_\theta$  (the scoring function from  $\mathbb{R}^e$  to  $\mathbb{R}$ ). Our training procedure is structured around 3 blocs A, B and C detailed in next sections and depicted in figure 1 and is summarised in the following main steps:

- Bloc A. step 1 Select a labelled dataset  $(\mathbf{X}, \mathbf{y}^*) \sim \mathcal{D}$
- Bloc A. step 2 Given a metric function  $r$  (output from bloc B step 2, or initialised randomly), we perform a clustering of dataset  $\mathbf{X}$ :  $\hat{\mathbf{y}} = \arg \max_{\mathbf{y}} r(\mathbf{X}, \mathbf{y})$
- Bloc B. step 1  $\mathbf{y}^*$  and  $\hat{\mathbf{y}}$  are represented as graphs where each clique represents a cluster.
- Bloc B. step 2 Graph convolutional autoencoders perform feature extraction from  $\hat{\mathbf{y}}$  and  $\mathbf{y}^*$  and output embeddings  $\hat{\mathbf{z}}$  and  $\mathbf{z}^*$
- Bloc C. step 1 The metric  $r$  is modelled by a WGAN critic that outputs evaluations of the clusterings:  $r(\mathbf{X}, \mathbf{y}^*) = c_\theta(\mathbf{z}^*)$  and  $r(\mathbf{X}, \hat{\mathbf{y}}) = c_\theta(\hat{\mathbf{z}})$
- Bloc C. step 2 Train the model using the error between  $r(\mathbf{X}, \mathbf{y}^*)$  and  $r(\mathbf{X}, \hat{\mathbf{y}})$ .



**Fig. 1:** Our framework’s 3 components: the clustering mechanism (A), the GAE (B) and the WGAN (C). (A) takes an unlabelled dataset  $\mathbf{X}$  as input and outputs a clustering  $\hat{\mathbf{y}}$  that maximises a metric  $r$ .  $\hat{\mathbf{y}}$  is then turned into a graph  $\mathcal{G}(\mathbf{X}, \hat{\mathbf{y}})$  then into an embedding vector  $\hat{\mathbf{z}}$  using (B). Same goes for the correctly labelled dataset, which is embedded as  $\mathbf{z}^*$ . Then, (C), which is the metric itself, evaluates  $\hat{\mathbf{z}}$  and  $\mathbf{z}^*$  using  $c_\theta$  and is trained to produce a new metric  $r$  which is then used for (A) in the next iteration.

### 3.1 Clustering mechanism

We seek the most suitable optimisation algorithm for clustering given  $r$ . Considering a neural network that performs the clustering, we need to find its weights  $w$  such that the metric is maximised (see equation (3)). The type of algorithm to use depends on the nature of the metric  $r$  to optimise on.

$$\text{CEM}_r(\mathbf{X}) \xrightarrow{\text{finds}} w^* = \arg \max_w r(\mathbf{X}, \mathbf{y}^w) \quad (3)$$

Where  $\mathbf{y}^w$  is a clustering obtained with the weights  $w$ . The metric is assumed to hold certain properties, discussed in 3.3:

**Algorithm 1** CEM Algorithm

---

**Input:** Dataset  $X \in \mathbb{R}^{n \times d}$ ; score function  $r$ ;  $\mu \in \mathbb{R}^d$  and  $\sigma \in \mathbb{R}^d$ ; elite percentage to retain  $p$ ;  $n$  samples of  $w_i \sim \mathcal{N}(\mu, \text{diag}(\sigma))$ ;  $T$  number of iterations  
**for** iteration = 1 **to**  $T$  **do**  
    Produce  $n$  samples of neural network weights  $w_i \sim \mathcal{N}(\mu, \text{diag}(\sigma))$   
    Produce clusterings  $y_i$  of  $X$  using each  $w_i$   
    Evaluate  $r_i = r(X, y_i)$   
    Constitute the elite set of  $p\%$  best  $w_i$   
    Fit a Gaussian distribution with diagonal covariance to the elite set and get a new  $\mu_t$  and  $\sigma_t$   
**end for**  
**return:**  $\mu, w^*$

---

- **Unique Maximum:** A unique optimal clustering.  $r$  has a unique maximum.
- **Continuity**<sup>3</sup>: Any two clusterings  $\mathbf{y}$  and  $\mathbf{y}'$  should be similar if  $r(\mathbf{y})$  and  $r(\mathbf{y}')$  are close in  $\mathbb{R}$  space. Hence,  $r$  has to satisfy a continuity constraint.

There is no guarantee that the best metric for the clustering task is differentiable. Given the above assumptions, conditions are favourable for evolutionary strategies (ES) to iteratively converge towards the optimal solution. Indeed, if  $r$  is continuous and the series  $((\mathbf{X}, \mathbf{y}_1), \dots, (\mathbf{X}, \mathbf{y}_p))$  converges towards  $(\mathbf{X}, \mathbf{y}^*)$  then  $(r(\mathbf{X}, \mathbf{y}_1), \dots, r(\mathbf{X}, \mathbf{y}_p))$  converges towards  $r(\mathbf{X}, \mathbf{y}^*)$ . We choose the Cross-Entropy Method (CEM) [3], a popular ES algorithm for its simplicity, to optimise the clustering neural network weights by solving Eq.(3) (algorithm 1).

### 3.2 Graph based dataset embedding

To capture global properties and be transferable across different datasets, we argue that it is necessary to input all the points of a dataset at once. Hence, instead of pairwise similarities between random pairs of points, we propose to get a representation of the relation between a bunch of neighbouring points. Thus, we represent each dataset by a graph structure  $\mathcal{G}(\mathbf{X}, \mathbf{y})$  where each node corresponds to a point in  $\mathbf{X}$  and where cliques represent clusters as shown in figure 1. This representation takes the form of a feature matrix  $X$  and an adjacency matrix  $A$ . Using  $X$ , and  $A$ , we embed the whole dataset into a vector  $\mathbf{z} \in \mathbb{R}^e$ . To do so, we use graph autoencoders (GAE). Our implementation is based on [9].

We obtain  $z \in \mathcal{M}_{n,m}$  which is dependent of the shape of the dataset (where  $m$  is a user specified hyper-parameter). In order to make it independent from the number of points in  $\mathcal{X}$ , we turn the matrix  $z$  into a square symmetrical one  $z \leftarrow z^T z \in \mathcal{M}_{m,m}$ . The final embedding corresponds to a flattened version of the principal triangular bloc of  $z^T z$ , which shape is  $\mathbf{e} = (\frac{m+1}{2}, 1)$ . However, the scale

<sup>3</sup> As a reminder, Let  $T$  and  $U$  be two topological spaces. A function  $f : T \mapsto U$  is continuous in the open set definition if for every  $t \in T$  and every open set  $u$  containing  $f(t)$ , there exists a neighbourhood  $v$  of  $t$  such that  $f(v) \subset u$ .

of the output still depends on the number of points in the dataset. This could cause an issue when transferring to datasets with a vastly different number of data points. It should therefore require some regularisation; in order to simplify, we decided to use datasets with approximately the same number of points.

### 3.3 A critic as a metric

With embedded vectors of the same shape, we compare the clusterings proposed  $\hat{\mathbf{z}}$  and the ground truth ones  $\mathbf{z}$  using the metric  $r$ .  $r$  is a function mapping an embedding vector  $\mathbf{z} \in \mathbb{R}^e$  to  $\mathbb{R}$ , we therefore parameterise it as:

$$r_\alpha(\mathbf{X}, \mathbf{y}) = r_\alpha(\mathbf{z}) = \alpha_1 \phi_1(\mathbf{z}) + \alpha_2 \phi_2(\mathbf{z}) + \dots + \alpha_h \phi_h(\mathbf{z}) \quad (4)$$

Where  $\phi_j(\mathbf{z}) \in \mathbb{R}$ . As per [13], learning a viable metric is possible provided both the following constraints: (1) maximising the difference between the quality of the optimal decision and the quality of the second best; (2) minimising the amplitude of the metric function as using small values encourages the metric function to be simpler, similar to regularisation in supervised learning.

When maximising the metric difference between the two clusterings that have the highest scores, we get a similarity score as in traditional metric learning problems. The problem is formulated by equation (5) where  $\mathcal{S}$  is a set of solutions (i.e., clustering proposals) found using  $r_\alpha$  and  $\mathbf{y}^*$  is the true clustering,  $\mathbf{y}^{\max}$  is the best solution found in  $\mathcal{S}$ :  $\mathbf{y}^{\max} = \arg \max_{\mathbf{y} \in \mathcal{S}} r_\alpha(\mathbf{X}, \mathbf{y})$ .

$$\begin{aligned} \min_{\alpha} r_\alpha(\mathbf{X}, \mathbf{y}^*) - \max_{\alpha} \min_{\mathbf{y}' \in \mathcal{S} \setminus \mathbf{y}^{\max}} r_\alpha(\mathbf{X}, \mathbf{y}^{\max}) - r_\alpha(\mathbf{X}, \mathbf{y}') \\ \text{s.t. } \mathbf{y}^* = \arg \max_{\mathbf{y} \in \mathbf{Y}} r(\mathbf{y}) \end{aligned} \quad (5)$$

To solve equation (5), we use a GAN approach where the clustering mechanism (i.e., CEM) plays the role of the generator while a critic (i.e., metric learning model) plays the role of the discriminator. In a classic GAN, the discriminator only has to discriminate between real and false samples, making it use a cross entropy loss. With this kind of loss, and in our case, the discriminator quickly becomes too strong. Indeed, the score output by the discriminator becomes quickly polarised around 0 and 1.

For this reason, we represent  $r$  as the critic of a WGAN [1]. This critic scores the realness or fakeness of a given sample while respecting a smoothing constraint. The critic measures the distance between data distribution of the training dataset and the distribution observed in the generated samples. Since WGAN assumes that the optimal clustering provided is unique, the metric solution found by the critic satisfies equation (5) constraints.  $r$  reaching a unique maximum while being continuous, the assumptions made in section 3.1 are correctly addressed. To train the WGAN, we use the loss  $\mathcal{L}$  in equation (6) where  $\hat{\mathbf{z}}$  is the embedding vector of a proposed clustering and  $\mathbf{z}$  is the embedding vector of the desired clustering. Our framework is detailed in algorithm 2.

$$\mathcal{L}(\mathbf{z}^*, \hat{\mathbf{z}}) = \max_{\theta} \mathbb{E}_{\mathbf{z}^* \sim p} [f_{\theta}(\mathbf{z}^*)] - \mathbb{E}_{\hat{\mathbf{z}} \sim p(\hat{\mathbf{z}})} [f_{\theta}(\hat{\mathbf{z}})] \quad (6)$$

**Algorithm 2** Critic2Metric (C2M)

---

**Input:**  $b$ : batch size,  $epoch$ : number of epochs;  $p$ : percentage of elite weights to keep;  
 $iteration$ : number of CEM iterations;  $population$ : number of weights to generate;  
 $\mu \in \mathbb{R}^d$ : CEM mean;  $\sigma \in \mathbb{R}^d$ : CEM standard deviation,  $\theta$ : critic's weights










```

for  $n = 1$  to  $epoch$  do
  for  $k = 1$  to  $b$  do
    Sample  $(\mathbf{X}_k, \mathbf{y}_k^*) \sim \mathcal{D}$  a correctly labelled dataset
    Generate ground truth embeddings  $\mathbf{z}(\mathbf{x}_k, \mathbf{y}_k^*) = GAE(\mathcal{G}(\mathbf{X}_k, \mathbf{y}_k^*))$ 
    Initialise clustering neural network weights  $\{w_j\}_{j=1}^{population}$ 
    for  $i = 1$  to  $iteration$  do
      for  $j = 1$  to  $population$  do
        Generate clusterings  $\hat{\mathbf{y}}_k^{w_j}$ 
        Convert  $\hat{\mathbf{y}}_k^{w_j}$  into a graph
         $\mathbf{z}(\mathbf{x}_k, \hat{\mathbf{y}}_k^{w_j}) = GAE(\mathcal{G}(\mathbf{X}_k, \hat{\mathbf{y}}_k^{w_j}))$ 
        Evaluate:  $r(\mathbf{X}_k, \hat{\mathbf{y}}_k^{w_j}) = c_\theta(\mathbf{z}(\mathbf{x}_k, \hat{\mathbf{y}}_k^{w_j}))$ 
      end
      Keep proportion  $p$  of best weights  $w_p$ 
       $w^* \leftarrow \text{CEM}(w_p, \mu, \sigma)$ 
    end
    Generate clustering  $\mathbf{y}_k^{w^*}$ 
     $\mathbf{z}(\mathbf{x}_k, \hat{\mathbf{y}}_k^{w^*}) = GAE(\mathcal{G}(\mathbf{X}_k, \hat{\mathbf{y}}_k^{w^*}))$ 
    Train critic as in [1] using  $\mathbf{z}(\mathbf{x}_k, \hat{\mathbf{y}}_k^{w^*})$  and  $\mathbf{z}(\mathbf{x}_k, \mathbf{y}_k^*)$ 
  end
end

```

---

## 4 Experiments

Dataset family	Synthetic data				MNIST			Street view house numbers	Omniglot
Dataset	Blob	Moon	Circles	Anisotropic	MNIST-digits [11]	letters MNIST [4]	fashion MNIST [18]	SVHN [12]	Omniglot [10]
Snapshot									
Feature dimension	2	2	2	2	$28 \times 28$	$28 \times 28$	$28 \times 28$	$32 \times 32$	$105 \times 105$
Maximum number of clusters	9 (custom)	9 (custom)	9 (custom)	9 (custom)	10	26	10	10	47
Size	200 (custom)	200 (custom)	200 (custom)	200 (custom)	60000	145600	60000	73257	32460

**Table 1:** Datasets description

For empirical evaluation, we parameterise our framework as follows: The critic (block C in Fig 1) is a 5 layer network of sizes 256, 256, 512, 512, and 1 (output) neurons. All activation functions are LeakyRelu ( $\alpha = 0.2$ ) except last layer (no activation). RMSprop optimizer with 0.01 initial learning rate and a decay rate of 0.95. The CEM-trained neural network (bloc A in Fig 1) has 1 hidden layer of size 16 with Relu activation, and a final layer of size  $k = 50$  (the maximum number of clusters). The GAE (bloc B in Fig 1) has 2 hidden layers; sized 32 and 16 for synthetic datasets, and 100 and 50 for real datasets.



We choose datasets based on 3 main criteria: having a similar compatible format; datasets should be large enough to allow diversity in subsampling configurations to guarantee against overfitting; datasets should be similar to the ones used in our identified baseline literature. All used datasets are found in table 1.

For training, we construct  $n$  sample datasets and their ground truth clustering, each containing 200 points drawn randomly from a set of 1500 points belonging to the training dataset. Each one of these datasets, along with their clustering is an input to our model. To test the learned metric, we construct 50 new sample datasets from datasets that are different from the training one (e.g., if we train the model on MNIST numbers, we will use datasets from MNIST letters or fashion to test the metric). The test sample datasets contain 200 points each for synthetic datasets and 1000 points each otherwise. The accuracies are then averaged accross the 50 test sample datasets. To test the ability of the model to learn using only a few samples, we train it using 5 (few shots) and 20 datasets (standard), each containing a random number of clusters. For few shots trainings, we train the critic for 1 epoch and 10 epochs for standard trainings.

To evaluate the clustering, we use Normalised-Mutual Information (NMI) [16] and clustering accuracy (ACC) [20]. NMI provides a normalised measure that is invariant to label permutations while ACC measures the one-to-one matching of labels. For clustering, we only need that the samples belonging to the same cluster are attributed the same label, independently from the label itself. However, since we want to analyse the behaviour of the metric learned through our framework, we are interested in seeing whether it is permutation invariant or not. Hence, we need the two measures.

#### 4.1 Results on 2D synthetic datasets

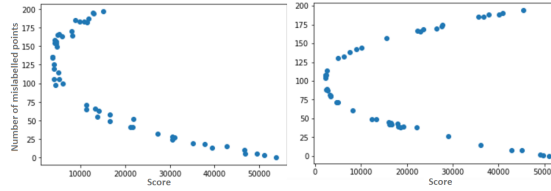
Analysis on synthetic datasets (see table 1) proves that our model behaves as expected. We do not compare our results to any baseline since existing unsupervised methods are well studied on them. We train our model using exclusively samples from blobs datasets. We then test the learned metric on the 4 different types of synthetic datasets (blobs, anisotropic, moons and circles). Results are displayed in table 2. We observe that the model obtains the best score on blobs since it is trained using this dataset. We can also notice that our model achieves high scores for the other types of datasets not included in training.

Types of datasets	Standard training		Few shots training	
	ACC	NMI	ACC	NMI
Blobs	98.4%	0.980	97.3%	0.965
Anisotropic	97.9%	0.967	97.2%	0.945
Circles	91.7%	0.902	92.7%	0.900
Moons	92.1%	0.929	92.8%	0.938

**Table 2:** Average ACC and NMI on synthetic test datasets.

Our model succeeds in clustering datasets presenting non linear boundaries like circles while blobs datasets used in training are all linearly separable. Hence, the model learns intrinsic properties of training dataset that are not portrayed in the initial dataset structure, and thus that the metric appears to be transferable.

**Critic’s ablation study.** To test if the critic behaves as expected, i.e., grades the clustering proposals proportionally to their quality, we test it on wrongly labelled datasets to see if the score decreases with the number of mislabelled points. We consider 50 datasets from each type of synthetic datasets, create 50 different copies and mislabel a random number of points in each copy. A typical result is displayed in figure 2 and shows that the critic effectively outputs an ordering metric as the score increases when the number of mislabelled points decreases, reaching its maximum when there is no mislabelled point. This shows that the metric satisfies the constraints stated in equation 5.



**Fig. 2:** Metric values (i.e., scores given by the critic) for several clusterings of a dataset. Plots are from an anisotropic dataset (left) and a moons dataset (right). In a 2 cluster case (right), the formula used to compute mislabelled points has been made sensitive to label permutation to verify if permuted labels can fool the critic. The critic assigns a high score either when all the labels match the given ground truth or when all the labels are permuted (which again does not affect the correctness of the clustering)

An interesting behaviour is shown in figure 2. Recall that since we are in the context of a clustering problem, we only need for the samples belonging to the same cluster to get the same label, independently from the cluster label itself. Thus, the formula used to compute mislabelled points has been made sensitive to label permutation to verify if permuted labels can fool the critic. For instance, in a 2 clusters case, one can switch the labels of all points in each cluster and still get the maximum score. Switching all labels makes all the points wrongly labelled compared to the given ground truth but nonetheless the clustering itself remains true. This explains the rounded shape in figure 2 where the used datasets in the right panel only consisted of 2 clusters. The critic assigns a high score either when all the labels match the given ground truth or when all the labels are permuted (which does not affect the correctness of the clustering).

## 4.2 Results on MNIST datasets

MNIST datasets give similar results both in terms of ACC and NMI on all test datasets regardless of the used training dataset (see table 3). Hence, the model effectively capture implicit features that are dataset independent. While standard training shows better results, the few shots training has close performance.

Training Dataset	Testing Dataset					
	Numbers		Letters		Fashion	
	ACC	NMI	ACC	NMI	ACC	NMI
Numbers (standard)	72.3%	0.733	81.3%	0.861	65.2%	0.792
Numbers (few shots)	68.5%	0.801	79.0%	0.821	61.8%	0.672
Letters (standard)	75.9%	0.772	83.7%	0.854	67.5%	0.800
Letters (few shots)	69.8%	0.812	78.7%	0.806	60.9%	0.641
Fashion (standard)	70.6%	0.706	83.4%	0.858	72.5%	0.762
Fashion (few shots)	70.1%	0.690	82.1%	0.834	70.7%	0.697

**Table 3:** Mean clustering performance on MNIST dataset.

Training Dataset	Testing Dataset					
	Numbers		Letters		Fashion	
	Best	Top 3	Best	Top 3	Best	Top 3
Numbers (standard)	78.3%	92.5%	86.0%	97.5%	69.2%	87.2%
Numbers (few shots)	75.8%	82.1%	83.3%	92.0%	65.1%	83.9%
Letters (standard)	77.4%	89.2%	88.8%	96.4%	70.2%	86.7%
Letters (few shots)	73.1%	80.6%	85.1%	91.5%	61.0%	76.3%
Fashion (standard)	70.1%	83.1%	85.0%	98.6%	76.9%	94.7%
Fashion (few shots)	67.9%	77.4%	83.5%	95.3%	70.2%	88.0%

**Table 4:** Critic based performance assessment: Best corresponds to the percentage of times the critic gives the best score to the desired solution. Top 3 is when this solution is among the 3 highest scores.

Table 4 shows the percentage of times the critic attributes the best score to the desired solution. It shows that ES algorithm choice has a significant impact on the overall performance. Even with a metric that attributes the best score to the desired clustering, the CEM may be stuck in a local optimum and fails to reconstruct back the desired clustering. Hence, a better optimisation can enhance the performance shown in table 3 closer to the one presented in table 4.

### 4.3 Comparative study

We compare our approach with baseline methods from the literature (table 5). For some methods, we followed the procedure in [8] and used their backbone neural network as a pairwise similarity metric. Table 5a reports results when training on SVHN and testing on MNIST numbers. We obtain close ACC values to CCN and ATDA [14]. These methods uses Omniglot as an auxiliary dataset to learn a pairwise similarity function, which is not required for our model. Our model only uses a small fraction of SVHN, has shallow networks and does not require any adaptation to its loss function to achieve comparable results. Finally, other cited methods require the number of clusters as an a priori indication. We achieve comparable results without needing this information. When the loss

adaptation through Omniglot is discarded (denoted source-only in table 5a), or if the number of clusters is not given, their accuracy falls and our model surpasses them by a margin.

Method	ACC		Method	ACC	NMI
	Loss	Adaptation Source Only			
DANN [5]	73.9%	54.9%	k-means	18.9%	0.464
LTR [15]	78.8%	54.9%	CSP [17]	65.4%	0.812
ATDA [14]	86.2%	70.1%	MPCK-means [2]	53.9%	0.816
CCN [8]	89.1%	52%	CCN [8]	78.18%	0.874
Ours (standard)	—	84.3%	DTC [6]	87.0%	0.945
Ours (few shots)	—	81.4%	Autonovel [7]	85.4%	—
			Ours (standard)	83.4%	0.891

(a) Unsupervised cross-task transfer from SVHN to MNIST digits. (b) Unsupervised cross-task transfer from Omniglot<sub>train</sub> to Omniglot<sub>test</sub> ( $k = 100$  for all).

**Table 5:** Comparative clustering performance

Table 5b reports results when training on Omniglot<sub>train</sub> and testing on Omniglot<sub>test</sub>. Values are averaged across 20 alphabets which have 20 to 47 letters. We set the maximum number of clusters  $k = 100$ . When the number of clusters is unknown, we get an ACC score relatively close to DTC and Autonovel. Compared to these two approaches, our method bears several significant advantages:

- **Deep Networks:** DTC and Autonovel used Resnets as a backbone which are very deep networks while we only used shallow networks (2 layers maximum)
- **Pairwise similarity:** in Autonovel the authors used a pairwise similarity statistic between datasets instances which we aimed to avoid due to its significant computational bottleneck. Moreover, this metric is recalculated after each training epoch, which adds more complexity.
- **Vision tasks:** While DTC can only handle vision tasks, we present a more general framework which includes vision but also tabular datasets.
- **Number of classes:** DTC and Autonovel used the labelled dataset as a probe dataset, and estimates the number of classes iteratively, and when the labelled clusters are correctly recovered, they used the ACC metric to keep the best clustering. This approach is effective, but requires access to the labelled dataset at inference time to estimate the number of classes. This is a shortcoming (memory or privacy limitations). Our approach does not require the labelled dataset once the metric is learned. Our metric automatically estimates the number of clusters required to any new unlabelled dataset.

## 5 Conclusion

We presented a framework for cross domain/task clustering by learning a transferable metric. This framework consisted of ES methods, and GAE alongside a critic. Our model extracts dataset-independent features from labelled datasets that characterise a given clustering, performs the clustering and grades its quality. We showed successful results using only small datasets and relatively shallow

architectures. Moreover, there is more room for improvement. Indeed, since our framework is composed of 3 different blocs (CEM, GAE, critic), overall efficiency can be enhanced by independently improving each bloc (i.e replacing CEM).

In future work, we will study the criteria that determine why some auxiliary datasets are more resourceful than others given a target dataset. In our case, this means to study for instance why using the MNIST letters dataset as training allowed a better performance on Fashion MNIST than when using MNIST numbers. This would allow to deliver a minimum performance guarantee at inference time by creating a transferability measure between datasets.

**Acknowledgements:** We gratefully acknowledge Orianne Debeaupuis for making the figure. We also acknowledge computing support from NVIDIA. This work was supported by funds from the French Program "Investissements d’Avenir".

## References

1. Arjovsky, M., et al.: Wasserstein generative adversarial networks. In: ICML. pp. 214–223 (2017)
2. Bilenko, M., Basu, S., Mooney, R.J.: Integrating constraints and metric learning in semi-supervised clustering. ICML p. 11 (2004)
3. de Boer, P.T., Kroese, D.P., et. al: A tutorial on the cross-entropy method. *Ann. Oper. Res.* **134**(1), 19–67 (feb 2005)
4. Cohen, G., et al.: Emnist: Extending mnist to handwritten letters. IJCNN (2017)
5. Ganin, et al.: Domain-adversarial training of neural networks. JMLR (2016)
6. Han, K., et al.: Learning to discover novel visual categories via deep transfer clustering (2019)
7. Han, K., Rebuffi, S.A., et. al: AutoNovel: Automatically discovering and learning novel visual categories. PAMI pp. 1–1 (2021)
8. Hsu, et al.: Learning to cluster in order to transfer across domains and tasks (2017)
9. Kipf, T.N., Welling, M.: Variational graph auto-encoders (2016)
10. Lake, B.M., Salakhutdinov, R., Tenenbaum, J.B.: Human-level concept learning through probabilistic program induction. *Science* **350**(6266), 1332–1338 (2015)
11. LeCun, Y., Cortes, C., Burges, C.: Mnist handwritten digit database (2010)
12. Netzer, Y., et. al: Reading digits in natural images with unsupervised feature learning. NIPS Workshop on Deep Learning and Unsupervised Feature Learning (2011)
13. Ng, A.Y., Russell, S.: Algorithms for inverse reinforcement learning. In: in Proc. 17th International Conf. on Machine Learning. pp. 663–670 (2000)
14. Saito, K., Ushiku, Y., Harada, T.: Asymmetric tri-training for unsupervised domain adaptation. ICML p. 2988–2997 (2017)
15. Sener, O., Song, H.O., et al.: Learning transfer able representations for unsupervised domain adaptation. NIPS p. 2110–2118 (2016)
16. Strehl, A., Ghosh, J.: Cluster ensembles—a knowledge reuse framework for combining multiple partitions. JMLR **3**(Dec), 583–617 (2002)
17. Wang, X., Qian, B., Davidson, I.: On constrained spectral clustering and its applications. *Data Mining and Knowledge Discovery* p. 1–30 (2014)
18. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms (2017)
19. Xie, J., Girshick, R., Farhadi, A.: Unsupervised deep embedding for clustering analysis. In: ICML. pp. 478–487 (20–22 Jun 2016)
20. Yang, Y., Xu, D., et. al: Image clustering using local discriminant models and global integration. *IEEE Transactions on Image Processing* **19**(10), 2761–2773 (2010)