Why Capsule Neural Networks Do Not Scale: Challenging the Dynamic Parse-Tree Assumption

Matthias Mitterreiter[1,4], **Marcel Koch**[2], **Joachim Giesen**[1], **Sören Laue**[3]
[1]Friedrich-Schiller-University, Jena, Germany
[2]Ernst Abbe University of Applied Sciences, Jena, Germany
[3]Technical University Kaiserslautern, Germany
[4]Data Assessment Solutions GmbH, Hannover, Germany
matthias.mitterreiter@uni-jena.de, marcel.koch@eah-jena.de, joachim.giesen@uni-jena.de, laue@cs.uni-kl.de

Abstract

Capsule neural networks replace simple, scalar-valued neurons with vector-valued capsules. They are motivated by the pattern recognition system in the human brain, where complex objects are decomposed into a hierarchy of simpler object parts. Such a hierarchy is referred to as a parse-tree. Conceptually, capsule neural networks have been defined to realize such parse-trees. The capsule neural network (CapsNet), by Sabour, Frosst, and Hinton, is the first actual implementation of the conceptual idea of capsule neural networks. CapsNets achieved state-of-the-art performance on simple image recognition tasks with fewer parameters and greater robustness to affine transformations than comparable approaches. This sparked extensive follow-up research. However, despite major efforts, no work was able to scale the CapsNet architecture to more reasonable-sized datasets. Here, we provide a reason for this failure and argue that it is most likely not possible to scale CapsNets beyond toy examples. In particular, we show that the concept of a parse-tree, the main idea behind capsule neuronal networks, is not present in CapsNets. We also show theoretically and experimentally that CapsNets suffer from a vanishing gradient problem that results in the starvation of many capsules during training.

1 Introduction

The concept of capsules (Hinton, Krizhevsky, and Wang 2011) describes a hypothetical system that parses a complex image scene into a hierarchy of visual entities that stand in part-whole relationship to each other (Hinton, Ghahramani, and Teh 1999). A capsule is conceptually defined as a highly informative, compact representation of a visual entity or object within an image. The idea of capsules is motivated by the pattern recognition system in the visual cortex of the human brain (Sabour, Frosst, and Hinton 2017). There is some psychological evidence that the human object recognition system assigns hierarchical structural descriptions to complex objects by decomposing them into parts (Hinton 1979). The theory of recognition by components (Biederman 1987) proposes that a relatively small set of simple 3D shapes, called geons, can be assembled in various arrangements to form virtually any complex object, which can then be recognized by decomposition into its respective parts (Biederman 1987).

A capsule may represent a visual entity by encapsulating its properties, also known as instantiation parameters, such as position, size, orientation, deformation, texture, or hue. A multi-level assembly of such capsules represents a complex image scene, where lower-level capsules model less abstract objects or object parts, and higher-level capsules model complex and composite objects. Lower-level capsules are connected to higher-level capsules if the corresponding entities are in a part-whole relationship. For a composite object, the hierarchy of capsules defines a syntactic structure like a parse-tree defines the syntactic structure of a sentence. Therefore, the hierarchy of capsules is also referred to as parse-tree. If an object or object part is present in an image, its respective capsule will be present within the parse-tree.

Ideally, the parse-tree is invariant under affine transformations as well as changes of viewpoint. That is, a slightly modified viewpoint on a visual entity should not change a capsule's presence within the parse-tree. Such parse-trees would be highly efficient distributed representations of image scenes (Sabour, Frosst, and Hinton 2017; Hinton, Ghahramani, and Teh 1999). Also, explainable machine learning can profit from interpretable capsules that stand for dedicated visual entities, and the discrete nature of trees may connect deep learning with a symbolic approach to AI. Furthermore, capsules can be related to inverse graphics, and there is hope that they can lead to debuggable, parameter efficient, and interpretable models with a broad range of applications for all kinds of image-related tasks like image classification or segmentation.

However, capsules are only conceptually defined, and the difficulty is finding an implementation with all the highly-desirable properties from above. The capsule neural network (CapsNet) by Sabour, Frosst, and Hinton (2017) aims at such an implementation of the conceptual capsule idea. It was specifically designed to surpass convolutional neural networks (ConvNets) (LeCun et al. 1989) as the latter were found to suffer from several limitations, including a lack of robustness to affine transformations and change of viewpoint, the susceptibility to adversarial attacks, exponential inefficiencies, and a general lack of interpretability in the network's decision-making process. Considering these limi-

tations, the parse-tree sounds particularly appealing with all its advantages.

Contributions. Here, our aim is a thorough investigation of the question, whether the CapsNets implementation as proposed by Sabour, Frosst, and Hinton (2017) realizes all the conceptual ideas that make capsule networks so appealing. We summarize this in two key assumptions. The **first key assumption** is that the CapsNet learns to associate a capsule with a dedicated visual entity within an input image (Sabour, Frosst, and Hinton 2017). The **second key assumption** is that the CapsNet's capsules can be organized hierarchically in a parse-tree that encodes part-whole relationships. We test both assumptions experimentally and have to reject them. We show that the CapsNet does not exhibit any sign of an emerging parse-tree. Thus, the CapsNet implementation cannot provide the theoretical benefits of capsule networks like invariance under affine transformations and change of viewpoint. Furthermore, we provide a theoretical analysis, exposing a vanishing gradient problem, that supports our experimental findings.

2 Related Work

Early references to the hierarchy of parts appear already in (Hinton 1979). The idea of parsing images into parse-trees was proposed by Hinton, Ghahramani, and Teh (1999) and the concept of capsules was established in (Hinton, Krizhevsky, and Wang 2011). An important addition by the CapsNet (Sabour, Frosst, and Hinton 2017) was the routing-by-agreement (RBA) algorithm that creates capsule parse-trees from images. With its introduction, the CapsNet demonstrated state-of-the-art classification accuracy on MNIST (LeCun et al. 1998) with fewer parameters and stronger robustness to affine transformations than the ConvNet baseline, which sparked a flood of follow-up research. This includes different routing mechanisms, such as EM-Routing (Hinton, Sabour, and Frosst 2018), Self-Routing (Hahn, Pyeon, and Kim 2019), Variational Bayes Routing (Ribeiro, Leontidis, and Kollias 2020), Receptor Skeleton (Chen et al. 2021) and attention-based routing (Ahmed and Torresani 2019; Tsai et al. 2020; Mazzia, Salvetti, and Chiaberge 2021; Gu 2021). Wang and Liu (2018) reframed the routing algorithm in (Sabour, Frosst, and Hinton 2017) as an optimization problem, and Rawlinson, Ahmed, and Kowadlo (2018) introduced an unsupervised learning scheme for CapsNets. Other work replaced the capsule vector representations by matrices (Hinton, Sabour, and Frosst 2018) or tensors (Rajasegaran et al. 2019), or added classic ConvNet features to the general routing mechanisms, such as dropout (Xiang et al. 2018) or skip-connections (Rajasegaran et al. 2019). The GLOM architecture, which was proposed by (Hinton 2021), suggests a routing-free approach for creating parse-trees from images, but has not been implemented yet. Furthermore, other publications focus on learning better first layer capsules (PrimeCaps), such as the Stacked Capsule Autoencoders (Kosiorek et al. 2019) and Flow Capsules (Sabour et al. 2021).

However, after a while it turned out that the CapsNet falls short of the anticipated benefits and promises of the capsule idea. To this date, CapsNets do not scale beyond small-scale datasets. Works that empirically report scaling issues include (Xi, Bing, and Jin 2017; Paik, Kwak, and Kim 2019). Although the CapsNet was introduced in the realm of computer vision, the best performing capsule implementation (Ahmed and Torresani 2019) achieves only 60.07% top-1 image classification accuracy on ImageNet (Deng et al. 2009), far behind state-of-the-art transformer-based approaches (Wortsman et al. 2022) and ConvNets (Pham et al. 2021) with 90.88% and 90.02% accuracy respectively. The original CapsNet itself has not been demonstrated to work on ImageNet.

Further negative results regarding CapsNets emerged, questioning the promised benefits and technical progress altogether. Paik, Kwak, and Kim (2019) observed that increasing the depth of various CapsNet variants did not improve accuracy, and routing algorithms, the core components of capsule implementations, do not provide any benefit regarding accuracy in image classification. Michels et al. (2019), and Gu, Wu, and Tresp (2021) showed that CapsNets can be as easily fooled as ConvNets when it comes to adversarial attacks. Gu, Tresp, and Hu (2021) showed that the individual parts of the CapsNet have contradictory effects on the performance on different tasks and conclude that with the right baseline, CapsNets are not generally superior to ConvNets. Finally, Gu and Tresp (2020) showed that removing the dynamic routing component improves affine robustness, and Rawlinson, Ahmed, and Kowadlo (2018) show that capsules do not specialize without supervision.

Here, we explain these shortcomings, which can be attributed to a lack of an emerging parse-tree.

3 The Capsule Neural Network

The CapsNet implements capsules as parameter vectors. An illustration of the CapsNet architecture, which consists of a multi-layer hierarchy of capsules, is shown in Figure 1. In the following, we introduce basic notations and definitions, the generic CapsNet architecture, and a loss function for training CapsNets. Furthermore, we discuss how CapsNets implement the crucial concept of a parse-tree.

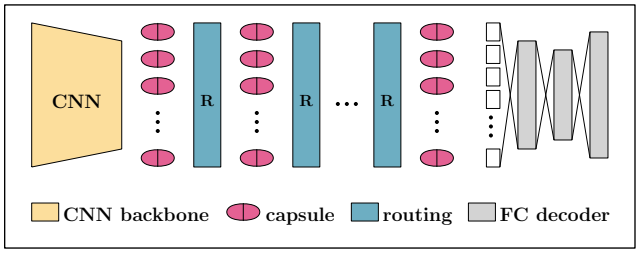


Figure 1: A generic CapsNet architecture.

3.1 Notation

Capsules. The matrix $U^l \in \mathbb{R}^{n^l \times d^l}$ holds n^l normalized capsules of dimension d^l for layer $l \in \{1, 2, \ldots, \ell\}$. The i-th capsule in U^l is the vector $u^l_{(i,:)} \in \mathbb{R}^{d^l}$, and $u^l_{(i,j)} \in \mathbb{R}$

is the j-th entry of capsule i on layer l.

Transformation matrices. The tensor $W^l \in \mathbb{R}^{n^{l+1} \times n^l \times d^{l+1} \times d^l}$ holds transformation matrices $W^l_{(j,i,:,:)} \in \mathbb{R}^{d^{l+1} \times d^l}$. The transformation matrix $W^l_{(j,i,:,:)}$ maps the i-th capsule of layer l to its unnormalized contribution to the j-the capsule of layer $l+1$.

Coupling coefficients. The matrix $C^l \in \mathbb{R}^{n^l \times n^{l+1}}$ holds coupling coefficients for the connections of capsules from layer l to layer $l+1$. The entry $c^l_{(i,j)} \in [0,1]$ specifies the coupling strength between capsule i on layer l and capsule j on layer $l+1$. The coupling coefficients satisfy $\sum_{j=1}^{n^{l+1}} c^l_{(i,j)} = 1$ for all $i \in \{1, 2, \ldots, n^l\}$.

Squashing function. The squashing function normalizes the length of a capsule vector $u \in \mathbb{R}^d$ into the range $[0, 1)$. Here, we use a slightly modified squashing function (Mazzia, Salvetti, and Chiaberge 2021),

$$g(u) = \left(1 - \frac{1}{\exp(\|u\|_2)}\right) \frac{u}{\|u\|_2} \tag{1}$$

that behaves similarly to the original squashing function proposed by Sabour, Frosst, and Hinton (2017), but is more sensitive to small changes near zero (Xi, Bing, and Jin 2017), which is required to stack multiple layers of capsules.

3.2 Architecture

First, the backbone network extracts features from an input image into a feature matrix in $\mathbb{R}^{n^1 \times d^1}$. The feature matrix is then normalized by applying the squashing function to each row, which constitutes the first capsule layer in $U^1 \in \mathbb{R}^{n^1 \times d^1}$. The capsules in U^1 are also called PrimeCaps. Starting from the PrimeCaps, consecutive layers of capsules are computed as follows: First, the linear contribution of capsule i on layer l to capsule j on layer $l+1$ is computed as

$$\hat{u}^{l+1}_{(i,j,:)} = W^l_{(j,i,:,:)} u^l_{(i,:)}, \tag{2}$$

where the entries in the matrix $\hat{U}^{l+1}_{(i)}$, which holds the vectors $\hat{u}^{l+1}_{(i,j,:)}$, are called **votes** from the i-th capsule on layer l. An upper layer capsule $u^{l+1}_{(j,:)}$ is the squashed, weighted sum over all votes from lower layer capsules $u^l_{(i,:)}$, that is,

$$u^{l+1}_{(j,:)} = g\left(\sum_{i=1}^{n^l} c^l_{(i,j)} \cdot \hat{u}^{l+1}_{(i,j,:)}\right),$$

where the the coupling coefficients $c^l_{(i,j)}$ are dynamically computed, that is, individually for every input image, by the Routing-by-agreement Algorithm (RBA), see Algorithm 1.

The number of output capsules on the last layer ℓ is set to match the number of classes in the respective dataset. Finally, the fully-connected decoder network reconstructs the input image from the capsules on layer ℓ.

3.3 Training

The parameters in the backbone network, in the reconstruction network, as well as the transformation tensors W^l and the RBA routing priors $b^l_{(i,j)}$ are all learned by minimizing a weighted sum of a supervised classification loss L_m and an unsupervised reconstruction loss L_r, that is, $L = L_m + \alpha \cdot L_r$, with $\alpha > 0$. The classification loss function

$$L_m = \sum_{j=1}^{n^\ell} t_j \cdot \max\{0, m^+ - \|u^\ell_{(j,:)}\|_2\}^2 + \lambda \cdot (1 - t_j) \cdot \max\{0, \|u^\ell_{(j,:)}\|_2 - m^-\}^2 \tag{3}$$

is only applied to the output capsules. Here $m^+, m^- > 0$ and $\lambda > 0$ are regularization parameters, and t_j is 1 if an object of the j-th class is present in the input image, and 0 otherwise. Output capsules that correspond to classes not present in the input image are masked by zeros. The reconstruction loss function is applied to the output of the reconstruction network and sums the distances between the reconstruction and the pixel intensities in the input image.

Algorithm 1: Routing-by-agreement (RBA)

Input: votes $\hat{u}$, number of iterations r, routing priors b
Output: coupling coefficients c

for r iterations **do**
  $c_{(i,j)} \leftarrow \frac{\exp(b_{(i,j)})}{\sum_k \exp(b_{(i,k)})}$
  $v_{(j,:)} \leftarrow g\left(\sum_i c_{(i,j)} \hat{u}_{(i,j,:)}\right)$
  $b_{(i,j)} \leftarrow b_{(i,j)} + \langle \hat{u}_{(i,j,:)}, v_{(j,:)} \rangle$
end for

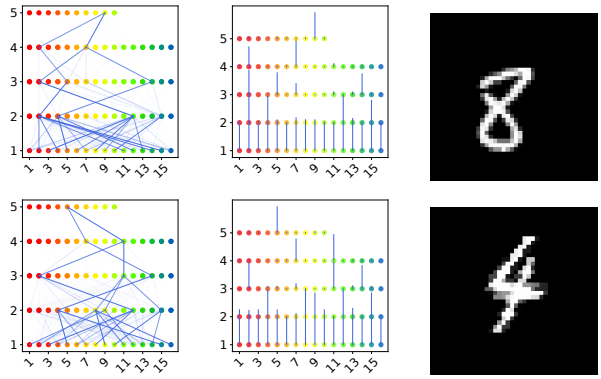
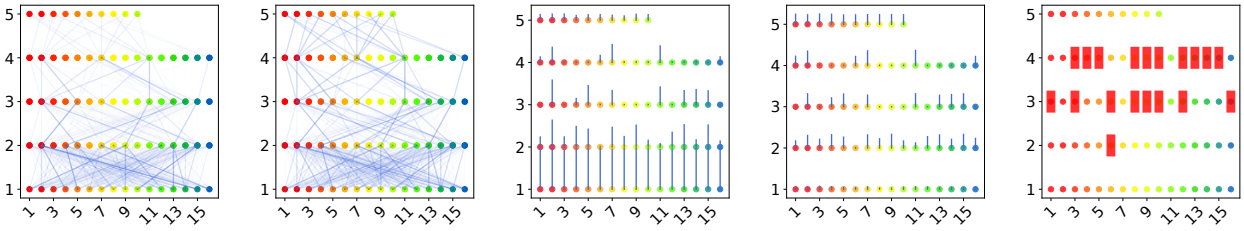


Figure 2: Fuzzy parse-trees for images from the AffNIST dataset for a model with five capsule layers, 16 capsules on each intermediate layer, and ten on the last layer. The figure shows the coupling coefficients as connections between capsules (left), the capsule norms/activations (middle), and the input image (right). The blue tone of the edges is darker for greater coupling coefficients.

3.4 Parse-Trees

The parse-tree is the most important concept that allows us to understand and interpret capsules and their connections. The CapsNet defines a parse-tree, where capsules represent nodes, and coupling coefficients represent fuzzy edges.



(a) mean couplings (b) std. couplings (c) mean activations (d) std. activations (e) dead capsules

Figure 3: Parse-tree statistics for the complete AffNIST validation dataset for a five-layer CapsNet model with 16/10 capsules. The mean (a) and the standard deviation (b) of the coupling coefficient matrices for each layer are visualized as connections between capsules. Higher coupling coefficients have a darker blue tone. The capsule norms' mean (c) and standard deviation (d) are visualized by bars. Dead capsules (e) are highlighted with a red bar.

The capsules magnitude, that is, the norm of the parameter vector, which is always in $[0, 1)$ after applying the squashing function, represents the probability that the corresponding entity is present in an input image. The capsules direction represents instantiation parameters of the entity like its position, size, or orientation. Changing the viewpoint on an entity does not affect its presence, but only its instantiation parameters. Therefore, the respective capsule's magnitude should be unaffected, whereas its direction can change.

In the CapsNet, the dynamic part-whole relationships are implemented by coupling coefficients between capsules on consecutive layers. The coupling coefficients in the routing layers are computed dynamically by the RBA Algorithm 1. Taking the row-wise *softmax* ensures that the coupling coefficients in $c^l_{(i,:)}$ are positive and sum up to one. Therefore, we can view the coupling coefficients as fuzzy edges that connect capsules $u^l_{(i,:)}$ and $u^{l+1}_{(j,:)}$ with probability $c^l_{(i,j)}$. The multi-layer hierarchy of capsule nodes, connected by fuzzy edges, defines the parse-tree analogously to a probabilistic context-free grammar. Examples are shown in Figure 2.

4 Challenging the Parse-Tree Assumption

As mentioned in the introduction, there are two key assumptions regarding the parse-tree: (1) The nodes of the parse-tree, the activated capsules, are viewpoint-invariant representations of visual entities present in the input image. (2) Lower-level capsules represent object parts, higher-level capsules represent composite objects, and part-whole relationships are represented by the edges of the parse tree, that is, by the coupling coefficients. In the following, we are going to challenge both assumptions.

If Assumption (2) holds, then the parse-tree computed by a CapsNet is a part-whole representation of the image scene, and the routing dynamics defined by the coupling coefficients is specific to the input image. We conduct experiments challenging this assumption in Section 4.1.

If Assumption (1) holds, then affine transformations of an image only change the direction of a parameter vector of a capsule, but not its magnitude. Hence, we take an image, transform it affinely, and analyze the resulting capsule activations. These experiments can be found in Section 4.2. Furthermore, we examine the capsule activation in general in Section 4.3.

Experimental Setup We use the AffNIST benchmark (Sabour, Frosst, and Hinton 2017) to assess a model's robustness to affine transformations, and we use the CIFAR10 dataset (Krizhevsky 2009) to test a model's performance on complex image scenes. We conduct extensive experiments using a total of 121 different model architectures of various scales, featuring different numbers of routing layers and different numbers and capsule dimensions. Shallow models resemble the original CapsNet implementation while deeper models allow for a more semantically expressive parse-tree.

We refer to the appendix for detailed architecture and dataset descriptions, training procedures, and full results for all models used in our experiments.

4.1 Routing Dynamics

We measure the diversity of parse-trees, that is, the routing dynamics, by assessing the diversity of routing targets for a single capsule u. For k input images, let $C \in \mathbb{R}^{k \times n}$ hold all the coupling coefficients that connect u to capsules on the next layer, which contains n target capsules. We use the standard deviation $\mathrm{std}(c_{(:,i)})$ with respect to all input images as a measure for the routing diversity of u to the i-th capsule on the next layer. A routing pr_n is called **perfect** if it always routes to exactly one capsule on the next layer and routes to all n capsules on the next layer equally likely. The standard deviation of a perfect routing computes to $\mathrm{std}(pr_n) = \sqrt{\left(1 - \frac{1}{n}\right)\frac{1}{n}}$. We use a perfect routing to define the **dynamic routing coefficient (dyr)** for capsule u as

$$\mathrm{dyr}(u) = \frac{1}{n} \sum_{i=1}^{n} \frac{\mathrm{std}(c_{(:,i)})}{\mathrm{std}(\mathrm{pr}_n)}$$

The expected number of target capsules **(dys)** for u is $\mathrm{dys}(u) = n \cdot \mathrm{dyr}(u)$. For a whole layer, we define the coefficients **dyr** and **dys** as the mean over all capsules of this layer.

Results In the following, we report the routing statistics for a CapsNet architecture with four routing layers, 16 capsules per layer in the first four layers, and ten capsules in the last layer. We set the capsule dimension to eight and train multiple models on the AffNIST dataset until a target accuracy of 99.2% is reached. The routing statistics for the models are summarized in Table 1 and the corresponding coupling coefficients of a single model are visualized in Figure 3. The dys values below two for Layers 2 and 3 indicate low routing dynamics. A route is mostly predetermined once a capsule is activated; hence, the routing is almost static. Only the last layer exhibits higher routing dynamics, which can be attributed to the supervisory effect of the classification loss L_m. Since the routing is almost static, we conclude that the parse-trees do not encode the information necessary for a distributed representation of diverse image scenes, violating Assumption (2). As can be seen from Tables 7 to 12 in the supplement, the results look similar for all models trained on AffNIST. The more complex data set CIFAR10 exhibits even worse routing dynamics; see Table 13 and Figure 16 in the appendix.

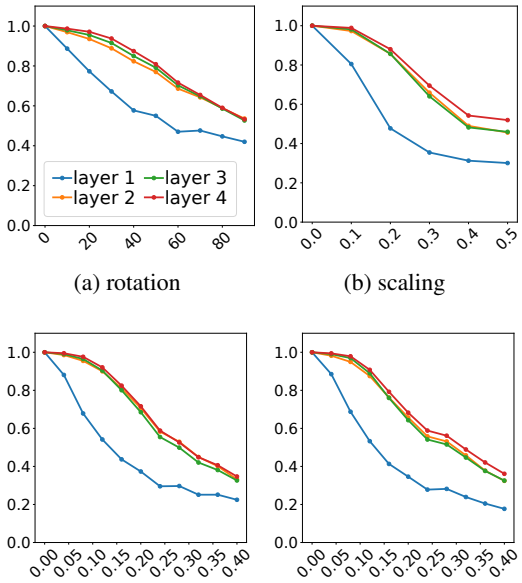
Layer	Capsules Alive	Routing Dynamics	
		Rate (dyr)	Mean (dys)
1	16.00 ± 0.00	0.30 ± 0.00	4.50 ± 0.17
2	14.90 ± 0.70	0.25 ± 0.01	1.72 ± 0.11
3	7.00 ± 0.63	0.30 ± 0.02	1.79 ± 0.16
4	5.90 ± 0.70	0.38 ± 0.04	3.78 ± 0.38
output	10.00 ± 0.00		

Table 1: The routing statistics for a CapsNet with four routing layers, 16/10 capsules per layer, and a capsule dimension of eight. We separately train and evaluate ten models on AffNIST the same way and report the mean and standard deviation.

4.2 Viewpoint Invariance

We investigate to which degree capsule activations are invariant to affine transformations of the input images. Let $x^{(1)}$ and $x^{(2)}$ show the same visual entity though differently instantiated. For one specific capsule, let $u^{(1)}, u^{(2)} \in \mathbb{R}^d$ be the corresponding capsule responses for the two images. For a viewpoint invariant parse-tree, it holds that $\left\|u^{(1)}\right\|_2 = \left\|u^{(2)}\right\|_2$, since $\|u\|_2$ measures the probability that the visual entity is present in the image. We repeat this process for a set of k input images and collect the corresponding capsule activations in the two vectors $v^{(1)}$ and $v^{(2)} \in \mathbb{R}^k$. We compute the empirical cross-correlation between these two vectors as $\frac{(v^{(1)})^\top v^{(2)}}{\left\|v^{(1)}\right\|_2 \cdot \left\|v^{(2)}\right\|_2}$. For one layer, we compute the average of this value over all capsules of this layer. For a viewpoint-invariant parse-tree, this value should be one.

Results We observe that the capsule activation correlation decreases for increasingly stronger affine transformations, see Figure 4. This observation holds for all intermediate capsule layers, and all tested affine transformations. See also Figure 5 for a qualitative example showing two different parse-trees that are expected to be identical. We conclude that the parse-tree is not invariant under affine transformations of the input image, violating Assumption (1). Furthermore, since already the activations of the PrimeCaps do not exhibit viewpoint-invariance, we believe that capsules need a different backbone that gives rise to better PrimeCaps.



(a) rotation (b) scaling

(c) horizontal translation (d) vertical translation

Figure 4: The capsule activation correlations for each layer with respect to increasing degree of affine transformations.

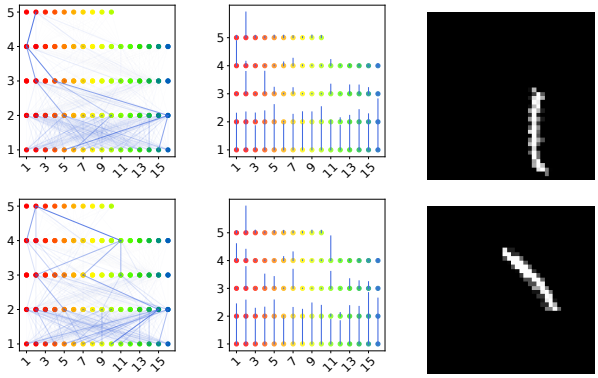


Figure 5: Similar input image, different parse tree.

4.3 Capsule Activation

In this section, we analyze the capsule activation and thus the parse-tree nodes. For a layer with n capsules of dimension d, let $U \in \mathbb{R}^{k \times n \times d}$ hold the n capsule responses of dimension d for k input images. We denote the entries of U by the lower letter u with corresponding lower indices. We define the **capsule norm sum (cns)** as the sum of capsule norms in

Capsule Layer	Capsule Norms		Capsule Activation		Capsule Deaths	
	Mean (cnm)	Sum (cns)	Rate (car)	Sum (cas)	Rate (cdr)	Sum (cds)
1	0.95 ± 0.00	15.25 ± 0.06	1.00 ± 0.00	16.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
2	0.32 ± 0.01	5.12 ± 0.18	0.70 ± 0.03	11.17 ± 0.48	0.07 ± 0.04	1.10 ± 0.70
3	0.18 ± 0.01	2.83 ± 0.08	0.35 ± 0.03	5.57 ± 0.42	0.56 ± 0.04	9.00 ± 0.63
4	0.12 ± 0.00	1.90 ± 0.07	0.22 ± 0.02	3.51 ± 0.27	0.63 ± 0.04	10.10 ± 0.70
5	0.15 ± 0.01	1.48 ± 0.05	0.30 ± 0.03	3.05 ± 0.00	0.00 ± 0.00	0.00 ± 0.00

Table 2: Capsule activation statistics for a CapsNet with five capsule layers, 16/10 capsules per layer, and a capsule dimension of eight. We separately train and evaluate ten models on AffNIST the same way and report the mean and standard deviation.

the respective layer averaged over all input images. For comparing layers with different numbers of capsules, we define the **capsule norm mean (cnm)**, which is the cns adjusted for the number of capsules that are present in the layer:

$$\mathrm{cns}(U) = \frac{1}{k}\sum_{i=1}^{k}\sum_{j=1}^{n}\left\|u_{(i,j,:)}\right\|_2, \qquad \mathrm{cnm}(U) = \frac{\mathrm{cns}(U)}{n}$$

We say a capsule j is active for input image i, if its norm exceeds a certain threshold ε, that is,

$$\mathbb{1}_{\mathrm{active}}(u_{(i,j,:)}) = \begin{cases} 1 & \left\|u_{(i,j,:)}\right\|_2 \geq \varepsilon \\ 0 & \text{otherwise.} \end{cases}$$

Furthermore, we define the **sum of active capsules (cas)** as the mean sum of active capsules per layer and the **rate of activate capsules** as the cas adjusted for the number of capsules:

$$\mathrm{cas}(U) = \frac{1}{k}\sum_{i=1}^{k}\sum_{j=1}^{n}\mathbb{1}_{\mathrm{active}}(u_{(i,j,:)}), \quad \mathrm{car}(U) = \frac{\mathrm{cas}(U)}{n}$$

We say that a capsule j is **dead** if the mean μ and the standard deviation σ of its norm over the k input images are below certain thresholds, that is,

$$\mathbb{1}_{\mathrm{dead}}(u_{(:,j,:)}) = \begin{cases} 1 & \mu(u_{(:,j,:)}) \leq \varepsilon_1 \text{ and } \sigma(u_{(:,j,:)}) \leq \varepsilon_2 \\ 0 & \text{otherwise.} \end{cases}$$

We compute the **sum of dead capsules (cds)** and the **rate of dead capsules (cdr)** as

$$\mathrm{cds}(U) = \sum_{j=1}^{n}\mathbb{1}_{\mathrm{dead}}\left(u_{(:,j,:)}\right), \quad \mathrm{cdr}(U) = \frac{\mathrm{cds}(U)}{n}$$

Results Table 2 summarizes the capsule activation statistics for the AffNIST experiment. As expected, the agreement of lower layer capsules, enforced by the RBA algorithm, results in a declining number of active capsules in the upper layers, as is witnessed by decreasing cas values. As a result, the overall sum of norms per layer drops, as can be seen in the cns values. Surprisingly, there is no sign of sparse activation within the PrimeCaps. All PrimeCaps are consistently active, as seen from the car values. This implies that PrimeCaps do not represent parts that are present in one image and not present in another, questioning the underlying assumption of distributed representation learning. It is another indication that the backbone does not deliver the representations required for PrimeCaps.

Furthermore, we observe that the number of dead capsules cdr increases with the depth of the model. For instance, 63% of the capsules on Layer 4 in the AffNIST experiment are dead. Figure 3e highlights the dead capsules. In other experiments, this value increases up to 84%, see Table 20 in the appendix. This has the following implications: First, the depth of a CapsNet is limited as the number of dead capsules rises with the number of layers. Second, the parse-tree cannot carry separate semantic information for each class if the number of active capsules is less than the number of classes. Third, dead capsules limit the capacity of a CapsNet as their respective parameters are not in use. This explains why baseline models trained with **uniform routing** perform better than models trained with RBA. Uniform routing, which uses all parameters, achieves better classification accuracies; see Tables 5 and 14 in the appendix. In uniform routing, all entries in the coupling coefficient matrix are set to the same fixed value. Our results stand in line with prior work (Paik, Kwak, and Kim 2019; Gu and Tresp 2020; Gu, Tresp, and Hu 2021) that also observed a negative impact of routing on model performance.

Theoretical Analysis In order to theoretically explain the dynamics of the activation of the capsules during training, we analyze the gradient of the loss function. We have the following theorem:

Theorem 1 *Let L_m be the margin loss function. The gradient of a single capsule $u^l_{(j,:)}$ of the upper-most layer l evaluates to:*

$$\frac{\partial L_m}{\partial u^l_{(j,:)}} = \Big(-t_j \max(0, m^+ - \left\|u^l_{(j,:)}\right\|_2) + \lambda(1-t_j)\max(0, \left\|u^l_{(j,:)}\right\|_2 - m^-)\Big) \cdot \frac{2u^l_{(j,:)}}{\left\|u^l_{(j,:)}\right\|_2}$$

The theorem follows directly from the definition of the classification loss function, Equation (3). The gradient is independent of the magnitude $\left\|u^l_{(j,:)}\right\|$ of the capsule activation. Hence, as long as the loss function is not zero, the gradient is large enough to force the capsules to either become active or inactive, depending on the label of the data point. Hence, all capsules on the upper-most layer will be active for the corresponding data points. This is in stark contrast to capsules

that are not on the upper-most layer. Here, it can happen that a capsule becomes dead during training. We observe, that once a capsule is dead, it never becomes active again, resulting in a starvation of capsules. The following theorem asserts this behavior.

Theorem 2 *Let L_m be the margin loss function. The gradient of a single capsule $u^l_{(i,:)}$, that does not belong to the upper-most layer evaluates to:*

$$\frac{\partial L_m}{\partial u^l_{(i,:)}} = \sum_j \frac{\partial L_m}{\hat{u}^{l+1}_{(i,j,:)}} \cdot W^l_{(j,i,:,:)}$$

The gradients of the corresponding weight matrices $W^l_{(j,i,:,:)}$ evaluate to:

$$\frac{\partial L_m}{\partial W^l_{(j,i,:,:)}} = \frac{\partial L_m}{\partial \hat{u}^{l+1}_{(i,j,:)}} \cdot u^l_{(i,:)}$$

The theorem follows from Equation (2). It states that the gradient of the weight matrix scales with the activation of the corresponding capsule, and the gradient of the capsule scales with the magnitude of the weight matrix. Hence, once both are small, they will not change sufficiently. In the limit, i.e., of magnitude zero, they will never change. In other terms, once a capsule becomes dead, it never becomes active again. Figure 10 in the appendix clearly show this behavior for the gradient of the capsule activation and Figures 11-12 in the appendix show this behavior for the gradient of the weight matrices. Dead capsules do not participate in the routing and are not part of any parse-tree. Also, note that the supervised loss forces the upper-most layers to be active. However, capsules can become dead on the intermediate layers where no supervised loss is directly present.

5 Comparing RBA with Self-Attention

We compare routing-by-agreement with the multi-head self-attention (MHSA) mechanism used in transformers (Vaswani et al. 2017) and more task-related vision transformers (Dosovitskiy et al. 2021). Like RBA, the MHSA mechanism operates on vectors and uses the softmax function to compute the normalized attention matrix, similar to the coupling coefficient matrix in RBA. However, contrary to RBA, which computes the softmax row-wise, MHSA enforces the softmax column-wise and, as a result, does not suffer from the previously discussed vanishing gradient problem. However, MHSA is not considered routing and does not intend to implement a parse-tree. Considering space and time complexity, we observe that RBA is extremely expensive compared to MHSA; see Table 3. This fact may contribute to the substantial interest in models relying on MHSA rather than RBA.

6 Broader Impact and Limitations

In this work, we focus on the original CapsNet with RBA routing since it is the predominant implementation of the capsule idea. An exhaustive investigation, including all capsule variants and follow-up models, is difficult because the absence of a formal definition of capsules makes the topic hard to cover. Different approaches from the vast literature are technically diverse. As a result, whether a follow-up work implements the concept of capsules is not easy to judge. However, our claims are general enough to cover many implementations. The softmax-based routing approach is part of many capsule implementations, see for instance (Xiang et al. 2018; Zhou et al. 2019; Mazzia, Salvetti, and Chiaberge 2021), and we expect that they face similar issues.

	Space	Time
MHSA	$O(d^2)$	$O(n^2 \cdot d + n \cdot d^2)$
RBA	$O(n^2 \cdot d^2)$	$O(n^2 \cdot d^2)$

Table 3: Comparing space and time complexity of routing-by-agreement and multi-head self-attention for a routing layer with n input and output vectors of dimension d.

7 Conclusion

The core concept of capsules is the part-whole hierarchy of an image represented by a parse-tree. While this concept has appealing properties like robustness under affine transformations, interpretability, and parameter efficiency, so far, implementations of the capsule concept have not taken over yet. Instead, some of their properties were questioned in recent work. Here, we have shown that the core idea of a parse-tree does not emerge in the CapsNet implementation. Furthermore, starvation of capsules caused by a vanishing gradient limits their capacity and depth. Our observations explain recently reported shortcomings, especially that CapsNets do not scale beyond small datasets. Hence, the CapsNet is not a sufficient implementation of the capsule idea.

Acknowledgments

This work was supported by the German Science Foundation (DFG) grant (GI-711/5-1) within the priority program (SPP 1736) Algorithms for Big Data and by the Carl Zeiss Foundation within the project "Interactive Inference".

References

Ahmed, K.; and Torresani, L. 2019. STAR-Caps: Capsule Networks with Straight-Through Attentive Routing. In *Advances in Neural Information Processing Systems, (NeurIPS)*.

Biederman, I. 1987. Recognition-by-components: A Theory of Human Image Understanding. *Psychological review*, 94(2): 115.

Chen, J.; Yu, H.; Qian, C.; Chen, D. Z.; and Wu, J. 2021. A Receptor Skeleton for Capsule Neural Networks. In Meila, M.; and Zhang, T., eds., *International Conference on Machine Learning, (ICML)*.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A Large-scale Hierarchical Image Database. In *Conference on Computer Vision and Pattern Recognition, (CVPR)*.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations, (ICLR)*.

Gu, J. 2021. Interpretable Graph Capsule Networks for Object Recognition. In *Conference on Artificial Intelligence, (AAAI)*.

Gu, J.; and Tresp, V. 2020. Improving the Robustness of Capsule Networks to Image Affine Transformations. In *Conference on Computer Vision and Pattern Recognition, (CVPR)*.

Gu, J.; Tresp, V.; and Hu, H. 2021. Capsule Network Is Not More Robust Than Convolutional Network. In *Conference on Computer Vision and Pattern Recognition, (CVPR)*.

Gu, J.; Wu, B.; and Tresp, V. 2021. Effective and Efficient Vote Attack on Capsule Networks. In *International Conference on Learning Representations, (ICLR)*.

Hahn, T.; Pyeon, M.; and Kim, G. 2019. Self-Routing Capsule Networks. In *Advances in Neural Information Processing Systems, (NeurIPS)*.

Hinton, G. E. 1979. Some Demonstrations of the Effects of Structural Descriptions in Mental Imagery. *Cognitive Science*, 3(3): 231–250.

Hinton, G. E. 2021. How to represent part-whole hierarchies in a neural network. *arXiv*, abs/2102.12627.

Hinton, G. E.; Ghahramani, Z.; and Teh, Y. W. 1999. Learning to Parse Images. In *Advances in Neural Information Processing Systems, (NIPS)*.

Hinton, G. E.; Krizhevsky, A.; and Wang, S. D. 2011. Transforming Auto-Encoders. In *International Conference on Artificial Neural Networks, (ICANN)*.

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

Kosiorek, A. R.; Sabour, S.; Teh, Y. W.; and Hinton, G. E. 2019. Stacked Capsule Autoencoders. In *Advances in Neural Information Processing Systems, (NeurIPS)*.

Krizhevsky, A. 2009. Learning multiple layers of features from tiny images. Technical report, University of Toronto.

LeCun, Y.; Boser, B. E.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W. E.; and Jackel, L. D. 1989. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4): 541–551.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Mazzia, V.; Salvetti, F.; and Chiaberge, M. 2021. Efficient-CapsNet: capsule network with self-attention routing. *Scientific reports*, 11(1): 1–13.

Michels, F.; Uelwer, T.; Upschulte, E.; and Harmeling, S. 2019. On the Vulnerability of Capsule Networks to Adversarial Attacks. *arXiv*, abs/1906.03612.

Paik, I.; Kwak, T.; and Kim, I. 2019. Capsule Networks Need an Improved Routing Algorithm. In *Asian Conference on Machine Learning, (ACML)*.

Pham, H.; Dai, Z.; Xie, Q.; and Le, Q. V. 2021. Meta Pseudo Labels. In *Conference on Computer Vision and Pattern Recognition, (CVPR)*.

Rajasegaran, J.; Jayasundara, V.; Jayasekara, S.; Jayasekara, H.; Seneviratne, S.; and Rodrigo, R. 2019. DeepCaps: Going Deeper With Capsule Networks. In *Conference on Computer Vision and Pattern Recognition, (CVPR)*.

Rawlinson, D.; Ahmed, A.; and Kowadlo, G. 2018. Sparse Unsupervised Capsules Generalize Better. *arXiv*, abs/1804.06094.

Ribeiro, F. D. S.; Leontidis, G.; and Kollias, S. D. 2020. Capsule Routing via Variational Bayes. In *Conference on Artificial Intelligence, (AAAI)*.

Sabour, S.; Frosst, N.; and Hinton, G. E. 2017. Dynamic Routing Between Capsules. In *Advances in Neural Information Processing Systems, (NIPS)*.

Sabour, S.; Tagliasacchi, A.; Yazdani, S.; Hinton, G. E.; and Fleet, D. J. 2021. Unsupervised Part Representation by Flow Capsules. In *International Conference on Machine Learning, (ICML)*.

Tsai, Y. H.; Srivastava, N.; Goh, H.; and Salakhutdinov, R. 2020. Capsules with Inverted Dot-Product Attention Routing. In *International Conference on Learning Representations, (ICLR)*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems, (NIPS)*.

Wang, D.; and Liu, Q. 2018. An Optimization View on Dynamic Routing Between Capsules. In *International Conference on Learning Representations, (ICLR)*.

Wortsman, M.; Ilharco, G.; Gadre, S. Y.; Roelofs, R.; Lopes, R. G.; Morcos, A. S.; Namkoong, H.; Farhadi, A.; Carmon, Y.; Kornblith, S.; and Schmidt, L. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning, (ICML)*.

Xi, E.; Bing, S.; and Jin, Y. 2017. Capsule Network Performance on Complex Data. *arXiv*, abs/1712.03480.

Xiang, C.; Zhang, L.; Tang, Y.; Zou, W.; and Xu, C. 2018. MS-CapsNet: A Novel Multi-Scale Capsule Network. *IEEE Signal Processing Letters*, 25(12): 1850–1854.

Zhou, Y.; Ji, R.; Su, J.; Sun, X.; and Chen, W. 2019. Dynamic Capsule Attention for Visual Question Answering. In *Conference on Artificial Intelligence, (AAAI)*.