Introducing Routing Uncertainty in Capsule Networks

Fabio De Sousa Ribeiro\textsuperscript{\textcopyright} Georgios Leontidis\textdagger \ Stefnanos Kollias\textsuperscript{\textcopyright}

\textsuperscript{\textcopyright} Machine Learning Group
University of Lincoln, UK
\{fdesousaribeiro,skollias\}@lincoln.ac.uk

\textdagger Department of Computing Science
University of Aberdeen, UK
georgios.leontidis@abdn.ac.uk

Abstract

Rather than performing inefficient local iterative routing between adjacent capsule layers, we propose an alternative global view based on representing the inherent uncertainty in part-object assignment. In our formulation, the local routing iterations are replaced with variational inference of part-object connections in a probabilistic capsule network, leading to a significant speedup without sacrificing performance. In this way, global context is also considered when routing capsules by introducing global latent variables that have direct influence on the objective function, and are updated discriminatively in accordance with the minimum description length (MDL) principle. We focus on enhancing capsule network properties, and perform a thorough evaluation on pose-aware tasks, observing improvements in performance over previous approaches whilst being more computationally efficient.

1 Introduction

Although capsule networks (CapsNets) have taken on a few different forms since their inception \cite{1,2,3,4}, they are generally built upon the following core assumptions and premises:

(i) Capturing equivariance w.r.t. viewpoints in neural activities, and invariance in the weights;
(ii) High-dimensional coincidences are effective feature detectors;
(iii) Viewpoint changes have nonlinear effects on pixels, but linear effects on object relationships;
(iv) Object parts belong to a single object, and each location contains at most a single object.

In theory, a perfect instantiation of the above premises could yield more sample efficient models, that leverage robust representations to better generalise to unseen cases. Unlike current methods, humans can extrapolate object appearance to novel viewpoints after a single observation. Evidence suggests that this is because we impose coordinate frames on objects \cite{5,6}. Capsules imitate this concept by representing neural activities as poses of objects w.r.t. a coordinate frame imposed by an observer, and attempt to disentangle salient features of objects into their composing parts. This is reminiscent of inverse graphics \cite{7}, but is not explicitly enforced in capsule formulations since the learned pose matrices are not constrained to interpretable geometric forms. Another argument for CapsNets, is one that views capsules as an extension to the very successful inductive biases already present in CNNs, by wiring in some additional complexity to deal with viewpoint changes. One of the desired effects is to align the learned representations with those perceptually consistent with humans, which would also make adversarial examples less effective \cite{8}. The additional complexity comes from replacing scalar neurons with vector valued neural activities, along with a high-dimensional coincidence filtering algorithm to detect capsule level features, known as capsule routing \cite{2,3}. This procedure is typically iterative, local and inefficient which has prompted further research on the topic \cite{9,10,11,12,13}.
1.1 Motivation & Contribution

Weaknesses of Capsule Networks. The memory bottleneck incurred by vector valued activations in addition to the iterative nature of capsule routing algorithms results in inefficient models. They are also prone to underfitting or overfitting if the number of routing iterations isn’t properly set \[2,3\]. To address the above weaknesses one may decide to naively replace the iterative nature of capsule routing with some faster alternative. However, to stay true to the premises of CapsNets, we argue that the four following points are of paramount importance for the research community to consider, when proposing algorithmic variants of CapsNets or capsule routing going forward:

(i) Whether viewpoint-invariance and affine transformation robustness properties are retained;
(ii) Changes in assumptions about part-object relationships are made explicit;
(iii) Whether capsules are still activated based on high-dimensional coincidences;
(iv) How do we handle the intrinsic uncertainty in assembling parts into objects.

Changes in the core assumptions of CapsNets aren’t always made clear in recent literature, but emerge incidentally via the proposed modifications. This leads to ambiguities regarding what qualifies as a capsule network, which can make comparisons between methods more difficult and hinder progress. In this paper, we focus on the core premises of capsule networks, and on enhancing their advantages over CNNs: viewpoint-invariance, and affine transformation robustness whilst being more efficient.

Contribution. Rather than performing local iterative routing between adjacent capsule layers which is inefficient, we propose an alternative global view based on representing the inherent uncertainty in part-object relationships, by approximating a posterior distribution over part-object connections. Sources of uncertainty in assembling objects via a composition of parts can arise from numerous sources, such as: (i) feature occlusions due to observed viewpoints; (ii) sensory noise in captured data; (iii) object symmetries for which poses may be ambiguous such as spherical objects/parts.

In our formulation, the local routing iterations are replaced with variational inference of part-object connections in a probabilistic capsule network, leading to a significant speedup (Figure 4). In this way, we encourage global context to be taken into account when routing information, by introducing global latent variables which have direct influence on the objective function, and are updated discriminatively in accordance with the minimum description length (MDL) principle \[14,15\]. Our experiments demonstrate that local iterative routing can be replaced by variational posterior inference of part-object connections in a global context setting, allowing the model to leverage the inherent uncertainty in assembling objects as a composition of parts to improve performance on pose-aware tasks.

2 Background: Capsule Networks

Capsules. A capsule \(c\) is a set of neurons \(c = \{a, M\}\). Each capsule is composed of either a vector \(m \in \mathbb{R}^d\) or matrix \(M \in \mathbb{R}^{\sqrt{d} \times \sqrt{d}}\) of neurons, and an activation probability \(a\). A single capsule is wired to represent a single entity, and its vector/matrix may learn to encode its pose w.r.t the coordinate frame imposed by an observer. The activation \(a\) simply represents an entity’s presence. A capsule network is composed of two or more capsule layers, with multiple capsules \(N\) in each layer. Capsule routing takes place between adjacent capsule layers, i.e. \(N_i\) capsules in a lower layer \(\ell_i\) are routed to \(N_j\) capsules in a higher layer \(\ell_j\), which can be seen as a form of cluster finding. Contextually, capsules in \(\ell_i\) are referred to as parts of objects (datapoints), and capsules in \(\ell_j\) are objects (clusters). Each part capsule uses its relationship to the viewer (pose), to posit a vote for what the pose of the object it is part of should be. To achieve this, part capsule poses \(M_i\) are multiplied with trainable viewpoint-invariant, affine transformation weight matrices:

\[
V_{j|i} = \{M_i \cdot W_{ij} \mid \forall c_i \in \ell_i , \forall c_j \in \ell_j\}, \quad W_{ij} \in \mathbb{R}^{\sqrt{d} \times \sqrt{d}}.
\]

(1)

where \(V_{j|i}\) denotes the \(i^{th}\) part capsule vote for the \(j^{th}\) object capsule pose, and \(W_{ij}\) are the weights.

Inducing Nonlinearity. Capsule poses \(M\) are not directly activated via nonlinear mappings but are compositions of affine/projective linear transformations, that increase in complexity as we traverse through the network. Nonlinearity is induced by the choice of routing algorithm \[2,3\], and the vote agreement measure used in calculating the activation probability \(a_j\) for each capsule \(c_j \in \ell_j\).
which are analogous to output channels in CNNs. Importantly, part capsules on the edge of feature maps vote for fewer objects than those in the middle (Figure 1), a fact which is very often overlooked in capsule research, leading to improper normalisation over objects and competition between capsules.

3 Uncertainty in Capsule Routing

Let $D$ denote a set of data given as $m$ pairs $\{x_i, y_i\}_{i=1}^m$, where $x_i \in \mathbb{R}^d$ denotes a datapoint, and $y_i \in \{1, \ldots, K\}$ its corresponding label. Let $z$ denote some latent variables associated with our observations $(x, y)$, that capture underlying structure in our data $D$ and help govern its distribution.

3.1 Defining Part-Object Connections

**Dense & Convolutional Voting.** In dense capsule voting, all part capsules are connected to all object capsules in the layer above. That is, each part capsule $c_i \in \ell_i$ votes $N_j$ times, therefore each object capsule $c_j \in \ell_j$ receives $N_i$ votes. The part-object connections are then $z_{\ell_i, \ell_j} \in \mathbb{R}^{N_i \times N_j}$. Alternatively, in a convolutional capsule layer with kernel size $k$ and stride $s$, the number of object capsules that each part capsule $c_i$ can vote for $N_{i \rightarrow j}$, is bounded above and below by

$$0 \leq N_{i \rightarrow j} \leq t_j \times \left \lceil \frac{k^2}{s} \right \rceil,$$

where $\lceil \cdot \rceil$ denotes the ceiling function, and $t_j$ denotes the number of output object capsule types, which are analogous to output channels in CNNs. Importantly, part capsules on the edge of feature maps vote for fewer objects than those in the middle (Figure 1), a fact which is very often overlooked in capsule research, leading to improper normalisation over objects and competition between capsules.

**Stochastic Variational Inference.** To represent our uncertainty about part-object relationships in a CapsNet, we look to approximate the (intractable) posterior distribution $p(z|D)$ over part-object connections $z$, with a chosen parameterised distribution $q_\phi(z|D) \approx p(z|D)$ via variational inference (VI). In general, $q_\phi(z|D)$ is optimised by updating the parameters $\phi$ such that the Kullback-Leibler (KL) divergence $D_{KL}(q_\phi(z|D) \| p(z|D))$ is minimised [15][16][17]. Next, we discuss and consider the inference of $q_\phi(z|D)$ under two main modelling paradigms: generative and discriminative.

**Generative.** Under generative frameworks, a set of local latent variables $z$ in models of the form $p_\theta(x, z) = p_\theta(x|z)p(z)$ are often employed, such as in the variational autoencoder (VAE) [13]. Specifically, latent variables $z = \{z_i\}_{i=1}^m$ are inferred for each $x = \{x_i\}_{i=1}^m$, and maximum likelihood (ML) or maximum a posteriori (MAP) inference is performed on global parameters. The model is fit by maximising the Evidence Lower Bound (ELBO) on the marginal log-likelihood

$$\log p(x) \geq \sum_{i=1}^m -D_{KL}(q_\phi(z_i|x_i) \| p(z_i)) + E_{q_\phi(z_i|x_i)}[\log p_\theta(x_i|z_i)] \triangleq \mathcal{L}_\text{local}(\phi, \theta).$$

**Discriminative.** Under the discriminative framework, global latent variables $z$ are often utilised and are shared among datapoints $\{x_i\}_{i=1}^m$, for instance when inferring the posterior on the weights of a neural network (NN) [15][17][19]. The bound is on the conditional marginal log-likelihood

$$\log p(y|x) \geq \sum_{i=1}^m -\frac{1}{m} D_{KL}(q_\phi(z) \| p(z)) + E_{q_\phi(z)}[\log p(y_i|x_i, z)] \triangleq \mathcal{L}_\text{global}(\phi).$$

To facilitate comparisons with the majority of research on CapsNets, we focus on the development and evaluation of our method in a discriminative setting. Formally, we are interested in estimating the conditional likelihood $p(y|x) = \prod_{i=1}^m p(y_i|x_i, z)$ using probabilistic capsule network models.

Figure 1: Our inference procedure in a given capsule layer (Left). Small example of part-object connections in convolutional voting for $k = 2$, drawn randomly from Dirichlet distributions (Right).
3.2 Posterior Inference of Part-Object Connections

Inference & Model Assumptions. Using stochastic VI tools, we intend to find the best approximation $q_\phi(z)$ that minimises $D_{KL}(q_\phi(z) \parallel p(z|D, W))$, where $z$ are global latent part-object connection variables, and $W$ are viewpoint-invariant transformation parameters, in a CapsNet with $L$ layers. We place a prior $p(z^{(\ell)})$ over each part capsule’s $c_i \in \ell$ connections to the objects they vote for $c_j \in \ell + 1$, and make the following factorised independence assumptions across capsule layers:

\[
  z^{(\ell)} = (z_1, z_2, \ldots, z_{N_{i-1}}) \sim p(z^{(\ell)}) \quad \forall c_i \in \ell, \quad p(z) = \prod_{\ell=1}^{L-1} \prod_{i=1}^{N_i} p(z^{(\ell)}_i). \tag{5}
\]

We then make a variational approximation $q_\phi(z_{\ell,\ell+1})$ to the posterior on part-object connection variables between adjacent capsule layers $\ell$ and $\ell+1$, for all capsule layers in the network. Our model’s likelihood $p(D|z, W)$, and mean-field variational family $q_\phi(z)$ are given by

\[
p(D|z, W) = \prod_{i=1}^{m} p(y_i|x_i, z, W), \quad q_\phi(z) = \prod_{\ell=1}^{L-1} \prod_{i=1}^{N_i} q_\phi(z^{(\ell)}_i). \tag{6}
\]

The model is defined hierarchically where the object capsules in $\ell$ are the parts of $\ell + 1$, and so forth.

Free Energy Objective. The model is fit end-to-end by maximising the following lower bound on the conditional marginal log-likelihood $\log p(y|x)$, which approximates its description length:

\[
  \mathcal{L}(y|x; \phi) \triangleq - \sum_{\ell=1}^{L-1} D_{KL}(q_\phi(z_{\ell,\ell+1}) \parallel p(z_{\ell})) + \sum_{i=1}^{m} E_{q_\phi(z)}[\log p(y_i|x_i, z, W)]. \tag{7}
\]

In the general case, we perform variational inference on the part-object connection latent variables $z$, and ML/MAP inference on $W$. We find this to work well enough in practice, whilst significantly reducing the number of parameters needed and assumptions made, which is especially important in CapsNets given that efficiency is a major concern. Nonetheless, for full posterior learning, we can make one further mean-field assumption by: $q_{\phi,\theta}(z, W) = q_\phi(z)q_\theta(W)$, where $q_\theta(W)$ is Gaussian and factorises similarly across layers, including any convolutional layers preceding the capsule layers.

3.3 Choosing Priors: Reflecting Part-Object Assumptions

Logistic-Normal. Recall from Eq. (2) that each part capsule $c_i$ votes for $N_{i-1}$ objects, we can introduce randomness in their part-object connections via a Gaussian-Softmax parameterisation:

\[
  \text{softmax}(z^{(i)})_j = \frac{\exp(z_j)}{\sum_k^{N_{i-1}} \exp(z_k)}, \quad z_j \sim \mathcal{N}(0, 1) \quad \text{for} \quad j = 1, 2, \ldots, N_{i-1}, \tag{8}
\]

with all components $z_j$ sampled independently from standard Gaussian priors. The approximate posterior then takes the form: $q_\phi(z^{(i)}) = \mathcal{N}(z^{(i)} | \mu^{(i)}, \sigma^{(i)}) \forall c_i \in \ell$. To obtain stochastic gradients of the lower bound w.r.t. the parameters $\phi$, we can parameterise samples from $q_\phi(z^{(i)})$ by: $z^{(i)} = f(\epsilon, \phi)$ where $f(\cdot)$ is differentiable and $\epsilon \sim \mathcal{N}(0, I)$, using the (local) reparameterisation trick \cite{18, 20}. These priors are generally attractive since reparameterising Gaussian samples is straightforward, and they have been shown to work well in other settings such as topic models \cite{21, 22}.

Dirichlet. Alternatively, multi-modality over categorical events is better captured by the Dirichlet distribution \cite{23}. We can also reduce the number of parameters as we only need to infer $\pi^{(i)}$ rather than $\{\mu^{(i)}, \sigma^{(i)}\}$ for each part capsule $c_i$, which is especially important in CapsNets, as explained in Section 1.1 since efficiency is a major concern. Our Dirichlet priors over $z$ are defined as

\[
  z^{(i)} = (z_1, z_2, \ldots, z_{N_{i-1}}) \sim \text{Dir}(\pi_0^{(i)}), \quad \pi_0^{(i)} = (\pi_1, \pi_2, \ldots, \pi_{N_{i-1}}), \tag{9}
\]

where $\pi_0^{(i)}$ are the prior concentration parameters for $c_i$, and the approximate posterior is then also Dirichlet distributed: $q_\phi(z^{(i)}) = \text{Dir}(\bar{\pi}^{(i)}) \forall c_i \in \ell_i$. In practice, we draw Dirichlet samples via independent standard Gamma distributions over each part-object connection:

\[
  \gamma_i^{(i)} = \{\gamma_j\}_{j=1}^{N_{i-1}}, \quad \gamma_j \sim \text{Gamma}(\pi_j, 1), \tag{10}
\]
algorithm 1 capsule layer with routing uncertainty. returns updated object capsules \( c_j = \{a_j, M_j\} \in \ell + 1 \), given part capsules \( c_i = \{a_i, M_i\} \in \ell \). performs ML/MAP inference of transformation weights \( W \), and variational inference of latent part-object connection variables \( z \).

1: function convcaps2d \((a_i, M_i)\) \( \triangleright \) input capsules from previous layer
2: initialise affine weights: \( W_{ij} \in \mathbb{R}^{\sqrt{2}\times\sqrt{2}} \) \( \forall i \neq j \)
3: set dirichlet priors: \( \pi_0(i) \in \mathbb{R}^{N_{\ell+1} \rightarrow \ell} \) \( \forall c_i \in \ell \)
4: \( V^j_{ji} \leftarrow \text{vote} (M_i, W_{ij}) \) \( \# \text{eqs.(1-3)} \) \( \triangleright \) capsules \( c_i \) vote for poses of capsules \( c_j \)
5: \( z_{\ell,\ell+1} \leftarrow \text{sample} q_0(\cdot) (a_i, \pi_0(i)) \) \( \# \text{eqs.(10-12)} \) \( \triangleright \) sample \( \mathbf{z}^{(i)} \) \( \forall c_i \) from approximate posterior
6: \( a_j, M_j \leftarrow \text{route} (z_{\ell,\ell+1}, V^j_{ji}) \) \( \# \text{eqs.(12,13)} \) \( \triangleright \) aggregate votes and activate capsules \( \mathbf{c}_j \)
7: return \( c_j = \{a_j, M_j\} \) \( \triangleright \) output capsules to next layer

Global Routing. Following from Eq. (1), part capsules \( c_i \in \ell \) cast votes \( V^j_{ji} \) for object capsules \( c_j \in \ell + 1 \), in all layers. During training we fit multivariate gaussians \( M_j \sim \mathcal{N}(\mu_j, \sigma_j) \), on each object’s \( d \) dimensional poses, and sample part-object connections from the approximate posterior:

\[
\mathbf{z}^{(i)} \sim q_0(\mathbf{z}_{\ell,\ell+1}) \quad \forall \mathbf{c}_i \in \ell, \quad 
\mu_j = \frac{\sum_i z^{(i)}_{\ell,\ell+1} V^j_{ji}}{\sum_i z^{(i)}_{\ell,\ell+1}}, \quad 
\sigma_j = \frac{\sum_i z^{(i)}_{\ell,\ell+1} (V^j_{ji} - \mu_j)^2}{\sum_i z^{(i)}_{\ell,\ell+1}}. \tag{12}
\]

The latent variables \( \mathbf{z}^{(i)} \) can act as soft assignments depending on our choice of prior, and one could interpret the training procedure as approximating the true posterior \( q^*(\mathbf{z}|D) \approx p(\mathbf{z}|D, W) \) over all layers under the global minimum description length objective in Eq. (7), rather than local (iterative) inference of \( \mathbf{z} \) in the E-step of EM routing [8] between all adjacent capsule layers. Alternatively, if we use the prior on \( \mathbf{z}^{(i)} \) be Beta distributed over each part-object connection, and omit the normalisation over objects, we can allow each part to route information to multiple objects at once. If one normalises over parts rather than objects, then routing closely resembles attention [29].

Agreement & Activation. To measure vote agreement for each object capsule, we compute the average negative entropy of its pose: \(-\mathcal{H}(M_j) \triangleq -d^{-1} \mathcal{H} [\mathcal{N}(\mu_j, \sigma_j)]\). Averaging yields a scale invariant measure w.r.t. the number of pose parameters \( d \). Agreement is weighted by the support for each object capsule, which is the amount of data received from its parts: \(-\mathcal{H}(M_j) \sum_i z^{(i)}_{\ell,\ell+1}\). Next, consider a Binomially distributed random variable \( S_j \sim \mathcal{B}(N_i, N_j^{-1}) \), describing the assignment of \( N_i \) parts to \( N_j \) objects with probability \( N^{-1} \). The expected amount of data each object receives in a given layer is then \( \mathbb{E}(S_j) \). We can use this value to normalise and offset the entropy term, which automatically scales logits according to the number of capsules in each layer:

\[
a_j \triangleq \frac{-\eta_j \mathcal{H}(M_j) - \mathbb{E}(S_j)}{\mathbb{E}(S_j)} = -\frac{\eta_j}{\mathbb{E}(S_j)} \mathcal{H}(M_j) - 1, \quad \eta_j \triangleq \sum_i z^{(i)}_{\ell,\ell+1}. \tag{13}
\]

\( a_j \) is then activated using the logistic function. In simple terms, if the uncertainty among votes is high — i.e. low negative entropy and poor agreement — assigning more data to capsule \( j \) decreases its activation. Alternatively, if the uncertainty among votes is low — i.e. high negative entropy and good agreement — assigning more data to capsule \( j \) increases its activation significantly. Activating capsules in this way simply encourages the model to meet the agreement and support activation criteria implicitly, but does not enforce them explicitly via learned \( \beta \) thresholds as in EM routing [8].
Although the model is partially Bayesian, we observe predictive entropies on out-of-distribution samples (AffNIST, FashionMNIST) to be consistent with model uncertainty representation as possible for all the benchmark results presented, i.e. flat Dirichlet: \( p(z^{(i)}) \sim \text{Dir}(1, \ldots, 1) \) \( \forall i \in \ell, \forall \ell \). These priors explicitly assume that each part capsule \( c_i \) is equally likely to belong to any object it votes for, with any level of certainty. Nonetheless, we conducted experiments to test sensitivity to the choice of prior, as presented in Figure 4. We observe tighter bounds for priors with central peaks, meaning that sampled part-object connections are closer to uniform over objects. Although tighter bounds are not always better \( [31] \), this suggests that parts prefer to spread their vote amongst multiple objects in CapsNets, which is reminiscent of Dropout’s effect on NN weights \( [32] \).

Capsule L2 Norm. Alternatively, we can activate capsules by computing the Frobenius norm of the mean votes for object poses \( ||\mu_j||_F \), then squashing it to a sensible \((0, 1)\) range \( \{2\} \). This encodes agreement in the norm of the poses and offers a considerable speedup at a performance cost.

<table>
<thead>
<tr>
<th>Method (Viewpoints)</th>
<th>Azimuth (Acc. %)</th>
<th>Elevation (Acc. %)</th>
<th># Param</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline CNN [3]</td>
<td>96.3</td>
<td>80.0</td>
<td>4.2M</td>
</tr>
<tr>
<td>CNN (AvgPool) [12]</td>
<td>91.5</td>
<td>78.2</td>
<td>0.15M</td>
</tr>
<tr>
<td>Our EM-Routing</td>
<td>96.29 ± 0.02</td>
<td>87.1 ± 0.42</td>
<td>0.17M</td>
</tr>
<tr>
<td>SR-Caps [12]</td>
<td>92.38</td>
<td>80.14</td>
<td>0.75M</td>
</tr>
<tr>
<td>STAR-Caps [11]</td>
<td>96.3</td>
<td>86.3</td>
<td>0.32M</td>
</tr>
<tr>
<td>EM-Routing [3]</td>
<td>96.3</td>
<td>86.5</td>
<td>0.31M</td>
</tr>
<tr>
<td>VB-Routing [13]</td>
<td>96.29</td>
<td>88.6</td>
<td>0.17M</td>
</tr>
<tr>
<td>{32, 8, 8, 8, 5}</td>
<td>96.3 ± 0.03</td>
<td>89.12 ± 0.7</td>
<td>0.06M</td>
</tr>
<tr>
<td>{64, 8, 16, 16, 5}</td>
<td>96.3 ± 0.02</td>
<td>91.06 ± 0.31</td>
<td>0.14M</td>
</tr>
<tr>
<td>{64, 16, 16, 16, 5}</td>
<td>96.29 ± 0.03</td>
<td>91.41 ± 0.46</td>
<td>0.22M</td>
</tr>
<tr>
<td>{128, 16, 32, 32, 5}</td>
<td>96.3 ± 0.02</td>
<td>91.85 ± 0.42</td>
<td>0.58M</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method (Viewpoints)</th>
<th>Azimuth (Acc. %)</th>
<th>Elevation (Acc. %)</th>
<th># Param</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline CNN [3]</td>
<td>96.3</td>
<td>80.0</td>
<td>4.2M</td>
</tr>
<tr>
<td>CNN (AvgPool) [12]</td>
<td>91.5</td>
<td>78.2</td>
<td>0.15M</td>
</tr>
<tr>
<td>Our EM-Routing</td>
<td>96.29 ± 0.02</td>
<td>87.1 ± 0.42</td>
<td>0.17M</td>
</tr>
<tr>
<td>SR-Caps [12]</td>
<td>92.38</td>
<td>80.14</td>
<td>0.75M</td>
</tr>
<tr>
<td>STAR-Caps [11]</td>
<td>96.3</td>
<td>86.3</td>
<td>0.32M</td>
</tr>
<tr>
<td>EM-Routing [3]</td>
<td>96.3</td>
<td>86.5</td>
<td>0.31M</td>
</tr>
<tr>
<td>VB-Routing [13]</td>
<td>96.29</td>
<td>88.6</td>
<td>0.17M</td>
</tr>
<tr>
<td>{32, 8, 8, 8, 5}</td>
<td>96.3 ± 0.03</td>
<td>89.12 ± 0.7</td>
<td>0.06M</td>
</tr>
<tr>
<td>{64, 8, 16, 16, 5}</td>
<td>96.3 ± 0.02</td>
<td>91.06 ± 0.31</td>
<td>0.14M</td>
</tr>
<tr>
<td>{64, 16, 16, 16, 5}</td>
<td>96.29 ± 0.03</td>
<td>91.41 ± 0.46</td>
<td>0.22M</td>
</tr>
<tr>
<td>{128, 16, 32, 32, 5}</td>
<td>96.3 ± 0.02</td>
<td>91.85 ± 0.42</td>
<td>0.58M</td>
</tr>
</tbody>
</table>

Capsule L2 Norm. Alternatively, we can activate capsules by computing the Frobenius norm of the mean votes for object poses \( ||\mu_j||_F \), then squashing it to a sensible \((0, 1)\) range \( \{2\} \). This encodes agreement in the norm of the poses and offers a considerable speedup at a performance cost.

### Experiments

In this study, we focus on demonstrating that our method enhances capsule properties and outperforms previous approaches on challenging pose-aware tasks used in CapsNet literature (Sections 4.1, 4.2 and 4.4), whilst being more computationally efficient (see Figure 4 for runtime comparisons) \[7\].

#### Network Architecture.

To ensure fair and direct comparisons with previous work, we use identical CapsNets to EM routing \[3\], A single \( 5 \times 5 \) Conv layer with \( f_0 \) filters and stride 2 precedes four capsule layers. The PrimaryCaps layer transforms \( f_0 \) feature maps into \( t_1 \) capsule types, each having \( H \times W \) number of capsules with \( 4 \times 4 \) poses. Next, two \( 3 \times 3 \) ConvCaps layers with \( t_2 \) and \( t_3 \) output capsule types, using strides 2 and 1. The last ConvCaps layer outputs \( t_4 \) class capsules, and shares weights across spatial dimensions \[3\]. Let \( \{f_0, t_1, t_2, t_3, t_4\} \) denote the complete architecture. In all experiments, we use Adam \[30\] with default parameters and a batch size of 128 for training.

#### Priors.

To show our method works well in the general case, we set the priors to be as uninformative as possible for all the benchmark results presented, i.e. flat Dirichlet: \( p(z^{(i)}) \sim \text{Dir}(1, \ldots, 1) \) \( \forall i \in \ell, \forall \ell \). These priors explicitly assume that each part capsule \( c_i \) is equally likely to belong to any object it votes for, with any level of certainty. Nonetheless, we conducted experiments to test sensitivity to the choice of prior, as presented in Figure 4. We observe tighter bounds for priors with central peaks, meaning that sampled part-object connections are closer to uniform over objects. Although tighter bounds are not always better \( [31] \), this suggests that parts prefer to spread their vote amongst multiple objects in CapsNets, which is reminiscent of Dropout’s effect on NN weights \( [32] \).

#### Inference.

In all benchmark results, we perform a deterministic inference at test time without sampling \( z \), by using the posterior means \( z^* = E[q_{\phi, \theta}(z_{\ell, t+1})] \) \( \forall \ell \), to compute predictions \( y^* = \text{arg max}_y p(y|z^*, W) \). Alternatively, we can draw \( T \) Monte Carlo samples of part-object connections from the approximate posterior, and calculate the predictive entropy:

\[
H(\tilde{y}|x, z, W) = - \sum_{k=1}^{K} \tilde{y}_k \log \tilde{y}_k, \quad \tilde{y} \approx \frac{1}{T} \sum_{t=1}^{T} p(y|x, z^t, W), \quad z^t \sim q_{\phi}(z|D). \quad (14)
\]

Under full posterior learning: \( q_{\phi, \theta}(z, W) \), the pose transformation matrices \( W \) are also sampled. Although the model is partially Bayesian, we observe predictive entropies on out-of-distribution dataset samples (AffNIST, FashionMNIST) to be consistent with model uncertainty representation as shown in Figure 3. We also observe entropic predictions on more challenging SmallNORB viewpoints as we vary azimuth, whilst holding the lowest/highest elevation viewpoints fixed (see Figure 4).
4.1 Generalisation to Novel Viewpoints

Viewpoint-Invariance. SmallNORB [33] consists of grey-level stereo 96 × 96 images of 5 objects: each given at 18 different azimuths (0-340), 9 elevations and 6 lighting conditions, with 24,300 training and test set examples. As in [3], we standardise the images and resize them to 48 × 48. During training we take 32 × 32 random crops, and centre crops at test time. We train on training set images with azimuths: \( A_{\text{train}} = \{300, 320, 340, 0, 20, 40\} \), denoted as familiar viewpoints, and test on test set images containing novel azimuths: \( A_{\text{test}} = \{60, 80, \ldots, 280\} \). Similarly, for the elevation viewpoints we train on \( E_{\text{train}} = \{30, 35, 40\} \), and test on \( E_{\text{test}} = \{45, 50, \ldots, 70\} \). As reported in Table 1 we observed notable performance improvements in viewpoint-invariance over previous CapsNets, and significant improvements over CNNs. Additional results on the standard SmallNORB train/test splits are found in Table 2.

4.2 Affine Transformation Robustness

Table 3: MNIST to AffNIST generalisation error (%). (†) unsupervised learning.

<table>
<thead>
<tr>
<th>Method</th>
<th>MNIST Error (%)</th>
<th>AffNIST Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline CNN</td>
<td>0.8</td>
<td>14.1</td>
</tr>
<tr>
<td>BCN [34]</td>
<td>2.5</td>
<td>8.4</td>
</tr>
<tr>
<td>Dynamic [6]</td>
<td>0.77</td>
<td>21</td>
</tr>
<tr>
<td>Sparse Caps [5]</td>
<td>1.0†</td>
<td>9.9</td>
</tr>
<tr>
<td>SCAE [9]†</td>
<td>1.5</td>
<td>7.79</td>
</tr>
<tr>
<td>EM-Routing</td>
<td>0.8</td>
<td>6.9</td>
</tr>
<tr>
<td>All Caps [22]</td>
<td>0.77</td>
<td>6.79</td>
</tr>
<tr>
<td>{32, 8, 8, 10}</td>
<td>0.8±0.01</td>
<td>5.02±0.28</td>
</tr>
<tr>
<td>{64, 16, 16, 10}</td>
<td>0.79±0.02</td>
<td>4.17±0.3</td>
</tr>
<tr>
<td>{64, 16, 16, 10}</td>
<td>0.78±0.02</td>
<td>3.88±0.34</td>
</tr>
<tr>
<td>{128, 16, 32, 32}</td>
<td>0.8±0.02</td>
<td>3.46±0.19</td>
</tr>
<tr>
<td>{128, 16, 32, 32}</td>
<td>0.28±0.01</td>
<td>2.31±0.03</td>
</tr>
</tbody>
</table>

Out-of-Distribution Generalisation. In this study we demonstrate our model’s robustness to affine transformations using the AffNIST dataset. AffNIST consists of MNIST images which have been uniquely transformed by 32 random affine transformations per image. Training is performed on the MNIST training set, and the test generalisation performance is measured on the AffNIST test set containing 320,000 examples. AffNIST images are 40 × 40 so for training we pad MNIST images, randomly placing the digits on 40 × 40 black backgrounds as in works we compare to [2][13]. Our models were never trained on AffNIST, and no further data augmentation was used. As shown in Table 3 we observed performance improvements over previous CapsNets, and significantly so over CNNs. Increasing the number of capsules used in our method also leads to better generalisation performance.
predictive $H(\hat{y} | x, z, W)$

MNIST (Train)
MNIST (Test)

0.75 1.00 1.25 1.50 1.75 2.00 2.25

predictive $H(\hat{y} | x, z, W)$

FashionMNIST (Test)

0.8 0.9 1.0 1.1 1.2 1.3 1.4 1.5 1.6

predictive $H(\hat{y} | x, z, W)$

NORB (Atrain)
NORB (Atest)

0 2 4 6 8
iteration (× 10^4)

− 5
− 4
− 3
− 2
− 1
ELBO
×102 Dirichlet Priors ($\pi_0$)

0.5
1
5
10
15
103 104 ... 13.2 16.1 11.0 9.8

36.4
22.5
11.3 6.8 5.6 5.4 4.1

1

L2 Norm
2Entropy
Runtime Comparisons

training
inference

Table 5: Comparing MultiMNIST test error and exact match ratio (MR) error, (†) dagger denotes results from using DiverseMultiMNIST.

<table>
<thead>
<tr>
<th>Method</th>
<th>MultiMNIST (#Param)</th>
<th># Param</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Test)</td>
<td>Error (%)</td>
<td>MR (%)</td>
</tr>
<tr>
<td>Baseline CNN</td>
<td>8.01</td>
<td>24.6M</td>
</tr>
<tr>
<td>Baseline CNN</td>
<td>-</td>
<td>15.2</td>
</tr>
<tr>
<td>Dynamic</td>
<td>5.2</td>
<td>-</td>
</tr>
<tr>
<td>IDP-Attention</td>
<td>10</td>
<td>8.83</td>
</tr>
<tr>
<td>All-Caps</td>
<td>4.51</td>
<td>-</td>
</tr>
<tr>
<td>{64, 8, 16, 16, 10}</td>
<td>3.3 ± 0.07</td>
<td>7.2 ± 0.21</td>
</tr>
<tr>
<td>{128, 16, 16, 16, 10}</td>
<td>2.4 ± 0.11</td>
<td>4.7 ± 0.18</td>
</tr>
<tr>
<td>{128, 16, 32, 32, 10}</td>
<td>1.8 ± 0.09</td>
<td>3.4 ± 0.17</td>
</tr>
</tbody>
</table>

Overlapping Digits. In this study we empirically demonstrate that our method is resilient under feature occlusions (which is a source of uncertainty). To that end, we replicated the experiment setup in [38], and trained our shallow models on the MultiMNIST dataset by generating occluded digit pairs on the fly. Digit pairs are formed by shifting each MNIST digit by up to 4 pixels in each direction, then adding them together. No further data augmentation was used. Our models were trained/validated on 60M overlapping digit pairs, and tested on 10M. Table 5 reports both lower test error and exact match ratio (MR) error compared to previous work. See Figure 5 for illustrations.
5 Related Work & Conclusion

Variational Inference. Our work lies at the intersection of CapsNets and variational Bayesian learning. Variational Inference (VI) has its roots in statistical physics [40, 41], leading to seminal work in the early-1990s [15] which offered a MDL [14] perspective on VI in NNs. VI was later formalised more generally in a series of important works [42, 43, 44, 16]. More recently, practical strategies for calculating biased/unbiased Monte Carlo gradients of variational objectives in deep NNs have been proposed [17, 19], which are complemented by ideas from deep generative modelling such as the reparameterisation trick [18, 24]. NNs with Dropout [32] have also been interpreted as being approximately Bayesian [45, 46], and are widely used to estimate uncertainty [47, 48, 49, 50].

Capsule Networks. Initial work on capsules began with the transforming autoencoder [1]. Other successful variants have since then been proposed, notably: Dynamic routing [2], EM routing [3], and stacked capsule autoencoders (SCAE) [4], all of which achieved state-of-the-art performance in pose-aware tasks. Much follow-up work focuses on algorithmic variants of local routing or in scaling up CapsNets: VB routing [13], KDE [9], Spectral [51], Subspace-Caps [52, 53]. Other interesting works improve on the equivariance properties of CapsNets directly using Group theory [35, 54], which is on the contrary to our approach, as we do not impose any specific equivariance restrictions into the model. Geometric approaches have also been explored by [55, 56, 57], extending CapsNets to work with point clouds and in 3D. Related work on probabilistic interpretations of CapsNets is limited, with the notable exception of [58] which considers a fully generative perspective of SCAE [4] that is unsupervised, in contrast to the discriminative probabilistic model with capsule structure presented in this paper. Our work builds primarily on both local EM/VB routing [3, 13]—to which we provide a global alternative using VI tools—and other recent non-iterative routing methods: Attention routing [59], STAR-Caps [11], Self-Routing [12], and Inverted Dot-Product routing [10]. Modifications in some of the latter methods have led to ambiguities regarding what qualifies as a CapsNet, as opposed to CNNs with attention. As explained in Section 1.1, this occurs whenever the fundamental premises of CapsNets are implicitly or explicitly altered, and their properties are not carefully verified or retained. With that in mind, we demonstrate empirically that our proposed end-to-end probabilistic approach leads to performance enhancements in benchmark pose-aware tasks commonly used in CapsNet literature, whilst being more computationally efficient.

5.1 Conclusion

In this paper we propose to replace inefficient local iterative routing with variational inference of a posterior on part-object connections in a probabilistic capsule network, leading to a significant speedup (Figure 4). In this way, we encourage global context to be taken into account when routing information, by introducing global latent variables which have direct influence on the objective function, and are updated discriminatively in accordance with the minimum description length principle. To facilitate comparisons, we developed our method in a discriminative setting, and performed a thorough evaluation on pose-aware tasks, demonstrating enhanced capsule properties over previous iterative and non-iterative routing methods. We believe further exploration of CapsNets as deep latent variable models (DLVMs) [24, 60, 61], to be a promising future research direction.
Broader Impact

With the advent of Deep Learning, the computational requirements in the field have increased significantly due to the ever increasing scale of our models. The environmental impact of training or deploying such models is therefore at an all time high. This raises concerns regarding the sustainability of our current practices, as the technologies we help develop are slowly integrated into all areas of society. Although it is important to continue on this path of discovery, we feel that an important shift towards efficiency is sorely needed. Concretely, the development of smaller scale models which are more robust and sample efficient, could significantly reduce the environmental impact of our technology with small sacrifices in performance. In general, we believe this can be achieved by introducing richer inductive priors into our models, which in turn require fewer examples to learn from, i.e. leading to increased sample efficiency. With that in mind, Capsule Networks have previously shown to possess superior generalisation properties than conventional CNNs in certain tasks, and in our work we enhance these properties further whilst being more computationally efficient than previous iterative routing methods. We also demonstrated competitive performances on sample efficiency tasks, which have broad applicability to limited data domains such as medical. When these properties are enhanced even further, they have the potential to make a significant positive impact on our societies by increasing the sustainability and efficiency of our machine learning models.

Acknowledgments and Disclosure of Funding

We would like to gratefully acknowledge the support of NVIDIA Corporation with the donation of GPUs used for this research. We also thank Francesco Caliva and Lewis Smith for fruitful discussions.

References


