Inference and Learning for Generative Capsule Models

Final m/s version of paper accepted for publication in *Neural Computation*.

Inference and Learning for Generative Capsule Models

Alfredo Nazabal*†
Amazon Development Centre Scotland
Edinburgh, UK
alfrena@amazon.com

Nikolaos Tsagkas*‡
School of Informatics, University of Edinburgh,
Edinburgh EH8 9AB, UK
n.tsagkas@ed.ac.uk

Christopher K. I. Williams
School of Informatics, University of Edinburgh,
Edinburgh EH8 9AB, UK, and
The Alan Turing Institute, London, UK
c.k.i.williams@ed.ac.uk

October 21, 2022

Abstract

Capsule networks (see e.g. Hinton et al., 2018) aim to encode knowledge of and reason about the relationship between an object and its parts. In this paper we specify a *generative* model for such data, and derive a variational algorithm for inferring the transformation of each model object in a scene, and the assignments of observed parts to the objects. We derive a learning algorithm for the object models, based on variational expectation maximization (Jordan et al., 1999). We also study an alternative inference algorithm based on the RANSAC method of Fischler and Bolles (1981). We apply these inference methods to (i) data generated from multiple geometric objects like squares and triangles ("constellations"), and (ii) data from a parts-based model of faces. Recent work by Kosiorek et al. (2019) has used amortized inference via stacked capsule autoencoders (SCAEs) to tackle this problem—our results show that we significantly outperform them where we can make comparisons (on the constellations data).

*Equal contribution.
†Work carried out while AN was at the Alan Turing Institute.
‡Part of this work was carried out when NT was a MSc student at the University of Edinburgh.

Keywords: Capsules, variational inference, permutation matrix, Sinkhorn-Knopp algorithm, RANSAC.

1 Introduction

An attractive way to set up the problem of object recognition is *hierarchically*, where an object is described in terms of its parts, and these parts are in turn composed of sub-parts, and so on. For example a face can be described in terms of the eyes, nose, mouth, hair, etc.; and a teapot can be described in terms of a body, handle, spout and lid parts. This approach has a long history in computer vision, see e.g. the work on Pictoral Structures by Fischler and Elschlager (1973), and Recognition-by-Components by Biederman (1987). More recently Felzenszwalb et al. (2009) used discriminatively-trained parts-based models to obtain state-of-the-art results (at the time) for object recognition. Advantages of recognizing objects by first recognizing their constituent parts include tolerance to the occlusion of some parts, and that parts may vary less under a change of pose than the appearance of the whole object.

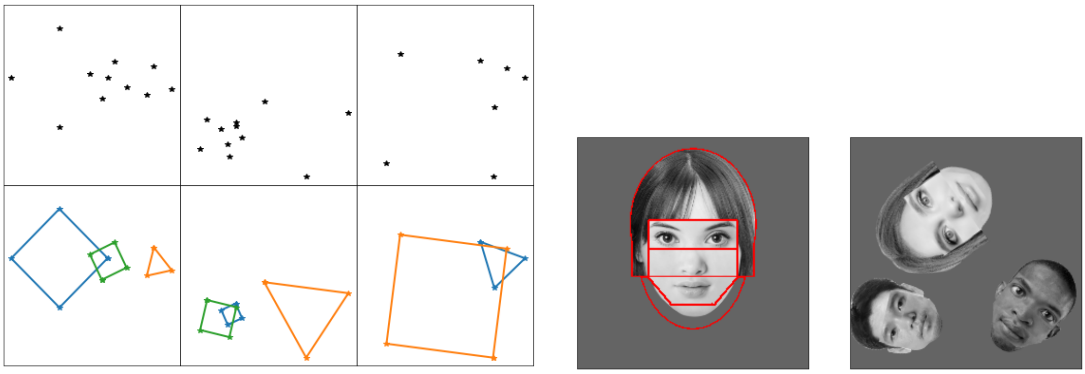
Recent work by Sabour et al. (2017) and Hinton et al. (2018) has developed *capsule networks*. These exploit the fact that if the pose[1] of the object changes, this can have very complicated effects on the pixel intensities in an image, but the geometric transformation between the object and the parts can be described by a simple linear transformation (as used in computer graphics). In these papers a part in a lower level can vote for the pose of an object in the higher level, and an object's presence is established by the agreement between votes for its pose in a process called "routing-by-agreement". Hinton et al. (2018, p. 1) describe this as a process which "updates the probability with which a part is assigned to a whole based on the proximity of the vote coming from that part to the votes coming from other parts that are assigned to that whole". Subsequently Kosiorek et al. (2019) framed inference for a capsule network in terms of an autoencoder, the Stacked Capsule Autoencoder (SCAE). Here, instead of the iterative routing-by-agreement algorithm, a neural network h^{caps} takes as input the set of input parts and outputs predictions for the object capsules' instantiation parameters $\{\mathbf{y}_k\}_{k=1}^K$. Further networks h_k^{part} are then used to predict part candidates from each $\mathbf{y}_k$.

The objective function used in Hinton et al. (2018) (their eq. 4) is quite complex (involving four separate terms), and is not derived from first principles. In this paper we argue that the description in the paragraph above is backwards—we find it more natural to first describe the generative process *by which an object gives rise to its parts*, and that the appropriate routing-by-agreement inference algorithm then falls out naturally from this principled formulation.

The contributions of this paper are to:

- Derive a novel variational inference algorithm for routing-by-agreement, based on a generative model of object-part relationships, including a relaxation of the permutation-matrix formulation for matching object parts to observations;

[1] i.e. the location and rotation of the object in 2D or 3D.



(a) (b) (c)

Figure 1: (a) Scenes composed of 2D points (upper figures) and their corresponding objects (lower figures). (b) A synthetic face. The red lines indicate the areas of the 5 part types (i.e. hair, eyes, nose, mouth and jaw). (c) Example scene with 3 randomly transformed faces.

- Focus on the problem of inference for *scenes containing multiple objects*. Much of the work on capsules considers only single objects (although sec. 6 in Sabour et al. (2017) and De Sousa Ribeiro et al. (2020b, sec. 4.4) are notable exceptions).
- Demonstrate the effectiveness of our algorithm on (i) "constellations" data generated from multiple geometric objects (e.g. triangles, squares) at arbitrary translations, rotations and scales; and (ii) data of multiple faces from a novel parts-based model of faces;
- Evaluate the performance of our algorithm and the RANSAC method vs. competitors on the constellations data.
- Derive a learning algorithm for the object models, based on variational expectation maximization (Jordan et al., 1999).

The structure of the remainder of the paper is as follows: in section 2 we provide an overview of the method. Section 3 gives details of the generative model and the matching distribution between observed and model parts. The variational inference algorithm derived from this model is given in section 4, and RANSAC inference is described in sec. 5. Related work is discussed in section 6, and approaches to learning generative capsule models are given in section 7. Our experiments and results are described in section 8. We conclude with a discussion.

2 Overview

Consider images of a set of objects in different poses, such as images of handwritten digits, faces, or geometric shapes in 2D or 3D. An object can be defined as an instantiation of a specific model object (or template) along with a particular pose (or geometric

transformation). Furthermore, objects, and thus templates, are decomposed in parts, which are the basic elements that comprise the objects. For example, faces can be decomposed into parts (e.g. mouth, nose etc.), or a geometric shape can be decomposed into vertices. These parts can have internal variability, (e.g. eyes open or shut).

More formally, let $T = \{T_k\}_{k=1}^{K}$ be the set of K templates that are used to generate a scene. Each template $T_k = \{\mathbf{p}_n\}_{n=1}^{N_k}$ is composed of N_k parts $\mathbf{p}_n$. We assume that scenes can only be generated using the available templates. Furthermore, every scene can present a different configuration of objects, with some objects missing in some scenes. For example, in scenes that could potentially contain all digits from 0 to 9 once, and if only the digits 2 and 5 are in the image, we consider that the other digits are missing. If all the templates were employed in the scene, then the number of observed parts M is equal to the sum of all the parts of all the templates $N = \sum_{k=1}^{K} N_k$.

Each observed template T_k in a scene is then transformed by an independent transformation $\mathbf{y}_k$, different for each template, generating a set of transformed parts $X_k = \{\mathbf{x}_n\}_{n=1}^{N_k}$

$$T_k \xrightarrow{\mathbf{y}_k} X_k. \tag{1}$$

The transformation $\mathbf{y}_k$ includes both the geometric transformation of the template, and also the appearance variability in the parts.

In this paper we assume that we are given a scene $X = \{\mathbf{x}_m\}_{m=1}^{M}$ composed of M observed parts coming from multiple templates. (For example, the Part Capsule Autoencoder (PCAE) of Kosiorek et al. (2019) outputs a set of parts.) The *inference problem* for X involves a number of different tasks. We need to determine which objects from the set of templates were employed to generate the scene. Also, we need to infer what transformation $\mathbf{y}_k$ was applied to each template to generate the objects. This allows us to infer the *correspondences* between the template parts and the scene parts.

We demonstrate our method on "constellations" data as shown in Fig. 1(a), and data containing multiple faces (Fig. 1(c)). In the constellations data, the real generators are triangle and square objects, but only their vertices are provided in the data. The faces data is generated from the parts-based model of faces shown in Fig. 1(b) and described in sec. 8.2.1.

3 A Generative Capsules Model (GCM)

We propose a generative model to describe the problem. Consider a template (or model) T_k for the kth object. T_k is composed of N_k parts $\{\mathbf{p}_n\}_{n=1}^{N_k}$.[2] Each part $\mathbf{p}_n$ is described in its reference frame by its *geometry* $\mathbf{p}_n^g$ and its *appearance* $\mathbf{p}_n^a$. Each object also has associated latent variables $\mathbf{y}_k$ which transform from the reference frame to the image frame, so $\mathbf{y}_k$ is split into geometric variables $\mathbf{y}_k^g$ and appearance variables $\mathbf{y}_k^a$.

Geometric transformations: Here we consider 2D templates and a similarity transformation (translation, rotation and scaling) for each object, but this can be readily

[2]For simplicity of notation we suppress the dependence of $\mathbf{p}_n$ on k for now.

extended to allow 3D templates and a scene-to-viewer camera transformation. We assume that $\mathbf{p}_n^g$ contains the x and y locations of the part, and also its size s_n and orientation ϕ_n relative to the reference frame.[3] The size and orientation are represented as the projected size of the part onto the x and y axes, as this allows us to use linear algebra to express the transformations (see below). Thus $\mathbf{p}_n^g = (p_{nx}^g, p_{ny}^g, s_n \cos\phi_n, s_n \sin\phi_n)^T$.

Consider a template with parts $\mathbf{p}_n^g$ for $n = 1, \ldots, N_k$ that we wish to scale by a factor s, rotate through with a clockwise rotation angle θ and translate by (t_x, t_y). We obtain a transformed object with geometric observations for the nth part $\mathbf{x}_n^g = (x_{nx}^g, x_{ny}^g, x_{nc}^g, x_{ns}^g)$, where the c and s subscripts denote the projections of the scaled and rotated part onto the x and y axes respectively (c and s are mnemonic for cosine and sine).

For each part in the template, the geometric transformation works as follows:

$$\begin{pmatrix} x_{nx} \\ x_{ny} \\ x_{nc} \\ x_{ns} \end{pmatrix} = \begin{pmatrix} 1 & 0 & p_{nx} & p_{ny} \\ 0 & 1 & p_{ny} & -p_{nx} \\ 0 & 0 & s_n \cos\phi_n & -s_n \sin\phi_n \\ 0 & 0 & s_n \sin\phi_n & s_n \cos\phi_n \end{pmatrix} \begin{pmatrix} t_x \\ t_y \\ s\cos\theta \\ s\sin\theta \end{pmatrix}. \tag{2}$$

Decoding the third equation, we see that $x_{nc} = s_n s \cos\phi_n \cos\theta - s_n s \sin\phi_n \sin\theta = s_n s \cos(\phi_n + \theta)$ using standard trigonometric identities. The x_{ns} equation is derived similarly. We shorten eq. 2 to $\mathbf{x}_n^g = F_{kn}^g \mathbf{y}_k^g$, where $\mathbf{y}_k^g$ is the $\mathbb{R}^4$ column vector, and $F_{kn}^g \in \mathbb{R}^{4\times 4}$ is the matrix to its left.[4] Allowing Gaussian observation noise with precision λ we obtain

$$p(\mathbf{x}_n^g | T_k, \mathbf{y}_k^g) \sim N\left(\mathbf{x}_n^g | F_{kn}^g \mathbf{y}_k^g, \lambda^{-1} I\right). \tag{3}$$

The prior distribution over similarity transformations $\mathbf{y}_k^g$ is modelled with a $\mathbb{R}^4$ Gaussian distribution with mean $\boldsymbol{\mu}_0^g$ and covariance matrix D_0^g:

$$p(Y^g) = \prod_{k=1}^{K} N(\mathbf{y}_k^g | \boldsymbol{\mu}_0^g, D_0^g), \tag{4}$$

where Y^g denotes the set $\{\mathbf{y}_k^g\}_{k=1}^K$. Notice that modelling $\mathbf{y}_k^g$ with a Gaussian distribution implies that we are modelling the translation (t_x, t_y) in $\mathbb{R}^2$ with a Gaussian distribution. If $\boldsymbol{\mu}_0^g = \mathbf{0}$ and $D_0^g = I_4$ then $s^2 = (y_{k3}^g)^2 + (y_{k4}^g)^2$ has a χ_2^2 distribution, and $\theta = \arctan y_{k4}^g / y_{k3}^g$ is uniformly distributed in its range $[-\pi, \pi]$ by symmetry. For more complex linear transformations (e.g. an affine transformation), we need only to increase the dimension of $\mathbf{y}_k^g$ and change the form of F_{kn}^g, but the generative model in (4) would remain the same. For the 3D case, note that Basri (1996) has shown that every affine projection of an object represents some uncalibrated paraperspective projection of the object.

[3] For the constellations data, the size and orientation information is not present, nor are there any appearance features.

[4] Here F is mnemonic for "features".

Appearance transformations: The appearance $\mathbf{x}_n^a$ of part n in the image depends on $\mathbf{y}_k^a$. For our faces data, $\mathbf{y}_k^a$ is a vector latent variable which models the co-variation of the appearance of the parts via a linear (factor analysis) model; see sec. 8.2.1 for a fuller description. Hence

$$p(\mathbf{x}_n^a|T_k, \mathbf{y}_k) \sim N\left(\mathbf{x}_n^a|F_{kn}^a\mathbf{y}_k^a + \boldsymbol{m}_{kn}^a, D_{kn}^a\right), \tag{5}$$

where F_{kn}^a maps from $\mathbf{y}_k^a$ to the predicted appearance features in the image, D_{kn}^a is a diagonal matrix of variances and $\boldsymbol{m}_{kn}^a$ allows for the appearance features to have a non-zero mean. The dimensionality of the nth part of the kth template is d_{kn}. The prior for $\mathbf{y}_k^a$ is taken to be a standard Gaussian, i.e. $N(\mathbf{0}, I)$. Combining (4) and the prior for $\mathbf{y}_k^a$, we have that $p(\mathbf{y}_k) = N(\boldsymbol{\mu}_0, D_0)$, where $\boldsymbol{\mu}_0$ stacks $\boldsymbol{\mu}_0^g$ and $\mathbf{0}$ from the appearance, and D_0 is a diagonal matrix with blocks D_0^g and I.

Joint distribution: Let $z_{mnk} \in \{0, 1\}$ indicate whether observed part $\mathbf{x}_m$ matches to part n of object k. The set of these match variables is denoted by Z. Every observation m belongs uniquely to a tuple (k, n), or in other words, a point $\mathbf{x}_m$ belongs uniquely to the part defined by $\mathbf{y}_k$ acting on the template matrix F_{kn}. The opposite is also partially true; every tuple (k, n) belongs uniquely to a point m, or it is unassigned if part n of template k is missing in the scene.

The joint distribution of the variables in the model is given by

$$p(X, Y, Z) = p(X|Y, Z)p(Y)p(Z), \tag{6}$$

where $p(X|Y, Z)$ is a Gaussian mixture model explaining how the points in a scene were generated from the templates

$$p(X|Y, Z) = \prod_{m=1}^{M}\prod_{k=1}^{K}\prod_{n=1}^{N_k} N\left(\mathbf{x}_m|F_{kn}\mathbf{y}_k + \boldsymbol{m}_{kn}, D_{kn}\right)^{z_{mnk}}, \tag{7}$$

where D_{kn} consists of the diagonal matrices $\lambda^{-1}I$ and D_{kn}^a and $\boldsymbol{m}_{kn}$ consists of a zero vector for the geometric features stacked on top of the mean for the appearance features $\boldsymbol{m}_{kn}^a$. Note that F_{kn} has blocks of zeros so that $\mathbf{x}_m^g$ does not depend on $\mathbf{y}_k^a$, and $\mathbf{x}_m^a$ does not depend on $\mathbf{y}_k^g$.

Annealing parameter: During inference, where the model is fitted to data, it is useful to modify the covariance matrix D_{kn} to $\beta^{-1}D_{kn}$, where β is a parameter < 1. The effect of this is to inflate the variances in D_{kn}, allowing greater uncertainty in the inferred matches early on in the fitting process, as used, e.g. in Revow et al. (1996). β is increased according to an annealing schedule during the fitting.

Match distribution $p(Z)$: In a standard Gaussian mixture model, the assignment matrix Z is characterized by a Categorical distribution, where each point $\mathbf{x}_m$ is assigned to one part

$$p(Z) = \prod_{m=1}^{M} \text{Cat}(\mathbf{z}_m|\boldsymbol{\pi}), \tag{8}$$

with $\mathbf{z}_m$ being a 0/1 vector with only one 1, and $\boldsymbol{\pi}$ being the probability vector for each tuple (k, n). However, the optimal solution to our problem occurs when each part of a template belongs uniquely to one observed part in a scene. This means that Z should be a permutation matrix, where each point m is assigned to a tuple (k, n) and vice versa. Notice that a permutation matrix is a square matrix, so if $M \leq N$, we add dummy rows to Z, which are assigned to missing points in the scene.

The set of permutation matrices of dimension N is a discrete set containing $N!$ permutation matrices. A discrete prior over permutation matrices assigns each valid matrix Z_i a probability π_i:

$$p_{perm}(Z) = \sum_{i=1}^{N!} \pi_i \; I[Z = Z_i] \tag{9}$$

with $\sum_{i=1}^{N!} \pi_i = 1$ and $I[Z = Z_i]$ being the indicator function, equal to 1 if $Z = Z_i$ and 0 otherwise.

The number of possible permutation matrices increases as $N!$, which makes exact inference over permutations intractable, except for very small N. An interesting property of $p_{perm}(Z)$ is that its first moment $\mathbb{E}_{p_{perm}}[Z]$ is a doubly-stochastic (DS) matrix, a matrix of elements in $[0, 1]$ whose rows and columns sum to 1. We propose to relax $p_{perm}(Z)$ to a distribution $p_{DS}(Z)$ that is characterized by the doubly-stochastic matrix A with elements a_{mnk}, such that $\mathbb{E}_{p_{DS}}[Z] = A$:

$$p_{DS}(Z) = \prod_{m=1}^{N} \prod_{k=1}^{K} \prod_{n=1}^{N_k} a_{mnk}^{z_{mnk}}. \tag{10}$$

A is fully characterized by $(N-1)^2$ elements. In the absence of any prior knowledge of the affinities, a uniform prior over Z with $a_{mnk} = \frac{1}{N}$ can be used. However, note that p_{DS} can also represent a particular permutation matrix Z_i by setting the appropriate entries of A to 0 or 1, and indeed we would expect this to occur during variational inference (see sec. 4) when the model converges to a correct solution.

Related models: Our model for a single object has both geometric and appearance variability (see eqs. 3 and 5). A similar model but with geometric features only was developed by Revow et al. (1996). Fergus et al. (2003) described a "constellation of parts" model, that used a joint Gaussian model for locations of the parts, and an image-patch model for each part appearance. However, the appearance model was a single Gaussian per part, without the correlations between parts afforded by the factor analysis model. This model was applied to images of single (foreground) objects, summing out over possible assignments Z. Rao and Ballard (1999) developed a hierarchical factor analysis model, but used it to model extended edges in natural image patches rather than correlations between the parts of an object. See sec. 7 for further discussion of parts-based models.

Hierarchical modelling: above we have described a two-layer model with part-object relations. This can of course be extended to a deeper hierarchy; for example

we would expect to find relationships between the objects in a scene, such as the relationships between a dining table and the dining chairs, or between a bed and its associated nightstand(s). Such scene-level relationships can be formulated in terms of graphical models for the groups of objects. We believe the most difficult aspect is handling the assignment Z between model parts and observed parts (as covered in this paper), and that inference in the graphical model above is fairly standard, and is left for future work.

4 Variational Inference

Variational inference for the above model can be derived similarly to the Gaussian Mixture model case (Bishop, 2006, Sec. 10.1). The variational distribution under the mean field assumption is given by $q(Z, Y) = q(Z)q(Y)$, The evidence lower error bound (ELBO) for this model is derived in Appendix A. Optimizing the ELBO with respect to either $q(Z)$ or $q(Y)$, the optimal solutions can be expressed as

$$\log q(Z) \propto \mathbb{E}_{q(Y)}[\log p(X, Y, Z)], \tag{11}$$
$$\log q(Y) \propto \mathbb{E}_{q(Z)}[\log p(X, Y, Z)]. \tag{12}$$

These alternating updates of $q(Y)$ and $q(Z)$ carry out routing-by-agreement.

For $q(Z)$ we obtain an expression with the same form as the prior in (10)

$$q(Z) \propto \prod_{m=1}^{N} \prod_{k=1}^{K} \prod_{n=1}^{N_k} \rho_{mnk}^{z_{mnk}}, \tag{13}$$

where ρ_{mnk} represents the unnormalized probability of point m being assigned to tuple (k, n) and vice versa. These unnormalized probabilities have a different form depending on whether we are considering a point that appears in the scene ($m \leq M$)

$$\log \rho_{mnk} = \log a_{mnk} - \frac{1}{2} \log |\beta^{-1} D_{kn}| - \frac{d_{kn}}{2} \log 2\pi - \\ \frac{\beta}{2} \mathbb{E}_{\mathbf{y}_k}[(\mathbf{x}_m - F_{kn}\mathbf{y}_k - \boldsymbol{m}_{kn})^T D_{kn}^{-1} (\mathbf{x}_m - F_{kn}\mathbf{y}_k - \boldsymbol{m}_{kn})], \tag{14}$$

or whether we are considering a dummy row of the prior ($m > M$),

$$\log \rho_{mnk} = \log a_{mnk}. \tag{15}$$

When a point is part of the scene (14), and thus $m \leq M$ the update of ρ_{mnk} is similar to the Gaussian mixture model case. However, if a point is not part of the scene (15), and thus $m > M$ then the matrix is not updated and the returned value is the prior a_{mnk}. The expectation term in (14) is given by:

$$\mathbb{E}_{\mathbf{y}_k}[(\mathbf{x}_m - F_{kn}\mathbf{y}_k - \boldsymbol{m}_{kn})^T D_{kn}^{-1} (\mathbf{x}_m - F_{kn}\mathbf{y}_k - \boldsymbol{m}_{kn})] = \\ (\mathbf{x}_m - F_{kn}\boldsymbol{\mu}_k - \boldsymbol{m}_{kn})^T D_{kn}^{-1} (\mathbf{x}_m - F_{kn}\boldsymbol{\mu}_k - \boldsymbol{m}_{kn}) + \text{trace}(F_{kn}^T D_{kn}^{-1} F_{kn} \Lambda_k^{-1}). \tag{16}$$

The normalized distribution $q(Z)$ becomes:

$$q(Z) = \prod_{m=1}^{N} \prod_{k=1}^{K} \prod_{n=1}^{N_k} r_{mnk}^{z_{mnk}}, \tag{17}$$

where $\mathbb{E}_{q(Z)}[z_{mnk}] = r_{mnk}$. The elements r_{mnk} of matrix R represent the posterior probability of each point m being uniquely assigned to the part-object tuple (n, k) and vice-versa. This means that R needs to be a DS matrix. This can be achieved by employing the Sinkhorn-Knopp algorithm (Sinkhorn and Knopp, 1967) as given in Algorithm 1, which updates a square non-negative matrix by normalizing the rows and columns alternately until the resulting matrix becomes doubly stochastic.

The use of the Sinkhorn-Knopp algorithm for approximating matching problems has also been described by Powell and Smith (2019) and Mena et al. (2020), but note that in our case we also need to alternate with inference for $q(Y)$.

Algorithm 1 Sinkhorn-Knopp algorithm, taking as input a square non-negative matrix M

procedure SINKHORNKNOPP(M)
  while M not doubly stochastic **do**
    Normalize rows of M: $m_{ij} = \frac{m_{ij}}{\sum_j m_{ij}}, \forall i$
    Normalize columns of M: $m_{ij} = \frac{m_{ij}}{\sum_i m_{ij}}, \forall j$
  return M

Furthermore, the optimal solution to the assignment problem occurs when R is a permutation matrix itself. When this happens we exactly recover a discrete posterior (with the same form as (9)) over permutation matrices where one of them has probability one, with the others being zero.

The distribution for $q(Y)$ is a Gaussian with

$$q(Y) = \prod_{k=1}^{K} N(\mathbf{y}_k | \boldsymbol{\mu}_k, \Lambda_k^{-1}), \tag{18}$$

$$\Lambda_k = D_0^{-1} + \beta \sum_{m}^{M} \sum_{n}^{N_k} r_{mnk} F_{kn}^T D_{kn}^{-1} F_{kn}, \tag{19}$$

$$\boldsymbol{\mu}_k = \Lambda_k^{-1} \left[D_0 \boldsymbol{\mu}_0 + \beta \sum_{m}^{M} \sum_{n}^{N_k} r_{mnk} F_{kn}^T D_{kn}^{-1} (\mathbf{x}_m - \boldsymbol{m}_{kn}) \right], \tag{20}$$

where the updates for both Λ_k and $\boldsymbol{\mu}_k$ depend explicitly on the annealing parameter β and the templates employed in the model. Note that the prediction from datapoint m to the mean of $\mathbf{y}_k$ depends on $r_{mnk} F_{kn}^T D_{kn}^{-1} (\mathbf{x}_m - \boldsymbol{m}_{kn})$, i.e. a weighted sum of the predictions of each part n with weights r_{mnk}. These expressions remain the same when considering a Gaussian mixture prior such as (8).

Algorithm 2 summarizes the inference procedure for this model.

Algorithm 2 Variational Inference

Initialize β, β_{max} and $R \sim U[0,1]^{N \times N}$, $\forall m, n, k$
$R = \text{SinkhornKnopp}(R)$
while not converged **do**
    Update Λ_k (19), $\forall k$
    Update $\boldsymbol{\mu}_k$ (20), $\forall k$
    Update $\log \rho_{mnk}$ (14)(15), $\forall m, n, k$
    Update $R = \text{SinkhornKnopp}(\rho)$
    if ELBO has converged **then**
        if $\beta < \beta_{max}$ **then**
            Anneal β
        else
            converged = True
return $R, \{\boldsymbol{\mu}_k, \Lambda_k\}$

4.1 Comparison with other objective functions

In Hinton et al. (2018) an objective function $cost_k^h$ is defined (their eq. 1) which considers inference for the pose of a higher-level capsule k on pose dimension h. Translating to our notation, $cost_k^h$ combines the predictions $\mathbf{v}_{mk}$ from each datapoint m for capsule k as $cost_k^h = \sum_m r_{mk} \ln P_{m|k}^h$, where $P_{m|k}^h$ is a Gaussian, and r_{mk} is the "routing softmax assignment" between m and k. It is interesting to compare this with our equation (20). Firstly, note that the vote of $\mathbf{x}_m$ to part n in object k is given explicitly by $\beta \Lambda_k^{-1} F_{kn}^T D_{kn}^{-1} (\mathbf{x}_m - \boldsymbol{m}_{kn})$, i.e. we do not require the introduction of an independent voting mechanism, this falls out directly from the inference. Secondly, note that our R must keep track not only of assignments to objects, but also to *parts* of the objects. In our experiments with the constellations data each observed part could match any object/part combination of the models, so this is necessary. For the faces data, the observed parts are of identifiable type (e.g. nose, mouth), so in this case they only need to vote for the object. In contrast to Hinton et al. (2018), our inference scheme is derived from variational inference on the generative model for X, rather than introducing an *ad hoc* objective function that corresponds to a clustering in $\mathbf{y}$-space.

De Sousa Ribeiro et al. (2020a) develop a variational Bayes extension of the model of Hinton et al. (2018). They write down a mixture model in $\mathbf{y}$-space where the "datapoints" are the votes $\mathbf{v}_{mk}$, and use Bayesian priors for the mixing proportions, and for the means and covariance matrices of the components. This can improve training stability, e.g. by reducing the problem of variance-collapse where a capsule claims sole custody of of a datapoint. However, this is still a model for clustering in $\mathbf{y}$-space, and not a generative model for X.

The specialization of the SCAE method of Kosiorek et al. (2019) to constellation data is called the "constellation capsule autoencoder (CCAE)" and discussed in their sec. 2.1. Under their equation 5, we have that

$$p(\mathbf{x}_{1:M}) = \prod_{m=1}^{M} \sum_{k=1}^{K} \sum_{n=1}^{N} \frac{a_k a_{k,n}}{\sum_i a_i \sum_j a_{ij}} N(\mathbf{x}_m | \mu_{k,n}, \lambda_{k,n}), \tag{21}$$

Algorithm 3 RANSAC approach

T: K templates of the scene
B_k: base matrix for template T_k
X: M points of the scene
$out = []$
for $\mathbf{x}_i \in X$ **do**
    for $\mathbf{x}_j \in X \setminus x_i$ **do**
        $\mathbf{x}_{ij} = \text{Vectorize}(\mathbf{x}_i, \mathbf{x}_j)$
        for $k = 1 : K$ **do**
            $\hat{\mathbf{y}}_k = B_k^{-1}\mathbf{x}_{ij}$
            $T_k \xrightarrow{\hat{\mathbf{y}}_k} \hat{X}_k$
            if SubsetMatch($\hat{X}_k, X$) **then**
                Add $(T_k, \hat{\mathbf{y}}_k, \hat{X}_k)$ to out
return out

where $a_k \in [0, 1]$ is the presence probability of capsule k, $a_{k,n} \in [0, 1]$ is the conditional probability that a given candidate part n exists, and $\mu_{k,n}$ is the predicted location of part k, n. The a_ks are predicted by the network h^{caps}, while the $a_{k,n}$s and $\mu_{k,n}$s are produced by separate networks h_k^{part} for each part k.

We note that (21) provides an autoencoder style reconstructive likelihood for $\mathbf{x}_{1:M}$, as the a's and μ's depend on the data. To handle the arbitrary number of datapoints M, the network h^{caps} employs a Set Transformer architecture (Lee et al., 2019). In comparison to our iterative variational inference, the CCAE is a "one shot" inference mechanism. This may be seen as an advantage, but in scenes with overlapping objects, humans may perform reasoning like "if that point is one of the vertices of a square, then this other point needs to be explained by a different object" etc, and it may be rather optimistic to believe this can be done in a simple forward pass. Also, the CCAE cannot exploit prior knowledge of the geometry of the objects, as it relies on an opaque network h^{caps} which requires extensive training.

5 A RANSAC Approach to Inference

A radical alternative to "routing by agreement" inference is to make use of a "random sample consensus" approach (RANSAC, Fischler and Bolles, 1981), where a minimal number of parts are used in order to instantiate an object. The original RANSAC fitted just one object, but Sequential RANSAC (see, e.g., Torr 1998; Vincent and Laganière 2001) repeatedly removes the parts associated with a detected object and re-runs RANSAC, so as to detect all objects.

For the constellations problem, we can try matching any pair of points on one of the templates to every possible pair of $M(M-1)$ points in the scene. The key insight is that a pair of known points is sufficient to estimate the 4-dimensional $\hat{\mathbf{y}}_k$ vector in the case of similarity transformations. Using the transformation $\hat{\mathbf{y}}_k$, we can then *predict* the location of the remaining parts of the template, and *check* if these are actually present. If so, this provides evidence for the existence of T_k and $\hat{\mathbf{y}}_k$. After considering

the $M(M-1)$ subsets, the algorithm then combines the identified instantiations to give an overall explanation of the scene.

More details of the RANSAC approach are given in Algorithm 3. Assume we have chosen parts n_1 and n_2 as the basis for object k, and that we have selected datapoints $\mathbf{x}_i$ and $\mathbf{x}_j$ as their hypothesized counterparts in the image. Let $\mathbf{x}_{ij}$ be the vector obtained by stacking $\mathbf{x}_i$ and $\mathbf{x}_j$, and B_k be the 4×4 square matrix obtained by stacking F_{kn_1} and F_{kn_2}. Then $\hat{\mathbf{y}}_k = B_k^{-1}\mathbf{x}_{ij}$. Finally, SubsetMatch$(\hat{X}_k, X)$ selects those points in X_k that are close to X with a given tolerance and add them to the output. Among them, the solution is given by the one that minimizes $\sum_{n=1}^{N_k}(\hat{x}_{nk} - x_{nk})^2$.

The above algorithm chooses a specific basis for each object, but one can consider all possible bases for each object. It is then efficient to use a hash table to store the predictions for each part, as used in Geometric Hashing (Wolfson and Rigoutsos, 1997). Geometric Hashing is traditionally employed in computer vision to match geometric features against previously defined models of such features. This technique works well with partially occluded objects, and is computationally efficient if the basis dimension is low.

For the faces data, each part has location, scale and orientation information, so a single part is sufficient to instantiate the whole object geometrically. For the appearance, we follow the procedure for inference in a factor analysis model with missing data, as given in Williams et al. (2019), to predict $\mathbf{y}_k^a$ given the appearance of the single part.

It is interesting to compare the RANSAC and routing-by-agreement approaches to identifying objects. In RANSAC a minimal basis is chosen so as to instantiate the object, which is then verified by finding the remaining parts in the predicted locations. In contrast routing-by-agreement makes predictions from all of the observed parts, and then seeks to adjust the R matrix so as to cluster these predictions into coherent objects. This is generally an *iterative process*, although the SCAE autoencoder architecture is "one shot".[5]

6 Related Work

The origins of capsule networks (including the name "capsule") can be traced back at least as far as the work of Hinton et al. (2011) on Transforming Auto-encoders. Capsules were further developed in Sabour et al. (2017). Above we have already discussed later work by Hinton and co-workers, including Hinton et al. (2018), Kosiorek et al. (2019), and also De Sousa Ribeiro et al. (2020a).

The paper by De Sousa Ribeiro et al. (2022) provides a thorough survey of capsules. Below we highlight a few papers on capsules; for example Li and Zhu (2019) develop a capsule Restricted Boltzmann Machine (RBM). This generative model contains a number of capsules, each having a binary existence variable and a vector of capsule variables similar to our $\mathbf{y}_k$. However, these capsule variables are not used to model geometric transformations (the key idea behind capsule networks), but instead to handle appearance variability of the parts.

[5]One can extend variational autoencoders to use iterative amortized inference, see Marino et al. (2018).

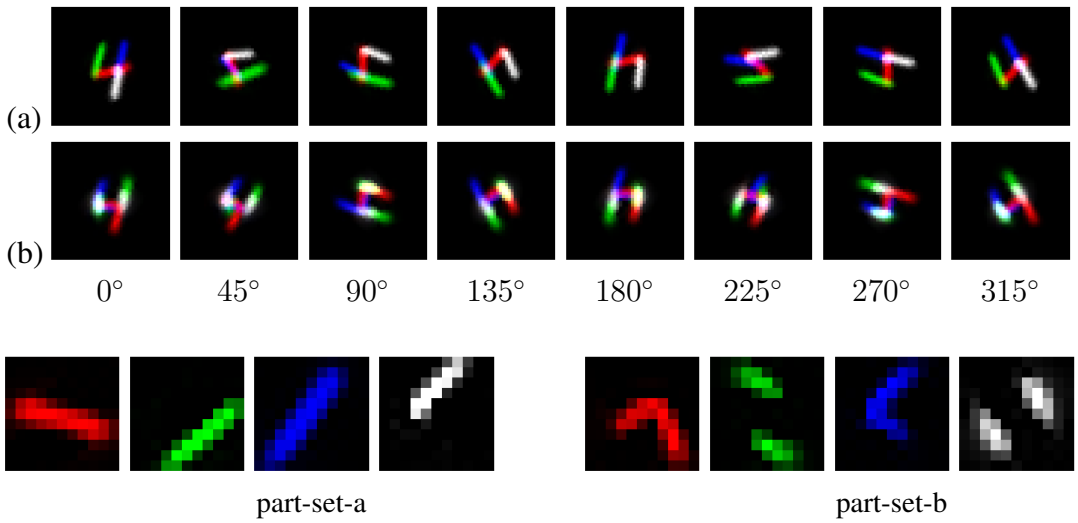


Figure 2: PCAE-based reconstruction for different angles of rotation of the input scenes. Row (a) corresponds to reconstructions with the learned part-set-a and row (b) reconstructions with the learned part-set-b. The parts are 11×11 pixels, and the images in rows (a) and (b) are 40×40.

Both Li et al. (2020) and Smith et al. (2021) propose a layered directed generative model. These both make use of the "single parent constraint", where a capsule in a layer can be connected to only one capsule (its parent) in the layer above. Such tree-structured relationships are similar to those studied in Hinton et al. (2000) and Storkey and Williams (2003). Like Li and Zhu (2019), Li et al. (2020) do not explicitly consider modelling geometric transformations with their network. This aspect is explicit in Smith et al. (2021), where their capsule variables (similar to our $\mathbf{y}_k$s) do model pose, but not appearance variability.

However, most importantly, these recent capsule models do not properly handle the fact that the input to a capsule should be a *set* of parts; instead in their work the first layer capsules that interact with the input image model specific location-dependent features/templates in the image, and their second layer capsules have interactions with the specific first layer capsules (e.g. the fixed affinity ρ_{ij}^k of Smith et al. (2021) parent i in layer k for child j). But if we consider a second-layer capsule that is, for example, detecting the conjunction of the 3 strokes comprising a letter “A”, then at different translations, scales and rotations of the A it will need to connect to different image level stroke features, so these connection affinities must depend on transformation variables of the parent. This point is illustrated in Fig. 2 of sec. 7 for the PCAE parts, and is also made in sec. 5.4 of Smith et al. (2021), where it is shown that the parts decomposition used of a digit image is not stable with respect to rotation of the digit.

7 Learning Generative Capsule Models

Above we have assumed that the models are given, i.e. that F_{kn} and $\boldsymbol{m}_{kn}$ are known for each part n and model k. One advantage of the interpretable nature of the GCM is that

one can use a "curriculum learning" approach (Bengio et al., 2009), where individual models can first be learned, and then composed together for inference in more complex situations.

For a given object, a key issue is the learning of the decomposition into parts. For the constellations data this is not an issue as the points (i.e. parts) are given, but for image data it must be addressed. An important issue is the degree of flexibility of each part—is it simply a fixed template, or are there internal degrees of flexibility? An example of the former is non-negative matrix factorization (NMF) (Lee and Seung, 1999), where, for example, aligned images of faces were decomposed into non-negative basis functions (parts) that were combined with non-negative coefficients. An example of a richer parts-based model is by Ross and Zemel (2006), who developed "multiple cause factor analysis" (MCFA) and applied it to faces, to learn the regions governed by each part and the variability within each part.[6] This work was also carried out on aligned, vertically oriented face images, so it was not necessary to factor out geometric transformations. For our work on faces, we made use of the parts-based model from the "PhotoFitMe" work described in section 8.2.1. This provided a ready-made parts decomposition, but we learned a factor analysis model on top to model the correlations between the parts.

Kosiorek et al. (2019) developed a Part Capsule Autoencoder (PCAE) to learn parts from images, and applied it to MNIST images. Each PCAE part is a template which can undergo an affine transformation, and it has "special features" that were used to encode the colour of the part. If the overall model is to be equivariant to geometric transformations, it is vital that the input part decomposition is stable to such variation, otherwise the model is building on shaky foundations. However, we have observed that the parts detected by PCAE are not equivariant to rotation. Figure 2 shows that the PCAE part decompositions inferred for a digit 4 are not stable to different angles of rotation: notice e.g. in panel (a) how the part coloured white maps to different parts of the 4 for $45°$-$180°$ and $225°$-$0°$. The details of this experiment are described in Appendix D.

As identified above, a strength of the GCM is that individual models can first be learned, before moving on to more complex situations. We can start with an initial guess for the object configuration, which can be chosen as one of the noisy configurations from the training set. We then run variational expectation maximization (variational EM; see sec. 6.2 in Jordan et al. 1999). In the E-step, variational inference is run as in sec. 4 with this model to infer the $\mathbf{y}$ and Z variables on each training example. In the M-step, the variatonal distributions determined in the E-step are held fixed, and the locations of the template points are updated so as to increase the ELBO, summed over the training cases. Details of this update are given in Appendix B for the constellations dataset, and experimental results are given in sec. 8.1.3.

[6] In contrast to our work they did not have a higher-level factor analyser to model correlations between part appearances, but did allow variability in the masks of the parts.

8 Experiments and Results

We first provide experimental details and results for inference and learning for the constellations data in sec. 8.1, and then give experimental details and results for the faces data in sec. 8.2.

8.1 Constellations: experiments and results

Below we provide details of the data generators, inference methods and evaluation criteria for the constellations data in sec. 8.1.1, present results for inference on the constellations data in sec. 8.1.2, and for learning constellation models in sec. 8.1.3.

8.1.1 Constellations experiments

In order to allow fair comparisons, we use the same dataset generator for geometric shapes employed by Kosiorek et al. (2019). We create a dataset of scenes, where each scene consists of a set of 2D points, generated from different geometric shapes. The possible geometric shapes (templates) are a square and an isosceles triangle, with parts being represented by the 2D coordinates of the vertices. We use the same dimensions for the templates as used by Kosiorek et al. (2019), side 2 for the square, and base and height 2 for the triangle. All templates are centered at $(0, 0)$. In every scene there are at most two squares and one triangle, so $N = 11$. Each shape is transformed with a random transformation to create a scene of 2D points given by the object parts. To match the evaluation of Kosiorek et al. (2019), all scenes are normalized so as the points lie in $[-1, 1]$ on both dimensions. When creating the scene, we select randomly (with probability 0.5) whether an object is going to be present or not, but delete empty scenes. A test set used for evaluation is comprised of 450-460 non-empty scenes, based on 512 draws.

Additionally, we study how the methods compare when objects are created from noisy templates. We consider that the original templates used for the creation of the images are corrupted with Gaussian noise with standard deviation σ. Once the templates are corrupted with noise, a random transformation $\mathbf{y}_k$ is applied to obtain the object X_k of the scene. As with the noise-free data, the objects are normalized to lie in [-1,1] on both dimensions.

The CCAE is trained by creating random batches of 128 scenes as described above and optimizing the objective function in (21). The authors run CCAE for $300K$ epochs, and when the parameters of the neural networks are trained, they use their model on the test dataset to generate an estimation of which points belong to which capsule, and where the estimated points are located in each scene.

The variational inference approach allows us to model scenes where the points are corrupted with some noise. The annealing parameter β controls the level of noise allowed in the model. We use an annealing strategy to fit β, increasing it every time the ELBO has converged, up to a maximum value of $\beta_{max} = 1$. We set the hyperparameters of the model to $\boldsymbol{\mu}_0^g = \mathbf{0}$, $D_0^g = I_4$, $\lambda = 10^4$ and $a_{mnk} = \frac{1}{N}$. We run Algorithm 2 with 5 different random initializations of R and select the solution with the best ELBO. Similarly to Kosiorek et al. (2019), we incorporate a sparsity constraint in our model,

that forces every object to explain at least two parts. Once our algorithm has converged, for a given k if any $r_{mnk} > 0.9$ and $\sum_m \sum_n r_{mnk} < 2$ it means that the model has converged to a solution where object k is assigned to less than 2 parts. In these cases, we re-run Algorithm 2 with a new initialization of R. Notice that this is also related to the minimum basis size necessary in the RANSAC approach for the types of transformations that we are considering.

The implementation of SubsetMatch in Algorithm 3 considers all matches between the predicted and the scene points where the distance between them is less than 0.1. Among them, it selects the matching with minimum distance between scene and predicted points.

For both the variational inference algorithm and the RANSAC algorithm, a training dataset is not necessary if we have prior knowledge of the target shapes. This contrasts with SCAE, which learns from whole scenes. If prior knowledge is not available, these shapes can be learned as described in sec. 7, and illustrated in sec. 8.1.3.

Unfortunately we do not have access to the code employed by Hinton et al. (2018), so we have been unable to make comparisons with it.

Evaluation: Three metrics are used to evaluate the performance of the different methods: variation of information (Meilă, 2003), adjusted Rand index (Hubert and Arabie, 1985) and segmentation accuracy (Kosiorek et al., 2019). They are based on partitions of the datapoints into those associated with each object, and those that are missing. Compared to standard clustering evaluation metrics, some modifications are needed to handle the missing objects. Details are provided in Appendix C. We also use an average scene accuracy metric, where a scene is correct if the method returns the full original scene, and is incorrect otherwise.

8.1.2 Constellation Inference Experiments

In Table 1 we show a comparison between CCAE, the variational inference method with a Gaussian mixture prior (8) (GCM-GMM), with a DS prior over permutation matrices (9) (GCM-DS), and the RANSAC approach for scenes without noise and with noise levels of $\sigma = 0.1,\ 0.25$.[7] For GCM-GMM and GCM-DS we show the results where the initial $\beta = 0.05$. The effect of different β initializations is discussed below.

We see that GCM-DS improves over CCAE and GCM-GMM in all of the metrics, with GCM-GMM being comparable to CCAE. Interestingly, for the noise-free scenarios, the RANSAC method achieves a perfect score for all of the metrics. Since there is no noise on the observations and the method searches over all possible solutions of $\mathbf{y}_k$, it can find the correct solution for any configuration of geometric shapes in a scene. For the noisy scenarios, all the methods degrade as σ increases. However, the relative performance between them remains the same, with RANSAC performing the best, followed by GCM-DS and then GCM-GMM.

Figure 3 shows some reconstruction examples from CCAE and GCM-DS for the noise-free scenario. In columns (a) and (b) we can see that CCAE recovers the correct parts assignments but the object reconstruction is inaccurate. In (a) one of the squares is reconstructed as a triangle, while in (b) the assignment between the reconstruction

[7] Code at: https://github.com/anazabal/GenerativeCapsules

Table 1: Comparison between the different methods. For Segmentation Accuracy, Adjusted Rand Index and Scene Accuracy the higher the better. For Variation of Information the lower the better. Different levels of Gaussian noise with standard deviation σ are considered.

Metric	Model	σ=0	σ=0.1	σ=0.25
Segmentation Accuracy ↑	CCAE	0.828	0.754	0.623
	GCM-GMM	0.753	0.757	0.744
	GCM-DS	0.899	0.882	0.785
	RANSAC	1	0.992	0.965
Adjusted Rand Index↑	CCAE	0.599	0.484	0.248
	GCM-GMM	0.586	0.572	0.447
	GCM-DS	0.740	0.699	0.498
	RANSAC	1	0.979	0.914
Variation of Information ↓	CCAE	0.481	0.689	0.988
	GCM-GMM	0.478	0.502	0.677
	GCM-DS	0.299	0.359	0.659
	RANSAC	0	0.034	0.135
Scene Accuracy ↑	CCAE	0.365	0.138	0.033
	GCM-GMM	0.179	0.173	0.132
	GCM-DS	0.664	0.603	0.377
	RANSAC	1	0.961	0.843

and the ground truth is not exact. For GCM-DS, if the parts are assigned to the ground truth properly, and there is no noise, then the reconstruction of the object is perfect. In column (c) all methods work well. In column (d), CCAE fits the square correctly (green), but adds an additional red point. In this case GCM-DS actually overlays two squares on top of each other. Both methods fail badly on column (e). Note that CCAE is not guaranteed to reconstruct an existing object correctly (square or triangle in this case). In column (f) we can see that CCAE fits an irregular quadrilateral (blue) to the assigned points, while GCM-DS obtains the correct fit. Additional examples for noisy cases with $\sigma = 0.25$ are shown in Appendix E.

To assess the effect of in the initial value of β, we considered 6 values: 0.005, 0.01, 0.05, 0.1, 0.2, and 0.5. We found that GCM-DS is always better than CCAE and GCM-GMM. As the initial β is increased, GCM-DS performs better across all the metrics. We found that the performance of GCM-DS and GCM-GMM degrades with $\beta > 0.1$.

We conducted paired t-tests between CCAE and GCM-GMM, GCM-DS and RANSAC on the three clustering metrics for $\sigma = 0$ and initial $\beta = 0.05$. The differences between CCAE and GCM-DS are statistically significant with p-values less than 10^{-7}, and between CCAE and RANSAC with p-values less than 10^{-28}. For CCAE and GCM-GMM the differences are not statistically significant.

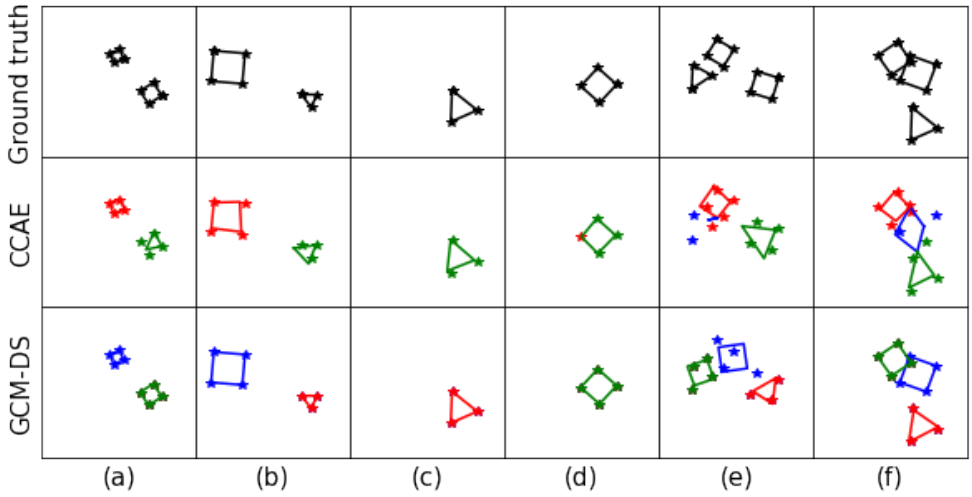


Figure 3: Reconstruction examples from CCAE and GCM-DS for noise-free data. The upper figures show the ground truth of the test images. The middle figures show the reconstruction and the capsule assignments (by colours) of CCAE. The lower figures show the reconstruction and the parts assignment of GCM-DS. Datapoints shown with a given colour are predicted to belong to the reconstructed object with the corresponding colour.

8.1.3 Learning Constellations

Making use of the interpretable nature of the GCM, we consider learning each template (triangle or square) individually from noisy data, with noise values of $\sigma = 0,\ 0.1,\ 0.25$. (Note that the full dataset as generated contains 1/7 = 14% single triangles, and 28% single squares which can be selected simply based on the number of points.)

To compare the learned template to the ground-truth (GT), we have to bear in mind that the learned template could be a rotated version, as this will still be centered and have the correct scale. To remove this rotational degree of freedom, we compute the Procrustes rotation (see, e.g., Mardia et al. 1979, sec. 14.7) that best aligns them. After this, we can compute the standardized mean squared error (SMSE) $\frac{1}{N}\sum_{n=1}^{N}(\mathbf{p}_n - \mathbf{p}_n^{GT})^T(\mathbf{p}_n - \mathbf{p}_n^{GT})$.

We use $S = 64$ examples to train each template, and use one random sample to seed the initial template (after translation, scaling and rotation). (In fact, for noise-free constellation data ($\sigma = 0$), a single observed triangle or square configuration serves as a perfect template.) As N is 3 or 4 in this case, we can use exact inference over permutations, rather than the variational approach. We initialized $\beta = 0.01$ and increased it after each iteration by a factor of two, similarly to the annealing strategy utilized in Algorithm 2. The stopping criterion for the learning process was that the SMSE between two consecutive updates of the learned template should not be greater than 10^{-4}.

For the triangle object, the SMSE of the learned template was 4.1×10^{-5} at $\sigma = 0.1$, and 9.6×10^{-3} at $\sigma = 0.25$. For the square object, the corresponding SMSEs were 1.8×10^{-4} and 2.1×10^{-2} respectively. Figure 4 shows example learned templates

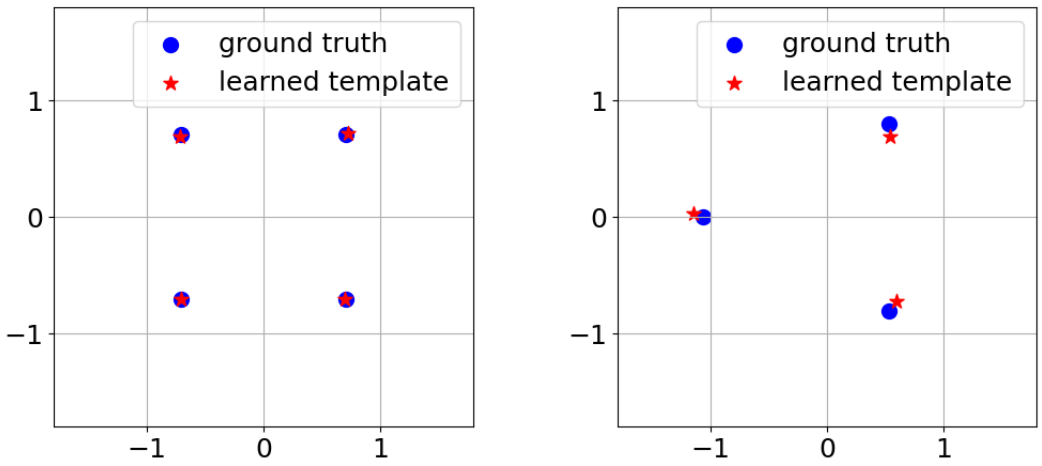


Figure 4: Visualization of GT templates and Procrustes transformed learned templates, after training with scenes of different noise levels.

(after Procrustes rotation) compared to the GT templates. We see that with only $S = 64$ examples, an accurate template can be learned for each object. As expected, the error increases with increasing σ. Note that $\sigma = 0.25$ gives quite noticeably distorted examples, see e.g. the examples in Fig. 6 (top row). In contrast, for the CCAE, the encoder network in the autoencoder is not able to benefit from curriculum learning, and must tackle the full problem for the start; Kosiorek et al. (2019) used 300k batches of 128 scenes to train this model.

8.2 Faces: experiments and results

Below we provide details of the data generator and inference methods for the faces data in sec. 8.2.1, and give inference results in sec. 8.2.2.

8.2.1 Parts-based face model

We have developed a novel hierarchical parts-based model for face appearances. It is based on five parts, namely eyes, nose, mouth, hair and forehead, and jaw (see Fig. 1(b)). Each part has a specified mask, and we have cropped the hair region to exclude highly variable hairstyles. This decomposition is based on the "PhotoFit Me" work and data of Prof. Graham Pike, see `https://www.open.edu/openlearn/PhotoFitMe`. For each part we trained a probabilistic PCA (PPCA) model to reduce the dimensionality of the raw pixels; the dimensionality is chosen so as to explain 95% of the variance. This resulted in dimensionalities of 24, 11, 12, 16 and 28 for the eyes, nose, mouth, jaw and hair parts respectively. We then add a factor analysis (FA) model on top with latent variables $\mathbf{y}_k^a$ to model the correlations of the PPCA coefficients across parts. The dataset used (from PhotoFit Me) is balanced by gender (female/male) and by race (Black/Asian/Caucasian), hence the high-level factor analyser can model regularities across the parts, e.g. wrt skin tone. $\mathbf{x}_n^a$ is predicted from $\mathbf{y}_k^a$ as $F_{kn}^a \mathbf{y}_k^a + \boldsymbol{m}_{kn}^a$ as in (5).

$\mathbf{y}_k^g$ would have an effect on the part appearance, e.g. by scaling and rotation, but this can be removed by running the PPCA part detectors on copies of the input image that have been rescaled and rotated.

The "PhotoFit Me" project utilizes 7 different part-images for each gender/race group, for each of the five part types. As a result, we generated 7^5 synthetic faces for each group, by combining these face-parts, which led to a total of $100,842$ faces. All faces were centered on a 224×224 pixel canvas. For each synthetic face we created an appearance vector $\mathbf{x}_n^a$, which consisted of the stacked vectors from the 5 different principal component subspaces. Finally, we created a balanced subset from the generated faces ($18,000$ images), which we used to train a FA model. We tuned the latent dimension of this model by training it multiple times with a different number of factors, and finally chose 12 factors, where a knee in the reconstruction loss on the face data was observed on a validation set.

To evaluate our inference algorithm we generated 224×224 pixel scenes of faces. These consisted of 2, 3, 4 or 5 randomly selected faces from a balanced test-set of $7,614$ synthetic faces, which were transformed with random similarity transformations. The face-objects were randomly scaled down by a minimum of 50% and were also randomly translated and rotated, with the constraint that all the faces fit the scene and did not overlap each other. An example of such a scene is shown in Fig. 1(c), and further examples are shown in Figure 5. For each scene it is easy to determine the correct number of faces, as the number of faces present is equal to the count of each of the parts detected. Afterwards, these two constraints were dropped to test the ability of our model to perform inference with occluded parts, see Figure 5(e) for an example. These occluded scenes were comprised of 3 faces. In our experiments we assume that the face parts are detected accurately, i.e. as generated.

In the case of facial parts—and parts with both geometric and appearance features in general—it only makes sense to assign the observed parts $\mathbf{x}_m$ to template parts $\mathbf{x}_{kn}$ of the same type (e.g. an observed "nose" part should be assigned only to a template "nose" part). We assume that this information is known, since the size of the appearance vector of each part-type is unique. Thus it no longer makes sense to initialize the assignment matrix uniformly for all entries, but rather only for the entries that correspond to templates of the observed part's type. Consequently, (14) is only utilized for m, n pairs of the same type. Similarly to the constellation experiments, we initialized the assignment matrix 5 times and selected the solution with the largest ELBO.

In the experiments we evaluated the part assignment accuracy of the algorithms. In a given scene, the assignment is considered correct if all the observed parts have been correctly assigned to their corresponding template parts with the highest probability. In order to evaluate the prediction of the appearance features, we measured the root mean square error (RMSE) between the input and generated scenes.

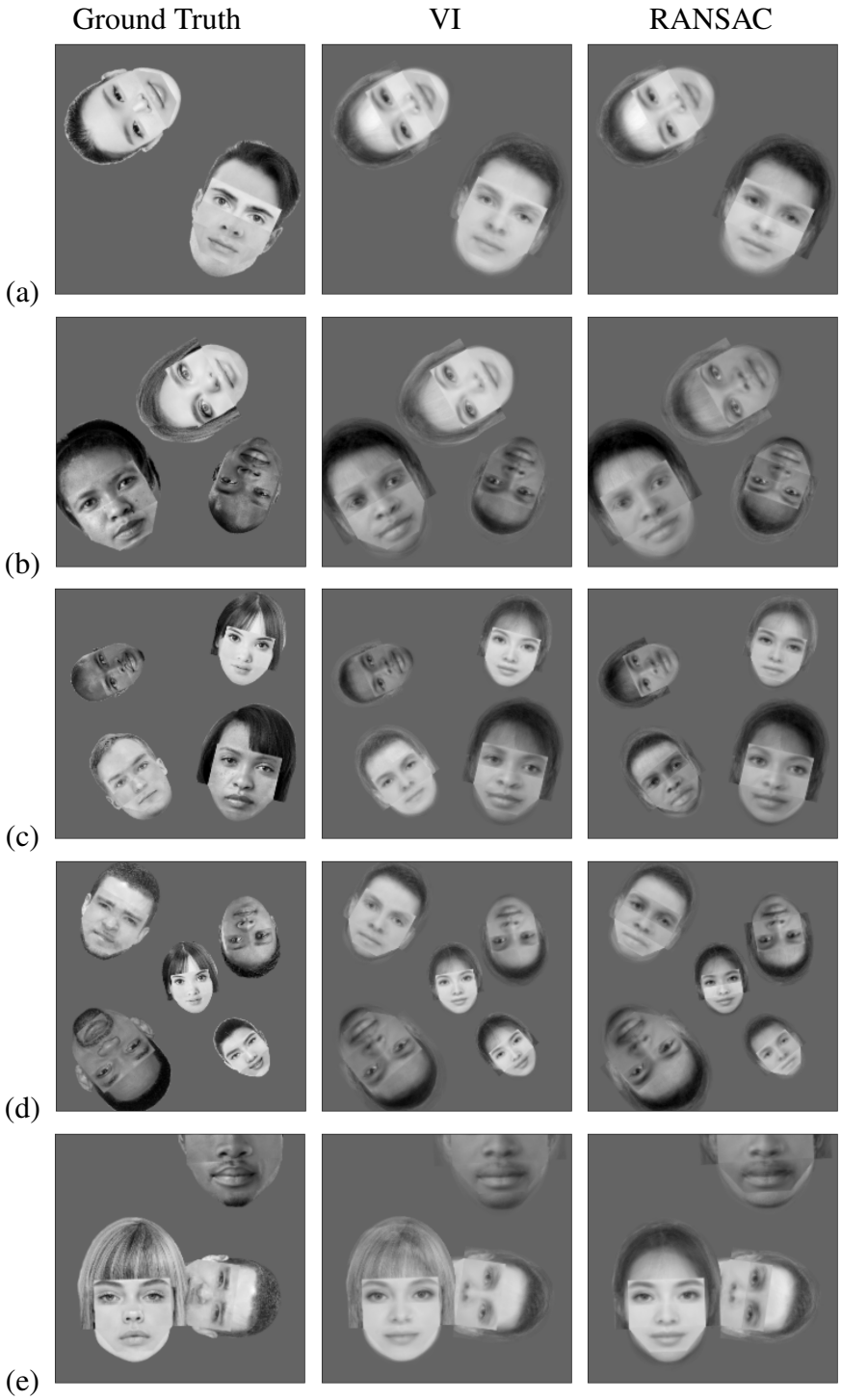


Figure 5: Reconstruction examples with our Variational Inference (VI) algorithm and the RANSAC-type algorithm: (a) scene with 2 faces, (b) scene with 3 faces, (c) scene with 4 faces (d) scene with 5 faces and (e) 3 faces with partially occluded faces. All faces have been randomly selected and transformed.

8.2.2 Face Experiments

Firstly, the VI algorithm was evaluated on scenes of multiple, randomly selected and transformed faces.[8] For scenes with 2, 3, 4 and 5 faces, the assignment accuracy was 100%, 100%, 99.2% and 93.7% respectively (based on 250 scenes per experiment). RANSAC gave 100% accurate assignments in all four cases. This is to be expected, since from each part the pose of the whole can be predicted accurately. However, RANSAC's ability to infer the appearance of the faces proved to be limited. More specifically, in 250 instances uniformly distributed across scenes of 2, 3, 4 and 5 faces, the VI algorithm had RMSE of 0.036 ± 0.004, while RANSAC scored 0.052 ± 0.006, with consistently higher error on *all* scenes. This is illustrated in the examples of Figure 5, where it is clear that RANSAC is less accurate in capturing key facial characteristics. If inference for $\mathbf{y}_k^a$ is run as a post-processing step for RANSAC using all detected parts in an object, this difference disappears.

The supplementary material contains a movie showing the fitting of the models to the data. It is not possible for us to make a fair comparison with the SCAE algorithm on the faces data, as the PCAE model used is not rich enough to model PCA subspaces.

Secondly, we evaluated the ability of our algorithm to perform inference in scenes where some parts have been occluded, either by overlapping with other faces or by extending outside of the scene. In 250 scenes with 3 partially occluded faces, both the VI and RANSAC algorithms were fully successful in assigning the observed parts to the corresponding template accurately; see Figure 5(e) for an example.

9 Discussion

In our experiments RANSAC was shown to often be an effective alternative to variational inference (VI). This is particularly the case when the basis in RANSAC is highly informative about the object. For the constellations experiments this was the case, even when the datapoints were corrupted by noise. However, as we saw in sec. 8.2.2 for the faces data, an individual part was less informative about the appearance than the geometry, and so led to worse reconstructions unless a post-processing step using all of the detected parts was used. Also, RANSAC's sampling-based inference may be less amenable to neural-network style parallel implementations than VI.

Above we have described a *generative* capsules model. The key features of this approach are:

- The model is *interpretable*, and thus admits the use of prior knowledge, e.g. if we already know some things about an object. The formulation is also *composable* in that models for individual objects can be learned separately, then combined together at inference time.

- The variational inference algorithm is obtained directly from a generative model for the observations X. In contrast other leading formulations set up an objective to produce clusters in $\mathbf{y}$-space.

[8] Code at: https://github.com/tsagkas/capsules

- The interpretable structure of the GCM allows other inference methods to be used, as demonstrated by our use of RANSAC.
- The GCM conforms to the view, as promoted in Kosiorek et al. (2019), that the input is regarded as a *set* of parts. This formulation ensures that if the parts can be detected equivariantly, then the inferences for the objects will also be equivariant. This was demonstrated in the constellations and faces experiments.

As noted above, for the GCM to be equivariant to large transformations of the input, the parts need to to be detected equivariantly. Some capsules papers have used the affNIST dataset[9], but this only used small rotations of up to $\pm 20^\circ$. Hinton et al. (2018, sec. 5.1) did investigate the use of very different viewpoints on the smallNORB dataset; while their capsules results in Table 2 did outperform a competitor CNN, it is noticeable that there is still a performance gap between novel and familiar viewpoints. We have demonstrated (see Fig. 2) that the PCAE decomposition is not equivariant to large rotations, and similar observations have been made by Smith et al. (2021) for their model. Thus we believe that further work on the equivariant extraction of parts is necessary in order to achieve equivariant object recognition.

Acknowledgements

We thank the anonymous referees for their helpful comments. This work was supported in part by The Alan Turing Institute under EPSRC grant EP/N510129/1. For the purpose of open access, the authors have applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising from this submission.

A Details for Variational Inference

The evidence lower bound (ELBO) $L(q) = \mathbb{E}_q[\log p(X, Y, Z) - \log q(Y, Z)]$ for this model is decomposed in three terms:

$$L(q) = \mathbb{E}_q[\log p(X|Y, Z)] - KL(q(Y)||p(Y)) - KL(q(Z)||p(Z)), \tag{22}$$

where $KL(q||p)$ is the Kullback-Leibler divergence between distributions q and p. The first term indicates how well the generative model $p(X|Y, Z)$ fits the observations under our variational model $q(Y, Z)$:

$$\mathbb{E}_q[\log p(X|Y, Z)] = -\sum_{m=1}^{M}\sum_{k=1}^{K}\sum_{n=1}^{N_k} r_{mnk}\Big[\frac{d_{kn}}{2}\log 2\pi + \frac{1}{2}\log|\beta^{-1}D_{kn}| + \frac{\beta}{2}(\mathbf{x}_m - F_{kn}\boldsymbol{\mu}_k - \boldsymbol{m}_{kn})^T D_{kn}^{-1}(\mathbf{x}_m - F_{kn}\boldsymbol{\mu}_k - \boldsymbol{m}_{kn}) + \frac{\beta}{2}\text{trace}(F_{kn}^T D_{kn}^{-1} F_{kn}\Lambda_k^{-1})\Big]. \tag{23}$$

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

The Kullback-Leibler divergence between the two Gaussian distributions $q(Y)$ and $p(Y)$ in our model has the following expression:

$$KL(q(Y)||p(Y)) = \\ \frac{1}{2}\sum_{k=1}^{K}\left(\text{trace}(D_0^{-1}\Lambda_k^{-1}) - d_k + (\boldsymbol{\mu}_k - \boldsymbol{\mu}_0)^T D_0^{-1}(\boldsymbol{\mu}_k - \boldsymbol{\mu}_0) + \log|D_0| + \log|\Lambda_k|\right), \tag{24}$$

where d_k is the dimensionality of $\mathbf{y}_k$.

The expression for $KL(q(Z)||p(Z))$ is given by

$$KL(q(Z)||p(Z)) = \sum_{m=1}^{N}\sum_{k=1}^{K}\sum_{n=1}^{N_k} r_{mnk}\log\frac{r_{mnk}}{a_{mnk}}. \tag{25}$$

B Learning for the Constellations Model

We specialize the ELBO given in Appendix A for one constellation, taking $K = 1$ (and hence dropping the index k). As there are no appearance features, we drop the g superscript on F_n and $\mathbf{y}$. Also the mean $\boldsymbol{m}_{kn}$ is only needed for the appearance features and thus can be omitted. D_n is specialized to $\lambda^{-1}I_2$, we set $\beta = 1$ in this appendix and allow λ to vary, and we assume $p(\mathbf{y}) \sim N(\mathbf{0}, \Lambda_0^{-1})$. Hence by specializing eq. 23 we obtain

$$\mathbb{E}_q[\log p(X|Y,Z)] = \\ -\sum_{m=1}^{M}\sum_{n=1}^{N} r_{mn}\Big[\log\frac{2\pi}{\lambda} + \frac{\lambda}{2}(\mathbf{x}_m - F_n\boldsymbol{\mu})^T(\mathbf{x}_m - F_n\boldsymbol{\mu}) + \frac{\lambda}{2}\text{trace}(F_n^T F_n\Lambda^{-1})\Big], \tag{26}$$

where $q(\mathbf{y}) \sim N(\boldsymbol{\mu}, \Lambda^{-1})$, with Λ and $\boldsymbol{\mu}$ specialized from eqs. 19 and 20 as

$$\Lambda = \Lambda_0 + \lambda\sum_{m=1}^{M}\sum_{n=1}^{N} r_{mn}F_n^T F_n, \qquad \boldsymbol{\mu} = \lambda\,\Lambda^{-1}(\sum_{m=1}^{M}\sum_{n=1}^{N} r_{mn}F_n^T\mathbf{x}_m). \tag{27}$$

Our goal in learning is to adapt the template parameters $\{F_n\}$ so as to increase the variational log likelihood (ELBO) $L(q)$. In the M-step of variational EM, the distributions $q(\mathbf{y})$ and $q(Z)$ (parameterized by $\boldsymbol{\mu}$, Λ and R) are held fixed, and the ELBO is optimized wrt $\{F_n\}$. Note that the terms $KL(q(Y)||p(Y))$ and $KL(q(Z)||p(Z))$ do not depend explicitly on $\{F_n\}$, and hence any derivative of these KL terms wrt F_n will be zero. Thus these terms can be omitted when optimizing the ELBO wrt $\{F_n\}$.

The trace term in eq. 26 can be simplified using $\sum_m r_{mn} = 1$, to give

$$\sum_{m=1}^{M}\sum_{n=1}^{N} r_{mn}\text{trace}(F_n^T F_n\Lambda^{-1}) = \text{trace}((\sum_{n=1}^{N} F_n^T F_n)\Lambda^{-1}). \tag{28}$$

It turns out that it is more convenient to write $F_n\boldsymbol{\mu}$ in terms of $\mathbf{p}_n = (p_{nx}, p_{ny})^T$ and the mean transformation parameters $\boldsymbol{\mu} = (\hat{t}_x, \hat{t}_y, \hat{s}_c, \hat{s}_s)^T$, where $\hat{s}_c$ is the posterior mean of $s\cos\theta$, and $\hat{s}_s$ is the same for $s\sin\theta$. Hence we have that

$$F_n\boldsymbol{\mu} = \begin{pmatrix} \hat{\mathbf{t}}_x \\ \hat{t}_y \end{pmatrix} + \begin{pmatrix} \hat{s}_c & \hat{s}_s \\ -\hat{s}_s & \hat{s}_c \end{pmatrix} \begin{pmatrix} p_{nx} \\ p_{ny} \end{pmatrix} \stackrel{def}{=} \hat{\mathbf{t}} + \hat{T}\mathbf{p}_n. \tag{29}$$

Hence $\mathbf{x}_m - F_n\boldsymbol{\mu} = \mathbf{x}_m - \hat{\mathbf{t}} - \hat{T}\mathbf{p}_n = \tilde{\mathbf{x}}_m - \hat{T}\mathbf{p}_n$, where $\tilde{\mathbf{x}}_m = \mathbf{x}_m - \hat{\mathbf{t}}$, and the quadratic form $(\mathbf{x}_m - F_n\boldsymbol{\mu})^T(\mathbf{x}_m - F_n\boldsymbol{\mu})$ can be rewritten as $(\tilde{\mathbf{x}}_m - \hat{T}\mathbf{p}_n)^T(\tilde{\mathbf{x}}_m - \hat{T}\mathbf{p}_n)$.

We can now rewrite the term $Q = \sum_m \sum_n r_{mn}(\mathbf{x}_m - F_n\boldsymbol{\mu})^T(\mathbf{x}_m - F_n\boldsymbol{\mu})$ in eq. 26 as

$$Q = \sum_m \sum_n r_{mn}(\tilde{\mathbf{x}}_m - \hat{T}\mathbf{p}_n)^T(\tilde{\mathbf{x}}_m - \hat{T}\mathbf{p}_n) \tag{30}$$

$$= \sum_m \sum_n r_{mn}\left(\mathbf{p}_n^T\hat{T}^T\hat{T}\mathbf{p}_n - \mathbf{p}_n^T\hat{T}^T\tilde{\mathbf{x}}_m - \tilde{\mathbf{x}}_m^T\hat{T}p_n + \tilde{\mathbf{x}}_m^T\tilde{\mathbf{x}}_m\right). \tag{31}$$

Using $\sum_m r_{mn} = 1$ and defining $\tilde{\mathbf{x}}_n^r = \sum_m r_{mn}\tilde{\mathbf{x}}_m$, we obtain

$$Q = \sum_n \mathbf{p}_n^T\hat{T}^T\hat{T}\mathbf{p}_n - \mathbf{p}_n^T\hat{T}^T\tilde{\mathbf{x}}_n^r - (\tilde{\mathbf{x}}_n^r)^T\hat{T}\mathbf{p}_n + (\sum_{m,n} r_{mn}\tilde{\mathbf{x}}_m^T\tilde{\mathbf{x}}_m). \tag{32}$$

This can be further simplified by noting that $\hat{T}^T\hat{T} = \hat{s}^2 I_2$, where $\hat{s}^2 = \hat{s}_c^2 + \hat{s}_s^2$.

The above derivation is all for one example X. Now summing Q over all training examples $\{X_i\}$ we obtain

$$Q^{tot} = \sum_i Q_i = \sum_i \sum_n \left(\hat{s}_i^2\mathbf{p}_n^T\mathbf{p}_n - \mathbf{p}_n^T\hat{T}^T\tilde{\mathbf{x}}_{ni}^r - (\tilde{\mathbf{x}}_{ni}^r)^T\hat{T}\mathbf{p}_n + \sum_m r_{mn}\tilde{\mathbf{x}}_{mi}^T\tilde{\mathbf{x}}_{mi}\right), \tag{33}$$

where $\tilde{\mathbf{x}}_{ni}^r$ denotes $\tilde{\mathbf{x}}_n^r$ in the ith example, and similarly for $\tilde{\mathbf{x}}_{mi}$.

Now consider the trace term $S_i = \text{trace}((\sum_{n=1}^N F_n^T F_n)\Lambda_i^{-1})$, where Λ_i is the precision matrix for $\mathbf{y}$ on the ith example. We have that

$$F_n^T F_n = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ p_{nx} & p_{ny} \\ p_{ny} & -p_{nx} \end{pmatrix} \begin{pmatrix} 1 & 0 & p_{nx} & p_{ny} \\ 0 & 1 & p_{ny} & -p_{nx} \end{pmatrix} = \begin{pmatrix} 1 & 0 & p_{nx} & p_{ny} \\ 0 & 1 & p_{ny} & -p_{nx} \\ p_{nx} & p_{ny} & p_{nx}^2 + p_{ny}^2 & 0 \\ p_{ny} & -p_{nx} & 0 & p_{nx}^2 + p_{ny}^2 \end{pmatrix}. \tag{34}$$

Assume that the template is centered so that $\sum_n p_{nx} = \sum_n p_{ny} = 0$, and scaled so that $\sum_n p_{nx}^2 + p_{ny}^2 = N$. Hence we have that $\sum_n F_n^T F_n = N I_4$. From eq. 27 and taking $\Lambda_0 = \alpha I_4$ we have

$$\Lambda = \Lambda_0 + \sum_m \sum_n r_{mn} F_n^T F_n = (\alpha + \lambda N) I_4. \tag{35}$$

Hence

$$\text{trace}((\sum_{n=1}^N F_n^T F_n)\Lambda^{-1}) = N\text{trace}(\Lambda^{-1}) = \frac{4N}{\alpha + \lambda N}. \tag{36}$$

Crucially, this term is *independent* of $\{\mathbf{p}_n\}$, as long as the template is centered and scaled correctly. Hence this term can be ignored when optimizing the $\mathbf{p}_n$s.

Thus optimizing the ELBO wrt $\mathbf{p}_n$ comes down to optimizing Q^{tot} wrt $\mathbf{p}_n$. Differentiating eq. 33 wrt $\mathbf{p}_n$ and setting it equal to zero we obtain

$$\frac{\partial Q^{tot}}{\partial \mathbf{p}_n} = \sum_i \left(2\hat{s}_i^2 \mathbf{p}_n - 2\hat{T}_i^T \tilde{\mathbf{x}}_{ni}^r \right) = 0, \tag{37}$$

which gives the update formula

$$\mathbf{p}_n = \frac{1}{\sum_i \hat{s}_i^2} \sum_i \hat{T}_i^T \tilde{\mathbf{x}}_{ni}^r. \tag{38}$$

It can also be shown that $\hat{T}^T = \hat{s}^2 \hat{T}^{-1}$, yielding the update equation

$$\mathbf{p}_n = \frac{\sum_i \hat{s}_i^2 \hat{T}_i^{-1} \tilde{\mathbf{x}}_{ni}^r}{\sum_i \hat{s}_i^2}. \tag{39}$$

This is quite intuitive—we first remove the effect of the translation $\hat{\mathbf{t}}$ by computing $\tilde{\mathbf{x}}_m$, then take into account the weighted assignments r_{mn} to give $\tilde{\mathbf{x}}_{ni}^r$, and then apply $\hat{T}_i^{-1}$ to remove the effect of the scaling and rotation. The summations are weighted by $\hat{s}_i^2$, which has the effect giving higher weight to examples with larger scale, where the relative effect of the noise $N(0, \lambda^{-1})$ is smaller.

One can also re-estimate λ using the variational EM algorithm. Differentiating eq. 26 wrt λ we obtain

$$\frac{1}{\lambda} = \frac{1}{2NS} \sum_{i=1}^{S} \sum_{m,n} r_{mn}^i (\mathbf{x}_{mi} - F_n \boldsymbol{\mu}_i)^T (\mathbf{x}_{mi} - F_n \boldsymbol{\mu}_i), \tag{40}$$

where S denotes the number of examples. As λ^{-1} is a variance, this equation makes sense in terms of an average of squared residuals. In the derivation of eq. 40 the dependence of the trace term $4N\lambda/(\alpha + N\lambda)$ on λ has been omitted, as for $N\lambda \gg \alpha$ this dependence is negligible.

C Evaluation metrics for the constellations data

In a given scene X there are M points, but we know that there are $N \geq M$ possible points that can be produced from all of the templates. Assume that $K' \leq K$ templates are active in this scene. Then the points in the scene are labelled with indices $1, \ldots, K'$, and we assign the missing points index 0. Denote the ground truth partition as $V = \{V_0, V_1, \ldots V_{K'}\}$. An alternative partition output by one of the algorithms is denoted by $\hat{V} = \{\hat{V}_0, \hat{V}_1, \ldots \hat{V}_{\hat{K}'}\}$. The predicted partition $\hat{V}$ may instantiate objects or points that were in fact missing, thus it is important to handle the missing data properly.

In Information Theory, the **variation of information** (VI) (Meilă, 2003) is a measure of the distance between two partitions of elements (or clusterings). For a given

set of elements, the variation of information between two partitions V and $\hat{V}$, where $N = \sum_i |V_i| = \sum_j |\hat{V}_j|$ is defined as:

$$VI(V, \hat{V}) = -\sum_{i,j} r_{ij} \left[\log \frac{r_{ij}}{p_i} + \log \frac{r_{ij}}{q_j} \right] \tag{41}$$

where $r_{ij} = \frac{|V_i \cap \hat{V}_j|}{N}$, $p_i = \frac{|V_i|}{N}$ and $q_j = \frac{|\hat{V}_j|}{N}$. In our experiments we report the average variation of information of the scenes in the dataset.

The **Rand index** (Rand, 1971) is another measure of similarity between two data clusterings. This metric takes pairs of elements and evaluates whether they do or do not belong to the same subsets in the partitions V and $\hat{V}$

$$RI = \frac{TP + TN}{TP + TN + FP + FN}, \tag{42}$$

where TP are the true positives, TN the true negatives, FP the false positives and FN the false negatives. The Rand index takes on values between 0 and 1. We use instead the **adjusted Rand index** (ARI) (Hubert and Arabie, 1985), the corrected-for-chance version of the Rand index. It uses the expected similarity of all pair-wise comparisons between clusterings specified by a random model as a baseline to correct for assignments produced by chance. Unlike the Rand index, the adjusted Rand index can return negative values if the index is less than the expected value. In our experiments, we compute the average adjusted Rand index of the scenes in our dataset.

The **segmentation accuracy** (SA) is based on obtaining the maximum bipartite matching between V and $\hat{V}$, and was used by Kosiorek et al. (2019) to evaluate the performance of CCAE. For each set V_i in V and set $\hat{V}_j$ in $\hat{V}$, there is an edge w_{ij} with the weight being the number of common elements in both sets. Let $W(V, \hat{V})$ be the overall weight of the maximum matching between V and $\hat{V}$. Then we define the average segmentation accuracy as:

$$SA = \sum_{i=1}^{I} \frac{W(V_i, \hat{V}_i)}{W(V_i, V_i)} = \frac{1}{N} \sum_{i=1}^{I} W(V_i, \hat{V}_i), \tag{43}$$

where I is the number of scenes. Notice that $W(V_i, V_i)$ represents a perfect assignment of the ground truth, both the observed and missing subsets, and thus $W(V_i, V_i) = N$.

There are some differences on how we compute the SA metric compared to Kosiorek et al. (2019). First, they do not consider the missing points as part of their ground truth, but as we argued above this is necessary. They evaluate the segmentation accuracy in terms of the observed points in the ground truth, disregarding possible points that were missing in the ground truth but predicted as observed in $\hat{V}$. Second, they average the segmentation accuracy across scenes as

$$SA = \frac{\sum_{i=1}^{I} W(V_i, \hat{V}_i)}{\sum_{i=1}^{I} W(V_i, V_i)}. \tag{44}$$

For them, $W(V_i, V_i) = M_i$, where M_i is the number of points present in a scene. In our case, both averaging formulae are equivalent since our $W(V_i, V_i)$ is the same across scenes.

D Failures of rotation equivariance for SCAE

We trained the SCAE model with digit "4" images from the training set of the MNIST dataset[10], after they had been uniformly rotated by up to 360° and uniformly translated by up to 6 pixels on the x and y axes. Since we used a single class in the dataset we modified the SCAE architecture to use only a single object capsule.

We repeated the training of SCAE multiple times for 8K epochs, and collected distinct sets of learned 11×11 parts that the digit "4" can be decomposed into. Two example part-sets are shown in Fig. 2. We then evaluated PCAE's ability to detect these parts in MNIST digit "4" images that had been rotated by multiples of 45°. Our results indicate that the PCAE model is not equivariant to rotations. This is apparent from Fig. 2, where the learned parts are inconsistently assigned to the regions of the digit-object, depending on the angle of rotation. We hypothesize that this phenomenon stems from the fact that PCAE seems to generate parts that are either characterized by an intrinsic symmetry, and thus their pose is ambiguous, as in the line features of part-set-a in Fig. 2); or pairs of parts that are transformed versions of themselves, and thus can be used interchangeably (e.g. the first and third templates of part-set-b in Fig. 2). This leads to identifiability issues, where the object can be decomposed into its parts in numerous ways.

E Examples of Noisy Cases

Figure 6 shows several examples of objects generated from noisy templates with a corruption level of $\sigma = 0.25$. GCM-DS and RANSAC do not allow for deformable objects to try to fit the points exactly, contrary to CCAE. Both methods try to find the closest reconstruction of the noisy points in the image by selecting the geometrical shapes that are a best fit to those points. Nonetheless, both methods can determine that which parts belong together to form a given object, even when the matching is not perfect.

References

Basri, R. (1996). Paraperspective ≡ Affine. *International Journal of Computer Vision*, 19:169–179.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum Learning. In *Proc. 26th International Conference on Machine Learning*.

Biederman, I. (1987). Recognition-by-Components: A Theory of Human Image Understanding. *Psychological Review*, 94:115–147.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

De Sousa Ribeiro, F., Duarte, K., Everett, M., Leontidis, G., and Shah, M. (2022). Learning with Capsules: A Survey. arXiv:2206.02664.

[10] http://yann.lecun.com/exdb/mnist .

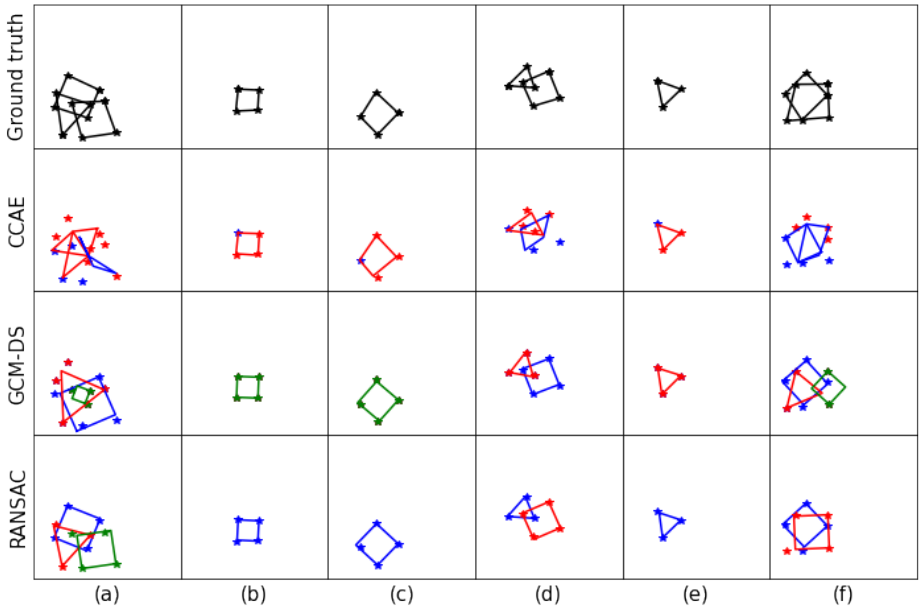


Figure 6: Reconstruction examples from CCAE, GCM-DS and RANSAC with Gaussian noise $\sigma = 0.25$.

De Sousa Ribeiro, F., Leontidis, G., and Kollias, S. (2020a). Capsule Routing via Variational Bayes. In *Proc. Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20)*, pages 3749–3756.

De Sousa Ribeiro, F., Leontidis, G., and Kollias, S. (2020b). Introducing Routing Uncertainty in Capsule Networks. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6490–6502. Curran Associates, Inc.

Felzenszwalb, P., Girshick, R., McAllester, D., and Ramanan, D. (2009). Object Detection with Discriminatively Trained Part Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645.

Fergus, R., Perona, P., and Zisserman, A. (2003). Object Class Recognition by Unsupervised Scale-Invariant Learning. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Fischler, M. A. and Bolles, R. C. (1981). Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6):381–395.

Fischler, M. A. and Elschlager, R. A. (1973). The Representation and Matching of Pictoral Structures. *IEEE Transactions on Computers*, 22(1):67–92.

Hinton, G. E., Ghahramani, Z., and Teh, Y. W. (2000). Learning to Parse Images. In Solla, S. A., Leen, T. K., and Müller, K.-R., editors, *Advances in Neural Information Processing Systems 12*, pages 463–469. MIT Press, Cambridge, MA.

Hinton, G. E., Krizhevsky, A., and Wang, S. D. (2011). Transforming Auto-encoders. In *Proceedings ICANN 2011*.

Hinton, G. E., Sabour, S., and Frosst, N. (2018). Matrix Capsules with EM Routing. In *International Conference on Learning Representations*.

Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of Classification*, 2(1):193–218.

Jordan, M. I., Ghahramani, Z., and Jaakkola, T. S. Saul, L. K. (1999). An Introduction to Variational Methods for Graphical Models. *Machine Learning*, 37:183–233.

Kosiorek, A., Sabour, S., Teh, Y. W., and Hinton, G. E. (2019). Stacked Capsule Autoencoders. In *Advances in Neural Information Processing Systems*, pages 15512–15522.

Lee, D. D. and Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791.

Lee, J., Lee, Y., Kim, J., Kosiorek, A. R., Choi, S., and Teh, Y.-W. (2019). Set Transformer: A Framework for Attention-based Permutation-invariant Neural Networks. In *Proc. of the 36th International Conference on Machine Learning*, pages 3744–3753.

Li, Y. and Zhu, X. (2019). Capsule Generative Models. In Tetko, I. V. et al., editors, *ICANN*, pages 281–295. LNCS 11727.

Li, Y., Zhu, X., Naud, R., and Xi, P. (2020). Capsule Deep Generative Model That Forms Parse Trees. In *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE.

Mardia, K. V., Kent, J. T., and Bibby, J. M. (1979). *Multivariate Analysis*. Academic Press, London.

Marino, J., Yue, Y., and Mandt, S. (2018). Iterative Amortized Inference. In *ICML*.

Meilă, M. (2003). Comparing Clusterings by the Variation of Information. In *Learning Theory and Kernel Machines*, pages 173–187. Springer.

Mena, G., Varol, E., Nejatbakhsh, A., Yemini, E., and Paninski, L. (2020). Sinkhorn Permutation Variational Marginal Inference. In *Symposium on Advances in Approximate Bayesian Inference*, pages 1–9. PMLR.

Nazabal, A., Tsagkas, N., and Williams, C. K. I. (2022). Inference and Learning for Generative Capsule Models. arXiv:2209.03115.

Powell, B. and Smith, P. A. (2019). Computing expectations and marginal likelihoods for permutations. *Computational Statistics*, pages 1–21.

Rand, W. M. (1971). Objective Criteria for the Evaluation of Clustering Methods. *Journal of the American Statistical Association*, 66(336):846–850.

Rao, R. P. N. and Ballard, D. H. (1999). Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature Neurosci.*, 2(1):79–87.

Revow, M., Williams, C. K. I., and Hinton, G. E. (1996). Using Generative Models for Handwritten Digit Recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(6):592–606.

Ross, D. and Zemel, R. (2006). Learning Parts-Based Representations of Data. *Journal of Machine Learning Research*, 7:2369–2397.

Sabour, S., Frosst, N., and Hinton, G. (2017). Dynamic Routing Between Capsules. In *Advances in Neural Information Processing Systems*, pages 3856–3866.

Sinkhorn, R. and Knopp, P. (1967). Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348.

Smith, L., Schut, L., Gal, Y., and van der Wilk, M. (2021). Capsule Networks—A Generative Probabilistic Perspective. https://arxiv.org/pdf/2004.03553.pdf.

Storkey, A. J. and Williams, C. K. I. (2003). Image Modelling with Position-Encoding Dynamic Trees. *IEEE Trans Pattern Analysis and Machine Intelligence*, 25(7):859–871.

Torr, P. H. S. (1998). Geometric Motion Segmentation and Model Selection. *Philosophical Trans. of the Royal Society A*, 356:1321–1340.

Vincent, E. and Laganière, R. (2001). Detecting planar homographies in an image pair. In *Proceedings of the 2nd International Symposium on Image and Signal Processing and Analysis (ISPA)*.

Williams, C. K. I., Nash, C., and Nazabal, A. (2019). Autoencoders and Probabilistic Inference with Missing Data: An Exact Solution for The Factor Analysis Case. arXiv:1801.03851.

Wolfson, H. J. and Rigoutsos, I. (1997). Geometric Hashing: An Overview. *IEEE Computational Science and Engineering*, 4(4):10–21.