Survey on Capsule Network's Depth Scalability and Learned Feature Spatial Relationships Retention

Ugenteraan Manogaran [⋆], Ya Ping Wong [⋆⋆], and Boon Yian Ng

Faculty of Computing and Informatics, Multimedia University, Persiaran Multimedia, 63100 Cyberjaya, Selangor, Malaysia

Abstract. In 2017 the first working architecture of capsule networks called Capsule Network with Dynamic Routing (CapsNet) was introduced and shown to perform better than Convolutional Neural Networks (CNNs) in the MNIST dataset. It was also speculated that capsule networks' ability to retain spatial relationships among learned features are better than CNNs. Ever since then, much research has been devoted to improving capsule network architectures. However, most proposed architectures are relatively shallow and not scalable depth-wise. Also, some works have shown that the said retention ability is not as good as speculated. In this survey, we reviewed research works on capsule networks that focus on depth scaling and the said retention ability. We found that the routing algorithm used in a capsule network plays a crucial role in both the depth scalability and the said retention ability. We believe that future research on capsule networks should be focused on improving the routing algorithm to ensure the success of capsule networks.

Keywords: CapsNet, Convolutional Neural Network, Spatial Relationships, Depth Scalability

1 Introduction

A Convolutional Neural Network (CNN) is a class of neural networks that is widely used for computer vision tasks. CNN was first introduced by LeCun, Y. et. al. [1] in 1989 but it was not a popular choice of approach back then due to the lack of computational resources and dataset availability. Only after Krizhevsky, A. et. al [2] showed that their CNN based approach was able to achieve a state-of-the-art result in the ImageNet challenge [3] in 2012, CNN gained tremendous popularity among both the research community and engineers.

Since then, many variants of CNN architecture have been built and used in countless computer vision tasks such as EfficientNet [4] in image classification, EfficientDet [5] in object detection, and Hierarchical Multi-Scale Attention [6] in semantic segmentation where they have achieved state-of-the-art results. CNNs

⋆ E-mail : m.ugenteraan15@gmail.com

⋆⋆ E-mail : ypwong@mmu.edu.my

are also widely used in real-world applications such as self-driving cars [7], speech recognition [8], and human crowd detection [9].

Unfortunately, CNN based models have two major pitfalls which are usually remedied by the use of massive amounts of training data. The first problem with CNN based models is the lack of rotational invariance [10]. In simpler words, if a feature that is known by a model is rotated to a certain extent, the model would fail to identify the feature. This causes false negative results by the model. The second problem is the failure to retain spatial relationships among features [11]. This means that features of an object in an image arranged in the relative positions that no longer resemble the original object may still cause the object to be identified positively as the original object by a CNN model.

To solve these problems, a novel neural network architecture named Capsule Network with Dynamic Routing (CapsNet) was introduced in 2017 by Sabour, S. et al. [12]. Capsule network is a relatively new class of neural network that produces outputs in the form of vectors instead of scalars as in the case of CNNs. These vectors are known as capsules. The authors were able to achieve new state-of-the-art results on MNIST at the time of its introduction. Capsule networks were also speculated to be able to retain the spatial relationships among learned features better than CNNs [13].

Since then, much research has been devoted to developing variants of capsule network architecture further for real-world applications. LaLonde, R. et. al. [14] developed a capsule based novel architecture known as SegCaps to perform image segmentation on the LUNA16 subset of the LIDC-IDRI database. It achieved a new state-of-the-art result even though SegCaps had a lower number of parameters compared to previous methods. Zhao, W. et. al. [15] introduced a scalable new framework that is built on top of existing capsule networks for natural language processing applications. The authors reported that their framework has outperformed other strong baseline architectures on multi-label text classification and question answering. More recently, Sun, C. et. al. [16] developed a novel BERT-based attention-guided capsule network for chemical-protein interaction extraction which has been experimentally proved to be superior to other state-of-the-art models on the CHEMPROT corpus.

However, most of the capsule network architectures that exist today are relatively shallow. Furthermore, most research such as the SegCaps [14] has been built based on the assumption that capsule networks retain the spatial relationships among learned features.

In this paper, we will survey and analyse research papers that have attempted to scale capsule network based models to relatively deep architectures and papers that have empirically analysed or improved the capsule network's ability to retain the spatial relationships among learned features.

2 CapsNet Compared to a Traditional CNN

2.1 Introduction

Although the first working architecture of capsule networks was introduced in 2017 [12], the idea of using capsules in a neural network architecture was first proposed by Hinton, G. et. al. [13] in 2012. The authors argued that the use of scalars in CNNs is the cause for CNNs not being able to retain the variations in position, scaling, orientation and lighting. Therefore, they proposed the idea of using vectors as outputs instead.

In addition to the first argument, the authors have also argued that the use of pooling layers in CNN architectures for subsampling causes CNNs to be unable to retain spatial relationships among learned features. In view of this, capsule networks do not use pooling layers to subsample its outputs. Instead, a routing algorithm is commonly used in capsule networks. In the proposed CapsNet architecture [12], a routing algorithm known as dynamic routing was used.

2.2 A Traditional CNN

A traditional convolutional neural network architecture consists of convolutional layers, activation functions, pooling layers, and fully-connected layers. Figure 1 below shows an example of a schematic representation of a relatively simple traditional CNN.

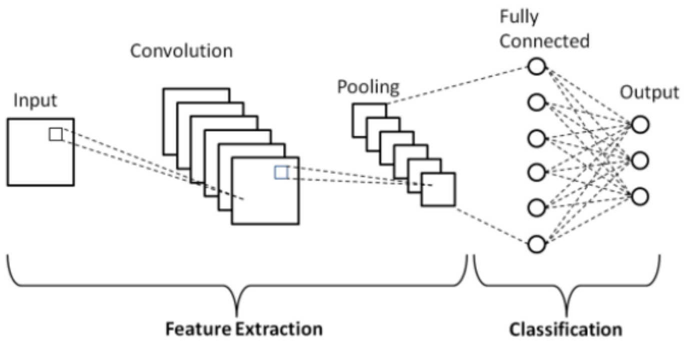

Fig. 1. Schematic diagram of a simple CNN architecture. [17].

2.2.1 Convolutional Layers

A convolutional layer consists of a fixed number of kernels where the size of each kernel is m x n. The kernels slide across the input image based on a set stride and perform element-wise multiplication with the overlapped region and sum the results. This operation can be viewed as a linear combination between the kernel and the overlapped region. For example, figure 2 below shows a 3 x 3 kernel consisting of arbitrary values.

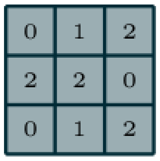

Fig. 2. A 3 x 3 kernel [18].

If the kernel above is used in a convolutional operation with a 5 x 5 input image, the operation can be illustrated as figure 3 below.

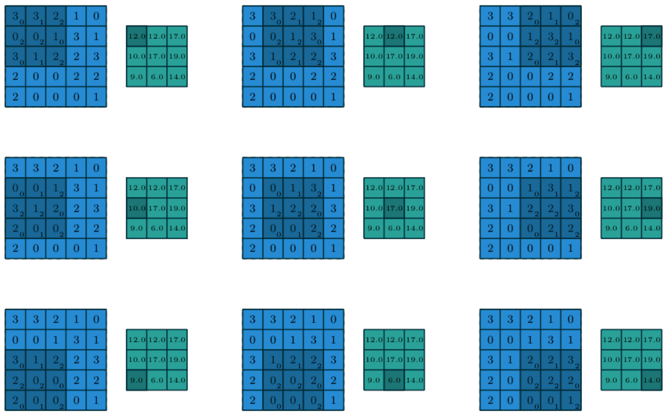

Fig. 3. Convolution operation between a 3 x 3 kernel and a 5 x 5 image [18].

The green 3 x 3 region is the result of the convolutional operation. The resulting region is known as a feature map. The feature map will then become an input to the next convolutional layer. The aim of the kernels is to extract learned patterns or features in the given input image or feature maps.

2.2.2 Activation Functions

Activation functions are functions that add non-linearity properties into the network. They are applied to the output of a layer in the network before the output is used as an input to the next layer. There are two main reasons for the use of the activation functions. First, most real-world data are non-linear by nature. Therefore, in order to extract information from the data efficiently, the neural network must have non-linearity properties. Secondly, neural networks have to be differentiable for optimization purposes. Adding activation functions into a neural network makes the network differentiable.

Common activation functions used in a neural network are Swish [19], Rectified Linear Unit (ReLU) [20], Sigmoid [21], and Hyperbolic Tangent (Tanh) [21], with ReLU being the most common. ReLU is defined as shown below in (1).

$$f(x) = x^{+} = \max(0, x) \tag{1}$$

Although the function appears to be simple, it delivers consistent results and is usually more effective than all the other activation functions. Hence, ReLU became the default choice of activation function [19].

2.2.3 Pooling Layers

Synonymous to convolution layers, pooling layers consist of a fixed number of same-sized pooling windows. However, in contrast to convolutional kernels, the purpose of these pooling windows is to subsample a given feature map. Normally the subsampling is done through either choosing the maximum value from the overlapped region or averaging all the values in the overlapped region. The pooling process significantly reduces the size of the feature maps while still retaining important information from the feature maps.

2.2.4 Fully-Connected Layers

The feature maps from either the final convolutional layer or the final pooling layer are first transformed into a single vector. This vector is known as a feature vector which will be used as an input to a fully-connected layer. Fully-connected layers are actually traditional neural networks or better known as multi-layer perceptrons.

Every neuron in a given layer is connected to every other neuron in the next layer. The last fully-connected layer has the number of neurons equal to the number of classes to be predicted. For example, if the purpose of a CNN is to predict whether a given image is an image of a dog or an image of a cat, then the last fully-connected layer of the CNN would have 2 neurons since there are only 2 classes to predict. Each neuron outputs a number that is treated as a probability. The neuron with the highest value represents the class of the given input image.

2.3 CapsNet

Since CapsNet is the first architecture of capsule networks, it is best to understand its building blocks so that better capsule network architectures can be built. CapsNet architecture consists of convolutional layers, primary capsule layer, activation function, dynamic routing, and digit capsules. Note that there are no pooling layers in CapsNet.

2.3.1 Convolutional Layers

The convolutional layers used in CapsNet are exactly how they are in a traditional CNN.

2.3.2 Primary Capsule Layer

The feature vector produced by the final convolutional layer is broken down into equal parts. Now, instead of a single high dimensional vector, there are now many lower dimensional vectors. These lower dimensional vectors are known as capsules and the layer where this process takes place is known as the primary capsule layer.

2.3.3 Activation Function

Since capsules are vectors, the activation functions that are used in CNN or traditional neural networks would be ineffective as these functions are meant for scalars. The idea of capsule networks is to treat each capsule as a single unit. Therefore a novel activation function known as squashing function was introduced in CapsNet as shown below in (2).

$$\mathbf{v}_j = \frac{||\mathbf{s}_j||^2}{1+||\mathbf{s}_j||^2}\frac{\mathbf{s}_j}{||\mathbf{s}_j||} \tag{2}$$

$\mathbf{v}_j$ is the vector output of capsule j and $\mathbf{s}_j$ is its total input. The squashing function ensures that the magnitude of a capsule is scaled to a value between 0 and 1. Furthermore, the squashing function also introduces non-linearity to the network.

2.3.4 Digit Capsules and Dynamic Routing

Since the idea of capsules is to represent the features present in a given input image, naturally the capsules in the deeper part of the network have to represent higher-level features. Therefore the capsules in the primary capsule layer are transformed into higher dimensional capsules before being used as an input in the digit capsule layer. The transformation is done through the use of a weight matrix, $\mathbf{W}_{ij}$ on a capsule in the primary capsule layer, $\mathbf{u}_i$ as shown in (3) below. A transformed capsule, $\mathbf{u}_{j|i}$ is known as a prediction vector. The letter i represents the i^{th} element in a capsule and the i^{th} column in the weight matrix while the letter j represents the j^{th} row in the weight matrix.

$$\hat{\mathbf{u}}_{j|i} = \mathbf{W}_{ij}\mathbf{u}_i \tag{3}$$

Digit capsule layer should have a number of capsules that corresponds to the number of classes to be predicted. Therefore, the prediction vectors have to be somehow routed to produce the required number of capsules for the digit capsule layer. A naive way to route is by using a pooling layer like in a CNN. However, Sabour, S. et al. [12] argued that in order for lower-level capsules to contribute to the right higher-level capsules, the prediction vectors should be routed to the capsules in the digit capsule layer based on how strong both the capsules and the prediction vectors "agree" with their predictions. The "agreement" is determined through the use of an iterative dot product operation. This process is known as dynamic routing.

In the first iteration, every prediction vector contributes equally to the capsules in the digit capsule layer. If $\mathbf{s}_j$ is a capsule in the digit capsule layer, where j is a value from 1 to the total number of capsules in the digit capsule layer, then the capsule is calculated by the equation shown in (4) below.

$$\mathbf{s}_j = \sum_i c_{ij}\hat{\mathbf{u}}_{j|i} \qquad \textit{where} \qquad c_{ij} = \frac{e^{b_{ij}}}{\sum_k e^{b_{ik}}} \tag{4}$$

All b_{ij} are set to 0 on the first iteration to ensure that c_{ij} are the same value for all the prediction vectors. In the following iterations, dot product operation is used between the prediction vectors and the capsules in the digit capsule layer to represent the agreement between them. The higher the value of the dot product, the higher the agreement between them. The operation is defined by the equation shown in (5) below.

$$a_{ij} = \mathbf{v}_j \cdot \hat{\mathbf{u}}_{j|i} \tag{5}$$

Every a_{ij} is then added to its corresponding b_{ij}. Once the total sum of b_{ij} is calculated, c_{ij} are then calculated by applying a softmax function as shown in the (4) above where k is a value from 1 to the total number of capsules in the digit capsule layer. The updated values of c_{ij} are then used as coefficients to route the prediction vectors to the digit capsule layer. The higher the coefficient on a certain prediction vector, the more the contribution of the prediction vector will be. The iterative process in the proposed CapsNet paper [12] was done for a total of 3 times. Finally, the magnitude of the capsules in the digit capsule layer is used as a probability to represent the class of the input image.

2.4 Advantages of CapsNet over Traditional CNNs

Since capsule networks use vectors to represent features unlike a convolutional neural network, capsule networks have the ability to retain the variations in the features' position, orientation, scaling, and lighting. This property has been demonstrated through the use of a reconstruction network in the CapsNet paper [12]. The reconstruction network is a traditional neural network that takes the predicted capsule from the digit capsule layer and uses the capsule as an input to reconstruct the original input image.

The reconstruction network is trained end-to-end in the CapsNet architecture. It acts as a regularizer for CapsNet and forces the capsules in primary capsule layer and digit capsule layer to learn the features' variations. figure 4 below shows the output from a reconstruction network from a CapsNet model that was trained using MNIST dataset along with the original input image.

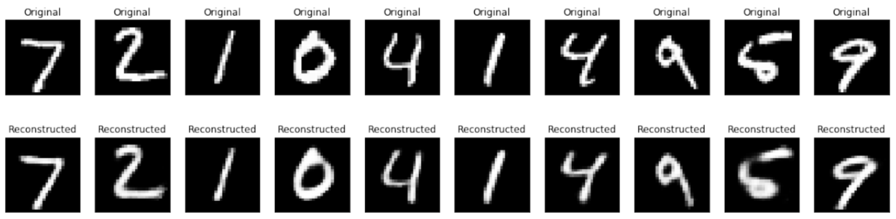

Fig. 4. Original input images (top row) vs. the reconstructed images (bottom row).

It is evident that CapsNet preserves the properties of the features relatively well. It is theorized that CapsNet is able to retain spatial relationships among learned features in contrast to a CNN. This is due to the use of dynamic routing. However, even without this ability, CNN based models are still dominating the computer vision field. The primary reason for the success of CNNs is their ability to use multiple feature extraction stages to learn relatively complex representations from input dataset [22]. Although CapsNet has been demonstrated to achieve state-of-the-art performance on MNIST dataset, its success on larger, and more complex dataset clearly depends on its ability to scale its depths and its effectiveness on retaining the spatial relationships among learned features.

3 Survey On Recent Capsule Network Implementations

3.1 Depth Scalability

After the successful demonstration of CapsNet on MNIST dataset [12], the next natural step is to test the performance of the model on more complex datasets such as Cifar10. However, since CapsNet is a relatively shallow model, it needs to be scaled to a deeper architecture in order to be used on more complex datasets. A naive approach to deepen the architecture would be to stack the hidden layers of CapsNet. Unfortunately, this approach has been demonstrated to perform relatively poorly [10]. This is caused by the dynamic routing. Since dynamic routing ensures the coupling coefficients add up to 1 in total, when there are too many capsules in a layer, the gradient flow gets dampened during backpropagation which results in poor learning in the layer [23].

Much research has been done on capsule networks since 2017 such as the work of LaLonde, R. et. al. [14], Mobiny, A., Van Nguyen, H. [24], Zhang, W. et. al. [25], Gu, J., Tresp, V. [26], Zhao, T. et. al. [27], and Xiang, C. et. al [28]. However, the architectures of the proposed capsule networks are still relatively shallow and are meant for only a specific application. There were no mentions of depth scalability as well. Only a handful of research has been done on the scaling of capsule network architectures depth-wise such as the work of Rajasegaran, J. et. al. [23], Venkatraman, S. et. al. [29], and Peer, D. et. al. [30].

Rajasegaran, J. et. al. [23] proposed to deepen the architecture by using ConvCaps layers. A ConvCaps layer may consist of either a 2D convolutional layer known as 2D ConvCaps layer or a 3D convolutional layer known as a 3D ConvCaps layer. A 2D ConvCaps layer is similar to a convolutional layer in a traditional CNN with the exception that the outputs are squashed to capsule forms [23]. The original dynamic routing algorithm [12] works in a fully-connected manner. This means that every capsule will be transformed using a weight matrix to prediction vectors and each of the prediction vectors will be routed to the next layer of capsules. In this work [23], dynamic routing is replaced with 3D dynamic routing which is implemented in the 3D ConvCaps layers. Since all capsules in ConvCaps layers are produced through convolutional operations, adjacent capsules would contain similar information. Therefore, Rajasegaran, J. et. al. [23] argue that routing all capsules from a lower-level layer to a higher-level layer is redundant [23]. In view of this, 3D dynamic routing only routes a part of every capsule to a higher-level layer.

Furthermore, skip connections [31] are utilized to further improve the gradient flow during backpropagation among the layers in DeepCaps. It has been reported [23] that DeepCaps surpassed the performance of other capsule network architectures in few of the datasets as shown in Table 1 below [3].

Model	CIFAR10	SVHN	F-MNIST	MNIST
DenseNet [34]	96.40%	98.41%	95.40%	-
ResNet [31]	96.57%	-	-	99.59%
DPN [35]	96.35%	-	95.70%	-
Wan et al. [36]	-	-	-	99.79%
Zhong et al. [37]	96.92%	-	96.35%	-
Sabour et al. [12]	89.40%	95.70%	93.60%	**99.75%**
Nair et al. [38]	67.53%	91.06%	89.80%	99.50%
HitNet [39]	73.30%	94.50%	92.30%	99.68%
DeepCaps [23]	91.01%	97.16%	94.46%	99.72%
DeepCaps (7-ensembles) [23]	**92.75%**	**97.56%**	**94.73%**	-

Table 1. Results of DeepCaps [23].

We have reproduced the work of Rajasegaran, J. et. al. [23] using Pytorch[1] to verify the claims made in the paper. To the best of our knowledge, this is the first work that focuses on the depth scalability of capsule network architectures. In our implementation, we achieved an accuracy of 88% on the FashionMNIST dataset contrary to the 94.73% accuracy reported in the paper. Due to our training being done on a relatively less powerful workstation with 16 GB of RAM, a GTX 1080Ti and i7 6th Gen processor, performing trial and error with different hyperparameters to achieve the reported accuracy is an extensive process as it takes a massive amount of time to complete even 1 epoch of training. However, we noted that it is possible to achieve a higher accuracy than 88% as the loss was still steadily decreasing even at our final epoch (1000th epoch).

In another research by Venkatraman, S. et. al [29], a group-based equivariant capsule network architecture known as Space-of-Variation network (SovNet) was proposed. In this paper, the authors argued that the way the primary capsules are constructed in a capsule network plays an important role in the performance of the entire network. This was argued based on the observation done by Do Rosario, V. M. et. al. [32] in their work where they divided the primary capsule layer into independent sets which are known as lanes. Each lane represents a set of operations that produces a predetermined number of primary capsules. Primary capsules from each lane then contribute to one of the dimensions in the final digit capsule layer. It has been noted in this work that the performance of the network differs significantly by changing the configurations of a lane such as the size of lanes, and the number of convolutional kernels.

Motivated by the work of Do Rosario, V. M. et. al. [32], Venkatraman, S. et. al [29] used group-equivariant convolutional [33] process to construct the primary capsules in their architecture. This is to ensure that the primary capsule layer is robust to translations and rotations of the input images. A novel routing algorithm known as degree-centrality based routing was also introduced in this work along with a proof that shows that this routing algorithm is equivariant. In addition to these, the authors have also shown that the depth of the SovNet can be scaled. The performance of SovNet was compared with DeepCaps and other capsule network architectures and the results are as shown in Table 2 below.

From the result of the comparison as shown in Table (2), it is evident that SovNet performs better than DeepCaps and other capsule network architectures in most cases. The superior performance might be owed to the way the primary capsules are constructed and the focus on equivariance.

In contrast to these two approaches, Peer, D. et. al. [30] solely focused on improving the routing algorithm to achieve depth scalability in capsule networks. In this research, the authors have shown that the original dynamic routing [12] does not achieve the objective of forming a parse tree in the network. The authors defined two metrics to measure the properties of the dynamic routing with its objective in mind. The authors argued that if a capsule network's objective is to form a parse tree structure, then the dynamic routing algorithm should satisfy these two properties :

[1] https://github.com/MMU-VisionLab/DeepCaps

Training Results on Untransformed FashionMNIST					
Method	$(0, 0°)$	$(2, 30°)$	$(2, 60°)$	$(2, 90°)$	$(2, 180°)$
CapsNet	91.23%	57.15%	37.98%	28.33%	22.38%
EMcaps	90.05%	59.75%	40.26%	30.17%	23.82%
G-Caps	86.56%	50.05%	35.05%	29.93%	27.10%
DeepCaps	93.27%	57.82%	37.06%	27.63%	21.86%
SOVNET	**94.72%**	**61.58%**	**41.01%**	**34.07%**	**27.63%**

Training Results on FashionMNIST Transformed by $(2, 30°)$					
Method	$(0, 0°)$	$(2, 30°)$	$(2, 60°)$	$(2, 90°)$	$(2, 180°)$
CapsNet	91.22%	89.57%	69.58%	50.17%	35.16%
EMcaps	90.17%	89.47%	68.39%	49.23%	37.02%
G-Caps	83.28%	80.12%	64.86%	53.71%	**52.54%**
DeepCaps	93.71%	93.40%	75.32%	53.35%	36.30%
SOVNET	**94.99%**	**94.36%**	**77.19%**	**58.59%**	**43.84%**

Training Results on FashionMNIST Transformed by $(2, 60°)$					
Method	$(0, 0°)$	$(2, 30°)$	$(2, 60°)$	$(2, 90°)$	$(2, 180°)$
CapsNet	89.98%	88.55%	88.15%	72.81%	46.89%
EMcaps	88.24%	87.30%	87.04%	71.72%	48.14%
G-Caps	82.04%	80.12%	78.94%	68.05%	59.25%
DeepCaps	93.36%	93.06%	92.84%	80.76%	49.90%
SOVNET	**94.49%**	**94.08%**	**94.20%**	**90.23%**	**73.48%**

Training Results on FashionMNIST Transformed by $(2, 90°)$					
Method	$(0, 0°)$	$(2, 30°)$	$(2, 60°)$	$(2, 90°)$	$(2, 180°)$
CapsNet	88.78%	87.18%	87.13%	86.19%	59.59%
EMcaps	86.43%	85.85%	85.82%	85.63%	61.15%
G-Caps	80.71%	79.55%	79.17%	79.21%	72.11%
DeepCaps	93.07%	92.93%	92.75%	92.51%	62.50%
SOVNET	**94.41%**	**94.03%**	**93.93%**	**93.98%**	**91.42%**

Training Results on FashionMNIST Transformed by $(2, 180°)$					
Method	$(0, 0°)$	$(2, 30°)$	$(2, 60°)$	$(2, 90°)$	$(2, 180°)$
CapsNet	86.90%	84.94%	84.93%	84.75%	84.72%
EMcaps	82.99%	82.67%	82.18%	82.32%	82.18%
G-Caps	80.65%	79.66%	79.46%	79.47%	79.37%
DeepCaps	92.07%	91.71%	91.70%	91.76%	91.66%
SOVNET	**94.11%**	**93.77%**	**93.56%**	**93.57%**	**93.60%**

Table 2. Performance Comparison on SovNet [29]

1. The relationship between a capsule layer and the next capsule layer should be many to one. That is, the capsules should represent a node in a parse tree.
2. The parent capsule should be dynamic. In other words, the parent capsule must change when the input to the capsule changes.

The authors proposed two metrics to measure these properties and evaluated the original dynamic routing [12] algorithm. The authors have found that the original dynamic routing algorithm does not ensure the forming of a parse tree in the network. All of the capsules from the lower level layer send their output almost equally to capsules in the following layer. This finding is consistent with the observation made by Rajasegaran, J. et. al. [23] where the authors stated that the gradient flow during backpropagation among the capsules gets dampened when there are too many capsules. If dynamic routing performs a uniform coupling among the capsules as reported by [30], then due to the use of softmax operation in dynamic routing, the coefficient of each capsule would be close to 0. This in turn would dampen the gradient flow among the capsules during backpropagation as theorized by Rajasegaran, J. et. al. [23].

To remedy this problem, Peer, D. et. al. [30] have proposed a novel deep dynamic routing algorithm which has been shown to perform better when evaluated with the metrics mentioned earlier. The deep dynamic routing also produces a dynamic parse tree network. As a result, the entire network can be scaled deeper when the proposed deep dynamic routing is used instead of the original dynamic routing [12]. In this paper, the authors compared a CapsNet implementation with original dynamic routing [12] and a CapsNet implementation with deep dynamic routing on the MNIST dataset. Both of the implementations had four hidden layers. The result was that the accuracy for the former implementation did not go above 10% with several different learning rates with 3 routing iterations. However for the latter implementation, the test accuracy increased to 87% with a learning rate of 0.001.

Despite all these different attempts to make capsule network architectures scalable in depth, CNN architectures still dominate the field of computer vision. However, note that among the capsule network architectures, the deeper architectures have been shown to perform better compared to the shallow architectures [23], [29].

3.2 Retention Of Spatial Relationships Among Learned Features

CapsNet [12] has been speculated to be better at retaining the spatial relationships among the learned features compared to a CNN [12], [28], [40]. The use of pooling layers in a deep CNN model is thought to be the cause for the loss of spatial information among the learned features [13]. Hence a capsule network architecture does not make use of any pooling layer at all. Instead, it uses a routing algorithm to ensure the relationship between the learned features are retained.

Venkatraman, S. R. et. al. [40] have shown experimentally in their research that a capsule network performs badly at detecting changes in the relationship among learned features as the entropy of the routing weight increases. Unfortunately, the original dynamic routing algorithm [12] has been reported [40] to have a high routing weight entropy and therefore not effective at retaining the spatial relationships among learned features. This correlates with our previous research [41] where we empirically proved that the said retention ability of CapsNet is not as good as a CNN. Venkatraman, S. R. et. al. [40] believe that in order to improve the ability of a capsule network on the retention of spatial relationships among learned features, routing algorithms that have low routing weight entropy have to be developed.

Recently, Sabour, S. et. al. [42] introduced a capsule network architecture called FlowCapsules. This work is focused on improving the capsule network's ability to retain the spatial relationships among learned features. Sabour, S. et. al. [42] introduced an unsupervised method to represent the parts of the objects present in the input image in the primary capsule layer. This layer is constructed based on a principle that the regions that move together often belong together. During the training of the model, the primary capsule layer takes a pair of consecutive frames as its input to learn to parse an object into parts. During inference, the primary capsule layer takes a single image as its input to detect and encode the moveable object parts in an image. It has been reported that FlowCapsules is relatively effective in learning meaningful representations and completing shapes even when partially occluded on both real and synthetic data [42].

3.3 Summary Of The Surveys

In this section, the main idea of each paper surveyed in the two previous subsections will be summarised in Table 3 and Table 4 below.

3.4 Discussion And Recommendation

Even though the first working architecture of the capsule network, CapsNet [12], outperformed CNN in the MNIST dataset, that does not mean that capsule networks are suitable for solving real world problems. Due to MNIST being a relatively simple problem to solve, a shallow capsule network architecture was sufficient. However, real world problems are commonly much more complex than MNIST. Raghu, M. et. al. [43] have shown that a neural network's capability to compute complex functions grows exponentially with the network's depth. Therefore, we argue that for capsule network based models to be applicable in real world situations, the models must have the ability to scale its depth with ease like CNN based models.

As pointed out earlier, many proposed capsule network architectures are designed to be shallow as they are focused to solve specific problems [14], [24], [25], [26], [27], [28]. These approaches may be able to improve the performance of capsule networks on specific problems and may even achieve state-of-the-art

Depth Scalability	
Title of the surveyed paper	**Main idea or contribution of the paper**
DeepCaps:Going Deeper with Capsule Networks [23]	Scaling the depth of capsule network architecture by introducing ConvCaps and by only routing a part of each lower-level capsule to higher-level capsule. Experimental results showed that DeepCaps is superior to other relatively shallow capsule network architectures.
Building Deep Equivariant Capsule Networks [29]	Ensures that the equivariant property of capsule networks are preserved through the use of group-equivariant convolutional process and introducing a routing algorithm that ensures the equivariant property is preserved. These modifications also allow the architecture to be scaled deeper. The experimental results showed that the proposed network outperforms even DeepCaps in most cases.
Training Deep Capsule Networks [30]	Revisited the objectives of a routing algorithm in capsule networks and proposed two metrics to measure the effectiveness of a routing algorithm in achieving their objectives. Also introduced a novel routing algorithm that allows for the depth scaling of a capsule network. Experimental results showed that a CapsNet [12] with the proposed routing performed better than a Capsnet with the original dynamic routing [12] on the same dataset. Both CapsNet implementations here had 4 hidden layers.

Table 3. Summary of the surveys on capsule network's depth scalability

Depth Scalability	
Title of the surveyed paper	**Main idea or contribution of the paper**
Learning Compositional Structures for Deep Learning: Why Routing-by-agreement is Necessary [40]	Experimentally shown that the ability of a capsule network to retain the spatial relationships among learned features gets better as the entropy of the routing weight decreases. Also shown that the original CapsNet's dynamic routing has a high routing weight entropy.
CapsNet vs CNN : Analysis Of The Effects Of Varying Feature Spatial Arrangement [41]	Empirically shown that the performance of CapsNet on retaining the spatial relationships among learned features are not as effective as speculated.
Unsupervised part representation by Flow Capsules [42]	Introduced a primary capsule layer that is able to learn to recognize parts of an object in an unsupervised manner. Experiments showed that the learning of object parts in the proposed architecture is effective in a variety of situations.

Table 4. Summary of the surveys on capsule network's ability to retain spatial relationships among learned features.

results. However they do not improve the capability of capsule networks to be applicable across many different domains like CNNs.

On the other hand, works such as that of [23], [29], and [30] would bring a bigger impact to the capsule networks field in the long run. It is also worth noting that one of the main advantages of capsule networks over CNNs is their ability to retain the spatial relationships among learned features. Therefore, we also add in our argument that the direction of research on capsule networks has to be focused on both the depth scalability of capsule network architectures and their said retention ability.

From the works that we have reviewed, especially the work of Peer, D. et. al. [30], it is evident that the original dynamic routing algorithm introduced in [12] prevents the capsule network to be scaled depth-wise by dampening the gradient flow among the capsules during backpropagation. Removing the routing algorithm altogether is not an option as it has been shown to be necessary to ensure the retention of spatial relationships among learned features in a capsule network [40]. Since the routing weight entropy influences the ability of a capsule network to retain the spatial relationships among learned features [40], the said ability of capsule networks can be improved by lowering the routing weight entropy.

Therefore, we believe that to improve the capsule network's ability on the retention of spatial relationships among learned features and its depth scalability, the most logical step to take is to modify the existing routing algorithms or to build a novel routing algorithm altogether. From the work of Peer, D. et. al. [30], we have learnt that in order to develop a routing algorithm that enables a capsule network to scale depth-wise, the algorithm must ensure the forming of a parse tree in a network. Also, from the work of Venkatraman, S. R. et. al. [40], it is noted that to improve a capsule network's ability on the retention of spatial relationships among learned features, the routing weight's entropy must have a low value. By developing a routing algorithm that adheres to these criteria, we believe that the performance of capsule networks can be significantly improved.

4 Conclusion

Many capsule networks related research so far have been focused on specific applications. Even though some of the works have achieved state-of-the-art results, the proposed architectures are not suitable for different applications unlike CNN architectures. This is due to the fact that the architectures are not scalable depth-wise. In this work, we have reviewed works on capsule networks that focus on the depth scalability and capsule networks' ability to retain spatial relationships among learned features. We believe that improving capsule network architectures based on these two criteria mentioned is pivotal in enabling the use of capsule networks on real world applications. During our reviews, we noticed that the routing algorithm plays a crucial role both in terms of depth scalability of the network and its ability in retaining the spatial relationships among learned features. Therefore, we conclude that modifying the existing routing algorithms or building a novel routing algorithm altogether is the key to the success of capsule networks.

5 Acknowledgement

The authors are grateful to the Ministry of Higher Education, Malaysia and Multimedia University for the financial support provided by the Fundamental Research Grant Scheme (FRGS/1/2015/ICT02/MMU/02/1) and MMU Internal Grant Scheme (MMUI/170110).

References

1. LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4), 541-551.
2. Krizhevsky, A., Sutskever, I., Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25, 1097-1105.
3. Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In IEEE Conference on Computer Vision and Pattern Recognition.
4. Tan, M., Le, Q. (2019, May). Efficientnet: Rethinking model scaling for convolutional neural networks. In International Conference on Machine Learning (pp. 6105-6114). PMLR.

5. Tan, M., Pang, R., Le, Q. V. (2020). Efficientdet: Scalable and efficient object detection. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 10781-10790).
6. Tao, A., Sapra, K., Catanzaro, B. (2020). Hierarchical multi-scale attention for semantic segmentation. arXiv preprint arXiv:2005.10821.
7. Codevilla, F., Müller, M., López, A., Koltun, V., Dosovitskiy, A. (2018, May). End-to-end driving via conditional imitation learning. In 2018 IEEE International Conference on Robotics and Automation (ICRA) (pp. 4693-4700). IEEE.
8. Hori, T., Watanabe, S., Zhang, Y., Chan, W. (2017). Advances in joint CTC-attention based end-to-end speech recognition with a deep CNN encoder and RNN-LM. arXiv preprint arXiv:1706.02737.
9. Tzelepi, M., Tefas, A. (2017). Human crowd detection for drone flight safety using convolutional neural networks. In 25th European Signal Processing Conference (EUSIPCO) (pp. 743-747). IEEE.
10. Xi, E., Bing, S., Jin, Y. (2017). Capsule network performance on complex data. arXiv preprint arXiv:1712.03480.
11. Algamdi, A. M., Sanchez, V., Li, C. T. (2019, May). Learning Temporal Information from Spatial Information Using CapsNets for Human Action Recognition. In ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 3867-3871). IEEE.
12. Sabour, S., Frosst, N., Hinton, G. E. (2017). Dynamic routing between capsules. arXiv preprint arXiv:1710.09829.
13. Hinton, G. E., Krizhevsky, A., Wang, S. D. (2011, June). Transforming auto-encoders. In International conference on artificial neural networks (pp. 44-51). Springer, Berlin, Heidelberg.
14. LaLonde, R., Bagci, U. (2018). Capsules for object segmentation. arXiv preprint arXiv:1804.04241.
15. Zhao, W., Peng, H., Eger, S., Cambria, E., Yang, M. (2019). Towards scalable and reliable capsule networks for challenging NLP applications. arXiv preprint arXiv:1906.02829.
16. Sun, C., Yang, Z., Wang, L., Zhang, Y., Lin, H., Wang, J. (2020). Attention guided capsule networks for chemical-protein interaction extraction. Journal of biomedical informatics, 103, 103392.
17. Phung, V. H., Rhee, E. J. (2019). A high-accuracy model average ensemble of convolutional neural networks for classification of cloud image patches on small datasets. Applied Sciences, 9(21), 4500.
18. Dumoulin, V., Visin, F. (2016). A guide to convolution arithmetic for deep learning. arXiv preprint arXiv:1603.07285.
19. Ramachandran, P., Zoph, B., Le, Q. V. (2017). Searching for activation functions. arXiv preprint arXiv:1710.05941.
20. Nair, V., Hinton, G. E. (2010, January). Rectified linear units improve restricted boltzmann machines. In Icml.
21. Nwankpa, C., Ijomah, W., Gachagan, A., Marshall, S. (2018). Activation functions: Comparison of trends in practice and research for deep learning. arXiv preprint arXiv:1811.03378.
22. Khan, A., Sohail, A., Zahoora, U., Qureshi, A. S. (2020). A survey of the recent architectures of deep convolutional neural networks. Artificial Intelligence Review, 53(8), 5455-5516.
23. Rajasegaran, J., Jayasundara, V., Jayasekara, S., Jayasekara, H., Seneviratne, S., Rodrigo, R. (2019). Deepcaps: Going deeper with capsule networks. In Proceedings of the IEEE/CVF

24. Mobiny, A., Van Nguyen, H. (2018, September). Fast capsnet for lung cancer screening. In International Conference on Medical Image Computing and Computer-Assisted Intervention (pp. 741-749). Springer, Cham.
25. Zhang, W., Tang, P., Zhao, L. (2019). Remote sensing image scene classification using CNN-CapsNet. Remote Sensing, 11(5), 494.
26. Gu, J., Tresp, V. (2020). Improving the robustness of capsule networks to image affine transformations. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 7285-7293).
27. Zhao, T., Liu, Y., Huo, G., Zhu, X. (2019). A deep learning iris recognition method based on capsule network architecture. IEEE Access, 7, 49691-49701.
28. Xiang, C., Zhang, L., Tang, Y., Zou, W., Xu, C. (2018). MS-CapsNet: A novel multi-scale capsule network. IEEE Signal Processing Letters, 25(12), 1850-1854.
29. Venkatraman, S., Balasubramanian, S., Sarma, R. R. (2019). Building Deep, Equivariant Capsule Networks. arXiv preprint arXiv:1908.01300.
30. Peer, D., Stabinger, S., Rodriguez-Sanchez, A. (2018). Training deep capsule networks. arXiv preprint arXiv:1812.09707.
31. He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
32. Do Rosario, V. M., Borin, E., Breternitz, M. (2019). The multi-lane capsule network. IEEE Signal processing letters, 26(7), 1006-1010.
33. Cohen, T., Welling, M. (2016, June). Group equivariant convolutional networks. In International conference on machine learning (pp. 2990-2999). PMLR.
34. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).
35. Chen, Y., Li, J., Xiao, H., Jin, X., Yan, S., Feng, J. (2017). Dual path networks. arXiv preprint arXiv:1707.01629.
36. Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., Fergus, R. (2013, May). Regularization of neural networks using dropconnect. In International conference on machine learning (pp. 1058-1066). PMLR.
37. Zhong, Z., Zheng, L., Kang, G., Li, S., Yang, Y. (2020, April). Random erasing data augmentation. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 34, No. 07, pp. 13001-13008).
38. Nair, P., Doshi, R., Keselj, S. (2021). Pushing the limits of capsule networks. arXiv preprint arXiv:2103.08074.
39. Deliege, A., Cioppa, A., Van Droogenbroeck, M. (2018). Hitnet: a neural network with capsules embedded in a hit-or-miss layer, extended with hybrid data augmentation and ghost capsules. arXiv preprint arXiv:1806.06519.
40. Venkatraman, S. R., Anand, A., Balasubramanian, S., Sarma, R. R. (2020). Learning Compositional Structures for Deep Learning: Why Routing-by-agreement is Necessary. arXiv preprint arXiv:2010.01488.
41. Manogaran, U., Wong, Y. P., Ng, B. Y. (2020, September). CapsNet vs CNN: Analysis of the Effects of Varying Feature Spatial Arrangement. In Proceedings of SAI Intelligent Systems Conference (pp. 1-9). Springer, Cham.
42. Sabour, S., Tagliasacchi, A., Yazdani, S., Hinton, G. E., Fleet, D. J. (2020). Unsupervised part representation by Flow Capsules. arXiv preprint arXiv:2011.13920.
43. Raghu, M., Poole, B., Kleinberg, J., Ganguli, S., Sohl-Dickstein, J. (2017, July). On the expressive power of deep neural networks. In international conference on machine learning (pp. 2847-2854). PMLR.