Abstract

Capsule Networks are a recently proposed alternative for constructing Neural Networks, and early indications suggest that they can provide greater generalisation capacity using fewer parameters. In capsule networks scalar neurons are replaced with capsule vectors or matrices, whose entries represent different properties of objects. The relationships between objects and its parts are learned via trainable viewpoint-invariant transformation matrices, and the presence of a given object is decided by the level of agreement among votes from its parts. This interaction occurs between capsule layers and is a process called routing-by-agreement. Although promising, capsule networks remain underexplored by the community, and in this paper we present a new capsule routing algorithm based on Variational Bayes for a mixture of transforming gaussians. Our Bayesian approach addresses some of the inherent weaknesses of EM routing such as the ‘variance collapse’ by modelling uncertainty over the capsule parameters in addition to the routing assignment posterior probabilities. We test our method on public domain datasets and outperform the state-of-the-art performance on smallNORB using ≃50% less capsules.

1 Introduction

Capsule Networks (CapsNets) are a recently proposed method of learning part-whole relationships between observed entities in data by using groups of neurons known as capsules. These entities could be anything that possesses a consistent underlying structure across different viewpoints. Capsules attempt to encode their intrinsic viewpoint-invariant properties, and learn to adjust some instantiation parameters as the entity varies across its appearance manifold [7]. CapsNets have shown to outperform standard Convolutional Neural Networks (CNNs) in specific tasks involving shape recognition and overlapping digit segmentation using fewer parameters. These tasks are notoriously difficult for standard CNNs as they struggle to exploit the frame of reference humans impose on objects, and thus often fail at generalising knowledge to new viewpoints. Although this drawback can usually be mitigated by data augmentation during training, it does not address the underlying issue directly. Nonetheless, Deep CNNs have proven to perform remarkably well in practice, achieving state-of-the-art performance in most vision tasks. In CNNs, the convolution operator combined with sparse weight sharing provides the useful property of equivariance under translation, and enables efficient knowledge transfer across spatial locations. CapsNets retain these benefits and only do away with max/average pooling operations in favour of learning more robust representations for disentangling factors of variation with routing-by-agreement [15, 8]. Capsule routing works in general because it can intuitively be seen as a form of coincidence filtering, since getting good agreement in high
Capsule Networks  CapsNets are composed of at least one layer of capsules in which capsules \( i \) from a lower layer \( \mathcal{L}_i \) are routed to capsules \( j \) in a higher layer \( \mathcal{L}_j \). Each layer can contain multiple lower capsules, each of which has a pose matrix \( M_i \in \mathbb{R}^{4 \times 4} \) of instantiation parameters and activation probability \( a_i \). The pose matrix may learn to encode the relationship of an entity to the viewer, and the activation probability \( a_i \) represents its presence. Each lower level capsule then uses its pose matrix \( M_i \) to posit a vote for what the pose of a higher level capsule should be, by multiplying it with a trainable viewpoint-invariant transformation weight matrix

\[
V_{ji} = W_{ji} M_i,
\]

where \( V_{ji} \) denotes the vote coming from capsules \( i \) to capsule \( j \) and \( W_{ji} \in \mathbb{R}^{4 \times 4} \) is the viewpoint-invariant transformation matrix. To compute the pose matrix \( M_j \) of any higher level capsule \( j \) we can simply take a weighted mean of the votes it received from capsules in \( \mathcal{L}_i \) as \( M_j = 1/R_j \sum_i V_{ji} R_{ij} \), where \( R_{ij} \) represents the posterior probabilities of each capsule \( j \) having seen capsules \( i \) in EM routing \([8]\), with \( R_j = \sum_i R_{ij} \). These routing coefficients are tuned via a routing-by-agreement variant of the EM algorithm for Gaussian Mixtures and thus can be interpreted as the responsibility that each capsule \( j \)'s gaussian takes for explaining capsules \( i \). The routing coefficients are updated by measuring the agreement between \( V_{ji} \) and \( M_j \), which in Dynamic Routing \([15]\) is simply the scalar product between capsule vectors and can be trivially extended to matrices with \( \| M_j - V_{ji} \|_F \).

Contribution  We propose to learn a mixture of transforming gaussians \([8]\) between adjacent capsule layers by Variational Bayes. Unlike previous approaches such as Dynamic or EM routing, our probabilistic approach provides several advantages over previous routing methods, including reduction of singularities that lead to overfitting in EM, more stable and flexible control over the complexity of the capsules by tuning the priors to induce sparsity, and reducing the ‘variance collapse’ problem inherent to MLE derivatives \([11]\). Furthermore, we provide some insight into capsule network training for practitioners including weight initialisation and normalisation schemes that mitigate the dead capsule problem at the start of training \([8]\). Lastly, we show that it is straightforward to transform our capsule network into a Capsule-VAE by sampling the latent code from capsule’s approximate parameter posterior. With our approach we outperform the state-of-the-art on the smallNORB dataset using \( \approx 50\% \) less capsules than the original \([8]\).

2 Variational Bayes Capsule Routing

Next we briefly outline some necessary background on Variational Inference (VI) for the reader’s benefit, prior to contextualising some of these well established ideas with our routing algorithm.

2.1 Variational Inference

The Evidence Lower Bound  Let \( x \) denote the observed data, \( z \) denote latent variables associated with \( x \), and let \( \theta \) represent some model parameters. Typically we’d like to infer the unknown latent variables, by evaluating the conditional \( p(z|x, \theta) = \frac{p(x,z|\theta)}{\int p(x,z|\theta) \, dz} \), which is the posterior on \( z \). However, this distribution cannot be computed for most complex models due to the intractability of the integral in the denominator. VI provides an elegant solution to posterior inference by posing it as an optimisation problem. We approximate the posterior \( p(z|x, \theta) \) by choosing a variational distribution over the latent variables \( q_\phi(z) \) from a tractable family, with its own variational parameters \( \phi \). We can measure the quality of our approximation via the Kullback-Leibler (KL) \( \text{KL}(q_\phi(z) \mid\mid p(z|x, \theta)) \) divergence between the two distributions which can be minimised via the variational parameters \( \phi \)

\[
\hat{\phi} = \arg \min_\phi \mathbb{E}_{q_\phi(z)} \left[ \log q_\phi(z) - \log p(z|x, \theta) \right].
\]
However, since \( p(z|x, \theta) \) is unknown we cannot minimise the KL directly, so instead we maximise the variational lower bound (ELBO) on the log marginal likelihood
\[
\log p(x|\theta) = \text{KL}(q_\phi(z) \parallel p(z|x, \theta)) + \mathcal{L}_{\text{ELBO}}(q_\phi(z)).
\]
where the ELBO can be derived using Jensen’s inequality \( \log(E[X]) \geq E[\log(X)] \) applied to \( \log p(x|\theta) \) giving
\[
\log p(x|\theta) \geq \mathcal{L}_{\text{ELBO}}(q_\phi(z)) = E_{q_\phi(z)}[\log p(x, z|\theta)] - E_{q_\phi(z)}[\log q_\phi(z)].
\]
Here we use the joint \( \log p(x, z|\theta) \) which is tractable, rather than the unknown posterior \( \log p(z|x, \theta) \).

Note that from the product rule of probability we have \( p(x, z|\theta) = p(x|z, \theta)p(z|\theta) \) and given that the log marginal likelihood of the observed data \( \log p(x|\theta) \) is always negative and is independent of \( q_\phi(z) \), maximising the ELBO is therefore equivalent to minimising the KL divergence.

\[ \textbf{Mean Field} \] A popular way of performing VI is to posit a factorised form of the approximating family of distributions \( q_\phi(z) \), such that each variable is assumed to be independent
\[
p(z|x, \theta) \approx q_\phi(z) = \prod_{i=1}^{N} q_{\phi_i}(z_i), \quad \sum_{z_i} q_{\phi_i}(z_i) = 1.
\]
Recall that the log marginal is given by \( \log p(x|\theta) = \log \sum_{z} p(x, z|\theta) \) and therefore the factorised objective can be summarised as
\[
\arg \max_{q_{\phi_i}(z_i) \in q_\phi(z)} \sum_{i=1}^{N} E_{q_{\phi_i}(z_i)}[\log p(x, z_i|\theta)] - E_{q_{\phi_i}(z_i)}[\log q_{\phi_i}(z_i)].
\]

2.2 Variational Bayes for a Mixture of Transforming Gaussians

The structure and rationale behind Capsules naturally lends itself to clustering logic. This is reflected in the fact that any higher level capsule \( j \) (cluster) is composed of, and receives votes from, many lower level capsules \( i \) (data points) within its receptive field. However, capsules do differ from regular clustering in a drastic way, as every cluster has its own viewpoint-invariant transformation matrix \( M_j \) with which it transforms its data points. Therefore, each cluster sees a different view of the data, and the algorithm converges much faster since it’s easier to break symmetry compared to simply initialising the gaussians with different means \([8]\). Next we propose our algorithm borrowing some ideas from \([1]\), and begin by picking up from our discussion in section 1.

\[ \textbf{Proposed Method} \] Let \( v_{j|i} \in \mathbb{R}^D \) denote a vectorised version of the 4x4 votes \( V_{j|i} \) matrix, and let \( \mu_j \in \mathbb{R}^D \) denote a vectorised version of capsule \( j \)'s 4x4 pose matrix \( M_j \), where \( D = 16 \). Assuming independence, consider the log likelihood function maximised in a Gaussian Mixture Model (GMM), applied to routing capsules \( i \) from a lower layer to capsules \( j \) in a higher layer
\[
\log p(v|\pi, \mu, \Lambda) = \arg \max_{\pi, \mu, \Lambda} \sum_{i \in \mathcal{L}_i} \log \left[ \sum_{j \in \mathcal{L}_j} \pi_j \text{Norm}(v_{j|i}|\mu_j, \Lambda_j^{-1}) \right].
\]
In EM routing, point estimates of the parameters \( \mu_j \) and \( \text{diag}(\Lambda_j) \) are computed in the M-step, and the routing probabilities \( R_{ij} \) are evaluated in the E-step. The mixing coefficients \( \pi_j \) however, are replaced with activations \( a_j \) which represent the probability of cluster \( j \) being switched on, and are computed by a shifting logistic non-linearity. The \( a_j \)'s play the role of the mixing proportions but \( \sum_j a_j \neq 1 \). Recall from section 1 that the votes play the roles of the data points and are computed as \( V_{j|i} = W_{ij}M_i \) using different transformation matrices \( W_{ij} \) for each capsule \( j \), so it follows that \( V_{j|i} \neq V_{j|i} \).

In order to model uncertainty over the capsule parameters in our algorithm, we place conjugate priors over \( \pi, \mu \) and \( \Lambda \). Our model’s generative process for any lower layer capsule \( i \)'s vectorised pose \( \mu_{i|\mathcal{L}_i} \) can be derived from the following
\[
\begin{align*}
v_{j|i} | z_i = j & \sim \text{Norm}(\mu_j, \Lambda_j^{-1}) \\
z_i & \sim \text{Cat}(z_i|\pi) \\
\pi | \alpha_0 & \sim \text{Dir}(\alpha_0) \\
\mu_j | \mu_0, \kappa_0, \Lambda_j & \sim \text{Norm}(\mu_0, (\kappa_0 \Lambda_j)^{-1}) \\
\Lambda_j | \Psi_0, \nu_0 & \sim \text{Wh}(\Psi_0, \nu_0),
\end{align*}
\]
then \( \mu_i \) can be retrieved by simply inverting the vectorised vote transformation \( \mu_i = w_i^{-1} v_{ji} \). The joint distribution of the model factorises as \( p(v, z, \tau, \mu, \Lambda) = p(v|z, \mu, \Lambda)p(z|\tau)p(\tau)p(\pi)p(\mu|\Lambda)p(\Lambda) \), where the latent variables \( z \) are a collection of \( L_i \) one-hot vectors denoting the cluster assignments of each of the lower capsules votes \( v_{ji} \), to their corresponding higher capsules’ gaussians. Following from the VI discussion in section 2.1, we approximate the posterior \( p(z, \tau, \mu, \Lambda|v) \propto p(v|z, \mu, \Lambda)p(z|\tau)p(\tau)p(\mu|\Lambda)p(\Lambda) \), with a factorised variational distribution

\[
p(z, \tau, \mu, \Lambda|v) \approx q(\tau)q(\pi)\prod_{j \in L_i} q(\mu_j, \Lambda_j),
\]

where the conjugate priors factor in the following standard form as in Bayesian Gaussian Mixtures

\[
p(\tau)p(\mu, \Lambda) = \text{Dir}(\tau|\alpha_0)\prod_{j \in L_i} \text{Norm}(\mu_j|m_0, (\kappa_0\Lambda_j)^{-1})\text{Wi}(\Lambda_j|\Psi_0, \nu_0).
\]

To parameterise diagonal precisions in practice, we simply let \( \lambda_j \in \mathbb{R}^D \) represent the diagonal entries of \( \text{diag}(\Lambda_j) \), and replace the Gaussian-Wishart prior with Gaussian-Gamma priors over each diagonal entry \( \lambda_j^d \)

\[
p(\mu|\lambda) = \prod_{j \in L_i} \prod_{d=1}^D \text{Norm}(\mu_j^d|m_0, (\kappa_0\lambda_j^d)^{-1})\text{Gam}(\lambda_j^d|\kappa_0, \nu_0).
\]

In our algorithm, we iterate between optimising capsule \( j \) parameter distributions with the responsibilities over capsules \( i \) fixed, and evaluating the new expected responsibilities given the current distributions over the capsule \( j \) parameters. See Algorithm 1 for the posterior update equations, which assume the same functional form as the priors through conjugacy, and refer to 11 for a more detailed explanation of this process.

**Capsule Agreement** We propose to measure agreement between the votes from lower capsules \( i \) using the weighted differential entropy of the higher capsule \( j \)'s gaussian. Firstly, the differential entropy of a multivariate gaussian distributed random variable \( x \) is by definition

\[
\mathbb{H}[x] \triangleq -\int_{-\infty}^{+\infty} f(x) \ln f(x) dx = -\mathbb{E}[\ln \text{Norm}(x|\mu, \Sigma)]
\]

\[
= \frac{1}{2} \ln \det(\Sigma) + \frac{D}{2} (1 + \ln(2\pi)) \propto \ln \det(\Sigma).
\]

Now let \( f(x) = q^*(\mu_j, \Lambda_j) = \text{Norm}(\mu_j|m_j, (\kappa_j\Lambda_j)^{-1})\text{Wi}(\Lambda_j|\Psi_j, \nu_j) \), where \( m_j, \kappa_j, \Psi_j \) and \( \nu_j \) are the updated prior parameters as in Algorithm 1, then we have

\[
\mathbb{H}[q^*(\mu_j, \Lambda_j)] \propto \mathbb{E}[\ln \det(\Lambda_j)] = \sum_{i=0}^{D-1} \psi\left(\frac{\nu_j - 1}{2}\right) + D \ln 2 + \ln \det(\Psi_j) \propto \ln \det(\Psi_j),
\]

where we use the expected log determinant of the precision matrix \( \Lambda_j \) to indirectly measure the differential entropy of capsule \( j \)'s distribution, up to constant factors. Intuitively, the determinant of the precision matrix measures the concentration of data points across the volume defined by the matrix. The higher the concentration the higher the agreement is among votes for capsule \( j \). Note that the updated parameters \( \mu_j, \kappa_j, \Psi_j \) and \( \nu_j \) all have a dependency on routing coefficients \( r_j = \sum_i r_{ij} \odot a_i \), which represent the amount of data assigned to capsule \( j \), weighted by the previous capsule layer activations. From the perspective of any capsule \( j \)'s gaussian, the previous layer activations \( a_i \) simply determine how important each data point is. To compute capsule \( j \)'s activation probability \( a_j \), we pass in its the expected mixing coefficient and the expected log determinant of its precision matrix, as a weighted measure of agreement, through a logistic non-linearity

\[
a_j = \text{sigmoid}\left(\beta_u - (\beta_u + \mathbb{E}[\ln \pi_j] + \mathbb{E}[\ln \det(\Lambda_j)]) \odot r_j\right),
\]

where \( \beta_u \) and \( \beta_u \) are learnable parameters that represent the description lengths of two different ways of coding the activated lower-level capsules assigned to \( j \). Unlike EM or Dynamic routing, we only activate the capsules after the routing iterations have been performed. This allows us to
We can easily transform our CapsNet into a Variational Autoencoder (VAE) [9] by sampling \( \epsilon \) where
\[
\text{weighted information and entropy principles, wherein one can imagine two separate low probability}
\]
add in the expected mixing coefficients as a weight on the differential entropy of each capsule \( j \)'s gaussian, encouraging a trade-off between activating the capsule with the largest amount of votes and the measure of how concentrated they are. This decision is in part motivated by context-dependent weighted information and entropy principles, wherein one can imagine two separate low probability events incurring equally high surprisal, but the informative value of one of them is contextually higher [5].

### 2.3 Capsule-VAE

We can easily transform our CapsNet into a Variational Autoencoder (VAE) [9] by sampling from the approximate posterior on the capsule parameters. We do so by saving the posterior parameters of the approximating distribution at the end of the routing procedure of the final layer, and output the capsule means and precisions as latent code. Recall that, for routing, the approximate posterior on the mean and precision of any capsule \( j \) is a Gaussian-Wishart

\[
q^*(\mu_j, \Lambda_j) = \text{Norm}(\mu_j | m_j, (\kappa_j \Lambda_j)^{-1})\text{Wi}(\Lambda_j | \Psi_j, \nu_j),
\]

and we can sample from it as

\[
\Lambda_j \mid \Psi_j, \nu_j \sim \text{Wi}(\Psi_j, \nu_j)
\]

\[
\mu_j \mid \Lambda_j, m_j, \kappa_j \sim \text{Norm}(m_j, (\kappa_j \Lambda_j)^{-1}).
\]

It is straightforward to condition the sample on the current label during training, and to make the whole process differentiable, we invert and square root the precision before applying the reparameterisation trick

\[
z \sim \text{Norm}(\mu_j, \sigma_j) = g_{\mu_j, \sigma_j}(\epsilon) = \mu_j + \epsilon \odot \sigma_j
\]

where \( \epsilon \sim \text{Norm}(0, I) \), and \( \sigma_j \) can be the square root of either the diagonal entries of the inverted full precision matrix, or the inverted precisions \( \Lambda_j^{-1} \) of the diagonal Gaussian-Gamma parameterisation.
denoted in Eq. (11). Capsule-VAE’s are interesting models since the output latent code is composed of capsule instantiation parameters, and we know from [15] that each capsule dimension learns to encode different variations of object properties that we can visualise/tweak. We leave further analysis and exploration of these ideas to future work.

### 3 Related Work

Capsules were first introduced in [7], wherein the logic of encoding instantiation parameters was established in a transforming autoencoder. More recently, further work on capsules [15] garnered some attention achieving state-of-the-art performance on MNIST, with a shallow Capsule Network using a Dynamic routing-by-agreement algorithm. Shortly after, a new Expectation Maximisation routing algorithm was proposed in [8], and capsule vectors were replaced by matrices to reduce the number of parameters. State-of-the-art performance was achieved on the smallNORB dataset using a relatively small CapsNet. Group Equivariant Capsule Networks were proposed in [10], leveraging ideas from group theory to guarantee equivariance and invariance properties. In [19] a new routing algorithm based on kernel density estimation is proposed, providing a speed up compared to EM routing among other benefits. Capsules have also been extended to action recognition in videos by [3], where the propose to average the votes before routing them. This speeds up the routing procedure but somewhat goes against capsule philosophy as being the replacement for pooling. Work in [18] proposes learning groups of capsule subspaces and project embedded features onto these subspaces. Despite these interesting works on capsules among others since their revival, the original state-of-the-art benchmarks are yet to be beaten. Our paper builds on previous work on capsules, and seeks to bring the benefits of Variational Bayesian learning into capsule networks.

### 4 Experiments

#### Capsule Network Architecture

Our CapsNet comprises 4 capsules layers, starting with primary capsules (PrimaryCaps) layer followed by three convolutional capsule (ConvCaps) layers. The stem of the network consists of a 5 × 5 Conv layer using 256 filters and stride 2, and is followed by a 1 × 1 Conv layer with 256 filters, both with BatchNorm and ReLU activations. The PrimaryCaps layer transforms the 256 filters into 16 capsule pose 4x4 matrices and activations with 1 × 1 convolutions, outputting 16 capsule types. This is followed by a 3 × 3 ConvCaps layer with 16 capsules types and stride 2, and a 3 × 3 ConvCaps layer with 16 capsule types with stride 1. The final ConvCaps layer shares weight matrices across spatial dimensions, yielding a capsule for each class, and we perform coordinate addition [8] by embedding the spatial coordinate of capsules into their vote matrices. Lastly, the reader is invited to refer to the supplementary material for more details/visualisations, and all source code will be made publicly available should the paper be accepted.

#### Objective Function

We experiment with both the standard negative likelihood loss \( \mathcal{L}_{\text{NLL}} \) and the spread loss function \( \mathcal{L}_{\text{SL}} \) presented in [8], and add the VAE reconstruction and KL penalty loss \( \mathcal{L}_{\text{VAE}} \) of the decoder as a regulariser

\[
\mathcal{L}_{\text{SL}} = \sum_{i \neq j} \max(0, m - (a_t - a_j))^2, \quad \mathcal{L}_{\text{NLL}} = - \sum_j \sum_k y_{jk} \log \hat{y}_{jk},
\]

\[
\mathcal{L}_{\text{VAE}} = \frac{1}{2} \sum_d \left( \sigma_{jd}^2 + \mu_{jd}^2 - \ln \sigma_{jd}^2 - 1 \right) + \frac{1}{K} \sum_k \|x_k - f(x_k)\|_F^2. \tag{17}
\]

The total loss is simply a linear combination of one of the classification losses, and the VAE loss as a regulariser, i.e. \( \mathcal{L} = \mathcal{L}_{\text{NLL}} + \alpha \mathcal{L}_{\text{VAE}} \). Regularisation by reconstruction was proposed in the first capsule paper [15] with a normal decoder network, to ensure capsules can encode and reconstruct the input. We extend this idea in our model by using a VAE.

#### Implementation Details

CapsNet is known to be difficult to train due to dead capsules at the beginning of training [8]. In this paper we provide some suggestions for practitioners on how to initialise the various parameters of the model that worked well for us experimentally, and helped stabilise training in general. We initialise the transformation weights \( W_{ij} \), as identity matrices with added random noise \( \epsilon \) from a uniform distribution \( \epsilon \sim \text{Unif}(0, 1) \) on the off diagonal entries. In
| Method          | smallNORB Test error rate % | Params | Fashion-MNIST Method | Fashion-MNIST Test error rate % | Params |
|-----------------|-----------------------------|--------|-----------------------|---------------------------------|--------|
| Dynamic [15]    | 2.7%                        | 8.2M   | MS-Caps [17]           | 7.3%                            | 10.8M  |
| FREM [19]       | 2.2%                        | 1.2M   | FREM [19]              | 6.2%                            | 1.2M   |
| LVQCaps [16]    | 5.6%                        | 442K   | Nair et al. [12]       | 10.2%                           | 8.2M   |
| DCNet++ [13]    | 4.7%                        | 13.4M  | HitNet [2]             | 7.7%                            | ≈8.2M  |
| FRMS [19]       | 2.6%                        | 1.2M   | MLCN [14]              | 7.37%                           | 10.6M  |
| EM-Routing [8]  | 1.8%                        | 310K   | MaxMin [20]            | 7.93%                           | ≈8.2M  |
| **Our Method**  | **1.55%**                   | **364K**|                      |                                 |        |

This way, at the start of training the transformations don’t stray too far from computing the identity function. In our method we activate the capsules $a_j$ outside of the routing loop, and activate only at the end as is customary in all neural net layers. We also normalise the argument of the capsule’s logistic activation function to have mean 0 and s.d. 1. This restricts the range of values fed through the logistic function from being too high/low resulting in close to 0 gradients. All network stem convolutional layers are initialised as He uniform [6], and the primary capsule layers are initialised as Xavier uniform [4] due to the sigmoid activation on the primary capsules.

**Choosing Priors** We keep it simple by setting the gaussian priors on the mean parameters $m_0$ to be zeros with precision scaling $\kappa_0 = 1$, and the wishart priors on the precision matrix $\Psi_0$ to be identities $I$ with degrees of freedom $\nu_0 = D + 1$. For the diagonal case, $\lambda_0$ is a vector of 1’s. These priors have a regularising effect since they encourage the capsule clusters to be remain close to the origin, and not to be too irregular in shape. The dirichlet prior on the mixing coefficients $\alpha$ is set to 1, and reducing this value favours solutions with less active capsules. We leave further analysis of the effect of priors on capsules to future work.

**smallNORB** The smallNORB dataset is the ideal dataset for testing Capsule Networks because it was carefully designed to be a tough shape-recognition task. The different viewpoints (azimuth, lighting and elevation) provided in the dataset help researchers evaluate how well Capsule perform at precisely the task they’re designed for. SmallNORB consists of 5 classes of grey-level stereo 96x96 images of toys. Each toy is given at 18 different azimuths (0-340), 9 elevations and 6 lighting conditions, and there are 24,300 images in the training and test sets each. Following [8], we standardize and resize the images to $48 \times 48$ and take random crops of $32 \times 32$ during training, and no further data augmentation techniques are used. At test time, we simply center crop the images.

**Fashion-MNIST** The Fashion-MNIST is a relatively new dataset which was designed to serve as a more difficult drop-in replacement for the MNIST dataset. Fashion-MNIST is also a popular choice for benchmarking Capsule Net research. The dataset resembles MNIST in that there are 10 classes, each is a different clothing item. The images are 28x28 and the training and test sets have 60,000 and 10,000 examples respectively. We standardize and resize the images to be $36 \times 36$, and randomly crop $32 \times 32$ patches during training. No further data augmentation is used. At test time we simply take center crops of the images.

**Results** In general we found that CapsNets have trouble with vanishing gradients due to the routing loops. Backpropagating through the unrolled routing loops proves difficult especially since the capsule activations are logistic and can yield small gradients naturally. In order to mitigate this problem we both normalise the argument of the capsule activation function to have mean 0 and standard deviation 1, and we add skip connections over each capsule layer. In total there are 3 skip connections, one over the PrimaryCaps layer and another two over the convolutional capsule layers (1 each) before the classification layer. Our best model was trained for 350 epochs using Spread Loss, Adam with exponentially decaying learning rate, 1e-6 weight decay, batch size 32, diagonal precision parameterisation, and number of variational bayes routing iterations {2, 2, 3} for each convolutional capsule layer respectively. As reported in Table 1, our approach outperforms previous methods on smallNORB. We achieve a test error rate of 1.55% compared to the previous state-of-the-art 1.8% as reported in the original Capsule EM routing paper [8]. Note that by averaging multiple
crops at test time they can get 1.4\% and we get 1.29\%. However, our result is obtained without adding random brightness/contrast or any other augmentations/deformations during training. We also stress that our network has \( \approx 50\% \) less capsules than the original in [8], and is therefore much more efficient at leveraging its routing capacity. For the Fashion-MNIST experiments we use the same small architecture containing an equal number of capsules as used in smallNORB. We only double the number of features maps in the first two convolutional layers of the network (before PrimaryCaps) from 256 to 512, to increase the network’s representation capacity before routing. Our best model was trained using the negative likelihood loss with the added VAE loss as a regulariser given by \( \mathcal{L} = \mathcal{L}_{\text{NLL}} + \alpha \mathcal{L}_{\text{VAE}} \) as opposed to the Spread Loss. We trained for 350 epochs, using Adam with an exponentially decaying learning rate, 1e-6 weight decay, 1e-7 \( \alpha \) for the VAE loss, batch size 32, and \{2, 2, 3\} routing iterations for each convolutional capsule layer respectively. The results in Table 1 demonstrate the greater generalisation capacity of our model compared to others in capsule literature, as our network has considerably less parameters and number of capsules but still performs better in almost all cases.

Routing Comparison & Weight Initialisation

We evaluate the performance of our routing algorithm against EM routing. We start by taking the architecture described in section 4 and train it on smallNORB using Variational Bayes routing between the convolutional capsules layers. Then we take the same network and simply replace the routing layers with our implementation of EM routing. We employ a diagonal precision parameterisation in all cases for fairer comparison, and the results can be seen in Figure 1a. All versions of our algorithm and architecture, namely with and without the VAE loss and/or skip connections, perform better than EM routing using the same network in our experiments. Finally, we perform some comparative analysis on how different weight initialisation schemes affect the training of capsule networks for the benefit of practitioners. To do so we build a smaller capsule network with Variational Bayes routing (without skip connections or a decoder), and train it using 4 different weight initialisation schemes for the 4x4 capsule pose matrices. We can see from the resulting Figure 1b that initialising the transformation matrices as noisy identities helps mitigate the problem of dead capsules [8] at the beginning of training and improves overall performance. We theorise that initialising the transformations matrices to be positive definite has beneficial geometrical properties, from convex optimisation type arguments. We leave further analysis of of this idea to future work.

5 Conclusion

In this paper we propose a new capsule routing algorithm based on Variational Bayes for a mixture of transforming gaussians. We exploit the natural structure of capsule networks and routing-by-agreement to model uncertainty over the capsule parameters, allowing more flexibility and control over capsule complexity when needed by tuning priors to induce sparsity and avoid overfitting. We
show that our method works well in practice, achieving state-of-the-art performance on smallNORB using $\approx 50\%$ less capsules, and without using any data augmentation/deformations except for cropping. These results suggest that greater generalisation capabilities can be obtained by placing priors over capsules’ transforming gaussians. In summary, we argue that the combination of Bayesian learning and CapsNets has potential since both domains value explainability in decision making, and in this paper we simply take the first step in this direction with hope to stimulate the research community. For future work, we intend to leverage the flexibility of CapsNets to obtain calibrated uncertainty estimates over predictions using larger networks and more complex datasets.

References

[1] C. M. Bishop. Pattern recognition and machine learning. springer, 2006.

[2] A. Deliège, A. Cioppa, and M. Van Droogenbroeck. An effective hit-or-miss layer favoring feature interpretation as learned prototypes deformations. In Thirty-Third AAAI Conference on Artificial Intelligence, 2019.

[3] K. Duarte, Y. Rawat, and M. Shah. Videocapsulenet: A simplified network for action detection. In Advances in Neural Information Processing Systems, pages 7610–7619, 2018.

[4] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the thirteenth international conference on artificial intelligence and statistics, pages 249–256, 2010.

[5] S. Guiasu. Weighted entropy. Reports on Mathematical Physics, 2(3):165–179, 1971.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the IEEE international conference on computer vision, pages 1026–1034, 2015.

[7] G. E. Hinton, A. Krizhevsky, and S. D. Wang. Transforming auto-encoders. In International Conference on Artificial Neural Networks, pages 44–51. Springer, 2011.

[8] G. E. Hinton, S. Sabour, and N. Frosst. Matrix capsules with em routing. In International Conference on Learning Representations (ICLR), 2018.

[9] D. P. Kingma and M. Welling. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013.

[10] J. E. Lenssen, M. Fey, and P. Libuschewski. Group equivariant capsule networks. In Advances in Neural Information Processing Systems, pages 8844–8853, 2018.

[11] K. P. Murphy. Machine learning: a probabilistic perspective. MIT press, 2012.

[12] P. Nair, R. Doshi, and S. Keselj. Pushing the limits of capsule networks. Technical note, 2018.

[13] S. S. R. Phaye, A. Sikka, A. Dhall, and D. Bathula. Dense and diverse capsule networks: Making the capsules learn better. arXiv preprint arXiv:1805.04001, 2018.

[14] V. M. d. Rosario, E. Borin, and M. Breternitz Jr. The multi-lane capsule network (mlcn). arXiv preprint arXiv:1902.08451, 2019.

[15] S. Sabour, N. Frosst, and G. E. Hinton. Dynamic routing between capsules. In Advances in Neural Information Processing Systems (NIPS), pages 3856–3866, 2017.

[16] S. Saralajew, S. Nooka, M. Kaden, and T. Villmann. Learning vector quantization capsules. Machine Learning Reports, 12:1–17, 2018.

[17] C. Xiang, L. Zhang, Y. Tang, W. Zou, and C. Xu. Ms-capnet: A novel multi-scale capsule network. IEEE Signal Processing Letters, 25(12):1850–1854, 2018.

[18] L. Zhang, M. Edraki, and G.-J. Qi. Cappronet: Deep feature learning via orthogonal projections onto capsule subspaces. In Advances in Neural Information Processing Systems, pages 5814–5823, 2018.

[19] S. Zhang, Q. Zhou, and X. Wu. Fast dynamic routing based on weighted kernel density estimation. In International Symposium on Artificial Intelligence and Robotics, pages 301–309. Springer, 2018.

[20] Z. Zhao, A. Kleinmans, G. Sandhui, I. Patel, and K. Unnikrishnan. Capsule networks with max-min normalization. arXiv preprint arXiv:1903.09662, 2019.
6 Supplementary Material

More comparisons In all cases here we use a small CapsNet (37k params) similar to the one described in section 4, but with only 32 feature maps in the first convolutional layer, followed by PrimaryCaps and 3 ConvCaps layers each with 8 capsules types. The network was trained on smallNORB and validated on a 20% subset of the training set. In all experiments in Figures 2, 3, and 4 we use the same architecture, Adam optimiser with default params, batch size 32, weight decay 1e-6, 3 routing iterations and negative log likelihood loss.

Figure 2: Example of VB routing outperforming EM routing using the exact same architecture, loss function, weight initialisation, parameters and hyperparameters during training.

Figure 3: Extreme example of dead capsules at the beginning of training using Xavier initialisation schemes for the pose transformation weight matrices.

Capsule Vote Analysis In order to better understand the mechanics of capsule training, we analyse the behaviour of capsule votes from each class in smallNORB by visualising the discrepancies between them and the respective target class capsule. We use the small network described above using Variational Bayes routing and the negative log likelihood loss regularised by the VAE decoder loss \( L = L_{\text{NLL}} + \alpha L_{\text{VAE}} \).

Figures 5-6 show time-histograms of the squared distances between votes \( V_{ij} \), averaged each of the individual class images, and each of the all 5 class capsules \( M_j \) throughout training. Routing iterations 1-3 are depicted per row, and each column represents a different class capsule. As can be seen below, the average votes from each class’ images learn to agree with the respective target class’ capsules during training. We can see that the discrepancies between the votes and the respective target class capsules increasingly gather around 0 over time, more so than other capsules.
Figure 4: Performance comparison between using different number of routing iterations. \{·,·,·\} denotes number of routing iterations used in each of the 3 layers. Little performance is gained from using more than 2 iterations so we use \{2, 2, 3\} for the speed gain.

Figure 5: Histograms of the squared distances (X axis) between votes $V_{j|i}$ averaged over all car images, and each of the all 5 class capsules $M_j$ throughout training (epochs on Y axis). We can see a very clear difference in the agreement between target (car) and non-target capsules even without inspecting the distances on the X axis.
Figure 6: Histograms of the squared distances (X axis) between votes $V_{ji}$ averaged over all **airplane** images, and each of the all 5 class capsules $M_j$ throughout training (epochs on Y axis). Routing iterations 1-3 are depicted per row, and each column represents a different class capsule. As can be seen above, the average votes from the airplane images learn to agree with the airplane class capsule during training, and therefore the discrepancies between the votes and the target capsule increasingly gather around 0 over time more than the other capsules.