HAL
open science

# Deep Transformation-Invariant Clustering

Tom Monnier, Thibault Groueix, Mathieu Aubry

## To cite this version:

Tom Monnier, Thibault Groueix, Mathieu Aubry. Deep Transformation-Invariant Clustering. Advances in Neural Information Processing Systems 33 (2020), Dec 2020, Vancouver, Canada. hal-03015110

**HAL Id: hal-03015110**

**https://hal.science/hal-03015110v1**

Submitted on 19 Nov 2020

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Deep Transformation-Invariant Clustering

**Tom Monnier** **Thibault Groueix** **Mathieu Aubry**

LIGM, Ecole des Ponts, Univ Gustave Eiffel, CNRS, France
{tom.monnier,thibault.groueix,mathieu.aubry}@enpc.fr

## Abstract

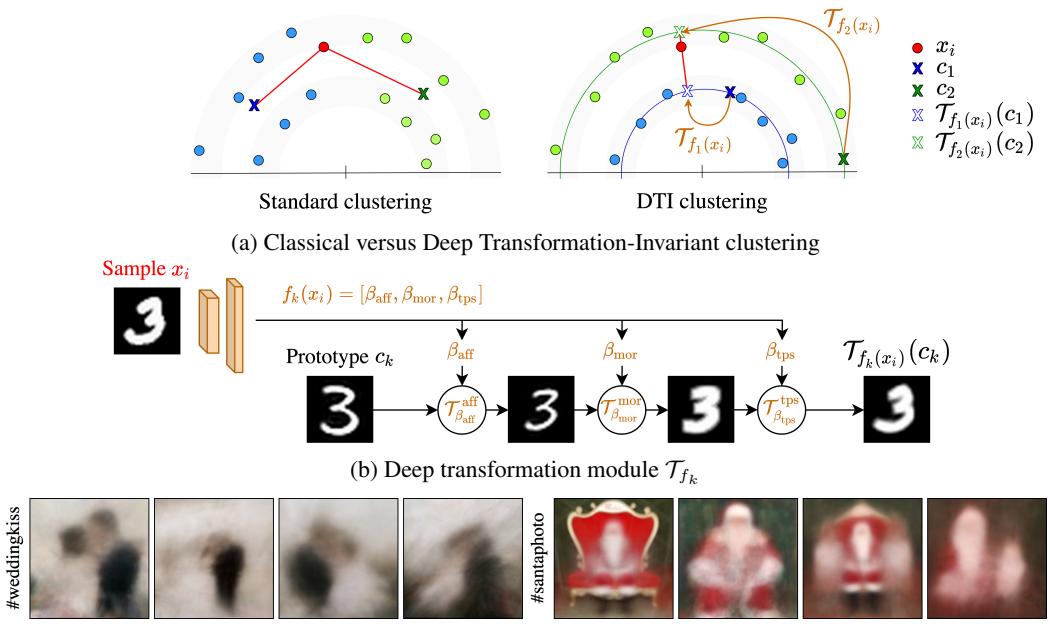
Recent advances in image clustering typically focus on learning better deep representations. In contrast, we present an orthogonal approach that does not rely on abstract features but instead learns to predict transformations and performs clustering directly in pixel space. This learning process naturally fits in the gradient-based training of K-means and Gaussian mixture model, without requiring any additional loss or hyper-parameters. It leads us to two new deep transformation-invariant clustering frameworks, which jointly learn prototypes and transformations. More specifically, we use deep learning modules that enable us to resolve invariance to spatial, color and morphological transformations. Our approach is conceptually simple and comes with several advantages, including the possibility to easily adapt the desired invariance to the task and a strong interpretability of both cluster centers and assignments to clusters. We demonstrate that our novel approach yields competitive and highly promising results on standard image clustering benchmarks. Finally, we showcase its robustness and the advantages of its improved interpretability by visualizing clustering results over real photograph collections.

## 1 Introduction

Gathering collections of images on a topic of interest is getting easier every day: simple tools can aggregate data from social media, web search, or specialized websites and filter it using hashtags, GPS coordinates, or semantic labels. However, identifying visual trends in such image collections remains difficult and usually involves manually organizing images or designing an ad hoc algorithm. Our goal in this paper is to design a clustering method which can be applied to such image collections, output a visual representation for each cluster and show how it relates to every associated image.

Directly comparing image pixels to decide if they belong to the same cluster leads to poor results because they are strongly impacted by factors irrelevant to clustering, such as exact viewpoint or lighting. Approaches to obtain clusters invariant to these transformations can be broadly classified into two groups. A first set of methods extracts invariant features and performs clustering in feature space. The features can be manually designed, but most state-of-the-art methods learn them directly from data. This is challenging because images are high-dimensional and learning relevant invariances thus requires huge amounts of data. For this reason, while recent approaches perform well on simple datasets like MNIST, they still struggle with real images. Another limitation of these approaches is that learned features are hard to interpret and visualize, making clustering results difficult to analyze. A second set of approaches, following the seminal work of Frey and Jojic on transformation-invariant clustering [11, 12, 13], uses explicit transformation models to align images before comparing them. These approaches have several potential advantages: (i) they enable direct control of the invariances to consider; (ii) because they do not need to discover invariances, they are potentially less data-hungry; (iii) since images are explicitly aligned, clustering process and results can easily be visualized. However, transformation-invariant approaches require solving a difficult joint optimization problem. In practice, they are thus often limited to small datasets and simple transformations, such as affine

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.




(a) Classical versus Deep Transformation-Invariant clustering

(b) Deep transformation module $\mathcal{T}_{f_k}$

(c) Prototypes learned from unfiltered Instagram images associated to different hashtags

Figure 1: **Overview. (a)** Given a sample $x_i$ and prototypes $c_1$ and $c_2$, standard clustering such as K-means assigns the sample to the closest prototype. Our DTI clustering first aligns prototypes to the sample using a family of parametric transformations - here rotations - then picks the prototype whose alignment yields the smallest distance. **(b)** We predict alignment with deep learning. Given an image $x_i$, each parameter predictor $f_k$ predicts parameters for a sequence of transformations - here affine $\mathcal{T}^{\text{aff}}_{\beta_{\text{aff}}}$, morphological $\mathcal{T}^{\text{mor}}_{\beta_{\text{mor}}}$, and thin plate spline $\mathcal{T}^{\text{tps}}_{\beta_{\text{tps}}}$ - to align prototype $c_k$ to $x_i$. **(c)** Examples of interpretable prototypes discovered from large images sets (15k each) associated to hashtags in Instagram using our DTI clustering with 40 clusters. Each cluster contains from 200 to 800 images.

transformations, and to the best of our knowledge they have never been evaluated on large standard image clustering datasets.

In this paper, we propose a deep transformation-invariant (DTI) framework that enables to perform transformation-invariant clustering at scale and uses complex transformations. Our main insight is to jointly learn deep alignment and clustering parameters with a single loss, relying on the gradient-based adaptations of K-means [38] and GMM optimization [9]. Not only is predicting transformations more computationally efficient than optimizing them, but it enables us to use complex color, thin plate spline and morphological transformations without any specific regularization. Because it is pixel-based, our deep transformation-invariant clustering is also easy to interpret: cluster centers and image alignments can be visualized to understand assignments. Despite its apparent simplicity, we demonstrate that our DTI clustering framework leads to results on par with the most recent feature learning approaches on standard benchmarks. We also show it is capable of discovering meaningful modes in real photograph collections, which we see as an important step to bridge the gap between theoretically well-grounded clustering approaches and semi-automatic tools relying on hand-designed features for exploring image collections, such as AverageExplorer [52] or ShadowDraw [32].

We first briefly discuss related works in Section 2. Section 3 then presents our DTI framework (Fig. 1a). Section 4 introduces our deep transformation modules and architecture (Fig. 1b) and discuss training details. Finally, Section 5 presents and analyzes our results (Fig. 1c).

**Contributions.** In this paper we present:

- a deep transformation-invariant clustering approach that jointly learns to cluster and align images,
- a deep image transformation module to learn spatial alignment, color modifications and for the first time morphological transformations,
- an experimental evaluation showing that our approach is competitive on standard image clustering benchmarks, improving over state-of-the-art on Fashion-MNIST and SVHN, and provides highly interpretable qualitative results even on challenging web image collections.

Code, data, models as well as more visual results are available on our project webpage[1].

## 2 Related work

Most recent approaches to image clustering focus on learning deep image representations, or features, on which clustering can be performed. Common strategies include autoencoders [48, 10, 25, 28], contrastive approaches [49, 5, 44], GANs [6, 51, 41] and mutual information based strategies [22, 18, 24]. Especially related to our work is [28] which leverages the idea of capsule [20] to learn equivariant image features, in a similar fashion of equivariant models [33, 45]. However, our method aims at being invariant to transformations but not at learning a representation.

Another type of approach is to align images in pixel space using a relevant family of transformations, such as translations, rotations, or affine transformations to obtain more meaningful pixel distances before clustering them. Frey and Jojic first introduced transformation-invariant clustering [11, 12, 13] by integrating pixel permutations as a discrete latent variable within an Expectation Maximization (EM) [9] procedure for a mixture of Gaussians. Their approach was however limited to a finite set of discrete transformations. *Congealing* generalized the idea to continuous parametric transformations, and in particular affine transformations, initially by using entropy minimization [40, 30]. A later version using least square costs [7, 8] demonstrated the relation of this approach to the classical Lukas-Kanade image alignment algorithm [37]. In its classical version, congealing only enables to align all dataset images together, but the idea was extended to clustering [36, 39, 34], for example using a Bayesian model [39], or in a spectral clustering framework [34]. These works typically formulate difficult joint optimization problems and solve them by alternating between clustering and transformation optimization for each sample. They are thus limited to relatively small datasets and to the best of our knowledge were never compared to modern deep approaches on large benchmarks. Deep learning was recently used to scale the idea of congealing for global alignment of a single class of images [1] or time series [46]. Both works build on the idea of Spatial Transformer Networks [23] (STN) that spatial transformation are differentiable and can be learned by deep networks. We also build upon STN, but go beyond single-class alignment to jointly perform clustering. Additionally, we extend the idea to color and morphological transformations. We believe our work is the first to use deep learning to perform clustering in pixel space by explicitly aligning images.

## 3 Deep Transformation-Invariant clustering

In this section, we first discuss a generic formulation of our deep transformation-invariant clustering approach, then derive two algorithms based on K-means [38] and Gaussian mixture model [9].

**Notation:** In all the rest of the paper, we use the notation $a_{1:n}$ to refer to the set $\{a_1, \dots, a_n\}$.

### 3.1 DTI framework

Contrary to most recent image clustering methods which rely on feature learning, we propose to perform clustering in pixel space by making the clustering invariant to a family of transformations. We consider $N$ image samples $x_{1:N}$ and aim at grouping them in $K$ clusters using a *prototype method*. More specifically, each cluster $k$ is defined by a prototype $c_k$, which can also be seen as an image, and prototypes are optimized to minimize a loss $\mathcal{L}$ which typically evaluates how well they represent the samples. We further assume that $\mathcal{L}$ can be written as a sum of a loss $l$ computed over each sample:

$$\mathcal{L}(c_{1:K}) = \sum_{i=1}^{N} l(x_i, \{c_1, \dots, c_K\}). \tag{1}$$

Once the problem is solved, each sample $x_i$ will be associated to the closest prototype.

Our key assumption is that in addition to the data, we have access to a group of parametric transformations $\{\mathcal{T}_\beta, \beta \in B\}$ to which we want to make the clustering invariant. For example, one can consider $\beta \in \mathbb{R}^6$ and $\mathcal{T}_\beta$ the 2D affine transformation parametrized by $\beta$. Other transformations are discussed in Section 4.1. Instead of finding clusters by minimizing the loss of Equation 1, one can

[1] http://imagine.enpc.fr/~monniert/DTIClustering/

minimize the following transformation-invariant loss:

$$\mathcal{L}_{\text{TI}}(c_{1:K}) = \sum_{i=1}^{N} \min_{\beta_{1:K}} l(x_i, \{\mathcal{T}_{\beta_1}(c_1), \dots, \mathcal{T}_{\beta_K}(c_K)\}). \tag{2}$$

In this equation, the minimum over $\beta_{1:K}$ is taken for each sample independently. This loss is invariant to transformations of the prototypes (see proof in Appendix B). Also note there is not a single optimum since the loss is the same if any prototype $c_k$ is replaced by $\mathcal{T}_\beta(c_k)$ for any $\beta \in B$. If necessary, for example for visualization purposes, this ambiguity can easily be resolved by adding a small regularization on the transformations. The optimization problem associated to $\mathcal{L}_{\text{TI}}$ is of course difficult. A natural approach, which we use as baseline (noted TI), is to alternatively minimize over transformations and clustering parameters. We show that performing such optimization using a gradient descent can already lead to improved results over standard clustering but is computationally expensive.

We experimentally show it is faster and actually better to instead learn $K$ (deep) predictors $f_{1:K}$ for each prototype, which aim at associating to each sample $x_i$ the transformation parameters $f_{1:K}(x_i)$ minimizing the loss, i.e. to minimize the following loss:

$$\mathcal{L}_{\text{DTI}}(c_{1:K}, f_{1:K}) = \sum_{i=1}^{N} l(x_i, \{\mathcal{T}_{f_1(x_i)}(c_1), \dots, \mathcal{T}_{f_K(x_i)}(c_K)\}), \tag{3}$$

where predictors $f_{1:K}$ are now shared for all samples. We found that using deep parameters predictors not only enables more efficient training but also leads to better clustering results especially with more complex transformations. Indeed, the structure and optimization of the predictors naturally regularize the parameters for each sample, without requiring any specific regularization loss, especially in the case of high numbers $N$ of samples and transformation parameters.

In the next section we present concrete losses and algorithms. We then describe differentiable modules for relevant transformations and discuss parameter predictor architecture as well as training in Section 4.

### 3.2 Application to K-means and GMM

**K-means.** The goal of K-means algorithm [38] is to find a set of prototypes $c_{1:K}$ such that the average Euclidean distance between each sample and the closest prototype is minimized. Following the reasoning of Section 3.1, the loss optimized in K-means can be transformed into a transformation-invariant loss:

$$\mathcal{L}_{\text{DTI K-means}}(c_{1:K}, f_{1:K}) = \sum_{i=1}^{N} \min_{k} \|x_i - \mathcal{T}_{f_k(x_i)}(c_k)\|^2. \tag{4}$$

Following batch gradient-based trainings [3] of K-means, we can then simply jointly minimize $\mathcal{L}_{\text{DTI K-means}}$ over prototypes $c_{1:K}$ and deep transformation parameter predictors $f_{1:K}$ using a batch gradient descent algorithm. In practice, we initialize prototypes $c_{1:K}$ with random samples and predictors $f_{1:K}$ such that $\forall k, \forall x, \mathcal{T}_{f_k(x)} = \text{Id}$.

**Gaussian mixture model.** We now consider that data are observations of a mixture of $K$ multivariate normal random variables $X_{1:K}$, i.e. $X = \sum_k \delta_{k,\Delta} X_k$ where $\delta$ is the Kronecker function and $\Delta \in \{1, \dots, K\}$ is a random variable defined by $P(\Delta = k) = \pi_k$, with $\forall k,\ \pi_k > 0$ and $\sum_k \pi_k = 1$. We write $\mu_k$ and $\Sigma_k$ the mean and covariance of $X_k$ and $G(\,\cdot\,; \mu_k, \Sigma_k)$ associated probability density function. The transformation-invariant negative log-likelihood can then be written:

$$\mathcal{L}_{\text{DTI GMM}}(\mu_{1:K}, \Sigma_{1:K}, \pi_{1:K}, f_{1:K}) = -\sum_{i=1}^{N} \log\Big(\sum_{k=1}^{K} \pi_k G\big(x_i\,; \mathcal{T}_{f_k(x_i)}(\mu_k), \mathcal{T}^*_{f_k(x_i)}(\Sigma_k)\big)\Big), \tag{5}$$

where $\mathcal{T}^*$ is slightly modified version of $\mathcal{T}$. Indeed, $\mathcal{T}$ may include transformations that one can apply to the covariance, such as spatial transformations, and other that would not make sense, such as additive color transformations. We jointly minimize $\mathcal{L}_{\text{DTI GMM}}$ over Gaussian parameters, mixing probabilities, and deep transformation parameters $f_{1:K}$ using a batch gradient-based EM procedure similar to [21, 15, 14] and detailed in Algorithm 1. In practice, we assume that pixels are independent resulting in diagonal covariance matrices.

In such gradient-based procedures, two constraints have to be enforced, namely the positivity and normalization of mixing probabilities $\pi_k$ and the non-negativeness of the diagonal covariance terms.

**Algorithm 1:** Deep Transformation-Invariant Gaussian Mixture Model

**Input:** data $\mathbf{X}$, number of clusters $K$, transformation $\mathcal{T}$
**Output:** cluster assignations, Gaussian parameters $\mu_{1:K}, \Sigma_{1:K}$, deep predictors $f_{1:K}$
**Initialization:** $\mu_{1:K}$ with random samples, $\Sigma_{1:K} = 0.5$, $\eta_{1:K} = 1$ and $\forall k, \forall x, \mathcal{T}_{f_k(x)} = \mathrm{Id}$
**while** *not converged* **do**

i. sample a batch of data points $x_{1:N}$
ii. compute mixing probabilities: $\pi_{1:K} = \mathrm{softmax}(\eta_{1:K})$
iii. compute per-sample Gaussian transformed parameters:

$$\forall k,\ \forall i,\ \ \tilde{\mu}_{ki} = \mathcal{T}_{f_k(x_i)}(\mu_k) \ \text{ and } \ \tilde{\Sigma}_{ki} = \mathcal{T}^*_{f_k(x_i)}(\Sigma_k) + \mathrm{diag}(\sigma^2_{\min})$$

iv. compute responsibilities: $\forall k,\ \forall i,\ \gamma_{ki} = \frac{\pi_k G(x_i\,; \tilde{\mu}_{ki}, \tilde{\Sigma}_{ki})}{\sum_j \pi_j G(x_i\,; \tilde{\mu}_{ji}, \tilde{\Sigma}_{ji})}$ (*E-step*)
v. minimize expected negative log-likelihood w.r.t to $\{\mu_{1:K}, \Sigma_{1:K}, \eta_{1:K}, f_{1:K}\}$:

$$\mathbb{E}[\mathcal{L}_{\text{DTI GMM}}] = -\sum_{i=1}^{N}\sum_{k=1}^{K} \gamma_{ki}\Big(\log\big(G(x_i\,; \tilde{\mu}_{ki}, \tilde{\Sigma}_{ki})\big) + \log(\pi_k)\Big) \qquad (\textit{M-step})$$

**end**

For the mixing probabilities constraints, we adopt the approach used in [21] and [14] which optimize mixing parameters $\eta_k$ used to compute the probabilities $\pi_k$ using a softmax instead of directly optimizing $\pi_k$, which we write $\pi_{1:K} = \mathrm{softmax}(\eta_{1:K})$. For the variance non-negativeness, we introduce a fixed minimal variance value $\sigma^2_{\min}$ which is added to the variances when evaluating the probability density function. This approach is different from the one in [14] which instead use clipping, because we found training with clipped values was harder. In practice, we take $\sigma_{\min} = 0.25$.

# 4 Learning image transformations

## 4.1 Architecture and transformation modules

We consider a set of prototypes $c_{1:K}$ we would like to transform to match a given sample $x$. To do so, we propose to learn for each prototype $c_k$, a separate deep predictor which predicts transformation parameters $\beta$. We propose to model the family of transformations $\mathcal{T}_\beta$ as a sequence of M parametric transformations such that, writing $\beta = (\beta^1, \ldots, \beta^M)$, $\mathcal{T}_\beta = \mathcal{T}^M_{\beta^M} \circ \ldots \circ \mathcal{T}^1_{\beta^1}$. In the following, we describe the architecture of transformation parameter predictors $f_{1:K}$, as well as each family of parametric transformation modules we use. Figure 1b shows our learned transformation process on a MNIST example.

**Parameters prediction network.** For all experiments, we use the same parameter predictor network architecture composed of a shared ResNet [19] backbone truncated after the global average pooling, followed by $K \times M$ Multi-Layer Perceptrons (MLPs), one for each prototype and each transformation module. For the ResNet backbone, we use ResNet-20 for images smaller than $64 \times 64$ and ResNet-18 otherwise. Each MLP has the same architecture, with two hidden layers of 128 units.

**Spatial transformer module.** To model spatial transformations of the prototypes, we follow the spatial transformers developed by Jaderberg et al. [23]. The key idea is to model spatial transformations as a differentiable image sampling of the input using a deformed sampling grid. We use affine $\mathcal{T}^{\text{aff}}_\beta$, projective $\mathcal{T}^{\text{proj}}_\beta$ and thin plate spline $\mathcal{T}^{\text{tps}}_\beta$ [2] transformations which respectively correspond to 6, 8 and 16 (a 4x4 grid of control points) parameters.

**Color transformation module.** We model color transformation with a channel-wise diagonal affine transformation on the full image, which we write $\mathcal{T}^{\text{col}}_\beta$. It has 2 parameters for greyscale images and 6 parameters for colored images. We first used a full affine transformation with 12 parameters, however the network was able to hide several patterns in the different color channels of a single prototype (Appendix C.4). Note that a similar transformation was theoretically introduced in capsules [28], but with the different goal of obtaining a color-invariant feature representation. Deep feature-based approaches often handle color images with a pre-processing step such as Sobel filtering [4, 24, 28]. We believe the way we align colors of the prototypes to obtain color invariance in pixel space is novel, and it enables us to directly work with colored images without using any pre-processing or specific invariant features.

**Morphological transformation module.** We introduce a new transformation module to learn morphological operations [16] such as dilation and erosion. We consider a greyscale image $x \in \mathbb{R}^D$ of size $U \times V = D$, we write $x[u, v]$ the value of the pixel $(u, v)$ for $u \in \{1, \dots, U\}$ and $v \in \{1, \dots, V\}$. Given a 2D region $A$, the dilation of $x$ by $A$, $\mathcal{D}_A(x) \in \mathbb{R}^D$, is defined by $\mathcal{D}_A(x)[u, v] = \max_{(u',v') \in A} \; x[u + u', v + v']$ and its erosion by $A$, $\mathcal{E}_A(x) \in \mathbb{R}^D$, is defined by $\mathcal{E}_A(x)[u, v] = \min_{(u',v') \in A} \; x[u + u', v + v']$. Directly learning the region $A$ which parametrizes these transformations is challenging, we thus propose to learn parameters $(\alpha, a)$ for the following soft version of these transformations:

$$\mathcal{T}^{\text{mor}}_{(\alpha,a)}(x)[u, v] = \frac{\sum_{(u',v') \in W} \; x[u + u', v + v'] \cdot a[u + u', v + v'] \cdot e^{\alpha x[u+u',v+v']}}{\sum_{(u',v') \in W} \; a[u + u', v + v'] \cdot e^{\alpha x[u+u',v+v']}}, \tag{6}$$

where $W$ is a fixed set of 2D positions, $\alpha$ is a softmax (positive values) or softmin (negative values) parameter and $a$ is a set of parameters with values between 0 and 1 defined for every position $(u', v') \in W$. Parameters $a$ can be interpreted as an image, or as a soft version of the region $A$ used for morphological operations. Note that if $a[u', v'] = \mathbf{1}_{\{(u',v') \in A\}}$, when $\alpha \to +\infty$ (resp. $-\infty$), it successfully emulates $\mathcal{D}_A$ (resp. $\mathcal{E}_A$). In practice, we use a grid of integer positions around the origin of size $7 \times 7$ for $W$. Note that since morphological transformations do not form a group, transformation-invariant denomination is slightly abusive.

### 4.2 Training

We found that two key elements were critical to obtain good results: empty cluster reassignment and curriculum learning. We then discuss further implementation details and computational cost.

**Empty cluster reassignment.** Similar to [4], we adopt an empty cluster reassignment strategy during our clustering optimization. We reinitialize both prototype and deep predictor of "tiny" clusters using the parameters of the largest cluster with a small added noise. In practice, the size of balanced clusters being $N/K$, we define "tiny" as less than 20% of $N/K$.

**Curriculum learning.** Learning to predict transformations is a hard task, especially when the number of parameters is high. To ease learning, we thus adopt a curriculum learning strategy by gradually adding more complex transformation modules to the training. Given a target sequence of transformations to learn, we first train our model without any transformation - or equivalently with an identity module - then iteratively add subsequent modules once convergence has been reached. We found this is especially important when modeling local deformations with complex transformations with many parameters, such as TPS and morphological transformations. Intuitively, prototypes should first be coarsely aligned before attempting to refine the alignment with more complex transformations.

**Implementation details.** Both clustering parameters and parameter prediction networks are learned jointly and end-to-end using Adam optimizer [27] with a $10^{-6}$ weight decay on the neural network parameters. We sequentially add transformation modules at a constant learning rate of 0.001 then divide the learning rate by 10 after convergence - corresponding to different numbers of epochs depending on the dataset characteristics - and train for a few more epochs with the smaller learning rate. We use a batch size of 64 for real photograph collections and 128 otherwise.

**Computational cost.** Training DTI K-means or DTI GMM on MNIST takes approximately 50 minutes on a single Nvidia GeForce RTX 2080 Ti GPU and full dataset inference takes 30 seconds. We found it to be much faster than directly optimizing transformation parameters (TI clustering) for which convergence took more than 10 hours of training.

## 5 Experiments

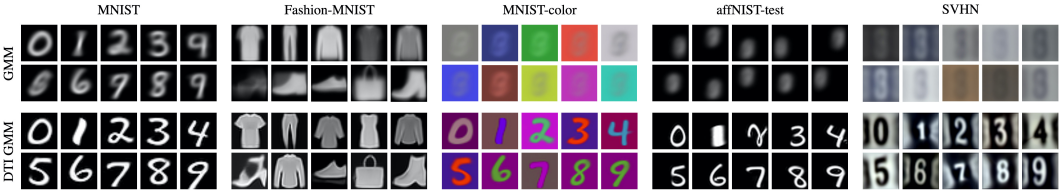
In this section, we first analyze our approach and compare it to state-of-the-art, then showcase its interest for image collection analysis and visualization.

### 5.1 Analysis and comparisons

Similar to previous work on image clustering, we evaluate our approach with global classification accuracy (ACC), where a cluster-to-class mapping is computed using the Hungarian algorithm [29],

Table 1: **Comparisons.** We report ACC and NMI in % on standard clustering benchmarks. Symbols mark methods that use data augmentation ($\triangledown$) and manually selected features as input ($\S$ for pretrained features from best VaDE run, $\dagger$ for GIST features, $\ddagger$ for Sobel filters) and are thus not directly comparable. For SVHN, we also report our results with our Gaussian weighted loss ($\star$). Eval column refers to the aggregate used: best run (*max*), average (*avg*) or run with minimal loss (*minLoss*).

| | | | MNIST | | MNIST-test | | USPS | | F-MNIST | | FRGC | | SVHN |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | Runs | Eval | ACC | NMI | ACC | NMI | ACC | NMI | ACC | NMI | ACC | NMI | ACC |
| *Clustering on a learned feature* | | | | | | | | | | | | | |
| DEC [48, 50] | 9 | max | 86.3 | 83.4 | 85.6 | 83.0 | 76.2 | 76.7 | 51.8 | 54.6 | 37.8 | 50.5 | - |
| InfoGAN [6, 41] | 5 | max | 89.0 | 86.0 | - | - | - | - | 61.0 | 59.0 | - | - | - |
| VaDE [25, 50] | 10 | max | 94.5 | 87.6 | - | - | 56.6 | 51.2 | 57.8 | 63.0 | - | - | - |
| ClusterGAN [41] | 5 | max | 95.0 | 89.0 | - | - | - | - | 63.0 | 64.0 | - | - | - |
| JULE [49] | 3 | avg | 96.4 | 91.3 | 96.1 | 91.5 | 95.0 | 91.3 | 56.3 | 60.8 | 46.1 | 57.4 | - |
| DEPICT [10] | 5 | avg | 96.5 | 91.7 | 96.3 | 91.5 | **96.4** | **92.7** | 39.2 | 39.2 | **47.0** | **61.0** | - |
| DSCDAN [50] | 10 | avg | **97.8** | **94.1** | **98.0** | **94.6** | 86.9 | 85.7 | **66.2** | **64.5** | - | - | - |
| *Clustering on a learned feature with data augmentation and/or ad hoc data representation* | | | | | | | | | | | | | |
| SpectralNet [44] | 5 | avg | 97.1$^{\S}$ | **92.4**$^{\S}$ | - | - | - | - | - | - | - | - | - |
| IMSAT [22] | 12 | avg | 98.4$^{\triangledown}$ | - | - | - | - | - | - | - | - | - | **57.3**$^{\triangledown\dagger}$ |
| ADC [18] | 20 | avg | 98.7$^{\triangledown}$ | - | - | - | - | - | - | - | **43.7**$^{\triangledown}$ | - | 38.6$^{\triangledown}$ |
| SCAE [28] | 5 | avg | 98.7$^{\triangledown}$ | - | - | - | - | - | - | - | - | - | 55.3$^{\ddagger}$ |
| IIC [24] | 5 | avg | 98.4$^{\triangledown}$ | - | - | - | - | - | - | - | - | - | - |
| | 5 | minLoss | **99.2**$^{\triangledown}$ | - | - | - | - | - | - | - | - | - | - |
| *Clustering on pixel values* | | | | | | | | | | | | | |
| K-means [38] | 10 | avg | 54.8 | 50.2 | 55.9 | 51.2 | 65.3 | 61.2 | 54.1 | 51.4 | 22.7 | 26.5 | 12.2 |
| GMM [9] | 10 | avg | 54.2 | 51.7 | 55.6 | 54.7 | 66.0 | 60.9 | 49.7 | 51.2 | 24.2 | 27.9 | 11.6 |
| **DTI K-means** | 10 | avg | **97.3** | **94.0** | 96.6 | 94.6 | 86.4 | 88.2 | 61.2 | 63.7 | 39.6 | 48.7 | 36.4 / 44.5$^{\star}$ |
| | 10 | minLoss | 97.2 | 93.8 | **98.0** | **95.3** | **89.8** | **89.5** | 57.4 | 64.1 | 41.1 | 49.7 | 39.6 / 62.6$^{\star}$ |
| **DTI GMM** | 10 | avg | 95.9 | 93.2 | 97.8 | 94.7 | 84.5 | 87.2 | 59.6 | 62.2 | 40.1 | 48.9 | 36.7 / 57.4$^{\star}$ |
| | 10 | minLoss | 97.1 | 93.7 | **98.0** | 95.1 | 87.3 | 89.0 | **68.2** | **66.3** | **41.6** | **51.1** | 39.5 / **63.3**$^{\star}$ |




(a) Prototypes learned for different datasets

(b) Transformations predicted for all prototypes for 4 MNIST images

Figure 2: **Qualitative results. (a)** compares prototypes learned from GMM and our DTI GMM, **(b)** shows transformed prototypes given query samples from MNIST and highlight the closest prototype.

and Normalized Mutual Information (NMI). Datasets and corresponding transformation modules we used are described in Appendix A.

**Comparison on standard benchmarks.** In Table 1, we report our results on standard image clustering benchmarks, i.e. digit datasets (MNIST [31], USPS [17]), a clothing dataset (Fashion-MNIST [47]) and a face dataset (FRGC [43]). We also report results for SVHN [42] where concurrent methods use pre-processing to remove color bias. In the table, we separate representation-based from pixel-based methods and mark results using data augmentation or manually selected features as input. Note that our results depend on initialization, we provide detailed statistics in Appendix C.1.

Our DTI clustering is fully unsupervised and does not require any data augmentation, ad hoc features, nor any hyper-parameter while performing clustering directly in pixel space. We report average performances and performances of the minimal loss run which we found to correlate well with high performances (Appendix C.2). Because this non-trivial criterion allows to automatically select a run in a fully unsupervised way, we argue it can be compared to average results from competing methods which don't provide such criterion.

First, DTI clustering achieves competitive results on all datasets, in particular improving state-of-the-art by a significant margin on SVHN and Fashion-MNIST. For SVHN, we first found that the

Table 2: **Augmented and specific datasets.** Clustering accuracy (%) with standard deviation for methods applied on raw images (no pre-processing). We used 10 runs for our method and 5 for the baselines.

| Method | Eval | MNIST-1k | MNIST-color | affNIST-test |
|---|---|---|---|---|
| VaDE [25] | avg | 49.6 (5.6) | 11.9 (1.2) | Div. |
| IMSAT [22] | avg | 67.9 (2.3) | 10.6 (0.1) | 18.2 (2.6) |
| IIC [24] | avg | 63.4 (0.4) | 10.6 (0.0) | 57.6 (0.0) |
| | minLoss | 63.2 | 10.6 | 57.6 |
| **DTI K-means** | avg | 79.8 (6.9) | 96.7 (0.1) | 95.5 (3.3) |
| | minLoss | **90.5** | **96.8** | **97.0** |
| **DTI GMM** | avg | 80.8 (7.2) | 96.0 (0.2) | 93.3 (5.9) |
| | minLoss | 87.1 | 95.8 | **97.0** |

Table 3: **Ablation study on MNIST.** Clustering accuracy (%) over 10 runs.

| Method | Avg | MinLoss |
|---|---|---|
| **DTI clustering (aff-morpho-tps)** | **97.3** | **97.2** |
| ordering: aff-tps-morpho | 95.5 | 96.9 |
| ordering: morpho-aff-tps | 27.5 | 97.0 |
| w/o morphological | 94.8 | 95.8 |
| w/o thin plate spline | 90.0 | 82.5 |
| w/o affine | 85.1 | 96.8 |
| affine only | 90.1 | 90.5 |
| w/o empty cluster reassignment | 80.9 | 78.6 |
| w/o curriculum learning | 83.9 | 78.9 |
| TI clustering (aff-morpho-tps, 1 run) | 26.3 | 26.3 |
| TI clustering (affine only) | 73.0 | 73.1 |

prototypes quality was harmed by digits on the side of the image. To pay more attention to the center digit, we weighted the clustering loss by a Gaussian weight ($\sigma = 7$). It led to better prototypes and allowed us to improve over all concurrent methods by a large margin. Compared to representation-based methods, our pixel-based clustering is highly interpretable. Figure 2a shows standard GMM prototypes and our prototypes learned with DTI GMM which appear to be much sharper than standard ones. This directly stems from the quality of the learned transformations, visualized in Figure 2b. Our transformation modules can successfully align the prototype, adapt the thickness and apply local elastic deformations. More alignment results are available on our project webpage.

**Augmented and specific datasets.** DTI clustering also works on small, colored and misaligned datasets. In Table 2, we highlight these strengths on specifics datasets generated from MNIST: MNIST-1k is a 1000 images subset, MNIST-color is obtained by randomly selecting a color for the foreground and background and affNIST-test[2] is the result of random affine transformations. We used an online implementation[3] for VaDE [25] and official ones for IMSAT [22] and IIC [24] to obtain baselines. Our results show that the performances of DTI clustering is barely affected by spatial and color transformations, while baseline performances drop on affNIST-test and are almost chance on MNIST-color. Figure 2a shows the quality and interpretability of our cluster centers on affNIST-test and MNIST-color. DTI clustering also seems more data-efficient than the baselines we tested.

**Ablation on MNIST.** In Table 3, we conduct an ablation study on MNIST of our full model trained following Section 4.2 with affine, morphological and TPS transformations. We first explore the effect of transformation modules. Their order is not crucial, as shown by similar minLoss performances, but can greatly affect the stability of the training, as can be seen in the average results. Each module contributes to the final performance, affine transformations being the most important. We then validate our training strategy showing that both empty cluster reassignment and curriculum learning for the different modules are necessary. Finally, we directly optimize the loss of Equation 2 (TI clustering) by optimizing the transformation parameters for each sample at each iteration of the batch clustering algorithm, without using our parameter predictors. With rich transformations which have many parameters, such as TPS and morphological ones, this approach fails completely. Using only affine transformations, we obtain results clearly superior to standard clustering, but worse than ours.

### 5.2 Application to web images

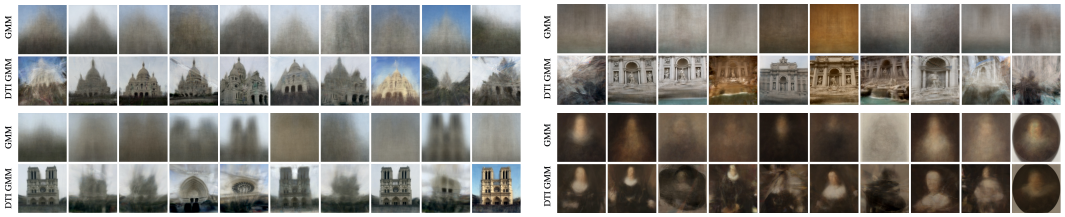
One of the main interest of our DTI clustering is that it allows to discover trends in real image collections. All images are resized and center cropped to $128 \times 128$. The selection of the number of clusters is a difficult problem and is discussed in Appendix C.3.

In Figure 1c, we show examples of prototypes discovered in very large unfiltered sets (15k each) of Instagram images associated to different hashtags[4] using DTI GMM applied with 40 clusters. While many images are noise and are associated to prototypes which are not easily interpretable, we show prototypes where iconic photos and poses can be clearly identified. To the best of our knowledge, we believe we are the first to demonstrate this type of results from raw social network image collections.

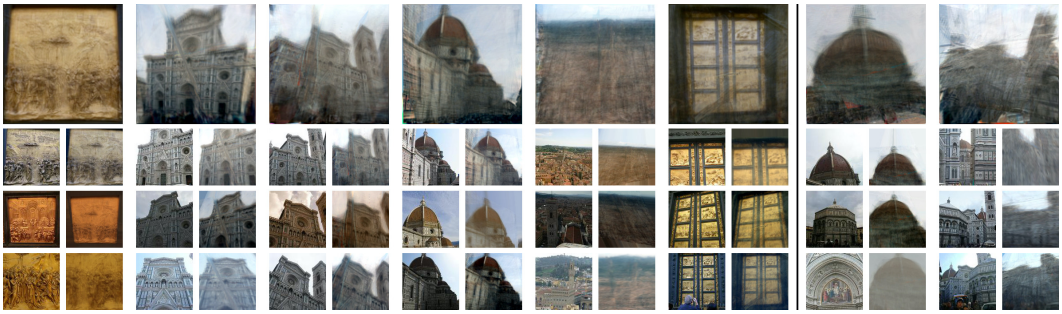
[2] https://www.cs.toronto.edu/~tijmen/affNIST/

[3] https://github.com/GuHongyang/VaDE-pytorch

[4] https://github.com/arc298/instagram-scraper was used to scrape photographs




(a) Full sets of prototypes discovered with GMM and DTI GMM



(b) Examples of cluster centers and aligned images with DTI GMM (20 clusters)

Figure 3: **Qualitative results on real photographs.** **(a)** Clustering results from photographs of different locations in [35] (1,089 Sacre Coeur top-left, 1,688 Trevi fountain top-right, 2,625 Notre-Dame bottom-left) and 980 Baroque portraits from [26] (bottom-right). **(b)** Clustering results from 1,892 Florence cathedral images from [35]. Top row shows learned prototypes while the three bottom rows show examples of images from each cluster and aligned prototypes. These clusters contain respectively 44, 154, 134, 64, 71, 133, 85 and 64 images. The left six examples are successful clusters while the two right clusters are relative failure cases.

Comparable results in AverageExplorer [52], e.g. on Santa images, could be obtained using ad hoc features and user interactions, while our results are produced fully automatically.

Figure 3 shows qualitative clustering results on MegaDepth [35] and WikiPaintings [26]. Similar to our results on image clustering benchmarks, our learned prototypes are more relevant and accurate than the ones obtained from standard clustering. Note that some of our prototypes are very sharp: they typically correspond to sets of photographs between which we can accurately model deformations, e.g. scenes that are mostly planar, with little perspective effects. On the contrary, more unique photographs and photographs with strong 3D effects that we cannot model will be associated to less interpretable and blurrier prototypes, such as the ones in the last two columns of Figure 3b. In Figure 3b, in addition to the prototypes discovered, we show examples of images contained in each cluster as well as the aligned prototype. Even for such complex images, the simple combination of our color and spatial modules manages to model real image transformations like illumination variations and viewpoint changes. More web image clustering results are shown on our project webpage.

## 6 Conclusion

We have introduced an efficient deep transformation-invariant clustering approach in raw input space. Our key insight is the online optimization of a single clustering objective over clustering parameters and deep image transformation modules. We demonstrate competitive results on standard image clustering benchmarks, including improvements over state-of-the-art on SVHN and Fashion-MNIST. We also demonstrate promising results for real photograph collection clustering and visualization. Finally, note that our DTI clustering framework is not specific to images and can be extended to other types of data as long as appropriate transformation modules are designed beforehand.

## Acknowledgements

This work was supported in part by ANR project EnHerit ANR-17-CE23-0008, project Rapid Tabasco, gifts from Adobe and HPC resources from GENCI-IDRIS (Grant 2020-AD011011697). We thank Bryan Russell, Vladimir Kim, Matthew Fisher, François Darmon, Simon Roburin, David Picard, Michaël Ramamonjisoa, Vincent Lepetit, Elliot Vincent, Jean Ponce, William Peebles and Alexei Efros for inspiring discussions and valuable feedback.

## Broader Impact

The impact of clustering mainly depends on the data it is applied on. For instance, adding structure in user data can raise ethical concerns when users are assimilated to their cluster, and receive targeted advertisement and newsfeed. However, this is not specific to our method and can be said of any clustering algorithm. Also note that while our clustering can be applied for example to data from social media, the visual interpretation of the clusters it returns via the cluster centers respects privacy much better than showing specific examples from each cluster.

Because our method provides highly interpretable results, it might bring increased understanding of clustering algorithm results for the broader public, which we think may be a significant positive impact.

## References


[1] R. Annunziata, C. Sagonas, and J. Cali. Jointly Aligning Millions of Images with Deep Penalised Reconstruction Congealing. In *ICCV*, 2019.

[2] F. Bookstein. Principal Warps: Thin-Plate Splines and the Decomposition of Deformations. *TPAMI*, 1989.

[3] L. Bottou and Y. Bengio. Convergence Properties of the K-Means Algorithms. In *NIPS*, 1995.

[4] M. Caron, P. Bojanowski, A. Joulin, and M. Douze. Deep Clustering for Unsupervised Learning of Visual Features. In *ECCV*, 2018.

[5] J. Chang, L. Wang, G. Meng, S. Xiang, and C. Pan. Deep Adaptive Image Clustering. In *ICCV*, 2017.

[6] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. In *NIPS*, 2016.

[7] M. Cox, S. Sridharan, S. Lucey, and J. Cohn. Least Squares Congealing for Unsupervised Alignment of Images. In *CVPR*, 2008.

[8] M. Cox, S. Sridharan, S. Lucey, and J. Cohn. Least-Squares Congealing for Large Numbers of Images. In *ICCV*, 2009.

[9] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*, 1977.

[10] K. G. Dizaji, A. Herandi, C. Deng, W. Cai, and H. Huang. Deep Clustering via Joint Convolutional Autoencoder Embedding and Relative Entropy Minimization. In *ICCV*, 2017.

[11] B. J. Frey and N. Jojic. Estimating Mixture Models of Images and Inferring Spatial Transformations Using the EM Algorithm. In *CVPR*, 1999.

[12] B. J. Frey and N. Jojic. Fast, large-scale transformation-invariant clustering. In *NIPS*, 2002.

[13] B. J. Frey and N. Jojic. Transformation-Invariant Clustering Using the EM Algorithm. *TPAMI*, 2003.

[14] A. Gepperth and B. Pfülb. Gradient-based training of Gaussian Mixture Models in High-Dimensional Spaces. *arXiv:1912.09379*, 2019.

[15] K. Greff, S. van Steenkiste, and J. Schmidhuber. Neural Expectation Maximization. In *NIPS*, 2017.

[16] R. M. Haralick, S. R. Sternberg, and X. Zhuang. Image Analysis Using Mathematical Morphology. *TPAMI*, 1987.

[17] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2001.

[18] P. Häusser, J. Plapp, V. Golkov, E. Aljalbout, and D. Cremers. Associative Deep Clustering: Training a Classification Network with no Labels. In *GCPR*, 2018.

[19] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016.

[20] G. E. Hinton, A. Krizhevsky, and S. D. Wang. Transforming Auto-Encoders. In *ICANN*, 2011.

[21] R. Hosseini and S. Sra. Matrix Manifold Optimization for Gaussian Mixtures. In *NIPS*, 2015.

[22] W. Hu, T. Miyato, S. Tokui, E. Matsumoto, and M. Sugiyama. Learning Discrete Representations via Information Maximizing Self-Augmented Training. In *ICML*, 2017.

[23] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial Transformer Networks. In *NIPS*, 2015.

[24] X. Ji, A. Vedaldi, and J. Henriques. Invariant Information Clustering for Unsupervised Image Classification and Segmentation. In *ICCV*, 2019.

[25] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou. Variational Deep Embedding: An Unsupervised and Generative Approach to Clustering. In *IJCAI*, 2017.

[26] S. Karayev, M. Trentacoste, H. Han, A. Agarwala, T. Darrell, A. Hertzmann, and H. Winnemoeller. Recognizing Image Style. In *BMVC*, 2014.

[27] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015.

[28] A. Kosiorek, S. Sabour, Y. W. Teh, and G. E. Hinton. Stacked Capsule Autoencoders. In *NeurIPS*, 2019.

[29] H. W. Kuhn and B. Yaw. The Hungarian method for the assignment problem. *Naval Research Logistic Quarterly*, 1955.

[30] E. G. Learned-Miller. Data Driven Image Models through Continuous Joint Alignment. *TPAMI*, 2005.

[31] Y. LeCun, L. Bottou, Y. Bengio, and P. Ha. Gradient-Based Learning Applied to Document Recognition. In *Proc. of the IEEE*, 1998.

[32] Y. J. Lee, C. L. Zitnick, and M. F. Cohen. ShadowDraw: Real-Time User Guidance for Freehand Drawing. In *SIGGRAPH*, 2011.

[33] K. Lenc and A. Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *CVPR*, 2015.

[34] Q. Li, Z. Sun, Z. Lin, R. He, and T. Tan. Transformation invariant subspace clustering. *Pattern Recognition*, 2016.

[35] Z. Li and N. Snavely. MegaDepth: Learning Single-View Depth Prediction from Internet Photos. In *CVPR*, 2018.

[36] X. Liu, Y. Tong, and F. W. Wheeler. Simultaneous Alignment and Clustering for an Image Ensemble. In *ICCV*, 2009.

[37] B. D. Lucas, T. Kanade, et al. An Iterative Image Registration Technique with an Application to Stereo Vision. In *IJCAI*, 1981.

[38] J. MacQueen. Some methods for classification and analysis of multivariate observations. *Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

[39] M. A. Mattar, A. R. Hanson, and E. G. Learned-Miller. Unsupervised Joint Alignment and Clustering using Bayesian Nonparametrics. In *UAI*, 2012.

[40] E. G. Miller, N. E. Matsakis, and P. A. Viola. Learning from One Example Through Shared Densities on Transforms. In *CVPR*, 2000.

[41] S. Mukherjee, H. Asnani, E. Lin, and S. Kannan. ClusterGAN : Latent Space Clustering in Generative Adversarial Networks. In *AAAI*, 2019.

[42] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading Digits in Natural Images with Unsupervised Feature Learning. In *NIPS Workshop*, 2011.

[43] P. J. Phillips, P. J. Flynn, T. Scruggs, K. W. Bowyer, Jin Chang, K. Hoffman, J. Marques, Jaesik Min, and W. Worek. Overview of the Face Recognition Grand Challenge. In *CVPR*, 2005.

[44] U. Shaham, K. Stanton, H. Li, B. Nadler, R. Basri, and Y. Kluger. SpectralNet: Spectral Clustering Using Deep Neural Networks. In *ICLR*, 2018.

[45] K. S. Tai, P. Bailis, and G. Valiant. Equivariant Transformer Networks. In *ICML*, 2019.

[46] R. A. S. Weber, M. Eyal, N. Skafte, O. Shriki, and O. Freifeld. Diffeomorphic Temporal Alignment Nets. In *NeurIPS*, 2019.

[47] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv:1708.07747*, 2017.

[48] J. Xie, R. Girshick, and A. Farhadi. Unsupervised Deep Embedding for Clustering Analysis. In *ICML*, 2016.

[49] J. Yang, D. Parikh, and D. Batra. Joint Unsupervised Learning of Deep Representations and Image Clusters. In *CVPR*, 2016.

[50] X. Yang, C. Deng, F. Zheng, J. Yan, and W. Liu. Deep Spectral Clustering Using Dual Autoencoder Network. In *CVPR*, 2019.

[51] P. Zhou, Y. Hou, and J. Feng. Deep Adversarial Subspace Clustering. In *CVPR*, 2018.

[52] J.-Y. Zhu, Y. J. Lee, and A. A. Efros. AverageExplorer: Interactive Exploration and Alignment of Visual Data Collections. In *SIGGRAPH*, 2014.

# A Dataset descriptions

Table 4: **Datasets and transformation sequences used**

| Dataset | Samples | Classes | Dimension | Transformation sequence |
|---|---|---|---|---|
| *Standard* | | | | |
| MNIST [31] | 70,000 | 10 | 1×28×28 | aff-morpho-tps |
| MNIST-test [31] | 10,000 | 10 | 1×28×28 | aff-morpho-tps |
| USPS [17] | 9,298 | 10 | 1×16×16 | col-aff-tps |
| Fashion-MNIST [47] | 70,000 | 10 | 1×28×28 | col-aff-tps |
| FRGC [43] | 2,462 | 20 | 3×32×32 | col-aff-tps |
| SVHN [42] | 99,289 + unlabeled extra | 10 | 3×28×28 | col-proj |
| *Augmented* | | | | |
| MNIST-1k | 1,000 | 10 | 1×28×28 | aff-morpho-tps |
| MNIST-color | 70,000 | 10 | 3×28×28 | col-aff-tps |
| affNIST-test | 320,000 | 10 | 1×40×40 | aff-morpho-tps |
| *Real photographs* | | | | |
| All | 1k to 15k | - | 3×128×128 | col-proj |

Table 4 summarizes dataset characteristics as well as the transformation sequences used. Datasets are:

- **MNIST** and **MNIST-test** [31] which respectively correspond to full and test subset of MNIST dataset. They depict binary white handwritten digits centered over a black background.
- **USPS** [17] is a handwritten digit dataset from USPS composed of greyscale images.
- **Fashion-MNIST** [47] is a 10-class clothing dataset composed of greyscale images of cloth over black background. Classes are: T-shirt, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, ankle boot.
- **FRGC** [43] is a colored face dataset. We use a subset of this dataset introduced in [49], where 20 subjects are selected and each image is cropped and resized to a constant size of 32×32.
- **SVHN** [42] is composed of digits extracted from house numbers cropped from Google Street View images. Following standard practice for clustering, we use both labeled samples (99,289) and unlabeled extra samples (~530k) for training and evaluate on the labeled subset only.
- **affNIST-test** is the test split of affNIST (https://www.cs.toronto.edu/ tijmen/affNIST/) an augmented dataset of MNIST where random affine transformations are applied.
- **MNIST-1k**: we randomly sampled 1,000 images from the test split of MNIST.
- **MNIST-color**: we augmented MNIST with random colors for background and foreground.

# B Transformation invariance

We consider $N$ image samples $x_{1:N}$, $K$ prototypes $c_{1:K}$ and a group of parametric transformations $\{\mathcal{T}_\beta,\, \beta \in B\}$. For $\beta_1, \beta_2 \in B$, we write $\beta_1\beta_2 \in B$ the element such that $\mathcal{T}_{\beta_1\beta_2} = \mathcal{T}_{\beta_1} \circ \mathcal{T}_{\beta_2}$. We have, for any $\alpha_1, \ldots, \alpha_K \in B$:

$$\mathcal{L}_{\text{TI}}(\{c_1, \ldots, c_K\}) = \mathcal{L}_{\text{TI}}(\{\mathcal{T}_{\alpha_1}(c_1), \ldots, \mathcal{T}_{\alpha_K}(c_K)\}).$$

Indeed:

$$\begin{aligned}
\mathcal{L}_{\text{TI}}(\{\mathcal{T}_{\alpha_1}(c_1), \ldots, \mathcal{T}_{\alpha_K}(c_K)\}) &= \sum_{i=1}^{N} \min_{\{\beta_1,\ldots,\beta_K\}\in B^K} l(x_i,\, \{\mathcal{T}_{\beta_1} \circ \mathcal{T}_{\alpha_1}(c_1), \ldots, \mathcal{T}_{\beta_K} \circ \mathcal{T}_{\alpha_K}(c_K)\}) \\
&= \sum_{i=1}^{N} \min_{\{\beta_1,\ldots,\beta_K\}\in B^K} l(x_i,\, \{\mathcal{T}_{\beta_1\alpha_1}(c_1), \ldots, \mathcal{T}_{\beta_K\alpha_K}(c_K)\}) \\
&= \sum_{i=1}^{N} \min_{\{\beta'_1,\ldots,\beta'_K\}\in B^K} l(x_i,\, \{\mathcal{T}_{\beta'_1}(c_1), \ldots, \mathcal{T}_{\beta'_K}(c_K)\}) \\
&= \mathcal{L}_{\text{TI}}(\{c_1, \ldots, c_K\}),
\end{aligned}$$

using the variable change $\beta'_k = \beta_k\alpha_k$, which is possible because for any $\alpha \in B$, $\alpha B = B$ as we assumed to have a group of transformations.

In some specific cases, the loss is also invariant to the samples, in particular when the loss $l$ is invariant to joint transformation of the prototype and the samples, i.e. for any $\beta \in B$, $l(x_i, \{c_1, \dots, c_K\} = l(\mathcal{T}_\beta(x_i), \{\mathcal{T}_\beta(c_1), \dots, \mathcal{T}_\beta(c_K)\}$. This is the case for example for K-means with a group of isometric transformations (e.g. rigid transformations), and it is also the case for GMM with the group of affine transformations applied to both the mean and covariance mixture parameters.

Note that we also tried to transform the samples to match the prototypes, which would lead to an invariance to sample transformation. However, a trivial solution to corresponding optimization problem is to learn "empty" prototypes and transformations of the samples into empty images. For examples, for the MNIST case with affine transformations, we observed that completely black prototypes were learned and any digit was transformed into a black image. Although a regularization term could have prevented such behaviour, we argue that keeping raw samples as target and transforming the prototypes is simpler and effective.

# C Analysis

## C.1 Statistics on standard clustering benchmarks

Table 5: **Detailed results.** We report statistics of our results on standard clustering benchmarks. For SVHN, we also report results with our Gaussian weighted loss ($\star$).

| | | | MNIST | | MNIST-test | | USPS | | F-MNIST | | FRGC | | SVHN |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | Runs | Stat | ACC | NMI | ACC | NMI | ACC | NMI | ACC | NMI | ACC | NMI | ACC |
| **DTI K-means** | 10 | avg | 97.3 | 94.0 | 96.6 | 94.6 | 86.4 | 88.2 | 61.2 | 63.7 | 39.6 | 48.7 | 36.4 / 44.5$^\star$ |
| | 10 | std | 0.1 | 0.1 | 4.1 | 1.5 | 4.1 | 1.6 | 2.0 | 0.3 | 1.7 | 2.2 | 1.9 / 9.6$^\star$ |
| | 10 | min | 97.1 | 93.8 | 84.9 | 90.4 | 83.2 | 87.1 | 57.4 | 63.2 | 35.9 | 43.9 | 34.5 / 37.0$^\star$ |
| | 10 | median | 97.3 | 94.0 | 97.9 | 95.1 | 85.0 | 87.4 | 61.9 | 63.3 | 40.2 | 49.3 | 35.8 / 39.6$^\star$ |
| | 10 | max | 97.5 | 94.2 | 98.0 | 95.3 | 96.4 | 92.0 | 63.3 | 64.2 | 41.1 | 51.4 | 39.6 / 62.6$^\star$ |
| | 10 | minLoss | 97.2 | 93.8 | 98.0 | 95.3 | 89.8 | 89.5 | 57.4 | 64.1 | 41.1 | 49.7 | 39.6 / 62.6$^\star$ |
| **DTI GMM** | 10 | avg | 95.9 | 93.2 | 97.8 | 94.7 | 84.5 | 87.2 | 59.6 | 62.2 | 40.1 | 48.9 | 36.7 / 57.4$^\star$ |
| | 10 | std | 3.9 | 1.5 | 0.1 | 0.2 | 2.0 | 0.8 | 4.7 | 2.4 | 1.4 | 1.5 | 2.3 / 5.1$^\star$ |
| | 10 | min | 84.7 | 89.1 | 97.7 | 94.4 | 82.0 | 86.3 | 56.1 | 59.7 | 38.4 | 46.8 | 34.0 / 49.9$^\star$ |
| | 10 | median | 97.1 | 93.7 | 97.8 | 94.7 | 84.3 | 87.1 | 57.2 | 60.9 | 39.6 | 49.1 | 36.4 / 57.4$^\star$ |
| | 10 | max | 97.3 | 93.9 | 98.0 | 95.1 | 87.3 | 89.0 | 68.2 | 66.3 | 41.9 | 51.1 | 39.5 / 64.6$^\star$ |
| | 10 | minLoss | 97.1 | 93.7 | 98.0 | 95.1 | 87.3 | 89.0 | 68.2 | 66.3 | 41.6 | 51.1 | 39.5 / 63.3$^\star$ |

## C.2 Accuracy and loss correlation

Similar to standard K-means and GMM, there is a variation in performances depending on the random initialization. We experimentally found that: (i) runs seem to be mainly grouped into distinct modes, each corresponding to roughly the same clustering quality; (ii) a run with a low loss usually leads to high clustering performances. We launched 100 runs on MNIST-test dataset and plot the loss with respect to the accuracy for each run in Figure 4. Except 2 outliers for the 100 runs, the runs with lower loss correspond to the runs with better performances. This is verified in most of our experiments, where the minLoss criterion clearly improves over the average performance.

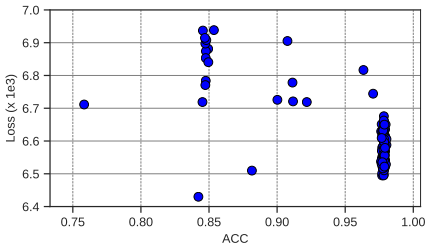
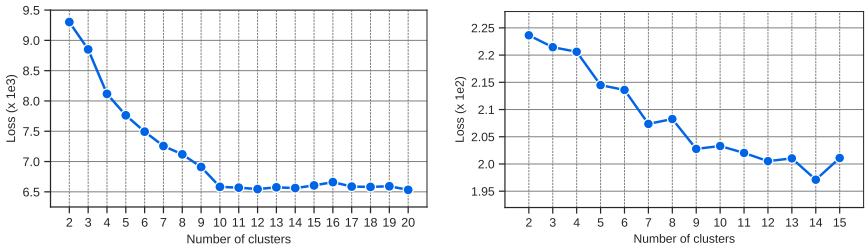


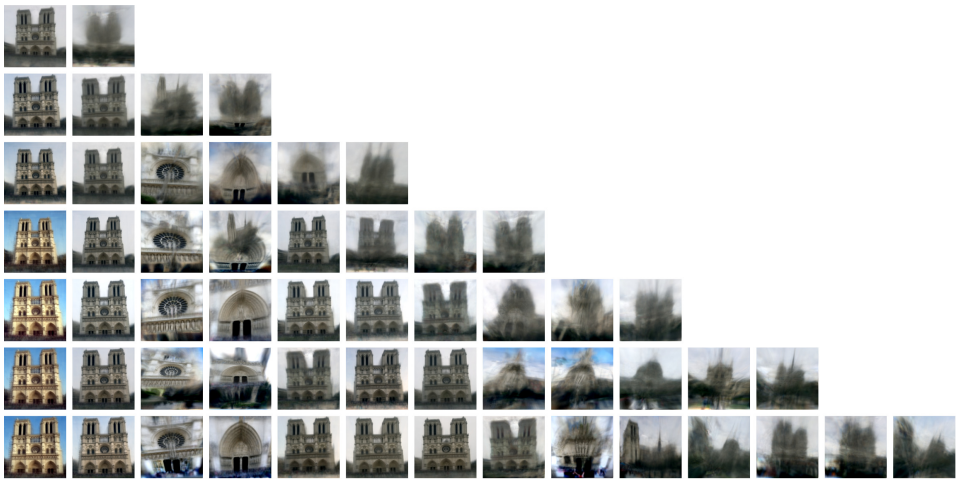

Figure 4: **Accuracy/loss correlation.** We report loss and accuracy for DTI K-means on MNIST-test.

## C.3 Effect of the number of clusters K

Similar to many clustering methods, the selection of the number of clusters is a challenge. We investigated if a purely quantitative analysis could be applied to select $K$. In Figure 5a, we plot the loss of DTI-Kmeans as a function of the number of clusters for MNIST-test (left) and Notre-Dame (right). For MNIST-test, it is clear an elbow method could be applied to select the good number of clusters. For Notre-Dame, the quantitative analysis is not as conclusive but in this case, the correct number of clusters is not clearly defined. In practice, we did not find the qualitative results on internet photo collections to be very sensitive to this choice, as shown in Figure 5b where learned prototypes are mostly consistent across the different clustering results.




(a) Loss w.r.t varying numbers of clusters for MNIST-test (left) and Notre-Dame (right)



(b) Prototypes learned on Notre-Dame for different numbers of clusters

Figure 5: **Effect of $K$. (a)** We report the loss of DTI K-means for different numbers of clusters. For MNIST-test (left), the loss is averaged over 5 runs and for Notre-Dame (right), the loss corresponds to a single run. **(b)** We show the prototypes learned on Notre-Dame for even numbers of clusters.

### C.4 Constraining color transformation

While evaluating our approaches on real photograph collections, we experimentally observed that a full affine color transformation module (12 parameters) was too flexible and as a result, prototypes were able to learn different patterns hidden in each color channel. In Figure 6, we show each R, G and B channel as a greyscale image for two prototypes learned using a full affine color transformation module. One can see that a second pattern is hidden in particular in the green channels. To avoid this effect, we restricted the color transformation module to be a diagonal affine transformation corresponding to 6 parameters in total.

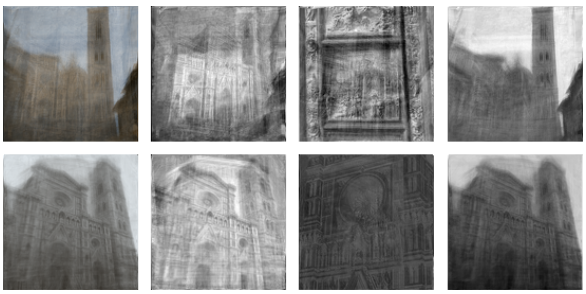


Figure 6: **Learned prototypes and RGB decomposition.** Two examples of learned prototypes (first column) on Florence cathedral collection from [35] using a full color transformation module (12 parameters). The 3 right columns correspond to R, G and B channels rescaled between 0 and 1. Note how the green channel is used to hide a completely different pattern from the other 2 channels.